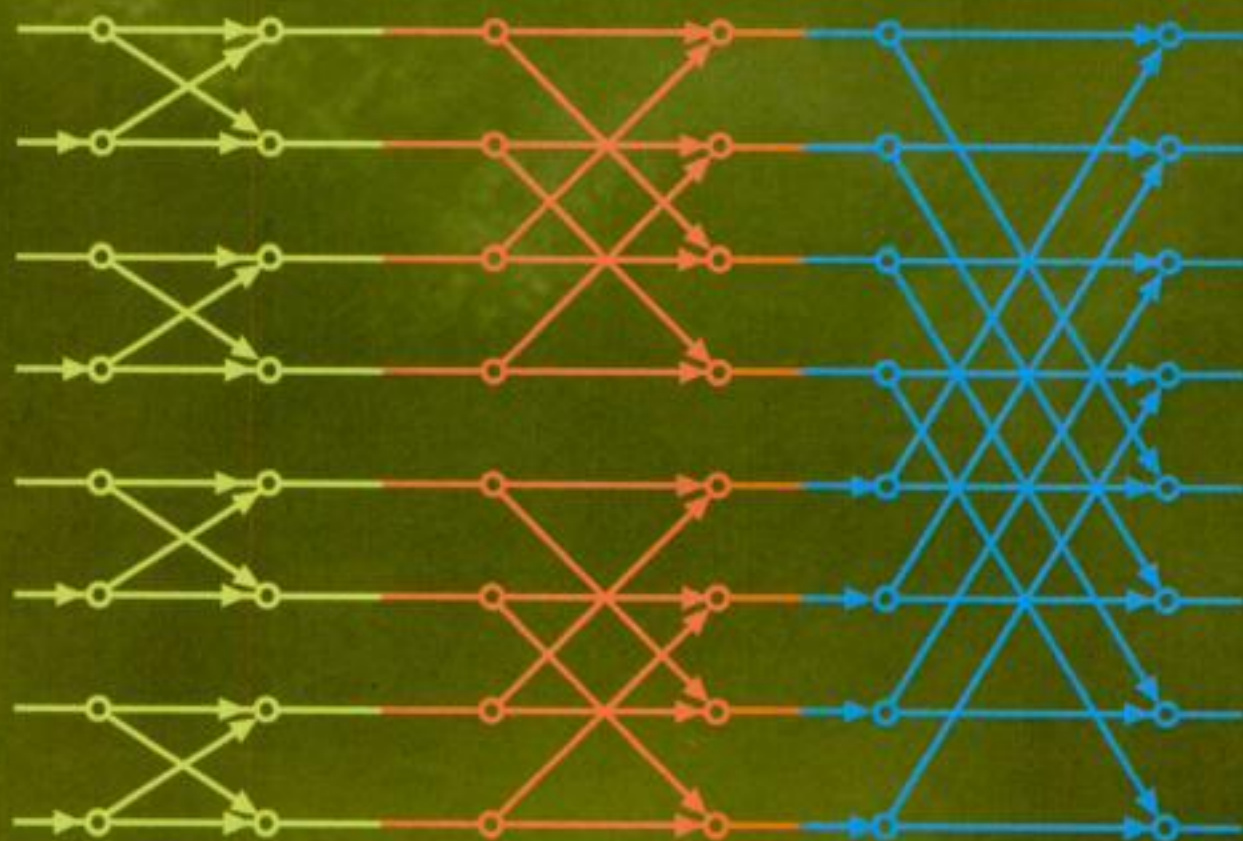


First Edition - 2008

Digital Signal Processing



© Programs Given In This Book Are Available On
www.vtubooks.com For **Free** Download

J. S. Chitode



Technical Publications Pune © All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or by any information storage and retrieval system, without the prior written permission of the publisher.

3

Digital Signal Processing

J. S. Chitode

M. E. Electronics (Digital Systems)
M.I.S.T.E., M.I.E.T.E., M.B.E.S.I.

Professor and Head
Department of Electronics Engineering
Bharati Vidyapeeth Deemed University
College of Engineering, Pune

Price Rs. 260/-

Visit us at : www.vtubooks.com



Technical Publications Pune®





Digital Signal Processing

ISBN 9788184314243

All rights reserved with Technical Publications. No part of this book should be reproduced in any form, Electronic, Mechanical, Photocopy or any information storage and retrieval system without prior permission in writing, from Technical Publications, Pune.

Published by :

Technical Publications Pune®

#1, Amit Residency, 412, Shaniwar Peth, Pune - 411 030, India.

Printer :

Alert DTPrinters
Sr.no. 10/3, Sinhgad Road,
Pune - 411 041

Preface

This book presents fundamental Digital Signal Processing techniques. The range of applications of Digital Signal Processing is vast. The Digital Signal Processing systems can be implemented on general purpose processors like PC or Digital Signal Processors. The Digital Signal Processing basically involves processing of discrete samples of data. Because of the availability of high speed computers, Digital Signal Processing (DSP) operations like filtering, convolution, correlation, FFT etc. can be implemented fast. The DSP processors are available. These processors are designed for fast implementations of DSP operations. Many real time applications use DSP processor based systems.

The first chapter introduces DSP and its scope of applications. The second chapter presents characteristics and properties of signals and systems. Mainly discrete time signals and discrete time systems are concentrated. The standard discrete time signals, properties of discrete time systems, linear time invariant systems, difference equations etc. are presented in this chapter.

The third chapter presents z-transform and its properties. The z-transform is used to analyze the discrete time systems. It presents pole-zero plots, causality and stability criteria for discrete time systems.

The fourth chapter describes various digital filter structures. The various techniques of realization of discrete time systems are presented in this chapter.

The digital filter design is discussed in sixth chapter. The FIR and IIR filter design is presented in this chapter. Butterworth approximation frequency transformation, least squares filter design etc is also discussed in this chapter.

Chapter 7 presents the architecture and features of DSP processors. The ADSP-21XX and ADSP-2106X are described. The brief instruction set, development tools etc. are described in this chapter. This chapter also presents features and architecture of TMS320C5X series of DSP processors.

Finite wordlength effects limit the performance of DSP systems. These effects are discussed in 8th chapter. Effects of coefficient quantization, A/D conversion noise, rounding and truncation in arithmetic operations etc. is discussed in this chapter. The dynamic scaling, limit cycles etc topic are also discussed.

The last that is 9th chapter presents applications of DSP. Applications in DTMF detection, speech, music, audio, image processing are presented. Oversampling A/D, D/A and applications of multirate signal processing are presented.

At the end of every chapter 'C' programs are presented. The implementation logic and results of these programs are also discussed. The list of these programs is given in the index.

Large number of solved examples are presented to make concepts clear. Unsolved examples are also given at the end of every chapter along with their answers for practice. Attempts are made to make this text as lucid as possible. Efforts are taken for consistency in various topics. However there is a chance of typing, alignment and organizational errors. Any criticism or suggestions in this regard will be highly appreciated.

Acknowledgement

I am thankful to the staff of Computer and Electronics departments for their encouragement and support. I also thank **the Publisher** and the team of Technical Publications to publish this book.

Author

J. S. Chitode

Dedicated to God

Table of Contents

Chapter 1	Introduction	1
1.1	Basic Elements of Digital Signal Processing	1
1.2	Digital Against Analog Signal Processing	2
1.3	DSP Applications	3
1.4	Technology Review	4
1.5	Study of DSP	4
Chapter 2	Discrete Time Signals and Systems	6
2.1	Introduction	6
2.2	Discrete Time Signals as Array of Values	6
2.2.1	Continuous Time Signals	6
2.2.2	Discrete Time Signals	7
2.2.3	Classification of Signals	12
2.2.4	Frequency Concept in Discrete Time Signals	14
2.3	Standard Discrete Time Signals (Sequences)	27
2.3.1	Unit Sample Sequence	27
2.3.2	Unit Step Sequence	27
2.3.3	Unit Ramp Sequence	28
2.3.4	Exponential Sequence	29
2.4	Classification of Discrete Time Signals	31
2.4.1	Energy Signals and Power Signals	31
2.4.2	Periodic Signals and Non-periodic Signals	31
2.4.3	Even and Odd Signals or Symmetric and Antisymmetric Signals	31
2.5	Discrete Time Systems	33
2.5.1	Representation of Discrete Time Systems	38
2.6	Classification of Discrete Time Systems	40
2.6.1	Static and Dynamic Systems (Dynamicity Property)	41
2.6.2	Shift Invariant and Shift Variant Systems (Shift Invariance Property)	42
2.6.3	Linear and Nonlinear Systems (Linearity Property)	45
2.6.4	Causal and Noncausal Systems (Causality Property)	47
2.6.5	Stable and Unstable Systems (Stability Property)	48
2.7	Linear Time Invariant (LTI) Systems	63

2.7.1	Discrete Time Signal as Weighted Impulses	63
2.7.2	Linear Convolution	66
2.7.3	Properties of Convolution	74
2.7.4	Causality of LTI Systems	77
2.7.5	Stability of LTI Systems	79
2.8	Difference Equations	107
2.8.1	Finite Impulse Response (FIR) Systems	108
2.8.2	Infinite Impulse Response (IIR) Systems	108
2.8.3	Nonrecursive Systems	109
2.8.4	Recursive Systems	110
2.8.5	Representation of Discrete Time Systems via Difference Equations	111
2.9	Correlation	113
2.9.1	Cross-correlation and Auto-correlation	113
2.9.2	Properties of Crosscorrelation and Autocorrelation Sequences	118
2.10	A/D Conversion Process	119
2.10.1	Sampling	119
2.10.2	Frequency Relationships	121
2.10.3	Aliasing	125
2.10.4	Quantization	126
2.10.5	Encoding	129
2.10.6	Sampling Theorem	129
2.10.7	Anti Aliasing Filter	131
2.11	Implementation of General Difference Equation using 'C'	135
2.12	Generation of Waveforms using 'C'	138
2.13	Linear Convolution using 'C'	143
	Computer Exercise	146
	Theory Questions	147
	Unsolved Examples	147

Chapter 3 The z-Transform 151

3.1	Introduction	151
3.2	z-transform	151
3.3	Properties of z-transform	159
3.3.1	Linearity	159
3.3.2	Time Shifting	159

3.3.3	Scaling in z-domain	160
3.3.4	Time Reversal	160
3.3.5	Differentiation in z-domain	161
3.3.6	Convolution in Time Domain	162
3.3.7	Correlation of Two Sequences	163
3.3.8	Multiplication of Two Sequences	164
3.3.9	Conjugation of a Complex Sequence	165
3.3.10	z-transform of Real Part of the Sequence	166
3.3.11	z-transform of Imaginary Part of the Sequence	166
3.3.12	Parseval's Relation	167
3.3.13	Initial Value Theorem	168
3.4	Inverse z-transform	181
3.4.1	Inverse z-transform using Partial Fraction Expansion	182
3.4.2	Inverse z-transform by Power Series Expansion	197
3.5	System Function and Pole-zero Plots from z-transform	211
3.5.1	Pole-zero Plots	211
3.5.2	System Function of the LTI System	215
3.6	Causality and Stability in Terms of z-transform	218
3.7	Computation of z-transform	229
3.8	Computation of Coefficients of a Difference Equation from Poles and Zeros	233
3.8.1	Logic for Computation	233
3.8.2	'C' Program for Computation of Coefficients from Poles and Zeros	235
3.9	Computation of Linear Convolution using z-transform	238
3.9.1	Logic for Computation	238
3.9.2	C Program for Computation of Linear Convolution using z-transform	239
	Computer Exercise	242
	Theory Questions	243
	Unsolved Examples	243

Chapter 4 Digital Filter Structures 246

4.1	Describing Equation	246
4.2	Structures for FIR Systems	248
4.2.1	Direct Form Structure of FIR System	248
4.2.2	Cascade Form Structure for FIR System	249
4.2.3	Frequency Sampling Structure for FIR Systems	251

4.2.4	Lattice Structure for FIR Systems	254
4.3	Structure for IIR Systems	256
4.3.1	Direct Form Structures for Systems	256
4.3.1.1	Direct Form-I Structure of IIR System	256
4.3.1.2	Direct Form-II Structure for IIR System	258
4.3.2	Cascade Form Structure for IIR Systems	263
4.3.3	Parallel Form Structure for IIR Systems	264
4.3.4	Lattice Structure for IIR Systems	265
4.4	Representation of Structures using Signal Flow Graphs	268
	Theory Questions	271

Chapter 5 DFT, FFT and their Applications 272

5.1	Introduction	272
5.2	Fourier Transform of Discrete Time Signals	272
5.2.1	Fourier Transform of Standard Signals	273
5.2.2	Properties of Fourier Transform	275
5.2.3	Inverse Fourier Transform	277
5.2.4	Magnitude / Phase Transfer Functions using Fourier Transform	278
5.2.5	Relationship Between Fourier Transform and z-transform	282
5.2.6	Frequency Scale on the Unit Circle	283
5.3	Discrete Fourier Transform (DFT)	289
5.3.1	Definition of DFT and IDFT	289
5.3.2	DFT of Standard Signals	298
5.3.3	Properties of DFT	299
5.3.3.1	Periodicity	299
5.3.3.2	Linearity	300
5.3.3.3	Circular Symmetries of a Sequence	300
5.3.3.4	Symmetry Properties	306
5.3.3.5	Circular Convolution	308
5.3.3.6	Time Reversal of a Sequence	326
5.3.3.7	Circular Time Shift of a Sequence	327
5.3.3.8	Circular Frequency Shift	330
5.3.3.9	Complex Conjugate Properties	330
5.3.3.10	Circular Correlation	331
5.3.3.11	Multiplication of Two Sequences	332

5.3.3.12	Parseval's Theorem	333
5.3.4	Relationship Between DFT and z-transform	335
5.3.5	Applications of DFT	335
5.3.5.1	DFT for Linear Filtering	335
5.3.5.2	Overlap Save Method for Linear Filtering	340
5.3.5.3	Overlap Add Method for Linear Filtering	341
5.3.5.4	Frequency or Spectrum Analysis using DFT	343
5.4	Fast Fourier Transform (FFT) Algorithms	345
5.4.1	Introduction to FFT	345
5.4.2	Properties of W_N	345
5.4.3	Classification of FFT Algorithms	347
5.5	Radix-2 FFT Algorithms	348
5.5.1	Radix-2 DIT-FFT Algorithm	348
5.5.1.1	Butterfly Computation	355
5.5.1.2	Computational Complexity Compared to Direct Computation	357
5.5.1.3	Memory Requirement and Inplace Computations	358
5.5.1.4	Bit Reversal	359
5.5.2	Radix-2 DIF FFT Algorithm	360
5.5.2.1	Butterfly Operation in DIF FFT	363
5.5.2.2	Computational Complexity	365
5.5.2.3	Memory Requirement and Inplace Computation	365
5.5.2.4	Bit Reversal	365
5.6	Goertzel Algorithm	365
5.7	Chirp-z Transform Algorithm	367
5.8	Use of FFT Algorithms in Linear Filtering and Correlation	369
5.9	Quantization Effects in FFT Algorithms	370
5.10	Power Spectrum Estimation using DFT	371
5.11	Computation of Fourier Transform	373
5.12	Computation of DIT FFT Algorithm	377
5.13	Inverse DFT and Computation of IDFT using FFT Algorithms	385
5.14	Computation of DFT and IDFT	388
5.14.1	Logic for Computation of DFT	388
5.14.2	'C' Program for Computation of DFT	388
5.14.3	Logic for Computation of IDFT	390
5.14.4	'C' Program for Computation of IDFT	391
5.15	Computation of Circular Convolution	392

5.15.1	Logic for Computation	392
5.15.2	'C' Program for Computation of Circular Convolution	393
5.16	Computation of Circular Convolution using DFT and IDFT	395
5.16.1	Logic for Computation	395
5.16.2	'C' Program for Computation of Circular Convolution using DFT and IDFT	396
5.17	Computation of Magnitude and Phase Transfer Function Plots	398
5.17.1	Logic for Computation and Magnitude and Phase	398
5.17.2	'C' Program for Magnitude and Phase Transfer Function Plot	400
	Computer Exercise	403
	Theory Questions	403
	Unsolved Examples	404

Chapter 6 Filter Design **405**

6.1	Introduction	405
6.2	Difference Between Analog Filters and Digital Filters	405
6.2.1	An Example of Analog Lowpass Filter	405
6.2.2	An Example of Digital Lowpass Filter	405
6.2.3	Implementation of Digital Filter	406
6.2.4	Comparison of Analog and Digital Filters	407
6.3	Types of Digital Filters	407
6.3.1	An Example of FIR Filter	408
6.3.2	An Example of IIR Filter	409
6.4	LTI Systems as Filters	410
6.4.1	Ideal Filter Characteristics	410
6.4.2	Realizability of Ideal Filters	412
6.5	Design of IIR Filters from Analog Filters	414
6.5.1	IIR Filter Design by Approximation of Derivatives	414
6.5.2	IIR Filter Design by Impulse Invariance	417
6.5.3	IIR Filter Design by Bilinear Transformation	422
6.6	IIR Filter Design using Butterworth Approximation	425
6.6.1	Necessity of Filter Approximation	425
6.6.2	Butterworth Filter Approximation	426
6.7	Frequency Transformations	448
6.8	FIR Filters	456
6.8.1	Inherent Stability of FIR Filters	456

6.8.2	Symmetric and Antisymmetric FIR Filters	456
6.8.3	Linear Phase in FIR Filters	457
6.8.4	Magnitude Characteristics and Order of FIR Filter	460
6.9	FIR Filter Design	461
6.9.1	Design of Linear Phase FIR Filters using Windows	461
6.9.1.1	Rectangular Window for FIR Filter Design	462
6.9.1.2	Gibbs Phenomenon	464
6.9.1.3	Commonly Used Window Functions	464
6.9.2	Design of Linear Phase FIR Filters using Frequency Sampling	472
6.9.3	Design of Optimum Equiripple Linear Phase FIR Filters	473
6.9.4	FIR Differentiators	475
6.9.5	Design of Hilbert Transformers	476
6.9.6	Comparison of Designing Methods	476
6.10	Least Squares Filter Design	477
6.11	Designing Digital Filters from Pole-Zero Placement	479
6.12	Comparison of FIR and IIR Filters	485
6.13	Butterworth Filter Design using Bilinear Transformation	486
6.13.1	Design Steps	487
6.13.2	Logic for Computation of System Function	488
6.13.3	C Program for Butterworth Filter Design using Bilinear Transformation	493
6.14	FIR Filter Design using Windows	497
6.14.1	Design Steps	497
6.14.2	Logic for Computation of Coefficients of FIR Filters	497
6.14.3	C Program for FIR Filter Design using Windows	500
6.15	Design of Filters using Pole-Zero Combination	503
6.15.1	Logic for Computation	503
6.15.2	C Program for Filter Design using Pole-Zero Combination	505
	Computer Exercise	509
	Theory Questions	509
	Unsolved Examples	509

Chapter 7 Hardware Architecture of DSP (DSP Processors) 511

7.1	Introduction	511
7.2	Desirable Features of DSP Processors	511
7.3	Types of Architectures	512

7.4	Internal Architecture of ADSP-21xx Family	513
7.4.1	Arithmetic /Logic Unit (ALU)	516
7.4.2	Multiplier-Accumulator	517
7.4.3	Barrel Shifter	519
7.4.4	Data Address Generators (DAG)	521
7.4.5	Program Sequencer	521
7.4.6	Cache Memory	522
7.5	Features of ADSP-21xx Family of Processors	522
7.5.1	Boot Address Generator	525
7.5.2	Serial Ports	525
7.5.3	Timer	525
7.5.4	Interrupts	525
7.5.5	Pin Definitions	526
7.6	System Interface	528
7.6.1	Clock Signals	528
7.6.2	Reset	529
7.6.3	Program Memory Interface	529
7.6.4	Program Memory Maps	529
7.6.5	Data Memory Interface	530
7.6.6	Data Memory Map	530
7.6.7	Boot Memory Interface	531
7.7	Instruction Set of ADSP-21xx	531
7.8	ADSP-21xx Development Tools	534
7.8.1	System Builder	534
7.8.2	Assembler	535
7.8.3	Linker	536
7.9	ADSP-2106x Processors	536
7.10	TMS DSP Processors	538
7.10.1	Features of Processors	538
7.10.2	Architecture of TMS320C5x Processors	539
7.11	Comparison Between DSP Processors and General Purpose Microprocessors	543
	Computer Exercise	544
	Theory Questions	544

Chapter 8 Analysis of Finite Word-length Effects **546**

8.1 Introduction	546
8.2 The Quantization Process and Errors	546
8.3 Analysis of Coefficient Quantization Effects in FIR Filters	548
8.4 A/D Conversion Noise Analysis	550
8.5 Analysis of Arithmetic Roundoff Errors	553
8.6 Dynamic Range Scaling	554
8.7 Low Sensitivity Digital Filters	554
8.8 Reduction of Product Roundoff Errors	556
8.9 Limit Cycles in IIR Filters	557
8.10 Roundoff Errors in FFT Algorithms	557
Theory Questions	557

Chapter 9 Applications **558**

9.1 Dual-Tone Multifrequency Signal Detection	558
9.2 Spectral Analysis using DFT	560
9.3 Short Term DFT	560
9.4 Musical Sound Processing	562
9.5 Voice Privacy	564
9.6 Subband Coding of Speech and Audio Signals	565
9.7 Oversampling A/D Converter	567
9.8 Oversampling D/A Converter	568
9.9 Applications of Multirate Signal Processing	568
9.10 Applications of DSP in Image Processing	569
9.11 Echo Cancellation	570
9.12 Vibration Analysis	571
Exercise	571

References **572**

Index **573**

List of Programs

The following programs are available on www.technicalpublicationspune.com for free download. You will get programs in 'zip' file from website. You will have to extract the programs from this 'zip' file using 'winzip' facility available in windows. There are three folders, The C_source folder contains source files of 'C' programs. The C_exe folder contains exe files of 'C' programs.

1)	diffeqn.cpp	(Chapter 2)	135
2)	signals.cpp	(_")	139
3)	linconv.cpp	(_")	143
4)	ztrans.cpp	(Chapter 3)	230
5)	plz2cff.cpp	(_")	235
6)	convztr.cpp	(_")	239
7)	fourier.cpp	(Chapter 5)	373
8)	ditfft.cpp	(_")	377
9)	idft.cpp	(_")	385
10)	dft.cpp	(_")	388
11)	invdft.cpp	(_")	391
12)	circonv.cpp	(_")	393
13)	circonv2.cpp	(_")	396
14)	magphse.cpp	(_")	400
15)	biliner.cpp	(Chapter 6)	493
16)	fir.cpp	(_")	500
17)	notch.cpp	(_")	505

Abbreviations and Symbols

A_p	Passband attenuation
A_s	Stopband attenuation
ALU	Arithmetic Logic Unit
BLT	Bilinear Transformation
B	Bandwidth
b	Binary digits for representation of one level
DAG	Data Address Generator
DMD	Data Memory Data bus
DMA	Data Memory Address bus
DFT	Discrete Fourier Transform
E_x	Energy of signal $x(n)$
$e_q(n)$	Quantization error
FFT	Fast Fourier Transform
FT	Fourier Transform
F	Continuous time frequency in Hz
F_s	Sampling frequency in Hz.
f	Discrete time frequency in cycles/sample
FIR	Finite Impulse Response Filter/system
$H(\omega)$	Transfer function
$H(z)$	System function
$h(n)$	Unit sample response
IIR	Infinite Impulse Response Filter/system
IZT	Inverse z-transform
$\text{Im}\{\}$	Imaginary part of
j	Imaginary constant, $\sqrt{-1}$
L	Number of quantization levels
MAC	Multiplier Accumulator
n	Sample index
N	Number of data samples or period of signal or order of digital filter
P_x	Power of $x(n)$
PMA	Program Memory Address
PMD	Program Memory Data
$\text{Re}\{\}$	Real part of
ROC	Region of convergence
s	Laplace domain variable
$T[\]$	Response of system to input in brackets
t	Continuous time variable

T_s or T	Sampling duration
$u_r(n)$	Unit ramp signal
$u(n)$	Unit step sequence
$X(\omega)$ or $X(f)$	Fourier transform of discrete time signal
$X(\Omega)$ or $X(F)$	Fourier transform of continuous time signal
$x_R(n)$	Real part of $x(n)$
$x_I(n)$	Imaginary part of $x(n)$
$x_q(n)$	Quantized value of $x(n)$
$x(n)$	Discrete time sequence at input
$y(n)$	Discrete time sequence at output
ZT	z-transform
z	z-domain variable
$\angle P$	Angle of variable P
\xrightarrow{F} or \xleftarrow{FT}	Fourier transform
$\tau_g(\omega)$	Envelope delay or group delay
σ	Real axis in s-plane
ω_p or Ω_p	Passband edge frequency
ω_s or Ω_s	Stopband edge frequency
ω_c or Ω_c	Cutoff frequency
Δ or δ	Quantization step size or resolution
$x(n) * h(n)$	Linear convolution of $x(n)$ and $h(n)$
$x(n) \textcircled{N} h(n)$	Circular convolution of $x(n)$ and $h(n)$
$ a $	Absolute value of variable 'a'
$x^*(n)$	Complex conjugate of $x(n)$
$r_{xy}(l)$	Crosscorrelation of $x(n)$ and $y(n)$
$r_{xx}(l)$	Autocorrelation of $x(n)$
$\rho_{xx}(l)$	Normalized autocorrelation sequence
\xleftrightarrow{z}	z-transform pair
\oint_C	Integration over closed contour
$\delta(n)$	Unit sample sequence
Ω	Continuous time frequency in radians/sec
ω	Discrete time frequency in radians/sample
π	Constant pi its value is $\frac{22}{7}$

Chapter 1

INTRODUCTION

Digital Signal Processing is common today. Everybody of us is directly or indirectly related to Digital Signal Processing and its applications. The subject of Digital Signal Processing (DSP) is mainly related to Electronics and Computer Engineering. Because of the advancements in computers, the applications range of Digital Signal Processing is growing. The signal processing takes place in amplifiers, attenuators, transformations, filters, transmission lines, channels etc. When the signal is analog, it is called Analog Signal Processing. When the signal is digital in nature, it is called *Digital Signal Processing*. All of us know that computer is a digital machine. Thus any type of signal processing done by the computers is basically a Digital Signal Processing. In this chapter we will introduce the Digital Signal Processing in brief.

1.1 Basic Elements of Digital Signal Processing

Fig. 1.1.1 shows the basic elements of digital signal processing system. Most of the signals generated are analog in nature. Hence these signals are converted to digital form by the analog to digital converter. Thus the Analog to Digital (A/D) converter generates an array of samples and gives it to the digital signal processor. This array of samples (or sequence of samples) is the digital equivalent of input analog signal. It is also called digital signal. The digital signal processor performs signal processing operations like filtering, multiplication, transformation, amplification etc. operations over this digital signal (sequence of samples) and generates another digital signal at its output. This digital signal processor can be the high speed digital computer or digital signal microprocessor. Such processors perform signal processing operations with the help of the software, which decides the type of operation. The digital signal processors are specially designed of digital signal processing. The digital output signal from the digital signal processor is given to digital to analog converter. The digital to analog (D/A) converter gets an analog equivalent of the output digital signal.

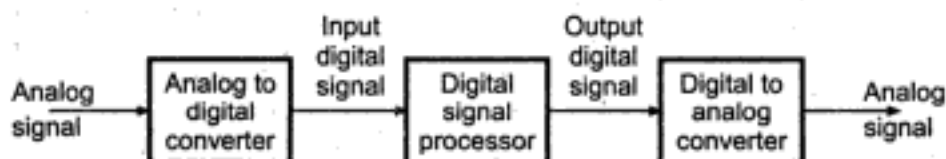


Fig. 1.1.1 Basic elements of a digital signal processing

The elementary digital signal processing discussed above can be used in number of applications as we will see next. Any type of analog signal is converted to digital form and processed by the digital signal processor. Some times the digital signal is available on the storage media like Hard disk floppies, magnetic tapes etc. Such signal is then given to the

digital signal processor. Hence two types of digital signal processings are possible, real time digital signal processing and nonreal time (or offline) digital signal processing. When the analog data is processed as it is generated, it is called real time application. For example, the processing radar signal or speech signal by the digital signal processor is mostly real time operation. In case of offline or nonreal time applications the digital data is stored on some storage media. The digital signal processor then performs signal processing on this data. The processing of satellite images after they are taken is offline type of application.

Here we have briefly introduced digital signal processing system. We observed that the digital signal processing operation is basically performed on the sequence of digital data samples. In the succeeding sections we will see the details like frequency and amplitude related concepts of this digital data sequences.

1.2 Digital Against Analog Signal Processing

The digital signal processing offers many advantages over analog signal processing. These advantages are discussed next.

1. Digital signal processing systems are flexible. The system can be reconfigured for some other operation by simply changing the software program. For example, the high pass digital filter can be changed to low pass digital filter by simply changing the software. For this, no changes in the hardware are required. Thus digital signal processing systems are highly flexible. But this type of change is not easily possible in analog system. An analog system which performing as high pass filter, is to be totally replaced to get lowpass filter operation.
2. Accuracy of digital signal processing systems is much higher than analog systems. The analog systems suffer from component tolerances, their breakdown etc. problems. Hence it is difficult to attain high accuracy in analog systems. But in digital signal processing systems, these problems are absent. The accuracy of digital signal processing systems is decided by resolution of A/D converter, number of bits to represent digital data, floating/fixed point arithmetic etc. But these factors are possible to control in digital signal processing systems to get high accuracy.
3. The digital signals can be easily stored on the storage media such as magnetic tapes, disks etc. Whereas the analog signals suffer from the storage problems like noise, distortion etc. Hence digital signals are easily transportable compared to analog signals. Thus remote processing of digital signals is possible compared to analog signals.
4. Mathematical operations can be accurately performed on digital signals compared to analog signals. Hence mathematical signal processing algorithms can be routinely implemented on digital signal processing systems. Whereas such algorithms are difficult to implement on analog systems.
5. When there is large complexity in the application, then digital signal processing systems are cheaper compared to analog systems. The software control algorithm can be complex, but it can be implemented accurately with less efforts.
6. The processing of the signals is completely digital in digital signal processing systems. Hence the performance of these systems is exactly repeatable. For example the lowpass filtering operation performed by digital filter today, will be exactly same even after ten years. But the performance may deteriorate in analog systems because of noise effects and life of components etc.
7. The digital signal processing systems are easily upgradable since they are software controlled. But such easy upgradation is not possible in analog systems.

8. The digital signal processing systems use digital computers or standard digital signal processors as their hardware. Almost all the applications use this as standard hardware with minor modifications. The operation of the digital signal processing is decided mainly by software program. Hence universal compatibility is possible in digital signal processing systems. Whereas it is not possible in analog systems. Since a simple analog low pass filter can be implemented by large number of ways.
9. The digital signal processing systems are small in size, more reliable and less expensive compared to the analog systems.

Disadvantages of Digital Signal Processing Systems

Eventhough the digital signal processing systems have all the above advantages, they have few drawbacks as follows :

1. When the analog signals have wide bandwidth, then high speed A/D converters are required. Such high speeds of A/D conversion are difficult to achieve for same signals. For such applications, analog systems must be used.
2. The digital signal processing systems are expensive for small applications. Hence the selection is done on the basis of cost complexity and performance.

The advantages of digital communication systems outweigh the above drawbacks.

1.3 DSP Applications

In the last section we discussed the advantages of DSP. Now let us see what is the range of applications of DSP. The summary of important DSP applications is presented below.

(1) DSP for Voice and Speech :

Speech recognition, voice mail, speech vocoding, speaker verification, speech enhancement, speech synthesis, text to speech etc.

(2) DSP for Telecommunications :

FAX, cellular phone, speaker phones, digital speech interpolation, video conferencing, spread spectrum communications, packet switching, echo cancellation, digital EPABXs, ADPCM transcoders, channel multiplexing, Modems adaptive equalizers, data encryption and line repeaters etc.

(3) DSP for Consumer Applications :

Digital audio/video/Television/Music systems, music synthesizer, Toys etc.

(4) DSP for Graphics and Imaging :

3-D and 2-D visualization, animation, pattern recognition, image transmission and compression, image enhancement, robot vision, satellite imaging for multipurpose applications etc.

(5) DSP for Military/Defence :

Radar processing, Sonar processing, Navigation, missile guidance, RF modems, secure communications.

(6) DSP for Biomedical Engineering :

X-ray storage and enhancement, ultrasound equipment, CT scanning equipments, ECG analysis, EEG brain mappers, hearing aids, patient monitoring systems, diagnostic tools etc.

(7) DSP for Industrial Applications :

Robotics, CNC, security access and power line monitors etc.

(8) DSP for Instrumentation :

Spectrum analysis, function generation, transient analysis, digital filtering, phase locked loops, seismic processing, pattern matching etc.

(9) DSP for Control Applications :

Servo control, robot control, laser printer control, disk control, engine control and motor control etc.

(10) DSP for Automotive Applications :

Vibration analysis, voice commands, digital radio, engine control, navigation, antiskid brakes, cellular telephones, noise cancellation, adaptive ride control etc.

Thus DSP has wide range of applications. We will be studying few applications in details in last chapter.

1.4 Technology Review

Here we will briefly see what is the technology used for DSP. A DSP system for the particular application can be implemented as Dedicated processor based DSP and General purpose processor based DSP.

(i) Dedicated Processor based DSP :

In such systems the DSP processors are used. The DSP processors of Analog Devices (ADSP 21XX series), Texas Instruments (TMS 320XXX) and Motorola (M 56XXX) are commonly used. These DSP processors are designed specially for array operations and multiply-accumulate operations. The DSP processors based systems are stand alone, portable, low cost and suitable for real time applications.

(ii) General purpose processor based DSP :

Such systems use general purpose micro-processors or computers. The software is developed to perform DSP operations on computers. For example, 'C' programs can be developed for digital filtering, z-transform, fourier transform, FFT etc. which run on computer. Thus utility of computers can be increased. Such systems are flexible and easily upgradable. The technologies of computers such as networking, storage, display, printing etc. can be shared. But such systems are computationally inefficient. If only DSP operations are to be performed, then it is better to use dedicated processor based systems.

1.5 Study of DSP

The DSP is to be first introduced through basic elements, application areas and technology.

Then the signals and their properties are to be studied which are used in DSP. Discrete time signals are used in DSP. Hence discrete time signals, their properties, generation etc, should be studied.

Next is analysis of signals. The discrete time signals can be analyzed in time domain as well as frequency domain. Fourier transform and discrete fourier transforms are the standard tools for analysis of signals.

DSP applications are implemented with the help of discrete time systems. For example digital filters, correlations etc. The properties of these discrete time systems are studied. Cascading, paralleling etc. of the discrete time systems is also studied.

z-transform is major tool for the analysis of discrete time systems. Hence computation of z-transform, pole-zero plots system function, transfer function etc. is studied.

Computational DSP consists of FFT algorithms, design and implementation of digital filters, estimation etc. This is the fundamental study for implementing any DSP application.

Lastly is to study use of DSP in few applications. The difference between dedicated and general purpose processor based DSP is to be studied. Few applications and their implementations is to be studied.

The above topics are systematically presented in this book. The concepts are supported with illustrative 'C' programs. Study of DSP processors and few application case studies are also presented at the end. The matter presented in this book is balanced for fundamental study of theoretical and practical concepts.

□□□

Chapter 2

DISCRETE TIME SIGNALS AND SYSTEMS

2.1 Introduction

From this chapter we are starting the study of digital signal processing. The digital signal processing involves discrete or digital signals and discrete time systems. Here we will study various types of signals, systems and their properties. The important properties of systems such as stability, causality, shift invariance etc. are discussed in this chapter. Similar properties do exist for analog systems also. The realizability of the discrete time systems can be tested with the help of these properties. In the signals and systems analysis we normally use standard signals. For analog systems, these signals are unit step, unit impulse, unit ramp etc. For discrete time systems, these signals are unit sample sequence, unit step sequence and unit ramp sequence. In this chapter we will also study the important concept of convolution of two sequences and its applications.

2.2 Discrete Time Signals as Array of Values

We know that all the signals in digital signal processing are discrete (or digital) in nature. The corresponding signal is analog for analog signal processing systems. The analog signals are also called continuous time signals.

2.2.1 Continuous Time Signals

Let us briefly look at what are continuous time and analog signals. Fig. 2.2.1 shows various continuous time signals. In this figure observe that exponential as well as sinusoid are the examples of analog signals. The special characteristic of analog signals is that they are continuous in amplitude and defined at every time instant. For example you can calculate value of exponential signal ' e^{-t} ' at $t = 0.1, 0.001, 0.5, 0.0082 \dots$ etc. time instants. Thus it is defined at all the time instants. Similarly ' e^{-t} ' can take values from 1 to zero with continuous variation. The other examples of analog signals are ECG signals, speech signals, Television signals, noise signals etc. Almost all the signals generated from various sources in the nature are basically analog.

Please refer Fig. 2.2.1 on next page.

Continuous time, discrete amplitude signals

Just now we were discussing about continuous time and continuous amplitude; i.e. analog signals. It is also possible to have the signals which have continuous time but discrete amplitudes. Fig. 2.2.2 shows a signal which is defined at all the times but have discrete amplitude levels. This signal can take the amplitude only in three steps but can be defined at any time instants. Hence it is called continuous time discrete amplitude signals.

Please refer Fig. 2.2.2 on next page.

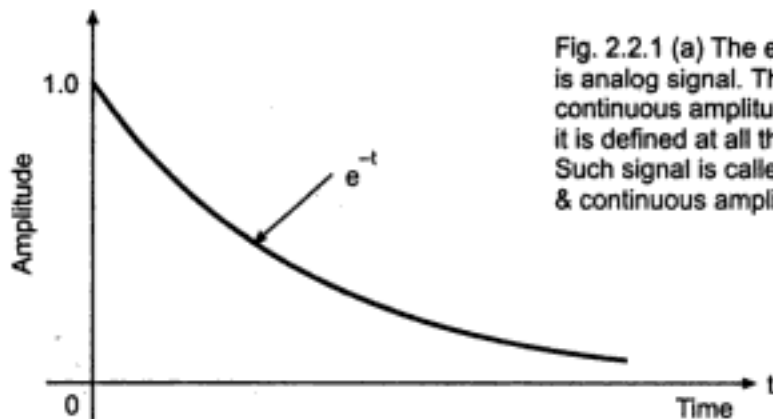


Fig. 2.2.1 (a) The exponential signal is analog signal. This signal has continuous amplitude as well as it is defined at all the time. Such signal is called continuous time & continuous amplitude or analog signal.

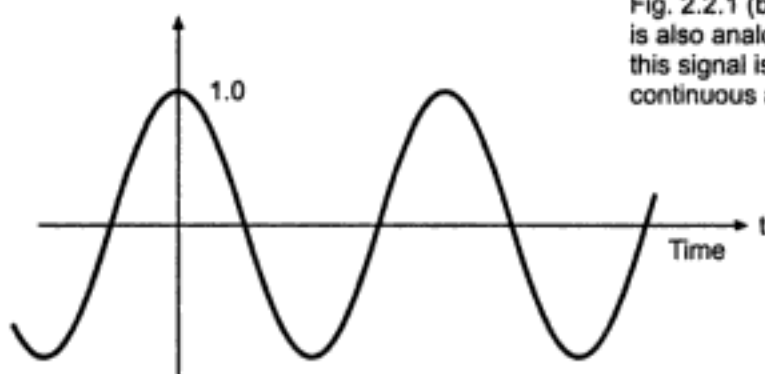


Fig. 2.2.1 (b) The sinusoidal signal is also analog signal. Observe that this signal is also continuous time continuous amplitude signal.

Fig. 2.2.1 Examples of continuous time signals

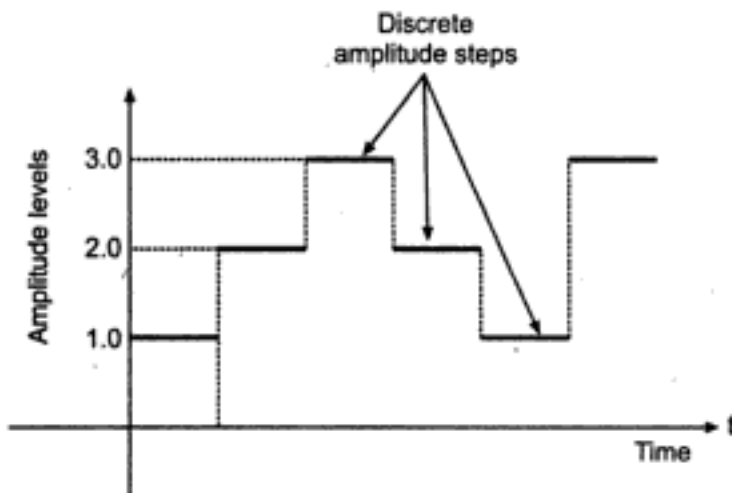


Fig. 2.2.2 An example of continuous time discrete amplitude signal. This signal is defined at all the time instants, but takes discrete amplitude levels

2.2.2 Discrete Time Signals

The discrete time signals are obtained by time sampling of continuous time signals. Hence the discrete time signals are defined only at sampling instants. Let us consider the exponential signal e^{-t} . This signal can be defined at $-\infty < t < \infty$. In this range it can be defined at any time instant. Let us consider that this exponential signal is sampled at the time instants separated by

period T . Fig. 2.2.3 shows the continuous time signal, sampling instants and sampled or discrete time signal. The sampling function of Fig. 2.2.3 (b) is the train of impulses. Fig. 2.2.3(c) shows the discrete time exponential function. Observe that the discrete time exponential function is defined only at sampling instants $0, \pm T, \pm 2T, \pm 3T, \dots$ etc.

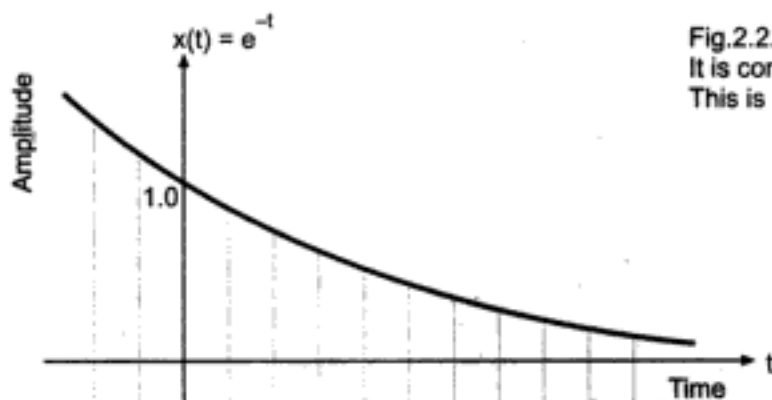


Fig.2.2.3 (a) An exponential function. It is continuous in time and amplitude. This is continuous time signal.

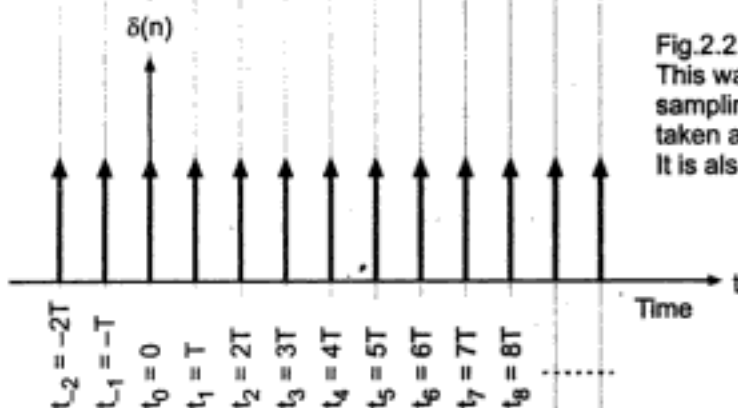


Fig.2.2.3 (b) A sampling function. This waveform indicates the sampling instants. Samples are taken at $\pm T, \pm 2T, \pm 3T, \dots$. It is also called instantaneous sampling.

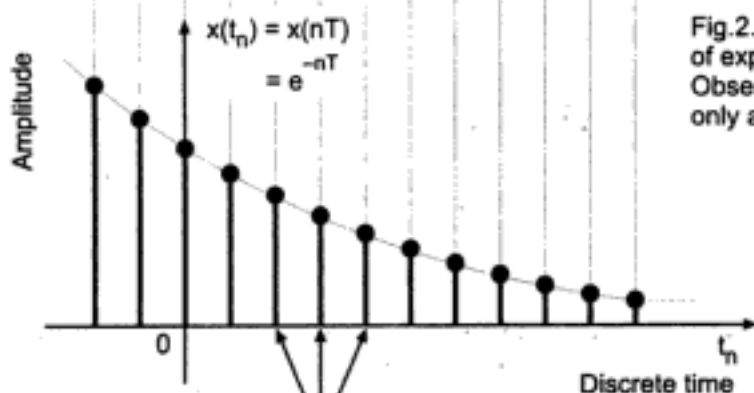


Fig.2.2.3 (c) A discrete time equivalent of exponential function of Fig.(a). Observe that this function is defined only at sampling instants.

This signal is defined only at sampling instants

Fig. 2.2.3 Representation of a discrete time signal

The sampling instants are normally represented by $t_n = nT$. The sampled or discrete time signal becomes,

$$x(t_n) = x(nT) \quad \dots (2.2.1)$$

Till now we have observed the discrete time signal in the graphical form. Let us assume that the sampling duration $T = 1$ sec and

$$x(nT) = \left. \begin{array}{l} e^{-nT} \quad \text{for } n \geq 0 \\ = 0 \quad \text{for } n < 0 \end{array} \right\} \quad \dots (2.2.2)$$

Here we are considering discrete time exponential signal only for $n \geq 0$ for convenience. Since we are assuming that sampling time $T = 1$ sec, then the above discrete time representation becomes,

$$x(n) = \left. \begin{array}{l} e^{-n} \quad \text{for } n \geq 0 \\ = 0 \quad \text{for } n < 0 \end{array} \right\} \quad \dots (2.2.3)$$

Let us evaluate $x(n)$ for $n = 0, 1, 2, \dots$ as follows :

$$\begin{aligned} x(n) &= \{e^0, e^{-1}, e^{-2}, e^{-3}, e^{-4}, e^{-5}, e^{-6}, \dots\} \\ &= \{1, 0.368, 0.135, 0.049, 0.018, 0.0067, \dots\} \quad \dots (2.2.4) \end{aligned}$$

The above array shows the discrete time exponential signal. Fig. 2.2.4 shows the graphical representation of this signal. In this figure observe that on y-axis we are plotting amplitude. On x-axis we are plotting the index or number of the samples. Thus in this representation we do not get any idea of sampling duration and timing parameters. Fig. 2.2.4 is just an array of samples. Equation 2.2.4 also represents an array of samples. From this array we do not get any information about sampling or timing. Thus the discrete time signals are basically arrays of samples.

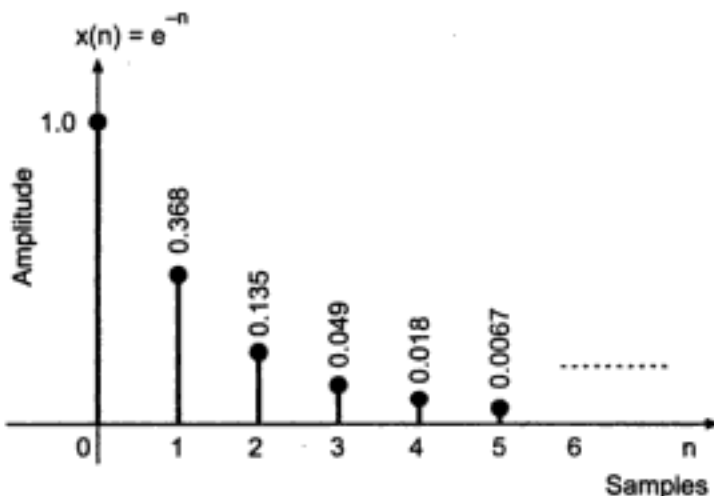


Fig. 2.2.4 Graphical representation of discrete time exponential array of samples

Let us consider the another signal which is discrete time cosine wave.

$$x(n) = \cos(0.3\pi n) \quad \dots (2.2.5)$$

The above signal is simply an array of samples for $n = 0, \pm 1, \pm 2, \pm 3, \pm 4, \pm 5, \dots$ and so on.

The following table illustrates the values of this array for various values of n .

Table 2.2.1 : Samples of discrete time cosine wave

n	$x(n) = (0.3 \pi n)$
:	:
:	:
- 4	$x(-4) = - 0.809$
- 3	$x(-3) = - 0.951$
- 2	$x(-2) = - 0.309$
- 1	$x(-1) = 0.588$
0	$x(0) = 1$
1	$x(1) = 0.588$
2	$x(2) = - 0.309$
3	$x(3) = - 0.951$
4	$x(4) = - 0.809$
5	$x(5) = 0$
6	$x(6) = 0.809$
7	$x(7) = 0.951$
8	$x(8) = 0.309$
9	$x(9) = - 0.588$
10	$x(10) = - 1$
11	$x(11) = - 0.588$
12	$x(12) = 0.309$
:	:
:	:

Thus the analog to digital converter samples cosine wave at fixed rate and it generates the sequence or array of samples as illustrated in Fig. 2.2.5. The array $x(n)$ is then used by the digital signal processor. The sampling frequency for the analog to digital converter or sampler is selected depending upon the maximum frequency content of the input analog signal. Now let us plot the discrete time cosine wave we have obtained in table 2.2.1. The plot is shown in Fig. 2.2.6. In the figure observe that sample values are defined at $n = 0, \pm 1, \pm 2, \pm 3, \pm 4, \dots$ etc. Note that $x(n)$ cannot be defined at $n = 1.5, 3.2, 10.2, \dots$ etc. type of values of 'n'. This is because the sample number 'n' can be only integer. For example what do you interpret from 15^{th} sample? Whereas $1^{\text{st}}, 2^{\text{nd}}, 3^{\text{rd}}, \dots$ etc. samples are defined.

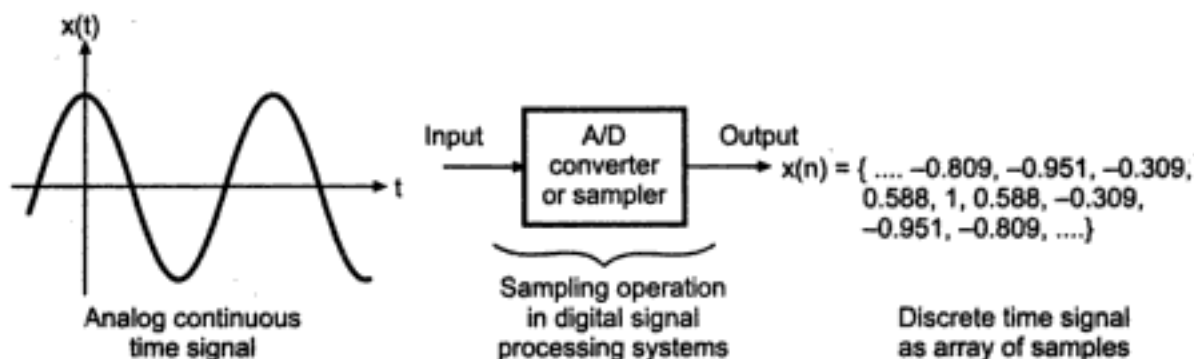


Fig. 2.2.5 A continuous time signal is converted to discrete time signal because of sampling operation in digital signal processing system

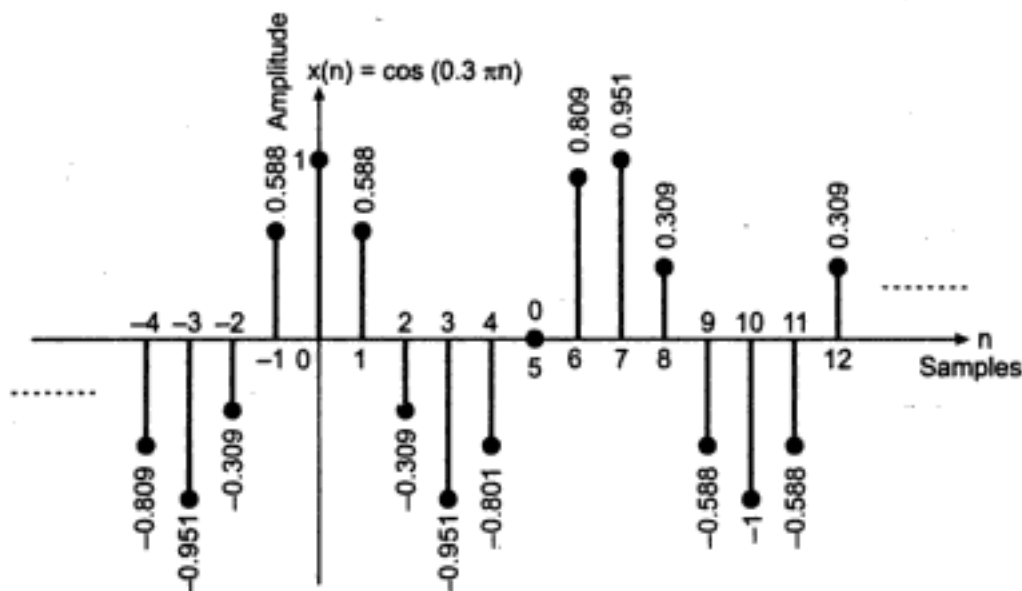


Fig. 2.2.6 Graphical representation of discrete time cosine wave

Sometimes discrete time signals do exist from the source itself. For example if the temperature of Pune city is recorded every year in March, then the discrete time signal resulted is the sequence containing temperatures. In this case sample index 'n' can be the year. It is shown in Table 2.2.2. Thus $x(n)$ represent the sequence which contains temperatures. Note that temperature is actually continuous time signal. But while recording it is sampled every year in March. Hence the recorded signal becomes discrete time in nature. Thus the sampling time for this signal is 1 year. The digital signal processing system can then determine the rise/fall, maximum/minimum etc. values of the temperatures from the sequence.

Table 2.2.2 Temperatures of Pune city taken every year in March

n = Year	x (n) = Temperature in March
:	:
:	:
1988	29.2
1989	30.1
1990	30.4

$n = \text{Year}$	$x(n) = \text{Temperature in March}$
1991	31.0
1992	29.7
1993	31.2
1994	32.0
1995	32.4
1996	32.8
1997	31.5
1998	31.8
1999	32.1
2000	32.2
:	:
:	:

2.2.3 Classification of Signals

Here we will briefly see how various types of signals are classified. The classification is as follows :

- (i) Continuous and discrete time signals
- (ii) Continuous and discrete amplitude signals
- (iii) Deterministic and random signals
- (iv) Digital and analog signals
- (v) Multichannel and multidimensional signals.

Continuous time signal : This signal can be defined at any time instant. The exponential function and sinusoidal function shown in Fig. 2.2.1 are the examples of continuous time signals.

Discrete time signal : This signal is defined only at sampling instants. These signals are basically represented as array of sample values. Fig. 2.2.4 shows the discrete time signal.

Continuous amplitude signals : The amplitude variation is continuous in such signals. Note that the continuous amplitude signals can be discrete or continuous in time. For example the signals in Fig. 2.2.1 and Fig. 2.2.4 are continuous amplitude signals since they can take any amplitude value.

Discrete amplitude signals : These signals take only discrete amplitude levels. Here note that the discrete amplitude signals can be continuous or discrete in time. Fig. 2.2.2 shows a discrete amplitude signal which is continuous in time. Fig. 2.2.7 shows the signal which is discrete in amplitude as well as time.

Please refer Fig. 2.2.7 on next page.

Digital signals : The signals which are discrete in time as well as amplitude are called digital signals. All the signal representation in computers and digital signal processors use digital signals. The digital signal can be binary (one bit), octal (3 bit) hex (4 bit), 16 bit, 32 bit or even 64 bit. The complete amplitude range of the analog signal is represented by these bit lengths. For example, the 4 bit representation will have $2^4 = 16$ levels of amplitude. If the

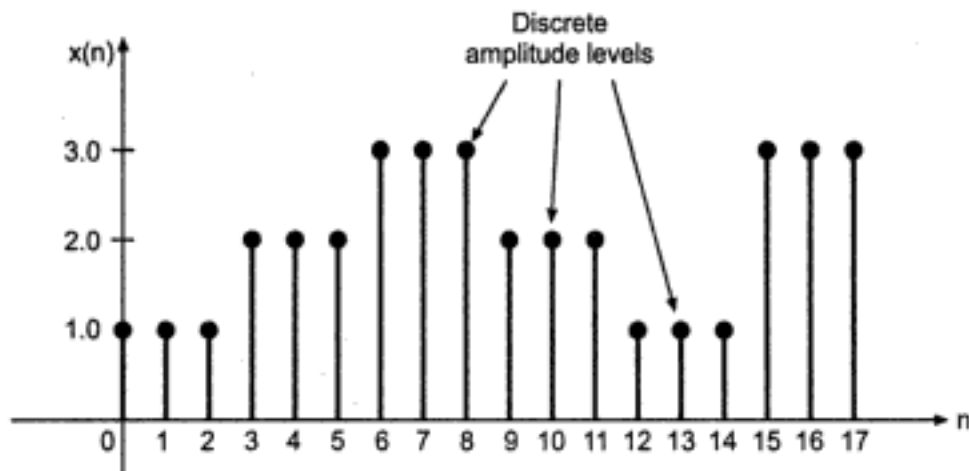


Fig. 2.2.7 A discrete amplitude and discrete time signal

analog signal has the amplitude range of 16 volts peak, then each level will be of one volt. This operation is illustrated in Fig. 2.2.8. In the figure observe that, the amplitude of the analog signal at sampling instant $n=1$ is slightly above 8 volts. It is quantized to nearest level of 8 volts. Similarly at $n=7$, the signal is quantized to a level of 9V. Thus the sampled sequence becomes,

$$x(n) = \{10, 8, 6, 4, 3, 3, 5, 9, 12, 14, 15, 15, 12, 7, 7, \dots\}$$

Here since the amplitude as well as time is discrete, it is called digital signal. This signal can be represented to binary form by simply writing each sample in binary i.e.,

$$x(n) = \{1010, 1000, 0110, 0100, 0011, 0011, 0101, 1001, 1100, \dots\}$$

The above sequence of samples is binary in nature. We will discuss more about sampling and quantization in section 2.9.

Please refer Fig. 2.2.8 on next page.

Analog signals : The signals which are continuous in time as well as amplitude are called analog signals. For example, the exponential function and sinusoidal function shown in Fig. 2.2.1 are examples of analog signals.

Deterministic signals : A signal which is completely described by the mathematical model is called deterministic signal. The value of the deterministic signal can be evaluated at any time (past, present and future) without uncertainty. For example, the sinusoidal signal.

$$x(t) = A \cos \omega t$$

is the deterministic signal since it's value can be calculated at any time precisely.

Random signals : The signals which cannot be described by the mathematical model are called random signals. For example the noise signal or speech signals are random signals. The random signals can be described with the help of their statistical properties.

Multichannel signals : When different signals are recorded from the same source they are called multichannel signals. For example, ECG signal can be recorded in 3 leads or 12 leads for the same person. This results in 3 channel or 12 channel ECG signal. The multichannel signals are useful in studying correlation properties of the source.

Multidimensional signals : When the amplitude of the signal depends upon two or more independent variables, it is called multidimensional signal. For example, the intensity or brightness at any point in the picture or image is the function of its x and y position. Hence it becomes two dimensional signal. The intensity of any point on the TV screen is the function of its x and y positions as well as time. Hence it becomes three dimensional signal.

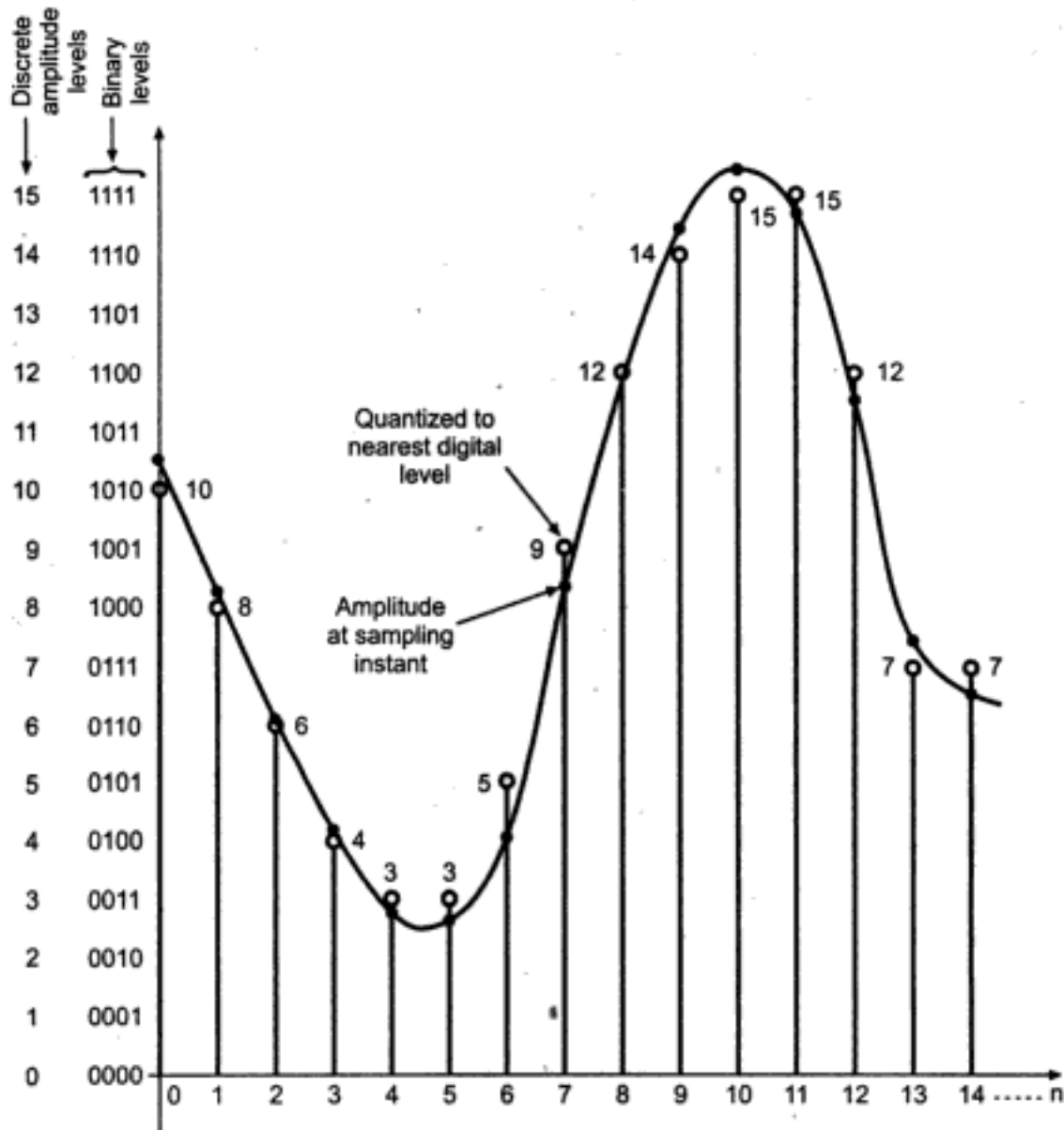


Fig. 2.2.8 Sampling and quantization to get digital signal

We will see more detailed classification of discrete time signals specifically further in this chapter.

2.2.4 Frequency Concept in Discrete Time Signals

In this subsection let us see how frequencies for discrete time signal are represented. We will also see the relationship between sampling frequency, continuous time frequency and discrete time frequency. Let us consider the continuous time cosine wave, which can be expressed as,

$$x_a(t) = A \cos(\Omega t + \theta), -\infty < t < \infty \quad \dots (2.2.6)$$

Here $x_a(t)$ represents analog signal.

A is the amplitude of sinusoid in volts.

and Ω is the frequency of analog sinusoid in radians per second.

θ is the phase in radians.

$$\text{And} \quad \Omega = 2\pi F \quad \dots (2.2.7)$$

Here F is the frequency in cycles per second or Hertz. Hence equation 2.2.6 can be written also as,

$$x_a(t) = A \cos(2\pi Ft + \theta), -\infty < t < \infty \quad \dots (2.2.8)$$

Fig. 2.2.9 shows the cosine wave represented by the signal $x_a(t)$.

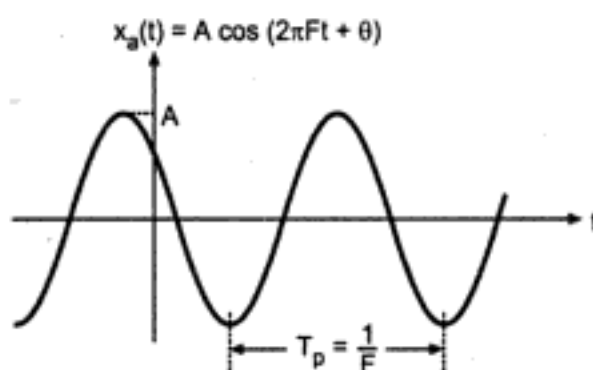


Fig. 2.2.9 An analog cosine wave of amplitude A , frequency F cycles/sec and phase shift θ radians

Since the cosine wave of equation 2.2.8 is periodic with period T_p , we know that any periodic signal satisfies following property :

$$x_a(t) = x_a(t + T_p) \quad \dots (2.2.9)$$

This means the signal repeats after the period T_p . As frequency 'F' is increased or decreased, we get cosine waves of different frequencies in equation 2.2.8. That is all the frequency waveforms will be distinct from each other. For the analog cosine wave of Fig. 2.2.9, it is possible to increase frequency F upto infinity. Similarly frequency can be reduced to zero. Hence the frequency 'F' satisfies the relation : $0 \leq F < \infty$.

Negative frequencies in analog signals :

Sometimes we come across negative frequencies in some mathematical operations. Consider the standard euler's identify,

$$\text{and} \quad \left. \begin{aligned} e^{j\phi} &= \cos \phi + j \sin \phi \\ e^{-j\phi} &= \cos \phi - j \sin \phi \end{aligned} \right\} \quad \dots (2.2.10)$$

We can write equation 2.2.6 in the form of complex exponential functions using above relation. Consider the following expression,

$$\begin{aligned} \frac{A}{2} e^{j(\Omega t + \theta)} + \frac{A}{2} e^{-j(\Omega t + \theta)} &= \frac{A}{2} [\cos(\Omega t + \theta) + j \sin(\Omega t + \theta)] \\ &\quad + \frac{A}{2} [\cos(\Omega t + \theta) - j \sin(\Omega t + \theta)] \end{aligned}$$

$$= \frac{A}{2} \times 2 \cos(\Omega t + \theta)$$

$$= A \cos(\Omega t + \theta)$$

In the above result observe that the RHS is equation 2.2.6. Thus,

$$x_a(t) = A \cos(\Omega t + \theta) = \frac{A}{2} \underbrace{e^{j(\Omega t + \theta)}}_{\text{Positive frequency}} + \frac{A}{2} \underbrace{e^{-j(\Omega t + \theta)}}_{\text{Negative frequency}} \dots (2.2.11)$$

Thus the cosine wave is represent in terms of two equal amplitude complex conjugate exponential functions. The first complex exponential function has positive frequency and second function has negative frequency. Here note that negative frequency as such is not possible in practice. The positive/negative frequencies representation of the signals as in equation 2.2.11 above, is extremely useful in mathematical analysis as we will see in succeeding chapters.

Discrete time signal representation :

Till now we have revised the frequency concepts for analog signals. Now let us see how these concepts can be applied to discrete time signals. The discrete time cosine wave can be expressed as,

$$x(n) = A \cos(\omega n + \theta), -\infty < n < \infty \quad \dots (2.2.11 (a))$$

Here $x(n)$ is the sequence of samples of discrete time cosine wave.

n is the index of the samples.

ω is the frequency in radians per sample.

θ is the phase in radians.

and A is the amplitude of cosine wave.

The frequency ω is expressed as *radians/samples*. Here samples has no unit. It is just the index of the samples. Hence the frequency is also expressed in *radians* only. Thus frequency in radians/sample and frequency in radians have the same meaning. Remember that for continuous time signals, the frequency is represented in radians/sec. Since ω is the angular frequency it can be expressed as,

$$\omega = 2\pi f \quad \dots (2.2.12)$$

Here f is the frequency in cycles per sample. Since again samples has no unit, the frequency f can also be expressed in cycles only. With the help of above result equation 2.2.11 can be written as,

$$x(n) = A \cos(2\pi f n + \theta), -\infty < n < \infty \quad \dots (2.2.13)$$

Ex. 2.2.1 Sketch the discrete time sinusoidal signal which is given as,

$$x(n) = A \cos(\omega n + \theta) \quad \dots (2.2.14)$$

for

$$\omega = \frac{\pi}{6} \text{ radians / sample and}$$

$$\theta = \frac{\pi}{3} \text{ radians.}$$

Sol. : The given signal $x(n)$ is the discrete time cosine wave of frequency ' ω ' and phase shift of ' θ '. Basically the given equation of $x(n)$ is same as that in equation 2.2.1. We know that the period of the repeating signal is 2π radians. Hence let us calculate how many samples of $x(n)$ are present in one period of 2π radians (i.e. one cycle). The given ' ω ' is,

$$\begin{aligned}
 \omega &= \frac{\pi}{6} \text{ radians / sample} \\
 &= \frac{2}{2} \times \frac{\pi}{6} \text{ multiplying numerator and denominator by 2} \\
 &= \frac{2\pi}{12} \frac{\text{radians}}{\text{sample}} \quad \dots (2.2.15)
 \end{aligned}$$

Here observe that there are 12 samples of $x(n)$ in one period (or cycle) of 2π radians. Now let us calculate the number of samples for the phase shift of $\theta = \frac{\pi}{3}$ radians. We know that there are 12 samples of $x(n)$ in one cycle or period of 2π radians. Hence number of samples in $\frac{\pi}{3}$ radians will be,

$$\begin{aligned}
 \text{Phase shift in samples} &= \frac{12 \times \frac{\pi}{3}}{2\pi} \\
 &= 2 \text{ samples.}
 \end{aligned}$$

Here the phase shift is positive ($+\theta$), hence the waveform of $x(n)$ will be phase advanced by 2 samples with respect to 0^{th} sample.

Calculation of sample values of $x(n)$:

Till now we have discussed about number of samples in one cycle and phase shift of $x(n)$. This can be verified by actually calculating samples of $x(n)$. We have,

$$x(n) = A \cos(\omega n + \theta)$$

Putting $\omega = \frac{\pi}{6}$ and $\theta = \frac{\pi}{3}$ in above equation we get,

$$x(n) = A \cos\left(\frac{\pi}{6}n + \frac{\pi}{3}\right)$$

Now let us calculate $x(n)$ for various values of n . The calculations are shown in Table 2.2.3.

Table 2.2.3 : Calculation of discrete time cosine wave

$x(n)$	$x(n) = A \cos\left(\frac{\pi}{6}n + \frac{\pi}{3}\right)$
$x(-6)$	-0.500A
$x(-5)$	0.000A
$x(-4)$	0.500A
$x(-3)$	0.866A
$x(-2)$	1.000A
$x(-1)$	0.866A
$x(0)$	0.500A
$x(1)$	0.000A

$x(n)$	$x(n) = A \cos\left(\frac{\pi}{6}n + \frac{\pi}{3}\right)$
$x(2)$	$-0.500A$
$x(3)$	$-0.866A$
$x(4)$	$-1.000A$
$x(5)$	$-0.866A$
$x(6)$	$-0.500A$
$x(7)$	$0.000A$
$x(8)$	$0.500A$
$x(9)$	$0.866A$
$x(10)$	$1.000A$
$x(11)$	$0.866A$
$x(12)$	$0.500A$
$x(13)$	$0.000A$
$x(14)$	$-0.500A$
$x(15)$	$-0.866A$
\vdots	\vdots
\vdots	\vdots

Fig. 2.2.10 shows the sketch of discrete time cosine wave based on the values calculated in above table.

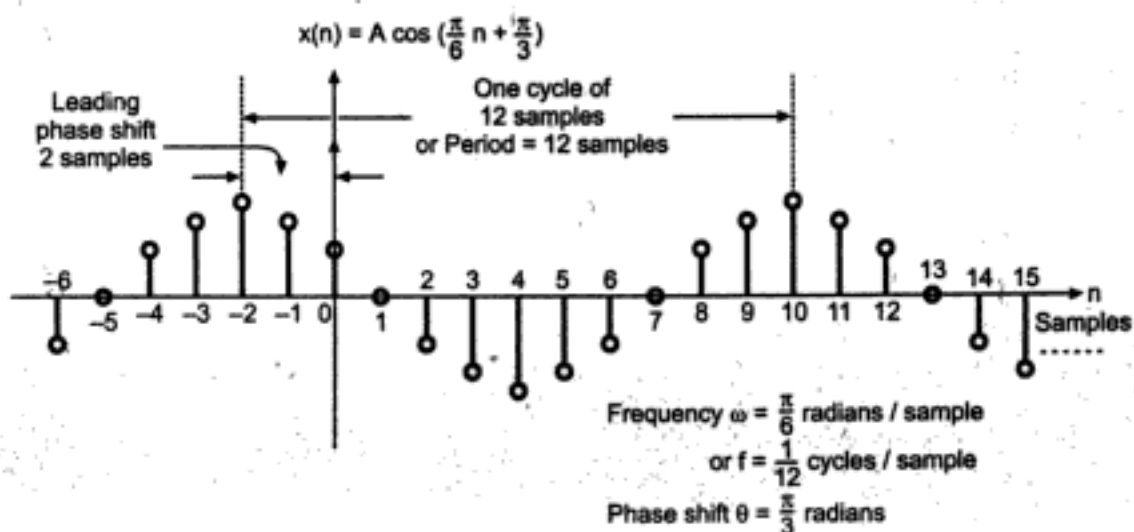


Fig. 2.2.10 A discrete time cosine wave of example 2.2.1

It is clear from the Fig. 2.2.10 that one cycle of $x(n)$ has 12 samples. Similarly the waveform is phase leading by 2 samples. This is same as we have discussed at the beginning of the solution. We know that,

$$\omega = 2\pi f$$

Hence
$$f = \frac{\omega}{2\pi}$$

Here ω is expressed in $\frac{\text{radians}}{\text{sample}}$ and 2π is basically period of one cycle. Hence 2π can be called as $\frac{\text{radians}}{\text{cycle}}$ means 2π radians per cycle. Hence unit of ' f ' becomes,

$$\begin{aligned} f &= \frac{\omega \text{ radians / sample}}{2\pi \text{ radians / cycle}} \\ &= \frac{\omega}{2\pi} \text{ cycles / sample} \end{aligned}$$

Thus the unit of ' f ' as cycles per samples which we have mentioned earlier. Putting value of $\omega = \frac{\pi}{6}$ in above equation we get,

$$f = \frac{\pi/6}{2\pi}$$

$\therefore f = \frac{1}{12}$ cycles / sample

Thus the frequency of discrete time cosine wave of Fig. 2.2.10 is $\frac{1}{12}$ cycles per sample. Since samples do not have any unit, the frequency ' f ' can be called simply as $\frac{1}{12}$ cycles also.

To calculate period 'N' of $x(n)$:

Now let us calculate period 'N' of the discrete time signal. i.e.,

$$\text{* Period } N = \frac{1}{\text{frequency } f} \quad \dots (2.2.16)$$

In the above equation putting the value of $f = \frac{1}{12}$ cycles / samples,

$$\text{Period } N = \frac{1}{\frac{1 \text{ cycles}}{12 \text{ sample}}} = 12 \text{ samples / cycles}$$

Thus the period is 12 samples per cycle as indicated in Fig. 2.2.10.

* It is defined in next example in equation 2.2.18 that a signal is periodic only if $f = \frac{k}{N}$. Here k and N are integers. N is the period. Equation 2.2.16 holds only when $k = 1$. Since $N = \frac{k}{f} = \frac{1}{f}$. For simplicity this relation is given.

Ex. 2.2.2 Show that the discrete time sinusoidal signal is periodic only if its frequency (f_0) can be expressed as the ratio of two integers i.e. if ' f_0 ' is rational.

Sol. : The discrete time signal is periodic only if,

$$x(n+N) = x(n) \text{ for all } n \quad \dots (2.2.17)$$

Here 'N' is the period of $x(n)$ in samples. Now let us consider the cosine wave signal,

$$x(n) = A \cos(2\pi f_0 n + \theta)$$

Hence $x(n+N)$ in the above equation becomes,

$$\begin{aligned} x(N+n) &= A \cos[2\pi f_0(N+n) + \theta] \\ &= A \cos[2\pi f_0 N + 2\pi f_0 n + \theta] \end{aligned}$$

For periodicity $x(N+n) = x(n)$ That is,

$$A \cos[2\pi f_0 N + 2\pi f_0 n + \theta] = A \cos(2\pi f_0 n + \theta)$$

The above equation is satisfied only if, $2\pi f_0 N$ is an integer multiple of 2π i.e.,

$$2\pi f_0 N = 2\pi k$$

Here k is some integer. Solving for f_0 , above equation becomes,

$$f_0 = \frac{k}{N} \quad \dots (2.2.18)$$

This result shows that the discrete time sinusoidal signal is periodic only if its frequency f_0 is rational. For example,

if
$$f_0 = \frac{30}{60} = \frac{1}{2}$$

Here $k=1$ and, $N=2$ is the fundamental period. Thus f_0 is expressed as the ratio of k and N such that no further simplification of $\frac{k}{N}$ is possible.

The discussion of the above example can be continued further and we can state that when the frequencies of discrete time sinusoids are separated by integer multiple of 2π , they cannot be distinguished from each other. For example let $x_1(n) = \cos(\omega_0 n + \theta)$.

Let $x_2(n)$ have frequency of $\omega_0 + 2\pi$, then we can write,

$$\begin{aligned} x_2(n) &= \cos[(\omega_0 + 2\pi)n + \theta] \\ &= \cos[\omega_0 n + 2\pi n + \theta] \\ &= \cos[\omega_0 n + \theta] = x_1(n) \end{aligned}$$

Thus $x_1(n)$ and $x_2(n)$ become same.

Ex. 2.2.3 Consider the following sinusoidal signal,

$$x(n) = \cos(\omega_0 n)$$

Sketch the discrete time sequence of $x(n)$ for following frequencies :

(i) $\omega_0 = 0$, (ii) $\omega_0 = \frac{\pi}{8}$, (iii) $\omega_0 = \frac{\pi}{4}$, (iv) $\omega_0 = \frac{\pi}{2}$ and (v) $\omega_0 = \pi$ radians.

Sol. : The values of samples can be obtained by values of ω_0 in the equation $x(n) = \cos(\omega_0 n)$. Table 2.2.4 shows the sample values of $x(n)$ for various values of ω_0 .

Table 2.2.4 : Sample values of $x(n) = \cos(\omega_0 n)$ for various values of ω_0

n	$\omega_0 = 0$	$\omega_0 = \frac{\pi}{8}$	$\omega_0 = \frac{\pi}{4}$	$\omega_0 = \frac{\pi}{2}$	$\omega_0 = \pi$
$x(-15)$	1.000	0.924	0.707	0.000	- 1.000
$x(-14)$	1.000	0.707	0.000	- 1.000	1.000
$x(-13)$	1.000	0.383	- 0.707	- 0.000	- 1.000
$x(-12)$	1.000	0.000	- 1.000	1.000	1.000
$x(-11)$	1.000	- 0.383	- 0.707	0.000	- 1.000
$x(-10)$	1.000	- 0.707	- 0.000	- 1.000	1.000
$x(-9)$	1.000	- 0.924	0.707	- 0.000	- 1.000
$x(-8)$	1.000	- 1.000	1.000	1.000	1.000
$x(-7)$	1.000	- 0.924	0.707	0.000	- 1.000
$x(-6)$	1.000	- 0.707	0.000	- 1.000	1.000
$x(-5)$	1.000	- 0.383	- 0.707	- 0.000	- 1.000
$x(-4)$	1.000	- 0.000	- 1.000	1.000	1.000
$x(-3)$	1.000	0.383	- 0.707	0.000	- 1.000
$x(-2)$	1.000	0.707	- 0.000	- 1.000	1.000
$x(-1)$	1.000	0.924	0.707	- 0.000	- 1.000
$x(0)$	1.000	1.000	1.000	1.000	1.000
$x(1)$	1.000	0.924	0.707	- 0.000	- 1.000
$x(2)$	1.000	0.707	- 0.000	- 1.000	1.000
$x(3)$	1.000	0.383	- 0.707	0.000	- 1.000
$x(4)$	1.000	- 0.000	- 1.000	1.000	1.000
$x(5)$	1.000	- 0.383	- 0.707	- 0.000	- 1.000
$x(6)$	1.000	- 0.707	0.000	- 1.000	1.000
$x(7)$	1.000	- 0.924	0.707	0.000	- 1.000
$x(8)$	1.000	- 1.000	1.000	1.000	1.000
$x(9)$	1.000	- 0.924	0.707	- 0.000	- 1.000
$x(10)$	1.000	- 0.707	- 0.000	- 1.000	1.000
$x(11)$	1.000	- 0.383	- 0.707	0.000	- 1.000
$x(12)$	1.000	0.000	- 1.000	1.000	1.000
$x(13)$	1.000	0.383	- 0.707	- 0.000	- 1.000
$x(14)$	1.000	0.707	0.000	- 1.000	1.000
$x(15)$	1.000	0.924	0.707	0.000	- 1.000
:	:	:	:	:	:

Fig. 2.2.11 shows the sketch of $x(n)$ for various frequencies of ω_0 based on the values calculated in the above table.

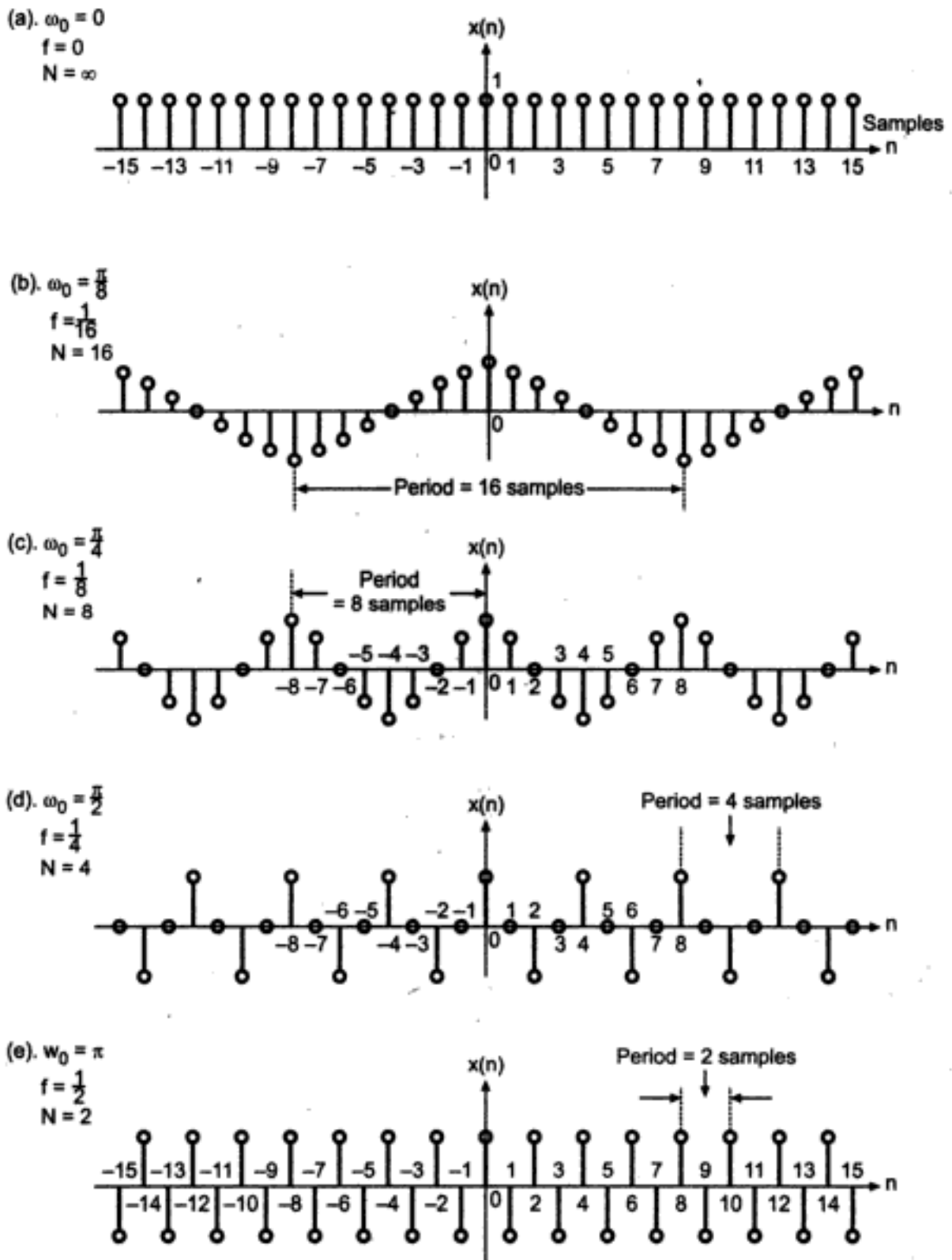


Fig. 2.2.11 Sketches of discrete time cosine wave for various values of frequency. Fig. 2.2.11 (a) shows sketch for frequency $\omega_0 = 0$, i.e. DC signal

Now let us interpret the sketches of $x(n)$ for various frequencies. The $\omega_0 = 0$ is actually dc signal. Hence the value of $x(n) = 1$ for all the sample values. This sketch is shown in Fig. 2.2.11(a). The next frequency is $\omega_0 = \frac{\pi}{8}$. It can be rearranged as,

$$\omega_0 = \frac{2\pi}{2 \times 8} = \frac{2\pi}{16}$$

Thus there are 16 samples in one cycle. This sketch is indicated in Fig. 2.2.11 (b). We have f_0 related to ω_0 as,

$$\begin{aligned} \omega_0 &= 2\pi f_0 \\ \therefore f_0 &= \frac{\omega_0}{2\pi} = \frac{\pi/8}{2\pi} \text{ putting value of } \omega_0 = \frac{\pi}{8} \\ \therefore f_0 &= \frac{1}{16} \text{ cycles per sample.} \end{aligned}$$

Hence period = $\frac{1}{f_0} = \frac{1}{1/16} = 16$ samples as shown in Fig. 2.2.11 (b).

Fig. 2.2.11 (c) shows the sketch of discrete time cosine wave for $\omega_0 = \frac{\pi}{4}$ radians per sample. And $f = \frac{1}{8}$ cycles per sample for this signal. As shown the period $N = 8$ samples. It can be calculated from frequency ' f ' also. i.e.,

$$f = \frac{k}{N} = \frac{1}{8} \text{ i.e. Period } N = 8 \text{ samples.}$$

This indicates that as frequency increases, the period reduces.

Further, Fig. 2.2.11 (d) shows the sketch of discrete time cosine wave for $\omega_0 = \frac{\pi}{2}$. Observe that the period is further reduced. There are only four samples of the cosine wave in one period of cycle.

Fig. 2.2.11 (e) shows the sketch of the waveform for $\omega_0 = \pi$ radians / sample. Here $f = \frac{1}{2}$ cycles per sample. The period $N = 2$ samples. This means in one cycle only two samples of the waveform are used. This also shows that the frequency ω_0 cannot be increased more than ' π ' since the waveform will not be represented. Observe in Fig. 2.2.11 (e) that it is not possible to reduce the period less two samples, since alternate samples are positive and negative.

Conclusion :

Based on the discussion of above example we can conclude that ; for the discrete time sinusoids, the maximum frequency is $\omega = \pi$ radians / sample or $f = \frac{1}{2}$ cycles / sample. It can also be shown easily that maximum negative value of ' ω ' will be ' $-\pi$ ' and that of ' f ' will be ' $-\frac{1}{2}$ '.

Just now we have established that frequency range in which sinusoids can be represented is limited to $-\pi$ to π . Now let us consider that we have the signal,

$$x_1(n) = \cos(\omega_1 n) \quad \pi \leq \omega_1 \leq 2\pi \quad \dots (2.2.19)$$

Here we have considered that the frequency ω_1 is between π to 2π . Let us select another sinusoid,

$$x_2(n) = \cos(\omega_2 n) \quad \dots (2.2.20)$$

Such that $\omega_2 = 2\pi - \omega_1$. Since $\pi \leq \omega_1 \leq 2\pi$, the range of ω_2 will be $0 \leq \omega_2 \leq \pi$. Putting $\omega_2 = 2\pi - \omega_1$ in the above equation we get,

$$\begin{aligned} x_2(n) &= \cos[(2\pi - \omega_1)n] \\ &= \cos[2\pi n - \omega_1 n] \\ &= \cos(-\omega_1 n) \\ &= \cos(\omega_1 n) \\ &= x_1(n) \end{aligned} \quad \dots (2.2.21)$$

Thus $x_2(n)$ is same as $x_1(n)$. The frequency of $x_1(n)$ is in the range $0 \leq \omega_1 \leq \pi$. We have seen that frequencies in this range are represented. But ω_2 has the range of $\pi \leq \omega_2 \leq 2\pi$, which is outside of the maximum range that can be represented. Equation 2.2.21 shows that $x_2(n)$ becomes same as $x_1(n)$. In this effect, high frequency ($> \pi$) takes the form of low frequency ($< \pi$). This effect is called *aliasing*. Here ω_2 is called alias of ω_1 . We will see more about the concept of aliasing further in the section on sampling of signals.

Ex. 2.2.4 A discrete time signal $x(n)$ is defined as,

$$x(n) = 1 + \frac{n}{3} \quad \text{for } -3 \leq n \leq -1$$

$$x(n) = 1 \quad \text{for } 0 \leq n \leq 3$$

$$x(n) = 0 \quad \text{elsewhere}$$

Determine its values and sketch the signal $x(n)$

[Dec - 99]

Sol. : $x(n) = 1 + \frac{n}{3}$ for $-3 \leq n \leq -1$:

$$\therefore n = -3 \Rightarrow x(n) = 1 + \frac{-3}{3} = 0$$

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

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

$x(n) = 1$ for $0 \leq n \leq 3$:

$$\therefore x(0) = 1$$

$$x(1) = 1$$

$$x(2) = 1$$

$$x(3) = 1$$

$x(n) = 0$ elsewhere :

i.e. for $n \leq -4, -5, -6, \dots$ etc $x(n) = 0$

and for $n \geq 4, 5, 6, \dots$ etc $x(n) = 0$

Hence the sequence $x(n)$ becomes,

$$x(n) = \left\{ \dots, 0, \frac{1}{3}, \frac{2}{3}, \underset{\uparrow}{1}, 1, 1, 1, 0, \dots \right\}$$

Fig. 2.2.12 shows the sketch of the above signal.

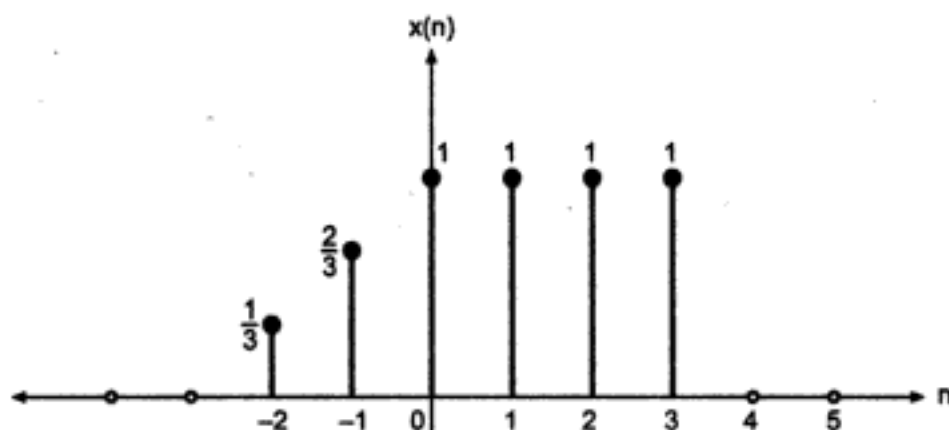


Fig. 2.2.12 Sketch of the sequence $x(n)$ of example 2.2.4

Ex. 2.2.5 Few discrete time sequences are given below :

(i) $\cos(0.01\pi n)$ (ii) $\cos(3\pi n)$ (iii) $\sin(3n)$ (iv) $\cos\left(\frac{n}{8}\right)\cos\left(\frac{\pi n}{8}\right)$

Determine whether they are periodic or nonperiodic. If the sequence is periodic, determine its fundamental period.

Sol. : In example 2.2.2 we have seen that, the discrete time signal is periodic if its frequency ' f ' can be expressed as ratio of two integers. i.e.,

$$f = \frac{k}{N} \quad \text{from equation 2.2.18}$$

(i) $\cos(0.01\pi n)$:

Here $\omega = 0.01\pi$

Since $\omega = 2\pi f$, we have $f = \frac{\omega}{2\pi}$

$$\therefore f = \frac{0.01\pi}{2\pi}$$

$$= \frac{1}{200} \quad \text{cycles per sample}$$

Here $k=1$ and $N=200$. Thus ' f ' is expressed as ratio of two integers. Hence given sequence is periodic. Here period, $N=200$ samples. The relation $N = \frac{1}{f}$ holds only when $k=1$.

Earlier we have considered all signals which have $k=1$. Hence we have used the relation $N = \frac{1}{f}$. But this is not always true.

(ii) $\cos 3\pi n$:

Here $\omega = 3\pi$

We have $f = \frac{\omega}{2\pi}$

$$\therefore f = \frac{3\pi}{2\pi} = \frac{3}{2}$$

We know that $f = \frac{k}{N}$ and if k and N are integers, then signal is periodic with period ' N '.

Thus the given signal is periodic with period $N = 2$ samples.

(iii) $\sin(3n)$

Here $\omega = 3$

We have $f = \frac{\omega}{2\pi}$

$$\therefore f = \frac{3}{2\pi}$$

Here $k = 3$ and $N = 2\pi$, which is not an integer. Thus ' f ' cannot be expressed as the ratio of two integers. Therefore this signal is nonperiodic.

(iv) $\cos\left(\frac{n}{8}\right)\cos\left(\frac{\pi n}{8}\right)$:

Here we can write given sequence as,

$\cos \omega_1 \cos \omega_2$

and $\omega_1 = \frac{1}{8}, \omega_2 = \frac{\pi}{8}$

$$\begin{aligned} \therefore f_1 &= \frac{\omega_1}{2\pi} \\ &= \frac{1/8}{2\pi} \\ &= \frac{1}{16\pi} \end{aligned}$$

Here $k_1 = 1$ and $N_1 = 16\pi$, which is not an integer. Hence f_1 is not the ratio of two integers. Therefore $\cos\left(\frac{n}{8}\right)$ is nonperiodic.

$$\begin{aligned} \text{Now } f_2 &= \frac{\omega_2}{2\pi} \\ &= \frac{\pi/8}{2\pi} \\ &= \frac{1}{16} \end{aligned}$$

' f_2 ' is the ratio of two integers with $N_2 = 16$. Hence $\cos\left(\frac{\pi}{8}\right)$ is periodic. The given signal is thus the product of nonperiodic signal and periodic signal. Hence the product signal $\cos\left(\frac{n}{8}\right)\cos\left(\frac{\pi n}{8}\right)$ is nonperiodic.

2.3 Standard Discrete Time Signals (Sequences)

In this section we will consider some of the standard discrete time signals and their representation. These signals are discrete time sinusoids, unit sample sequence, unit step signal, unit ramp signal and exponential signal etc. These standard signals are extremely useful in the analysis of discrete time systems. We have discussed discrete time sinusoids such as $x(n) = A \cos(\omega_0 n)$ or $x(n) = A \sin(\omega_0 n)$ in detail in last section. Hence we will see the representation of other standard signals in this section.

2.3.1 Unit Sample Sequence

The unit sample sequence is represented by $\delta(n)$. It is very much similar to the standard unit impulse signal $\delta(t)$. The unit sample sequence $\delta(n)$ is defined as,

$$\left. \begin{aligned} \delta(n) &= 1 && \text{for } n = 0 \\ &= 0 && \text{for } n \neq 0 \end{aligned} \right\} \quad \dots (2.3.1)$$

In the sequence form it can be represented as,

$$\delta(n) = \{\dots, 0, 0, 0, \underset{\uparrow}{1}, 0, 0, 0, \dots\} \quad \dots (2.3.2)$$

In the above sequence ' \uparrow ' this arrow represents 0^{th} sample. The above sequence can also be represented simply as,

$$\delta(n) = \{1\} \quad \dots (2.3.3)$$

Fig. 2.3.1 shows the sketch of unit sample sequence.

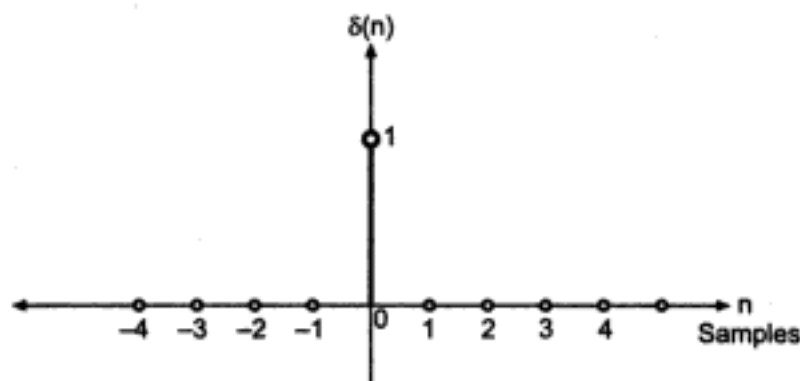


Fig. 2.3.1 Graphical representation of unit sample sequence

2.3.2 Unit Step Sequence

The unit step sequence is denoted by $u(n)$ and all its samples have a value of '1' for $n \geq 0$. i.e.,

$$\left. \begin{aligned} u(n) &= 1 && \text{for } n \geq 0 \\ &= 0 && \text{for } n < 0 \end{aligned} \right\} \dots (2.3.4)$$

Thus the unit step sequence of ten samples can be written as,

$$u(n) = \{0, 0, \underset{\uparrow}{1}, 1, 1, 1, 1, 1, 1, 1, 1\}$$

Here the '↑' arrow indicates 0th sample. Or it can also be written as,

$$u(n) = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}$$

As shown in the above representation an '↑' arrow is absent. In such representation, the first sample is considered as the 0th sample. Fig. 2.3.2 shows the sketch the unit step sequence.

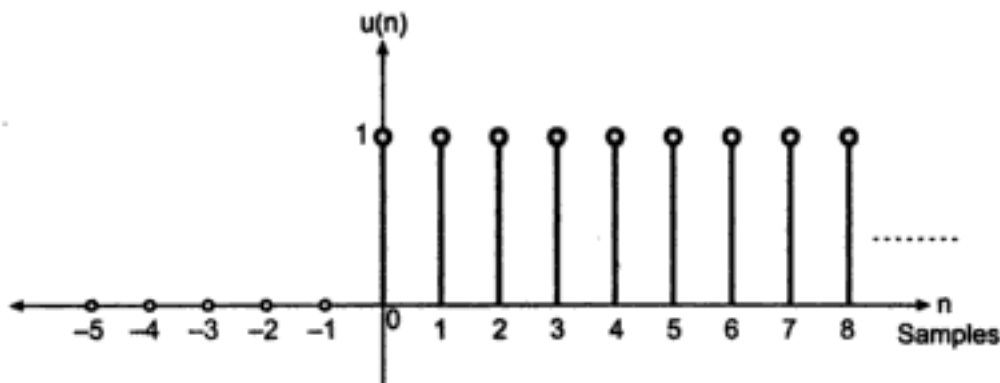


Fig. 2.3.2 Graphical representation of the unit step signal

The unit step sequence is similar to standard unit step signal $u(t)$ used for continuous time systems. Hence unit step sequence can also be called as discrete time representation of unit step signal.

2.3.3 Unit Ramp Sequence

The unit ramp sequence is denoted by $u_r(n)$. Its value increases with the sample number 'n' linearly. It is defined as,

$$\left. \begin{aligned} u_r(n) &= n && \text{for } n \geq 0 \\ &= 0 && \text{for } n < 0 \end{aligned} \right\} \dots (2.3.5)$$

The sequence for a unit ramp of 10 samples can be written as follows.

$$u_r(n) = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

Fig. 2.3.3 shows the sketch of unit ramp sequence.

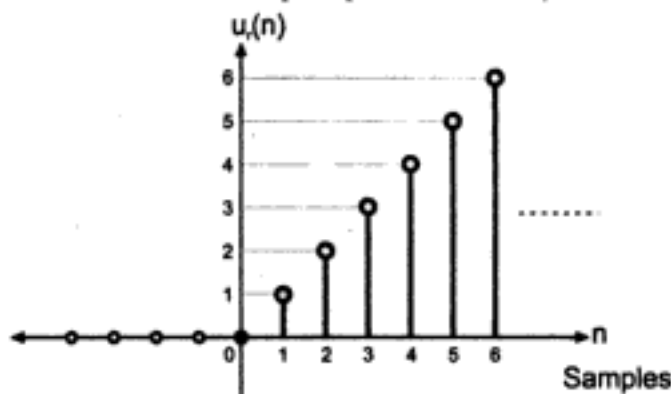


Fig. 2.3.3 A sketch of unit ramp sequence

The unit ramp sequence is similar to the unit ramp signal $r(t)$. In the above figure observe that the amplitudes of the samples increase linearly with the number of the sample.

2.3.4 Exponential Sequence

The exponential sequence can be represented as,

$$x(n) = a^n \quad \dots (2.3.6)$$

Fig. 2.3.4 (a) shows the plot of a^n for $0 < a < 1$. Observe that it is the decaying signal. If $a > 1$, then $x(n)$ becomes the rising exponential sequence. The exponential sequence can be real or complex valued. If 'a' is complex valued, then it can be represented as,

$$a = r e^{j\theta} \quad \dots (2.3.7)$$

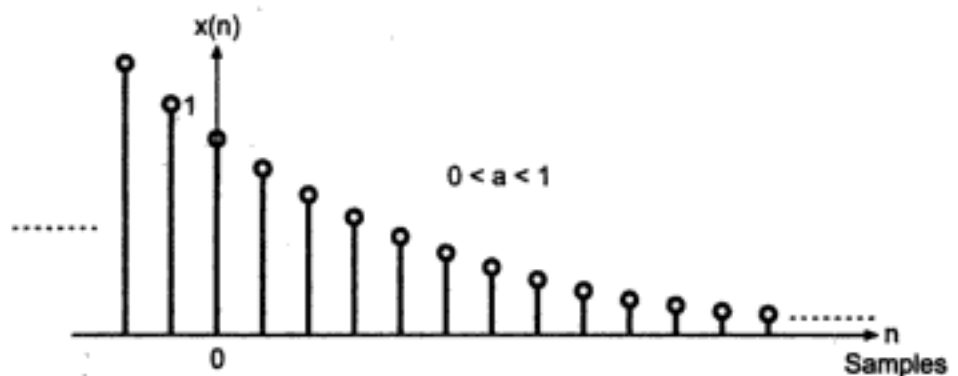


Fig. 2.3.4 (a) A sketch of decaying exponential sequence

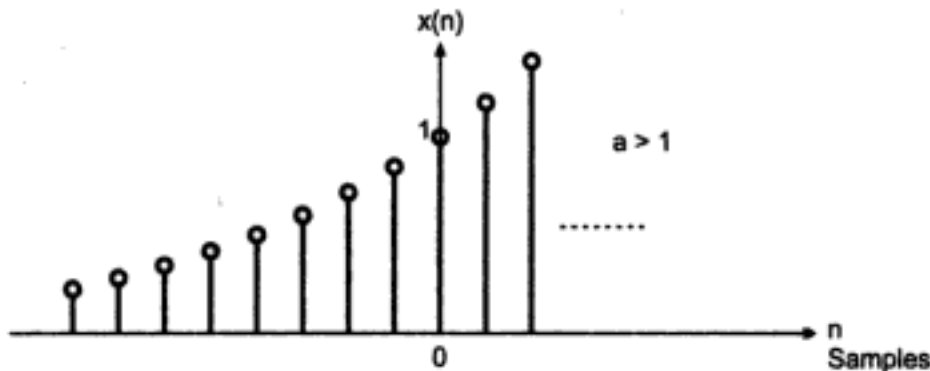


Fig. 2.3.4 (b) A sketch of rising exponential sequence

Here 'r' is the magnitude of 'a' and 'θ' is the angle of 'a'. Hence the sequence $x(n)$ of equation 2.3.6 becomes,

$$\begin{aligned} x(n) &= a^n \\ &= [r e^{j\theta}]^n \quad \text{Since } a = r e^{j\theta} \\ &= r^n e^{j\theta n} \quad \dots (2.3.8) \end{aligned}$$

Here let us use euler's identity : $e^{j\theta} = \cos \theta + j \sin \theta$. Hence above equation becomes,

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

Thus each sample of sequence $x(n)$ has real part and imaginary part. i.e.

$$\left. \begin{array}{l} \text{Real part of } x(n) \Rightarrow x_R(n) = r^n \cos(\theta n) \\ \text{and Imaginary part of } x(n) \Rightarrow x_I(n) = r^n \sin(\theta n) \end{array} \right\} \dots (2.3.9)$$

Similarly from equation 2.3.8 we can write magnitude and phase of $x(n)$ as,

$$\text{Magnitude of } x(n) \Rightarrow |x(n)| = A(n) = r^n \dots (2.3.10)$$

$$\text{Phase of } x(n) \Rightarrow \angle x(n) = \phi(n) = \theta n \dots (2.3.11)$$

The above two results can also be obtained from equation 2.3.9.

Ex. 2.3.1 Prove the following

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

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

$$(iii) u(n) = \sum_{k=0}^{\infty} \delta(n-k)$$

Sol. : (i) $\delta(n) = u(n) - u(n-1)$

We know that

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

And
$$u(n-1) = \begin{cases} 1 & \text{for } n \geq 1 \\ 0 & \text{for } n < 1 \end{cases}$$

Hence,
$$u(n) - u(n-1) = \begin{cases} 0 & \text{for } n \geq 1 \text{ i.e. } n > 0 \\ 1 & \text{for } n = 0 \\ 0 & \text{for } n < 0 \end{cases}$$

The above equation is nothing but $\delta(n)$. i.e.,

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

$$(ii) u(n) = \sum_{k=-\infty}^n \delta(k) :$$

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

The right hand side of above equation is unit sample sequence $u(n)$. Hence the equation is proved.

$$(iii) u(n) = \sum_{k=0}^{\infty} \delta(n-k) :$$

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

The right hand side of above equation is unit sample sequence $u(n)$. Hence the given equation is proved.

2.4 Classification of Discrete Time Signals

Earlier we have seen the general classification of signals. The discrete time signals are one of them. Thus the discrete time signals are classified as follows :

- (i) Energy signals and power signals
- (ii) Periodic signals and nonperiodic signals
- (iii) Even and odd signals
- (iv) Multichannel and multidimensional signals.
- (v) Deterministic and random signals.

Actually all the above types of signals exists in continuous time signals also. Let us define briefly the above types of signals.

2.4.1 Energy Signals and Power Signals

The energy of the discrete time signal $x(n)$ is denoted by E . It is given as,

$$\text{Energy, } E = \sum_{n=-\infty}^{\infty} |x(n)|^2 \quad \dots (2.4.1)$$

Thus the energy of the signal is equal to summation of magnitude squared values of $x(n)$. Observe that the summation is carried out over all the samples of $x(n)$. The signal $x(n)$ is called energy signal if its energy is finite i.e.,

$$x(n) \text{ is energy signal if, } 0 < E < \infty \quad \dots (2.4.2)$$

The average power of the discrete time signal $x(n)$ is denoted by ' P '. It is given as,

$$\text{Power, } P = \lim_{N \rightarrow \infty} \frac{1}{(2N+1)} \sum_{n=-N}^N |x(n)|^2 \quad \dots (2.4.3)$$

Thus in the above equation, it is desirable that ' N ' should be very large. The signal $x(n)$ is called power signal if its power is finite i.e.,

$$x(n) \text{ is power signal if, } 0 < P < \infty \quad \dots (2.4.3 (a))$$

There are few signals, which are neither energy signals nor power signals.

2.4.2 Periodic Signals and Non-periodic Signals

The signal $x(n)$ is said to be periodic if,

$$x(n+N) = x(n) \quad \text{for all } n \quad \dots (2.4.4)$$

Here ' N ' is the period of the signal. If the signal $x(n)$ do not satisfy above property for all values of n , then the signal is said to be non periodic. In example 2.2.2, we have seen earlier that a discrete time sinusoid is periodic for all values of n only if its frequency f_0 is rational. That is if the frequency f_0 can be expressed as the ratio of,

$$f_0 = \frac{k}{N} \quad \dots (2.4.5)$$

Here N is the fundamental period and ' k ' is some integer.

2.4.3 Even and Odd Signals or Symmetric and Antisymmetric Signals

The signal $x(n)$ is said to be even or symmetric if,

$$x(-n) = x(n) \quad \dots (2.4.6)$$

For example consider the signal shown in Fig. 2.4.1.

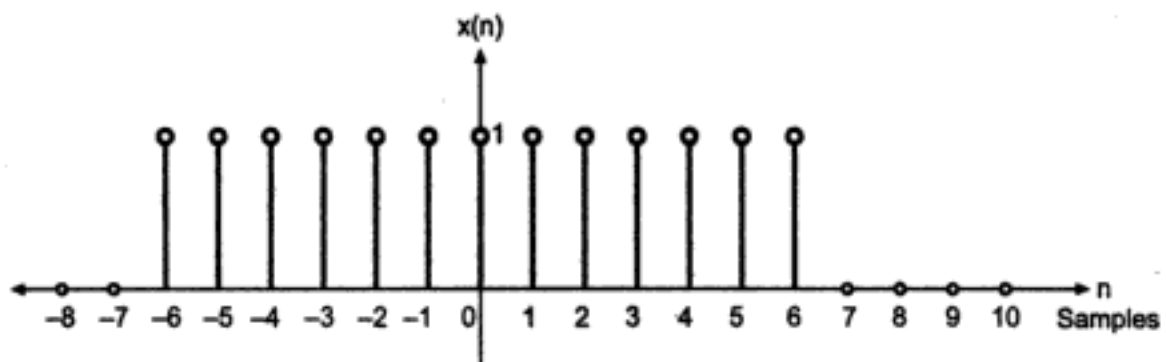


Fig. 2.4.1 Sketch of an even or symmetric signal

In this figure observe that,

$$x(-1) = x(1)$$

$$x(-2) = x(2)$$

$$x(-3) = x(3)$$

$$\vdots \quad \vdots$$

$$\vdots \quad \vdots$$

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

Hence it is even or symmetric signal.

The signal $x(n)$ is called odd or antisymmetric if

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

... (2.4.7)

For example consider the signal shown in Fig. 2.4.2.

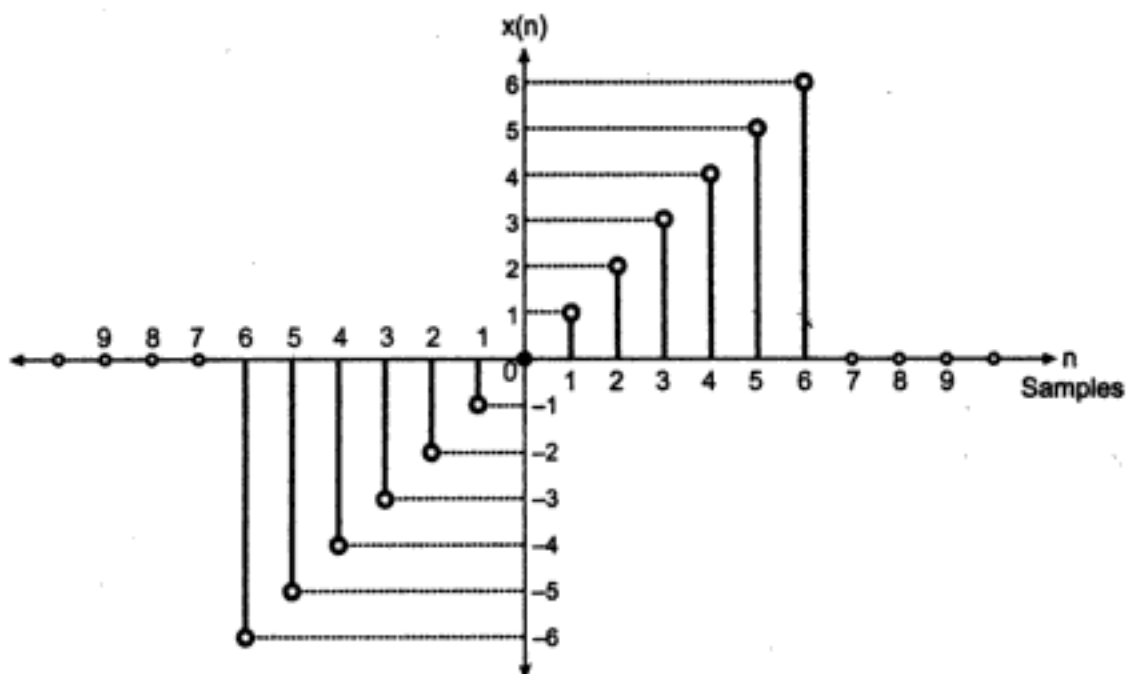


Fig. 2.4.2 A sketch of odd or antisymmetric signal

In the Fig. 2.4.2 it is clear that

$$x(-1) = -x(1)$$

$$x(-2) = -x(2)$$

$$\vdots$$

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

Hence it is odd or antisymmetric signal.

Earlier we have seen multidimensional and multichannel signals. We have also seen deterministic and random signals in section 2.2.3. The same discussion applies to discrete time signals also.

2.5 Discrete Time Systems

Till now we were discussing about discrete time signals and their classification. In this section we will discuss about discrete time systems. The discrete time system is a device or algorithm that performs some prescribed operation on the discrete time signal. Thus the discrete time system has an input or excitation and the output or response as shown in Fig. 2.5.1.

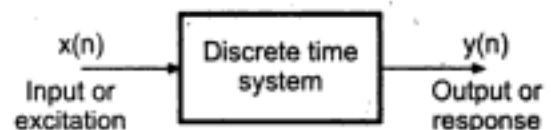


Fig. 2.5.1 A discrete time system

As shown in the figure, $y(n)$ is the response to the excitation $x(n)$. The input output relationship for a discrete time system is represented as,

$$y(n) = T[x(n)] \quad \dots (2.5.1)$$

or $x(n) \xrightarrow{T} y(n) \quad \dots (2.5.2)$

Here 'T' represents transformation operation. This transformation operation depends upon the characteristics of the discrete time system. Let us consider the following example which illustrates the data processing in discrete time system.

Ex. 2.5.1 Consider the discrete time system of Fig. 2.5.1, is excited by following sequence,

$$\left. \begin{aligned} x(n) &= 1 && \text{for } 0 \leq n \leq 3 \\ &= 0 && \text{elsewhere} \end{aligned} \right\} \quad \dots (2.5.3)$$

Find out the response $y(n)$ if $x(n)$ and $y(n)$ are related by following relations :

- (i) $y(n) = x(-n)$
- (ii) $y(n) = x(n-1)$
- (iii) $y(n) = x(n+1)$
- (iv) $y(n) = x(n-1) + x(n+1)$
- (v) $y(n) = 2x(n)$

Sol. : Since this is the first example on digital signal processing in a discrete time system we will study it graphically as well as analytically.

(i) $y(n) = x(-n)$: **Folding operation :**

This is basically folding of the sequence. Let us sketch the sequence $x(n)$. It is shown in Fig. 2.5.2.

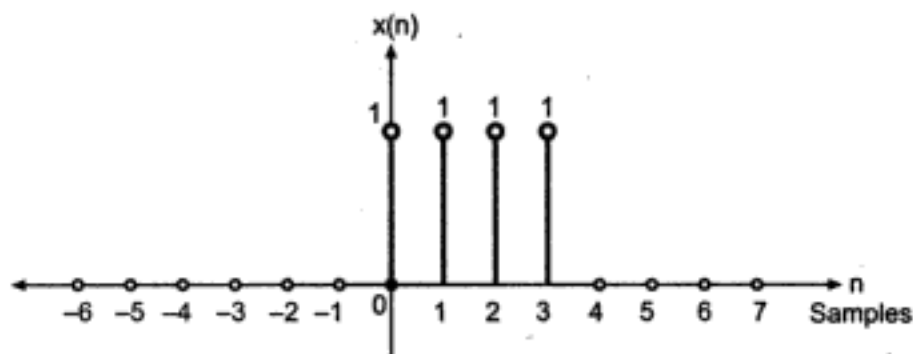


Fig. 2.5.2 Graphical representation of input sequence $x(n)$

Thus it is clear from above figure that $x(n) = 1$ for $0 \leq n \leq 3$. $x(n) = 0$ for $n = 4, 5, 6, \dots \infty$. Similarly $x(n) = 0$ for $n = -1, -2, -3, -4, \dots \infty$. Now we have $y(n) = x(-n)$. Hence,

$$\begin{aligned} n = 0 &\Rightarrow y(0) = x(0) = 1 \\ n = 1 &\Rightarrow y(1) = x(-1) = 0 \\ n = 2 &\Rightarrow y(2) = x(-2) = 0 \\ n = 3 &\Rightarrow y(3) = x(-3) = 0 \quad \dots \text{and so on.} \end{aligned}$$

Similarly

$$\begin{aligned} n = -1 &\Rightarrow y(-1) = x[-(-1)] = x(1) = 1 \\ n = -2 &\Rightarrow y(-2) = x(2) = 1 \\ n = -3 &\Rightarrow y(-3) = x(3) = 1 \\ n = -4 &\Rightarrow y(-4) = x(4) = 0 \\ n = -5 &\Rightarrow y(-5) = x(5) = 0 \quad \dots \text{and so on.} \end{aligned}$$

The sketch of $y(n)$ based on above calculations is shown in Fig. 2.5.3 below.

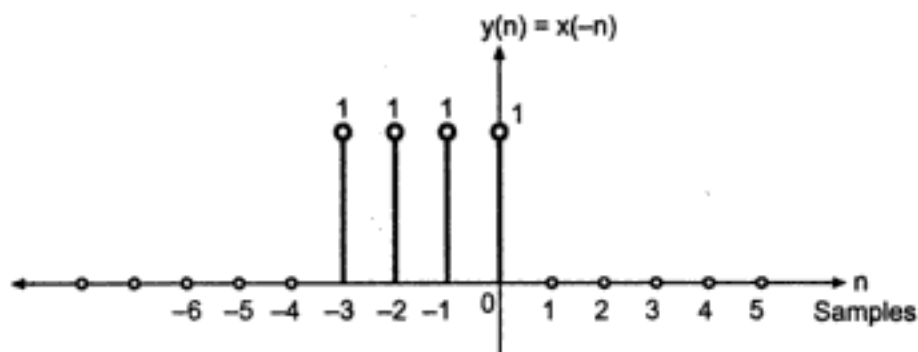


Fig. 2.5.3 Folding or reflection operation of the sequence

Now compare the sequence $x(-n)$ shown in Fig. 2.5.3 with the sequence $x(n)$ in Fig. 2.5.2. It is clear that sequence $x(-n)$ is the mirror image of $x(n)$ at $n=0$. It is also called folding or reflection of $x(n)$ at $n=0$. Thus $x(-n)$ is obtained by folding $x(n)$ with respect to time origin $n=0$. Thus we can also say that $y(n)$ is obtained by just folding $x(n)$ to get,

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

The folding of sequences is required in convolution and FFT algorithms as we will see next.

(ii) $y(n) = x(n-1)$: Delay operation :

This indicates delaying the sequence. The sketch and values of $x(n)$ are given in Fig. 2.5.2. Let us use these values to calculate $y(n)$ as follows.

$$n=0 \Rightarrow y(0) = x(0-1) = x(-1) = 0$$

$$n=1 \Rightarrow y(1) = x(1-1) = x(0) = 1$$

$$n=2 \Rightarrow y(2) = x(2-1) = x(1) = 1$$

$$n=3 \Rightarrow y(3) = x(3-1) = x(2) = 1$$

$$n=4 \Rightarrow y(4) = x(4-1) = x(3) = 1$$

$$n=5 \Rightarrow y(5) = x(5-1) = x(4) = 0$$

$$n=6 \Rightarrow y(6) = x(6-1) = x(5) = 0 \quad \dots \text{ and so on.}$$

Similarly for negative values of n ,

$$n=-1 \Rightarrow y(-1) = x(-1-1) = x(-2) = 0$$

$$n=-2 \Rightarrow y(-2) = x(-2-1) = x(-3) = 0$$

$$n=-3 \Rightarrow y(-3) = x(-3-1) = x(-4) = 0 \quad \dots \text{ and so on.}$$

Based on the above values, Fig. 2.5.4 (b) shows the sequence $y(n)$. Fig. 2.5.4 (a) shows $x(n)$ for reference.

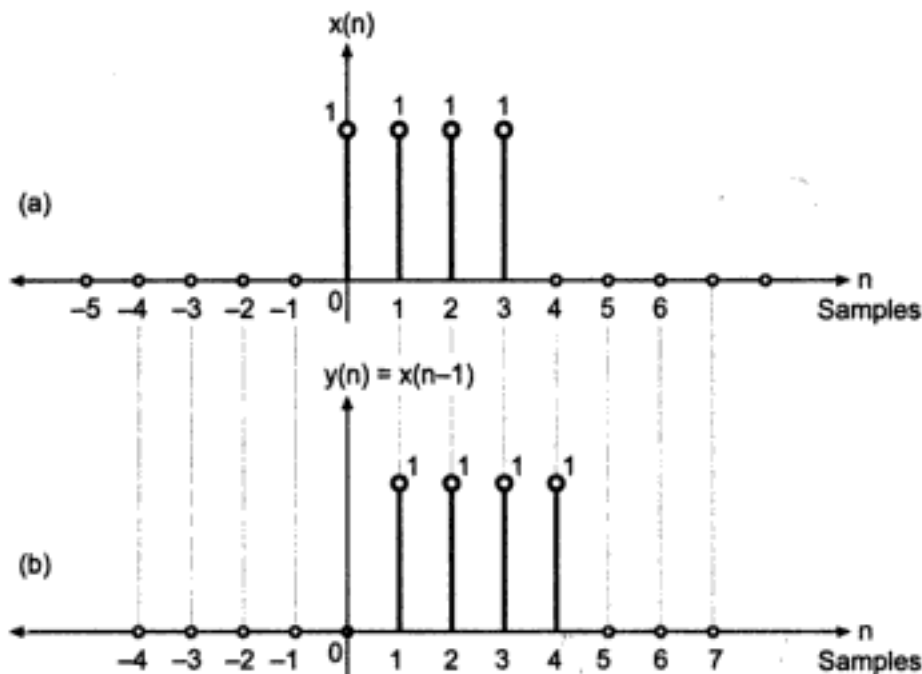


Fig. 2.5.4 (a) The sketch of sequence $x(n)$

(b) Sketch of sequence $y(n)$. It is delayed by one sample.

On comparing $x(n)$ and $y(n)$ shown in above figure, it is clear that $y(n)$ is delayed by one sample. Thus delay of one sample indicates shifting the sequence right by one sample. Similarly if,

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

then $y(n)$ can be obtained by shifting $x(n)$ to right by 'k' samples.

(iii) $y(n) = x(n+1)$: Advancing operation :

This indicates advancing the sequence. We have the sketch and values of $x(n)$ as shown in Fig. 2.5.4 (a). Let us calculate $y(n) = x(n+1)$ for various values of n as follows :

$$n = 0 \Rightarrow y(0) = x(0+1) = x(1) = 1$$

$$n = 1 \Rightarrow y(1) = x(1+1) = x(2) = 1$$

$$n = 2 \Rightarrow y(2) = x(2+1) = x(3) = 1$$

$$n = 3 \Rightarrow y(3) = x(3+1) = x(4) = 0$$

$$n = 4 \Rightarrow y(4) = x(4+1) = x(5) = 0 \quad \dots \text{ and so on.}$$

Similarly for negative values of 'n' we have,

$$n = -1 \Rightarrow y(-1) = x(-1+1) = x(0) = 1$$

$$n = -2 \Rightarrow y(-2) = x(-2+1) = x(-1) = 0$$

$$n = -3 \Rightarrow y(-3) = x(-3+1) = x(-2) = 0$$

$$n = -4 \Rightarrow y(-4) = x(-4+1) = x(-3) = 0 \quad \dots \text{ and so on.}$$

The sketch of $x(n)$ and $y(n)$ together is shown in Fig. 2.5.5. (a) and (b) based on above values.

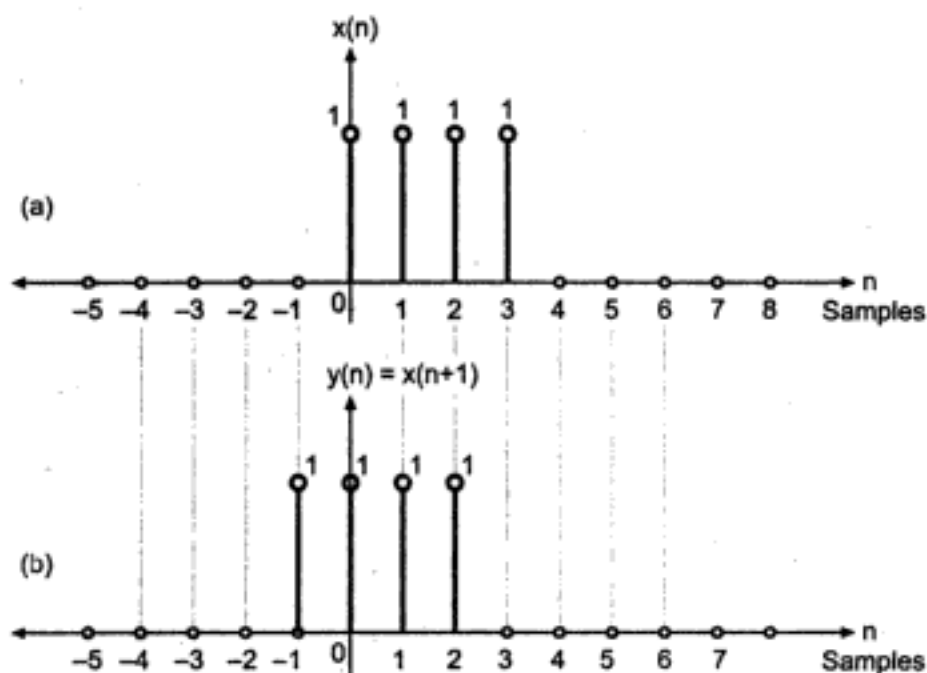


Fig. 2.5.5 (a) The sketch of sequence $x(n)$

(b) Sketch of sequence $y(n)$. It is advanced by one sample.

On comparing the above two sequences it is clear that sequence $y(n)$ is time advanced by one sample with respect to sequence $x(n)$. Hence advancing sequence by one sample indicates the shift of sequence towards left by one sample. Similarly if,

$$y(n) = x(n+k)$$

then $y(n)$ will be obtained by shifting $x(n)$ towards left by 'k' samples.

(iv) $y(n) = x(n-1) + x(n+1)$: Adding operation :

This is the operation of adding two sequences. Let us evaluate $y(n)$ by using the available sequence $x(n)$ of Fig. 2.5.5 (a).

$$n = 0 \Rightarrow y(0) = x(0-1) + x(0+1) = x(-1) + x(1) = 1$$

$$n = 1 \Rightarrow y(1) = x(1-1) + x(1+1) = x(0) + x(2) = 2$$

$$n = 2 \Rightarrow y(2) = x(2-1) + x(2+1) = x(1) + x(3) = 2$$

$$n = 3 \Rightarrow y(3) = x(3-1) + x(3+1) = x(2) + x(4) = 1$$

$$n = 4 \Rightarrow y(4) = x(4-1) + x(4+1) = x(3) + x(5) = 1$$

$$n = 5 \Rightarrow y(5) = x(5-1) + x(5+1) = x(4) + x(6) = 0$$

$$n = 6 \Rightarrow y(6) = x(6-1) + x(6+1) = x(5) + x(7) = 0$$

... and so on.

Similarly for negative values of 'n' we get,

$$n = -1 \Rightarrow y(-1) = x(-1-1) + x(-1+1) = x(-2) + x(0) = 1$$

$$n = -2 \Rightarrow y(-2) = x(-2-1) + x(-2+1) = x(-3) + x(-1) = 0$$

$$n = -3 \Rightarrow y(-3) = x(-3-1) + x(-3+1) = x(-4) + x(-2) = 0$$

... and so on.

Based on the above calculations, the sketch of $x(n)$, $x(n-1)$, $x(n+1)$, $y(n)$ is shown in Fig. 2.5.6 below. For this sketch, $x(n)$ and $x(n-1)$ of Fig. 2.5.4 is used. And $x(n+1)$ of Fig. 2.5.5 is used.

Please refer Fig. 2.5.6 on next page.

In the Fig. 2.5.6 observe that $y(n)$ is obtained by adding $x(n-1)$ and $x(n+1)$ on sample to sample basis. That is, for example 1st sample of $y(n)$ is obtained by adding 1st samples of $x(n-1)$ and $x(n+1)$. Such summation operation is often used in convolution, FFT and filtering operations.

(v) $y(n) = 2x(n)$: **Scaling operation :**

This is the simple operation in which the samples of the sequence are scaled by some magnitude. Here, the samples of $x(n)$ are multiplied by 2. Thus,

$$n = 0 \Rightarrow y(0) = 2 \cdot x(0) = 2 \times 1 = 2$$

$$n = 1 \Rightarrow y(1) = 2 \cdot x(1) = 2 \times 1 = 2$$

$$n = 2 \Rightarrow y(2) = 2 \cdot x(2) = 2 \times 1 = 2$$

$$n = 3 \Rightarrow y(3) = 2 \cdot x(3) = 2 \times 1 = 2$$

$$n = 4 \Rightarrow y(4) = 2 \cdot x(4) = 2 \times 0 = 0$$

$$n = 5 \Rightarrow y(5) = 2 \cdot x(5) = 2 \times 0 = 0$$

... and so on.

Similarly,

$$n = -1 \Rightarrow y(-1) = 2 \cdot x(-1) = 2 \times 0 = 0$$

$$n = -2 \Rightarrow y(-2) = 2 \cdot x(-2) = 2 \times 0 = 0 \quad \dots \text{ and so on.}$$

The sketch of $y(n)$ based on above values is shown in Fig. 2.5.7.

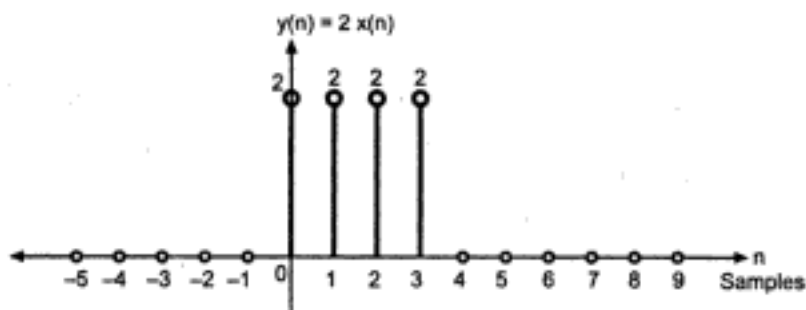


Fig. 2.5.7 Scaling operation in discrete time systems

The scaling operation is used to amplify or attenuate the sample values.

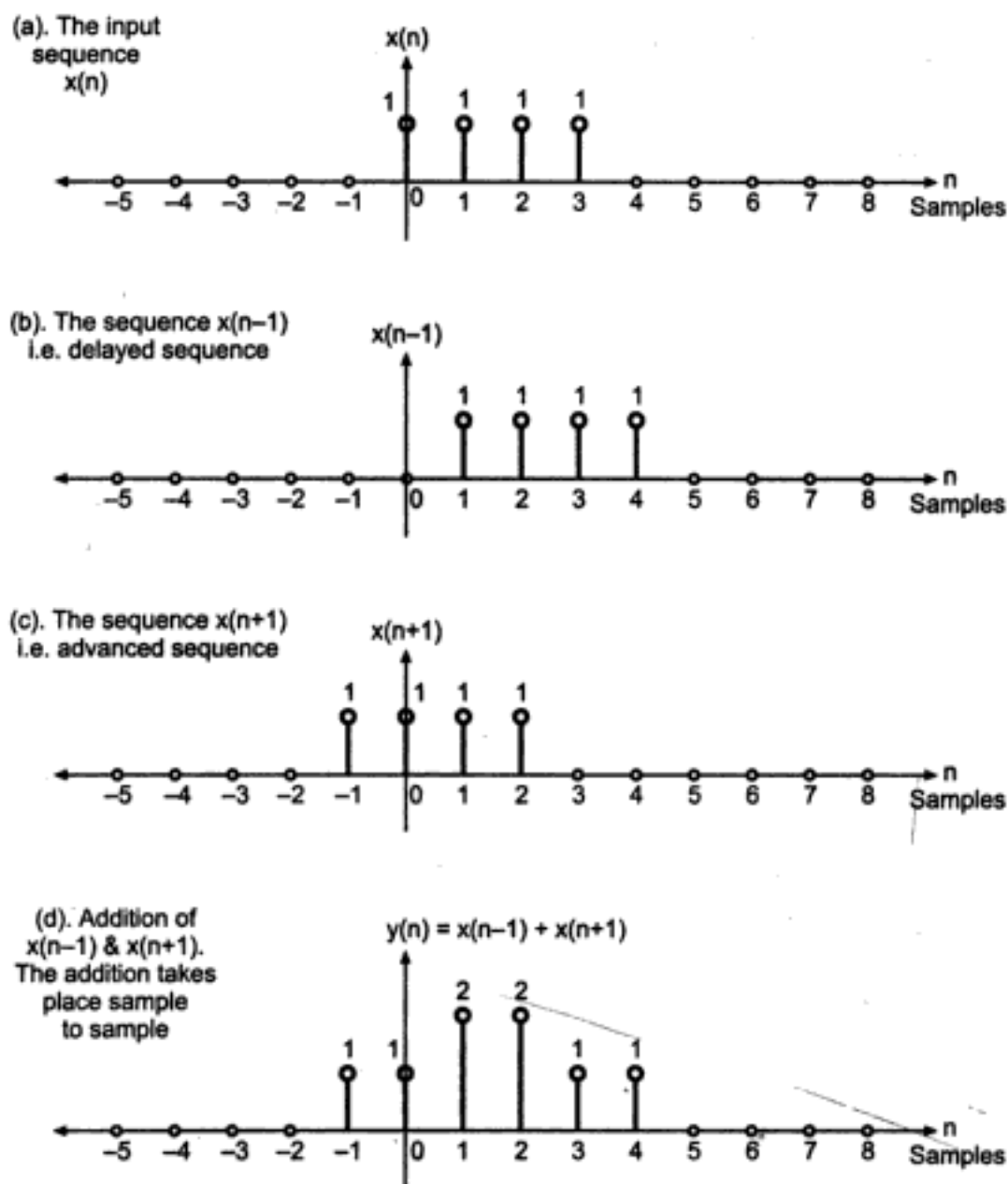


Fig. 2.5.6 Addition operation in discrete time systems

2.5.1 Representation of Discrete Time Systems

In this section we will see, how discrete time systems are represented in the block diagram form. The operation of the discrete time system can be completely described by these block diagrams. The block diagrams of discrete time systems are prepared with the help of some fundamental building blocks. Such building blocks are adders, multipliers, delay and advance elements etc. Let us now define those blocks.

1. Adder :

An adder is used to perform the addition of two sequences. It generates the output sequence which is the sum of input sequences. Fig. 2.5.8 shows the symbol of an adder which adds two sequences $x_1(n)$ and $x_2(n)$.

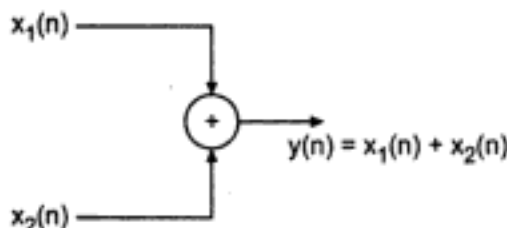


Fig. 2.5.8 Symbol for adder used in the discrete time systems

The adding operation is carried out on sample to sample basis. Hence there is no need to store any samples. Hence this addition is also called memoryless. Some time the symbol of an adder is also used as shown in Fig. 2.5.9. In this figure symbol Σ (sigma) represents summation the signs '+' or '-' are used to indicate Addition or Subtraction.

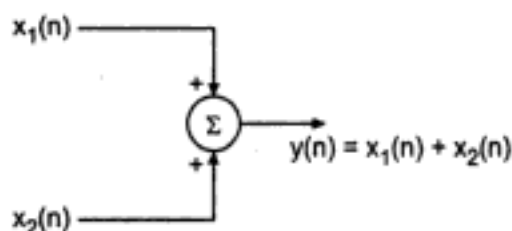


Fig. 2.5.9 Alternate symbol used to represent adder

2. Constant multiplier :

This symbol is used to show the scaling operation. This symbol is shown in Fig. 2.5.10. This operation also does not need any storage or memory. As shown in above figure $x(n)$ is multiplied by a constant 'a' to give output sequence $y(n)$.



Fig. 2.5.10 A symbol used to represent scaling operation

3. Signal multiplier :

This symbol is used to indicate the multiplication of two sequences. It is shown in Fig. 2.5.11. As shown in the figure, the two sequences $x_1(n)$ and $x_2(n)$ are multiplied on the sample to sample basis to give sequence $y(n)$. This operation does not need any storage or memory.



Fig. 2.5.11 A symbol used to represent signal multiplier

4. Unit delay block :

The unit delay block delays the input signal by one sample. For example when the input is $x(n)$ the output is $x(n-1)$. Fig. 2.5.12 shows the representation of unit delay block. Consider for example when input sample is $x(1)$, then $y(1) = x(1-1) = x(0)$. Thus at the output of unit delay block, there is just previous sample. Such delay is produced for all the samples. The unit delay block stores the sample value for one sample duration. Hence it needs storage or memory for operation. The unit delay is represented by z^{-1} in mathematical operations. This will be explained in details in the chapter on z-transforms.



Fig. 2.5.12 A unit delay block

5. Unit advance block :

This is used to represent advancing of the sequence by one sample. Fig. 2.5.13 shows the symbol of this block. As shown in the figure the output is $y(n) = x(n+1)$ when the input is $x(n)$. For example when input sample is $x(1)$, then output sample is $y(1) = x(2)$. Thus at the output of unit advance block there is next sample available.

This operation is not realizable in practice. This is because, it is not possible to get next sample when present sample is the input. Here 'z' indicates the unit time advance operation.



Fig. 2.5.13 A unit advance block

Ex. 2.5.2 Consider the discrete time signal derived in example 2.2.4. Sketch the following signals :

- (i) First fold $x(n)$ and then delay resulting signal by four samples.
- (ii) First delay $x(n)$ by four samples and then fold the resulting signal.

And then prove that folding and time delay operations are not commutative.

Sol. : In example 2.2.4 we have derived $x(n)$ as,

$$x(n) = \left\{ \frac{1}{3}, \frac{2}{3}, 1, 1, 1, 1 \right\}$$

Fig. 2.5.14 (a) shows the sketch of this sequence $x(n)$. The folded sequence $x(-n)$ is shown in Fig. 2.5.14 (b). Then this sequence is delayed by four samples. It becomes $x(-n+4)$. Its sketch is shown in Fig. 2.5.14 (c).

Please refer Fig. 2.5.14 on next page.

Fig. 2.5.15 (a) shows the sketch of $x(n)$. This sequence is delayed by four samples. It becomes $x(n-4)$. It is shown in Fig. 2.5.15 (b). This delayed sequence is then folded and it becomes $x(-n-4)$. This sequence is shown in Fig. 2.5.15 (c).

Please refer Fig. 2.5.15 on page 42.

Observe the two sequences obtained in Fig. 2.5.14 (c) and Fig. 2.5.15 (c) after folding and time delay operations. These two sequences are totally different. This shows that time delay and folding operations are not commutative.

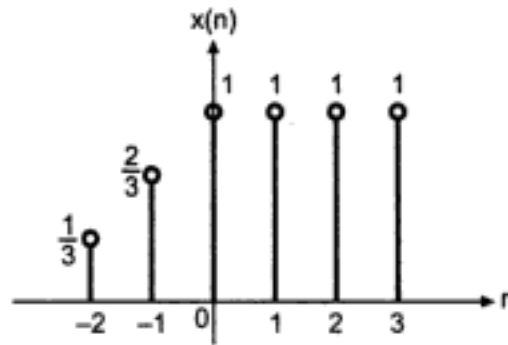
2.6 Classification of Discrete Time Systems

In the last section we defined discrete time systems. In this section we will discuss the important properties of the discrete time systems according to which they are classified. The discrete time systems are classified as,

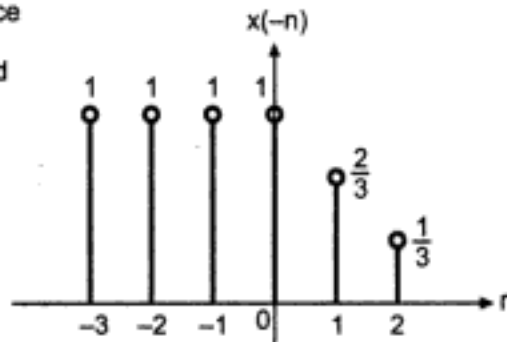
- (i) Static and dynamic systems
- (ii) Time invariant and time variant systems
- (iii) Linear and non linear systems
- (iv) Causal and non causal systems
- (v) Stable and unstable systems

Actually the above mentioned types of classifications describe the dynamicity, shift invariance, linearity, causality and stability properties of discrete time systems respectively.

(a). Sketch of sequence $x(n]$



(b). The sequence $x[n]$ is folded. It is represented by $x[-n]$



(c). The folded sequence $x[-n]$ is delayed by four samples.

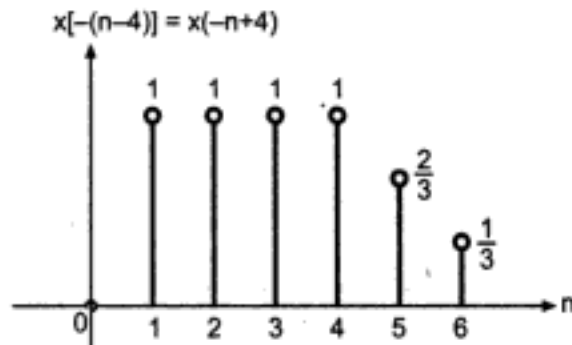


Fig. 2.5.14 Folding and time delay operation

2.6.1 Static and Dynamic Systems (Dynamicity Property)

When the output of the system depends only upon the present input sample, then it is called *static* system. For example,

$$y(n) = 10 \cdot x(n)$$

or

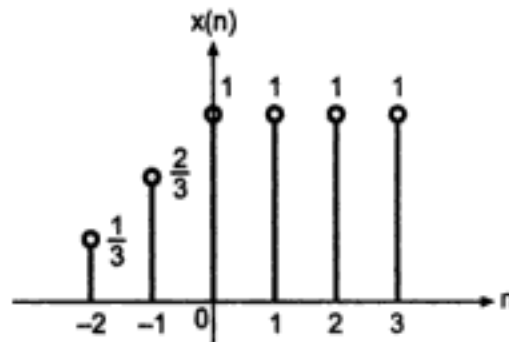
$$y(n) = 15 \cdot x^2(n) + 10x(n)$$

are the static systems. Here the $y(n)$ depends only upon n^{th} input sample. Hence such systems do not need memory for its operation. A system is said to be *dynamic* if the output depends upon the past values of input also. For example,

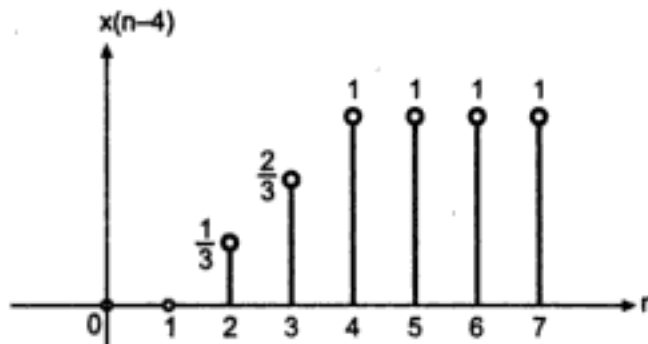
$$y(n) = x(n) + x(n-1)$$

This is the dynamic system. In this system the n^{th} output sample value depends upon n^{th} input sample and just previous i.e. $(n-1)^{th}$ input sample. This systems needs to store the previous sample value. Consider the following equation of a system.

(a). Sketch of sequence $x(n]$



(b). The sequence $x[n]$ is delayed by four samples. i.e. $x[n-4]$



(c). The delayed sequence $x[n-4]$ is folded.

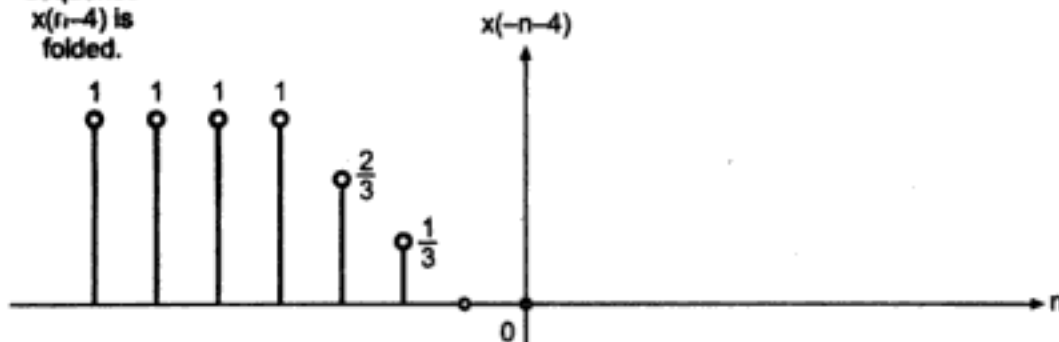


Fig. 2.5.15 Time delay and folding operation

$$y(n) = \sum_{k=0}^4 x(n-k)$$

Expanding this equation,

$$y(n) = x(n) + x(n-1) + x(n-2) + x(n-3) + x(n-4)$$

This also represents a dynamic system. The output depends upon the present input and preceding four input samples. Hence the preceding four input samples i.e. $x(n-1)$, $x(n-2)$, $x(n-3)$ and $x(n-4)$ are stored in the memory.

2.6.2 Shift Invariant and Shift Variant Systems (Shift Invariance Property)

If the input output characteristics of the system do not change with shift of time origin, such systems are called *shift invariant* or *time invariant* systems. Consider the system has response $y(n)$ for the excitation of $x(n)$. i.e.,

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

Then the system is shift invariant or time invariant if and only if,

$$x(n-k) \xrightarrow{T} y(n-k) \quad k\text{-constant} \quad \dots (2.6.1)$$

This means if the input is shifted by 'k' samples, then output is also shifted by the same number of samples for shift invariant system.

To test for shift invariance property :

To test for shift invariance, excite the system by the input $x(n)$ and get output $y(n)$. Then delay (or shift) the input by 'k' samples and calculate the output. Let this output be,

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

Thus $y(n, k) \Rightarrow$ response due to delayed / shifted input.

Now we have $y(n)$ computed earlier. Hence obtain $y(n-k)$ from $y(n)$ by delaying by 'k' samples. i.e., $y(n-k) \Rightarrow$ output delayed / shifted directly. Then the system is shift invariant or time invariant if,

$$y(n, k) = y(n-k) \quad \text{for all values of 'k'}$$

And the system is time variant or shift variant if,

$$y(n, k) \neq y(n-k) \quad \text{even for single value of 'k'}$$

Cooking a Rice is the shift invariant operation since every day it requires the same amount of time. It is independent of time / day / year of cooking. Ambient temperature is the shift variant parameter since it depends upon the period of time. For example temperature is high in May-June whereas it is minimum in November-December. Like these, there are many physical examples of shift invariant and shift variant systems. Few of them with their classification are mentioned below :

Table 2.6.1 Physical examples of shift invariant and shift variant systems

Sr.No.	Operation or system	Type
1	Bus and train arrival and departures	Shift invariant
2	Rainfall per month	Shift variant
3	Thermal noise in electronic components	Shift invariant
4	Noise effects in the radio communication channels	Shift variant
5	File handling in C language	Shift invariant
6	Printing documents by the printer	Shift invariant

Ex. 2.6.1 Determine whether the following systems are shift invariant or not ?

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

(ii) $y(n) = n x(n)$

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

(iv) $y(n) = x(n) \cos \omega_0 n$

Sol. : (i) Consider the system described by

$$y(n) = T[x(n)] = x(n) - x(n-1) \quad \dots (2.6.1 \text{ (a)})$$

Let us apply the input to this system which is delayed by 'k' samples. Then the output will be,

$$\begin{aligned} y(n, k) &= T[x(n-k)] \\ &= x(n-k) - x(n-k-1) \end{aligned} \quad \dots (2.6.2)$$

Now let us delay the output $y(n)$ given by equation 2.6.1 by ' k ' samples i.e.,

$$y(n-k) = x(n-k) - x(n-k-1) \quad \dots (2.6.3)$$

Here observe that

$$y(n, k) = y(n-k)$$

Hence the system is shift invariant.

(ii) The given discrete time system equation is

$$y(n) = T[x(n)] = nx(n) \quad \dots (2.6.4)$$

When input $x(n)$ is delayed by ' k ' samples, the response is,

$$\begin{aligned} y(n, k) &= T[x(n-k)] \\ &= nx(n-k) \end{aligned} \quad \dots (2.6.5)$$

Here observe that only input $x(n)$ is delayed. The multiplier ' n ' is not part of the input. Hence it cannot be written as $(n-k)$. Now let us shift or delay the output $y(n)$ given by equation 2.6.4 by ' k ' samples. i.e.,

$$y(n-k) = (n-k)x(n-k) \quad \dots (2.6.6)$$

Here both n and $x(n)$ in the equation $y(n) = nx(n)$ will be shifted by ' k ' samples since they are part of the output sequence. It is clear from equation 2.6.5 and equation 2.6.6 that,

$$y(n, k) \neq y(n-k)$$

Hence the system is shift variant.

(iii) The given discrete time system equation is,

$$\begin{aligned} y(n) &= T[x(n)] \\ &= x(-n) \end{aligned} \quad \dots (2.6.7)$$

Here let us delay the input $x(n)$ by ' k ' samples. The output $y(n)$ is equal to folded input, i.e. $x(-n)$. Hence $x(-n)$ will also be delayed by ' k ' samples. i.e.,

$$\begin{aligned} y(n, k) &= T[x(n-k)] \\ &= x[(-n)-k] \end{aligned} \quad \dots (2.6.8)$$

$$= x(-n-k) \quad \dots (2.6.8 (a))$$

Here observe that we cannot replace ' n ' by simply $n-k$. Since we are delaying $x(n)$; $x(-n)$ will also be delayed by same amount. The equation 2.6.8 is specifically written to indicate this operation.

Now let us delay the output $y(n)$ given by equation 2.6.7 by ' k ' samples. i.e.,

$$y(n-k) = x[-(n-k)] \quad \dots (2.6.9)$$

$$= x(-n+k) \quad \dots (2.6.9 (a))$$

Here observe that we are delaying the output $y(n)$. That is n is converted to $n-k$ in the equation for $y(n)$. This is specifically indicated in equation 2.6.9. On comparing equation 2.6.9 and equation 2.6.8 we observe that,

$$y(n, k) \neq y(n-k),$$

Hence the system is shift variant.

(iv) The given response equation is,

$$y(n) = T[x(n)] = x(n) \cos \omega_0 n \quad \dots (2.6.10)$$

The response to the delayed input $x(n - k)$ will be,

$$y(n, k) = x(n - k) \cos \omega_0 n \quad \dots (2.6.10 (a))$$

Here observe that 'n' in $\cos \omega_0 n$ is not written as $(n - k)$ since the term ' $\cos \omega_0 n$ ' is not part of the input. Now let us delay the output $y(n)$ by 'k' samples given by equation 2.6.10,

$$y(n - k) = x(n - k) \cos \omega_0 (n - k) \quad \dots (2.6.10 (b))$$

Here 'n' is converted to $(n - k)$ since both $x(n)$ and $\cos \omega_0 n$ in equation 2.6.10 are part of output $y(n)$. On comparing equation 2.6.10 (a) and (b) we find that,

$$y(n, k) \neq y(n - k)$$

Hence the system is shift variant.

From the above example it is clear that if the system alters the timing properties of the signal, then it is shift variant system.

2.6.3 Linear and Nonlinear Systems (Linearity Property)

A system is said to be linear if it satisfies the superposition principle. Let $x_1(n)$ and $x_2(n)$ be the two input sequences. Then the system is said to be linear if and only if

$$T \{a_1 x_1(n) + a_2 x_2(n)\} = a_1 T[x_1(n)] + a_2 T[x_2(n)] \quad \dots (2.6.11)$$

Here a_1 and a_2 are arbitrary constants. The above condition states that the system is said to be linear if the combined response due to $x_1(n)$ and $x_2(n)$ together is same as the sum of individual responses. Fig. 2.6.1 (a) shows the implementation of LHS of equation 2.6.11.

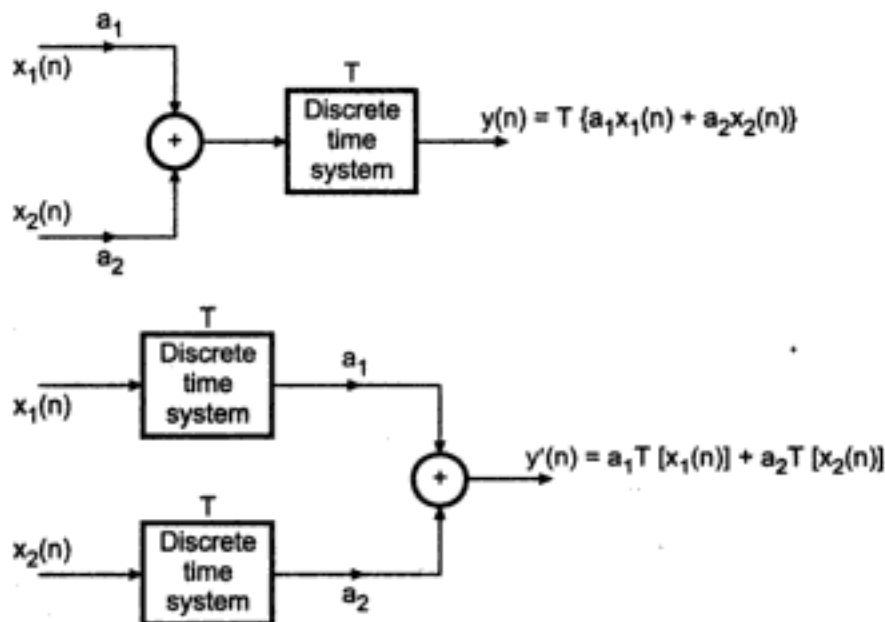


Fig. 2.6.1 (a) Representation of LHS of equation 2.6.11

Fig. 2.6.1 (b) Representation of RHS of equation 2.6.11

For the discrete time 'T' to have linear $y(n)$ in Fig. (a) must be same as $y'(n)$ in Fig. (b)

Equation 2.6.11 can be extended further, that is if there are 'M' number of input signals i.e.,

$$x(n) = \sum_{k=1}^{M-1} a_k x_k(n)$$

Then the output of such system is given as,

$$y(n) = \sum_{k=1}^{M-1} a_k y_k(n)$$

That is if the system is linear, we have,

$$\mathcal{T} \left\{ \sum_{k=1}^{M-1} a_k x_k(n) \right\} = \sum_{k=1}^{M-1} a_k y_k(n) \quad \dots (2.6.12)$$

A linear system has the important property that, it produces zero output if the input is zero under the relaxed condition. If the system does not satisfy this condition it is called nonlinear system. The nonlinear system does not satisfy superposition principle also.

Ex. 2.6.2 Determine whether the following systems

(i) $y(n) = x(n^2)$

(ii) $y(n) = x^2(n)$

are linear or nonlinear.

Sol. : (i) The given equation is,

$$y(n) = x(n^2)$$

For the two separate inputs $x_1(n)$ and $x_2(n)$ the system produces the response of,

$$y_1(n) = x_1(n^2) \quad \dots (2.6.13 \text{ (a)})$$

and

$$y_2(n) = x_2(n^2) \quad \dots (2.6.13 \text{ (b)})$$

The response of the system to the linear combination of $x_1(n)$ and $x_2(n)$ will be,

$$y_3(n) = T [a_1 x_1(n) + a_2 x_2(n)]$$

Since the linear systems satisfy *additive* property, the above equation can be written as,

$$y_3(n) = T \{a_1 x_1(n)\} + T \{a_2 x_2(n)\}$$

The linear systems also satisfy *scaling* property. Hence we can write above equation as,

$$\begin{aligned} y_3(n) &= a_1 T \{x_1(n)\} + a_2 T \{x_2(n)\} \\ &= a_1 x_1(n^2) + a_2 x_2(n^2) \text{ from equation 2.6.13} \quad \dots (2.6.14) \end{aligned}$$

This is the response of the system to linear combination of two inputs. This type of operation is illustrated in Fig. 2.6.1 (a). Now the response of the system due to linear combination of two outputs will be,

$$y_3'(n) = a_1 y_1(n) + a_2 y_2(n)$$

This type of operation is illustrated in Fig. 2.6.1 (b). From equation 2.6.13, we can write above equation as,

$$y_3'(n) = a_1 x_1(n^2) + a_2 x_2(n^2) \quad \dots (2.6.15)$$

On comparing equation 2.6.14 and above equation we observe that,

$$y_3(n) = y_3'(n)$$

Hence the system is linear.

(ii) The given equation is,

$$y(n) = x^2(n)$$

When the inputs $x_1(n)$ and $x_2(n)$ are applied separately, the responses $y_1(n)$ and $y_2(n)$ will be,

$$y_1(n) = x_1^2(n) \quad \dots (2.6.15 (a))$$

and $y_2(n) = x_2^2(n) \quad \dots (2.6.15 (b))$

The response of the system to the linear combination of $x_1(n)$ and $x_2(n)$ will be,

$$\begin{aligned} y_3(n) &= T\{a_1 x_1(n) + a_2 x_2(n)\} \\ &= [a_1 x_1(n) + a_2 x_2(n)]^2 \quad \text{Since } y(n) = x^2(n) \\ &= a_1^2 x_1^2(n) + 2a_1 a_2 x_1(n)x_2(n) + a_2^2 x_2^2(n) \quad \dots (2.6.16) \end{aligned}$$

Observe that this operation is illustrated in Fig. 2.6.1 (a).

The linear combination of two outputs given by equation 2.6.15 will be,

$$y_3'(n) = a_1 x_1^2(n) + a_2 x_2^2(n)$$

On comparing the output of above equation with that of equation 2.6.16 we find that,

$$y_3(n) \neq y_3'(n)$$

Hence the system is nonlinear.

We will see some more examples on linearity property of systems further.

2.6.4 Causal and Noncausal Systems (Causality Property)

In the causal system the output depends upon past and present inputs only. That is the output is the function of $x(n)$, $x(n-1)$, $x(n-2)$, $x(n-3)$... and so on. The system is noncausal if its output depends upon the future inputs also. i.e. $x(n+1)$, $x(n+2)$ and so on. Thus the noncausal systems are physically unrealizable. The following example illustrates causal and noncausal systems.

Ex. 2.6.3 Check whether the systems described by following equations are causal or noncausal.

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

(ii) $y(n) = x(n) + x(n+1)$

(iii) $y(n) = x(2n)$

Sol. : (i) The given system equation is,

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

Here $y(n)$ depends upon $x(n)$ and $x(n-1)$. $x(n)$ is the present input and $x(n-1)$ is the previous input. Hence the system is causal.

(ii) The given system equation is,

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

Here $y(n)$ depends upon $x(n)$ and $x(n+1)$. $x(n)$ is the present input and $x(n+1)$ is the next input. Hence the system is noncausal.

(iii) The given system equation is,

$$y(n) = x(2n)$$

Here when

$$n=1 \Rightarrow y(1) = x(2)$$

$$n=2 \Rightarrow y(2) = x(4) \dots$$

Thus the output $y(n)$ depends upon the future inputs. Hence the system is noncausal.

2.6.5 Stable and Unstable Systems (Stability Property)

When the every bounded input produces a bounded output, then the system is called Bounded input Bounded output (BIBO) stable. The input $x(n)$ is said to be bounded if there exists some finite number M_x such that,

$$|x(n)| \leq M_x < \infty \quad \dots (2.6.17 (a))$$

Similarly output $y(n)$ is bounded if there exists some finite number M_y such that,

$$|y(n)| \leq M_y < \infty \quad \dots (2.6.17 (b))$$

If the output is unbounded for any bounded input, then the system is unstable. The unstable systems produce erratic outputs.

We will see some examples on stability of systems further.

Ex. 2.6.4 Determine whether the following system is linear or nonlinear.

$$y(n) = \log_{10} (|x(n)|)$$

Sol. : The given equation is,

$$y(n) = T \{x(n)\} = \log_{10} (|x(n)|) \quad \dots (2.6.18)$$

Let the two inputs $x_1(n)$ and $x_2(n)$ be applied separately. Then the response will be,

$$y_1(n) = \log_{10} (|x_1(n)|) \quad \dots (2.6.19 (a))$$

and

$$y_2(n) = \log_{10} (|x_2(n)|) \quad \dots (2.6.19 (b))$$

The response of the system to linear combination of $x_1(n)$ and $x_2(n)$ will be,

$$y_3(n) = T \{a_1 x_1(n) + a_2 x_2(n)\}$$

We know that $T \{x(n)\} = \log_{10} (|x(n)|)$. Then above equation becomes,

$$y_3(n) = \log_{10} (|a_1 x_1(n) + a_2 x_2(n)|) \quad \dots (2.6.20)$$

This is the response of the system to linear combination of two inputs. Now consider the linear combination of two outputs given by equation 2.6.19. i.e.,

$$\begin{aligned} y_3(n) &= a_1 y_1(n) + a_2 y_2(n) \\ &= a_1 \log_{10} (|x_1(n)|) + a_2 \log_{10} (|x_2(n)|) \end{aligned}$$

Clearly the two outputs,

$$y_3(n) \neq y_3(n)$$

Hence the system is nonlinear.

This can be proved very easily as follows :

Let $x_1(n) = 100$ and $x_2(n) = 10$

$\therefore y_1(n) = \log_{10} (|x_1(n)|) = 2$

and $y_2(n) = \log_{10} (|x_2(n)|) = 1$

$\therefore y_3(n) = \log_{10} (|x_1(n) + x_2(n)|) = 2.04139$

and $y_3(n) = y_1(n) + y_2(n) = 2 + 1 = 3$

Thus $y_3(n) \neq y_3(n)$ and system is nonlinear. For simplicity we have not considered scaling constants a_1 and a_2 . Eventhough this later procedure looks simple, it is advised that reader should follow the first (basic) procedure only.

Ex. 2.6.5 Few discrete time systems are given below :

$$(i) \quad y(n) = \cos [x(n)]$$

$$(ii) \quad y(n) = \sum_{k=-\infty}^{n+1} x(k)$$

$$(iii) \quad y(n) = x(n) \cos(\omega_0 n)$$

$$(iv) \quad y(n) = x(-n+2)$$

$$(v) \quad y(n) = |x(n)|$$

$$(vi) \quad y(n) = x(n)u(n)$$

$$(vii) \quad y(n) = x(n) + nx(n+1)$$

$$(viii) \quad y(n) = x(2n)$$

$$(ix) \quad y(n) = x(-n)$$

$$(x) \quad y(n) = \text{sgn}[x(n)]$$

Check whether these systems are :

1. Static or dynamic
2. Linear or nonlinear
3. Shift invariant or shift varying
4. Causal or noncausal
5. Stable or unstable.

Sol. : (i) $y(n) = \cos [x(n)]$:

1. A system is static if its output depends only upon the present input sample. Here since $y(n)$ depends upon cosine of $x(n)$, i.e. present input sample, the system is static.
2. For two separate inputs the system produces the response of,

$$y_1(n) = T\{x_1(n)\} = \cos [x_1(n)]$$

$$\text{and} \quad y_2(n) = T\{x_2(n)\} = \cos [x_2(n)]$$

The response of the system to linear combination of two inputs will be,

$$y_3(n) = T\{a_1 x_1(n) + a_2 x_2(n)\} = \cos [a_1 x_1(n) + a_2 x_2(n)]$$

The linear combination of two outputs will be,

$$y_3'(n) = a_1 y_1(n) + a_2 y_2(n) = a_1 \cos [x_1(n)] + a_2 \cos [x_2(n)]$$

Clearly $y_3(n) \neq y_3'(n)$. hence system is nonlinear.

3. The system is said to be shift invariant or time invariant if its characteristics do not change with shift of time origin. The given system is,

$$y(n) = T\{x(n)\} = \cos [x(n)] \quad \dots (2.6.21)$$

Let us delay the input by k samples. Then output will be,

$$y(n, k) = T\{x(n-k)\} = \cos [x(n-k)] \quad \dots (2.6.22)$$

Now let us delay the output $y(n)$ given by equation 2.6.21 by ' k ' samples, i.e. $y(n-k)$. This is equivalent to replacing n by $n-k$ in equation 2.6.21. i.e.,

$$y(n-k) = \cos [x(n-k)]$$

Comparing above equation with equation 2.6.22 we observe that,

$$y(n, k) = y(n-k)$$

This shows that the system is shift invariant.

4. The system is said to be causal if output depends upon past and present inputs only. The output is given as,

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

Here observe that n^{th} sample of output depends upon n^{th} sample of input $x(n)$. Hence the system is a causal system.

5. For any bounded value of $x(n)$ the cosine function has bounded value. Hence $y(n)$ has bounded value. Therefore the system is said to be BIBO stable.

Thus the given system is,

Static, nonlinear, shift invariant, causal and stable.

(ii) $y(n) = \sum_{k=-\infty}^{n+1} x(k) :$

- Here observe that the system's output for n^{th} sample is equal to summation of all past input samples, present input sample and one next input sample. Hence the system needs to store these samples. Therefore the system requires memory. Hence this system is dynamic system.
- The output $y(n)$ can be written as,

$$y(n) = x(-\infty) + \dots + x(-2) + x(-1) + x(0) + x(1) + x(2) + x(3) + \dots + x(n) + x(n+1)$$

From this equation it is clear that the system is linear since it is the summation of individual inputs. We know that the summation operation is a linear operation. Hence the system is a linear system.

- The given system produces output $y(n)$ as a linear summation of inputs. If we shift the inputs, then there will be corresponding shift in the output. Hence the system is shift invariant.
- The output $y(n)$ can be expressed as,

$$y(n) = x(-\infty) + \dots + x(-2) + x(-1) + x(0) + x(1) + x(2) + x(3) + \dots + x(n) + x(n+1)$$

In the above equation the present output $y(n)$ depends upon past inputs like $x(-\infty), x(-2), x(-1), x(0), \dots, x(n-1)$. It also depends upon the present input $x(n)$ and future or next input $x(n+1)$. Hence the system is noncausal since its output depends upon the future inputs also.

5. Consider that the system has input $x(k)$ as unit sample sequence $u(k)$. We know that,

$$u(k) = \begin{cases} 1 & \text{for } k \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Hence the system equation becomes,

$$y(n) = \sum_{k=0}^{n+1} u(k)$$

$= 1+1+1+1+\dots(n+1)$ number of times '1' are added

For example if $n=5$, then above equation becomes,

$$y(5) = 1+1+1+1+1+1+1=7$$

if $n=10$, $y(10) = 1+1+1+1+1+1+1+1+1+1+1+1=12$

It is clear from above equation that as $n \rightarrow \infty, y(n) \rightarrow \infty$. Here the input $u(k)$ is the unit sample sequence and it is bounded. But output $y(n)$ is unbounded as $n \rightarrow \infty$. Hence the system is unstable. Thus the given system is,

Dynamic, linear, shift invariant, noncausal and unstable.

(iii) $y(n) = x(n) \cos(\omega_0 n)$:

1. This is a static system since, the output of the system depends only upon the present input sample. i.e., n^{th} output sample depends upon n^{th} input sample. Hence this is a static system.

2. We know that the given system is,

$$y(n) = T\{x(n)\} = x(n) \cos(\omega_0 n)$$

When the two inputs $x_1(n)$ and $x_2(n)$ are applied separately, the responses $y_1(n)$ and $y_2(n)$ will be,

$$\left. \begin{aligned} y_1(n) &= T\{x_1(n)\} = x_1(n) \cos(\omega_0 n) \\ y_2(n) &= T\{x_2(n)\} = x_2(n) \cos(\omega_0 n) \end{aligned} \right\} \dots (2.6.23)$$

The response of the system due to linear combination of inputs will be,

$$\begin{aligned} y_3(n) &= T\{a_1 x_1(n) + a_2 x_2(n)\} = [a_1 x_1(n) + a_2 x_2(n)] \cos(\omega_0 n) \\ &= a_1 x_1(n) \cos(\omega_0 n) + a_2 x_2(n) \cos(\omega_0 n) \end{aligned} \dots (2.6.24)$$

Now the linear combination of the two outputs will be,

$$\begin{aligned} y_3(n) &= a_1 y_1(n) + a_2 y_2(n) \\ &= a_1 x_1(n) \cos(\omega_0 n) + a_2 x_2(n) \cos(\omega_0 n) \text{ from equation 2.6.23} \end{aligned}$$

From above equation and equation 2.6.24,

$$y_3(n) = y_3(n). \text{ Hence the system is linear.}$$

3. The system equation is,

$$y(n) = x(n) \cos(\omega_0 n)$$

The response of the system to delayed input will be,

$$\begin{aligned} y(n, k) &= T\{x(n-k)\} \\ &= x(n-k) \cos(\omega_0 n) \end{aligned} \dots (2.6.25)$$

Now let us delay or shift the output $y(n)$ by 'k' samples. i.e.,

$$y(n-k) = x(n-k) \cos[\omega_0(n-k)]$$

Here every 'n' is replaced by $n-k$. On comparing above equation with equation 2.6.25 we find that,

$$y(n, k) \neq y(n-k) \text{ Hence the system is shift variant.}$$

4. In the given system, $y(n)$ depends upon $x(n)$, i.e. present output depends upon present input. Hence the system is causal.

5. The given system equation is,

$$y(n) = x(n) \cos(\omega_0 n)$$

Here value of $\cos(\omega_0 n)$ is always bounded. Hence as long as $x(n)$ is bounded, $y(n)$ is also bounded. Hence the system is stable.

This system is,

Static, linear, shift variant, causal and stable.

(iv) $y(n) = x(-n + 2)$:

1. It is clear from above equation that n^{th} sample of output is equal to $(-n + 2)^{\text{th}}$ sample of input. Hence the system needs memory storage. Therefore the system is Dynamic.
2. It is very easy to prove that this system is linear.
3. The output $y(n)$ for delayed input will be,

$$\begin{aligned} y(n, k) &= T\{x(n - k)\} \\ &= x(-n + 2 - k) \end{aligned} \quad \dots (2.6.26)$$

Now the delayed output will be obtained by replacing n by $n - k$ in the system equation i.e.,

$$\begin{aligned} y(n - k) &= x[-(n - k) + 2] \\ &= x(-n + 2 + k) \end{aligned}$$

On comparing above equation with equation 2.6.26 we find that,

$$y(n, k) \neq y(n - k) \text{ Hence the system is shift variant}$$

4. In the given system equation, when we put $n = 0$ we get,

$$y(0) = x(2)$$

Thus the output depends upon future inputs. Hence the system is noncausal.

5. It is clear from the given system equation that, as long as input is bounded, the output is bounded. Hence the system is a stable system.

Thus the given system is,

Dynamic, linear, shift variant, noncausal and stable.

(v) $y(n) = |x(n)|$

1. The output is equal to magnitude of present input sample. Hence the system does not need memory storage. Therefore the system is static.
2. The given system equation is,

$$y(n) = T\{x(n)\} = |x(n)|$$

For two separate inputs $x_1(n)$ and $x_2(n)$ the system has the response of,

$$\left. \begin{aligned} y_1(n) &= T\{x_1(n)\} = |x_1(n)| \\ y_2(n) &= T\{x_2(n)\} = |x_2(n)| \end{aligned} \right\} \quad \dots (2.6.27)$$

The response of the system to linear combination of two inputs $x_1(n)$ and $x_2(n)$ will be,

$$\begin{aligned} y_3(n) &= T\{a_1 x_1(n) + a_2 x_2(n)\} \\ &= |a_1 x_1(n) + a_2 x_2(n)| \end{aligned} \quad \dots (2.6.28)$$

Now the linear combination of two outputs will be,

$$\begin{aligned} y_3(n) &= a_1 y_1(n) + a_2 y_2(n) \\ &= a_1 |x_1(n)| + a_2 |x_2(n)| \end{aligned}$$

Here observe that $y_3(n) \neq y_3(n)$. Hence the system is nonlinear.

3. Delaying the input by ' k ' samples, output will be,

$$\begin{aligned} y(n, k) &= T\{x(n - k)\} \\ &= |x(n - k)| \end{aligned}$$

And the delayed output will be,

$$y(n-k) = |x(n-k)|$$

Since $y(n, k) = y(n-k)$, the system is shift invariant.

- The system equation is $y(n) = |x(n)|$. The output depends upon present input. Hence the system is causal.
- From the given equation it is clear that as long as $x(n)$ is bounded, $y(n)$ will be bounded. Hence the system is stable.

Thus the given system is,

Static, nonlinear, shift invariant, causal and stable.

(vi) $y(n) = x(n)u(n)$

- The output depends upon present input only. Hence the system is static.
- The given system equation is,

$$y(n) = T\{x(n)\} = x(n)u(n)$$

The response of this system to the two inputs $x_1(n)$ and $x_2(n)$ when applied separately will be,

$$\left. \begin{aligned} y_1(n) &= T\{x_1(n)\} = x_1(n)u(n) \\ y_2(n) &= T\{x_2(n)\} = x_2(n)u(n) \end{aligned} \right\} \dots (2.6.28)$$

The response of the system to linear combination of inputs $x_1(n)$ and $x_2(n)$ will be,

$$\begin{aligned} y_3(n) &= T\{a_1 x_1(n) + a_2 x_2(n)\} \\ &= [a_1 x_1(n) + a_2 x_2(n)]u(n) \\ &= a_1 x_1(n)u(n) + a_2 x_2(n)u(n) \end{aligned} \dots (2.6.29)$$

The linear combination of two outputs of equation 2.6.28 will be,

$$\begin{aligned} y_3(n) &= a_1 y_1(n) + a_2 y_2(n) \\ &= a_1 x_1(n)u(n) + a_2 x_2(n)u(n) \end{aligned}$$

From above equation and equation 2.6.29 we find that,

$$y_3(n) = y_3(n), \text{ hence the system is linear.}$$

- The response of the system to delayed input will be,

$$\begin{aligned} y(n, k) &= T\{x(n-k)\} \\ &= x(n-k)u(n) \end{aligned} \dots (2.6.30)$$

The delayed output will be obtained by replacing 'n' by $n-k$. i.e.,

$$y(n-k) = x(n-k)u(n-k) \dots (2.6.31)$$

Here, on comparing above equation and equation 2.6.30, we find that,

$$y(n, k) \neq y(n-k) \text{ Hence the system is shift variant.}$$

- The system equation is, $y(n) = x(n)u(n)$. The output depends upon present input only. Hence the system is causal.
- We know that $u(n) = 1$ for $n \geq 0$ and $u(n) = 0$ for $n < 0$. This means $u(n)$ is a bounded sequence. Hence as long as $x(n)$ is bounded, $y(n)$ is also bounded. Hence this system is stable.

Thus, the given system is,

Static, linear, shift variant, causal and stable.

(vii) $y(n) = x(n) + nx(n+1)$:

1. From the given equation it is clear that, the output depends upon the present input and next input. Hence system is dynamic.
2. The given system equation is,

$$y(n) = T\{x(n)\} = x(n) + nx(n+1) \quad \dots (2.6.32)$$

If we apply two inputs $x_1(n)$ and $x_2(n)$ separately, then the outputs become,

$$\left. \begin{aligned} y_1(n) &= T\{x_1(n)\} = x_1(n) + nx_1(n+1) \\ \text{and } y_2(n) &= T\{x_2(n)\} = x_2(n) + nx_2(n+1) \end{aligned} \right\} \dots (2.6.32 (a))$$

Response of the system to linear combination of inputs $x_1(n)$ and $x_2(n)$ will be,

$$\begin{aligned} y_3(n) &= T\{a_1 x_1(n) + a_2 x_2(n)\} \\ &= a_1 [x_1(n) + nx_1(n+1)] + a_2 [x_2(n) + nx_2(n+1)] \end{aligned} \quad \dots (2.6.33)$$

The linear combination of two outputs given by equation 2.6.32 (a) will be,

$$\begin{aligned} y_3(n) &= a_1 y_1(n) + a_2 y_2(n) \\ &= a_1 [x_1(n) + nx_1(n+1)] + a_2 [x_2(n) + nx_2(n+1)] \end{aligned}$$

On comparing above equation with equation 2.6.33 we observe that,

$$y_3(n) = y_3(n), \text{ Hence the system is linear.}$$

3. The given system equation is,

$$y(n) = T\{x(n)\} = x(n) + nx(n+1) \quad \dots (2.6.34)$$

Response of the system to delayed input will be,

$$\begin{aligned} y(n, k) &= T\{x(n-k)\} \\ &= x(n-k) + nx(n-k+1) \end{aligned} \quad \dots (2.6.35)$$

Now let us delay the output of equation 2.6.34 by 'k' samples. i.e.,

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

Here we have replaced 'n' by 'n-k'. On comparing above equation with equation 2.6.35 we observe that,

$$y(n, k) \neq y(n-k), \text{ Hence the system is shift variant.}$$

4. The given system equation is,

$$y(n) = x(n) + nx(n+1)$$

Here observe that n^{th} output sample depends upon $(n+1)^{\text{th}}$ i.e. next input sample.

That is the output depends upon future input. Hence the system is noncausal.

5. In the given system equation observe that as $n \rightarrow \infty, y(n) \rightarrow \infty$ even if $x(n)$ is bounded. Hence the system unstable.

Thus the given system is,

Dynamic, linear, shift variant, noncausal and unstable.

(viii) $y(n) = x(2n)$:

1. By putting $n=1$ in the given system equation,

$$y(1) = x(2) \text{ similarly,}$$

$$n = 2 \Rightarrow y(2) = x(4)$$

$$n = 3 \Rightarrow y(3) = x(6) \text{ and so on.}$$

Thus the system needs to store the future input samples. Hence it requires memory. Therefore the system is dynamic.

2. The given system equation is,

$$y(n) = T\{x(n)\} = x(2n) \quad \dots (2.6.36)$$

For two separate inputs $x_1(n)$ and $x_2(n)$ the system produces the response of

$$\left. \begin{aligned} y_1(n) &= T\{x_1(n)\} = x_1(2n) \\ \text{and } y_2(n) &= T\{x_2(n)\} = x_2(2n) \end{aligned} \right\} \quad \dots (2.6.37)$$

The response of the system to linear combination of $x_1(n)$ and $x_2(n)$ will be,

$$\begin{aligned} y_3(n) &= T\{a_1 x_1(n) + a_2 x_2(n)\} \\ &= a_1 x_1(2n) + a_2 x_2(2n) \end{aligned} \quad \dots (2.6.38)$$

Now the linear combination of two outputs given by equation 2.6.37 will be,

$$\begin{aligned} y_3'(n) &= a_1 y_1(n) + a_2 y_2(n) \\ &= a_1 x_1(2n) + a_2 x_2(2n) \end{aligned}$$

On comparing above equation with equation 2.6.38 we find that,

$$y_3(n) = y_3'(n), \text{ Hence the system is linear.}$$

3. The given system equation is,

$$y(n) = T\{x(n)\} = x(2n) \quad \dots (2.6.39)$$

The response of the system to delayed input $x(n-k)$ will be,

$$y(n, k) = T\{x(n-k)\} = x(2n-k) \quad \dots (2.6.40)$$

Now let us delay the output $y(n)$ given by equation 2.6.39 by ' k ' samples. This is obtained by replacing ' n ' by $n-k$ in equation 2.6.39. i.e.,

$$\begin{aligned} y(n-k) &= x[2(n-k)] \\ \therefore y(n-k) &= x(2n-2k) \end{aligned}$$

On comparing above equation with equation 2.6.40 we find that,

$$y(n, k) \neq y(n-k). \text{ Hence the system is shift variant.}$$

4. The given system equation is,

$$y(n) = x(2n)$$

Here output depends upon future inputs. i.e. n^{th} sample of output depends upon $(2n)^{\text{th}}$ sample of input. Clearly the system is noncausal.

5. As long as $x(n)$ is bounded, then $x(2n)$ is also bounded. Hence output $y(n)$ is also bounded. Therefore the system is stable.

Thus the given system is,

Dynamic, linear, shift variant, noncausal and stable.

(ix) $y(n) = x(-n)$:

1. If the present input is $x(n)$, then output $y(n)$ is $x(-n)$. That is if input is $x(4)$ then output is $x(-4)$. Thus the system has to store input sequence in the memory. Hence the system is static.
2. The given system equation is,

$$y(n) = T\{x(n)\} = x(-n) \quad \dots (2.6.41)$$

When the two inputs $x_1(n)$ and $x_2(n)$ are applied separately, then the responses $y_1(n)$ and $y_2(n)$ will be,

$$\left. \begin{aligned} y_1(n) &= T\{x_1(n)\} = x_1(-n) \\ y_2(n) &= T\{x_2(n)\} = x_2(-n) \end{aligned} \right\} \dots (2.6.42)$$

The response of the system to the linear combination of $x_1(n)$ and $x_2(n)$ will be,

$$\begin{aligned} y_3(n) &= T\{a_1 x_1(n) + a_2 x_2(n)\} \\ &= a_1 x_1(-n) + a_2 x_2(-n) \end{aligned} \dots (2.6.43)$$

The linear combination of two outputs $y_1(n)$ and $y_2(n)$ given by equation 2.6.42 will be,

$$\begin{aligned} y_3(n) &= a_1 y_1(n) + a_2 y_2(n) \\ &= a_1 x_1(-n) + a_2 x_2(-n) \end{aligned} \dots (2.6.44)$$

On comparing above equation with equation 2.6.43 we find that,

$$y_3(n) = y_3(n), \text{ Hence the system is a linear system.}$$

3. Let us apply the delayed input to the system. Then the response will be,

$$\begin{aligned} y(n, k) &= T\{x(n-k)\} \\ &= x(-n-k) \end{aligned} \dots (2.6.45)$$

We know that the system equation is given as,

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

Now let us delay the output $y(n)$ by ' k ' samples. This can be obtained by replacing ' n ' by $(n-k)$ in the above equation. i.e.,

$$\begin{aligned} y(n-k) &= x[-(n-k)] \\ &= x(-n+k) \end{aligned} \dots (2.6.46)$$

On comparing above equation with equation 2.6.45 we observe that,

$$y(n, k) \neq y(n-k), \text{ Hence the system is shift variant.}$$

This same result we have obtained in example 2.6.1 also.

4. The given system equation is,

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

$$\text{For } n = -2 \Rightarrow y(n) = x(2)$$

$$\text{For } n = -1 \Rightarrow y(n) = x(1) \text{ etc.}$$

Thus the output depends upon future inputs. Hence the system is noncausal.

5. As long as $x(n)$ is bounded, $x(-n)$ will also be bounded. Hence the output $y(n)$ is also bounded. Therefore the system is stable.

Thus the given system is,

Static, linear, shift variant, noncausal and stable.

(x) $y(n) = \text{sgn}[x(n)]$:

- Since output depends upon the present input only, then this system is static.
- The given system equation is,

$$y(n) = T\{x(n)\} = \text{sgn}[x(n)]$$

When the two inputs $x_1(n)$ and $x_2(n)$ are applied separately, the responses $y_1(n)$ and $y_2(n)$ will be,

$$\left. \begin{aligned} y_1(n) &= T\{x_1(n)\} = \text{sgn}[x_1(n)] \\ y_2(n) &= T\{x_2(n)\} = \text{sgn}[x_2(n)] \end{aligned} \right\} \dots (2.6.47)$$

The response of the system to linear combination of $x_1(n)$ and $x_2(n)$ will be,

$$\begin{aligned} y_3(n) &= T\{a_1 x_1(n) + a_2 x_2(n)\} \\ &= \text{sgn}[a_1 x_1(n) + a_2 x_2(n)] \end{aligned} \quad \dots (2.6.48)$$

Basically $\text{sgn}[x(n)] = 1$ for $n > 0$ and $\text{sgn}[x(n)] = -1$ for $n < 0$. In the above equation $y_3(n)$ will have a value of '1' for $n > 0$ and '-1' for $n < 0$.

The linear combination of two outputs given by equation 2.6.47 will be,

$$\begin{aligned} y_3'(n) &= a_1 y_1(n) + a_2 y_2(n) \\ &= a_1 \text{sgn}[x_1(n)] + a_2 \text{sgn}[x_2(n)] \end{aligned} \quad \dots (2.6.49)$$

In the above equation $\text{sgn}[x_1(n)] = 1$ for $n > 0$ and -1 for $n < 0$. Similarly $\text{sgn}[x_2(n)] = 1$ for $n > 0$ and -1 for $n < 0$. Hence the values of $y_3'(n)$ of above equation and $y_3(n)$ of equation 2.6.48 are not same.

i.e. $y_3(n) \neq y_3'(n)$. Hence the system is nonlinear.

3. The given system equation is,

$$y(n) = T\{x(n)\} = \text{sgn}[x(n)] \quad \dots (2.6.50)$$

The response of the system to delayed input $x(n-k)$ will be,

$$y(n, k) = T\{x(n-k)\} = \text{sgn}[x(n-k)] \quad \dots (2.6.51)$$

$$\begin{aligned} \text{Here } \text{sgn}[x(n-k)] &= 1 \text{ for } n > 0 \text{ and} \\ &= -1 \text{ for } n < 0 \end{aligned}$$

The delayed output $y(n-k)$ can be obtained from equation 2.6.50 by replacing 'n' by $(n-k)$ i.e.,

$$y(n-k) = \text{sgn}[x(n-k)] \quad \dots (2.6.52)$$

On comparing above equation with equation 2.6.51 we find that,

$$y(n, k) = y(n-k), \text{ Hence the system is shift invariant.}$$

4. Since $y(n)$ depends upon the present input. This is a causal system.

5. Since $\text{sgn}[x(n)]$ has a value of ± 1 depending upon 'n', this is a stable system.

Thus the given system is,

Static, nonlinear, shift invariant, causal and stable.

Ex. 2.6.6 For the systems represented by following functions, determine whether every system is,

(i) Stable (ii) Causal (iii) Linear (iv) Shift invariant.

1) $T[x(n)] = e^{x(n)}$

2) $T[x(n)] = ax(n) + 6$

[Dec - 99]

Sol. : (1) $y(n) = T[x(n)] = e^{x(n)}$:

(i) The given system equation is,

$$y(n) = T\{x(n)\} = e^{x(n)} \quad \dots (2.6.53)$$

For bounded $x(n)$, $e^{x(n)}$ is also bounded. [Note that value of 'e' is basically 2.7182818]. Hence the given system is a stable system.

(ii) Present value of output depends upon present value of input. Hence this is a causal system.

(iii) The response of the system to linear combination of two inputs $x_1(n)$ and $x_2(n)$ will be,

$$y_3(n) = T\{a_1 x_1(n) + a_2 x_2(n)\}$$

From equation 2.6.53 we can write,

$$\begin{aligned} y_3(n) &= e^{a_1 x_1(n) + a_2 x_2(n)} \\ &= e^{a_1 x_1(n)} \cdot e^{a_2 x_2(n)} \end{aligned} \quad \dots (2.6.54)$$

The response of the system to two inputs $x_1(n)$ and $x_2(n)$ when applied separately is given as,

$$\left. \begin{aligned} y_1(n) &= T\{x_1(n)\} = e^{x_1(n)} \\ y_2(n) &= T\{x_2(n)\} = e^{x_2(n)} \end{aligned} \right\} \quad \dots (2.6.55)$$

The linear combination of two outputs given by above equation will be,

$$\begin{aligned} y_3(n) &= a_1 y_1(n) + a_2 y_2(n) \\ &= a_1 e^{x_1(n)} + a_2 e^{x_2(n)} \end{aligned} \quad \text{Putting values from eq. 2.6.55}$$

On comparing above equation with equation 2.6.54 we find that,

$$y_3(n) \neq y_3(n), \text{ Hence this system is nonlinear.}$$

(iv) The response of the system to delayed input will be,

$$\begin{aligned} y(n, k) &= T\{x(n-k)\} \\ &= e^{k(n-k)} \end{aligned} \quad \dots (2.6.56)$$

The delayed output can be obtained by replacing 'n' by 'n-k' in equation 2.6.53. i.e.,

$$y(n-k) = e^{x(n-k)}$$

On comparing above equation with equation 2.6.56 we find that,

$$y(n, k) = y(n-k), \text{ Hence the system is shift invariant.}$$

Thus this given system is,

Stable, causal, nonlinear and shift invariant.

$$(2) \quad y(n) = T\{x(n)\} = a x(n) + 6 :$$

(i) In the given system equation 'a' is constant. As long as $x(n)$ is bounded, $y(n)$ is also bounded. Hence the system is stable.

(ii) Present output depends upon present input only. Hence this is as causal system.

(iii) The response of the system to linear combination of two inputs $x_1(n)$ and $x_2(n)$ will be,

$$\begin{aligned} y_3(n) &= T\{a_1 x_1(n) + a_2 x_2(n)\} \\ &= a[a_1 x_1(n) + a_2 x_2(n)] + 6 \end{aligned} \quad \dots (2.6.57)$$

Here above equation is written from given system equation with $x(n) = a_1 x_1(n) + a_2 x_2(n)$. The response of the system to two inputs $x_1(n)$ and $x_2(n)$ when applied separately is given as,

$$\left. \begin{aligned} y_1(n) &= T\{x_1(n)\} = a x_1(n) + 6 \\ \text{and } y_2(n) &= T\{x_2(n)\} = a x_2(n) + 6 \end{aligned} \right\} \quad \dots (2.6.58)$$

The linear combination of two outputs given by above equation will be,

$$y_3(n) = a_1 y_1(n) + a_2 y_2(n)$$

$$= a_1 [a x_1(n) + 6] + a_2 [a x_2(n) + 6]$$

Putting values from equation 2.6.58

On comparing above equation with equation 2.6.57 we find that,

$$y_3(n) \neq y'_3(n), \text{ Hence the system is nonlinear.}$$

(iv) The response of the system to the input delayed by 'k' samples will be,

$$\begin{aligned} y(n, k) &= T\{x(n-k)\} \\ &= a x(n-k) + 6 \end{aligned} \quad \dots (2.6.59)$$

And the delayed output by 'k' samples will be obtained by replacing 'n' by (n-k) in given system equation. i.e.,

$$y(n-k) = a x(n-k) + 6$$

On comparing above equation with equation 2.6.59 we find that,

$$y(n, k) = y(n-k), \text{ Hence the system is shift invariant.}$$

Thus the given system is,

Stable, causal, nonlinear and shift invariant.

Ex. 2.6.7 A discrete time system is given as,

$$y(n) = y^2(n-1) + x(n)$$

A bounded input of $x(n) = 2\delta(n)$ is applied to the system. Assume that the system is initially relaxed. Check whether this system is stable or unstable.

Sol. : The input $x(n) = 2\delta(n)$ is applied. We know that,

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

$$\therefore x(n) = \begin{cases} 2 & \text{for } n = 0 \\ 0 & \text{for } n \neq 0 \end{cases}$$

$$\therefore y(0) = y^2(0-1) + x(0)$$

Since system is initially relaxed, $y^2(0-1) = y^2(-1) = 0$

and $x(0) = 2$ as illustrated above.

$$\therefore y(0) = 2$$

$$\begin{aligned} \text{Now } y(1) &= y^2(1-1) + x(1) \\ &= y^2(0) + x(1) \end{aligned}$$

Since $x(n) = 0$ when $n \neq 0$, above equation becomes,

$$y(1) = y^2(0) = 2^2$$

$$\begin{aligned} \text{Similarly, } y(2) &= y^2(2-1) + x(2) \\ &= y^2(1) \end{aligned}$$

$$= (2^2)^2 = 2^4 = 2^{2^2}$$

$$y(3) = y^2(3-1) + x(3)$$

$$\begin{aligned}
 &= y^2(2) \\
 &= (2^4)^2 = 2^8 = 2^{2^3} \\
 y(4) &= y^2(4-1) + x(4) \\
 &= y^2(3) \\
 &= (2^8)^2 = 2^{16} = 2^{2^4}
 \end{aligned}$$

Similarly $y(n) = 2^{2^n}$

Here as $n \rightarrow \infty$, $y(n) \rightarrow \infty$. Thus the input $x(n) = 2\delta(n)$ is bounded but output $y(n)$ is not bounded for all 'n'. Hence the system is unstable.

Ex. 2.6.8 Two systems are connected in cascade as shown in Fig. 2.6.2.

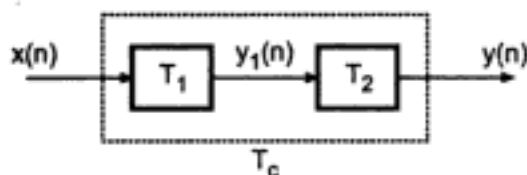


Fig. 2.6.2 Cascade connection of two systems

Prove that

(i) $T_c = T_2 T_1$

(ii) If T_1 and T_2 are shift invariant, the T_c is also shift invariant.

Sol. : The output of first system is $y_1(n)$. It is given as,

$$y_1(n) = T_1 \{x(n)\} \quad \dots (2.6.60)$$

The output of second system is $y(n)$. It is given as,

$$y(n) = T_2 \{y_1(n)\}$$

Putting for $y_1(n)$ from equation 2.6.60 we can write,

$$\begin{aligned}
 y(n) &= T_2 \{T_1 [x(n)]\} \\
 &= T_2 T_1 \{x(n)\}
 \end{aligned} \quad \dots (2.6.61)$$

If we consider overall system T_c , then we can write,

$$y(n) = T_c \{x(n)\}$$

From above equation and equation 2.6.61 we can write,

$$T_c = T_2 T_1 \quad \dots (2.6.62)$$

(ii) Since T_1 is shift invariant we can write,

$$y_1(n-k) = T_1 \{x(n-k)\} \quad \dots (2.6.63)$$

Similarly since T_2 is shift invariant we can write,

$$y(n-k) = T_2 \{y_1(n-k)\}$$

Putting for $y_1(n-k)$ from equation 2.6.63 in above equation,

$$\begin{aligned}
 y(n-k) &= T_2 \{T_1 [x(n-k)]\} \\
 &= T_2 T_1 \{x(n-k)\}
 \end{aligned}$$

Since $T_c = T_2 T_1$ above equation becomes,

$$y(n-k) = T_c \{x(n-k)\}$$

This shows that the overall cascade system T_c is also shift invariant.

Ex. 2.6.9 Two systems are connected in parallel as shown in Fig. 2.6.3.

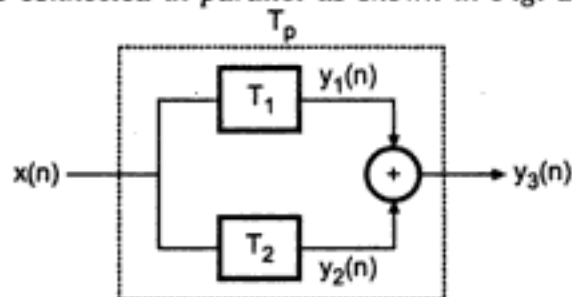


Fig. 2.6.3 Parallel connection of two systems

Prove that $T_p = T_1 + T_2$

Sol. : The outputs of systems T_1 and T_2 are related to input as,

$$y_1(n) = T_1 \{x(n)\}$$

and

$$y_2(n) = T_2 \{x(n)\}$$

The output $y_3(n)$ can be given as,

$$y_3(n) = y_1(n) + y_2(n)$$

$$= T_1 \{x(n)\} + T_2 \{x(n)\}$$

$$= (T_1 + T_2) \{x(n)\} \quad \dots (2.6.64)$$

For the overall system,

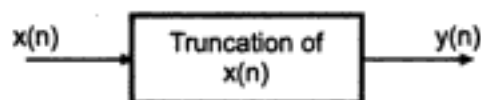
$$y_3(n) = T_p \{x(n)\}$$

Hence from above equation and equation 2.6.64 we can write,

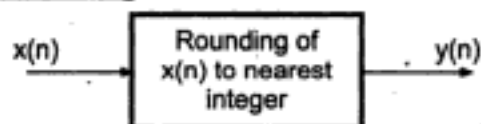
$$T_p = T_1 + T_2 \quad \dots (2.6.65)$$

Ex.2.6.10 Following systems are used in sampling and quantization of signals.

1. Truncation



2. Rounding



3. Sampling

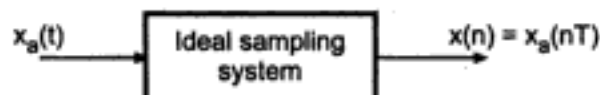


Fig. 2.6.4 Truncation, Rounding and Sampling systems

State whether these systems are

(i) Static (ii) Linear (iii) Shift invariant (iv) Causal and (v) Stable.

Sol. : 1. Truncation of $x(n)$:

(i) Since the output $y(n)$ is the integer obtained by truncation of $x(n)$, there is no need to store any samples of $x(n)$. Present output depends upon present input. Hence truncation operation is static.

(ii) Let the two input samples be,

$$x_1(n) = 2.713$$

$$x_2(n) = 3.521$$

When these two inputs are applied separately the outputs of truncation become,

$$y_1(n) = \text{Trunc} \{x_1(n)\} = \text{Trunc} \{2.713\} = 2$$

$$\text{and } y_2(n) = \text{Trunc} \{x_2(n)\} = \text{Trunc} \{3.521\} = 3 \quad \dots (2.6.66)$$

The linear combination of these two outputs become,

$$\begin{aligned} y_3(n) &= y_1(n) + y_2(n) \\ &= 2 + 3 \text{ from equation 2.6.66} \\ &= 5 \quad \dots (2.6.67) \end{aligned}$$

Let us consider the output of the truncation system to linear combination of two inputs. i.e.,

$$\begin{aligned} y_3'(n) &= \text{Trunc} \{x_1(n) + x_2(n)\} \\ &= \text{Trunc} \{2.713 + 3.521\} \\ &= \text{Trunc} \{6.234\} \\ &= 6 \end{aligned}$$

On comparing $y_3'(n)$ obtained above with $y_3(n)$ of equation 2.6.67, we find that,

$y_3(n) \neq y_3'(n)$, Hence truncation is nonlinear.

(iii) If the input $x(n)$ is delayed, there will be no effect on the result of truncation. Hence truncation operation is shift invariant.

(iv) The truncation operation is done only on present input sample. Hence the truncation operation is causal.

(v) As long as input $x(n)$ is bounded, the output integer obtained by truncation is also bounded. Hence truncation is stable system.

Thus truncation system is,

Static, nonlinear, shift invariant, causal and stable.

(2) Rounding of $x(n)$:

On the same lines as truncation, it can be shown easily that rounding system is,

Static, nonlinear, shift invariant, causal and stable.

(3) Ideal sampling system :

(i) There is no need to store the input analog signal. At the time of sampling the output $x(n)$ is equal to amplitude of $x(t)$. Hence ideal sampling is the static operation.

(ii) The sampled amplitude i.e. value of $x(n)$ is exactly equal to the instantaneous amplitude of $x_a(t)$. Hence the system is linear.

(iii) The sampling operation is independent of time shift. Hence it is shift invariant system.

(iv) The output $x(n)$ depends upon amplitude of input at time ' nT '. Hence this is a causal system.

(v) As long as input $x_a(t)$ is bounded, its samples, $x(n) = x_a(nT)$ are also bounded. Hence this is stable system.

Thus ideal sampling system is,

Static, linear, shift invariant, causal and stable.

2.7 Linear Time Invariant (LTI) Systems

We classified discrete time systems in the last section. We also studied the properties like linearity, causality, shift invariance, stability etc. Majority of the discrete time systems in practice are linear and shift invariant. Hence we will study Linear Shift Invariant (LTI) systems in more details in this section. Such systems are also called as Linear Time Invariant (LTI) systems. In this section we will also introduce the Linear Convolution. This is a powerful analytical tool for studying LTI systems. We will also demonstrate that the LTI system can be completely characterized by its unit sample response.

2.7.1 Discrete Time Signal as Weighted Impulses

Here we will first see how a discrete time signal can be expressed in the form of weighted impulses. Consider the arbitrary discrete time signal $x(n)$ of five samples :

$$x(n) = \{2, 1, 3, -2, 1\} \quad \dots (2.7.1)$$

Here $x(-2) = 2$, $x(-1) = 1$, $x(0) = 3$, $x(1) = -2$ and $x(2) = 1$. This signal is shown graphically in Fig. 2.7.1. below.

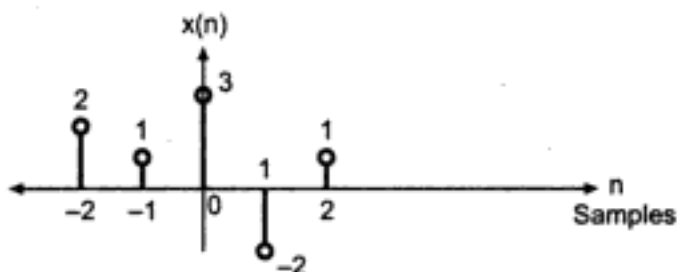


Fig. 2.7.1 Sketch of discrete time signal of equation 2.7.1

Now let us consider the unit sample sequence $\delta(n)$ defined in section 2.3.1. Fig. 2.3.1 shows the sketch of $\delta(n)$. It is defined as,

$$\left. \begin{aligned} \delta(n) &= 1 && \text{for } n = 0 \\ &= 0 && \text{for } n \neq 0 \end{aligned} \right\} \text{By equation 2.3.1} \quad \dots (2.7.2)$$

Now if $\delta(n)$ is delayed by one sample, then above equation will be,

$$\left. \begin{aligned} \delta(n-1) &= 1 && \text{for } n = 1 \\ &= 0 && \text{for } n \neq 1 \end{aligned} \right\}$$

Similarly if $\delta(n)$ is advanced by one sample, then it can be represented as,

$$\left. \begin{aligned} \delta(n+1) &= 1 && \text{for } n = -1 \\ &= 0 && \text{for } n \neq -1 \end{aligned} \right\}$$

This can be generalized. The $\delta(n)$ function delayed by 'k' samples can be represented as,

$$\left. \begin{aligned} \delta(n-k) &= 1 && \text{for } n = k \\ &= 0 && \text{for } n \neq k \end{aligned} \right\} \quad \dots (2.7.3)$$

Here k can be positive or negative. Fig. 2.7.2 shows the sketch of a signal $\delta(n-k)$ for $k=2$.

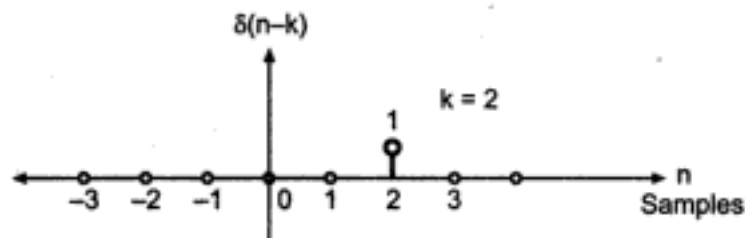


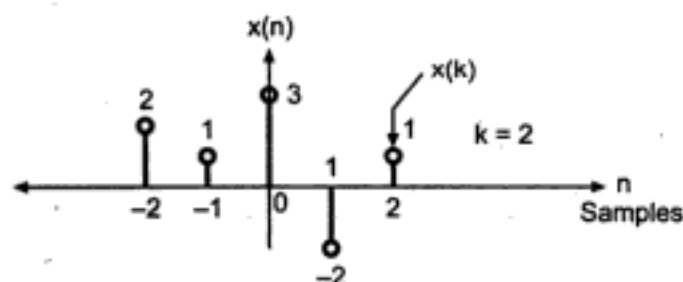
Fig. 2.7.2 Sketch of a delayed unit sample sequence. Here delay, $k=2$ samples

Now let us multiply $x(n)$ and $\delta(n-k)$. The value of $\delta(n-k)$ is zero everywhere except at $n=k$. Hence the result of multiplication becomes,

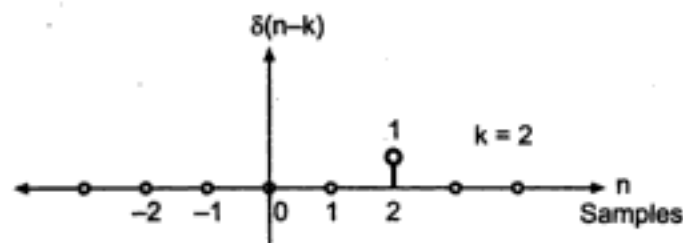
$$x(n)\delta(n-k) = x(k)\delta(n-k) \quad \dots (2.7.4)$$

This operation is indicated in Fig. 2.7.3. Fig. 2.7.3 (a) shows the sequence $x(n)$ of Fig. 2.7.1. Fig. 2.7.3 (b) shows the delayed unit sample sequence $\delta(n-k)$ of Fig. 2.7.2. Fig. 2.7.3(c) shows the product sequence $x(n)\delta(n-k)$. Thus equation 2.7.4 is clear since $x(n)=x(k)$ at $n=k$. Fig. 2.7.4 shows the various product sequences for $x(n)$ of Fig. 2.7.3. From Fig. 2.7.4 it is clear that if we add all the product sequences $x(k)\delta(n-k)$, then we get

(a). A discrete-time sequence $x(n)$



(b). A delayed unit sample sequence $\delta(n-k)$. Here $k=2$.



(c). Product sequence $x(n)\delta(n-k)$; which is same as $x(k)\delta(n-k)$. Here $k=2$.

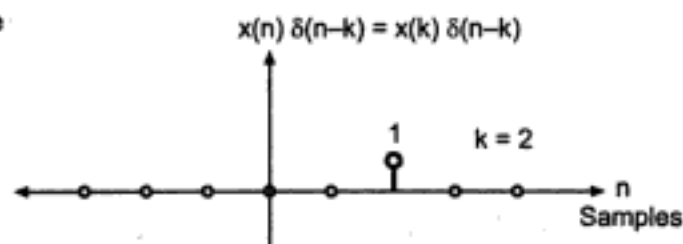
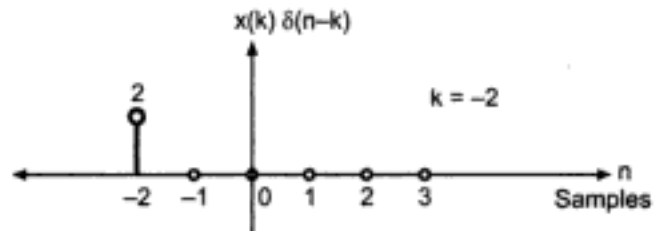
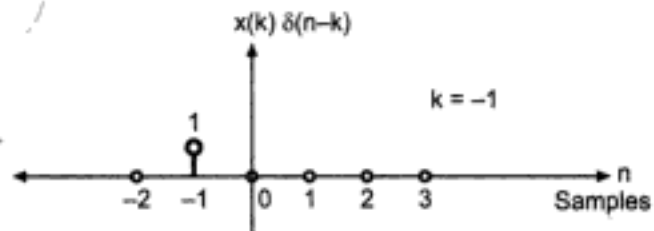


Fig. 2.7.3 The product $x(n)\delta(n-k) = x(k)\delta(n-k)$ since $\delta(n-k)=1$ at $n=k$ and $x(n)=x(k)$

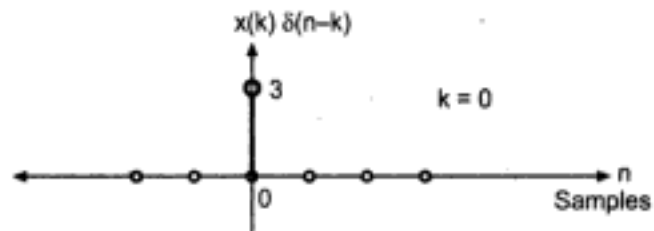
(a). The product sequence at $k = -2$. It is unit sample weighted by $x(-2) = 2$ at $n = -2$.



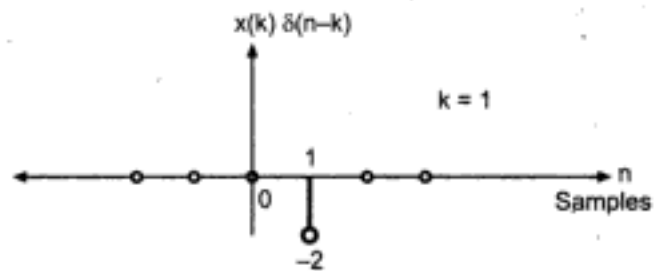
(b). The product sequence at $k = -1$. It is unit sample weighted by $x(-1) = 1$ at $n = -1$.



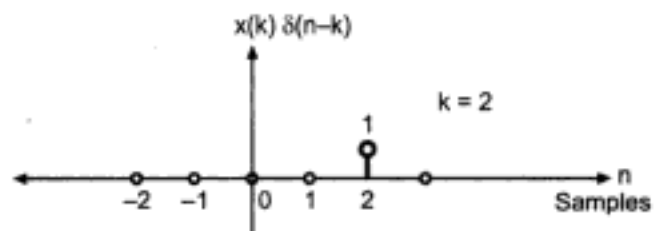
(c). $k = 0$. Unit sample weighted by $x(0) = 3$ at $n = 0$.



(d). $k = 1$. Unit sample weighted by $x(1) = -2$ at $n = 1$.



(e). $k = 2$. Unit sample weighted by $x(2) = 1$ at $n = 2$.



(f). A summation of product sequences in Fig (a) to (e) results in sequence $x(n)$

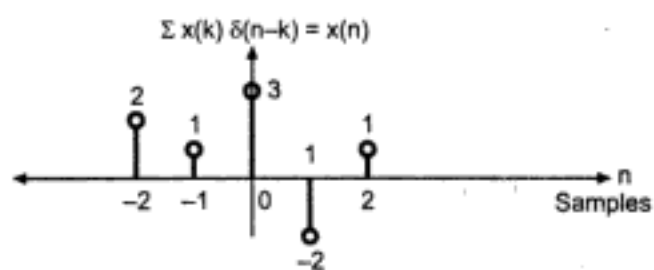


Fig. 2.7.4 A discrete time signal $x(n)$ is expressed as sum of weighted unit samples

$x(n)$. Here ' k ' varies from -2 to $+2$. For the generalized case the range of ' k ' will be $-\infty < k < \infty$, to accommodate all possible samples of $x(n)$. Thus,

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k) \quad \dots (2.7.5)$$

In this result the infinite number of unit sample sequences are added to get $x(n)$. The unit sample sequence $\delta(n-k)$ takes an amplitudes of $x(k)$. The result of equation 2.7.5 is an important one useful for convolution as we will see.

For the sequence $x(n)$ given by equation 2.7.1, the range of ' k ' will be, $-2 \leq k \leq 2$. Hence equation 2.7.5 above can be written as,

$$x(n) = \sum_{k=-2}^2 x(k)\delta(n-k)$$

Expanding above summation, we get,

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

Putting the values of $x(n)$ from equation 2.7.1, above equation becomes,

$$x(n) = 2\delta(n+2) + \delta(n+1) + 3\delta(n) - 2\delta(n-1) + \delta(n-2) \quad \dots (2.7.6)$$

Thus $x(n)$ is equal to the summation of unit samples. The amplitudes of unit samples are basically sample values of $x(n)$. The unit samples represent the location of particular sample or its delay in the sequence $x(n)$.

2.7.2 Linear Convolution

Linear convolution is a very powerful technique used for the analysis of LTI systems. In the last subsection we have seen that how the sequence $x(n)$ can be expressed as sum of weighted impulses. It is given by equation 2.7.5 as,

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k) \quad \dots (2.7.7)$$

If $x(n)$ is applied as an input to the discrete time system, then response $y(n)$ of the system is given as,

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

Putting for $x(n)$ in above equation from equation 2.7.7,

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

The above equation can be expanded as,

$$y(n) = T \left[\dots + x(-3)\delta(n+3) + x(-2)\delta(n+2) + x(-1)\delta(n+1) + x(0)\delta(n) \right. \\ \left. + x(1)\delta(n-1) + x(2)\delta(n-2) + x(3)\delta(n-3) + \dots \right] \quad \dots (2.7.8)$$

Since the system is *linear*, the above equation can be written as,

$$y(n) = \dots + T[x(-3)\delta(n+3)] + T[x(-2)\delta(n+2)] + T[x(-1)\delta(n+1)] + T[x(0)\delta(n)] \\ + T[x(1)\delta(n-1)] + T[x(2)\delta(n-2)] + T[x(3)\delta(n-3)] + \dots \quad \dots (2.7.9)$$

The linearity property states that output due to linear combination of inputs [equation 2.7.8] is same as the sum of outputs due to individual inputs [equation 2.7.9]. Thus we have written equation 2.7.9 from eq. 2.7.8 with the help of linearity property. In the above equation [equation 2.7.9] the sample values $\dots x(-3), x(-2), x(-1), x(0), x(1), x(2)\dots$ etc. are constants. Hence with the help of *scaling* property of *linear* systems we can write equation 2.7.9 as,

$$y(n) = \dots + x(-3)T[\delta(n+3)] + x(-2)T[\delta(n+2)] + x(-1)T[\delta(n+1)] + x(0)T[\delta(n)] + x(1)T[\delta(n-1)] + x(2)T[\delta(n-2)] + x(3)T[\delta(n-3)] + \dots \quad \dots (2.7.10)$$

The above equation we have written on the basis of scaling property. It states that if $y(n) = T[ax(n)]$, then $y(n) = aT[x(n)]$ for $a = \text{constant}$. The above equation can be written in compact form with the help of ' \sum ' sign. i.e.,

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)T[\delta(n-k)] \quad \dots (2.7.11)$$

The response of the system to unit sample sequence $\delta(n)$ is given as,

$$T[\delta(n)] = h(n) \quad \dots (2.7.12)$$

Here $h(n)$ is called unit sample response or impulse response of the system. If the discrete time system is shift invariant, then above equation can be written as,

$$T[\delta(n-k)] = h(n-k) \quad \dots (2.7.13)$$

Here ' k ' is some shift in samples. The above equation indicates that; if the excitation of the shift invariant system is delayed, then its response is also delayed by the same amount. Putting for $T[\delta(n-k)] = h(n-k)$ in equation 2.7.11 we get,

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \quad \dots (2.7.14)$$

This equation gives the response of linear shift invariant (LTI) system or LTI system to an input $x(n)$. The behaviour of the LTI system is completely characterized by the unit sample response $h(n)$. The above equation is basically linear convolution of $x(n)$ and $h(n)$. This linear convolution gives $y(n)$. Thus,

Linear Convolution : $y(n) = x(n) * h(n)$

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \quad \dots (2.7.15)$$

$y(n)$ is the response, $x(n)$ is the input to the system and $h(n)$ depends upon characteristics of the system. Fig. 2.7.5 illustrates this relationship.

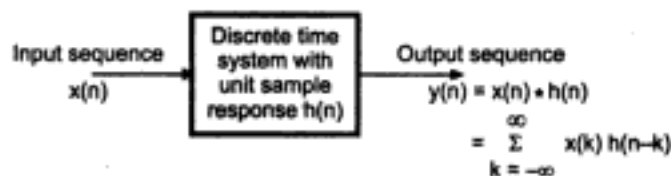


Fig. 2.7.5 Convolution of unit sample response $h(n)$ and input sequence $x(n)$ gives output $y(n)$ in LTI system

Computation of Linear Convolution :

Now let us see how to calculate $y(n)$ by using linear convolution of equation 2.7.15. With $n = 0$, equation 2.7.15 becomes,

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

Here $x(k)$ and $h(-k)$ are multiplied on sample to sample basis and added together to get $y(0)$. Observe that basically $h(-k)$ is the folded sequence of $h(k)$. Similarly with $n=1$, equation 2.7.15 becomes,

$$\begin{aligned} n = 1 \Rightarrow y(1) &= \sum_{k=-\infty}^{\infty} x(k) h(1-k) \\ &= \sum_{k=-\infty}^{\infty} x(k) h[-(k-1)] \text{ By rearranging the equation.} \end{aligned}$$

Here again $x(k)$ and $h[-(k-1)]$ are multiplied on sample to sample basis and added together to give $y(1)$. Here $h[-(k-1)]$ indicates shifted (or delayed) version of $h(-k)$ by one sample since $n=1$.

Similarly $y(n)$ can be calculated for other values. Thus the operations in computation of convolution are as follows :

- (i) **Folding** : Sequence $h(k)$ is folded at $k=0$, to get $h(-k)$
- (ii) **Shifting** : $h(-k)$ is shifted depending upon the value of 'n' in $y(n)$.
- (iii) **Multiplication** : $x(k)$ and $h(n-k)$ are then multiplied on sample to sample basis.
- (iv) **Summation** : The product sequence obtained by multiplication of $x(k)$ and $h(n-k)$ is added over all values of 'k' to get value of $y(n)$.

These operations will be more clear through the following example.

Ex. 2.7.1 Convolve the following two sequences $x(n)$ and $h(n)$ to get $y(n)$

$$x(n) = \{1, 1, 1, 1\}$$

$$h(n) = \{2, 2\}$$

Sol. : Here upward arrow (\uparrow) is not shown in $x(n)$ as well as $h(n)$ means, the first sample in the sequence is 0th sample. Thus the sample values are :

$$x(k=0) = 1$$

$$x(k=1) = 1$$

$$x(k=2) = 1$$

$$x(k=3) = 1$$

and

$$h(k=0) = 2$$

$$h(k=1) = 2$$

The convolution of $x(n)$ and $h(n)$ is given by equation 2.7.15 as,

$$y(n) = \sum_{k=-\infty}^{\infty} x(k) h(n-k) \quad \dots (2.7.15 \text{ (a)})$$

for $n=0$, the above equation gives 0th sample of $y(n)$ i.e.,

$$n \Rightarrow 0, \quad y(0) = \sum_{k=-\infty}^{\infty} x(k) h(-k) \quad \dots (2.7.16)$$

Here $h(-k)$ is obtained by folding $h(k)$ around $k=0$. Fig. 2.7.6 (a) shows $x(k)$, Fig. 2.7.6(b) shows $h(k)$ and Fig. 2.7.6 (c) shows $h(-k)$. The product $x(k)h(-k)$ is obtained by multiplying sequences of Fig. 2.7.6 (a) and Fig. 2.7.6 (c) sample to sample.

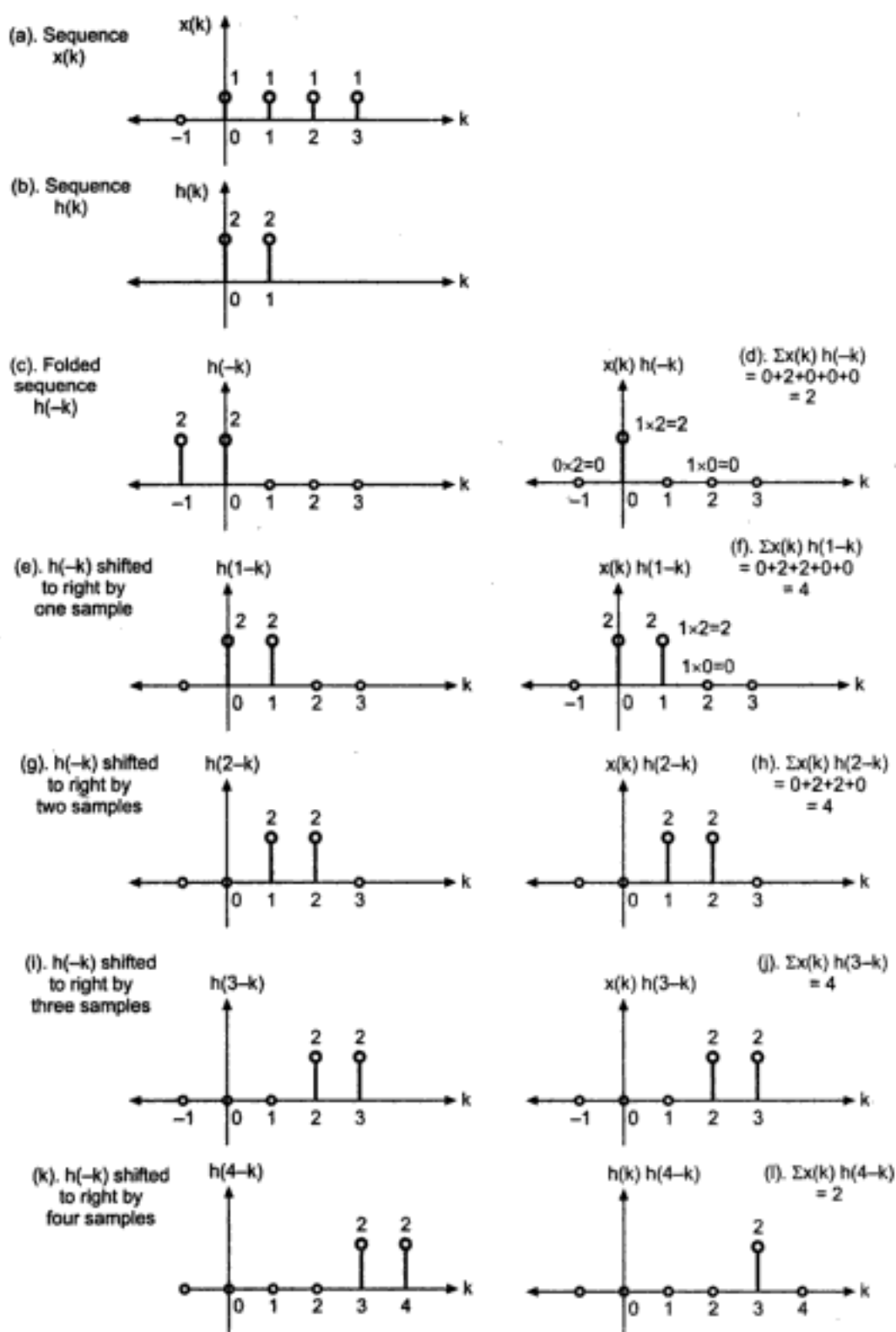


Fig. 2.7.6 Illustration of linear convolution

The product of sequence is shown in Fig. 2.7.6 (d).

The summation of samples of this product sequence gives $y(0)$ i.e.,

$$y(0) = \sum_{\text{over all samples}} x(k) h(-k) = 0 + 2 + 0 + 0 + 0 = 2$$

$$y(0) = 2$$

Now let $n=1$ in equation 2.7.15 to determine value of $y(1)$ i.e.,

$$n \Rightarrow 1 \quad y(1) = \sum_{k=-\infty}^{\infty} x(k) h(1-k) \quad \dots (2.7.17)$$

Here $h(1-k)$ can be written as $h[-(k-1)]$. This is nothing but $h(-k)$ shifted to right by one sample. This shifted sequence $h(1-k)$ is shown in Fig. 2.7.6 (e). The product sequence $x(k)h(1-k)$ is obtained by multiplying $x(k)$ of Fig. 2.7.6 (a) and $h(1-k)$ of Fig. 2.7.6 (e) sample to sample. This product sequence is shown in Fig. 2.7.6 (f). The summation of samples of this product sequence gives $y(1)$ i.e.,

$$y(1) = \sum_{\text{over all samples}} x(k) h(1-k) = 0 + 2 + 2 + 0 + 0 = 4$$

$$y(1) = 4$$

Now let $n=2$ in equation 2.7.15 to determine value of $y(2)$ i.e.,

$$n = 2 \Rightarrow \quad y(2) = \sum_{k=-\infty}^{\infty} x(k) h(2-k) \quad \dots (2.7.18)$$

Here $h(2-k)$ can be written as $h[-(k-2)]$. This is nothing but sequence $h(-k)$ shifted to right by two samples. This delayed sequence is shown in Fig. 2.7.6 (g). The product sequence $x(k)h(2-k)$ is obtained by multiplying $x(k)$ of Fig. 2.7.6 (a) and $h(2-k)$ of Fig. 2.7.6 (g) sample to sample. This product sequence is shown in Fig. 2.7.6 (h). The summation of the samples of this product sequence gives $y(2)$ [see equation 2.7.18]. i.e.,

$$y(2) = \sum_{\text{over all samples}} x(k) h(2-k) = 0 + 0 + 2 + 2 + 0 = 4$$

$$y(2) = 4$$

Similarly for $n=3$, equation 2.7.15 becomes,

$$n = 3 \Rightarrow \quad y(3) = \sum_{k=-\infty}^{\infty} x(k) h(3-k)$$

$h(3-k)$ is obtained by shifting $h(-k)$ to right by 3 samples. i.e. delaying by 3 samples. Fig. 2.7.6 (i) shows $h(3-k)$. The product sequence $x(k)h(3-k)$ is shown in Fig. 2.7.6 (j). Hence

$$y(3) = \sum_{\text{over all samples}} x(k) h(3-k) = 0 + 0 + 0 + 2 + 2 = 4$$

$$y(3) = 4$$

This is not the last sample of $y(n)$. With $n=4$ in equation 2.7.15 we get,

$$n = 4 \Rightarrow \quad y(4) = \sum_{k=-\infty}^{\infty} x(k) h(4-k)$$

$h(4-k)$ is obtained by delaying or shifting $h(-k)$ to right by four samples. This is shown in Fig. 2.7.6 (k). The product sequence $x(k)h(4-k)$ is shown in Fig. 2.7.6 (l). Hence

$$y(4) = \sum_{\text{over all samples}} x(k) h(4-k) = 0 + 0 + 0 + 0 + 2 + 0 = 2$$

$$y(4) = 2$$

Now let us see whether we get $y(5)$. That is with $n=5$ in equation 2.7.15 we get,

Hidden page

Now let us see whether we get some samples of $y(n)$ for $n < 0$. Let $n = -1$ in equation 2.7.15 we get,

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

Here $h(-1-k)$ can be written as $h[-(k+1)]$. This is nothing but sequence $h(-k)$ advanced by one sample. Such sequence can be obtained by shifting $h(-k)$ of Fig. 2.7.6 (c) left by one sample. This shifted sequence is shown in Fig. 2.7.7 (d). Here observe that $x(k) = 0$ for $k < 0$ and $h(-1-k) = 0$ for $k > -1$. Hence the product sequence $x(k) h(-1-k)$ is zero for all samples as shown in fig. 2.7.7 (e). Thus,

$$y(-1) = \sum_{\substack{\text{over all} \\ \text{samples}}} x(k) h(-1-k) = 0$$

$y(n \leq -1) = 0$

Hence all next samples of $y(n)$ for $n \leq -1$ will be zero only.

Thus the sequence $y(n)$ becomes as shown below :

$$y(n) = \{ \dots, 0, 0, \underset{\uparrow}{2}, 4, 4, 4, 2, 0, 0, \dots \}$$

or it can also be written as,

$$y(n) = \{ 2, 4, 4, 4, 2 \}$$

Here note that we have used graphical sketches to evaluate convolution.

Comments :

1. Convolution involves folding, shifting, multiplication and summation operations as discussed in this example.
2. If there are 'M' number of samples in $x(n)$ and 'N' number of samples in $h(n)$; then the maximum number of samples in $y(n)$ is equal to $M + N - 1$.

In this example $M = 4$ and $N = 2$. Hence number of samples in $y(n)$ are $4 + 2 - 1 = 5$. Thus there are maximum five samples in $y(n)$.

Ex. 2.7.2 *Recompute the sequence $y(n)$ in example 2.7.1 with the help of basic convolution equation.*

Sol. : The given $x(n)$ and $h(n)$ are,

$$\begin{aligned} x(0) &= 1 & h(0) &= 2 \\ x(1) &= 1 & h(1) &= 2 \\ x(2) &= 1 \\ x(3) &= 1 \end{aligned}$$

The linear convolution of $x(n)$ and $h(n)$ is given by equation 2.7.15 as,

$$y(n) = \sum_{k=-\infty}^{\infty} x(k) h(n-k) \quad \dots (2.7.19)$$

Here clearly $x(k) = 0$ for $k < 0$ in the given sequence. Hence lower limit on 'k' will be 0 in above equation.

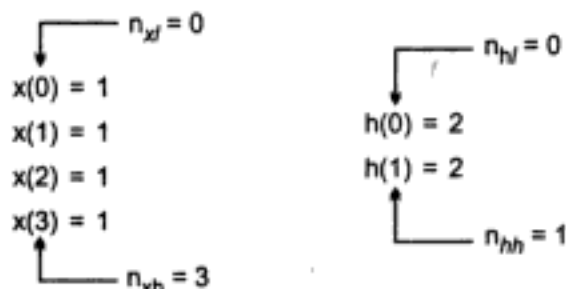
Similarly $x(k) = 0$ for $k > 3$ in the given sequence. Hence upper limit on 'k' will be 3 in above equation. Hence equation 2.7.19 can be written as,

$$y(n) = \sum_{k=0}^3 x(k) h(n-k) \quad \dots (2.7.20)$$

To determine range of 'n' in $y(n)$:

Let us represent lowest index of $x(n)$ as n_{xl} , which is zero for given sequence.

Similarly let n_{xh} be the highest index in $x(n)$. For given sequence of $x(n)$, $n_{xh} = 3$. Similarly let lowest index in $h(n)$ be n_{hl} ; which is zero for given $h(n)$. And let n_{hh} be the highest index in $h(n)$; which is 1 for given sequence of $h(n)$.



Then lowest index of 'n' for $y(n)$ is given as,

$$n_{yl} = n_{xl} + n_{hl} = 0 + 0 = 0 \quad \dots (2.7.21)$$

And highest index of 'n' for $y(n)$ is given as,

$$n_{yh} = n_{xh} + n_{hh} = 3 + 1 = 4 \quad \dots (2.7.22)$$

Thus the range of n for $y(n)$ will be $0 \leq n \leq 4$.

$n = 0$ in equation 2.7.20 gives $y(0)$:

$$\begin{aligned}
 y(0) &= \sum_{k=0}^3 x(k) h(-k) \\
 &= x(0) h(0) + x(1) h(-1) + x(2) h(-2) + x(3) h(-3) \\
 &= (1 \times 2) + 0 + 0 + 0 \quad \text{since } h(-1) = h(-2) = h(-3) = 0 \\
 &= 2 \quad \boxed{y(0) = 2}
 \end{aligned}$$

$n = 1$ in equation 2.7.20 gives $y(1)$:

$$\begin{aligned}
 y(1) &= \sum_{k=0}^3 x(k) h(1-k) \\
 &= x(0) h(1) + x(1) h(0) + x(2) h(-1) + x(3) h(-2) \\
 &= (1 \times 2) + (1 \times 2) + 0 + 0 \quad \text{since } h(-1) = h(-2) = 0 \\
 &= 2 + 2 = 4 \quad \boxed{y(1) = 4}
 \end{aligned}$$

$n = 2$ in equation 2.7.20 gives $y(2)$:

$$\begin{aligned}
 y(2) &= \sum_{k=0}^3 x(k) h(2-k) \\
 &= x(0) h(2) + x(1) h(1) + x(2) h(0) + x(3) h(-1) \\
 &= 0 + (1 \times 2) + (1 \times 2) + 0 \quad \text{since } h(2) = h(-1) = 0 \\
 &= 4 \quad \boxed{y(2) = 4}
 \end{aligned}$$

$n = 3$ in equation 2.7.20 gives $y(3)$:

$$\begin{aligned} y(3) &= \sum_{k=0}^3 x(k) h(3-k) \\ &= x(0)h(3) + x(1)h(2) + x(2)h(1) + x(3)h(0) \\ &= 0 + 0 + (1 \times 2) + (1 \times 2) \text{ since } h(3) = h(2) = 0 \\ &= 4 \end{aligned}$$

$$y(3) = 4$$

$n = 4$ in equation 2.7.20 gives $y(4)$:

$$\begin{aligned} y(4) &= \sum_{k=0}^3 x(k) h(4-k) \\ &= x(0)h(4) + x(1)h(3) + x(2)h(2) + x(3)h(1) \\ &= 0 + 0 + 0 + (1 \times 2) \text{ since } h(4) = h(3) = h(2) = 0 \\ &= 2 \end{aligned}$$

$$y(4) = 2$$

Thus the calculated sequence due to convolution of $x(n)$ and $h(n)$ is as follows :

$$y(n) = \{2, 4, 4, 4, 2\}$$

Observe that this sequence is similar to the one calculated in example 2.7.1.

Comments :

- The range of 'n' in $y(n)$ is given as,
Lowest index = sum of lowest indices of sequences to be convolved.
and Highest index = sum of highest indices of sequences to be convolved.
- The given equation for $y(n)$ as convolution of $x(n)$ and $h(n)$ is,

$$y(n) = \sum_{k=-\infty}^{\infty} x(k) h(n-k) \quad \dots (2.7.23)$$

If lowest index for $x(k)$ is n_{x_l} and highest index is n_{x_h} . Then,

$$x(k) = 0 \text{ for } k < n_{x_l} \text{ and } k > n_{x_h}$$

Hence the product $x(k)h(n-k)$ becomes zero for $k < n_{x_l}$ and $k > n_{x_h}$. Hence limits of 'k' in equation 2.7.23 can be changed as $n_{x_l} \leq k \leq n_{x_h}$ i.e.,

$$y(n) = \sum_{k=n_{x_l}}^{n_{x_h}} x(k) h(n-k) \quad \dots (2.7.24)$$

$$\text{and } (n_{x_l} + n_{h_l}) \leq n \leq (n_{x_h} + n_{h_h})$$

This equation is then easily computable.

2.7.3 Properties of Convolution

In the last subsection we discussed convolution with the help of examples. Here we will discuss some of the important properties of convolution.

1. Commutative property of convolution :

This property states that convolution is commutative operation. The linear convolution is given by equation 2.7.15 as,

$$y(n) = \sum_{k=-\infty}^{\infty} x(k) h(n-k) \quad \dots (2.7.25)$$

Let us define the new index of summation as,

$$m = n - k \text{ and hence } k = n - m.$$

The limits of m will be same as k , i.e. $(-\infty, \infty)$. Hence above equation becomes,

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

Here observe that ' m ' is the dummy index and it can be replaced by any character, the meaning remains same. Hence replacing ' m ' by ' k ' in the above equation we get,

$$y(n) = \sum_{k=-\infty}^{\infty} x(n-k) h(k) \quad \dots (2.7.26)$$

This equation is alternate form of convolution given by equation 2.7.25. In the above equation $x(k)$ is folded and shifted, where as impulse response $h(k)$ is unaltered. Equation 2.7.25 and equation 2.7.26 are the two equations for convolution representing the same output. i.e.,

$$y(n) = \sum_{k=-\infty}^{\infty} x(k) h(n-k) = x(n) * h(n) \quad \dots (2.7.27)$$

and

$$y(n) = \sum_{k=-\infty}^{\infty} x(n-k) h(k) = h(n) * x(n) \quad \dots (2.7.28)$$

Here we have used asterisk (*) to represent convolution of two sequence. It is standard symbolic notation. From the above two equations of convolution it is clear that,

$$x(n) * h(n) = h(n) * x(n) = y(n) \quad \dots (2.7.29)$$

Hence it is proved that linear convolution is commutative. Fig. 2.7.8 illustrates this commutative property.

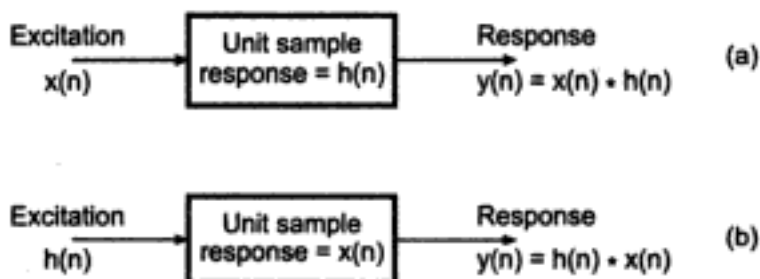


Fig. 2.7.8 Illustration of commutative property of linear convolution

2. Associative property of convolution :

The linear convolution has associative property which can be stated as :

$$[x(n) * h_1(n)] * h_2(n) = x(n) * [h_1(n) * h_2(n)] \quad \dots (2.7.30)$$

Consider the left hand side of the above equation and let,

$$y_1(n) = x(n) * h_1(n) \quad \dots (2.7.31)$$

This means $x(n)$ is applied to the system whose unit sample response is $h_1(n)$ and output is denoted as $y_1(n)$. This output is applied to another system whose unit sample response is $h_2(n)$. Let the output of this system be $y(n)$. i.e.,

$$y(n) = y_1(n) * h_2(n) \quad \dots (2.7.32)$$

Putting for $y_1(n)$ from equation 2.7.31 in above equation,

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

This is the left hand side of equation 2.7.30. Such operation is illustrated in Fig. 2.7.9.

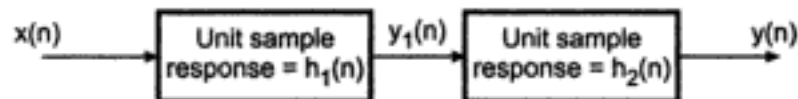


Fig. 2.7.9 Two linear shift invariant systems connected in cascade

Thus the left hand side of equation 2.7.30 equivalent to connecting two LTI systems in cascade. The right hand side of equation 2.7.30 is,

$$\begin{aligned} y(n) &= x(n) * [h_1(n) * h_2(n)] \\ &= x(n) * h(n) \end{aligned}$$

Here $h(n) = h_1(n) * h_2(n) \quad \dots (2.7.33)$

This means $x(n)$ is applied to the system whose unit sample response is the convolution of two unit sample responses $h_1(n)$ and $h_2(n)$. Such system is indicated in Fig. 2.7.10. Thus the unit sample response $h(n)$ of this system is equivalent to convolution of $h_1(n)$ and $h_2(n)$. And the operation performed by systems in Fig. 2.7.9 and Fig. 2.7.10 is same. This indicates that convolution is associative.

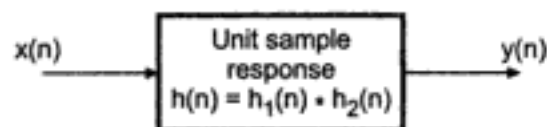


Fig. 2.7.10 The cascade connection of two systems can be combined into one system and overall unit sample response is equal to convolution of individual unit sample responses

If there are 'L' number of LTI systems connected in cascade with unit sample responses of $h_1(n), h_2(n), h_3(n) \dots h_L(n)$; then the overall unit sample response is,

$$h(n) = h_1(n) * h_2(n) * h_3(n) \dots * h_L(n) \quad \dots (2.7.34)$$

This also indicates that a LTI system can be decomposed into cascade connection of systems and their unit sample responses are related by above equation.

3. Distributive property of convolution :

The linear convolution satisfies distributive property, which is stated as,

$$x(n) * [h_1(n) + h_2(n)] = x(n) * h_1(n) + x(n) * h_2(n) \quad \dots (2.7.35)$$

Consider the right handside of above equation, in which let,

$$y_1(n) = x(n) * h_1(n)$$

and

$$y_2(n) = x(n) * h_2(n)$$

And right handside of equation 2.7.35 is equal to $y_1(n) + y_2(n)$. This means there are two LTI systems excited by same input $x(n)$ and their outputs are added. This operation is illustrated in Fig. 2.7.11 (a).

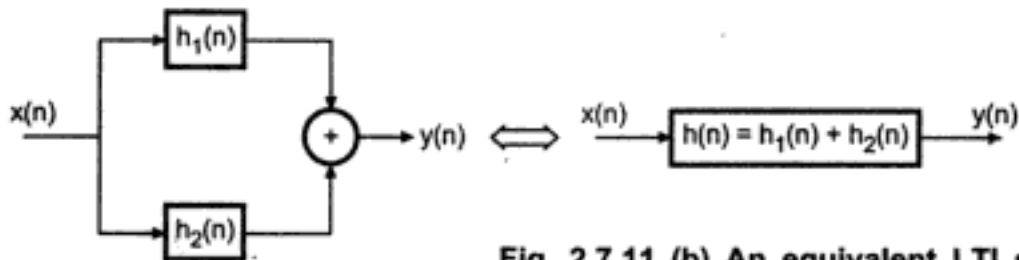


Fig. 2.7.11 (a) Two individual LTI systems applied by same input. representing RHS of equation 2.7.35

Fig. 2.7.11 (b) An equivalent LTI system of Fig. 2.7.11 (a). For parallel connection of Fig. 2.7.11(a), $h_1(n)$ and $h_2(n)$ are added in this system.

Observe that it is the parallel connection of two systems. The unit sample response of the overall system is equal to sum of $h_1(n)$ and $h_2(n)$ i.e.

$$h(n) = h_1(n) + h_2(n) \quad \dots (2.7.36)$$

Hence output of such system is,

$$\begin{aligned} y(n) &= x(n) * h(n) \\ &= x(n) * [h_1(n) + h_2(n)] \text{ putting } h(n) \text{ from equation 2.7.36} \end{aligned}$$

This is the left hand side of equation 2.7.35. Thus the overall unit sample response is obtained by adding individual unit sample responses. Fig. 2.7.11 (b) illustrates this operation. Thus it is possible to decompose a complex system into parallel connection of small and simplified systems, i.e.,

$$\begin{aligned} h(n) &= h_1(n) + h_2(n) + h_3(n) + \dots + h_L(n) \\ &= \sum_{j=1}^L h_j(n) \quad \dots (2.7.37) \end{aligned}$$

Here $h(n)$ is the unit sample response of complex system and $h_1(n), h_2(n), h_3(n), \dots, h_L(n)$ are the unit sample responses of individual parallel connected subsystems.

2.7.4 Causality of LTI Systems

Earlier we have defined causality property of the systems. The output of the causal system depends only upon the present and past inputs. For example, the output of the causal system at $n = n_0$ depends only upon inputs $x(n)$ for $n \leq n_0$. This condition for causality can be expressed in terms of unit sample response $h(n)$ for the LTI systems. The output $y(n)$ is given as convolution of $h(n)$ and $x(n)$ for LTI systems.

$$y(n) = \sum_{k=-\infty}^{\infty} h(k) x(n-k) \text{ from equation 2.7.28}$$

At $n = n_0$, the output $y(n_0)$ will be,

$$y(n_0) = \sum_{k=-\infty}^{\infty} h(k) x(n_0 - k) \quad \dots (2.7.38)$$

Let us rearrange the $\sum_{k=-\infty}^{\infty}$ in the above equation for the values of $k \geq 0$ and $k < 0$ in two separate terms as follows :

$$y(n_0) = \sum_{k=0}^{\infty} h(k)x(n_0 - k) + \sum_{k=-\infty}^{-1} h(k)x(n_0 - k)$$

Let us expand the terms of summation in the above equation,

$$y(n_0) = [h(0)x(n_0) + h(1)x(n_0 - 1) + h(2)x(n_0 - 2) + \dots] \\ + [h(-1)x(n_0 + 1) + h(-2)x(n_0 + 2) + h(-3)x(n_0 + 3) + \dots]$$

Here $x(n_0)$ is present input and $x(n_0 - 1), x(n_0 - 2), \dots$ etc are past inputs. And $x(n_0 + 1), x(n_0 + 2), x(n_0 + 3), \dots$ etc are the future inputs. We know that the output of causal system at $n = n_0$ depends upon the inputs for $n \leq n_0$. Hence for causality,

$$h(-1) = h(-2) = h(-3) \dots = 0$$

This is because $x(n_0 + 1), x(n_0 + 2), \dots$ etc need not be zero compulsorily, since they are inputs. But $h(n)$ is the unit sample response of the system. In other words, it is the characteristic of the system. Hence for the system to be causal,

$$h(n) = 0 \quad \text{for } n < 0 \quad \text{for causal system}$$

We know that, $h(n) = T[\delta(n)]$

That is, $h(n)$ is the unit sample response of the system for unit sample $\delta(n)$ applied at $n = 0$. Hence for the system to be causal, $h(n) = 0$ for $n < 0$. Thus this condition is two fold for causality. Hence we can state,

A LTI system is causal if and only if

$$h(n) = 0 \quad \text{for } n < 0 \quad \dots (2.7.39)$$

This is the necessary and sufficient condition for causality of the system.

The convolution of $x(n)$ and $h(n)$ is given as,

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

For the causal LTI system, $h(n) = 0$ for $n < 0$. Hence $h(k) = 0$ for $k < 0$ in the above equation. This is just the change of index from 'n' to 'k'. Hence the convolution equation for causal LTI system becomes,

$$y(n) = \sum_{k=0}^{\infty} h(k)x(n - k) \quad \dots (2.7.40)$$

Here let us change the index as,

$$m = n - k \quad \text{hence } k = n - m$$

when $k = 0,$ $m = n$

and when $k = \infty,$ $m = n - \infty = -\infty$

Hence equation 2.7.40 becomes,

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

This equation can also be written simply as,

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

In the above equation replacing the index 'm' by 'k' does not change the meaning. Here we are doing it for consistent notations. Hence above equation becomes,

$$y(n) = \sum_{k=-\infty}^n x(k)h(n-k) \quad \dots (2.7.41)$$

When the sequence $x(n) = 0$. For $n < 0$, then it is called causal sequence. Hence $x(k) = 0$ for $k < 0$. Hence above equation becomes,

$$y(n) = \sum_{k=0}^n x(k)h(n-k) \quad \dots (2.7.42)$$

This is the equation for linear convolution. It gives output of the causal LTI system for causal input sequence. The causality of LTI system is imposed by unit sample response $h(n)$ and input sequence $x(n)$ is also causal. Hence the limits of summation in \sum are changed from $(-\infty, \infty)$ to $(0, n)$. In the above equation $y(n) = 0$ for $n < 0$, hence the response of causal LTI system is also causal for causal input sequence.

2.7.5 Stability of LTI Systems

We have defined the stability criteria for systems earlier. The system is said to be stable if it produces bounded output for every bounded input. In this section we will derive the stability criteria for Linear Time Invariant (LTI) systems in terms of their unit sample response. The linear convolution is given as,

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

Taking the absolute value of both the sides,

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

The absolute value of the total sum is always less than or equal to sum of the absolute values of individual terms. Hence right hand side of the above equation can be written as,

$$|y(n)| \leq \sum_{k=-\infty}^{\infty} |h(k)||x(n-k)| \quad \dots (2.7.43)$$

If the input sequence $x(n)$ is bounded, then there exists a finite number M_x , such that

$$|x(n)| \leq M_x < \infty \quad \dots (2.7.44)$$

Putting this condition for bounded input in equation 2.7.43 we get,

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

Here M_x is the finite number. Then for the $|y(n)|$ to be finite in the above equation, the condition is,

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

With this condition, the sum of impulse response is finite and hence the output $|y(n)|$ is also finite. Thus bounded input $x(n)$ produces bounded output $y(n)$ in the LTI system only if,

$$\sum_{k=-\infty}^{\infty} |h(k)| < \infty \quad \dots (2.7.45)$$

When this condition is satisfied, the system will be stable. The above condition states that the LTI system is stable if its unit sample response is absolutely summable. This is the necessary and sufficient condition for the stability of LTI system.

Ex.2.7.3 Determine the range of values 'a' and 'b', for which the LTI system with impulse response

$$h(n) = \begin{cases} a^n, & n \geq 0 \\ b^n, & n < 0 \end{cases} \quad \dots (2.7.46)$$

is stable.

[Dec - 99]

Sol. : The condition for stability is given by equation 2.7.45 as,

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

Splitting the summation according to equation 2.7.46 and putting for $h(n)$ in above equation we get,

$$\sum_{n=-\infty}^{\infty} |h(n)| = \sum_{n=0}^{\infty} |a^n| + \sum_{n=-\infty}^{-1} |b^n| \quad \dots (2.7.47)$$

Let us consider the summation terms in above equation separately. The first summation can be written as,

$$\sum_{n=0}^{\infty} |a^n| = 1 + |a| + |a^2| + |a^3| + \dots$$

This is a standard geometric series and it converges to $\frac{1}{1-|a|}$ if $|a| < 1$. If $|a| > 1$, the series does not converge and it becomes unstable. Thus,

$$\sum_{n=0}^{\infty} |a^n| = \frac{1}{1-|a|}, \text{ if } |a| < 1 \quad \dots (2.7.48)$$

Now let us consider the second summation in equation 2.7.47. It can be rearranged as follows :

$$\begin{aligned} \sum_{n=-\infty}^{-1} |b^n| &= \sum_{n=1}^{\infty} \frac{1}{|b^n|} \\ &= \frac{1}{|b|} + \frac{1}{|b^2|} + \frac{1}{|b^3|} + \frac{1}{|b^4|} + \dots \\ &= \frac{1}{|b|} \left[1 + \frac{1}{|b|} + \frac{1}{|b^2|} + \dots \right] \quad \dots (2.7.49) \end{aligned}$$

The part inside square brackets in above equation is the geometric series and it converges to, $\frac{1}{1-\frac{1}{|b|}}$, if $\frac{1}{|b|} < 1$ i.e. $|b| > 1$. Hence equation 2.7.49 becomes,

$$\begin{aligned}\sum_{n=-\infty}^{-1} |b^n| &= \frac{1}{|b|} \cdot \frac{1}{1 - \frac{1}{|b|}}, \text{ if } |b| > 1 \\ &= \frac{1}{|b|-1}, \text{ if } |b| > 1\end{aligned}\quad \dots (2.7.50)$$

Putting the values of summations obtained in the above equation and equation 2.7.48 in equation 2.7.47 we get,

$$\sum_{n=-\infty}^{\infty} |h(n)| = \frac{1}{1-|a|} + \frac{1}{|b|-1}, \text{ if } |a| < 1 < |b|$$

Thus the geometric series converges if $|a| < 1 < |b|$. Thus the system will be stable if $|a| < 1 < |b|$.

Ex.2.7.4 Determine the range of values of the parameter 'a' for which the linear time invariant system with impulse response $h(n)$ is stable.

$$\begin{aligned}h(n) &= a^n, \quad n \geq 0 \text{ and } n \text{ even} \\ &= 0 \quad \text{otherwise}\end{aligned}$$

[Dec - 99]

Sol. : The condition for stability is given by equation 2.7.45 as,

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

Putting the given value of $h(n)$ in above equation,

$$\sum_{n=-\infty}^{\infty} |h(n)| = \sum_{\substack{n=0 \\ n \text{ even}}}^{\infty} |a^n|$$

Here 'n' takes only even values. Let us put $n = 2m$. Hence m takes values of 0, 1, 2, 3, ... ∞ and 'n' will take values of 0, 2, 4, 6, ... ∞ . Then above equation becomes,

$$\begin{aligned}\sum_{n=-\infty}^{\infty} |h(n)| &= \sum_{m=0}^{\infty} |a^m| \\ &= \sum_{m=0}^{\infty} |a|^m \\ &= 1 + |a| + |a|^2 + |a|^3 + \dots\end{aligned}$$

This is a geometric series and it converges to $\frac{1}{1-|a|}$ if $|a| < 1$. If the series does not converges, then the system becomes unstable. Thus the given system is stable if $|a| < 1$.

Ex.2.7.5 The convolution of $x(n)$ and $h(n)$ is given as,

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

Then show that

$$\sum_{n=-\infty}^{\infty} y(n) = \sum_{n=-\infty}^{\infty} x(n) \cdot \sum_{n=-\infty}^{\infty} h(n) \quad \dots (2.7.51)$$

Sol. : The linear convolution of $x(n)$ and $h(n)$ is given by equation 2.7.15 as,

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

Let us take summation of $y(n)$ from $-\infty$ to $+\infty$ i.e.,

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

Let $n-k=m$ in second summation term in above equation. Since limits for ' n ' are $(-\infty, \infty)$, the limits for ' m ' will also be $(-\infty, \infty)$. Hence above equation becomes,

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

In the above equation ' k ' and ' m ' are just indices and any alphabet like ' n ' can be used. Thus equation 2.7.51 is proved. The above equation indicates that the sum of all values of $y(n)$ is equal to product of sums of $x(n)$ and $h(n)$. Hence equation 2.7.51 can also be written as follows :

$$\sum y = \sum x \sum h \quad \dots (2.7.52)$$

Here observe that indices are avoided.

To verify equation 2.7.52 for the result obtained in example 2.7.2 :

Refer to example 2.7.2. In this example $x(n)$ and $h(n)$ are given as,

$$x(n) = \{1, 1, 1, 1\}$$

$$h(n) = \{2, 2\}$$

And $y(n)$ is calculated in example 2.7.2 as,

$$y(n) = \{2, 4, 4, 4, 2\}$$

$$\therefore \sum x(n) = 1+1+1+1=4$$

$$\sum h(n) = 2+2=4$$

$$\text{and } \sum y(n) = 2+4+4+4+2=16$$

According to equation 2.7.52,

$$\begin{aligned} \sum y &= \sum x \cdot \sum h = 4 \times 4 \\ &= 16 \end{aligned}$$

Which is same as calculated above. Thus this equation can be used as a 'check' to verify the results of convolution.

Ex.2.7.6 Easy method to compute convolution

The sequences $x(n)$ and $h(n)$ are given as follows :

$$x(n) = \{1, 1, 0, 1, 1\}$$

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

Compute the convolution of these two sequences

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

Sol. : This method is relatively easy and is based on the technique similar to multiplication. The two sequences are multiplied as we multiply multiple digit numbers. The result of this multiplication is nothing but the convolution of two sequences. The complete procedure is illustrated below in Fig. 2.7.12.

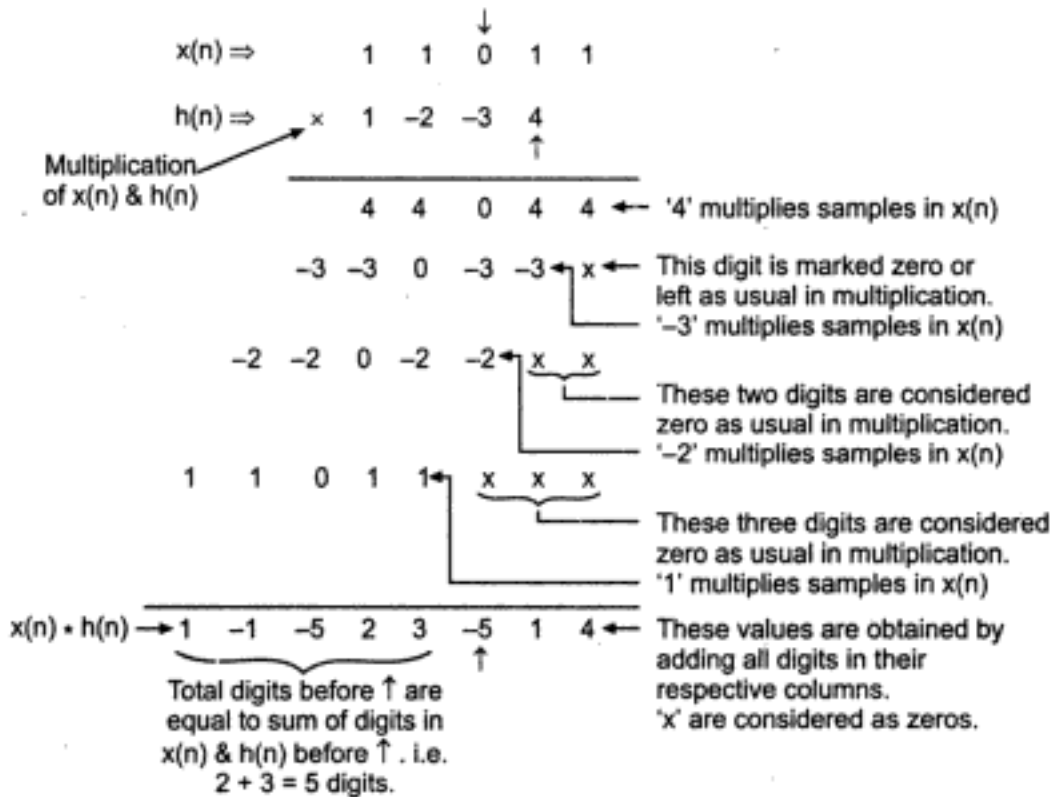


Fig. 2.7.12 Computation of convolution using multiplication

In the sequence $x(n)$ observe that there are '2' digits before the zero mark \uparrow arrow. Similarly there are '3' digits before the zero mark \uparrow arrow in $h(n)$. Hence there will be $2+3=5$ digits before the zero mark \uparrow arrow in $y(n)$. Thus the result of convolution is obtained in Fig. 2.7.12 as,

$$y(n) = \{1, -1, -5, 2, 3, -5, 1, 4\} \quad \dots (2.7.53)$$

To verify the result obtained.

The result obtained in this example can be verified by using equation 2.7.52 and also by using basic convolution equation.

(i) To verify result by equation 2.7.52 :

Equation 2.7.52 states that,

$$\sum y = \sum x \cdot \sum h \quad \dots (2.7.54)$$

From equation 2.7.53 we have $y(n)$ as,

$$y(n) = \{1, -1, -5, 2, 3, -5, 1, 4\}$$

$$\therefore \sum y = 1 - 1 - 5 + 2 + 3 - 5 + 1 + 4 = 0 \quad \dots (2.7.55)$$

$x(n)$ and $h(n)$ are given as,

$$x(n) = \{1, 1, 0, 1, 1\}$$

$$\therefore \sum x = 1+1+0+1+1=4$$

$$\text{and } h(n) = \{1, -2, -3, 4\}$$

$$\therefore \sum h = 1-2-3+4=0$$

Putting values in equation 2.7.54 we have,

$$\begin{aligned} \sum y &= \sum x \cdot \sum h \\ &= 4 \times 0 \end{aligned}$$

= 0 which is same as that obtained in equation 2.7.55

Thus $\sum y = \sum x \cdot \sum h$ is proved.

(ii) To verify result using $y(n) = \sum_{k=-\infty}^{\infty} x(k) \cdot h(n-k)$:

Here let us compute the convolution using the basic convolution equation. The given sequences are :

$$x(n) = \{1, 1, 0, 1, 1\}$$

↑

and

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

↑

These two sequences can also be written as,

$$\begin{array}{ll} x(-2)=1 & h(-3)=1 \\ x(-1)=1 & h(-2)=-2 \\ x(0)=0 \leftarrow & h(-1)=-3 \\ x(1)=1 & h(0)=4 \leftarrow \\ x(2)=1 & \end{array}$$

From above we can write the following :

$$\text{Lowest index of } x(n) \Rightarrow n_{x_l} = -2$$

$$\text{Highest index of } x(n) \Rightarrow n_{x_h} = 2$$

$$\text{Lowest index of } h(n) \Rightarrow n_{h_l} = -3$$

and Highest index of $h(n) \Rightarrow n_{h_h} = 0$

The convolution equation is given by equation 2.7.24 as,

$$y(n) = \sum_{k=n_{x_l}}^{n_{x_h}} x(k) \cdot h(n-k) \text{ and } (n_{x_l} + n_{h_l}) \leq n \leq (n_{x_h} + n_{h_h}) \quad \dots (2.7.56)$$

Putting values of highest and lowest indices in above equation we get,

$$y(n) = \sum_{k=-2}^2 x(k) h(n-k) \text{ and } (-2-3) \leq n \leq (2+0)$$

$$\text{i.e. } y(n) = \sum_{k=-2}^2 x(k) h(n-k) \text{ and } -5 \leq n \leq 2 \quad \dots (2.7.57)$$

Now let us compute $y(n)$ for the complete range.

$n = -5$ in equation 2.7.57 gives $y(-5)$:

$$\begin{aligned} y(-5) &= \sum_{k=-2}^2 x(k)h(-5-k) \\ &= x(-2)h(-3) + x(-1)h(-4) + x(0)h(-5) + x(1)h(-6) + x(2)h(-7) \\ &= (1 \times 1) + (1 \times 0) + (0 \times 0) + (1 \times 0) + (1 \times 0) \\ &\quad \text{Since } h(-4) = h(-5) = h(-6) = h(-7) = 0 \\ &= 1 + 0 + 0 + 0 + 0 \\ &= 1 \end{aligned}$$

$$y(-5) = 1$$

$n = -4$ in equation 2.7.57 gives $y(-4)$:

$$\begin{aligned} y(-4) &= \sum_{k=-2}^2 x(k)h(-4-k) \\ &= x(-2)h(-2) + x(-1)h(-3) + x(0)h(-4) + x(1)h(-5) + x(2)h(-6) \\ &= (1 \times -2) + (1 \times 1) + 0 + 0 + 0, \quad \text{since } h(-4) = h(-5) = h(-6) = 0 \\ &= -2 + 1 + 0 + 0 + 0 \\ &= -1 \end{aligned}$$

$$y(-4) = -1$$

$n = -3$ in equation 2.7.57 gives $y(-3)$:

$$\begin{aligned} y(-3) &= \sum_{k=-2}^2 x(k)h(-3-k) \\ &= x(-2)h(-1) + x(-1)h(-2) + x(0)h(-3) + x(1)h(-4) + x(2)h(-5) \\ &= (1 \times -3) + (1 \times -2) + (0 \times 1) + 0 + 0, \quad \text{since } h(-n) = h(-5) = 0 \\ &= -3 - 2 + 0 + 0 + 0 \\ &= -5 \end{aligned}$$

$$y(-3) = -5$$

$n = -2$ in equation 2.7.57 gives $y(-2)$:

$$\begin{aligned} y(-2) &= \sum_{k=-2}^2 x(k)h(-2-k) \\ &= x(-2)h(0) + x(-1)h(-1) + x(0)h(-2) + x(1)h(-3) + x(2)h(-4) \\ &= (1 \times 4) + (1 \times -3) + (0 \times -2) + (1 \times 1) + 0, \quad \text{since } h(-4) = 0 \\ &= 4 - 3 + 0 + 1 + 0 \\ &= 2 \end{aligned}$$

$$y(-2) = 2$$

$n = -1$ in equation 2.7.57 gives $y(-1)$:

$$\begin{aligned} y(-1) &= \sum_{k=-2}^2 x(k)h(-1-k) \\ &= x(-2)h(1) + x(-1)h(0) + x(0)h(-1) + x(1)h(-2) + x(2)h(-3) \\ &= 0 + (1 \times 4) + (0 \times -3) + (1 \times -2) + (1 \times 1), \quad \text{since } h(1) = 0 \\ &= 0 + 4 + 0 - 2 + 1 \\ &= 3 \end{aligned}$$

$$y(-1) = 3$$

$n = 0$ in equation 2.7.57 gives $y(0)$:

$$\begin{aligned} y(0) &= \sum_{k=-2}^2 x(k)h(-k) \\ &= x(-2)h(2) + x(-1)h(1) + x(0)h(0) + x(1)h(-1) + x(2)h(-2) \\ &= 0 + 0 + (0 \times 4) + (1 \times -3) + (1 \times -2), \text{ since } h(2) = h(1) = 0 \\ &= 0 + 0 + 0 - 3 - 2 \\ &= -5 \end{aligned}$$

$$y(0) = -5$$

$n = 1$ in equation 2.7.57 gives $y(1)$:

$$\begin{aligned} y(1) &= \sum_{k=-2}^2 x(k)h(1-k) \\ &= x(-2)h(3) + x(-1)h(2) + x(0)h(1) + x(1)h(0) + x(2)h(-1) \\ &= 0 + 0 + 0 + (1 \times 4) + (1 \times -3), \text{ since } h(3) = h(2) = h(1) = 0 \\ &= 0 + 0 + 0 + 4 - 3 \\ &= 1 \end{aligned}$$

$$y(1) = 1$$

$n = 2$ in equation 2.7.57 gives $y(2)$:

$$\begin{aligned} y(2) &= \sum_{k=-2}^2 x(k)h(2-k) \\ &= x(-2)h(4) + x(-1)h(3) + x(0)h(2) + x(1)h(1) + x(2)h(0) \\ &= 0 + 0 + 0 + 0 + (1 \times 4), \text{ since } h(4) = h(3) = h(2) = h(1) = 0 \\ &= 0 + 0 + 0 + 0 + 4 \\ &= 4 \end{aligned}$$

$$y(2) = 4$$

Thus the sequence $y(n)$ obtained using basic computations of convolution is as follows :

$$y(n) = \{1, -1, -5, 2, 3, -5, 1, 4\} \quad \dots (2.7.58)$$

The sequence $y(n)$ given above is exactly same as that obtained in equation 2.7.53. Thus the results are verified.

Ex.2.7.7 Another easy method to compute convolution.

This example illustrates another easy method to compute convolution.

$$x(n) = \{1, 1, 0, 1, 1\} \text{ and } h(n) = \{1, -2, -3, 4\}$$

of example 2.7.6 are considered. Sometimes this method is also called tabulation method.

Sol. : The values of $x(n)$ and $h(n)$ can be written as follows :

$$\begin{array}{rcl} x(-2) = 1 & & h(-3) = 1 \\ x(-1) = 1 & & h(-2) = -2 \\ x(0) = 0 & \leftarrow & h(-1) = -3 \\ x(1) = 1 & & h(0) = 4 \leftarrow \\ x(2) = 1 & & \end{array}$$

The above values of $x(n]$ and $h(n]$ are tabulated as shown below in Fig. 2.7.13.

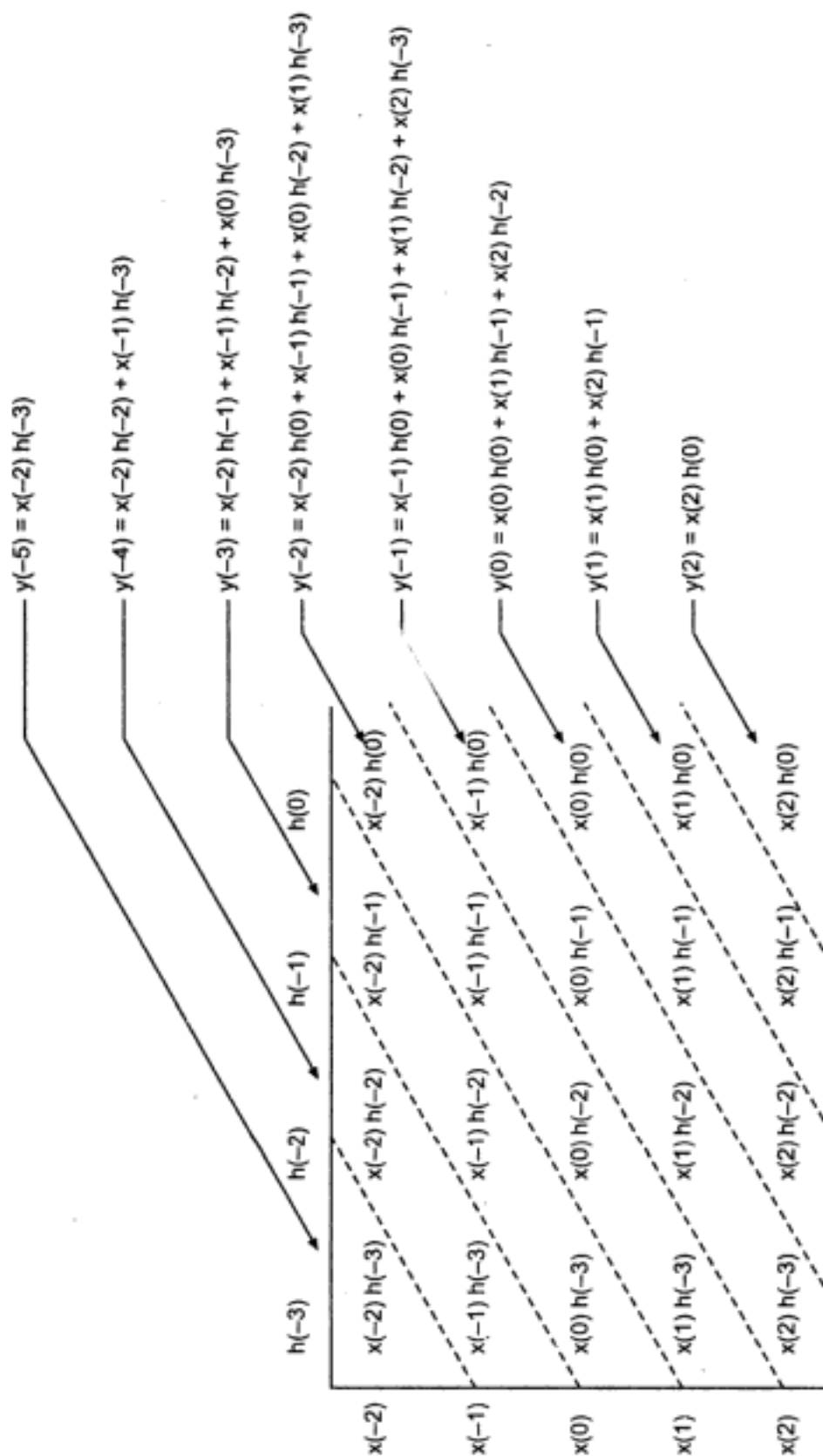


Fig. 2.7.13 Computation of convolution using tabulation

Thus as shown in above figure $h(-3), h(-2), h(-1)$ and $h(0)$, form the columns of table. And $x(-2), x(-1), x(0), x(1), x(2)$ form the rows of the table. In the table the multiplications of $x(n)$ and $h(n)$ are written as shown. Then the multiplications are separated diagonally as shown by dotted lines in Fig. 2.7.13.

From given sequences $x(n)$ and $h(n)$ we have,

lowest index of $x(n) \Rightarrow n_{x1} = -2$

lowest index of $h(n) \Rightarrow n_{h1} = -3$

Hence lowest index of $y(n) \Rightarrow n_{y1} = n_{x1} + n_{h1} = -2 - 3$

$\therefore n_{y1} = -5$

Thus first element in $y(n)$ will be $y(-5)$. This element is equal to top left diagonal array. It contains only one multiplication. i.e.,

$y(-5) = x(-2)h(-3)$

The other diagonal arrays are successively $y(-4), y(-3), y(-2)$ as shown in Fig. 2.7.13. Finally the last element in the array is $y(2)$ and it is the bottom right element in table. i.e.,

$y(2) = x(2)h(0)$

Let us put values in table of Fig. 2.7.13. Such table is shown below in Fig. 2.7.14.

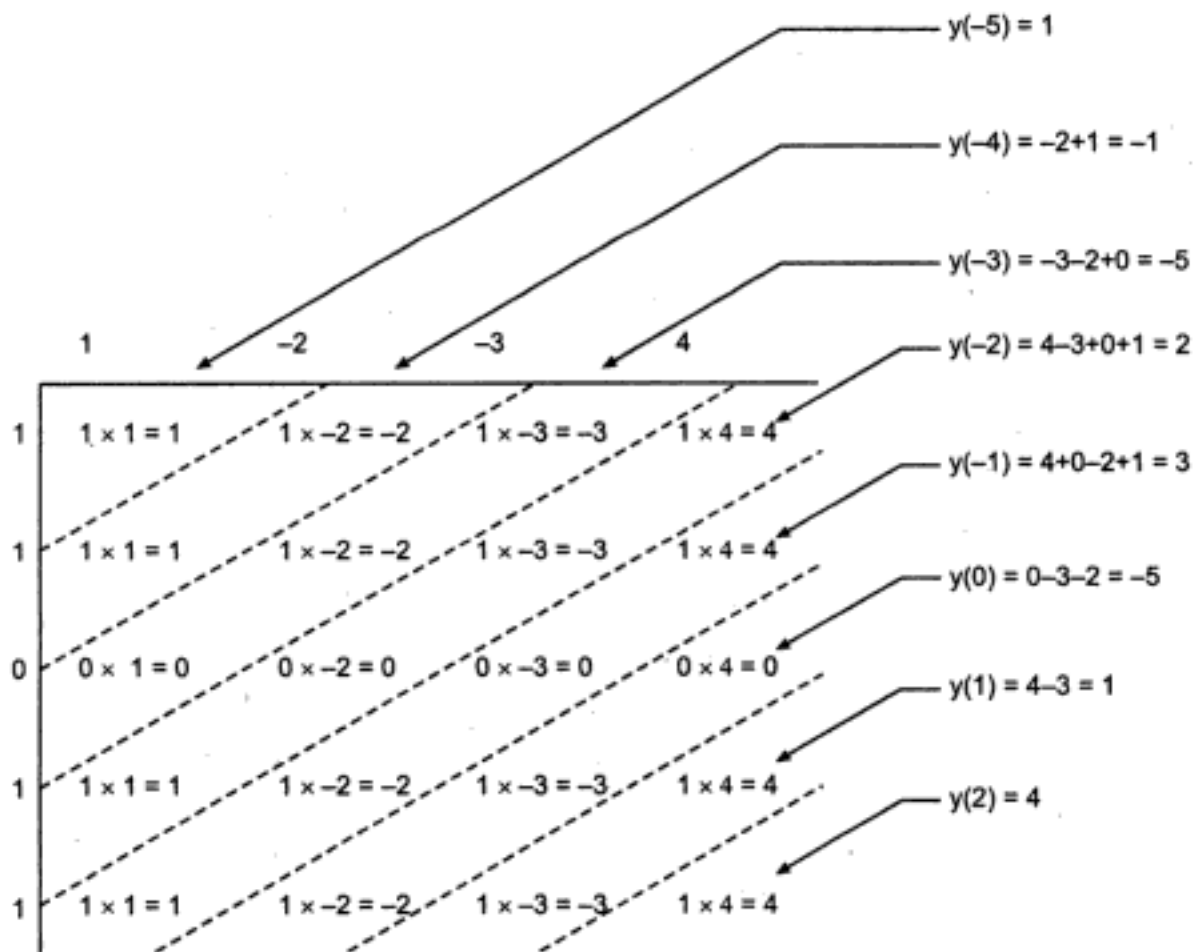


Fig. 2.7.14 Computation of convolution

Thus the computed sequence is,

$$y(n) = \{1, -1, -5, 2, 3, -5, 1, 4\}$$

↑

This sequence is exactly similar to that calculated in last example given by equation 2.7.58. Thus tabulation method and multiplication methods are easier methods for computation of convolution.

Comments :

1. Convolution can be computed easily by tabulation and multiplication methods.
2. If $y(n) = x(n) * h(n)$, then result of convolution satisfies $\sum y = \sum x \cdot \sum h$. This equation can be used to check correctness of the result.
3. After equation 2.7.57, $y(-5), y(-4), \dots$ etc are calculated using basic convolution equation. Observe tabulation of multiplications and equations for $y(-5), y(-4), y(-3), \dots$ etc in Fig. 2.7.13 carefully. It shows that multiplications and summations are arranged in specific format in the table. This makes the computations easy.
4. Observe the multiplication method in Fig. 2.7.12 carefully. Here the summations are actually summations in basic convolution computation.

Thus tabulation and multiplication techniques are derived from basic convolution method only. They provide easier and faster computations of convolution.

Ex.2.7.8 Prove and explain graphically the difference between the relations :

$$x(n) \delta(n - n_0) = x(n_0)$$

$$\text{and } x(n) * \delta(n - n_0) = x(n - n_0)$$

[Dec - 99]

Sol. : (i) Interpretation of $x(n) \delta(n - n_0) = x(n_0)$:

The discussion of this expression is given in section 2.7.1. Let us consider some sequence for $x(n)$. i.e.,

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

↑

... (2.7.59)

Here,

$$\begin{aligned} x(-2) &= 2 \\ x(-1) &= 1 \\ x(0) &= 3 \\ x(1) &= -2 \\ x(2) &= 2 \end{aligned}$$

This signal is shown graphically in Fig. 2.7.15 (a).

We know that,

$$\left. \begin{aligned} \delta(n) &= 1 \text{ for } n = 0 \\ &= 0 \text{ for } n \neq 0 \end{aligned} \right\} \text{By definition}$$

This is a unit sample sequence. We have shown in equation 2.7.3 that the $\delta(n)$ function delayed by n_0 samples is given as,

$$\left. \begin{aligned} \delta(n - n_0) &= 1 \text{ for } n = n_0 \\ &= 0 \text{ for } n \neq n_0 \end{aligned} \right\} \dots (2.7.60)$$

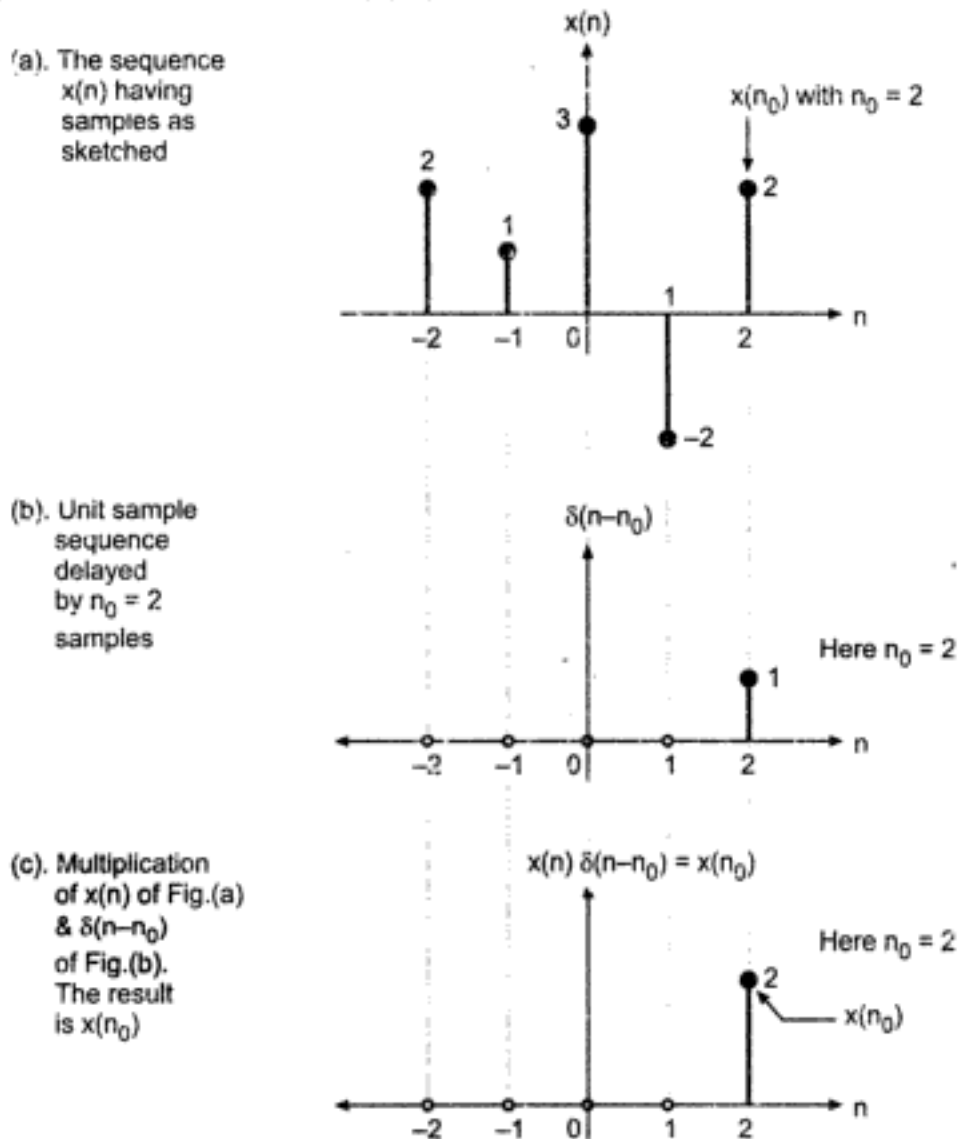


Fig. 2.7.15 Illustration of $x(n) \delta(n - n_0) = x(n_0)$

The sketch of $\delta(n - n_0)$ for $n_0 = 2$ samples is shown in Fig. 2.7.15 (b). When we multiply $x(n)$ and $\delta(n - n_0)$ we get all samples of zero values except at $n = n_0$. Thus,

$$x(n) \cdot \delta(n - n_0) = x(n_0) \cdot 1 \text{ since } \delta(n - n_0) = 1$$

$$\therefore x(n) \cdot \delta(n - n_0) = x(n_0)$$

And remaining all samples in the product will be zero. This is shown in Fig. 2.7.15 (c).

(ii) Interpretation of $x(n) * \delta(n - n_0) = x(n - n_0)$:

This can be best explained with the help of an example. For example consider,

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

and $n_0 = 1$ Hence $\delta(n - n_0) = 1$ for $n = n_0 = 1$
 $= 0$ for $n \neq n_0$ i.e.1

Let us represent, $h(n) = \delta(n - n_0)$ with $n_0 = 1$

By definition linear convolution is given as,

$$\begin{aligned}
 x(n) * \delta(n - n_0) &= x(n) * h(n) \quad \text{Here } h(n) = \delta(n - n_0) \\
 &= \sum_{k=-\infty}^{\infty} x(k) \cdot h(n - k) \quad \dots (2.7.6i)
 \end{aligned}$$

And now we have $x(k)$ and $h(k)$ as.

$$x(k = -1) = 1$$

$$x(k = 0) = 2$$

$$x(k = 1) = -1$$

And
$$\begin{aligned}
 h(k) = \delta(n - n_0) &= 1 \quad \text{for } n = n_0 \text{ i.e. } n = 1 \\
 &\quad \text{i.e. for } k = 1 \text{ since } k = n \\
 &= 0 \quad \text{elsewhere}
 \end{aligned}$$

Thus,
$$\begin{aligned}
 h(k = -1) &= 0 \\
 h(k = 0) &= 0 \\
 h(k = 1) &= 1 \quad \text{and } h(k = 2) = 0 \quad \text{and so on.}
 \end{aligned}$$

The sketches of sequences $x(k)$ and $h(k)$ as given above are shown in Fig. 2.7.16 (a) and Fig. 2.7.16 (b) respectively. The convolution equation of equation 2.7.61 can be written as,

$$y(n) = \sum_{k=-\infty}^{\infty} x(k) h(n - k) \quad \dots (2.7.62)$$

Here $y(n)$ is denoted as convolution of $x(n)$ and $h(n)$.

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

Thus $y(0)$ is equal to sum of multiplications of $x(k)$ and $h(-k)$. $h(-k)$ is the folded sequence and shown in Fig. 2.7.16 (c). The product sequence $x(k)h(-k)$ is shown in Fig. 2.7.16 (d). From Fig. 2.7.16 (d) it is clear that,

$$y(0) = \sum_{\text{over all samples}} x(k) h(-k) = 1 + 0 + 0 = 1 \quad \boxed{y(0) = 1}$$

$n = 1$ in equation 2.7.62 gives,

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

The sequence $h(1 - k)$ is shown in Fig. 2.7.16 (e). It is obtained by shifting $h(-k)$ to right side by one sample. The product sequence $x(k)h(1 - k)$ is shown in Fig. 2.7.16 (f). From Fig. 2.7.16 (f),

$$y(1) = \sum_{\text{over all samples}} x(k) h(1 - k) = 0 + 2 + 0 = 2 \quad \boxed{y(1) = 2}$$

Similarly from Fig. 2.7.16 (g) and Fig. 2.7.16 (h),

$$y(2) = \sum_{\text{over all samples}} x(k) h(2 - k) = 0 + 0 - 1 = -1 \quad \boxed{y(2) = -1}$$

By putting $n = 3$ it can be verified easily that, there will be no overlap between sequences $x(k)$ and $h(3 - k)$. Hence product sequence $x(k)h(3 - k)$ will have all its samples zero. And hence all values of $y(n)$ will be zero for $n \geq 3$.

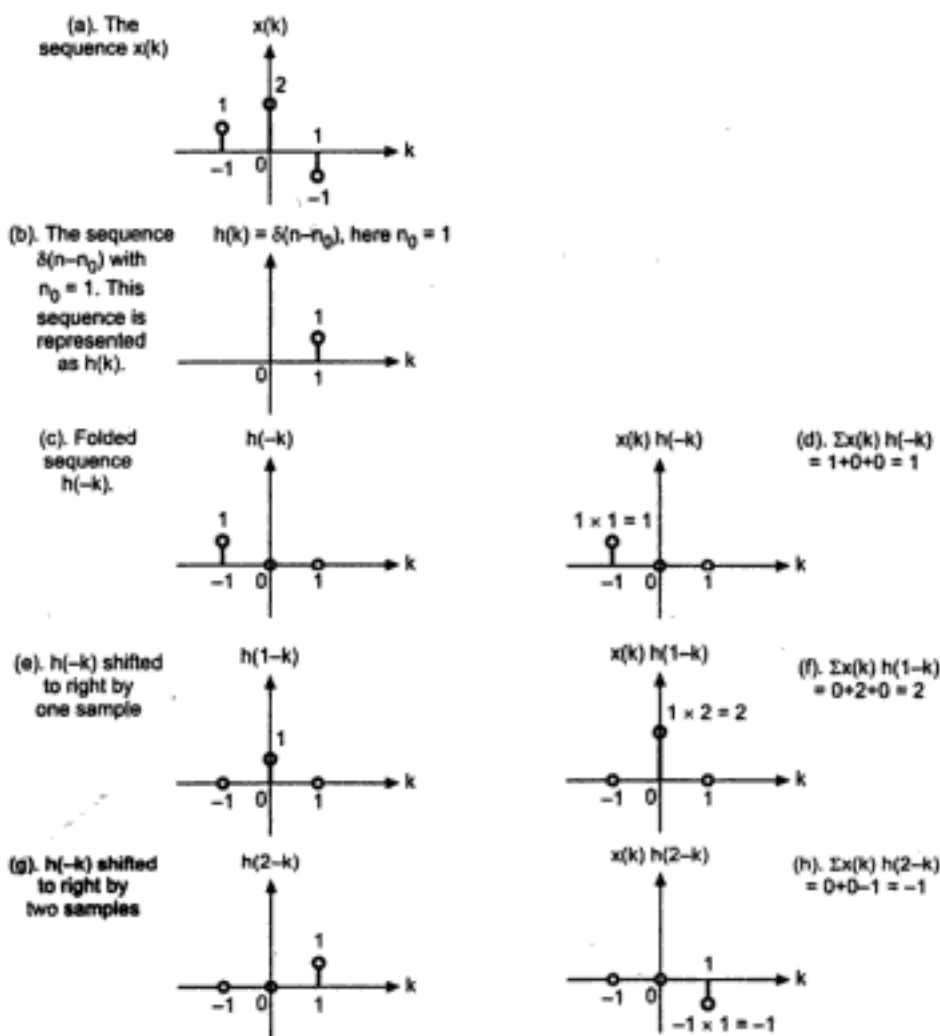


Fig. 2.7.16 Illustration of linear convolution of $x(n)$ and $\delta(n - n_0)$. Here $x(n)$ is represented as $x(k)$ and $\delta(n - n_0)$ is represented as $h(k)$. $n_0 = 1$

Similarly it can be verified easily that, for $n = -1$, we have to shift $h(-k)$ to left by one sample. This sequence becomes $h(-1-k)$. And there will be no overlap between samples of $x(k)$ and $h(-1-k)$. Hence the product sequence $x(k)h(-1-k)$ will have all its samples zero. And hence all values of $y(n)$ will be zero for $n \leq -1$. Thus we obtained $y(n)$ as,

$$y(n) = \{1, 2, -1\}$$

We know that $y(n)$ is the convolution of $x(n)$ and $\delta(n - n_0)$, since $h(n) = \delta(n - n_0)$ with $n_0 = 1$. Fig. 2.7.17 shows $x(n)$ and $y(n)$.

Please refer Fig. 2.7.17 on next page.

In the above figure observe $x(n)$ and $y(n)$ carefully. The sequence $y(n)$ is basically $x(n)$ delayed by one sample (i.e. $n_0 = 1$).

$$y(n) = x(n - n_0) \text{ Here } n_0 = 1 \text{ in Fig. 2.7.17 (b)}$$

Since

$$y(n) = x(n) * \delta(n - n_0), \text{ we can write,}$$

$$x(n) * \delta(n - n_0) = x(n - n_0) \quad \dots (2.7.63)$$

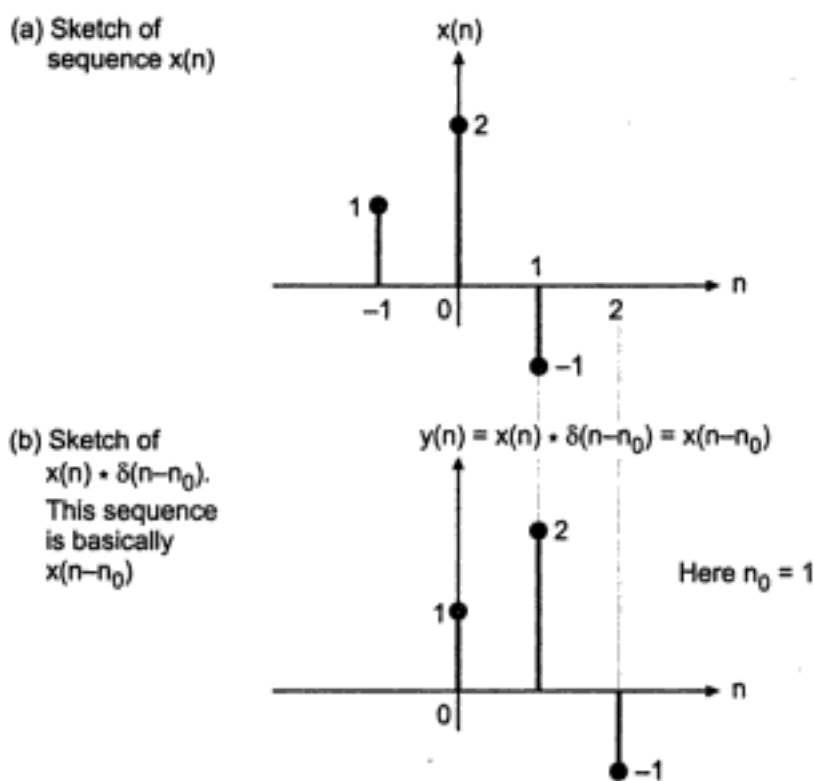


Fig. 2.7.17 Illustration of $x(n) * \delta(n-n_0) = x(n-n_0)$

Thus the convolution of any sequence with $\delta(n-n_0)$ is equivalent to delay that sequence by ' n_0 ' samples.

Ex.2.7.9 The impulse response of a linear time invariant system is

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

Find out the response of the system to the input signal

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

Sol. : We know that output $y(n)$ of the system is given as convolution of input $x(n)$ and impulse response $h(n)$. i.e.,

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

Please refer Fig. 2.7.18 on next page.

Fig. 2.7.18 shows the convolution of $x(n)$ and $h(n)$ by the method of multiplication. $x(n)$ and $h(n)$ are multiplied by the regular method. Observe that there are no samples in $x(n)$ before \uparrow arrow. There is one sample in $h(n)$ before \uparrow arrow. Hence total number of samples combinely due to $h(n)$ and $x(n)$ before \uparrow are $(1+0=1)$ one. Hence there is one sample before \uparrow arrow in $y(n)$. Thus the result of convolution is,

				↓		
	$h(n) \Rightarrow$	1		2	1	-1
	$x(n) \Rightarrow$		1	2	3	1
			↑			
		$1 \times 1 = 1$		$1 \times 2 = 2$	$1 \times 1 = 1$	$1 \times (-1) = -1$
		$3 \times 1 = 3$	$3 \times 2 = 6$	$3 \times 1 = 3$	$3 \times (-1) = -3$	x
	$2 \times 1 = 2$	$2 \times 2 = 4$	$2 \times 1 = 2$	$2 \times (-1) = -2$	x	x
$1 \times 1 = 1$	$1 \times 2 = 2$	$1 \times 1 = 1$	$1 \times (-1) = -1$	x	x	x
1	$2 + 2 = 4$	$3 + 4 + 1 = 8$	$1 + 6 + 2 - 1 = 8$	$2 + 3 - 2 = 3$	$1 - 3 = -2$	-1
	↑					

$\therefore y(n) = \{1, 4, 8, 8, 3, -2, -1\}$
↑

Fig. 2.7.18 Convolution of $x(n)$ and $h(n)$ to get $y(n)$

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

↑

This is the required output sequence.

Ex.2.7.10 Prove that the convolution of any sequence with the unit sample sequence results in the same sequence.

OR

$$x(n) * \delta(n) = x(n) \quad \dots (2.7.64)$$

OR

$$x(n) * h(n) = x(n) \quad \text{if } h(n) = \{1\} \quad \dots (2.7.65)$$

Sol. : We know that convolution of $x(n)$ and $h(n)$ is given as,

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

Since convolution is commutative, above equation can be written as,

$$x(n) * h(n) = \sum_{k=-\infty}^{\infty} h(k) x(n-k) \quad \dots (2.7.66)$$

$$h(k) = \delta(k) = \begin{cases} 1 & \text{at } k = 0 \\ 0 & \text{if } k \neq 0 \end{cases}$$

Hence the summation of equation 2.7.64 is evaluated only at $k = 0$. i.e.,

$$\begin{aligned} x(n) * h(n) &= h(k) x(n-k) \Big|_{k=0} \\ &= 1 \cdot x(n-0) \\ &= x(n) \end{aligned}$$

Thus $x(n) * h(n) = x(n) * \delta(n) = x(n)$ if $h(n) = \delta(n)$.

Ex.2.7.11 Determine the convolution between $x(n)$ and $h(n)$ as given below :

$$(i) \quad \begin{array}{l} h(n) = u(n) \quad \text{and} \\ x(n) = \{1, 1, 2\} \\ \quad \quad \quad \uparrow \end{array}$$

$$(ii) \quad \begin{array}{l} x(n) = \{1, 1, 2, 5\} \\ \quad \quad \quad \uparrow \end{array}$$

$$(iii) \quad \begin{array}{l} x(n) = \{4, 1, 3, 2\} \\ \quad \quad \quad \uparrow \end{array}$$

And hence determine the generalized relation for convolution of $u(n)$ with finite duration sequence.

Sol. : (i) Consider the first sequence,

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

Convolution is given as,

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

Here $x(k)$ has the values from 0 to 2. Hence above equation becomes,

$$y(n) = \sum_{k=0}^2 x(k) h(n-k) \quad \dots (2.7.67)$$

Since n_{x_l} as well as n_{h_l} are zero, the lowest index for $y(n)$ will be $n_{y_l} = n_{x_l} + n_{h_l} = 0$. Hence, $n = 0$ in equation 2.7.67 gives,

$$y(0) = \sum_{k=0}^2 x(k) h(a-k)$$

We know that $h(n) = u(n)$. Hence $h(k) = u(k)$ and hence $h(-k)$ is obtained by folding $u(k)$. $x(k)$ and $h(-k)$ are shown in Fig. 2.7.19 (a) and Fig. 2.7.19 (c) respectively. Hence $y(0)$ becomes,

$$y(0) = \sum_{k=0}^2 x(k) h(-k) = 1 \quad \dots (2.7.68 (a))$$

Similarly $n = 1$ in equation 2.7.67 becomes,

$$y(1) = \sum_{k=0}^2 x(k) h(1-k)$$

$h(1-k)$ is obtained by shifting $h(-k)$ to right by one sample. Fig. 2.7.19 (d) shows the sketch of $h(1-k)$. From Fig. 2.7.19 (a) and Fig. 2.7.19 (d) it is clear that,

$$y(1) = \sum_{k=0}^2 x(k) h(1-k) = 2 \quad \dots (2.7.68 (b))$$

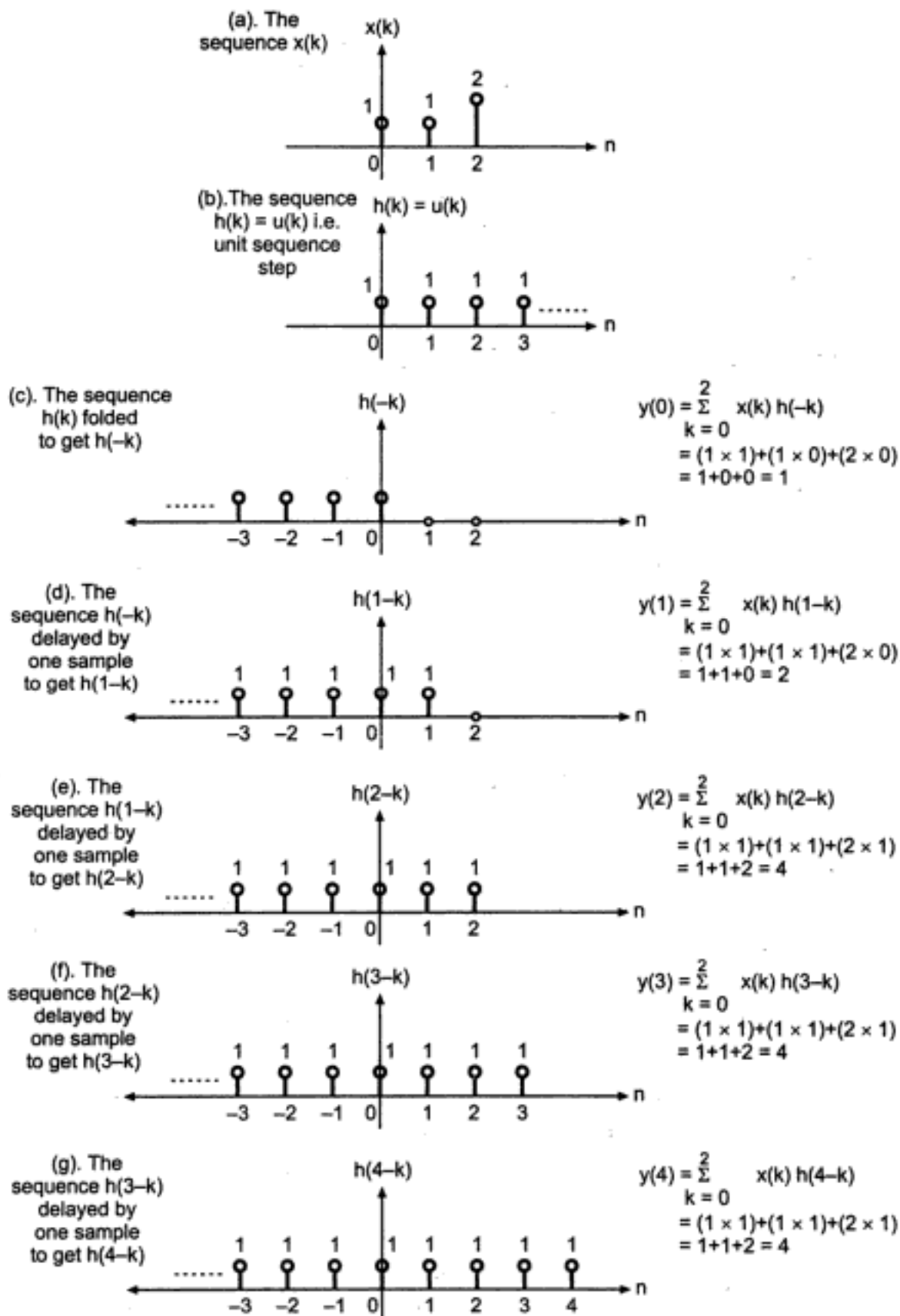


Fig. 2.7.19 Convolution of finite length $x(n)$ with unit step sequence $u(n)$

Similarly from Fig. 2.7.19 other values of $y(n)$ are obtained as,

$$y(2) = 4 \quad \dots (2.7.68 (c))$$

$$y(3) = 4 \quad \dots (2.7.68 (d))$$

$$y(4) = 4 \quad \dots \text{and so on.}$$

Thus $y(n)$ is an infinite sequence.

By careful observation of equation 2.7.68 (a), equation 2.7.68 (b), equation 2.7.68 (c), equation 2.7.68 (d) etc we can derive the generalized relation for $y(n)$ as,

$$y(n) = u(n) + u(n-1) + 2u(n-2) \quad \dots (2.7.69)$$

This can be verified easily as follows :

$$y(0) = u(0) + u(-1) + 2u(-2), \text{ with } n=0 \text{ in equation 2.7.69}$$

$$\text{Here } u(0)=1, \quad u(-1)=u(-2)=0, \quad \text{Hence } y(0)=1$$

$$y(1) = u(1) + u(0) + 2u(-1), \text{ with } n=1 \text{ in equation 2.7.69}$$

$$\text{Here } u(1) = u(0) = 1 \text{ and } u(-1) = 0, \quad \text{Hence } y(1) = 1 + 1 = 2$$

$$y(2) = u(2) + u(1) + 2u(0), \text{ with } n=2 \text{ in equation 2.7.69}$$

$$\text{Here } u(2) = u(1) = u(0) = 1. \quad \text{Hence } y(2) = 1 + 1 + 2 = 4$$

Thus equation 2.7.69 is the generalized equation for convolution. This equation can be generalized further. We know that,

$$x(0) = 1, \quad x(1) = 1 \text{ and } x(2) = 2$$

Hence equation 2.7.69 can be written as,

$$y(n) = x(0)u(n) + x(1)u(n-1) + x(2)u(n-2) \quad \dots (2.7.70)$$

This is more generalized form of convolution of sequence $x(n)$ with $u(n)$. It can be extended further as,

$$y(n) = x(0)u(n) + x(1)u(n-1) + x(2)u(n-2) + x(3)u(n-3) + \dots \quad \dots (2.7.71)$$

$$(ii) \quad y(n) = x(n) * u(n)$$

Here

$$x(n) = \{1, 1, 2, 5\}$$

↑

From equation 2.7.71 we can write the sequence $y(n)$ quickly as,

$$y(n) = u(n) + u(n-1) + 2u(n-2) + 5u(n-3)$$

$$(iii) \quad y(n) = x(n) * u(n)$$

Here

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

↑

Here observe that there is one sample before 0^{th} sample. Hence equation 2.7.71 can be modified as follows :

$$y(n) = x(-1)u(n+1) + x(0)u(n) + x(1)u(n-1) + x(2)u(n-2) + \dots \quad \dots (2.7.72)$$

In the above equation observe that the term $x(-1)u(n+1)$ is added to incorporate the sample $x(-1)$. Putting the values of sequence $x(n)$ in above equation we get,

$$y(n) = 4u(n+1) + u(n) + 3u(n-1) + 2u(n-2)$$

The correctness of above equation can be verified easily by actually carrying out convolution for first few samples of $y(n)$. Equation 2.7.72 can be generalized further as,

$$y(n) = \dots + x(-3)u(n+3) + x(-2)u(n+2) + x(-1)u(n+1) + x(0)u(n) + x(1)u(n-1) + x(2)u(n-2) + \dots \quad \dots (2.7.73)$$

Ex.2.7.12 The impulse response of the relaxed LTI system is given as,

$$h(n) = a^n u(n) \text{ and } |a| < 1$$

Determine the response of this system if it is excited by unit step sequence.

Sol. : Here excitation is unit step sequence, means,

$$x(n) = u(n)$$

And we have to determine response $y(n)$. i.e.,

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

Here we have to perform convolution of unit step sequence with another infinite duration sequence. This is because $h(n) = a^n u(n)$ is infinite duration sequence. In the previous example we have derived the generalized expression for convolution of unit step sequence with finite duration sequence. This expression is given by equation 2.7.73 i.e.,

$$y(n) = \dots + x(-3)u(n+3) + x(-2)u(n+2) + x(-1)u(n+1) + x(0)u(n) \\ + x(1)u(n-1) + x(2)u(n-2) + \dots$$

In this equation input is $x(n)$ and $h(n) = u(n)$. But if $x(n) = u(n)$, then above equation can be modified easily by commutative property of convolution. i.e.,

$$y(n) = \dots + h(-3)u(n+3) + h(-2)u(n+2) + h(-1)u(n+1) + h(0)u(n) \\ + h(1)u(n-1) + h(2)u(n-2) + \dots \quad \dots (2.7.74)$$

Even though we have derived equation 2.7.73 and above equation for finite duration sequences, they can be used for infinite duration sequences as well.

In this example $x(n) = u(n)$ and

$$h(n) = a^n u(n) = \begin{cases} a^n & \text{for } n \geq 0 \\ 0 & \text{for } n < 0 \end{cases}$$

The individual samples of $h(n)$ can be written as,

$$h(0) = 1$$

$$h(1) = a$$

$$h(2) = a^2$$

$$h(3) = a^3 \dots \text{and so on.}$$

i.e.,
$$h(n) = \{1, a, a^2, a^3, a^4, \dots\}$$

Thus $h(n)$ is infinite duration sequence. The convolution of any sequence with unit step sequence is given by equation 2.7.73 and equation 2.7.74. Here putting individual sample values of $h(n)$ in equation 2.7.74 we get,

$$y(n) = u(n) + au(n-1) + a^2u(n-2) + a^3u(n-3) + \dots \quad \dots (2.7.75)$$

This is the result of convolution. Let us compute sequence for $y(n)$. Putting $n = 0$ in above equation we get,

$$y(0) = u(0) + au(-1) + a^2u(-2) + a^3u(-3) + \dots \\ = 1 \quad \text{since } u(0) = 1 \text{ and } u(-1) = u(-2) = \dots = 0$$

Putting $n = 1$ in equation 2.7.75,

$$y(1) = u(1) + au(0) + a^2u(-1) + a^3u(-2) + \dots$$

$$= 1 + a \text{ since } u(1) = u(0) = 1 \text{ and } u(-1) = u(-2) = \dots = 0$$

Putting $n = 2$ in equation 2.7.75,

$$\begin{aligned} y(2) &= u(2) + a u(1) + a^2 u(0) + a^3 u(-1) + \dots \\ &= 1 + a + a^2 \text{ since } u(2) = u(1) = u(0) = 1 \text{ and } u(-1) = \dots = 0 \end{aligned}$$

Similarly the n^{th} sample of $y(n)$ can be obtained as,

$$y(n) = 1 + a^2 + a^3 + a^4 + \dots + a^n \quad \dots (2.7.76)$$

Compare equation 2.7.75 and above equation. There is no difference and $u(n) = u(n-1) = u(n-2) = \dots = 1$ for $n \geq 0$. With the help of standard relation in series, above equation can be written as,

$$y(n) = \frac{a^{n+1} - 1}{a - 1} \quad \dots (2.7.77)$$

The above equation is written with the help of following standard series

$$\sum_{k=0}^N a^k = \frac{a^{N+1} - 1}{a - 1} = \frac{1 - a^{N+1}}{1 - a} \quad \dots (2.7.78)$$

Ex.2.7.13 Two discrete time LTI systems are connected in cascade as shown in Fig. 2.7.20. Determine the unit sample response of this cascade connection.

Sol. : The cascade connection of two LTI systems is illustrated in Fig. 2.7.9 earlier. The unit sample response of the cascade connection is given by equation 2.7.33, i.e.,

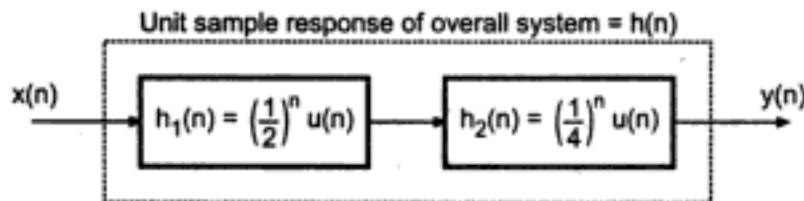


Fig. 2.7.20 Cascade connection of two discrete time systems

$$h(n) = h_1(n) * h_2(n) \quad \dots (2.7.79)$$

$$= \sum_{k=-\infty}^{\infty} h_1(k) \cdot h_2(n-k) \quad \dots (2.7.80)$$

Consider the first term i.e. $h_1(k)$. Since $h_1(n) = \left(\frac{1}{2}\right)^n u(n)$, $h_1(k) = \left(\frac{1}{2}\right)^k u(k)$. We know that $u(k) = 0$ for $k < 0$. Hence,

$$h_1(k) = \left(\frac{1}{2}\right)^k u(k) \text{ for } k \geq 0 \quad \dots (2.7.81)$$

Consider the second term in convolution of equation 2.7.80 i.e. $h_2(n-k)$. It is given that $h_2(n) = \left(\frac{1}{4}\right)^n u(n)$. Hence $h_2(n-k) = \left(\frac{1}{4}\right)^{n-k} u(n-k)$. We know that $u(n-k) = 0$ for $n < k$. Hence,

$$h_2(n-k) = \left(\frac{1}{4}\right)^{n-k} u(n-k) \text{ for } n \geq k \quad \dots (2.7.82)$$

Putting the values of equation 2.7.81 and above equation in equation 2.7.80 we can write,

$$h(n) = \sum_{k=-\infty}^{\infty} \left(\frac{1}{2}\right)^k u(k) \left(\frac{1}{4}\right)^{n-k} u(n-k) \quad \dots (2.7.83)$$

Since $k \geq 0$, the lower limit of summation in above equation will be $k = 0$. From equation 2.7.82 we know that $n \geq k$ always. Hence upper limit of summation in above equation will be $k = n$. This means if $n = \infty$, $k = \infty$; But k will have maximum value equal to n . Thus we can write equation 2.7.83 as,

$$h(n) = \sum_{k=0}^n \left(\frac{1}{2}\right)^k \cdot \left(\frac{1}{4}\right)^{n-k}$$

In the above equation, we have not written $u(k)$ and $u(n-k)$ since the limits of summation are modified accordingly. Above equation can be further simplified as follows :

$$\begin{aligned} h(n) &= \sum_{k=0}^n \left(\frac{1}{2}\right)^k \cdot \left(\frac{1}{4}\right)^n \cdot \left(\frac{1}{4}\right)^{-k} \\ &= \left(\frac{1}{4}\right)^n \sum_{k=0}^n \frac{1}{2^k} \cdot \frac{1}{4^{-k}} \\ &= \left(\frac{1}{4}\right)^n \sum_{k=0}^n \frac{1}{2^k} \cdot 4^k \\ &= \left(\frac{1}{4}\right)^n \sum_{k=0}^n \frac{1}{2^k} \cdot (2 \times 2)^k \\ &= \left(\frac{1}{4}\right)^n \sum_{k=0}^n \frac{1}{2^k} \cdot 2^k \times 2^k \\ &= \left(\frac{1}{4}\right)^n \sum_{k=0}^n 2^k \quad \dots (2.7.84) \end{aligned}$$

Using equation 2.7.78 we can write $\sum_{k=0}^n 2^k$ as,

$$\begin{aligned} h(n) &= \left(\frac{1}{4}\right)^n \cdot \frac{2^{n+1} - 1}{2 - 1} \\ &= \left(\frac{1}{4}\right)^n [2^{n+1} - 1] \end{aligned}$$

This is the unit sample response of the cascade connection.

Ex.2.7.14 Determine the response of the system whose input $x(n]$ and unit sample response $h(n]$ is given as follows :

$$x(n) = \begin{cases} \frac{1}{3}n & \text{for } 0 \leq n \leq 6 \\ 0 & \text{elsewhere} \end{cases}$$

$$h(n) = \begin{cases} 1 & \text{for } -2 \leq n \leq 2 \\ 0 & \text{elsewhere} \end{cases}$$

Sol. : The input sequence $x(n]$ is,

$$x(n) = \left\{ 0, \frac{1}{3}, \frac{2}{3}, \frac{3}{3}, \frac{4}{3}, \frac{5}{3}, \frac{6}{3} \right\}$$

i.e.
$$x(n) = \left\{ 0, \frac{1}{3}, \frac{2}{3}, 1, \frac{4}{3}, \frac{5}{3}, 2 \right\}$$

The unit sample response $h(n]$ is,

$$h(n) = \{ 1, 1, 1, 1, 1 \}$$

This example can be solved graphically, by multiplication or tabulation. Let us use multiplication method as given below in Fig. 2.7.21 from Fig. 2.7.21, it is clear that the response of the system is given as,

$$y(n) = \left\{ \frac{1}{3}, 1, 2, \frac{10}{3}, 5, 6, \frac{16}{3}, 5, \frac{11}{3}, 2 \right\}$$

$x(n) \Rightarrow$	0	$\frac{1}{3}$	$\frac{2}{3}$	1	$\frac{4}{3}$	$\frac{5}{3}$	2				
	↑										
$h(n) \Rightarrow$		1	1	1	1	1					
			↑								
			0	$\frac{1}{3}$	$\frac{2}{3}$	1	$\frac{4}{3}$	$\frac{5}{3}$	2		
		0	$\frac{1}{3}$	$\frac{2}{3}$	1	$\frac{4}{3}$	$\frac{5}{3}$	2	×		
		0	$\frac{1}{3}$	$\frac{2}{3}$	1	$\frac{4}{3}$	$\frac{5}{3}$	2	×		
		0	$\frac{1}{3}$	$\frac{2}{3}$	1	$\frac{4}{3}$	$\frac{5}{3}$	2	×		
	0	$\frac{1}{3}$	$\frac{2}{3}$	1	$\frac{4}{3}$	$\frac{5}{3}$	2	×	×		
$y(n) =$	0	$\frac{1}{3}$	1	2	$\frac{10}{3}$	5	6	$\frac{16}{3}$	5	$\frac{11}{3}$	2
			↑								

Fig. 2.7.21 Computation of convolution of example 2.7.14

Ex.2.7.15 Determine the response of the LTI system whose input and unit sample response is given as,

$$x(n) = \begin{cases} n+1 & \text{for } 0 \leq n \leq 2 \\ 0 & \text{elsewhere} \end{cases}$$

and $h(n) = a^n u(n)$ for all 'n'

Sol. : Here $x(n)$ is finite duration and $h(n)$ is of infinite duration. In such cases the convolution should be performed carefully. We know that convolution is given as,

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

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

By commutative property of convolution.

The given sequence $x(n)$ is,

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

↑

and $h(n) = a^n u(n)$

In convolution we have to fold one sequence. It is always convenient to fold infinite duration sequence.

Here $h(n)$ is infinite duration sequence. In first convolution equation $h(n-k)$ means $h(n)$ is folded and shifted. Hence we will use first convolution equation. i.e.,

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

We have $x(0)=1$, $x(1)=2$ and $x(2)=3$. Hence above equation becomes,

$$y(n) = \sum_{k=0}^2 x(k) h(n-k)$$

Here observe that $x(k)$ is finite duration sequence. $x(n)=0$ for $k \leq -1$ and $k \geq 3$. Hence the product $x(k) h(n-k) = 0$ for $k \leq -1$ and $k \geq 3$. Therefore the limits of summation are $k=0$ to $k=2$ in above equation. Now let us put the value of $h(n)$ in above equation. i.e.,

$$y(n) = \sum_{k=0}^2 x(k) a^{n-k} u(n-k) \quad \dots (2.7.85)$$

In above equation $u(n-k)=1$ for $n \geq k$ and $u(n-k)=0$ for $n < k$. Hence above equation can be written as,

$$y(n) = \sum_{k=0}^2 x(k) a^{n-k} \quad \text{for } n \geq k \quad \dots (2.7.86)$$

First few samples of $y(n)$ will be,

$$y(0) = \sum_{k=0}^2 x(k) a^{-k} = x(0) \quad \text{since } n \geq k \text{ i.e. } k \leq n \text{ or } k \leq 0$$

$$= 1$$

$$y(1) = \sum_{k=0}^2 x(k) a^{1-k} = x(0)a + x(1) \text{ since } k \leq n \text{ i.e. } k \leq 1$$

$$= a + 2$$

$$y(2) = \sum_{k=0}^2 x(k) a^{2-k} = x(0)a^2 + x(1)a + x(0)$$

$$= a^2 + 2a + 3$$

$$y(3) = \sum_{k=0}^2 x(k) a^{3-k} = x(0)a^3 + x(1)a^2 + x(0)a$$

$$= a^3 + 2a^2 + 3a \dots \text{ and so on.}$$

Thus other samples can be computed. And $y(n) = 0$ for $n < 0$. This is because $y(n)$ is computed for $n \geq k$ as given by equation 2.7.86 and lowest value of $k = 0$.

From equation 2.7.85 we can write $y(n)$ alternately as,

$$y(n) = x(0)a^n u(n) + x(1)a^{n-1} u(n-1) + x(2)a^{n-2} u(n-2)$$

Here summation is expanded. Putting values of $x(n)$,

$$y(n) = a^n u(n) + 2a^{n-1} u(n-1) + 3a^{n-2} u(n-2) \dots (2.7.87)$$

Compare above two equations with equation 2.7.71. Check whether these equations are same if $a=1$. Then above equation can be directly obtained from equation 2.7.71.

Ex.2.7.16 Determine the output of the LTI system whose input and unit sample response are given as follows :

$$x(n) = b^n u(n)$$

and

$$h(n) = a^n u(n)$$

Sol. : Here both $x(n)$ and $h(n)$ are infinite duration sequences. We have already solved this type of convolution in example 2.7.13. By definition of convolution,

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

Putting the given sequences in above equation,

$$y(n) = \sum_{k=-\infty}^{\infty} b^k u(k) \cdot a^{n-k} u(n-k) \dots (2.7.88)$$

Here
$$u(k) = \begin{cases} 1 & \text{for } k \geq 0 \\ 0 & \text{for } k < 0 \end{cases}$$

Hence lower limit of summation in equation 2.7.88 becomes $k = 0$ and $u(k) = 1$. i.e.,

$$y(n) = \sum_{k=0}^{\infty} b^k a^{n-k} u(n-k) \dots (2.7.89)$$

In above equation
$$u(n-k) = \begin{cases} 1 & \text{for } n \geq k \\ 0 & \text{for } n < k \text{ or } k > n \end{cases}$$

Hence upper limit of summation in equation 2.7.89 becomes 'n' and $u(n-k) = 1$, i.e.,

$$\begin{aligned}
 y(n) &= \sum_{k=0}^n b^k a^{n-k} \quad \text{Here } n \geq k \geq 0 \\
 &= \sum_{k=0}^n b^k a^n \cdot a^{-k} \\
 &= a^n \sum_{k=0}^n (b a^{-1})^k
 \end{aligned}$$

Using equation 2.7.78 we can write the summation in above equation as,

$$\begin{aligned}
 y(n) &= a^n \cdot \frac{(b a^{-1})^{n+1} - 1}{(b a^{-1}) - 1} \quad \text{for } n \geq 0 \\
 &= a^n \cdot \frac{\left(\frac{b}{a}\right)^{n+1} - 1}{\frac{b}{a} - 1} \\
 &= a^n \cdot a \frac{\left(\frac{b}{a}\right)^{n+1} - 1}{b - a} \\
 &= a^{n+1} \cdot \frac{b^{n+1} - a^{n+1}}{b - a} \\
 &= \frac{b^{n+1} - a^{n+1}}{b - a} \quad \text{for } n \geq 0 \text{ and } a \neq b
 \end{aligned}$$

Ex.2.7.17 Determine the response of the system whose unit sample response and input are given as follows :

$$x(n) = u(n+1) - u(n-4) - \delta(n-5)$$

and

$$h(n) = [u(n+2) - u(n-3)] \cdot (3 - |n|)$$

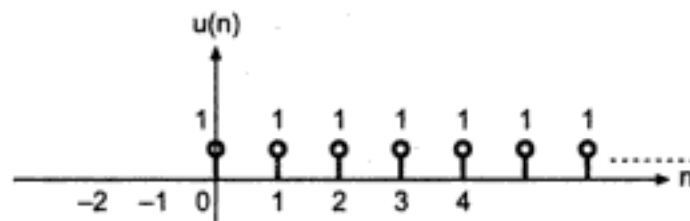
Sol. : Let us first determine the sequences $x(n)$ and $h(n)$. These sequence can be best obtained from graphical interpretation. Fig. 2.7.22 shows how sequence $x(n)$ is obtained. Fig. 2.7.22 (a) shows the unit step sequence $u(n)$. The sequence $u(n+1)$ can be obtained by advancing $u(n)$ by one sample. That is to shift $u(n)$ left by one sample. This $u(n+1)$ sequence is shown in Fig. 2.7.22 (b). The sequence $u(n-4)$ is obtained by delaying $u(n)$ by four samples. That is to shift $u(n)$ right by four samples. This $u(n-4)$ sequence is shown in Fig. 2.7.22 (c). The sequence $\delta(n-5)$ is basically a unit sample delayed by five samples. This is shown in Fig. 2.7.22 (d).

Now let us perform the addition and subtractions on $u(n+1)$, $u(n-4)$ and $\delta(n-5)$ on sample to sample basis as per given $x(n)$. i.e.,

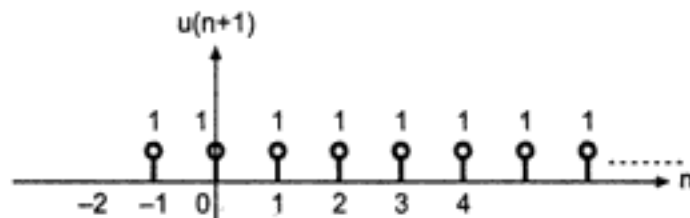
$$x(n) = u(n+1) - u(n-4) - \delta(n-5)$$

The sequence $x(n)$ is shown in Fig. 2.7.22 (e). From this figure we can write $x(n)$ as,

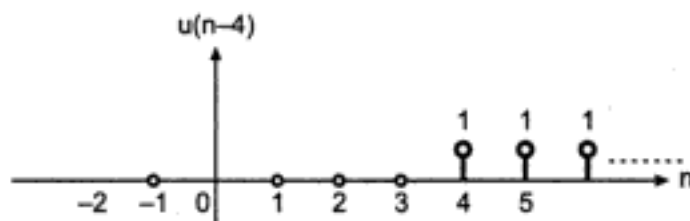
(a). Sketch of unit step sequence $u(n)$



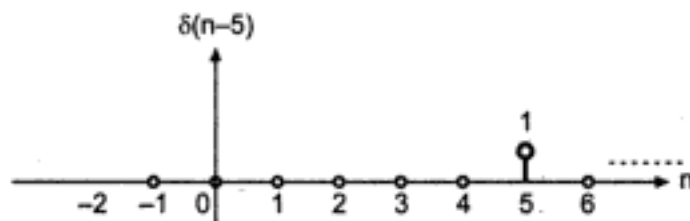
(b). $u(n)$ advanced by one sample



(c). $u(n)$ delayed by four samples



(d). The unit sample delayed by five samples



(e). $x(n)$ is obtained from sequences of Fig. (b), (c), & (d)

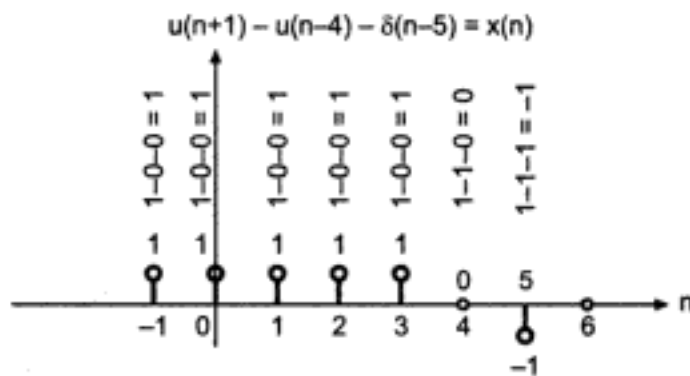
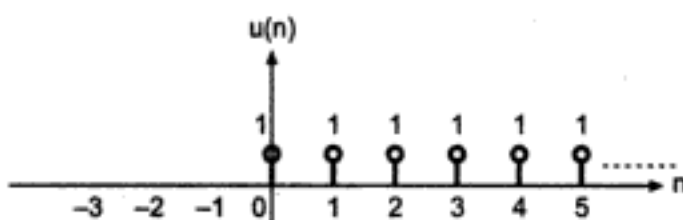


Fig. 2.7.22 To obtain $x(n) = u(n+1) - u(n-4) - \delta(n-5)$. Here $x(n) = \{1, 1, 1, 1, 1, 0, -1\}$

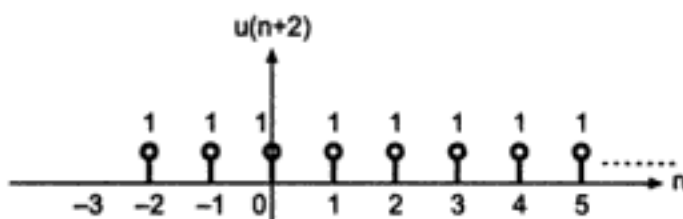
$$x(n) = \{1, 1, 1, 1, 1, 0, -1\} \quad \dots (2.7.90)$$

Now let us see how to obtain $h(n)$. Fig. 2.7.23 (a) shows the unit step sequence $u(n)$. Fig. 2.7.23 (b) shows the sequence $u(n+2)$. This is obtained by advancing (i.e. shifting left) $u(n)$ by two samples. Fig. 2.7.23 (c) shows the sequence $u(n-3)$. This is obtained by delaying (i.e. shifting right) $u(n)$ by three samples. Fig. 2.7.23 (d) shows

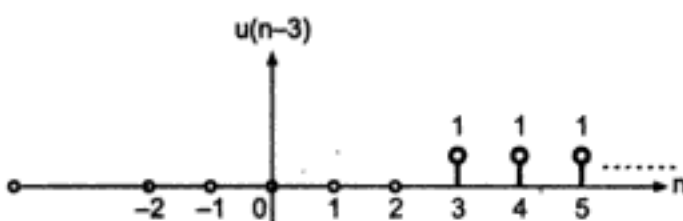
(a). Sketch of unit step sequence



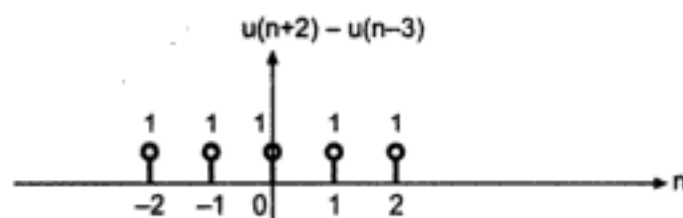
(b). $u(n+2)$ is obtained by advancing unit step sequence by two samples



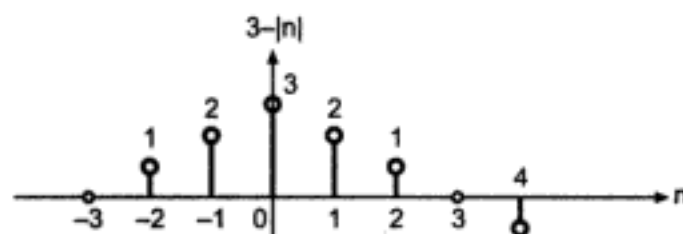
(c). $u(n-3)$ is obtained by delaying unit step sequence by three samples



(d). $u(n-3)$ of Fig.(c) is subtracted from $u(n+2)$ of Fig.(b) on sample to sample basis.



(e). The sequence $3-|n|$



(f). Multiplication of $[u(n+2) - u(n-3)]$ of Fig.(d) and $3-|n|$ of Fig.(e). This is sequence $h(n)$.

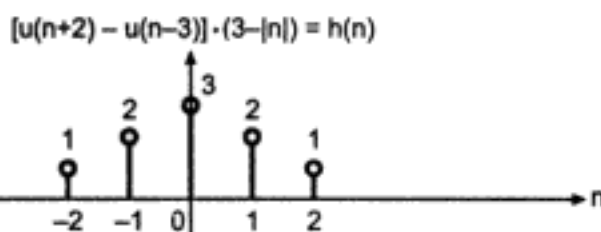


Fig. 2.7.23 To obtain $h(n) = [u(n+2) - u(n-3)] \cdot (3-|n|)$ Here $h(n) = \begin{Bmatrix} 1, & 2, & 3, & 2, & 1 \\ & & \uparrow & & \end{Bmatrix}$

$[u(n+2) - u(n-3)]$. Fig. 2.7.23 (e) shows the sketch of $(3-|n|)$. Fig. 2.7.23 (f) shows $h(n)$ which is obtained by multiplication of sequences of Fig. 2.7.23 (d) and Fig. (e). Thus from Fig. 2.7.23 (f), $h(n)$ is obtained as,

$$h(n) = \{1, 2, 3, 2, 1\} \quad \dots (2.7.91)$$

Output $y(n)$ is obtained by convolution of $x(n)$ and $h(n)$. The convolution using tabulat method is shown below in Fig. 2.7.24.

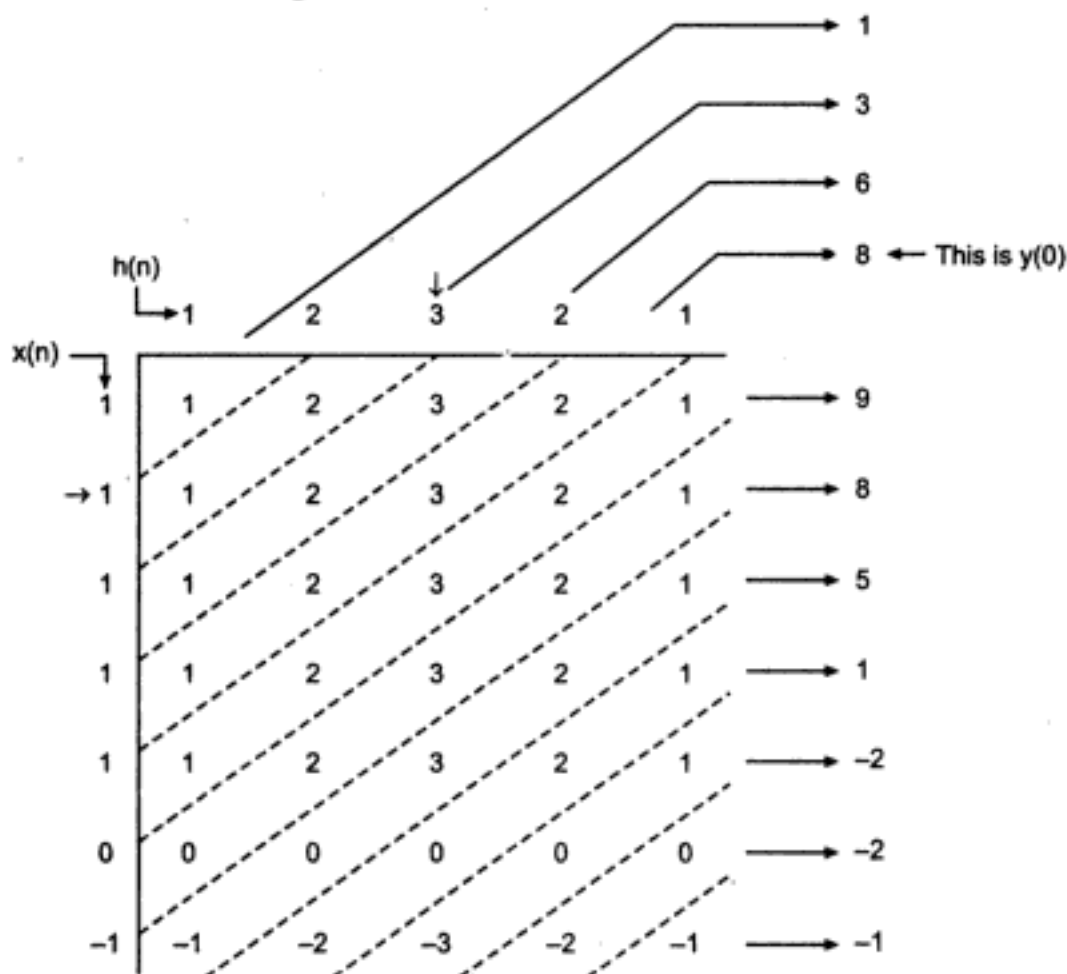


Fig. 2.7.24 Computation of convolution using tabulation method

From above figure it is clear that the computed sequence of $y(n)$ is,
 $y(n) = \{1, 3, 6, 8, 9, 8, 5, 1, -2, -2, -1\}$

The total number of elements before \uparrow in $x(n)$ and $h(n)$ is 'three'. Hence there are 'three' elements before \uparrow in $y(n)$.

2.8 Difference Equations

Till now we were discussing about discrete time signals, systems and linear shift invariant systems. In this section we will introduce difference equations which is an efficient way to implement discrete time systems. We know that the convolution of input sequence $x(n)$ and unit sample response $h(n)$ gives the output $y(n)$ i.e.,

$$y(n) = \sum_{k=-\infty}^{\infty} h(k) x(n-k) \quad \dots (2.8.1)$$

This is the standard convolution equation. If the LTI system is causal,

$$h(n) = 0 \text{ for } n < 0 \quad \dots (2.8.2)$$

then the output of such causal LTI becomes,

$$y(n) = \sum_{k=0}^{\infty} h(k)x(n-k) \quad \dots (2.8.3)$$

The above equation is obtained by putting the causality condition of equation 2.8.2 in equation 2.8.1. Two types of systems are possible depending upon the length of unit sample response $h(k)$ in equation 2.8.3. They are defined as follows :

2.8.1 Finite Impulse Response (FIR) Systems

In the equation 2.8.3 we have stated that the LTI system is causal. Hence $h(k) = 0$ for $k < 0$. But now if $h(k) = 0$ for $k > M$, then we can write equation 2.8.3 as,

$$y(n) = \sum_{k=0}^{M-1} h(k)x(n-k) \quad \dots (2.8.4)$$

Here observe that the unit sample response is non zero for $k = 0$ to $M - 1$. Thus there are finite number of terms in the unit sample response. These terms are,

$h(0), h(1), h(2), \dots, h(M-1)$: Number of terms are 'M'.

The systems for which unit sample response $h(n)$ has finite number of terms, they are called Finite Impulse Response (FIR) systems. And the output of such FIR systems is given by equation 2.8.4 above. Expanding equation 2.8.4 we get,

$$y(n) = h(0)x(n) + h(1)x(n-1) + h(2)x(n-2) + \dots + h(M-1)x(n-M+1) \quad \dots (2.8.5)$$

Thus the output $y(n)$ is equal to weighted sum of $x(n), x(n-1), x(n-2), \dots, x(n-M+1)$ input samples. These input samples are most recent 'M' samples. Thus the FIR system has to store $x(n-1), x(n-2), x(n-3) \dots, x(n-M+1)$ i.e. $(M-1)$ number of input samples in the memory.

2.8.2 Infinite Impulse Response (IIR) Systems

Now let us consider equation 2.8.3,

$$y(n) = \sum_{k=0}^{\infty} h(k)x(n-k) \quad \dots (2.8.6)$$

Here we have considered the causal LTI system and hence $h(k) = 0$ for $k < 0$. Now let $h(k)$ be non zero for all values of $k > 0$. That is, $h(0), h(1), h(2), h(3), \dots, h(\infty)$ all these terms will be non zero. Hence in the computation of output $y(n)$, we have to consider all values of $h(k)$ for $0 \leq k \leq \infty$. These are infinite number of terms in $h(k)$. The systems, for which such infinite number of unit sample response terms are to be considered are called as Infinite Impulse Response (IIR) systems. Thus we can expand equation 2.8.6 for IIR system as,

$$y(n) = h(0)x(n) + h(1)x(n-1) + h(2)x(n-2) + h(3)x(n-3) + \dots + h(\infty)x(n-\infty) \quad \dots (2.8.7)$$

This equation shows that computation of $y(n)$ involves use of $x(n)$ and all past inputs. Hence all past inputs are to be stored in the memory. Ideally, $x(n), x(n-1), x(n-2), \dots, x(n-\infty) \approx x(\infty)$ are the infinite number of inputs to be used for computation and stored in the memory. Thus the IIR systems needs infinite memory.

Comment :

Here we briefly defined the FIR and IIR systems. FIR systems can be implemented with the help of convolution given by equation 2.8.4. Such operation needs $(M - 1)$ number of past input samples are to be stored in the memory. But the computation becomes inefficient when 'M' becomes large.

But IIR systems described by convolution operation of equation 2.8.6 needs infinite number of samples to be stored in the memory. Hence memory requirement is infinite. This is not possible practically. Such IIR systems are implemented with the help of difference equations. Such implementation is very efficient in terms of computation and memory requirements.

2.8.3 Nonrecursive Systems

When the output $y(n)$ of the system depends upon present and past inputs, then it is called nonrecursive system. i.e.,

$$y(n) = F[x(n), x(n-1), x(n-2), \dots, x(n-M)] \quad \dots (2.8.8)$$

Here $F[\cdot]$ denotes the function of the quantity contained within the brackets $[\cdot]$. We know that the output $y(n)$ of causal FIR system is given as,

$$y(n) = \sum_{k=0}^M h(k)x(n-k) \quad \dots (2.8.9)$$

Here we have considered $(M + 1)$ terms of unit sample response compared to these of equation 2.8.4. [In equation 2.8.4 we have considered 'M' term of $h(n)$]. In the above causal FIR system $h(n)$ is non zero for $0 \leq n \leq M$. Expanding above equation we get,

$$\begin{aligned} y(n) &= h(0)x(n) + h(1)x(n-1) + h(2)x(n-2) + \dots + h(M)x(n-M) \quad \dots (2.8.10) \\ &= F[x(n), x(n-1), x(n-2), \dots, x(n-M)] \end{aligned}$$

Here $y(n)$ is the function of present and past M input samples. This is nonrecursive system. Thus the causal LTI FIR systems defined by linear convolution are basically nonrecursive systems. Fig. 2.8.1 shows the basic form for such nonrecursive systems. These systems do not use any feedback from the output $y(n)$. But observe that the system has to store $x(n-1), x(n-2), \dots, x(n-M)$ all these 'M' number of past input samples in the memory. Hence nonrecursive systems need large memory if 'M' is large. For causal IIR systems, the convolution equation is given by equation 2.8.6 as,

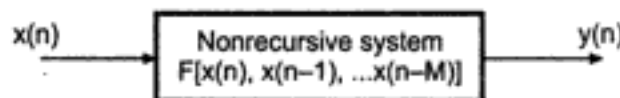


Fig. 2.8.1 Basic form for causal nonrecursive system

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

$$\begin{aligned} \therefore y(n) &= h(0)x(n) + h(1)x(n-1) + h(2)x(n-2) + \dots + h(\infty)x(n-\infty) \\ &= F[x(n), x(n-1), x(n-2), x(n-3), \dots, x(\infty)] \end{aligned}$$

This shows that for the nonrecursive implementation of IIR systems all (ideally infinite) input samples are to be stored in the memory. This is practically impossible. Hence implementation of causal IIR systems is not possible in nonrecursive form.

2.8.4 Recursive Systems

When the output $y(n]$ of the system depends upon the present and past inputs as well as past outputs, it is called recursive system. The causal FIR and IIR LTI systems can be efficiently implemented using recursive systems. The recursive systems are efficient in terms of memory requirement and computations.

Consider the system which is basically nonrecursive,

$$y(n) = \sum_{k=0}^n x(k) \quad \dots (2.8.11)$$

This can be assumed to be convolution equation with $h(n-k)=1$ for all terms. Let us expand the above equation for various values of n .

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

$$n=1 \Rightarrow y(1) = \sum_{k=0}^1 x(k) = x(0) + x(1) = y(0) + x(1) \text{ from above equation}$$

$$n=2 \Rightarrow y(2) = \sum_{k=0}^2 x(k) = x(0) + x(1) + x(2) = y(1) + x(2) \text{ from above equation}$$

$$n=3 \Rightarrow y(3) = \sum_{k=0}^3 x(k) = x(0) + x(1) + x(2) + x(3) = y(2) + x(3) \text{ from above equation}$$

: :
: :

$$n=n-1 \Rightarrow y(n-1) = \sum_{k=0}^{n-1} x(k) = x(0) + x(1) + x(2) + x(3) + \dots + x(n-1)$$

$$n=n \Rightarrow y(n) = \sum_{k=0}^n x(k) = x(0) + x(1) + x(2) + x(3) + \dots + x(n-1) + x(n) = y(n-1) + x(n) \text{ from above equation}$$

Thus we have expressed the equation 2.8.11 as,

$$y(n) = y(n-1) + x(n) \quad \dots (2.8.12)$$

Here $y(n-1)$ is the previous output. Thus the present output $y(n)$ is calculated using previous output $y(n-1)$ and present input $x(n)$. This is recursive system. In the recursive systems the previous outputs are used to calculate next outputs. Fig. 2.8.2 shows the implementation of recursive system of equation 2.8.12. Observe that the feedback is present since it is recursive system. This system needs only one memory location to store the previous

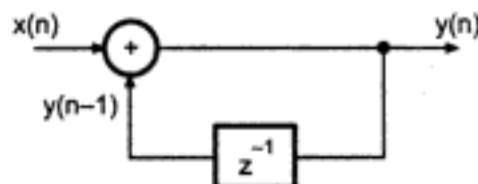


Fig. 2.8.2 Block diagram representation of the recursive system of equation 2.8.12

output $y(n-1)$. The z^{-1} block indicates delay of one sample. And there is only one addition. Whereas if we implement equation 2.8.11 using nonrecursive, then the system has to store all previous outputs. Similarly all the terms are to be added. This requirement of memory increases as 'n' increases. Thus recursive systems are efficient in terms of memory and computations. The IIR systems are also converted in the recursive form to make their implementation easy. Here we have shown that the output $y(n)$ is some function of previous outputs and present and past inputs in recursive systems. i.e.,

$$y(n) = F [y(n-1), y(n-2), y(n-3) \dots y(n-N), x(n), x(n-2) x(n-3) \dots x(n-M)] \quad \dots (2.8.13)$$

Fig. 2.8.3 shows the basic form of recursive system based on above equation.

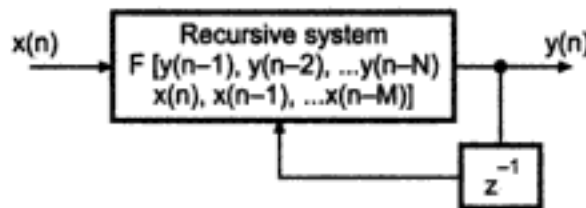


Fig. 2.8.3 Basic representation of recursive system

In the above block diagram the delay block shown separately is crucial. Since it provides previous output for calculations. It is not possible to calculate $y(n)$ in terms of $y(n)$, which would be the case if delay block is absent !

2.8.5 Representation of Discrete Time Systems via difference Equations.

In the last subsection we studied about how recursive equations can be written for FIR and IIR systems. Now let us see the generalized form of constant coefficient difference equations used to represent LTI systems. Basically the recursive equation given by equation 2.8.12 is a constant coefficient difference equation. Observe that the coefficients of $y(n-1)$ and $x(n)$ are constants, i.e. 1. Now let us consider the recursive system having,

$$y(n) = a y(n-1) + x(n) \quad \dots (2.8.14)$$

Observe that this equation has constant coefficients of 'a' and '1'. Let us find out $y(n)$ in above equation for different values of n . i.e.,

$$n=0 \Rightarrow y(0) = a y(-1) + x(0)$$

$$n=1 \Rightarrow y(1) = a y(0) + x(1) = a^2 y(-1) + a x(0) + x(1)$$

$$n=2 \Rightarrow y(2) = a y(1) + x(2) = a^3 y(-1) + a^2 x(0) + a x(1) + x(2)$$

$$n=3 \Rightarrow y(3) = a y(2) + x(3) = a^4 y(-1) + a^3 x(0) + a^2 x(1) + a x(2) + x(3)$$

:

:

$$y(n) = a y(n-1) + x(n) = a^{n+1} y(-1) + a^n x(0) + a^{n-1} x(1) + \dots + a x(n-1) + x(n)$$

$$= a^{n+1} y(-1) + \sum_{k=0}^n a^k x(n-k) \quad \dots (2.8.15)$$

Here since we have considered the causal system, the output $y(n)$ is calculated for $n \geq 0$. For this calculation we need value of $y(-1)$. This value of $y(-1)$ is basically initial condition of the system. When $y(-1) = 0$, the system is said to be relaxed initially. This is also called zero state of the system. And the output obtained with zero initial state is called zero state response or forced response of the system. It is called forced response, since it is forced due to input. It is denoted by $y_{zs}(n)$. Hence equation 2.8.15 becomes,

$$y_{zs}(n) = \sum_{k=0}^n a^k x(n-k) \quad \text{with } y(-1) = 0 \quad \dots (2.8.16)$$

If the system is not relaxed initially, i.e. $y(-1) \neq 0$ and input $x(n) = 0$ for all n , then the response obtained is called zero input response or natural response. It is denoted by $y_{zi}(n)$, i.e., with $x(n) = 0$ for all n in equation 2.8.15,

$$y_{zi}(n) = a^{n+1} y(-1) \quad \dots (2.8.17)$$

Thus the nonrelaxed system produces output even if input is absent. The total response of the system is given as,

$$y(n) = y_{zi}(n) + y_{zs}(n) \quad \dots (2.8.18)$$

Then the total response is linear combination of two responses. This can also be considered as the condition for linearity of the system.

Till now we have considered two recursive systems given by equation 2.8.12 and equation 2.8.14. These are basically linear constant coefficient difference equations. The generalized form of such equation is given as,

$$y(n) = - \sum_{k=1}^N a_k y(n-k) + \sum_{k=0}^M b_k x(n-k) \quad \dots (2.8.19)$$

Here 'N' represents the order of the difference equation and hence order of the system.

Linearity, shift invariance and stability of systems represented by difference equations:

Here we will consider the linearity, shift invariance and stability properties of recursive systems described by constant coefficient difference equations.

Linearity :

A system is said to be linear if it satisfies the following requirements :

(i) The total response $y(n)$ is equal to the sum of zero input response $y_{zi}(n)$ and zero state response $y_{zs}(n)$ i.e.,

$$y(n) = y_{zi}(n) + y_{zs}(n)$$

(ii) Zero input response should be linear, and

(iii) Zero state response should be linear.

Shift invariance :

The difference equations clearly state the input/output relationship. The coefficients a_k and b_k in difference equations are constants and independent of shift. Hence the recursive systems described by constant coefficient difference equations are shift invariant. Such systems are linear also. Hence they are Linear Shift Invariant (LTI) systems.

Stability :

The recursive system described by linear constant coefficient difference equation is BIBO stable if for every bounded input and every bounded initial condition, the output is bounded.

2.9 Correlation

The correlation technique is very much similar to convolution. Correlation is always computed for two sequences. Correlation provides the information about similarity between the two sequences. Correlation is used in many applications where digital signal extraction is required. Such applications are Digital Communication systems, Radars, Spread spectrum communications, Mobile communications etc. In such applications the incoming / received signal is correlated with the standard set of signals. From this set a signal is selected, which shows maximum correlation with the incoming / received signal. Thus the received signal is identified or detected. This is the main application of correlation.

2.9.1 Cross-correlation and Auto-correlation

The correlation can be of two types cross correlation and auto correlation. When the correlation of two different sequences $x(n)$ and $y(n)$ is performed, it is called cross correlation. Crosscorrelation is basically a sequence and it is denoted by $r_{xy}(l)$. It is given as,

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n)y(n-l), \quad l=0, \pm 1, \pm 2, \dots \quad \dots (2.9.1)$$

or

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n+l)y(n), \quad l=0, \pm 1, \pm 2, \dots \quad \dots (2.9.2)$$

In the first equation $y(n)$ is delayed with respect to $x(n)$. And in the second equation $x(n)$ is advanced with respect to $y(n)$. Both of these operations are equivalent. Hence above two equations provide identical crosscorrelation sequences.

The crosscorrelation sequence $r_{yx}(l)$ are defined as follows :

$$r_{yx}(l) = \sum_{n=-\infty}^{\infty} y(n)x(n-l) \quad \dots (2.9.3)$$

or

$$r_{yx}(l) = \sum_{n=-\infty}^{\infty} y(n+l)x(n) \quad \dots (2.9.4)$$

The autocorrelation of $x(n)$ is denoted by $r_{xx}(l)$ and it is given as,

$$r_{xx}(l) = \sum_{n=-\infty}^{\infty} x(n)x(n-l) \quad \dots (2.9.5)$$

or

$$r_{xx}(l) = \sum_{n=-\infty}^{\infty} x(n+l)x(n) \quad \dots (2.9.6)$$

Both of the above equations are written from definitions of crosscorrelation with $x(n)=y(n)$.

If $x(n)$ and $y(n)$ are causal sequences of length 'N', then crosscorrelation and auto-correlation sequences are given as,

$$r_{xy}(l) = \sum_{n=i}^{N-|k|-1} x(n)y(n-l) \quad \dots (2.9.7)$$

and

$$r_{xx}(l) = \sum_{n=i}^{N-|k|-1} x(n)x(n-l) \quad \dots (2.9.8)$$

Here $i = l$ and $k = 0$ for $l \geq 0$
and $i = 0$ and $k = l$ for $l < 0$

Ex. 2.9.1 Prove the following

(i) $r_{xy}(l) = r_{yx}(-l)$ i.e. correlation is not commutative

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

Sol. : (i) Consider equation 2.9.1,

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

Since product $x(n)y(n-l) = y(n-l)x(n)$ we can write above equation as,

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

We can write $y(n-l)$ as $y[n+(-l)]$ i.e.,

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

Compare the RHS of above equation with RHS of equation 2.9.4. We have,

$$r_{xy}(l) = r_{yx}(-l) \quad \text{from equation 2.9.4} \quad \dots (2.9.9)$$

This proves that correlation is not commutative.

(ii) Consider the linear convolution of two sequences $x(n)$ and $y(n)$. i.e.,

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

Since 'n' is just the index, we can write 'l' in place of 'n'. i.e.,

$$x(l) * y(l) = \sum_{k=-\infty}^{\infty} x(k)y(l-k)$$

Again 'k' is just the index. We can write 'n' in place of 'k'. i.e.,

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

This is the convolution of two sequences $x(l)$ and $y(l)$. If we fold $y(l)$, then folded sequence becomes $y(-l)$. The convolution equation for $x(l)$ and $y(-l)$ can be obtained by folding $y(l)$ in above equation. i.e.,

$$x(l) * y(-l) = \sum_{n=-\infty}^{\infty} x(n)y[-(l-n)]$$

In the above equation observe that we have inverted the signs of indices of 'y'. The above equation (RHS) can be further simplified as,

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

The RHS of above equation is basically $r_{xy}(l)$ of equation 2.9.1. Thus we have,

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

or $r_{xy}(l) = x(l) * y(-l) \quad \dots (2.9.10)$

Thus cross correlation of $x(n)$ and $y(n)$ is equivalent to convolution of $x(n)$ and $y(-n)$. Here $y(-n)$ is obtained by folding $y(n)$.

Ex. 2.9.2 Determine the crosscorrelation sequence $r_{xy}(l)$ of the following sequences :

$$x(n) = \{2, -1, 3, 7, 1, 2, -3\}$$

↑

$$y(n) = \{1, -1, 2, -2, 4, 1, -2, 5\}$$

↑

[Dec - 98]

Sol. : Here given sequences are,

$$\begin{aligned} x(-4) &= 2 & y(-4) &= 1 \\ x(-3) &= -1 & y(-3) &= -1 \\ x(-2) &= 3 & y(-2) &= 2 \\ x(-1) &= 7 & y(-1) &= -2 \\ x(0) &= 1 \leftarrow & y(0) &= 4 \leftarrow \\ x(1) &= 2 & y(1) &= 1 \\ x(2) &= -3 & y(2) &= -2 \\ & & y(3) &= 5 \end{aligned}$$

The cross correlation sequence $r_{xy}(l)$ is given by equation 2.9.1 as,

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

For $x(n)$, the index ' n ' varies from -4 to 2 ,

Hence above equation becomes,

$$r_{xy}(l) = \sum_{n=-4}^2 x(n)y(n-l)$$

The above equation can be expanded as,

$$\begin{aligned} r_{xy}(l) &= x(-4)y(-4-l) + x(-3)y(-3-l) + x(-2)y(-2-l) \\ &\quad + x(-1)y(-1-l) + x(0)y(-l) + x(1)y(1-l) \\ &\quad + x(2)y(2-l) \end{aligned} \quad \dots (2.9.11)$$

In the above equation all the ' y ' terms are 'zero' for $l \leq -8$. Hence starting from $l = -7$,

$$\begin{aligned} r_{xy}(-7) &= x(-4)y(-4+7) + x(-3)y(-3+7) + x(-2)y(-2+7) \\ &\quad + x(-1)y(-1+7) + x(0)y(7) + x(1)y(1+7) + x(2)y(2+7) \\ &= x(-4)y(3) + x(-3)y(4) + x(-2)y(5) + x(-1)y(6) \\ &\quad + x(0)y(7) + x(1)y(8) + x(2)y(9) \\ &= x(-4)y(3) \quad \text{since } y(4) = y(5) = \dots = y(9) = 0 \\ &= 2 \times 5 = 10 \end{aligned}$$

$$\boxed{r_{xy}(-7) = 10}$$

Putting $l = -6$ in equation 2.9.11 we get,

$$\begin{aligned} r_{xy}(-6) &= x(-4)y(2) + x(-3)y(3), \quad \text{since remaining terms are zero} \\ &= 2 \times (-2) + (-1) \times 5 = -9 \end{aligned}$$

$$\boxed{r_{xy}(-6) = -9}$$

On the similar lines other values of $r_{xy}(l)$ can be obtained by putting $l = -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5$, and 6 in equation 2.9.11. These values are,

$$\begin{aligned}
 r_{xy}(-5) &= 19, & r_{xy}(-4) &= 36, & r_{xy}(-3) &= -14 \\
 r_{xy}(-2) &= 33, & r_{xy}(-1) &= 0, & r_{xy}(0) &= 7 \\
 r_{xy}(1) &= 13, & r_{xy}(2) &= -18, & r_{xy}(3) &= 16 \\
 r_{xy}(4) &= -7, & r_{xy}(5) &= 5, & r_{xy}(6) &= -3
 \end{aligned}$$

If we put $l=7$ in equation 2.9.11, all the 'y' terms will be zero. Thus,

$$r_{xy}(l) = 0 \text{ for } l \geq 7 \text{ and } l \leq -8.$$

Thus the crosscorrelation sequence is,

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

↑

... (2.9.12)

To obtain correlation sequence using convolution :

From equation 2.9.10 we know that,

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

Here cross correlation sequence is equal to convolution of $x(l)$ and $y(-l)$. And $y(-l)$ is obtained by folding $y(l)$.

The sequence $y(l)$ is,

$$y(l) = \{1, -1, 2, -2, 4, 1, -2, 5\}$$

↑

The folded sequence $y(-l)$ becomes,

$$y(-l) = \{5, -2, 1, 4, -2, 2, -1, 1\}$$

↑

Here observe that the folding is done around $l=0$. And $y(-4)$ becomes $y(4)$ in the folded sequence and so on. Fig. 2.9.1 shows the convolution of $x(l)$ and $y(-l)$ using multiplication method.

$x(l) \Rightarrow$	2	-1	3	7	1	2	-3	
					↓			
$y(l) \Rightarrow$	5	-2	1	4	-2	2	-1	1
			↑					
		2	-1	3	7	1	2	-3
		-2	1	-3	-7	-1	-2	3
	4	-2	6	14	2	4	-6	x
-4	2	-6	-14	-2	-4	6	x	x
8	-4	12	28	4	8	-12	x	x
2	-1	3	7	1	2	-3	x	x
-4	2	-6	-14	-2	-4	6	x	x
10	-5	15	35	5	10	-15	x	x
$r_{xy}(l) \Rightarrow$	10	-9	19	36	-14	33	0	7
							↑	

Fig. 2.9.1 Computation of cross correlation using convolution

From Fig. 2.9.1 the cross correlation sequence obtained by convolution is,

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

↑

Observe that the above sequence is the same as one obtained earlier in equation 2.9.12. There are 4 samples of $x(l)$ before ↑ and 3 samples of $y(-l)$ before ↑. Hence there are $4+3=7$ samples in $r_{xy}(l)$ before ↑ as computed in Fig. 2.9.1.

Ex. 2.9.3 Determine the autocorrelation sequences of the following signals :

(i) $x(n) = \{1, 2, 1, 1\}$

↑

(ii) $y(n) = \{1, 1, 2, 1\}$

↑

What is your conclusion ?

[Dec - 99]

Sol. : The autocorrelation of the sequence is given by equation 2.9.5 as,

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

Using equation 2.9.10 we can write above equation as,

$$r_{xx}(l) = x(l) * x(-l) \dots (2.9.13)$$

This equation shows that autocorrelation of the sequence is obtained by convolving the sequence with its folded version.

Consider the first sequence $x(n)$. i.e.,

$$x(l) = \{1, 2, 1, 1\}$$

↑

The folded sequence $x(-l)$ becomes,

$$x(-l) = \{1, 1, 2, 1\}$$

↑

The above sequence $x(-l)$ is obtained by folding $x(l)$ around $l=0$. Fig. 2.9.2 shows the convolution of $x(l)$ and $x(-l)$ using multiplication method.

$x(l) \Rightarrow$	↓	1	2	1	1			
$x(-l) \Rightarrow$		1	1	2	1	↑		
			1	2	1	1		
			2	4	2	2	×	
		1	2	1	1	×	×	
		1	2	1	1	×	×	
$r_{xx}(l) \Rightarrow$		1	3	5	7	5	3	1
					↑			

Fig. 2.9.2 Computation of autocorrelation

As shown in above figure, the autocorrelation sequence $r_{xx}(l)$ is,

$$r_{xx}(l) = \{1, 3, 5, 7, 5, 3, 1\} \quad \dots (2.9.14)$$

From equation 2.9.9 we know that,

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

Then for auto correlation sequence above equation becomes,

$$r_{xx}(l) = r_{xx}(-l) \quad \dots (2.9.15)$$

Observe that above relation is satisfied by auto correlation sequence of equation 2.9.14.

Now let us consider the second sequence,

$$y(n) = y(l) = \{1, 1, 2, 1\}$$

The folded sequence $y(-l)$ can be obtained as,

$$y(-l) = \{1, 2, 1, 1\}$$

Hence the auto correlation sequence can be obtained using equation 2.9.13 as,

$$r_{yy}(l) = y(l) * y(-l)$$

If we perform the convolution of $y(l)$ and $y(-l)$

We get,

$$r_{yy}(l) = \{1, 3, 5, 7, 5, 3, 1\}$$

Observe that $r_{yy}(l)$ obtained above is same as $r_{xx}(l)$ of equation 2.9.14. This is because $y(n) = x(-n+3)$. This is equivalent to reversing the sequence $x(n)$. Hence the autocorrelation sequences are same.

2.9.2 Properties of Crosscorrelation and Autocorrelation Sequences

Here we will see some of the important properties of correlation sequences.

1. The crosscorrelation is not commutative. i.e.,

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

We have proved this property earlier.

2. The autocorrelation sequence is an even function. i.e.,

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

This property follows from the first property when $x(n) = y(n)$.

3. The crosscorrelation is equivalent to convolution of one sequence with folded version of another sequence. i.e.,

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

This property also we have proved earlier.

4. The autocorrelation sequence attains maximum value at zero lag i.e. $l=0$. i.e.,

$$|r_{xx}(l)| \leq r_{xx}(0) = E_x \quad \dots (2.9.16)$$

This can be proved very easily. Consider the equation for autocorrelation sequence. i.e.,

$$r_{x x}(l) = \sum_{n=-\infty}^{\infty} x(n)x(n-l) \quad \text{By equation 2.9.5}$$

With zero lag i.e. $l=0$ above equation becomes,

$$\begin{aligned} r_{x x}(l) &= \sum_{n=-\infty}^{\infty} x^2(n) \\ &= E_x \end{aligned}$$

5. The crosscorrelation sequence satisfies the condition of,

$$|r_{x y}(l)| \leq \sqrt{r_{x x}(0)r_{y y}(0)} = \sqrt{E_x E_y}$$

The above equation gives upper limit on $r_{x y}(l)$.

6. The shape of the crosscorrelation sequence depends upon shapes of $x(n)$ and $y(n)$, rather their instantaneous amplitudes.

2.10 A/D Conversion Process

Most of the signals are analog in nature when they are generated from the primary source. Analog signals are continuous in time and amplitude. The signals such as speech, video, radar, seismic signals, ECG, EMG etc signals are basically analog in nature. If these signals are to be processed digitally, they should be converted to digital form. Such conversion is performed by Analog to Digital (A/D) converters. The A/D converters are available with standard specifications in the market. The output digital signal is converted back to analog form by Digital to Analog (D/A) converters. Thus A/D and D/A converters are always used with DSP systems.

Fig. 2.10.1 illustrates the process of A/D conversion in brief. It consists of three basic block of A/D converter.



Fig. 2.10.1 Basic block diagram of A/D converter

We will briefly study these three basic operations of A/D conversion in following subsections.

2.10.1 Sampling

The continuous time signal $x_a(t)$ is converted to discrete time signal $x(n)$ by sampling. The sampler takes the samples at regular time intervals. The sampling interval is denoted by T . Then continuous time variable t and discrete time variable n are related as,

$$t = nT \quad n = 0, 1, 2, 3, \dots \quad \dots (2.10.1)$$

This means $x_a(t)$ is represented in its samples at $0, T, 2T, 3T, 4T, \dots$. Hence $x_a(t)$ and $x(n)$ are related as,

$$x_a(t) = x_a(nT) = x(n) \quad n = 0, 1, 2, 3, \dots \quad \dots (2.10.2)$$

Thus discrete time signal is discrete time signal $x(n)$ is defined only at $n = 0, 1, 2, 3, \dots$. And its value is equal to $x_a(nT)$. Here T is sampling interval.

The sampler is normally a switch and it chops off the incoming analog signal. It is illustrated in Fig. 2.10.2. As shown in the figure, the sampler switch operates at the sampling rate of F_s , which is given as,

$$\text{Sampling rate } F_s = \frac{1}{\text{Sampling interval } T} \quad \dots (2.10.3)$$

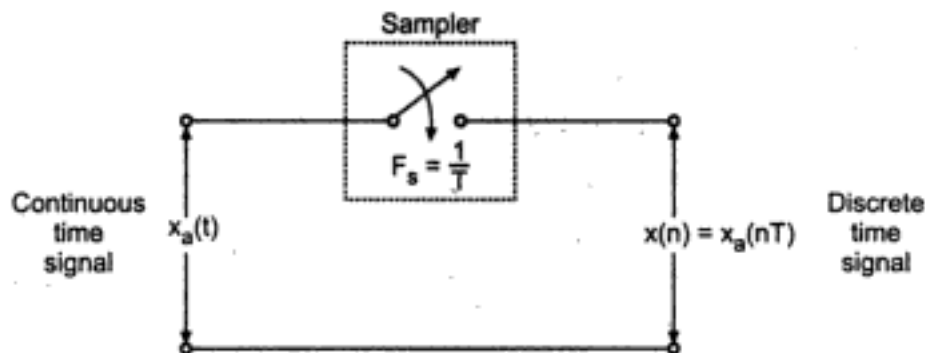


Fig. 2.10.2 Sampler

Here ' F_s ' is also called sampling frequency. Thus ' F_s ' can be expressed as samples per second or in hertz. The sampling interval ' T ' is in seconds. Fig. 2.10.3 shows the input signal $x_a(t)$ and its sampled version $x(nT)$ or $x(n)$. In this figure observe that $x(n)$ takes the

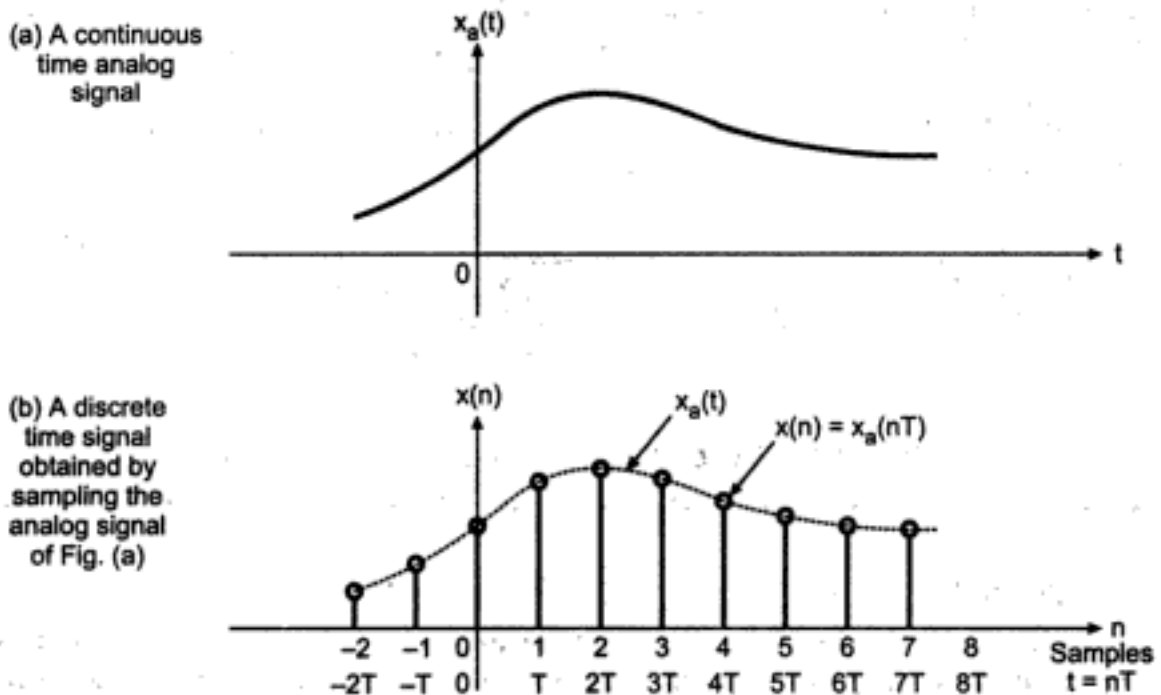


Fig. 2.10.3 Sampling of the analog signals

amplitude of $x_a(t)$ at the sampling instants $t = nT$. The values of ' n ' can be only positive for real time processing. But ' n ' is considered in the range $-\infty < n < \infty$ for mathematical simplicity. Hence we can rewrite equation 2.10.2 as,

$$x(n) = x_a(nT) \quad -\infty < n < \infty \quad \dots (2.10.4)$$

Since $t = nT$ and $F_s = \frac{1}{T}$ we can write an important relationship as,

$$t = nT = \frac{n}{F_s} \quad \dots (2.10.5)$$

This equation gives the relationship between time t , samples ' n ' and sampling frequency F_s .

2.10.2 Frequency Relationships

Here we relate the continuous time frequencies (F, Ω) with the discrete time frequencies (f, ω) in terms of sampling frequency (F_s). This relationship is extremely important for digital filter design. It is also useful for the conversion of discrete and continuous time domains. Consider the analog sinusoidal signal,

$$x_a(t) = A \cos(2\pi Ft + \theta) \quad \dots (2.10.6)$$

Let this signal be sampled at the regular intervals. Hence $x(n)$ can be obtained by putting $t = nT$ in the above equation. i.e.,

$$\begin{aligned} x(n) &= x_a(t)|_{t=nT} = x_a(nT) \\ &= A \cos(2\pi F n T + \theta) \end{aligned} \quad \dots (2.10.7)$$

The sampling frequency $F_s = \frac{1}{T}$ or $T = \frac{1}{F_s}$, hence above equation becomes,

$$x(n) = A \cos\left(2\pi n \frac{F}{F_s} + \theta\right) \quad \dots (2.10.8)$$

Earlier we have represented discrete time signal $x(n)$. It is given by equation 2.2.13 as,

$$x(n) = A \cos(2\pi fn + \theta) \quad \dots (2.10.9)$$

On comparing above equation with equation 2.10.8 we obtain,

$$f = \frac{F}{F_s} = FT \quad \dots (2.10.10)$$

Since $2\pi F = \Omega$, equation 2.10.8 becomes,

$$x(n) = A \cos\left(n \frac{\Omega}{F_s} + \theta\right) \quad \dots (2.10.11)$$

The similar equation we have obtained earlier. It is given by equation 2.2.11 as,

$$x(n) = A \cos(\omega n + \theta) \quad \dots (2.10.12)$$

On comparing above equation with equation 2.10.11 we obtain,

$$\omega = \frac{\Omega}{F_s} = \Omega T \quad \dots (2.10.13)$$

The above equation can be directly obtained from equation 2.10.10 by simply multiplying both sides by 2π and letting $\omega = 2\pi f$ and $\Omega = 2\pi F$. We know that the continuous time frequencies have infinite range of distinct frequencies. i.e.,

$$-\infty < F < \infty \quad \text{and} \quad -\infty < \Omega < \infty \quad \dots (2.10.14)$$

In the conclusion of example 2.2.3 we have seen that the maximum range of discrete time frequencies is,

$$-\frac{1}{2} \leq f \leq \frac{1}{2} \quad \text{and} \quad -\pi \leq \omega \leq \pi \quad \dots (2.10.15)$$

This maximum frequency is also illustrated for the sinusoidal signal in Fig. 2.2.10 also. [Readers are advised to study example 2.2.2 in detail]. From equation 2.10.10 we know that $f = \frac{F}{F_s}$. Hence above equation becomes,

$$-\frac{1}{2} \leq \frac{F}{F_s} \leq \frac{1}{2}$$

$$\therefore -\frac{F_s}{2} \leq F \leq \frac{F_s}{2} \dots (2.10.16)$$

This equation shows that the maximum range of frequencies that can be represented by the discrete time signal is from $-\frac{F_s}{2}$ to $\frac{F_s}{2}$. Or highest signal frequency that can be represented by discrete time signal if sampling frequency is F_s , will be,

$$F_{\max} = \frac{F_s}{2} = \frac{1}{2T} \dots (2.10.17)$$

Since $\Omega = 2\pi F$, equation 2.10.16 will be,

$$-\frac{F_s}{2} \leq \frac{\Omega}{2\pi} \leq \frac{F_s}{2}$$

$$\therefore -\pi F_s \leq \Omega \leq \pi F_s \dots (2.10.18)$$

And maximum frequency Ω_{\max} is,

$$\Omega_{\max} = \pi F_s = \frac{\pi}{T} \dots (2.10.19)$$

The various frequencies, their ranges and conversions are summarized in Table 2.10.1.

Table 2.10.1 : Summary of continuous and discrete time frequency relationships

Continuous time frequencies	Ω, F
Discrete time frequencies	ω, f
Sampling frequency	F_s or $\frac{1}{T}$
Conversion relations	$\omega = \frac{\Omega}{F_s} = \Omega T$ $f = \frac{F}{F_s} = FT$
Range of continuous time frequencies	$-\infty < \Omega < \infty$ $-\infty < F < \infty$
Range of discrete time frequencies	$-\frac{1}{2} \leq f \leq \frac{1}{2}$ $-\pi \leq \omega \leq \pi \text{ or}$ $-\frac{F_s}{2} \leq F \leq \frac{F_s}{2}$ $-\pi F_s \leq F \leq \pi F_s$

Here we have concluded that if signal frequency lies in the range $-\frac{F_s}{2} \leq F \leq \frac{F_s}{2}$, it can be represented by discrete time signal. Now let us see what happens if signal frequency $F > \frac{F_s}{2}$. This concept is illustrated with the help of an example considered next.

Ex.2.10.1 *The following four analog sinusoidal signals are sampled with the sampling frequency of 40 Hz. Find out the corresponding discrete time signals and comment on the result.*

$$x_1(t) = \cos 2\pi(10)t$$

$$x_2(t) = \cos 2\pi(50)t$$

$$x_3(t) = \cos 2\pi(90)t$$

$$x_4(t) = \cos 2\pi(130)t$$

Sol. : Here clearly observe that the signal frequencies are,

$$\text{Frequency of } x_1(t) \text{ is } F_1 = 10 \text{ Hz}$$

$$\text{Frequency of } x_2(t) \text{ is } F_2 = 50 \text{ Hz}$$

$$\text{Frequency of } x_3(t) \text{ is } F_3 = 90 \text{ Hz}$$

$$\text{Frequency of } x_4(t) \text{ is } F_4 = 130 \text{ Hz}$$

Since the sampling frequency $F_s = 40 \text{ Hz}$, the maximum discrete time frequency that can be represented properly is $\frac{F_s}{2}$ i.e. 20 Hz. So here F_1 will be represented properly and let us see

what happens for F_2 , F_3 and F_4 . We know that $x(n)$ is obtained by putting $t = \frac{n}{F_s}$ in $x(t)$.

Here we put $t = \frac{n}{40}$ in given analog signal equations. Consider $x_1(t)$; we obtain $x_1(n)$ by

putting $t = \frac{n}{F_s} = \frac{n}{40}$, hence,

$$\begin{aligned} x_1(n) &= \cos 2\pi 10 \cdot \frac{n}{40} \\ &= \cos 2\pi \left(\frac{10}{40} \right) n \\ &= \cos 2\pi \left(\frac{1}{4} \right) n \end{aligned} \quad \dots (2.10.20)$$

Comparing this equation with discrete time equation $x(n) = \cos 2\pi f n$ we get,

$$f_1 = \frac{1}{4} \text{ cycles / sample} \quad \dots (2.10.21)$$

Note that this frequency is less than $f \leq \frac{1}{2}$. Hence it will be represented properly.

Now consider the signal $x_2(t)$. Its sampled version at $F_s = 40 \text{ Hz}$ becomes,

$$\begin{aligned} x_2(n) &= \cos 2\pi 50 \cdot \frac{n}{40} \\ &= \cos 2\pi \frac{5}{4} n \end{aligned} \quad \dots (2.10.22)$$

Here frequency of $x_2(n)$ is $f_2 = \frac{5}{4}$ cycles / sample. Since this frequency is greater than $\frac{1}{2}$, it will not be represented properly in discrete time domain. We can write above equation as,

$$\begin{aligned} x_2(n) &= \cos 2\pi \frac{5}{4} n \\ &= \cos 2\pi \left(1 + \frac{1}{4}\right) n \\ &= \cos \left(2\pi n + 2\pi \cdot \frac{1}{4} n\right) \end{aligned}$$

Since $\cos(2\pi n + \phi) = \cos \phi$, the above equation becomes,

$$\begin{aligned} x_2(n) &= \cos 2\pi \cdot \frac{1}{4} n \\ &= x_1(n) \quad \text{from equation 2.10.20} \end{aligned}$$

Thus signal $x_2(n)$ is same as $x_1(n)$.

Now let us consider the signal $x_3(t)$. Its sampled version at $F_s = 40$ Hz becomes,

$$\begin{aligned} x_3(n) &= \cos 2\pi 90 \cdot \frac{n}{40} \\ &= \cos 2\pi \frac{9}{4} n \quad \dots (2.10.23) \end{aligned}$$

Here $f_3 = \frac{9}{4}$. Again this frequency is greater than $\frac{1}{2}$. Hence it will not be represented properly. Let us rearrange the above equation as,

$$\begin{aligned} x_3(n) &= \cos 2\pi \frac{9}{4} n \\ &= \cos 2\pi \left(2 + \frac{1}{4}\right) n \\ &= \cos \left(2\pi \cdot 2n + 2\pi \frac{1}{4} n\right) \end{aligned}$$

Since $\cos(4\pi n + \phi) = \cos \phi$, above equation becomes,

$$\begin{aligned} x_3(n) &= \cos 2\pi \frac{1}{4} n \\ &= x_1(n) \quad \text{from equation 2.10.20} \end{aligned}$$

Thus $x_3(n)$ is same as $x_1(n)$.

Consider $x_4(t)$. Its sampled sequence at $F_s = 40$ Hz becomes,

$$\begin{aligned} x_4(n) &= \cos 2\pi 130 \frac{n}{40} \\ &= \cos 2\pi \frac{13}{4} n \quad \dots (2.10.24) \end{aligned}$$

Here $f_4 = \frac{13}{4}$ which is greater than $\frac{1}{2}$. Hence this frequency will not be represented properly.

Let us rearrange above equation as,

$$\begin{aligned}
 x_4(n) &= \cos 2\pi \cdot \frac{13}{4} n \\
 &= \cos 2\pi \left(3 + \frac{1}{4} \right) n \\
 &= \cos \left(2\pi \cdot 3n + 2\pi \cdot \frac{1}{4} n \right)
 \end{aligned}$$

Here $\cos(2\pi kn + \phi) = \cos \phi$ for $k = 0, 1, 2, 3, \dots$. Hence above equation becomes

$$\begin{aligned}
 x_4(n) &= \cos 2\pi \frac{1}{4} n \\
 &= x_1(n)
 \end{aligned}$$

Thus $x_4(n)$ is same as $x_1(n)$.

Comments :

1. Here all the four signals have same discrete time sequence. i.e. $x_1(n) = x_2(n) = x_3(n) = x_4(n)$. This happens because f_2, f_3 and f_4 are greater than $\frac{1}{2}$, which is maximum discrete time frequency that can be represented.
2. The relationship between F_2, F_3 and F_4 and F_s are as follows,

$$F_2 = F_1 + F_s = 10 + 40 = 50 \text{ Hz}$$

$$F_3 = F_1 + 2F_s = 10 + 2 \times 40 = 90 \text{ Hz}$$

$$F_4 = F_1 + 3F_s = 10 + 3 \times 40 = 130 \text{ Hz}$$

These F_2, F_3, F_4 are called *alias* of F_1 at $F_s = 40$ Hz.

This can be generalized further :

All the frequencies of $(F_1 + kF_s), k = 1, 2, 3, 4, \dots$ will be *alias* of F_1 at sampling frequency of F_s .

2.10.3 Aliasing

The concept of aliasing was introduced in last example. The maximum signal frequency F_{\max} , that can be represented properly in discrete time domain is given as,

$$F_{\max} = \frac{F_s}{2} = \frac{1}{2T} \quad \text{By equation 2.10.17}$$

Corresponding to this, there is maximum discrete time frequency which is given as,

$$f_{\max} = \frac{1}{2}$$

If the maximum signal frequency is more than $\frac{F_s}{2}$, then the corresponding discrete time frequency is more than $\frac{1}{2}$. As a result of this *aliasing* occurs. In aliasing, the higher frequencies take the form of lower frequencies. In example 2.10.1 we observed that all the signal frequencies $(F_1 + kF_s), k = 1, 2, 3, \dots$ are represented as F_1 only. This can be verified easily,

$$x_k(t) = A \cos [2\pi(F_1 + kF_s)t], \quad k = 1, 2, 3, \dots$$

The corresponding sampled version of this signal becomes,

$$\begin{aligned}
 x_k(n) &= A \cos \left[2\pi \frac{F_1 + k F_s}{F_s} n \right] \quad \text{By putting } t = \frac{n}{F_s} \\
 &= A \cos \left[2\pi kn + 2\pi \frac{F_1}{F_s} n \right]
 \end{aligned}$$

Since $\cos(2\pi kn + \phi) = \cos \phi$, the above equation becomes,

$$x_k(n) = A \cos \left(2\pi \frac{F_1}{F_s} n \right)$$

This shows that all sequences $x_k(n), k=1, 2, 3, \dots$ have same frequency of F_1 Hz. Hence all the frequencies of $F_1 + F_s, F_1 + 2F_s, F_1 + 3F_s, \dots$ etc are called alias frequencies of F_1 at sampling frequency F_s . Such alias frequencies are infinite. It can be verified easily that $F_1 - F_s, F_1 - 2F_s, F_1 - 3F_s, \dots$ etc are also alias frequencies of F_1 at sampling frequency F_s . This concept is illustrated in Fig. 2.10.4. This figure shows that the relationship between 'f' and 'F' is linear in the range $-\frac{F_s}{2} \leq F < \frac{F_s}{2}$. And there is no ambiguity or aliasing in this range. Here $\frac{F_s}{2}$ is also called as folding frequency, since all frequencies are reflected or folded with respect to this frequency.

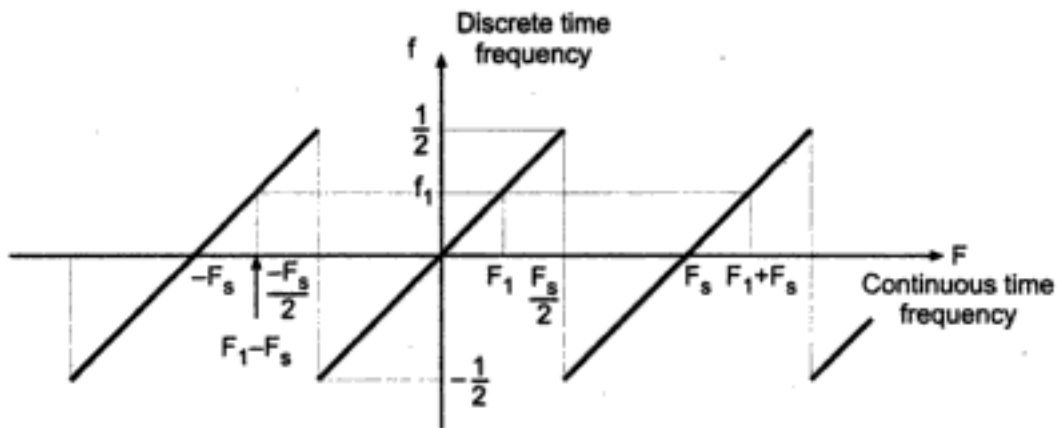


Fig. 2.10.4 Frequencies $F_1, F_1 + F_s, F_1 - F_s$ etc are represented by same frequency f_1 . Actually f_1 represents $(F_1 + k F_s), k = \pm 1, \pm 2, \pm 3, \dots$, such infinite number of frequencies

2.10.4 Quantization

When the analog signal is converted to discrete time signal, it is given to quantizer. The quantizer converts the continuous amplitude signal to discrete amplitude signal. Each sample is represented by fixed number of digits in the processor. These digits determine the discrete amplitude levels. For example if the processor uses 4 bits to represent data, then it will have $2^4 = 16$ different levels. Consider the quantization of the following signal,

$$\left. \begin{aligned}
 x(n) &= 10(0.9)^n \quad \text{for } n \geq 0 \\
 &= 0 \quad \text{for } n < 0
 \end{aligned} \right\} \dots (2.10.25)$$

This signal is obtained by sampling $x_a(t) = 10(0.9)^t$ with sampling frequency of $F_s = 1 \text{ Hz}$. First 10 sample values of $x(n)$ are given in Table 2.10.2. The value of $x(n)$ is then quantized to nearest integer by rounding operation. This quantized signal is denoted by $x_q(n)$ and shown in table 2.10.2. Observe that $x_q(n)$ has fixed amplitude levels. Fig. 2.10.5 shows this quantization operation graphically. This figure shows the continuous time signal $x_a(t) = 10(0.9)^t$. The figure also shows the sampled signal $x(n) = 10(0.9)^n$ with (·) such dots on the curve of $x_a(t)$. The samples of $x(n)$ are quantized to nearest integer by rounding operation. This quantized signal $x_q(n)$ is shown in Table 2.10.2. Fig. 2.10.5 shows the quantized samples of $x_q(n)$ by (·) such dots. Observe that quantizer has 16 levels (0 to 15).

Please refer Fig. 2.10.5 on next page.

Table 2.10.2 Quantization and encoding of $x(n) = 10(0.9)^n$ into four digits

n	Discrete time signal $x(n) = 10(0.9)^n$	Quantized signal $x_q(n)$ by rounding	Quantization error $e_q(n) = x_q(n) - x(n)$	Encoded digital signal
0	10	10	0	1010
1	9	9	0	1001
2	8.1	8	- 0.1	1000
3	7.29	7	- 0.29	0111
4	6.561	7	0.439	0111
5	5.9049	6	0.0951	0110
6	5.31441	5	- 0.31441	0101
7	4.782969	5	0.217031	0101
8	4.3046721	4	- 0.3046721	0100
9	3.87420489	4	0.12579511	0100
:	:	:	:	:

The difference between the two quantization levels is called as *quantization step* or *resolution*. The quantization step is denoted by Δ . As shown in the figure the quantization step is,

$$\text{Quantization step } \Delta = 1$$

The *quantization error* $e_q(n)$ is defined as,

$$\text{Quantization error } e_q(n) = x_q(n) - x(n) \quad \dots (2.10.26)$$

The quantization error in rounding operation is limited to $\left| \frac{\Delta}{2} \right|$. Thus,

$$\text{Maximum quantization error } e_q(n)_{\max} = \left| \frac{\Delta}{2} \right| \quad \dots (2.10.27)$$

This is clear from the calculated values of $e_q(n)$ in Table 2.10.2. Here since $\Delta = 1$. The maximum quantization error is,

$$e_q(n)_{\max} = \left| \frac{\Delta}{2} \right| = \left| \frac{1}{2} \right|$$

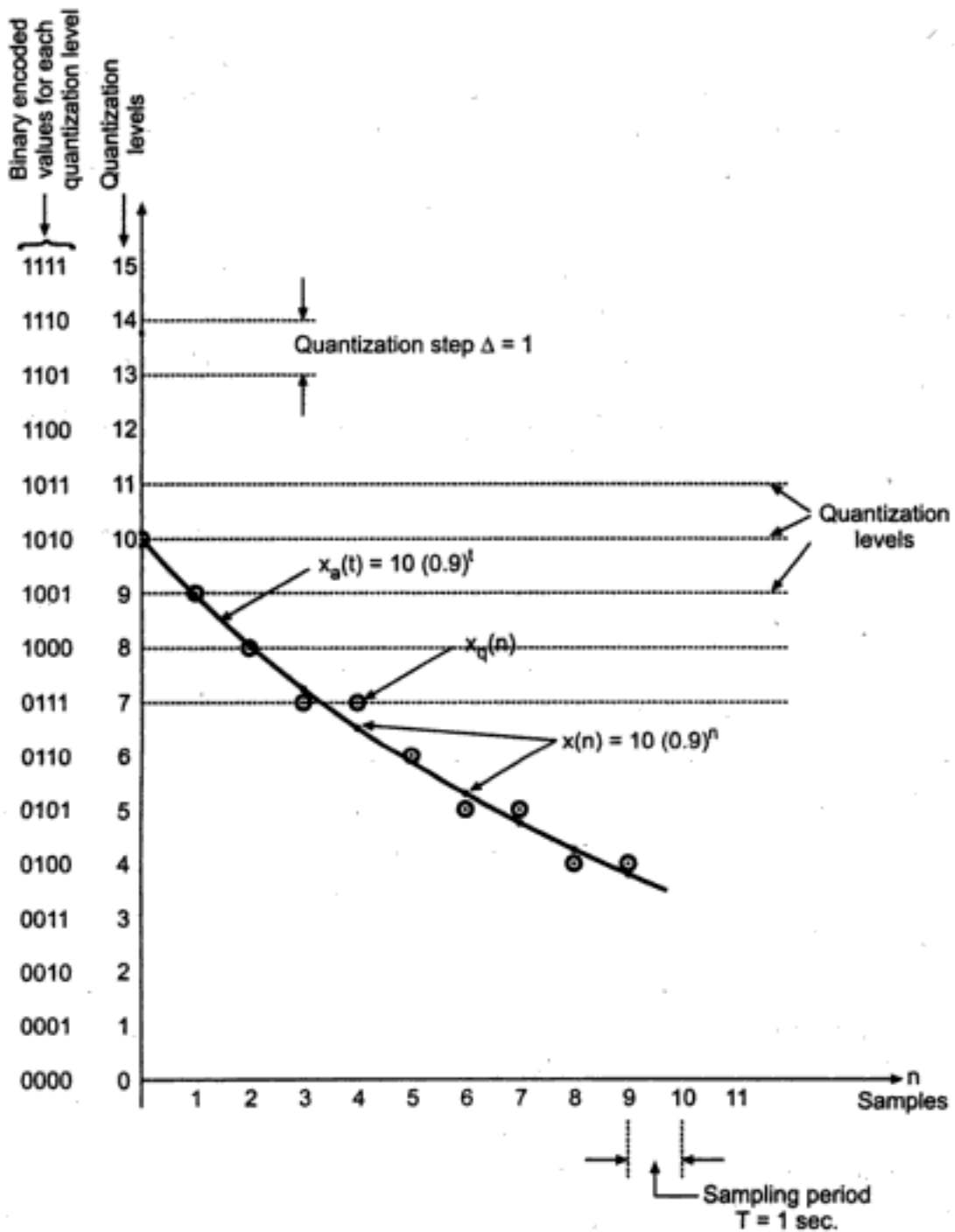


Fig. 2.10.5 Quantization and encoding in A/D converter

Thus error is less than ± 0.5 .

In general, the quantization step is given as,

$$\Delta = \frac{x_{\max} - x_{\min}}{L - 1} \quad \dots (2.10.28)$$

Here Δ is quantization step or resolution.

x_{\max} is maximum value to be represented

x_{\min} is minimum value to be represented

L is number of levels.

Here, from Fig. 2.10.5 we have following values of above quantities, $\Delta = \frac{15-0}{16-1} = 1$ which is same as earlier.

If number of levels are increased, then there is small quantization step. This results in small quantization error, and the signal is represented more accurately.

In quantization operation, the signal is converted to a fixed quantization level. This conversion is done either by truncation or rounding operation. This always results in net loss of information. This loss cannot be recovered. Hence quantization process is irreversible process. The only way to reduce this loss is to increase the number of quantization levels. For every application the ratio of signal to quantization error is specified, such that information loss is within the tolerable limits.

2.10.5 Encoding

After quantization, the next operation is encoding. Each quantization level is assigned a unique binary code. In the encoding operation, the quantized sample value is converted to the binary equivalent of that quantization level. Fig. 2.10.5 shows the binary values for quantization levels. Since there are 16 quantization levels, 4 bits are required. Thus each quantized sample is converted to four binary digits. Table 2.10.2 shows the digital encoded signal for each sample of $x(n)$. The bits are denoted by 'b' and they depend upon the quantization levels. i.e.,

$$2^b \geq L \quad \dots (2.10.29)$$

or $b \geq \log_2 L \quad \dots (2.10.30)$

Thus the number of binary digits required are smallest integer greater than or equal to $\log_2 L$. In Fig. 2.10.5, $L=16$ hence $b = \log_2 16 = 4$ bits. The standard A/D converters are available with 8, 16 or 24 bits. Obviously, a 24 bit A/D converter will have better resolution compared to 16 bit one. The section of the bit length is done on the type of signal to be coded.

Normally sampling, quantization and encoding operations are performed by the single chip or device and they are commonly called A/D converters.

2.10.6 Sampling Theorem

The sampling theorem provides the standard mathematical base for sampling and reconstructing the signals. Selection of the sampling frequency F_s is important issue. This frequency is related to frequency content of the signal to be sampled. If maximum frequency content of the signal is known, then it is possible to select sampling frequency correctly. Earlier in this section we have established that the maximum frequency F_{\max} of the signal should be less than or equal to half of sampling frequency F_s i.e.,

$$F_{\max} \leq \frac{F_s}{2} \quad \text{from equation 2.10.17} \quad \dots (2.10.31)$$

This means the sampling frequency should be at least double of the maximum signal frequency. Let us consider the more general range of signal frequencies,

$$-\frac{F_s}{2} \leq F \leq \frac{F_s}{2} \quad \text{from equation 2.10.16} \quad \dots (2.10.32)$$

We have seen that the signal frequencies in this range are represented in discrete time *without ambiguity*. But if the signal frequency is outside this range (see equation 2.10.32), then *aliasing* occurs. Because of aliasing the frequencies are not represented properly and they are lost during sampling. Since negative range of frequencies in equation 2.10.32 are just of mathematical interest, we can state that the sampling frequency should be at least double of the maximum signal frequency i.e.,

$$\text{Sampling frequency } F_s \geq 2 F_{\max} \text{ from equation 2.10.31 and equation 2.10.32} \quad \dots (2.10.33)$$

If the sampling frequency satisfies this relationship, then all the signal frequencies will be represented in discrete time properly (i.e. $-\frac{1}{2} \leq f \leq \frac{1}{2}$) and there will be no ambiguity i.e. no aliasing.

Normally it is possible to detect the maximum signal frequency F_{\max} depending upon the type of the signal. For example speech frequencies are in the range of 50 to 5000 Hz. Hence $F_{\max} = 5 \text{ kHz}$ can be selected for speech. Similarly TV and video signals have the bandwidth of 5 MHz. Hence F_{\max} of 5 MHz can be considered for such signals. Another example is of ECG signal. It contains major components upto 75 Hz. Hence F_{\max} can be selected accordingly. Once the maximum signal frequency is fixed up like this, then the sampling frequency can be taken as $F_s \geq F_{\max}$ to avoid aliasing.

Statement of sampling theorem :

A continuous time signal can be completely represented in its samples and recovered back if the sampling frequency $F_s \geq 2B$. Here F_s is the sampling frequency and B is the maximum frequency present in the signal. That is, $F_{\max} = B$.

The theorem says that the information of the signal is completely represented by its samples if $F_s \geq B$ (Here $F_{\max} = B$).

And also the signal can be obtained back from its samples if $F_s \geq B$ (Here $F_{\max} = B$). Thus there are two parts of sampling theorem. The first part is about sampling and second part is about reconstruction from samples. For the reconstruction of the signal an interpolation function is used.

Nyquist rate :

When the sampling rate becomes exactly equal to $2B$ (i.e. $2F_{\max}$) samples per second for the signal bandwidth of 'B' Hz, then it is called Nyquist rate. Then the nyquist interval is obtained as,

$$\text{Nyquist interval} = \frac{1}{2B} = \frac{1}{2F_{\max}} \quad \dots (2.10.34)$$

$$\text{and Nyquist rate} = 2B = 2F_{\max} \quad \dots (2.10.35)$$

The proof of sampling theorem needs the knowledge of Fourier transform. The proof of sampling theorem is presented in Appendix.

Ex.2.10.2 Consider the signal $x_a(t) = 10 \cos 2\pi(1000)t + 5 \cos 2\pi(5000)t$ is to be sampled.

(i) Determine the Nyquist rate for this signal.

(ii) If the signal is sampled at 4 kHz, will the signal be recovered from its samples?

Sol. : (i) Consider the given signal

$$x_a(t) = 10 \cos 2\pi(1000)t + 5 \cos 2\pi(5000)t$$

Clearly, this signal contains two cosine waves

$$F_1 = 1000 \text{ Hz and } A_1 = 10$$

and

$$F_2 = 5000 \text{ Hz and } A_2 = 5$$

Since the highest frequency in the given signal is $B = F_{\max} = 5000 \text{ Hz}$, the nyquist rate is given from equation 2.10.35 as,

$$\text{Nyquist rate} = 2 F_{\max} = 2 \times 5000 \text{ Hz} = 10 \text{ kHz}$$

(ii) This signal is sampled at $F_s = 4000 \text{ Hz}$. When the signal $x_a(t)$ is sampled, we get $x(n)$ as,

$$\begin{aligned} x(n) &= 10 \cos 2\pi \frac{F_1}{F_s} n + 5 \cos 2\pi \frac{F_2}{F_s} n \\ &= 10 \cos 2\pi \frac{1000}{4000} n + 5 \cos 2\pi \frac{5000}{4000} n \\ &= 10 \cos 2\pi \left(\frac{1}{4}\right) n + 5 \cos 2\pi \left(\frac{5}{4}\right) n \end{aligned}$$

The second term can be rearranged in above equation as,

$$\begin{aligned} x(n) &= 10 \cos 2\pi \left(\frac{1}{4}\right) n + 5 \cos 2\pi \left(1 + \frac{1}{4}\right) n \\ &= 10 \cos 2\pi \left(\frac{1}{4}\right) n + 5 \cos \left(2\pi n + 2\pi \frac{1}{4} n\right) \quad \dots (2.10.36) \end{aligned}$$

Here the second term can be simplified as,

$$\cos(2\pi nk + \phi) = \cos \phi \quad \text{for } k = 1, 2, 3, \dots$$

Hence we can write equation 2.10.36 as,

$$\begin{aligned} x(n) &= 10 \cos 2\pi \left(\frac{1}{4}\right) n + 5 \cos 2\pi \left(\frac{1}{4}\right) n \\ &= 15 \cos 2\pi \left(\frac{1}{4}\right) n \quad \dots (2.10.37) \end{aligned}$$

This is the sampled signal. Observe that there is only one frequency $f = \frac{1}{4}$ in the sampled signal. When this signal is reconstructed we get,

$$x_a(t) = 15 \cos 2\pi(1000)t \quad \text{from equation 2.10.37} \quad \dots (2.10.38)$$

Here observe that the reconstructed signal contains only one frequency of 1000 Hz and amplitude of 15. Thus the effect is that one signal of $F_2 = 5000 \text{ Hz}$ is completely lost and amplitude of $F_1 = 1000 \text{ Hz}$ is increased. This shows that with the sampling rate 4000 Hz, the signal is not recovered from its samples. This is because the sampling theorem is not satisfied. The nyquist rate as calculated is (i) is 10 kHz. Hence minimum sampling frequency should be 10 kHz to avoid aliasing.

2.10.7 Anti Aliasing Filter

When processing the analog signal using DSP system, it is sampled at some rate depending upon the bandwidth. For example if speech signal is to be processed the frequencies upto 3 kHz can be used. Hence the sampling rate of 6 kHz can be used. But the speech signal also contains some frequency components more than 3 kHz. Hence a sampling rate of 6 kHz will introduce aliasing because of such frequency components. Hence it is necessary that the

speech signal should not contain any frequencies above 3 kHz strictly. This can be the situation with other signals also. Hence the signal should be strictly bandlimited to avoid aliasing. The bandlimiting of the signal means to ensure that it contains frequencies within the specific band only.

The signal can be bandlimited by passing it through a filter which blocks or highly attenuates all the frequency components outside the specific bandwidth. Such filter is normally a low pass type of filter. For example, a speech signal can be passed through the low pass filter of bandwidth 3 kHz. Hence all the frequency components below 3 kHz are passed through the filter. And all the frequency components above 3 kHz are blocked or attenuated by this filter. Hence with the nyquist rate of sampling (i.e. 6 kHz), there will be no aliasing. Thus as analog is passed through a filter and then it is given to the A/D converter for sampling. This filter avoids aliasing due to high frequency components and hence it is called Anti aliasing filter. Fig. 2.10.6 shows the block diagram of DSP system for an analog signal.

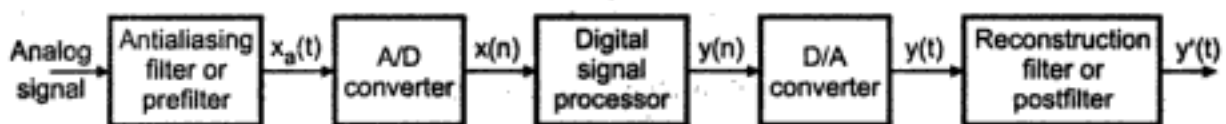


Fig. 2.10.6 Block diagram of a DSP system for processing of an analog signal

As shown in the above figure, the analog signal is first passed through antialiasing filter for bandlimiting. This filter is also called prefilter. Then the signal is given to A/D converter for sampling. The sequence $x(n)$ is processed through DSP processor and the output sequence $y(n)$ is applied to D/A converter. The signal $y(t)$ from the D/A converter is passed through the reconstruction filter. This is interpolation low pass filter and used to generate smooth signal from individual samples. The reconstruction filter is also called postfilter. Its bandwidth depends upon the frequency content of the signal to be reconstructed.

Ex.2.10.3 A digital communication link carries binary coded words representing samples of an input signal $x_a(t) = 3 \cos 600\pi t + 2 \cos 800\pi t$. The link is operated at 10,000 bits/sec and each input sample is quantized into 1024 different voltage levels.

- What is the sampling frequency and folding frequency?
- What is the Nyquist rate for the signal $x_a(t)$?
- What are the frequencies in resulting discrete time signal $x(n)$?
- What is the resolution ' Δ '?

[Dec - 99]

Sol. : (i) Each input sample is quantized into $L = 1024$ levels. The number of bits/sample are denoted by ' b ' and are given by equation 2.10.30 as,

$$\begin{aligned} b &= \log_2 L \\ &= \log_2 1024 = \frac{\log_{10} 1024}{\log_{10} 2} \\ &= 10 \text{ bits} \end{aligned}$$

The bit rate in the digital communication link is given as,

$$\text{Bit rate} = \text{samples/sec} \times \text{bits/sample} \quad \dots (2.10.39)$$

In above equation Bit rate = 10,000 bits/sec and 10 bits/sample. Samples/sec is nothing but sampling frequency F_s . Thus,

$$\begin{aligned}
 F_s \text{ (samples/sec)} &= \frac{\text{Bit rate}}{\text{bits / sample}} \\
 &= \frac{10,000}{10} \\
 &= 1000 \text{ Hz}
 \end{aligned}$$

Folding frequency is the frequency after which folding or aliasing starts. We know that aliasing starts after $\frac{F_s}{2}$. Hence folding frequency becomes,

$$\text{Folding frequency} = \frac{F_s}{2} = 500 \text{ Hz}$$

(ii) There are two frequencies in $x_a(t)$. i.e.,

$$x_a(t) = A_1 \cos 2\pi F_1 t + A_2 \cos 2\pi F_2 t \quad \dots (2.10.40)$$

The given signal is,

$$x_a(t) = 3 \cos 600 \pi t + 2 \cos 800 \pi t$$

The above equation can be rearranged as,

$$x_a(t) = 3 \cos 2\pi 300 t + 2 \cos 2\pi 400 t \quad \dots (2.10.41)$$

Comparing above equation with equation 2.10.40 we get,

$$F_1 = 300 \text{ Hz} \text{ and } A_1 = 3$$

$$F_2 = 400 \text{ Hz} \text{ and } A_2 = 2$$

Here $F_{\max} = F_2 = 400 \text{ Hz}$

Nyquist rate is given by equation 2.10.35 as,

$$\begin{aligned}
 \text{Nyquist rate} &= 2 F_{\max} \\
 &= 2 \times 400 = 800 \text{ Hz}
 \end{aligned}$$

(iii) The sampling frequency is $F_s = 1000 \text{ Hz}$.

The discrete time signal $x(n)$ is given by putting $t = nT = \frac{n}{F_s}$ in equation 2.10.41. i.e.,

$$\begin{aligned}
 x(n) &= 3 \cos 2\pi 300 \cdot \frac{n}{1000} + 2 \cos 2\pi 400 \times \frac{n}{1000} \\
 &= 3 \cos 2\pi 0.3 n + 2 \cos 2\pi 0.4 n \quad \dots (2.10.42)
 \end{aligned}$$

The discrete time signal having two frequencies is represented as,

$$x(n) = A_1 \cos 2\pi f_1 n + A_2 \cos 2\pi f_2 n$$

On comparing above equation with equation 2.10.42 we find that there are two frequencies in discrete time signal. i.e.,

$$f_1 = 0.3 \text{ and } f_2 = 0.4$$

Here both the frequencies are distinct since they are maximum value of $\frac{1}{2}$ i.e. 0.5.

(iv) The two amplitudes are $A_1 = 3$ and $A_2 = 5$. Hence maximum amplitude will be $x_{\max} = A_1 + A_2 = 5$. Since these are sinusoidal signals, minimum amplitude will be $x_{\min} = -x_{\max} = -5$. From equation 2.10.28 step size Δ is given as,

$$\Delta = \frac{x_{\max} - x_{\min}}{L - 1}$$

Putting values in above equation,

$$\Delta = \frac{5 - (-5)}{1024 - 1}$$

$$\Delta = 0.009775$$

Ex.2.10.4 Develop an algorithm for sampling a signal having frequency range of 10 Hz to 1 kHz. You can use an 8 bit A/D converter with conversion time of 50 μ sec. Store at least 1024 samples of signal in a data file. **[Dec - 99]**

Sol. : Before actually going to the steps of algorithm we should discuss few important points in such case. The A/D converter can be used along with PC or DSP processor. Hence the conversion routine can be written in assembly language or advanced language like C/C++ for PC. If it is DSP processor, then conversion routine can be written in assembly language of DSP processor. We will assume that the A/D converter is interfaced to PC or DSP processor in usual standard way.

The input signal frequency is 10 Hz to 1 kHz. The conversion time of A/D converter is 50 μ sec. This means the individual samples will be separated by minimum of $T_{\min} = 50 \mu$ sec. This imposes a limit on maximum sampling frequency. i.e.,

$$F_{s(\max)} = \frac{1}{T_{\min}} = \frac{1}{50 \mu\text{sec}} = 20,000 \text{ Hz}$$

Thus the sampling frequency should be less than 20 kHz for given system.

The A/D converter gives 8 bits/sample. This means each sample requires 1 byte. We have to store 1024 samples in the data file. This means the data file size will be of 1 K bytes.

The steps of the A/D conversion algorithm are shown below :

Algorithm :

1. Get the name of the data file from user in which digital data is to be stored. If the name is not given, use default name.
2. Open the existing data file with specified name to write binary coded samples. If file doesnot exist, create new one.
3. Initialize the A/D converter channel from which the signal is to be converted. Sometimes this initialization is done when start of conversion command is given.
4. Initialize the count for 1024 samples. Fix up sampling time/frequency if specified, otherwise use default values.
5. Write header into the data file consisting of sampling rate, number of channels of input signal, minimum and maximum amplitudes of input signal, number of bits/sample, total number of samples, storage mode of data file etc.
6. Give start of conversion (SOC) signal to A/D converter. Sometimes writing into the output port of A/D converter performs this job.
7. Wait for the conversion time. Sometimes this conversion time is used for processing purposes.
8. Read digital value of the sample from output port of A/D converter.
9. Store the digital value of the sample in the data file in the specified format.
10. Wait till the sampling time 'T'. This time includes all delays like conversion, storage, processing etc.
11. Check whether 1024 samples are stored. If samples are less than 1024, repeat steps 6 to 11. If 1024 samples are stored, then go to next step.

12. Stop A/D conversion process. Close the data file and return control to the user indicating that 1024 samples are digitized successfully.

2.11 Implementation of General Difference Equation using C

We have seen earlier that a recursive as well as nonrecursive LTI systems can be well described with the help of constant coefficient difference equation. Almost all the discrete time systems can be implemented with the help of this difference equation. The generalized form of such difference equation is given by equation 2.8.19 as,

$$y(n) = - \sum_{k=1}^N a_k y(n-k) + \sum_{k=0}^{M-1} b_k x(n-k) \quad \dots (2.11.1)$$

[Here M is written as $M-1$, it doesn't make any difference]

The above equation can be expanded as,

$$y(n) = -[a_1 y(n-1) + a_2 y(n-2) + a_3 y(n-3) + \dots] \\ + b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + \dots \quad \dots (2.11.2)$$

This equation is implemented by the C program.

```
file name : diffeqn.cpp
// Implementation of general difference equation based array mapping
//
// This program implements the difference equation of the LSI
// system which is given as,
//
//      y(n) = -[a1*y(n-1)+a2*y(n-2)+a3*y(n-3)+.....]
//             +b0*x(n)+b1*x(n-1)+b2*x(n-2)+.....
//
// Inputs: 1. Number of coefficients ak, denoted as N.
//          2. Values of a1,a2,a3,.....etc.
//          3. Number of coefficients bk, denoted as M.
//          4. Values of b0,b1,b2,.....etc.
//          5. Number of samples of x(n), denoted as L.
//          6. Values of x(0),x(1),x(2),.....etc.
//
// Outputs : The computed output sequence y(n)
//            according to the specified format of
//            difference equation as above.
//
// Assumptions : 1. The number of samples computed for y(n)
//                are same as number of input samples.
//                2. All initial conditions are assumed zero.
//
// Note :       Actually there is no specific formula
//              for number of samples in y(n). Because
//              samples in y(n) depend upon type of input
//              x(n), coefficients ak and bk and initial conditions.
//
//-----
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
    float a[10],b[10],x[100],y[100],sumXn_k,sumYn_k;
```

```

int N,M,k,L,n;
clrscr();

printf("    Implementation of general difference"
       " equation based array mapping\n");
printf("\nEnter the number of coefficients a[k] i.e. N = ");
scanf("%d",&N);           // number of ak i.e N
printf("Enter the values coefficients a[k] \n");
for(k = 1; k <= N; k++)    // values of a1,a2,a3,...etc.
{
    printf("a%d = ",k);
    scanf("%f",&a[k]);
}

printf("\nEnter the number of coefficients b[k] i.e. M = ");
scanf("%d",&M);           // number of bk i.e M
printf("Enter the values coefficients b[k] \n");
for(k = 0; k < M; k++)    // values of b0,b1,b2,...etc.
{
    printf("b%d = ",k);
    scanf("%f",&b[k]);
}

printf("\nEnter the number of samples of x(n) ");
scanf("%d",&L);           // number of samples of x(n)
for(k = 0; k < L; k++)    // values of x(0),x(1),x(2),...etc.
{
    printf("x(%d) = ",k);
    scanf("%f",&x[k]);
}

printf("\nThe computed values of y(n) are as follows...\n");
for(n = 0; n < L; n++)    // computation of y(n)
{
    sumYn_k = 0;
    sumXn_k = 0;
    for(k = 1; (k <= n)&&(k <= N); k++)
    {
        // computation of a1*y(n-1)+a2*y(n-2)+a3*y(n-3)+.....
        sumYn_k += a[k] * y[n-k];
    }
    for(k = 0; (k <= n)&&(k < M); k++)
    {
        // computation of b0*x(n)+b1*x(n-1)+b2*x(n-2)+.....
        sumXn_k += b[k] * x[n-k];
    }
    y[n] = - sumYn_k + sumXn_k; // final value of y(n)
    printf("\ny(%d) = %f",n,y[n]);
}
}
//-----End of program-----

```

The program first accepts number of coefficients a_k (i.e. N) and b_k (i.e. M). Then the program accepts values of a_k and b_k . Then the input samples (L) and their values are accepted by the program.

The program computes $y(n)$ upto ' L ' number of samples. Thus $y(n)$ has same number of samples as in $x(n)$. This is assumption. Actually there is no relationship between the number of samples in the output $y(n)$ and input $x(n)$. Nonzero values of $y(n)$ can be obtained even if

$x(n)$ is absent. Here, just to simplify the program, output samples are taken same as input samples.

In the program there is last `for` loop for computation of $y(n)$. In this loop there are two `for` loops. i.e. first `for` loop is,

```
for(k = 1; (k <= n) && (k <= N); k++)
{
    sumYn_k += a[k] * y[n_k];
}
```

This loop implements first part of equation 2.11.2, i.e.,

$$a_1 y(n-1) + a_2 y(n-2) + a_3 y(n-3) + \dots \quad \dots (2.11.3)$$

Here the sum is stored in `sumYn_k` variable. Observe that the `for` loop starts from $k = 1$. But upper condition is $(k \leq N) \&\& (k \leq N)$. This means equation 2.11.3 is computed for $k \leq n$ and $k \leq N$. This is done since we don't know $y(-1), y(-2), \dots$ etc. Indirectly all these values are assumed zero. This is crucial part of the program.

The next `for` loop computes second part of equation 2.11.2. i.e.,

$$b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + \dots \quad \dots (2.11.4)$$

And the corresponding `for` loop in the program is,

```
for(k = 0; (k <= n) && (k < M); k++)
{
    sumXn_k += b[k] * x[n_k];
}
```

The sum of equation 2.11.4 is stored in `sumXn_k` variable. Here observe the difference. This `for` loop starts from $k = 0$. Hence there is little difference in upper condition, i.e. $(k \leq n) \&\& (k < M)$. Here it is $k < M$ and not $k \leq M$. Similarly since we don't know $x(-1), x(-2), \dots$ etc; this condition avoids these values by using $(k \leq n)$. This means equation 2.11.4 is computed for $k \leq n$ and $k < M$. This indirectly assumes $x(-1), x(-2), \dots$ etc. to zero. This is also crucial part of the program.

To run the program

Let us test this program for following values :

$$N = 1, \quad a_1 = 1$$

and $M = 1, \quad b_0 = 1$

Putting these values in equation 2.11.2 we get,

$$y(n) = -a_1 y(n-1) + b_0 x(n) \quad \dots (2.11.5)$$

Let the input $x(n)$ has 10 samples as follows :

$$x(n) = \{1, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$$

Thus $x(0) = 1$ and rest all $x(n)$ have zero values.

Since $a_1 = 1$ and $b_0 = 1$ in equation 2.11.5 we have,

$$y(n) = -y(n-1) + x(n) \quad \dots (2.11.6)$$

With $n = 0$, above equation becomes,

$$y(0) = -y(-1) + x(0)$$

Here program doesn't compute $y(-1)$ since $n < k$. Hence $y(-1)$ is assumed zero. Thus,

$$y(0) = x(0) = 1 \quad \text{since} \quad x(0) = 1$$

With $n=1$ in equation 2.11.6 we have,

$$y(1) = -y(0) + x(1) = -1 + 0 = -1$$

Similarly,

$$y(2) = -y(1) + x(2) = -(-1) + 0 = 1$$

$$y(3) = -y(2) + x(3) = -(1) + 0 = -1$$

Thus we keep on getting $y(n)$, even if $x(n)=0$. The program computes upto $y(9)$ since input $x(n)$ is specified upto $n=9$. Thus $y(n)$ is,

$$y(n) = \{1, -1, 1, -1, 1, -1, 1, -1, 1, \dots\}$$

The results of the program are shown below for the same conditions as calculated above. Observe that program provides same values of $y(n)$.

----- Results -----

Implementation of general difference equation based array mapping

Enter the number of coefficients a[k] i.e. N = 1

Enter the values coefficients a[k]

a1 = 1

Enter the number of coefficients b[k] i.e. M = 1

Enter the values coefficients b[k]

b0 = 1

Enter the number of samples of x(n) 10

x(0) = 1

x(1) = 0

x(2) = 0

x(3) = 0

x(4) = 0

x(5) = 0

x(6) = 0

x(7) = 0

x(8) = 0

x(9) = 0

The computed values of $y(n)$ are as follows...

y(0) = 1.000000

y(1) = -1.000000

y(2) = 1.000000

y(3) = -1.000000

y(4) = 1.000000

y(5) = -1.000000

y(6) = 1.000000

y(7) = -1.000000

y(8) = 1.000000

y(9) = -1.000000

2.12 Generation of Waveforms Using C

In this section we will see about how the discrete time sequences for sine, cosine, exponential, square, sawtooth and random signals are generated. These sequences can be generated using 'C' as well as standard signal processing softwares such as MATLAB.

Here we present the program in 'C' which generates the samples of cosine, sine, square exponential and random signals. The source code of the program is listed below :

```

file name : signals.cpp
/*--- Generation of samples of some standard signals -----*/
//
// This program generates the samples of cosine, sine, square,
// exponential and random noise at specified sampling
// frequencies.
//
// Inputs : 1. Frequencies of cosine, sine and
//           square waves.
//           2. Sampling frequency
//           3. Choice for the signal to be generated.
//
// Outputs : 1. The samples of the discrete time
//            signal are stored in the array.
//            2. The discrete time signal is displayed
//               on the screen in graphics mode.
//
// Assumptions : The number of samples of the signal
//                and their amplitudes are assumed in the
//                'program.
//-----
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
#include<graphics.h>
void main()
(
    float x[700],A,F,Fs,n,Y;
    int i,gd,gm,X,choice,k;
    clrscr();

    printf("          Generation of discrete time signals \n");
    printf("\nEnter the frequency of analog signal F = ");
    scanf("%f",&F); //frequency of the signal
    printf("Enter the sampling frequency Fs = ");
    scanf("%f",&Fs); //sampling frequency
    printf("\nEnter your choice");
    printf("\ncosine wave (Enter 1)"); //cosine wave
    printf("\nsine wave (Enter 2)"); //sine wave
    printf("\nsquare wave (Enter 3)"); //square wave
    printf("\nexponential signal (Enter 4)"); //exponential signal
    printf("\nRandom noise (Enter 5)\nChoice = "); //random noise
    scanf("%d",&choice);
    i = 640; //640 samples of the signal will be generated
    A = 1.0; //maximum amplitude of the signals is 1
    switch(choice)
    {
        case 1 : for(n = 0; n < i; n++) // cosine wave
                x[n] = A * cos( 2 * 3.1415927 * F * (n/Fs));
                break;
        case 2 : for(n = 0; n < i; n++) // sine wave
                x[n] = A * sin( 2 * 3.1415927 * F * (n/Fs));
    }
}

```

```

        break;
    case 3 : k = 0; // square wave
            do
            {
                for(n=(k*Fs)/(2*F); n < ((k+1)*Fs)/(2*F); n++)
                { // positive half cycle of aquare wave
                    x[n] = A; if(n>i)break;
                }
                for(n=((k+1)*Fs)/(2*F); n<((k+2)*Fs)/(2*F); n++)
                { // negative half cycle of aquare wave
                    x[n] = -A; if(n>i) break;
                }
                k=k+2; //this count is modified for next cycle
            } while(n<i);
        break;
    case 4 : for(n = 0; n < i; n++) // exponential signal
            x[n] = A * exp( -(n/Fs));
            break;
    case 5 : for(n = 0; n < i; n++) // random noise
            x[n] = (float)(rand()%1000)/1000.0;
            break;
    }
//----- next part of the program displays the generated signal
Y = X = 0;
gd = DETECT;
initgraph(&gd, &gm, ""); // initialize screen in graphics mode
for(n = 0; n < i; n++)
{ // this loop displays 640 samples of discrete time signal
  Y = 200 - x[n]*100; //scaling of x(n) for proper display
  putpixel(X, Y, WHITE); //x(n) is displayed as putting pixels
  X++;
}
getch();
closegraph();
}
//-----End of program-----

```

The program asks for frequency 'F' of analog signal. This frequency is asked for all the waveforms, but it is ignored for exponential and random noise signals. The program also asks for sampling frequency 'F_s'. Then the choice for the sequence to be generated is asked.

This program displays the generated sequence on the screen in graphics mode. A resolution of 640 × 480 is assumed. Hence 640 samples can be displayed on the screen. Hence program generates 640 samples of sequence. Observe the statements in program.

$i = 640;$ ← This statement is counter for 640 samples.

$A = 1.0;$ ← This statement sets maximum amplitude of samples.

The next switch statement generates samples of the signal depending upon choice given by user.

Example : How sine wave is generated in program ?

We know that sine wave (analog) is given as,

$$x_a(t) = A \sin(2\pi Ft) \quad \dots (2.12.1)$$

A discrete time sine wave can be obtained by putting $t = nT = \frac{n}{F_s}$ since $T = \frac{1}{F_s}$. Hence above equation becomes,

$$x(n) = A \sin\left(2\pi F \cdot \frac{n}{F_s}\right)$$

Here value of $\pi = 3.1415927$ substituted in the program. i.e.,

$$x(n) = A \sin\left(2 \times 3.1415927 \times F \times \frac{n}{F_s}\right) \quad \dots (2.12.2)$$

This statement is implemented in the program observe case 2. There is a for loop which operates from $n = 0$ to $n < i$ (i.e. $n < 640$). And this for loop executes following statement :

$$x[n] = A * \sin(2 * 3.1415927 * F * (n / F_s)) ;$$

The above 'C' statement implements equation 2.12.2. Depending upon F and F_s , samples of sine wave are generated.

Cosine wave

The samples of cosine wave are generated by using the same logic discussed for sine wave. Case 1 statement generates cosine wave.

Logic for generation of square wave :

The logic for generation of square wave is very straight forward. Fig. 2.12.1 shows the waveform of square wave.

From above waveform we can write following,

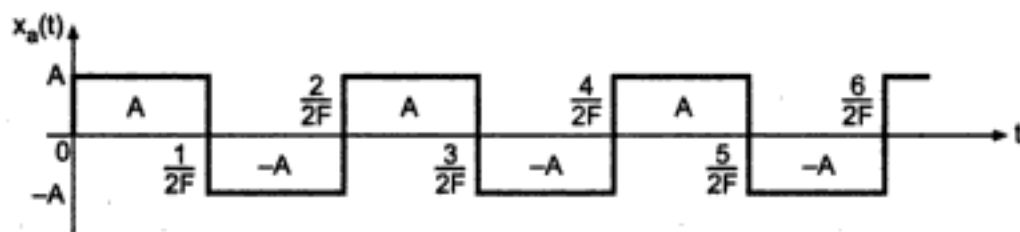


Fig. 2.12.1 Waveform of square wave of frequency F .
Its various periods are marked in terms of F

$$\begin{aligned} x_a(t) &= A, \text{ for } 0 \leq t < \frac{1}{2F}; \frac{2}{2F} \leq t < \frac{3}{2F}; \frac{4}{2F} \leq t < \frac{5}{2F}; \dots \\ &= -A, \text{ for } \frac{1}{2F} \leq t < \frac{2}{2F}; \frac{3}{2F} \leq t < \frac{4}{2F}; \frac{5}{2F} \leq t < \frac{6}{2F}; \dots \end{aligned}$$

The discrete time version of this signal is obtained by putting $t = nT = \frac{n}{F_s}$ in above equation. i.e.,

$$\begin{aligned} x(n) &= A, \text{ for } 0 \leq \frac{n}{F_s} < \frac{1}{2F}; \frac{2}{2F} \leq \frac{n}{F_s} < \frac{3}{2F}; \frac{4}{2F} \leq \frac{n}{F_s} < \frac{5}{2F}; \dots \\ &= -A, \text{ for } \frac{1}{2F} \leq \frac{n}{F_s} < \frac{2}{2F}; \frac{3}{2F} \leq \frac{n}{F_s} < \frac{4}{2F}; \frac{5}{2F} \leq \frac{n}{F_s} < \frac{6}{2F}; \dots \end{aligned}$$

The above equation can be further simplified as,

$$\begin{aligned}
 x(n) &= A \quad \text{for } 0 \leq n < \frac{F_s}{2F}; \quad \frac{2F_s}{2F} \leq n < \frac{3F_s}{2F}; \quad \frac{4F_s}{2F} \leq n < \frac{5F_s}{2F}; \dots \\
 &= -A \quad \text{for } \frac{F_s}{2F} \leq n < \frac{2F_s}{2F}; \quad \frac{3F_s}{2F} \leq n < \frac{4F_s}{2F}; \quad \frac{5F_s}{2F} \leq n < \frac{6F_s}{2F}; \dots
 \end{aligned}
 \tag{2.12.3}$$

The above equation can be generalized as follows :

$$x(n) = \begin{cases} A & \text{for } \frac{k F_s}{2F} \leq n < \frac{(k+1) F_s}{2F} \\ -A & \text{for } \frac{(k+1) F_s}{2F} \leq n < \frac{(k+2) F_s}{2F} \end{cases}
 \tag{2.12.4}$$

The above equation is implemented by the do-while loop in case 3 statements to generate square wave in the program. The first for loop implements positive cycle i.e. $x(n) = A$ of the square wave and second for loop implements negative cycle i.e. $x(n) = -A$ of the square wave. The value of 'k' is incremented by '2' as it is clear from equation 2.12.3.

Generation of exponential sequence :

The decaying analog exponential signal is given as,

$$x_a(t) = A e^{-t} \tag{2.12.5}$$

$x(n)$ can be obtained by putting $t = nT = \frac{n}{F_s}$, i.e.,

$$x(n) = A e^{-\frac{n}{F_s}} \tag{2.12.6}$$

This equation is implemented by the case 4 statements, i.e.,

$$x[n] = A * \exp(- (n / F_s));$$

The for loop generates 640 samples of exponential sequence and stores in $x[n]$.

Generation of random noise :

The random noise can be generated by number of ways. A standard `rand()` function of 'C' is used here to generate samples of random noise.

Displaying discrete time signals on screen :

The `initgraph` function initializes screen in graphics mode. Note that the 'bgi' files should be present in the same directory in which this program is executed. Otherwise proper path should be provided. The for loop displays 640 samples as pixels on the screen. The statement,

$$Y = 200 - x[n] * 100 ;$$

Scales sample value for proper display on the screen. The samples are displayed on the screen by 'putpixel' function.

To run this program :

Let us test this program. The signal frequency is $F = 10 \text{ Hz}$, sampling frequency is $F_s = 1000 \text{ Hz}$ and 'sine' wave is selected. The results of program are shown below :

```
//----- results -----
      Generation of discrete time signals
```

Enter the frequency of analog signal F = 10
 Enter the sampling frequency Fs = 1000

Enter your choice
 cosine wave (Enter 1)
 sine wave (Enter 2)
 square wave (Enter 3)
 exponential signal (Enter 4)
 Random noise (Enter 5)
 Chioce = 2



Fig. 2.12.2 A sine wave of frequency 10 Hz and sampled at 1000 Hz displayed on the screen

2.13. Linear Convolution using 'C'

Linear convolution is one of the most important operation in signal processing. It relates the input, output and unit sample response of the system. i.e.,

$$\begin{aligned}
 y(n) &= x(n) * h(n) \\
 &= \sum_{k=-\infty}^{\infty} x(k) h(n-k) \quad \dots (2.13.1)
 \end{aligned}$$

Here $h(n)$ is the unit sample response,

$x(n)$ is the input sequence and

$y(n)$ is the output sequence.

The 'C' program for implementing convolution is given below :

```

file name : linconv.cpp
/*----- Linear Convolution of two sequences -----*/
//
// This program computes the Linear Convolution of two causal
// sequences x(n) and h(n).
//
// Inputs : 1. Number of samples in h(n)
//          2. Samples of h(n) in the form
//            h[0],h[1],h[2],.....h[n-1]
//          3. Number of samples in x(n)
//          4. Samples of x(n) in the form
//            x[0],x[1],x[2],.....x[n-1]
//
// Outputs : Convolution sequence of x(n)
//           and h(n)

```

```

//
// Assumptions: The given sequences are causal.
//-----
#include<stdio.h>
#include<conio.h>
#include<math.h>
main()
{
float h[20],x[20],y[20],sum;
int N,M,n,k;
clrscr();
printf("                                Linear Convolution\n\n");

printf("Enter the number of samples in h(n) = ");
scanf("%d",&N); // number of samples in h(n)
printf("Enter the sequence h(n)\n");
for(n = 0; n < N; n++)
{
printf("h[%d] = ",n); // Entering the sequence h(n)
scanf("%f",&h[n]);
}

printf("Enter the number of samples in x(n) = ");
scanf("%d",&M); // number of samples in x(n)
printf("Enter the sequence x(n)\n");
for(n = 0; n < M; n++)
{
printf("x[%d] = ",n); // Entering the sequence x(n)
scanf("%f",&x[n]);
}

printf("\nThe result of convolution is...");
for(n = 0; n < (N+M-1); n++) // convolution calculation
{
sum = 0.0;
for(k = 0; k < M; k++) // summation loop
{
if(n < k || (n-k)>= N) continue;
sum += x[k] * h[n-k];
}
y[n] = sum;
printf("\ny[%d] = %f",n,y[n]);
}
}
//----- End of program -----

```

The first for loop is used to enter the sequence $h(n)$. The next for loop enters the sequence $x(n)$. To make program simple, causal sequences are considered. The last, i.e. 3rd for loop implements linear convolution of equation 2.13.1. We know that there are $(N + M - 1)$ number of samples in $y(n)$. Hence the upper limit on 'n' in this for loop is $n < N + M - 1$. Since sequences are causal, $y(n)$ starts from $n = 0$. The summation loop is an inner for loop

and it implements $\sum_{k=0}^{M-1} x(k)h(n-k)$ for every value of 'n'. In the inner for loop there is one

statement i.e.,

```
if (n < k || (n - k) >= N) continue;
```

This statement avoids the execution of

```
sum += x[k] * h[n - k];
```

if $n < k$ or $(n - k) \geq N$. This is because $h(n)$ is given only for $n = 0$ to $N - 1$. Similarly the lower limit on k in the inner for loop (i.e. summation loop) is zero, i.e. $k = 0$ since $x(k)$ is a causal sequence.

To run the program :

Let us consider the convolution of

$$h(n) = \left\{ \begin{array}{c} 1, 1, 1 \\ \uparrow \end{array} \right\}$$

and

$$x(n) = \left\{ \begin{array}{c} 1, 1, 1 \\ \uparrow \end{array} \right\}$$

By multiplication method we can obtain convolution of $h(n)$ and $x(n)$ as given below in Fig. 2.13.1.

Thus the computed output sequence is,

$$y(n) = \left\{ \begin{array}{c} 1, 2, 3, 2, 1 \\ \uparrow \end{array} \right\}$$

There are '5' samples in $y(n)$. The results of this program for above example are given below :

```
----- results -----
          Linear Convolution

Enter the number of samples in h(n) = 3
Enter the sequence h(n)
h[0] = 1
h[1] = 1
h[2] = 1
Enter the number of samples in x(n) = 3
Enter the sequence x(n)
x[0] = 1
x[1] = 1
x[2] = 1

The result of convolution is...
y[0] = 1.000000
y[1] = 2.000000
y[2] = 3.000000
y[3] = 2.000000
y[4] = 1.000000
-----
```

Observe that the results of C program are exactly similar to these calculated earlier.

Important equations to be remembered :

1. If analog signal is $x_a(t)$. Then $x(n)$ is obtained by putting $t = nT = \frac{n}{F_s}$, i.e.,

$$x(n) = x_a(t) \Big|_{t = nT = \frac{n}{F_s}}$$

2. Linear convolution is given as,

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

3. LSI system is described by following constant coefficient difference equation.

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

4. Following are important frequency relationships.

$$(i) \quad \omega = \frac{\Omega}{F_s} = \Omega T$$

$$(ii) \quad f = \frac{F}{F_s} = FT$$

$$(iii) \quad -\frac{1}{2} \leq f \leq \frac{1}{2} \quad \text{and} \quad -\pi \leq \omega \leq \pi$$

$$(iv) \quad -\frac{F_s}{2} \leq F \leq \frac{F_s}{2} \quad \text{and} \quad -\pi F_s \leq \omega \leq \pi F_s$$

5. Sampling frequency $F_s \geq 2 F_{\max}$

$$\begin{aligned} 6. \text{ Crosscorrelation } r_{xy}(l) &= \sum_{n=-\infty}^{\infty} x(n) y(n-l) \\ &= \sum_{n=-\infty}^{\infty} x(n+l) y(n) \end{aligned}$$

7. Frequently required summation equation.

$$\sum_{k=N_1}^{N_2} a^k = \frac{a^{N_1} - a^{N_2+1}}{1-a}, \quad N_2 \geq N_1$$

If $a=1$, then above equation becomes

$$\sum_{k=N_1}^{N_2} a^k = N_2 - N_1 + 1$$

Computer Exercise

1. Consider the 'C' program for general difference equation based array mapping presented in section 2.11. Modify this program so that it should accept all the data from data file and store output in another file.

- Consider the 'C' program presented in section 2.12 to generate samples of cosine, sine, square, exponential and random noise signals. Add the function in this program to generate sawtooth, unit step and unit sample sequences. Your program should accept number of samples to be generated from user. Also modify the program to write samples in data file.
- A 'C' program for linear convolution is presented in section 2.13. Modify this program so that it should work for noncausal sequences as well. Your program should accept input sequences from one data file and store $y(n)$ in another file.

Theory Questions

- Define discrete time signals and classify them.
- Define and explain unit sample and unit step sequences.
- What is discrete time system? How discrete time systems are classified?
- Explain Linear Shift Invariant (LSI) system.
- Explain how any sequence can be explained by weighted impulses.
- Prove the formula for linear convolution.
- State and explain/prove the properties of convolution.
- Prove that the necessary and sufficient condition for LSI system to be causal is,

$$h(n) = 0 \quad \text{for } n < 0$$
- Prove that the necessary and sufficient condition for LSI system to be stable is,

$$\sum_{k=-\infty}^{\infty} |h(k)| < \infty$$
- Explain the FIR and IIR systems.
- Explain non recursive and recursive systems.
- Explain how LSI systems are represented using constant coefficient difference equations.
- What is autocorrelation and crosscorrelations? State its properties.
- Explain the frequency relationships between continuous time and discrete time signals.
- What is aliasing? Explain with the help of an example.
- Explain quantization, quantization error and encoding in the sampling of signals.
- State and explain sampling theorem and nyquist rate.
- What is antialiasing filter?

Unsolved Examples

- Consider the discrete time sequence as given below :

$$x(n) = \left\{ \underset{\uparrow}{1}, 1, 1, 1, \frac{1}{2}, \frac{1}{2} \right\}$$

(i) Sketch $x(n)$ (ii) find out $x(n)u(2-n)$ and sketch it.

Hint : First determine $u(-n)$. The delay this sequence by two samples to get $u(-n+2)$ or $u(2-n)$. Multiply $x(n) \cdot u(2-n)$.

$$\text{Ans.: } x(n)u(2-n) = \left\{ \underset{\uparrow}{1}, 1, 1, 1, 0, 0 \right\}$$

(iii) Find out $x(n-1]\delta(n-3)$ and sketch it.

Hint : First delay $x(n)$ by one sample to get $x(n-1)$. Then multiply $x(n-1)$ by $\delta(n-3)$.

$$\text{Ans. : } x(n-1)\delta(n-3) = \{0, 0, 0, 1, 0, 0\}$$

↑

2. State whether the following systems are (i) Static (ii) Linear (iii) Shift invariant (iv) Causal and (v) Stable.

(a) $y(n) = nx(n)$

Ans. : Static, linear, shift variant, causal, unstable.

(b) $y(n) = ax(n)$

Ans. : Static, linear, shift invariant, causal, stable.

(c) $y(n) = x(n^2)$

Ans. : Dynamic, linear, shift invariant, noncausal, stable.

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

Ans. : Dynamic, linear, shift invariant, causal, unstable.

(e) $y(n) = x(n) + 3u(n+1)$

Ans. : Static, nonlinear, shift variant, noncausal, stable.

(f) $y(n) = g(n)x(n)$ Here $g(n)$ is another sequence,

Ans. : Static, linear, shift variant, causal, stable.

$h(n) \Rightarrow$	↓			
	1	1	1	
$x(n) \Rightarrow$	1	1	1	
	↑			
	1	1	1	
	1	1	1	x
	1	1	1	x
$y(n) \Rightarrow$	1	2	3	2
	↑			

Fig. 2.13.1 Convolution of $x(n)$ and $h(n)$

3. The unit sample response of the LSI system is given as $\left(\frac{1}{4}\right)^n u(n)$. If the input

sequence to this system is $\left(\frac{1}{2}\right)^n u(n)$, determine its response.

Ans. :

$$y(n) = \left(\frac{1}{4}\right)^n [2^{n+1} - 1]$$

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

then
$$x(-n) \xleftrightarrow{z} X(z^{-1}) \text{ ROC : } \frac{1}{r_2} < |z| < \frac{1}{r_1} \quad \dots (3.3.4)$$

Proof :

By definition of z-transform we can write,

$$Z[x(-n)] = \sum_{n=-\infty}^{\infty} x(-n) z^{-n}$$

In the RHS of above equation let us change the index as $-n = m$. Hence we can write,

$$Z[x(-n)] = \sum_{m=\infty}^{-\infty} x(m) z^m$$

The above equation can be rearranged as follows :

$$\begin{aligned} Z[x(-n)] &= \sum_{m=-\infty}^{\infty} x(m) (z^{-1})^{-m} \\ &= X(z^{-1}) \end{aligned}$$

Thus folding the signal in time domain is equivalent to replacing z by z^{-1} in z-domain. Replacing z by z^{-1} is called inversion. Hence folding or reflection in time domain is equivalent to inversion in z-domain.

The ROC for $X(z)$ is $r_1 < |z| < r_2$, then the ROC for $X(z^{-1})$ will be,

$$\begin{aligned} r_1 &< |z^{-1}| < r_2 \\ \therefore \frac{1}{r_2} &< |z| < \frac{1}{r_1} \end{aligned}$$

3.3.5 Differentiation in z-domain

This property states that if,

$$x(n) \xleftrightarrow{z} X(z)$$

then
$$n x(n) \xleftrightarrow{z} -z \frac{d}{dz} X(z) \quad \dots (3.3.5)$$

Proof :

Consider the basic definition of z-transform, i.e.,

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

Differentiating both the sides of above equation with respect to z we get,

$$\begin{aligned} \frac{d}{dz} X(z) &= \sum_{n=-\infty}^{\infty} \frac{d}{dz} [x(n) z^{-n}] \\ &= \sum_{n=-\infty}^{\infty} x(n) \frac{d}{dz} z^{-n} \end{aligned}$$

$$\begin{aligned}
 &= \sum_{n=-\infty}^{\infty} x(n)(-n)z^{-n-1} \\
 &= - \sum_{n=-\infty}^{\infty} x(n) \cdot n \cdot z^{-n} \cdot z^{-1} \\
 &= -z^{-1} \sum_{n=-\infty}^{\infty} [nx(n)]z^{-n} \\
 &= -z^{-1} Z[nx(n)] \quad \text{By definition of z-transform.}
 \end{aligned}$$

$$\therefore Z[nx(n)] = -z \frac{d}{dz} X(z)$$

which is proved. The ROC of $Z[nx(n)]$ is same as that of $X(z)$.

3.3.6 Convolution in Time Domain

Convolution of two sequences is very useful property. It states that if,

$$x_1(n) \xleftrightarrow{Z} X_1(z)$$

and

$$x_2(n) \xleftrightarrow{Z} X_2(z)$$

then

$$x_1(n) * x_2(n) \xleftrightarrow{Z} X_1(z) \cdot X_2(z) \quad \dots (3.3.6)$$

Proof :

Let $x(n)$ represent the convolution of $x_1(n)$ and $x_2(n)$. i.e.,

$$\begin{aligned}
 x(n) &= x_1(n) * x_2(n) \\
 &= \sum_{k=-\infty}^{\infty} x_1(k) x_2(n-k) \quad \dots (3.3.7)
 \end{aligned}$$

By definition, z-transform of $x(n)$ is given as,

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

Putting for $x(n)$ from equation 3.3.7 in above formula we have,

$$\begin{aligned}
 X(z) &= Z[x_1(n) * x_2(n)] \\
 &= \sum_{n=-\infty}^{\infty} \left\{ \sum_{k=-\infty}^{\infty} x_1(k) x_2(n-k) \right\} z^{-n}
 \end{aligned}$$

Interchanging the order of summations in above equation we get,

$$X(z) = \sum_{k=-\infty}^{\infty} x_1(k) \left\{ \sum_{n=-\infty}^{\infty} x_2(n-k) z^{-n} \right\} \quad \dots (3.3.8)$$

In the above equation, the term in { } brackets represents the z-transform of $x_2(n-k)$. From the time shifting property of equation 3.3.2 we have,

$$Z[x(n-k)] = z^{-k} X(z)$$

Applying this property to the term in { } brackets in equation 3.3.8 we have,

$$X(z) = \sum_{k=-\infty}^{\infty} x_1(k)z^{-k} X_2(z)$$

In the above equation $\sum_{k=-\infty}^{\infty} x_1(k) \cdot z^{-k}$ represents the z-transform of $x_1(k)$, i.e. $X_1(z)$.

Hence above equation becomes,

$$X(z) = X_1(z) \cdot X_2(z)$$

Thus the convolution property of equation 3.3.6 is proved. It indicates that the convolution of two sequences in time domain is equivalent to multiplication of their z-transforms. The ROC of the product $X_1(z) \cdot X_2(z)$ is the intersection or overlap of ROC of two individual sequences.

3.3.7 Correlation of Two Sequences

This property states that if,

$$x_1(n) \xleftrightarrow{z} X_1(z)$$

and $x_2(n) \xleftrightarrow{z} X_2(z)$

then $\sum_{n=-\infty}^{\infty} x_1(n)x_2(n-l) \xleftrightarrow{z} X_1(z)X_2(z^{-1})$... (3.3.9)

Proof :

Here the correlation of two sequences $x_1(n)$ and $x_2(n)$ is denoted as $r_{x_1x_2}(l)$ and is given as,

$$r_{x_1x_2}(l) = \sum_{n=-\infty}^{\infty} x_1(n)x_2(n-l) \text{ By definition}$$

Let us rearrange the term $x_2(n-l)$ as $x_2[-(l-n)]$ in above equation i.e.,

$$r_{x_1x_2}(l) = \sum_{n=-\infty}^{\infty} x_1(n)x_2[-(l-n)]$$

The RHS of above equation represents the convolution of $x_1(l)$ and $x_2(-l)$. Hence we can write above equation as,

$$r_{x_1x_2}(l) = x_1(l) * x_2(-l)$$

By taking z-transform of both the sides of above equation,

$$Z[r_{x_1x_2}(l)] = Z[x_1(l) * x_2(-l)]$$

From the convolution property of equation 3.3.6 we can write RHS of above equation as multiplication of z-transforms of $x_1(l)$ and $x_2(-l)$. i.e.,

$$Z[r_{x_1x_2}(l)] = Z[x_1(l)] Z[x_2(-l)]$$

In the above equation $Z[x_1(l)] = X_1(z)$. And from the time reversal property of equation 3.3.4 we have $Z[x_2(-l)] = X_2(z^{-1})$. Hence above equation becomes,

$$Z[r_{x_1x_2}(l)] = X_1(z) \cdot X_2(z^{-1})$$

which is proved. The ROC of the z-transform of correlation is intersection or overlap of ROC of two individual sequences.

3.3.8 Multiplication of Two Sequences

This property states that if,

$$x_1(n) \xleftrightarrow{z} X_1(z)$$

and

$$x_2(n) \xleftrightarrow{z} X_2(z)$$

$$\text{then } x_1(n)x_2(n) \xleftrightarrow{z} \frac{1}{2\pi j} \oint_C X_1(v)X_2\left(\frac{z}{v}\right)v^{-1}dv \quad \dots (3.3.10)$$

Here C is a closed contour. It encloses the origin and lies in the ROC which is common to both $X_1(v)$ and $X_2\left(\frac{1}{v}\right)$.

Proof :

Let $x(n)$ represent the multiplication of $x_1(n)$ and $x_2(n)$ i.e.,

$$x(n) = x_1(n) \cdot x_2(n) \quad \dots (3.3.11)$$

Inverse z-transform of $x_1(n)$ is given as,

$$x_1(n) = \frac{1}{2\pi j} \oint_C X_1(v)v^{n-1}dv \quad \dots (3.3.12)$$

This is the standard formula gives $x_1(n)$ from its z-transform. We will see more about inverse z-transforms and this formula in the section on inverse z-transforms. Putting this expression for $x_1(n)$ in equation 3.3.11.

$$x(n) = \frac{1}{2\pi j} \oint_C X_1(v)v^{n-1}dv \cdot x_2(n)$$

Taking z-transform of this function,

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} = \sum_{n=-\infty}^{\infty} \left[\frac{1}{2\pi j} \oint_C X_1(v)v^{n-1}dv \cdot x_2(n) \right] z^{-n}$$

Let us rearrange the above equation as,

$$\begin{aligned} X(z) &= \frac{1}{2\pi j} \oint_C X_1(v) \sum_{n=-\infty}^{\infty} v^n \cdot v^{-1} x_2(n) z^{-n} dv \\ &= \frac{1}{2\pi j} \oint_C X_1(v) \sum_{n=-\infty}^{\infty} x_2(n) \frac{1}{v^{-n}} z^{-n} v^{-1} dv \\ &= \frac{1}{2\pi j} \oint_C X_1(v) \sum_{n=-\infty}^{\infty} \left[x_2(n) \left(\frac{z}{v} \right)^{-n} \right] v^{-1} dv \end{aligned}$$

In the above equation $\sum_{n=-\infty}^{\infty} x_2(n) \left(\frac{z}{v} \right)^{-n}$ is equal to $X_2\left(\frac{z}{v}\right)$ by definition of z-transform.

Hence we can write,

$$X(z) = \frac{1}{2\pi j} \oint_C X_1(v) \cdot X_2\left(\frac{z}{v}\right) v^{-1} dv$$

which is the required result. Let the ROC of $X_1(v)$ be,

$$r_{1l} < |v| < r_{1u} \quad \dots (3.3.13)$$

and that of $X_2(z)$ be,

$$r_{2l} < |z| < r_{2u}$$

Then the ROC of $X_2\left(\frac{z}{v}\right)$ will be,

$$r_{2l} < \left|\frac{z}{v}\right| < r_{2u}$$

Hence the ROC for $X(z)$ will be,

$$r_{2l} |v| < |z| < r_{2u} |v| \quad \text{from above equation}$$

The range of values of $|v|$ is specified in equation 3.3.13.

3.3.9 Conjugation of a Complex Sequence

This property states that if $x(n)$ is a complex sequence and if,

$$x(n) \xrightarrow{z} X(z)$$

then z-transform of complex conjugate of $x(n)$ is,

$$x^*(n) \xrightarrow{z} X^*(z^*) \quad \dots (3.3.14)$$

Proof :

By definition, the z-transform of $x^*(n)$ will be,

$$Z[x^*(n)] = \sum_{n=-\infty}^{\infty} x^*(n) z^{-n}$$

Let us rearrange the above equation as,

$$\begin{aligned} Z[x^*(n)] &= \sum_{n=-\infty}^{\infty} [x(n)(z^*)^{-n}]^* \\ &= \left[\sum_{n=-\infty}^{\infty} x(n)(z^*)^{-n} \right]^* \end{aligned}$$

In the above equation the quantity in [] brackets is equal to $X(z^*)$. Hence,

$$\begin{aligned} Z[x^*(n)] &= [X(z^*)]^* \\ &= X^*(z^*) \end{aligned}$$

This is the required proof. The ROC of z-transform of conjugate sequence is same as that of $X(z)$.

3.3.10 z-transform of Real Part of the Sequence

This property states that if $x(n)$ is a complex valued sequence and

$$x(n) \xrightarrow{z} X(z)$$

$$\text{then } \operatorname{Re}[x(n)] \xrightarrow{z} \frac{1}{2} [X(z) + X^*(z^*)] \quad \dots (3.3.15)$$

Proof :

Consider the RHS of equation 3.3.15 above. By definition of z-transform we can write,

$$\frac{1}{2} [X(z) + X^*(z^*)] = \frac{1}{2} \{Z[x(n)] + Z[x^*(n)]\}$$

In the above equation we have used the conjugation property of equation 3.3.14 to write $X^*(z^*) = Z[x^*(n)]$. Since z-transform satisfies the linearity property we can write above equation as,

$$\frac{1}{2} [X(z) + X^*(z^*)] = \frac{1}{2} Z\{x(n) + x^*(n)\} \quad \dots (3.3.16)$$

$$\text{We know that } x(n) = \operatorname{Re}[x(n)] + j \operatorname{Im} x(n)$$

$$\text{Hence } x^*(n) = \operatorname{Re}[x(n)] - j \operatorname{Im} x(n)$$

Putting the values of $x(n)$ and $x^*(n)$ in equation 3.3.16 and simplifying we get,

$$\frac{1}{2} [X(z) + X^*(z^*)] = Z\{\operatorname{Re}[x(n)]\}$$

This is the required proof. The ROC of above z-transform includes ROC of $X(z)$.

3.3.11 z-transform of Imaginary Part of the Sequence

This property states that if $x(n)$ is a complex valued sequence and

$$x(n) \xrightarrow{z} X(z)$$

$$\text{then } \operatorname{Im}[x(n)] \xrightarrow{z} \frac{1}{2j} [X(z) - X^*(z^*)] \quad \dots (3.3.17)$$

Proof :

Consider the RHS of equation 3.3.17 above. By definition of z-transform we can write,

$$\frac{1}{2j} [X(z) - X^*(z^*)] = \frac{1}{2j} \{Z[x(n)] - Z[x^*(n)]\}$$

In the above equation we have used the conjugation property of equation 3.3.14 to write $X^*(z^*) = Z[x^*(n)]$. Since z-transform satisfies the linearity property we can write above equation as,

$$\frac{1}{2j} [X(z) - X^*(z^*)] = \frac{1}{2j} Z\{x(n) - x^*(n)\} \quad \dots (3.3.18)$$

$$\text{We know that } x(n) = \operatorname{Re}[x(n)] + j \operatorname{Im}[x(n)]$$

$$\text{Hence } x^*(n) = \operatorname{Re}[x(n)] - j \operatorname{Im}[x(n)]$$

Putting the values of $x(n)$ and $x^*(n)$ in equation 3.3.18 and simplifying we get,

$$\begin{aligned} \frac{1}{2j} [X(z) - X^*(z^*)] &= \frac{1}{2j} Z \{2j \operatorname{Im} [x(n)]\} \\ &= Z \{ \operatorname{Im} [x(n)] \} \end{aligned}$$

This is the required proof. The ROC of the above z-transform includes the ROC of $X(z)$.

3.3.12 Parseval's Relation

If $x_1(n)$ and $x_2(n)$ are complex valued sequences, then the Parseval's relation states that,

$$\sum_{n=-\infty}^{\infty} x_1(n) x_2^*(n) = \frac{1}{2\pi j} \oint_c X_1(v) X_2^* \left(\frac{1}{v^*} \right) v^{-1} dv \quad \dots (3.3.19)$$

Proof :

From equation 3.3.12, the inverse z-transform of $x_1(n)$ is given as,

$$x_1(n) = \frac{1}{2\pi j} \oint_c X_1(v) v^{n-1} dv$$

Consider the LHS of Parseval's relation,

$$\sum_{n=-\infty}^{\infty} x_1(n) \cdot x_2^*(n) = \sum_{n=-\infty}^{\infty} \frac{1}{2\pi j} \oint_c X_1(v) v^{n-1} dv x_2^*(n)$$

By putting $x_1(n)$ from above equation

$$= \frac{1}{2\pi j} \oint_c X_1(v) \left[\sum_{n=-\infty}^{\infty} x_2^*(n) v^{n-1} \right] dv$$

In the above equation $v^{n-1} = v^n \cdot v^{-1}$

$$= (v^{-1})^{-n} \cdot v^{-1}$$

$$= \left(\frac{1}{v} \right)^{-n} v^{-1}$$

$$\therefore \sum_{n=-\infty}^{\infty} x_1(n) \cdot x_2^*(n) = \frac{1}{2\pi j} \oint_c X_1(v) \left[\sum_{n=-\infty}^{\infty} x_2^*(n) \left(\frac{1}{v} \right)^{-n} \right] v^{-1} dv \quad \dots (3.3.20)$$

$$\text{In the above equation } \sum_{n=-\infty}^{\infty} x_2^*(n) \left(\frac{1}{v} \right)^{-n} = \sum_{n=-\infty}^{\infty} \left[x_2(n) \left(\frac{1}{v^*} \right)^{-n} \right]^*$$

$$= \left[\sum_{n=-\infty}^{\infty} x_2(n) \left(\frac{1}{v^*} \right)^{-n} \right]^*$$

$$= \left[X_2 \left(\frac{1}{v^*} \right) \right]^*$$

$$= X_2^* \left(\frac{1}{v^*} \right)$$

With this result we can write equation 3.3.20 as,

$$\sum_{n=-\infty}^{\infty} x_1(n) \cdot x_2^*(n) = \frac{1}{2\pi j} \oint_c X_1(v) X_2^* \left(\frac{1}{v^*} \right) v^{-1} dv$$

which is same as the required relation. Thus Parseval's relation is proved.

3.3.13 Initial Value Theorem

If $x(n]$ is causal sequence, then its initial value is given by,

$$x(0) = \lim_{z \rightarrow \infty} X(z) \quad \dots (3.3.21)$$

Proof :

By definition of z-transform,

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

Since $x(n]$ is a causal sequence, $x(n) = 0$ for $n < 0$. Hence above equation becomes,

$$\begin{aligned} X(z) &= \sum_{n=0}^{\infty} x(n) z^{-n} \\ &= x(0) + x(1)z^{-1} + x(2)z^{-2} + x(3)z^{-3} + \dots \end{aligned}$$

Applying the limits as $z \rightarrow \infty$

$$\lim_{z \rightarrow \infty} X(z) = \lim_{z \rightarrow \infty} x(0) + \lim_{z \rightarrow \infty} x(1)z^{-1} + \lim_{z \rightarrow \infty} x(2)z^{-2} + \dots$$

Thus,

$$= x(0) + 0 + 0 + 0 \dots$$

$$\text{Initial value } x(0) = \lim_{z \rightarrow \infty} X(z)$$

Ex.3.3.1 Delayed unit sample sequence.

Determine the z-transform of following function.

$$x_1(n) = \delta(n - k)$$

$$x_2(n) = \delta(n + k)$$

Sol. : In example 3.2.5 we have obtained the z-transform of unit sample sequence $\delta(n)$. It is given by equation 3.2.21 as,

$$Z[\delta(n)] = 1 \quad \text{ROC : Entire z-plane.}$$

Consider $x_1(n) = \delta(n - k)$. This is the unit sample sequence of k samples right shift. The time shifting property of z-transform is given by equation 3.3.2 as,

$$Z[x(n - k)] = z^{-k} X(z)$$

Applying this property to $x_1(n) = \delta(n - k)$ we have,

$$\begin{aligned} Z[x_1(n)] &= Z[\delta(n - k)] \\ &= z^{-k} Z[\delta(n)] && \text{By time shifting property} \\ &= z^{-k} \cdot 1 \\ &= z^{-k} \end{aligned}$$

This z-transform has infinite value at $z = 0$, since $Z[x_1(n)] = z^{-k} = \frac{1}{z^k}$. At $z = 0$ this becomes infinite. It has finite values for remaining values of z . Hence ROC for $Z[x_1(n)]$ is entire z-plane except $z = 0$. Thus we have obtained a z-transform pair,

$$\delta(n-k) \xleftrightarrow{Z} z^{-k}, \quad \text{ROC : Entire } z\text{-plane except } z=0 \quad \dots (3.3.22)$$

Now let us consider the second sequence $x_2(n) = \delta(n+k)$. Again applying the time shifting property of z-transform we have,

$$\begin{aligned} Z[\delta(n+k)] &= z^k Z[\delta(n)] \\ &= z^k \cdot 1 \\ &= z^k \end{aligned}$$

This z-transform has finite values everywhere except at $z = \infty$. Hence ROC for this z-transform is entire z-plane except $z = \infty$. Thus the z-transform pair is,

$$\delta(n+k) \xleftrightarrow{Z} z^k, \quad \text{ROC : Entire } z\text{-plane except } z = \infty \quad \dots (3.3.23)$$

Ex.3.3.2 Determine the z-transform and ROC of the following sequence.

$$x(n) = \left(-\frac{1}{3}\right)^n u(n) - \left(\frac{1}{2}\right)^n u(-n-1) \quad \dots (3.3.24)$$

Sol. : The z-transform of $x(n)$ is,

$$\begin{aligned} X(z) &= Z[x(n)] \\ &= Z\left\{\left(-\frac{1}{3}\right)^n u(n) - \left(\frac{1}{2}\right)^n u(-n-1)\right\} \end{aligned}$$

Since z-transform satisfies linearity property we can write above equation as,

$$X(z) = Z\left\{\left(-\frac{1}{3}\right)^n u(n)\right\} + Z\left\{-\left(\frac{1}{2}\right)^n u(-n-1)\right\} \quad \dots (3.3.25)$$

Here let us use the standard results derived earlier. i.e.,

$$Z\{a^n u(n)\} = \frac{1}{1-az^{-1}}, \quad \text{ROC : } |z| > |a| \text{ from equation 3.2.12}$$

$$\therefore Z\left\{\left(-\frac{1}{3}\right)^n u(n)\right\} = \frac{1}{1+\frac{1}{3}z^{-1}}, \quad \text{ROC : } |z| > \frac{1}{3}$$

$$\text{And } Z\{-a^n u(-n-1)\} = \frac{1}{1-az^{-1}}, \quad \text{ROC : } |z| < |a| \text{ from equation 3.2.19}$$

$$\therefore Z\left\{-\left(\frac{1}{2}\right)^n u(-n-1)\right\} = \frac{1}{1-\frac{1}{2}z^{-1}}, \quad \text{ROC : } |z| < \frac{1}{2}$$

With these results we can write equation 3.3.25 as,

$$X(z) = \frac{1}{1+\frac{1}{3}z^{-1}} + \frac{1}{1-\frac{1}{2}z^{-1}}, \quad \text{ROC : } |z| > \frac{1}{3} \text{ and } |z| < \frac{1}{2}$$

The ROC of $|z| > \frac{1}{3}$ and $|z| < \frac{1}{2}$ can also be written as $\frac{1}{3} < |z| < \frac{1}{2}$. Thus ROC is the region between the circles of radius of $\frac{1}{3}$ and $\frac{1}{2}$. Fig. 3.3.1 shows this ROC.

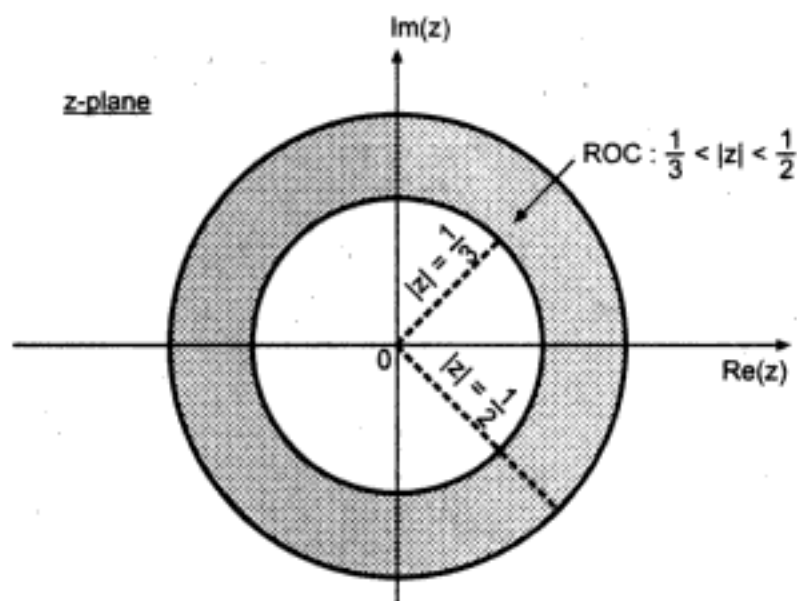


Fig. 3.3.1 ROC of the function of example 3.3.2

Thus the ROC is the intersection or overlap of ROC of individual functions. It is annular region for $\frac{1}{3} < |z| < \frac{1}{2}$ looking like a disk.

Ex.3.3.3 Determine the z-transform of folded unit step sequence i.e.

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

Sol. : In example 3.2.3 we have obtained the z-transform of unit step sequence i.e.,

$$Z[u(n)] = \frac{1}{1-z^{-1}}, \quad \text{ROC} : |z| > 1 \quad \text{By equation 3.2.14}$$

The time reversal property of equation 3.3.4 states that,

$$\text{If } x(n) \xleftrightarrow{Z} X(z), \quad \text{ROC} : r_1 < |z| < r_2$$

$$\text{then } x(-n) \xleftrightarrow{Z} X(z^{-1}), \quad \text{ROC} : \frac{1}{r_2} < |z| < \frac{1}{r_1}$$

Using this property we can write z-transform of $u(-n)$ as,

$$Z[u(-n)] = \frac{1}{1-z}, \quad \text{ROC} : |z| < 1$$

Thus we have one more z-transform pair,

$$u(-n) \xleftrightarrow{Z} \frac{1}{1-z}, \quad \text{ROC} : |z| < 1 \quad \dots (3.3.26)$$

Ex.3.3.4 Determine the z-transform of the signal

$$x(n) = n a^n u(n)$$

And hence determine z-transform of the unit ramp signal $n u(n)$.

[Dec - 99]

Sol. : Let $x_1(n) = a^n u(n)$

From equation 3.2.12 the z-transform of $a^n u(n)$ is given as

$$Z[a^n u(n)] = \frac{1}{1 - az^{-1}}, \quad \text{ROC : } |z| > |a|$$

$$\therefore X_1(z) = \frac{1}{1 - az^{-1}}, \quad \text{ROC : } |z| > |a|$$

Now the given function is,

$$x(n) = n a^n u(n)$$

i.e. $x(n) = n x_1(n)$ since $x_1(n) = a^n u(n)$

$$\therefore X(z) = Z[n x_1(n)] \quad \dots (3.3.27)$$

The differentiation in z-domain property of equation 3.3.5 states that,

$$Z[n x(n)] = -z \frac{d}{dz} X(z)$$

Applying this property to equation 3.3.27 we have,

$$\begin{aligned} X(z) &= -z \frac{d}{dz} X_1(z), \quad \text{ROC : } |z| > |a| \\ &= -z \frac{d}{dz} \frac{1}{1 - az^{-1}} \\ &= \frac{az^{-1}}{(1 - az^{-1})^2} \end{aligned}$$

Thus we have obtained the z-transform pair,

$$n a^n u(n) \xleftrightarrow{z} \frac{az^{-1}}{(1 - az^{-1})^2}, \quad \text{ROC : } |z| > |a| \quad \dots (3.3.28)$$

z-transform of unit ramp sequence :

The unit ramp sequence is given as,

$$x(n) = n u(n)$$

$$\therefore X(z) = Z[n u(n)]$$

By putting $a=1$ in equation 3.3.28 we can get the z-transform of unit ramp sequence.

Hence,

$$n u(n) \xleftrightarrow{z} \frac{z^{-1}}{(1 - z^{-1})^2}, \quad \text{ROC : } |z| > 1 \quad \dots (3.3.29)$$

Ex.3.3.5 Find out the z-transform of causal cosine sequence which is given as,

$$x(n) = \cos(\omega_0 n) u(n)$$

Sol. : Using Eulers identity we have,

$$\cos \theta = \frac{e^{j\theta} + e^{-j\theta}}{2}$$

Hence
$$\cos(\omega_0 n) = \frac{e^{j\omega_0 n} + e^{-j\omega_0 n}}{2}$$

$$\begin{aligned} \therefore x(n) &= \frac{1}{2} [e^{j\omega_0 n} + e^{-j\omega_0 n}] u(n) \\ &= \frac{1}{2} e^{j\omega_0 n} u(n) + \frac{1}{2} e^{-j\omega_0 n} u(n) \end{aligned}$$

Taking z-transform of above function,

$$X(z) = Z \left\{ \frac{1}{2} e^{j\omega_0 n} u(n) + \frac{1}{2} e^{-j\omega_0 n} u(n) \right\}$$

Using the linearity property of equation 3.3.1 we can write above equation as,

$$X(z) = Z \left\{ \frac{1}{2} e^{j\omega_0 n} u(n) \right\} + Z \left\{ \frac{1}{2} e^{-j\omega_0 n} u(n) \right\}$$

Here let $a = e^{j\omega_0}$ and $b = e^{-j\omega_0}$, then we can write,

$$X(z) = \frac{1}{2} Z [a^n u(n)] + \frac{1}{2} Z [b^n u(n)] \quad \dots (3.3.30)$$

From equation 3.2.12 we can write,

$$Z [a^n u(n)] = \frac{1}{1 - az^{-1}}, \quad \text{ROC : } |z| > |a| \quad \dots (3.3.31)$$

We have
$$a = e^{j\omega_0} = \cos \omega_0 + j \sin \omega_0$$

$$\therefore |a| = \sqrt{\cos^2 \omega_0 + \sin^2 \omega_0} = 1$$

Hence we can write equation 3.3.31 as,

$$Z [a^n u(n)] = \frac{1}{1 - e^{j\omega_0} z^{-1}}, \quad \text{ROC : } |z| > 1 \quad \dots (3.3.32)$$

Similarly the z-transform of second term in equation 3.3.30 can be written as,

$$Z [b^n u(n)] = \frac{1}{1 - bz^{-1}}, \quad \text{ROC : } |z| > |b| \quad \text{from equation 3.2.12} \quad \dots (3.3.33)$$

We have
$$b = e^{-j\omega_0} = \cos \omega_0 - j \sin \omega_0$$

$$\therefore |b| = \sqrt{\cos^2 \omega_0 + \sin^2 \omega_0} = 1$$

Hence we can write equation 3.3.33 as,

$$Z [b^n u(n)] = \frac{1}{1 - e^{-j\omega_0} z^{-1}}, \quad \text{ROC : } |z| > 1 \quad \dots (3.3.34)$$

Putting the individual z-transform values from above equation and equation 3.3.32 in equation 3.3.30 we get,

$$\begin{aligned} X(z) &= \frac{1}{2} \cdot \frac{1}{1 - e^{j\omega_0} z^{-1}} + \frac{1}{2} \cdot \frac{1}{1 - e^{-j\omega_0} z^{-1}}, \text{ ROC : } |z| > 1 \\ &= \frac{1}{2} \left\{ \frac{1}{1 - e^{j\omega_0} z^{-1}} + \frac{1}{1 - e^{-j\omega_0} z^{-1}} \right\} \\ &= \frac{1}{2} \left\{ \frac{1 - e^{-j\omega_0} z^{-1} + 1 - e^{j\omega_0} z^{-1}}{1 - e^{-j\omega_0} z^{-1} - e^{j\omega_0} z^{-1} + e^{j\omega_0} z^{-1} e^{-j\omega_0} z^{-1}} \right\} \\ &= \frac{1}{2} \left\{ \frac{2 - z^{-1} (e^{j\omega_0} + e^{-j\omega_0})}{1 - z^{-1} (e^{j\omega_0} + e^{-j\omega_0}) + z^{-2}} \right\} \end{aligned}$$

Here let us use Euler's identity i.e. $2 \cos \theta = e^{j\theta} + e^{-j\theta}$. Hence above equation can be further simplified as,

$$X(z) = \frac{1}{2} \left\{ \frac{2 - 2z^{-1} \cos \omega_0}{1 - 2z^{-1} \cos \omega_0 + z^{-2}} \right\}$$

$$\therefore X(z) = \frac{1 - z^{-1} \cos \omega_0}{1 - 2z^{-1} \cos \omega_0 + z^{-2}}, \text{ ROC : } |z| > 1$$

Thus we obtained the standard z-transform pair as,

$$\cos(\omega_0 n) u(n) \xleftrightarrow{z} \frac{1 - z^{-1} \cos \omega_0}{1 - 2z^{-1} \cos \omega_0 + z^{-2}}, \text{ ROC : } |z| > 1 \quad \dots (3.3.35)$$

Ex.3.3.6 Determine the z-transform of

$$x(n) = \sin(\omega_0 n) u(n)$$

Sol. : Using Euler's identity we have

$$\sin \theta = \frac{e^{j\theta} - e^{-j\theta}}{2j}$$

$$\therefore \sin(\omega_0 n) = \frac{1}{2j} [e^{j\omega_0 n} - e^{-j\omega_0 n}]$$

$$\begin{aligned} \therefore x(n) &= \frac{1}{2j} [e^{j\omega_0 n} - e^{-j\omega_0 n}] u(n) \\ &= \frac{1}{2j} e^{j\omega_0 n} u(n) - \frac{1}{2j} e^{-j\omega_0 n} u(n) \end{aligned}$$

Taking z-transform of above function,

$$\begin{aligned} X(z) &= Z \left\{ \frac{1}{2j} e^{j\omega_0 n} u(n) - \frac{1}{2j} e^{-j\omega_0 n} u(n) \right\} \\ &= \frac{1}{2j} Z \left\{ e^{j\omega_0 n} u(n) \right\} - \frac{1}{2j} Z \left\{ e^{-j\omega_0 n} u(n) \right\} \text{ By linearity property} \quad \dots (3.3.36) \end{aligned}$$

In the previous example, we have obtained z-transform of $e^{j\omega_0 n} u(n)$, i.e.,

$$Z \left\{ e^{j\omega_0 n} u(n) \right\} = \frac{1}{1 - e^{j\omega_0} z^{-1}}, \quad \text{ROC} : |z| > 1 \quad \text{By equation 3.3.32}$$

Similarly z-transform of $e^{-j\omega_0 n} u(n)$ also we have obtained in previous example. i.e.,

$$Z \left\{ e^{-j\omega_0 n} u(n) \right\} = \frac{1}{1 - e^{-j\omega_0} z^{-1}}, \quad \text{ROC} : |z| > 1 \quad \text{By equation 3.3.34}$$

Putting the above two results in equation 3.3.36 we get,

$$\begin{aligned} X(z) &= \frac{1}{2j} \cdot \frac{1}{1 - e^{j\omega_0} z^{-1}} - \frac{1}{2j} \cdot \frac{1}{1 - e^{-j\omega_0} z^{-1}}, \quad \text{ROC} : |z| > 1 \\ &= \frac{1}{2j} \left\{ \frac{1}{1 - e^{j\omega_0} z^{-1}} - \frac{1}{1 - e^{-j\omega_0} z^{-1}} \right\} \\ &= \frac{1}{2j} \left\{ \frac{1 - e^{-j\omega_0} z^{-1} - 1 + e^{j\omega_0} z^{-1}}{1 - e^{j\omega_0} z^{-1} - e^{-j\omega_0} z^{-1} + e^{j\omega_0} z^{-1} \cdot e^{-j\omega_0} z^{-1}} \right\} \\ &= \frac{1}{2j} \left\{ \frac{z^{-1} (e^{j\omega_0} - e^{-j\omega_0})}{1 - z^{-1} (e^{j\omega_0} + e^{-j\omega_0}) + z^{-2}} \right\} \\ &= \left\{ \frac{z^{-1} \frac{e^{j\omega_0} - e^{-j\omega_0}}{2j}}{1 - z^{-1} (e^{j\omega_0} + e^{-j\omega_0}) + z^{-2}} \right\} \end{aligned}$$

Applying the Euler's identity to above equation we can write,

$$X(z) = \frac{z^{-1} \sin \omega_0}{1 - 2z^{-1} \cos \omega_0 + z^{-2}}, \quad \text{ROC} : |z| > 1$$

Thus we have obtained the z-transform pair as,

$$\sin(\omega_0 n) u(n) \xleftrightarrow{z} \frac{z^{-1} \sin \omega_0}{1 - 2z^{-1} \cos \omega_0 + z^{-2}}, \quad \text{ROC} : |z| > 1 \quad \dots (3.3.37)$$

Ex.3.3.7 Determine the z-transform of the discrete time signal,

$$x(n) = a^n \cos(\omega_0 n) u(n)$$

[Dec - 98]

Sol. : Let $x_1(n) = \cos(\omega_0 n) u(n)$

The z-transform of $\cos(\omega_0 n) u(n)$ is given by equation 3.3.35 i.e.,

$$X_1(z) = \frac{1 - z^{-1} \cos \omega_0}{1 - 2z^{-1} \cos \omega_0 + z^{-2}}, \quad \text{ROC} : |z| > 1 \quad \dots (3.3.38)$$

Now consider the given equation i.e.,

$$\begin{aligned}
 x(n) &= a^n \cos(\omega_0 n) u(n) \\
 &= a^n x_1(n) \\
 \therefore X(z) &= Z \left\{ a^n x_1(n) \right\} \quad \dots (3.3.39)
 \end{aligned}$$

Here let us use the scaling property of equation 3.3.3 i.e.,

$$Z \left\{ a^n x(n) \right\} = X \left(\frac{z}{a} \right), \quad \text{ROC} : |a|r_1 < |z| < |a|r_2$$

Applying this property to equation 3.3.39 we have,

$$X(z) = X_1 \left(\frac{z}{a} \right)$$

$X_1 \left(\frac{z}{a} \right)$ is obtained by replacing z by $\frac{z}{a}$ in $X_1(z)$ of equation 3.3.38 i.e.,

$$X(z) = \frac{1 - \left(\frac{z}{a} \right)^{-1} \cos \omega_0}{1 - 2 \left(\frac{z}{a} \right)^{-1} \cos \omega_0 + \left(\frac{z}{a} \right)^{-2}}, \quad \text{ROC} : |z| > 1 \cdot |a|$$

Thus we obtained the z-transform pair as,

$$a^n \cos(\omega_0 n) u(n) \xleftrightarrow{z} \frac{1 - \left(\frac{z}{a} \right)^{-1} \cos \omega_0}{1 - 2 \left(\frac{z}{a} \right)^{-1} \cos \omega_0 + \left(\frac{z}{a} \right)^{-2}}, \quad \text{ROC} : |z| > |a| \quad \dots (3.3.40)$$

Ex.3.3.8 Determine the z-transform of the following signal,

$$x(n) = a^n \sin(\omega_0 n) u(n)$$

Sol. : Let

$$x_1(n) = \sin(\omega_0 n) u(n)$$

The z-transform of $\sin(\omega_0 n) u(n)$ is given by equation 3.3.37 i.e.,

$$X_1(z) = \frac{z^{-1} \sin \omega_0}{1 - 2z^{-1} \cos \omega_0 + z^{-2}}, \quad \text{ROC} : |z| > 1 \quad \dots (3.3.41)$$

Now consider the given equation i.e.,

$$x(n) = a^n \sin(\omega_0 n) u(n)$$

$$= a^n x_1(n)$$

$$\therefore X(z) = Z \left\{ a^n x_1(n) \right\} \quad \dots (3.3.42)$$

Here let us use the scaling property of equation 3.3.3 i.e.,

$$Z \left\{ a^n x(n) \right\} = X \left(\frac{z}{a} \right), \quad \text{ROC} : |a|r_1 < |z| < |a|r_2$$

Applying this property to equation 3.3.42 we have,

$$X(z) = X_1\left(\frac{z}{a}\right)$$

Here $X_1\left(\frac{z}{a}\right)$ is obtained by replacing z by $\frac{z}{a}$ in $X_1(z)$ of equation 3.3.41 i.e.,

$$X(z) = \frac{\left(\frac{z}{a}\right)^{-1} \sin \omega_0}{1 - 2\left(\frac{z}{a}\right)^{-1} \cos \omega_0 + \left(\frac{z}{a}\right)^{-2}}, \quad \text{ROC : } |z| > 1 \cdot |a|$$

Thus we have obtained the z-transform pair as,

$$a^n \sin(\omega_0 n) u(n) \xleftrightarrow{z} \frac{\left(\frac{z}{a}\right)^{-1} \sin \omega_0}{1 - 2\left(\frac{z}{a}\right)^{-1} \cos \omega_0 + \left(\frac{z}{a}\right)^{-2}}, \quad \text{ROC : } |z| > |a| \quad \dots (3.3.43)$$

Ex.3.3.9 Determine the z-transform of the following signal with the help of linearity and shifting properties.

$$x(n) = \begin{cases} 1 & \text{for } 0 \leq n \leq N-1 \\ 0 & \text{elsewhere} \end{cases}$$

Sol. : The given function $x(n)$ is a pulse having ' N ' samples of amplitude 1. This pulse can be obtained from two unit step sequences as follows :

$$x(n) = u(n) - u(n-N)$$

Taking z-transform, $X(z) = Z\{u(n) - u(n-N)\}$

By linearity property we can write above equation as,

$$X(z) = Z[u(n)] - Z[u(n-N)] \quad \dots (3.3.44)$$

The z-transform of unit step sequence is given by equation 3.2.14 as,

$$Z[u(n)] = \frac{1}{1-z^{-1}}, \quad \text{ROC : } |z| > 1 \quad \dots (3.3.45)$$

The time shifting property of z-transform states that,

$$x(n-k) \xleftrightarrow{z} z^{-k} X(z)$$

$$\therefore Z[u(n-N)] = z^{-N} Z[u(n)]$$

$$= z^{-N} \cdot \frac{1}{1-z^{-1}}$$

Putting results of above equation and equation 3.3.45 in equation 3.3.44 we obtain,

$$\begin{aligned} X(z) &= \frac{1}{1-z^{-1}} - \frac{z^{-N}}{1-z^{-1}} \\ &= \frac{1-z^{-N}}{1-z^{-1}}, \quad \text{ROC : } |z| > 1 \end{aligned}$$

Ex.3.3.10 Determine the z-transform and the ROC of the signal,

$$x(n) = [3 \cdot (4^n) - 4(2^n)] u(n)$$

[May-97, Dec-99]

Sol. : To find z-transform of given signal let us use the standard relation given in table 3.3.2, i.e.,

$$a^n u(n) \xleftrightarrow{z} \frac{1}{1 - az^{-1}}, \quad \text{ROC : } |z| > |a|$$

$$\therefore 4^n u(n) \xleftrightarrow{z} \frac{1}{1 - 4z^{-1}}, \quad \text{ROC : } |z| > 4$$

and $2^n u(n) \xleftrightarrow{z} \frac{1}{1 - 2z^{-1}}, \quad \text{ROC : } |z| > 2$

Therefore z-transform of given signal becomes,

$$\begin{aligned} X(z) &= 3 Z \left\{ (4)^n u(n) \right\} - 4 Z \left\{ (2)^n u(n) \right\} \\ &= \frac{3}{1 - 4z^{-1}} - \frac{4}{1 - 2z^{-1}}, \quad \text{ROC : } |z| > 4 \end{aligned}$$

The common ROC for $|z| > 4$ and $|z| > 2$ is equal to $|z| > 4$.

Ex.3.3.11 Determine the z-transform including ROC for the following.

(i) $\left(\frac{1}{2}\right)^n u(-n)$ (ii) $\left(\frac{1}{2}\right)^n \{u(n) - u(n-10)\}$

[May-2000]

Sol. : (i) The given function is,

$$x(n) = \left(\frac{1}{2}\right)^n u(-n)$$

In example 3.4.10 we have obtained the z-transform of $a^n u(-n)$. It is given by equation 3.4.50 as,

$$Z \left\{ a^n u(-n) \right\} = \frac{1}{1 - a^{-1}z}, \quad \text{ROC : } |z| < |a|$$

Here in the given example, $a = \frac{1}{2}$, i.e.,

$$Z \left\{ \left(\frac{1}{2}\right)^n u(-n) \right\} = \frac{1}{1 - \left(\frac{1}{2}\right)^{-1}z}, \quad \text{ROC : } |z| < \frac{1}{2}$$

This is the required z-transform of given function.

(ii) The given function is,

$$x(n) = \left(\frac{1}{2}\right)^n \{u(n) - u(n-10)\}$$

$$\begin{aligned}
 &= \left(\frac{1}{2}\right)^n u(n) - \left(\frac{1}{2}\right)^n u(n-10) \\
 \therefore X(z) &= Z\left\{\left(\frac{1}{2}\right)^n u(n)\right\} - Z\left\{\left(\frac{1}{2}\right)^n u(n-10)\right\} \quad \dots (3.3.46)
 \end{aligned}$$

From Table 3.3.2 we can write z-transform of $\left(\frac{1}{2}\right)^n u(n)$ as,

$$\left(\frac{1}{2}\right)^n u(n) \xleftrightarrow{z} \frac{1}{1 - \left(\frac{1}{2}\right)z^{-1}}, \quad \text{ROC: } |z| > \frac{1}{2} \quad \dots (3.3.47)$$

Now let us consider z-transform of $\left(\frac{1}{2}\right)^n u(n-10)$

$\left(\frac{1}{2}\right)^n u(n-10)$ can be written as,

$$\left(\frac{1}{2}\right)^n u(n-10) = \begin{cases} \left(\frac{1}{2}\right)^n & \text{for } n \geq 10 \\ 0 & \text{for } n < 10 \end{cases}$$

$$\begin{aligned}
 \therefore Z\left\{\left(\frac{1}{2}\right)^n u(n-10)\right\} &= \sum_{n=10}^{\infty} \left(\frac{1}{2}\right)^n z^{-n} \\
 &= \sum_{n=10}^{\infty} \left(\frac{1}{2}z^{-1}\right)^n
 \end{aligned}$$

Here we use the standard relation given by equation 3.2.24. i.e.,

$$\begin{aligned}
 Z\left\{\left(\frac{1}{2}\right)^n u(n-10)\right\} &= \frac{\left(\frac{1}{2}z^{-1}\right)^{10} - \left(\frac{1}{2}z^{-1}\right)^{\infty}}{1 - \left(\frac{1}{2}\right)z^{-1}} \\
 &= \frac{\left(\frac{1}{2}z^{-1}\right)^{10}}{1 - \left(\frac{1}{2}\right)z^{-1}} \quad \dots (3.3.48)
 \end{aligned}$$

Putting the values obtained in above equation and equation 3.3.47 in equation 3.3.46 we get,

$$X(z) = \frac{1}{1 - \left(\frac{1}{2}\right)z^{-1}} - \frac{\left(\frac{1}{2}z^{-1}\right)^{10}}{1 - \left(\frac{1}{2}\right)z^{-1}}$$

$$= \frac{1 - \left(\frac{1}{2} z^{-1}\right)^{10}}{1 - \left(\frac{1}{2}\right) z^{-1}}, \quad \text{ROC : } |z| > \frac{1}{2}$$

This is the required z-transform.

Ex.3.3.12 Determine the z-transform of

$$x(n) = \frac{a^n}{n!} \quad \text{for } n \geq 0$$

Sol. : By definition of z-transform we have,

$$\begin{aligned} X(z) &= \sum_{n=-\infty}^{\infty} x(n) z^{-n} \\ &= \sum_{n=0}^{\infty} \frac{a^n}{n!} z^{-n} \\ &= \sum_{n=0}^{\infty} \frac{(az^{-1})^n}{n!} \end{aligned} \quad \dots (3.3.49)$$

Here we will use the standard exponential series,

$$e^{-x} = \sum_{n=0}^{\infty} \frac{x^n}{n!} \quad \dots (3.3.50)$$

\therefore Equation 3.3.49 becomes,

$$X(z) = e^{az^{-1}}$$

This is the required z-transform.

Table 3.3.1 : Properties of z-transform

Sr. No.	Name of the property	Time domain sequence	z-domain representation	Region of convergence
		$x(n)$ $x_1(n)$ $x_2(n)$	$X(z)$ $X_1(z)$ $X_2(z)$	$r_1 < z < r_2$ ROC1 ROC2
1	Linearity	$a_1 x_1(n) + a_2 x_2(n)$	$a_1 X_1(z) + a_2 X_2(z)$	Intersection of ROC1 and ROC2
2	Time shifting	$x(n-k)$	$z^{-k} X(z)$	ROC of $X(z)$ except $z=0$ if $k > 0$ and $z=\infty$ if $k < 0$
3	Scaling in z-domain	$a^n x(n)$	$X\left(\frac{z}{a}\right)$	$ a r_1 < z < a r_2$

Sr. No.	Name of the property	Time domain sequence	z-domain representation	Region of convergence
4	Time reversal	$x(-n)$	$X(z^{-1})$	$\frac{1}{r_2} < z < \frac{1}{r_1}$
5	Differentiation in z-domain	$nx(n)$	$-z \frac{d}{dz} X(z)$	Same as ROC of $X(z)$
6	Convolution in time domain	$x_1(n) * x_2(n)$	$X_1(z) \cdot X_2(z)$	Intersection of ROC 1 and ROC 2
7	Correlation	$x_1(l) * x_2(-l)$	$X_1(z) \cdot X_2(z^{-1})$	Intersection of ROC of $X_1(z)$ and $X_2(z^{-1})$
8	Multiplication	$x_1(n) \cdot x_2(n)$	$\frac{1}{2\pi j} \oint_c X_1(v) X_2\left(\frac{z}{v}\right) v^{-1} dv$	$r_{2l} < v < r_{2u} < v $ and $r_{1l} < v < r_{1u}$
9	Conjugation	$x^*(n)$	$X^*(z^*)$	Same as ROC of $X(z)$
10	Real part of $x(n)$	$Re[x(n)]$	$\frac{1}{2} [X(z) + X^*(z^*)]$	Includes ROC of $X(z)$
11	Imaginary part of $x(n)$	$Im[x(n)]$	$\frac{1}{2j} [X(z) - X^*(z^*)]$	Includes ROC of $X(z)$
12	Initial value theorem	Causal $x(n)$	$x(0) = \lim_{z \rightarrow \infty} X(z)$	
13	Parseval's relation	$\sum_{n=-\infty}^{\infty} x_1(n) x_2^*(n) = \frac{1}{2\pi j} \oint_c X_1(v) X_2^*\left(\frac{1}{v}\right) v^{-1} dv$		

Table 3.3.2 : Standard z-transfer pairs

Sr.No.	Time domain sequence $x(n)$	z-transform $X(z)$	ROC
1	$\delta(n)$	1	Complete z-plane
2	$u(n)$	$\frac{1}{1-z^{-1}}$	$ z > 1$
3	$a^n u(n)$	$\frac{1}{1-az^{-1}}$	$ z > a $
4	$-a^n u(-n-1)$	$\frac{1}{1-az^{-1}}$	$ z < a $

Sr.No.	Time domain sequence $x(n)$	z-transform $X(z)$	ROC
5	$n a^n u(n)$	$\frac{az^{-1}}{(1-az^{-1})^2}$	$ z > a $
6	$-n a^n u(-n-1)$	$\frac{az^{-1}}{(1-az^{-1})^2}$	$ z < a $
7	$\cos(\omega_0 n) u(n)$	$\frac{1-z^{-1} \cos \omega_0}{1-2z^{-1} \cos \omega_0 + z^{-2}}$	$ z > 1$
8	$\sin(\omega_0 n) u(n)$	$\frac{z^{-1} \sin \omega_0}{1-2z^{-1} \cos \omega_0 + z^{-2}}$	$ z > 1$
9	$a^n \cos(\omega_0 n) u(n)$	$\frac{1-\left(\frac{z}{a}\right)^{-1} \cos \omega_0}{1-2\left(\frac{z}{a}\right)^{-1} \cos \omega_0 + \left(\frac{z}{a}\right)^{-2}}$	$ z > a $
10	$a^n \sin(\omega_0 n) u(n)$	$\frac{\left(\frac{z}{a}\right)^{-1} \sin \omega_0}{1-2\left(\frac{z}{a}\right)^{-1} \cos \omega_0 + \left(\frac{z}{a}\right)^{-2}}$	$ z > a $
11	a^n for $0 \leq n \leq N-1$ 0 otherwise	$\frac{1-a^N z^{-N}}{1-az^{-1}}$	$ z > 0$
12	$2 A_k r_k^n \cos(\beta_k n + \alpha_k) u(n)$ $A_k = A_k e^{j\alpha_k}$ and $p_k = r_k e^{j\beta_k}$	$\frac{A_k}{1-p_k z^{-1}} + \frac{A_k^*}{1-p_k^* z^{-1}}$	$ z > p_k = r_k$

3.4 Inverse z-transform

In the last two sections we studied z-transform and its properties. Always it is required to transfer the signal to z-domain and get it back in time domain. The signal can be converted from z-domain to time domain with the help of inverse z-transform. The inverse z-transform can be obtained by the following methods.

- (i) Contour integration
- (ii) Partial fraction expansion
- (iii) Power series expansion

Inverse z-transform can be obtained by the contour integration and this is the basic method. Without going into much mathematical theory, we can state the inversion formula based on contour integration as,

$$x(n) = \frac{1}{2\pi j} \oint_c X(z) z^{n-1} dz \quad \dots (3.4.1)$$

This formula is based on cauchy residue theorem. Almost all the z-transforms we use in practice are rational. A partial fraction expansion is the more simpler way to find inverse z-transform. Next subsection discusses this technique.

3.4.1 Inverse z-transform Using Partial Fraction Expansion

In this method, the function $X(z)$ is expressed as,

$$X(z) = \alpha_1 X_1(z) + \alpha_2 X_2(z) + \dots + \alpha_k X_k(z) \quad \dots (3.4.2)$$

Here $\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_k$ are the constants and $X_1(z), X_2(z), \dots, X_k(z)$ are the standard z-transforms where inverse z-transforms are known. For example, their inverse z-transforms are $x_1(n), x_2(n), \dots, x_k(n)$ respectively i.e.,

$$x_1(n) \xleftrightarrow{z} X_1(z), \quad x_2(n) \xleftrightarrow{z} X_2(z), \dots, \quad x_k(n) \xleftrightarrow{z} X_k(z)$$

Then by linearity property of z-transform we can obtain the sequence $x(n)$ as,

$$x(n) = \alpha_1 x_1(n) + \alpha_2 x_2(n) + \dots + \alpha_k x_k(n) \quad \dots (3.4.3)$$

This method is possible for the z-transforms which are rational in nature and can be expressed as the ratio of two polynomials i.e.,

$$X(z) = \frac{N(z)}{D(z)} = \frac{b_0 + b_1 z^{-1} + \dots + b_M z^{-M}}{a_0 + a_1 z^{-1} + \dots + a_N z^{-N}} \quad \dots (3.4.4)$$

Here $a_0 = 1$. If $a_0 \neq 1$, then the polynomials are adjusted accordingly. In the above equation $a_N \neq 0$ and $M < N$, then it is called proper form. Otherwise the function $X(z)$ is arranged such that it becomes proper. With $a_0 = 1$, above equation can be written as,

$$X(z) = \frac{N(z)}{D(z)} = \frac{b_0 + b_1 z^{-1} + \dots + b_M z^{-M}}{1 + a_1 z^{-1} + \dots + a_N z^{-N}} \quad \dots (3.4.5)$$

Multiplying both the numerator and denominator by z^N in above equation we get,

$$X(z) = \frac{b_0 z^N + b_1 z^{N-1} + \dots + b_M z^{N-M}}{z^N + a_1 z^{N-1} + \dots + a_N}$$

Let us write the above equation as,

$$\frac{X(z)}{z} = \frac{b_0 z^{N-1} + b_1 z^{N-2} + \dots + b_M z^{N-M-1}}{z^N + a_1 z^{N-1} + \dots + a_N} \quad \dots (3.4.6)$$

The denominator polynomial can be written in the factor form as,

$$z^N + a_1 z^{N-1} + \dots + a_N = (z - p_1)(z - p_2) \dots (z - p_N)$$

Hence equation 3.4.6 can be written as,

$$\frac{X(z)}{z} = \frac{b_0 z^{N-1} + b_1 z^{N-2} + \dots + b_M z^{N-M-1}}{(z - p_1)(z - p_2) \dots (z - p_N)}$$

The above equation can be written in partial fraction expansion form as follows,

$$\frac{X(z)}{z} = \frac{A_1}{z - p_1} + \frac{A_2}{z - p_2} + \dots + \frac{A_N}{z - p_N} \quad \dots (3.4.7)$$

Here the coefficient A_k can be obtained as,

$$A_k = \left. \frac{(z - p_k) X(z)}{z} \right|_{z=p_k}, \quad k=1, 2, \dots, N \quad \dots (3.4.8)$$

This formula works only when $p_1 \neq p_2 \neq p_3 \neq \dots \neq p_N$, i.e. they have distinct values. Equation 3.4.7 can be further expressed as,

$$\begin{aligned} X(z) &= \frac{A_1 z}{z - p_1} + \frac{A_2 z}{z - p_2} + \dots + \frac{A_N z}{z - p_N} \\ &= \frac{A_1}{1 - p_1 z^{-1}} + \frac{A_2}{1 - p_2 z^{-1}} + \dots + \frac{A_N}{1 - p_N z^{-1}} \end{aligned} \quad \dots (3.4.9)$$

Here we use two standard z-transform pairs,

$$a^n u(n) \xleftrightarrow{z} \frac{1}{1 - az^{-1}}, \quad \text{ROC} : |z| > |a|$$

and $-a^n u(-n-1) \xleftrightarrow{z} \frac{1}{1 - az^{-1}}, \quad \text{ROC} : |z| < |a|$

Depending upon the specified ROC, the sequence $x(n)$ can be obtained from equation 3.4.9 by applying above two standard relations. Thus for equation 3.4.9 we can write,

$$\text{IZT} \left\{ \frac{1}{1 - p_k z^{-1}} \right\} = \begin{cases} (p_k)^n u(n) & \text{if ROC} : |z| > |p_k| \\ \text{causal sequences} & \\ -(p_k)^n u(-n-1) & \text{if ROC} : |z| < |p_k| \\ \text{anticausal sequences} & \end{cases} \quad \dots (3.4.10)$$

Here 'IZT' means inverse z-transform of the quantity in brackets. The following example illustrates this procedure.

Ex.3.4.1 Determine the inverse z-transform of the following function

$$X(z) = \frac{1}{1 - 15z^{-1} + 05z^{-2}}$$

For following ROC

(i) ROC : $|z| > 1$

(ii) ROC : $|z| < 0.5$

(iii) ROC : $0.5 < |z| < 1$

Sol. : Here $N(z) = 1 \quad \therefore \quad M = 0$

and $D(z) = 1 - 15z^{-1} + 05z^{-2} \quad \therefore \quad N = 2, a_0 = 1 \text{ and } a_N = 0.5$

Hence $M < N$, $a_N \neq 0$ and $a_0 = 1$. This shows that the given function is in proper form. Therefore we can proceed with the next step. That is to multiply numerator and denominator by z^N . Here we have to multiply by z^2 . Hence the given equation becomes,

$$\begin{aligned} X(z) &= \frac{z^2}{z^2} \times \frac{1}{1 - 15z^{-1} + 05z^{-2}} \\ &= \frac{z^2}{z^2 - 15z + 05} \end{aligned} \quad \dots (3.4.11)$$

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

there is one zero at $z = -1$.

Thus both the poles p_1 and p_2 of this system are inside the unit circle $|z|=1$. The system is stable if poles of $H(z)$ are inside the unit circle. Hence this system is stable.

The input to the system is,

$$x(n) = nu(n)$$

The z-transform of $nu(n)$ is given by equation 3.3.29 as,

$$X(z) = Z\{nu(n)\} = \frac{z^{-1}}{(1-z^{-1})^2}$$

Let us rearrange $X(z)$ such that powers of z will be positive i.e.,

$$X(z) = \frac{z}{(z-1)^2}$$

The z-transform of output $Y(z)$ is given as,

$$\begin{aligned} Y(z) &= H(z) \cdot X(z) \\ &= \frac{z+1}{(z-0.4)(z-0.3)} \cdot \frac{z}{(z-1)^2} \end{aligned}$$

The above equation can be written as,

$$\frac{Y(z)}{z} = \frac{z+1}{(z-0.4)(z-0.3)(z-1)^2} \quad \dots (3.6.8)$$

In above equation there is $(z-1)^2$ term in the denominator. Such equation can be expanded in partial fraction as,

$$\frac{Y(z)}{z} = \frac{A_1}{z-0.4} + \frac{A_2}{z-0.3} + \frac{A_3}{z-1} + \frac{A_4}{(z-1)^2} \quad \dots (3.6.9)$$

Here A_1, A_2, A_4 can be obtained using equation 3.4.8 as follows :

$$\begin{aligned} A_1 &= (z-0.4) \left. \frac{Y(z)}{z} \right|_{z=0.4} = \left. \frac{z+1}{(z-0.3)(z-1)^2} \right|_{z=0.4} \\ &= \frac{0.4+1}{(0.4-0.3)(0.4-1)^2} = 38.89 \end{aligned}$$

$$\begin{aligned} A_2 &= (z-0.3) \left. \frac{Y(z)}{z} \right|_{z=0.3} = \left. \frac{z+1}{(z-0.4)(z-1)^2} \right|_{z=0.3} \\ &= \frac{0.3+1}{(0.3-0.4)(0.3-1)^2} = -26.53 \end{aligned}$$

$$A_4 = (z-1)^2 \left. \frac{Y(z)}{z} \right|_{z=1} = \left. \frac{z+1}{(z-0.4)(z-0.3)} \right|_{z=1}$$

$$= \frac{1+1}{(1-0.4)(1-0.3)} = 4.76$$

A_3 can be obtained by equation 3.4.32 i.e.,

$$\begin{aligned} A_3 &= \frac{d}{dz} \left[(z-1)^2 \cdot \frac{Y(z)}{z} \right]_{z=1} \\ &= \frac{d}{dz} \left[\frac{z+1}{(z-0.4)(z-0.3)} \right]_{z=1} \\ &= \frac{(z-0.4)(z-0.3)1 - (z+1)(2z-0.7)}{(z^2 - 0.7z + 0.12)^2} \Big|_{z=1} \\ &= -1235 \end{aligned}$$

Thus equation 3.6.9 becomes,

$$\frac{Y(z)}{z} = \frac{3889}{z-0.4} - \frac{2653}{z-0.3} - \frac{1235}{z-1} + \frac{476}{(z-1)^2}$$

$$\therefore Y(z) = \frac{3889z}{z-0.4} - \frac{2653z}{z-0.3} - \frac{1235z}{z-1} + \frac{476z}{(z-1)^2}$$

$$\therefore Y(z) = \frac{3889}{1-0.4z^{-1}} - \frac{2653}{1-0.3z^{-1}} - \frac{1235}{1-z^{-1}} + \frac{476z^{-1}}{(1-z^{-1})^2}$$

The inverse z-transform of this equation can be obtained from Table 3.3.2 as,

$$y(n) = 3889(0.4)^n u(n) - 2653(0.3)^n u(n) - 1253(1)^n u(n) + 476 \cdot n(1)^n u(n)$$

This equation can be simplified as,

$$y(n) = [3889(0.4)^n - 2653(0.3)^n - 1253 + 476n] u(n)$$

This is the required output equation.

Ex.3.6.4 For the following system obtain the system function $H(z)$ and plot its poles and zeros against unit circle.

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

[May-98]

Sol. : Taking z-transform of the given difference equation,

$$Y(z) = a z^{-1} Y(z) + X(z)$$

$$\therefore (1 - a z^{-1}) Y(z) = X(z)$$

$$\therefore \frac{Y(z)}{X(z)} = \frac{1}{1 - a z^{-1}}$$

Since $H(z) = \frac{Y(z)}{X(z)}$

$$H(z) = \frac{1}{1 - az^{-1}}$$

This is the system function. Converting powers of z positive in above equation,

$$H(z) = \frac{z}{z - a} = \frac{z - z_1}{z - p_1}$$

This system has one zero at $z_1 = 0$ and one pole at $p_1 = a$. The pole zero diagram is shown Fig. 3.6.3.

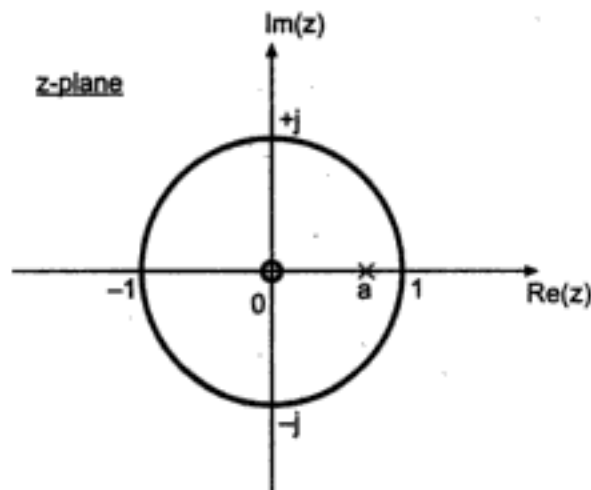


Fig. 3.6.3 Pole zero plot of the function of example 3.6.4

Ex.3.6.5 For the following systems

(i) $y(n) = 0.5y(n-1) + x(n) + x(n-1)$

(ii) $y(n) = x(n) + 3x(n-1) + 3x(n-2) + x(n-3)$

Plot the pole-zero diagram and assuming $y(n) = 0$ for $n < 0$, find values of $y(n)$ for $n = 0, 1, 2, 3, 4, 5$ for $x(n) = \delta(n)$. [Dec-99]

Sol. : (i) The given difference equation is,

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

Taking z-transform of above equation,

$$Y(z) = \frac{1}{2}z^{-1}Y(z) + X(z) + z^{-1}X(z)$$

$$\therefore \left(1 - \frac{1}{2}z^{-1}\right)Y(z) = (1 + z^{-1})X(z)$$

$$\therefore \frac{Y(z)}{X(z)} = \frac{1 + z^{-1}}{1 - \frac{1}{2}z^{-1}}$$

We know that $\frac{Y(z)}{X(z)} = H(z)$. i.e.,

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 + z^{-1}}{1 - \frac{1}{2}z^{-1}} \quad \dots (3.6.10)$$

$$= \frac{z+1}{z-\frac{1}{2}} = \frac{z-z_1}{z-p_1}$$

Thus $H(z)$ has one zero at $z_1 = -1$ and one pole at $p_1 = \frac{1}{2}$.

Fig. 3.6.4 shows the pole zero diagram.

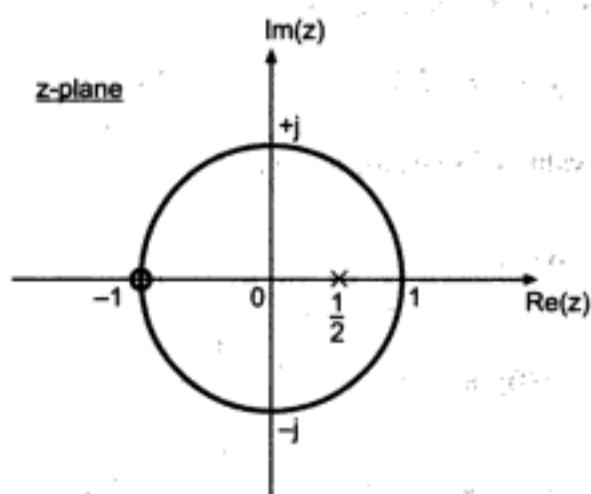


Fig. 3.6.4 Pole zero diagram of system of example 3.6.5

From equation 3.6.10 we have,

$$Y(z) = \frac{1+z^{-1}}{1-\frac{1}{2}z^{-1}} X(z) \quad \dots (3.6.11)$$

We know that input $x(n) = \delta(n)$

Hence $X(z) = Z\{\delta(n)\} = 1$

Putting this value of $X(z)$ in equation 3.6.11 we get,

$$Y(z) = \frac{1+z^{-1}}{1-\frac{1}{2}z^{-1}} = \frac{z+1}{z-\frac{1}{2}}$$

$$\therefore \frac{Y(z)}{z} = \frac{z+1}{z\left(z-\frac{1}{2}\right)} = \frac{A_1}{z} + \frac{A_2}{z-\frac{1}{2}}$$

$$\therefore A_1 = z \cdot \frac{Y(z)}{z} \Big|_{z=0} = \frac{z+1}{z-\frac{1}{2}} \Big|_{z=0} = -2$$

and

$$A_2 = \left(z-\frac{1}{2}\right) \cdot \frac{Y(z)}{z} \Big|_{z=\frac{1}{2}} = \frac{z+1}{z} \Big|_{z=\frac{1}{2}} = 3$$

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

$$\begin{aligned}
 &= \left[\sum_{k=-\infty}^{\infty} h(k) e^{-j\omega k} \right] e^{j\omega n} \\
 &= H(\omega) e^{j\omega n} \quad \dots (5.2.23)
 \end{aligned}$$

Here,
$$H(\omega) = \sum_{k=-\infty}^{\infty} h(k) e^{-j\omega k} \quad \dots (5.2.24)$$

Thus $H(\omega)$ is the Fourier transform of $h(k)$. And $h(k)$ is the unit sample response. $H(\omega)$ is called Transfer function of the system. $H(\omega)$ is complex valued function of ω in the range $-\pi \leq \omega \leq \pi$. The transfer function $H(\omega)$ can be expressed in polar form as,

$$H(\omega) = |H(\omega)| e^{j\angle H(\omega)} \quad \dots (5.2.25)$$

Here $|H(\omega)|$ is magnitude of $H(\omega)$

and $\angle H(\omega)$ is angle of $H(\omega)$.

By Euler's identity we can write $e^{\theta} = \cos \theta + j \sin \theta$. Hence equation 5.2.24 can be written as,

$$\begin{aligned}
 H(\omega) &= \sum_{k=-\infty}^{\infty} h(k) [\cos(\omega k) - j \sin(\omega k)] \\
 &= \sum_{k=-\infty}^{\infty} h(k) \cos(\omega k) - j \sum_{k=-\infty}^{\infty} h(k) \sin(\omega k) \quad \dots (5.2.26)
 \end{aligned}$$

Here $H_R(\omega) = \text{real part of } H(\omega) = \sum_{k=-\infty}^{\infty} h(k) \cos(\omega k) \quad \dots (5.2.27)$

and $H_I(\omega) = \text{Imaginary part of } H(\omega) = - \sum_{k=-\infty}^{\infty} h(k) \sin(\omega k) \quad \dots (5.2.28)$

and $|H(\omega)| = \sqrt{H_R^2(\omega) + H_I^2(\omega)} \quad \dots (5.2.29)$

and $\angle H(\omega) = \tan^{-1} \frac{H_I(\omega)}{H_R(\omega)} \quad \dots (5.2.30)$

Ex.5.2.6 The difference equation for the low pass filter is given as,

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

Obtain the magnitude/phase transfer function plots for this filter.

Sol. : The given difference equation is,

$$y(n) = \frac{1}{2} x(n) + \frac{1}{2} x(n-1) \quad \dots (5.2.31)$$

The linear convolution of unit sample response $h(n)$ and input $x(n)$ gives output $y(n)$. i.e.,

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

Hidden page

$$= \cos \frac{\omega}{2} \quad \dots (5.2.33)$$

$$\text{Phase of } H(\omega) = \angle H(\omega) = \tan^{-1} \frac{H_I(\omega)}{H_R(\omega)}$$

$$= \tan^{-1} \left[\frac{-\sin \frac{\omega}{2} \cos \frac{\omega}{2}}{\cos^2 \frac{\omega}{2}} \right]$$

$$= \tan^{-1} \left(-\tan \frac{\omega}{2} \right)$$

$$= -\frac{\omega}{2} \quad \dots (5.2.34)$$

Table 5.2.3 shows the calculations of magnitude and phase of $H(\omega)$ for few values of ω

Table 5.2.3 Calculation of $|H(\omega)|$ and $\angle H(\omega)$

ω	Magnitude $ H(\omega) = \cos \frac{\omega}{2}$	Phase $\angle H(\omega) = -\frac{\omega}{2}$
$-\pi$	0	$\frac{\pi}{2}$
$-\frac{2\pi}{3}$	0.5	$\frac{\pi}{3}$
$-\frac{\pi}{2}$	$\frac{1}{\sqrt{2}}$	$\frac{\pi}{4}$
$-\frac{\pi}{3}$	$\frac{\sqrt{3}}{2}$	$\frac{\pi}{6}$
0	1	0
$\frac{\pi}{3}$	$\frac{\sqrt{3}}{2}$	$-\frac{\pi}{6}$
$\frac{\pi}{2}$	$\frac{1}{\sqrt{2}}$	$-\frac{\pi}{4}$
$\frac{2\pi}{3}$	0.5	$-\frac{\pi}{3}$
π	0	$-\frac{\pi}{2}$

Fig. 5.2.1 (a) shows the magnitude and Fig. 5.2.1 (b) shows the phase plot of transfer function based on calculations in above table.

Please refer Fig. 5.2.1 on next page.

Hidden page

Hidden page

In the z -plane, $z = r e^{j\omega}$ represents a phasor. It has magnitude $r = |z|$ from origin and angle ' ω ' with respect to $Re[z]$ axis. This is illustrated in Fig. 5.2.2.

As shown in Fig. 5.2.2, the angle ' ω ' is basically frequency ' ω ' in fourier transform. Thus the unit circle becomes frequency axis. The frequency $\omega = 0$ i.e. DC is along the positive $Re[z]$ axis. The frequency of $\omega = \frac{\pi}{2}$ is along the positive $Im[z]$ axis. And the frequency $\omega = \pi$ is along the negative $Re[z]$ axis. Observe that the point $\omega = 0$ and $\omega = 2\pi$ is same on unit circle. Similarly the point $\omega = \pi$ and $\omega = -\pi$ is same i.e. $(-1, 0)$ on the unit circle. We have seen that maximum discrete time frequency that can be represented is π . Hence upper part of the unit circle represents actually used frequency scale. This z -domain frequency scale will be useful in the design of filters.

Ex.5.2.7 The difference equation of the system is given as,

$$y(n) = \frac{1}{2}x(n) + \frac{1}{2}x(n-1) \quad \dots (5.2.43)$$

- Find out (i) System function (ii) Unit sample response (iii) Pole zero plot
(iv) Transfer function and its magnitude phase plot
(v) Magnitude/phase response using geometric interpretation.

Sol. : (i) To find system function :

Taking z -transform of given difference equation,

$$Z\{y(n)\} = Z\left\{\frac{1}{2}x(n) + \frac{1}{2}x(n-1)\right\}$$

Applying the linearity and time shift properties,

$$\begin{aligned} Y(z) &= \frac{1}{2}X(z) + \frac{1}{2}z^{-1}X(z) \\ &= \left(\frac{1}{2} + \frac{1}{2}z^{-1}\right)X(z) \end{aligned}$$

$$\therefore \frac{Y(z)}{X(z)} = \frac{1}{2} + \frac{1}{2}z^{-1}$$

Hence system function $H(z) = \frac{Y(z)}{X(z)}$ becomes,

$$H(z) = \frac{1}{2} + \frac{1}{2}z^{-1} \quad \dots (5.2.44)$$

(ii) To find unit sample response :

A unit sample response $h(n)$ is obtained by taking inverse z -transform of $H(z)$. i.e.,

$$h(n) = IZT\{H(z)\} = IZT\left\{\frac{1}{2} + \frac{1}{2}z^{-1}\right\}$$

Applying the linearity and time shifting properties and from z -transform pair $\delta(n) \leftrightarrow 1$, we can write,

$$h(n) = \frac{1}{2}\delta(n) + \frac{1}{2}\delta(n-1) \quad \dots (5.2.45)$$

(iii) To determine pole zero plot :

Consider the system function $H(z)$,

$$H(z) = \frac{1}{2} + \frac{1}{2}z^{-1} \quad \dots (5.2.46)$$

Let us convert the powers of 'z' to positive by rearranging the equation as follows ;

$$H(z) = \frac{1}{2} \cdot \frac{z+1}{z}$$

Hence zero is at $z = -1$ i.e. $-1 + j0$

and pole is at $z = 0$ i.e. origin

Fig. 5.2.3 shows the pole-zero plot.

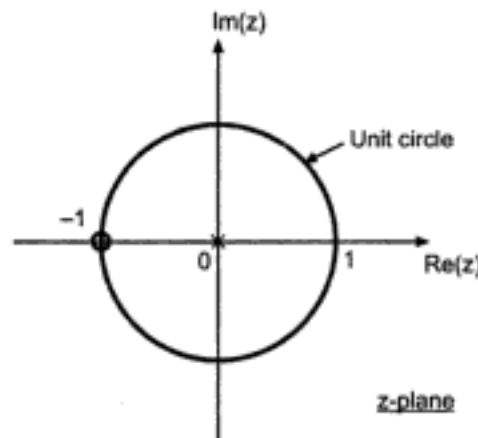


Fig. 5.2.3 Pole-zero plot of the system function of equation 5.2.46

(iv) To obtain transfer function $H(\omega)$ and its magnitude/phase plot :

The transfer function $H(\omega)$ can be obtained from system function $H(z)$ by putting $z = e^{j\omega}$. Hence equation 5.2.46 becomes,

$$\begin{aligned} H(\omega) &= H(z)|_{z=e^{j\omega}} = \frac{1}{2} + \frac{1}{2}e^{-j\omega} \\ &= \frac{1}{2}(1 + e^{-j\omega}) \quad \dots (5.2.47) \\ &= \frac{1}{2} \cdot e^{-j\omega/2} (e^{j\omega/2} + e^{-j\omega/2}) \end{aligned}$$

By rearranging the equation

By euler's identity we know that $\frac{e^{j\theta} + e^{-j\theta}}{2} = \cos \theta$. Hence above equation can be written as,

$$H(\omega) = \cos\left(\frac{\omega}{2}\right) \cdot e^{-j\omega/2} \quad \dots (5.2.48)$$

This is the required transfer function. This transfer function can be expressed in phasor form as,

$$H(\omega) = |H(\omega)| e^{j\angle H(\omega)} \quad \dots (5.2.49)$$

On comparing above equation with equation 5.2.48 we get,

$$\text{Magnitude of } H(\omega) \Rightarrow |H(\omega)| = \cos\left(\frac{\omega}{2}\right) \quad \dots (5.2.50)$$

$$\text{Phase of } H(\omega) \Rightarrow \angle H(\omega) = -\frac{\omega}{2} \quad \dots (5.2.51)$$

' ω ' is the continuous function and varies from $-\pi$ to π . Table 5.2.4 shows the values of $|H(\omega)|$ and $\angle H(\omega)$ for few values of ω .

Table 5.2.4 : Calculation of magnitude and phase response

ω	$ H(\omega) = \cos\left(\frac{\omega}{2}\right)$	$\angle H(\omega) = -\frac{\omega}{2}$
$-\pi$	0	$\frac{\pi}{2}$
$-\frac{3\pi}{4}$	0.383	$\frac{3\pi}{8}$
$-\frac{\pi}{2}$	0.707	$\frac{\pi}{4}$
$-\frac{\pi}{4}$	0.924	$\frac{\pi}{8}$
0	1	0
$\frac{\pi}{4}$	0.924	$-\frac{\pi}{8}$
ω	$ H(\omega) = \cos\left(\frac{\omega}{2}\right)$	$\angle H(\omega) = -\frac{\omega}{2}$
$\frac{\pi}{2}$	0.707	$-\frac{\pi}{4}$
$\frac{3\pi}{4}$	0.383	$-\frac{3\pi}{8}$
π	0	$-\frac{\pi}{2}$

Fig. 5.2.4 shows the magnitude and phase response of the transfer function. (See Fig. on next page)

Note : We know that the range of ω is, $-\pi \leq \omega \leq \pi$. Hence useful range in above magnitude and phase plot is from $-\pi$ to π . Reader can just try out by plotting $|H(\omega)|$ and $\angle H(\omega)$ outside this range. If we plot $|H(\omega)|$ and $\angle H(\omega)$ from π to 3π , the similar response repeats.

Here observe that the magnitude/phase plot given in Fig. 5.2.4 is same as that of Fig. 5.2.1, since difference equation of this example is same as that of Ex. 5.2.6.

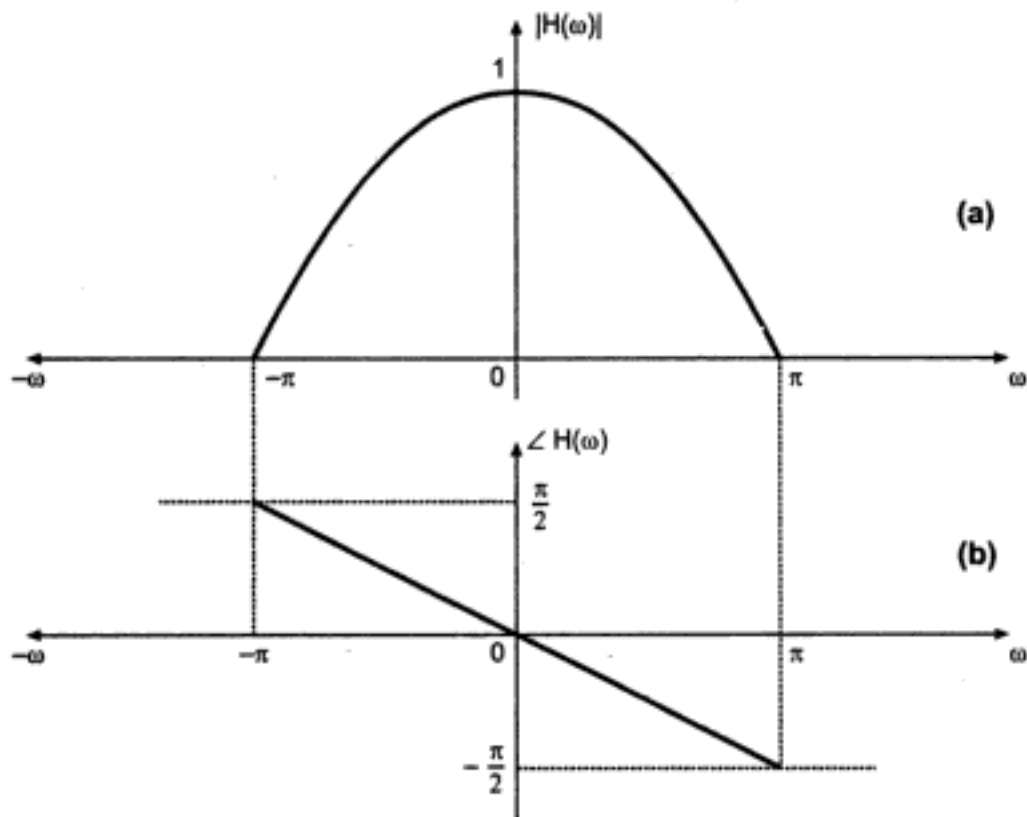


Fig. 5.2.4 Plots of magnitude and phase of transfer function
 (a) : Plot of magnitude transfer function $|H(\omega)|$ versus ω
 (b) : Plot of phase of the transfer function $\angle H(\omega)$ versus ω

Ex.5.2.8 A discrete time causal system has a transfer function $H(z)$ as,

$$H(z) = \frac{1 - z^{-1}}{1 - 0.2z^{-1} - 0.15z^{-2}}$$

(i) Determine the difference equation of the system

(ii) Show pole-zero diagram and hence find magnitude at $\omega = 0$ and $\omega = \pi$

(iii) Find impulse response of the system.

[Dec-99]

Sol. : (i) We know that $H(z) = \frac{Y(z)}{X(z)}$. Hence we have,

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 - z^{-1}}{1 - 0.2z^{-1} - 0.15z^{-2}}$$

Cross multiplying we get,

$$Y(z) [1 - 0.2z^{-1} - 0.15z^{-2}] = X(z) [1 - z^{-1}]$$

$$\therefore Y(z) - 0.2z^{-1}Y(z) - 0.15z^{-2}Y(z) = X(z) - z^{-1}X(z)$$

Taking inverse z-transform of above equation,

$$y(n) - 0.2y(n-1) - 0.15y(n-2) = x(n) - x(n-1)$$

$$\therefore y(n) = 0.2y(n-1) + 0.15y(n-2) + x(n) - x(n-1)$$

Hidden page

Hidden page

signal processor is impossible. Hence it is necessary to compute $X(\omega)$ only at discrete values of ω . When Fourier transform is calculated at only discrete points, it is called Discrete Fourier Transform (DFT). The DFT is denoted by $X(k)$ and it is given as,

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}, \quad k = 0, 1, 2, \dots, N-1 \quad \dots (5.3.2)$$

Here $X(k)$ is the DFT and it is computed at $k = 0, 1, 2, \dots, N-1$ i.e. 'N' discrete points. Thus DFT $X(k)$ is the sequence of 'N' samples. The sequence $x(n)$ can be obtained back from $X(k)$ by taking Inverse Discrete Fourier Transform, i.e. IDFT. It is given as,

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi kn/N}, \quad n = 0, 1, 2, \dots, N-1 \quad \dots (5.3.3)$$

Here $x(n)$ is the sequence of 'N' samples. Thus $X(k)$ and $x(n)$ both contains 'N' number of samples.

Let us define,
$$W_N = e^{-j2\pi/N} \quad \dots (5.3.4)$$

This is called twiddle factor. Hence DFT and IDFT equation can be written as,

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad k = 0, 1, \dots, N-1 \quad \dots (5.3.5)$$

and
$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn}, \quad n = 0, 1, \dots, N-1 \quad \dots (5.3.6)$$

The above DFT and IDFT equation are obtained by putting $e^{-j2\pi/N} = W_N$ in equation 5.3.2 and equation 5.3.3. Let us represent sequence $x(n)$ as vector x_N of N samples. i.e.,

$$x_N = \begin{matrix} n=0 \\ n=1 \\ \vdots \\ \vdots \\ n=N-1 \end{matrix} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ \vdots \\ x(N-1) \end{bmatrix}_{N \times 1} \quad \dots (5.3.7)$$

and $X(k)$ can be represented as a vector X_N of 'N' samples. i.e.,

$$X_N = \begin{matrix} k=0 \\ k=1 \\ \vdots \\ \vdots \\ k=N-1 \end{matrix} \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ \vdots \\ X(N-1) \end{bmatrix}_{N \times 1} \quad \dots (5.3.8)$$

The values of W_N^{kn} can be represented as a matrix $[W_N]$ of size $N \times N$ as follows :

$$[W_N] = \begin{matrix} & n=0 & n=1 & n=2 & \dots & n=N-1 \\ \begin{matrix} k=0 \\ k=1 \\ k=2 \\ \vdots \\ k=N-1 \end{matrix} & \begin{bmatrix} W_N^0 & W_N^0 & W_N^0 & \dots & W_N^0 \\ W_N^0 & W_N^1 & W_N^2 & \dots & W_N^{N-1} \\ W_N^0 & W_N^2 & W_N^4 & \dots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ W_N^0 & W_N^{N-1} & W_N^{2(N-1)} & \dots & W_N^{(N-1)(N-1)} \end{bmatrix} & \end{matrix}_{N \times N} \quad \dots (5.3.9)$$

Here the individual elements are written as W_N^{kn} with 'k' rows and 'n' columns. Then N-point DFT of equation 5.3.5 can be represented as,

$$X_N = [W_N] x_N \quad \dots (5.3.10)$$

Similarly IDFT of equation 5.3.6 can be expressed in matrix form as,

$$x_N = \frac{1}{N} [W_N^*] X_N \quad \dots (5.3.11)$$

Here $W_N^{kn} = [W_N]$, hence $[W_N^*] = W_N^{-kn}$ as written above.

Periodicity property of W_N :

Now let us see the values of W_N for $N=8$ we know that W_N is given as,

$$W_N = e^{-j\frac{2\pi}{N}}$$

With $N=8$ above equation becomes,

$$W_8 = e^{-j\frac{2\pi}{8}} = e^{-j\frac{\pi}{4}} \quad \dots (5.3.12)$$

Table 5.3.1 shows values of $W_8^0, W_8^1, W_8^2, \dots, W_8^{15}$

Table 5.3.1 Values of W_N for $N=8$

kn	$W_8^{kn} = e^{-j\frac{\pi}{4} \times kn}$	Comments
0	$W_8^0 = e^0$	Phasor of magnitude 1 and angle 0
1	$W_8^1 = e^{-j\frac{\pi}{4} \times 1} = e^{-j\frac{\pi}{4}}$	Phasor of magnitude 1 and angle $-\frac{\pi}{4}$
2	$W_8^2 = e^{-j\frac{\pi}{4} \times 2} = e^{-j\frac{\pi}{2}}$	Magnitude 1, Angle $-\frac{\pi}{2}$
3	$W_8^3 = e^{-j\frac{\pi}{4} \times 3} = e^{-j\frac{3\pi}{4}}$	Magnitude 1, Angle $-\frac{3\pi}{4}$
4	$W_8^4 = e^{-j\frac{\pi}{4} \times 4} = e^{-j\pi}$	Magnitude 1, Angle $-\pi$
5	$W_8^5 = e^{-j\frac{\pi}{4} \times 5} = e^{-j\frac{5\pi}{4}}$	Magnitude 1, Angle $-\frac{5\pi}{4}$
6	$W_8^6 = e^{-j\frac{\pi}{4} \times 6} = e^{-j\frac{3\pi}{2}}$	Magnitude 1, Angle $-\frac{3\pi}{2}$
7	$W_8^7 = e^{-j\frac{\pi}{4} \times 7} = e^{-j\frac{7\pi}{4}}$	Magnitude 1, Angle $-\frac{7\pi}{4}$
8	$W_8^8 = e^{-j\frac{\pi}{4} \times 8} = e^{-j2\pi}$	Magnitude 1, angle -2π Angle of -2π is same as '0' $\therefore W_8^8 = W_8^0$

kn	$W_8^{kn} = e^{-j\frac{\pi}{4} \times kn}$	Comments
9	$W_8^9 = e^{-j\frac{\pi}{4} \times 9} = e^{-j\left(2\pi + \frac{\pi}{4}\right)}$	Magnitude 1, angle $-\left(2\pi + \frac{\pi}{4}\right)$ This angle is same as $-\frac{\pi}{4}$ $\therefore W_8^9 = W_8^1$
10	$W_8^{10} = e^{-j\frac{\pi}{4} \times 10} = e^{-j\left(2\pi + \frac{\pi}{2}\right)}$	Magnitude 1, angle $-\left(2\pi + \frac{\pi}{2}\right)$ This angle is same as $-\frac{\pi}{2}$ $\therefore W_8^{10} = W_8^2$
Similarly following values can be verified		
11	$W_8^{11} = e^{-j\frac{\pi}{4} \times 11} = e^{-j\left(2\pi + \frac{3\pi}{4}\right)}$	$W_8^{11} = W_8^3$
12	$W_8^{12} = e^{-j\frac{\pi}{4} \times 12} = e^{-j(2\pi + \pi)}$	$W_8^{12} = W_8^4$
13	$W_8^{13} = e^{-j\frac{\pi}{4} \times 13} = e^{-j\left(2\pi + \frac{5\pi}{4}\right)}$	$W_8^{13} = W_8^5$
⋮	⋮	⋮
⋮	⋮	⋮
⋮	⋮	For all further values the above logic is applicable

In the above table observe the value of W_8^1 , i.e.,

$$\begin{aligned} W_8^1 &= e^{-j\frac{\pi}{4} \times 1} = e^{-j\frac{\pi}{4}} \\ &= \text{Magnitude } e^{-j} \text{ Angle} \end{aligned}$$

Thus magnitude = 1 and angle = $-\frac{\pi}{4}$. In the above table all phasors W_8^{kn} have magnitude

'1'. Fig. 5.3.1 shows these phasors in complex plane against unit circle.

Please refer Fig. 5.3.1 on next page.

The values of these phasors are also shown in above figure observe that,

$$\begin{aligned} W_8^1 &= W_8^9 = \dots = \cos \frac{\pi}{4} - j \sin \frac{\pi}{4} \\ &= \frac{1}{\sqrt{2}} - j \frac{1}{\sqrt{2}} \end{aligned}$$

Similarly other values are obtained.

Hidden page

The values given above can be verified easily. For example consider W_4^3 .

$$W_4^3 = e^{-j\frac{2\pi}{4} \times 3} = e^{-j\frac{3\pi}{2}}$$

This has magnitude '1' and angle $-\frac{3\pi}{2}$ and,

$$W_4^3 = e^{-j\frac{3\pi}{2}} = \cos \frac{3\pi}{2} - j \sin \frac{3\pi}{2} = j$$

We have the matrix $[W_4]$ of 4×4 size. Its individual elements will be as shown below :

$$[W_4] = \begin{matrix} & n=0 & n=1 & n=2 & n=3 \\ \begin{matrix} k=0 \\ k=1 \\ k=2 \\ k=3 \end{matrix} & \begin{bmatrix} W_4^0 & W_4^0 & W_4^0 & W_4^0 \\ W_4^0 & W_4^1 & W_4^2 & W_4^3 \\ W_4^0 & W_4^2 & W_4^4 & W_4^6 \\ W_4^0 & W_4^3 & W_4^6 & W_4^9 \end{bmatrix} \end{matrix}$$

In the above matrix individual elements are W_4^{kn} with $k=0$ to 3 rows and $n=0$ to 3 columns. From Fig. 5.3.2, the values of various elements in above matrix are,

$$[W_4] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

Also we have

$$x_4 = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}$$

From equation 5.3.10 we can obtain 4 point DFT as

$$X_N = [W_N] x_N$$

With $N=4$, $X_4 = [W_4] x_4$

Putting values of W_4 and x_4 ,

$$\begin{aligned} X_4 &= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix} \\ &= \begin{bmatrix} 0+1+2+3 \\ 0-j-2+3j \\ 0-1+2-3 \\ 0+j-2-3j \end{bmatrix} = \begin{bmatrix} 6 \\ -2+2j \\ -2 \\ -2-2j \end{bmatrix} \end{aligned}$$

Thus we obtained 4 point DFT as,

$$X_4 = \begin{bmatrix} X(0) = 6 \\ X(1) = -2 + 2j \\ X(2) = -2 \\ X(3) = -2 - 2j \end{bmatrix}$$

This is the required DFT.

Ex. 5.3.2 Calculate 8-point DFT of the following signal

$$x(n) = \{1, 1, 1, 1\}$$

Assume imaginary part is zero. Also calculate magnitude and phase of $X(k)$

[May-98]

Sol. : The given sequence is of length '4'. Since 8-point DFT is asked, we should make length of the sequence to be '8'. This is done by appending four zeros at the end of $x(n)$. i.e.,

$$x(n) = \{1, 1, 1, 1, 0, 0, 0, 0\}$$

Thus appending zeros at the end of sequence does not change its meaning. This is also called as *zero padding*.

DFT is given by equation 5.3.5 as,

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

For 8 point DFT, above equation becomes,

$$X(k) = \sum_{n=0}^7 x(n) W_8^{kn}, \quad k = 0, 1, \dots, 7$$

This equation is formulated in the matrix form by equation 5.3.10 as,

$$X_N = [W_N] x_N$$

with $N = 8$, $X_8 = [W_8] x_8$

These matrices can be expanded as,

$$X_8 = \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} \quad \text{and} \quad x_8 = \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

And the 8×8 matrix $[W_8]$ can be written from equation 5.3.9 as follows,

$$[W_8] = \begin{bmatrix} W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 \\ W_8^0 & W_8^1 & W_8^2 & W_8^3 & W_8^4 & W_8^5 & W_8^6 & W_8^7 \\ W_8^0 & W_8^2 & W_8^4 & W_8^6 & W_8^8 & W_8^{10} & W_8^{12} & W_8^{14} \\ W_8^0 & W_8^3 & W_8^6 & W_8^9 & W_8^{12} & W_8^{15} & W_8^{18} & W_8^{21} \\ W_8^0 & W_8^4 & W_8^8 & W_8^{12} & W_8^{16} & W_8^{20} & W_8^{24} & W_8^{28} \\ W_8^0 & W_8^5 & W_8^{10} & W_8^{15} & W_8^{20} & W_8^{25} & W_8^{30} & W_8^{35} \\ W_8^0 & W_8^6 & W_8^{12} & W_8^{18} & W_8^{24} & W_8^{30} & W_8^{36} & W_8^{42} \\ W_8^0 & W_8^7 & W_8^{14} & W_8^{21} & W_8^{28} & W_8^{35} & W_8^{42} & W_8^{49} \end{bmatrix}$$

In Table 5.3.1 and Fig. 5.3.1 we have obtained the values of W_N^{kn} for $N=8$. From Table 5.3.1 and Fig. 5.3.1 we obtain following.

$$W_8^0 = W_8^8 = W_8^{16} = W_8^{24} = W_8^{32} = W_8^{40} = \dots = 1$$

$$W_8^1 = W_8^9 = W_8^{17} = W_8^{25} = W_8^{33} = W_8^{41} = W_8^{49} = \dots = \frac{1}{\sqrt{2}} - j \frac{1}{\sqrt{2}}$$

$$W_8^2 = W_8^{10} = W_8^{18} = W_8^{26} = W_8^{34} = W_8^{42} = \dots = -j$$

$$W_8^3 = W_8^{11} = W_8^{19} = W_8^{27} = W_8^{35} = W_8^{43} = \dots = -\frac{1}{\sqrt{2}} - j \frac{1}{\sqrt{2}}$$

$$W_8^4 = W_8^{12} = W_8^{20} = W_8^{28} = W_8^{36} = W_8^{44} = \dots = -1$$

$$W_8^5 = W_8^{13} = W_8^{21} = W_8^{29} = W_8^{37} = W_8^{45} = \dots = -\frac{1}{\sqrt{2}} + j \frac{1}{\sqrt{2}}$$

$$W_8^6 = W_8^{14} = W_8^{22} = W_8^{30} = W_8^{38} = W_8^{46} = \dots = j$$

$$W_8^7 = W_8^{15} = W_8^{23} = W_8^{31} = W_8^{39} = W_8^{47} = \dots = \frac{1}{\sqrt{2}} + j \frac{1}{\sqrt{2}}$$

Putting different values in the equation for X_8 , we obtain,

$$X_8 = [W_8] \cdot x_8 \quad \text{i.e.,}$$

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \frac{1}{\sqrt{2}} - j \frac{1}{\sqrt{2}} & -j & -\frac{1}{\sqrt{2}} - j \frac{1}{\sqrt{2}} & -1 & -\frac{1}{\sqrt{2}} + j \frac{1}{\sqrt{2}} & j & \frac{1}{\sqrt{2}} + j \frac{1}{\sqrt{2}} \\ 1 & -j & -1 & j & 1 & -j & -1 & j \\ 1 & -\frac{1}{\sqrt{2}} - j \frac{1}{\sqrt{2}} & j & \frac{1}{\sqrt{2}} - j \frac{1}{\sqrt{2}} & -1 & \frac{1}{\sqrt{2}} + j \frac{1}{\sqrt{2}} & -j & -\frac{1}{\sqrt{2}} + j \frac{1}{\sqrt{2}} \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -\frac{1}{\sqrt{2}} + j \frac{1}{\sqrt{2}} & -j & \frac{1}{\sqrt{2}} + j \frac{1}{\sqrt{2}} & -1 & \frac{1}{\sqrt{2}} - j \frac{1}{\sqrt{2}} & j & -\frac{1}{\sqrt{2}} - j \frac{1}{\sqrt{2}} \\ 1 & j & -1 & -j & 1 & j & -1 & -j \\ 1 & \frac{1}{\sqrt{2}} + j \frac{1}{\sqrt{2}} & j & -\frac{1}{\sqrt{2}} + j \frac{1}{\sqrt{2}} & -1 & -\frac{1}{\sqrt{2}} - j \frac{1}{\sqrt{2}} & -j & \frac{1}{\sqrt{2}} - j \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Hidden page

Hidden page

$$\sum_{k=N_1}^{N_2} a^k = \frac{a^{N_1} - a^{N_2+1}}{1-a}$$

Hence equation 5.3.15 can be written as,

$$\begin{aligned} X(k) &= \frac{(a e^{-j2\pi k/N})^0 - (a e^{-j2\pi k/N})^N}{1 - a e^{-j2\pi k/N}} \\ &= \frac{1 - a^N e^{-j2\pi k}}{1 - a e^{-j2\pi k/N}} \end{aligned} \quad \dots (5.3.16)$$

Here $e^{-j2\pi k} = \cos 2\pi k - j \sin 2\pi k$
 $= 1 - j0 = 1$ always

Hence equation 5.3.16 can be written as,

$$X(k) = \frac{1 - a^N}{1 - a e^{-j2\pi k/N}} \quad \dots (5.3.17)$$

This is the DFT of exponential sequence a^n .

5.3.3 Properties of DFT

We have defined DFT with the help of some examples. A signal $x(n)$ and its DFT $X(k)$ is represented as a pair by shorthand notation as,

$$x(n) \xleftrightarrow[N]{\text{DFT}} X(k)$$

Here 'N' represents 'N' point DFT. Various properties of DFT are described in this subsection.

5.3.3.1 Periodicity

Let $x(n)$ and $X(k)$ be the DFT pair.

Then, if $x(n+N) = x(n)$ for all n, then
 $X(k+N) = X(k)$ for all k ... (5.3.18)

Proof :

By definition DFT is given as,

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \quad \text{by equation 5.3.5} \quad \dots (5.3.18 \text{ (a)})$$

Let us replace k by k + N, then above equation becomes,

$$\begin{aligned} X(k+N) &= \sum_{n=0}^{N-1} x(n) W_N^{(k+N)n} \\ &= \sum_{n=0}^{N-1} x(n) W_N^{kn} \cdot W_N^{Nn} \end{aligned} \quad \dots (5.3.18 \text{ (b)})$$

We know that, $W_N = e^{-j\frac{2\pi}{N}}$

$$\begin{aligned} \therefore W_N^{Nn} &= e^{-j\frac{2\pi}{N}Nn} = e^{-j2\pi n} \\ &= \cos(2\pi n) - j\sin 2\pi n \\ &= 1 - j0 \\ &= 1 \text{ always} \end{aligned}$$

This is because for all values of 'n', $\cos(2\pi n)=1$ and $\sin(2\pi n)=0$. Hence equation 5.3.18 (b) becomes,

$$\begin{aligned} X(k+N) &= \sum_{n=0}^{N-1} x(n) W_N^{kn} \\ &= X(k) \quad \text{by equation 5.3.18 (a)} \end{aligned}$$

Similarly by using IDFT formula it can be shown that $x(N+n) = x(n)$.

5.3.3.2 Linearity

The linearity property of DFT states that

$$\text{if } x_1(n) \xleftrightarrow[N]{\text{DFT}} X_1(k)$$

$$\text{and } x_2(n) \xleftrightarrow[N]{\text{DFT}} X_2(k) \quad \text{then,}$$

$$a_1 x_1(n) + a_2 x_2(n) \xleftrightarrow[N]{\text{DFT}} a_1 X_1(k) + a_2 X_2(k) \quad \dots (5.3.19)$$

Here a_1 and a_2 are constants.

Proof :

By definition of DFT, we can write,

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

Let $x(n) = a_1 x_1(n) + a_2 x_2(n)$, then above equation becomes,

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} [a_1 x_1(n) + a_2 x_2(n)] W_N^{kn} \\ &= \sum_{n=0}^{N-1} a_1 x_1(n) W_N^{kn} + \sum_{n=0}^{N-1} a_2 x_2(n) W_N^{kn} \\ &= a_1 \sum_{n=0}^{N-1} x_1(n) W_N^{kn} + a_2 \sum_{n=0}^{N-1} x_2(n) W_N^{kn} \\ &= a_1 X_1(k) + a_2 X_2(k) \end{aligned}$$

5.3.3.3 Circular Symmetries of a Sequence

Till now we have seen that $x(n)$ is the sequence containing 'N' samples. And $X(k)$ is the 'N' point DFT. When we take IDFT we get a periodic sequence $x_p(n)$ which is related to $x(n)$ as,

$$x_p(n) = \sum_{l=-\infty}^{\infty} x(n-lN) \quad \dots (5.3.20)$$

Thus $x_p(n)$ is nothing but periodic repetition of $x(n)$. And $x_p(n)$ contains such infinite periodic repetitions. On the basis of this conclusion we can say that if we take DFT of $x_p(n)$ we get the same DFT $X(k)$ which is obtained due to $x(n)$. i.e.,

$$x(n) \xleftrightarrow[N]{\text{DFT}} X(k) \quad \dots (5.3.21)$$

$$x_p(n) \xleftrightarrow[N]{\text{DFT}} X(k) \quad \dots (5.3.22)$$

And $x(n)$ and $x_p(n)$ are related by equation 5.3.20.

Or we can write,

$$x(n) = \begin{cases} x_p(n) & \text{for } 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases} \quad \dots (5.3.23)$$

Now let $x_p(n)$ be shifted by 'k' units to the right. Let this new sequence be $x'_p(n)$. i.e.

$$x'_p(n) = x_p(n-k) \quad \dots (5.3.24)$$

$$= \sum_{l=-\infty}^{\infty} x(n-k-lN) \quad \dots (5.3.24 (a))$$

Then the corresponding sequence $x'(n)$ can be obtained from equation 5.3.23 as,

$$x'(n) = \begin{cases} x'_p(n) & \text{for } 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases} \quad \dots (5.3.25)$$

This sequence $x'(n)$ is related to $x(n)$ by the circular shift. This concept is illustrated in Fig. 5.3.3. Fig. 5.3.3 (a) shows sequence $x(n)$ containing $N = 4$ samples. The periodic repetition of $x(n)$ is $x_p(n)$ and this sequence is shown in Fig. 5.3.3 (b). Fig. 5.3.3 (c) shows the sequence $x'_p(n)$ which is obtained by shifting $x_p(n)$ to the right by two samples. Fig. 5.3.3 (d) shows sequence $x'(n)$ as per equation 5.3.25. The sequence $x'(n)$ is basically one period of $x'_p(n)$ from $0 \leq n \leq 3$.

Please refer Fig. 5.3.3 on next page.

The sequence $x'(n)$ is related to $x(n)$ by a circular shift. It is represented as follows :

$$x'(n) = x(n-k, \text{ modulo } N) \quad \dots (5.3.26)$$

The shorthand notation for $(n-k, \text{ modulo } N)$ is $((n-k))_N$ i.e.,

$$x'(n) = x((n-k))_N \quad \dots (5.3.27)$$

Now let us evaluate samples of $x'(n)$ as per above equation with $k = 2$ and $N = 4$. Then above equation becomes,

$$\text{Hence, } \left. \begin{aligned} x'(n) &= x((n-2))_4 \\ x'(0) &= x((-2))_4 = x(2) \\ x'(1) &= x((-1))_4 = x(3) \\ x'(2) &= x((0))_4 = x(0) \\ x'(3) &= x((1))_4 = x(1) \end{aligned} \right\} \quad \dots (5.3.28)$$

The above relation indicate that $x'(n)$ is obtained by shifting $x(n)$ circularly by two samples. Such relationship can be better understood by plotting $x(n)$ anticlockwise along the circle as shown in Fig. 5.3.4 (a). Fig. 5.3.4 (b) indicates the sequence $x(n)$ delayed circularly by one sample, i.e. $x((n-1))_4$. This shift is anticlockwise, Fig. 5.3.4 (c) indicates

Hidden page

Hidden page

Hidden page

Hidden page

5.3.3.4 Symmetry Properties

Let the sequence $x(n)$ be complex valued and expressed as,

$$x(n) = X_R(n) + jX_I(n), \quad 0 \leq n \leq N-1 \quad \dots (5.3.32)$$

Let the DFT of $x(n)$ be complex valued and expressed as,

$$X(k) = X_R(k) + jX_I(k), \quad 0 \leq k \leq N-1 \quad \dots (5.3.33)$$

By definition of DFT,

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{kn} \\ &= \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \quad \text{since } W_N = e^{-j\frac{2\pi}{N}} \\ &= \sum_{n=0}^{N-1} [X_R(n) + jX_I(n)] e^{-j2\pi kn/N} \quad \text{by putting for } x(n) \\ &= \sum_{n=0}^{N-1} [X_R(n) + jX_I(n)] \left[\cos\left(\frac{2\pi kn}{N}\right) - j\sin\left(\frac{2\pi kn}{N}\right) \right] \\ &= \sum_{n=0}^{N-1} \left[X_R(n) \cos\left(\frac{2\pi kn}{N}\right) + X_I(n) \sin\left(\frac{2\pi kn}{N}\right) \right] \\ &\quad - j \sum_{n=0}^{N-1} \left[X_R(n) \sin\left(\frac{2\pi kn}{N}\right) - X_I(n) \cos\left(\frac{2\pi kn}{N}\right) \right] \end{aligned} \quad \dots (5.3.34)$$

We know that $X(k) = X_R(k) + jX_I(k)$. Hence comparing with above equation we obtain real and imaginary parts of $X(k)$ as follows.

$$X_R(k) = \sum_{n=0}^{N-1} \left[X_R(n) \cos\left(\frac{2\pi kn}{N}\right) + X_I(n) \sin\left(\frac{2\pi kn}{N}\right) \right] \quad \dots (5.3.35)$$

and

$$X_I(k) = - \sum_{n=0}^{N-1} \left[X_R(n) \sin\left(\frac{2\pi kn}{N}\right) - X_I(n) \cos\left(\frac{2\pi kn}{N}\right) \right] \quad \dots (5.3.36)$$

Similarly the real and imaginary parts of sequence $x(n)$ can be obtained in terms of its DFTs as follows (using IDFT formula):

$$x_R(n) = \frac{1}{N} \sum_{k=0}^{N-1} \left[X_R(k) \cos\left(\frac{2\pi kn}{N}\right) - X_I(k) \sin\left(\frac{2\pi kn}{N}\right) \right] \quad \dots (5.3.37)$$

and

$$x_I(n) = \frac{1}{N} \sum_{k=0}^{N-1} \left[X_R(k) \sin\left(\frac{2\pi kn}{N}\right) + X_I(k) \cos\left(\frac{2\pi kn}{N}\right) \right] \quad \dots (5.3.38)$$

(i) Symmetry property for real valued $x(n)$:

This property states that if $x(n)$ is real, then

$$X(N-k) = X^*(k) = X(-k) \quad \dots (5.3.39)$$

Hidden page

(iii) Real and odd sequence :

A real and odd sequence is expressed as,

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

Then DFT of such sequence becomes,

$$X(k) = -j \sum_{n=0}^{N-1} x(n) \sin\left(\frac{2\pi kn}{N}\right), \quad 0 \leq k \leq N - 1 \quad \dots (5.3.45)$$

For real and odd sequence $X_R(k)$ of equation 5.3.35 will be zero. This is because $x_I(n)$ is zero since sequence is real, and terms of $x_R(n) \cos\left(\frac{2\pi kn}{N}\right)$ in equation 5.3.35 cancel each other since the sequence $x_R(n)$ is odd and 'cosine' is even function. Equation 5.3.45 is obtained from equation 5.3.36 with $X(k) = X_I(k)$ and $x_I(n)$ is zero. Similarly IDFT is given from equation 5.3.37 as,

$$x(n) = j \frac{1}{N} \sum_{k=0}^{N-1} X(k) \sin\left(\frac{2\pi kn}{N}\right), \quad 0 \leq n \leq N - 1 \quad \dots (5.3.46)$$

(iv) Purely imaginary sequence :

The purely imaginary sequence is given as,

$$x(n) = j x_I(n)$$

Then DFT can be obtained from equation 5.3.35 and 5.3.36 as,

$$\left. \begin{aligned} X_R(k) &= \sum_{n=0}^{N-1} x_I(n) \sin\left(\frac{2\pi kn}{N}\right) \\ \text{and} \quad X_I(k) &= \sum_{n=0}^{N-1} x_I(n) \cos\left(\frac{2\pi kn}{N}\right) \end{aligned} \right\} \quad \dots (5.3.47)$$

Here if $x_I(n)$ is odd, then $X_I(k)$ becomes zero since 'cos' is even function. Similarly if $x_I(n)$ is even, then $X_R(k)$ becomes zero. Since 'sin' is odd function.

5.3.3.5 Circular Convolution

This property states that if

$$x_1(n) \xleftrightarrow[N]{\text{DFT}} X_1(k)$$

and

$$x_2(n) \xleftrightarrow[N]{\text{DFT}} X_2(k)$$

$$\text{Then } x_1(n) \circledR x_2(n) \xleftrightarrow[N]{\text{DFT}} X_1(k) X_2(k) \quad \dots (5.3.48)$$

Here $x_1(n) \circledR x_2(n)$ means circular convolution of $x_1(n)$ and $x_2(n)$. This property states that multiplication of two DFTs is equivalent to circular convolution of their sequences in time domain.

Proof :

By definition two DFTs, $X_1(k)$ and $X_2(k)$ are given as,

$$X_1(k) = \sum_{n=0}^{N-1} x_1(n) e^{-j 2\pi kn/N}, \quad k = 0, 1, \dots, N - 1 \quad \dots (5.3.49)$$

Hidden page

Here $e^{j2\pi k(m-n-l)} = 1$ always. Hence above equation becomes,

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

$$= 0 \quad \text{when } (m-n-l) \text{ is not multiple of } N. \quad \dots (5.3.56)$$

Thus we obtained from above equation and equation 5.3.55 as,

$$\sum_{k=0}^{N-1} e^{j2\pi k(m-n-l)/N} = \begin{cases} N & \text{when } (m-n-l) \text{ is multiple of } N \\ 0 & \text{otherwise} \end{cases} \quad \dots (5.3.57)$$

Putting this value in equation 5.3.53 we get,

$$x_3(m) = \frac{1}{N} \sum_{n=0}^{N-1} x_1(n) \sum_{l=0}^{N-1} x_2(l) \cdot N \quad \begin{matrix} \text{When } (m-n-l) \text{ is} \\ \text{multiple of } N \end{matrix}$$

$$= \sum_{n=0}^{N-1} x_1(n) \sum_{l=0}^{N-1} x_2(l) \quad \begin{matrix} \text{When } (m-n-l) \text{ is} \\ \text{multiple of } N \end{matrix}$$

$$\dots (5.3.58)$$

In the above equation $(m-n-l)$ multiple of N can be written as,

$$(m-n-l) = pN \quad \text{Here } p \text{ is some integer.}$$

Since integer multiple can be positive or negative both, we can write above condition for convenience as,

$$m-n-l = -pN$$

$$\therefore l = m-n+pN \quad \dots (5.3.59)$$

When ' l ' is given by above equation, the condition of equation 5.3.58 is satisfied. Hence putting for ' l ' from above equation in equation 5.3.58 we get,

$$x_3(m) = \sum_{n=0}^{N-1} x_1(n) x_2(m-n+pN) \quad \dots (5.3.60)$$

Here observe that the summation for $x_2(\)$ is dropped since it is redundant because of change of index. There is no ' l ' term in above equation.

In the above equation $x_2(m-n+pN)$ represents it is periodic sequence with period N . This periodic sequence is delayed by ' n ' samples. Such type of sequences we have treated in the beginning of section 5.3.3.3.

$x_2(m-n+pN)$ represents sequence $x_2(m)$ shifted circularly by ' n ' samples. This concept is explained in Fig. 5.3.3. Such sequence is represented by equation 5.3.26 as,

$$x_2(m-n+pN) = x_2(m-n, \text{ modulo } N) \quad \dots (5.3.61)$$

or by equation 5.3.27 above sequence can be represented as,

$$x_2(m-n+pN) = x_2((m-n))_N \quad \dots (5.3.62)$$

Putting this sequence in equation 5.3.60 we get,

$$x_3(m) = \sum_{n=0}^{N-1} x_1(n) x_2((m-n))_N, \quad m = 0, 1, \dots, N-1 \quad \dots (5.3.63)$$

Let us compare the above equation with linear convolution which is given as,

Hidden page

Hidden page

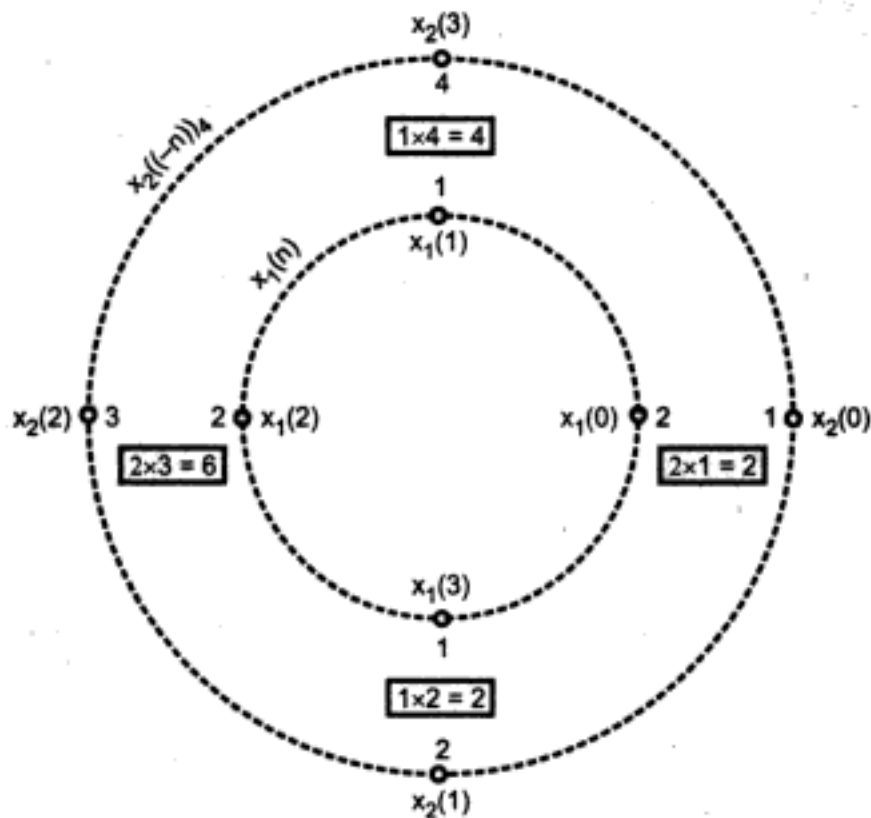


Fig. 5.3.8 $x_1(n)$ and $x_2((-n))_4$ plotted on two concentric circles. They are multiplied point by point

To obtain $x_3(1)$, put $m = 1$ in equation 5.3.65 :

Putting $m = 1$ in equation 5.3.65 we get,

$$x_3(1) = \sum_{n=0}^3 x_1(n)x_2((1-n))_4 \quad \dots (5.3.67)$$

Here the sequence $x_2((1-n))_4$ can be written as $x_2((-[n-1]))_4$. This means $x_2((-n))_4$ delayed by one sample. The delay of one sample is equivalent to shifting $x_2((-n))_4$ anticlockwise by one sample. [See equation 5.3.27 and its relevant description]. Thus $x_2((1-n))_4$ is obtained by shifting $x_2((-n))_4$ anticlockwise by one position or sample. This sequence is shown in Fig. 5.3.9 (b). Fig. 5.3.9 (a) shows $x_2((-n))_4$ for reference.

Please refer Fig. 5.3.9 on next page.

Now we have to obtain products $x_1(n)x_2((1-n))_4$ and their sum. This is obtained easily by plotting $x_1(n)$ and $x_2((1-n))_4$ on concentric circles as shown below in Fig. 5.3.10. The point by point products of the two sequences are also shown in Fig. 5.3.10. Hence $x_3(1)$ of equation 5.3.67 becomes,

$$x_3(1) = 4 + 1 + 8 + 3 = 16$$

Thus $x_3(1) = 16$

Please refer Fig. 5.3.10 on next page.

Hidden page

Hidden page

Thus $x_3(2)$ of equation 5.3.68 can be obtained from above figure as,

$$\begin{aligned} x_3(2) &= 6 + 2 + 2 + 4 \\ &= 14 \end{aligned}$$

Thus $x_3(2) = 14$

To find $x_3(3)$, put $m = 3$ in equation 5.3.65 :

Putting $m = 3$ in equation 5.3.65 we get,

$$x_3(3) = \sum_{n=0}^3 x_1(n)x_2((3-n))_4 \quad \dots (5.3.69)$$

Here $x_2((3-n))_4$ is obtained by shifting $x_2((2-n))_4$ anticlockwise by one sample position. This is equivalent to shifting $x_2((-n))_4$ by three sample positions. Now here let us use the sequences plotted on concentric circles in Fig. 5.3.12. Sequence $x_1(n)$ is plotted on inner circle, and it will remain as it is. This sequence is shown in Fig. 5.3.13 (inner circle) below. The $x_2((2-n))_4$ plotted on outer circle in Fig. 5.3.12 should be shifted anticlockwise by one sample position to get $x_2((3-n))_4$. This new shifted sequence is shown below in Fig. 5.3.13 on outer circle.

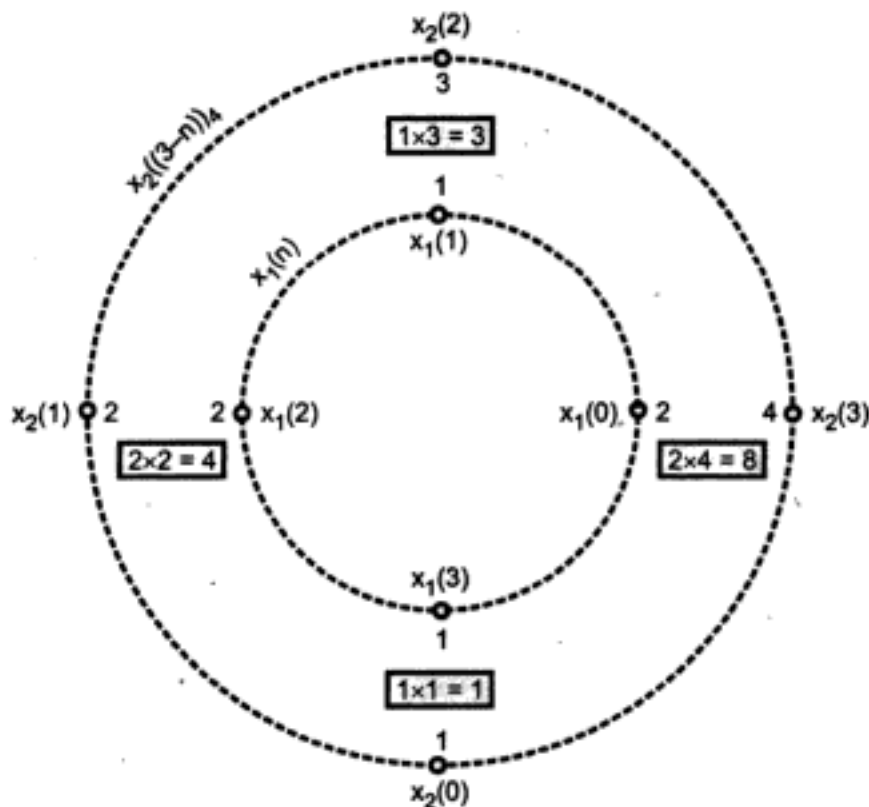


Fig. 5.3.13 To obtain $x_3(3)$

The point by point products $x_1(n)x_2((3-n))_4$ are shown in above figure. Then $x_3(3)$ of equation 5.3.69 can be obtained by adding all these product terms, i.e.,

$$\begin{aligned} x_3(3) &= 8 + 3 + 4 + 1 \\ &= 16 \end{aligned}$$

Thus $x_3(3) = 16$

As per equation 5.3.65, 'm' varies from 0 to 3. This means there will be four samples in sequence $x_3(\)$ i.e. $x_3(0)$, $x_3(1)$, $x_3(2)$ and $x_3(3)$ as we calculated. Thus sequence $x_3(\)$ is,

Hidden page

Hidden page

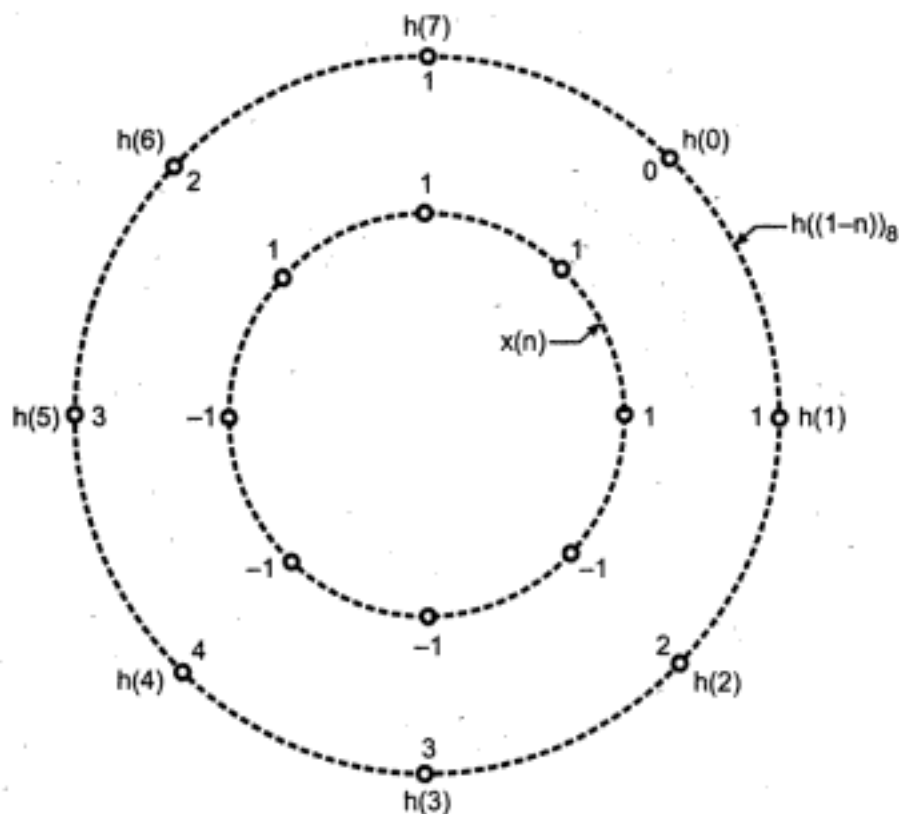


Fig. 5.3.16 Computation of $y(1) = \sum_{n=0}^7 x(n) h((1-n))_8$

Now let us shift $h(n)$ plotted on outer circle by one sample in anticlockwise direction. Then multiply two sequences point to point. If we add all these multiplications, we will get $y(2)$.

Please refer Fig. 5.3.17 on next page.

Thus from Fig. 5.3.17

$$\begin{aligned}
 y(2) &= (1 \times 2) + (1 \times 1) + (1 \times 0) + (1 \times 1) + (-1 \times 2) \\
 &\quad + (-1 \times 3) + (-1 \times 4) + (-1 \times 3) \\
 &= 2 + 1 + 0 + 1 - 2 - 3 - 4 - 3 \\
 &= -8
 \end{aligned}$$

$y(2) = -8$

Next, further shift $h(n)$ plotted on outer circle by one sample in anticlockwise direction. Then multiply the two sequences point to point.

Then adding all the multiplications gives $y(3)$. i.e. from Fig. 5.3.18 we can write,

$$\begin{aligned}
 y(3) &= (1 \times 3) + (1 \times 2) + (1 \times 1) + (1 \times 0) + (-1 \times 1) \\
 &\quad + (-1 \times 2) + (-1 \times 3) + (-1 \times 4) \\
 &= 3 + 2 + 1 + 0 - 1 - 2 - 3 - 4
 \end{aligned}$$

$= -4$

$y(3) = -4$

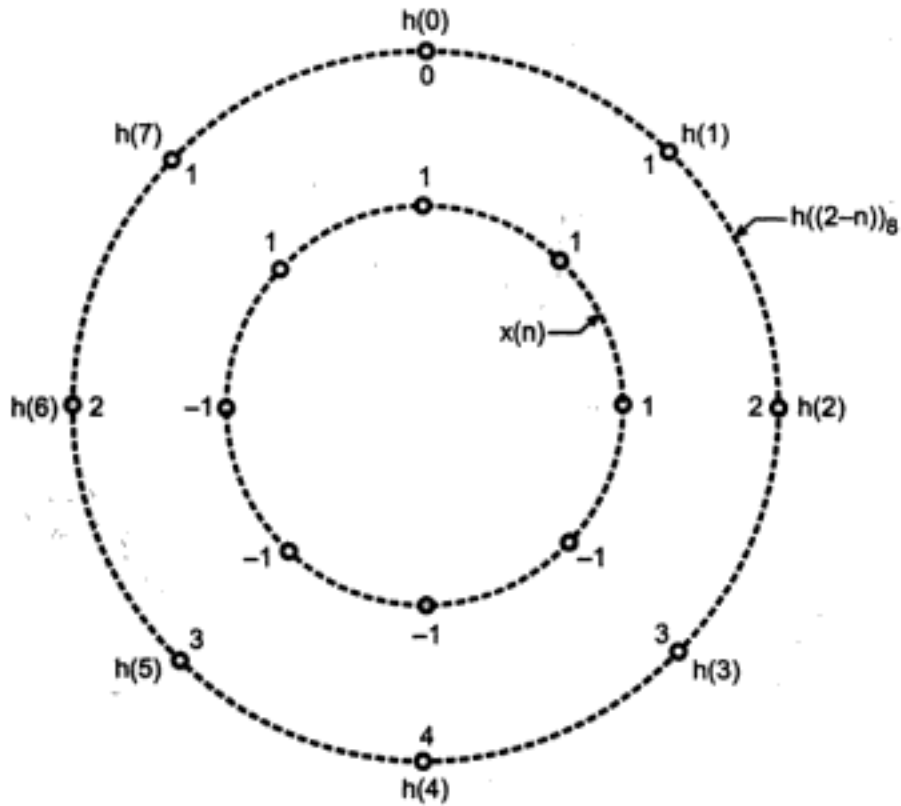


Fig. 5.3.17 Computation of $y(2) = \sum_{n=0}^7 x(n) h((2-n))_8$

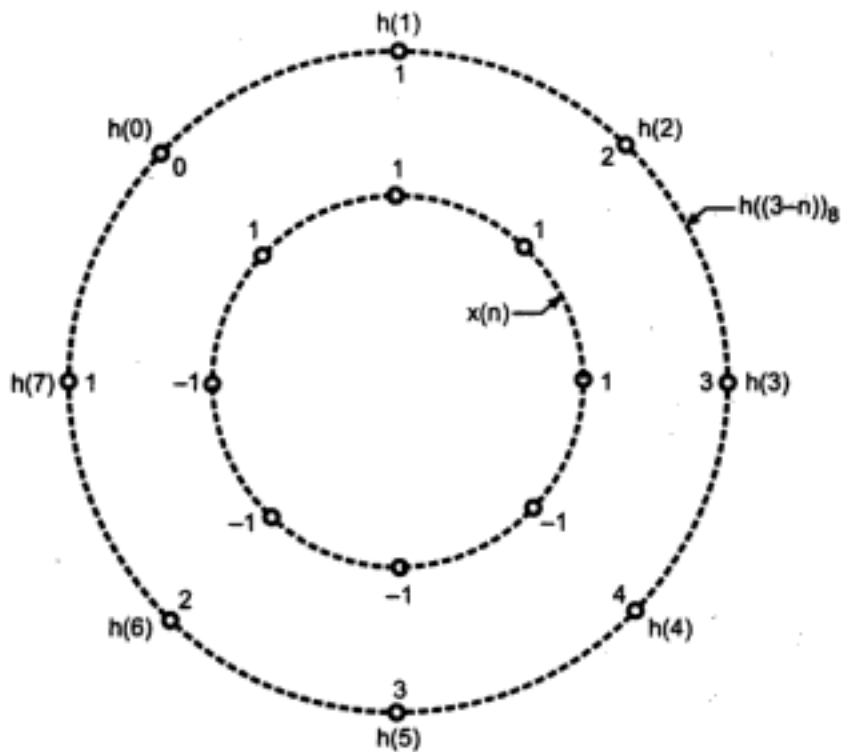


Fig. 5.3.18 Computation of $y(3) = \sum_{n=0}^7 x(n) h((3-n))_8$

Hidden page

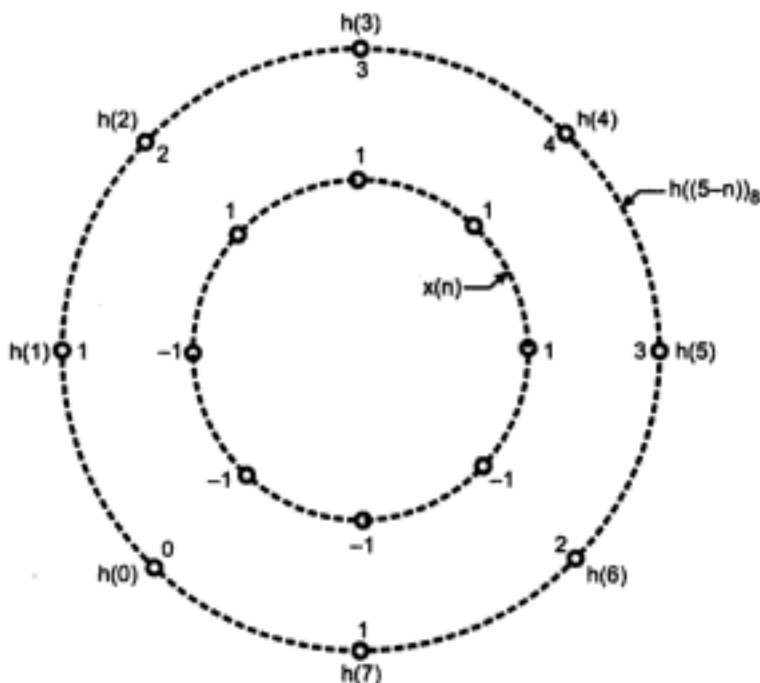


Fig. 5.3.20 Computation of $y(5) = \sum_{n=0}^7 x(n) h((5-n))_8$

To obtain $y(6)$, shift $h(n)$ anticlockwise by one sample position. Then multiply the two sequences point to point. Summation of these multiplications give the value of $y(6)$. Thus from Fig. 5.3.21 we can write,

$$\begin{aligned}
 y(6) &= (1 \times 2) + (1 \times 3) + (1 \times 4) + (1 \times 3) + (-1 \times 2) \\
 &\quad + (-1 \times 1) + (-1 \times 0) + (-1 \times 1) \\
 &= 2 + 3 + 4 + 3 - 2 - 1 - 0 - 1 \\
 &= 8
 \end{aligned}$$

$y(6) = 8$

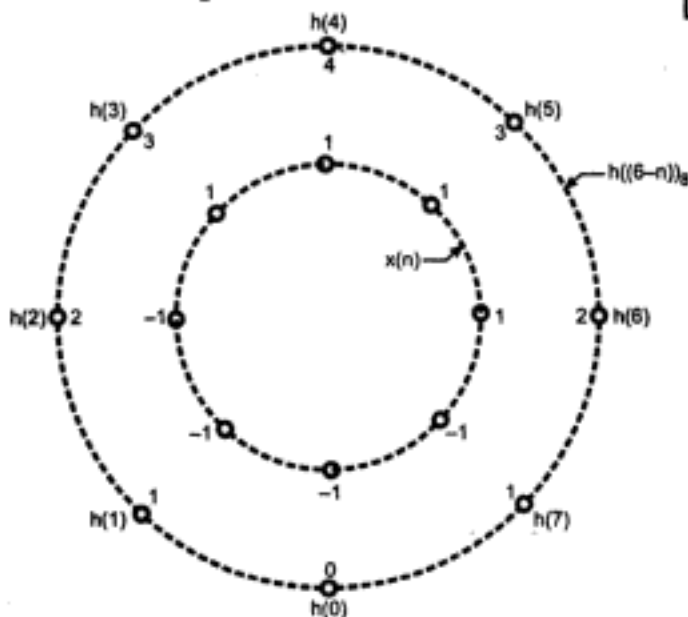


Fig. 5.3.21 Computation of $y(6) = \sum_{n=0}^7 x(n) h((6-n))_8$

Hidden page

Hidden page

and $x(n) = \{1, 0.5, 1, 0.5, 1, 0.5, 1, 0.5\}$

Putting above values in equation 5.3.76 we get,

$$\begin{aligned}
 \begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \\ y(4) \\ y(5) \\ y(6) \\ y(7) \end{bmatrix} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 3 & 2 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 3 & 2 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 & 3 \\ 3 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0.5 \\ 1 \\ 0.5 \\ 1 \\ 0.5 \\ 1 \\ 0.5 \end{bmatrix} \\
 &= \begin{bmatrix} (0 \times 1) + (0 \times 0.5) + (0 \times 1) + (0 \times 0.5) + (0 \times 1) + (3 \times 0.5) + (2 \times 1) + (1 \times 0.5) \\ (1 \times 1) + (0 \times 0.5) + (0 \times 1) + (0 \times 0.5) + (0 \times 1) + (0 \times 0.5) + (3 \times 1) + (2 \times 0.5) \\ (2 \times 1) + (1 \times 0.5) + (0 \times 1) + (0 \times 0.5) + (0 \times 1) + (0 \times 0.5) + (0 \times 1) + (3 \times 0.5) \\ (3 \times 1) + (2 \times 0.5) + (1 \times 1) + (0 \times 0.5) + (0 \times 1) + (0 \times 0.5) + (0 \times 1) + (0 \times 0.5) \\ (0 \times 1) + (3 \times 0.5) + (2 \times 1) + (1 \times 0.5) + (0 \times 1) + (0 \times 0.5) + (0 \times 1) + (0 \times 0.5) \\ (0 \times 1) + (0 \times 0.5) + (3 \times 1) + (2 \times 0.5) + (1 \times 1) + (0 \times 0.5) + (0 \times 1) + (0 \times 0.5) \\ (0 \times 1) + (0 \times 0.5) + (0 \times 1) + (3 \times 0.5) + (2 \times 1) + (1 \times 0.5) + (0 \times 1) + (0 \times 0.5) \\ (0 \times 1) + (0 \times 0.5) + (0 \times 1) + (0 \times 0.5) + (3 \times 1) + (2 \times 0.5) + (1 \times 1) + (0 \times 0.5) \end{bmatrix} \\
 &= \begin{bmatrix} 0 + 0 + 0 + 0 + 0 + 1.5 + 2 + 0.5 \\ 1 + 0 + 0 + 0 + 0 + 0 + 3 + 1 \\ 2 + 0.5 + 0 + 0 + 0 + 0 + 0 + 1.5 \\ 3 + 1 + 1 + 0 + 0 + 0 + 0 + 0 \\ 0 + 1.5 + 2 + 0.5 + 0 + 0 + 0 + 0 \\ 0 + 0 + 3 + 1 + 1 + 0 + 0 + 0 \\ 0 + 0 + 0 + 1.5 + 2 + 0.5 + 0 + 0 \\ 0 + 0 + 0 + 0 + 3 + 1 + 1 + 0 \end{bmatrix} \\
 &= \begin{bmatrix} 4 \\ 5 \\ 4 \\ 5 \\ 4 \\ 5 \\ 4 \\ 5 \end{bmatrix}
 \end{aligned}$$

Thus we obtained sequence $y(n)$ as,

$$y(n) = \{4, 5, 4, 5, 4, 5, 4, 5\}$$

Hidden page

Hidden page

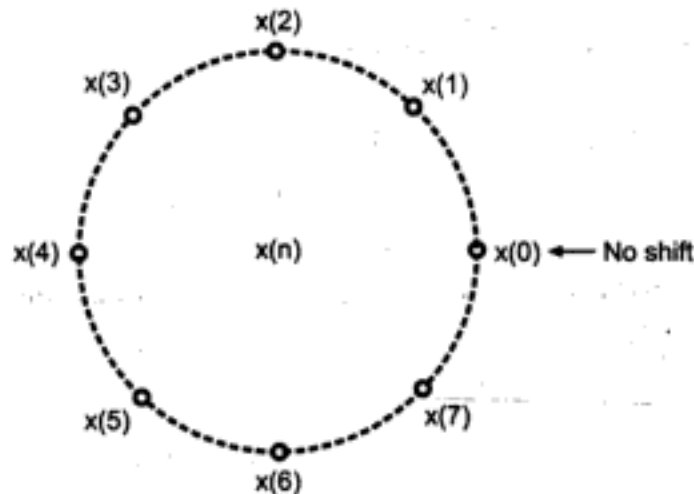


Fig. 5.3.24 A sequence having eight ($N = 8$) samples plotted across the circle. This sequence has no shift

shifted sequence $x((n-l))_N$ with $l=2$ and $N=8$ is shown below in Fig. 5.3.25. The DFT of this circularly shifted sequence can be splitted into two parts. It is given as [see two DFTs in Fig. 5.3.25],

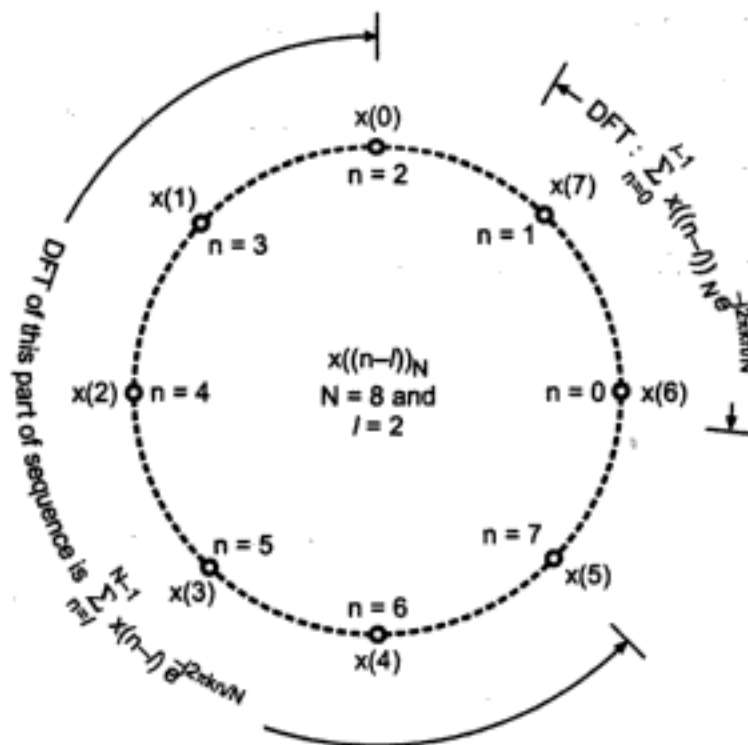


Fig. 5.3.25 DFT of a circularly shifted sequence is split in two parts as shown

$$DFT \{x((n-l))_N\} = \sum_{n=l}^{N-1} x(n-l) e^{-j2\pi kn/N} + \sum_{n=0}^{l-1} x((n-l))_N e^{-j2\pi kn/N} \quad \dots (5.3.85)$$

Here $x((n-l))_N$ can be written as $x(N-l+n)$ since this is circular shift. Hence second summation in above equation becomes,

Hidden page

Hidden page

Hidden page

$$DFT \{y^*(-l)\} = \sum_{l=0}^{N-1} y^*(-l) e^{-j2\pi kl/N} \quad \dots (5.3.97)$$

Let $n = -l$, then limits of summation will be,

When $l = 0$, $n = 0$ and

When $l = N - 1$, $n = -(N - 1)$

Then equation 5.3.97 becomes,

$$DFT \{y^*(-l)\} = \sum_{n=0}^{-(N-1)} y^*(n) e^{j2\pi kn/N}$$

The sequence $y^*(n)$ is circular in nature. Hence summation from 0 to $-(N - 1)$ will be same as from 0 to $(N - 1)$. Hence above equation can be written as,

$$\begin{aligned} DFT \{y^*(-l)\} &= \sum_{n=0}^{N-1} y^*(n) e^{j2\pi kn/N} \\ &= \left[\sum_{n=0}^{N-1} y(n) e^{-j2\pi kn/N} \right]^* \\ &= [Y(k)]^* \quad \text{by definition of DFT} \\ &= Y^*(k) \end{aligned}$$

Putting this value in equation 5.3.96 we get,

$$\tilde{R}_{xy}(k) = X(k) Y^*(k) \quad \dots (5.3.98)$$

Thus equation 5.3.93 is proved. When $x(n) = y(n)$ we get circular autocorrelation. Then equation 5.3.93 can be written as,

$$\tilde{r}_{xx}(l) \xleftrightarrow[N]{DFT} \tilde{R}_{xx}(k) = X(k) X^*(k) \quad \dots (5.3.99)$$

We know that $X(k) X^*(k) = |X(k)|^2$, Hence above equation becomes,

$$\tilde{r}_{xx}(l) \xleftrightarrow[N]{DFT} \tilde{R}_{xx}(k) = |X(k)|^2 \quad \dots (5.3.100)$$

5.3.3.11 Multiplication of Two Sequences

This property states that if

$$x_1(n) \xleftrightarrow[N]{DFT} X_1(k) \quad \text{and}$$

$$x_2(n) \xleftrightarrow[N]{DFT} X_2(k) \quad \text{then,}$$

$$x_1(n) x_2(n) \xleftrightarrow[N]{DFT} \frac{1}{N} X_1(k) \circledast X_2(k) \quad \dots (5.3.101)$$

This means multiplication of two sequences in time domain results in circular convolution of their DFTs in frequency domain.

Proof :

This property can be proved by the same method which is discussed for circular convolution. The proof can be started with DFT of $x_3(m)$ which is product of $x_1(n)$ and $x_2(n)$. Then putting for IDFTs of $x_1(n)$ and $x_2(n)$, the derived equation 5.3.101 can be obtained.

5.3.3.12 Parseval's Theorem

Consider the complex valued sequences $x(n)$ and $y(n)$. Then if,

$$\begin{aligned} x(n) &\xleftrightarrow{\frac{\text{DFT}}{N}} X(k) \quad \text{and} \\ y(n) &\xleftrightarrow{\frac{\text{DFT}}{N}} Y(k) \quad \text{then} \\ \sum_{n=0}^{N-1} x(n)y^*(n) &= \frac{1}{N} \sum_{k=0}^{N-1} X(k)Y^*(k) \end{aligned} \quad \dots(5.3.102)$$

When $y(n) = x(n)$ above equation becomes,

$$\sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2 \quad \dots (5.3.103)$$

The above two equations are relations are Parseval's Theorem. Above equation give energy of finite duration sequence in terms of its frequency components.

Proof :

Circular correlation is given by equation 5.3.94 as,

$$\tilde{r}_{xy}(l) = \sum_{n=0}^{N-1} x(n)y^*((n-l))_N \quad \dots (5.3.104)$$

For $l=0$ above equation becomes,

$$\tilde{r}_{xy}(0) = \sum_{n=0}^{N-1} x(n)y^*(n) \quad \dots (5.3.105)$$

From equation 5.3.93 we know that

$$\text{DFT} \{ \tilde{r}_{xy}(l) \} = X(k)Y^*(k)$$

i.e. ,
$$\tilde{r}_{xy}(l) = \text{IDFT} \{ X(k)Y^*(k) \}$$

By definition of IDFT,

$$\tilde{r}_{xy}(l) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)Y^*(k) e^{j2\pi kl/N}$$

With $l=0$, above equation becomes,

$$\tilde{r}_{xy}(0) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)Y^*(k) \quad \dots (5.3.106)$$

Equating above equation with equation 5.3.105 we have,

$$\sum_{n=0}^{N-1} x(n) y^*(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) Y^*(k)$$

This is the required relation.

The properties of DFT are summarized below in Table 5.3.3.

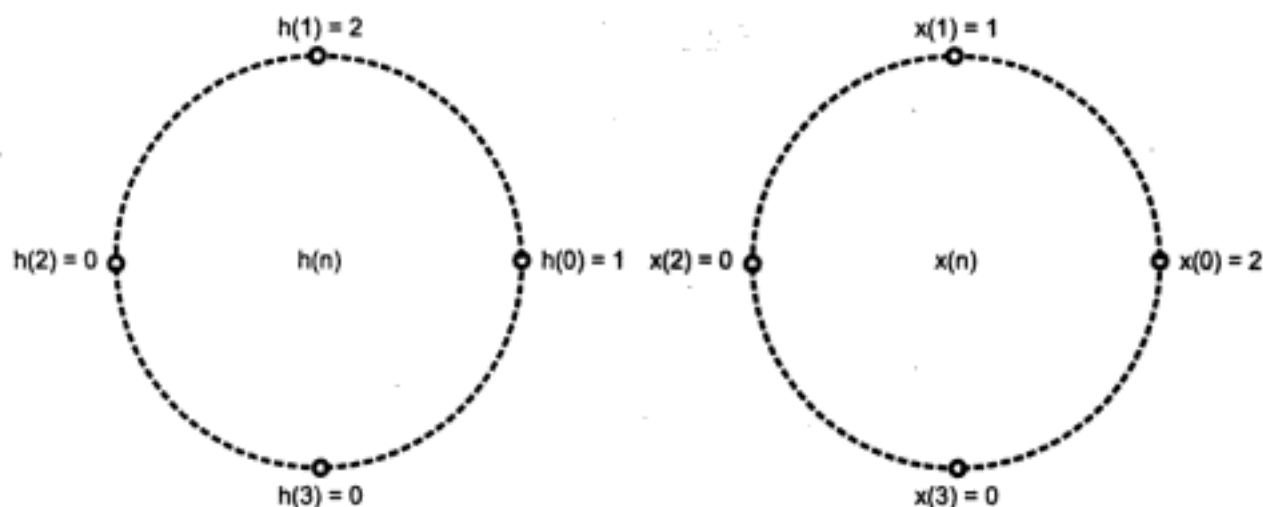
Table 5.3.3 Properties of DFT

Sr. No.	Name of the property	Time domain representation	Frequency domain representation
1	Periodicity	$x(n) = x(n + N)$	$X(k) = X(k + N)$
2	Linearity	$a_1 x_1(n) + a_2 x_2(n)$	$a_1 X_1(k) + a_2 X_2(k)$
3	Symmetry properties	$x^*(n)$ $x^*(N - n)$ For real valued $x(n)$	$X^*(N - k)$ $X^*(k)$ $X(k) = X^*(N - k)$ $X_R(k) = X_R(N - k)$ $X_I(k) = -X_I(N - k)$ $ X(k) = X(N - k) $ $\angle X(k) = -\angle X(N - k)$
4	Circular convolution	$x_1(n) \circledast x_2(n)$	$X_1(k) X_2(k)$
5	Circular time reversal	$x((-n))_N = x(N - n)$	$X((-k))_N = X(N - k)$
6	Circular time shift	$x((n - l))_N$	$X(k) e^{-j2\pi kl/N}$
7	Circular frequency shift	$x(n) e^{j2\pi ln/N}$	$X((k - l))_N$
8	Complex conjugate properties	$x^*(n)$ $x^*((-n))_N = x^*(N - k)$	$X^*((-k))_N = X^*(N - k)$ $X^*(k)$
9	Circular correlation	$\tilde{r}_{xy}(l) = x(l) \circledast y^*(-l)$	$X(k) Y^*(k)$
10	Multiplication of two sequences	$x_1(n) x_2(n)$	$\frac{1}{N} X_1(k) \circledast X_2(k)$
11	Parsevals Theorem	$\sum_{n=0}^{N-1} x(n) ^2$ $\sum_{n=0}^{N-1} x(n) y^*(n)$	$\frac{1}{N} \sum_{k=0}^{N-1} X(k) ^2$ $\frac{1}{N} \sum_{k=0}^{N-1} X(k) Y^*(k)$

Hidden page

Hidden page

Hidden page

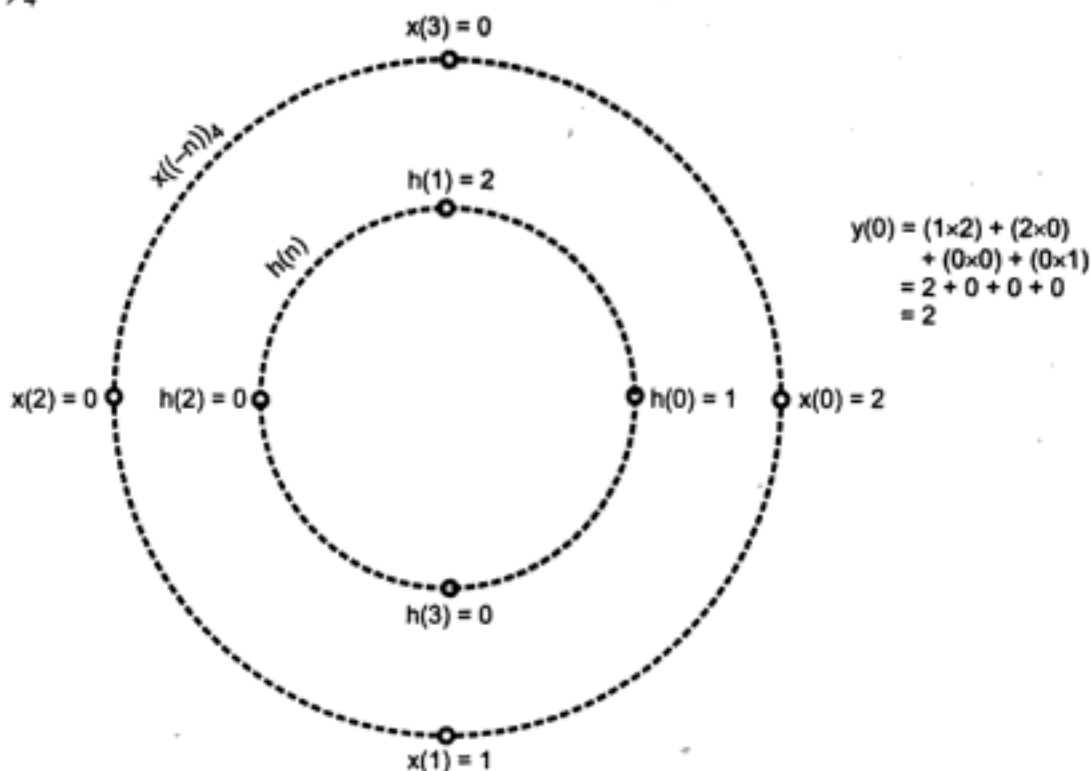
Fig. 5.3.28 (a) Sequence $h(n)$ (b) Sequence $x(n)$

The circular convolution of $x(n)$ and $h(n)$ is given as,

$$y(m) = \sum_{n=0}^3 h(n) x((m-n))_4, \quad m = 0, 1, 2, 3$$

$$\therefore y(0) = \sum_{n=0}^3 h(n) x((-n))_4$$

Fig. 5.3.29 shows $h(n)$ and $x((-n))_4$ plotted on concentric circles to obtain $y(0)$. $x((-n))_4$ is obtained by circular folding of $x(n)$ of Fig. 5.3.28 (b).

Fig. 5.3.29 Circular convolution to obtain $y(0)$

As shown in Fig. 5.3.29 the two sequences are multiplied point by point and the products are added. The value of $y(0) = 2$ as obtained above.

Hidden page

Hidden page

$$h(n) = \left\{ \underbrace{h(0), h(1), \dots, h(M-1)}_{\substack{\text{'M' samples of unit sample response} \\ \text{(L-1) zeros are padded to make } N=L+M-1 \text{ total samples}}} , \underbrace{0, 0, \dots, (L-1 \text{ zeros})}_{\substack{\text{(L-1) zeros are padded to make } N=L+M-1 \text{ total samples}}} \right\} \quad \dots (5.3.120)$$

Thus as shown above $h(n)$ also contains $N = L + M - 1$ samples. The N -point DFT of this $h(n)$ will be $H(k)$. Let the DFT of m^{th} input data block be $X_m(k)$ and corresponding DFT of output be $\hat{Y}_m(k)$. i.e.,

$$\hat{Y}_m(k) = H(k) X_m(k), \quad k = 0, 1, \dots, N-1 \quad \dots (5.3.121)$$

The sequence $\hat{y}_m(n)$ can be obtained by taking N -point IDFT of $\hat{Y}_m(k)$. Then the individual samples of this sequence can be represented as,

$$\hat{y}_m(n) = \{ \hat{y}_m(0), \hat{y}_m(1), \dots, \hat{y}_m(M-1), \hat{y}_m(M), \hat{y}_m(M+1), \dots, \hat{y}_m(N-1) \} \quad \dots (5.3.122)$$

This type of sequence will be obtained due to input data blocks $x_1(n), x_2(n), x_3(n), \dots$ etc. The sequence $\hat{y}_m(n)$ contains $N = L + M - 1$ samples. Observe that in $x_1(n), x_2(n), x_3(n), \dots$ etc. initial $M - 1$ samples are taken from previous segment i.e. overlap, and last 'L' samples are actual input samples. Because of this overlap of initial $(M - 1)$ samples in input data block, there is aliasing in initial $(M - 1)$ samples in the corresponding output data block i.e. $\hat{y}_m(n)$. The aliasing effect occurs because of circular shift and overlap of samples in computation of DFT. Hence the initial $(M - 1)$ samples of $\hat{y}_m(n)$ must be discarded. The last 'L' samples of $\hat{y}_m(n)$ are the correct output samples. Hence the actual output samples to be considered in every output block are,

$$\hat{y}_m(n) = y_m(n), \quad \text{for } n = M, M+1, \dots, N-1 \quad \dots (5.3.123)$$

Such blocks are fitted one after another to get the final output. The overlap save method discussed above is illustrated in Fig. 5.3.31.

Please refer Fig. 5.3.31 on next page.

5.3.5.3 Overlap Add Method for Linear Filtering

In this method the data blocks of length $N = L + M - 1$ are formed by taking 'L' samples from input sequence and padding $M - 1$ zeros as shown below.

$$x_1(n) = \left\{ \underbrace{x(0), x(1), \dots, x(L-1)}_{\substack{\text{'L' samples of input data sequence } x(n) \\ \text{(M-1) zeros are padded at the end}}} , \underbrace{0, 0, \dots, 0}_{\substack{\text{(M-1) zeros are padded at the end}}} \right\} \quad \dots (5.3.124)$$

$$x_2(n) = \left\{ \underbrace{x(L), x(L+1), \dots, x(2L-1)}_{\substack{\text{Next 'L' samples of input sequence } x(n) \\ \text{(M-1) zeros are padded at the end}}} , \underbrace{0, 0, \dots, 0}_{\substack{\text{(M-1) zeros are padded at the end}}} \right\} \quad \dots (5.3.125)$$

$$x_3(n) = \left\{ \underbrace{x(2L), x(2L+1), \dots, x(3L-1)}_{\substack{\text{Next 'L' samples of input sequence } x(n) \\ \text{(M-1) zeros}}} , \underbrace{0, 0, \dots, 0}_{\substack{\text{(M-1) zeros}}} \right\} \quad \dots (5.3.126)$$

Thus each data block is of length 'N'. The N -point DFT $Y_m(k)$ of the output is obtained by multiplying $H(k)$ and $X_m(k)$. i.e.,

$$Y_m(k) = H(k) \cdot X_m(k), \quad k = 0, 1, \dots, N-1 \quad \dots (5.3.127)$$

Here $H(k)$ is N -point DFT of unit sample response $h(n)$ and $X_m(k)$ is DFT of m^{th} data block.

Hidden page

The sequence $y_m(n)$ is obtained by taking N-point IDFT of $Y_m(k)$. Thus samples of sequence $y_m(n)$ will be,

$$y_1(n) = \{y_1(0), y_1(1), \dots, y_1(L-1), y_1(L), y_1(L+1), \dots, y_1(N-1)\} \quad \dots (5.3.128)$$

$$y_2(n) = \{y_2(0), y_2(1), \dots, y_2(L-1), y_2(L), y_2(L+1), \dots, y_2(N-1)\} \quad \dots (5.3.129)$$

Similarly other sequences are obtained. We know that each data block is terminated with $M-1$ zeros. Hence the last $M-1$ samples of each output sequence must be overlapped and added to first $M-1$ samples of succeeding output sequence. This is done since the sequence is appended with zeros at the end. And hence the name overlap and add is given. For example the output $y(n)$ due to overlap and adding of $y_1(n)$ and $y_2(n)$ is given as follows :

$$y(n) = \{y_1(0), y_1(1), \dots, y_1(L-1), [y_1(L) + y_2(0)], [y_1(L+1) + y_2(1)], \dots, [y_1(N-1) + y_2(M-1)], y_2(M), \dots, y_2(N-1)\} \quad \dots (5.3.130)$$

This process continues till the end of input sequence. This algorithm is illustrated in Fig.3.3.32 on next page.

Here note that the $(M-1)$ overlapping samples of the output blocks are not discarded. This is because the input data blocks are padded with $(M-1)$ zeros to make their length equal to 'N'. Hence there will be no aliasing (effect due to circular shifting and overlap in DFT) in the output data blocks. Therefore for the last $(M-1)$ samples of current output block must be added to the first $(M-1)$ samples of next output block. This is the major difference between this method and overlap save method. In overlap save method the initial $(M-1)$ samples of every output block are discarded since they are aliased due to overlap.

Comments :

Even though overlap save and overlap add methods seem to be complicated, the computations involved actually are less compared to linear convolution. This is because DFT can be computed fast using FFT algorithms. These algorithms compute DFT with very small number of computations. These algorithms are discussed in next section.

5.3.5.4 Frequency or Spectrum Analysis using DFT

When we want to analyze the spectrum of the signal its fourier transform is taken. For example consider the sequence $x(n)$ and its fourier transform $X(\omega)$. Since $X(\omega)$ is continuous function of ' ω ', it is not possible to analyze $X(\omega)$ on digital computer or digital signal processor. Hence DFT of the signal can be used for spectrum analysis.

The signal to be analyzed is passed through the antialiasing filter and sampled at the rate of $F_s \geq 2F_{\max}$. Hence highest frequency in the sampled signal is $\frac{F_s}{2}$. For spectrum analysis

some finite length of the sequence is taken. Let 'T' be the sampling interval and 'L' number of samples of input sequence are taken. Then the time interval of the sequence will be $T_0 = LT$. It can be shown that the time interval of the sequence should be as large as possible. Because the smallest frequency resolution is given as $\frac{1}{T_0}$. Let the 'L' number of samples of the sequence

$x(n)$ be obtained by multiplying $x(n)$ by rectangular window $w(n)$ of length 'L'. i.e.,

$$\hat{x}(n) = x(n)w(n) \quad \dots (5.3.131)$$

Here $w(n)$ is the rectangular window which is given as,

$$w(n) = \begin{cases} 1, & \text{for } 0 \leq n \leq L-1 \\ 0 & \text{otherwise} \end{cases} \quad \dots (5.3.132)$$

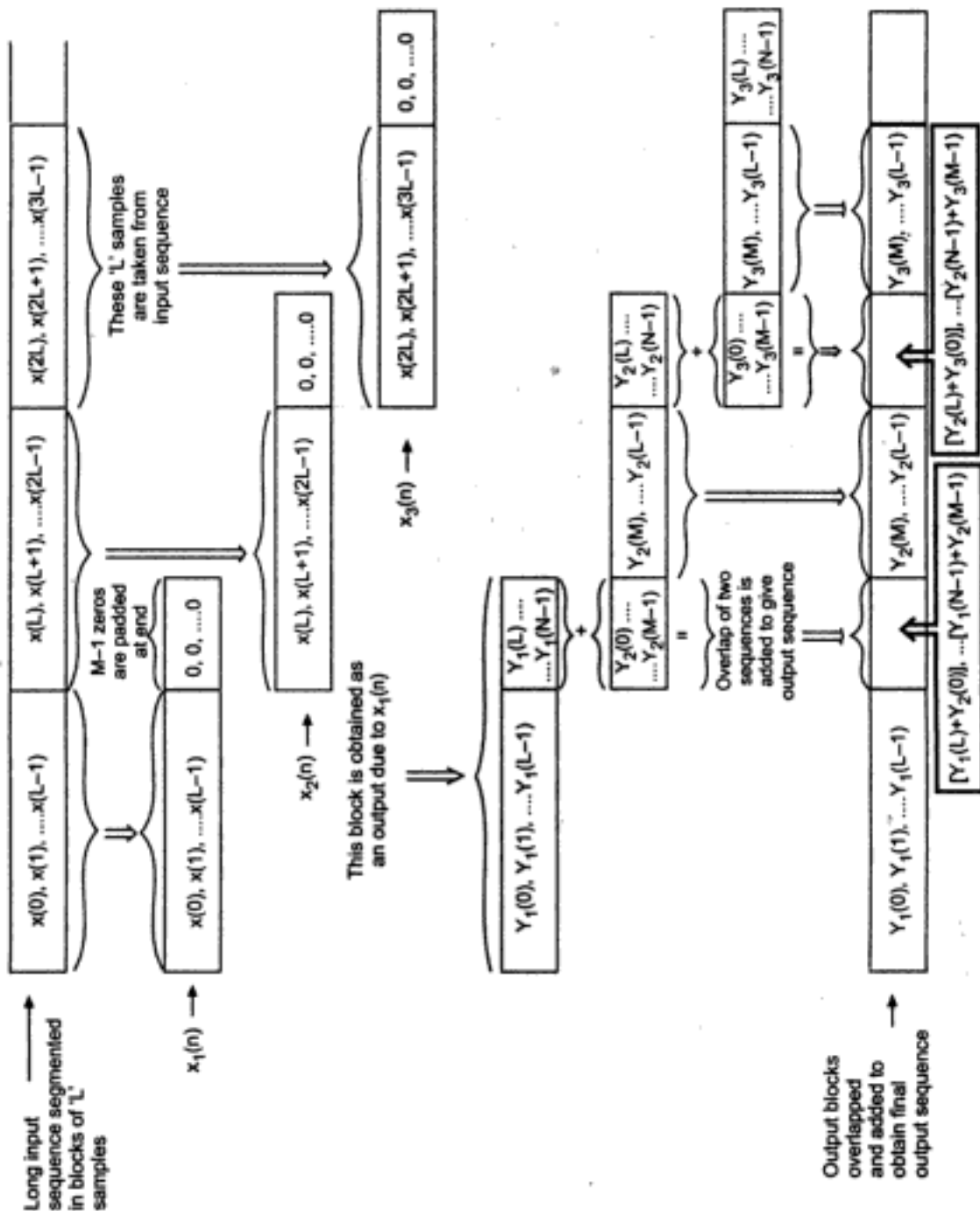


Fig. 5.3.32 Overlap add method for filtering of long data sequences using DFT

Let $x(n)$ be the cosine wave consisting single frequency which is given as,

$$x(n) = \cos(\omega_0 n) \quad \dots (5.3.133)$$

Then the finite length sequence $\hat{x}(n)$ becomes,

$$\hat{x}(n) = \cos(\omega_0 n) \quad \text{for} \quad 0 \leq n \leq L-1 \quad \dots (5.3.134)$$

Hidden page

Hidden page

Hence equation 5.4.6 becomes,

$$\begin{aligned} W_N^{k + \frac{N}{2}} &= -e^{-j \frac{2\pi}{N} k} \\ &= -W_N^k, \quad \text{since } e^{-j \frac{2\pi}{N}} = W_N \end{aligned}$$

Prove that $W_N^2 = W_{N/2}$... (5.4.7)

We know that $W_N = e^{-j \frac{2\pi}{N}}$

In this equation replace 'N' by N/2, i.e.

$$\begin{aligned} W_{N/2} &= e^{-j \frac{2\pi}{N/2}} \\ &= e^{-j \frac{2\pi}{N} \cdot 2} \\ &= W_N^2 \quad \text{since } e^{-j \frac{2\pi}{N}} = W_N \end{aligned}$$

which is same as equation 5.4.7.

The direct computation of DFT does not use these properties of W_N . The FFT algorithms exploit these properties of W_N to reduce calculations of DFT as discussed next.

5.4.3 Classification of FFT Algorithms

The FFT algorithms are based on two basic methods. The first one is divide and conquer approach. In this method the 'N' point DFT is divided successively to 2-point DFTs to reduce calculations. In this method, Radix-2, Radix-4, decimation in time, decimation in frequency etc type of FFT algorithms are developed.

The second one is based on linear filtering. Based on this method, there are two algorithms. Goertzel algorithm and the chirp-z transform algorithm. This complete classification is listed below in Fig. 5.4.1.

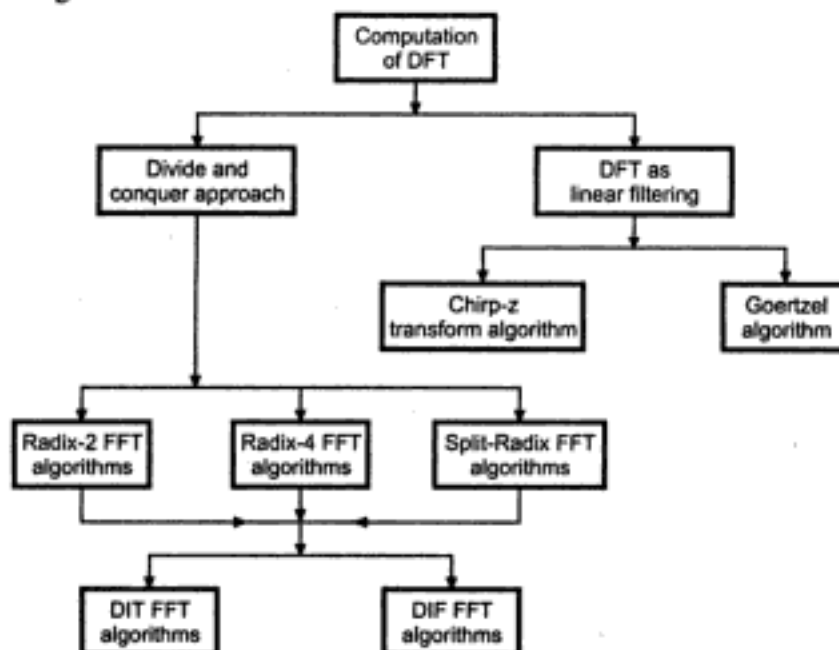


Fig. 5.4.1 Various FFT algorithms and their classification

Hidden page

Hidden page

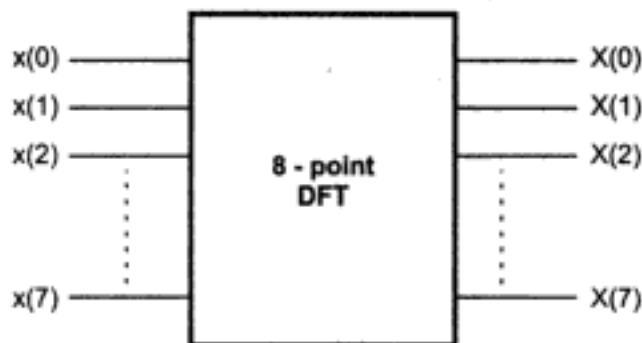


Fig. 5.5.1 Direct computation of 8-point DFT

diagram for this operation. In Fig. 5.5.2 observe that for 8-point $x(n)$, the sequences $f_1(m)$ and $f_2(m)$ are given as [see equation 5.5.1],

$$\left. \begin{aligned} f_1(m) &= x(2n) = \{x(0), x(2), x(4), x(6)\} \\ f_2(m) &= x(2n+1) = \{x(1), x(3), x(5), x(7)\} \end{aligned} \right\} - \frac{N}{2} \text{ point sequences} \quad \dots (5.5.9)$$

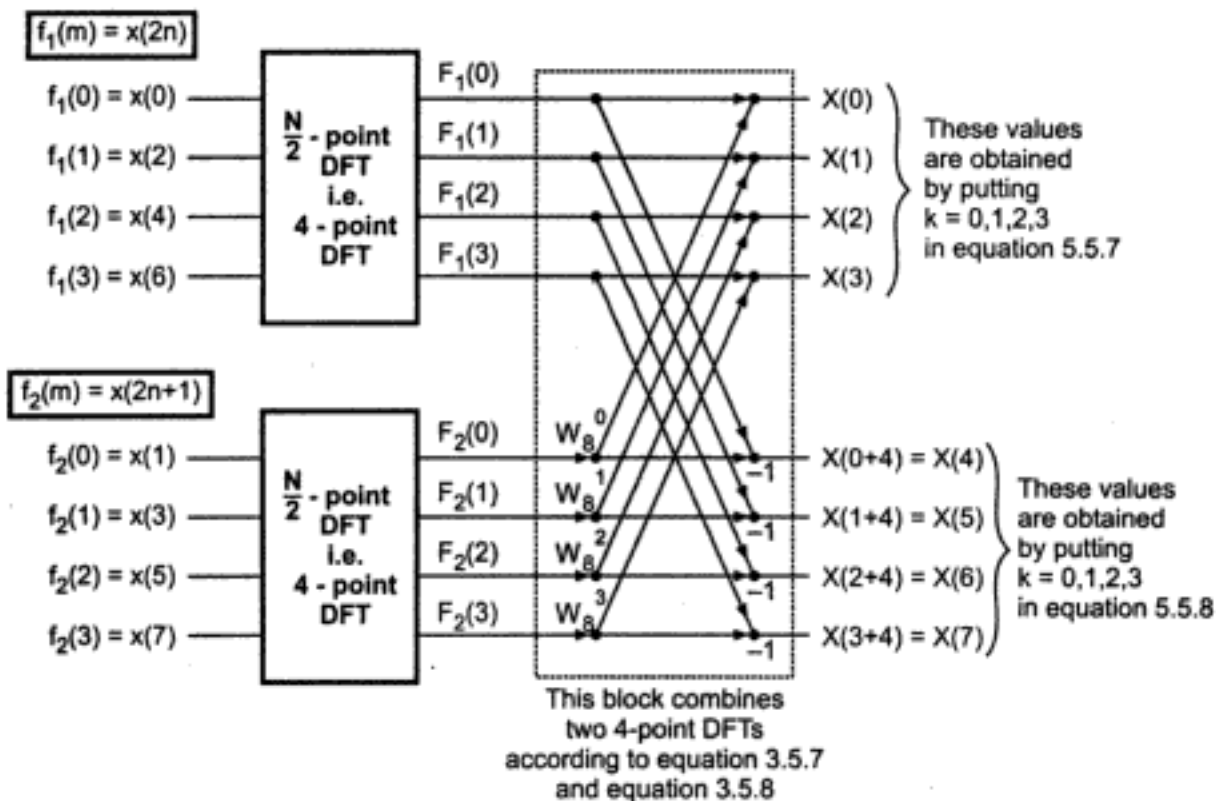


Fig. 5.5.2 8-point DFT of $X(k)$ is obtained by combining two 4-point DFTs $F_1(k)$ and $F_2(k)$

As shown in Fig. 5.5.2 two $\frac{N}{2}$ - point DFTs are to be computed separately and then they are combined as per equation 5.5.7 and equation 5.5.8 to get N-point DFT.

Second stage of decimation on $f_1(n)$ and $f_2(n)$:

We know that $f_1(n)$ and $f_2(n)$ are $\frac{N}{2}$ point sequences. Let $f_1(n)$ be splitted into even numbered samples and odd numbered samples as follows :

$$\left. \begin{aligned} v_{11}(n) &= f_1(2n), & n = 0, 1, \dots, \frac{N}{4} - 1 \\ \text{and } v_{12}(n) &= f_1(2n+1), & n = 0, 1, \dots, \frac{N}{4} - 1 \end{aligned} \right\} \dots (5.5.10)$$

Here observe that $v_{11}(n)$ is even numbered sequence of $f_1(n)$ and $v_{12}(n)$ is odd numbered sequence of $f_1(n)$. Since $f_1(n)$ contain $N/2$ samples. $v_{11}(n)$ and $v_{12}(n)$ contain $N/4$ samples.

Similarly $f_2(n)$ can be splitted into even and odd numbered sequences as follows :

$$\left. \begin{aligned} v_{21}(n) &= f_2(2n), & n = 0, 1, \dots, \frac{N}{4} - 1 \\ \text{and } v_{22}(n) &= f_2(2n+1), & n = 0, 1, \dots, \frac{N}{4} - 1 \end{aligned} \right\} \dots (5.5.11)$$

$v_{21}(n)$ is even numbered sequence of $f_2(n)$ and

$v_{22}(n)$ is odd numbered sequence of $f_2(n)$. Both $v_{21}(n)$ and $v_{22}(n)$ contain $\frac{N}{4}$ samples each.

We obtained $X(k)$ and $X\left(k + \frac{N}{2}\right)$ of equation 5.5.7 and equation 5.5.8 from $F_1(k)$ and $F_2(k)$ with $f_1(n)$ and $f_2(n)$ as decimated sequences. The length of DFT was $\frac{N}{2}$. By using the similar method we can obtain $F_1(k)$ and $F_1\left(k + \frac{N}{4}\right)$ from $V_{11}(k)$ and $V_{12}(k)$. Here $V_{11}(k)$ is $\frac{N}{4}$ point DFT of $v_{11}(n)$ and $V_{12}(k)$ is $\frac{N}{4}$ point DFT of $v_{12}(n)$ of equation 5.5.10. i.e.,

$$F_1(k) = V_{11}(k) + W_{N/2}^k V_{12}(k), \quad k = 0, 1, \dots, \frac{N}{4} - 1 \quad \dots (5.5.12)$$

$$F_1\left(k + \frac{N}{4}\right) = V_{11}(k) - W_{N/2}^k V_{12}(k), \quad k = 0, 1, \dots, \frac{N}{4} - 1 \quad \dots (5.5.13)$$

Compare the above two equations with equation 5.5.7 and equation 5.5.8. In equation 5.5.7 and equation 5.5.8 N -point DFT is obtained from two $\frac{N}{2}$ -point DFTs. In the above equations $\frac{N}{2}$ point DFT (i.e. $F_1(k)$) is obtained from two $\frac{N}{4}$ point DFTs.

Similarly we can write equations for $F_2(k)$ as follows,

$$F_2(k) = V_{21}(k) + W_{N/2}^k V_{22}(k), \quad k = 0, 1, \dots, \frac{N}{4} - 1 \quad \dots (5.5.14)$$

$$F_2\left(k + \frac{N}{4}\right) = V_{21}(k) - W_{N/2}^k V_{22}(k), \quad k = 0, 1, \dots, \frac{N}{4} - 1 \quad \dots (5.5.15)$$

Here $V_{21}(k)$ is $\frac{N}{4}$ point DFT of $v_{21}(n)$ and $V_{22}(k)$ is $\frac{N}{4}$ point DFT of $v_{22}(n)$ of equation 5.5.11.

Recall the example of 8-point DFT :

Now let us recall the example of 8-point DFT. As shown in Fig. 5.5.2, $F_1(k)$ and $F_2(k)$ are two 4-point DFTs computed directly. These two DFTs are represented by single blocks symbolically as shown below in Fig. 5.5.3 below.

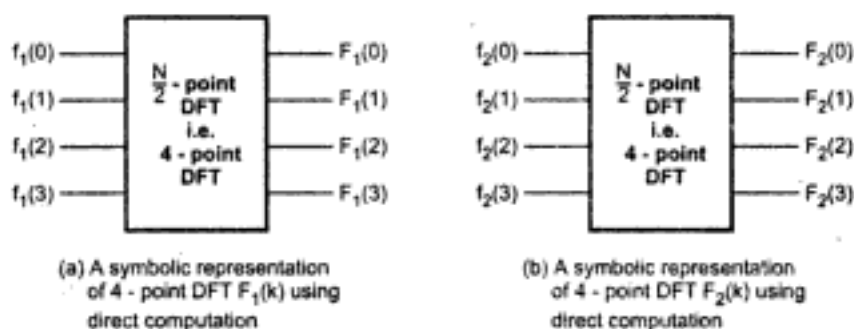


Fig. 5.5.3

$F_1(k)$ can be obtained from $V_{11}(k)$ and $V_{12}(k)$ as per equation 5.5.12 and equation 5.5.13.

For $N=8$, $V_{11}(k)$ and $V_{12}(k)$ will be $\frac{N}{4}$ i.e. 2-point DFTs.

Fig. 5.5.4 (a) shows the symbolic diagram for this operation. In this figure observe that for 4-point $f_1(n)$ the sequences $v_{11}(n)$ and $v_{12}(n)$ are given as (see equation 5.5.10),

$$\left. \begin{aligned} v_{21}(n) &= f_1(2n) = x(4n) = \{x(0), x(4)\}, n = 0, 1 \\ v_{22}(n) &= f_1(2n+1) = x(4n+2) = \{x(2), x(6)\}, n = 0, 1 \end{aligned} \right\} - \frac{N}{4} \text{ point sequences} \quad \dots (5.5.16)$$

Thus as shown in Fig. 5.5.4(a) the two $\frac{N}{4}$ - point DFTs are to be computed separately and then they are combined as per equation 5.5.12 and equation 5.5.13 to get $\frac{N}{2}$ point DFT of $F_1(k)$.

Fig. 5.5.4 (b) shows the computation of $F_2(k)$ as per equation 5.5.14 and equation 5.5.15. In this figure observe that the two $\frac{N}{4}$ point DFTs are combined. In this figure observe that the $\frac{N}{4}$ point sequences $v_{21}(n)$ and $v_{22}(n)$ sequences are given as [see equation 5.5.11],

$$\left. \begin{aligned} v_{21}(n) &= f_2(2n) = x(4n+1) = \{x(1), x(5)\}, n = 0, 1 \\ v_{22}(n) &= f_2(2n+1) = x(4n+3) = \{x(3), x(7)\}, n = 0, 1 \end{aligned} \right\} - \frac{N}{4} \text{ point sequences} \quad \dots (5.5.17)$$

Computation of 2-point DFTs :

The $\frac{N}{4}$ point sequences of equation 5.5.10 and equation 5.5.11 are further splitted in their even and odd parts. Hence we get next stage of decimation and the sequences will be of length $\frac{N}{8}$. The procedure discussed till now can be repeated further to compute the DFTs, till

we reach to 2-point DFT. Since $N = 2^v$, we reach to 2-point DFT after $(v-1)$ decimations. For example for $N=8$, we performed $3-1=2$ (since $v=3$) decimations and we reached to 2-point DFTs as shown in Fig. 5.5.4. Now let us see how 2-point DFTs can be evaluated.

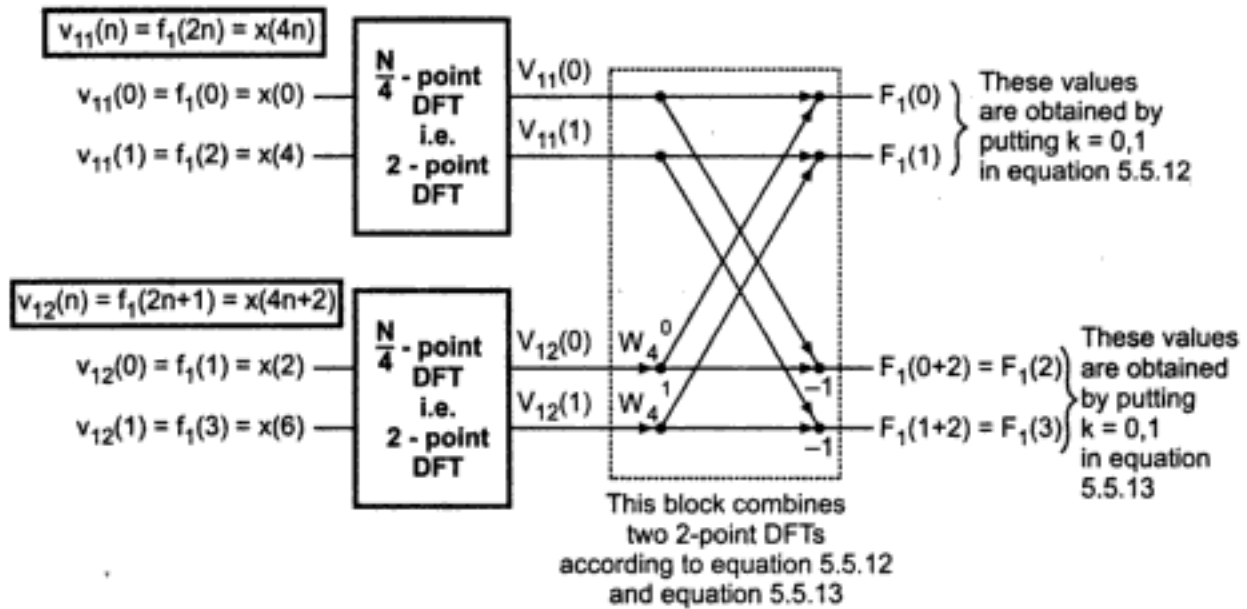


Fig. 5.5.4 (a) 4-point DFT of $F_1(k)$ is obtained by combining two 2-point DFTs of $V_{11}(k)$ and $V_{12}(k)$

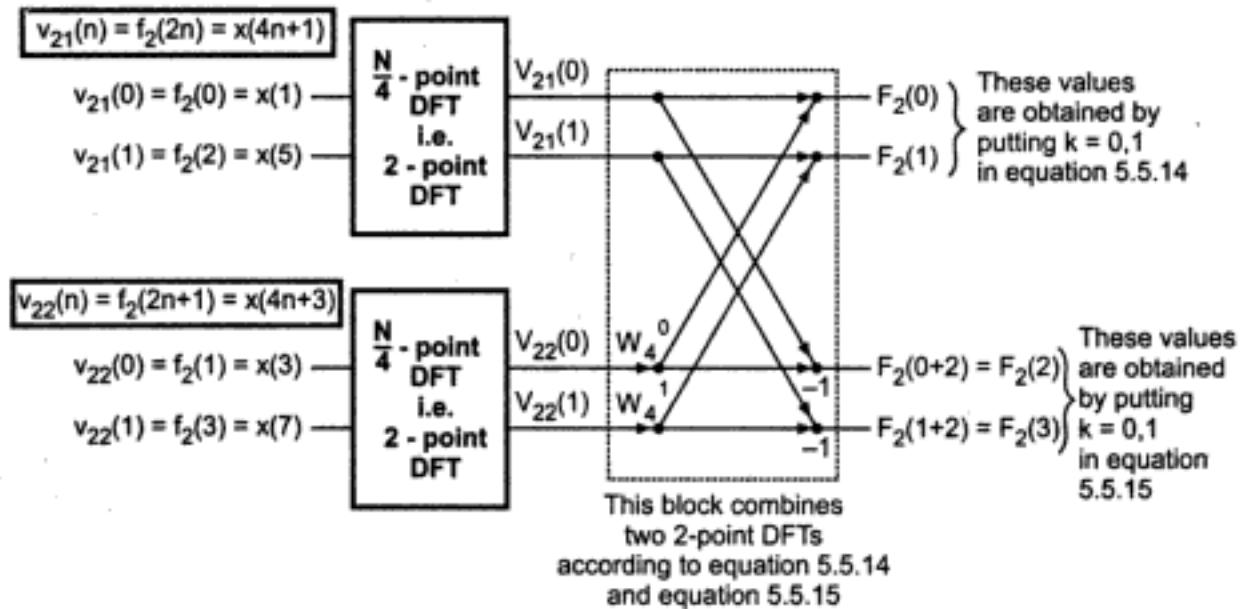


Fig. 5.5.4 (b) 4-point DFT of $F_2(k)$ is obtained by combining two 2-point DFTs of $V_{21}(k)$ and $V_{22}(k)$

By definition of N-point DFT,

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad k = 0, 1, \dots, N-1 \quad \dots (5.5.18)$$

Let us see the computation of 2-point DFT $V_{11}(k)$ of Fig. 5.5.4 (a).

From Fig. 5.5.5 we can write equation 5.5.18 as,

$$V_{11}(k) = \sum_{n=0}^1 v_{11}(n) W_2^{kn}, \quad k = 0, 1 \quad \dots (5.5.19)$$

Hidden page

Hidden page

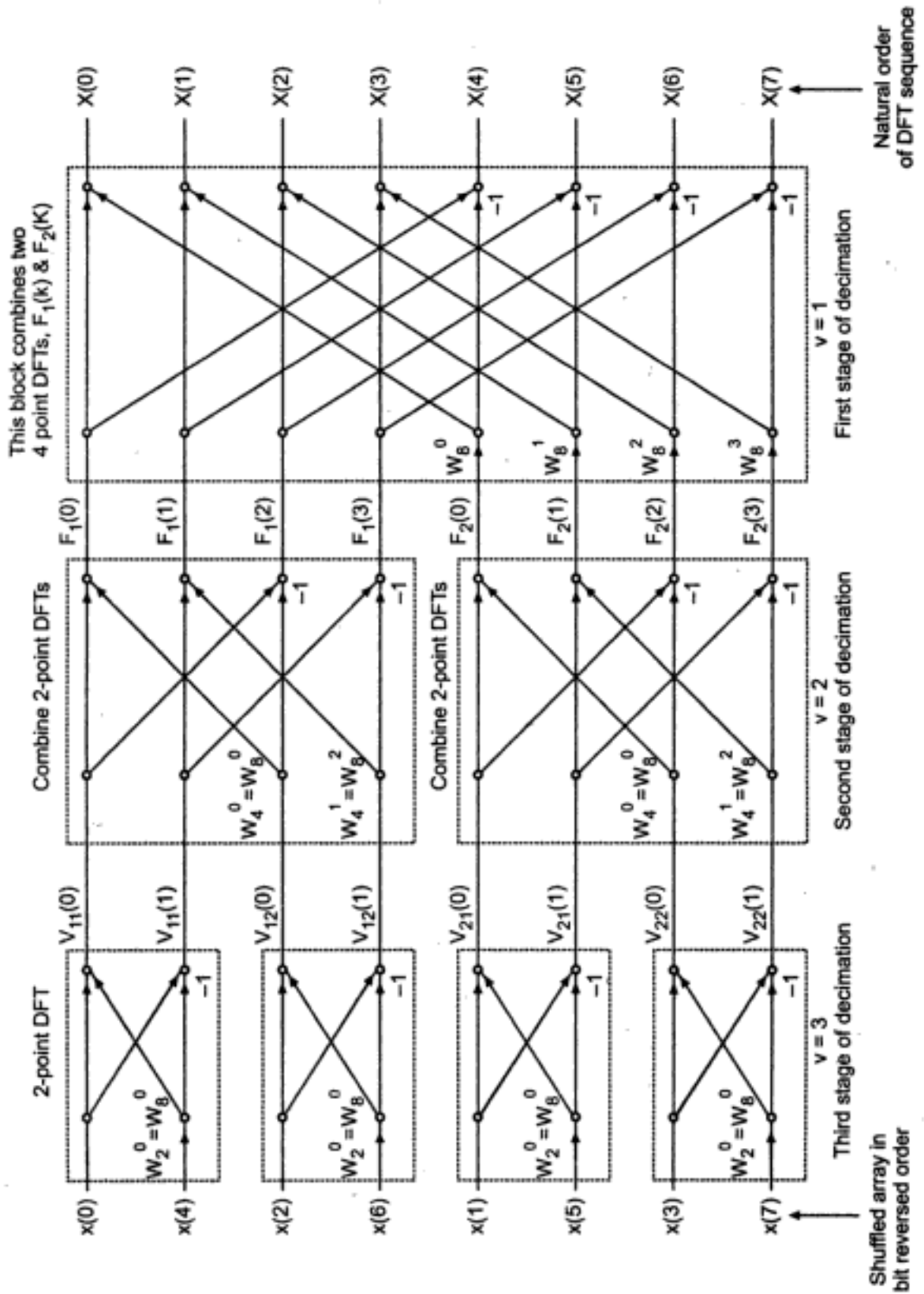


Fig. 5.5.7 Signal flow graph and stages of computation of Radix-2 DIT DFT algorithm for $N = 8$

5.5.2.2 Computational Complexity

Observe that the butterfly operation of DIF FFT given in Fig. 5.5.11 is similar to that of DIT FFT given in Fig. 5.5.9 except multiplication by factor W_N^r . Hence the butterfly of Fig. 5.5.11 also needs one complex multiplication and two complex additions. Since there are $\frac{N}{2}$ butterflies per stage and there are $v = \log_2 N$ stages, there are total $\frac{N}{2} \times v$ butterflies in the computation. Hence we can write,

$$\begin{aligned} \text{Number of complex multiplications} &= \frac{N}{2} \times v \\ &= \frac{N}{2} \log_2 N \quad \dots (5.5.50) \end{aligned}$$

$$\text{Number of complex additions} = 2 \times \left(\frac{N}{2} \times v \right) = N \log_2 N \quad \dots (5.5.51)$$

Thus the computational complexity of DIT FFT and DIF FFT algorithms is same (see equation 5.5.27 and equation 5.5.28).

5.5.2.3 Memory Requirement and Inplace Computation

The butterfly operation of this algorithm can be implemented similar to that of DIT FFT. The value of 'A' can be stored where 'a' was stored and 'B' can be stored where 'b' was stored. This is called inplace computation. $A - a$ and $B - b$ are complex valued. Hence two memory locations are required to store each of them. Therefore the memory locations required for in place computation of one butterfly are [see equation 5.5.31],

$$\text{Memory locations for one butterfly} = 2 \times 2 = 4 \quad \dots (5.5.52)$$

The computations are performed stagewise. There are $\frac{N}{2}$ butterflies in one stage. Hence memory requirement for computation of N-point DFT becomes [see equation 5.5.33],

$$\text{Memory requirement of N-point DFT} = 4 \times \frac{N}{2} = 2N \quad \dots (5.5.53)$$

Thus the memory requirement of DIF FFT is same as that of DIT FFT. The maximum memory requirement including storage of twiddle factors will be $2N + \frac{N}{2}$.

5.5.2.4 Bit Reversal

In the signal flow graph of Fig. 5.5.12 observe that the input sequence is in natural order but output DFT sequence is in bit reversed order. In DIT FFT, the input sequence is in bit reversed order. Hence to get DFT in natural order it should be read in bit reversed order. The common bit reversal algorithm can be developed. This algorithm can be used to reshuffle the input sequence in bit reversed order. Such algorithm can be used for DIF as well as DIT algorithms.

5.6 Goertzel Algorithm

We know that the FFT algorithms take N-data points of the sequence $x(n)$ and compute N-points of DFT. This requires $\frac{N}{2} \log_2 N$ number of complex multiplications and $N \log_2 N$ complex additions. In some applications the DFT is to be computed only at selected values of

k (i.e. frequencies). When these selected values are less than $\log_2 N$, then direct computation of DFT becomes more efficient than FFT. This direct computation of DFT for selected values of frequencies can be realized through linear filtering of $x(n)$. Such linear filtering for computation of DFT can be implemented using Goertzel algorithm, which is discussed below.

Consider the factor W_N^{-kN} ,

$$\begin{aligned} W_N^{-kN} &= e^{-j\frac{2\pi}{N} \cdot (-kN)} \quad \text{since } W_N = e^{-j\frac{2\pi}{N}} \\ &= e^{j2\pi} \\ &= \cos 2\pi + j \sin 2\pi = 1 + j 0 \\ &= 1 \quad \text{always} \end{aligned} \quad \dots (5.6.1)$$

By definition, N-point DFT is given as,

$$X(k) = \sum_{m=0}^{N-1} x(m) W_N^{km}$$

If we multiply RHS of above equation by W_N^{-kN} , the value of $X(k)$ will be unaffected since $W_N^{-kN} = 1$ i.e.,

$$\begin{aligned} X(k) &= \sum_{m=0}^{N-1} x(m) W_N^{km} \cdot W_N^{-kN} \\ &= \sum_{m=0}^{N-1} x(m) W_N^{-k(N-m)} \end{aligned} \quad \dots (5.6.2)$$

Let us consider the LTI system shown in Fig. 5.6.1 which has input $x(n)$. Let the unit sample response of this system be given as,

$$h_k(n) = W_N^{-kn} u(n) \quad \dots (5.6.3)$$

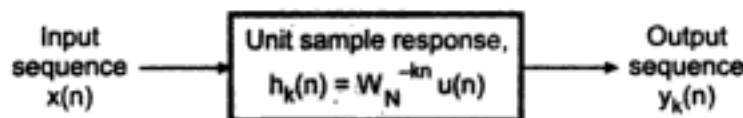


Fig. 5.6.1 The LTI system with unit sample response $h_k(n) = W_N^{-kn} u(n)$ and input $x(n)$

Here ' k ' is fixed (constant) for this system. We know that linear convolution of $x(n)$ and $h_k(n)$ gives the output sequence $y_k(n)$ of LTI system. i.e.,

$$y_k(n) = \sum_{m=-\infty}^{\infty} x(m) h_k(n-m) \quad \dots (5.6.4)$$

Putting for $h_k(n)$ in above equation from equation 5.6.3 we get,

$$y_k(n) = \sum_{m=-\infty}^{\infty} x(m) W_N^{-k(n-m)} u(n-m)$$

In this equation $x(m)$ is given for $0 \leq m \leq N-1$ for N values, hence limits on summation will be changed as follows,

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

5.11 Computation of Fourier Transform

We know that fourier transform of the discrete time signal $x(n)$ is given as,

$$X(\omega) = \sum_{N=-\infty}^{\infty} x(n) e^{-j\omega n} \quad \dots (5.11.1)$$

Let the sequence be causal having 'N' samples. Then above equation can be written as,

$$X(\omega) = \sum_{n=0}^{N-1} x(n) e^{-j\omega n} \quad \dots (5.11.2)$$

We know that $e^{-j\omega n} = \cos(\omega n) - j \sin(\omega n)$. Hence above equation can be written as,

$$\begin{aligned} X(\omega) &= \sum_{n=0}^{N-1} x(n) [\cos(\omega n) - j \sin(\omega n)] \\ &= \sum_{n=0}^{N-1} x(n) \cos(\omega n) - j \sum_{n=0}^{N-1} x(n) \sin(\omega n) \end{aligned} \quad \dots (5.11.3)$$

Thus $X(\omega)$ has real and imaginary parts. The real part of $X(\omega)$ is given as,

$$\text{Re} [X(\omega)] = \sum_{n=0}^{N-1} x(n) \cos(\omega n) \quad \dots (5.11.4)$$

And imaginary part of $X(\omega)$ is given as,

$$\text{Im} [X(\omega)] = - \sum_{n=0}^{N-1} x(n) \sin(\omega n) \quad \dots (5.11.5)$$

The 'C' program for computation of fourier transform is given below :

```
//file name : fourier.cpp
/*- Fourier Transform of the sequence and computation of transfer function -*/
//
// This program computes the fourier transform of the sequence x(n)
// and plots its magnitude and phase transfer function
// characteristics on the screen.
//
// Inputs : 1. Number of samples of x(n)
//          2. Values of samples of x(n)
//          3. Frequency w at which fourier transform
//             is to be evaluated
//
// Outputs : 1. X(w) at given value is displayed.
//           2. Magnitude and Phase plots for 0 to pi
//              values of w are displayed on screen.
//
// Assumptions : 1. This program is for written real
//                 values of sequence x(n)
//                 2. Magnitude and phase transfer function
//                    plots are computed for x(n) for 0 to pi
//                    values of w.
//-----
```


Hidden page

```

//calculation of magnitude of transfer function
phase[k] = atan2(XwImag,XwReal);
} //calculation of phase of transfer function

// Next part of program displays the magnitude & phase on the screen
// which is calculated at 640 points by previous 'for' loop

detectgraph(&gd, &gm); // detect graphics mode & graphics driver
initgraph(&gd, &gm, ""); // initialize screen in graphics mode
setlinestyle(DOTTED_LINE, 1, 1); // dotted line style for x-axis
line(0, 250, 640, 250); // x-axis line for magnitude
line(0, 350, 640, 350); // x-axis line for phase
for(k = 0; k < 640; k++) // this loop puts 640 pixels of
{ // magnitude and phase on the screen
yMag = 250 - mag[k]*200; //magnitude scaled for proper display
putpixel(k, yMag, YELLOW); //put pixel of magnitude on the screen
yPhase = 350 - phase[k]*50; // scaling of phase
putpixel(k, yPhase, GREEN); // put pixel of phase
}
outtextxy(500, 260, "Magnitude plot"); // label of magnitude
outtextxy(500, 450, "Phase plot"); // label of phase
getch(); // wait for observation
closegraph();
}
//----- End of program -----

```

The above program first accepts the total number of samples 'N' and their values. The first for loop performs this function. Next the fourier transform is computed at particular value of frequency ' ω '. This frequency is to be entered by the user (between 0 to π).

$$XwReal = XwReal + x[n] * \cos((w*n));$$

$$XwImag = XwImag + x[n] * \sin(w*n);$$

The first statement computes real part of $X(\omega)$ according to equation 5.11.4. The second statement computes imaginary part of $X(\omega)$ according to equation 5.11.5. $XwImage$ is made negative as required by equation 5.11.5 after the for loop i.e.,

$$XwImag = XwImag * (-1.0);$$

Then the program prints value of $X(\omega)$ at given value of ' ω ' as

$$X(\omega) = \text{Real part} + j \text{Imaginary part}$$

The next part of the program computes magnitude and phase of $X(\omega)$ in the range of 0 to π . $X(\omega)$ is continuous function of ' ω '. But magnitude and phase of $X(\omega)$ is computed at 640 points in 0 to π . Hence ' ω ' is increased in steps of $\frac{\pi}{640}$.

$$wStep = \pi/640.0;$$

This statement computes the incremental step of ' ω '.

Next there is for loop which computes magnitude and phase of $X(\omega)$ for 640 values of ' ω ' in the range of 0 to π . The array $mag[k]$ stores magnitude of $X(\omega)$ at k^{th} value of ω . Similarly the array $phase[k]$ stores phase of $X(\omega)$ at k^{th} value of ω . The two statements,

$$mag[k] = \sqrt{XwReal * XwReal + XwImag * XwImag}$$

$$phase[k] = \text{atan2}(XwImag, XwReal)$$

The first statement computes magnitude of $X(\omega)$ at $\omega_k = \frac{\pi}{640}k$ i.e. k^{th} value of ω . The second statement computes phase of $X(\omega)$ at $\omega_k = \frac{\pi}{640}k$ i.e. k^{th} value of ω .

The next part of the program displays magnitude and phase on the screen. Graphics mode initializes screen in 640×480 pixels. Thus we can plot 640 pixels on the screen. The last for loop displays these pixels. Magnitude and phase both are computed at 640 points. These 640 points are displayed as pixels on the screen. The pixels are displayed after proper scaling.

An example to test this program :

Refer to example 5.2.6. In this example we have computed fourier transform for $h(n)$ which is given as,

$$h(n) = \{0.5, 0.5\}$$

The magnitude and phase plots of fourier transform of $h(n)$ i.e. $H(\omega)$ are given in Fig.5.2-1 for $-\pi \leq \omega \leq \pi$

Let us consider the same example to test this program. Here we will call $h(n)$ as $x(n)$. This is just notation. Hence enter the number of samples as 2 and,

$$x(0) = 0.5$$

$$x(1) = 0.5$$

Let us compute fourier transform at $\omega = \frac{2\pi}{3} = 2.0943951$.

The results of the program for these inputs are given below.

```
//----- Results -----
Fourier Transform and computation of transfer function
Enter the number samples in the sequence x(n) 2
Enter the samples of sequence x(n)
x(0) = 0.5
x(1) = 0.5
Enter the frequency w at which fourier transform is
to be evaluated w(btween 0 to pi) = 2.0943951
The value of fourier transform is
X(w = 2.094395) = 0.250000 + j(-0.433013)
press any key to see magnitude and phase transfer function plots
```

As shown in above results,

$$X(w = 2.094395) = 0.250000 + j(-0.433013)$$

From equation 5.2.32 in example 5.2.6 we have obtained expression for $H(\omega)$, [here $X(\omega)$] as,

$$X(\omega) = \frac{1}{2}(1 + \cos \omega) - j \frac{1}{2} \sin \omega$$

At $\omega = 2.094395$ above equation becomes,

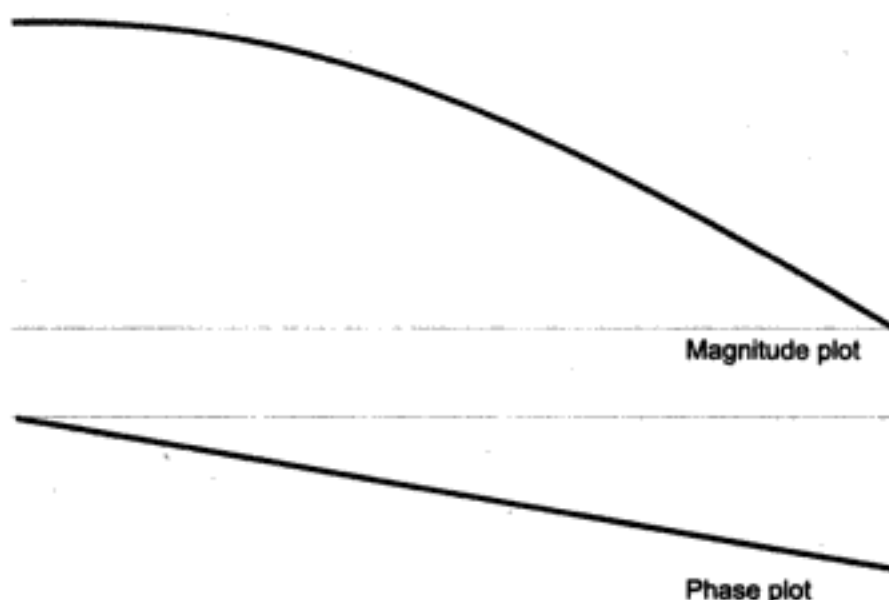


Fig. 5.11.1 Magnitude and phase plot of $X(\omega)$ displayed on the screen for $x(n) = \{0.5, 0j\}$ at 640 points in 0 to π

$$X\left(\omega = \frac{2\pi}{3} = 2.094395\right) = 0.25 - j 0.4330127$$

This value is same as that computed by the program.

Fig. 5.11.1 shows the magnitude and phase plot of $X(\omega)$ displayed on the screen. Observe that these plots are successive display of 640 pixels on the screen. The curves look like continuous plots on the screen. The plots of Fig. 5.11.1 are shown for 0 to π . Observe that these plots are similar to those shown in Fig. 5.2.1 earlier. The plots of Fig. 5.2.1 are for $-\pi$ to π .

3.12 Computation of DIT FFT Algorithm

Here let us see the implementation of radix-2 DIT FFT algorithm. The approach is simplified and based on the signal flow graph. A 'C' program is given below. This program computes the DFT of the complex valued sequence of length 'N'. Note that $N = 2^v$ and v is some integer.

```
//file name : ditfft.cpp
/*----- Radix-2 DIT FFT algorithm -----*/
//
// This program implements the radix-2 DIT FFT algorithm for
// the complex valued sequence of length N
//
// Inputs : 1. Number of samples of x(n)
//          2. Values of samples of x(n)
//
// Outputs : real and imaginary part of
//           DFT X(k).
//
// Assumptions : This program is for written for
//               inplace computation. The data storage
```

Hidden page

```

for(i = 1; i <= 2*N;)
{
    // this loop computes N/2 butterflies in one
    // stage of decimation
    // following four statements compute real and imaginary parts
    // one butterfly
    real_k0 = data[i]+cosTheta*data[i+p]+sinTheta*data[i+1+p];
    imag_k0 = data[i+1]+cosTheta*data[i+1+p]-sinTheta*data[i+p];
    real_k1 = data[i]-cosTheta*data[i+p]-sinTheta*data[i+1+p];
    imag_k1 = data[i+1]-cosTheta*data[i+1+p]+sinTheta*data[i+p];

    data[i] = real_k0; // | the newly computed real and
    data[i+1] = imag_k0; // | imaginary parts of the
    data[i+p] = real_k1; // | butterfly are stored in
    data[i+p+1] = imag_k1; // | place of previous values
    // | this is "in place computation"

    if(n<k)
    {
        // |
        i = i + 2; n++; // |
        t = t + 1; // |
        Theta = ((2*pi)/(float)p)*t; // | this if-else loop
        cosTheta = cos(Theta); // | adjusts the
        sinTheta = sin(Theta); // | counters and
    } else // | phase factors
    { // | as required
        // | in one stage
        i = i + p+2; // | of decimation
        n = 1; t = 0; // |
        Theta = ((2*pi)/(float)p)*t; // |
        cosTheta = cos(Theta); // |
        sinTheta = sin(Theta); // |
    }
}
k = k << 1; // adjust counter for next stage of decimation
m++;
}
//----- DIT FFT computation completes -----

//----- Next part displays computed DFT on the screen -----

printf("\nThe DFT is as follows...\n");
printf("\nreal part of X(k)      imaginar part of X(k)");
for(n = 1; n <= N; n++)
{
    printf("\n%f \t\t %f",data[2*n-1],data[2*n]);
}
}
//----- End of program -----

```

The first part of the above program accepts the complex valued sequence $x(n)$. The variable N represents the number of samples of the sequence. The array `data[2050]` is used to store upto 1024 samples of input sequence $x(n)$. Here note that the length of the array `data` can be increased or reduced. The sequence $x(n)$ is stored in array `data` as follows :

Hidden page

Hidden page

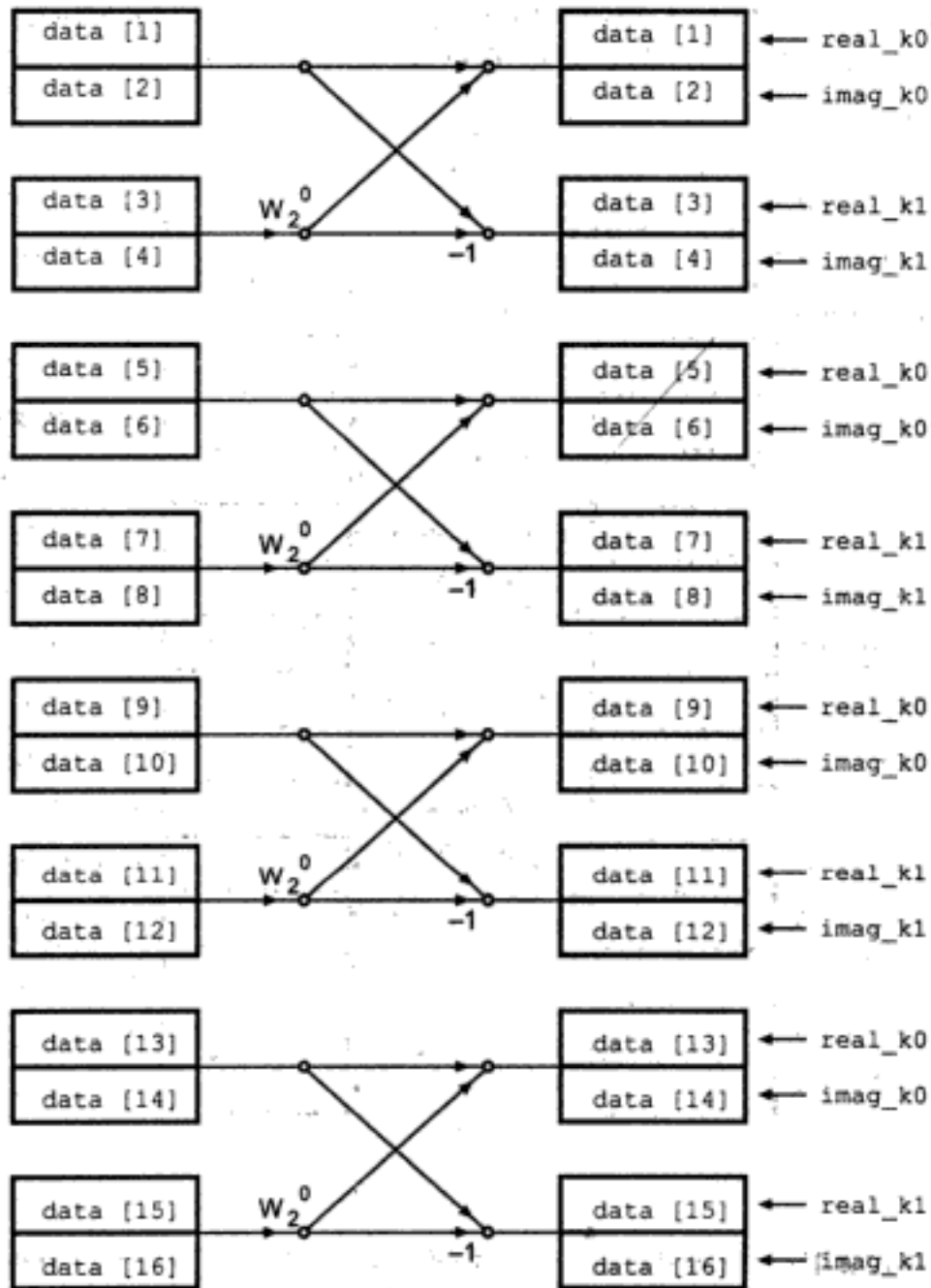


Fig. 5.12.3 First stage of computations. The results are stored in the same array

The computations of various butterflies as shown above are performed repeatedly according to the following process. Fig. 5.12.4 shows the generalized butterfly in any stage of decimation/ computation.

Please refer Fig. 5.12.4 on next page.

The computations in above butterfly are performed as follows :

Let
$$\text{Theta} = \frac{2\pi}{P} \cdot t \quad \dots (5.12.1)$$

Hidden page

These are the statements used in the program. The `for` loop thus computes values of one butterfly. There are such $\frac{N}{2}$ butterflies in every stage.

In the same `for` loop, there is `if-else` statements. These statements adjust the counters of array and phase factors $e^{-j\frac{2\pi}{P} \cdot t}$ as required for computation of generalized butterfly in any stage.

The last part of the program displays computed DFT on the screen.

To test the program :

Now let us check this program. Consider the example 5.3.1. In this example, sequence $x(n)$ is given as,

$$x(n) = \{0, 1, 2, 3\}$$

The DFT of this sequence is obtained in the example as,

$$X(k) = \{6, -2+2j, -2, -2-2j\}$$

The sequence to be entered to the program as,

$$x(0) = 0 + j 0 \text{ i.e. } 0 \ 0$$

$$x(1) = 1 + j 0 \text{ i.e. } 1 \ 0$$

$$x(2) = 2 + j 0 \text{ i.e. } 2 \ 0$$

$$x(3) = 3 + j 0 \text{ i.e. } 3 \ 0$$

And the expected DFT is,

$$x(0) = 6 + j 0 \quad \text{i.e. } 6 \ 0$$

$$x(1) = -2 + 2j \quad \text{i.e. } -2 \ 2$$

$$x(2) = -2 + j 0 \quad \text{i.e. } -2 \ 0$$

$$x(3) = -2 - 2j \quad \text{i.e. } -2 \ -2$$

The results obtained by the program are presented below :

```
//----- results -----
Radix-2 DIT FFT algorithm
```

Enter the number of samples in the sequence $x(n)$, $N = 4$

Enter the samples of sequence $x(n)$

real part imaginary part

$x(0) = 0 \ 0$

$x(1) = 1 \ 0$

$x(2) = 2 \ 0$

$x(3) = 3 \ 0$

The DFT is as follows...

real part of X(k)	imaginar part of X(k)
6.000000	0.000000
-1.998736	2.000000
-2.000000	0.000000
-2.001264	-2.000000

Thus the program provides the same results.

5.13 Inverse DFT and Computation of IDFT using FFT Algorithms

From equation 5.3.2 we know that DFT of the sequence $x(n)$ is given as,

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}, \quad k = 0, 1, 2, \dots, N-1 \quad \dots (5.13.1)$$

And from equation 5.3.3 IDFT is given as,

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi kn/N}, \quad n = 0, 1, 2, \dots, N-1 \quad \dots (5.13.2)$$

In the above equation observe that terms under summation are similar to that of DFT except the sign of exponential term. Hence it is possible to use FFT algorithms for the computation of IDFT. For example if we want to use DIT FFT algorithm for computation of IDFT, we have to arrange $X(k)$ in bit reversed order. Then interchange the 'signs' of phase factors. The result obtained by the algorithm is divided by 'N' to get sequence $x(n)$.

For example let us see what will be the modifications in the program presented in last section to compute IDFT. The phase factor of equation 5.12.2 changes the sign. Hence for computation of IDFT equation 5.12.2 will be,

$$e^{j\Theta} = \cos(\Theta) + j \sin(\Theta) \quad \dots (5.13.3)$$

Equation 5.12.3 and equation 5.12.4 are then solved accordingly and the statements of 'C' are modified. Lastly every sample of output sequence should be divided by 'N' to get sequence $x(n)$. Thus same program can be used to compute DFT and IDFT with manipulating 'signs'.

Such modified 'C' program is given below.

```
//file name : idft.cpp
/*----- Radix-2 DIT FFT algorithm to compute IDFT -----*/
//
// This program implements the radix-2 DIT FFT algorithm for
// the complex valued sequence of length N to compute IDFT
//
// Inputs : 1. Number of samples of X(k)
//          2. Values of samples of X(k)
//
// Outputs : real and imaginary part of
//           x(n).
//
// Assumptions : This program is for written for
//               inplace computation. The data storage
//               format is given in the discription.
//-----
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define SWAP(a,b) tempr = (a); (a) = (b); (b) = tempr;
void main()
{
    float data[2050], real_k0, imag_k0, real_k1, imag_k1, tempr;
    float pi, cosTheta, sinTheta, t, Theta;
    unsigned long N, n, m, j, i, k, p;
```

Hidden page

```

data[i] = real_k0; // | the newly computed real and
data[i+1] = imag_k0; // | imaginary parts of the
data[i+p] = real_k1; // | butterfly are stored in
data[i+p+1] = imag_k1; // | place of previous values
// | this is "in place computation"

if(n<k)
{
// |
i = i + 2; n++; // |
t = t + 1; // |
Theta = ((2*pi)/(float)p)*t; // | this if-else loop
cosTheta = cos(Theta); // | adjusts the
sinTheta = sin(Theta); // | counters and
} else // | phase factors
{ // | as required
i = i + p+2; // | in one stage
n = 1; t = 0; // | of decimation
Theta = ((2*pi)/(float)p)*t; // |
cosTheta = cos(Theta); // |
sinTheta = sin(Theta); // |
}
k = k << 1; // adjust counter for next stage of decimation
m++;
}
//----- DIT FFT computation completes -----
//----- Next part displays computed x(n) on the screen -----

printf("\nThe sequence x(n) is as follows...\n");
printf("\nreal part of x(n)      imaginary part of x(n)");
for(n = 1; n <= N; n++)
{
printf("\n%f \t\t %f", data[2*n-1]/N, data[2*n]/N);
} // each sample is divided by N since it is IDFT
}
//-----End of program -----

```

Observe that the program given above is exactly similar to that used for computation of DFT. In the while loop there is for loop. In this loop there are four statements to compute $real_k0$, $imag_k0$ and $imag_k1$. In these statements 'signs' of $sinTheta$ are made opposite to that for DFT.

And lastly while printing, the samples in data are divided by N. These are the only two changes made in the earlier program to compute IDFT.

This program is tested with the DFT $X(k)$ obtained by earlier program. The results are presented below.

```

----- results -----
Radix-2 DIT FFT algorithm to compute IDFT

Enter the number of samples in the sequence X(k), N = 4

Enter the samples of sequence X(k)
real part imaginary part
X(0) = 6 0
X(1) = -2 2
X(2) = -2 0

```

$X(3) = -2 -2$

The sequence $x(n)$ is as follows...

real part of $x(n)$	imaginary part of $x(n)$
0.000000	0.000000
1.000000	-0.000632
2.000000	0.000000
3.000000	0.000632

Observe that the sequence $x(n) = \{0, 1, 2, 3\}$ is obtained back by the program. This shows that IDFT can be computed using FFT algorithms.

Note : A single program can be used to compute DFT as well as IDFT. Based on whether DFT or IDFT, the 'signs' of $\sin\theta$ can be changed. This is very easily possible.

5.14 Computation of DFT and IDFT

In this section we will study the computation of DFT and IDFT using their basic definition. In the previous sections we have seen the computation of DFT and IDFT using FFT algorithms

5.14.1 Logic for Computation of DFT

The N-point DFT of the sequence $x(n)$ is given as,

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}, \quad k = 0, 1, \dots, N-1 \quad \dots(5.14.1)$$

We know that $e^{-j\theta} = \cos\theta - j \sin\theta$. Hence equation (5.14.1) becomes,

$$X(k) = \sum_{n=0}^{N-1} x(n) \left[\cos\left(\frac{2\pi kn}{N}\right) - j \sin\left(\frac{2\pi kn}{N}\right) \right]$$

The real and imaginary parts of $X(k)$ become,

$$X_{\text{Real}}[k] = \sum_{n=0}^{N-1} x(n) \cos\left(\frac{2\pi kn}{N}\right) \quad \text{and} \quad \dots(5.14.2)$$

$$X_{\text{Imag}}[k] = - \sum_{n=0}^{N-1} x(n) \sin\left(\frac{2\pi kn}{N}\right) \quad \dots(5.14.3)$$

Here $X_{\text{Real}}[k]$ is real part of $X(k)$ and $X_{\text{Imag}}[k]$ is imaginary part of $X(k)$. In equation 5.14.2 and equation 5.14.3 $x(n)$ is considered real sequence.

5.14.2 C Program for Computation of DFT

The C program based on logic discussed in previous subsection is given below.

```
//file name : dft.cpp
/*----- Discrete Fourier Transform(DFT) -----*/
//
// This program computes the N point DFT of the sequence x(n)
// of length N.
//
// Inputs : 1. Number of samples of x(n), i.e. N
//          2. Values of samples of x(n)
//
// Outputs : N point DFT X(k) with its real
//          real and imaginary parts.
```

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

```

h[2] = 2
h[3] = 1
Enter the sequence x(n)
x[0] = 1
x[1] = 2
x[2] = 3
x[3] = 4

The result of convolution is...
y[0] = 14.000000
y[1] = 16.000000
y[2] = 14.000000
y[3] = 16.000000

```

As shown above, the same sequence of $\{14, 16, 14, 16\}$ is obtained by the program..

5.16 Computation of Circular Convolution using DFT and IDFT

We know that circular convolution satisfies the following property from equation 5.3.48,

$$x_1(n) \text{ N } x_2(n) \xleftrightarrow[N]{DFT} X_1(k)X_2(k) \quad \dots (5.16.1)$$

$$\text{or } h(n) \text{ N } x(n) \xleftrightarrow[N]{DFT} H(k)X(k) \quad \dots (5.16.2)$$

This means multiplication of two DFTs is equivalent to the circular convolution of their time domain sequences. This provides another method for computation of circular convolution. That is, first taken DFTs of two sequences. Then multiply these DFTs on sample to sample basis i.e.,

$$Y(k) = H(k) \cdot X(k) \quad \dots (5.16.3)$$

Then take IDFT of $Y(k)$ i.e.

$$y(n) = \text{IDFT}\{Y(k)\}$$

According to equation 3.11.2, $Y(n)$ is circular convolution of $h(n)$ and $x(n)$. i.e.

$$y(n) = h(n) \text{ N } x(n) \quad \dots (5.16.4)$$

5.16.1 Logic for Computation

Let the two DFT pairs be as follows

$$h(n) \text{ N } \xleftrightarrow[N]{DFT} H(k)$$

$$x(n) \text{ N } \xleftrightarrow[N]{DFT} X(k)$$

And the real and imaginary parts of $H(k)$ $X(k)$ be given as,

$$H(k) = \text{HReal}[k] + j \text{HImag}[k]$$

$$X(k) = \text{XReal}[k] + j \text{XImag}[k]$$

Then the multiplication of two DFTs will be,

$$\begin{aligned}
Y(k) &= H(k) \cdot X(k) \\
&= \{\text{HReal}[k] + j \text{HImag}[k]\} \{\text{XReal}[k] + j \text{XImag}[k]\}
\end{aligned}$$

Hidden page

Hidden page

The real and imaginary parts of $Y(k)$ are obtained according to equation 3.16.5 and equation 5.16.6. Then the inverse DFT of $Y(k)$ is computed according to equation 5.14.5. This is because sequence $Y(n)$ will be real if $h(n)$ and $x(n)$ are real. The last for loop prints $y(n)$ on the screen.

To Test the Program :

Let us test this program for the circular convolution of $\{2, 1, 2, 1\}$ and $\{1, 2, 3, 4\}$. In example 5.3.5 we have obtained circular convolution of these sequences as $\{14, 16, 14, 16\}$.

When these sequences are given as input to the program, the results are shown below.

```
----- Results -----
Circular Convolution using DFT and IDFT

Enter the value of N = 4
Enter the sequence h(n)
h[0] = 2
h[1] = 1
h[2] = 2
h[3] = 1
Enter the sequence x(n)
x[0] = 1
x[1] = 2
x[2] = 3
x[3] = 4

The circular of convolution is...
y[0] = 14.003798
y[1] = 15.998674
y[2] = 13.996202
y[3] = 16.001299
-----
```

These results are same as one we have obtained earlier.

5.17 Computation of Magnitude and Phase Transfer Function Plots

In this section we will discuss about how to compute magnitude and phase of the transfer function.

5.17.1 Logic for Computation and Magnitude and Phase

We know that the difference equation is given as,

$$y(n) = -[a_1 y(n-1) + a_2 y(n-2) + \dots + a_N y(n-N)] + b_0 x(n) + b_1 x(n-1) + \dots + b_{M-1} x(n-M+1) \dots \quad (5.17.1)$$

This difference equation has 'N' number of $y(n)$ terms and 'M' number of $x(n)$ terms. Taking z-transform of above equation we get,

$$Y(z) = -[a_1 z^{-1} Y(z) + a_2 z^{-2} Y(z) + \dots + a_N z^{-N} Y(z)] + b_0 X(z) + b_1 z^{-1} X(z) + \dots + b_{M-1} z^{-M+1} X(z)$$

On rearranging above equation the ratio $\frac{Y(z)}{X(z)}$ called system function $H(z)$ becomes,

Hidden page

Thus $wstep = \frac{\pi \text{ (i.e. } p)}{640}$ is the angle with which ' ω ' is increased to get range of 0 to π

Thus we get 640 values of $|H(\omega)|$ and $\angle H(\omega)$. These values are then displayed on the screen.

5.17.2 'C' Program for Magnitude and Phase Transfer Function Plot

The 'C' program based on above logic is given below :

```

//file name : magphase.cpp
/* Magnitude and phase transfer function plots of a difference equation */
//
// This program accepts the coefficients of the difference
// equation and generates the magnitude and phase transfer
// function plots.
//
// The format of the difference equation is,
//  $y(n) = -[a_1*y(n-1)+a_2*y(n-2)+a_3*y(n-3)+\dots]$ 
//  $+b_0*x(n)+b_1*x(n-1)+b_2*x(n-2)+\dots$ 
//
// Inputs : 1. Number of coefficients of  $x(n)$  i.e. M.
//          2. Values of coefficients of  $x(n)$ ,
//             i.e.  $b_0, b_1, b_2, \dots$  etc.
//          3. Number of coefficients of  $y(n)$  i.e. N.
//          4. Values of coefficients of  $y(n)$ ,
//             i.e.  $a_1, a_2, a_3, \dots$  etc.
//
// Outputs : Magnitude and phase transfer function
//            plot  $H(e^{j\omega})$  for  $\omega = 0$  to  $\pi$ .
//
// Assumptions : This program is written for up to ten coefficients.
//-----
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
void main()
{
    float RealNum, ImagNum, RealDen, ImagDen;
    float mag[640], phase[640], pi, w, wStep;
    float static a[10], b[10], yMag, yPhase;
    int M, N, i, k, gd, gm;

    clrscr();
    printf(" \nThis program displays the magnitude and phase"
           " transfer function plots\nfrom given coefficients of"
           " difference equation \n\n");
    printf("\nenter the number of coefficients of x(n), M = ");
    scanf("%d", &M);
    for(i = 0; i < M; i++)
    {
        // coefficients of x(n)
        printf("b%d = ", i);
        scanf("%f", &b[i]); // b0, b1, b2, ....
    }
    printf("\nenter the number of coefficients of y(n), N = ");
    scanf("%d", &N);
    a[0] = 1.0;
    for(i = 1; i <= N; i++)

```

```

    { // coefficients of y(n)
        printf("a%d = ", i);
        scanf("%f", &a[i]); // a1, a2, a3, .....
    }
    pi = 22.0/7.0; // value of pi
    wStep = pi/640.0; // range of w = 0 to pi divided into small
// steps of pi/640 for displaying on screen.
    for(k = 0; k < 640; k++)
    { // this loop computes magnitude and phase at 640 points
// in the range of w = 0 to pi.
        w = w + wStep; // steps by which w is increased.
        RealNum = b[0]; // real part of numerator
        ImagNum = 0; // imaginary part of numerator
        for(i = 1; i < M; i++) // this loop computes total real &
        { // imaginary parts of numerator seperately.
            RealNum = RealNum + b[i]*cos(i*w); //total real part.
            ImagNum = ImagNum + b[i]*sin(i*w); //total imaginary part.
        }
        ImagNum = ImagNum * (-1.0); // change sign of imaginary part.
        RealDen = a[0]; // real part of denominator
        ImagDen = 0; // imaginary part of denominator
        for(i = 1; i <= N; i++) // this loop computes total real &
        { // imaginary parts of denominator seperately.
            RealDen = RealDen + a[i]*cos(i*w); // total real part
            ImagDen = ImagDen + a[i]*sin(i*w); // total imaginary part
        }
        ImagDen = ImagDen * (-1.0); // change sign of imaginary part
        mag[k] = sqrt(RealNum*RealNum + ImagNum*ImagNum) /
            sqrt(RealDen*RealDen + ImagDen*ImagDen);
            //calculation of magnitude of transfer function
        phase[k] = atan2(ImagNum, RealNum) - atan2(ImagDen, RealDen);
    } //calculation of phase of transfer function

// Next part of program displays the magnitude & phase on the screen
// which is calculated at 640 points by previous 'for' loop

    detectgraph(&gd, &gm); // detect graphics mode & graphics driver
    initgraph(&gd, &gm, ""); // initialize screen in graphics mode
    setlinestyle(DOTTED_LINE, 1, 1); // dotted line style for x-axis
    line(0, 250, 640, 250); // x-axis line for magnitude
    line(0, 350, 640, 350); // x-axis line for phase
    for(k = 0; k < 640; k++) // this loop puts 640 pixels of
    { // magnitude and phase on the screen
        yMag = 250 - mag[k]*25; // magnitude scaled for proper display
        putpixel(k, yMag, WHITE); // put pixel of magnitude on the screen
        yPhase = 350 - phase[k]*50; // scaling of phase
        putpixel(k, yPhase, WHITE); // put pixel of phase
    }
    outtextxy(500, 200, "Magnitude plot"); // label of magnitude
    outtextxy(500, 450, "Phase plot"); // label of phase
    getch(); // wait for observation
    closegraph(); // close graphics mode
}
//----- End of program -----

```

As shown in above program the first for loop is used to enter the coefficients of $x(n)$. The second for loop is used to enter coefficients of $y(n)$. The statement

```
wstep = pi/640.0;
```

Computes the steps in which ' ω ' is incremented in the range of 0 to π . The next, i.e. 3rd for loop computes magnitude and phase at 640 discrete values in the range of 0 to π . In this for loop, there are two statements :

```
ImagNum = ImagNum * (-1.0);
```

and

```
Imagden = ImagDen * (-1.0);
```

These statements absorb the negative sign of ImagNum and ImagDen in equation 5.17.8.

The statement,

```
RealNum = RealNum + b[i] * cos (i * w);
```

implements equation 5.17.1. Similarly other equations are also implemented.

The arrays mag[k] stores magnitude of $H(\omega)$ of 640 points and phase[k] stores phase of $H(\omega)$ of 640 points.

The last part of the program then displays the samples of mag[k] and phase[k] arrays on the screen after proper scaling.

An example and results :

Consider the example of

$M = 1$ with $b_0 = 1$ and

$N = 1$ with $a_1 = -0.9$ and $a_0 = 1$ internally.

The difference equation of equation 5.17.1 becomes,

$$Y(n) = -[-0.9 Y(n-1)] + x(n)$$

Hence the system function $H(z)$ becomes,

$$y(z) = 0.9z^{-1} y(z) + X(z)$$

$$\therefore H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - 0.9z^{-1}}$$

Hence transfer function $H(\omega)$ becomes,

$$H(\omega) = H(z)|_{z=e^{j\omega}}$$

$$\therefore H(\omega) = \frac{1}{1 - 0.9e^{-j\omega}} \quad \dots (5.17.11)$$

The calculation of magnitude and phase of $H(\omega)$ of above equation is left as an exercise to the readers. The results of the above 'C' program for this difference equation are presented next.

```
//----- results -----
```

```
This program displays the magnitude and phase transfer function plots
from given coefficients of difference equation
```

```
enter the number of coefficients of x(n), M = 1
b0 = 1
```

```
enter the number of coefficients of y(n), N = 1
a1 = -0.9
```

```
-----
```

Hidden page

7. With the help of $N = 8$, explain radix-2 DIF FFT algorithm for computation of DFT.
8. Write short notes on the following :
 - (i) Goertzel algorithm.
 - (ii) Computational complexity of FFT algorithms.
 - (iii) In place computations.
 - (iv) Butterfly operation
 - (v) Overlap-save and overlap-add algorithms.
 - (vi) Use of DFT for power spectrum estimation.
 - (vii) Chirp-z transform Algorithm.
9. Explain the relationships between z-transform and DFT and fourier transform.

Unsolved Examples

1. Compute N-point DFT of following signals :

(i) $x(n) = \delta(n - n_0)$

(Ans. : $X(k) = e^{-j2\pi kn_0/N}$)

(ii) $x(n) = \begin{cases} 1, & 0 \leq n \leq \frac{N}{2} - 1 \\ 0, & \frac{N}{2} \leq n \leq N - 1 \end{cases}$

(Ans. : $X(k) = \frac{1 - (-1)^k}{1 - W_N^k}$)

2. Compute the circular convolution of the following sequences.

(i) $x_1(n) = \{1, 1, 1, 1, 0, 0, 0, 0\}$

$x_2(n) = \sin \frac{3\pi}{8} n, \quad 0 \leq n \leq 7$

(Ans. : $\{1.25, 2.55, 2.55, 1.25, 0.25, -1.06, -1.06, 0.25\}$)

(ii) $x(n) = \{1, 3, 5, 3\}$
 $h(n) = \{2, 3, 1, 1\}$

(Ans. : $\{19, 17, 23, 25\}$)

□□□

Chapter 6

FILTER DESIGN

6.1 Introduction

The digital filters are the discrete time systems used mainly for filtering of arrays. The arrays or sequences are obtained by sampling the input analog signals. The digital filters perform the frequency related operations such as low pass, high pass, band reject (notch), bandpass and all pass etc. The design specifications include cut off frequency, sampling frequency of input signal, pass band variation, stop band attenuation, approximation, type of filter and realization form etc. Digital filters can be realized through hardware or software. Actually speaking, the software digital filters need digital hardware for their operation.

6.2 Difference Between Analog Filters and Digital Filters

All of us are familiar with analog filters that uses inductors, capacitors and resistors.

6.2.1 An Example of Analog Low Pass Filter

Consider the LC lowpass filter shown in Fig. 6.2.1. This is a very standard and commonly used filter to remove ripple and noise in the input signal.

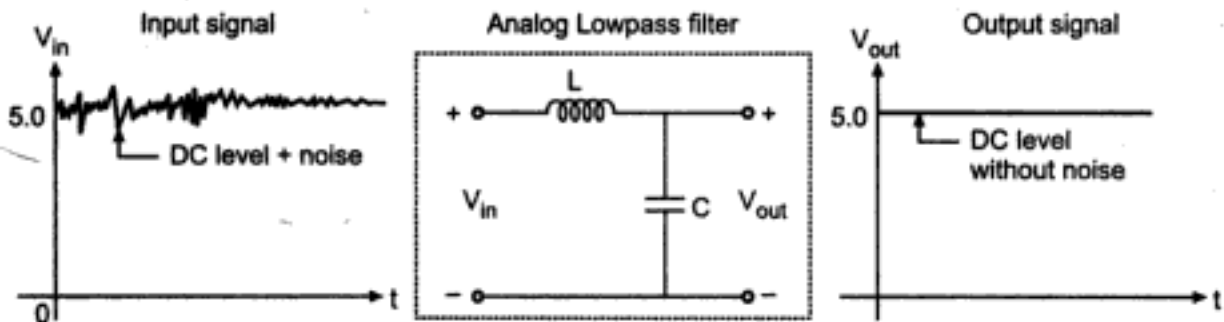


Fig. 6.2.1 Analog lowpass filter

In the above figure observe that input signal is noisy DC level of 5 volts. The noise is removed by the LC filter and the output signal is pure DC level of 5 volts. The noise is high frequency signal compared to DC level. The lowpass filter removes this noise and passes only DC level. This is how the simple LC lowpass filter works.

6.2.2 An Example of Digital Lowpass Filter

The digital filters do not use resistance capacitors or inductors. We know that the discrete time systems are represented by the difference equation. Similarly a digital filter is also represented by the difference equation. Consider a simple lowpass filter whose difference equation is given as,

$$y(n) = \frac{1}{2}x(n) + \frac{1}{2}x(n-1) \quad \dots (6.2.1)$$

Fig. 6.2.2 shows the implementation of this digital lowpass filter. The noisy analog 5V signal is sampled and we get $x(n)$. Observe that the samples of $x(n)$ vary above and below 5V level. We have to remove this noise and all the samples at the output should be near 5V level. The noisy $x(n)$ is given to the digital lowpass filter as shown in figure. Observe that the digital lowpass filter is represented by the difference equation $y(n) = \frac{1}{2}x(n) + \frac{1}{2}x(n-1)$. The digital filter then computes various values of $y(n)$ according to this difference equation. The above figure shows the computation of first few samples of $y(n)$. For initial calculation we need $x(-1)$. It is assumed to be 5.8. Observe the values of $y(n)$ in figure. The noise in these samples is reduced compared to $x(n)$. This is how digital filter works.

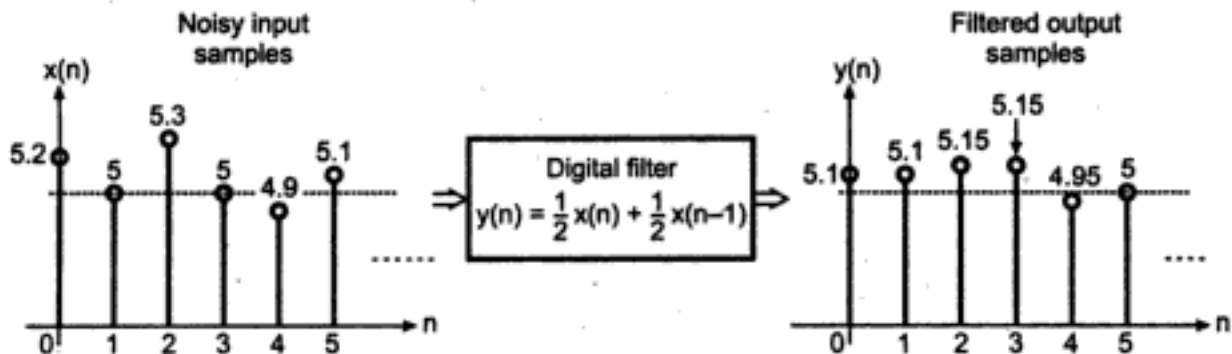


Fig. 6.2.2 Digital lowpass filter

Assume $x(-1) = 5$

Then output samples $y(n)$ for various values of 'n' can be obtained as follows by using $y(n) = \frac{1}{2}x(n) + \frac{1}{2}x(n-1)$.

$$y(0) = \frac{1}{2}x(0) + \frac{1}{2}x(-1) = \frac{1}{2}(5.2) + \frac{1}{2}(5) = 5.1$$

$$y(1) = \frac{1}{2}x(1) + \frac{1}{2}x(0) = \frac{1}{2}(5) + \frac{1}{2}(5.2) = 5.1$$

$$y(2) = \frac{1}{2}x(2) + \frac{1}{2}x(1) = \frac{1}{2}(5.3) + \frac{1}{2}(5) = 5.15$$

$$y(3) = \frac{1}{2}x(3) + \frac{1}{2}x(2) = \frac{1}{2}(5) + \frac{1}{2}(5.3) = 5.15$$

$$y(4) = \frac{1}{2}x(4) + \frac{1}{2}x(3) = \frac{1}{2}(4.9) + \frac{1}{2}(5) = 4.95$$

$$y(5) = \frac{1}{2}x(5) + \frac{1}{2}x(4) = \frac{1}{2}(5.1) + \frac{1}{2}(4.9) = 5 \quad \dots \text{ and so on.}$$

6.2.3 Implementation of Digital Filter

The difference equation of the digital filter described above can be implemented in software like 'C' or assembly language. Basically such languages are compiled and an executable code for the processor is prepared. This code runs on the memory, data bus, shift registers, counters, ALU etc to give required output. Thus digital filter is implemented by the digital hardware such as flipflops, counters, shift registers, ALU of the general purpose processor. The software code simply manipulates the operations of these digital circuits.

The digital filter can also be implemented by the dedicated hardware. Such hardware is basically a digital circuit consisting of counters, shift registers, flipflops, ALU etc. But the dedicated hardware can perform only one type of filtering operation, since it cannot be changed.

When the digital filters are written in softwares like 'C' or assembly, they are highly flexible. The single program is capable of performing different filtering operations. The major advantage is that these all programs use the same digital hardware. For example different programs in 'C' can be written for lowpass, bandpass, highpass, notch etc filters. And all these programs run on same computer.

Thus digital filters have large number of advantages over analog filters. The differences will be more clear when we will study design and implementation of digital filters.

6.2.4 Comparison of Analog and Digital Filters

Following Table 6.2.1 presents some of the differences (advantages/disadvantages) between analog and digital filters.

Table 6.2.1 Comparison/advantages and disadvantages of analog and digital filters

Sr.No.	Parameter	Analog filter	Digital filter
1.	Input/output signals	Analog	Digital (discrete time sequences)
2.	Composition	Lumped elements such as R, L and C or analog IC.	Software + digital hardware
3.	Filter representation	In terms of system components	By difference equation
4.	Flexibility	Not flexible	Highly flexible
5.	Portability	Not easily portable	Portable
6.	Design objective and result	Specifications to values of R, L and C components	Specifications to difference equation
7.	Environmental effects	Environmental parameters affects the performance	Negligible effect of environmental parameters
8.	Interference noise and other effects	Maximum effect	Minimum/negligible effect
9.	Storage/maintenance failure	Difficult storage and maintenance and higher failure rate	Easier storage and maintenance and reduced failure rate.

6.3 Types of Digital Filters

We are introduced with the digital filters. Now let us see the types of digital filters. The digital filters are of two types :

- (i) Finite Impulse Response (FIR) filters.
- and (ii) Infinite Impulse Response (IIR) filters.

These two types of FIR and IIR filters we have introduced earlier in chapter-2. Basically these are Linear Time Invariant (LTI) systems which are characterized by unit sample response. The FIR system has finite duration unit sample response. i.e.,

$$h(n) = 0 \quad \text{for } n < 0 \text{ and } n \geq M \quad \dots (6.3.1)$$

This unit sample response exists only for the duration from 0 to $M - 1$. Hence this is FIR system.

The IIR system has infinite duration unit-sample response. i.e.,

$$h(n) = 0 \quad \text{for } n < 0$$

This unit sample response exists for the duration from 0 to ∞ . Hence this is IIR system.

We have also seen earlier that FIR and IIR systems can be best described by difference equations. IIR systems can be easily described by recursive systems. FIR systems are nonrecursive. Readers are advised to revise all these concepts in chapter-2. Thus output of FIR filter depends only upon present and past inputs since it is nonrecursive, i.e. it does not use feedback. The IIR filters are recursive i.e. they use feedback. Hence output of IIR filter depends upon present input as well as past inputs and outputs.

We know that the difference equation of the LTI system is given as,

$$y(n) = - \sum_{k=1}^N a_k y(n-k) + \sum_{k=0}^M b_k x(n-k) \quad \dots (6.3.2)$$

In the above equation the terms under first summation are past outputs, and terms under second summation are present and past inputs. Hence the above difference equation represents IIR filter. For the FIR filter the first summation will be absent and the difference equation becomes,

$$y(n) = \sum_{k=0}^M b_k x(n-k) \quad \dots (6.3.3)$$

We know that output of the LTI system is given by convolution as,

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

For the FIR filter unit sample response $h(k)$ exists for the duration from 0 to $M - 1$ as we have seen in equation 6.3.1. Hence output of FIR filter becomes,

$$y(n) = \sum_{k=0}^{M-1} h(k) x(n-k) \quad \dots (6.3.4)$$

Here observe that above equation as well as equation 6.3.3 represent the output of FIR filter. Hence unit sample response of FIR filter is given as,

$$h(k) = b_k \quad \dots (6.3.5)$$

This is unit sample response of FIR filter can be directly obtained in terms of coefficients of difference equation. But there is no such easy and direct relation for unit sample response of IIR filter.

6.3.1 An Example of FIR Filter

To clear the difference between FIR and IIR filters, let us consider their examples. Actually we have seen an example of FIR filter as lowpass filter earlier, consider again the same example. We know that difference equation of the FIR filter is given by equation 6.3.3 as,

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

If we select $b_0 = \frac{1}{2}$ and $b_1 = \frac{1}{2}$, then above equation becomes,

$$y(n) = \frac{1}{2}x(n) + \frac{1}{2}x(n-1) \quad \dots (6.3.6)$$

This is the digital lowpass filter we have considered in equation 6.2.1 earlier. From equation 6.3.5 we have the unit sample response of this filter as,

$$h(k) = b_k, \quad k=0,1$$

$$\therefore h(0) = b_0 = \frac{1}{2}$$

and
$$h(1) = b_1 = \frac{1}{2}$$

$$\therefore \text{Unit sample response } h(n) = \left\{ \frac{1}{2}, \frac{1}{2} \right\}$$

This is the unit sample response of the digital lowpass filter we have considered. Observe that the unit sample response is finite in duration i.e. only two samples. Hence this is FIR filter.

6.3.2 An Example of IIR Filter

Now let us consider a simplest example of IIR filter. We know that IIR filters are described by the difference equation given by equation 6.3.2 as,

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

Here let $a_1 = -1$ and $b_0 = 1$ and rest of the coefficients are zero, then above equation becomes,

$$y(n) = y(n-1) + x(n) \quad \dots (6.3.7)$$

To show that this equation really represents IIR filter, we will obtain its unit sample response. Taking z-transform of above equation we get,

$$Y(z) = z^{-1}Y(z) + X(z)$$

$$\therefore Y(z) [1 - z^{-1}] = X(z)$$

$$\therefore \frac{Y(z)}{X(z)} = \frac{1}{1 - z^{-1}}$$

We know that $\frac{Y(z)}{X(z)}$ is the system function $H(z)$. i.e.,

$$H(z) = \frac{1}{1 - z^{-1}}$$

The inverse z-transform of system function $H(z)$ is called unit sample response $h(n)$. Hence taking inverse z-transform of above equation from standard z-transform tables we get,

$$h(n) = u(n) \text{ i.e. unit step function} \quad \dots (6.3.8)$$

Hidden page

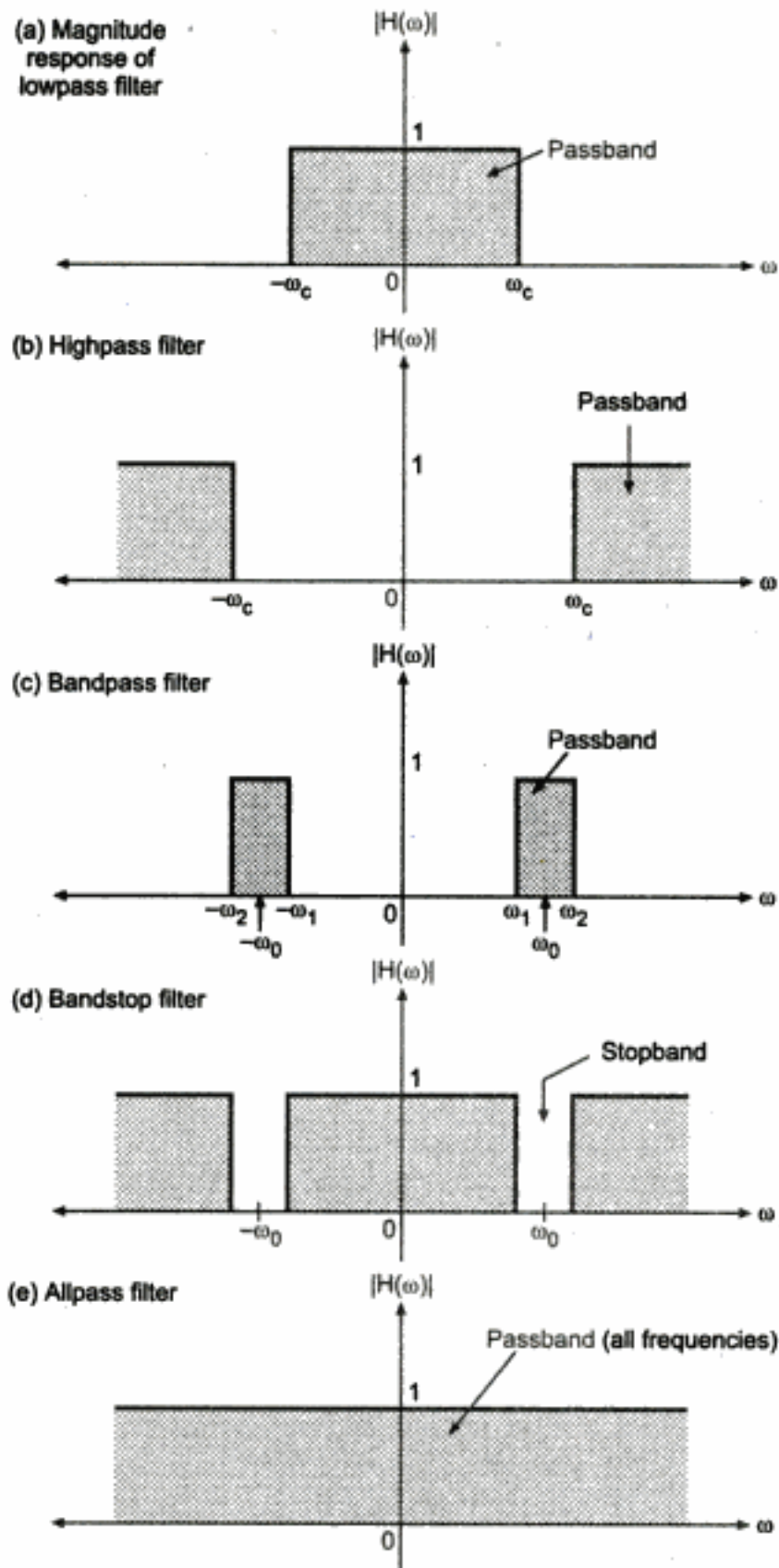


Fig. 6.4.1 Magnitude response characteristics (Ideal) of filters. The shaded area shows passband

Hidden page

Hidden page

6.5 Design of IIR Filters from Analog Filters

The analog filter design theory is well developed. Hence IIR digital filters are designed from analog filters. The analog filter is described by following system function.

$$H_a(s) = \frac{B(s)}{A(s)} = \frac{\sum_{k=0}^M \beta_k s^k}{\sum_{k=0}^N \alpha_k s^k} \quad \dots (6.5.1)$$

The system function $H_a(s)$ can also be obtained from impulse response $h(t)$, i.e.,

$$H_a(s) = \int_{-\infty}^{\infty} h(t) e^{-st} dt \quad \dots (6.5.2)$$

Thus laplace transform of $h(t)$ gives system function. The analog filter can also be described by a linear constant coefficient differential equation i.e.,

$$\sum_{k=0}^N \alpha_k \frac{d^k y(t)}{dt^k} = \sum_{k=0}^M \beta_k \frac{d^k x(t)}{dt^k} \quad \dots (6.5.3)$$

Here $x(t)$ is input to the filter and
 $y(t)$ is the output of the filter.

Normally the design of IIR digital filter is started from specifications of analog filter. The system function of the analog filter is then obtained. The system function of the digital filter is then obtained through some transformation. We know that the analog filter is stable if its poles fall in the left half of the s -plane. Hence the transformation techniques should have following desirable properties :

- 1) The $j\Omega$ axis of s -plane should map on the unit circle in the z -plane. This provides the direct relationship between the frequencies in s -and z -plane.
- 2) The left half plane of s -plane should be mapped inside of the unit circle in z -plane. Because of this, the stable analog filter is converted to a stable digital filter.

Further, we will consider few important techniques for design of digital filters from analog filters.

6.5.1 IIR Filter Design by Approximation of Derivatives

In this method, the differential equation of analog filter is approximated by an equivalent difference equation of the digital filter. For the derivative $\frac{dy(t)}{dt}$ at $t = nT$, following substitution is made,

$$\left. \frac{dy(t)}{dt} \right|_{t=nT} = \frac{y(nT) - y(nT - T)}{T}$$

Here T is the sampling interval and $y(n) = y(nT)$, hence above equation can be written as,

$$\frac{dy(t)}{dt} = \frac{y(n) - y(n-1)}{T} \quad \dots 4)$$

The system function of the differentiator having output $\frac{dy(t)}{dt}$ is,

$$H(s) = s \quad \dots (6.5.5)$$

The system function of the digital which produces output $\frac{y(n) - y(n-1)}{T}$ is

$$H(z) = \frac{1 - z^{-1}}{T} \quad \dots (6.5.6)$$

Thus the analog domain to digital domain transformation can be obtained (from equation 6.5.4, equation 6.5.5 and equation 6.5.6) as

$$s = \frac{1 - z^{-1}}{T} \quad \dots (6.5.7)$$

Similarly it can be shown that,

$$s^k = \left(\frac{1 - z^{-1}}{T} \right)^k \quad \dots (6.5.8)$$

Here 'k' represents the order of the derivative. Thus the system function of the digital filter can be obtained from the system function of the analog filter by approximation of derivatives as follows.

$$H(z) = H_a(s) \Big|_{s = \frac{1 - z^{-1}}{T}} \quad \dots (6.5.9)$$

From equation 6.5.7 we have,

$$z = \frac{1}{1 - sT} \quad \dots (6.5.10)$$

We know that $s = \sigma + j\Omega$, hence above equation becomes,

$$\begin{aligned} z &= \frac{1}{1 - (\sigma + j\Omega)T} \\ &= \frac{1}{1 - \sigma T - j\Omega T} \\ &= \frac{1 - \sigma T + j\Omega T}{(1 - \sigma T)^2 + (\Omega T)^2} \\ &= \frac{1 - \sigma T}{(1 - \sigma T)^2 + (\Omega T)^2} + j \frac{\Omega T}{(1 - \sigma T)^2 + (\Omega T)^2} \quad \dots (6.5.11) \end{aligned}$$

Let us see how $j\Omega$ axis is mapped in z-plane. For this, substitute $\sigma = 0$ in above equation. Hence we get,

$$z = \frac{1}{1 + (\Omega T)^2} + j \frac{\Omega T}{1 + (\Omega T)^2} \quad \dots (6.5.12)$$

The above equation shows that complete $j\Omega$ axis ($-\infty$ to $+\infty$) is mapped on the circle of radius $1/2$ and center at $z = \frac{1}{2}$. This is shown in Fig. 6.5.1. This circle is inside the unit circle.

From equation 6.5.11 it can be shown that left hand plane of $j\Omega$ axis maps inside

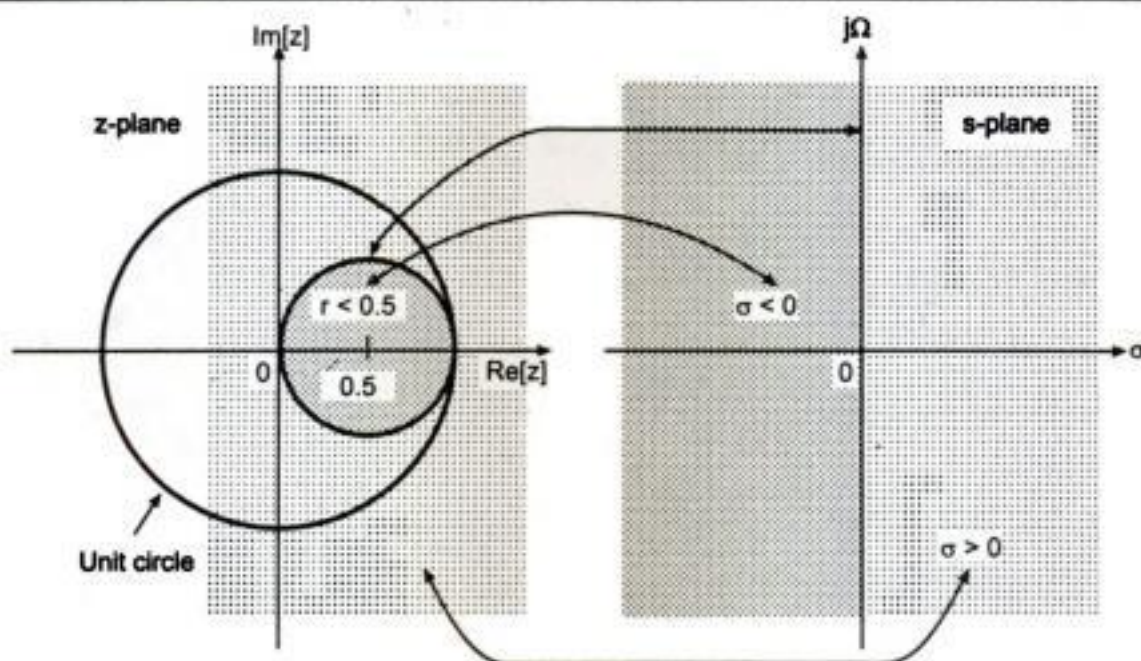


Fig. 6.5.1 Mapping from s plane to z plane for $s = \frac{1-z^{-1}}{T}$ | i.e. approximation of derivatives method

the circle of radius $\frac{1}{2}$ centered at $z = \frac{1}{2}$. And right hand plane of $j\Omega$ axis maps outside this circle. This is shown in Fig. 6.5.1. Thus a stable analog filter is converted to stable digital filter.

Ex. 6.5.1 Obtain the system function of the digital filter by approximation of derivatives if the system function of the analog filter is as follows :

$$H_a(s) = \frac{1}{(s+0.1)^2 + 9} \quad \dots (6.5.13)$$

Sol. : The derivative approximation is obtained by putting,

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

Hence equation 6.5.13 becomes,

$$\begin{aligned} H(z) &= H_a(s) \Big|_{s=\frac{1-z^{-1}}{T}} \\ &= \frac{1}{\left(\frac{1-z^{-1}}{T} + 0.1\right)^2 + 9} \\ &= \frac{T^2}{(1+0.2T+9.01T^2)} \\ &= \frac{1}{1 - \frac{2(1+0.1T)}{1+0.2T+9.01T^2} z^{-1} + \frac{1}{1+0.2T+9.01T^2} z^{-2}} \end{aligned}$$

This is the required system function.

6.5.2 IIR Filter Design by Impulse Invariance

Let the impulse responses of the analog filter be $h_a(t)$. Then the unit sample response of the corresponding digital filter is obtained by uniformly sampling the impulse response of the analog filter. We know that sampled signal is obtained by putting $t = nT$ i.e.,

$$h(n) = h_a(nT), \quad n = 0, 1, 2, \dots \quad \dots (6.5.14)$$

Here $h(n)$ is the unit sample response of digital filter and T is the sampling interval.

Let the system function of the analog filter be denoted as $H_a(s)$. Let us assume that the poles of analog filter are distinct. Then its partial fraction expansion can be written as,

$$H_a(s) = \sum_{k=1}^N \frac{c_k}{s - p_k} \quad \dots (6.5.15)$$

Here $\{p_k\}$ are the poles of the analog filter and $\{c_k\}$ are the coefficients of partial fraction expansion. The impulse response of the analog filter i.e. $h_a(t)$ can be obtained by taking inverse laplace transform of the system function $H_a(s)$ given by equation 6.5.15. From the standard relations of laplace transform we can obtain $h_a(t)$ from equation 6.5.15 as,

$$h_a(t) = \sum_{k=1}^N c_k e^{p_k t} \quad t \geq 0 \quad \dots (6.5.16)$$

The unit sample response of the digital filter is obtained by uniform sampling of $h_a(t)$. i.e.,

$$\begin{aligned} h(n) &= h_a(t)|_{t=nT} = h_a(nT) \\ &= \sum_{k=1}^N c_k e^{p_k nT} \end{aligned} \quad \dots (6.5.17)$$

Now the system function of the IIR filter can be obtained by taking z-transform of $h(n)$. i.e.,

$$\begin{aligned} H(z) &= \sum_{n=0}^{\infty} h(n) z^{-n} \quad \text{By definition of z-transform} \\ &= \sum_{n=0}^{\infty} \left[\sum_{k=1}^N c_k e^{p_k nT} \right] z^{-n} \\ &= \sum_{k=1}^N c_k \sum_{n=0}^{\infty} \left[e^{p_k T} z^{-1} \right]^n \end{aligned}$$

We know that,

$\sum_{n=0}^{\infty} a^n = \frac{1}{1-a}$ using this standard relation we can write above equation of $H(z)$ as,

$$H(z) = \sum_{k=1}^N \frac{c_k}{1 - e^{p_k T} z^{-1}} \quad \dots (6.5.18)$$

Thus we have obtained the transformation of analog system function [equation 6.5.15] to digital system function [equation 6.5.18]. i.e.,

$$\frac{1}{s - p_k} \rightarrow \frac{1}{1 - e^{p_k T} z^{-1}} \quad \dots (6.5.19)$$

Hidden page

We have seen that the discrete time system is stable if its poles lie inside the unit circle. The analog system is stable if its poles lie in left side plane of $j\Omega$ axis. Since left side plane of analog system is mapped inside the unit circle, a stable analog system is converted into stable digital system. The entire $j\Omega$ axis is mapped on the unit circle. Hence this mapping is not one to one. We know that range of ' ω ' is $-\pi \leq \omega \leq \pi$. Hence corresponding range of ' Ω ' is,

$$\omega = \Omega T$$

$$\therefore -\pi \leq \omega \leq \pi \text{ corresponds to } -\pi \leq \Omega T \leq \pi \text{ i.e. } -\frac{\pi}{T} \leq \Omega \leq \frac{\pi}{T}$$

$$\text{Thus } -\frac{\pi}{T} \leq \Omega \leq \frac{\pi}{T} \text{ maps on } -\pi \leq \omega \leq \pi$$

We know that $\pi \leq \omega \leq 3\pi$ is same as $-\pi \leq \omega \leq \pi$ on the unit circle.

$$\therefore \pi \leq \omega \leq 3\pi \text{ corresponds to } \pi \leq \Omega T \leq 3\pi \text{ i.e. } \frac{\pi}{T} \leq \Omega \leq \frac{3\pi}{T}$$

Thus $\frac{\pi}{T} \leq \Omega \leq \frac{3\pi}{T}$ maps on $-\pi \leq \omega \leq \pi$. This shows that the mapping of $j\Omega$ axis is many to one on unit circle. The segments $\frac{(2k-1)\pi}{T} \leq \Omega \leq \frac{(2k+1)\pi}{T}$ of $j\Omega$ axis are all mapped on the unit circle $-\pi \leq \omega \leq \pi$. This effect takes place because of sampling.

The following transformations can be useful in designing IIR filters by using impulse invariant method.

$$\frac{1}{(s+p_k)^m} \rightarrow \frac{(-1)^{m-1}}{(m-1)!} \cdot \frac{d^{m-1}}{d p_k^{m-1}} \cdot \frac{1}{1 - e^{-p_k T} z^{-1}} \quad \dots (6.5.24)$$

$$\frac{s+a}{(s+a)^2 + b^2} \rightarrow \frac{1 - e^{-aT} (\cos bT) z^{-1}}{1 - 2 e^{-aT} (\cos bT) z^{-1} + e^{-2aT} z^{-2}} \quad \dots (6.5.25)$$

$$\frac{b}{(s+a)^2 + b^2} \rightarrow \frac{e^{-aT} (\sin bT) z^{-1}}{1 - 2 e^{-aT} (\cos bT) z^{-1} + e^{-2aT} z^{-2}} \quad \dots (6.5.26)$$

Note that zeros of the system are not mapped according to the transformation discussed here.

Ex. 6.5.2 The system function of the analog filter is given as,

$$H_a(s) = \frac{s+0.1}{(s+0.1)^2 + 9}$$

Obtain the system function of the IIR digital filter by using impulse invariance method.

Sol. : The denominator of $H_a(s)$ has roots at

$$p_1 = -0.1 + j3, \quad p_2 = -0.1 - j3$$

$$\therefore H_a(s) = \frac{s+0.1}{(s+0.1-j3)(s+0.1+j3)}$$

Let us expand $H_a(s)$ in partial fractions,

$$H_a(s) = \frac{A_1}{s + 0.1 - j3} + \frac{A_2}{s + 0.1 + j3} \quad \dots (6.5.27)$$

Values of A_1 and A_2 can be obtained as follows :

$$A_1 = (s + 0.1 - j3) \cdot H_a(s) \Big|_{s = -0.1 + j3}$$

$$= \frac{s + 0.1}{s + 0.1 + j3} \Big|_{s = -0.1 + j3}$$

$$= \frac{-0.1 + j3 + 0.1}{-0.1 + j3 + 0.1 + j3} = \frac{1}{2}$$

$$A_2 = (s + 0.1 + j3) \cdot H_a(s) \Big|_{s = -0.1 - j3}$$

$$= \frac{s + 0.1}{s + 0.1 - j3} \Big|_{s = -0.1 - j3}$$

$$= \frac{-0.1 - j3 + 0.1}{-0.1 - j3 + 0.1 - j3} = \frac{1}{2}$$

Hence equation 6.5.27 becomes,

$$H_a(s) = \frac{\frac{1}{2}}{s + 0.1 - j3} + \frac{\frac{1}{2}}{s + 0.1 + j3} \quad \dots (6.5.28)$$

We know that impulse invariance transformation is given by equation 6.5.19 as,

$$\frac{1}{s - p_k} \rightarrow \frac{1}{1 - e^{p_k T} z^{-1}}$$

Using this relation we can obtain the system function for digital filter from equation 6.5.28 as,

$$H(z) = \frac{\frac{1}{2}}{1 - e^{-0.1T} + j3T z^{-1}} + \frac{\frac{1}{2}}{1 - e^{-0.1T} - j3T z^{-1}}$$

This system function can be simplified further as,

$$H(z) = \frac{1 - (e^{-0.1T} \cos 3T) z^{-1}}{1 - (2e^{-0.1T} \cos 3T) z^{-1} + e^{-0.2T} z^{-2}}$$

Observe that this equation can also be obtained using equation 6.5.25.

Frequency response of digital filter designed using impulse invariance :

Always we are required to find frequency response of digital filter. When the digital filter is designed using impulse invariance its frequency response is related to that of analog filter by following relation,

$$H(\omega) = \frac{1}{T} H_a \left(j \frac{\omega}{T} \right) \quad -\pi \leq \omega \leq \pi \quad \dots (6.5.29)$$

The proof of this relation is not presented here just to avoid complex mathematics.

Ex.6.5.3 If $H_a(s) = \frac{1}{(s+1)(s+2)}$, find the corresponding $H(z)$ using impulse invariance method for sampling frequency of 5 samples/sec. **[Dec-98]**

Sol. : Let us first expand $H_a(s)$ in partial fractions. i.e.,

$$H_a(s) = \frac{c_1}{s+1} + \frac{c_2}{s+2}$$

$$\therefore c_1 = (s+1)H_a(s)\Big|_{s=-1} = \frac{1}{s+2}\Big|_{s=-1}$$

$$= \frac{1}{-1+2} = 1$$

$$c_2 = (s+2)H_a(s)\Big|_{s=-2} = \frac{1}{s+1}\Big|_{s=-2}$$

$$= \frac{1}{-2+1} = -1$$

Hence $H_a(s)$ becomes,

$$H_a(s) = \frac{1}{s+1} - \frac{1}{s+2} \quad \dots (6.5.30)$$

It is given that sampling frequency $F_s = 5$ Hz

$$\therefore \text{Sampling period } T = \frac{1}{F_s} = \frac{1}{5} = 0.2$$

From equation 6.6.6, impulse invariance transformation is given as,

$$\frac{1}{s-p_k} \rightarrow \frac{1}{1-e^{p_k T} z^{-1}}$$

Now let us apply this transformation to individual terms of $H_a(s)$ of equation 6.5.30 i.e.,

$$\frac{1}{s+1} \rightarrow \frac{1}{1-e^{-1 \times 0.2} z^{-1}}, \quad \text{Here } p_1 = -1$$

and,

$$\frac{1}{s+2} \rightarrow \frac{1}{1-e^{-2 \times 0.2} z^{-1}}, \quad \text{Here } p_2 = -2$$

Hence $H(z)$ of digital filter becomes,

$$H(z) = \frac{1}{1-e^{-1 \times 0.2} z^{-1}} - \frac{1}{1-e^{-2 \times 0.2} z^{-1}}$$

$$\therefore H(z) = \frac{1}{1-e^{-0.2} z^{-1}} - \frac{1}{1-e^{-0.4} z^{-1}}$$

$$= \frac{1}{1-0.818 z^{-1}} - \frac{1}{1-0.67 z^{-1}}$$

On simplifying this equation we get,

$$H(z) = \frac{0.148 z}{z^2 - 1.48 z + 0.548}$$

6.5.3 IIR Filter Design by Bilinear Transformation

In the impulse invariance method the impulse response of analog filter is sampled. We know that whenever sampling takes place problems due to aliasing occur. This aliasing takes place in frequency domain. Hence to design higher frequency filters using impulse invariance, sampling frequencies should be high. This limits the use of impulse invariance method to only low pass (or lower frequencies) type of filters.

Hence we will discuss a different type of mapping from analog to digital domain which overcomes the limitations of impulse invariance method. It is called bilinear transformation and given as

$$s = \frac{2}{T} \left(\frac{1-z^{-1}}{1+z^{-1}} \right) \quad \dots (6.5.31)$$

The above equation can also be written as,

$$s = \frac{2}{T} \left(\frac{z-1}{z+1} \right) \quad \dots (6.5.32)$$

We know that $s = \sigma + j\Omega$ and polar form of 'z' is $z = r e^{j\omega}$. Putting this value of 'z' in above equation we get,

$$s = \frac{2}{T} \frac{r e^{j\omega} - 1}{r e^{j\omega} + 1}$$

We know that $e^{j\omega} = \cos \omega + j \sin \omega$, hence above equation becomes,

$$s = \frac{2}{T} \cdot \frac{r (\cos \omega + j \sin \omega) - 1}{r (\cos \omega + j \sin \omega) + 1}$$

Separating the real and imaginary parts of this equation we get,

$$s = \frac{2}{T} \left(\frac{r^2 - 1}{1 + r^2 + 2r \cos \omega} + j \frac{2r \sin \omega}{1 + r^2 + 2r \cos \omega} \right) \quad \dots (6.5.33)$$

We know that $s = \sigma + j\Omega$. Comparing with above equation, we get the values of σ and Ω as follows :

$$\sigma = \frac{2}{T} \cdot \frac{r^2 - 1}{1 + r^2 + 2r \cos \omega} \quad \dots (6.5.34)$$

and

$$\Omega = \frac{2}{T} \cdot \frac{2r \sin \omega}{1 + r^2 + 2r \cos \omega} \quad \dots (6.5.35)$$

From equation 6.5.34 we have following :

- (i) If $r > 1$, then $\sigma > 0$
- (ii) If $r < 1$, then $\sigma < 0$
- (iii) If $r = 1$, then $\sigma = 0$

The first statement indicates that the right hand side of s-plane (i.e. $\sigma > 0$) maps outside of the unit circle (i.e. $r > 1$).

The second statement indicates that the left hand side of s-plane (i.e. $\sigma < 0$) maps inside of the unit circle (i.e. $r < 1$).

The third statement indicates that the $j\Omega$ axis in s -plane (i.e. $\sigma = 0$) maps on the unit circle (i.e. $r = 1$).

This mapping is similar to that of impulse invariance method shown in Fig. 6.5.2. But in impulse invariance the mapping was valid only for poles. But bilinear transform maps poles as well as zeros. The mapping shows that a stable analog filter is converted to stable digital filter.

Now let us see how the imaginary axis $j\Omega$ is mapped on unit circle. Equation 6.5.35 gives relationship between Ω and ω i.e.,

$$\Omega = \frac{2}{T} \frac{2r \sin \omega}{1+r^2 + 2r \cos \omega}$$

If we want the relationship of $j\Omega$ axis in s -plane (i.e. $\sigma = 0$) to unit circle in z -plane (i.e. $r = 1$) we have to put $r = 1$ in above equation. Then we get,

$$\begin{aligned} \Omega &= \frac{2}{T} \frac{2 \sin \omega}{1+1+2 \cos \omega} \\ &= \frac{2}{T} \frac{\sin \omega}{1+\cos \omega} = \frac{2}{T} \frac{2 \sin \frac{\omega}{2} \cos \frac{\omega}{2}}{2 \cos^2 \frac{\omega}{2}} \\ &= \frac{2}{T} \tan \frac{\omega}{2} \quad \dots (6.5.36) \end{aligned}$$

$$\text{or} \quad \omega = 2 \tan^{-1} \frac{\Omega T}{2} \quad \dots (6.5.37)$$

This equation shows that the entire range of ' Ω ' maps only once in $-\pi \leq \omega \leq \pi$. This mapping is highly nonlinear. Fig. 6.5.3 shows this mapping.

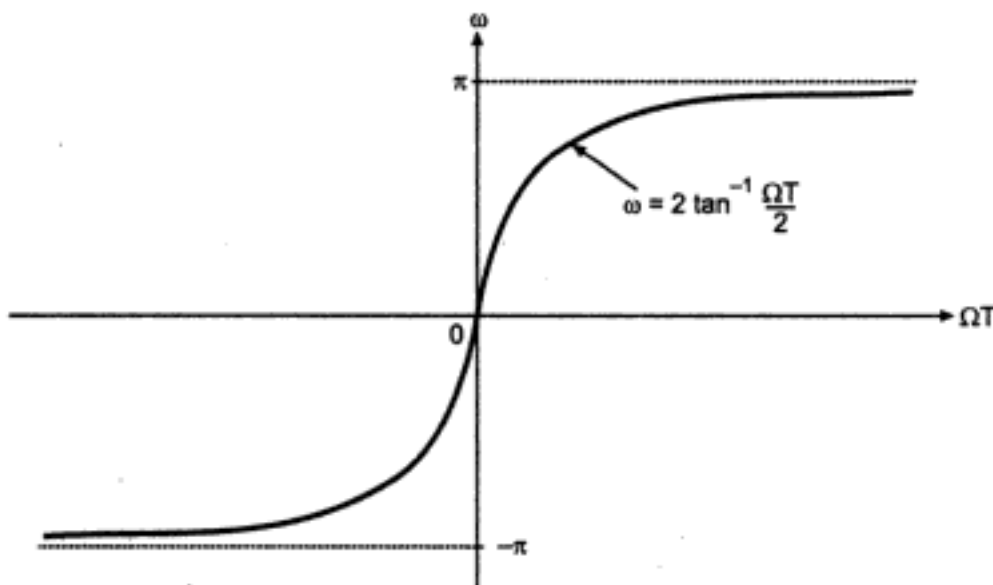


Fig. 6.5.3 Mapping between ' Ω ' and ' ω ' in bilinear transformation

Observe the nonlinearity in the relationship between ' ω ' and ' Ω '. It is called *frequency warping*.

Hidden page

Hidden page

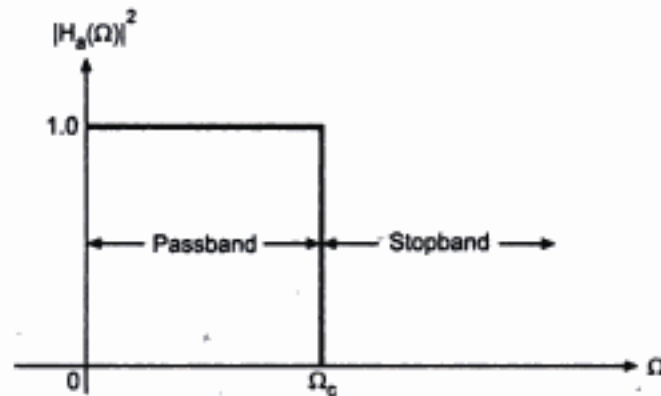


Fig. 6.6.1 Magnitude squared frequency response of ideal lowpass filter

Here we will discuss butterworth filter approximation in more details.

6.6.2 Butterworth Filter Approximation

The magnitude squared frequency response of the butterworth filter (lowpass) is given as,

$$|H_a(\Omega)|^2 = \frac{1}{1 + \left(\frac{\Omega}{\Omega_c}\right)^{2N}} \quad \dots (6.6.1)$$

Here N is the order of the filter and Ω_c is -3 dB cutoff or critical frequency. The above equation is also given as,

$$|H_a(\Omega)|^2 = \frac{1}{1 + \varepsilon^2 \left(\frac{\Omega}{\Omega_p}\right)^{2N}} \quad \dots (6.6.2)$$

Here Ω_p is the passband edge frequency and $\frac{1}{1 + \varepsilon^2}$ is the bandedge value of $|H_a(\Omega)|^2$.

To determine poles of $H_a(s)$:

At $s = j\Omega$, magnitude of $H(s)$ and $H(-s)$ is same, i.e.,

$$\begin{aligned} H_a(s) \cdot H_a(-s) &= |H_a(\Omega)| \cdot |H_a(\Omega)| \\ &= |H_a(\Omega)|^2 \end{aligned} \quad \dots (6.6.3)$$

We can write equation 6.6.1 as,

$$|H_a(\Omega)|^2 = \frac{1}{1 + \left(\frac{\Omega^2}{\Omega_c^2}\right)^N} \quad \dots (6.6.4)$$

At $s = j\Omega$, $\Omega^2 = -s^2$. Hence putting this value in above equation and from equation 6.6.3 we can write,

$$H_a(s) \cdot H_a(-s) = \frac{1}{1 + \left(\frac{-s^2}{\Omega_c^2}\right)^N} \quad \dots (6.6.5)$$

The poles of $H_a(s) \cdot H_a(-s)$ can be obtained by finding roots of denominator in above equation. i.e.,

$$1 + \left(\frac{-s^2}{\Omega_c^2} \right)^N = 0$$

$$\therefore \left(\frac{-s^2}{\Omega_c^2} \right)^N = -1$$

$$\therefore \frac{-s^2}{\Omega_c^2} = (-1)^{\frac{1}{N}} \quad \dots (6.6.6)$$

Let us consider $e^{j(2k+1)\pi}$ for $k = 0, 1, \dots, N-1$

$$e^{j(2k+1)\pi} = \cos(2k+1)\pi + j \sin(2k+1)\pi$$

Since $(2k+1)\pi = \pi, 3\pi, 5\pi, \dots$ etc, then above equation becomes,

$$e^{j(2k+1)\pi} = -1 + j0 \quad \text{for } k = 0, 1, 2, \dots, N-1$$

$$= -1 \quad \text{always for } k = 0, 1, 2, \dots, N-1$$

Putting for (-1) from above equation in equation 6.6.6 we get,

$$\frac{-s^2}{\Omega_c^2} = \left[e^{j(2k+1)\pi} \right]^{\frac{1}{N}} \quad \text{for } k = 0, 1, 2, \dots, N-1$$

$$= e^{j(2k+1)\pi/N} \quad \text{for } k = 0, 1, 2, \dots, N-1$$

$$\therefore s^2 = -\Omega_c^2 e^{j(2k+1)\pi/N} \quad \dots (6.6.7)$$

$$= (-1) \cdot \Omega_c^2 e^{j(2k+1)\pi/N}$$

Taking square root of both sides we get poles of equation 6.6.5. i.e.,

$$= \pm \sqrt{(-1)} \cdot \Omega_c \cdot \left[e^{j(2k+1)\pi/N} \right]^{\frac{1}{2}}$$

We know that $\sqrt{-1} = j$, hence we can write above equation as,

$$p_k = \pm j \Omega_c \cdot e^{j(2k+1)\pi/2N} \quad \text{for } k = 0, 1, 2, \dots, N-1 \quad (6.6.8)$$

Consider the value of $e^{j\frac{\pi}{2}}$ i.e.,

$$e^{j\frac{\pi}{2}} = \cos \frac{\pi}{2} + j \sin \frac{\pi}{2}$$

$$= 0 + j1 = j \quad \text{always.}$$

Hence putting for $j = e^{j\frac{\pi}{2}}$ in equation 6.6.8 we get,

$$p_k = \pm \Omega_c e^{j\frac{\pi}{2}} e^{j(2k+1)\pi/2N}$$

$$= \pm \Omega_c e^{j \left[\frac{\pi}{2} + \frac{(2k+1)\pi}{2N} \right]}$$

$$= \pm \Omega_c e^{j(N+2k+1)\pi/2N}$$

$$\left. \begin{array}{l} \\ \\ \end{array} \right\} k = 0, 1, 2, \dots, N-1 \quad (6.6.9)$$

This equation gives the pole position of $H_a(s) \cdot H_a(-s)$. For the filter to be stable we have to consider the poles left hand side of $j\Omega$ axis in s -plane. From above equation we

Hidden page

Here values of Ω_p and Ω_s are determined from ω_p and ω_s as follows :

(i) For impulse invariance method (see equation 6.5.23)

$$\Omega = \frac{\omega}{T} \quad \dots (6.6.12)$$

(ii) For bilinear transformation (see equation 6.5.36)

$$\Omega = \frac{2}{T} \tan \frac{\omega}{2} \quad \dots (6.6.13)$$

The butterworth filters are monotonic. Hence in equation 6.6.11 observe that if $|H_a(\Omega)| \geq A_p$ at $\Omega = \Omega_p$, then automatically the first condition of equation 6.6.11 is satisfied. Hence we can write equation 6.6.11 as,

$$\left. \begin{aligned} |H_a(\Omega)| &\geq A_p \text{ for } \Omega \leq \Omega_p \\ \text{and } |H_a(\Omega)| &\leq A_s \text{ for } \Omega_s \leq \Omega \end{aligned} \right\} \quad \dots (6.6.14)$$

Now let us consider the squared magnitude response of butterworth filters which is given by equation 6.6.1 as,

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

Hence according to the conditions of equation 6.6.14 above equation can be written as follows :

$$\frac{1}{1 + \left(\frac{\Omega_p}{\Omega_c}\right)^{2N}} \geq A_p^2 \quad \text{By putting } \Omega = \Omega_p \text{ in first condition of equation 6.6.14 and}$$

$$\frac{1}{1 + \left(\frac{\Omega_s}{\Omega_c}\right)^{2N}} \leq A_s^2 \quad \text{By putting } \Omega = \Omega_s \text{ in second condition of equation 6.6.14}$$

The above two equations can be written as follows :

$$\left(\frac{\Omega_p}{\Omega_c}\right)^{2N} \leq \frac{1}{A_p^2} - 1$$

$$\text{and } \left(\frac{\Omega_s}{\Omega_c}\right)^{2N} \geq \frac{1}{A_s^2} - 1$$

We need values of 'N' and ' Ω_c ' to determine order and poles of analog filter. Hence 'N' and ' Ω_c ' can be obtained from above equations by considering equalities. i.e.,

$$\left. \begin{aligned} \left(\frac{\Omega_p}{\Omega_c}\right)^{2N} &= \frac{1}{A_p^2} - 1 \\ \text{and } \left(\frac{\Omega_s}{\Omega_c}\right)^{2N} &= \frac{1}{A_s^2} - 1 \end{aligned} \right\} \quad \dots (6.6.15)$$

Taking the ratio of second equation to first in above equations we get,

Hidden page

Hidden page

If A_p and A_s are given in dB, then above values are given as,

$$\left. \begin{aligned} \epsilon &= \left(10^{0.1 A_p \text{ dB}} - 1 \right)^{\frac{1}{2}} \\ \text{and } \delta &= \left(10^{0.1 A_s \text{ dB}} - 1 \right)^{\frac{1}{2}} \end{aligned} \right\} \dots (6.6.24)$$

Similarly order 'N' can be given in terms of ϵ and δ . i.e. from equation 6.6.23 and equation 6.6.16 we get,

$$\begin{aligned} N &= \frac{1}{2} \frac{\log \left(\frac{\delta^2}{\epsilon^2} \right)}{\log \left(\frac{\Omega_s}{\Omega_p} \right)} \\ &= \frac{\log (\delta / \epsilon)}{\log (\Omega_s / \Omega_p)} \end{aligned} \dots (6.6.25)$$

These are the various equations which can be used to determine order 'N'. Fig. 6.6.3 illustrates various parameters discussed in these equations.

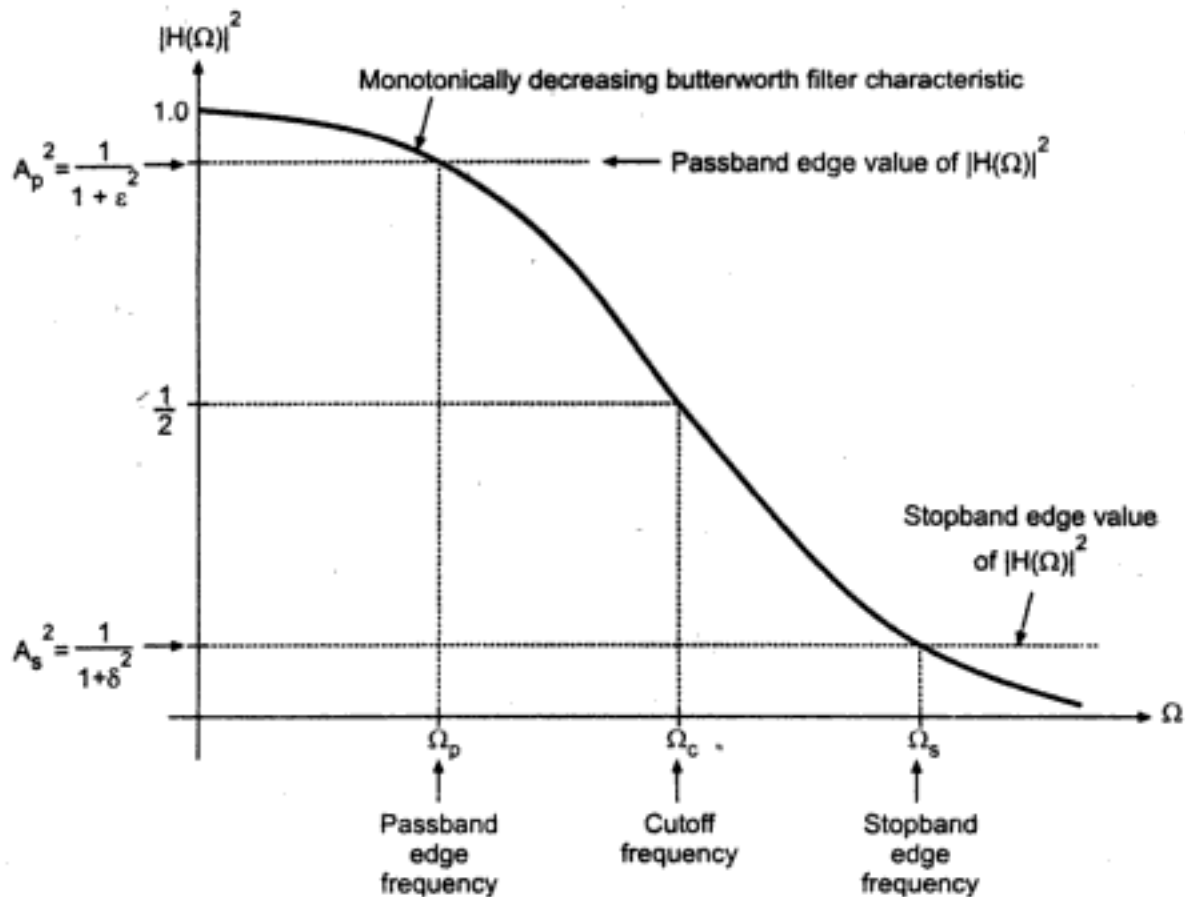


Fig. 6.6.3 Various frequencies and attenuations in filter characteristics

Ex.6.6.1 Design a discrete time lowpass filter using butterworth approximation by impulse invariance and realize the filter using parallel/cascade structure. The response is shown in Fig. 6.6.4 (a). [Dec-99]

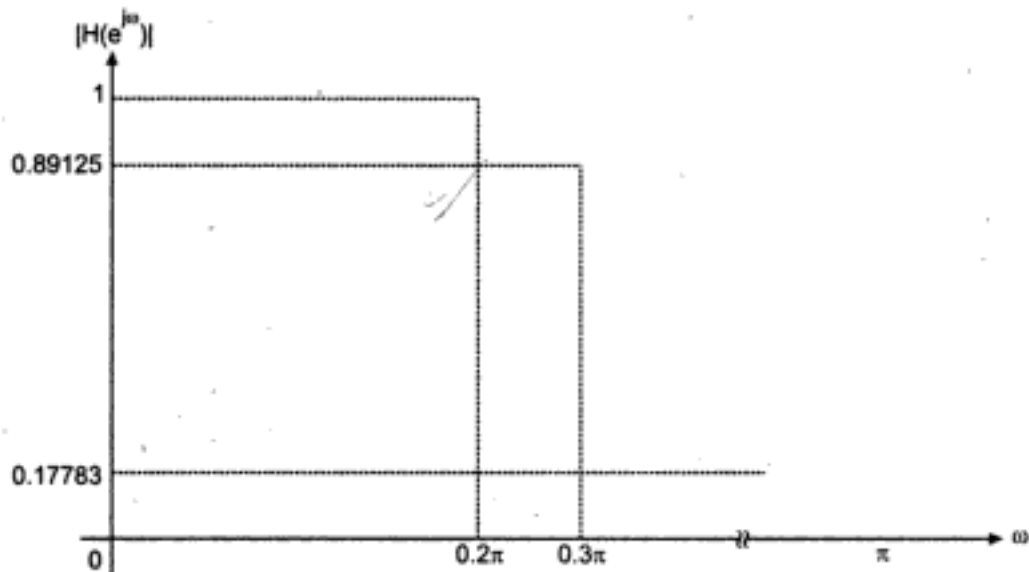


Fig. 6.6.4 (a) Specification of filter for example 6.6.1

Sol. : To identify specifications of given digital filter :

The given specifications in Fig. 6.6.4 (a) are little difficult to understand. It is more clearly shown in Fig. 6.6.4 (b) below :

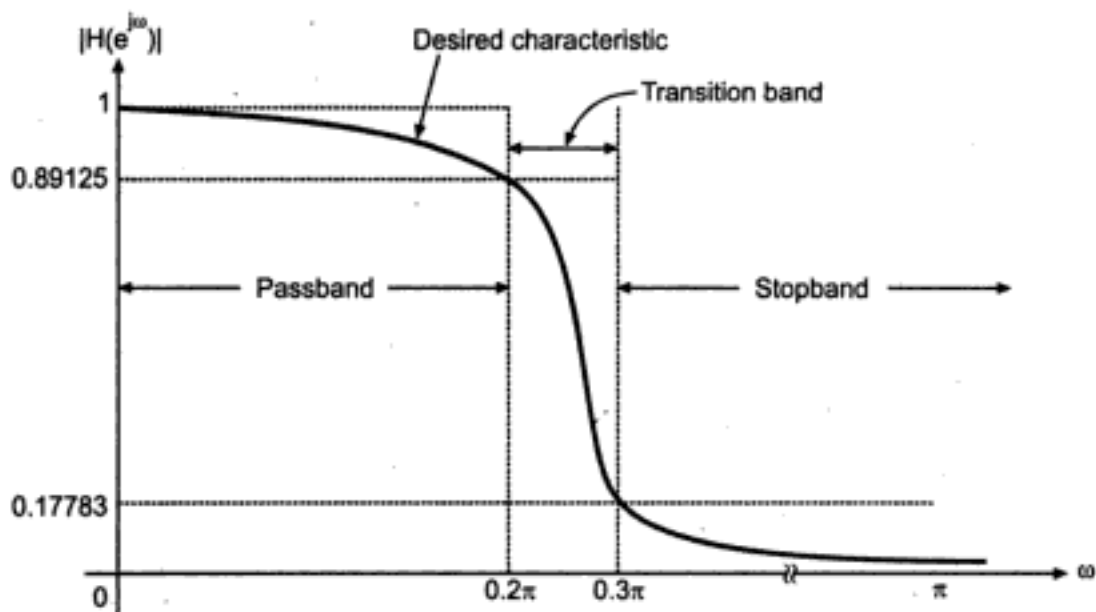


Fig. 6.6.4 (b) Specifications with desired characteristic

The above figure shows the desired lowpass filter characteristic passing through given specifications. These specifications are given for discrete time (i.e. digital) filter. Actually $|H(e^{j\omega})|$ is $|H(\omega)|$ i.e. magnitude of digital filter. The specifications of Fig. 6.6.4 (b) can be expressed mathematically as follows :

$$\left. \begin{aligned} 0.89125 \leq |H(\omega)| \leq 1 & \quad \text{for } 0 \leq \omega \leq 0.2 \pi \\ |H(\omega)| \leq 0.17783 & \quad \text{for } 0.3 \pi \leq \omega \leq \pi \end{aligned} \right\} \dots (6.6.26)$$

To convert specifications to equivalent analog filter :

In designing digital filter we need system function of analog filter. To obtain the system function of analog filter i.e. $H_a(s)$ we have to first convert the given specifications of digital filter to analog filter. Here we are using impulse invariance method. In this method we know that analog and discrete time frequencies are related by equation 6.5.23 as,

$$\omega = \Omega T$$

Hence the specifications of equation 6.6.26 can be written as,

$$\begin{aligned} 0.89125 \leq |H(\Omega)| \leq 1 & \quad \text{for } 0 \leq \Omega T \leq 0.2 \pi \\ \text{and } |H(\Omega)| \leq 0.17783 & \quad \text{for } 0.3 \pi \leq \Omega T \leq \pi \end{aligned}$$

The above two equations can be written as,

$$\left. \begin{aligned} 0.89125 \leq |H(\Omega)| \leq 1 & \quad \text{for } 0 \leq \Omega \leq \frac{0.2 \pi}{T} \\ \text{and } |H(\Omega)| \leq 0.17783 & \quad \text{for } \frac{0.3 \pi}{T} \leq \Omega \leq \frac{\pi}{T} \end{aligned} \right\} \dots (6.6.27)$$

These are the specifications of equivalent analog filter.

To determine order of the filter :

Compare the specifications of equation 6.6.27 with those of 6.6.11. Hence we get,

$$\left. \begin{aligned} A_p = 0.89125, \quad \Omega_p = \frac{0.2 \pi}{T} \\ \text{and } A_s = 0.17783, \quad \Omega_s = \frac{0.3 \pi}{T} \end{aligned} \right\} \dots (6.6.28)$$

Hence order of the filter is given by equation 6.6.16 as,

$$N = \frac{1}{2} \frac{\log \left[\left(\frac{1}{A_s^2} - 1 \right) / \left(\frac{1}{A_p^2} - 1 \right) \right]}{\log \left(\frac{\Omega_s}{\Omega_p} \right)}$$

Putting various values from equation 6.6.28 in above equation we get,

$$N = \frac{1}{2} \frac{\log \left\{ \left[\frac{1}{(0.17783)^2} - 1 \right] / \left[\frac{1}{(0.89125)^2} - 1 \right] \right\}}{\log \left(\frac{0.3 \pi / T}{0.2 \pi / T} \right)}$$

$$\therefore N = 6.8857$$

Here observe that the order of the filter should be integer number. In such situation always nearest higher integer value of 'N' is selected to satisfy the specifications. Hence, order of the filter, $N = 6$.

Hidden page

$$\begin{aligned}
 k=2 &\Rightarrow p_2 = \pm 0.7032 e^{j 11 \pi/12} \\
 &= -0.679 + j 0.182 \text{ and } 0.679 - j 0.182 \\
 k=3 &\Rightarrow p_3 = 0.7032 e^{j 13 \pi/12} \\
 &= -0.679 - j 0.182 \text{ and } 0.679 + j 0.182 \\
 k=4 &\Rightarrow p_4 = 0.7032 e^{j 15 \pi/12} \\
 &= -0.497 - j 0.497 \text{ and } 0.497 + j 0.497 \\
 k=5 &\Rightarrow p_5 = 0.7032 e^{j 17 \pi/12} \\
 &= -0.182 - j 0.679 \text{ and } 0.182 + j 0.679
 \end{aligned}$$

Thus there are total 12 poles as calculated above. Figure 6.6.5 shows poles plotted in the s -plane. Observe that all the poles lie on the circle of radius $\Omega_c = 0.7032$.

Please refer Fig. 6.6.5 on next page.

The poles shown in Fig. 6.6.5 are due to $H_a(s) \cdot H_a(-s)$. If we want stable filter then $H(s)$ should have all the poles in left half of s -plane. Hence we have to consider the poles lying in left half of the s -plane. These poles are as follows :

Poles of $H_a(s)$:

$$\begin{aligned}
 -0.182 \pm j 0.679 &\Rightarrow \text{pair 1} \\
 -0.497 \pm j 0.497 &\Rightarrow \text{pair 2} \\
 -0.679 \pm j 0.182 &\Rightarrow \text{pair 3}
 \end{aligned}$$

The above poles are combined as complex conjugate pairs.

Let us name these pole pairs as,

$$\begin{aligned}
 s_1 &= -0.182 + j 0.679 \text{ and } s_1^* = -0.182 - j 0.679 \\
 s_2 &= -0.497 + j 0.497 \text{ and } s_2^* = -0.497 - j 0.497 \\
 s_3 &= -0.679 + j 0.182 \text{ and } s_3^* = -0.679 - j 0.182
 \end{aligned}$$

To determine system function $H_a(s)$:

Here we have to combine complex conjugate poles so that all the coefficients will be real. For $N=6$, $H_a(s)$ is given as,

$$H_a(s) = \frac{\Omega_c^6}{(s-s_1)(s-s_1^*) \cdot (s-s_2)(s-s_2^*) \cdot (s-s_3)(s-s_3^*)}$$

Here observe that numerator is Ω_c^6 is made according to the theory of butterworth filters. For example for second order butterworth filter, the numerator will be Ω_c^2 . Putting values in above equation we get,

$$\begin{aligned}
 H_a(s) &= \frac{(0.7032)^6}{[(s+0.182-j0.679)(s+0.182+j0.679) \cdot (s+0.497-j0.497)(s+0.497+j0.497) \cdot (s+0.679-j0.182)(s+0.679+j0.182)]} \\
 &= \frac{0.1209}{\left[(s+0.182)^2 + (0.679)^2 \right] \cdot \left[(s+0.497)^2 + (0.497)^2 \right] \cdot \left[(s+0.679)^2 + (0.182)^2 \right]}
 \end{aligned}$$

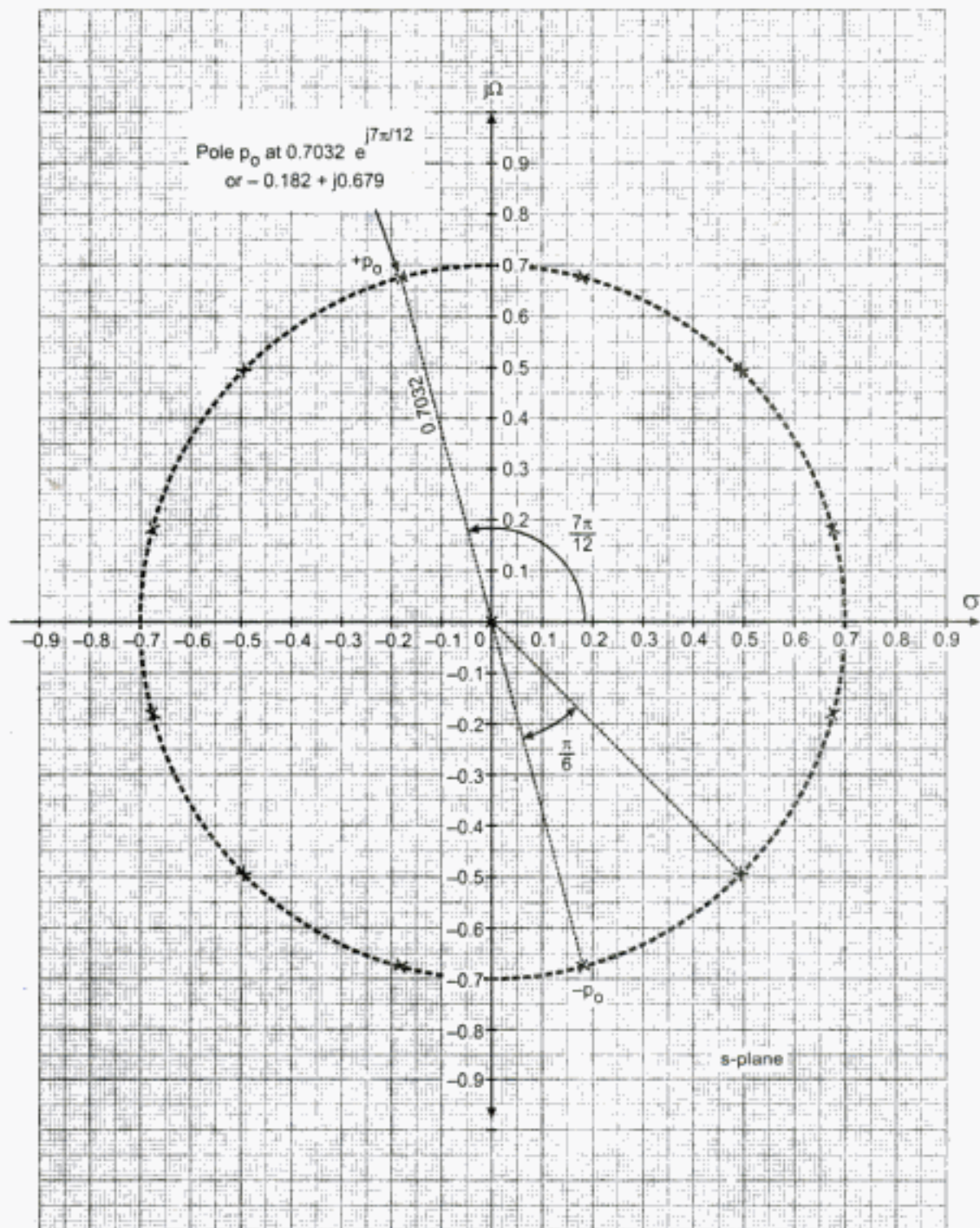


Fig. 6.6.5 Poles of the function $H_B(s) \cdot H_B(-s)$. These poles are located on circle of radius $\Omega_c = 0.7032$. Twelve poles are located at angular distance of $\frac{\pi}{6}$

Hidden page

Defining specifications for digital filter :

Order of the filter, $N = 2$ (given) cutoff frequency f_c of the digital filter can be obtained by applying the formula for conversion of continuous to discrete time frequencies.

(i.e. $f = \frac{F}{F_s}$) Hence,

$$f_c = \frac{F_c}{F_s} = \frac{1000}{10000} = 0.1 \text{ cycles/sample}$$

We know that angular frequency of the discrete time signal is given as ($\omega = 2\pi f$) i.e.,

$$\omega_c = 2\pi f_c = 2\pi \times 0.1 = 0.2 \pi \text{ radians/sample}$$

This is the angular discrete time cutoff frequency required.

To obtain specifications of equivalent analog filter for Bilinear Transformation :

In this example we are using bilinear transformation. We have cutoff frequency of the digital filter as $\omega_c = 0.2 \pi$. Now we should determine the equivalent cutoff frequency (Ω_c) of analog filter according to bilinear transformation. From equation 6.5.36, the frequency relationship in bilinear transformation is given as,

$$\Omega = \frac{2}{T} \tan \frac{\omega}{2}$$

$$\Omega_c = \frac{2}{T} \tan \frac{\omega_c}{2}$$

Here 'T' is the sampling duration and it is given as $T = \frac{1}{F_s}$. Here $F_s = 10000$ Hz. Hence

$T = \frac{1}{10000}$ sec. Putting these values above equation becomes,

$$\begin{aligned} \Omega_c &= \frac{2}{\left(\frac{1}{10000}\right)} \tan \left(\frac{0.2 \pi}{2}\right) \\ &= 6498.4 \text{ radians/sec} \end{aligned}$$

Thus we have the specifications of analog filter for bilinear transformation as,

$$\begin{aligned} \Omega_c &= 6498.4 \text{ radians/sec and} \\ N &= 2 \end{aligned}$$

Important note :

Here observe that first we obtained digital filter cutoff frequency. Then we obtained analog filter cutoff frequency according to bilinear transformation frequency relationship. Some times this procedure is called *prewarping*. This prewarping removes the warping effect when we apply bilinear transformation to analog filter system function. Hence the cutoff frequencies are mapped properly. The same procedure was done in previous example also for impulse invariance method, even though there is no warping concept. Hence this is the standard procedure for impulse invariance as well as bilinear transformation.

To obtain poles of $H_a(s)$:

The poles of $H_a(s) \cdot H_a(-s)$ are given by equation 6.6.9 as,

$$p_k = \pm \Omega_c e^{j(N+2k+1)\pi/2N}, \quad k = 0, 1, 2, \dots, N-1$$

For $\Omega_c = 6498.4$ and $N = 2$ above equation becomes,

$$p_k = \pm 6498.4 e^{j(3+2k)\pi/4}, \quad k = 0, 1 \quad \dots (6.6.37)$$

With $k = 0$ in above equation we get,

$$\begin{aligned} \therefore p_0 &= \pm 6498.4 e^{j3\pi/4} \\ &= -4595.0627 + j 4595.0627 \quad \text{and} \quad 4595.0627 - j 4595.0627 \end{aligned}$$

With $k = 1$ in equation 6.6.37 we get

$$\begin{aligned} p_1 &= \pm 6498.4 e^{j5\pi/4} \\ &= -4595.0627 - j 4595.0627 \quad \text{and} \quad 4595.0627 + j 4595.0627 \end{aligned}$$

The poles of $H_a(s)$ will be the poles lying in left half of the s-plane. i.e.,

$$\begin{aligned} s_1 &= -4595 + j 4595.0627 \quad \text{and} \\ s_1^* &= -4595 - j 4595.0627 \end{aligned}$$

Observe that the two poles lying in left half of s-plane are complex conjugate of each other. Since this is second order filter there are two poles. Note that the poles always occur in complex conjugate pairs if 'N' is even. If 'N' is odd, then $(N-1)$ poles occur in complex conjugate pairs and one pole lies on real axis. Hence the coefficients are not imaginary.

To obtain system function $H_a(s)$:

For butterworth approximation, the system function for second order filter is given as,

$$H_a(s) = \frac{\Omega_c^2}{(s - s_1)(s + s_1^*)}$$

Here the order is '2' hence numerator is equal to Ω_c^2 . Putting values in above equation we get,

$$\begin{aligned} H_a(s) &= \frac{(6498.4)^2}{(s + 4595.0627 - j 4595.0627)(s + 4595.0627 + j 4595.0627)} \\ &= \frac{(6498.4)^2}{(s + 4595.0627)^2 + (4595.0627)^2} \\ &= \frac{(6498.4)^2}{s^2 + 9190.125s + 42.23 \times 10^6} \quad \dots (6.6.38) \end{aligned}$$

This is the system function of analog filter.

To obtain $H(z)$ using bilinear transformation :

Next step is to obtain system function of the digital filter i.e. $H(z)$ by applying bilinear transformation to $H_a(s)$. Bilinear transformation is given by equation 6.5.31 as,

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

We have $T = \frac{1}{10000}$, then above equation becomes,

$$\begin{aligned}
 s &= \frac{2}{\left(\frac{1}{10000}\right)} \left(\frac{1-z^{-1}}{1+z^{-1}}\right) \\
 &= 2 \times 10^4 \left(\frac{1-z^{-1}}{1+z^{-1}}\right) \quad \dots (6.6.39)
 \end{aligned}$$

Putting for 's' from above equation in equation 6.6.38 we get $H(z)$ i.e.,

$$H(z) = \frac{(6498.4)^2}{\left[2 \times 10^4 \left(\frac{1-z^{-1}}{1+z^{-1}}\right)\right]^2 + 9190.125 \left[2 \times 10^4 \left(\frac{1-z^{-1}}{1+z^{-1}}\right)\right] + 42.23 \times 10^6}$$

On simplifying above equation we get,

$$H(z) = \frac{0.0676 (1 + 2z^{-1} + z^{-2})}{1 - 1.143z^{-1} + 0.4128z^{-2}} \quad \dots (6.6.40)$$

This is the required system function of digital lowpass second order butterworth filter. It can be easily realized in direct form I or II.

Ex.6.6.3 To show that $\frac{2}{T}$ can be assumed to be '1' while solving problems of filter design using bilinear transformation.

Explain this concept with the help of redesigning second order lowpass filter of last example (i.e. example 6.6.2)

Sol. : Important (What we are doing ?) :

We have already designed a second order lowpass filter of $F_c = 1 \text{ kHz}$ and $F_s = 10,000 \text{ Hz}$ in previous example. Here we will consider the same filter design but we will assume $\frac{2}{T} = 1$.

This means we will show that $\frac{2}{T}$ cancels out.

Given data :

Order to filter $N = 2$

Analog filter cutoff frequency $F_c = 1000 \text{ Hz}$ sampling frequency $F_s = 10000 \text{ Hz}$

Defining Specifications for digital filter :

Order of the filter, $N = 2$ (given). Cutoff frequency f_c of the digital filter can be obtained by applying the formula for conversion of continuous to discrete time frequencies $\left(\text{i.e. } f = \frac{F}{F_s}\right)$

Hence,

$$f_c = \frac{F_c}{F_s} = \frac{1000}{10000} = 0.1 \text{ cycles/sample}$$

Hence

$$\omega_c = 2\pi f_c = 0.1 \times 2\pi = 0.2 \pi \text{ radius/sample}$$

To obtain specifications of equivalent analog filter for bilinear transformation :

Here we are applying bilinear transformation hence the cutoff frequency of digital signal i.e. ω_c should be converted to its equivalent value of analog filter i.e. Ω_c according to bilinear transformation frequency relationship. This relationship is given by equation 6.5.36 as,

$$\Omega = \frac{2}{T} \tan \frac{\omega}{2}$$

Now we will consider $\frac{2}{T} = 1$. Hence,

$$\begin{aligned} \Omega_c &= \tan \frac{\omega_c}{2} \\ &= \tan \frac{0.2\pi}{2} = 0.325 \end{aligned}$$

Thus we have specifications of equivalent analog filter according to bilinear transformation are,

$$\Omega_c = 0.325$$

and $N = 2$

To obtain poles of $H_a(s)$:

The poles of $H_a(s) \cdot H_a(-s)$ are given by equation 6.6.9 as,

$$p_k = \pm \Omega_c e^{j(N+2k+1)\pi/2N}, \quad k = 0, 1, 2 \dots N-1$$

For $N = 2$ and $\Omega_c = 0.325$ above equation becomes

$$p_k = \pm 0.325 e^{j(3+2k)\pi/4}, \quad k = 0, 1$$

With $k = 0$ in above equation,

$$\begin{aligned} p_0 &= \pm 0.325 e^{j3\pi/4} \\ &= -0.229 + j 0.229 \text{ and } 0.229 - j 0.229 \end{aligned}$$

Similarly with $k = 1$ we get,

$$\begin{aligned} p_1 &= \pm 0.325 e^{j5\pi/4} \\ &= -0.229 - j 0.229 \text{ and } 0.229 + j 0.229 \end{aligned}$$

For stable filter we have to consider poles lying in left half of s-plane. Hence poles of $H_a(s)$ will be,

$$\begin{aligned} s_1 &= -0.229 + j 0.229 \quad \text{and} \\ s_1^* &= -0.229 - j 0.229 \end{aligned}$$

To determine system function $H_a(s)$:

The system function of second order butterworth lowpass filter is given as,

$$H_a(s) = \frac{\Omega_c^2}{(s - s_1)(s - s_1^*)}$$

Here numerator is Ω_c^2 since it is second order butterworth filter. Putting the values of s_1 and s_1^* in above equation we get,

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Here we want first order function. Hence $N=1$ therefore above equation will have only one value at $k=0$. i.e.,

$$\begin{aligned} p_0 &= \pm e^{j2\pi/2} \\ &= \pm e^{j\pi} = -1, +1 \end{aligned}$$

Since we want poles in left side of $j\Omega$ axis for stable filter, the pole of our lowpass filter is,

$$s_1 = -1$$

Hence the system function of butterworth lowpass filter will be,

$$H_{aLP}(s) = \frac{1}{s+1} \quad \dots (6.7.5)$$

Here observe that numerator is '1' since $\Omega_c=1$ i.e. normalized lowpass filter.

To obtain system function of analog highpass filter by frequency transformation :

Now we have to apply frequency transformation on system function of prototype lowpass filter to obtain system function of highpass filter. The lowpass to highpass transformation is given by equation 6.7.2 as,

$$s \rightarrow \frac{\Omega_p \Omega_{HP}}{s}$$

Here Ω_p is passband edge frequency of lowpass filter. This is Ω_c , which we have assumed '1'. Hence $\Omega_p=1$. Hence above transformation will be,

$$s \rightarrow \frac{\Omega_{HP}}{s}$$

We have obtained Ω_{HP} as 217.963 i.e. prewarped cutoff frequency of highpass filter. Hence,

$$s \rightarrow \frac{217.963}{s}$$

Hence system function of highpass filter is given as,

$$\begin{aligned} H_{aHP}(s) &= H_{aLP}(s) \Big|_{s \rightarrow \frac{\Omega_{HP}}{s}} \\ &= \frac{1}{s+1} \Big|_{s \rightarrow \frac{217.963}{s}} \\ &= \frac{1}{\frac{217.963}{s} + 1} \\ &= \frac{s}{s + 217.963} \quad \dots (6.7.6) \end{aligned}$$

This is the system function of analog highpass filter.

To obtain $H(z)$ by bilinear transformation :

We know that bilinear transformation is given as,

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

Hidden page

Hidden page

$$s \rightarrow \frac{s^2 + 1299.68 \times 2038.1}{1(2038.1 - 1299.68)}$$

$$\therefore s \rightarrow \frac{s^2 + 2.64887 \times 10^6}{738.42 s}$$

Hence system function of bandpass filter becomes,

$$\begin{aligned} H_{aBP}(s) &= H_{aLP}(s) \Big|_{s \rightarrow \frac{s^2 + 2.64887 \times 10^6}{738.42 s}} \\ &= \frac{1}{s+1} \Big|_{s \rightarrow \frac{s^2 + 2.64887 \times 10^6}{738.42 s}} \\ &= \frac{1}{\left(\frac{s^2 + 2.64887 \times 10^6}{738.42 s} \right) + 1} \\ &= \frac{738.42 s}{s^2 + 738.42s + 2.64887 \times 10^6} \end{aligned} \quad \dots (6.7.10)$$

To obtain $H(z)$ by bilinear transformation :

We know that bilinear transformation is given as,

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

Putting for $T = \frac{1}{F_s} = \frac{1}{150}$, above equation becomes,

$$\begin{aligned} s &= \frac{2}{\left(\frac{1}{2000} \right)} \left(\frac{1-z^{-1}}{1+z^{-1}} \right) \\ &= 4000 \left(\frac{1-z^{-1}}{1+z^{-1}} \right) \end{aligned}$$

Putting this value of s in equation 6.7.10 we get,

$$H(z) = \frac{738.42 \times 4000 \left(\frac{1-z^{-1}}{1+z^{-1}} \right)}{\left[4000 \left(\frac{1-z^{-1}}{1+z^{-1}} \right) \right]^2 + 738.42 \left[4000 \left(\frac{1-z^{-1}}{1+z^{-1}} \right) \right] + 2.64887 \times 10^6}$$

On simplifying above equation we get,

$$H(z) = \frac{0.1367 (1-z^{-1})}{1 - 1.237 z^{-1} + 0.726 z^{-2}}$$

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

For even value of M we can write equation 6.8.14 directly as,

$$H(\omega) = e^{-j\omega\left(\frac{M-1}{2}\right)} \left\{ 2 \sum_{n=0}^{\frac{M}{2}-1} h(n) \cos \omega \left(n - \frac{M-1}{2} \right) \right\} \quad \dots (6.8.18)$$

Comparing above equation with the polar form of $H(\omega)$ of equation 6.8.15 we get,

$$\text{Magnitude} \quad |H(\omega)| = 2 \sum_{n=0}^{\frac{M}{2}-1} h(n) \cos \omega \left(n - \frac{M-1}{2} \right) \quad \dots (6.8.19)$$

And phase $\angle H(\omega)$ is given as,

$$\angle H(\omega) = \begin{cases} -\omega \left(\frac{M-1}{2} \right) & \text{for } |H(\omega)| > 0 \\ -\omega \left(\frac{M-1}{2} \right) + \pi & \text{for } |H(\omega)| < 0 \end{cases} \quad \dots (6.8.20)$$

The above equation show that phase is piecewise linear. Similar results can be obtained for antisymmetric unit sample response. Thus for linear phase FIR filter,

$$h(n) = \pm h(M-1-n) \quad \dots (6.8.21)$$

The above condition should be satisfied.

Linear phase is the most important feature of FIR filters. IIR filters cannot be designed with linear phase. The linear phase in FIR filters can be obtained if unit sample response satisfies equation 6.8.21. Many applications need linear phase filtering. For example the speech related applications require linear phase. Similarly in data transmission applications, linear phase prevents pulse dispersion and the detection becomes more accurate. Hence whenever linear phase is desired, FIR filtering is used.

6.8.4 Magnitude Characteristics and Order of FIR Filter

The magnitude specifications of FIR filters are given in different way compared to those of IIR filters. Fig. 6.8.3 shows the magnitude specifications from which FIR filter is to be designed.

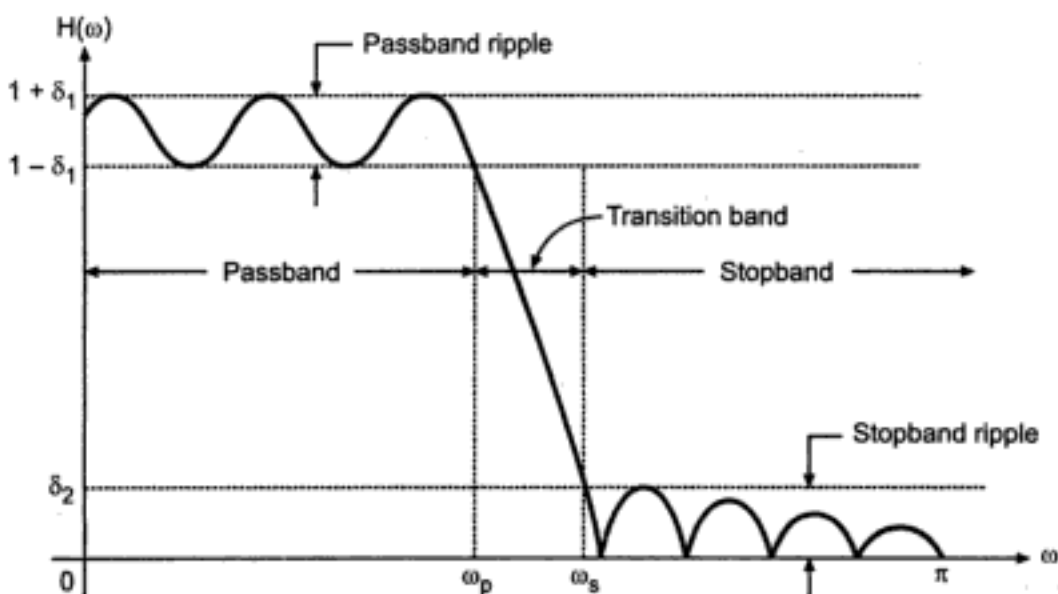


Fig. 6.8.3 Magnitude specifications used for FIR filter design

The magnitude response given in above figure can be expressed mathematically as,

$$\left. \begin{aligned} 1 - \delta_1 \leq |H(\omega)| \leq 1 + \delta_1 & \quad \text{for} \quad 0 \leq \omega \leq \omega_p \\ 0 \leq |H(\omega)| \leq \delta_2 & \quad \text{for} \quad \omega_s \leq \omega \leq \pi \end{aligned} \right\} \dots (6.8.22)$$

The approximate empirical formula for order N is given as,

$$N = \frac{-10 \log_{10} (\delta_1 \cdot \delta_2) - 15}{14 \Delta f} \dots (6.8.23)$$

Here $\Delta f = \frac{\omega_s - \omega_p}{2\pi}$ is the transition band.

or $\Delta f = f_s - f_p$, where $\omega_s = 2\pi f_s$ and $\omega_p = 2\pi f_p$.

And length of the filter i.e. $M = N$.

For the similar magnitude specifications the FIR filters have higher order than IIR filters. This is because FIR filters do not use feedback, hence they need long sequences for $h(n)$ (i.e. higher order) to get sharp cutoff filters. Because of increased number of coefficients, FIR filters require large time for processing. This processing time can be reduced by using FFT algorithms.

6.9 FIR Filter Design

In the last section we discussed various characteristics of FIR filters such as stability linear phase, order of filter etc. Now let us see how FIR filters are designed. There are three methods mainly used for FIR filter design. They are

- (i) FIR filter design using windows
- (ii) FIR filter design using frequency sampling or inverse fourier transform and
- (iii) Optimal or minimax FIR filter design.

Here we will discuss first two methods in detail.

6.9.1 Design of Linear Phase FIR Filters Using Windows

Let us consider that the digital filter which is to be designed have the frequency response $H_d(\omega)$. This is also called desired frequency response. Let the corresponding unit sample response (desired) be $h_d(n)$. We know that $H_d(\omega)$ is fourier transform of $h_d(n)$. i.e.,

$$H_d(\omega) = \sum_{n=0}^{\infty} h_d(n) e^{-j\omega n} \dots (6.9.1)$$

And $h_d(n)$ can be obtained by taking inverse fourier transform of $H_d(\omega)$. i.e.,

$$h_d(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(\omega) e^{j\omega n} d\omega \dots (6.9.2)$$

That is the desired unit sample response is obtained from desired frequency response by above equation.

Generally the unit sample response obtained by equation 6.9.2 is infinite in duration. Since we are designing a finite impulse response filter, the length of $h_d(n)$ should be made finite. If we want the unit sample response of length 'M', then $h_d(n)$ is truncated to length 'M'. This is equivalent to multiplying $h_d(n)$ by a window sequence $w(n)$. This concept can be best explained by considering a particular type of window sequence. Here we consider rectangular window.

Hidden page

$$\begin{aligned}
 W_R(\omega) &= \frac{e^{-j\omega \frac{M}{2}} \cdot 2 \sin\left(\omega \frac{M}{2}\right)}{e^{-j\frac{\omega}{2}} \cdot 2 \sin\left(\frac{\omega}{2}\right)} \\
 &= e^{-j\omega \left(\frac{M-1}{2}\right)} \cdot \frac{\sin\left(\frac{\omega M}{2}\right)}{\sin\left(\frac{\omega}{2}\right)} \quad \dots (6.9.10)
 \end{aligned}$$

The polar form of $W_R(\omega)$ is given as $|W_R(\omega)| e^{j\angle W_R(\omega)}$. Comparing with above equation we get magnitude response of rectangular window as,

$$|W_R(\omega)| = \frac{\left| \sin\left(\frac{\omega M}{2}\right) \right|}{\left| \sin\left(\frac{\omega}{2}\right) \right|} \quad \dots (6.9.11)$$

Fig. 6.9.1 shows the response of rectangular window given by above equation for $M = 50$.

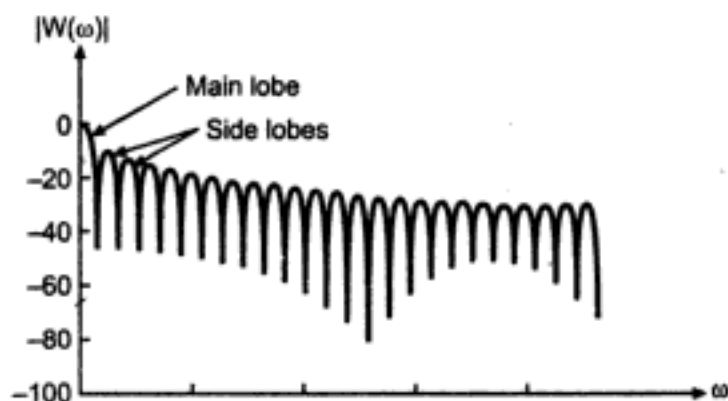


Fig. 6.9.1 Magnitude spectrum of rectangular window for $M = 50$

In the magnitude spectrum of rectangular window given above, observe that there is one main lobe and many side lobes. As 'M' increases the main lobe becomes narrower. The area under the side lobes remain same irrespective of changes in 'M'.

From equation 6.9.4 we know that unit sample response of FIR filter is given as,

$$h(n) = h_d(n) w(n)$$

Here $w(n)$ represents generalized window function. The frequency response of FIR filter can be obtained by taking fourier transform of above equation. i.e.,

$$H(\omega) = F.T. \{h_d(n) \cdot w(n)\}$$

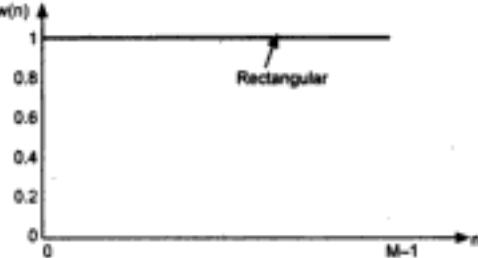
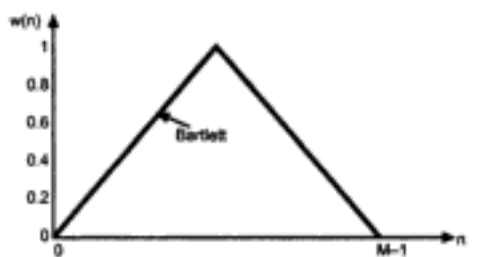
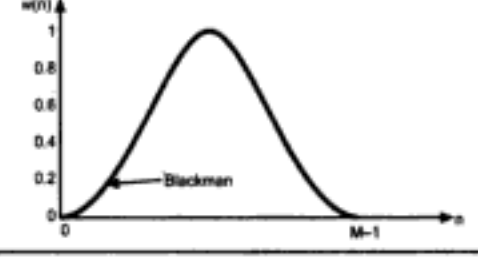
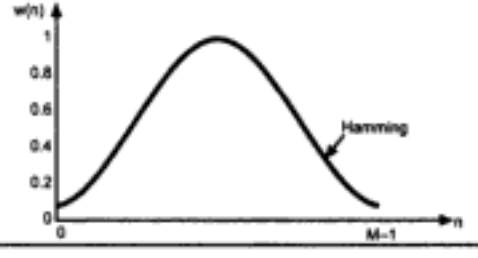
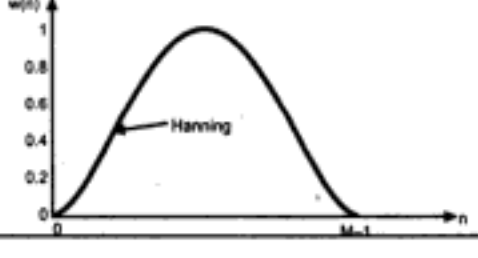
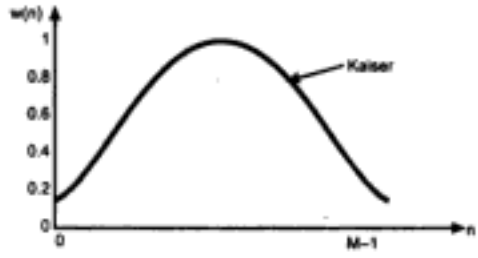
We know that fourier transform of multiplication of two signals is equal to convolution of their individual fourier transforms. Then above equation becomes,

$$H(\omega) = H_d(\omega) * W(\omega) \quad \dots (6.9.12)$$

This equation show that the response of FIR filter is equal to convolution of desired frequency response with that of window function. Because of convolution, $H(\omega)$ has the smoothing effect. The sidelobes of $W(\omega)$ create undesirable ringing effects in $H(\omega)$.

Hidden page

Table 6.9.1 Various window functions and their shapes

Sr. No.	Name of window	Time-domain sequence, $w(n), 0 \leq n \leq M-1$	Shape of window function
1.	Rectangular	1	
2.	Bartlett (triangular)	$1 - \frac{2 \left n - \frac{M-1}{2} \right }{M-1}$	
3.	Blackman	$0.42 - 0.5 \cos \frac{2\pi n}{M-1} + 0.08 \cos \frac{4\pi n}{M-1}$	
4.	Hamming	$0.54 - 0.46 \cos \frac{2\pi n}{M-1}$	
5.	Hanning	$\frac{1}{2} \left(1 - \cos \frac{2\pi n}{M-1} \right)$	
6.	Kaiser	$\frac{I_0 \left[\alpha \sqrt{\left(\frac{M-1}{2} \right)^2 - \left(n - \frac{M-1}{2} \right)^2} \right]}{I_0 \left[\alpha \left(\frac{M-1}{2} \right) \right]}$	

Hidden page

Hidden page

$$h_d(n) = \begin{cases} \frac{\sin(n-\tau)}{\pi(n-\tau)} & \text{for } n \neq \tau \\ \frac{1}{\pi} & \text{for } n = \tau \end{cases} \quad \dots (6.9.16)$$

Now let us determine value of τ . We know that the filter is symmetric. Hence,

$$h(n) = h(M-1-n)$$

We know that $h(n) = h_d(n) \cdot w(n)$. Hence above condition becomes,

$$h_d(n) w(n) = h_d(M-1-n) w(n)$$

$$\therefore h_d(n) = h_d(M-1-n)$$

$$\therefore \frac{\sin(n-\tau)}{\pi(n-\tau)} = \frac{\sin(M-1-n-\tau)}{\pi(M-1-n-\tau)}$$

The above condition is satisfied if,

$$-(n-\tau) = M-1-n-\tau$$

$$\therefore \tau = \frac{M-1}{2} \quad \dots (6.9.17)$$

Hence $h_d(n)$ of equation 6.9.16 becomes,

$$h_d(n) = \begin{cases} \frac{\sin\left(n - \frac{M-1}{2}\right)}{\pi\left(n - \frac{M-1}{2}\right)} & \text{for } n \neq \frac{M-1}{2} \\ \frac{1}{\pi} & \text{for } n = \frac{M-1}{2} \end{cases} \quad \dots (6.9.18)$$

With $M=7$ above equation becomes,

$$h_d(n) = \begin{cases} \frac{\sin(n-3)}{\pi(n-3)} & \text{for } n \neq 3 \\ \frac{1}{\pi} & \text{for } n = 3 \end{cases} \quad \dots (6.9.19)$$

Table 6.9.2 shows the values of $h_d(n)$ calculated according to above equation.

Table 6.9.2 Calculation of $h_d(n)$

n	Value of coefficient $h_d(n)$ according to equation 6.9.19
0	$h_d(0) = \frac{\sin(-3)}{-3\pi} = 0.01497$
1	$h_d(1) = \frac{\sin(-2)}{-2\pi} = 0.14472$
2	$h_d(2) = \frac{\sin(-1)}{-\pi} = 0.26785$

n	Value of coefficient $h_d(n)$ according to equation 6.9.19
3	$h_d(3) = \frac{1}{\pi} = 0.31831$
4	$h_d(4) = \frac{\sin(1)}{\pi} = 0.26785$
5	$h_d(5) = \frac{\sin(2)}{2\pi} = 0.14472$
6	$h_d(6) = \frac{\sin(3)}{3\pi} = 0.01497$

To obtain $h(n)$ by windowing :

Here observe that we have calculated only 7 values of $h_d(n)$. However according to equation 6.9.19 we can calculate infinite values of $h_d(n)$. Since we are using a window of 7 values, we will need only seven values of $h_d(n)$. i.e.,

$$h(n) = h_d(n) \cdot w(n)$$

We know that for rectangular window,

$$w(n) = \begin{cases} 1 & \text{for } 0 \leq n \leq 6 \\ 0 & \text{otherwise} \end{cases}$$

Hence,

$$h(n) = \begin{cases} h_d(n) & \text{for } 0 \leq n \leq 6 \\ 0 & \text{otherwise} \end{cases}$$

Hence from Table 6.9.2, coefficients of FIR filter become,

$$h(0) = 0.01497$$

$$h(1) = 0.14472$$

$$h(2) = 0.26785$$

$$h(3) = 0.31831$$

$$h(4) = 0.26785$$

$$h(5) = 0.14472$$

$$h(6) = 0.01497$$

This is the unit sample response of required FIR filter. Since the filter is symmetric, above coefficients satisfy the condition of $h(n) = h(M-1-n)$ for $M=7$. i.e.

$$h(n) = h(6-n)$$

i.e. $h(0) = h(6)$

$$h(1) = h(5)$$

$$h(2) = h(4)$$

Ex.6.9.2 Design the FIR filter of example 6.9.1 using hanning window.

Sol. : To calculate values of $w(n)$:

From Table 6.9.1 observe that hanning window is given by following equation :

$$w(n) = \frac{1}{2} \left(1 - \cos \frac{2\pi n}{M-1} \right) \quad \dots (6.9.20)$$

Hidden page

n	$h_d(n)$ (Table 6.9.2)	$w(n)$ (Table 6.9.3)	$h(n) = h_d(n) \cdot w(n)$
4	0.26785	$\frac{3}{4}$	$h(4) = 0.20089$
5	0.14472	$\frac{1}{4}$	$h(5) = 0.03618$
6	0.01497	0	$h(6) = 0$

In the above table also observe that unit sample response is symmetric and it satisfies the condition of $h(n) = h(M-1-n)$.

Comments :

In example 6.9.1 we calculated value of τ as,

$$\tau = \frac{M-1}{2}$$

The $H_d(\omega)$ of equation 6.9.13 can also be given as,

$$H_d(\omega) = \begin{cases} 1 \cdot e^{-j\omega \left(\frac{M-1}{2}\right)} & \text{for } |\omega| \leq \omega_c \\ 0 & \text{otherwise} \end{cases} \quad \dots (6.9.21)$$

The time shift property of fourier transform states that,

$$F.T. \{x(n-k)\} = e^{-j\omega k} X(\omega)$$

This means multiplication by $e^{-j\omega k}$ to the fourier transform is equivalent to delaying the time domain sequence by 'k' units. Hence in equation 6.9.21 observe that $H_d(\omega)$ has magnitude of 1 for $|\omega| < \omega_c$. And multiplication by $e^{-j\omega \left(\frac{M-1}{2}\right)}$ indicates delaying $h_d(n)$ by $\frac{M-1}{2}$ samples. This is clear from equation 6.9.18 also.

Hence we can write magnitude response $|H_d(\omega)|$ and phase $\angle H_d(\omega)$ from equation 6.9.21 as,

$$|H_d(\omega)| = \begin{cases} 1 & \text{for } |\omega| \leq \omega_c \\ 0 & \text{otherwise} \end{cases}$$

and
$$\angle H_d(\omega) = -\omega \left(\frac{M-1}{2}\right)$$

Fig. 6.9.4 shows the magnitude and phase response of this desired filter.

Please refer Fig. 6.9.4 (a) and (b) on next page.

The above figure shows that $H_d(\omega)$ represents ideal lowpass filter. Thus in example 6.9.1 we have designed ideal lowpass filter using rectangular window. But because of windowing (i.e. truncation of $h_d(n)$), the frequency response of FIR filter i.e. $H(\omega)$ is no more ideal.

Important :

Here we started from desired frequency response $H_d(\omega)$. Then we obtained $h_d(n)$ by inverse fourier transform of $H_d(\omega)$. Then $h(n)$ is obtained by windowing of $h_d(n)$. This is also called as FIR filter design using *inverse fourier transform*, since $h_d(n)$ is obtained by inverse fourier transform. But windowing is more commonly used name for this method.

Hidden page

Hence equation 6.9.26 becomes,

$$H(M - k) e^{j 2\pi n(M-k)/M} = H(M - k) e^{-j 2\pi kn/M}$$

Normally $|H(M - k)| = |H(k)|$, this can be easily verified from theory of DFT. This relation is based on the fact that magnitude of DFT from 0 to π is same as that from π to 2π . Hence we can write above term as,

$$H(M - k) e^{j 2\pi n(M-k)/M} = H(k) e^{-j 2\pi kn/M} \quad \dots (6.9.27)$$

The term $H(k) e^{-j 2\pi kn/M}$ is complex conjugate of $H(k) e^{j 2\pi kn/M}$. Hence $H(M - k) e^{j 2\pi n(M-k)/M}$ is complex conjugate of $H(k) e^{j 2\pi kn/M}$. i.e.,

$$H(M - k) = H^*(k) \quad \dots (6.9.28)$$

Using this relation of complex conjugate terms, the equation 6.9.25 is simplified to,

$$h(n) = \frac{1}{M} \left\{ H(0) + 2 \sum_{k=1}^P \text{Re} \left[H(k) e^{j 2\pi kn/M} \right] \right\} \quad \dots (6.9.29)$$

Here
$$P = \begin{cases} \frac{M-1}{2} & \text{if } M \text{ is odd} \\ \frac{M}{2} - 1 & \text{if } M \text{ is even} \end{cases} \quad \dots (6.9.30)$$

This equation is obtained by combining complex conjugate terms in equation 6.9.25. The above equation can be used to compute coefficients of FIR filter.

6.9.3 Design of Optimum Equiripple Linear Phase FIR Filters

Earlier we have discussed windows and frequency sampling methods for designing FIR filters. In those methods the critical frequencies ω_p and ω_s cannot be controlled precisely. The method discussed in this section is based on the chebyshev approximation. The weighted approximation error between desired frequency response and actual frequency response is spread evenly across the passband and stopband to minimize the maximum error. Ripples are introduced in the response of filter in passband and stopband.

Let $H_r(\omega)$ represents the real valued frequency response characteristic. From Fig. 5.9.3 we can write the frequency response specifications for lowpass filter as,

$$1 - \delta_1 \leq H_r(\omega) \leq 1 + \delta_1 \quad |\omega| \leq \omega_p \quad \dots (6.9.31)$$

and
$$-\delta_2 \leq H_r(\omega) \leq \delta_2 \quad |\omega| > \omega_s \quad \dots (6.9.32)$$

Now depending upon the symmetric/antisymmetric and odd/even length, there are four cases possible. Table 6.9.5 shows the relations for $H_r(\omega)$ and $h(n)$ for these four cases. In the table $H_r(\omega)$ is expressed as,

$$H_r(\omega) = Q(\omega) \cdot P(\omega) \quad \dots (6.9.33)$$

Table 9.6.5 $H_r(\omega) = Q(\omega) P(\omega)$ and corresponding $h(n)$ for linear phase FIR filters

	Filter type	$Q(\omega)$	$P(\omega)$	Relationship of $h(k)$ with $a(k)$, $\tilde{b}(k)$, $\tilde{c}(k)$ and $\tilde{d}(k)$
1.	$h(n) = h(M-1-n)$ M odd	1	$\sum_{k=0}^{(M-1)/2} a(k) \cos \omega k$	$a(k) = \begin{cases} h\left(\frac{M-1}{2}\right), & k=0 \\ 2h\left(\frac{M-1}{2}-k\right), & k=1, 2, \dots, \frac{M-1}{2} \end{cases}$

2.	$h(n) = h(M-1-n)$ M even	$\cos \frac{\omega}{2}$	$\sum_{k=0}^{\frac{M-1}{2}} \tilde{b}(k) \cos \omega k$	$\tilde{b}(0) = \frac{1}{2} b(1)$ $\tilde{b}(k) = 2b(k) - \tilde{b}(k-1), k=1, 2, \dots, \frac{M}{2} - 2$ $b(k) = 2h\left(\frac{M}{2} - k\right), k=1, 2, \dots, \frac{M}{2}$
3.	$h(n) = -h(M-1-n)$ M odd	$\sin \omega$	$\sum_{k=0}^{\frac{(M-3)}{2}} \tilde{c}(k) \cos \omega k$	$\tilde{c}(k-1) - \tilde{c}(k+1) = 2c(k) \quad 2 \leq k \leq \frac{M-5}{2}$ $\tilde{c}(0) + \frac{1}{2} \tilde{c}(2) = c(1)$ $c(k) = 2h\left(\frac{M-1}{2} - k\right), k=1, 2, \dots, \frac{M-1}{2}$
4.	$h(n) = -h(M-1-n)$ M even	$\sin \frac{\omega}{2}$	$\sum_{k=0}^{\frac{M-1}{2}} \tilde{d}(k) \cos \omega k$	$\tilde{d}(k-1) - \tilde{d}(k) = 2d(k) \quad 2 \leq k \leq \frac{M}{2} - 1$ $\tilde{d}(0) - \frac{1}{2} \tilde{d}(1) = d(1)$ $d(k) = 2h\left(\frac{M}{2} - k\right), k=1, 2, \dots, \frac{M}{2}$

In the above table observe that $P(\omega)$ has the common form,

$$P(\omega) = \sum_{k=0}^L \alpha(k) \cos \omega k \quad \dots (6.9.34)$$

Here $\alpha(k)$ represents parameters of the filter which are related to unit sample response $h(n)$ as shown in Table 6.9.5. The upper limit 'L' in the summation of above equation depends upon various cases given in Table 6.9.5. Let us denote real valued desired frequency response by $H_{dr}(\omega)$. The value of $H_{dr}(\omega)$ will be simple unity in the passband and zero in the stopband. Let us denote the weighting function on approximation error be $W(\omega)$. It can be defined as,

$$W(\omega) = \begin{cases} \delta_2 & \omega \text{ in passband} \\ \delta_1 & \omega \text{ in stopband} \end{cases} \quad \dots (6.9.35)$$

The weighted approximation error is given as,

$$E(\omega) = W(\omega)[H_{dr}(\omega) - H_r(\omega)] \quad \dots (6.9.36)$$

The relative size of errors is decided by function $W(\omega)$. Since $H_r(\omega) = Q(\omega)P(\omega)$ above equation call be written as,

$$\begin{aligned} E(\omega) &= W(\omega)[H_{dr}(\omega) - Q(\omega)P(\omega)] \\ &= W(\omega)Q(\omega)\left[\frac{H_{dr}(\omega)}{Q(\omega)} - P(\omega)\right] \end{aligned}$$

Here let $\hat{W}(\omega) = W(\omega)Q(\omega)$ and $\hat{H}_{dr}(\omega) = \frac{H_{dr}(\omega)}{Q(\omega)}$ then the above equation becomes,

Hidden page

Hidden page

Hidden page

Let the cascade configuration shown in above figure be excited by unit sample sequence $\delta(n)$. The response of $H_d(z)$ to $\delta(n)$ is $h_d(n)$ as expected and $h_d(n)$ becomes an input to $\frac{1}{H(z)}$ as shown in above figure. The output of this inverse filter is shown as $y(n)$. Actually if $H_d(z)$ and $H(z)$ are same, then $y(n) = \delta(n)$. This is because filtering and inverse filtering in cascade has no effect and output is same as input. Note that $H_d(z)$ is the desired filter and $H(z)$ is the approximation of $H_d(z)$ obtained through design process. From equation (6.10.1), $\frac{1}{H(z)}$ will be,

$$\frac{1}{H(z)} = \frac{1 + \sum_{k=1}^N a_k z^{-k}}{b_0}$$

The input to this filter is $h_d(n)$, hence its output $y(n)$ is given as,

$$b_0 y(n) = h_d(n) + \sum_{k=1}^N a_k h_d(n-k)$$

$$y(n) = \frac{1}{b_0} \left[h_d(n) + \sum_{k=1}^N a_k h_d(n-k) \right] \quad \dots (6.10.2)$$

It is desired that $y(n) = \delta(n)$. We know that $\delta(0) = 1$, hence $y(0) = 1$. With $n = 0$, above equation becomes,

$$y(0) = \frac{1}{b_0} \cdot h_d(0)$$

$$\therefore 1 = \frac{1}{b_0} h_d(0) \cdot \text{since } y(0) = 1$$

Hence, $b_0 = h_d(0) \quad \dots (6.10.3)$

In Fig. 6.10.1 observe that 'error' is obtained by subtraction of $y(n)$ and $\delta(n)$. i.e.

$$\text{Error} = y(n) - \delta(n)$$

For $n > 0$, $\delta(n) = 0$ and $\text{Error} = y(n)$. Let the sum of squares of the error sequence be denoted by ϵ i.e.,

$$\epsilon = \sum_{n=1}^{\infty} y^2(n) \quad \dots (6.10.4)$$

Putting the value of $y(n)$ from equation (6.10.2) above,

$$\epsilon = \sum_{n=1}^{\infty} \left\{ \frac{1}{b_0} \left[h_d(n) + \sum_{k=1}^N a_k h_d(n-k) \right] \right\}^2$$

Since $b_0 = h_d(0)$, above equation becomes,

$$\epsilon = \frac{\sum_{n=1}^{\infty} \left[h_d(n) + \sum_{k=1}^N a_k h_d(n-k) \right]^2}{h_d^2(0)} \quad \dots (6.10.5)$$

Hidden page

Fig. 6.11.1 shows the poles placement.

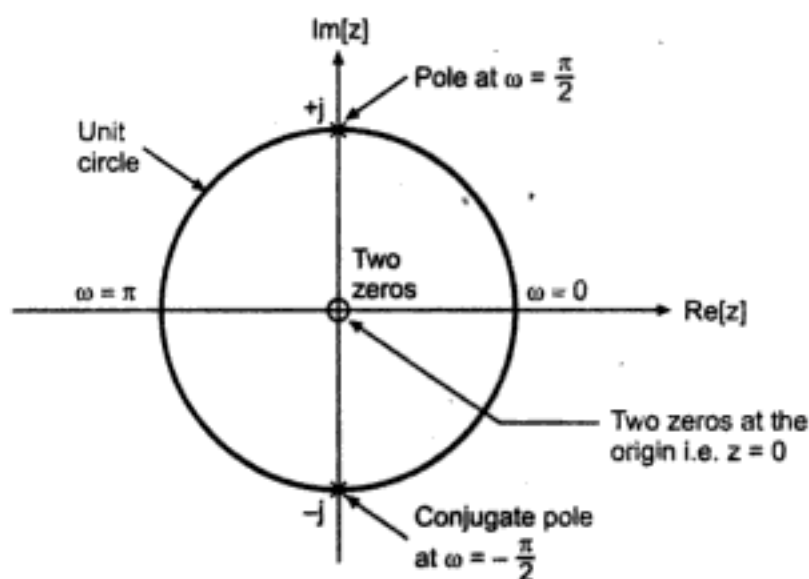


Fig. 6.11.1 Poles-zero placement for bandpass filter with frequency of $\frac{\pi}{2}$

In the above Fig. 6.11.1 observe that one pole is placed at $\omega = \frac{\pi}{2}$ (i.e. $z = 0 + j$). The filter will be realizable if all the poles and zeros occur in complex conjugate pairs. Hence we should place conjugate pole at $z = 0 - j$ i.e. at $\omega = -\frac{\pi}{2}$. Hence this pole is placed at $\omega = -\frac{\pi}{2}$ (i.e. $\omega = \frac{3\pi}{2}$). To make the system physically realizable, the number of poles and zeros should be equal. Since we have placed two poles, we should also place two zeros. Normally such zeros are placed at the origin i.e. $z = 0$. These zeros do not affect the frequency response since they are placed at $z = 0$. Thus we have poles and zeros of the bandpass filter as follows.

poles at $z = 0 + j$ and $0 - j$

zeros at $z = 0$ (Two zeros).

To determine system function :

Now the system function can be obtained from the poles and zeros as follows :

$$H(z) = \frac{(z - z_1)(z - z_2)}{(z - p_1)(z - p_1^*)}$$

Here

$$p_1 = 0 + j, \quad p_1^* = 0 - j$$

$$z_1 = z_2 = 0$$

\therefore

$$\begin{aligned} H(z) &= \frac{(z - 0)(z - 0)}{(z - j)(z + j)} \\ &= \frac{z^2}{z^2 + 1} \end{aligned}$$

The above system function can also be written as,

$$H(z) = \frac{1}{1+z^{-2}} \quad \dots (6.11.1)$$

This is the system function of digital filter. It can be implemented by direct form.

To determine frequency response of $H(z)$:

We know that frequency response of $H(z)$ can be obtained by putting $z = e^{j\omega}$, i.e.,

$$\begin{aligned} H(\omega) &= H(z) \Big|_{z=e^{j\omega}} \\ &= \frac{1}{1+e^{-j2\omega}} \quad \text{from equation 6.11.1} \end{aligned}$$

Let us rearrange this equation as follows :

$$\begin{aligned} H(\omega) &= \frac{1}{e^{-j\omega} \cdot e^{j\omega} + e^{-j\omega} \cdot e^{-j\omega}} \\ &= \frac{1}{e^{-j\omega} (e^{j\omega} + e^{-j\omega})} \\ &= \frac{e^{j\omega}}{e^{j\omega} + e^{-j\omega}} \end{aligned}$$

Here use $e^{j\theta} + e^{-j\theta} = 2 \cos \theta$. Hence,

$$H(\omega) = \frac{e^{j\omega}}{2 \cos \omega} = \frac{1}{2 \cos \omega} \cdot e^{j\omega}$$

Hence magnitude response of $H(\omega)$ becomes,

$$|H(\omega)| = \left| \frac{1}{2 \cos \omega} \right|$$

Table 6.11.1 below shows $|H(\omega)|$ calculated for various values of ω .

Table 6.11.1 Magnitude response of $H(\omega)$ calculated at few values of ω

ω	$ H(\omega) = \left \frac{1}{2 \cos \omega} \right $
0	$\frac{1}{2}$
$\frac{\pi}{6}$	0.577
$\frac{2\pi}{6} = \frac{\pi}{3}$	1
$\frac{3\pi}{6} = \frac{\pi}{2}$	∞

Hidden page

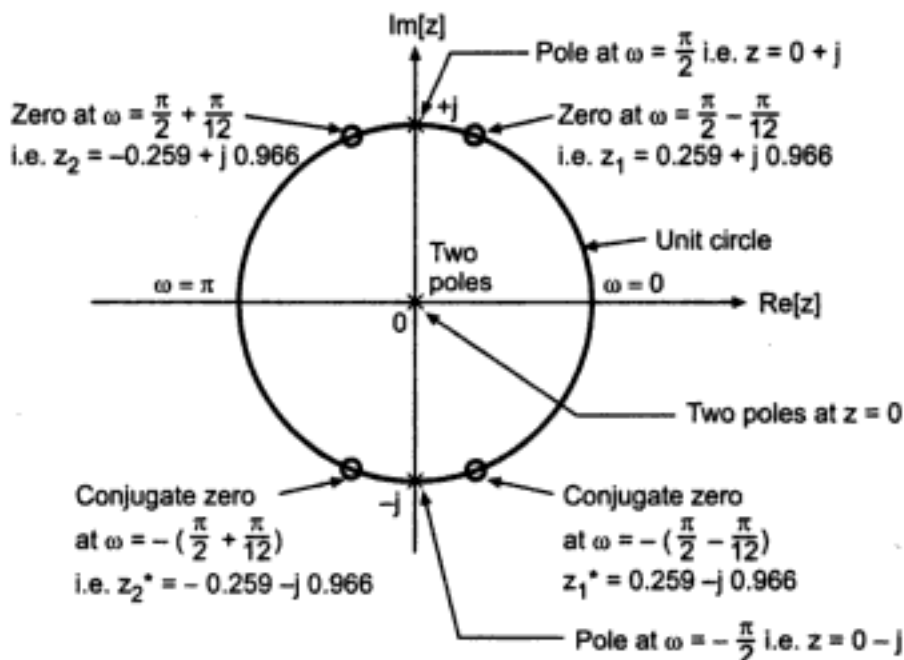


Fig. 6.11.3 Pole zero placement for sharp passband

In the above figure observe that we have put two zeros at $\omega = \frac{\pi}{2} \pm \frac{\pi}{12}$. These zeros will attenuate frequencies near $\omega = \frac{\pi}{2}$. Since poles and zero should have their complex conjugates, we have to put two zeros at $\omega = -\left(\frac{\pi}{2} \pm \frac{\pi}{12}\right)$. Thus there are total four zeros. To make poles and zeros equal for realizable system, we have to place two poles at origin i.e. at $z = 0$. Thus the poles and zeros of the system are as follows :

- Poles :**
- $$p_1 = p_2 = 0$$
- $$p_3 = 0 + j \quad , \quad p_3^* = 0 - j$$
- Zeros :**
- $$z_1 = 0.259 + j 0.966 \quad , \quad z_1^* = 0.259 - j 0.966$$
- $$z_2 = -0.259 + j 0.966 \quad , \quad z_2^* = -0.259 - j 0.966$$

System function for this new pole-zero placement can be obtained as,

$$H(z) = \frac{(z - z_1)(z - z_1^*)(z - z_2)(z - z_2^*)}{(z - p_1)(z - p_2)(z - p_3)(z - p_3^*)}$$

Putting for poles and zeros in above equation we get,

$$H(z) = \frac{(z - 0.259 - j 0.966)(z - 0.259 + j 0.966)(z + 0.259 - j 0.966)(z + 0.259 + j 0.966)}{(z - 0)(z - 0)(z - j)(z + j)}$$

$$= \frac{z^4 + 1.732z^2 + 1}{z^4 + z^2}$$

$$\therefore H(z) = \frac{1 + 1.732z^{-2} + z^{-4}}{1 + z^{-2}}$$

This is the system function of the required sharp bandpass filter. The frequency response can be obtained by evaluating at $z = e^{j\omega}$.

Important note :

Poles are normally placed slightly inside the unit circle. We know that digital filters are unstable if poles are outside the unit circle. Hence to avoid this problem poles are placed inside the unit circle. This do not affect the frequency response, since it depends on the angle of the pole. Zeros can be placed anywhere since they do not affect stability of the filters.

Ex.6.11.2 Design a lowpass filter to have cutoff frequency of 250 Hz using pole zero combination. The sampling frequency is 2000 Hz.

Sol. : To determine frequency specifications :

Cutoff frequency of lowpass filter, $F_c = 250$ Hz

Sampling frequency, $F_s = 2000$ Hz

Hence corresponding discrete frequencies will be,

$$\begin{aligned} f_c &= \frac{F_c}{F_s} \\ &= \frac{250}{2000} = \frac{1}{8} \text{ cycles/sample} \end{aligned}$$

\therefore

$$\begin{aligned} \omega_c &= 2\pi f_c \\ &= 2\pi \cdot \frac{1}{8} = \frac{\pi}{4} \text{ radians/sample.} \end{aligned}$$

Pole-zero placement :

We want the frequencies above $\frac{\pi}{4}$ to be blocked. Hence we will place one zero at $\omega = \frac{\pi}{4}$.

Then complex conjugate zero will occur at $\omega = -\frac{\pi}{4}$. This zero at $\omega = \frac{\pi}{4}$ will attenuate

frequencies near $\omega = \frac{\pi}{4}$. Hence we will put one pole at $\omega = \frac{\pi}{4} - \frac{\pi}{36}$ to reduce attenuation of

frequencies of $\omega < \omega_c$. Its complex conjugate pole will be present at $\omega = -\left(\frac{\pi}{4} - \frac{\pi}{36}\right)$ Fig. 6.11.4

shows the pole zero placement.

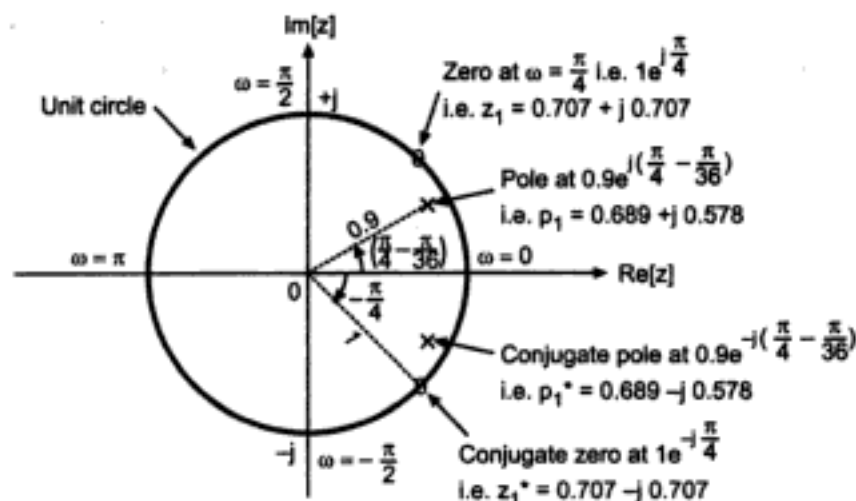


Fig. 6.11.4 Pole-zero placement for a lowpass filter

Hidden page

Sr. No.	Parameter or characteristic	IIR filters	FIR filters
4.	Phase characteristic	Nonlinear phase response.	Linear phase response. (for symmetric $h(n)$).
5.	Stability of the filter	These filters are to be designed for stability.	These are inherently stable filters.
6.	Number of multiplications required	Less.	More.
7.	Complexity of implementation	More.	Less.
8.	Memory requirement	Less memory is required.	More memory is required.
9.	Can simulate prototype analog filters	Yes.	No.
10.	Order of filter for similar specifications	Requires lower order.	Requires higher order.
11.	Availability of design softwares	Good.	Very good.
12.	Design procedure	Complicated.	Less complicated.
13.	Processing time	Less time is required.	More time is required.
14.	Design methods	(i) Bilinear transform. (ii) Impulse invariance.	(i) Windowing. (ii) Frequency sampling.
15.	Applications	Can be used where sharp cutoff characteristics with minimum order are required.	Used where linear phase characteristic is essential.

The selection between FIR and IIR filter is based on following criteria :

- (i) If linear phase requirement is critical then FIR filters are used.
- (ii) If sharp cutoff characteristic with minimum order is required, then IIR filters are required.

Thus IIR filters are used in maximum number of applications when phase characteristic is not very important. For the design of FIR as well as IIR filters good number of standard softwares are available.

Till now we have not discussed about finite wordlength effects. IIR and FIR filters can be compared on the basis of these effects also. FIR filters are least affected due to finite wordlength effects, whereas IIR filters are affected more.

6.13 Butterworth Filter Design Using Bilinear Transformation

In this section we will see about how to compute the system function of a butterworth using bilinear transformation.

6.13.1 Design Steps

Here we will summarize the steps used for design of butterworth filter using bilinear transformation.

Assumption : Assume that specifications of digital filter are given. These specifications are passband and stopband attenuations i.e. A_p and A_s ; passband and stopband edge frequencies i.e. ω_p and ω_s .

Step 1 : Redefine the specifications for equivalent analog filter by using bilinear frequency relationship i.e.,

$$\Omega = \frac{2}{T} \tan \frac{\omega}{2}$$

Here $\frac{2}{T}$ can be assumed to be 1.

Step 2 : Calculate the order 'N' of the filter by using following relation.

$$N = \frac{1}{2} \frac{\log \left[\left(\frac{1}{A_s^2} - 1 \right) / \left(\frac{1}{A_p^2} - 1 \right) \right]}{\log \left(\frac{\Omega_s}{\Omega_p} \right)}$$

or

$$N = \frac{1}{2} \frac{\log \left[\frac{10^{0.1 A_s \text{ dB}} - 1}{10^{0.1 A_p \text{ dB}} - 1} \right]}{\log \left(\frac{\Omega_s}{\Omega_p} \right)}$$

Step 3 : Determine cutoff frequency of equivalent analog filter i.e. Ω_c . It is given by following bilinear relationship.:

$$\Omega_c = \tan \frac{\omega_c}{2}$$

Here note that we have assumed $\frac{2}{T} = 1$.

Step 4 : Calculate poles of $H_a(s)$ by the following equation,

$$p_k = \Omega_c e^{j(N+2k+1)\pi/2N}, \quad k = 0, 1, 2, \dots, N-1$$

Step 5 : Organize the poles p_k as complex conjugate pairs. i.e.,

$$s_1 \text{ and } s_1^*, s_2 \text{ and } s_2^*, s_3 \text{ and } s_3^*, \dots$$

Step 6 : Compute system function of analog filter by following equation

$$H_a(s) = \frac{\Omega_c^N}{(s-s_1)(s-s_1^*)(s-s_2)(s-s_2^*) \dots}$$

Step 7 : System function of digital filter by bilinear transformation can be obtained as,

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

Hidden page

Putting $s = \frac{1-z^{-1}}{1+z^{-1}}$ in equation 6.13.4 we get,

$$H_1(z) = \frac{\Omega_c^2}{\left(\frac{1-z^{-1}}{1+z^{-1}}\right)^2 + 2 \text{pReal} \left(\frac{1-z^{-1}}{1+z^{-1}}\right) + (\text{pReal})^2 + (\text{pImag})^2}$$

$$= \frac{\Omega_c^2 (1+z^{-1})^2}{(1-z^{-1})^2 + 2 \text{pReal} (1-z^{-1})(1+z^{-1}) + [(\text{pReal})^2 + (\text{pImag})^2] (1+z^{-1})^2}$$

The above equation can be simplified as follows :

$$= B_1 \frac{b_{00} + b_{01} z^{-1} + b_{02} z^{-2}}{a_{00} + a_{01} z^{-1} + a_{02} z^{-2}} \quad \dots (6.13.5)$$

Here,

$$B_1 = \frac{\Omega_c^2}{1 + 2 \text{pReal} + (\text{pReal})^2 + (\text{pImag})^2} \quad \dots (6.13.6)$$

$$\left. \begin{aligned} b_{00} &= 1 \\ b_{01} &= 2 \\ b_{02} &= 1 \end{aligned} \right\} \quad \dots (6.13.7)$$

$$\left. \begin{aligned} a_{00} &= 1 \\ a_{01} &= \frac{2 [(\text{pReal})^2 + (\text{pImag})^2] - 2}{1 + 2 \text{pReal} + (\text{pReal})^2 + (\text{pImag})^2} \\ a_{02} &= \frac{1 - 2 \text{pReal} + (\text{pReal})^2 + (\text{pImag})^2}{1 + 2 \text{pReal} + (\text{pReal})^2 + (\text{pImag})^2} \end{aligned} \right\} \quad \dots (6.13.8)$$

Similarly other second order sections of $H_a(s)$ of equation 6.13.2 can be converted to equivalent digital filter section. i.e.,

$$H(z) = B_1 \frac{b_{00} + b_{01} z^{-1} + b_{02} z^{-2}}{a_{00} + a_{01} z^{-1} + a_{02} z^{-2}} \times B_2 \frac{b_{10} + b_{11} z^{-1} + b_{12} z^{-2}}{a_{10} + a_{11} z^{-1} + a_{12} z^{-2}} \times \dots \quad \dots (6.13.9)$$

$$= \prod_{k=1}^{N/2} B_k \frac{b_{k0} + b_{k1} z^{-1} + b_{k2} z^{-2}}{a_{k0} + a_{k1} z^{-1} + a_{k2} z^{-2}} \quad \text{for even N} \quad \dots (6.13.10)$$

Now let us consider $H_a(s)$ with odd N. It will contain $\left(\frac{N-1}{2}\right)$ second order sections and one first order section. Then equation 6.13.2 can be written as,

$$H_a(s) = \frac{\Omega_c^2}{(s-s_1)(s-s_1^*)} \times \frac{\Omega_c^2}{(s-s_2)(s-s_2^*)} \times \dots \times \frac{\Omega_c}{s + \Omega_c} \quad \dots (6.13.11)$$

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

$$\left. \begin{aligned} b_0 &= 1 \\ b_1 &= -2 \operatorname{Real}Z \\ b_2 &= (\operatorname{Real}Z)^2 + (\operatorname{Imag}Z)^2 \\ a_1 &= -2 \operatorname{Real}P \\ a_2 &= (\operatorname{Real}P)^2 + (\operatorname{Imag}P)^2 \end{aligned} \right\} \dots (6.15.8)$$

Thus the coefficients for two poles and two zeros are obtained. This logic can be extended for higher number of poles and zeros.

6.15.2 C Program for Filter Design Using Pole-Zero Combination

Now let us consider the C program based on logic discussed in previous subsection. The program is presented below :

```
//file name : notch.cpp
/*----- Filter design using pole zero combination -----*/
//
// This program accepts the location of pole and zero
// and gives the coefficients of difference equation.
// It also shows the magnitude and phase plot of the filter.
//
// The format of the difference equation is,
//  $y(n) = -[a_1*y(n-1)+a_2*y(n-2)] + b_0*x(n)+b_1*x(n-1)+b_2*x(n-2)$ 
//
// Inputs : Angular position of pole and zero
// as position =  $r e^{j(\text{Theta})}$ 
// Outputs : 1. Values of coefficients, i.e.  $a_1, a_2, b_0, b_1$  &  $b_2$ 
// 2. Magnitude and phase transfer function
// plot  $H(e^{jw})$  for  $w = 0$  to  $\pi$ .
//
// Assumptions : 1. This program is written for second order filter.
// It can be used for lowpass, highpass, bandpass,
// notch any type of pole zero combination.
// 2. The program accepts location of one pole and one
// zero and their conjugates are considered internally
// see explanation for details)
//-----
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
void main()
{
    float rZ, ThetaZ, rP, ThetaP;
    float RealZ, ImagZ, RealP, ImagP;
    float RealNum, ImagNum, RealDen, ImagDen;
    float mag[640], phase[640], pi, w, wStep;
    float static a[10], b[10], yMag, yPhase;
    int M, N, i, k, gd, gm;

    clrscr();
    printf("\t\t\tFilter design using using pole zero combination");
    printf("\n\nEnter the angular position of pole and zero "
        "\nthe program takes their conjugates internally "
```

Hidden page

Hidden page

----- Results -----

Filter design using using pole zero combination

Enter the angular position of pole and zero
the program takes their conjugates internally

pole or zero = $r e^{j(\text{Theta})}$
location of zero, r Theta : 0 0
location of pole, r Theta : 1 1.5707963

The coefficients of the filter are as follows...

b[0] = 1.000000
b[1] = -0.000000 a[1] = -0.000000
b[2] = 0.000000 a[2] = 1.000000
press any key to see magnitude/phase response

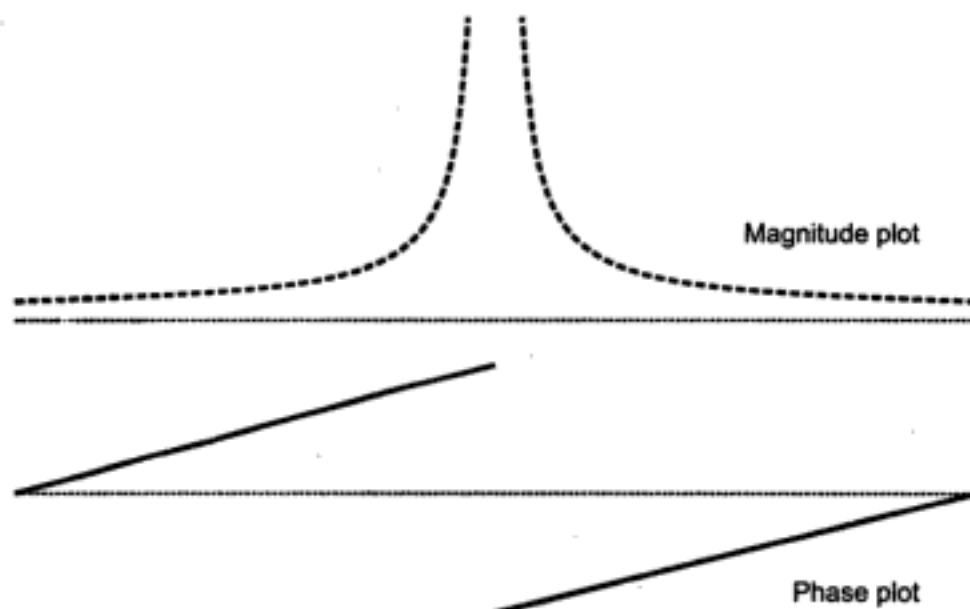


Fig. 6.15.1 Magnitude and phase plot of notch bandpass filter obtained by C program

From the above results it is clear that,

$$b_0 = 1, \quad b_1 = b_2 = 0 \text{ and} \\ a_2 = 1, \quad a_1 = 0$$

Hence the system function of equation 6.15.6 becomes,

$$H(z) = \frac{1}{1+z^{-2}}$$

This same as that we have obtained in example 6.11.1 and given by equation 6.11.1. Similarly the magnitude response computed by program as given above in Fig. 6.15.1 is similar to that of Fig. 6.11.2.

Note : This program can be used to calculate coefficients of any type of filter which has two poles and two zeros. The frequency response of the filter can be shaped by placement of poles and zeros. Thus it is also possible to have, lowpass, highpass, bandpass, bandstop, etc types of responses from this program. The only requirement that poles and zeros should be in complex conjugate pairs. This logic can be extended for higher number of poles and zeros.

Computer Exercise

1. The C program for butterworth filter design using bilinear transformation is given in section 6.13. Modify this program so that it should accept passband and stopband edge frequencies and attenuations, and compute coefficients accordingly.
2. The C program for FIR filter design using windows is presented in section 6.14. Add Blackmann window to this program.
3. Modify the C program presented in section 6.15 for higher number of poles and zeros. From this program design a notch bandstop filter to remove 50 Hz noise from the signal which is sampled at 200 Hz.

Theory Questions

1. What is the difference between analog filters and digital filters, compare them.
2. What do you understand by all pass filter ? Where it is used ?
3. Determine the unit sample response of the ideal lowpass filter. Why it is not realizable?
4. Explain the design of IIR filters by approximation of derivatives.
5. Explain the method of IIR filter design by impulse invariance.
6. Explain the Bilinear transform method of IIR filter design. What is warping effect ? Explain poles and zeros mapping procedure clearly.
7. What are the advantages of FIR filter ? Discuss the design steps of FIR filters using windows.
8. Explain in detail the design of FIR filters using rectangular window.
9. Explain FIR filter design using frequency sampling technique.
10. Explain the design of digital filters using pole-zero placement.
11. Explain the design method of optimum equiripple linear phase FIR filters
12. Explain the design of filters of filters using least squares method.
13. Write short notes on the following :
 - (i) Various smoothing window functions
 - (ii) Comparison of FIR and IIR filters
 - (iii) Gibbs phenomenon
 - (iv) Frequency transformation
 - (v) Digital filter design using butterworth approximation
 - (vi) Realization forms of digital filters
 - (vii) FIR differentiators
 - (viii) Hilbert transformers

Unsolved Examples

1. Design a highpass FIR filter having cutoff frequency $\omega_c = 2$ radians/sample and length of 7. Use rectangular smoothing window.

$$\text{Ans. : } h(0) = h(6) = 0.02965$$

$$h(1) = h(5) = 0.12045$$

$$h(2) = h(4) = -0.2894$$

$$h(3) = 0.36338$$

Hidden page

Chapter 7

HARDWARE ARCHITECTURE OF DSP (DSP PROCESSORS)

7.1 Introduction

In this chapter we will briefly introduce DSP processor architecture and their features. In the previous chapters we have implemented various DSP algorithms such as convolution, FFT, filtering, correlation etc in C. There are few things common to all DSP algorithms such as,

- (i) Processing on arrays is involved.
- (ii) Majority of operations are multiply and accumulate.
- (iii) Linear and circular shifting of arrays is required.

These operations require large time when they are implemented on general purpose processors. This is because the hardware of general purpose processors is not optimized to perform such operations fast. Hence general purpose processors are not suitable for DSP operations. Particularly, real time DSP operations are very difficult on general purpose processors. Hence DSP processors having architecture suitable for DSP operations are developed.

7.2 Desirable Features of DSP Processors

Now let us see what features DSP processors should have so that DSP operations will be performed fast.

- (i) DSP processors should have multiple registers so that data (i.e. arrays) exchange from register to register is fast.
- (ii) DSP operations require multiple operands simultaneously. Hence DSP processor should have multiple operand fetch capacity.
- (iii) DSP processors should have circular buffers to support circular shift operations.
- (iv) The DSP processor should be able to perform multiply and accumulate operations very fast.
- (v) DSP processors should have multiple pointers to support multiple operands, jumps and shifts.
- (vi) Since DSP processors can be used with general processors, they should have multi processing ability.
- (vii) To support DSP operations fast, the DSP processors should have on chip memory.
- (viii) For real time applications interrupts and timers are required. Hence DSP processors should have powerful interrupt structure and timers.

The architectures of DSP processors are designed to have these features. The DSP processors from Analog Devices, Texas Instruments, Motorola etc are commonly used.

7.3 Types of Architectures

There are three types of standard architectures for microprocessors. They are as follows :

(i) Von-Neumann Architecture :

General purpose processors normally have this type of architecture. The architecture shares same memory for program and data. The processors perform instruction fetch, decode and execute operations sequentially. In such architecture, the speed can be increased by pipelining. This type of architecture contains common internal address and data bus, ALU, accumulator, I/O devices and common memory for program and data. This type of architecture is not suitable for DSP processors.

(ii) Harvard Architecture :

The harvard architecture has separate memories for program and data. There are also separate, address and data buses for program and data. Because of these separate on chip memories and internal buses, the speed of execution in harvard architecture is high.

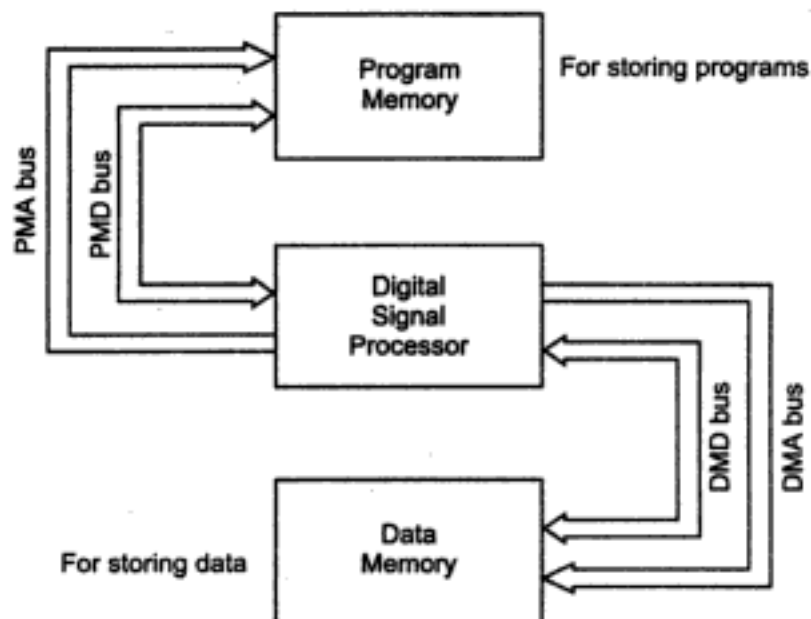


Fig. 7.3.1 Harvard architecture showing separate program and data memories

In the above figure observe that there is Program Memory Address (PMA) bus and Program Memory Data (PMD) bus separate for program memory. Similarly there is separate Data Memory Data (DMD) bus and Data Memory Address (DMA) bus for data memory. This is all on chip. The digital signal processor includes various registers, address generators, ALUs etc.

The harvard achitecture has multiple bus structure and separate memories. Hence its speed is increased. It is possible to fetch next instruction when current instruction is executed. That is, the fetch, decode and execute operations are done parallely.

(iii) Modified Harvard Architecture :

In this architecture data memory can be shared by data as well as programs. Fig. 7.3.2 illustrates this concept.

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

Hidden page

References

1. John G. Proakis and D.G. Manolakis, "Introduction to Digital Signal Processing", Macmillan Publishing Company, New York, 1989.
2. S.K. Mitra, "Digital Signal Processing", Tata McGraw-Hill Publishing Co Ltd, New Delhi, 1998.
3. Johnny Johnson, "Introduction to Digital Signal Processing", Prentice Hall of India Pvt Ltd, New Delhi, 1997.
4. Andreas Antoniou, "Digital Filters Analysis and Design", Tata McGraw-Hill Publishing Co Ltd, New Delhi, 1979.
5. Roman Kuc, "Introduction to Digital Signal Processing", McGraw-Hill Book Company, New York, 1988.
6. L.R. Rabiner and Bernard Gold, "Theory and Application of Digital Signal Processing", Prentice Hall of India Pvt Ltd, New Delhi, 1993.
7. Paul Embree and Bruce Kimble, "C Language Algorithms for Digital Signal Processing", Prentice Hall Englewood Cliffs, New Jersey, 1991.
8. Richard Lyons, "Understanding Digital Signal Processing", Addison Wesley Longman Pvt. Ltd, Singapore, 1999.
9. D.J. Defatta, Lucas and Hodgkiss, "Digital Signal Processing", John Wiley and sons, New York, 1988.
10. A.V. Oppenheim and R.W. Schaffer, "Discrete Time Signal Processing", Prentice Hall, New Jersey, 1999.
11. John G. Proakis, "Digital Communications", 3rd edition, McGraw-Hill Inc., New York, 1995.
12. William Press and others, "Numerical Recipes in C", Cambridge University Press, 1988.
13. J.N. Holmes, "Speech Synthesis and Recognition", Van Nostrand Reinhold (UK) Co Ltd, 1988.
14. B.W. Kernighan and D.M. Ritchie, "The C Programming Language", Prentice Hall of India Pvt Ltd, New Delhi, 1988.
15. Yashvant Kanetkar, "Let us C", BPB Publications, New Delhi, 1999.
16. V.R. Udpikar, "A Digital Signal Processing Training Program", Wavelet Group, Pune, 1999.
17. www.analogdevices.com.
18. www.ti.com
19. Watkins, Sadun and Marenka, "Modern Image Processing", Academic Press Inc., New York, 1993.
20. R.C. Gonzalez and R.E. Woods, "Digital Image Processing", Addison Wesley, 1992.
21. A.K. Jain, "Fundamentals of Digital Image Processing", Prentice Hall of India Pvt Ltd, New Delhi, 1997.

□□□

Index

A

A/D Conversion [119](#)
Adder [38](#)
Adding operation [36](#)
ADSP-2105 processor [528](#)
ADSP-2106x Processors [530](#)
ADSP-21xx [513](#)
ADSP-21xx Development Tools [528](#)
Advancing operation [36](#)
Aliasing [24](#), [125](#)
ALU [516](#)
ALU Instructions [531](#)
Analog Low Pass Filter [405](#)
Analog signals [13](#)
Analysis of Signals [272](#)
Anti Aliasing Filter [131](#)
Antisymmetric FIR filters [457](#)
Antisymmetric signal [32](#)
Applications of DFT [335](#)
Architecture Description (.ACH) file [529](#)
Architecture of ADSP-21xx [513](#), [524](#)
Arithmetic/Logic Unit [516](#)
Array of Values [6](#)
Assembler [529](#)
Associative property of convolution [75](#)
Auto-correlation [113](#)

B

Bandlimiting [132](#)
Barrel Shifter [519](#)
Bartlett triangular [465](#)
Bilinear Transformation [422](#)
Bit rate [132](#)
Bit Reversal [359](#), [365](#)
Blackman [465](#)
Boot Address Generator [525](#)
Boot Memory Interface [531](#)
Butterfly Computation [355](#)
Butterfly Operation [363](#)
Butterworth Approximation [425](#)
Butterworth filters [428](#)

C

Cache Memory [522](#)
Cache memory monitor [522](#)
Cascade connection [60](#), [99](#)
Cascade form realization of FIR system [251](#)
Cascade Form Structure for IIR Systems [263](#)
Causal system [47](#)
Causality [218](#)
Causality of LSI Systems [77](#)

Causality Property [47](#)
Chebyshev filter approximation [425](#)
Chirp-Z transform [347](#)
Circular Convolution [308](#)
Circular Correlation [331](#)
Circular Frequency Shift [330](#)
Circular Symmetries of a Sequence [300](#)
Circular Time Shift [327](#)
Circularly even sequence [302](#)
Circularly folded sequence [305](#)
Circularly odd sequence [304](#)
Classification of Discrete Time Systems [40](#)
Classification of Signals [12](#)
Clock Signals [528](#)
Commutative property of convolution [74](#)
Comparison of Analog and Digital Filters [407](#)
Comparison of FIR and IIR Filters [485](#)
Complex Conjugate Properties [330](#)
Computation of DIT FFT Algorithm [377](#)
Computation of Fourier Transform [373](#)
Computation of z-transform [229](#)
Computational Complexity [357](#), [365](#)
Conjugation of a Complex Sequence [165](#)
Constant multiplier [39](#)
Continuous amplitude signals [12](#)
Continuous Time Signals [6](#)
Contour integration [181](#)
Convolution [66](#)
Convolution in Time Domain [162](#)
Correlation [113](#)
Correlation of Two Sequences [163](#)
Cross-correlation [113](#)

D

DAGs [521](#)
Data Address Generators [521](#)
Data Memory Interface [530](#)
Data Memory Map [530](#)
Data Move Instruction [526](#)
Decimation in Time [348](#)
Delay operation [35](#)
Deterministic signals [13](#)
Development tools [528](#)
DFT [289](#)
DFT for Linear Filtering [335](#)
DFT of Standard Signals [298](#)
DIF FFT Algorithm [360](#)
Difference Equations [107](#)
Differentiation in z-domain [161](#)
Digital filters [246](#)
Digital Lowpass Filter [405](#)
Digital signals [12](#)
Direct form realization of FIR system [250](#)

Direct form-I structure of IIR system 256
Direct form-II structure for IIR system 258
Discrete amplitude signals [12](#)
Discrete Fourier Transform 272, 289
Discrete Time Signals [7](#)
Discrete Time Systems [33](#)
Distributive property of convolution [76](#)
DIT-FFT Algorithm 348
DMA bus 515
DSP Processors [511](#)
Dynamic system [42](#)
Dynamicity Property [41](#)

E

Elements of Digital Signal Processing 1
Elliptic filter approximation [425](#)
Encoding [127](#)
Energy Signals [31](#)
Even or symmetric signal [32](#)
Exponential Sequence [29](#)

F

Fast Fourier Transform 345
Features of ADSP-21xx 522
Features of DSP Processors [511](#)
FFT 345
Filter Approximation 425
FIR Filter [408](#)
FIR Filter Design [461](#)
FIR Filter Design Using Windows [461](#)
FIR Filters 456
FIR system [108](#)
Folding [68](#)
Folding operation [33](#)
Fourier Transform 272
Frequency Concept [14](#)
Frequency Relationships [121](#)
Frequency Transformations 448
frequency warping [423](#)
FT 272

G

Gibbs Phenomenon 464
Goertzel Algorithm [365](#)
Group delay 412

H

Hamming [465](#)
Hanning [465](#)
Harvard Architecture [512](#)

I

Ideal Filter Characteristics 410

IDFT 289
IIR Filter [409](#)
IIR system [108](#)
Impulse Invariance [417](#)
Initial Value Theorem [168](#)
Inplace Computation 358, [365](#)
Instruction cache 522
Instruction Set of ADSP-21xx [531](#)
Interrupts 525
Inverse Fourier Transform 277
Inverse z-transform [181](#)

K

Kaiser [465](#)

L

Linear Convolution [66](#)
Linear Phase in FIR Filters 457
Linear system [46](#)
Linearity 159, 300
Linearity Property [45](#)
Linker 530
LSI systems [63](#)

M

MAC 517
MAC Instructions 526
Memory map 529
Modified Harvard Architecture [512](#)
Multichannel signals [13](#)
Multidimensional signals [13](#)
Multifunction Instructions 526
Multiplication [68](#)
Multiplication methods [89](#)
Multiplication of Two Sequences [164](#), [332](#)
Multiplier-Accumulator 517

N

Negative frequencies [15](#)
Noncausal Systems [47](#)
Nonlinear Systems [45](#)
Non-periodic Signals [31](#)
Nonrecursive Systems [109](#)
Normalized filter 448
Nyquist interval [130](#)
Nyquist rate [130](#)

O

Odd Signals [31](#)
OR/PASS logic 519
Overlap Add Method [341](#)
Overlap Save Method [340](#)

P

Parallel Form Structure 264
Parseval's Relation [167](#)
Parseval's Theorem [333](#)
Partial Fraction Expansion
Passband edge frequency 428
Passband edge value 431
Periodic Signals [31](#)
Periodicity 299
Periodicity property of W_N [291](#), 346
Phase response [286](#)
Pin Definitions 526
PMA bus 521
PMD bus 515
Pole-zero Plots [211](#)
Postfilter [132](#)
Power Series Expansion 197
Power Signals [31](#)
Prefilter [132](#)
Prewarping [439](#)
Program Flow Instructions 526
Program Memory Interface 529
Program Memory Maps 529
Program Sequencer [521](#)
Properties of Convolution [74](#)
Properties of correlation [118](#)
Properties of DFT [299](#)
Properties of Fourier Transform 275
Properties of W_N 345
Properties of z-transform 159
Prototype lowpass filter 452

Q

Quantization [126](#)
Quantization error [127](#)
Quantization step [127](#)

R

Radix-2 FFT Algorithms 348
Random signals [13](#)
Rational z-transform [211](#)
R-bus 515
Realizability of Ideal Filters 412
Realization structures 247
Reconstruction filter [132](#)
Rectangular Window 462
Recursive Systems [110](#)
Region of Convergence 151
Reset 529
Resolution [127](#)
ROC 152
Rounding [61](#)

S

Sampling 61, 119
Sampling frequency [120](#)
Sampling Theorem [129](#)
Scaling in z-domain 160
Scaling operation [37](#)
Serial Ports 525
Shift Invariance [42](#)
Shift variant systems [43](#)
Shifter block (SB) register 520
Shifter Exponent (SE) register 520
Shifter Instructions 526
Sifting [68](#)
Signal flow graph 268, 355
Signal multiplier [39](#)
Signals and Systems [6](#) to [132](#)
Spectrum Analysis using DFT [343](#)
Stability 218
Stability of FIR Filters 456
Stability of LSI Systems [79](#)
Stability Property [48](#)
Standard Signals [27](#)
Static system [41](#)
Status registers 522
Stopband edge frequency 428
Stopband edge value [431](#)
Structure for IIR Systems 256
Structures for FIR Systems 248
Study of DSP [4](#)
Summation [68](#)
Symmetric FIR filters 456
Symmetric signal [32](#)
Symmetry Properties [306](#)
Symmetry property of W_N 346
System Builder 528
System Function 211
System Interface 528
System specification file 528

T

Tabulation method [89](#)
Technology for DSP [4](#)
Time-variant systems [42](#)
Time reversal 160
Time reversal of a Sequence 326
Time shifting 159
Timer 5
Transfer function 399
Transfer function Plot [400](#)
Transfer functions 278
Truncation [61](#)
Twiddle factor [290](#)
Types of digital Filters [407](#)

U

Unit advance block [40](#)
Unit delay block [39](#)
Unit Ramp Sequence [28](#)
Unit Sample Sequence [27](#)
Unit Step Sequence [27](#)
Unstable systems [48](#)

V

Von-Neumann Architecture [512](#)

W

Window Functions [464](#)

Z

z-plane [154](#)
z-transfer pairs [180](#)
z-transform [151](#)

Hidden page

Contents

- Digital against analog processing, Application of DSP, Technology review, Application of DSP in speech processing, Biomedical engineering, Vibration analysis, Picture (image) Processing (case studies).
- The z-transform and its inverse, Systems function, Poles and zeros, Discrete time signals and systems, Generation of discrete time signals, Properties and algebraic manipulation, Sampling theorem ADC, DAC, Difference equations, Representation of discrete system via difference equation, Convolutions (linear and circular), Linear time invariant system, Casuality, Stability.
- Digital filter structure, Describing equation, System transfer function, filter categories, Direct form I and II structures, Cascade combination of second order section, Parallel combination of second order sections, FIR filter structure, Frequency sampling structure of FIR filters, Lattice-ladder structure.
- Definition and properties of Discrete Fourier Transform, Fast Fourier Transform, Decimation in frequency, Decimation in time, GOETZEL algorithm Chirp-z-transform algorithm, Use of FFT algorithm in linear filtering and correlation, Quantization effect of FFT, Frequency analysis of discrete time signals, Power density, Energy density, Discrete time aperiodic signals its energy density, Convergence effect.
- **Filter Design**
Design of linear phase FIR filters using windows, Rectangular windows, Gibb's phenomenon, Triangular window, Hamming window, Blackman window, Kaiser window, Hanning window, Design of linear-phase FIR filters using frequency sampling, Design of optimum equiripple linear phase FIR filters, FIR differentiators, Design of Hilbert transforms, Comparison of design methods. IIR filters : Design of IIR filters from analog filters, Approximation of derivatives, Impulse invariance, Bilinear transform, Least square filter design.
- **Hardware Architecture of DSP**
Study of DSP chip architecture as an examples :(chip of Texas instruments or analog devices), Features of DSP chip architecture and instructions, Comparison with microprocessor chip.
- **Analysis of Finite Word**
Length effects, The quantization process and errors, Analysis of coefficient, Quantization effects in FIR filters, A/D conversion noise analysis, Analysis of arithmetic round effect errors, Dynamic range scaling, Low sensitivity digital filters, Reduction of product round off errors, Limit cycles in IIR filters, Round-off errors in FFT algorithms.
- **Applications**
Dual - tone multiply signal detection, Spectral analysis using DFT, Short term DFT, Musical sound processing, Voice privacy, Sub band coding of speech and special audio signals, Over sampling D/A, Over sampling A/D, Applications of multirate signal processing.



Technical Publications Pune

1, Amit Residency, 412 Shaniwar Peth, Pune - 411030, M.S., India.
Telefax : +91 (020) 24495496/97, Email : technical@vtubooks.com

Visit us at : www.vtubooks.com

First Edition : 2008

Rs. 260/-

ISBN 978-81-8431-424-3

