Digital Signal Processing Fundamentals with Hands-on Experiments

by

Andreas Spanias, Ph.D.

June 24, 2009
School of Electrical, Computer and Energy Engineering
Arizona State University
Tempe, AZ 85287-5706
480 965 1837
480 965 8325

DSP Primer – June 24, 2009
Sponsored in part by NSF 0817596

Disclaimer

These course notes cover the fundamentals and select applications of Digital Signal Processing and are intended solely for education. No other use is intended or authorized. No warranty or implied warranty is given that any of the material is fit for a particular purpose, application, or product. Although the author believes that the concepts, algorithms, software, and data presented are accurate, he provides no guarantee or implied guarantee that they are free of error. The material presented should not be used without extensive verification. If you do not wish to be bound by the above then please do not use these notes.

Contents

Digital Signal Processing (DSP) Introduction

- Digital Signal Processing (DSP) is a branch of signal processing that emerged from the rapid development of VLSI technology that made feasible real-time digital computation.
- DSP involves time and amplitude quantization of signals and relies on the theory of discrete-time signals and systems.
- DSP emerged as a field in the 1960s.
- Early applications of off-line DSP include seismic data analysis, voice processing research.

Digital vs Analog Signal Processing

Advantages of digital over analog signal processing:

- flexibility via programmable DSP operations,
- storage of signals without loss of fidelity,
- off-line processing,
- lower sensitivity to hardware tolerances,
- rich media data processing capabilities,
- opportunities for encryption in communications,
- Multimode functionality and opportunities for software radio.

- Disadvantages :

- Large bandwidth and CPU demands

DSP Historical Perspective

- Nyquist Theorem 1920's.
- Statistical Time Series, PCM 1940's.
- Digital Filtering, FFT, Speech Analysis mid 1960s (MIT, Bell Labs, IBM).
- Adaptive Filters, Linear Prediction (Stanford, Bell Labs, Japan 1960s).
- Digital Spectral Estimation, Speech Coding (1970s).

DSP Historical Perspective (2)

- First Generation DSP Chips (Intel microcontroler, TI, AT&T, Motorola, Analog Devices (early 1980s)
- Low-cost DSPs (late 1980s)
- Vocoder Standards for civilian applications (late 1980s)
- Migration of DSP technologies in general purpose CPU/Controllers "native" DSP (1990s)
- High Complexity Rich Media Applications
- Low Power (Portable) Applications

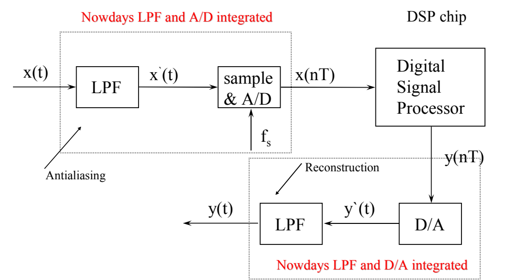
DSP Applications

- Military Applications (target tracking, radar, sonar, secure communications, sensors, imagery)
- Telecommunications (cellular, channel equalization, vocoders, software radioetc)
- PC and Multimedia Applications (audio/video on demand, streaming data applications, voice synthesis/recognition)
- Entertainment (digital audio/video compression, MPEG, CD, MD, DVD, MP3)
- Automotive (Active noise cancellation, hands-free communications, navigation-GPS, IVHS)
- Manufacturing, instrumentation, biomedical, oil exploration, robotics
- Remote sensing, security

Communications and DSP

- DTMF (use of the FFT and digital oscillators)
- Adaptive echo cancellation (Hands-free telephony, Speakerphones)
- Speech coding (speech coding in cellular phones)
- Modem (data/computer connectivity)
- Software radio (multi-mode/multi standard wireless communications)
- Channel estimation (equalization)
- Antenna beamforming (space division multiple access - SDMA)
- CDMA (modulating with random sequences)

Typical Digital Signal Processing System



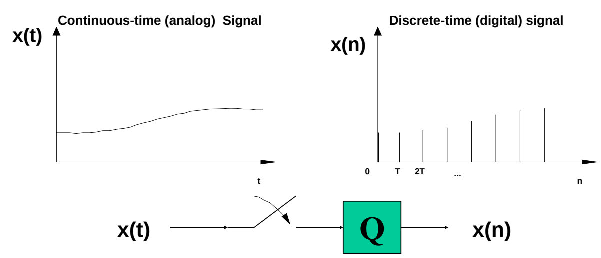
Remarks: The diagram shows the sampling, processing, and reconstruction of an analog signal. There are applications where processing stops at the digital signal processor, e.g., speech recognition.

Symbols and Notation

$$x_a(t)|_{t=nT} = x(nT) = x(n) \quad ;\text{discrete} - \text{time input}$$
$$y(n) \quad ;\text{discrete} - \text{time output}$$
$$H(.) \quad ;\text{transfer and frequency response functions}$$
$$h(.) \quad ;\text{impulse response (system function)}$$
$$n \quad ;\text{discrete} - \text{time index}$$

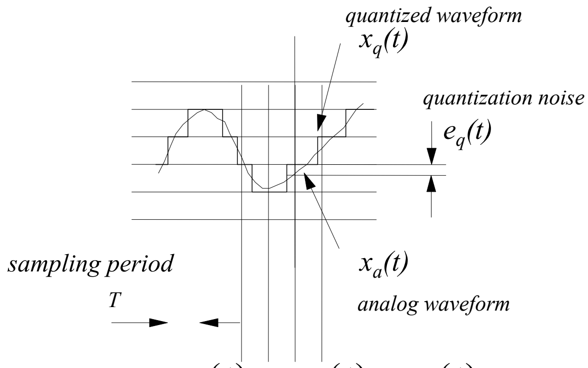
Remarks: In general and unless otherwise stated lower case symbols will be used for time-domain signals and upper case symbols will be used for transform domain signals. Bold face or underlined face symbols will be Be generally used for vectors or matrices.

Continuous vs Discrete-time



Remarks: A continuous-time signal is converted to discrete-time using sampling and quantization. As a result aliasing and quantization noise is introduced. This noise can be controlled by properly designing the quantizer and anti-aliasing filter.

Quantization Noise



$$x_q(t) = x_\alpha(t) + e_q(t)$$

Simplest Quantization Scheme - Uniform PCM

Performance in terms of Signal to Noise Ratio (SNR)

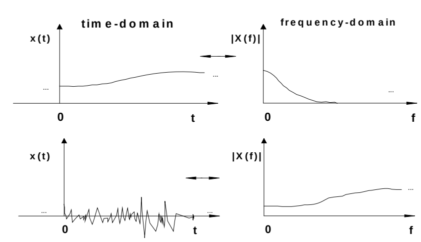
$$SNR_{PCM} = 6.02 R_b + K_1$$

where R_b is the number of bits and the value of K_1 depends on signal statistics. For telephone speech $K_1 = -10$

Oversampling Δ/Σ or Σ/Δ Conversion

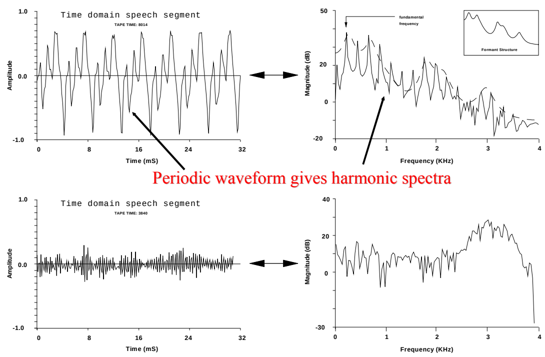
- Integrated oversampling and 1-bit quantization
- Very compact and inexpensive circuitry (some low power applications as well)
- Lowers analog circuit complexity with a modest increase in software (DSP MIPS) complexity
- Uses concepts from multirate signal processing and Delta Modulation
- Will be described in the context of multirate signal processing

Time vs Frequency Domain



Remarks: Slowly time-varying signals tend to have low-frequency content while signals with abrupt changes in their amplitudes have high frequency content. The frequency content of signals can be estimated using Fourier techniques.

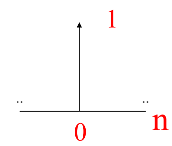
Example: Time vs Frequency Domain Speech



Some Important Signals

Discrete-time Impulse

$$\delta(n) =$$

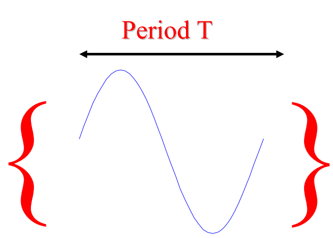


Think of signals as a weighted sum of impulses.
Impulses help in analyzing signals and filters

Some Important Signals (3)

The sinusoid

$$\sin(\omega t) = \sin(\frac{2\pi}{T} t) =$$



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

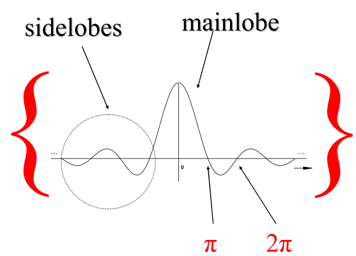
units: ω(rad/s) f(Hz) T(s)

Sinusoids are used in analyzing or synthesizing acoustic and other signals

Some Important Signals (4)

The sinc function

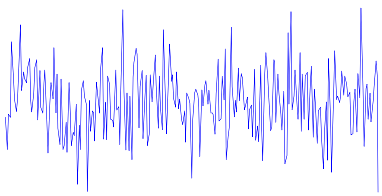
$$\text{sinc}(t) = \frac{\sin(t)}{t} =$$



Sinc functions often appear in signal and filter analysis
particularly when considering frequency domain behavior

Some Important Signals (5)

Random noise



Encountered in communication systems and other application
Characterized by their mean and variance

Representing Periodic Signals with Sinusoids

Fourier series: Trigonometric form:

$$x(t) = a_0 + \sum_{k=1}^{\infty} a_k \cos(k\omega_o t) + \sum_{k=1}^{\infty} b_k \sin(k\omega_o t)$$

Fourier series: Complex (magnitude/phase) form:

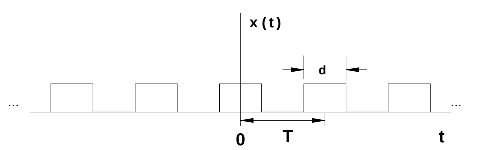
Preferred in engineering-- ≫ $$x(t) = \sum_{k=-\infty}^{\infty} X_k e^{jk\omega_o t}$$

$\boldsymbol{X_k}$ are complex F.S. coefficients and provide spectral magnitude and phase info

and $$e^{jk\omega_o t} = \cos(k\omega_o t) + j\sin(k\omega_o t)$$

Fourier Series Analysis Example

Representing a Periodic Pulse Train as a Sum of Harmonic Sinusoids



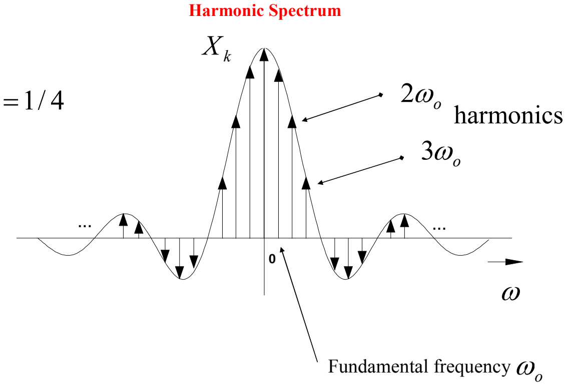
$$X_k = \frac{1}{T}\int_{-d/2}^{d/2} 1 e^{-jk\omega_o t}\,dt = \frac{d}{T}\,\text{sinc}\left(\frac{k\omega_o d}{2}\right)$$

Remarks: A periodic pulse signal has a discrete F.S. spectrum described by samples that fall on a sinc (sinc(x)=sin(x)/x) function. As the period increases the F.S. components become more dense in frequency and weaker in amplitude. If T goes to infinity periodicity is lost and the F.S. vanishes.

Fourier Series Example (2)

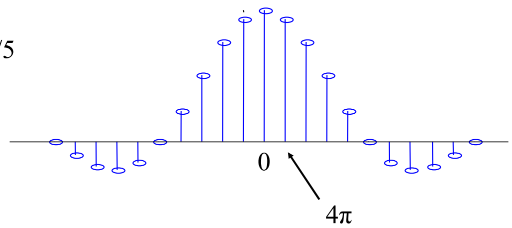
Harmonic Spectrum

$$\frac{d}{T} = 1/4$$

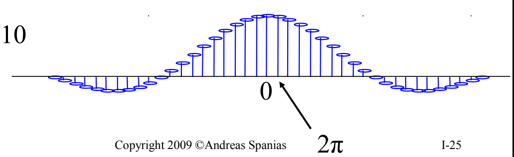


Fourier Series Example (3)

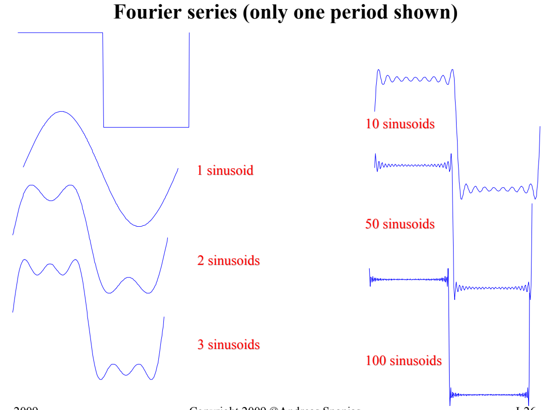
d/T=1/5



d/T=1/10



Use Sinusoids to synthesize a periodic pulse using the Fourier series (only one period shown)



The Continuous Fourier Transform (CFT) Equations

The Fourier transform — Analysis Expression

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt$$

The inverse Fourier transform — Synthesis Expression

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) e^{j\omega t} d\omega$$

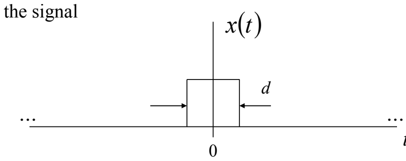
A Fourier transform pair is designated by: $x(t) \leftrightarrow X(\omega)$

Remarks: Both time and frequency are continuous variables. The CFT can handle non-periodic signals as long as they are integrable. Periodic signals can be handled using the impulse and CFT properties.

Fourier transform of a time-limited pulse

(Represent a single pulse by sinusoids)

Given the signal

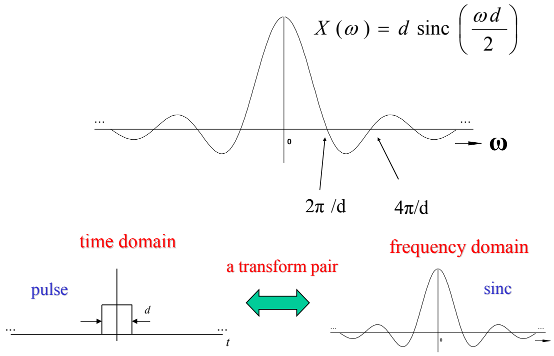


$$X(\omega) = \int_{-d/2}^{d/2} e^{-j\omega t} dt = d \ \text{sinc}\left(\frac{\omega d}{2}\right)$$

Remarks: Note that a time-limited signal has a non-bandlimited CFT spectrum. The sinc function has zero crossings at integer multiples of 2π/d. As the pulse width increases the sinc function "shrinks". In the limit, if T goes to infinity (i.e., pulse becomes D.C. signal) the sinc function collapses to a unit impulse.

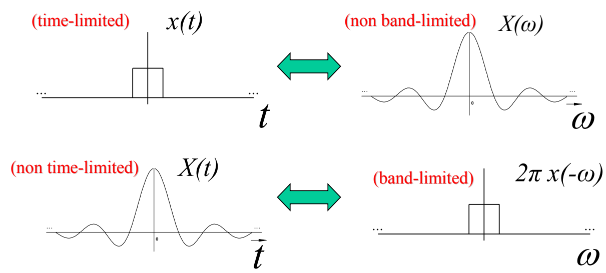
Fourier transform of a time-limited pulse(Cont.)

$$X(\omega) = d \ \mathrm{sinc}\left(\frac{\omega d}{2}\right)$$



Symmetry of the Fourier transform

$$if \quad x(t) \leftrightarrow X(\omega) \quad then \quad X(t) \leftrightarrow 2\pi x(-\omega)$$



The Time-Domain Convolution (Filtering) Property

$$x(t) \leftrightarrow X(\omega)$$

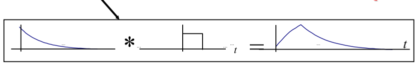
$$h(t) \leftrightarrow H(\omega)$$

$$h(t) * x(t) \leftrightarrow H(\omega)X(\omega)$$

$$h(t) * x(t) = \int h(\tau)x(t-\tau)d\tau$$

Muliplication in frequency
is essentially a filtering operation

convolution in time is multiplication in frequency

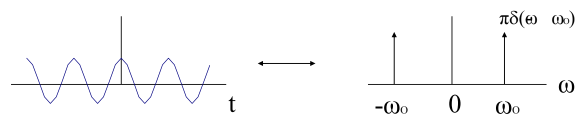


DEMO

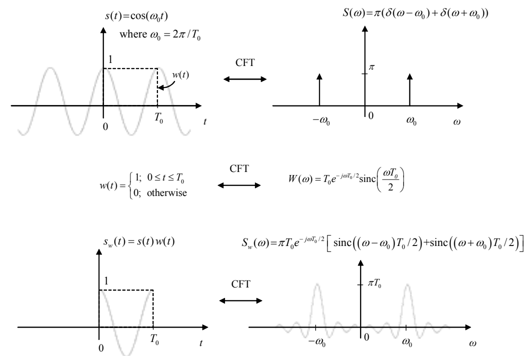
Example: Convolution of an exponential with a pulse

Important Fourier Transform Pairs

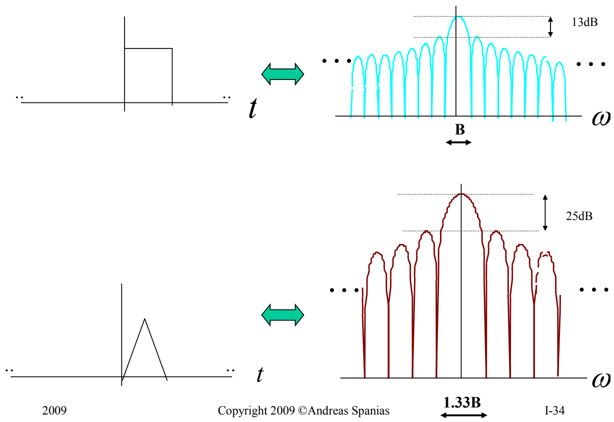
$$\cos(\omega_0 t) \leftrightarrow \pi(\delta(\omega-\omega_0)+\delta(\omega+\omega_0))$$



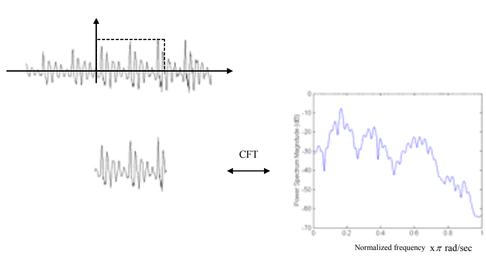
Truncating a Cosine



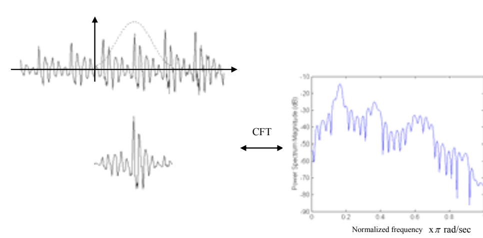
Truncating Signals with Tapered Windows



Truncating Speech



Truncating Speech (tapered window)



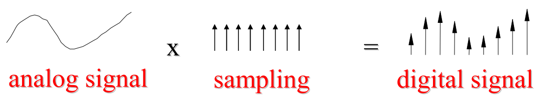
The Sampling Process

A bandlimited signal that has no spectral components at or above B can be uniquely represented by its sampled values spaced at uniform intervals that are not more than π/B seconds apart.

$$T \le \frac{\pi}{B}$$

or a signal that is bandlimited to B must be sampled at a rate of ω_s where

$$\omega_s \ge 2B \quad or \quad f_s \ge \frac{B}{\pi}$$



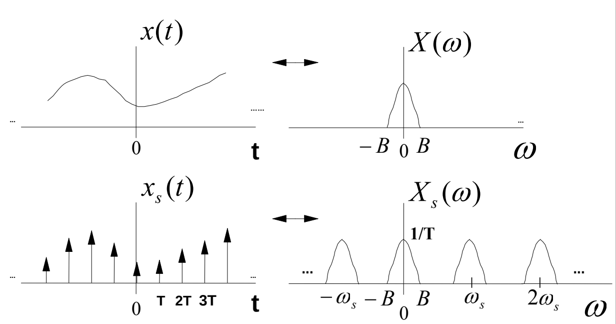
Example: Audio - Bandwidth

200 - 3200 Hz	Basic Telephone Speech Intelligible Preserves Speaker Identity
50 - 7000 Hz	Wideband Speech AM-grade audio
50 - 15000 Hz	Near High Fidelity FM-grade Audio
20 - 20000 Hz	High-Fidelity CD/DAT Quality Voice

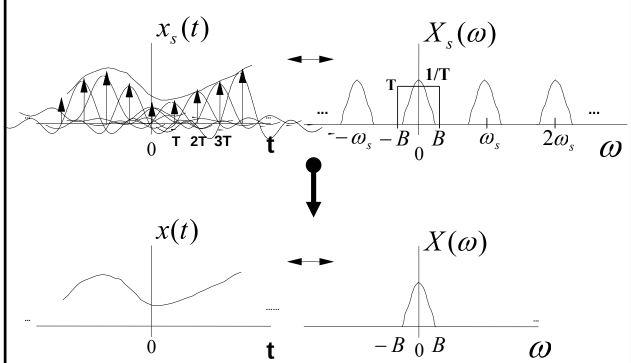
Example: Sampling of Audio Signals

Format	Bandwidth	Sampling frequency
Telephony	3.2 kHz	8 kHz
Wideband audio	7 kHz	16 kHz
High-fidelity, CD	20 kHz	44.1 kHz
Digital audio tape (DAT)	20 kHz	48 kHz
Super audio CD (SACD)	100 kHz	2.8224 MHz
DVD audio (DVD-A)	96 kHz	192 kHz

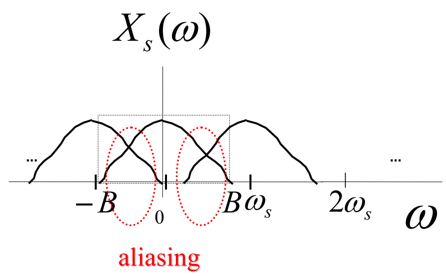
Sampling and Periodic Spectra



Signal Reconstruction using an Ideal Filter



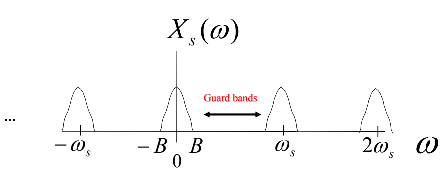
ALIASING (UNDERSAMPLING) $\omega_s<2B$



aliasing

the signal can not be recovered perfectly even with an ideal filter
only a distorted version of the signal can be recovered

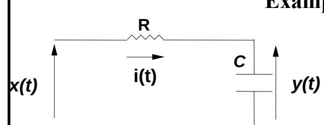
Oversampling $\omega_s>>2B$



Oversampling relaxes the requirements on antialiasing filters

It is used in Σ/Δ (Δ/ Σ) A-to-D converters

Example



Differential equation:

$$\frac{dy(t)}{dt}+\frac{1}{RC}y(t)=\frac{1}{RC}x(t)$$

Transfer function:

$$H(s)=\frac{1}{1+sRC}$$

Frequency response function:

$$H(\omega)=\frac{1}{1+j\omega RC}$$

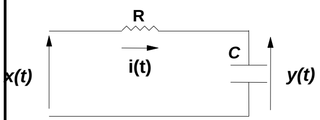
if $x(t) = \sin(\omega t)$ — sinusoid in

then — sinusoid out

$$y^{ss}(t)=|H(\omega)|\sin(\omega t+\angle H(\omega))$$

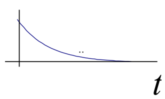
Example - Impulse Response

Consider the circuit below with R=1M, C=1x10^{-6}



$$\frac{dh(t)}{dt}+\frac{1}{RC}h(t)=\frac{1}{RC}\delta(t)$$

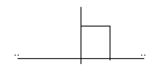
The solution: $h(t)=\frac{1}{RC}e^{-\frac{t}{RC}}=e^{-t}$ for $t \geq 0$



Example - Convolve and obtain an output

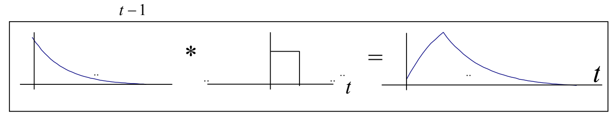
Consider the RC with impulse response $h(t) = e^{-t}u(t)$

and the input $x(t) = u(t) - u(t-1)$



$$y(t) = \int_0^t e^{-\tau} d\tau = 1 - e^{-t} \quad \text{for} \quad 0 < t < 1$$

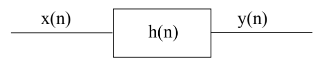
$$y(t) = \int_{t-1}^t e^{-\tau} d\tau = -e^{-t} + e^{-(t-1)} \quad \text{for} \quad t > 1$$



Discrete-time Linear Systems

Digital Filters

Discrete time Linear Systems – Digital Filters



The output is produced by convolving the input with the impulse response

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

This operation can also involve a finite-length impulse response(FIR) sequence

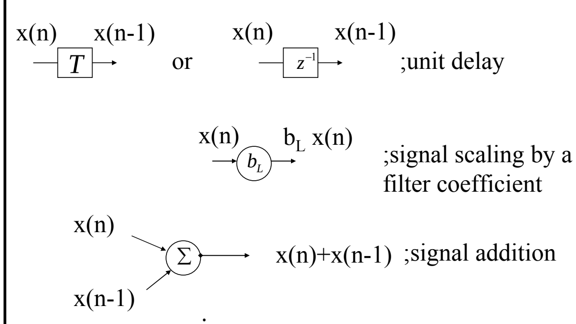
$$y(n) = \sum_{m=0}^{L} h(m)x(n-m)$$

An FIR filter is programmed using a multiply-accumulate instruction

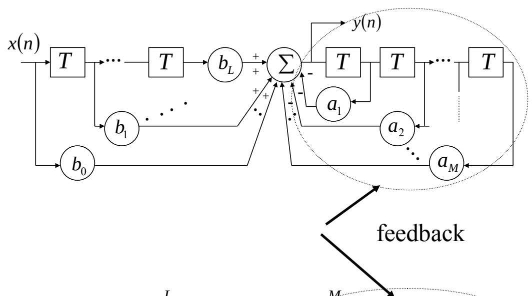
Some Definitions

- •A digital filter is linear if it has the property of generalized superposition
- •A digital filter is causal if it non anticipatory, i.e., the present output does not depend on future inputs.
- •All real-time systems are causal.
- • Non-causalities arise in image processing where the signal indexes are spatial instead of temporal.
- • Unless otherwise stated all systems in this course will be assumed causal.

Some More Definitions

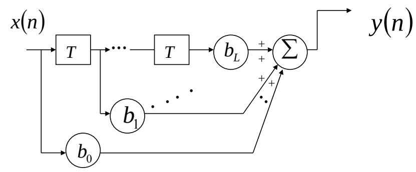


IIR Digital Filter Structure



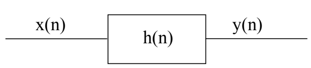
$$y(n) = \sum_{i=0}^{L} b_i x(n-i) - \sum_{i=1}^{M} a_i y(n-i)$$

FIR Digital Filter Structure



$$y(n) = \sum_{i=0}^{L} b_i x(n-i)$$

Two i/p o/p Equations for Digital Filters



One can compute the output using the *convolution sum*

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

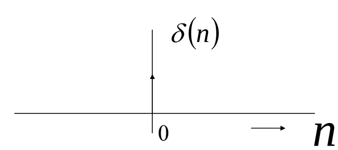
or by using the *difference equation*

$$y(n) = \sum_{i=0}^{L} b_i x(n-i) - \sum_{i=1}^{M} a_i y(n-i)$$

Remark: The impulse response h(n) can be determined by solving the difference equation.

Unit Impulse

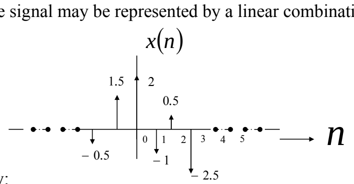
The analysis of digital filters in the frequency domain is facilitated using sinusoids. In the time domain a unique input signal is used for analysis, namely the unit impulse. That is defined as:



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

Signal Representation with Unit Impulses

Any discrete-time signal may be represented by a linear combination of unit impulses



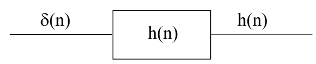
is represented by:

$$x(n) = -.5\delta(n+2) + 1.5\delta(n+1) + 2\delta(n) - \delta(n-1) + .5\delta(n-2) - 2.5\delta(n-3)$$

Impulse Response

The response of a digital filter to a unit impulse is known as impulse response and is given by

$$h(n) = b_0\delta(n) + b_1\delta(n-1) + \ldots + b_L\delta(n-L) - a_1h(n-1) - a_2h(n-2) - \ldots - a_Mh(n-M)$$

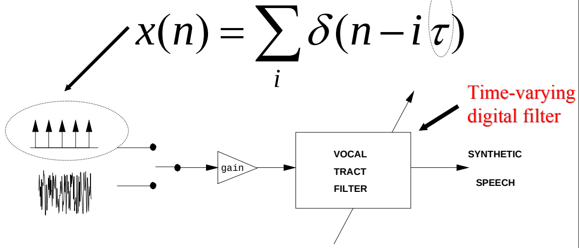


Impulses as input to source-system LPC Vocoders

Vowels are typically synthesized by exciting a filter representing the mouth and nasal (vocal tract) cavity with a train of periodic impulses

pitch period

$$x(n) = \sum_i \delta(n - i\tau)$$



Finite-Length Impulse Response (FIR)

If the digital filter has no feedback terms the impulse response is finite length

$$h(n) = b_0\delta(n) + b_1\delta(n-1) + \ldots + b_L\delta(n-L)$$

Note that

$$h(0) = b_0$$

$$h(1) = b_1$$

$$h(2) = b_2$$

$$\vdots$$

$$h(L) = b_L$$

Remark: The filter has a finite-length impulse response and is called FIR. The values of the impulse response sequence are the coefficients themselves. The filter is always stable.

Example – The Moving Average Filter

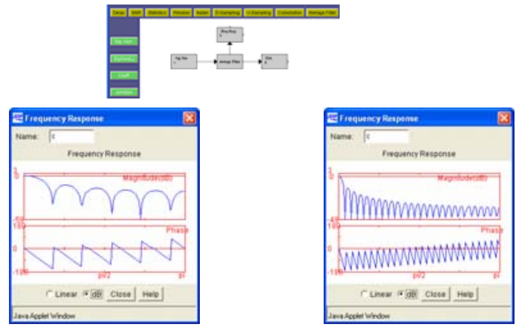
$$y(n) = \frac{1}{L+1}\sum_{i=0}^{L} x(n-i)$$

$$h(n) = \frac{1}{L+1}\{\delta(n) + \delta(n-1) + ... + \delta(n-L)\}$$

$$h(n) = \frac{1}{L+1} \qquad 0 \le n \le L$$

Remark: The moving average is essentially a low-pass (smoothing) filter. Later on we will see that this filter is also optimal in estimation problems.

J-DSP Simulation of Averaging Filter



L=10 L=50

Infinite Length Impulse Response (IIR)

If the digital filter has feedback terms then the impulse response is infinite length

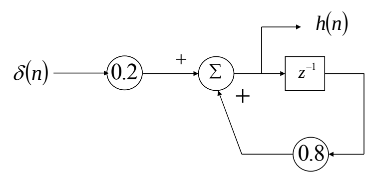
$$h(n) = \sum_{i=0}^{L} b_i \delta(n-i) - \sum_{i=1}^{M} a_i h(n-i)$$

Example: $h(n) = \delta(n) - a_1 h(n-1)$

$$h(0) = 1 \quad h(1) = -a_1 \quad h(2) = a_1^{\,2} \quad :: \quad h(n) = (-a_1)^n$$

Remark: Note that if the coefficient a_1 has magnitude larger than one the impulse response will go to infinity and hence the filter would be unstable.

IIR – Another First Order Example



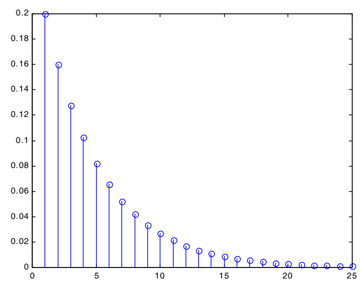
$$h(n) = 0.2\delta(n) + 0.8h(n-1)$$

$$h(n) = 0.2\ (0.8)^n \qquad n \ge 0$$

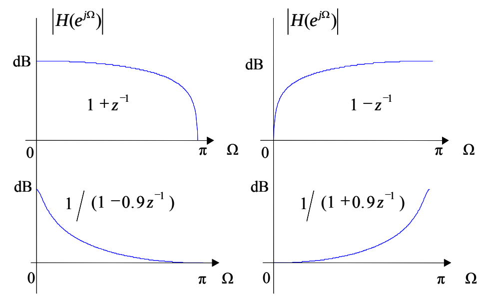
Remark: This particular IIR filter is also a low-pass filter behaving in similar manner like the the averaging FIR filter.

IIR – Another First Order Example (Plot)

$$h(n) = 0.2\ (0.8)^n \qquad n \geq 0$$



Frequency Responses



Impulse Response and Stability

For the causal digital filter

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

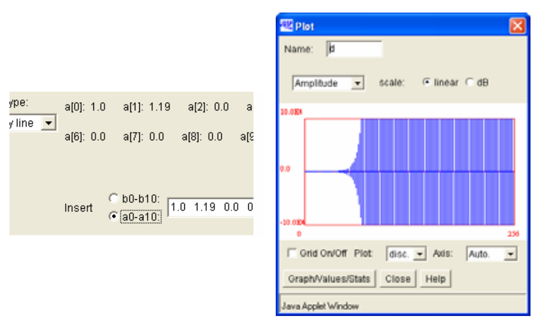
Bounded Input Bounded Output (BIBO) stability is defined as

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

The condition above is guaranteed if

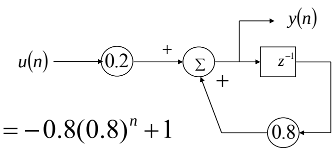
$$|p_i| < 1 \quad \textbf{\textit{for all}} \quad \boldsymbol{i = 1, 2, \ldots, M}$$

An Unstable Filter

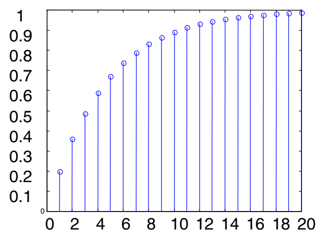


Example of Transient and Steady State Response

$u(n) = 1 \quad n \geq 0$



$y(n) = y^{tr}(n) + y^{ss}(n) = -0.8(0.8)^n + 1$



Steady-State Sinusoidal Response of Digital Filters

A special case of interest is the steady-state response to input sinusoids and is formulated as follows. For the IIR filter

$$y(n) = \sum_{i=0}^{L} b_i x(n-i) - \sum_{i=1}^{M} a_i y(n-i)$$

The frequency response function is given by:

$$H(e^{j\Omega}) = \frac{b_0 + b_1 e^{-j\Omega} + b_2 e^{-j2\Omega} + \ldots + b_L e^{-jL\Omega}}{1 + a_1 e^{-j\Omega} + a_2 e^{-j2\Omega} + \ldots + a_M e^{-jM\Omega}}$$

Steady-State Sinusoidal Response of Linear Discrete Systems (Cont.)

The frequency response function is periodic and an example of the steady state sinusoidal response is given below

if:

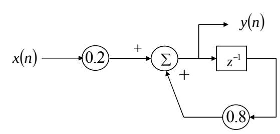
$$x(n) = \sin(n\Omega)$$

then:

$$y^{ss}(n) = \left|H(e^{j\Omega})\right| \sin(\Omega n + \angle H(e^{j\Omega}))$$

$$\Omega = \frac{2\pi f}{f_s}$$;normalized frequency

Example of Steady State Sinusoidal Response



$$H(e^{j\Omega}) = \frac{0.2}{1 - 0.8e^{-j\Omega}}$$

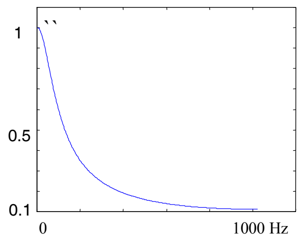
The filter is excited by a 500 Hz sinusoid and the Sampling rate is 2000Hz.

$$x(n) = \sin(2\pi n 500/2000) = \sin(\frac{\pi n}{2})$$

$$y^{ss}(n) = \left|\frac{0.2}{1+0.8j}\right| \sin(\frac{\pi n}{2} + \arg(\frac{0.2}{1+0.8j}))$$

Frequency Response Plot

$$H(e^{j\Omega}) = \frac{0.2}{1 - 0.8e^{-j\Omega}}$$



The Z-Transform in DSP

The Z Transform

•The z-transform plays a similar role in DSP as the Laplace transform in analog circuits and systems.

•It provides intuition that is sometimes not evident in time-domain analysis

•Simplifies time-domain operations – time domain-convolution maps to Z-domain multiplication

•Used to define transfer functions

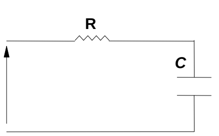
•Could be used to determine responses of systems using a table look-up process

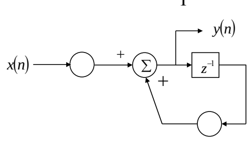
From the Laplace Transform to the Z Transform

$$H_a(s) \Rightarrow H_d(z)$$

s-domain transfer function **z-domain transfer function**

$$H_a(s) = \frac{1}{1+sRC} \longrightarrow H_d(z) = \frac{b_o}{1+a_1 z^{-1}}$$





The Z Transform- Definition

Given the signal: $x(n)$

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

For causal signals, i.e., $x(n) = 0 \quad for \quad n < 0$

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

Selected Properties of the Z Transform

Linearity: if: $x(n) \leftrightarrow X(z)$ and $y(n) \leftrightarrow Y(z)$

then

$$\alpha x(n) + \beta y(n) \leftrightarrow \alpha X(z) + \beta Y(z)$$

Shifting: $$x(n \pm m) \leftrightarrow z^{\pm m} X(z)$$

Convolution: $$x(n) * y(n) \leftrightarrow X(z) Y(z)$$

Scaling (bandwidth expansion): $$a^n x(n) \leftrightarrow X(z/a)$$

Delay of 7 Samples

$$x(n-7) \leftrightarrow z^{-7} X(z)$$

Selected Z Transform Pairs

Unit-Impulse: $$\delta(n) \leftrightarrow 1$$

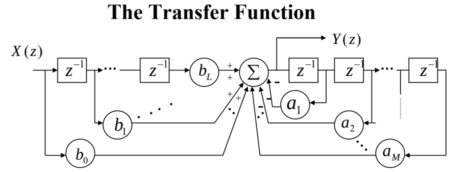
Sinusoids: $$\{\sin(\Omega n), n \geq 0\} \leftrightarrow \left\{ \frac{z^{-1}\sin(\Omega)}{1 - 2z^{-1}\cos(\Omega) + z^{-2}}, |z| > 1 \right\}$$

$$\{\cos(\Omega n), n \geq 0\} \leftrightarrow \left\{ \frac{1 - z^{-1}\cos(\Omega)}{1 - 2z^{-1}\cos(\Omega) + z^{-2}}, |z| > 1 \right\}$$

Sampled Unit-Step: $$\{1, n \geq 0\} \leftrightarrow \left\{ \frac{1}{1 - z^{-1}}, |z| > 1 \right\}$$

Exponential Signals: $$\{a^n, n \geq 0\} \leftrightarrow \left\{ \frac{z}{z - a}, |z| > |a| \right\}$$

The Transfer Function



$$y(n) = \sum_{i=0}^{L} b_i x(n-i) - \sum_{i=1}^{M} a_i y(n-i)$$

To write the transfer function put the difference equation in the z-domain

$x(n) \leftrightarrow X(z)$ and $y(n) \leftrightarrow Y(z)$

$$Y(z) = \sum_{i=0}^{L} b_i X(z) z^{-i} - \sum_{i=1}^{M} a_i Y(z) z^{-i}$$

The Transfer Function (Cont.)

$$X(z)\left(\sum_{i=0}^{L} b_i\ z^{-i}\right) = Y(z)\left(1+\sum_{i=1}^{M} a_i\ z^{-i}\right)$$

The transfer function *H(z)* is defined as:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \ldots + b_L z^{-L}}{1 + a_1 z^{-1} + \ldots + a_M z^{-M}}$$

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

Note that feedback terms are in the denominator

The Transfer Function and the Impulse Response

Convolution maps to multiplications in the z domain

$$y(n) = x(n) * h(n) \quad \leftrightarrow \quad Y(z) = X(z)H(z)$$

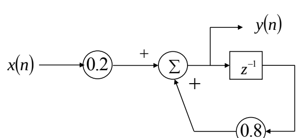
It can be shown that a transfer function *H(z)* is related to the impulse response sequence *h(n)* by:

$$H(z) = \sum_{n=0}^{\infty} h(n) z^{-n}$$

or

$$h(n) \leftrightarrow H(z)$$

Example: Write the Transfer Function of a First Order IIR Filter



Step 1: Write the Difference Equation

$$y(n) = 0.2\,x(n) + 0.8\,y(n-1)$$

Step 2: Transform all signals to the z domain

$$y(n) \Leftrightarrow Y(z) \qquad x(n) \Leftrightarrow X(z)$$

$$y(n-1) \Leftrightarrow z^{-1}Y(z)$$

Example: Transfer Function of a First Order IIR Filter (cont.)

Step 3: Since z-transform is a linear operation I can write

$$Y(z) = 0.2\,X(z) + 0.8\,z^{-1}\,Y(z)$$

Step 4: Form the ratio Y(z)/X(z) to get the transfer function

$$H(z) = \frac{Y(z)}{X(z)} = \frac{0.2}{1-0.8z^{-1}} = \frac{0.2z}{z-0.8}$$

Note below that the impulse response can be written by inspection of H(z)

$$h(n) = 0.2\ \ 0.8^n\,u(n) \ \Leftrightarrow\ H(z) = \frac{0.2z}{z-0.8} \quad |z| > 0.8$$

The Transfer Function of FIR Systems

For the FIR filter the transfer function is:

$$H(z) = b_0 + b_1 z^{-1} + \ldots + b_L z^{-L} = \sum_{i=0}^{L} b_i \; z^{-i}$$

this is also in agreement with

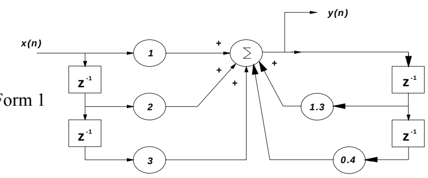
$$H(z) = \sum_{n=0}^{L} h(n) z^{-n}$$

Note that for FIR filters

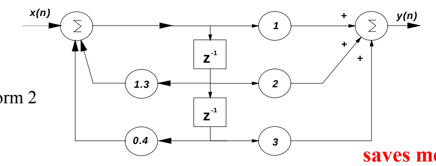
$$\{h(0),\; h(1), \ldots,\; h(L)\} = \{b_0, b_1, \ldots,\; b_L\}$$

Equivalent Filter Realizations

Direct Form 1



Direct Form 2



saves memory

Poles and Zeros of H(z)

In general the transfer function is rational; it has a numerator and a denominator polynomial.

The roots of the numerator and denominator polynomials are called the zeros and the poles respectively.

Pole-zero decompositions of *H(z)* are quite useful and provide intuition in signal analysis and filter design.

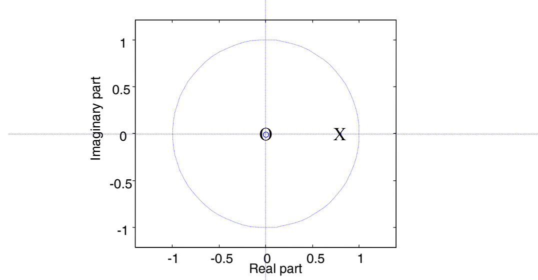
$$H(z) = G\frac{(z-\zeta_1)(z-\zeta_2)\ldots(z-\zeta_L)}{(z-p_1)(z-p_2)\ldots(z-p_M)} = G\frac{\prod_{i=1}^{L}(z-\zeta_i)}{\prod_{i=1}^{M}(z-p_i)}$$

where G is a gain factor

Example: Poles and Zeros of H(z)

$$H(z) = \frac{0.2z}{z-0.8}$$

- the zero (O) of this transfer function is at z=0
- the pole (X) of this transfer function is at z=0.8

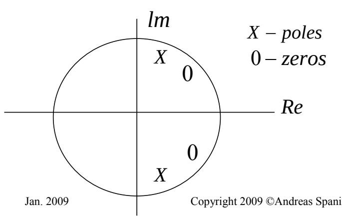


Example: Poles-Zeros of a Second Order System

$$H(z)=\frac{1-1.3435\,z^{-1}+0.9025\,z^{-2}}{1-0.45\,z^{-1}+0.55\,z^{-2}}$$

$$H(z)=\frac{(z-.95e^{j45^\circ})(z-.95e^{-j45^\circ})}{(z-.7416e^{j72.34^\circ})(z-.7416e^{-j72.34^\circ})}$$

Note that the filter coefficients are real valued and therefore poles and zeros occur in complex conjugate pairs.



Poles and Zeros and Stability

The poles are related to the stability of the filter since they are related to the impulse response of the system. In fact, the poles of For stability all the poles must be inside the unit circle, that is

$$|p_i|<1 \qquad \textbf{\textit{for all}} \quad \textbf{\textit{i = 1, 2, . . . , M}}$$

IIR filters may be all-pole or pole-zero and stability is always a concern. FIR or all-zero filters are always stable.

The Frequency Response Function

The transfer function is

$$H(z)=\frac{b_0+b_1z^{-1}+\ldots+b_Lz^{-L}}{1+a_1z^{-1}+\ldots+a_Mz^{-M}}$$

by evaluating on the unit circle, i.e. for $z=e^{j\Omega}$

$$H(e^{j\Omega})=\frac{b_0+b_1e^{-j\Omega}+\ldots+b_Le^{-jL\Omega}}{1+a_1e^{-j\Omega}+\ldots+a_Me^{-jM\Omega}}$$

The Frequency Response Function (Cont.)

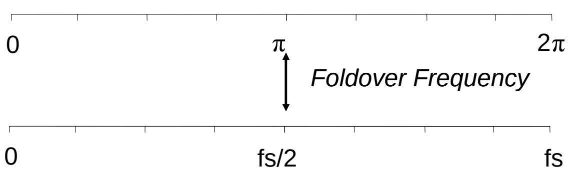
The frequency response function and is a complex and periodic With period 2o. The normalized frequencies Ω are associated to the sampling frequencies f_s by

$$\Omega=\omega T=2\pi\frac{f}{f_s}$$

Sampling period (T), *rad* (Ω), *rad/s* (ω), *Sampling frequency* (f_s)

where f_s is the sampling frequency and f is any frequency of interest. In practice, one determines the frequency response up to half the sampling frequency (fold-over frequency).

The Frequency Response Function (Cont.)



- The frequency response is usually plotted w.r.t. normalized frequencies (*Ω*)
- The frequency response is periodic with period f_s **(2 π)**
- Since frequencies of interest are up to the bandwidth of the analog signal the spectrum is usually plotted up to ***f_s/2, (π) the foldover frequency***

The Frequency Response and Poles and Zeros

The magnitude frequency response function

$$\left|H(e^{j\Omega})\right| = G\frac{\prod_{i=1}^{L}\left|e^{j\Omega}-\zeta_i\right|}{\prod_{i=1}^{M}\left|e^{j\Omega}-p_i\right|}$$

The phase frequency response function

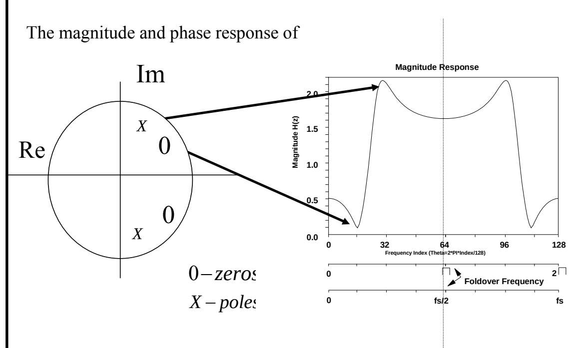
$$\arg(H(e^{j\Omega})) = \sum_{i=1}^{L}\arg(e^{j\Omega}-\zeta_i) - \sum_{i=1}^{M}\arg(e^{j\Omega}-p_i)$$

Remarks on Effects of Poles and Zeros on $H(e^{j\Omega})$

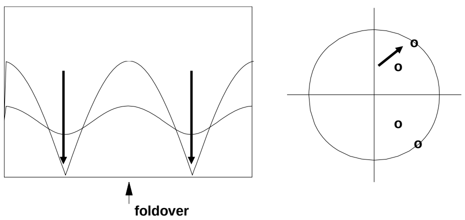
- Poles tend to create peaks in the magnitude frequency response
- Zeros tend to create valleys in the magnitude frequency response
- Very selective filters are designed efficiently by placing poles close to the unit circle
- Sharp notches are achieved efficiently with zeros very close to the unit circle
- A sharp notch in the frequency response needs many poles (high order) if we are restricted to an all-pole filter (not efficient).
- A sharp peak in the frequency response needs many zeros (long or high order FIR) if an all-zero filter is to be used (not efficient).

Magnitude and Phase Response

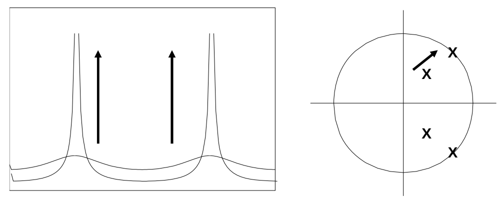
The magnitude and phase response of



Zero Locations and Frequency Response



Pole Locations and Frequency Responses



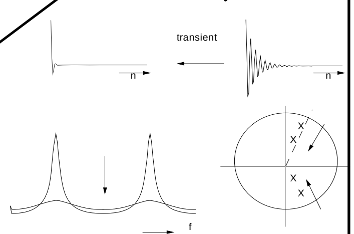
As the poles move outwards we get sharper peaks and if the poles are on the unit circle we get an oscillator.

Bandwidth Expansion
Moving poles inwards stabilizes numerically a filter

$$\gamma^n h(n) \leftrightarrow H(z/\gamma) \quad 0 < \gamma < 1$$

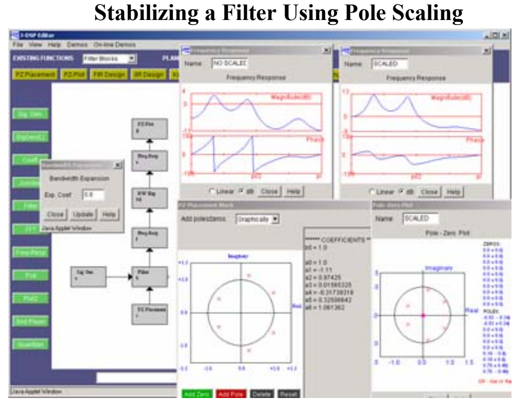
example

$$H(z/\gamma) = \frac{1}{1 + \sum_{i=1}^{M} \gamma^i a_i \, z^{-i}}$$



The scaling property of the z-transform is exploited for bandwidth expansion in LPC vocoders

Stabilizing a Filter Using Pole Scaling



Computing Filter Responses Using the Inverse Z-Transform

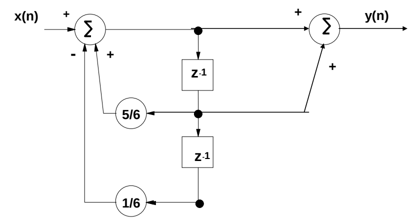
Partial Fractions: Given a the transfer function

$$H(z) = \frac{b_0 z^L + b_1 z^{L-1} + \ldots + b_L}{z^M + a_1 z^{M-1} + \ldots + a_M}$$

for distinct poles write H(z) as:

$$H(z) = c_0 + c_1 \frac{z}{z-p_1} + \ldots + c_M \frac{z}{z-p_M}$$

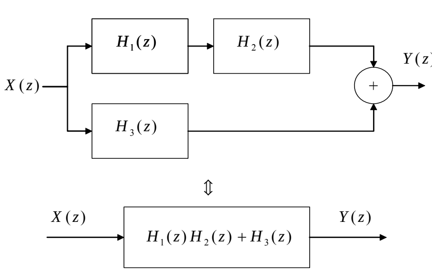
Example: Find the Impulse Response of the Filter



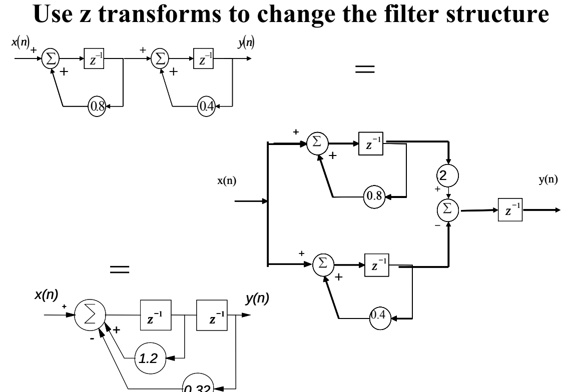
$$H(z) = \frac{z^2 + z}{z^2 - \frac{5}{6}z + \frac{1}{6}} \longrightarrow H(z) = 9\frac{z}{z - \frac{1}{2}} - 8\frac{z}{z - \frac{1}{3}}$$

$$h(n) = 9\left(\frac{1}{2}\right)^n - 8\left(\frac{1}{3}\right)^n, \; n \geq 0$$

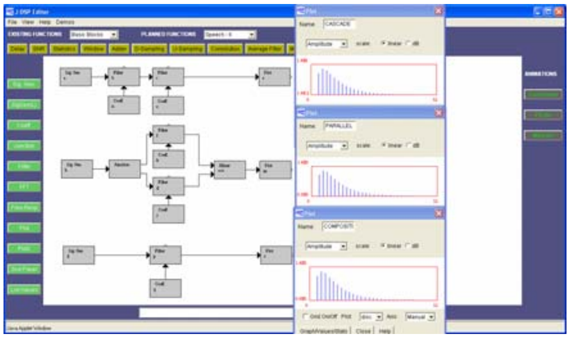
Filter Configurations



Use z transforms to change the filter structure



J-DSP Simulation



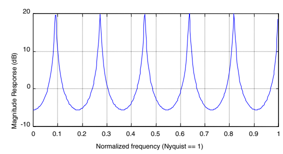
Interesting Transfer Functions
Long Term Prediction Synthesis Filter

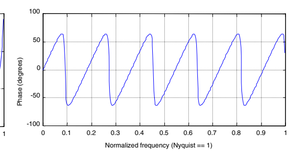
$$H(z) = \frac{1}{1 + a_p z^{-p}}$$

Example p=10>>

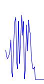
$$H(z) = \frac{1}{1 - 0.9 z^{-10}}$$

- Called long term because p is a long-term delay
- Used in Vocoders

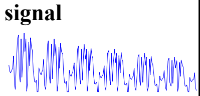




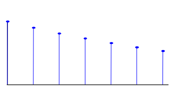
LTP excited by a random signal



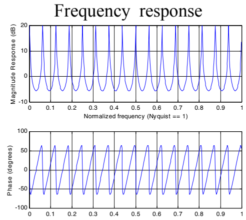
$$\frac{1}{1 - 0.95 z^{-30}}$$



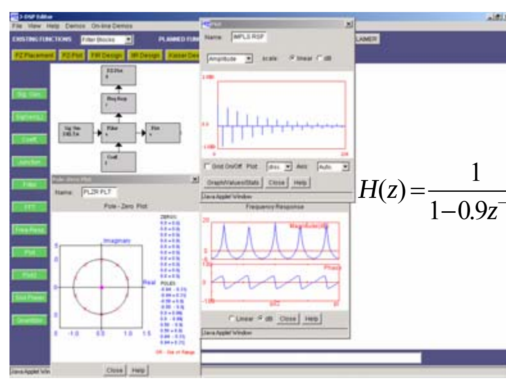
Impulse response



Frequency response



J DSP Simulation of LTP

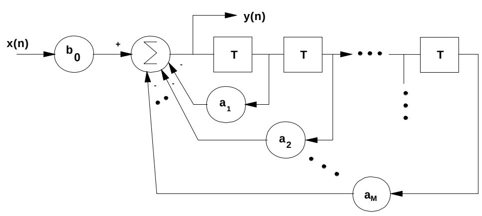


$$H(z) = \frac{1}{1 - 0.9 z^{-10}}$$

Interesting Transfer Functions
All-Pole Filters

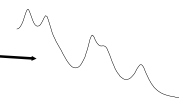
$$H(z)=\frac{b_o}{1+\sum_{i=1}^{M} a_i\ z^{-i}}$$

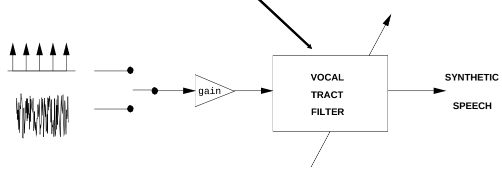
Typically used in speech processing (vocoders) as well as in spectral estimation applications



Interesting Transfer Functions
All-Pole Filters (2)

$$H(z)=\frac{b_0}{1+\sum_{i=1}^{M} a_i\ z^{-i}}$$





Interesting Transfer Functions
Digital Oscillators

If we realize the z-transform of a sinusoid as a transfer function and we excite it with a unit impulse we get a sinusoidal output

$$\{\sin(\Omega n), n\geq 0\}\leftrightarrow\left\{\frac{z^{-1}\sin(\Omega)}{1-2z^{-1}\cos(\Omega)+z^{-2}}, |z|>1\right\}$$

If we excite *H(z)* below

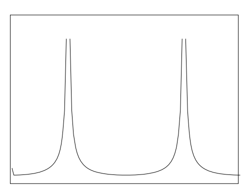
$$H(z)=\frac{z^{-1}\sin(\Omega)}{1-2z^{-1}\cos(\Omega)+z^{-2}}$$

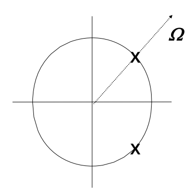
with $\delta(n)$ then the output will be a sinusoid

Interesting Transfer Functions
Digital Oscillators (2)

Note that H(z) has its poles on the unit circle

$$H(z)=\frac{z^{-1}\sin(\Omega)}{1-2z^{-1}\cos(\Omega)+z^{-2}}=\frac{z\sin(\Omega)}{(z-e^{j\Omega})(z-e^{-j\Omega})}$$

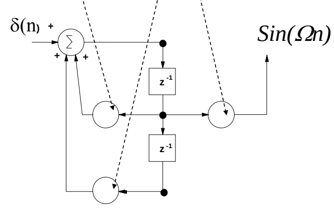




Interesting Transfer Functions Digital Oscillators (3)

$$H(z) = \frac{z^{-1}\sin(\Omega)}{1 - 2\cos(\Omega)\, z^{-1} + 1\, z^{-2}}$$

$$b_0 = 0$$
$$b_1 = \sin(\Omega)$$
$$a_1 = -2\cos(\Omega)$$
$$a_2 = 1$$

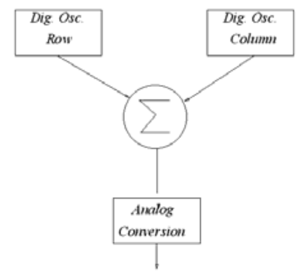


DTMF Applications and Digital Oscillators (2)

Dual Tone Multi-frequency Encoder

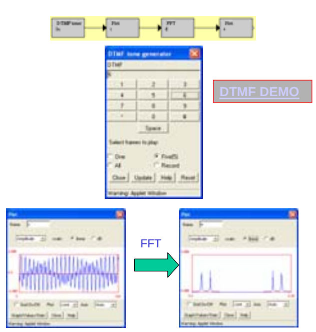
Row Frequencies

Column Frequencies	1.209 kHz	1.336 kHz	1.477 kHz
0.697 kHz	1	2	3
0.77 kHz	4	5	6
0.852 kHz	7	8	9
0.941 kHz	*	0	#



DTMF Functionality

- Generates dual-tone-multi-frequency (DTMF) tones used in landline telephony applications.
- The tones can be played back using the J-DSP provided sound player, and used in a DSP simulation.



$y = \cos(2\pi\, f_1 nT) + \cos(2\pi\, f_2 nT)$

where f_1 and f_2 are chosen from the tone frequencies (697, 770, 852, 941, 1209, 1336, 1477 (Hz)). The sampling frequency is 8 KHz, i.e., T = 0.125ms

Design of FIR Digital Filters

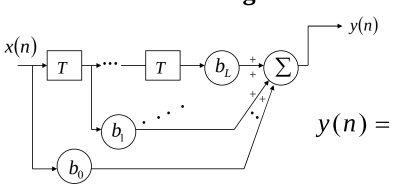
FIR Digital Filters

Advantages:
- Linear Phase Design
- Quite Efficient for designing notch filters
- Always Stable

Disadvantages:
- Requires High Order for Narrowband Design

Applications:
- Speech Processing, Telecommunications
- Data Processing, Noise Suppression, Radar
- Adaptive Signal Processing, Noise Cancellation, Echo Cancellation, Multipath channels

FIR Digital Filters



$$y(n) = \sum_{i=0}^{L} b_i x(n-i)$$

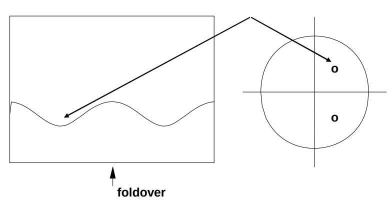
$$Y(z) = b_0 X(z) + b_1 X(z) z^{-1} + \ldots + b_L X(z) z^{-L}$$

$$H(z) = \frac{Y(z)}{X(z)} = b_0 + b_1 z^{-1} + \ldots + b_L z^{-L}$$

FIR Filter Frequency Response

$$H(e^{j\Omega}) = b_0 + b_1 e^{-j\Omega} + \ldots + b_L e^{-jL\Omega}$$

$$\Omega = 2\pi \frac{f}{f_s}$$



LINEAR PHASE DESIGN

Linear Phase (constant time delay) FIR filter design is important in pulse transmission applications where pulse dispersion must be avoided. The frequency response function of the FIR filter is written as:

$$H(e^{j\Omega}) = b_0 + b_1 e^{-j\Omega} + b_2 e^{-j2\Omega} + \ldots + b_L e^{-jL\Omega}$$

where

$$H(e^{j\Omega}) = M(\Omega) e^{\Phi(\Omega)}$$

$$M(\Omega) = \left| H(e^{j\Omega}) \right|, \qquad \Phi(\Omega) = \arg(H(e^{j\Omega}))$$

GROUP DELAY

The time delay or group delay of a filter is defined as

$$\tau = -\frac{d\Phi(\Omega)}{d\Omega}$$

therefore if $\Phi(\Omega)$ is a linear function of Ω then τ is a constant.

τ is given in terms of samples

LINEAR PHASE AND IMPULSE RESPONSE SYMMETRIES

It can be shown that linear phase is achieved if

$$h(n) = h(L-n)$$

where $h(n)$ is the impulse response of the filter. For $L = odd$

$$H(z) = \sum_{n=0}^{\frac{L-1}{2}} h(n)(z^{-n} + z^{-(L-n)})$$

$$H(e^{j\Omega}) = e^{-j\frac{\Omega L}{2}} \sum_{n=0}^{\frac{L-1}{2}} 2\, h(n) \cos\left(\Omega\left(\frac{L}{2} - n\right)\right)$$

SYMMETRIC AND ANTI-SYMMETRIC LINEAR PHASE FILTERS

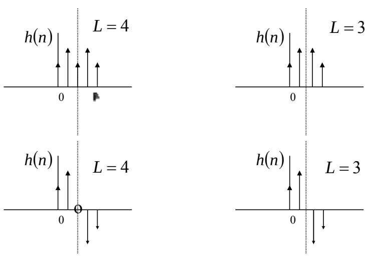
Two Anti-symmetries for L=*even* or L=*odd* for

$$h(n) = -h(L-n)$$

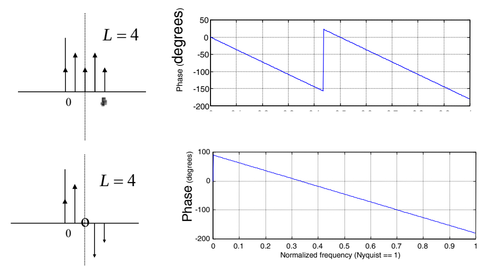
Two Symmetries for L=*even* or L=*odd* for

$$h(n) = h(L-n)$$

EXAMPLES OF SYMMETRIES



EXAMPLES OF PHASE AND SYMMETRY IN h(n)



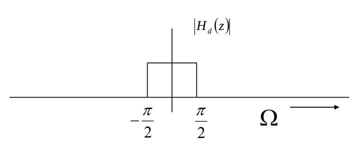
Symmetries and Linear Phase Simulated with J-DSP

Fourier Series Design Example

For the ideal low pass filter the impulse response sequence is an infinite length sampled sinc function. Lets say the sampling frequency is 8 KHz and we wish to have a cutoff frequency at 2 KHz. This results in

$$\Omega_c = 2\pi \frac{f_c}{f_s} = 2\pi \frac{2000}{8000} = \frac{\pi}{2}$$

That is



Fourier Series Design Example (Cont.)

The ideal impulse response $h_d(n)$ is given by

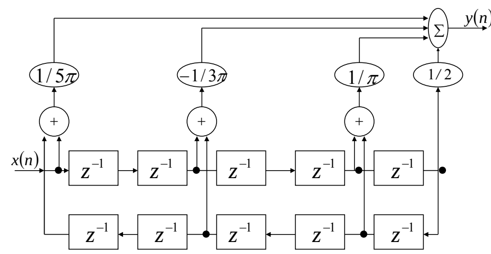
$$h_d(n) = \frac{1}{2}\text{sinc}\left(\frac{n\pi}{2}\right), n = 0, \pm 1, \pm 2, ...$$

For an FIR filter of 11 coefficients

$$h(n) = \left\{...0,0,\frac{1}{5\pi},0,-\frac{1}{3\pi},0,\frac{1}{\pi},\frac{1}{2},\frac{1}{\pi},0,-\frac{1}{3\pi},0,\frac{1}{5\pi},0,0,...\right\}$$

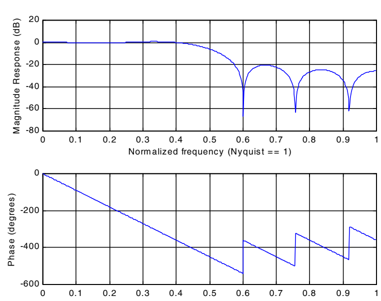
This impulse response is not causal, however a shift operator with 5 delays (z^{-5}) will convert it into a causal DF.

REALIZATION



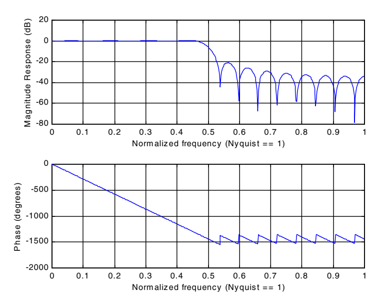
Fourier Series Design Example (Cont.)

$$b_0 = \frac{1}{5\pi}, b_2 = -\frac{1}{3\pi}, b_4 = \frac{1}{\pi}, b_5 = \frac{1}{2}, b_6 = \frac{1}{\pi}, b_8 = -\frac{1}{3\pi}, b_{10} = \frac{1}{5\pi}$$



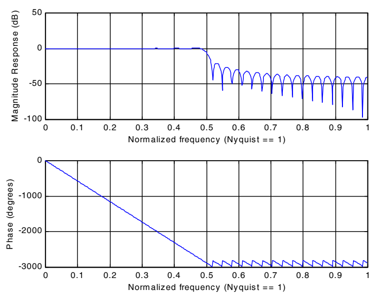
Fourier Series Design Example L=32

$$b_n = \frac{1}{2}\text{sinc}\left(\frac{(n-16)\pi}{2}\right), n = 0,1,....32$$



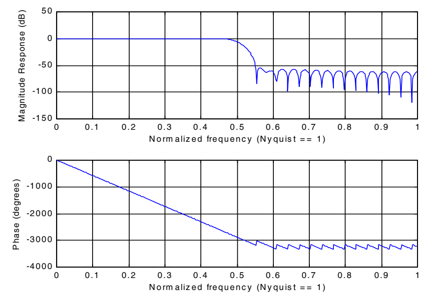
Fourier Series Design Example L=64

$$b_n = \frac{1}{2}\text{sinc}\left(\frac{(n-32)\pi}{2}\right), n = 0,1,\ldots,64$$

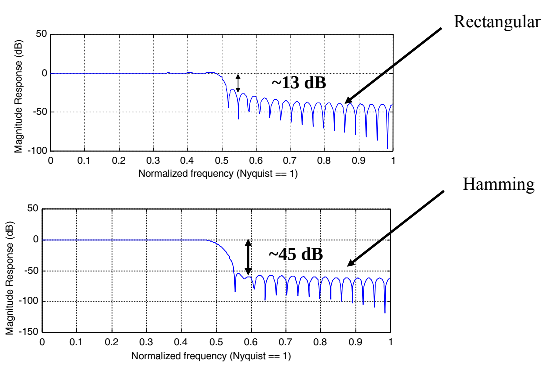


Truncating with Hamming Window L=64

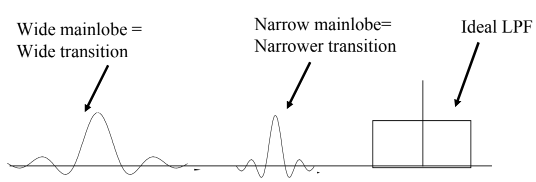
$$b_n = \frac{1}{2}\textbf{sinc}\left(\frac{(n-32)\pi}{2}\right) \quad x \quad \textbf{Hamming(L)}$$



F.S. Design; Rectangular vs Hamming Window - L=64

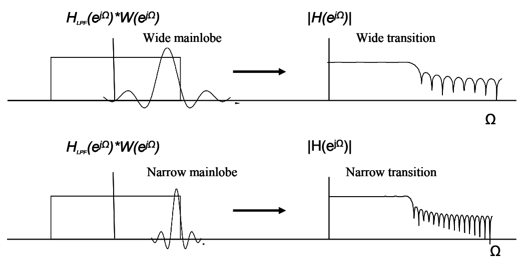


Truncating in time- frequency convolution and F.S. design



- The main-lobe width determines transition characteristics
- The sidelobe level determines rejection characteristics

Truncating with Short/Long Windows and F.S. design



Notes On Fourier Series Design

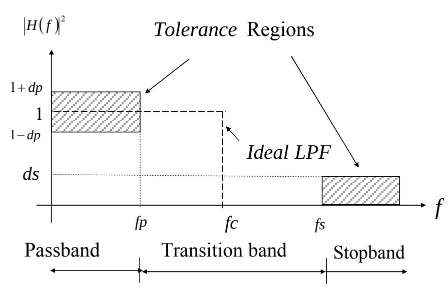
•The design performed in the previous example involved truncation of an ideal symmetric impulse response.A symmetric impulse response produces a linear phase design.

•Truncation involves the use of a window function which is multiplied with the impulse response. Multiplication in the time domain maps into frequency domain convolution and the spectral characteristics of the window function affect the design.

•The main-lobe width determines transition characteristics

•The sidelobe level determines rejection characteristics

DEFINING DESIGN SPECIFICATIONS



DESIGN USING THE KAISER WINDOW

The Kaiser window is parametric and its bandwidth as well as its sidelobe energy can be designed. Mainlobe bandwidth controls the transition characteristics and sidelobe energy affects the ripple characteristics.

$$w(n) = \frac{I_0\left(\beta\left[1-\left(\frac{n-\alpha}{\alpha}\right)^2\right]^{1/2}\right)}{I_0(\beta)}, 0 \leq n \leq L-1$$

α = L/2 ; associated with the order of the filter

β is a design parameter that controls the shape of the window

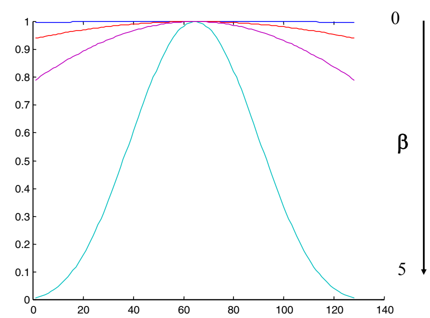
$I_0(.)$ is a zeroth order modified Bessel function of the first kind

DESIGN USING THE KAISER WINDOW (Cont.)

$$I_0(x) = 1 + \sum_{k=1}^{\infty} \left[\frac{1}{k!} \left(\frac{x}{2} \right)^k \right]^2$$

25 terms from the Bessel function are sufficient

EXAMPLES OF KAISER WINDOW



KAISER WINDOW DESIGN EQUATIONS

Given f_p, f_s, T and dp, ds determine the FIR filter coefficients.

$$\delta = \min(ds, dp)$$

$$A = -20 \log_{10} \delta$$

$$\Delta\Omega = 2\pi (f_s - f_p) T$$

The filter order is $L = \dfrac{A - 8}{2.285 \, \Delta\Omega} (\pm 2)$

and the kaiser parameter β is given by

$$\beta = \begin{cases} 0.1102 \, (A - 8.7), \; A > 50 \\ 0.5842 \, (A - 21)^{0.4} + 0.07886 \, (A - 21), 21 \le A \le 50 \\ 0, \quad A < 21 \end{cases}$$

DESIGN PROCEDURE

1. Determine the cutoff frequency for the ideal Fourier Series method.

$$f_c = \frac{f_s - f_p}{2}$$

2. Design the ideal LPF using the Fourier Series.
3. Design the Kaiser window
4. Shift and truncate the ideal impulse response

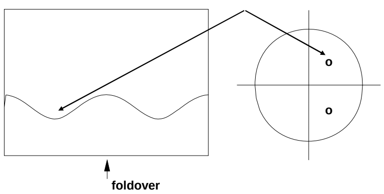
$$h_{LPF}(n) = w(n) \, h_d\left(n - \frac{L}{2}\right), \quad 0 \le n \le L$$

Note that this procedure can be generalized for the design of BPF, HPF, and BSF.

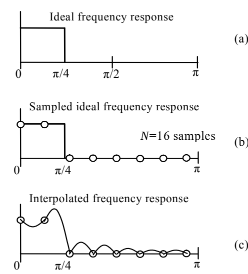
Design by Zero-Placement

As zeros are placed towards the unit circle the frequency response magnitude decreases at and in the vicinity of the frequency of the zeros.

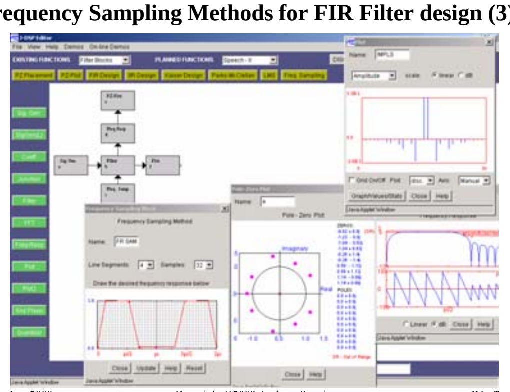
.



Frequency Sampling Methods for FIR Filter design



Frequency Sampling Methods for FIR Filter design (3)



Min-Max and Parks-McClellan Optimum FIR Design

The Parks-McClellan design is based on Min-Max

Equiripple and linear phase design is possible

This class of methods involve minimizing the maximum error between the designed FIR filter frequency response and a prototype

$$\min_{\{h(i),i=0,1,\ldots,L\}} \left\{ \max \left| E(e^{j\Omega}) \right| \right\}$$

where

$$E(e^{j\Omega}) = W(e^{j\Omega})(H_d(e^{j\Omega}) - H(e^{j\Omega}))$$

FIR Filter Design Using MATLAB+

IN THE MATLAB SP TOOLBOX

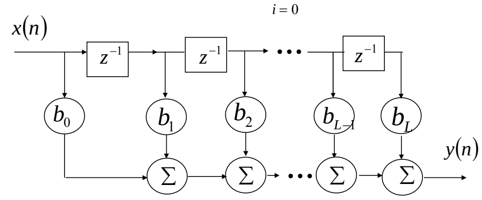
cremez - Complex and nonlinear phase equiripple FIR filter design.
fir1 - Window based FIR filter design - low, high, band, stop, multi.
fir2 - Window based FIR filter design - arbitrary response.
fircls - Constrained Least Squares filter design - arbitrary response.
fircls1 - Constrained Least Squares FIR filter design - low and highpass
firls - FIR filter design - arbitrary response with transition bands.
firrcos - Raised cosine FIR filter design.
intfilt - Interpolation FIR filter design.
kaiserord - Window based filter order selection using Kaiser window.
remez - Parks-McClellan optimal FIR filter design.
remezord - Parks-McClellan filter order selection.

+ MATLAB is registered trade mark of the MathWorks

FIR Filter Realizations

Direct Realizations

$$H(z) = \sum_{i=0}^{L} b_i z^{-i}$$

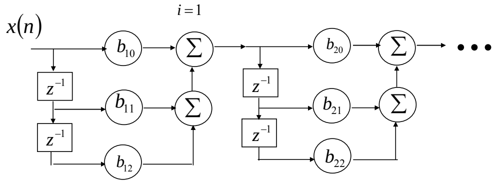


•Requires multiply accumulate instructions

FIR Filter Cascade Realizations

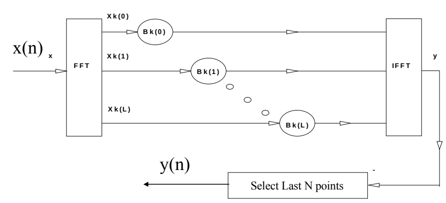
Cascade Realizations

$$H(z) = \prod_{i=1}^{q} (b_{i0} + b_{i1} z^{-1} + b_{i2} z^{-2})$$



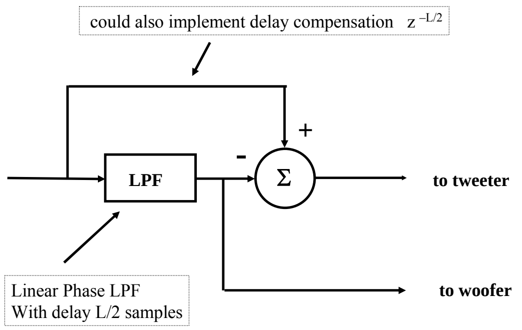
•Reduced Effects from Coefficient Quantization and round-off
•In Fixed-Point implementation signal scaling must be done carefully at each stage

Transform-Domain FIR Filter Realizations



A Transform domain realization is possible using the overlap and save and the FFT. This yields computational savings for high order implementations. Input data is organized in 2N-point blocks and blocks are shifted N points at a time. The data blocks and N zero-padded coefficients are transformed and multiplied and the results is inverse transformed. The last N-points are selected as the result. The blocks are updated and the process is repeated.

Implementing Efficiently Digital Cross-Over Using Subtractive Operations



DESIGN OF IIR DIGITAL FILTERS

IIR DIGITAL FILTERS

Advantages:

Efficient in terms of order

- Poles create narrow-band peaks efficiently
- Arbitrarily long impulse responses with few feedback
- coefficients

Disadvantages:

- Feedback and stability concerns
- Sensitive to Finite Word Length Effects
- Generally non-Linear Phase

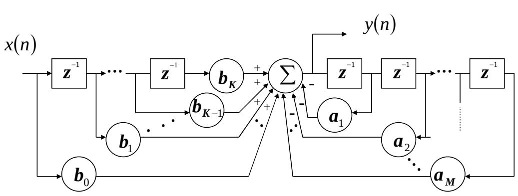
Applications:

- Speech Processing, Telecommunications
- Data Processing, Noise Suppression, Radar

IIR FILTERS

The difference equation is:

$$y(n) = \sum_{i=0}^{L} b_i x(n-i) - \sum_{i=1}^{M} a_i y(n-i)$$



IIR FILTERS (Cont.)

The transfer function:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + ... + b_L z^{-L}}{1 + a_1 z^{-1} + ... + a_M z^{-M}}$$

The frequency-response function :

$$H(e^{j\Omega}) = \frac{b_0 + b_1 e^{-j\Omega} + ... + b_L e^{-jL\Omega}}{1 + a_1 e^{-j\Omega} + ... + a_M e^{-jM\Omega}}$$

IIR Filter Design by Analog Filter Approximation

The idea is to use many of the successful analog filter designs to design digital filters

This can be done by either:

- by sampling the analog impulse response (*impulse invariance*) and then determining a digital transfer function

or

- by transforming directly the analog transfer function to a digital filter transfer function using the *bilinear transformation*

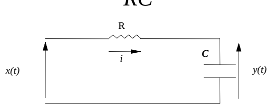
IIR Filter Design by Analog Filter Approxination

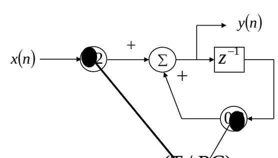
The *impulse invariance* method suffers from aliasing and is rarely used

The *bilinear transformation* does not suffer from aliasing and is by more popular than the impulse invariance method.
The frequency relationship from the s-plane to the z-plane is non-linear, and one needs to compensate by pre-processing the critical frequencies such that after the transformation the desired response is realized. Stability is maintained in this transformation since the left-half s-plane maps onto the interior of the unit circle.

Impulse Invariance

$$h_a(t) = \frac{1}{RC} e^{-\frac{t}{RC}} u(t) \quad \text{and} \quad H_a(j\omega) = \frac{1}{1 + j\omega RC}$$





$$h(n) = \frac{T}{RC} e^{-\frac{nT}{RC}} u(n) \qquad H(z) = \frac{(T/RC)}{1 - e^{-(T/RC)} z^{-1}}$$

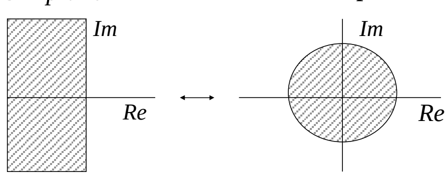
Impulse Invariance (2)

$$H_a(s) = \frac{1}{s - p_1} + \frac{1}{s - p_2} + \ldots.. + \frac{1}{s - p_M}$$

$$H(z) = \frac{T}{1 - e^{-p_1 T} z^{-1}} + \frac{T}{1 - e^{-p_2 T} z^{-1}} + \ldots.. + \frac{T}{1 - e^{-p_M T} z^{-1}}$$

The Bilinear Transformation

Bilinear Transform $\Rightarrow \; z = \dfrac{1+s}{1-s}$



$$\boldsymbol{H}(\boldsymbol{z}) = \boldsymbol{H}\left(\boldsymbol{s} = \frac{\boldsymbol{z}-1}{\boldsymbol{z}+1}\right)$$

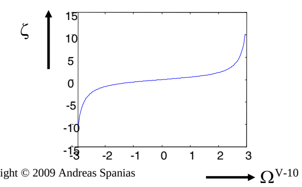
The Bilinear Transformation (Cont.)

The bilinear transformation compresses the frequency axis

$$\omega[-\infty,\infty] \leftrightarrow \Omega[-\pi,\pi]$$

The non-linear frequency transformation (frequency warping function) is given by

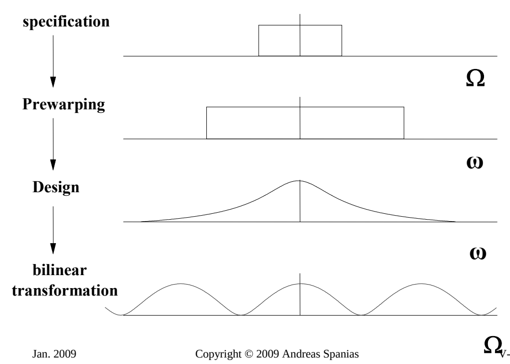
$$\omega = \tan\left(\frac{\Omega}{2}\right)$$



Procedure for Analog Filter Approximation

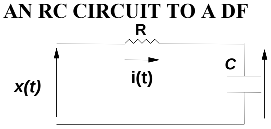
1. **Consider Critical Frequencies**
2. **Pre-warp critical frequencies**
3. **Analog Filter Design**
4. **Bilinear Transformation**
5. **Realization**

Applying the Bilinear Transformation



EXAMPLE: TRANSFORMING AN RC CIRCUIT TO A DF

Suppose we want a first-order (R-C LPF) appoximation



$$H(\omega)=\frac{1}{1+j\omega RC}$$

Say we have the following DF specs:

Step 1: $\Omega_c=\pi/2$

Apply pre-warping

Step 2: $\omega_c=\tan(\frac{\Omega_c}{2})=\tan(\frac{\pi}{4})=1$

Step 3: Design the analog filter. In this case the analog filter function is a first order LPF similar to an RC circuit →

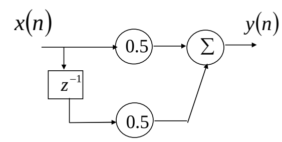
$$H(s)=\frac{1}{1+s}$$

TRANSFORMING AN RC CIRCUIT TO A DF (2)

Step 4: Apply the Bilinear Transform

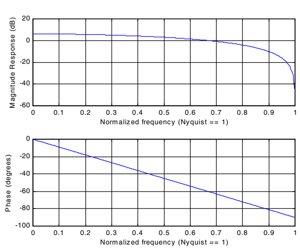
$$H(z)=H(s=\frac{z-1}{z+1})=\frac{1}{1+\frac{z-1}{z+1}}=0.5+0.5z^{-1}$$

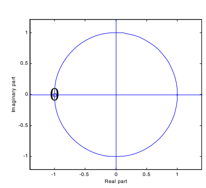
Step 5: Realization



TRANSFORMING AN RC CIRCUIT TO A DF (3)

Frequency Response $H(e^{j\Omega})=0.5+0.5e^{-j\Omega}$





Notice that there is no aliasing effect with the bilinear transformation. Although in this simple R-C example the resultant digital filter is FIR, more complex analog filters will yield IIR digital filters.

Analog Filter Designs

- •Butterworth - Maximally flat in passband
- •Chebyshev I - Equiripple in passband
- •Chebyshev II - Equiripple in stopband
- •Elliptic - Equiripple in passband and stopband

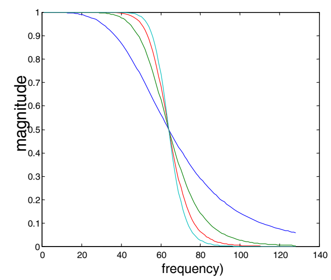
Butterworth Filter Design

- Maximally Flat in the Passband and Stopband

$$|H(\omega)|^2 = \frac{1}{1+(\frac{\omega}{\omega_C})^{2M}}$$

Butterworth Max. Flat in Passband and Stopband

Butterworth frequency response - transition is steeper as order increase



Butterworth Transfer Function

$$H(s)H(-s) = \frac{1}{1+(\frac{s}{j\omega_C})^{2M}}$$

Poles on a circle of radius ζ_c

$$(\frac{s}{j\omega_C})^{2M} = -1$$

$$s_k = (-1)^{1/2M}\, j\omega_C = \omega_C\, e^{\frac{j\pi(2k+M-1)}{2M}}$$

$$k = 0,1,..., 2M-1$$

note that roots can not fall on imaginary axis

Design Example - Butterworth

- A Butterworth filter is designed by finding the poles of H(s)H(-s)
- The poles fall on a circle with radius ζ_c
- The poles falling on the left hand s-plane (stable poles) are chosen to form H(s)
- H(s) is transformed to H(z)

Examples of IIR Filter Design Using MATLAB

FUNCTIONS IN THE SP TOOLBOX

IIR digital filter design.
butter - Butterworth filter design.
cheby1 - Chebyshev type I filter design.
cheby2 - Chebyshev type II filter design.
ellip - Elliptic filter design.
maxflat - Generalized Butterworth lowpass filter design.
yulewalk - Yule-Walker filter design.

IIR filter order selection.
buttord - Butterworth filter order selection.
cheb1ord - Chebyshev type I filter order selection.
cheb2ord - Chebyshev type II filter order selection.
ellipord - Elliptic filter order selection.

Butterworth Design in MATLAB

```
• % Design an IIR Butterworth filter
• clear
• N=256; %for the computation of N discrete frequencies
• Wp=0.4; %passband edge
• Ws=0.6; %stopband edge
• Rp=1; % max dB deviation in passband
• Rs=40; %min dB rejection in stopband
• [M,Wn]=buttord(Wp,Ws,Rp,Rs);
• [b,a]=butter(M,Wn);

• theta=[(2*pi/N).*[0:(N/2)-2]]; % precompute the set of discrete frequencies up
  to fs/2
• H=freqz(b,a,theta); % compute the frequency response
• plot(angle(H))
• pause
• H=(20*log10(abs(H))); % plot the magnitude of the frequency response
• plot(H)
• title('frequency response')
• xlabel('discrete frequency index (N is the sampling freq.)')
• ylabel('magnitude (dB)')
```

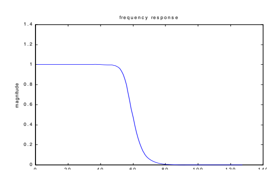

Butterworth Design in MATLAB (2)

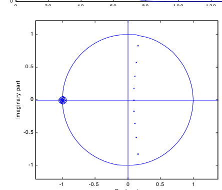
```
• Wp=0.4; %passband edge
• Ws=0.6; %stopband edge
• Rp=1; % max dB deviation in passband
• Rs=40; %min dB rejection in stopband
```

- b = 0.0021 0.0186 0.0745 0.1739 0.2609 0.2609 0.1739
- 0.0745 0.0186 0.0021

- a = 1.0000 -1.0893 1.6925 -1.0804 0.7329 -0.2722 0.0916
- -0.0174 0.0024 -0.0001

Butterworth Design in MATLAB (3)



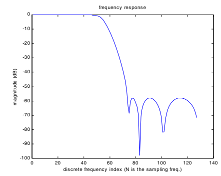


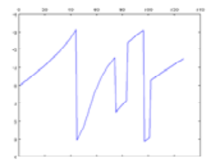
MATLAB Chebyshev II Design Example

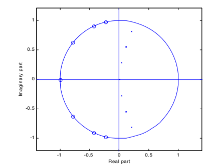
Chebyshev I - Equiripple in passband
Chebyshev II - Equiripple in stopband

```
% Design an IIR Chebyshev II filter - Ex 2.20

clear
N=256; %for the computation of N discrete frequencies

Wp=0.4; passband edge
Ws=0.5; stopband edge
Rp=1; ripple in passband (dB)
Rs=60; rejection (dB)

[M,Wn] = cheb2ord(Wp,Ws, Rp, Rs); % determine order

[b,a] = cheby2(M,58,Wn); %determine coefficients
size(a)
size(b)

% use routines to plot frequency response
```


IIR Chebyshev II Example







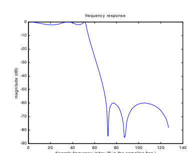
b = 0.0274 0.1065 0.2290 0.3252 0.3252 0.2290 0.1065 .0274

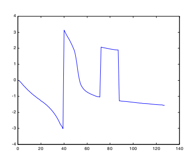
a =1.0000 -0.7484 1.2644 -0.4555 0.3427 -0.0454 0.0186 -0.0002

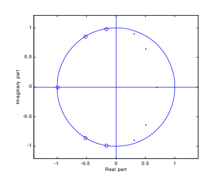
MATLAB Elliptic Design Example

```
% Design an IIR Elliptic filter
clear
N=256; %for the computation of N discrete frequencies

Wp=0.4; %passband edge
Ws=0.6; %stopband edge
Rp=2; % max dB deviation in passband
Rs=60; %min dB rejection in stopband
[M,Wn] = ellipord(Wp,Ws,Rp,Rs);
[b,a] = ellip(M,Rp,Rs,Wn); %design filter
size(a)
size(b)

theta=[(2*pi/N).*[0:(N/2)-2]]; % precompute the set of discrete frequencies up to fs/2
H=freqz(b,a,theta); % compute the frequency response
plot(angle(H))
pause
H=(20*log10(abs(H))); % plot the magnitude of the frequency response
plot(H)
title('frequency response')
xlabel('discrete frequency index (N is the sampling freq.)')
ylabel('magnitude (dB)')
pause
zplane(b,a) ;% z plane plot
```


IIR Elliptic







b = 0.0181 0.0431 0.0675 0.0675 0.0431 0.0181

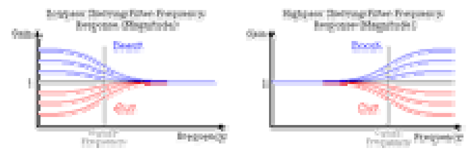
a = 1.0000 -2.3214 3.3196 -2.8409 1.5154 -0.4151

Introduction to Special types of Digital Filters

- Shelving Filter
- Peaking Filter
- Graphic Equalizer

Shelving Filters

Shelving filters realize tone controls in audio systems. The frequency response of a low-pass (bass) and high-pass (treble) shelving filter is shown below.



Frequency response of a low-pass and high-pass shelving filter.

Tone Control Block in J-DSP

The low frequencies are affected by bass adjustments with the audio signal processed through low-pass shelving filters. The high frequencies are affected by treble adjustments with the audio signal processed through high-pass shelving filters. A J-DSP simulation using the Tone Control .

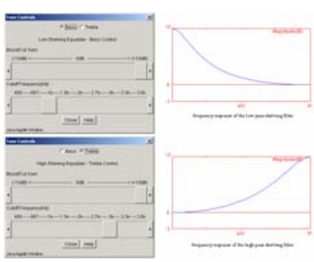


Figure J-DSP simulation using the tone control block.

Peaking Filter Block in J-DSP

A J DSP simulation using the Peaking Filter block is shown below

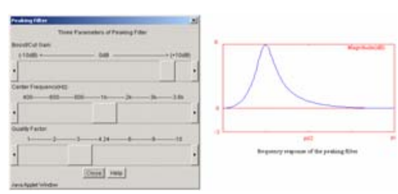
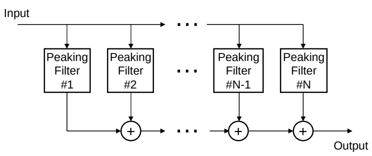


Figure 9: J-DSP simulation using the peaking filter block.

Graphic Equalizer

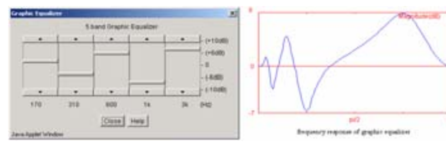
- A graphic equalizer uses a cascade of peaking filters
- It alters the frequency response of each band by varying the corresponding peaking filter's gain.



A diagram of a graphic equalizer with N bands of control.

Graphic Equalizer Block in J-DSP

A J DSP simulation using the Graphic Equalizer block is shown below. The sliders are a graphic representation of the frequency response applied to the input audio signal, hence the name "graphic" equalizer.



-DSP simulation using the graphic equalizer block.

FIR vs IIR Digital Filters

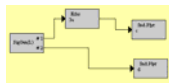
FIR	IIR
Always stable	Not always stable
Transversal	Recursive
All-zero model	All-pole or Pole-zero model
Moving Average(MA) model	Autoregressive(AR) or Autoregressive Moving Average (ARMA) model
Inefficient for spectral peaks	Efficient for spectral peaks (all-pole, pole-zero)
Efficient for spectral notches	All pole inefficient for spectral notches

FIR vs IIR Digital Filters (Cont.)

FIR	IIR
	Pole-zero efficient for both notches and peaks
Requires high order design	Generally requires lower order design
Less sensitive to finite word length implementation	More sensitive to finite word length implementation
Linear phase design	Generally non-linear phase

Digital Audio Filters (1)

Echo Effects



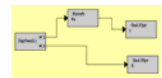
- The echo effect is obtained by mixing the input signal with its delayed version.
- The proportion of the delayed signal to the "clean" original signal determines how obvious the echo is, and the delay signifies the echo period.

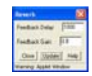
➔ ***y(n)= x(n) + b . x(n-R)***

- *R* = the number of echo delay in samples.
- In order to have a distinguishable echo, *R* should be relatively large.
- *b* is the attenuation constant ($|b| < 1$).

Digital Audio Filters (2)

Reverberation Effects

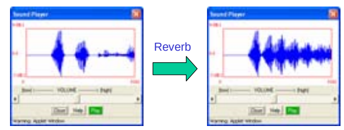




- Reverberation is obtained by mixing the input signal with the ***delayed*** versions of its ***feedback***.
- The effect of the feedback results in ***multiple echoes.***

➔ ***y(n) = x(n) + b . y(n-R)***

- R = feedback delay in samples.
- b is the attenuation constant ($|b| < 1$).



Other Methods for Digital Filtering

Median Filters

$$y(n) = \underset{i=0,1,2,\ldots,L}{median} \{x(n-i)\}$$

The median operation ranks the samples in the memory of the filter and picks the sample that falls in the middle of the rank and assigns it to the output *y(n)*

Used for impulsive noise. One application reported is scratch noise removal in vinyl record restoration

2-D Filters for Image Applications

- FIR 2-D filter

$$y(n_1,n_2) = \sum_{l=0}^{L} \sum_{m=0}^{M} h(n_1,n_2)x(n_1-l,n_2-m)$$

- IIR realizations also possible
- Theory very similar to 1-D and described in multidimensional signal processing books

Implementation of 2D Filters

LPF

HPF

Low-pass filtering of a natural image

High-pass filtering of a natural image

THE DISCRETE AND THE FAST FOURIER TRANSFORM

The DTFT of a finite sequence

if::

$$x(n) = \begin{cases} 1.....,0 \le n \le N-1 \\ 0.....,elsewhere \end{cases}$$

then

$$X(e^{j\Omega}) = \sum_{n=0}^{N-1} e^{-jn\Omega} = \frac{1-e^{-jN\Omega}}{1-e^{-j\Omega}}$$

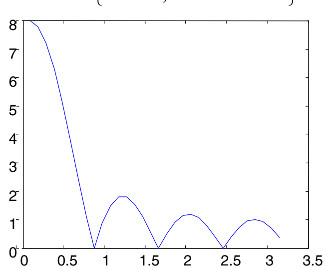
or

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

Remark: The sin(.)/sin(.) function is known as a *digital sinc* or a *Dirichlet* function.

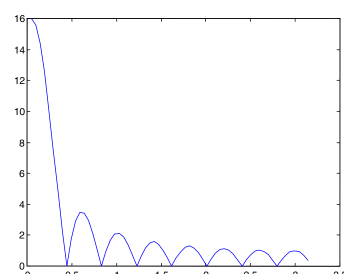
The DTFT of a finite sequence (Cont.)

$$x(n) = \begin{cases} 1.....,0 \le n \le 7 \\ 0.....,elsewhere \end{cases}$$



The DTFT of a finite sequence (Cont.)

$$x(n) = \begin{cases} 1.....,0 \le n \le 15 \\ 0.....,elsewhere \end{cases}$$



The Discrete Fourier Transform (DFT)

$$X(k)=\sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N} \quad \text{and} \quad k = 0, 1, \ldots, N-1$$

The inverse Discrete Fourier Transform (IDFT) of the sequence *x(n)*

$$x(n)=\frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j2\pi kn/N} \quad \text{and} \quad n = 0, 1, \ldots, N-1$$

The DFT transform pair is denoted by

$$\{x(n)\}\leftrightarrow\{X(k)\}$$

The DFT Matrix

The DFT and the IDFT may be expressed in terms of matrices, i.e.,

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ . \\ . \\ X(N-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \zeta^{-1} & \zeta^{-2} & \cdots & \zeta^{-(N-1)} \\ 1 & \zeta^{-2} & \zeta^{-4} & \cdots & \zeta^{-2(N-1)} \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ 1 & \zeta^{-(N-1)} & \zeta^{-2(N-1)} & \cdots & \zeta^{-(N-1)^2} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ . \\ . \\ x(N-1) \end{bmatrix}$$

where $\zeta^{-k} = e^{-j2\pi k/N}$ and $\underline{F}^{-1} = \frac{1}{N}\underline{F}^{H}$

a more compact form $\underline{X} = \underline{F}\,\underline{x}$ and $\underline{x} = \underline{F}^{-1}\underline{X}$

The DFT Matrix (2)

N=2, N=4, and N=8

$$\mathbf{F} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -0.5-j0.866 & -0.5+j0.866 \\ 1 & -0.5+j0.866 & -0.5-j0.866 \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

Selected Properties of the DFT

Linearity: $\{\alpha x(n)+\beta y(n)\} \leftrightarrow \{\alpha X(k)+\beta Y(k)\}$

Shifting: $\{x(n-m) \bmod N\} \leftrightarrow e^{-j2\Pi km/N}\{X(k)\}$

Circular Convolution: $x(n)\otimes h(n) \leftrightarrow X(k)H(k)$

where $x(n)\otimes h(n) = \sum_{m=0}^{N-1} h(m)\, x((n-m)_{\text{mod}\,N})$

Freq. Circular Convolution: $x(n)w(n) \leftrightarrow \frac{1}{N} X(k)\otimes W(k)$

Parseval's Theorem: $\sum_{n=0}^{N-1}|x(n)|^2 = \frac{1}{N}\sum_{k=0}^{N-1}|X(k)|^2$

Frequency resolution of the DFT

The frequency resolution of the N-point DFT is

$$f_r = \frac{f_s}{N}$$

- The DFT can resolve exactly only the frequencies falling exactly at: k fs/N. There is spectral leakage for components falling between the DFT bins
- Typically we use an FFT that is as large as we can afford
- Zero-padding is often use to provide more resolution in the frequency components
- Zero padding is often combined with tapered windows

Spectral Estimates over Finite-time Data windows

Frequency domain representations are appropriately defined by the Fourier Transform integrals over an infinite time span.

The DFT, however, estimates the spectrum over finite time

The DFT essentially applies a window to truncate the data.

The simplest data window is the rectangular (boxcar).

Truncation in time is convolution in frequency

The frequency domain characteristics of the data window, namely its bandwidth and sidelobes, affect the DFT spectral estimate.

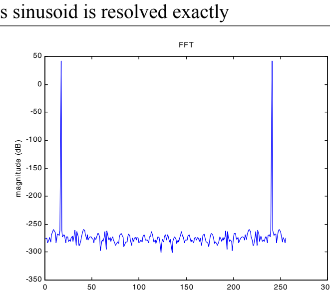
WINDOWS

The spectral characteristics of the window affect the spectral estimates. The rectangular window has the narrowest mainlobe width but the wordt sidelobes. Tapered windows have wider mainlobe width but better behaved bandwidth.

N-point Window	Mainlobe width	Sidelobe Level
Rectangular	$4\pi/(N+1)$	-13 dB
Triangular	$8\pi/N$	- 25 dB
Hamming	$8\pi/N$	- 41 dB
Hanning	$8\pi/N$	- 31 dB
Blackman	$12\pi/N$	- 57 dB

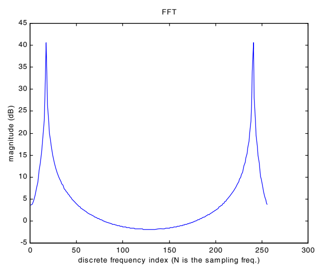
FFTs of Sinusoidal Signals (1)

256-point FFT of a 500 Hz sinusoid (fs=8 kHz). Notice that this sinusoid is resolved exactly



FFTs of Sinusoidal Signals (2)

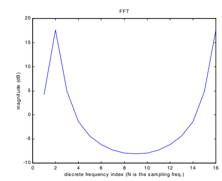
256-point FFT of a 510 Hz sinusoid (fs=8 kHz). Notice that this sinusoid is NOT resolved exactly

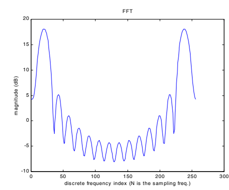


Jan. 2009 Copyright (c) 2009 Andreas Spanias VII-13

Zero-padded FFTs of Sinusoidal Signals (2)

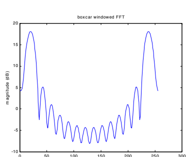
16-point FFT of a 16-point 590 Hz sinusoid (fs=8 kHz).
Vs
256-point FFT of a 16-point 590 Hz sinusoid (fs=8 kHz).
Notice that although this sinusoid is NOT resolved exactly the frequency of the peak in the zero-padded case is closer to actual



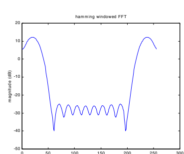


Jan. 2009 Copyright (c) 2009 Andreas Spanias VII-14

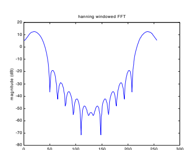
Windowed FFTs on Sinusoids



boxcar.



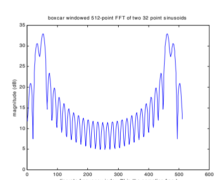
hamming



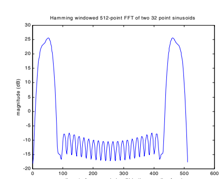
hanning

Jan. 2009 Copyright (c) 2009 Andreas Spanias VII-15

Window Comparison with Closely Spaced Periodicities

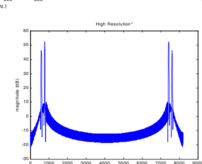


boxcar.



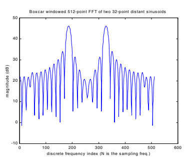
hamming

high resolution ---->

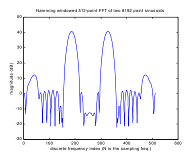


Jan. 2009 Copyright (c) 2009 Andreas Spanias VII-16

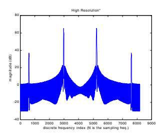
Window Comparison with Distant Periodicities



boxcar.



hamming



High Resolution

The FFT-DIT Algorithm

The FFT decimates the sequence and performs a DFT by processing results of smaller size DFTs. This is done by decomposing the N-point DFT to 2-point DFTs and using "butterfly" operations to obtain the result. For a Decimation in Time (DIT) FFT algorithm the following steps are taken:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N}$$

By decimating x(n) we can write

$$X(k) = \sum_{n=0}^{(N/2)-1} x(2n) e^{-j2\pi 2nk/N} + \sum_{n=0}^{(N/2)-1} x(2n+1) e^{-j2\pi(2n+1)k/N}$$

The FFT-DIT Algorithm (Cont.)

if we define $x_1(n) = x(2n)$ $\quad x_2(n) = x(2n+1)$ $\quad W_N^{nk} = e^{-j2\pi nk/N}$

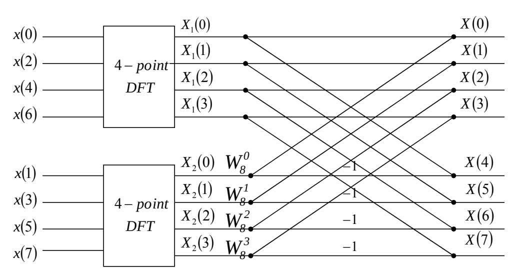
$$X_1(k) = \sum_{n=0}^{(N/2)-1} x_1(n) W_{N/2}^{nk} \qquad X_2(k) = \sum_{n=0}^{(N/2)-1} x_2(n) W_{N/2}^{nk}$$

and

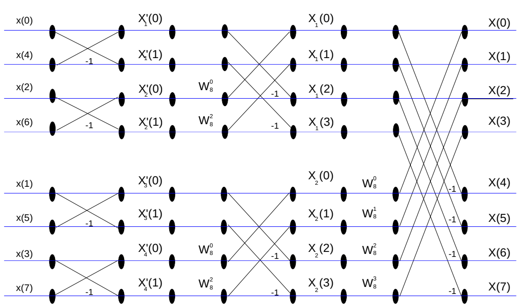
$$X(k) = X_1(k) + W_N^k X_2(k), \qquad k = 0,1,\ldots,N/2-1$$
$$X(k+N/2) = X_1(k) - W_N^k X_2(k)$$

Remarks: The N-point DFT is broken down to two N/2-point DFTs. We then write the N/2-point DFTs as a combination of two N/4-point DFTs and so forth.

The FFT-DIT Algorithm (Cont.)



The FFT-DIT Algorithm (Cont.)



The DFT and the FFT Complexity

The N-point DFT requires N^2 multiplications and $N^2 - 1$ additions to compute the discrete frequency spectrum.

The complexity of the DFT is reduced using the FFT to $N/2 \log_2 N$ multiplications and $N \log_2 N$ additions.

For example if N=4096 the DFT requires 16,777,216 multiplications while the FFT requires 49,152 multiplications.

FFT ALGORITHMS

IN FFT DECIMATION-IN-TIME
-the frequency-domain (output) indices are in place while the time-domain (input) indices are bit-reversed

IN FFT DECIMATION-IN-FREQUENCY
-the time-domain indices are in place while the frequency-domain indices are bit-reversed

VARIANTS OF FFT ALGORITHMS:
Low-Complexity "Pruned" FFTs
- For computing fewer frequency bins
- when time-domain values are systematically zero (ex: zero padded FFTs)

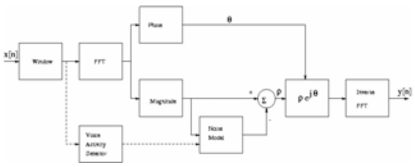
Radix 4 and Mixed-radix FFTs, Gortzel Algorithm (computes only one frq. bin), Rader, Prime Factor, Winograd, Zoom FFTs

Reference: E. Brigham, "The FFT and its Applications," Prentice Hall, NJ 1988
Links on FFT: http://www.fftw.org/links.html
FFT laboratory: http://sepwww.stanford.edu/oldsep/hale/FftLab.html

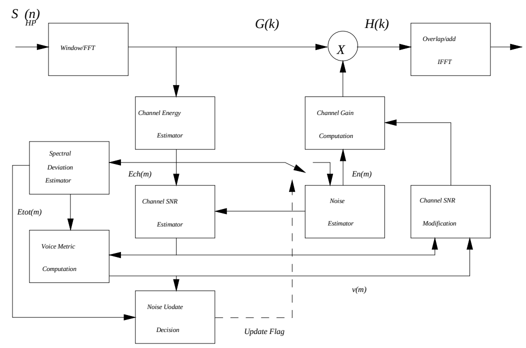
FFT Applications

FFT AND SPEECH ENHANCEMENT

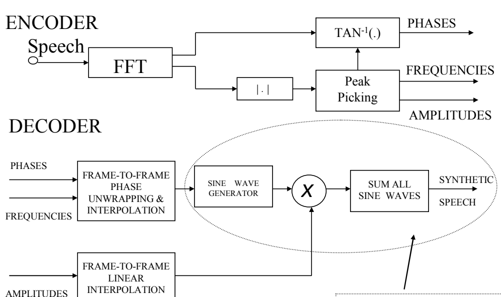
- If we have speech corrupted by background noise, the spectrum of background noise can be estimated during speech pauses.
- Speech is enhanced in the spectral domain by subtracting the estimated noise spectrum from the noisy speech spectrum
- FFT-based spectral subtraction is used in military applications
- Also used in CDMA cellular IS-127 telephony standard



FFT SPEECH ENHANCEMENT IN CDMA



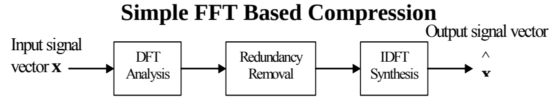
FFT AND SINUSOIDAL TRANSFORM CODING



Implemented by IFFT and overlap&add procedure

R. McAulay and T. Quatieri, "Sinsoidal Coding,"Ch. 4, Speech Coding and Synthesis, W.B. Kleijn and K. Paliwal, eds, Elsevier, 1995.

Simple FFT Based Compression



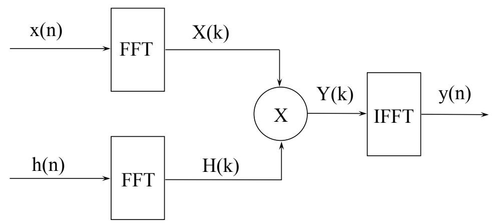
Fast Convolution Using the FFT

- Fast Circular Convolution

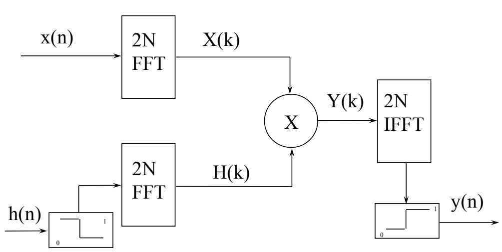
b. Fast Linear Convolution

FFTs are often used to compute efficiently convolutions of very long sequences. Such convolutions arise in adaptive filters that are used in noise and echo cancellation.

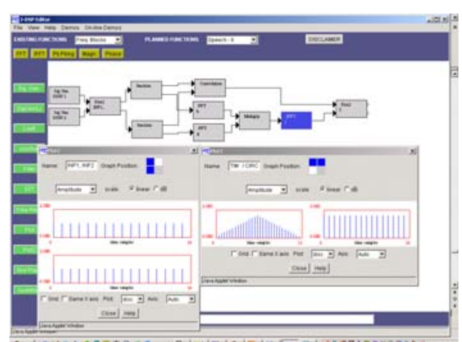
Fast N-point Circular Convolution



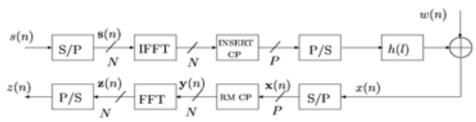
Fast N-point Linear Convolution using the Overlap/Save



J-DSP and Fast Convolution



Orthogonal Frequency Division Multiplexing (OFDM) and FFTs



$$z_k(n) = H(2\pi k / N)\, s_k(n) + w_k(n) \quad k = 1, \ldots, N$$
$$\mathbf{z}(n) = \mathbf{D}_{\mathrm{H}}\, \mathbf{s}(n) + \mathbf{v}(n)$$

