

Theoretical Digital Signal Processing

Duy-Ky Nguyen
© All Rights Reserved

12-06-96

1. Digital Signal Systems

Nowadays, most signal systems are implemented as digital systems rather than analog ones since digital systems are more flexible and they can implement highly complicated functions. Simple filters can be implemented with analog filters, however they are less reliable due to the drifting problem of op-amps and the leakage problem of capacitors, the aging problem of components such as resistors, capacitors etc. In this section, we first start with a reality of sampling process to convert an analog signal into digital signal to derive the *starred transform*. The *Z-transform* will be developed on the basis of this transform. Eventually, the *Z-transform* will define a mapping between the continuous-time *s*-plane and the discrete-time *z*-plane as a foundation to design a discrete-time controller for a digital signal system.

1.1. Some Definitions

In the real world, all signals are *analog signals* whose amplitudes take an *infinite* values within some ranges. Their values can be any real numbers with fraction and the difference of successive values is *infinitesimal*.

Analog signals are continuous-time signals. *Discrete-time signals*, or *discrete signals* for short, take finite values of any real numbers, hence the difference of successive values is finite and is a real number.

Digital signal is a subset of discrete-time signal. A values of a digital signal must be an *integer*, thus the difference of successive values is finite and is an integer. These integers are within the range determined by the length of registers in a ADC, the range is $[0 \sim 2^n]$ or $[-2^{n-1} \sim 2^{n-1} - 1]$ for a *n*-bit ADC.

Discrete-time control theory is used to design a discrete-time controller which is employed to implement a digital controller for a digital control system.

A continuous-time system is discretized using the discrete-time signal theory, there is no digitization of a system since it is a special case of discretization as a real number is rounded off to an integer. An analog signal is digitized, also known as quantized, by an ADC, there is no discretization of an analog signal since there is no such converter available nowadays.

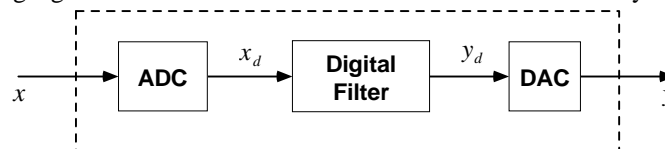


Fig. 1.1: Digital Signal System

where *x*, *y* are analog signals while *x_d*, *y_d* are digital signals. ADC and DAC are *analog-to-digital* and *digital-to-analog* converters, respectively. Digital filter can be either a micro-controller or a digital signal processor (DSP) where ADC and DAC are on-chip (internal) or a general purpose computer with external ADC/DAC. For 8-bit processors such as 8051 Intel family or 68HC11 Motorola family, ADC and DAC are also 8-bit. However, higher bit ADC's are expensive, 10-bit or 12-bit ADC are usually used for 16-bit processors such as 80196 Intel family, 68000 Motorola family, 32-bit 486-PC, 64-bit Pentium-PC.

1.2. ADC as Physical Sampler

Practically, ADC is a *zero-order-hold* sampler of *successive approximation* type

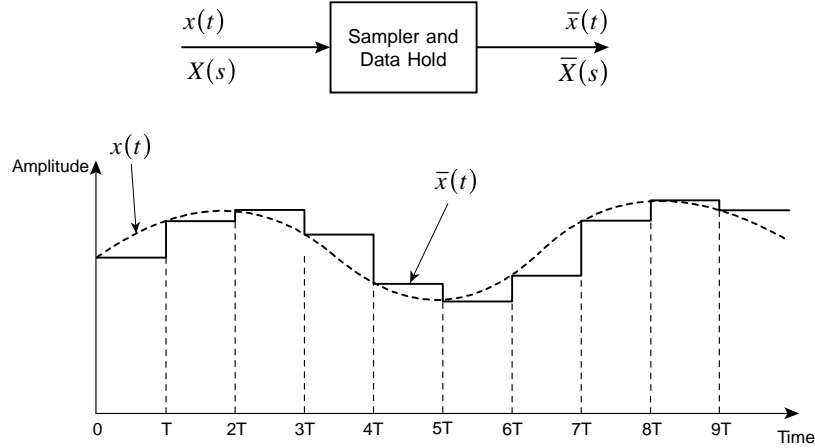


Fig. 1.2: Sample and Hold Signal

The *sampled and zero hold signal* $\bar{x}(t)$ can be mathematically expressed as

$$\bar{x}(t) = \sum_{k=-\infty}^{\infty} x(kT) \{ \mathcal{U}(t - kT) - \mathcal{U}[t - (k+1)T] \} \quad (1.1)$$

where $\mathcal{U}(t)$ is the unit step function, that is

$$\mathcal{U}(t) = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0 \end{cases} \quad (1.2)$$

Using the shifting theorem of Laplace transform gives

$$\bar{X}(s) = \sum_{k=-\infty}^{\infty} x(kT) \left\{ \frac{e^{-ksT}}{s} - \frac{e^{-(k+1)sT}}{s} \right\} = \left(\frac{1 - e^{-sT}}{s} \right) \sum_{k=-\infty}^{\infty} [x(kT)e^{-ksT}] \quad (1.3)$$

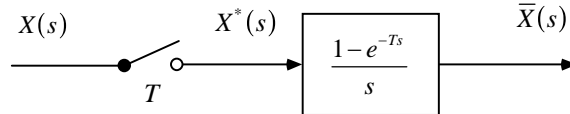
1.3. Starred Transform as Ideal Sampler

On the basis of Eq.(1.3), the *starred transform* is defined as

$$X^*(s) = \sum_{k=-\infty}^{\infty} [x(kT)e^{-ksT}] \quad (1.4)$$

then Eq.(1.3) can be written as

$$\bar{X}(s) = \left(\frac{1 - e^{-Ts}}{s} \right) X^*(s) \quad (1.5)$$



Eq.(1.4) gives the inverse Laplace transform of $X^*(s)$ as

$$x^*(t) = \mathcal{L}^{-1} \{ X^*(s) \} = \sum_{k=-\infty}^{\infty} x(kT) \delta(t - kT) \quad (1.6)$$

or in the *linear convolution* form

$$x(n) = \sum_{k=-\infty}^{\infty} x(k) \delta(n - k) = x(n) * \delta(n) \quad (1.7)$$

where $t = nT$ and $\delta(t)$ is a Dirac delta function.

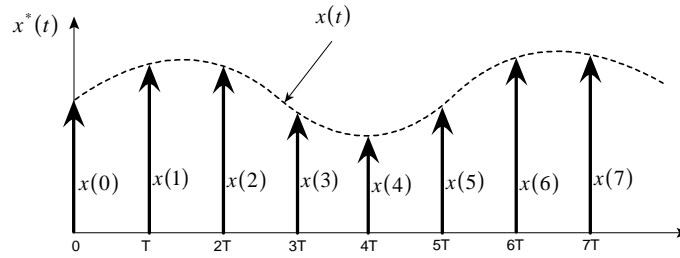


Fig. 1.3: A representation of $e^*(t)$

1.4. Sampling Characteristics and Signal Reconstruction

Sampling characteristics and signal reconstruction is determined by the following theorem

Theorem 1: Laplace Transform of Sampled Signal

An alternative expression for the starred transform is given by

$$X^*(s) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X(s + jk\omega_s) \quad (1.8)$$

thus the Laplace transform of a sampled signal is equal to infinite sum of Laplace transform of the Laplace transform of its original signal. The s -plane may be divided into a *primary strip* and *complementary strips* as shown in Fig. 1.4.

Proof

Let

$$\mathcal{P}(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT) \quad (1.9)$$

then Eq.(1.6) can be written as

$$x^*(t) = x(t) \cdot \mathcal{P}(t) \quad (1.10)$$

Since $\mathcal{P}(t)$ is a periodic function $\delta(t)$ whose period is T , its Fourier series is

$$\mathcal{P}(t) = \sum_{k=-\infty}^{\infty} \left(\frac{1}{T} \int_0^T \delta(t) e^{-j2\pi k\tau/T} d\tau \right) e^{j2\pi k t/T} = \frac{1}{T} \sum_{k=-\infty}^{\infty} e^{jk\omega_s t} \quad (1.11)$$

where

$$\omega_s = \frac{2\pi}{T} \quad (1.12)$$

thus Eq.(1.10) can be read as

$$x^*(t) = \frac{1}{T} \sum_{k=-\infty}^{\infty} x(t) e^{jk\omega_s t} \quad (1.13)$$

and its Laplace transform is

$$X^*(s) = \mathcal{L}\{x^*(t)\} = \int_0^{\infty} \frac{1}{T} \sum_{k=-\infty}^{\infty} x(t) e^{-(s-jk\omega_s)t} dt = \frac{1}{T} \sum_{k=-\infty}^{\infty} \int_0^{\infty} \frac{1}{T} x(t) e^{-(s-jk\omega_s)t} dt = \frac{1}{T} \sum_{k=-\infty}^{\infty} X(s + jk\omega_s)$$

Q.E.D.

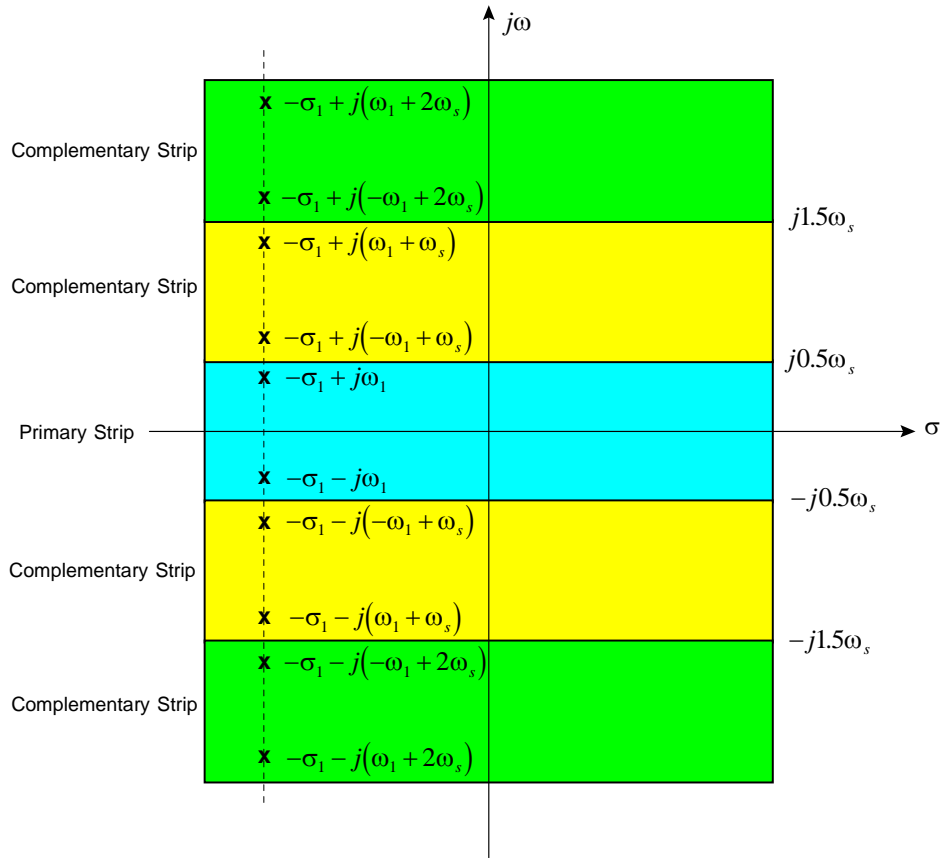


Fig. 1.4: Primary and Complementary Strips

1.4.1. Signal Reconstruction

To see the significance of Theorem 1.1, consider the case of frequency response, when $s = j\omega$

$$X^*(s) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X[j(\omega + k\omega_s)] \quad (1.14)$$

thus the spectrum of sampled signal consists of an infinite number of replicas of the analog signal spectrum scaled by the factor $1/T$ and the frequency shifted by multiples of ω_s .

If the sampling time T is low enough such that $\omega_{\max} < \omega_s/2$, then the sampled spectrum retains the shape of the analog one and the analog signal can be reconstructed from its sampled spectrum within the band $[-\omega_s/2, \omega_s/2]$.

For signal reconstruction, the sampling rate must satisfy the sampling theorem, also known as Shannon Sampling Theorem, state that the sampling frequency ω_s must be at least equal twice the value of the highest significant frequency in the signal. Since an ideal low-pass reconstruction filter cannot be implemented, one rule of thumb is to choose T as one-tenth of the smallest process time constant or the desired closed-loop time constant. Another convenient rule suggests sampling at the rate of 6 to 10 times per cycle.

Proposition 1: Choice of Sampling Time

In this work, the sampling time is chosen such that $\omega_s \geq 10\omega_{\max}$. With this choice, ignoring the sampling effect will introduce a maximum tolerance of 5%.

Proof:

We have

$$e^{-sT} = \frac{e^{-sT/2}}{e^{sT/2}} = \frac{1 - \frac{sT}{2} + \frac{(sT)^2}{2^2 2!} - \dots}{1 + \frac{sT}{2} + \frac{(sT)^2}{2^2 2!} + \dots} \approx \frac{1 - \frac{sT}{2}}{1 + \frac{sT}{2}}$$

then Eq.(1.5) gives

$$G_h(s) = \frac{1 - e^{-sT}}{s} \approx \frac{1}{s} \left(1 - \frac{sT}{2} \right) = \frac{T}{\frac{T}{2}s + 1}$$

Since the overall DC gain will be determined at the final design stage, the sampling time will be discarded to have

$$G_h(s) \approx \frac{1}{\frac{T}{2}s + 1}$$

thus ignoring the sampling effect will introduce a maximum tolerance of

$$G_h(j\omega_{\max}) \approx \frac{1}{1 + j\frac{0.1}{2}} = \frac{1}{1 + j0.05}$$

Q.E.D

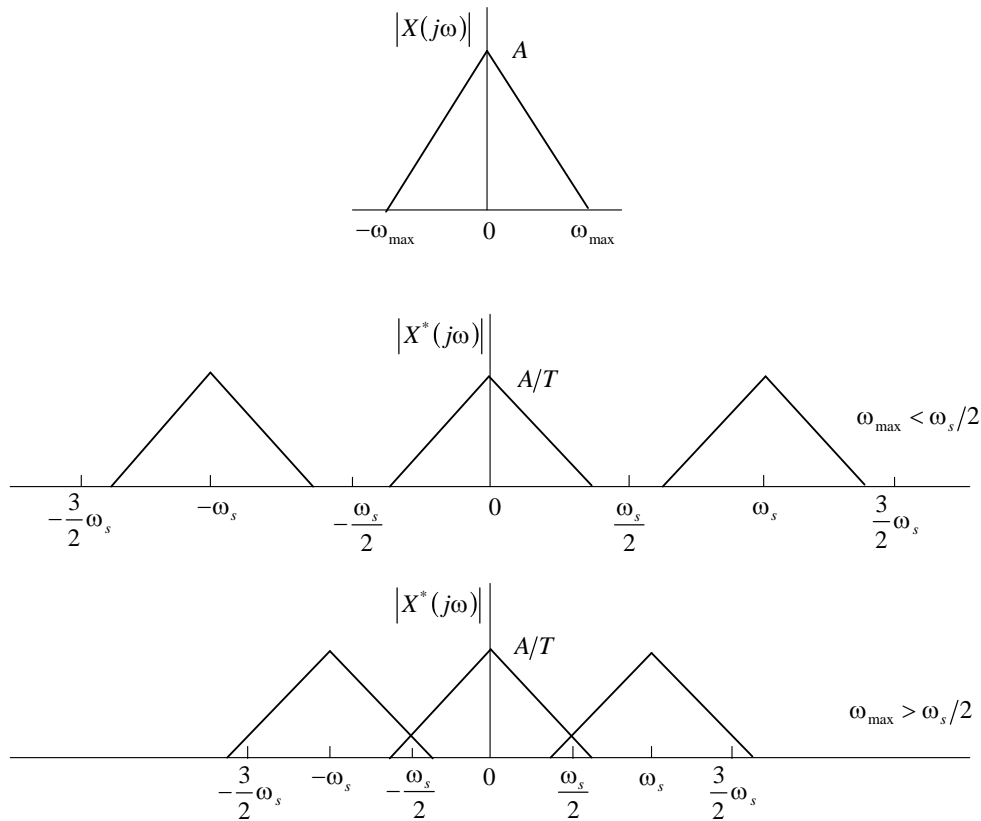


Fig. 1.5: Spectrum of Analog and Sampled Signal

1.4.2. Aliasing or Folding

Eq.(1.7) reveals that if $X(s)$ has a pole at p_0 , then $X^*(s)$ will have poles at $(p_0 + jn\omega_s)$ where $n = 0, \pm 1, \pm 2, \dots$. For example, suppose that $X(s)$ has one pair of poles located at $-a \pm j0.6\omega_s$, outside the primary strip. The sampling will then “fold” this pair back into the primary strip, to the location $-a \pm j0.4\omega_s$. So $x^*(t)$ contains components inside the primary strip that do not occur in $x(t)$. This property applies for poles only, not for zeros, since poles of a transfer function component in Eq.(1.7) will be poles of the whole transfer function; however, zeros of a component are not necessarily the zeros of the whole.

An anti-aliasing analog filter is used to solve this problem by filtering out all noise. It is a low-pass filter with the cut-off frequency is based on the highest frequency of the analog signal.

1.5. Z-Transform

On the basis of Eq.(1.4), the *Z transform* is defined as

$$X(z) = \mathcal{Z}\{x(k)\} = \sum_{k=-\infty}^{\infty} x(k)z^{-k} \quad (1.15)$$

where the mapping between the *s*-plane and the *z*-plane is defined as

$$z = e^{sT} \Leftrightarrow s = \frac{1}{T} \ln(z) \quad (1.16)$$

since comparing Eqs.(1.4) and (1.15) yields

$$X^*(s) = X(z) \Big|_{z=e^{Ts}} \quad (1.17)$$

The Laplace transform is used in obtaining a *control transfer function* for implementing a analog controller. We will see that the Z-transform will be employed in obtaining a *control difference equation* for implementing a digital controller.

Three of the most important properties of the Z-transform are the linearity, the real translation and the linear convolution as in Eq.(1.7) are given below

$$\mathcal{Z}\{ax_1(k) + bx_2(k)\} = a\mathcal{Z}\{x_1(k)\} + b\mathcal{Z}\{x_2(k)\} \quad (1.18)$$

and

$$\mathcal{Z}\{x(k-n)\} = z^{-n} \mathcal{Z}\{x(k)\} \quad (1.19)$$

and

$$\mathcal{Z}\{x(k) * y(k)\} = \mathcal{Z}\{x(k)\} \times \mathcal{Z}\{y(k)\} = X(z) \times Y(z) \quad (1.20)$$

where n is a positive integer.

Since by the definition of the Z-transform, we have

$$\mathcal{Z}\{ax_1(k) + bx_2(k)\} = \sum_{k=-\infty}^{\infty} [ax_1(k) + bx_2(k)]z^{-k} = a \sum_{k=-\infty}^{\infty} x_1(k)z^{-k} + b \sum_{k=-\infty}^{\infty} x_2(k)z^{-k} = a\mathcal{Z}\{x_1(k)\} + b\mathcal{Z}\{x_2(k)\}$$

and

$$\mathcal{Z}\{x(k-n)\} = \sum_{k=-\infty}^{\infty} [x(k-n)]z^{-k} = \sum_{m=-\infty}^{\infty} x(m)z^{-(n+m)} = z^{-n} \sum_{m=-\infty}^{\infty} x(m)z^{-m} = z^{-n} \mathcal{Z}\{x(k)\}$$

and

$$\mathcal{Z}\{x(k) * y(k)\} = \mathcal{Z}\left\{\sum_{n=-\infty}^{\infty} x(n)y(k-n)\right\} = \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(n)y(k-n)z^{-k} = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(n)y(m)z^{-(m+n)} = \sum_{m=-\infty}^{\infty} x(n)z^{-n} \sum_{n=-\infty}^{\infty} y(m)z^{-m} = X(z) \times Y(z)$$

where $m = k - n$

1.6. Computation of Z-Transform

If the sampling process is taken into account, we have

$$\mathcal{Z}\left\{\frac{1-e^{-sT}}{s}G(s)\right\} = \mathcal{Z}\left\{\frac{1-z^{-1}}{s}G(s)\right\} = (1-z^{-1})\mathcal{Z}\left\{\frac{1}{s}G(s)\right\} \quad (1.21)$$

1.7. Some S-Z Transformations

The Z-transform defines the mapping in Eq.(1.16) as the exact transformation between the s -plane and the z -plane. This transform should be used to convert a continuous-time system into a discrete-time one. A direct substituting Eq.(1.16) into a transfer function in s -plane will produce an infinite difference equation due to the Taylor expansion of the natural logarithmic function. This necessitates an approximate transformation to obtain finite difference equations if the substitution method is used in place of the z -transform. From the single term s , we can have s and $1/s$ corresponding to a differentiator and an integrator.

The backward difference transformation approximates a differentiator as

$$\frac{d}{dt} y(t) \approx \frac{y(t) - y(t-T)}{T}$$

taking Laplace transform gives

$$sY(s) \approx \frac{Y(s) - e^{-Ts}Y(s)}{T}$$

so

$$s \approx \frac{1 - e^{-Ts}}{T} = \frac{1 - z^{-1}}{T} \Leftrightarrow z \approx \frac{1}{1 - Ts} \quad (1.22)$$

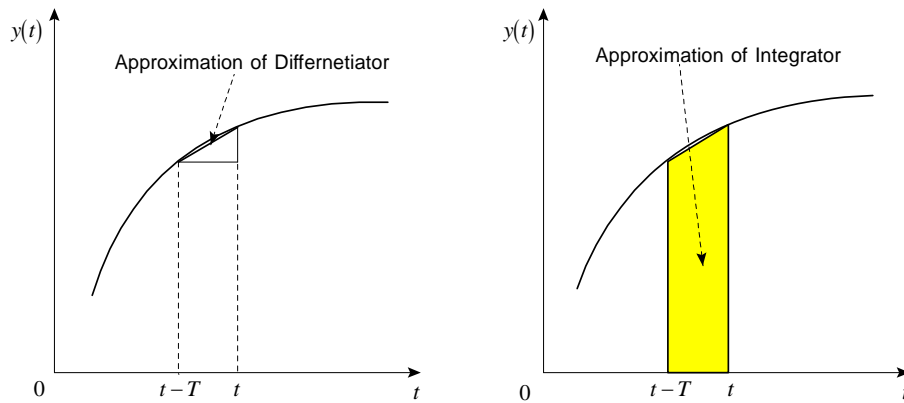


Fig. 1.6: Approximate Transformations for Differentiator and Integrator

The bilinear transformation approximates an integrator as the area of the trapezoidal

$$\int_0^t y(\tau) \cdot d\tau - \int_0^{t-T} y(\tau) \cdot d\tau \approx \frac{T}{2} [y(t-T) + y(t)]$$

taking Laplace transform gives

$$\frac{1}{s} Y(s) - \frac{1}{s} e^{-Ts} Y(s) \approx \frac{T}{2} [e^{-Ts} Y(s) + Y(s)]$$

thus

$$\frac{1}{s} \approx \frac{T}{2} \frac{1 + e^{-Ts}}{1 - e^{-Ts}} = \frac{T}{2} \frac{1 + z^{-1}}{1 - z^{-1}} \Leftrightarrow z \approx \frac{(T/2) + s}{(T/2) - s} \quad (1.23)$$

The stability region in the s -plane is the RHP, to find its mapping in the z -plane, substituting the frequency contour $s = j\omega$ into Eqs.(1.15), (1.22) and (1.23) produces

$$z = e^{j\omega T} \quad (1.24)$$

$$z = \frac{1}{1 - j\omega T} = \frac{1}{2} + \frac{1}{2} \frac{1 + j\omega T}{1 - j\omega T} = \frac{1}{2} + \frac{1}{2} e^{j\theta}, \quad \theta = \tan^{-1}(2\omega T) \quad (1.25)$$

$$z \approx \frac{(T/2) + j\omega T}{(T/2) - j\omega T} = e^{j\theta}, \quad \theta = \tan^{-1}(2\omega T) \quad (1.26)$$

Remark 1: Approximation Transformation in place of Z-Transform

Approximation transformations are used to avoid using z -transform, the exact transformation is defined by z -transform in Eq.(1.16)

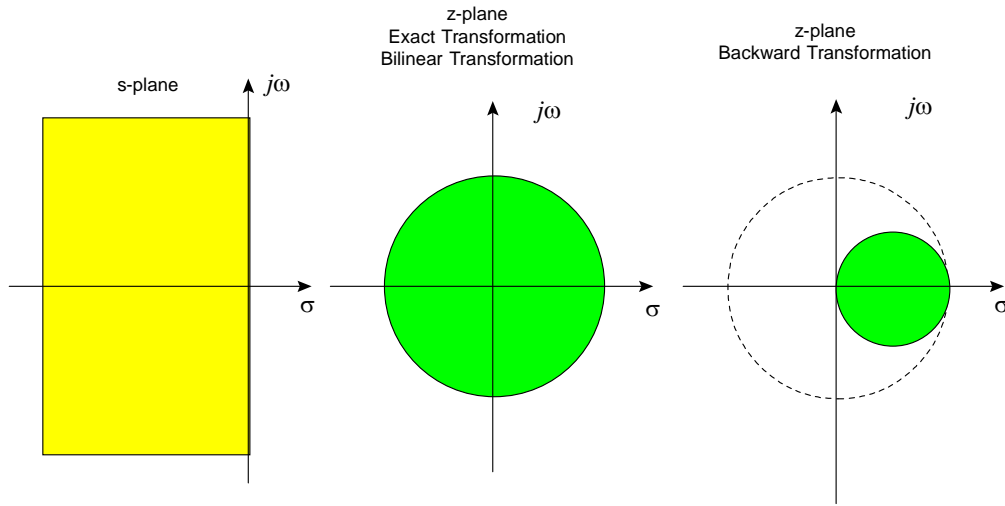


Fig. 1.7: Stability Regions

Thus a stable controller could be unstable under the backward difference transformation. This is a limitation of this transformation.

1.8. Linear Time-Invariant Systems



For a linear time invariant system, by Eq.(1.7) we have

$$y(n) = LTI\{x(n)\} = LTI\left\{\sum_{k=-\infty}^{\infty} x(k)\delta(n-k)\right\} = \sum_{k=-\infty}^{\infty} x(k) \times LTI\{\delta(n-k)\}$$

or

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = x(n) * h(n) \quad (1.27)$$

where

$$h(n) = LTI\{\delta(n)\} \quad (1.28)$$

is defined as an *impulse response* of a linear time invariant system.

In view of Eq.(1.20), Eq.(1.27) can be written as

$$Y(z) = H(z).X(z) \quad (1.29)$$

where $H(z)$ is defined as the *impulse transfer function* of a linear time invariant system.

2. Fourier Analysis

2.1. Fourier Series

For a *periodic* function, we have the following theorem to compute Fourier series

Theorem 2: Fourier Series

A function of period P has a Fourier series as

$$x(t) = \sum_{n=-\infty}^{\infty} c_n e^{j2\pi nt/P} \quad (2.1)$$

where

$$c_n = \frac{1}{P} \int_0^P x(t) e^{-j2\pi nt/P} dt = \frac{1}{P} \int_{-P/2}^{P/2} x(t) e^{-j2\pi nt/P} dt \quad (2.2)$$

Proof:

Multiplying Eq.(2.1) with $e^{-j2\pi mt/P}$ and taking integral gives

$$\int_0^P x(t) e^{-j2\pi mt/P} dt = \int_0^P \left(\sum_{n=-\infty}^{\infty} c_n e^{j2\pi nt/P} \right) e^{-j2\pi mt/P} dt = \sum_{n=-\infty}^{\infty} \left(c_n \int_0^P e^{j2\pi(n-m)t/P} dt \right)$$

or

$$\int_0^P x(t) e^{-j2\pi mt/P} dt = c_m P + \sum_{\substack{n=-\infty \\ n \neq m}}^{\infty} \left(c_n \frac{P}{j2\pi(n-m)} \left[\underbrace{e^{j2\pi(n-m)t/P}}_0 - e^0 \right] \right)$$

Q.E.D.

2.2. 2.2. Fourier Transform

For an *aperiodic* function, we have the following theorem to compute Fourier transform

Theorem 3: Fourier Transform

An aperiodic function has a Fourier transform pair as

$$X(w) = \int_{-\infty}^{\infty} x(t) e^{-jw t} dt \quad (2.3)$$

and

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(w) e^{jw t} dw \quad (2.4)$$

Proof:

We will start from Fourier series of a periodic function, then let the period approach ∞ . Eqs.(2.1) and (2.2) give

$$x(t) = \sum_{n=-\infty}^{\infty} \left(\frac{1}{P} \int_{-P/2}^{P/2} x(t) e^{-j2\pi nt/P} dt \right) e^{j2\pi nt/P} \quad (2.5)$$

Let

$$w = n \frac{2\pi}{P} \Rightarrow \Delta w = \frac{2\pi}{P} \quad (2.6)$$

then Eq.(2.5) can be read as

$$x(t) = \frac{1}{2\pi} \sum_{n=-\infty}^{\infty} \left(\int_{-P/2}^{P/2} x(t) e^{-jw t} dt \right) e^{jw t} \Delta w \quad (2.7)$$

Let $P \rightarrow \infty \Rightarrow \Delta w \rightarrow 0$, Eq.(2.7) becomes

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} x(t) e^{-jw t} dt \right) e^{jw t} dw \quad (2.8)$$

Q.E.D.

2.3. Discrete-Time Fourier Transform (DTFT)

Taking Fourier transform of Eq.(1.6) gives

$$X^*(j\omega) = \int_{-\infty}^{\infty} x^*(t) e^{-j\omega t} dt = \int_{-\infty}^{\infty} \sum_{k=-\infty}^{\infty} [x(k)\delta(t-kT)] e^{-j\omega t} dt = \sum_{k=-\infty}^{\infty} x(k) \int_{-\infty}^{\infty} \delta(t-kT) e^{-j\omega t} dt = \sum_{k=-\infty}^{\infty} x(k) e^{-j\omega kT}$$

thus

$$X^*(j\omega) = \sum_{k=-\infty}^{\infty} x(k) e^{-j\omega kT} = X(z) \Big|_{z=e^{j\omega T}} = X(e^{j\omega T}) \quad (2.9)$$

Let Ω be a *relative frequency*, that is

$$\Omega = \omega T = \frac{\omega}{f_s} = \frac{2\pi f}{f_s} \quad (2.10)$$

then a DTFT is defined as

$$X(e^{j\Omega}) = \sum_{k=-\infty}^{\infty} x(k) e^{-jk\Omega} \quad (2.11)$$

as

$$X(e^{j\Omega}) = \sum_{k=-\infty}^{\infty} x(k) e^{-jk\Omega} = \sum_{k=-\infty}^{\infty} x(k) e^{-jk(\Omega+2\pi n)} = X(e^{j(\Omega+2\pi n)}) \quad (2.12)$$

Eq.(1.7) is valid for all signal. For a signal which is bandlimited to a frequency range below π/T rad/sec ($f_s/2$ Hz), $X(j\omega)$ is zero for all values of ω with $|\omega| \geq \pi/T$. It follows that

$$X^*(j\omega) = \frac{1}{T} X(j\omega), \quad -\frac{\pi}{T} \leq \omega \leq \frac{\pi}{T} \quad (2.13)$$

Eqs.(2.4), (2.9), and (2.13) give

$$x(k) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\omega) e^{jkT\omega} d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} TX^*(j\omega) e^{jkT\omega} d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} X^*(j\omega) e^{jk\Omega} d\Omega$$

By Eq.(2.10), the *inverse DTFT* is

$$x(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\Omega}) e^{jk\Omega} d\Omega \quad (2.14)$$

Remark 2: Range of Ω

Eq.(2.14) reveals

$$-\pi \leq \Omega \leq \pi$$

and it conforms with the sampling theorem by Eq.(2.10)

2.4. Discrete Fourier Transform (DFT)

The discrete-time Fourier transform is applied for an infinite sequence of samples, while a *discrete Fourier transform* defined in this section should be used for an N -sample sequence.

It would be sufficient to store M values of $X(e^{j\Omega})$ only in the range $0 \leq \Omega < 2\pi$ since it is repetitive at intervals of 2π . Let

$$\Omega_n = \frac{2\pi n}{M} \quad (2.15)$$

and Eq.(2.10) gives

$$X(e^{j\Omega_n}) = \sum_{k=0}^{N-1} x(k) e^{-jk\Omega_n} \quad (2.16)$$

Multiplying Eq.(2.16) by $e^{jm\Omega_n}$ and summing over the block of M frequency domain samples

$$\sum_{n=0}^{M-1} X(e^{j\Omega_n}) e^{jm\Omega_n} = \sum_{k=0}^{M-1} \sum_{k=0}^{M-1} x(k) e^{j(m-k)\Omega_n} = \sum_{k=0}^{M-1} x(k) \sum_{n=0}^{M-1} e^{j2\pi n(m-k)/M} = Mx(m) \quad (2.17)$$

since it can be shown by summing the geometric series that

$$\sum_{n=0}^{M-1} e^{j2\pi n(m-k)/M} = \begin{cases} M, & \text{if } m = k \\ 0, & \text{if } m \neq k \end{cases} \quad (2.18)$$

The discrete Fourier transform is defined with $M = N$, ie. from Eq.(2.16)

$$X(n) = \sum_{k=0}^{N-1} x(k) W^{nk}, \quad n = 0, 1, \dots, N-1 \quad (2.19)$$

and from Eq.(2.17)

$$x(k) = \frac{1}{N} \sum_{n=0}^{N-1} X(n) W^{-nk}, \quad k = 0, 1, \dots, N-1 \quad (2.20)$$

where Ω_n is replaced by Eq.(2.15) with $M = N$, ie.

$$\Omega_n = \frac{2\pi n}{N} \quad (2.21)$$

and

$$W = e^{-j2\pi/N} \quad (2.22)$$

Note that

$$W^{mN} = e^{-j2\pi m} = 1, \quad m = 0, \pm 1, \pm 2, \dots \quad (2.23)$$

By Eq.(2.23), DFT in Eq.(2.19) has the periodic property

$$X(n + mN) = X(n), \quad m = 0, \pm 1, \pm 2, \dots \quad (2.24a)$$

and IDFT in Eq.(2.20) has the periodic property

$$x(n + mN) = x(n), \quad m = 0, \pm 1, \pm 2, \dots \quad (2.24b)$$

and for real sequence

$$X(N-n) = \bar{X}(n) \Leftrightarrow \begin{cases} \text{Re}\{X(N-n)\} = \text{Re}\{X(n)\} \\ \text{Im}\{X(N-n)\} = -\text{Im}\{X(n)\} \end{cases} \quad (2.25)$$

By Eq.(1.14) illustrated by Fig. 1.5, N corresponds to the sampling frequency, that is

$$N \leftrightarrow F_s \quad (2.26)$$

Remark 3: Spectrum Analyzer

By Eqs.(2.25) and (2.26), it is sufficient to have $N/2$ points in a spectrum of a real signal.

2.4.1. Direct DFT Computation

In general, let

$$x(k) = u(k) + jv(k) \quad (2.27)$$

and

$$X(n) = \mathcal{U}(n) + j\mathcal{V}(n) \quad (2.28)$$

thus DFT in Eq.(2.19) becomes

$$\mathcal{U}(n) + j\mathcal{V}(n) = \sum_{k=0}^{N-1} [u(k) + jv(k)] \times \left[\cos\left(\frac{2\pi nk}{N}\right) - j \sin\left(\frac{2\pi nk}{N}\right) \right]$$

hence

$$\mathcal{U}(n) = \sum_{k=0}^{N-1} \left[v(k) \sin\left(\frac{2\pi nk}{N}\right) + u(k) \cos\left(\frac{2\pi nk}{N}\right) \right] \quad (2.29a)$$

$$\mathcal{V}(n) = \sum_{k=0}^{N-1} \left[v(k) \cos\left(\frac{2\pi nk}{N}\right) - u(k) \sin\left(\frac{2\pi nk}{N}\right) \right] \quad (2.29b)$$

Remark 4: Direct DFT Computation

There are $2N^2$ additions and $4N^2$ multiplications in the direct DFT computation above.

Code for Direct DFT Computation

```
w1 = 6.2832/N;  
for k=1:N  
    w2 = w1*(k-1);  
    Ut = u(1);  
    Vt = v(1);  
    for j=2:N  
        w3 = w2*(j-1);  
        C = cos(w3);  
        S = -sin(w3);  
        Ut = Ut + C*u(j) - S*v(j);  
        Vt = Vt + C*v(j) + S*u(j);  
    end  
    U(k) = Ut;  
    V(k) = Vt;  
end
```

2.5. Fast Fourier Transform (FFT)

There are many algorithms to compute DFT faster than the basic computation in Eqs.(2.29). Only 2 of them will be considered in this work, they are Goertzel algorithm and Cooley-Turkey algorithm also known as FFT.

2.5.1. Derivation of DFT Goertzel Algorithm

By Eq.(2.23), Eq.(2.19) can be written as

$$X(n) = \sum_{k=0}^{N-1} x(k)W^{-n(N-k)} \quad (2.30)$$

Let

$$y_n(m) = \sum_{k=0}^{N-1} x(k)W^{-n(m-k)} \quad (2.31)$$

then

$$X(n) = y_n(N) \quad (2.32)$$

thus $y_n(m)$ is the linear convolution of the finite-duration input sequence $x(k)$ of length N with a filter that has an impulse response

$$h_n(k) = W^{-nk}u(k) \quad (2.33)$$

or

$$H_n(z) = \frac{1}{1 - W^{-n}z^{-1}} \quad (2.34)$$

since

$$\mathcal{Z}\{a^k u(k)\} = \sum_{k=-\infty}^{\infty} a^k u(k)z^{-k} = \sum_{k=0}^{\infty} (az^{-1})^k = \frac{1}{1 - az^{-1}} \quad (2.35)$$

where $|az^{-1}| < 1 \Rightarrow |z| > |a|$.

Eq.(2.34) gives

$$y_n(k) = W^{-n}y_n(k-1) + x(k) \quad (2.36)$$

this equation is known as the first-order Goertzel algorithm and involves complex multiplication. To avoid it, Eq.(2.36) can be rewritten in the second-order form as

$$H_n(z) = \frac{1 - W^n z^{-1}}{1 - W^n z^{-1}} \frac{1}{1 - W^{-n} z^{-1}} = \frac{1 - W^n z^{-1}}{1 - 2 \cos(2\pi n/N)z^{-1} + z^{-2}} \quad (2.37)$$

thus

$$Y_n(z) = \frac{N(z)}{D(z)} X(z) = N(z).Q(z) \quad (2.38)$$

where

$$Q(z) = \frac{X(z)}{D(z)} \quad (2.39)$$

Eqs.(2.37), (2.38) and (2.39) give

$$q_n(k) = 2 \cos(2\pi n/N) \cdot q_n(k-1) - q_n(k-2) + x(k) \quad (2.40a)$$

and

$$y_n(k) = q_n(k) - W^n q_n(k-1) = q_n(k) - \cos\left(\frac{2\pi n}{N}\right) q_n(k-1) - j \sin\left(\frac{2\pi n}{N}\right) q_n(k-1) \quad (2.40b)$$

where $q_n(-1) = q_n(-2) = 0$. The complex multiplication has been moved to the output Eq.(2.41), but it is evaluated only once after k reaches N .

Eqs.(2.32) & (2.40b) give

$$|X(n)|^2 = |y_n(N)|^2 = q_n^2(N) + q_n^2(N-1) - 2 \cos\left(\frac{2\pi n}{N}\right) \cdot q_n(N) \cdot q_n(N-1) \quad (2.41)$$

Code for 2nd-Order DFT Goertzel Algorithm

```
w = 2*pi/N;
for k=0:N-1
    S = sin(w*k);
    C = cos(w*k);
    CC = 2*C;
    U2 = 0;
    U1 = u(1);
    V2 = 0;
    V1 = v(1);
    for j=1:N-1
        T = U1;
        U1 = CC*U1 - U2 + u(j+1);
        U2 = T;
        T = V1;
        V1 = CC*V1 - V2 + v(j+1);
        V2 = T;
    end
    U(k+1) = C*U1 - U2 - S*V1;
    V(k+1) = S*U1 + C*V1 - V2;
end
```

2.5.2. Derivation of FFT (Cooley-Turkey Algorithm)

FFT is an algorithm efficiently to compute DFT with $N = 2^v$. For $N = 2^v$ in Eq.(2.19), n and k will be written in the binary form as

$$n = 2^{v-1} n_{v-1} + 2^{v-2} n_{v-2} + \dots + n_0$$

and

$$k = 2^{v-1} k_{v-1} + 2^{v-2} k_{v-2} + \dots + k_0$$

where n_i, k_j take value of 0 or 1, then

$$W^{np} = W^{(2^{v-1} n_{v-1} + 2^{v-2} n_{v-2} + \dots + n_0) \times (2^{v-1} k_{v-1} + 2^{v-2} k_{v-2} + \dots + k_0)}$$

If n 's are broken down then we have the *decimation-in-frequency* FFT. Otherwise, if k 's are broken down, then we have the *decimation-in-time* FFT as below

$$W^{np} = W^{(2^{v-1} n_{v-1} + 2^{v-2} n_{v-2} + \dots + n_0) 2^{v-1} k_{v-1}} \times W^{(2^{v-1} n_{v-1} + 2^{v-2} n_{v-2} + \dots + n_0) 2^{v-2} k_{v-2}} \times \dots \times W^{(2^{v-1} n_{v-1} + 2^{v-2} n_{v-2} + \dots + n_0) k_0} \quad (2.42)$$

By Eq.(2.23), Eq.(2.42) can be written as

$$W^{np} = W^{n_0 2^{v-1} k_{v-1}} \times W^{(2^{v-1} n_1 + 2^{v-2} n_0) k_{v-2}} \times \dots \times W^{(2^{v-1} n_{v-1} + 2^{v-2} n_{v-2} + \dots + n_0) k_0} \quad (2.43)$$

then Eq.(2.19) can be read as

$$X(n_{v-1}, n_{v-2}, \dots, n_0) = \sum_{k_0=0}^1 \sum_{k_1=0}^1 \dots \sum_{k_{v-1}=0}^1 x(k_{v-1}, k_{v-2}, \dots, k_0) W^{2^{v-1} n_0 k_{v-1}} \times W^{(2^{v-1} n_1 + 2^{v-2} n_0) k_{v-2}} \times \dots \times W^{(2^{v-1} n_{v-1} + 2^{v-2} n_{v-2} + \dots + n_0) k_0}$$

Performing each of the summations separately and labelling the intermediate results, we obtain

$$\begin{aligned}
 x_1(n_0, k_{v-2}, \dots, k_0) &= \sum_{k_{v-1}=0}^1 x(k_{v-1}, k_{v-2}, \dots, k_0) W^{2^{v-1} n_0 k_{v-1}} = x(0, k_{v-2}, \dots, k_0) + x(1, k_{v-2}, \dots, k_0) W^{2^{v-1} n_0} \\
 x_2(n_0, n_1, k_{v-2}, \dots, k_0) &= \sum_{k_{v-2}=0}^1 x_1(n_0, k_{v-2}, \dots, k_0) W^{(2^{v-1} n_1 + 2^{v-2} n_0) k_{v-2}} = x_1(n_0, 0, \dots, k_0) + x_1(n_0, 1, \dots, k_0) W^{(2^{v-1} n_1 + 2^{v-2} n_0)} \\
 &\dots \\
 x_v(n_0, n_1, \dots, n_{v-1}) &= \sum_{k_0=0}^1 x_{v-1}(n_0, n_1, \dots, k_0) W^{(2^{v-1} n_{v-1} + 2^{v-2} n_{v-2} + \dots + n_0) k_0} = x_{v-1}(n_0, n_1, \dots, 0) + x_{v-1}(n_0, n_1, \dots, 1) W^{(2^{v-1} n_{v-1} + 2^{v-2} n_{v-2} + \dots + n_0)}
 \end{aligned}$$

then

$$X(n_{v-1}, n_{v-2}, \dots, n_0) = x_v(n_0, n_1, \dots, n_{v-1}) \quad (2.44)$$

where

$$W^{2^{v-1} n_0} = e^{-j2\pi 2^{v-1} n_0 / N} = \cos\left(\frac{2\pi 2^{v-1} n_0}{N}\right) - j \sin\left(\frac{2\pi 2^{v-1} n_0}{N}\right) \quad (2.45)$$

Remark 5: FFT Computation

- Note the bit reversal in Eq.(2.44)
- There are $v = \log_2(N)$ computation stages. Eq.(2.45) reveals that there 2 additions in each stage. The first stage has $2 \times 2^{v-1}$ multiplications since Eq.(2.45) produces the term 2, the (k_{v-2}, \dots, k_0) produces the term 2^{v-1} . The second stage has $2 \times 2^{v-2}$ multiplications, ..., The last stage has only 2 multiplications. There are thus $2v = 2 \log_2(N)$ additions and $2v(1 + 2 + 2^2 + \dots + 2^{v-1}) = 2v \frac{2^v - 1}{2 - 1} = 2 \log_2(N) \times (N - 1)$ multiplications. Note that there are $2N^2$ additions and $4N^2$ multiplications in the direct DFT computation above.
- For FFT computation, any sequence can be rounded up to 2^v by *zero padding* since it is unchanged by Eq.(2.19).

Code for FFT

```

M = round(log(N)/log(2));

% In-place Bit Reversal
j = 0;
N2 = N/2;
N1 = N - 1;
for i = 0:N1-1
    if i < j
        tmp = u(j+1);
        u(j+1) = u(i+1);
        u(i+1) = tmp;
    end
    k = N / 2;
    while(k <= j)
        j = j - k;
        k = k / 2;
    end
    j = j + k;
end

% Perform the DIT FFT computation for each stage
ln = 1;
for m = 1:M
    ln = 2*ln;
    wr = 1;
    cs = cos(wt);
    for j = 0:ln1-1
        for i = j:ln:N-1
            ip = i + ln1;
            tr = u(ip+1)*wr -v(ip+1)*wi;
            u(ip+1) = u(i+1) -tr;
            u(i+1) = u(i+1) +tr;
        end
        tmp = wr*cs -wi*sn;
    end
    ln1 = ln / 2;
    wi = 0;
    sn = sin(wt);
    wt = -pi/ln1;
end

```

```

    wi = wi*cs +wr*sn;
    wr = tmp;
end
end

```

2.5.3. Revisit of Derivation of FFT (Cooley-Turkey Algorithm)

We have N -DFT

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi n}{N}k}, \quad k = 0, 1, \dots, N-1 \quad (2.19)$$

The key idea of C-T FFT is to calculate recursive in pair of half of odd and even, so N must be of power of 2, we will see that it could do so with zero-padding. For an even number, $N = 2N_0$, we have

$$\begin{cases} n = 2m \\ n = 2m + 1 \end{cases}, \quad m = 0, 1, \dots, N_0 - 1 \quad (2.46)$$

so, Eq(2.19) becomes

$$X_k = \underbrace{\sum_{m=0}^{N_0-1} x_{2m} e^{-j\frac{2\pi m}{N_0}k}}_{\text{Even}} + e^{-j\frac{2\pi}{N}k} \underbrace{\sum_{m=0}^{N_0-1} x_{2m+1} e^{-j\frac{2\pi m}{N_0}k}}_{\text{Odd}} = E_k + e^{-j\frac{2\pi}{N}k} O_k$$

Both E_k and O_k are N_0 -DFT, and we have $N_0 = 2N_1$ and keep doing the same to split in half, eventually we have

$$\begin{cases} X_0 = x_0 + x_1 \\ X_k = x_0 - x_1 \end{cases}$$

We have to split $\log_2(N)$ times, this is number of recursion times

2.6. Spectrum Analyzer

```

% Spectrum Analyzer using FFT

clear

J = sqrt(-1);

Fs = 8e3;
Ts = 1/Fs;

N = 64;
N2 = N/2;

tt = Ts*[0:N-1];
ff = (Fs/N)*[0:N2-1];

Fx1 = 500;
Fx2 = 1e3;

xx = sin(2*pi*Fx1*tt) + sin(2*pi*Fx2*tt);

xr = xx;
xi = zeros(1,N);

num_mul = 0;
num_add = 0;

N1 = N;

M = round(log(N)/log(2));

```

```

% In-place Bit Reversal
j = 0;
N2 = N/2;
N1 = N - 1;
for i=0:N1-1
    if i<j
        tmp = xr(j+1);
        xr(j+1) = xr(i+1);
        xr(i+1) = tmp;
    end
    k = N/2;
    num_mul = num_mul + 1;
    while(k<=j)
        j = j - k;
        k = k / 2;
        num_mul = num_mul + 1;
        num_add = num_add + 1;
    end
    j = j + k;
    num_add = num_add + 1;
end

xb = xr;

% Perform the DIT FFT computation for each stage
ln = 1;
for m=1:M
    ln = 2*ln;
    ln1 = ln/2;
    wr = 1;
    wi = 0;
    wt = -pi/ln1;
    cs = cos(wt);
    sn = sin(wt);
    num_mul = num_mul + 3;
    for j=0:ln1-1
        for i=j:ln:N-1
            ip = i + ln1;
            tr = xr(ip+1)*wr -xi(ip+1)*wi;    ti = xi(ip+1)*wr +xr(ip+1)*wi;
            xr(ip+1) = xr(i+1) -tr;          xi(ip+1) = xi(i+1) -ti;
            xr(i+1) = xr(i+1) +tr;          xi(i+1) = xi(i+1) +ti;
            num_mul = num_mul + 4;
            num_add = num_add + 7;
        end
        tmp = wr*cs -wi*sn;
        wi = wi*cs +wr*sn;
        wr = tmp;
        num_mul = num_mul + 4;
        num_add = num_add + 2;
    end
end

fprintf('Number of Multiplications = %d, Number of Additions = %d\n', num_mul, num_add);

ri = xr +J*xi;
Mag = (1/N)*abs(ri(1:N/2));

i_pk1 = find(Mag==max(Mag));
f_pk1 = (Fs/N)*(i_pk1 - 1);

Mag2 = Mag;
Mag2(i_pk1) = [];
i_pk2 = find(Mag2==max(Mag2));
f_pk2 = (Fs/N)*(i_pk2 - 1);

fprintf('Peak1 = %.2f Hz, Peak2 = %.2f Hz\n', f_pk1, f_pk2);

tic
y1 = (1/N)*fft(xx,N);

```

```

toc
Mag1 = abs(y1(1:N/2));
dm = Mag1 - Mag;
max(abs(dm))

figure(gcf)
subplot(211), plot(tt,xx),xlabel('Time [s]'),ylabel('500Hz & 1000 Hz'),grid
subplot(212), plot(ff,Mag),xlabel('Freq [Hz]'),ylabel('FFT'),grid

```

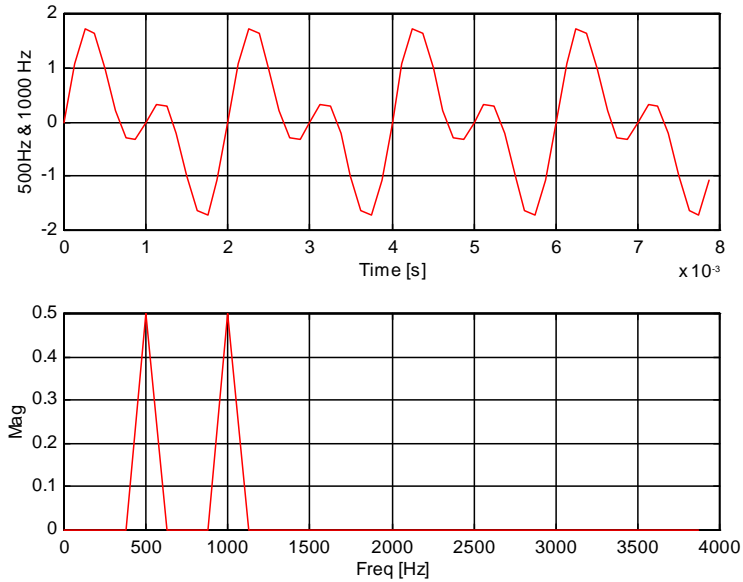
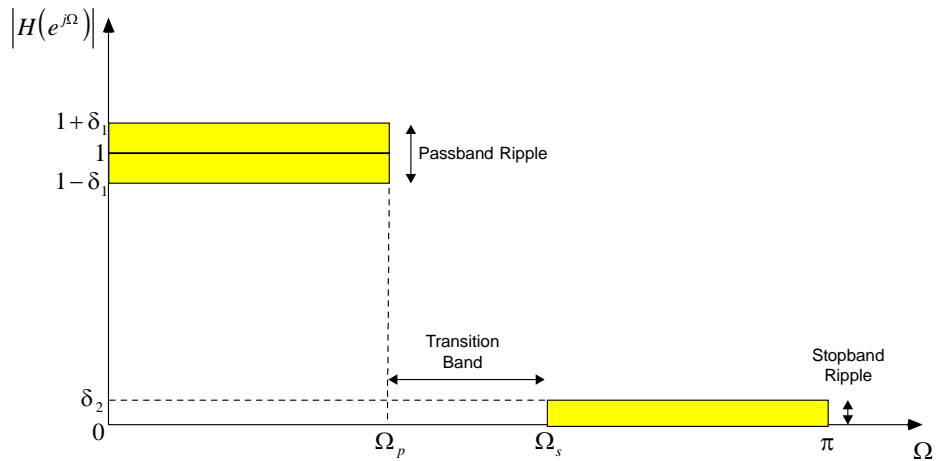


Fig. 8: Spectrum Analyzer

3. Digital Filter Design

Both FIR (Finite-Duration Impulse Response) and IIR (Infinite-Duration Impulse Response) filters will be considered in this section.



3.1. FIR Filter Design

There are 3 approaches will be considered: window design technique, frequency sampling design technique and optimal equiripple design technique.

3.1.1. Window Design Technique

An ideal LPF of bandwidth $\Omega_c < \pi$ is given by

$$H_d(e^{j\Omega}) = \begin{cases} 1, & |\Omega| \leq \Omega_c \\ 0, & \Omega_c < |\Omega| \leq \pi \end{cases} \quad (3.1)$$

where Ω_c is *cutoff* frequency. Eq.(2.14) gives the inverse DTFT as

$$h_d(k) = F^{-1}\{H_d(e^{j\Omega})\} = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(e^{j\Omega}) e^{jk\Omega} d\Omega = \frac{1}{2\pi} \int_{-\Omega_c}^{\Omega_c} e^{jk\Omega} d\Omega = \frac{\sin(k\Omega_c)}{k\pi} \quad (3.2)$$

This infinite sequence must be truncated to obtain a N -length FIR as

$$h(k) = \begin{cases} h_d(k), & 0 \leq k \leq N-1 \\ 0, & \text{otherwise} \end{cases} \quad (3.3)$$

This operation is called "windowing". In general, $h(k)$ can be thought of as being formed by the product of $h_d(k)$ and a window function $w(k)$ as

$$h(k) = h_d(k) \cdot w(k) \quad (3.4)$$

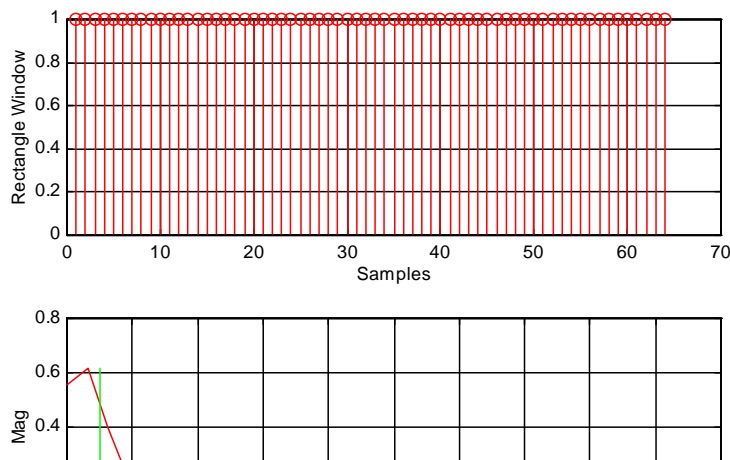
where

$$w(k) = \begin{cases} \text{symmetric function wrt } 0, & 0 \leq k \leq N-1 \\ 0, & \text{otherwise} \end{cases}$$

3.1.1.1. Rectangle Window

This is the simplest window function but provides the worst performance from the viewpoint of stop band attenuation

$$w(k) = \begin{cases} 1, & 0 \leq k \leq N-1 \\ 0, & \text{otherwise} \end{cases} \quad (3.5)$$



Remark 6: Gibb phenomenon

The large stopband ripple due to the sudden transition from 0 to 1 (or 1 to 0) in the rectangle window, this is known as *Gibbs phenomenon*. The following windows solves this problem.

```

clear

J = sqrt(-1);

Ts = 0.005;
Fs = 1/Ts;

N = 64;
nn = [0:N-1] + eps;
nn = nn';

N2 = N/2;

ff = (Fs/N)*[0:N2-1];

Fp = 5;
Wp = 2*pi/Fs*Fp;

Hd = sin(Wp*nn)./(pi*nn);

Win = boxcar(N);

H = Hd .* Win;

yy = fft(H,N);

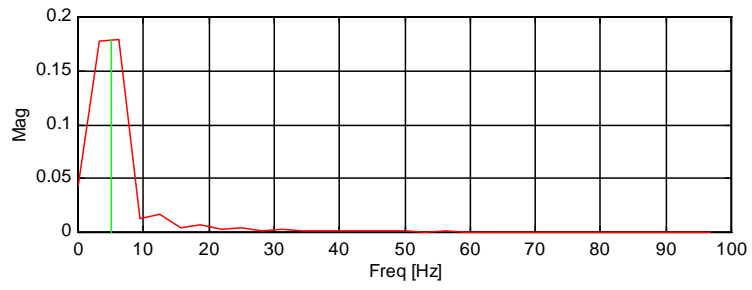
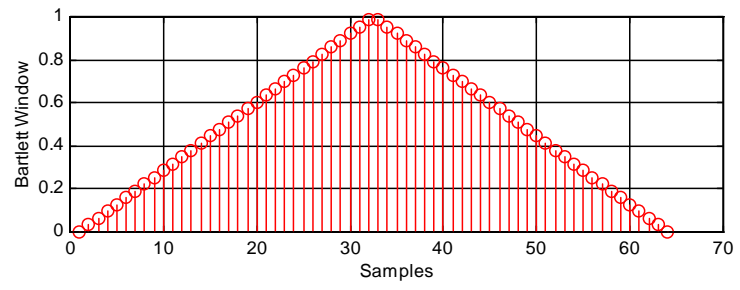
Mag = abs(yy(1:N/2));

figure(gcf)
subplot(211), stem(Win),xlabel('Samples'),ylabel('Rectangle Window'),grid
subplot(212), plot(ff,Mag,'-',[Fp,Fp],[0,max(Mag)],'-'),xlabel('Freq [Hz]'),ylabel('Mag'),grid

```

3.1.1.2. Barlett Window (Triangle Window)

$$w(k) = \begin{cases} \frac{2k}{N-1}, & 0 \leq k \leq \frac{N-1}{2} \\ 2 - \frac{2k}{N-1}, & \frac{N-1}{2} \leq k \leq N-1 \\ 0, & \text{otherwise} \end{cases} \quad (3.6)$$



```
Win = bartlett(N);
```

3.1.1.3. Hanning Window (Cosine Window)

$$w(k) = \begin{cases} 0.5 \left[1 - \cos\left(\frac{2\pi k}{N-1}\right) \right], & 0 \leq k \leq N-1 \\ 0, & \text{otherwise} \end{cases} \quad (3.7)$$

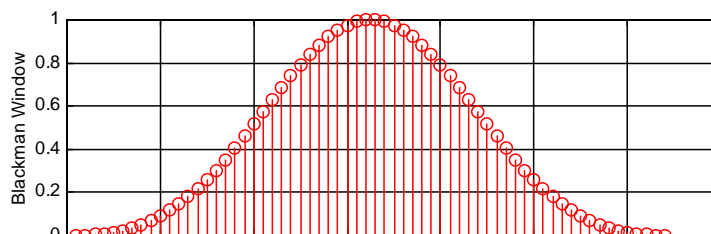
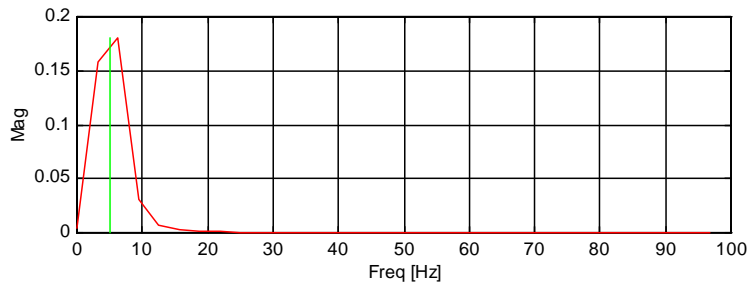
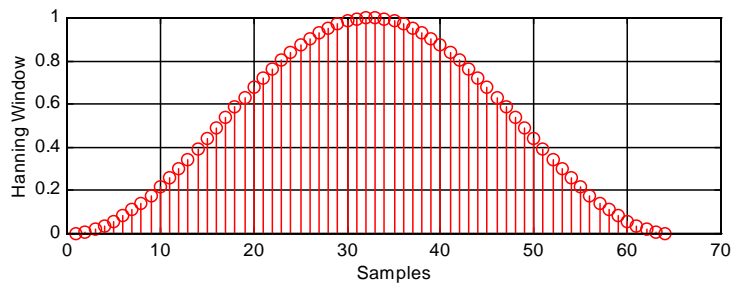
```
Win = hanning(N);
```

3.1.1.4. Blackman Window (Modified Cosine Window)

The second harmonic term is included to give

$$w(k) = \begin{cases} 0.42 - 0.5 \cos\left(\frac{2\pi k}{N-1}\right) + 0.08 \cos\left(\frac{4\pi k}{N-1}\right), & 0 \leq k \leq N-1 \\ 0, & \text{otherwise} \end{cases} \quad (3.8)$$

```
Win = blackman(N);
```



3.1.1.5. Kaiser Window

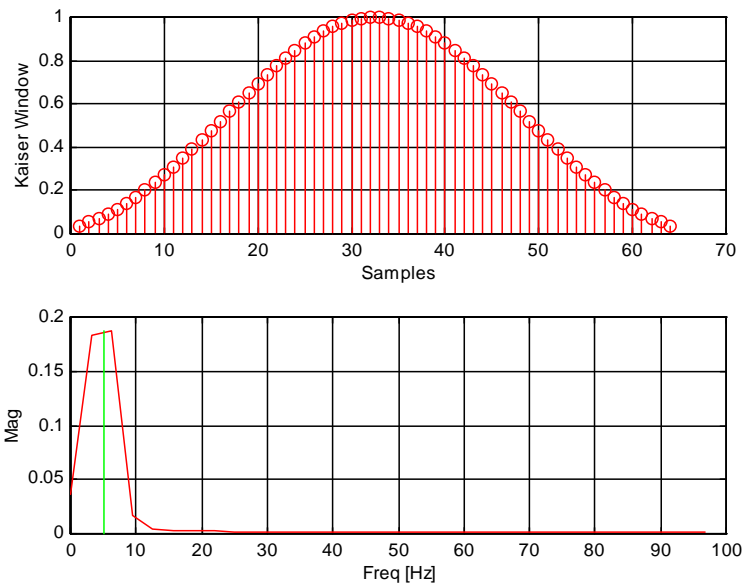
This is one of the most useful and optimum windows. It is optimum in the sense of providing a large main lobe width for a given stopband attenuation, which implies the sharpest transition width.

$$w(k) = \begin{cases} \frac{I_0 \left[\beta \sqrt{1 - \left(1 - \frac{2k}{N-1}\right)^2} \right]}{I_0[\beta]}, & 0 \leq k \leq N-1 \\ 0, & \text{otherwise} \end{cases} \quad (3.9)$$

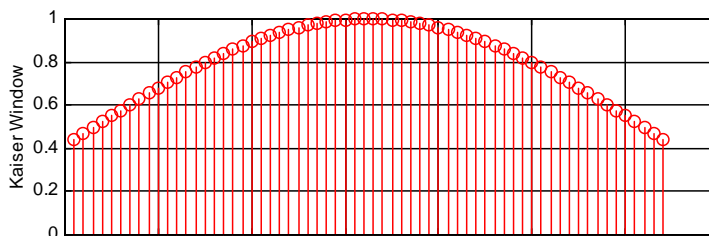
where $I_0(*)$ is the *modified zero-order Bessel function*, and β is a parameter that depends on N and can be chosen to yield various transition widths and near-optimum stopband attenuation.

```
bta = inp('Beta = <5> = ', 5);  
Win = kaiser(N, bta);
```

Choose $\beta = 5$, we have



Choose $\beta = 2$, we have



3.1.1.5. Gaussian Window (NDK)

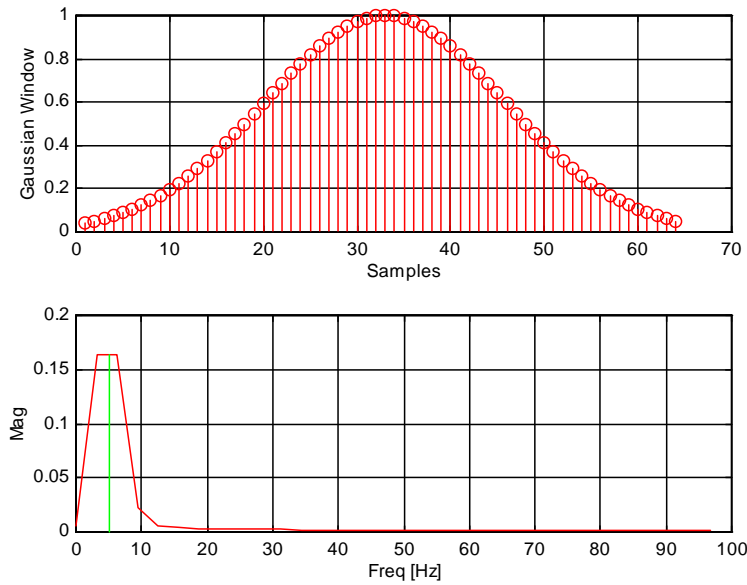
A Gaussian window is proposed to replace Kaiser which uses complicated Bessel function

$$w(k) = \begin{cases} \exp\left[-\left(\frac{k-N/2}{\gamma}\right)^2\right], & 0 \leq k \leq N \\ 0, & \text{otherwise} \end{cases} \quad (3.10)$$

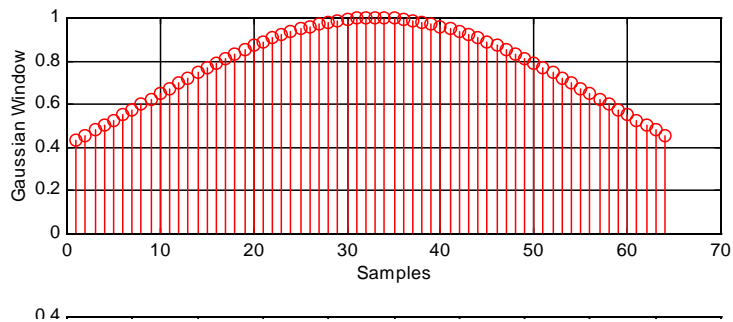
where γ is chosen equivalent to β in Kaiser window

```
gama = inp('Gamma = <5> = ', 5);  
Win = gauss(N, gama);
```

Choose $\gamma = 5$, we have



Choose $\gamma = 2$, we have



3.1.2. Frequency Sampling Design Technique

This design technique is based on a desired frequency response which has 2 features: amplitude and phase. It is normally to have a desired amplitude response, but a desired phase response must be specified to combine with the desired amplitude response to design a filter. One of the most interested feature of a FIR filter is its linear phase property which is considered in the sequence.

3.1.2.1. Linear Phase Property of a FIR Filter

A FIR filter can be characterized by

$$H(z) = \sum_{k=0}^{N-1} h(k)z^{-k} \quad (3.11)$$

Its frequency response is given by

$$H(e^{j\omega T}) = H(z) \Big|_{z=e^{j\omega T}} = \sum_{k=0}^{N-1} h(k)e^{-j\omega kT} \quad (3.12)$$

thus

$$\theta(\omega) = \angle H(e^{j\omega T}) = \tan^{-1} \left(\frac{-\sum_{k=0}^{N-1} h(k) \sin(\omega kT)}{\sum_{k=0}^{N-1} h(k) \cos(\omega kT)} \right) \quad (3.13)$$

For a linear phase, we have

$$\theta(\omega) = \theta_0 - \tau\omega \quad (3.14)$$

then Eq.(3.13) becomes

$$\frac{-\sum_{k=0}^{N-1} h(k) \sin(\omega kT)}{\sum_{k=0}^{N-1} h(k) \cos(\omega kT)} = \tan(\theta_0 - \tau\omega) = \frac{\sin(\theta_0 - \tau\omega)}{\cos(\theta_0 - \tau\omega)}$$

or

$$\sum_{k=0}^{N-1} h(k) \sin(\theta_0 - \tau\omega + \omega kT) = 0 \quad (3.15)$$

- If $\theta_0 = m\pi$, $m = 0, \pm 1$, then Eq.(3.15) becomes

$$h(0)\sin(m\pi - \tau\omega) + h(1)\sin(m\pi - \tau\omega + \omega T) + \dots + h(N-2)\sin[m\pi - \tau\omega + (N-2)\omega T] + h(N-1)\sin[m\pi - \tau\omega + (N-1)\omega T] = 0$$

thus

$$\tau = \frac{(N-1)T}{2} \quad (3.16)$$

and

$$h(k) = h(N-1-k) \quad (3.17)$$

Let

$$M = \frac{N-1}{2}, \quad \bar{M} = \text{round}(M) \quad (3.18)$$

then Eqs.(3.14) and (3.16) give

$$\theta(\omega) = m\pi - M\omega T \quad (3.19)$$

thus Eq.(3.12) becomes

$$H(e^{j\omega T}) = e^{j\theta(\omega)} \sum_{k=0}^{N-1} h(k) e^{-j[\theta(\omega) + \omega kT]} = e^{-jM\omega T} \sum_{k=0}^{N-1} h(k) e^{j(M-k)\omega T}$$

or

$$H(e^{j\omega T}) = e^{-jM\omega T} \left[h(0)e^{jM\omega T} + h(1)e^{j(M-1)\omega T} + \dots + h(N-2)e^{-j(M-1)\omega T} + h(N-1)e^{-jM\omega T} \right]$$

Eq.(3.17) gives

- **Type 1: N odd:** $N = 2P + 1 \Rightarrow M = \frac{N-1}{2} = P$

$$H(e^{j\omega T}) = e^{-jM\omega T} \left[h(M) + 2 \sum_{k=0}^{M-1} h(k) \cos(M-k)\omega T \right] \Rightarrow \left| H(e^{j\omega T}) \Big|_{\omega T=0} \right| > \left| H(e^{j\omega T}) \Big|_{\omega T=\pi} \right| > 0 \quad (3.20)$$

or

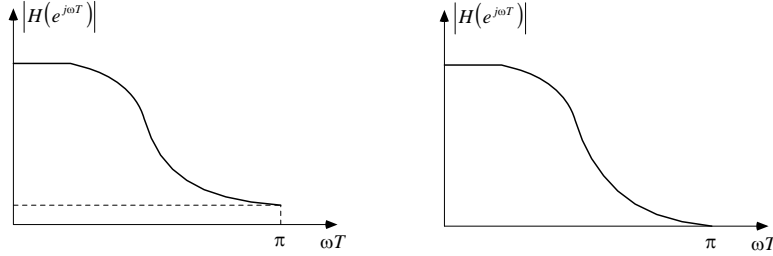
$$H(n) = e^{-j2\pi n M/N} \left\{ h(M) + 2 \sum_{k=0}^{M-1} h(k) \cos \left[(M-k) \frac{2\pi n}{N} \right] \right\}, \quad \omega T = \frac{2\pi f}{f_s} = \frac{2\pi n}{N} \quad (3.21)$$

- **Type 2: N even:** $N = 2P \Rightarrow M = \frac{N-1}{2} = P - \frac{1}{2}$

$$H(e^{j\omega T}) = e^{-jM\omega T} \left[2 \sum_{k=0}^{N/2-1} h(k) \cos(M-k)\omega T \right] \Rightarrow \begin{cases} |H(e^{j\omega T})|_{\omega T=0} > 0 \\ |H(e^{j\omega T})|_{\omega T=\pi} = 0 \end{cases} \quad (3.22)$$

or

$$H(n) = e^{-j2\pi nM/N} \left\{ 2 \sum_{k=0}^{N/2-1} h(k) \cos \left[(M-k) \frac{2\pi n}{N} \right] \right\} \quad (3.23)$$



- If $\theta_0 = \pm \frac{\pi}{2}$, then Eq.(3.15) becomes

$$h(0)\cos(-\tau\omega) + h(1)\cos(-\tau\omega + \omega T) + \dots + h(N-2)\cos[-\tau\omega + (N-2)\omega T] + h(N-1)\cos[-\tau\omega + (N-1)\omega T] = 0$$

thus

$$\tau = \frac{(N-1)T}{2} \quad (3.24)$$

and

$$h(k) = -h(N-1-k) \quad (3.25)$$

thus

$$\theta(\omega) = \pm \frac{\pi}{2} - M\omega T \quad (3.26)$$

then Eq.(3.12) becomes

$$H(e^{j\omega T}) = e^{j\theta(\omega)} \sum_{k=0}^{N-1} h(k) e^{-j[\theta(\omega) + \omega k T]} = e^{j\theta(\omega)} \sum_{k=0}^{N-1} h(k) e^{j[\pm \frac{\pi}{2} + (M-k)\omega T]} = e^{-jM\omega T} \sum_{k=0}^{N-1} h(k) e^{j(M-k)\omega T}$$

or

$$H(e^{j\omega T}) = e^{-jM\omega T} \left[h(0)e^{jM\omega T} + h(1)e^{j(M-1)\omega T} + \dots + h(N-2)e^{-j(M-1)\omega T} + h(N-1)e^{-jM\omega T} \right]$$

Eq.(3.23) gives

- **Type 3: N odd:** $N = 2P + 1 \Rightarrow M = \frac{N-1}{2} = P$

$$H(e^{j\omega T}) = e^{j(\frac{\pi}{2} - M\omega T)} \left[2 \sum_{k=0}^{M-1} h(k) \sin(M-k)\omega T \right] \Rightarrow |H(e^{j\omega T})|_{\omega T=0} = |H(e^{j\omega T})|_{\omega T=\pi} = 0 \quad (3.27)$$

or

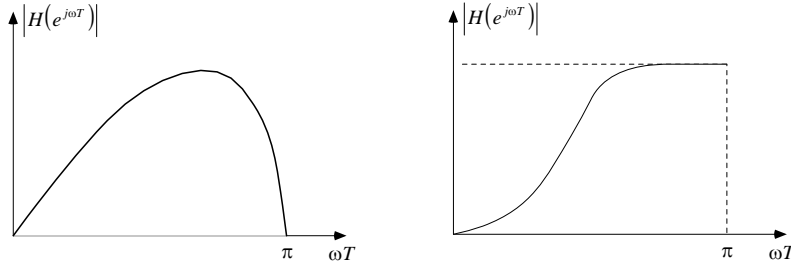
$$H(n) = e^{j(\frac{\pi}{2} - 2\pi nM/N)} \left\{ 2 \sum_{k=0}^{M-1} h(k) \sin \left[(M-k) \frac{2\pi n}{N} \right] \right\} \quad (3.28)$$

- **Type 4: N even:** $N = 2P \Rightarrow M = \frac{N-1}{2} = P - \frac{1}{2}$

$$H(e^{j\omega T}) = e^{j(\frac{\pi}{2} - M\omega T)} \left[2 \sum_{k=0}^{N/2-1} h(k) \sin(M-k)\omega T \right] \Rightarrow \begin{cases} |H(e^{j\omega T})|_{\omega T=0} = 0 \\ |H(e^{j\omega T})|_{\omega T=\pi} > 0 \end{cases} \quad (3.29)$$

or

$$H(n) = e^{j(\frac{\pi}{2} - 2\pi nM/N)} \left\{ 2 \sum_{k=0}^{N/2-1} h(k) \sin \left[(M-k) \frac{2\pi n}{N} \right] \right\} \quad (3.30)$$



Let

$$H(n) = A(n)e^{j\theta(n)} \quad (3.31)$$

then for type 1&2, we have

$$\theta(n) = -\frac{2\pi nM}{N}, \quad 0 \leq n \leq N-1 \quad (3.32)$$

so

$$A(N-n) = A(n), \quad 0 \leq n \leq \bar{M}-1 \quad (3.33)$$

as

$$\cos\left[(M-k)\frac{2\pi(N-n)}{N}\right] = \cos\left[(M-k)2\pi - (M-k)\frac{2\pi n}{N}\right] = \cos\left[(M-k)\frac{2\pi n}{N}\right]$$

and for type 3&4, we have

$$\theta(n) = \begin{cases} \frac{\pi}{2} - \frac{2\pi nM}{N}, & 0 \leq n \leq \bar{M}-1 \\ -\frac{\pi}{2} - \frac{2\pi nM}{N}, & \bar{M} \leq n \leq N-1 \end{cases} \quad (3.34)$$

so

$$A(N-n) = A(n), \quad 0 \leq n \leq \bar{M}-1 \quad (3.35)$$

as

$$\sin\left[(M-k)\frac{2\pi(N-n)}{N}\right] = \sin\left[(M-k)2\pi - (M-k)\frac{2\pi n}{N}\right] = -\sin\left[(M-k)\frac{2\pi n}{N}\right] \quad (3.36)$$

where $\omega(n)$ is subtracted by π in the second half length to obtain Eq.(3.35) due to the negative sign in Eq.(3.36).

3.1.2.2. Frequency Sampling Technique

For a desired frequency response $A(n)$, we can design a linear phase FIR filter corresponding to 4 types above.

- **Type 1: N odd:** $N = 2P+1 \Rightarrow M = \frac{N-1}{2} = P$

Eq1.(3.31) & (3.32) gives

$$h(k) = \frac{1}{N} \sum_{n=0}^{N-1} H(n)e^{j\frac{2\pi nk}{N}} = \frac{1}{N} \sum_{n=0}^{N-1} A(n)e^{j\left[\theta(n) + \frac{2\pi nk}{N}\right]} = \frac{1}{N} \left[A(0) + A(1)e^{-j\frac{2\pi(M-k)}{N}} + \dots + A(N-2)e^{j\frac{2\pi(M-k)}{N}} + A(N-1)e^{j\frac{2\pi(M-k)}{N}} \right]$$

Eq.(3.35) gives

$$h(k) = \frac{1}{N} \left\{ A(0) + 2 \sum_{n=1}^{(N-1)/2} A(n) \cos\left[\frac{2\pi n(M-k)}{N}\right] \right\}, \quad 0 \leq k \leq (N-1)/2 \quad (3.37)$$

and

$$h(N-1-k) = h(n)$$

- **Type 2: N even:** $N = 2P \Rightarrow M = \frac{N-1}{2} = P - \frac{1}{2}$

Similarly

$$h(k) = \frac{1}{N} \left\{ A(0) + 2 \sum_{n=1}^{(N/2)-1} A(n) \cos\left[\frac{2\pi n(M-k)}{N}\right] \right\}, \quad A(N/2) = 0, \quad 0 \leq k \leq (N/2) \quad (3.38)$$

and

$$h(N-1-k) = h(n)$$

- **Type 3: N odd:** $N = 2P + 1 \Rightarrow M = \frac{N-1}{2} = P$

Eq1.(3.31) & (3.34) gives

$$h(k) = \frac{1}{N} \sum_{n=0}^{N-1} H(n) e^{j \frac{2\pi nk}{N}} = \frac{1}{N} \sum_{n=0}^{N-1} A(n) e^{j \left[\theta(n) + \frac{2\pi nk}{N} \right]} = j \frac{1}{N} \left[A(0) + A(1) e^{-j \frac{2\pi(M-k)}{N}} + \dots - A(N-2) e^{j \frac{2\pi 2(M-k)}{N}} - A(N-1) e^{j \frac{2\pi(M-k)}{N}} \right]$$

Eq.(3.35) gives

$$h(k) = \frac{1}{N} \left[2 \sum_{n=1}^{(N-1)/2} A(n) \sin\left(\frac{2\pi nk}{N}\right) \right], \quad 0 \leq k \leq (N-1)/2, \quad A(0) = 0 \quad (3.39)$$

and

$$h(N-1-k) = -h(n)$$

- **Type 4: N even:** $N = 2P \Rightarrow M = \frac{N-1}{2} = P - \frac{1}{2}$

$$h(k) = \frac{1}{N} \sum_{n=0}^{N-1} H(n) e^{j \frac{2\pi nk}{N}} = \frac{1}{N} \sum_{n=0}^{N-1} A(n) e^{j \left[\theta(n) + \frac{2\pi nk}{N} \right]} = j \frac{1}{N} \left[A(0) + A(1) e^{-j \frac{2\pi(M-k)}{N}} + \dots - A(N-2) e^{j \frac{2\pi 2(M-k)}{N}} - A(N-1) e^{j \frac{2\pi(M-k)}{N}} \right]$$

Eq.(3.35) gives

$$h(k) = \frac{1}{N} \left[2 \sum_{n=1}^{(N/2)-1} A(n) \sin\left(\frac{2\pi nk}{N}\right) \right], \quad 0 \leq k \leq N/2, \quad A(0) = 0 \quad (3.40)$$

and

$$h(N-1-k) = -h(n)$$

```
% Frequency Sampling FIR Design
% Type 1 & 2: Low-Pass, Type 2 better
% Type 3: Differentiator
% Type 4: High-Pass
```

```
clear
```

```
J = sqrt(-1);
```

```
N = inp('N <64> = ', 64); % Odd: Type 1; Even: Type 2
```

```
Fp = 5;
```

```
Ts = 0.005;
```

```
Fs = 1/Ts;
```

```
Fn = Fs/2;
```

```
FF = Fn*[0:N/2-1]/N;
```

```
%%% Desired Frequency Response
```

```
M = (N-1)/2;
```

```
Mn = round(M + 0.5);
```

```
Np = round(N*Fp/Fn) + 1;
```

```
Ad = zeros(Mn,1);
```

```
Ad(1:Np) = ones(Np,1);
```

```
N2 = floor(N/2);
```

```
%%% Impulse Response
```

```
for k=1:Mn
```

```
    ht = Ad(1);
```

```
    for n=2:Mn
```

```
        ht = ht + 2*Ad(n)*cos(2*pi*(n-1)*(M-k+1)/N);
```

```
    end
```

```
    h(k) = ht/N;
```

```
end
```

```
h = h';
```

```
%%% Frequency Response
```

```
for n=1:Mn
```

```
    At = rem(N,2)*h(Mn);
```

```
    for k=1:N2
```

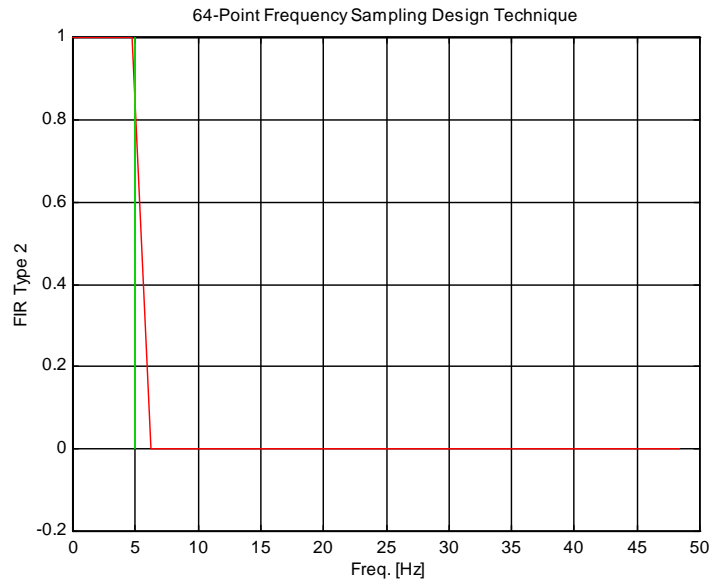
```
        At = At + 2*h(k)*cos(2*pi*(n-1)*(M-k+1)/N);
```

```
    end
```

```
    Ar(n) = At;
```

```
end
```

```
figure(gcf)
subplot(111),plot(FF,Ar),xlabel('Freq. [Hz]'),ylabel('Freq. Smpl Type 2'),grid
```



3.1.3. 3.1.3. Optimal Equiripple Design Technique

```
clear

N = inp('N <64> = ', 64);

Fp = 5;
Fs = 10;
Ts = 0.005;
Fn = 0.5/Ts;
FF = Fn*[0:N/2-1]/N;

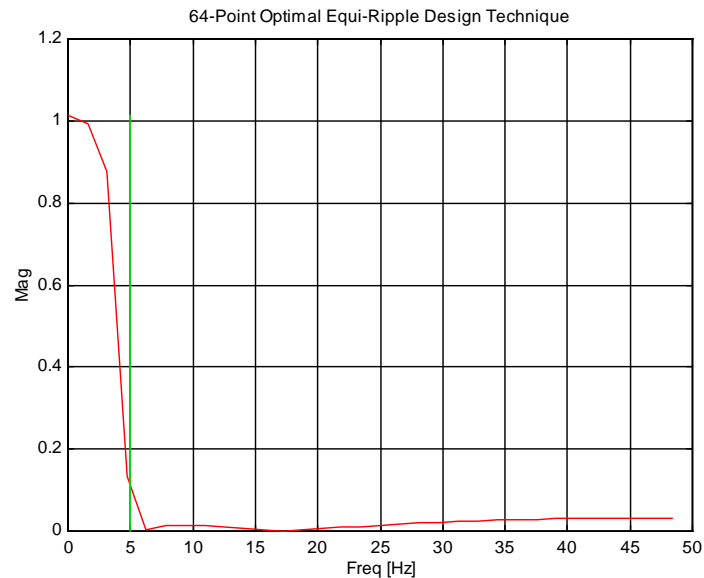
ff = [0, Fp, Fs, Fn]/Fn;
mm = [1, 1, 0, 0];
ww = [5, 1];

hh = remez(N, ff, mm, ww);

HH = fft(hh,N);

Mag = abs(HH(1:N/2));

figure(gcf)
plot(FF,Mag,'-',[Fp,Fp],[0,max(Mag)],'-'),xlabel('Freq [Hz]'), ylabel('Mag'), grid
title('64-Point Optimal Equi-Ripple Design Technique')
```



3.2. IIR Filter Design

This design is based on analog filter prototypes, then they can be converted into digital filters either by the exact or approximation transformations, *ie.* z -transform, backward difference, bilinear transformations, etc.

3.2.1. Butterworth Lowpass Filter

This filter is characterized by the property that its magnitude response is flat in both passband and stopband. A *normalized LP* Butterworth analog filter is determined by its order N and its cutoff frequency ω_c

3.2.2. Chebyshev Lowpass Filter

There are 2 types of Chebyshev filters. The Chebyshev-I filters have *equiripple response in the passband*, while the Chebyshev-II filters have *equiripple response in the stopband*. Butterworth filters have monotonic response in both bands. By choosing a filter that has an equiripple rather than monotonic behavior, we can obtain a lower-order filter. Therefore Chebyshev filters provide lower order than Butterworth filters for the same specifications.

A Chebyshev-II filter is related to the Chebyshev-I filter through a simple transformation. it has a monotone passband and an equiripple stopband. It has the phase response more linear in the passband than the Chebyshev-I filter.

3.2.3. Elliptic Lowpass Filter

These filters exhibit equiripple behavior in the passband as well as in stopband. They are similar in magnitude response characteristics to the FIR equiripple filters. Therefore elliptic filoters are optimum filters in that they achieve the minimum order N for the given specifications (or alternately, achieve the sharpest transition band for the given order N)

```
clear
```

```
N = inp('N <3> = ', 3);
```

```
Fp = 5;
```

```
Ts = 0.005;
```

```
Fs = 1/Ts;
```

```
Fn = Fs/2;
```

```

L = 200;

%%% Butterworth Digital Filter
[bb, ab] = butter(N, Fp/Fn);
[Hb,Fz] = freqz(bb,ab, L, Fs);
Magb = abs(Hb);

%%% Chebyshev-I Digital Filter
Rp = inp('Passband Ripple <0.5> = ', 0.5);
[bc1, ac1] = cheby1(N, Rp, Fp/Fn);
[Hc1,Fz] = freqz(bc1,ac1, L, Fs);
Magc1 = abs(Hc1);

%%% Chebyshev-II Digital Filter
Rs = inp('Stopband Ripple <20> = ', 20);
[bc2, ac2] = cheby2(N, Rs, Fp/Fn);
[Hc2,Fz] = freqz(bc2,ac2, L, Fs);
Magc2 = abs(Hc2);

%%% Elliptic Digital Filter
[be, ae] = ellip(N, Rp, Rs, Fp/Fn);
[He,Fz] = freqz(be,ae, L, Fs);
Mage = abs(He);

close(gcf)
subplot(221), plot(Fz, Magb, '-',[Fp,Fp],[0,max(Magb)], '-'),
ylabel('Magnitude'), grid
title('Butterworth Digital Filter')
axis([0,40, 0,1])

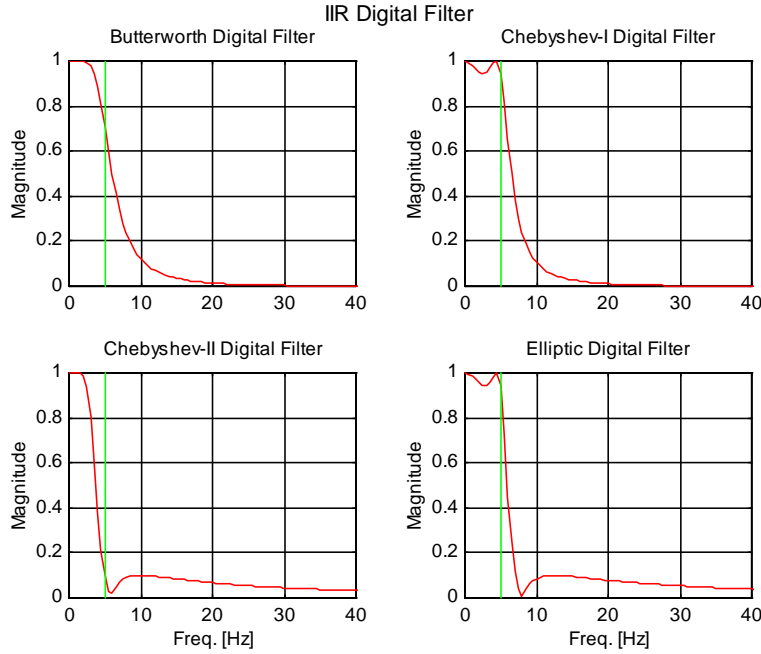
subplot(222), plot(Fz, Magc1, '-',[Fp,Fp],[0,max(Magc1)], '-'),
ylabel('Magnitude'),grid
title('Chebyshev-I Digital Filter')
axis([0,40, 0,1])

subplot(223), plot(Fz, Magc2, '-',[Fp,Fp],[0,max(Magc2)], '-'),
xlabel('Freq. [Hz]'),ylabel('Magnitude'),grid
title('Chebyshev-II Digital Filter')
axis([0,40, 0,1])

subplot(224), plot(Fz, Mage, '-',[Fp,Fp],[0,max(Mage)], '-'),
xlabel('Freq. [Hz]'),ylabel('Magnitude'),grid
title('Elliptic Digital Filter')
axis([0,40, 0,1])

toptitle('IIR Digital Filter')

```



4. Convolution

4.1. Linear and Circular Convolution

In reality, we have a *linear* convolution as in Eq.(1.7) for an infinite sequences. For 2 finite sequences x_1 of length N_1 and x_2 of length N_2 , a linear convolution is defined as

$$y(n) = \sum_k x_1(k)x_2(n-k) \quad (4.1)$$

The length N of y is determined by letting some specific values for N_1 and N_2 , say $N_1 = 3$ and $N_2 = 4$, we have

$$\begin{aligned} y(0) &= x_1(0).x_2(0) \\ y(1) &= x_1(0).x_2(1) + x_1(1).x_2(0) \\ y(2) &= x_1(0).x_2(2) + x_1(1).x_2(1) + x_1(2).x_2(0) \\ y(3) &= x_1(0).x_2(3) + x_1(1).x_2(2) + x_1(2).x_2(1) \\ y(4) &= x_1(1).x_2(3) + x_1(2).x_2(2) \\ y(5) &= x_1(2).x_2(3) \end{aligned}$$

Thus the length of y is

$$N = N_1 + N_2 - 1 \quad (4.2)$$

DFT requires periodic sequences by Eq.(2.24), this leads to a circular convolution defined as

$$\tilde{y}(n) = \sum_{k=0}^{N-1} x_1(k)_N \cdot x_2(n-k)_N, \quad n \in [0, N-1] \quad (4.3)$$

where both x_1 and x_2 are extended to the same length N in Eq.(4.2) by zero padding and both have the periodic property

$$x_i(k+mN) = x_i(k), \quad i = 1, 2 \quad (4.4)$$

We need to verify that $\tilde{y}(n) = y(n)$ in the above illustration with

$$x_1 = \{x_1(0), x_1(1), x_1(2), 0, 0, 0\}$$

and

$$x_2 = \{x_2(0), x_2(1), x_2(2), x_2(3), 0, 0\}$$

then Eqs(4.3) and (4.4) give

$$\tilde{y}(0) = x_1(0)_6 \cdot x_2(0)_6 + x_1(1)_6 \cdot x_2(-1)_6 + x_1(2)_6 \cdot x_2(-2)_6 + x_1(3)_6 \cdot x_2(-3)_6 + x_1(4)_6 \cdot x_2(-4)_6 + x_1(5)_6 \cdot x_2(-5)_6$$

or

$$\tilde{y}(0) = x_1(0)_6 \cdot x_2(0)_6 + x_1(1)_6 \cdot x_2(5)_6 + x_1(2)_6 \cdot x_2(4)_6 + x_1(3)_6 \cdot x_2(3)_6 + x_1(4)_6 \cdot x_2(2)_6 + x_1(5)_6 \cdot x_2(1)_6 = x_1(0)_6 \cdot x_2(0)_6$$

thus

$$\tilde{y}(0) = x_1(0).x_2(0)$$

Similarly, we have

$$\begin{aligned}\tilde{y}(1) &= x_1(0).x_2(1) + x_1(1).x_2(0) \\ \tilde{y}(2) &= x_1(0).x_2(2) + x_1(1).x_2(1) + x_1(2).x_2(0) \\ \tilde{y}(3) &= x_1(0).x_2(3) + x_1(1).x_2(2) + x_1(2).x_2(1) \\ \tilde{y}(4) &= x_1(1).x_2(3) + x_1(2).x_2(2) \\ \tilde{y}(5) &= x_1(2).x_2(3)\end{aligned}$$

Therefore, we can convert a linear convolution for filtering into a circular convolution for computing using DFT which has the fast computation FFT. In doing so, it is required that a zero-padding to extend to the length in Eq.(4.2) and a periodicity in Eq.(4.4).

```
function y = cirshfft(x,m,N)
% Circular shift of m samples wrt size N in sequence x: (time domain)
% -----
% [y] = cirshfft(x,m,N)
% y = output sequence containing the circular shift
% x = input sequence of length <= N
% m = sample shift
% N = size of circular buffer
% Method: y(n) = x((n-m) mod N)

% Check for length of x
if length(x) > N
    error('N must be >= the length of x')
end
x = [x zeros(1,N-length(x))];
n = [0:1:N-1];
n = mod(n-m,N);
y = x(n+1);

function y = circonvt(x1,x2,N)
% N-point circular convolution between x1 and x2: (time-domain)
% -----
% [y] = circonvt(x1,x2,N)
% y = output sequence containing the circular convolution
% x1 = input sequence of length N1 <= N
% x2 = input sequence of length N2 <= N
% N = size of circular buffer
% Method: y(n) = sum (x1(m)*x2((n-m) mod N))

% Check for length of x1
if length(x1) > N
    error('N must be >= the length of x1')
end

% Check for length of x2
if length(x2) > N
    error('N must be >= the length of x2')
end

x1=[x1 zeros(1,N-length(x1))];
x2=[x2 zeros(1,N-length(x2))];

m = [0:1:N-1];
x2 = x2(mod(-m,N)+1);
H = zeros(N,N);

for n = 1:1:N
    H(n,:) = cirshfft(x2,n-1,N);
end

y = x1*H';

clear

x1 = 1:3;
x2 = 4:8;
```

```

N1 = length(x1);
N2 = length(x2);

conv(x1,x2)

for n=max(N1,N2):N1+N2-1
    circonvt(x1,x2,n)
end

ans =     4     13     28     34     40     37     24           % Linear Convolution of Length 7

ans =     41     37     28     34     40                               % Circular Convolution of Length 5

ans =     28     13     28     34     40     37                       % Circular Convolution of Length 6

ans =     4     13     28     34     40     37     24           % Circular Convolution of Length 7

ans =     4     13     28     34     40     37     24           0 % Circular Convolution of Length 8

```

Remark 7: Length of Circular Convolution

For 2 sequences x_1 and x_2 of length N_1 and N_2 , respectively for a circular convolution of length N . Let $N_{req} = N_1 + N_2 - 1$, $N_{min} = \min(N_1, N_2)$, and $N_{max} = \max(N_1, N_2)$, then

- $N_{max} \leq N \leq N_{req}$: the first $(N - N_{max})$ samples of the circular convolution are erroneous
- $N = N_{max}$: the first $N_{min} - 1$ are erroneous
- $N = N_{req}$: the results of the circular and linear convolution are exactly the same
- $N > N_{req}$: the first N_{req} samples of the circular convolution are exactly the same the linear convolution, the rest of circular convolution are zeros

4.2. Fast Convolution

Consider a circular convolution as an implementation of a linear convolution as above

$$y(m) = \sum_{k=0}^{N-1} x_1(k) \cdot x_2(m-k) \quad (4.5)$$

then its DFT is

$$Y(n) = \sum_{m=0}^{N-1} y(m) e^{-j2\pi nm/N} = \sum_{m=0}^{N-1} \left[\sum_{k=0}^{N-1} x_1(k) \cdot x_2(m-k) \right] e^{-j2\pi nm/N} = \sum_{m=0}^{N-1} \left[\sum_{k=0}^{N-1} x_1(k) e^{-j2\pi nk/N} \cdot x_2(m-k) e^{-j2\pi n(m-k)/N} \right]$$

or

$$Y(n) = \sum_{k=0}^{N-1} x_1(k) e^{-j2\pi nk/N} \cdot \sum_{m=0}^{N-1} x_2(m-k) e^{-j2\pi n(m-k)/N} = X_1(n) \cdot X_2(n)$$

thus

$$x_1(k) * x_2(k) = \text{IDFT} \left\{ \text{DFT} [x_1(k)] \times \text{DFT} [x_2(k)] \right\} \quad (4.6)$$

The conv function in MATLAB is implemented using the direct convolution coded in C, fft and ifft functions are also coded in C. We will demonstrate the effectiveness of the fast convolution above. We first compare the results of the direct and fast convolutions.

```

clear

x1 = [1, 2, 3];
x2 = [4, 5, 6, 7];

N1 = length(x1);
N2 = length(x2);
N = N1 + N2 - 1;
nu = ceil(log(N)/log(2));
N = 2^nu;

y1 = conv(x1, x2);

y2 = real(ifft(fft(x1,N).*fft(x2,N)));

```

y1 =	4	13	28	34	32	21		
y2 =	4.00	13.00	28.00	34.00	32.00	21.00	0.00	0.00

```

clear

conv_time = zeros(1,150);
fft_time = zeros(1,150);

for L=1:150
    tc = 0;
    tf = 0;
    N = 2*L - 1;
    nu = ceil(log(N)/log(2));
    N = 2^nu;
    for I=1:100
        h = randn(1,L);
        x = rand(1,L);

        t0 = clock;
        y1 = conv(h,x);
        t1 = etime(clock,t0);
        tc = tc + t1;

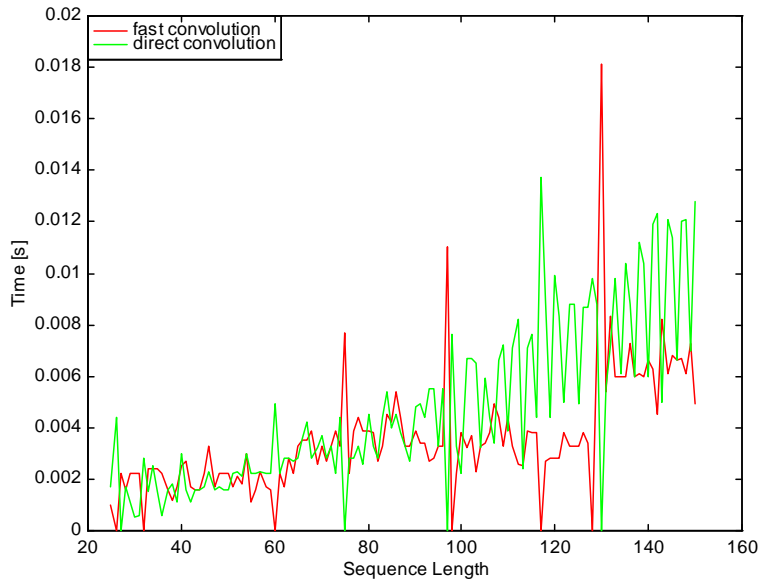
        t0 = clock;
        y2 = ifft(fft(h,N).*fft(x,N));
        t2 = etime(clock,t0);
        tf = tf + t2;
    end

    conv_time(L) = tc/100;
    fft_time(L) = tf/100;
end

figure(gcf)
n = 1:150;
subplot(111),plot(n(25:150), fft_time(25:150),'-',n(25:150), conv_time(25:150),'-')
xlabel('Sequence Length'),ylabel('Time [s]')
legend('fast convolution ', 'direct convolution ')
toptitle('Comparison of Fast and Direct Convolution Time')

```

Comparison of Fast and Direct Convolution Time

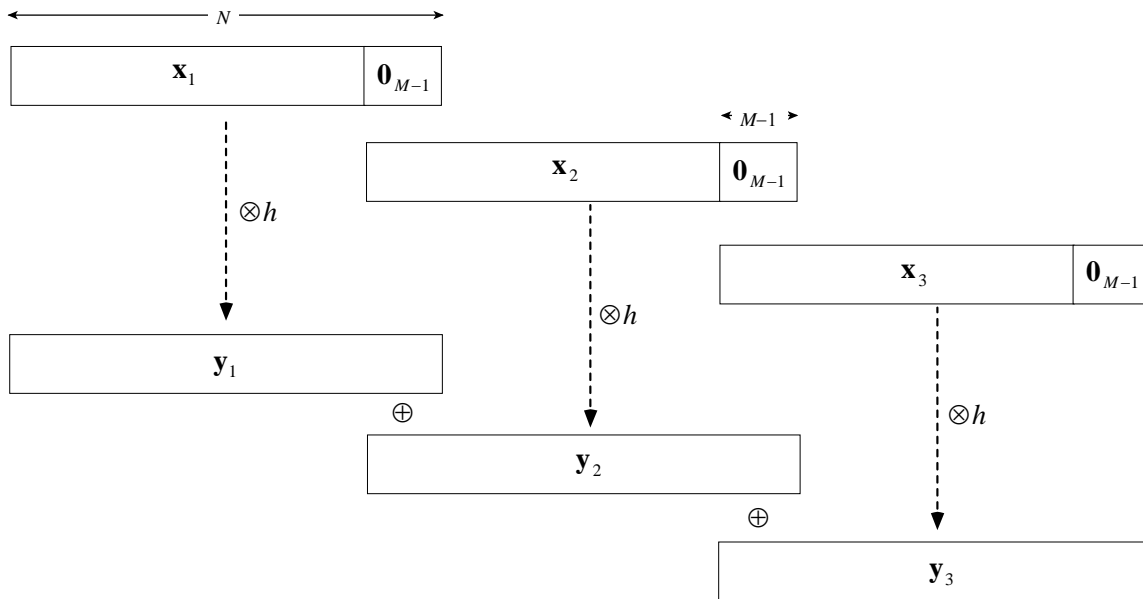


4.3. Block Convolution

If we want to filter (using convolution) an input sequence received continuously as an infinite sequence, then we experience some practical problems. We will have to compute a large DFT (for fast convolution) which is generally impractical. Furthermore, output samples are not available until all input samples are processed. This introduces an unacceptable large amount of delay. Therefore, we have to segment the infinite input sequence into smaller sections (or blocks), process each sections using DFT, and finally assemble the output sequence from outputs of each section. This procedure is called a *block convolution* operation.

Consider an input sequence x of length N_x segmented into N -sequence block, and an impulse sequence h of length M . From Remark 4, we have the first $(M - 1)$ samples in N -circular convolution are erroneous.

4.3.1. Overlap-Add Method



clear

```

x = 1:10;
h = 11:13;

N = 6;
M = length(h);
M1 = M - 1;
x1 = [ 1  2  3  4  0  0];
x2 = [ 5  6  7  8  0  0];
x3 = [ 9 10  0  0  0  0];

yt1 = circonvt(x1,h,6)
yt2 = circonvt(x2,h,6)
yt3 = circonvt(x3,h,6)

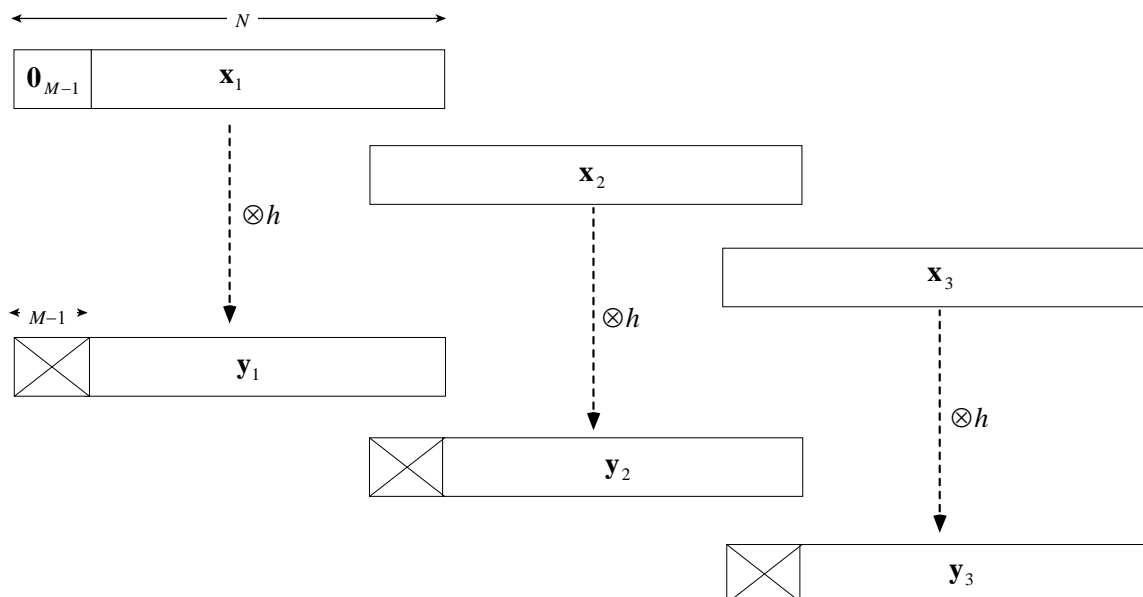
yt2(1:M1) = yt2(1:M1) + yt1(N-M1+1:N);
yt3(1:M1) = yt3(1:M1) + yt2(N-M1+1:N);

yy = [yt1(1:N-M1), yt2(1:N-M1), yt3(1:N-M1)]
y = conv(x,h)

```

yt1 =	11	34	70	106	87	52						
yt2 =	55	126	214	250	187	104						
yt3 =	99	218	237	130	0	0						
yy =	11	34	70	106	142	178	214	250	286	322	237	130
y =	11	34	70	106	142	178	214	250	286	322	237	130

4.3.2. Overlap-Save Method



```
clear
```

```
x = 1:10;  
h = 11:13;
```

```
N = 6;  
M = length(h);  
M1 = M - 1;  
x1 = [ 0 0 1 2 3 4];  
x2 = [ 3 4 5 6 7 8];  
x3 = [ 7 8 9 10 0 0];
```

```
yt1 = circonvt(x1,h,6)  
yt2 = circonvt(x2,h,6)  
yt3 = circonvt(x3,h,6)
```

```
yy = [yt1(M:N), yt2(M:N), yt3(M:N)]  
y = conv(x,h)
```

yt1 =	87	52	11	34	70	106						
yt2 =	220	184	142	178	214	250						
yt3 =	77	172	286	322	237	130						
yy =	11	34	70	106	142	178	214	250	286	322	237	130
y =	11	34	70	106	142	178	214	250	286	322	237	130