A. Enis Cetin

Lecture Notes on Discrete-Time Signal Processing

EE424 Course @ Bilkent University

April 19, 2013

BILKENT

Foreword

This is version 1 of my EE 424 Lecture Notes. I am not a native English speaker. Therefore the language of this set of lecture notes will be Globish. I will later (hopefully) revise this version and make it English with the help of my native English speaker son Sinan.

I have been studying, teaching contributing to the field of Discrete-time Signal Processing for more than 25 years. I tought this course at Bilkent University, University of Toronto and Sabanci University in Istanbul. My treatment of filter design is different from most textbooks and I only include material that can be covered in a single semester course.

The notes are organized according to lectures and I have X lectures.

We assume that the student took a Signals and Systems course and he or she is familier with Continuous Fourier Transform and Discrete-time Fourier Transform.

I would like to thank Kivanc Kose, Berkan Dulek, Onur Yorulmaz, and San Gultekin who helped with the typing of this book in Latex.

There may be typos in the notes. So be careful!

Note: This book will not be published. It will remain as an online resource.

Ankara, October 2011 *A. Enis Cetin*

Contents

Chapter 1
Introduction, Sampling Theorem and Notation

The first topic that we study is multirate signal processing. We need to review Shannon's sampling theorem, Continuous-time Fourier Transform (CTFT) and Discrete-time Fourier Transform (DTFT) before introducing basic principles of multirate signal processing. We use the Shannon sampling theorem to establish the relation between discrete-time signals sampled at different sampling rates.

Shannon's sampling theorem has been studied and proved by Shannon and other researchers including Kolmogorov in 1930's and 40's. Nyquist first noticed that telephone speech with a bandwidth of 4 KHz can be reconstructed from its samples, if it is sampled at 8 KHz at Bell Telephone Laboratories in 1930's.

It should be pointed out that this is not the only sampling theorem. There are many other sampling theorems.

We assume that student is familiar with periodic sampling from his third year Signals and Systems class. Let $x_c(t)$ be a continuous-time signal. The subscript "c" indicates that the signal is a continuous-time function of time. The discrete-time signal: $x[n] = x_c(nT_s), \quad n = 0, \pm1, \pm2, \pm3, \ldots$ where T_s is the sampling period.

1.1 Shannon's Sampling Theorem

Let $x_c(t)$ be a band-limited continuous-time signal with the highest frequency w_b. The sampling frequency w_s should be larger than $w_s > 2w_b$ to construct the original signal $x_c(t)$ from its samples $x[n] = x_c(nT_s), \quad n = 0, \pm1, \pm2, \pm3, \ldots$. The angular sampling frequency $\omega_s = 2\pi/T_s$ is called the Nyquist sampling rate.

Example: Telephone speech has a bandwidth of 4 kHz. Therefore the sampling frequency is 8 kHz, i.e., we get 8000 samples per second from the speech signal.

Example: In CD's and MP3 players, the audio sampling frequency is $f_s = 44.1$ kHz.

If the signal is not band-limited, we apply a low-pass filter first and then sample the signal. A-to-D converters convert audio and speech into digital form in PC's

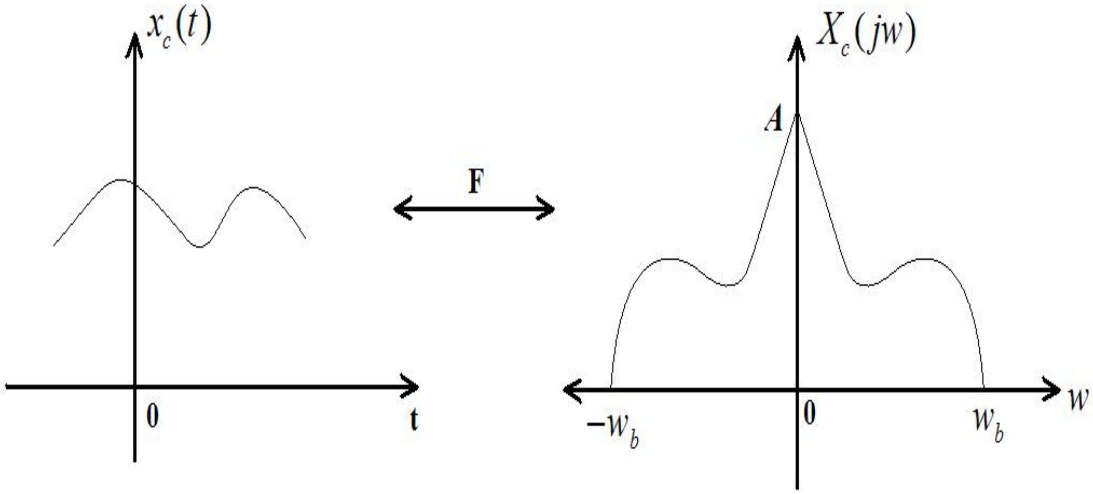


Fig. 1.1 The continuous-time signal $x_c(t)$ and its continuous-time Fourier Transform $X_c(jw)$. In general, Fourier Transform (FT) of a signal is complex but we use a real valued plot to illustrate basic concepts. This is just a graphical representation. It would be clumsy to plot the both the real and imaginary parts of the FT.

and phones etc and they have a built-in low-pass filter whose cut-off frequency is determined according to the sampling frequency.

The discrete-time signal $x[n] = x_c(nT_s), \quad n = 0, \pm 1, \pm 2, \pm 3, \ldots$ with the sampling period $T_s = \frac{1}{f_s} = \frac{2\pi}{w_s}$, $w_s = 2\pi f_s$ is equivalent to the continuous-time signal:

$$x_p(t) = \sum_{n=-\infty}^{\infty} x_c(nT_s)\delta(t - nT_s) \tag{1.1}$$

where $\delta(t - nT_s)$ is a Dirac-delta function occurring at $t = nT_s$. The signal $x_p(t)$ is not a practically realizable signal but we use it to prove the Shannon's sampling theorem. The sampling process is summarized in Figure **??**. The signal $x_p(t)$ and the discrete-time signal $x[n]$ are not equal because one of them is a discrete-time signal the other one is a continuous-time signal but they are equivalent because they contain the same samples of the continuous time signal $x_c(t)$:

$$x_p(t) \equiv x[n], \qquad x_p(t) \neq x[n] \tag{1.2}$$

The continuous-time signal $x_p(t)$ can be expressed as follows:

$$x_p(t) = x_c(t)p(t), \tag{1.3}$$

where

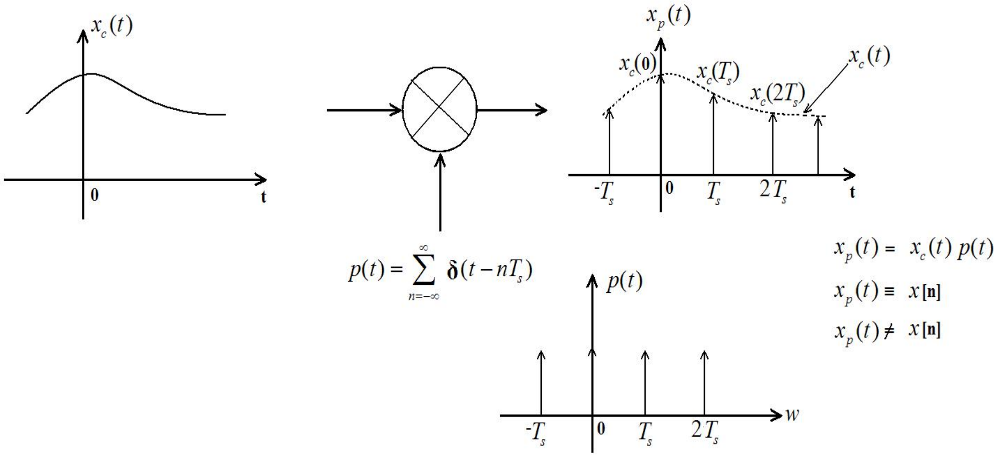


Fig. 1.2 The signal $x_p(t)$ contains the samples of the continuous-time signal $x_c(t)$.

$$p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_s)$$

is a uniform impulse train with impulses occurring at $t = nT_s, \quad n = 0, \pm 1, \pm 2, \pm 3, \ldots$.

The continuous-time Fourier Transform of $x_p(t)$ is given by

$$X_p(jw) = \frac{1}{2\pi} P(jw) * X_c(jw)$$

where $P(jw)$ is the CTFT of the impulse train $p(t)$

$$P(jw) = \frac{2\pi}{T_s} \sum_{k=-\infty}^{\infty} \delta(w - kw_s)$$

$P(jw)$ is also an impulse train in the Fourier domain (see Fig. **??**). Notice that Fourier domain impulses occur at $w = kw_s$ and the strength of impulses are $1/T_s$. Convolution with an impulse only shifts the original function therefore

$$X_c(jw) * \delta(w - w_s) = X_c(j(w - w_s))$$

Similarly,

$$X_c(jw) * \delta(w - kw_s) = X_c(j(w - kw_s))$$

As a result we obtain

$$X_p(jw) = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} X_c(j(w - kw_s))$$

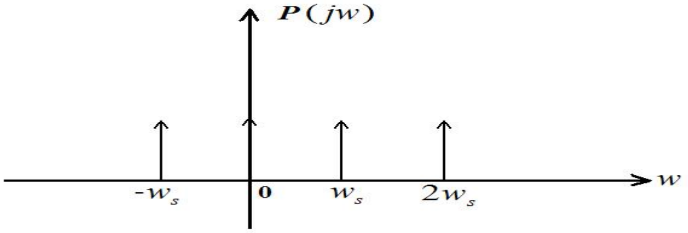


Fig. 1.3 $P(jw)$ is the CTFT of signal $p(t)$.

which consists of shifted replicas of $X_c(jw)$ occurring at $w = kw_s, k = 0, \pm 1, \pm 2, \pm 3, \ldots$ as shown in Figure **??**. Notice that it is assumed that $w_s - w_b > w_b$ in Fig. **??**, so that

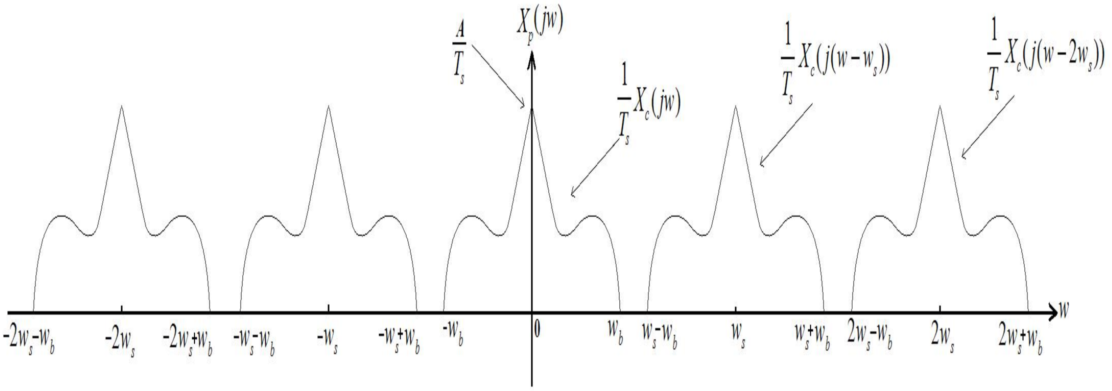


Fig. 1.4 The $X_p(jw)$ which is the CTFT of signal $x_p(t)$ with the assumption $w_s > 2w_b$.

there is no overlap between $(1/T_s)X_c(jw)$ and $(1/T_s)X_c(jw \pm w_s)$. This means that the original signal $x_c(t)$ can be recovered from its samples $x_p(t)$ by simple low-pass filtering:

$$X_c(jw) = H_c(jw)X_p(jw) \tag{1.4}$$

where $H_c(jw)$ is a perfect low-pass filter with cut-off $w_s/2$ and an amplification factor T_s. Continuous-time to discrete-time (C/D) conversion process is summarized in Figure **??**. *Notice that we do not compute Fourier Transforms during signal sampling (C/D conversion). We use the Fourier analysis to prove Shannon's sampling theorem.*. In practice:

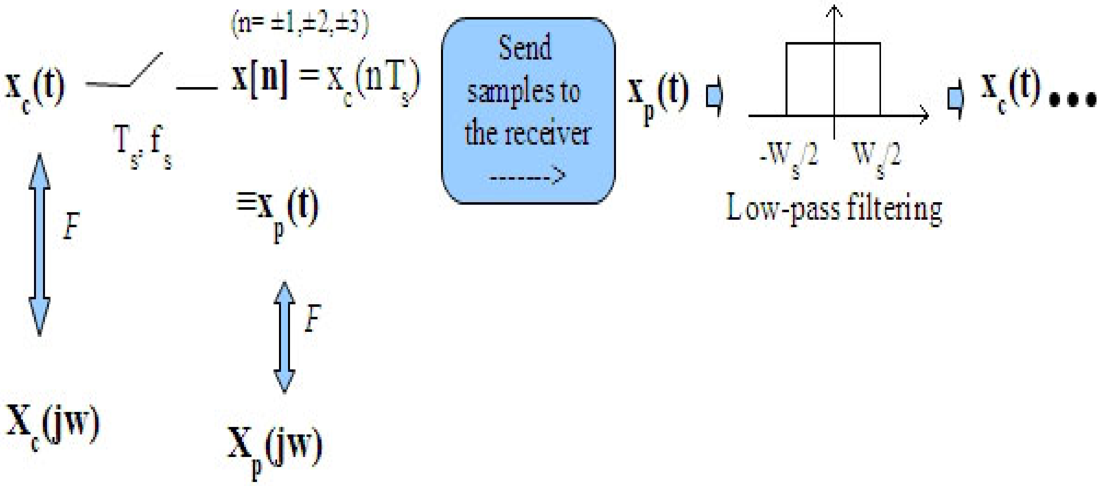


Fig. 1.5 Summary of signal sampling and signal reconstruction from samples.

- We cannot realize a perfect low-pass filter. We use an ordinary analog low-pass filter to reconstruct the continuous-time signal from its samples. Therefore, the reconstructed signal $\tilde{x}_c(t) \neq x_c(t)$ but it is very close to the original signal provided that we satisfy the Nyquist rate $w_s > 2w_b$. A practical signal reconstruction system is shown in Fig. **??**.
- The signal $x_p(t)$ is not used as an input to the low-pass filter during reconstruction, either, but a staircase signal is used. This is because we can not generate impulses.
- In Analog to Digital (A/D) converters, there is a built-in low-pass filter with cut-off frequency $\frac{f_s}{2}$ to minimize aliasing.
- In digital communication systems samples $x[n]$ are transmitted to the receiver instead of the continuous-time signal $x_c(t)$. In audio CD's samples are stored in the CD. In MP3 audio, samples are further processed in the computer and parameters representing samples are stored in the MP3 files.
- In telephone speech, $f_s = 8$ kHz, although a typical speech signal has frequency components up to 15 KHz. This is because we can communicate or understand the speaker even if the bandwidth is less than 4KHz. Telephone A/D converters apply a low-pass filter with a 3dB cut-off frequency at 3.2 KHz before sampling the speech at 8KHz. That is why we hear ”mechanical sound” in telephones.
- All finite-extent signals have infinite bandwidths. Obviously, all practical message signals are finite extent signals (even my mother-in-law cannot talk forever). Therefore, we can have approximately low-pass signals in practice.
- We use the angular frequency based definition of the Fourier Transform in this course:

$$X_c(jw) = \int_{-\infty}^{\infty} x_c(t)e^{-jwt}dt$$

where $w = 2\pi f$. In this case the inverse Fourier Transform becomes

$$x_c(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X_c(w)e^{jwt}dw$$

In most introductory telecommunications books they use

$$\hat{X}(f) = \int_{-\infty}^{\infty} x_c(t)e^{-j2\pi ft}dt$$

which leads to the inverse Fourier Transform:

$$x_c(t) = \int_{-\infty}^{\infty} \hat{X}(f)e^{j2\pi ft}df.$$

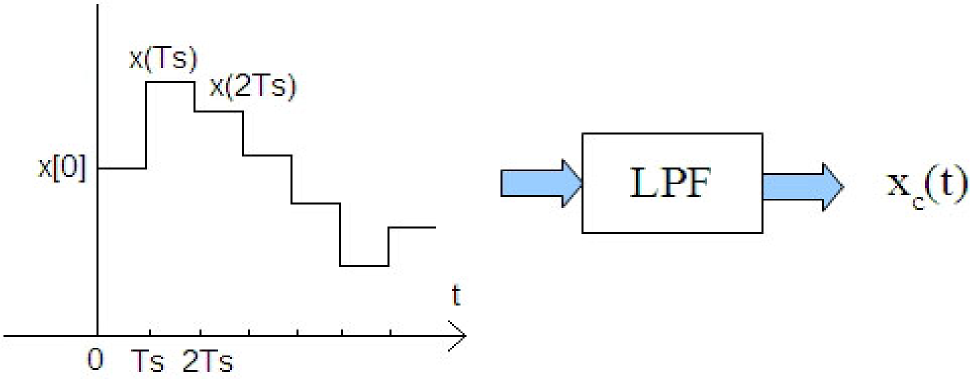


Fig. 1.6 Practical digital to Analog (D/A) conversion: Signal reconstruction from samples $x[n] = x_c(nT_s), \quad n = 0, \pm1, \pm2, \pm3, \ldots$. Ideal analog low-pass filter does not have a flat response in pass-band but this is very hard to achieve in practice because the low-pass filter is constructed from analog components.

1.2 Aliasing

We cannot capture frequencies above $\frac{w_s}{2}$ when the sampling frequency is w_s.

When $w_s < 2w_b$ the high frequency components of $X_c(jw)$ are corrupted during the sampling process and it is impossible to retrieve $x_c(t)$ from its samples $x[n] = x_c(nT_s)$. This phenomenon is called aliasing (see Figure **??**). I will put an aliased speech signal into course web-page. Visit Prof. Cevdet Aykanat's web-page and take a look at his jacket using Firefox. Unusual patterns in his jacket are due to

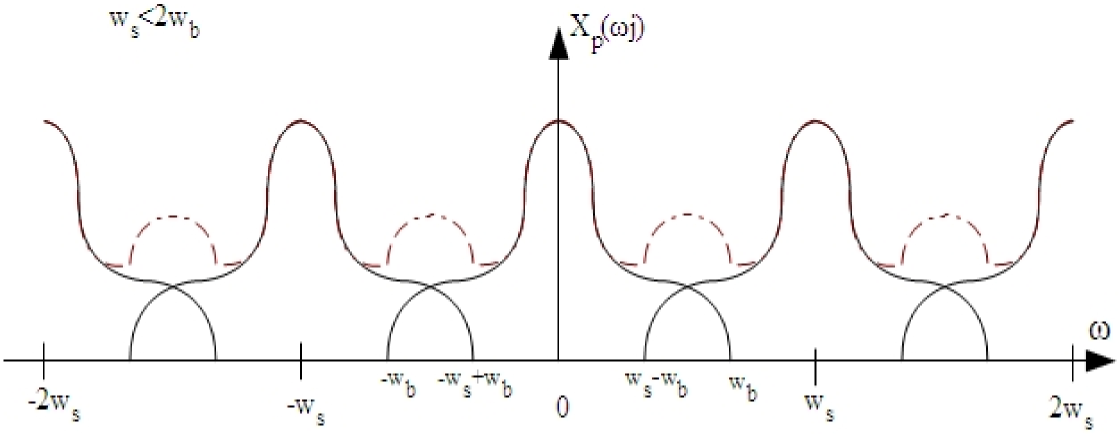


Fig. 1.7 Aliasing occurs when $w_s < 2w_b$.

Fig. 1.8 Prof. Cevdet Aykanat of Bilkent Universiy.

undersampling (see Fig.**??**). Firefox engineers do not know basic multi-rate signal processing theory that we will study in this course (perhaps there are no electrical

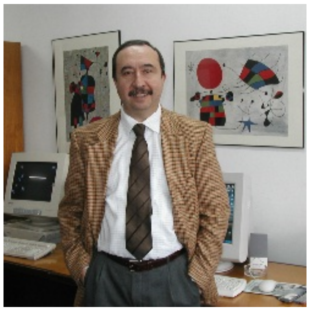


Fig. 1.9 Prof. Cevdet Aykanat's aliased image. Take a look at the artificial patterns in his jacket because of aliasing. The image in Fig. **??** is horizontally and vertically downsampled by a factor of 2.

engineers among Firefox developers). We contacted them in December 2010 and they said that they would fix this ”bug” in the future. On the other hand Google's Chrome and MS-Explorer provide smoother patterns because they use a low-pass filter before downsampling. Visit the same web-page using MS-Explorer or Google-Chrome.

1.3 Relation between the DTFT and CTFT

The Discrete-Time Fourier Transform (DTFT) and CTFT are two different transforms but they are related to each other. The CTFT $X(j\Omega)$ of the continuous-time signal $x_c(t)$ is given by

$$X_c(j\Omega) = \int_{-\infty}^{\infty} x_c(t) e^{-j\Omega t} dt \tag{1.5}$$

DTFT $X(e^{j\omega})$ of a discrete-time signal $x[n]$ is defined as follows:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} \tag{1.6}$$

Notice that I need to use two different angular frequencies in the above two equations. From now on I will use Ω for the actual angular frequency which is used in the CTFT and ω for the normalized angular frequency of the DTFT, respectively. This is the notation used in Oppenheim and Schaefer's book [**?**]. In McClellan's book they use ω for actual angular frequency and $\hat{\omega}$ for the normalized angular frequency [**?**]. So the Fourier Transform of a sampled version $x_p(t)$ of a band-limited signal $x_a(t)$ is shown in Figure **??**.

The normalized angular frequency $\omega = \pi$ corresponds to the actual angular frequency $\Omega_s/2$ because

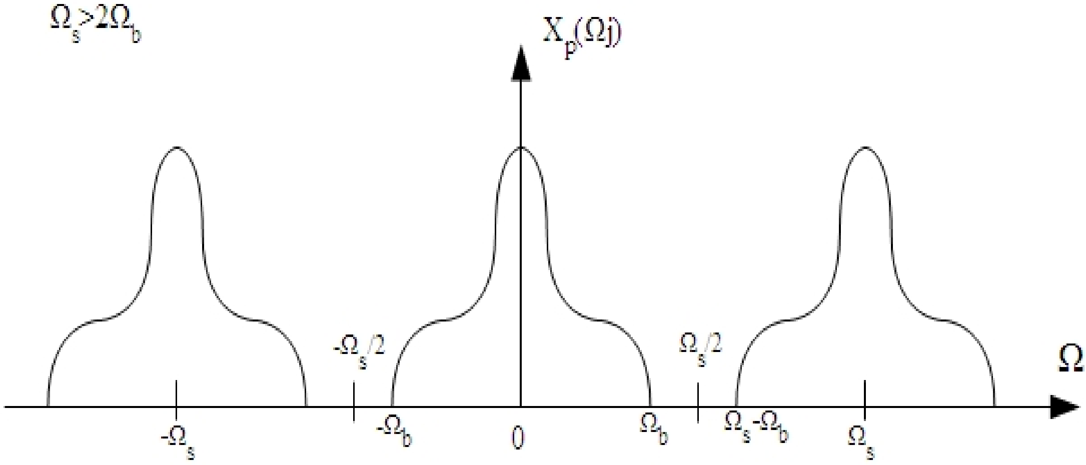


Fig. 1.10 The CTFT of $x_p(t)$. Sampling frequency $\Omega_s > 2\Omega_b$. This is the same plot as the Fig. **??**.

$$\omega = \frac{\Omega_s}{2} T_s = \frac{1}{2}\left(\frac{2\pi}{T_s}\right) T_s = \pi$$

Therefore the highest frequency that we can have in discrete-time Fourier Transform is the half of the sampling frequency.

1.4 Continuous-Time Fourier Transform of $x_p(t)$

The signal $x_p(t)$ is a continuous-time signal but its content is discrete in nature. It just contains impulses whose strength are determined by the analog signal samples. As you know $x_p(t)$ can be expressed as follows:

$$x_p(t) = \sum_{n=-\infty}^{\infty} x_a(nT_s)\,\delta(t - nT_s)$$

Let us now compute the CTFT $X_p(j\Omega)$ of $x_p(t)$:

$$X_p(j\Omega) = \int_{-\infty}^{\infty}\left(\sum_{n=-\infty}^{\infty} x_a(nT_s)\,\delta(t - nT_s)\right) e^{-j\Omega t} dt$$

$$= \sum_{n=-\infty}^{\infty} x_a(nT_s) \int_{-\infty}^{\infty} \delta(t - nT_s) e^{-j\Omega t} dt$$

$$= \sum_{n=-\infty}^{\infty} x_a(nT_s)e^{-j\Omega nT_s} \int_{-\infty}^{\infty} \delta(t-nT_s)dt$$

Therefore the CTFT $X_p(j\Omega)$ of $x_p(t)$ can be expressed as a sum as follow

$$X_p(j\Omega) = \sum_{n=-\infty}^{\infty} x(nT_s)e^{-j\Omega T_s n}, \tag{1.7}$$

Now, consider the discrete-time signal $x[n] = x(nT_s)$ which is equivalent to the continuous-time signal $x_p(t)$. The discrete-time Fourier Transform (DTFT)of $x[n]$ is defined as follows

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

This is the same as Equation (7) when we set $\omega = \Omega T_s$.

As you see, DTFT did not come out of blue and ω is called the normalized angular frequency. The normalization factor is determined by the sampling frequency f_s or equivalently by the sampling period T_s.

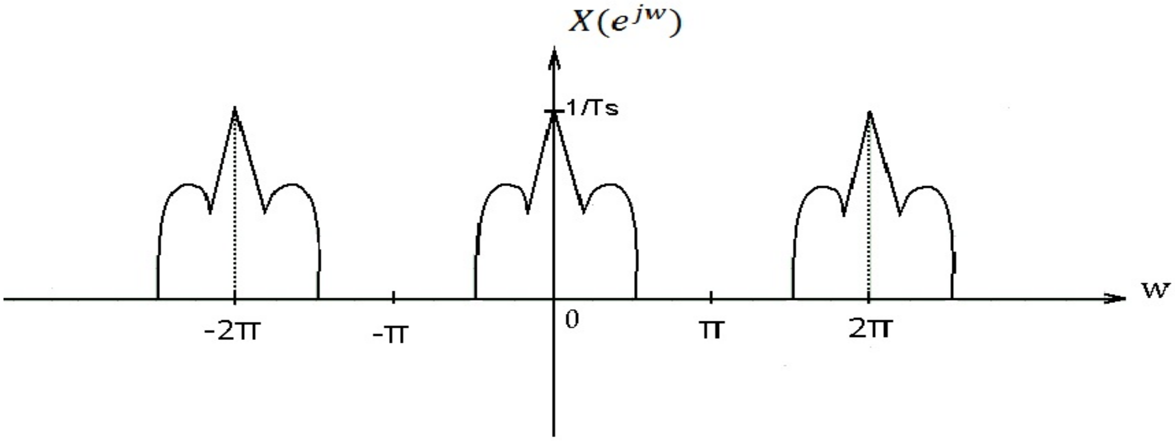


Fig. 1.11 The DTFT $X(e^{j\omega})$ of $x[n]$. It has the same shape as the CTFT of $x_p(t)$. Only the horizontal axis is normalized by the relation $\omega = \Omega T_s$ (The amplitude A is selected as 1: $A = 1$).

Since the CTFT $X_p(j\Omega)$ is periodic with period Ω_s the DTFT $X(e^{j\omega})$ is 2π periodic. This is due to the fact that $\omega = \Omega T_s$. The normalized angular frequency $\omega = \pi$ corresponds to the actual angular frequency $\Omega_s/2$ because

$$\omega = \frac{\Omega_s}{2}T_s = \frac{1}{2}\left(\frac{2\pi}{T_s}\right)T_s = \pi$$

Therefore, $\omega = \pi$ is the highest frequency that we can have in discrete-time Fourier Transform. The normalized angular frequency $\omega = \pi$ corresponds to the actual angular frequency of $\Omega_s/2$ which is the half of the sampling frequency.

Here is a table establishing the relation between the actual angular frequency and the normalized frequency.

ω	Ω
0	0
2π	$2\Omega_s$
$(\Omega_s T_s)/2 = \pi$	Ω_s
$(\Omega_o T_s)/2 = \omega_o/2$	Ω_o

When the sampling frequency is $2\Omega_s$, the highest normalized frequency π corresponds to Ω_s.

1.5 Inverse DTFT

Inverse DTFT is computed using the following formula

$$x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})\, e^{j\omega n} d\omega,\ n = 0, \pm 1, \pm 2, \ldots \tag{1.8}$$

Sometimes we may get an analytic expression for the signal $x[n]$ but in general we have to calculate the above integral for each value of n to get the entire $x[n]$ sequence.

Since the DTFT is 2π periodic function limits of the integral given in Eq. (1.8) can be any period covering 2π.

1.6 Inverse CTFT

Inverse CTFT of $X_c(j\Omega)$ is obtained using the following formula

$$x_c(t) = \frac{1}{2}\int_{-\infty}^{\infty} X_c(j\Omega) e^{j\Omega t} d\Omega \tag{1.9}$$

In some books the forward and the inverse CTFT expressions are given as follows:

$$\hat{X}_c(f) = \frac{1}{2}\int_{-\infty}^{\infty} x_c(t) e^{-j2\pi f t} dt \tag{1.10}$$

and

$$x_c(t) = \int_{-\infty}^{\infty} \hat{X}_c(f) e^{j2\pi f t} df \tag{1.11}$$

Let $\Omega = 2\pi f$ in Eq. (**??**). As a result $d\Omega = 2\pi df$ and we obtain Eq. (**??**).

This confuses some of the students because the CTFT of $\cos(2\pi f_o t)$ is $0.5(\delta(f - f_o) + \delta(f + f_o)$ according to (**??**) and $\pi(\delta(\Omega - 2\pi f_o) + \delta(\Omega + 2\pi f_o)$ according to (**??**). This is due to the fact that the CTFT is defined in terms of the angular frequency in (**??**) and in terms of frequency in (**??**), respectively.

1.7 Filtering Analog Signals in Discrete-time Domain

It is possible to use sampling theorem to filter analog (or continuous-time) signals in discrete-time domain.

Let us assume that $x_c(t)$ is a band-limited signal with bandwidth Ω_0. We want to filter this signal with a low-pass filter with cut-off frequency of $\Omega_c = \frac{\Omega_0}{2}$. In this case, we sample $x_c(t)$ with $\Omega_s = 2\Omega_0$ and obtain the discrete-time filter:

$$x[n] = x_c(nTs),\ n = 0, \pm 1, \pm 2, \ldots \tag{1.12}$$

The angular cut-off frequency $\frac{\Omega_0}{2}$ corresponds to normalized angular frequency of ω_c:

$$\omega_c = \frac{\Omega_0}{2} T_s = \frac{\Omega_0}{2} \frac{2\pi}{2\Omega_0} = \frac{\pi}{2} \tag{1.13}$$

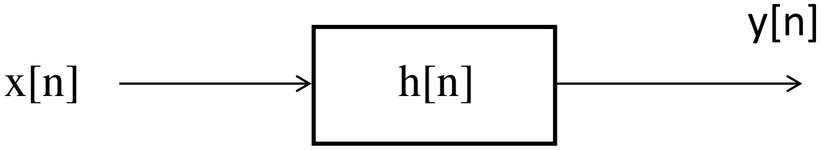
Therefore, we can use a discrete-time filter with cut-off frequency $\omega_c = \frac{\pi}{2}$ to filter $x[n]$ and obtain $x_0[n]$. Finally, we use a D/A converter to convert $x_0(t)$ to the analog domain and we achieve our goal. In general, if the cut-off frequency of the analog filter is Ω_c then the cut-off frequency of the discrete-time filter $\omega_c = \Omega_c T_s$. Similarly, we can perform band-pass, band-stop, and high-pass filtering in discrete-time domain. In general, this approach is more reliable and robust than analog filtering because analog components (resistors, capacitors and inductors) used in an analog filter are not perfect [**?**].

We can even low-pass, band-pass and band-stop filter arbitrary signals in discrete-time domain. All we have to do is to select a sampling frequency Ω_s well-above the highest cut-off frequency of the filter. Practical A/D converters have built-in analog low-pass filters to remove aliasing. Therefore, they remove the high-frequency components of the analog signal. In discrete-time domain the full-band corresponds to 0 to $\frac{\Omega_2}{2}$ of the original signal.

1.8 Exercises

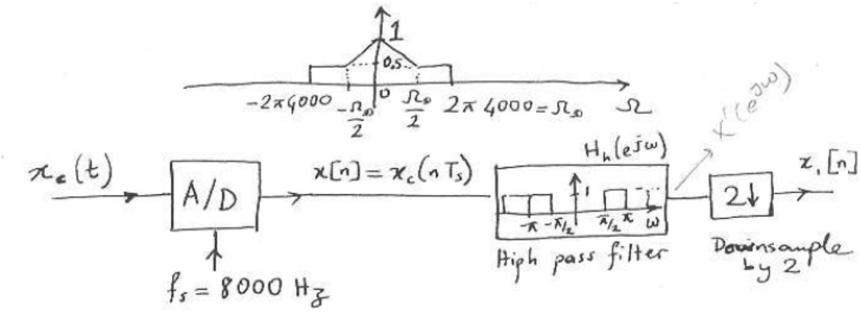
1. Consider the following LTI system characterized by the impulse response $h[n] = \left\{ -\frac{1}{2}, \underbrace{1}_{n=0}, -\frac{1}{2} \right\}$.

(a) Is this a causal system? Explain.



(b) Find the frequency response $H(e^{j\omega})$ of $h[n]$.
(c) Is this a low-pass or a high-pass filter?
(d) Let $x[n] = \left\{ \underbrace{1}_{n=0}, 2, 2 \right\}$. Find $y[n]$.

2. Let $x_c(t)$ be a continuous time signal with continuous time Fourier transform $X_c(j\Omega)$.
Plot the frequency domain functions $X(e^{j\omega})$ and $X_1(e^{j\omega})$.



3. Let the sampling frequency be $f_s = 8$ kHz. Normalized angular frequency $\omega_0 = \pi/4$ corresponds to which actual frequency in kHz?

4.

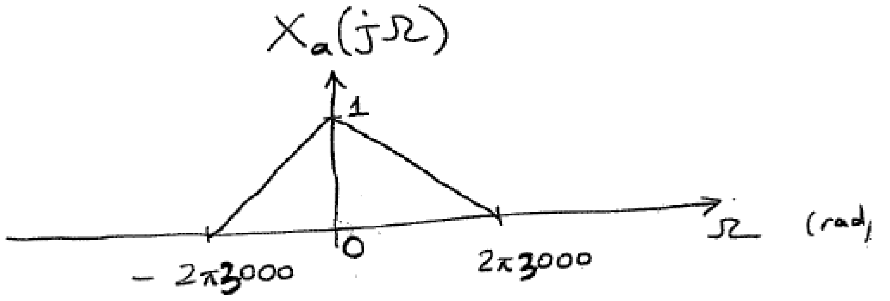
$$H(z) = \frac{z+0.8}{z^2 - 1.4z + 0.53}$$

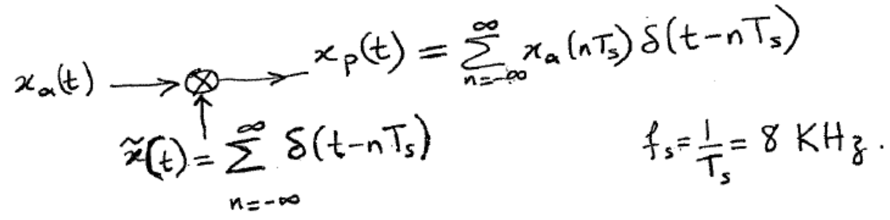
(a) Plot the locations of poles and zeros on the complex plane.
(b) How many different LTI filters may have the given $H(z)$. What are their properties? Indicate the associated regions of convergence.
(c) If the system is causal, is it also stable?
(d) Make a rough sketch of the magnitude response $|H(e^{j\omega})|$. What kind of filter is this?
(e) Give an implementation for the causal system using delay elements, vector adders and scalar real multipliers.

(f) Let $H_1(z) = H(z^2)$. Roughly plot $|H_1(e^{j\omega})|$ in terms of your plot in part (d).
(g) Repeat part (e) for H_1.
5. Let the sampling frequency be $f_s = 10$ kHz. Actual frequency is $f_0 = 2$ kHz. What is the normalized angular frequency ω_0 corresponding to $f_0 = 2$ kHz?
6. Given

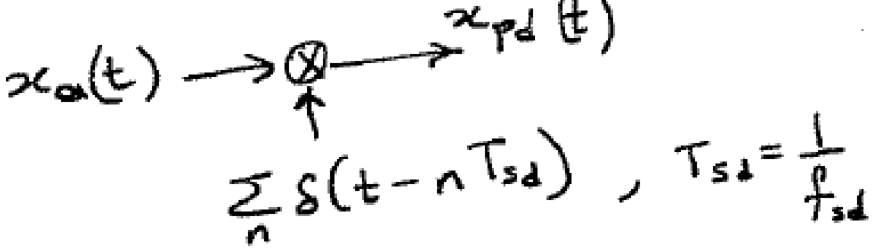
$$H(z) = \frac{1}{1 - \frac{1}{2}z^{-1}}$$

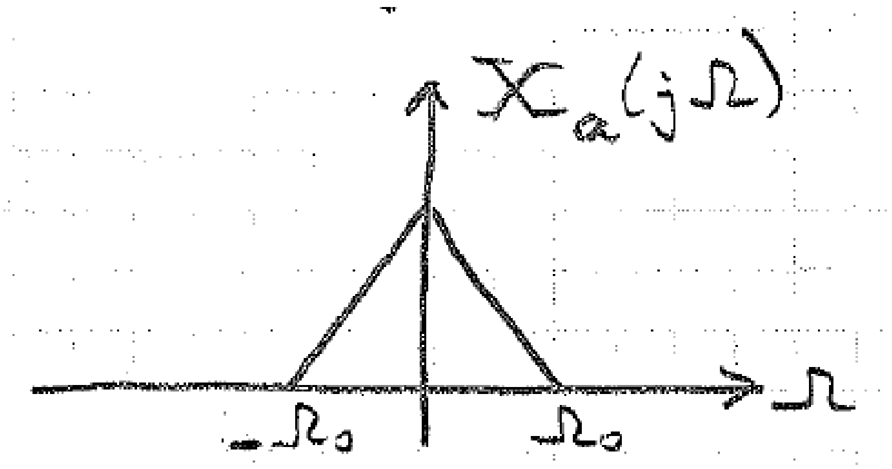
(a) Find the time domain impulse responses corresponding to $H(z)$.
(b) Indicate if they are stable or not.
7. Given $x_a(t)$ with continuous time Fourier Transform: (a) Plot $X_p(j\Omega)$ where




$x_a(t)$
$x_p(t) = \sum_{n=-\infty}^{\infty} x_a(nT_s)\,\delta(t-nT_s)$
$\tilde{x}(t) = \sum_{n=-\infty}^{\infty} \delta(t-nT_s)$
$f_s = \frac{1}{T_s} = 8$ KHz.

(b)Let the sampling frequency be $f_{sd} = 4$ kHz. Plot $X_{pd}(j\Omega)$ where
(c) Plot the DTFT of $x[n] = \{x_a(nT_s)\}_{n=-\infty}^{\infty}$.
(d) Plot the DTFT of $x[n] = \{x_d(nT_{sd})\}_{n=-\infty}^{\infty}$.
(e) Can you obtain $x_d[n]$ from $x[n]$? If yes, draw the block diagram of your system obtaining $x_d[n]$ from $x[n]$. If no, explain.
(f) Can you obtain $x[n]$ from $x_d[n]$? If yes, draw the block diagram of your system obtaining $x[n]$ from $x_d[n]$. If no, explain.
8. Given $x_a(t)$. We want to sample this signal. Assume that you have an A/D converter with a high sampling rate. How do you determine an efficient sampling frequency for $x_a(t)$?
9. Let $X_a(j\Omega)$ be the CTFT of $x_a(t)$:





We sample $x_a(t)$ with $\omega_s = 4\omega_0$ which results in $x_p(t)$. Plot $X_p(j\Omega)$.

10. Let $h[n] = \left\{ \frac{1}{4}, \underbrace{1}_{n=0}, \frac{1}{4} \right\}$. Calculate and plot the frequency response.

11. Let $x(t)$ be a continuous time signal (bandlimited) with maximum angular frequency $\Omega_0 = 2\pi 2000$ rad/sec. What is the minimum sampling frequency Ω_s which enables a reconstruction of $x(t)$ from its samples $x[n]$?

12. Consider the continuous-time signal $x(t) = sin(2\pi at) + sin(2\pi bt)$, where $b > a$.
(a) Plot the continuous-time Fourier-Transform $X(j\Omega)$ of $x(t)$.
(b) What is the lower bound for the sampling frequency so that $x(t)$ can be theoretically reconstructed from its samples?
(c) Plot the block-diagram of the system which samples $x(t)$ to yield the discrete-time signal $x[n]$ without aliasing. Specify all components. Hint: use impulse-train.
(d) Plot the block-diagram of the system which reconstructs $x(t)$ from $x[n]$. Specify all components.

13. Consider the FIR filter $y[n] = h[n] * x[n]$, where $h[n] = \delta[n+1] - 2\delta[n] + \delta[n-1]$.
(a) Compute output of $x[n] = \delta[n] - 3\delta[n-1] + 2\delta[n-2] + 5\delta[n-3]$.
(b) Calculate the frequency response $H(e^{j\omega})$ of $h[n]$.

(c) Determine a second order FIR filter $g[n]$ so that the combined filter $c[n] = g[n] * h[n]$ is causal. Calculate $c[n]$.
(d) Compute the output of the input sequence given in part (a) using the filter $c[n]$. Compare with the result of part (a).
(e) Calculate the frequency response $H_c(e^{j\omega})$ of the filter $c[n]$. Compare with $H(e^{j\omega})$ from part (b).
14. Consider the IIR filter $y[n] = x[n] + y[n-1] - y[n-2]$.
(a) Compute output of $y[n]$, $n = 0, \ldots, 8$ of this filter for $x[n] = 4\delta[n] - 3\delta[n-1] + \delta[n-2]$. Assume $y[n] = 0$ for $n < 0$.
(b) Determine $y[k+1+6n]$ for $k = 0, \ldots, 5$, $n \geq 0$. E.g. $y[1+6n] = \ldots$, $y[2+6n] = \ldots, \ldots, y[6+6n] = \ldots$.
(c) Compute the z-transform $H(z)$ of the IIR filter.
(d) Compute the corresponding frequency response $H(e^{j\omega})$.
(e) Plot the flow-diagram of the filter.

Chapter 2
Multirate Signal Processing

Multirate signal processing is used to modify windows in a computer screen or to enlarge or to reduce the sizes of images in a computer screen. It is also used in wavelet theory [**?**] and telecommunications.

We first study interpolation.

2.1 Interpolation

There are many ways to interpolate a discrete-time signal or a sequence of numbers. The straightforward approach is to use the linear interpolation in which the average of the consecutive samples are taken as the interpolated signal values. In this way, we reduce the sampling rate from T_s to $T_s/2$ or equivalently, we increase the angular sampling frequency from Ω_s to $2\Omega_s$. If the signal has N samples, the interpolated signal has 2N samples.

We assume that the discrete-time signal $x[n]$ is obtained from an analog signal $x_c(t)$[1] by sampling. We also assume that $x_c(t)$ is a band-limited signal as shown in Fig. **??** .

We have two discrete-time signals associated with the continuous-time signal $x_c(t)$ in Figure **??**. These are $x[n]$ and $x_i[n]$:

$$x[n] = x_c(nT_s) \qquad \text{sampled with } \Omega_s$$

$$x_i[n] = x_c(n\frac{T_s}{2}) \qquad \text{sampled with } 2\Omega_s$$

The signal $x_i[n]$ contains the samples of $x[n]$ because its sampling rate is $T_s/2$.

We want to obtain $x_i[n]$ from $x[n]$ using discrete-time domain processing. This is called discrete-time interpolation of $x[n]$ by a factor of $L = 2$.

Let $\tilde{x}_p(t)$ be the continuous-time signal equivalent to $x_i[n]$, i.e.,

[1] I use both $x_a(t)$ and $x_c(t)$ for continuous-time signals.

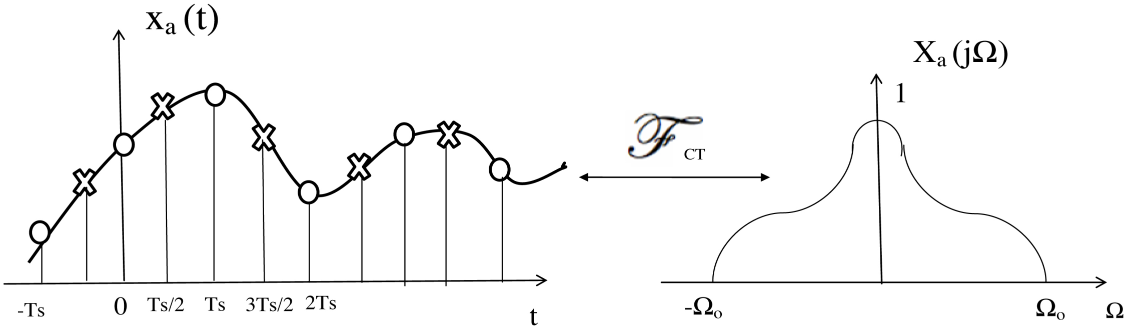


Fig. 2.1 $x_c(t)$ and its CTFT $X_c(j\Omega)$. The interpolation problem: obtain samples marked with "x" from the samples marked with "o".

$$x_i[n] \equiv \tilde{x}_p(t) = x_a(t) \times \sum_{n=-\infty}^{\infty} \delta(t - n\frac{T_s}{2}) = \sum_{n=-\infty}^{\infty} x_a(nT_s/2)\delta(t - n\frac{T_s}{2})$$

The CTFT $\tilde{X}_p(j\Omega)$ of $\tilde{x}_p(t)$ is shown in Fig. **??**

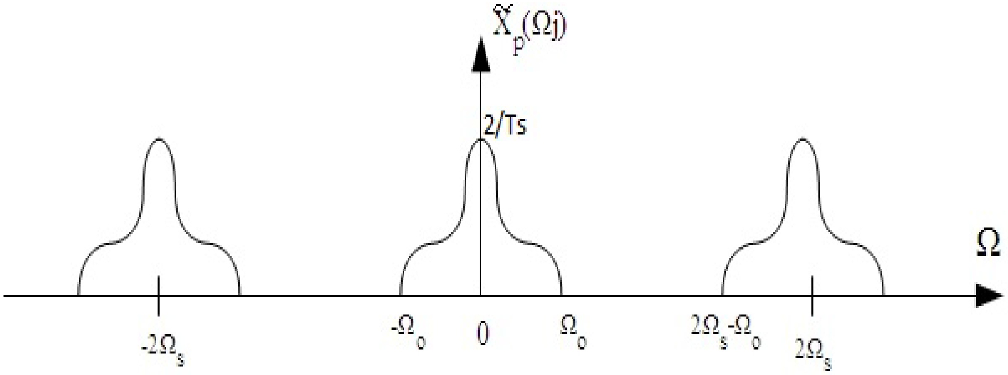


Fig. 2.2 The CTFT $\tilde{X}_p(j\Omega)$ of $\tilde{x}_p(t)$

The CTFT $\tilde{X}_p(j\Omega)$ is related with the DTFT $X_i(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x_i[n]e^{-j\omega n}$ as shown in Fig. **??**: Since the sampling period is $T_s/2$ the highest normalized angular frequency π corresponds to Ω_s. Therefore, the highest frequency component of the signal $x_i[n]$ is now $\omega_o/2$ as shown in Figure **??**

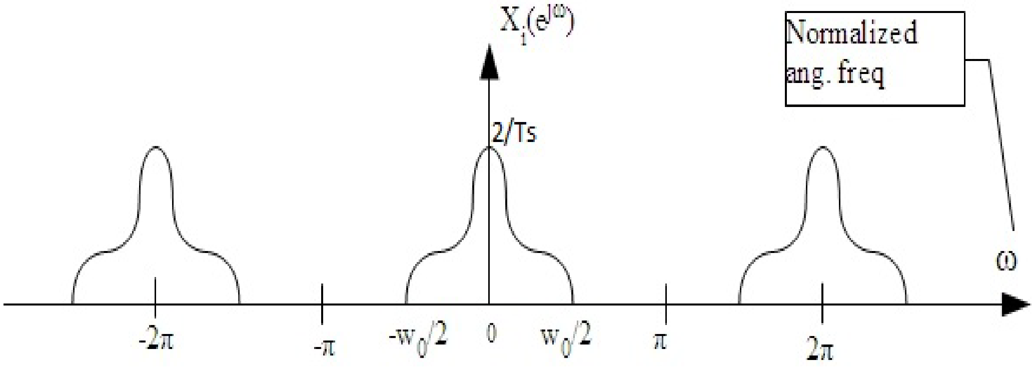


Fig. 2.3 The DTFT $X_i(e^{j\omega})$ of $x_i[n]$.

The Fourier transform of $x[n]$ is shown in Fig. **??**.

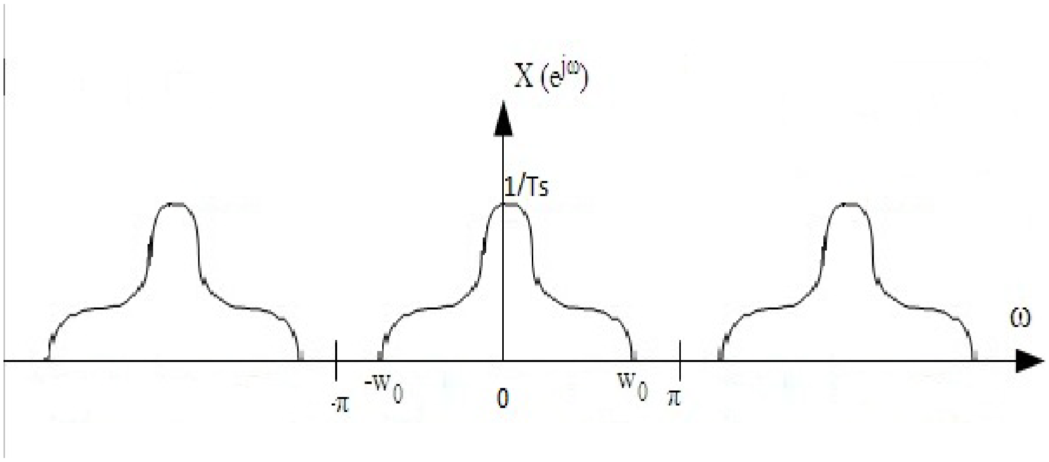


Fig. 2.4 The DTFT $X(e^{j\omega})$ of $x[n]$.

We cannot obtain $x_i[n]$ from $x[n]$ via low-pass filtering or any other filtering operation. We need to use a system called ”up-sampler” first to increase the number of samples of $x[n]$. This is done by simply inserting a zero valued sample between every other sample of $x[n]$ as follows:

$$x_u[n] = \begin{cases} x[n/2] & \text{, n even} \\ 0 & \text{, n odd} \end{cases}$$

The upsampling operation can be also considered as a modification of the sampling rate. In other words, the upsampler changes the sampling rate from T_s to $T_s/2$. Block diagram of an upsampler by a factor of L=2 is shown in Figure **??**.

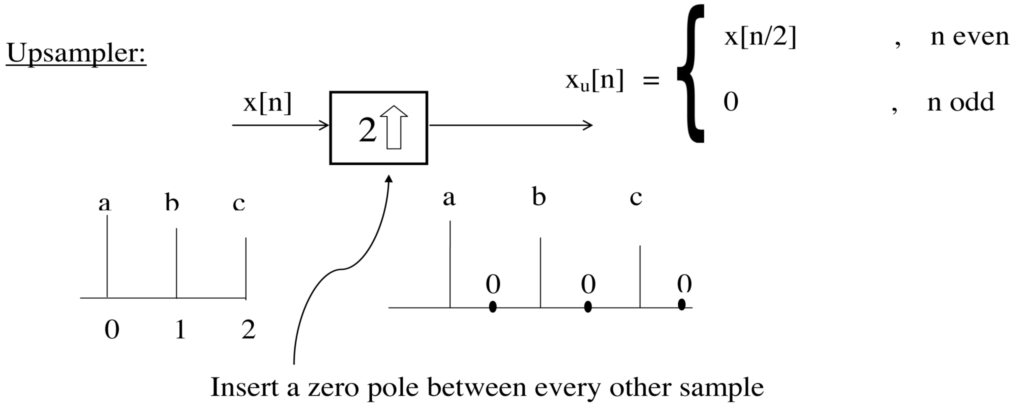

Upsampler:
x[n]
2⇧
$x_u[n] = \begin{cases} x[n/2] & , \text{ n even} \\ 0 & , \text{ n odd} \end{cases}$
a
b
c
0
1
2
a
b
c
0
0
0
Insert a zero pole between every other sample

Fig. 2.5 Upsampler by a factor of L=2. It inserts a zero-valued sample between every other sample of $x[n]$.

The effective sampling rate of $x_u[n]$ is $T_s/2$ as $x_i[n]$. Let us compute the DTFT of $x_u[n]$ and see if we can obtain x_i from x_u.

$$X_u(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x_u[n]e^{-j\omega n} \tag{2.1}$$

which can be simplified because all odd indexed samples of $x_u[n]$ are equal to zero. Therefore

$$X_u(e^{j\omega}) = x_u[0]e^{-j\omega 0} + x_u[1]e^{-j\omega 1} + x_u[-1]e^{j\omega 1} + x_u[2]e^{-j\omega 2} + x_u[-2]e^{j\omega 2} + \ldots$$

or

$$\begin{aligned} X_u(e^{j\omega}) = \overbrace{x_u[0]}^{=x[0]} + \overbrace{x_u[1]}^{=0} e^{-j\omega} + \overbrace{x_u[2]}^{=x[1]} e^{-j2\omega} + \cdots \\ + \underbrace{x_u[-1]}_{=0} e^{j\omega} + \underbrace{x_u[-2]}_{=x[-1]} e^{j2\omega} + \cdots \end{aligned}$$

This is because $x_u[n] = 0$ for all odd n values and $x_u[n/2] = x[n]$ for all even n values, therefore we have

$$X_u(e^{j\omega}) = x[0] + x[1]e^{-j\omega 2} + x[-1]e^{j\omega 2} + \ldots \tag{2.2}$$

and

$$X_u(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j2\omega n} \tag{2.3}$$

or

$$X_u(e^{j\omega}) = X(e^{j2\omega}) \tag{2.4}$$

The notation is somewhat clumsy but we have $(G(w) = H(2w))$ type relation between the two DTFT's in Equation (11). Therefore $X_u(e^{j\omega})$ has a period of π as shown in Figure **??**. We can simply use a low-pass filter with cut off $\pi/2$ to get rid of the high-frequency components of $x_u[n]$ and obtain $x_i[n]$. Notice that the DTFT of the low-pass filter is also 2π periodic.

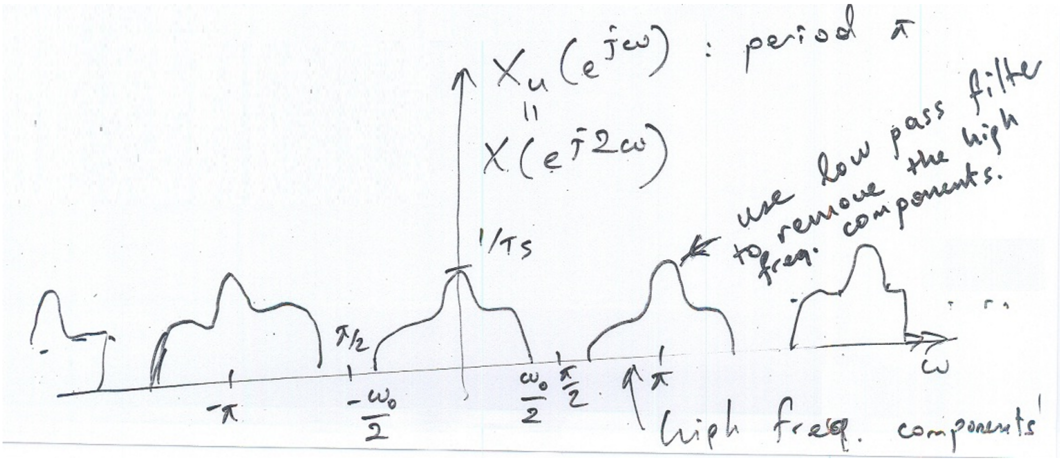


Fig. 2.6 The DTFT $X_u(e^{j\omega})$ of $x_u[n]$.

We need to amplify the low-pass filter output by a factor of 2 to match the amplitudes of Fig. **??** and Fig. **??**. Therefore a basic interpolation system consists of two steps: First upsample the input signal, then apply a low-pass filter with cut off $\pi/2$, and an amplification factor of 2. The block diagram of the Interpolation by a factor of 2 system is shown in Figure **??**. The discrete-time low-pass filter should be a perfect low-pass filter. In practice we cannot implement a perfect low-pass filter because the impulse response of an ideal low-pass filter extends from minus infinity to plus infinity. We have to design a practical low-pass filter approximating the perfect low-pass filter in some sense [**?**]. We will discuss the design of discrete time filters later in Chapter 4.

Here are some remarks:

- Upsampler is a linear operator but it is not a time-invariant system. Therefore it does not have an impulse response.
- Upsampler can be represented by a matrix and upsampling operation can be implemented using matrix-vector multiplication. See Ref. [**?**] for further details.

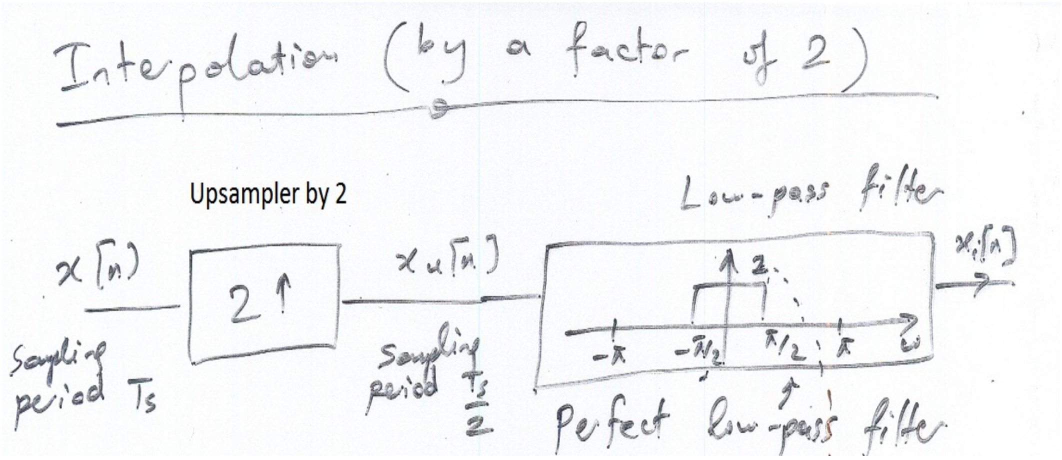


Fig. 2.7 Interpolation by a factor of L=2. Notice that the filter has an amplification factor of 2.

- As an exercise prove that the upsampler is a time varying system.

  Example: You can use the simple low-pass filter

$$h[n] = \{1/4, h[0] = 1/2, 1/4\}$$

as an interpolating low-pass filter. This filter has a frequency response

$$H(e^{j\omega}) = 0.5 + 0.5\cos(\omega)$$

It is not a great low pass filter but it is a low-pass filter. This filter is obviously periodic with period 2π.

Since $H(e^{j0}) = 1$ we need to amplify the filter by a factor of 2. Therefore, the filter $g[n] = 2h[n] = \{1/2, g[0] = 1, 1/2\}$.

The input/output (I/O) relation for this filter is

$$y[n] = 0.5x[n+1] + x[n] + 0.5x[n-1] \tag{2.5}$$

When we use this filter in Figure **??**, we obtain the following result:

$$x_i[n] = x[n/2] \; for \; even \;\; n \tag{2.6}$$

$$x_i[n] = 0.5x[(n-1)/2] + 0.5x[(n+1)/2] \; for \; odd \;\; n$$

This is simply linear interpolation. We take the average of two consecutive samples of $x[n]$ to estimate the odd indexed samples of $x_i[n]$. We can use filters with much nicer frequency response to perform interpolation and achieve better interpolation results. As pointed out earlier we will discuss FIR filter design later.

The filter in Eq. (12) is an anticausal filter but anticausality is not a major problem in discrete-time filtering. You can simple compute $y[n]$ whenever $x[n+1]$ becomes available.

2.2 Interpolation by an integer M

We need to use an upsampler by M first before low-pass filtering. Upsampler introduces an $M-1$ zeros between every other sample of $x[n]$.

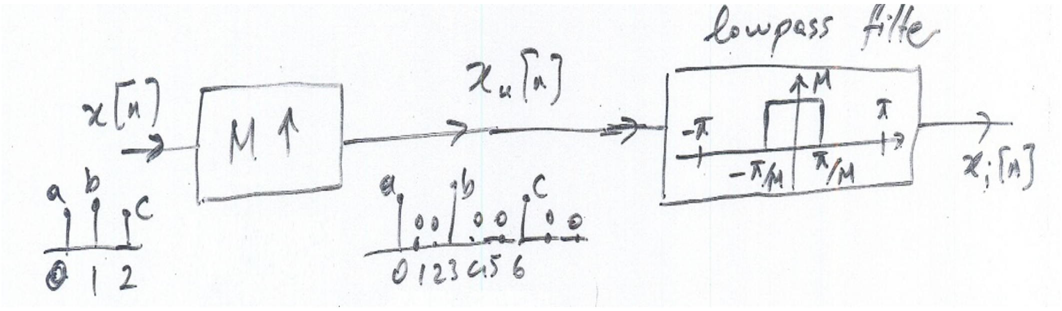


Fig. 2.8 Interpolation by a factor of M.

As a result the DTFT domain relation between $X_u(e^{j\omega})$ and $X(e^{j2\omega})$ is given by

$$X_u(e^{j\omega}) = X(e^{jM\omega}) \tag{2.7}$$

Therefore $X_u(e^{j\omega})$ has a period of $2\pi/M$ because $X(e^{j\omega})$ is periodic with period 2π as shown in Figure **??**.

By inserting zeros we also introduce high-frequency components as discussed in the previous section. We must remove the high frequency components by using a low-pass filter. The cut-off frequency of the low-pass filter must be π/M. It should amplify the input signal by a factor of M.

2.3 Decimation by a factor of 2

To reduce the number of samples of a given signal or image we can drop some of the samples. However, we cannot arbitrarily drop samples because we may suffer from aliasing because dropping samples corresponds to an increase in sampling period (or equivalently, it corresponds to a decrease in sampling frequency). A typical example is shown in Figure **??**. Prof. Aykanat's jacket has artificial stripes and lines. This is due to aliasing.

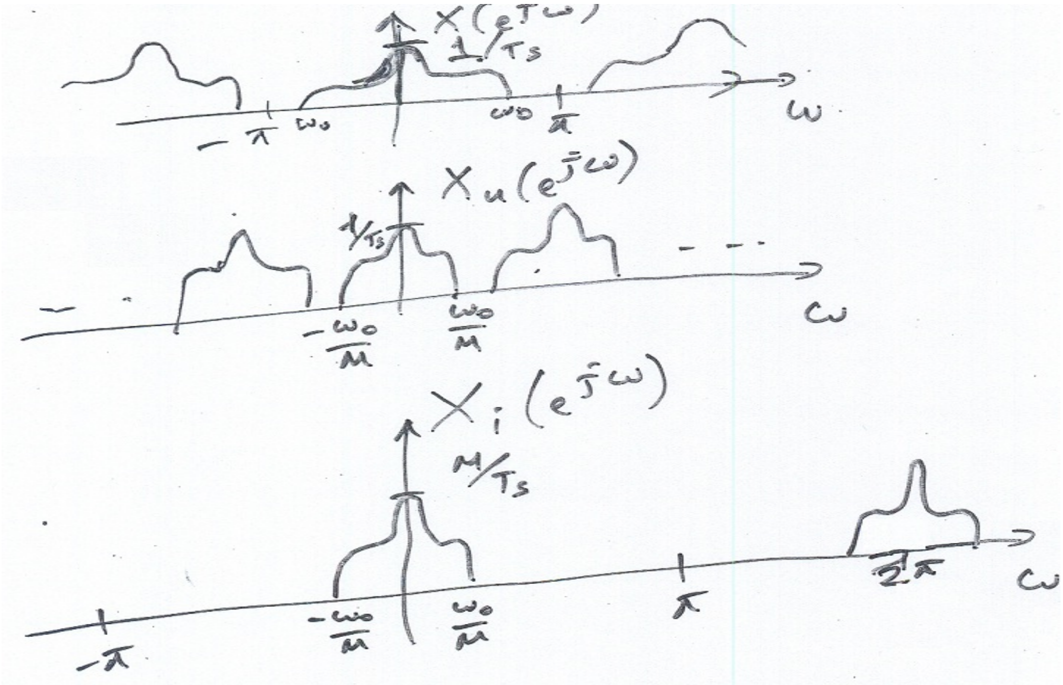


Fig. 2.9 Interpolation by a factor of M.

We use the down-sampling block shown in Figure **??** to represent the sampling rate change. In Figure **??** the system is a downsampling block by a factor of two, i.e., the output $x_d[n] = x[2n]$.

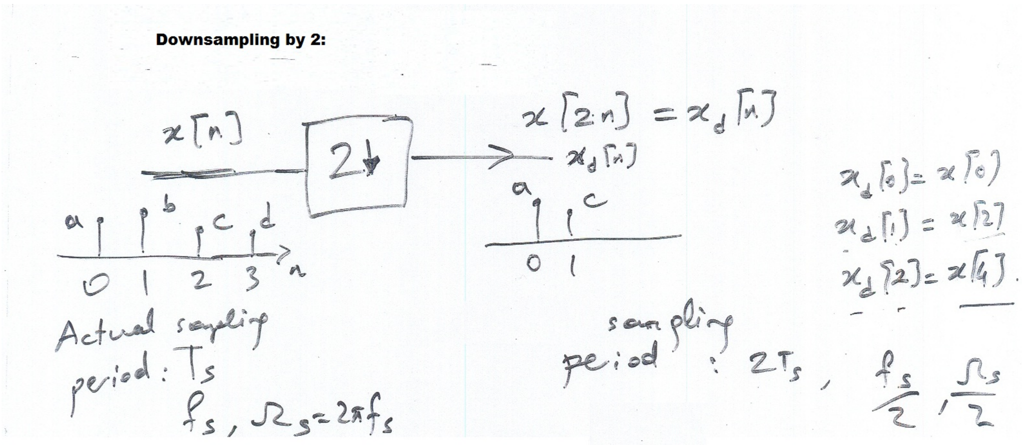


Fig. 2.10 Downsampling by a factor of L=2.

Example:

$$
\begin{aligned}
x_d[n] &= x[2n] \\
x[n] &= \{a,b,c,d,0,0,\cdots\} \\
x_d[n] &= \{a,c,0,0,\cdots\} \\
x_d[0] &= x[0], \quad x_d[1] = x[2], \quad x_d[2] = x[4]
\end{aligned}
$$

The effective sampling frequency is reduced by a factor of 2 by the downsampler.

Since $x[n]$ has a sampling period of $T_s\,(f_s,\,\Omega_s = 2\pi f_s)$ the down-sampled signal $x_d[n]$ has a sampling period of $2T_s\,(\frac{f_s}{2},\,\frac{\Omega_s}{2})$.

In Figure **??** the DTFT $X_d(e^{j\omega})$ of the signal $x_d[n]$ is shown (the bottom plot). The DTFT $X_d(e^{j\omega})$ is obtained from $X_{pd}(j\Omega)$ which is the CTFT of $x_{pd}(t)$. The signal $x_{pd}(t)$ is equivalent to $x_d[n]$:

$$x_d[n] \equiv x_{pd}(t) = x_a(t) \sum_{n=-\infty}^{\infty} \delta(t - n2T_s)$$

where $x_{pd}(t)$ is obtained from $x_a(t)$ via sampling with period $2T_s$. Therefore, the DTFT $X_d(e^{j\omega})$ is equivalent to $X_{pd}(j\Omega)$. Since $\Omega_o > \frac{\Omega_s}{2}$, we may have aliasing as shown in Figure **??**. When $\Omega_o > \frac{\Omega_s}{4}$, there must be aliasing in the downsampled signal. You cannot simply throw away samples!

By comparing the bottom plot of Fig. **??** with the DTFT of $x[n]$ shown in Fig. **??** we conclude that

$$X_d(e^{jw}) = \frac{1}{2}(X(e^{\frac{jw}{2}} + X(e^{\frac{jw+2\pi}{2}})) \tag{2.8}$$

The first component in Eq **??** $X(e^{\frac{jw}{2}})$ has a period of 4π and centered at $w = 0$. The second component $X(e^{\frac{jw+2\pi}{2}})$ is also periodic with period 4π but it is centered at $\mp 2\pi$. Neither $X(e^{\frac{jw}{2}})$ nor $X(e^{\frac{jw+2\pi}{2}})$ are valid DTFT expressions by themselves because they are not periodic with period 2π. But $X_d(e^{jw})$ in Eq. **??** is periodic with period 2π and it is the DTFT of $x_d[n]$.

If we had the continuous-time signal $x_c(t)$ we would low-pass filter this signal by an analog low-pass filter with cut-off frequency of $\Omega_s/4$ before sampling it with a sampling frequency of $\Omega_s/2$ to avoid aliasing. This analog filter can be implemented in discrete-time domain with a low-pass filter with a cut-off frequency of $\pi/2$ because $\Omega_s/4$ corresponds to the normalized angular frequency of $\pi/2$ when the sampling frequency is Ω_s. Therefore, we have to low-pass filter $x[n]$ with a low-pass filter with a cut-off frequency of $\pi/2$ before down-sampling as shown in Figure **??**.

Decimation by 2: Low-pass + Downsampling by 2

First low-pass filter the signal $x[n]$ with cut-off frequency $\pi/2$ and then downsample by a factor of 2. The term ”decimation” is a technical term and it refers to low-pass filtering and downsampling.

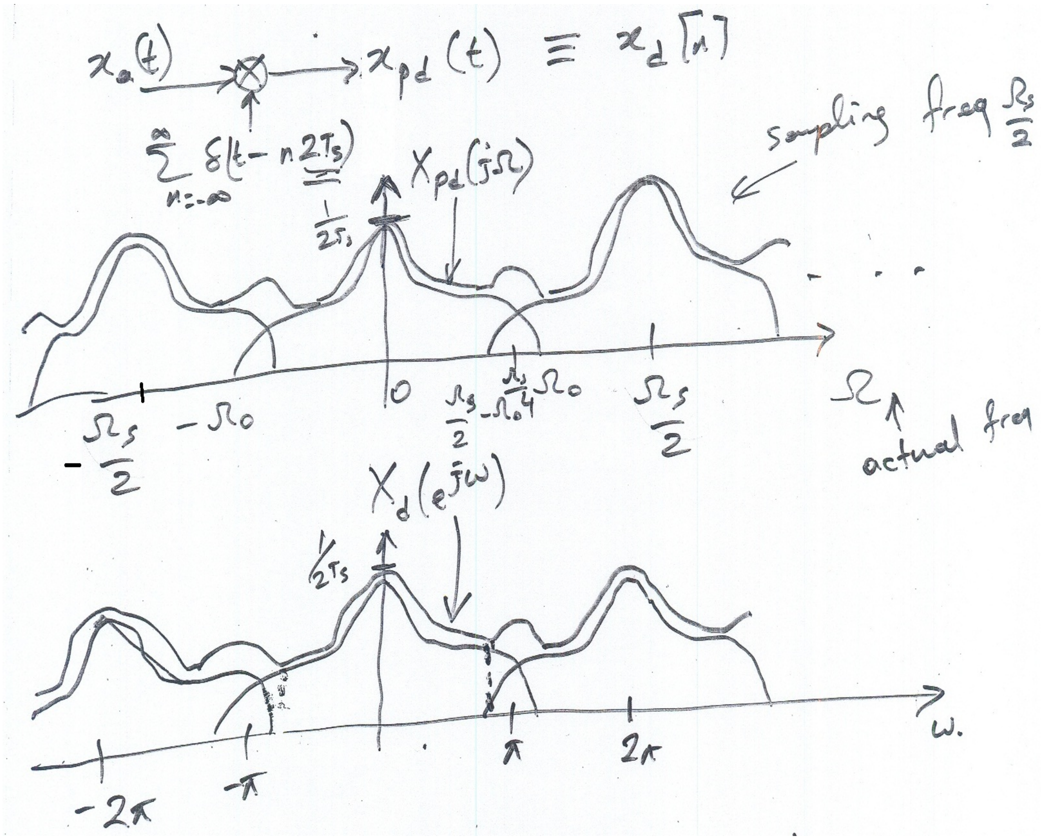


Fig. 2.11 The DTFT $X_d(e^{j\omega})$ of the signal $x_d[n]$ (bottom plot).

Decimation is a lossy operation in general because we low-pass filter the input first. As a result we remove the high-frequency components for good. After low-pass filtering it is not possible to retrieve the high-frequency bands of the input signal.

Example: The following low-pass filter can be used both in interpolation and decimation. In interpolation, it has to be amplified by a factor of 2!

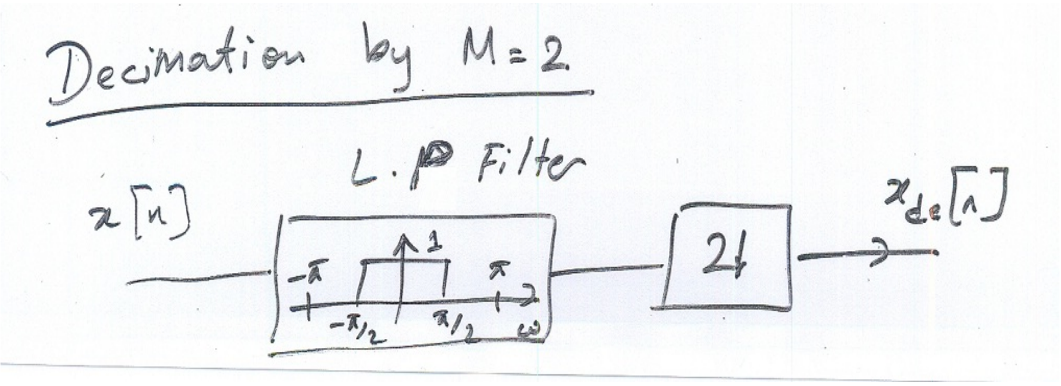


Fig. 2.12 Decimation by a factor of 2 is a two-stage operation.

Fig. 2.13 Prof. Cevdet Aykanat's properly decimated image. This image is a blurred version of the original image Fig. **??** but it does not have artificial patterns as Fig **??** . The image in is horizontally and vertically decimated by a factor of 2.

$$
\begin{aligned}
h[n] &= \{h[0] = 1/4, 1/2, 1/4\} \leftarrow \text{ Causal}\\
H(e^{j\omega}) &= \sum_{k=0}^{2} h[k] e^{-j\hat{\omega}k}\\
&= \frac{1}{4}e^{-j\hat{\omega}0} + \frac{1}{2}e^{-j\hat{\omega}1} + \frac{1}{4}e^{-j\hat{\omega}2}\\
&= e^{-j\hat{\omega}}\left(\frac{1}{4}e^{j\hat{\omega}} + \frac{1}{2} + \frac{1}{4}e^{-j\hat{\omega}}\right)\\
&= \left(\frac{1}{2} + \frac{1}{2}\cos(\hat{\omega})\right)e^{-j\hat{\omega}}
\end{aligned}
$$

The magnitude response of this filter is:

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

and the phase response of the filter is given by:

$$\phi(\hat{\omega}) = -\hat{\omega} \bmod (2\pi)$$

It is not a great half-band low-pass filter but it is a low-pass filter. *Example:* Here is a better half-band low-pass filter:

$$h[n] = \{h[0] = -1/32, 0, 9/32, 1/2, 9/32, 0, -1/32\} \tag{2.9}$$

This filter also has a cut-off frequency of $\pi/2$. That is why it is called "half-band" because the full-band refers to $[0,\pi]$. The frequency response of this filter is shown in **??**. The filter has a linear phase as shown in Fig. **??**

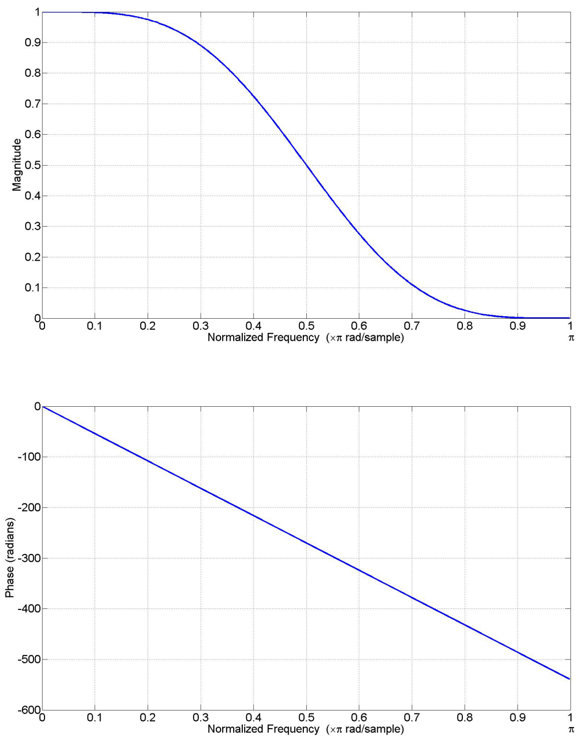


Fig. 2.14 Magnitude and the phase response of the filter given in Eq. (**??**).

The anticausal filter

$$h_a[n] = \{-1/32, 0, 9/32, h[0] = 1/2, 9/32, 0, -1/32\} \tag{2.10}$$

has the same magnitude response as shown in Fig. **??** but its phase response is zero: $\phi_a(w) = 0$.

2.4 Decimation by M

Decimation by integer $M > 2$ is achieved in two-stages as in decimation by a factor of 2 case. First low-pass filter the signal $x[n]$ with cut-off frequency π/M and then downsample by a factor of M.

Decimation is a lossy operation in general. This is because of the low-pass filtering operation. We remove the high-frequency components of the original filter using the low-pass filter. However, low-pass filtering is necessary to avoid aliasing.

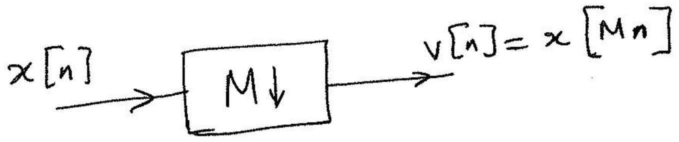


Fig. 2.15 Downsampling by a factor of M.

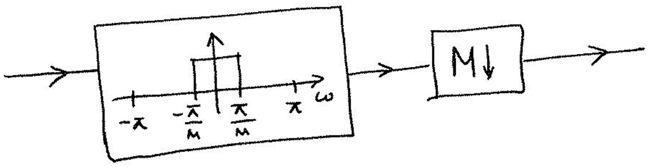


Fig. 2.16 Decimation by a factor of M.

2.5 Sampling Rate Change by a factor of L/M

You should first interpolate the original signal by L and then decimate. This is because the decimation operation is a lossy operation. When you remove the high-frequency components of the original signal by the low-pass filter you cannot retrieve those frequency components back. On the other hand we do not remove any signal component during interpolation.

Another advantage of this process is that you can combine the low-pass filters and perform this operation using a single low-pass filter.

Example: We first interpolate the input signal by a factor of L. Therefore, we insert L-1 zeros between every other sample of the input signal. Therefore we effectively decrease the sampling period from T_s to T_s/L. We low-pass filter the zero-padded signal with a low-pass filter with cut-off frequency π/L. This completes the interpolation stage. The interpolated signal is first low-pass filter with cut-off π/M by the decimation filter. Therefore we can combine the two filters and perform a single convolution (or filtering), if we use a low-pass filter with cut-off frequency of $\omega_c = \min(\pi/M, \pi/L)$. The amplification factor of the low-pass filter should be L, which is the amplification factor of the interpolation filter. After low-pass filtering we down-sample by a factor of M. Therefore the new sampling period becomes $\frac{MT_s}{L}$ after down-sampling. The corresponding sampling frequency becomes $\frac{Lf_s}{M}$.

In discrete-time domain, if the original signal has a length of N, it will have a length of NL after interpolation. After decimation its length will be NL/M.

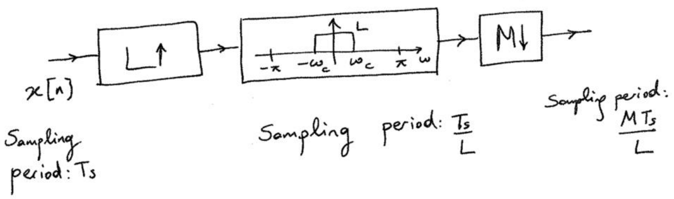


Fig. 2.17 Sampling rate change by L/M.

2.6 Interpolation Formula

Digital to analog conversion part of Shannon's sampling theorem states that, the bandlimited continuous-time signal $x_c(t)$ can be recovered exactly from its samples by perfect low-pass filtering of the signal $x_p(t) = \sum x_c(nT_s)\delta(t - nT_s)$. This leads to the well-known WhittakerShannon interpolation formula:

$$x_c(t) = \sum_{n=-\infty}^{\infty} x[n] \cdot \text{sinc}\left(\frac{t - nT_s}{T_s}\right) \tag{2.11}$$

where $x[n] = x_c(nT_s), n = 0$, is the discrete-time signal. The interpolation formula tells us that we can determine any value of $x_c(t)$ from the samples $x[n]$ but it is not implementable because (i) it is an infinite sum and (ii) the sinc function is an infinite-extent function. Therefore WhittakerShannon interpolation formula is not practical at all!

2.7 Downsampler and Upsampler are Linear Operators

The last remark that I have before closing this chapter is that both the down-sampler and the upsampler are linear operators but they are both time-varying operators.

2.8 Computer Project

1. Download the *shirt.jpg*(Fig. **??**) and *shirt-small.jpg*(Fig. **??**) images and load them into Matlab by using the following function:

```
     imagename = imread('filename.jpg');
```

Comment on the sizes of image matrices. What are the width and height of the images? What do these numbers correspond to?

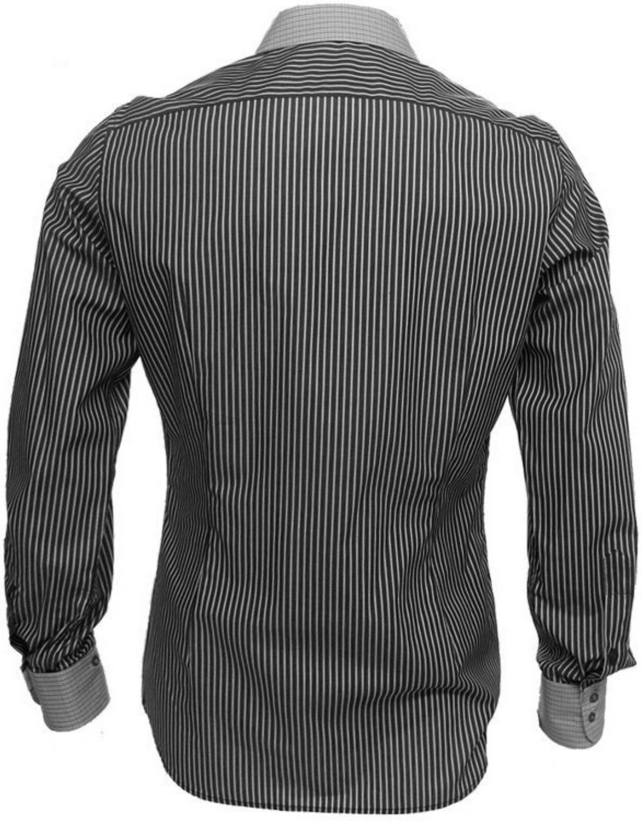


Fig. 2.18 Image of a shirt

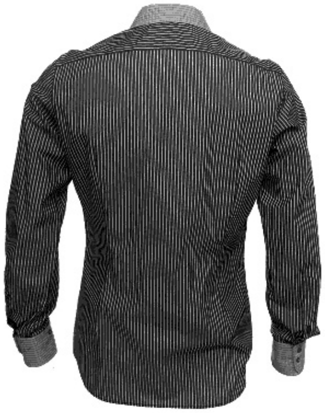


Fig. 2.19 Small image of the shirt

2. On these images, find the color values of the pixels at locations $(x = 230, y = 230)$ and $(x = 148, y = 373)$.
3. Write the *shirt.jpg* image from Matlab into your hard drive. You can use `imwrite` function to do this. Use bitmap file format by the following code:

```
    imwrite(imagename,'filename.bmp','bmp');
```

To find more info about `imwrite` function, you can type `help imwrite` in Matlab.
4. Compare the file sizes of *shirt.bmp* (from step 3) and *shirt.jpg* images. Which file is larger? Comment on differences.
5. View *wall.jpg*(Fig. **??**), and observe the patterns on the wall. Comment on the patterns that you observed.

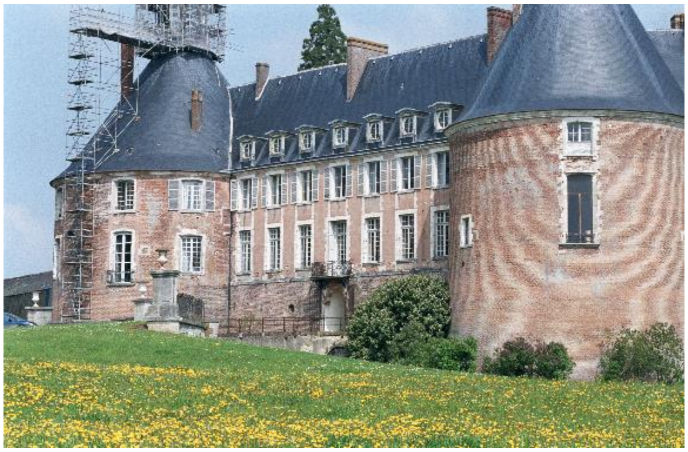


Fig. 2.20 Wall Image

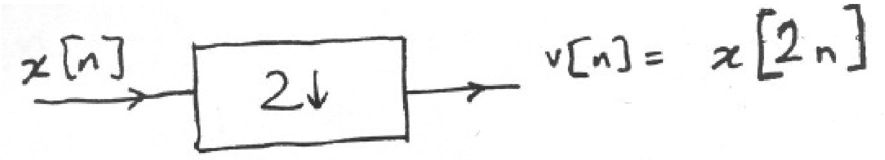
6. What is decimation? Explain in one or two sentences.
7. Decimate the *shirt.jpg* image horizontally by using the following filter: [0.250.50.25]. In order to do this first apply the filter to the rows and then down-sample the columns by 2. Comment on the size of the resulting image matrix.
8. Decimate the *shirt.jpg* image vertically by using the following filter: [0.25; 0.5; 0.25]. Comment on the size of the resulting image matrix.
9. Now, first decimate the *shirt.jpg* horizontally and then decimate the resulting image vertically. What are the width and height values of the final image? Also observe the final image and compare it with *shirt-small.jpg*. Comment on the differences.
10. Are down-sampling and up-sampling Linear Time Invariant (LTI) processes? Prove your answers.

2.9 Exercises

1. Given the following input/output relation:

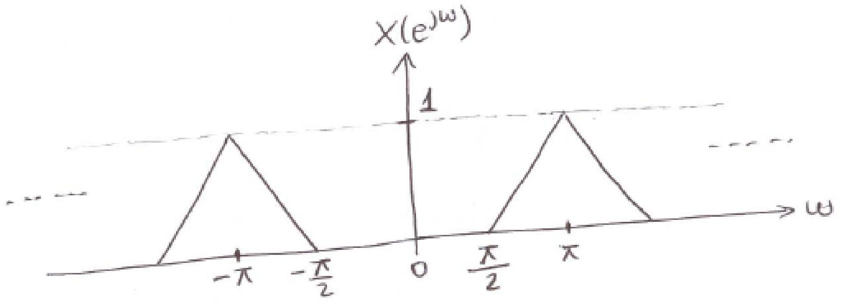
$$y[n] = \frac{1}{2}x[n] + \frac{1}{4}x[n-1]$$

(a) Is this system linear? Prove your answer.
(b) Is this system time-invariant? Prove your answer.
(c) Find the impulse response.
(d) Consider the 'downsampler by 2'. Is it linear? Prove your answer.



(e) Is down-sampler time-invariant?
(f) Let $x[n] = \delta[n]$. What is $v[n]$?
2. Given $X(e^{j\omega}) = \mathscr{F}\{x[n]\}$ (a) $y[n] = x[2n]$. Plot $Y(e^{j\omega})$.
(b) Is it possible to retrieve $x[n]$ from $y[n]$? Explain your answer.

3. Let $x[n] = \left\{ \ldots, 1, \underbrace{1}_{n=0}, 1, 2, 2, 2, 1 \right\}$ and given the low-pass filter $h[n] = \left\{ \frac{1}{4}, \underbrace{\frac{1}{2}}_{n=0}, \frac{1}{4} \right\}$.
(a) Decimate $x[n]$ by a factor of 2 using the above low-pass filter.
(b) Interpolate $x[n]$ by a factor of 2 using the same low-pass filter.
(c) Plot the frequency response of $h[n]$.
4. Draw the block diagram of the system which rescales $x[n]$ by $\alpha = \frac{3}{5}$.
5. Let $x_d[n]$ be the downsampled version of $x[n]$ (see Fig. **??**).



$$X_d(e^{jw}) = \sum_{n=-\infty}^{\infty} x_d[n]e^{-jwn}$$

where $x_d[n] = x_d[2n]$. Define $p[n] = \frac{1}{2}(1+(-1)^n)$. Use $p[n]$ to establish the relation between $X(e^{jw})$ and $X_d(e^{jw})$

Chapter 3
Discrete Fourier Transform (DFT)

3.1 DFT Definition

The Discrete Fourier Transform (DFT) of a finite extent signal $x[n]$ is defined as follows

$$X[k] \triangleq \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn}, \quad k = 0, 1, \ldots, N-1 \tag{3.1}$$

where N is called the size of the DFT. The DFT $X[k]$ of $x[n]$ is a sequence of complex numbers. That is why we use square brackets in (3.1). Obviously, $x[n]$ is assumed to be zero outside the input data window $n = 0, 1, \ldots, N-1$ in Eq. (3.1). Signal samples can be computed from the DFT coeficients using a similar equation as follows:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}kn}, \quad n = 0, 1, \ldots, N-1 \tag{3.2}$$

which is called the Inverse DFT (IDFT). We will prove the IDFT equation later.

Here are some remarks about DFT:

- It is a finite sum, therefore it can be computed using an ordinary computer.
- The DFT $X[k]$ is computed at discrete indices $k = 0, 1, \ldots, N-1$ unlike the DTFT $X(e^{j\omega})$ which has to be computed for all real ω between 0 and 2π.
- The DFT computation is basically a matrix-vector multiplication. We will discuss the computational cost of DFT computation in the next chapter.

Let $x[n]$ be a finite-extent signal, i.e., $x[n] = 0$ for $n < 0$ and $n \geq L (\leq N-1)$ then

$$\begin{aligned} \text{DTFT:} \quad X(e^{j\omega}) &= \sum_{n=0}^{L-1} x[n] e^{-j\omega n}, && \omega \text{ is a cont. variable} \\ \text{DFT:} \quad X[k] &= \sum_{n=0}^{L-1} x[n] e^{-j\frac{2\pi}{N}kn}, && k = 0, 1, \ldots, N-1 \end{aligned}$$

Therefore the relation between the DTFT and DFT for a finite extent signal is given by the following relation

$$X[k] = \quad X(e^{j\omega})\big|_{\omega=\frac{2\pi}{N}k}\,, \qquad k = 0,1,\ldots,N-1 \tag{3.3}$$

In other words, the DFT contains the samples of DTFT computed at angular frequency values $\omega = \frac{2\pi}{N}k$ for $k = 0,1,\ldots,N-1$, when $x[n]$ is a finite-extent signal as shown in Figure **??**.

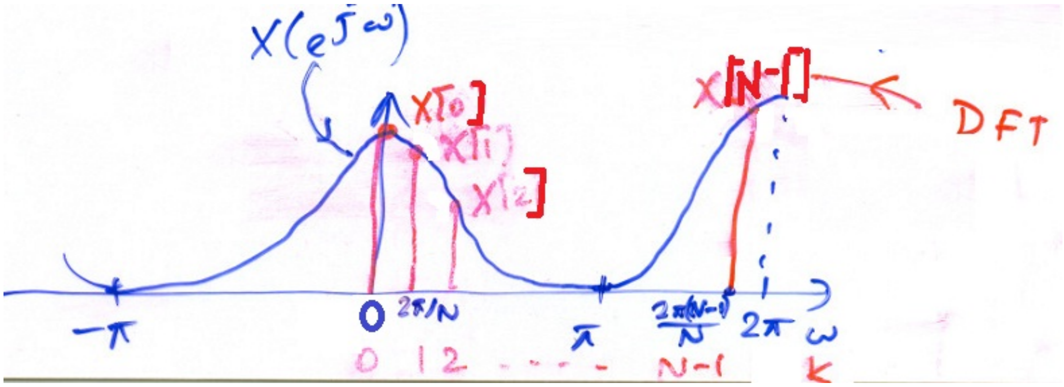


Fig. 3.1 Relation between the DFT and the DTFT of a finite extent signal $x[n]$

Theorem 1: The DFT has a period of N (when we compute DFT outside the range $k = 0,1,\ldots,N-1$).

This is because the DTFT is 2π periodic and we sample the DTFT at N locations. You can also use the definition of DFT to prove the above theorem as follows:

$$\begin{aligned}
X[N] &= \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}Nn} && = \sum_{n=0}^{N-1} x[n] \\
X[0] &= \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}0n} && = \sum_{n=0}^{N-1} x[n] \\
X[N+l] &= \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}(N+l)n} && = \sum_{n=0}^{N-1} x[n]\overbrace{e^{-j\frac{2\pi}{N}Nn}}^{=1}e^{-j\frac{2\pi}{N}ln} \\
&= X[l] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}ln}
\end{aligned}$$

The conjugate symmetry property of DFT is described in the following theorem: Theorem: For real signals $X[k] = X^*[-k]$ and $X[N-l] = X^*[l]$.

A straightforward implication of the above result in terms of magnitudes and phases of the DFT coefficients are given as follows:

$$X[N-l] = X^*[l] \Rightarrow |X[N-l]| = |X[l]| \text{ and } \sphericalangle X[N-l] = \sphericalangle X[l]$$

This is because $X(e^{j\omega})$ is 2π periodic and $X(e^{j\omega}) = X^*(e^{-j\omega})$.

Another important implication of the conjugate symmetry property is that the DFT coefficients $X[1], X[2]$ to $X[N/2]$ (even N) determines the remaining set of DFT coefficients for real $x[n]$.

3.2 Approximate Computation of CTFT using DFT

We can use the DFT to approximately compute the CTFT of a continuous time signal. To establish the relation between the CTFT and the DFT let us review the sampling theorem once again. Let us assume that $x_c(t)$ is a band-limited signal with bandwith Wc as shown in Fig. 3.2. We sample the signal with $\Omega_s > 2\Omega_c$ and obtain $X_p(j\Omega)$ which is the CTFT of $x_p(t)$. The signal $x_p(t)$ is defined in Chapter 1. The CTFT $X_p(j\Omega)$ is shown in Fig. 3.3.

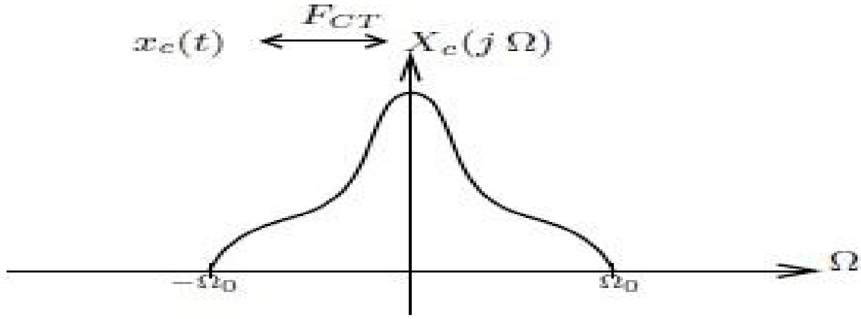


Fig. 3.2 The CTFT of the bandlimited signal $x_c(t)$.

The DTFT $X(e^{j\omega})$ discrete time signal $x[n] = x_c(nT_s), n = 0, \pm1, \pm2, ...$ is shown in Fig. 3.4. The CTFT $X_p(j\Omega)$ is equivalent to $X(e^{j\omega})$ except that the horizontal axis is normalized according to $\omega = \Omega T_s$. The signal $x[n]$ is an infinite extent signal because all band-limited signals are infinite extent signals. However, they decay to zero as n tends to infinity or negative infinity. Therefore we can select an appropriate finite window of data such that $x[n]$ is approximately 0 for $n \geq N$ and$n < 0$ (we may shift the data to fit into the range $n = 0, 1, ..., N-1$). After this truncation we can compute the N-point DFT of $x[n]$ and assume that

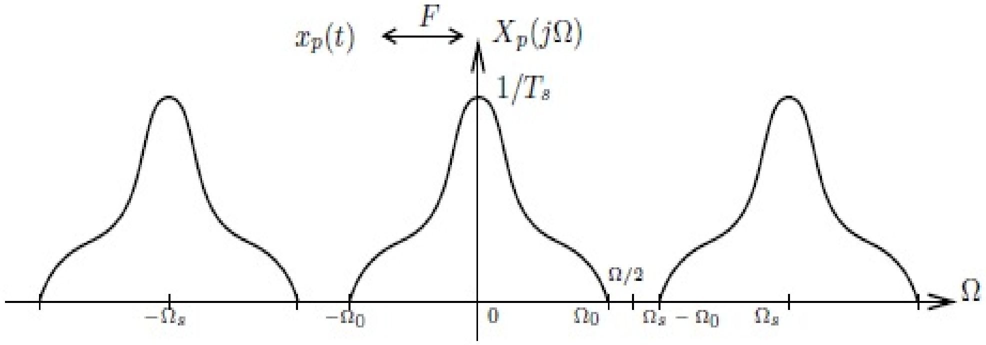


Fig. 3.3 The CTFT of the signal $x_p(t)$.

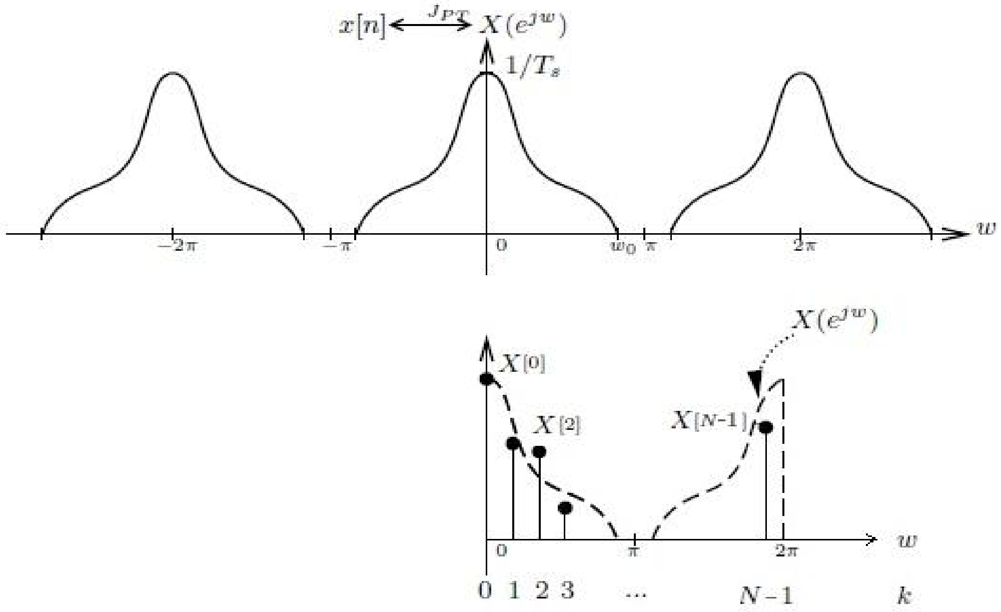


Fig. 3.4 The DTFT of the signal $x[n]$ (top plot) and the corresponding DFT coefficients $X[k]$ (bottom plot- this plot will be corrected. please see Fig 3.1 for the correct plot).

$$X[k] \cong \quad X(e^{j\omega})\big|_{\omega=\frac{2\pi}{N}k}\,, \qquad k=0,1,\ldots,N-1 \tag{3.4}$$

as shown in Figure 3.4 (bottom plot).

Since $X(e^{j\omega})$ samples are related with CTFT samples, you can approximately compute CTFT samples as well! For example, $X[N/2]$, (N even) corresponds to

$X(e^{j\pi})$ which, in turn, corresponds to $X(j\Omega_s/2)$, i.e.,

$$X(j\Omega_s/2) \cong T_s X[N/2]$$

and in general

$$X[k] \cong \frac{1}{T_s} X_c(j\frac{2\pi k}{NT_s}), \quad for\, k = 0,1,2,...,N/2 \tag{3.5}$$

Therefore it is possible to compute the CTFT using the DFT. Since there are computationally efficient algorithms for computing the DFT the Fourier analysis is an important tool for signal analysis.

It is also possible to use the Rieman sum to approximate the Fourier integral but this will not lead to a different result. Rieman sum simply becomes the DFT after some algebraic manipulations.

Example: DFT of a sinuoidial signal:

Let us assume that the sinusoid $x_c(t) = \cos(2\pi 2000t), \quad -\infty < t < \infty$ is sampled with sampling frequency $f_s = 8$ KHz. The CTFT of this sinusoid consists of two impulses

$$X_c(j\Omega) = \pi(\delta(\Omega - \Omega_o) + \delta(\Omega + \Omega_o))$$

where $\Omega_o = 2\pi 2000$.

In practice we can neither compute impulses using a computer nor we can generate a sinusoid from $-\infty$ to ∞. In practice, we observe or create a finite duration sinusoid $\tilde{x}_c(t) = x_c(t)w(t)$ where

$$w(t) = \begin{cases} 1 & 0 < t < T_o \\ 0 & otherwise \end{cases}$$

is a finite-duration time window. The CTFT $W(j\Omega)$ of a box function $w(t)$ is a sinc type waveform. Therefore the CTFT $\tilde{X}_c(j\Omega)$ of $\tilde{x}_c(t)$ is formed by convolving $X_c(j\Omega)$ and the sinc-type Fourier transform. As a result we get two sincs centered at Ω_o and $-\Omega_o$. Therefore we can assume that $\tilde{X}_c(j\Omega)$ is more or less a bandlimited Fourier Transform because sincs decay to zero as Ω tends to infinity and minus infinity. Therefore, we can approximately estimate the CTFT of $\tilde{X}_c(j\Omega)$ using the samples

$$x[n] = \tilde{x}_c(nT_s), \quad n = 0,1,\ldots,N-1 \qquad NT_s \approx T_o.$$

for a large N such as $N = 1024$. The DTFT of $x[n]$ should exhibit two peaks at $\omega_o = \Omega_o T_s = 2\pi 2000 \frac{1}{8000} = \frac{\pi}{2}$ and $-\omega_o$. As a result we should observe a peak at $k = \frac{N}{4}$ in N point DFT $X[k]$. From the location of the peak $\frac{N}{4}$ in DFT domain, we can determine the actual frequency (2KHz) of the sinusoid. We should also observe another peak at $N - N/4$ due to the conjugate symmetry property of the DFT.

Here is a table establishing the relation between the DFT index k, and the normalized frequency ω and the actual frequency Ω:

Ω	0	$2\pi*2000$	$2\pi*4000$	$-2\pi*2000$	$-2\pi*4000$
ω	0	$\pi/2$	π	$3\pi/2$	π
k	0	$N/4$	$N/2$	$3N/4$	$N/2$

Here is another example:

In Figure **??** the N=64 point DFT of $x[n] = cos(0.2\pi n), \quad n = 0,1,...,N-1$ is shown.

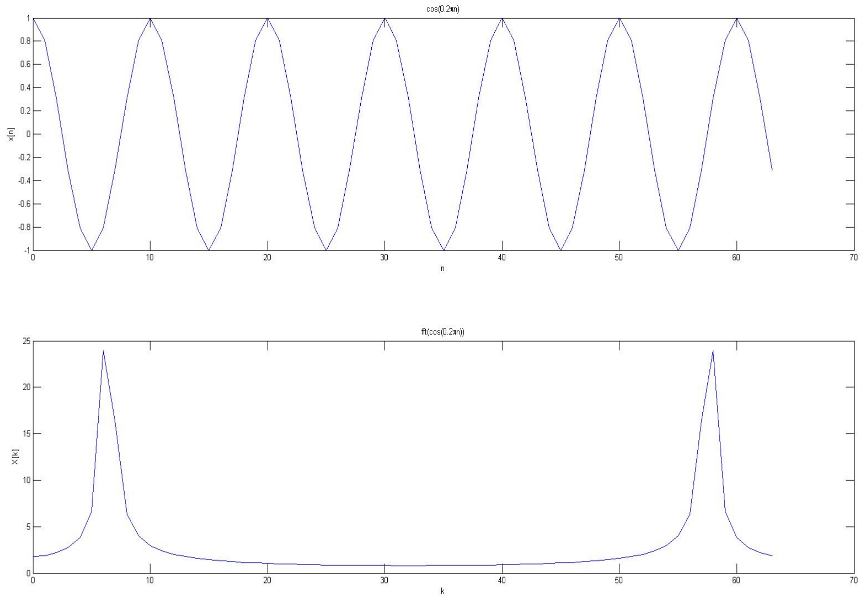


Fig. 3.5 The magnitude plot of 64 point DFT $X[k]$ of the signal $x[n] = cos(0.2\pi n)$ (bottom plot). Samples of the sinusoid are plotted in the top plot.

3.2.1 Computer Project: DTMF (Dual/Dial-Tone-Multi-Frequency)

When we dial a phone number we generate a dual-tone sound consisting of two sinusoids. For example when we press ”5” we produce

$$[\mathbf{x_5(t)}] = \cos(\Omega_b t) + \cos(\Omega_2 t), \quad 0 < t \leq T_o$$

The following frequency values are used to generate the DTMF signals.

	$\cos(\Omega_1 t)$	$\cos(\Omega_2 t)$	$\cos(\Omega_3 t)$
$\cos(\Omega_a t)$	1	2	3
$\cos(\Omega_b t)$	4	5	6
$\cos(\Omega_c t)$	7	8	9
$\cos(\Omega_d t)$	$*$	0	#

where $f_1 = 1209Hz, f_2 = 1336Hz, f_3 = 1477Hz$, and $f_4 = 1633Hz$, and $f_a = 697Hz, f_b = 770Hz, f_c = 852Hz$, and $f_d = 941Hz$. Since the speech is sampled at 8KHz all of the frequencies of sinusoids are between 0 and 4 KHz, i.e.,

$$0 < \Omega_1, \Omega_2, \Omega_3, \Omega_a, \Omega_b, \Omega_c, \Omega_d < 2\pi 4KHz$$

and the corresponding normalized angular frequency values are:

$$\omega_b = \Omega_b \cdot T_s \text{ and } \omega_2 = \Omega_2 \cdot T_s$$

where $T_s = 1/8000sec$.

Therefore, when you take the N point DFT (let N=1024) you observe two significant peaks between $k = 0$ and $k = N/2$ in the DFT spectrum plot. Let the peaks be k_1^* and k_2^*, respectively. From k_1^* and k_2^* it is possible to estimate Ω values and determine the number dialed!

To determine a regular phone number, you have to compute the DFT in short-time windows. DFT based approach is not the only approach to determine DTMF frequencies. There are other algorithms to determine DTMF tones.

3.3 Convolution using DFT

Given two discrete-time signals

$$x_1[n], \quad n = 0, 1, \ldots, M-1,$$

and

$$x_2[n], \quad n = 0, 1, \ldots, L-1.$$

Let $x[n]$ be their convolution:

$$x[n] = x_1[n] * x_2[n], \quad n = 0, 1, \ldots, N-1;$$

where $N = M + L - 1$. The length of the convolved signal is longer than the lengths of $x_1[n]$ and $x_2[n]$. Let $X[k]$ be the $N = M + L - 1$ point DFT of $x[n]$. In this case,

$$X[k] = X_1[k] \cdot X_2[k], \quad k = 0, 1, \ldots, N-1. \tag{3.6}$$

where $X_1[k]$ and $X_2[k]$ are N-point DFT's of $x_1[n]$ and $x_2[n]$, respectively. The above relation given in (**??**) is also valid when $N \geq M + L - 1$.

Let us define the signal $x_p[n]$ using the IDFT relation as follows:

$$x_p[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi}{N}kn} \tag{3.7}$$

Since any signal computed using the DFT equation or IDFT equation is periodic with period N^1. the signal $x_p[n]$ is a periodic signal with period N and

$$x_p[n] = x[n], \;\; n = 0,1,2,...,N-1 \tag{3.8}$$

In other words, the signal $x_p[n]$ is a periodic extension of the convolution result $x[n]$. This is because it is defined using the IDFT equation (**??**).

In general, inner product (dot product) of two DFT vectors corresponds to circular convolution of the corresponding signals in time domain. This subject is covered in the next section.

3.4 Circular Convolution

Let us now discuss what happens when we use an arbitrary DFT size, say K.

$$x_1[n] \overset{K-DFT}{\longleftrightarrow} \bar{X}_1[k]$$
$$x_2[n] \overset{K-DFT}{\longleftrightarrow} \bar{X}_2[k]$$

Clearly, $K-$point DFTs $\bar{X}_1[k]$ and $\bar{X}_2[k]$ are different from $N-$point DFTs $X_1[k]$ and $X_2[k]$, respectively. Let us define

$$X_3[k] = \bar{X}_1[k]\cdot\bar{X}_2[k], \quad k = 0,1,\ldots,K-1\,. \tag{3.9}$$

Inverse DFT of X_3 produces

$$x_3[n] = x_1[n]Ⓚx_2[n] \;\; ,n = 0,1,...,K-1$$

which is the $K-$point circular convolution of x_1 and x_2.

$$x_3[n] = \sum_{l=0}^{K-1} x_1[l]x_2[(n-l)_K], \; n = 0,1,...,K-1 \tag{3.10}$$

where $(n-l)_K$ represents the value of $(n-l)$ modulo K. Therefore we restrict the set of indices to $n = 0,1,...,K-1$. Outside this index range we can only get the periodic extension of $x_3[n]$.

Let us present the proof of circular convolution theorem (**??**)-(**??**). Let $x_3[n] = x_1[n]Ⓚx_2[n]$ be defined as follows:

[1] The proof of this statement is very similar to the proof of Theorem 1.

$$x_3[n] = \sum_{l=0}^{K-1} x_1[l]x_2[(n-l)_K]\,.$$

Let us compute the K-point DFT of $x_3[n]$ as follows:

$$X_3[k] = \sum_{n=0}^{K-1}\left(\sum_{l=0}^{K-1} x_1[l]x_2[(n-l)_K]\right)e^{-j\frac{2\pi}{K}kn}\,, \quad k = 0,1,\ldots,K-1\,.$$

We change the order of summations and obtain:

$$\begin{aligned} X_3[k] &= \sum_{l=0}^{K-1} x_1[l]\sum_{n=0}^{K-1} x_2[(n-l)_K]e^{-j\frac{2\pi}{K}kn}\,, \quad k = 0,1,\ldots,K-1 \\ X_3[k] &= \sum_{l=0}^{K-1} x_1[l]\sum_{m=0}^{K-1} x_2[m_K]e^{-j\frac{2\pi}{K}k(m+l)}\,. \end{aligned}$$

We can take $e^{-j\frac{2\pi kl}{K}}$ outside the inner sum and obtain:

$$\begin{aligned} X_3[k] &= \underbrace{\sum_{l=0}^{K-1} x_1[l]e^{-j\frac{2\pi}{K}kl}}_{}\underbrace{\sum_{m=0}^{K-1} x_2[m]e^{-j\frac{2\pi}{K}km}}_{} \\ X_3[k] &= \quad \bar{X}_1[k] \qquad \cdot \qquad \bar{X}_2[k]\,, \quad k = 0,1,\ldots,K-1 \end{aligned}$$

which proves the statements in (**??**)-(**??**).

When $K < M+L-1$, we cannot use the DFT to compute the regular convolution of x_1 and x_2 but we can compute the circular convolution of x_1 and x_2. In general, we should try to avoid the circular convolution because some of the samples of the circular convolution turn out to be corrupted. Circular convolution produces the same coefficients of regular convolution when $K \geq M+L-1$. Circular convolution is useful when we filter streaming data in DFT domain (we will discuss this in Chapter 5).

Let us consider the following example. Given two sequences $x[n]$ and $h[n]$:

$$x[n] = \begin{cases} 1, & n = 0,1 \\ 0, & otherwise \end{cases}$$

$$h[n] = \begin{cases} 0.9^n, & n = 0,1,\ldots,4 \\ 0, & otherwise \end{cases}$$

In this case, the regular convolution $y[n] = x[n] * h[n]$ has a length of $6 = 5+2-1$. Let us also compute the $6-$point circular convolution of $x[n]$ and $h[n]$

$x_3[n] = x[n] \textcircled{6} h[n]$

For $n=0$, $\quad x_3[0] = \sum_{n=0}^{5} h[n]x[(0-n)_6] = h[0]x[(0)_6] = 0.9^0 = 1 = y[0]$,

for $n=1$, $\quad x_3[1] = \sum_{n=0}^{5} h[n]x[(1-n)_6] = h[0]x[(1)_6] + h[1]x[(0)_6] = 1 + 0.9 = y[1]$,

for $n=2$, $\quad x_3[2] = \sum_{n=0}^{5} h[n]x[(2-n)_6] = h[1]x[(1)_6] + h[2]x[(0)_6] = 0.9 + 0.9^2 = y[2]$,

for $n=3$, $\quad x_3[3] = \sum_{n=0}^{5} h[n]x[(3-n)_6] = h[2]x[(1)_6] + h[3]x[(0)_6] = 0.9^2 + 0.9^3 = y[3]$,

for $n=4$, $\quad x_3[4] = 0.9^4 + 0.9^3 = y[4]$,

for $n=5$, $\quad x_3[5] = \sum_{n=0}^{5} h[n]x[(5-n)_6] = h[4]x[(1)_6] = 0.9^4 = y[5]$,

and for $n=6$, $\quad x_3[(6)_6] = x_3[0]$.

Therefore $x_3[n] = y[n]$, for $n = 0, 1, \ldots, 5$.

For $M = 5$ we have a problem. Let $x_2[n]$ be the 5−point circular convolution of $x[n]$ and $h[n]$:

$$x_2[n] = x[n] \textcircled{5} h[n].$$

Let us compute $x_2[0]$:

$$\begin{aligned} \text{For } n=0, \quad x_2[0] &= \sum_{n=0}^{4} h[n]x[(0-n)_5] = h[0]x[(0)_5] + h[4]x[(-4)_5] \\ &= h[0]x[(0)_5] + h[4]x[1] = 1 + 0.9^4 \end{aligned}$$

which is not equal to $y[0]$. However,

$$\begin{aligned} &\text{for } n=1, \quad x_2[1] = y[1] \\ &\text{for } n=2, \quad x_2[2] = y[2] \\ &\text{for } n=3, \quad x_2[3] = y[3] \\ &\text{and for } n=4, \quad x_2[4] = y[4]. \end{aligned}$$

It turns out that:

$$x_2[0] = y[0] + y[5] \longleftarrow \text{ corrupting term}$$

Since there is no room for $y[5]$, it turned around the modulo circle and settled on $y[0]$.

Let $y[n] = x_1[n] * x_2[n]$ and $x_3[n] = x_1[n] \textcircled{M} x_2[n]$. The circular convolution results in $x_3[n]$, that is periodic with period M, and it is related with $y[n]$ as follows:

$$x_3[n] = \sum_{l=-\infty}^{\infty} y[n-lM] = y[n] + y[n-M] + \dots$$
$$+ y[n+M] + \dots \tag{3.11}$$

$y[n]$ and its shifted versions are overlapped and added to obtain $x_3[n]$. This obviously corrupts some samples of x_3 when M is shorter than the length of $y[n]$.

3.4.1 Computation of DFT of Anticausal Sequences

Equations (3.1) and (3.2) assume that the discrete-time signal $x[n]$ are causal sequences. Let us consider the following example.
Example: Compute the DFT of a two-sided signal $x[n] = \{e, d, \underbrace{a}_{n=0}, b, c\}$.

There are two ways to compute the DFT.

- Shift this signal $\bar{x}[n] = x[n-2]$ and compute the N-point DFT of $\bar{x}[n]$ and use the relation $\bar{X}(e^{j\omega}) = X(e^{j\omega})e^{-j2\omega}$ to determine $X[k]$, therefore

$$\bar{X}[k] = X[k]e^{-j2(2\pi k)/N}, k = 0, 1, ..., N-1. \tag{3.12}$$

  Therefore, we can compute the DFT of the causal sequence and use the above equation to determine the N-point DFT of $x[n]$:

$$X[k] = \bar{X}[k])e^{jm(2\pi k)/N}, k = 0, 1, ..., N-1. \tag{3.13}$$

  where m is the amount of time shift (which is $m = 2$ in this example).
- There is a second way. This is based on the periodic nature of DFT and IDFT. Assume a periodic extension of $x[n]$ as follows:

$$x_p[n] = \sum_{l=-\infty}^{\infty} x[n-lN], \qquad N \geq 5$$
$$= x[n] + x[n-5] + x[n+5] + \dots$$

where it is assumed that $N = 5$. The first period of $x_p[n]$ is given as follows:

$$x_p[n] = \{\underbrace{a}_{n=0}, b, c, e, d\}, \text{ for } n = 0, 1, \dots, 4.$$

After this step, we can compute the $N-$point DFT:

$$\bar{x}[n] \overset{N-DFT}{\longleftrightarrow} \bar{X}[k]$$
$$x_p[n] \overset{N-DFT}{\longleftrightarrow} \bar{X}_p[k]$$

Magnitudes of $\bar{X}[k]$ and $\bar{X}_p[k]$ are equal to each other, i.e., $|X_p[k]| = |\bar{X}[k]|$. Only a linear phase difference term exists between the two DFTs. This subject is covered in the following property of DFT.

Periodic Shift Property of DFT:

$$x[n] \overset{N-DFT}{\longleftrightarrow} X[k]$$
$$x[(n-m)_N] \overset{N-DFT}{\longleftrightarrow} X[k]e^{-j\frac{2\pi}{N}km}$$

An ordinary time shift of $x[n]$ may produce non-zero terms after the index N. Therefore, we need the modulo operation $(n-m)_M$ to keep all the coefficients of $x[(n-m)_M]$ in the range of $n = 0,1,\ldots,N-1$.

Linearity Property:

For all $\alpha, \beta \in \mathbb{R}$; and signals x_1 and x_2, we have

$$x_1[n] \overset{N-DFT}{\longleftrightarrow} X_1[k]$$
$$x_2[n] \overset{N-DFT}{\longleftrightarrow} X_2[k]$$
$$\alpha x_1[n] + \beta x_2[n] \overset{N-DFT}{\longleftrightarrow} \alpha X_1[k] + \beta X_2[k], \quad k = 0,1,\ldots,N-1$$

Therefore, the DFT is a linear transform. Notice that you cannot linearly combine an N-point DFT with and L-point DFT.

Example: Compute the $N-$point DFT of the following signal $x[n]$:

$$x[n] = \begin{cases} 1, & n = m \\ 0, & \text{otherwise} \end{cases}$$

We compute the DFT coefficients one by one:

$$X[0] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} x[n] = x[m] = 1$$
$$X[1] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn} = x[m]e^{-j\frac{2\pi}{N}1m} = e^{-j\frac{2\pi}{N}m} = W_N^m,$$

where $W_N = e^{-j\frac{2\pi}{N}}$. Similarly,

$$X[2] = W_N^{2m} = e^{-j\frac{2\pi}{N}2m}, \ldots$$
$$\text{and } X[N-1] = W_N^{(N-1)m}.$$

Therefore,

$$X[k] = W_N^{km} = e^{-j\frac{2\pi}{N}km}, \quad k = 0,1,\ldots,N-1 \qquad (3.14)$$

Let us compute the IDFT of the DFT coefficients given in (**??**):

Inverse DFT produces an interesting identity:

$$x[n] = \begin{cases} 1, & n = m \\ 0, & \text{otherwise} \end{cases} = \frac{1}{N}\sum_{k=0}^{N-1} e^{-j\frac{2\pi}{N}km} e^{-j\frac{2\pi}{N}kn} = \delta(n-m)$$

$$\sum_{k=0}^{N-1} e^{-j\frac{2\pi}{N}(n-m)k} = \begin{cases} N, & \text{for } n-m = 0, \pm N, \pm 2N, \ldots \\ 0, & \text{otherwise} \end{cases}$$

Periodic extension is due to the fact that the IDFT expression also produces a periodically extended signal when n is computed outside the range $0 \le n \le N-1$.
Example: Compute the DFT of

$$x[n] = \cos\left(\frac{2\pi rn}{N}\right), \qquad 0 \le n \le N-1, \quad r = 0, 1, \ldots, N-1\,.$$

We can express $x[n]$ in the following form using Euler's formula:

$$x[n] = \frac{1}{2}\left(W_N^{-rn} + W_N^{rn}\right)$$

Let us compute the DFT of each term separately.

$$X[k] = \frac{1}{2}\sum_{n=0}^{N-1} W_N^{(r-k)n} + \frac{1}{2}\sum_{n=0}^{N-1} W_N^{(r+k)n}$$

We can now use the previous example and get

$$X[k] = \begin{cases} N/2, & k = r \\ N/2, & k = N-r \\ 0, & \text{otherwise} \end{cases}$$

3.5 Inverse DFT of an Infinite Extent Signal

Let $x[n]$ be an arbitrary signal with DTFT $X(e^{j\omega})$. We sample $X(e^{j\omega})$ in the Fourier domain at N locations as follows:

$$X[k] = X(e^{j\omega})\big|_{\omega = W_N^k}\,, \quad k = 0, 1, \ldots, N-1\,.$$

Since $x[n]$ can be an infinite extent signal, we may have an infinite sum as shown below:

$$X[k] = X(e^{j\frac{2\pi}{N}k}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\frac{2\pi}{N}kn}$$

We can divide the infinite sum into finite sums:

$$\begin{aligned} X[k] &= \cdots + \sum_{n=N}^{2N-1} x[n]e^{-j\frac{2\pi}{N}kn} + \sum_{n=-N}^{-1} x[n]e^{-j\frac{2\pi}{N}kn} + \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn} + \cdots \\ &= \sum_{l=-\infty}^{\infty} \sum_{n=lN}^{lN+N-1} x[n]e^{-j\frac{2\pi}{N}kn} \end{aligned}$$

Define a new variable as $n = m + lN$

$$\begin{aligned} X[k] &= \sum_{l=-\infty}^{\infty} \sum_{m=0}^{N-1} x[m+lN]e^{-j\frac{2\pi}{N}km} \\ &= \sum_{m=0}^{N-1} \left(\sum_{l=-\infty}^{\infty} x[m+lN] \right) e^{-j\frac{2\pi}{N}km}, \quad k = 0,1,\dots,N-1\,. \end{aligned}$$

The last equation is the DFT of $\sum_{l=-\infty}^{\infty} x[m+lN]$. Let us define the signal $x_p[m]$ based on this signal as follows

$$x_p[m] = \sum_{l=-\infty}^{\infty} x[m+lN] \tag{3.15}$$

The signal $x_p[m]$ is the overlapped and added versions of $x[m], x[m+N], x[m-N], x[m+2N], \dots$. Therefore, $x_p[m]$ is some sort of time-aliased version of $x[n]$. It is also periodic with period N when extended outside the index range $n = 0,1,\dots,N-1$. That is because any sequence obtained using the IDFT relation has to be periodic with period N.

Property of IDFT: The signal $x_p[n]$ defined using the IDFT relation

$$x_p[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi}{N}kn}$$

is periodic with period N.

Consider

$$\begin{aligned} x_p[n+N] &= \frac{1}{N} \sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi}{N}k(n+N)} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi}{N}kn} \end{aligned}$$

because $e^{j\frac{2\pi N}{N}k} = 1$ for all integer k. Similarly, $x_p[n] = x_p[n+lN]$ for all integer l.

Example: (a) Compute the 2−point DFT of $x = \{1,1\}$.

$$\begin{aligned} X[0] &= 1 + 1 = 2 \\ X[1] &= 1 + 1e^{-j\frac{2\pi}{2}1\cdot 1} = 0 \end{aligned}$$

(b) Compute the 3−point DFT of $x = \{1,1\}$.

$$
\begin{aligned}
X_3[0] &= 1+1 = 2\\
X_3[1] &= 1+1e^{-j\frac{2\pi}{3}1\cdot 1} = 1+e^{-j2\pi/3}\\
X_3[2] &= 1+1e^{-j\frac{2\pi}{3}1\cdot 2} = 1+e^{-j4\pi/3}
\end{aligned}
$$

As you see $X[k] \neq X_3[k]$ for all k except $k=0$. This is because

$$X[k] = X(e^{j\omega})\big|_{\omega=\frac{2\pi k}{2}}\ ,\ k = 0,1$$

and

$$X_3[k] = X(e^{j\omega})\big|_{\omega=\frac{2\pi k}{3}}\ ,\ k = 0,1,2$$

Although, $X[k]$ and $X_3[k]$ are samples of $X(e^{j\omega})$ they can be different from each other because they sample $X(e^{j\omega})$ at different frequency locations.

3.6 DFT and Inverse DFT using Matrix Notation

The N-point DFT can be expressed as a matrix vector multiplication as follows:

$$
\begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix}
=
\begin{bmatrix}
1 & 1 & \cdots & 1 \\
1 & e^{-j\frac{2\pi}{N}} & \cdots & e^{-j\frac{2\pi(N-1)}{N}} \\
\vdots & \vdots & & \vdots \\
1 & e^{-j\frac{2\pi(N-1)}{N}} & \cdots & e^{-j\frac{2\pi(N-1)^2}{N}}
\end{bmatrix}
\begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}
$$

The N by N transform matrix $\mathbf{W_N}$

$$
\mathbf{W_N} =
\begin{bmatrix}
1 & 1 & \cdots & 1 \\
1 & e^{-j\frac{2\pi}{N}} & \cdots & e^{-j\frac{2\pi(N-1)}{N}} \\
\vdots & \vdots & & \vdots \\
1 & e^{-j\frac{2\pi(N-1)}{N}} & \cdots & e^{-j\frac{2\pi(N-1)^2}{N}}
\end{bmatrix}
$$

is called the forward DFT matrix. Notice that the last row can be simplified $\left[1\ e^{-j\frac{2\pi(N-1)}{N}}\ e^{-j\frac{2\pi(N-2)}{N}}\ \cdots\ e^{-j\frac{2\pi}{N}}\right]$. It can easily be shown that DFT is a symmetric matrix.

Similar to the forward DFT, Inverse DFT can be expressed as a matrix vector multiplication:

$$
\begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ X[N-1] \end{bmatrix}
= \mathbf{W_N^{-1}}
\begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix}
$$

where $\mathbf{W_N}^{-1}$ is the inverse DFT matrix which can be expressed in terms of $\mathbf{W_N}$ as follows

$$\mathbf{W_N^{-1}} = \frac{1}{N}\mathbf{W_N^*} \tag{3.16}$$

where $*$ denotes complex conjugate transpose. This is because the DFT matrix is an orthogonal matrix. Furthermore, the DFT is a symmetric matrix therefore there is no need to take the transpose of $\mathbf{W_N}$. Since the inverse DFT matrix is the complex conjugate of the forward DFT matrix, we directly obtain the forward DFT formula from Equation (3.13) as follows:

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi k}{N}n}, \qquad n = 0, 1, \ldots, N-1 \tag{3.17}$$

Therefore, for finite extent sequences, there is no need to compute the inverse DTFT expression

$$x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})\, e^{j\omega n} d\omega \quad \leftarrow \text{inverse DTFT} \tag{3.18}$$

which is an integral.

What happens if n is outside the index set $0, 1, \ldots, N-1$

$$\begin{aligned}
x[N] &= \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi k}{N}N} = \frac{1}{N}\sum_{k=0}^{N-1} X[k] \\
x[0] &= \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi k}{N}0} = \frac{1}{N}\sum_{k=0}^{N-1} X[k] \\
x[N+1] &= \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi k}{N}(N+1)} = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi k}{N}} = x[1] \\
x[-1] &= \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi k}{N}(-1)} = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi k}{N}(N-1)} = x[N-1]
\end{aligned}$$

Periodic extension: $x_p[n]$

$$\begin{aligned}
x_p[n] &= \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi k}{N}n}, \qquad n = 0, \pm 1, \pm 2, \ldots \\
x_p[n] &= x[n], \qquad n = 0, 1, 2, \ldots, N-1 \quad \leftarrow \text{Basic period} \\
x_p[n] &= \sum_{l=-\infty}^{\infty} x[n-lN] \quad \leftarrow \text{in general}
\end{aligned}$$

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi k}{N}n}, \qquad n = 0,1,\ldots,N-1$$

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} W_N^{-kn}, \qquad n = 0,1,\ldots,N-1$$

Proof: Forward DFT is given as follows:

$$\begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{-j\frac{2\pi}{N}} & \cdots & e^{-j\frac{2\pi(N-1)}{N}} \\ \vdots & \vdots & & \vdots \\ 1 & e^{-j\frac{2\pi(N-1)}{N}} & \cdots & e^{-j\frac{2\pi(N-1)^2}{N}} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$$

The last row is $\left[1 \; e^{-j\frac{2\pi(N-1)}{N}} \; e^{-j\frac{2\pi(N-2)}{N}} \; \cdots \; e^{-j\frac{2\pi}{N}}\right]$.

- $\mathbf{W_N}$ is a symmetric matrix.
- Rows of $\mathbf{W_N}$ are orthogonal to each other.

$$\mathbf{W_N^{-1}} = \frac{1}{N}\mathbf{W_N^H} = \frac{1}{N}\left(\mathbf{W_N^*}\right)^T = \frac{1}{N}\mathbf{W_N^*}$$

Therefore;

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi k}{N}n}, \qquad n = 0,1,\ldots,N-1$$

Example:

$$x[n] = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases} \Longrightarrow X[k] = ? \qquad k = 0,1,\ldots,N-1$$

$$X[0] = \sum_{n=0}^{N-1} x[n] = 1$$

$$X[1] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi 1}{N}n} = 1.e^{-j\frac{2\pi 1}{N}0} \; + \; 0 \; + \; \cdots \; + \; 0 = 1$$

$$\vdots$$

$$X[N-1] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi(N-1)}{N}n} = 1.e^{-j\frac{2\pi(N-1)}{N}0} \; + \; 0 \; + \; \cdots \; + \; 0 = 1$$

Inverse DFT expression:

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1}\underbrace{1}_{X[k]} e^{j\frac{2\pi k}{N}n} = \begin{cases} 1 & n = 0, \pm N, \pm 2N, \ldots \\ 0 & \text{ow } (n = 1, 2, \ldots, N-1) \end{cases}$$

3.7 Parseval's Relation

Parseval's relation for DTFT:

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

Parseval's relation for DFT:

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N}\sum_{k=0}^{N-1} |X[k]|^2$$

Proof:

$$\begin{aligned}
x[n] &\overset{N-DFT}{\longleftrightarrow} X[k], \quad k = 0, 1, \ldots, N-1 \\
x^*[-n] &\overset{N-DFT}{\longleftrightarrow} X^*[k], \quad k = 0, 1, \ldots, N-1 \\
x^*[n] &\overset{N-DFT}{\longleftrightarrow} X^*[-k], \quad k = 0, 1, \ldots, N-1 \\
v[n] &= x[n] \textcircled{N} x^*[-n] \overset{N-DFT}{\longleftrightarrow} V[k] = X[k]X^*[k] = |X[k]|^2, \quad k = 0, 1, \ldots, N-1 \\
v[n] &= \sum_{l=0}^{N-1} x[l]x^*[l-n] \\
v[0] &= \sum_{l=0}^{N-1} x[l]x^*[l] = \sum_{n=0}^{N-1} |x[l]|^2 \\
v[n] &= \frac{1}{N}\sum_{k=0}^{N-1} V[k]e^{j\frac{2\pi}{N}kn} \\
v[0] &= \frac{1}{N}\sum_{k=0}^{N-1} |X[k]|^2 \cdot 1 = \sum_{l=0}^{N-1} |x[l]|^2
\end{aligned}$$

□

3.8 Mini Projects

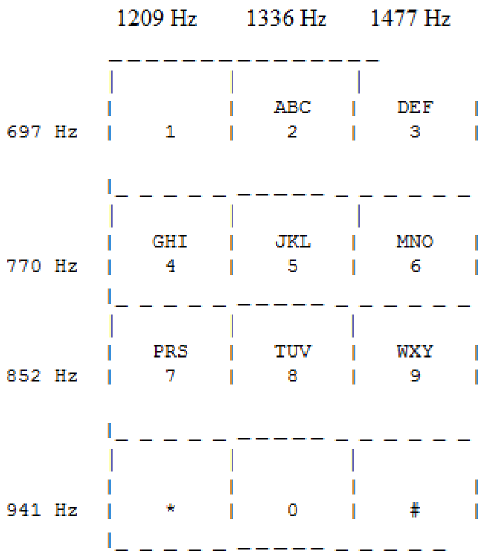
PART 1. DTMF (Dual/Dial-Tone-Multi-Frequency)

Which teaching assistant am I trying to call: *dtmf.wav*

- You can read the above file using *wavread* function in Matlab.
- You have to identify the corresponding 4−digit dialed number (XXXX).
- Show all your work and plot all necessary graphics in the report.

Summary:

A DTMF signal consists of the sum of two sinusoids - or tones - with frequencies taken from two mutually exclusive groups. Each pair of tones contains one frequency of the low group (697 Hz, 770 Hz, 852 Hz, 941 Hz) and one frequency of the high group (1209 Hz, 1336 Hz, 1477 Hz) and represents a unique symbol.



	1209 Hz	1336 Hz	1477 Hz
697 Hz	1	ABC 2	DEF 3
770 Hz	GHI 4	JKL 5	MNO 6
852 Hz	PRS 7	TUV 8	WXY 9
941 Hz	*	0	#

Example: Following is a DTMF signal in continuous time domain

$$x(t) = \sin(2\pi f_{\text{low}} t) + \sin(2\pi f_{\text{high}} t)$$

choosing $f_{\text{low}} = 697$ Hz and $f_{\text{high}} = 1209$ Hz, you'll have the dial tone for symbol $\{1\}$.

PART 2.

Which dial tone do need to use to prevent Hugo to be hit by the rock and then falling down: *hugo.jpg*

Write a short Matlab code to generate the DTMF signal. Plot the FFT of the DTMF signal that you generated. Clearly indicate the sampling frequency of the generated signal.

Note: Hugo is a superhero however he cannot jump over the rock.

3.9 Exercises

1. (a) Let $x_a(t) = \sin 2\pi 1000t + \cos 2\pi 1000t$. Plot $X_a(j\Omega)$
(b) This signal is sampled with $f_s = 8$ kHz, and $x[n] = x_a(nT_s),\ n = 0, \pm1, \pm2, \ldots$ is obtained. Plot $X(e^{j\omega})$.
(c) Assume that we have $x[n],\ n = 0, 1, \ldots, 1023$. Approximately plot $|X[k]|$ which is the 1024−point DFT of $x[n]$.

2. (a) Use $N = 5$ point DFT to compute the convolution of $x[n] = \left\{ \underbrace{1}_{n=0}, 2, 2 \right\}$ and $h[n] = \left\{ \underbrace{-\frac{1}{2}}_{n=0}, 1, -\frac{1}{2} \right\}$.
(b) Can you use $N = 4$ point DFT to compute $y[n] = h[n] * x[n]$? Explain. Find $v[n] = IDFT_4^{-1}\{H_4[k]X_4[k]\}$ where $H_4[k]$ and $X_4[k]$ are 4−point DFT's of $h[n]$ and $x[n]$, respectively.

3. Find the (a) DTFT $X(e^{j\omega})$ and (b) 32−point DFT of $x[n] = \left\{ \frac{1}{4}, \underbrace{\frac{1}{2}}_{n=0}, \frac{1}{4} \right\}$

4. Given $x_1[n] = \left\{ \underbrace{1}_{n=0}, 1, 0, 0 \right\}$ and $x_2[n] = \left\{ \underbrace{1}_{n=0}, 1, 1, 1 \right\}$
(a) Compute the 4−point DFT's of x_1 and x_2.
(b) Compute the 4−point circular convolution of x_1 and x_2 using DFT.
(c) What should be the size of DFT such that the circular convolution produces the actual convolution result?

5. Given $x[n] = \left\{ \frac{1}{4}, \frac{1}{2}, \underbrace{1}_{n=0}, \frac{1}{2}, \frac{1}{4} \right\}$
(a) Compute the 8−point DFT of $x[n]$.
(b) Find $X(e^{j\omega})$. What is the relation between $X(e^{j\omega})$ and the 8−point DFT $X[k]$?
(c) Let $Y[k] = \left[\underbrace{1}_{k=0}, 1, 1, 1, \ldots, \underbrace{1}_{k=1023} \right]$. Find $y[n]$.
(d) Let $y[n] = \left[\underbrace{1}_{n=0}, 1, 1, 1, \ldots, \underbrace{1}_{n=1023} \right]$. Find $Y[k]$.
(e) Prove that $X[k] = X^*[N-k]$ where x is real.

6. Let $x_c(t) = \cos 2\pi 450t$. This signal is sampled with $f_s = 1.5kHz$: $x[n] = x_c(nT_s),\ n = 0, \pm1, \ldots$ where $T_s = 1/f_s$. We have 512 samples of $x[n]$.
(a) Plot $X(e^{j\omega})$ of $x[n],\ n = 0, \pm1, \pm2, \ldots$
(b) Approximately plot the DFT magnitude $|X[k]|$. The DFT size is $N = 512$.

7. Calculate 10 elements of $y[n] = x_1[n] ⑤ x_2[n]$ where $x_1[n] = \underbrace{1}_{n=0}, 2$ and $x_2[n] = \underbrace{-1}_{n=0}, 0, 3$.

8. Compute the DFT of $x[n] = \left\{\frac{1}{2}, 0, \frac{1}{2}\right\}$ by first calculating the DTFT and sampling it ($N = 10$).

9. Write down the $6-$ point periodic extension $x_p[n]$ of $x[n] = \left\{\underbrace{a}_{n=0}, b\right\}$.

10. Find the $3-$point DFT of $x[n] = \left\{1, \underbrace{0}_{n=0}, 1\right\}$. Verify your result by calculating the IDFT.

11. Convolve $x_1[n] = \left\{-1, \underbrace{2}_{n=0}, 1\right\}$ and $x_2[n] = \left\{\underbrace{0}_{n=0}, 1, 3\right\}$ using DFT.

Chapter 4
Fast Fourier Transform (FFT) Algorithms

4.1 Introduction

Fast Fourier Transform (FFT) is not a transform. It is an algorithm to compute the DFT. As you will see in the next section, the FFT based implementation of DFT requires about $(N\log_2 N)$ complex multiplications compared to direct implementation which requires N^2 complex multiplications.
FFT algorithm made Fourier analysis of signals feasible because $N\log_2 N$ is much smaller than N^2 when N is large. For example, $N^2 = 1048576$, on the other hand $N\log_2 N = 1024 \times 10 = 10240$ for $N = 1024$. FFT is a recursive algorithm. It computes N-point DFT using $\frac{N}{2}$-point DFTs. $\frac{N}{2}$-point DFTs are then computed using $\frac{N}{4}$-point DFTs etc.

4.2 DFT Computation by Matrix Multiplication

As we discussed in Chapter 3 the N-point DFT:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi k}{N}n}, \qquad k = 0, 1, \ldots, N-1 \tag{4.1}$$

is essentially a matrix-vector multiplication:

$$\mathbf{X} = \mathbf{W_N x}$$

where $\mathbf{X} = [X[0]\ X[1]\ \ldots\ X[N-1]]^T$ is the vector of DFT coefficients, $\mathbf{x} = [x[0]\ x[1]\ \ldots\ x[N-1]]^T$ is the input vector and the matrix

$$\mathbf{W}_N = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{-j\frac{2\pi}{N}} & \cdots & e^{-j\frac{2\pi(N-1)}{N}} \\ \vdots & \vdots & & \vdots \\ 1 & e^{-j\frac{2\pi(N-1)}{N}} & \cdots & e^{-j\frac{2\pi(N-1)^2}{N}} \end{bmatrix}$$

The matrix $\mathbf{W_N}$ is the $N-$point DFT matrix representing the following set of computations.

$$\begin{aligned} X[0] &= \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}0n} \\ X[1] &= \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}1n} \\ &\vdots \\ X[N-1] &= \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}(N-1)n} \end{aligned} \tag{4.2}$$

Each equation in (**??**) corresponds to an inner product (or a dot product) and N complex multiplications are required to calculate each $X[k]$ value. Therefore, the total computational cost is N^2 complex multiplications to obtain the complete DFT domain data $X[0], X[1], \ldots, X[N-1]$.

Therefore direct implementation cost of DFT is N^2 complex multiplications and $N(N-1)$ complex additions.

Example: If $N = 1024 = 2^{10} \Rightarrow N^2 \approx 10^3 \times 10^3 = 10^6$ complex multiplications are required. On the other hand, decimation in frequency FFT algorithm requires $O((N/2)\log_2 N)$ complex multiplications $\approx \frac{10^3}{2}\log_2 1024 \approx \frac{10^4}{2}$ and $N\log_2 N \approx 10^4$ complex additions. Therefore, FFT algorithm is really a fast algorithm. When N is large, computational savings are really significant.

4.3 Decimation-in-Frequency FFT Computation Algorithm

This algorithm is developed by Cooley and Tukey in 1960s. It is a divide and conquer type algorithm and it requires $N = 2^p$, p is an integer. Let us express the DFT sum given in (**??**) in two parts:

$$X[k] = \sum_{n=0}^{N/2-1} x[n]W_N^{kn} \quad + \quad \sum_{n=N/2}^{N-1} x[n]W_N^{kn},$$

where $W_N = e^{-j\frac{2\pi}{N}}$. We define $n = l + N/2$ and rewrite the above equation as follows:

$$X[k] = \sum_{n=0}^{N/2-1} x[n]W_N^{kn} \quad + \quad \sum_{l=0}^{N/2-1} x\left[l+N/2\right] W_N^{kl} W_N^{kN/2},$$

where $W_N^{kN/2} = (-1)^k$ and it can go outside the second sum:

$$X[k] = \sum_{n=0}^{N/2-1} x[n]W_N^{kn} \quad + \quad (-1)^k \sum_{n=0}^{N/2-1} x\left[n+N/2\right] W_N^{kn},$$

where we replace l by n. To combine the two summations into a single sum:

$$X[k] = \sum_{n=0}^{N/2-1} \Big(x[n] + (-1)^k x\left[n+N/2\right]\Big) W_N^{kn}, \qquad k = 0,1,2,\ldots,N-1$$

The next step is to divide DFT coefficients into two groups according to their indices. This is called decimation in even and odd frequency samples. We do not throw away any samples as in Chapter 2. Therefore, it is actually downsampling $X[k]$ into even and odd samples. The first group contains even indexed DFT coefficients

$$X[2l] = \sum_{n=0}^{N/2-1} \Big(x[n] + (-1)^{2l} x\left[n+N/2\right]\Big) W_N^{2ln}, \quad l = 0,1,2,\ldots,N/2-1$$

where k is replaced by $2l$ and the second group contains odd indexed DFT coefficients

$$X[2l+1] = \sum_{n=0}^{N/2-1} \Big(x[n] + (-1)^{2l+1} x\left[n+N/2\right]\Big) W_N^{(2l+1)n}, \quad l = 0,1,2,\ldots,N/2-1$$

where k is replaced by 2l+1.

Even indexed $N-$point DFT coefficients can be expressed as follows:

$$X[2l] = \sum_{n=0}^{N/2-1} g[n] W_{N/2}^{ln}, \quad l = 0,1,2,\ldots,N/2-1 \tag{4.3}$$

where $g[n] = x[n] + x\left[n+N/2\right]$ and $W_{N/2}^{ln} = e^{-j\frac{2\pi ln}{N/2}} = e^{-j\frac{2\pi 2ln}{N}} = W_N^{2ln}$. So, (**??**) is the $\frac{N}{2}$-point DFT of the sequence $g[n], \quad n = 0,1,2,\ldots,N/2-1$.

$$G[l] = \sum_{n=0}^{N/2-1} g[n] W_{N/2}^{ln}, \quad l = 0,1,2,\ldots,N/2-1$$

where $G[l] = X[2l]$. The odd indexed $N-$point DFT coefficients can be also expressed as $N/2-$point DFT. Notice that

$$X[2l+1] = \sum_{n=0}^{N/2-1} \tilde{h}[n] W_N^{2ln} W_N^n, \quad l = 0,1,2,\ldots,N/2-1 \tag{4.4}$$

where $\tilde{h}[n] = x[n] - x[n+N/2]$ since $(-1)^{2l+1} = -1$. Eq. (**??**) is the $\frac{N}{2}$-point DFT of the sequence

$$h[n] = (x[n] - x[n+N/2])\, W_N^n, \quad n = 0,1,\ldots,N/2-1$$

Therefore,

$$H[l] = \sum_{n=0}^{N/2-1} h[n] W_{N/2}^{ln}, \quad l = 0,1,\ldots,N/2-1$$

where $H[l] = X[2l+1]$. This process is described in Fig **??**. The flow-graph of Eq. **??** and **??** are shown in Fig. 4.1. Equations (**??**) and (**??**) represent the main idea behind the recursive decimation-in-frequency FFT algorithm. At this stage, the computational cost of implementing the DFT is $\frac{N}{2} + 2\left(\frac{N}{2}\right)^2 = \frac{N^2}{2} + \frac{N}{2}$ complex multiplications, which is less than N^2 for $N \geq 4\,(N = 2^p)$. Since this is adventageous we can continue dividing $\frac{N}{2}-$point DFTs into $\frac{N}{4}-$point DFTs. The flow graph of expressing N-point DFTs is shown in Fig. **??** for N = 8. Use (**??**) and (**??**) to divide each $\frac{N}{2}-$point DFT into two $\frac{N}{4}-$point DFTs. Then, the computational cost becomes $\frac{N}{2} + \frac{N}{2} + 4\left(\frac{N}{4}\right)^2$ complex multiplications. For $N = 8,\ \frac{N}{4} = 2$.

For $N = 2$, we have the so-called ‘butterfly’:

$$\begin{aligned}
B[k] &= \sum_{n=0}^{1} b[n] W_N^{kn}, \quad k = 0,1 \\
B[0] &= b[0] + b[1] \\
B[1] &= b[0] W_N^0 + b[1] W_{N=2}^1 = b[0] - b[1]
\end{aligned}$$

which is the main computational unit of the FFT algorithms and the last DFT stage of the N = 8 point DFT. The $N = 8 = 2^3$ point DFT can be computed in $p = 3 = \log_2 8$ stages and in each stage we perform $\frac{N}{2} = 4$ complex multiplications. Therefore, the computational cost of the decimation-in-frequency algorithm for $N = 2^p$-point DFT: $\frac{N}{2}.p = N^2 \log_2 N$ complex multiplications.

Butterflies are the building blocks of the radix-2 FFT algorithm, $N = 2^p$. The term radix-2 is used for FFTs constructed from 2-point butterflies.

In decimation-in-frequency algorithm, the input vector goes in an orderly fashion. But we shuffle the DFT coefficients after each stage. It turns out that the FFT output comes out in bit-reversed order as shown in Fig. **??**. Therefore it is very easy to rearrange the DFT coefficients. For example, the forth output of the decimation in frequency algorithm is $X[6] = X[110]$. The bit reversed version of 110 is 011 which is equal to 3. This works for all $N = 2^p$.

One complex multiplication is equivalent to 4 real multiplications. So the cost of N-FFT is $2N \log_2 N$ real multiplications for a complex input $x[n]$.

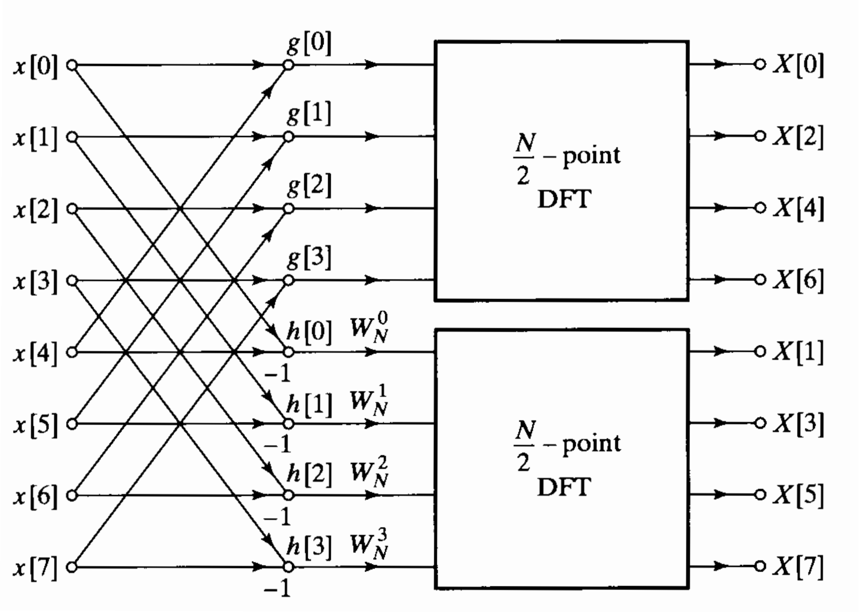


Fig. 4.1 Flowgraph of decimation-in-frequency algorithm of an $N = 8$-point DFT computation into two $\frac{N}{2} = 4$-point DFT computations.

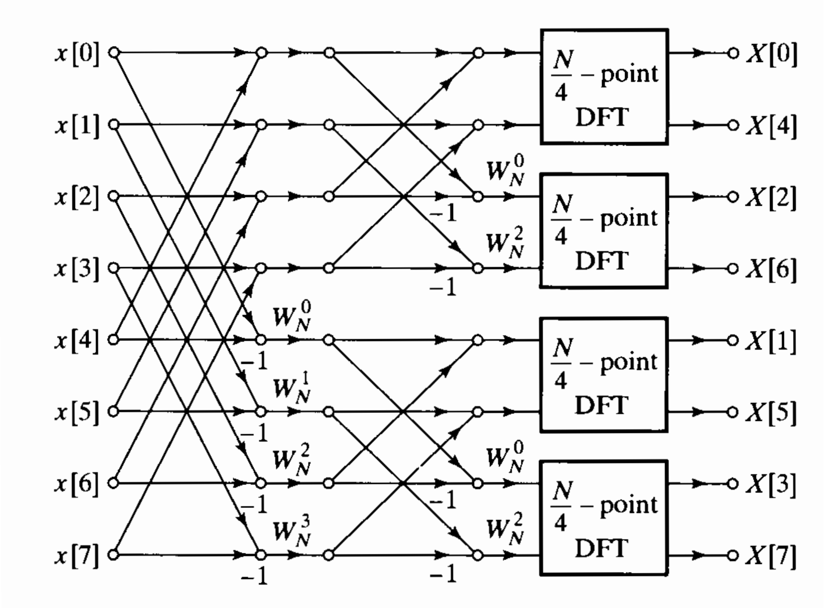


Fig. 4.2 Flowgraph of decimation-in-frequency algorithm of an $N = 8$-point DFT computation in terms of four $\frac{N}{4} = 2$-point DFT computations

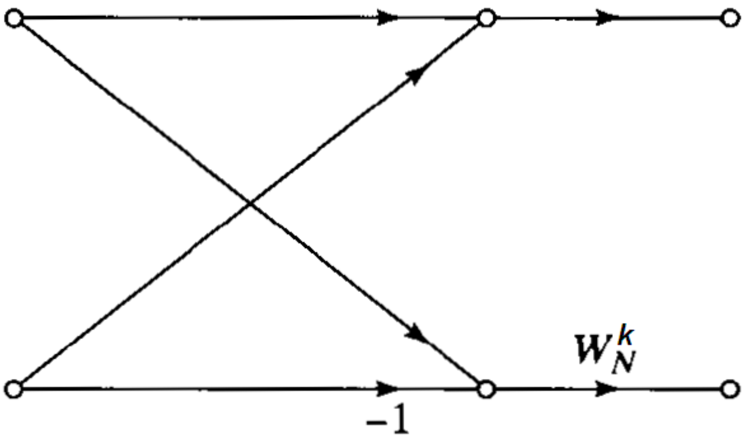


Fig. 4.3 Flowgraph of a typical butterfly used in decimation in frequency algorithm.

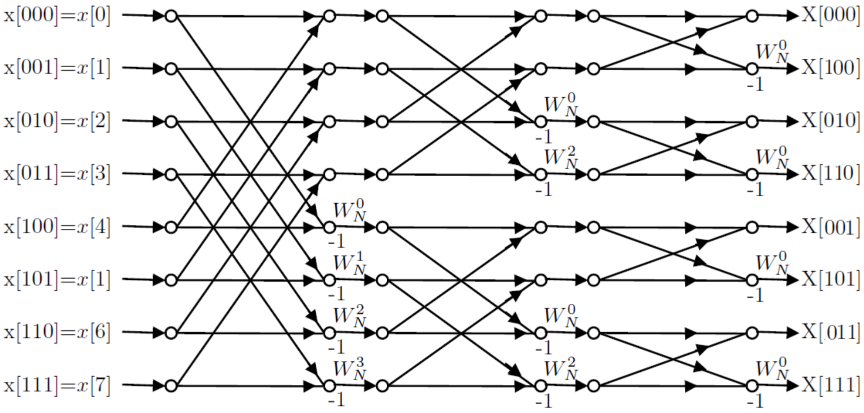


Fig. 4.4 Flowgraph of decimation-in-frequency decomposition of an N = 8-point DFT computation. It consists of $\log_2 N = 3$ stages. Output appears in bit-reversed order.

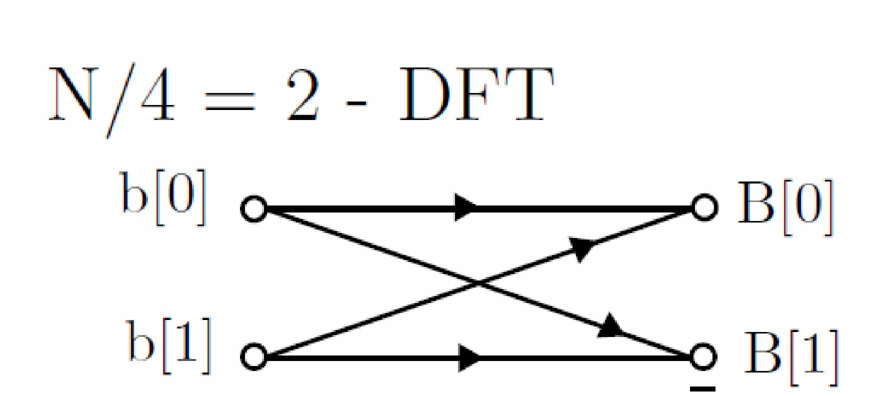


Fig. 4.5 Flowgraph of the 2-point DFT. They constitute the last stage of 8-point decimation-in-time FFT algorithm.

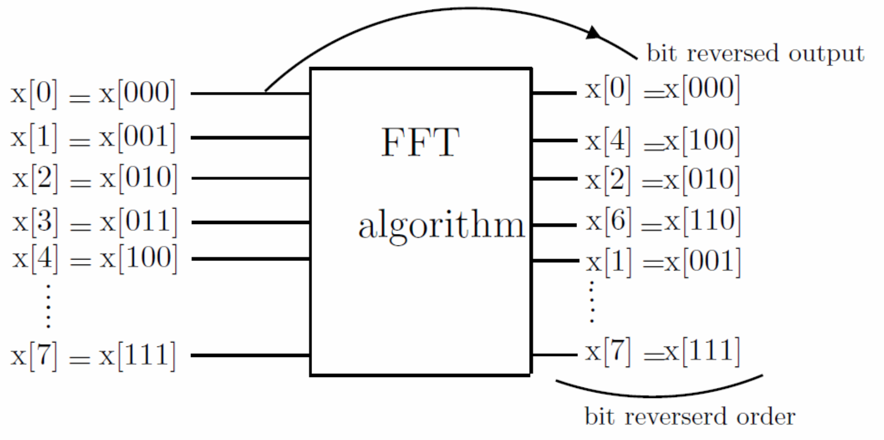


Fig. 4.6 DFT coefficients come out shuffled after decimation in frequency FFT algorithm, but they can be easily organized.

4.4 Decimation-in-Time FFT Computation Algorithm

Decimation-in-time algorithm is another way of computing DFT in a fast manner. The computational cost of decimation-in-time algorithm is also $N\log_2 N$ complex multiplications for an N-point DFT computation.

Similar to the decimation-in-frequency algorithm, we form two sums from Eq. **??** but this time we group the even and odd indexed time-domain samples. The DFT sum of Eq. **??** can be expressed in two parts as follows:

$$X[k] = \sum_{\substack{n=0 \\ n \text{ even}}}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} + \sum_{\substack{n=0 \\ n \text{ odd}}}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} \qquad k = 0, 1, \ldots, N-1$$

In the first sum we have the even indexed samples, and in the second sum we have the odd indexed samples, respectively. Therefore, we replace n by $2l$ in the first sum and n by $2l+1$ in the second sum, respectively.

$$X[k] = \sum_{l=0}^{N/2-1} x[2l] e^{-j\frac{2\pi}{N}k2l} \quad + \quad \sum_{l=0}^{N/2-1} x[2l+1] e^{-j\frac{2\pi}{N}k(2l+1)}$$

$$\textit{and}$$

$$X[k] = \sum_{l=0}^{N/2-1} x[2l] W_N^{2kl} \quad + \quad W_N^k \sum_{l=0}^{N/2-1} x[2l+1] W_N^{2kl}$$

Notice that $W_N^{2kl} = e^{-j\frac{2\pi}{N}2kl} = e^{-j\frac{2\pi}{N/2}kl} = W_{N/2}^{kl}$. Therefore,

$$X[k] = \sum_{l=0}^{N/2-1} x[2l] W_{N/2}^{kl} \quad + \quad W_N^k \sum_{l=0}^{N/2-1} x[2l+1] W_{N/2}^{kl} \tag{4.5}$$

In equation Eq. **??** the first sum is the $N/2-$point DFT of $x[2l]$ and the second sum is the $N/2-$point DFT of $x[2l+1]$, respectively. Therefore, we expressed $N-$point DFT in terms of two $N/2-$point DFTs as follows

$$X[k] = G[k] \, + \, W_N^k H[k], \qquad k = 0, 1, \ldots, N-1 \tag{4.6}$$

where $G[k],\ k = 0, 1, \ldots, N/2-1$ is the $N/2-$point DFT of $x[2l]$ and $H[k],\ k = 0, 1, \ldots, N/2-1$ is the $N/2-$point DFT of $x[2l+1]$, respectively. In (**??**), we also need $G[N/2], G[N/2+1], \ldots, G[N-1]$ and $H[N/2], H[N/2+1], \ldots, H[N-1]$. We take advantage of the periodicity of DFT and do not compute these DFT coefficients because $G[k+N/2] = G[k]$ and $H[k+N/2] = H[k]$.

The flowdiagram based on Eq. (**??**) is shown in Fig. **??** for $N = 8$. Similar to decimation-in-frequency algorithm, we can recursively continue computing $\frac{N}{2}$point DFTs using $\frac{N}{4}$-point DFTs etc. Flowgraphs of decimation-in-time FFT algorithm is shown in Figures **??** and **??**. There are $\log_2 N$ stages.

The basic building block of decimation-in-time algorithm is also a butterfly as shown in Fig. **??**.

In decimation-in-time algorithm, input samples have to be shuffled according to the following rule: $x[6] = x[(110)_2] \Rightarrow X[(011)_2] = X[3]$. Inputs are in bit-reversed order, whereas the outputs are regular.

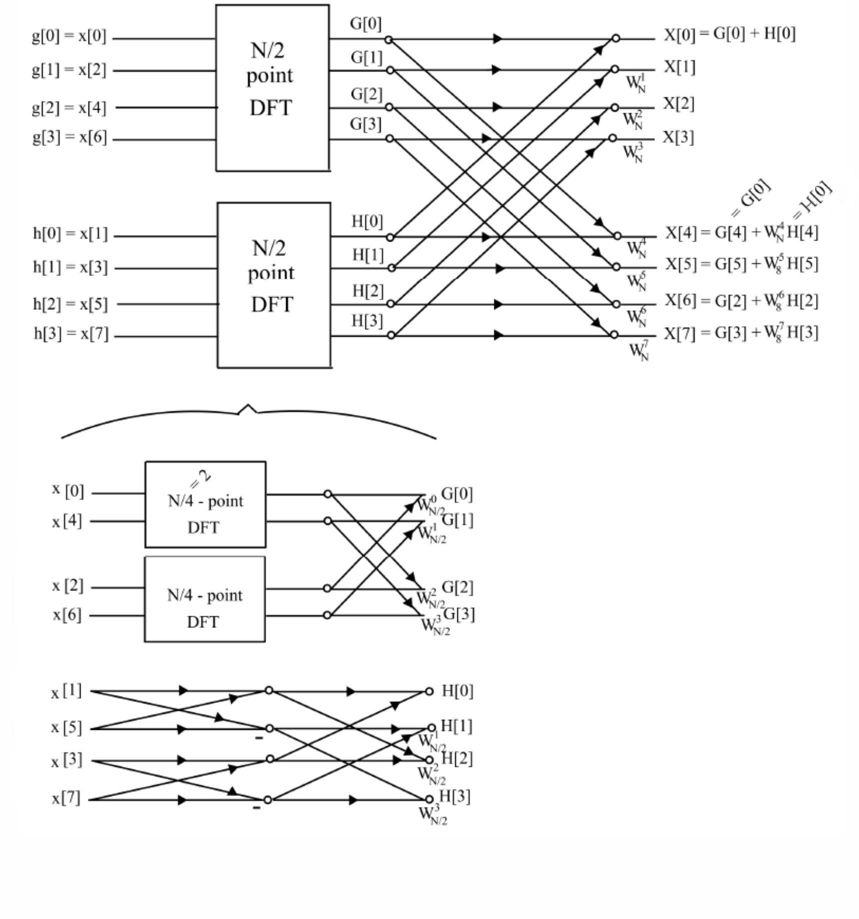


Fig. 4.7 Flowgraph based on Eq. (**??**): This is a decimation-in-time decomposition of an $N = 8$-point DFT computation into two $\frac{N}{2} = 4$-point DFT computations. Notice that $G[k]$ and $H[k]$ are periodic with period $\frac{N}{2} = 4$.

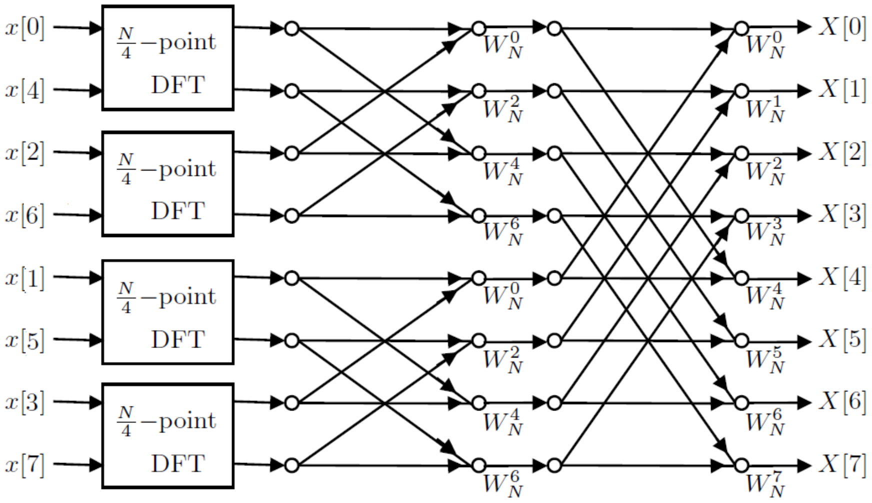


Fig. 4.8 Flowgraph of decimation-in-time algorithm of an 8-point DFT computation in three stages.

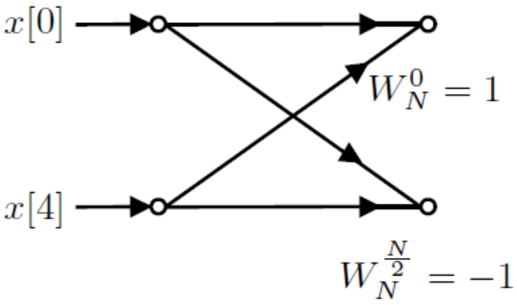


Fig. 4.9 Flowgraph of a typical $\frac{N}{4} = 2-$point DFT as required in the last stage of decimation-in-time algorithm.

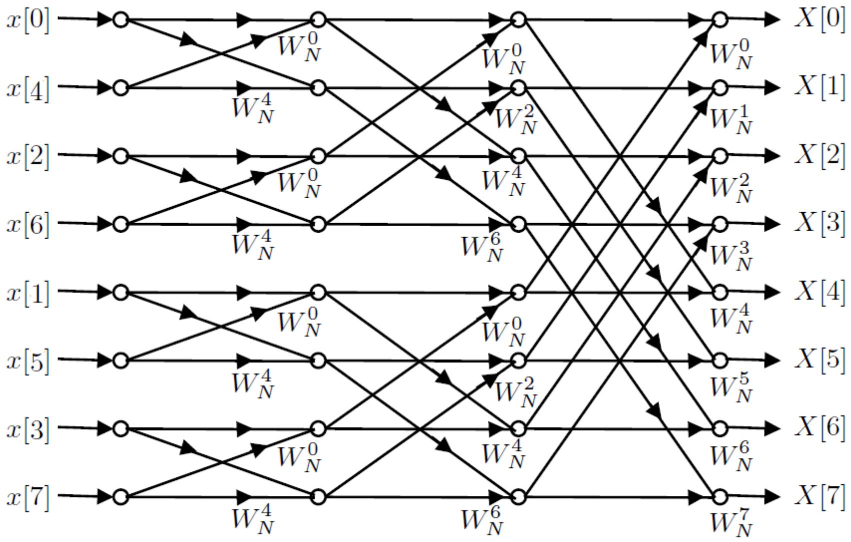


Fig. 4.10 Flowgraph of complete decimation-in-time decomposition of an $8-$point DFT computation.

4.5 FFT for an arbitrary N

If N can be expressed as a multiplication of two integers p and q, i.e., $N = p \cdot q$, then we can take advantage of this fact to develop an FFT algorithm similar to the algorithms described in Section 4.2 and 4.3.
Example: $p = 3$

$$X[k] = \sum_{\underbrace{n=0,3,6,\ldots}_{n=3l}}^{N-1} x[n]W_N^{kn} \quad + \sum_{\underbrace{n=1,4,7,\ldots}_{n=3l+1}}^{N-1} x[n]W_N^{kn} \quad + \sum_{\underbrace{n=2,5,8,\ldots}_{n=3l+2}}^{N-1} x[n]W_N^{kn}$$

$$X[k] = \sum_{l=0}^{N/3-1} x[3l]W_N^{k3l} \quad + \sum_{l=0}^{\frac{N}{3}-1} x[3l+1]W_N^{k(3l+1)} \quad + \sum_{l=0}^{N/3-1} x[3l+2]W_N^{k(3l+2)}$$

$$X[k] = \sum_{l=0}^{N/3-1} x[3l]W_{N/3}^{kl} \quad + \quad W_N^k \sum_{l=0}^{\frac{N}{3}-1} x[3l+1]W_{N/3}^{kl} \quad + \quad W_N^{2k} \sum_{l=0}^{\frac{N}{3}-1} x[3l+2]W_{N/3}^{kl}$$

where we take $N/3-$point DFTs:

$$X[k] = G[k] + W_N^k H[k] + W_N^{2k} V[k] \qquad k = 0,1,\ldots,N-1 \tag{4.7}$$

where

$$G[k] \text{ is the } N/3\text{– point DFT of } \{x[0],x[3],\ldots,x[N-3]\}$$
$$H[k] \text{ is the } N/3\text{– point DFT of } \{x[1],x[4],\ldots,x[N-2]\}$$
$$V[k] \text{ is the } N/3\text{– point DFT of } \{x[2],x[5],\ldots,x[N-1]\}$$

In (**??**), we need $G[k],H[k]$ and $V[k]$, $k = N/3,\ldots,N-1$. We do not compute these values. We simply use the periodic nature of DFT to generate the missing DFT coefficients: $G[k] = G[k+N/3] = G[k+2N/3]$, $k = 0,1,2,\ldots.$

After this single stage decomposition, we compute the $N-$point DFT of $x[n]$ using three $\frac{N}{3}$–point DFTs. The total computational cost is $3\left(\frac{N}{3}\right)^2 + 2N < N^2$ for large N.

In general, we factor N into its primes and

$$N = \underbrace{p \cdot q \cdots r}_{l \text{ primes}} \tag{4.8}$$

and perform the DFT in l stages because we have l prime numbers forming N.

It turns out that radix–4 DFT is the most efficient FFT because in 4–point DFT we do not perform any actual multiplication:

$$X[k] = \sum_{n=0}^{3} x[n]\, e^{-j\frac{2\pi kn}{4}}, \qquad k = 0,1,2,3$$

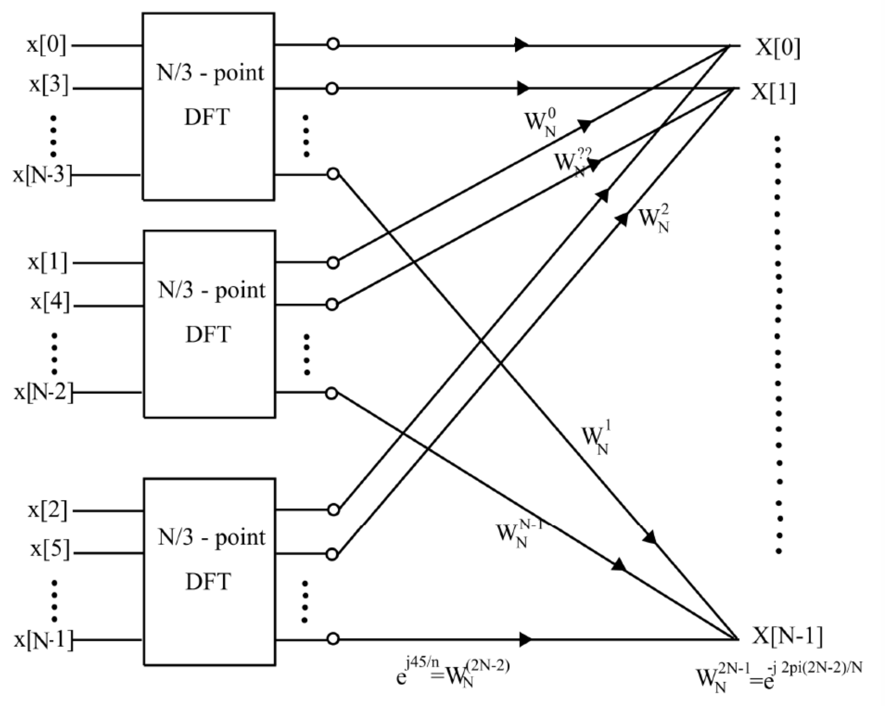


Fig. 4.11 Flowgraph of complete decimation-in-time decomposition of an 8−point DFT computation.

because $e^{-j\frac{2\pi kn}{4}}$ can take only $j, -j, 1$, and -1. The 4-point DFT is described in matrix form as follows:

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \end{bmatrix}$$

The computational cost in terms of number of multiplications is also Order$(N \log N)$.

4.6 Convolution using DFT (FFT)

Since FFT is a fast algorithm it may be feasible to compute convolution or filtering in the DFT domain. Let us consider the following convolution operation:

$$\underbrace{y[n]}_{\text{length } L_1+L_2-1} = \underbrace{h[n]}_{\text{length } L_1} * \underbrace{x[n]}_{\text{length } L_2} \xleftrightarrow{N-DFT} \underbrace{Y[k] = H[k]X[k]}_{N>L_1+L_2-1}$$

where $h[n]$ is the impulse response of the filter and $x[n]$ is the input. We can compute $y[n]$ in DFT domain as follows:

1. Compute $H[k]$ (Computational cost: $\frac{N}{2}\log_2 N$) This step may not be necessary in some cases because we can store $H[k]$ instead of $h[n]$ in the memory of the computer.
2. Compute $X[k]$ (Computational cost: $\frac{N}{2}\log_2 N$)
3. Compute $H[k]X[k], for \quad k = 0, 1, \ldots, N-1$ (Computational cost: N)
4. Compute DFT^{-1} [H[k]X[k]] (Computational cost: $\frac{N}{2}\log_2 N$)

Therefore the total cost is $N + 3\frac{N}{2}\log_2 N$ complex $\otimes = 4N + 6N\log_2 N$ real $\otimes$.

In general for long (or large order) filters, convolution using FFT can be more advantageous. One has to compute the required number of multiplications in time domain and frequency domain and compare the cost for each specific case. Consider the following two examples:

Example: Filter order $L_1 = 11$ and the length of the signal $L_2 = 900$. $y[n] = \sum_{k=0}^{L_1-1} h[k]x[n-k]$.
For a single $y[n]$, we need to perform 11 $\otimes$. Total cost of time domain convolution $\leq L_1 \cdot (L_1 + L_2 - 1) = 11 \cdot 910 \approx 10000 \otimes$. Frequency domain convolution requires $(N = 1024) \quad 4N + 6N\log_2 N \approx 4000 + 60000 \otimes$. Time domain convolution is better in this example.

Example: Filter order $L_1 = 101$ and $L_2 = 900$. Time domain convolution $\approx 100 \cdot 910 \approx 90000 \otimes$. Frequency domain convolution $\approx 4N + 6N\log_2 N = 64000 \otimes$. Frequency domain convolution is computationally better.

4.7 Exercises

1. $x[n]$ is defined as $x[n] = \sin\left(\frac{2\pi 4n}{N}\right)$ where $N = 16$.

a) Find the 16−point DFT of $x[n]$ analytically.
b) Calculate the 16−point DFT of $x[n]$ using Matlab.

2. Find the N−point DFTs of $x_1[n]$ and $x_2[n]$ where the sequences have the length N and $0 \leq n \leq N-1$.

$$x_1[n] = \cos^2\left(\frac{2\pi n}{N}\right) \qquad \text{and} \qquad x_2[n] = \cos^3\left(\frac{2\pi n}{N}\right)$$

3. $x[n] = \{5, 3, -2, 4, 3, -7, -1, 6\}$ is defined for $0 \leq n \leq 7$. $X[k]$ is the $8-$point DFT of $x[n]$. Determine the following values:

a) $X[0]$
b) $X[4]$

4. Given the sequence $x[n] = \delta[n] + 3\delta[n-1] + 2\delta[n-2]$, $X[k]$ is the $6-$ point DFT of $x[n]$. Moreover, $Y[k] = X^2[k]$ where $Y[k]$ is the DFT of $y[n]$.

a) Find values of $y[n]$ for $0 \leq n \leq 5$.
b) If $X[k]$ is defined to be the $N-$point DFT of $x[n]$, what should be the minimum value of N so that $y[n] = x[n] * x[n]$.

4.8 Computer Projects

PART 1.

1. Divide a *speech signal* or *music signal* into frames of size $N = 64$.

2. Compute the $N-$point DFT of each frame.
3. Save only first $L = 20$ DFT coefficients.
4. Reconstruct the frame from these L coefficients. (Do not disturb the symmetry of DFT during inverse DFT computation. $X[k] = X^*[N-k]$ for real signals, $N = 64$ here. Pad zeros for missing DFT coefficients.)
5. Comment on the quality of reconstructed and the original speech signal.
6. What is the effective data compression ratio? Note that DFT coefficients may be complex valued!
7. Repeat the above experiment for $L = 10$ and $L = 5$.
8. Repeat all above with DCT instead of DFT.

Hints:
1. Pad zeros at the end of the original signal in order to get the number of total samples to have a multiple of $N = 64$.
2. Effective Data Compression Rate (EDCR) can be calculated by:

$$\text{EDCR} = \frac{\text{number of samples in the original signal}}{\text{numberofsavedrealcoefficients} + (2 \times \text{number of saved complex coefficients})}$$

PS: You may use this formula for "per frame" or for the whole signal. These will give actually the same result.
3. You will save first L coefficients of DFT/DCT of original frame. And set the remaining coefficients to zero. Then, in reconstruction step of a frame, you should be careful about the conjugate symmetry of the DFT signal which is $X[k] = X^*[N-k]$.
PS: The first coefficient of DFT is always real, and $X[0] = X^*[N] = X[N]$ by the above formula!
4. Using these L coefficients and corresponding $(L-1)$ conjugate symmetric coefficients, (in between of these are all zeros), you take the IDFT of each frame. By the missing coefficients, the IDFT might be complex valued with the imaginary parts in the order of $10^{-18} - 10^{-20}$. You may ignore the imaginary parts and use the real parts of the reconstructed signal samples.

Please make comments and state any conclusions in your report. Plot the original signal and reconstructed signals to get some idea about the compression quality. Also listen to those signals and make comments about intelligibility. You may use

```
    soundsc(signal_name,8000)
```

command to listen them in Matlab.

PART 2.

1. Given the *signal* in the *mat* file, describe a way of compression with MSE below 10^{-28}. The signal is sampled at 200 Hz.
2. If you did not use the compression method given in Part 1, then please compare your method and the one given in Part 1. What are the advantages and disadvantages of your method? Is it possible to use your new idea on the signals given in Part 1?

4.9 Exercises

1. $x[n]$ is defined as $x[n] = \sin\left(\frac{2\pi 4n}{N}\right)$ where $N = 16$.

a) Find the 16−point DFT of $x[n]$ analytically.
b) Calculate the 16−point DFT of $x[n]$ using Matlab.

2. Find the $N-$point DFTs of $x_1[n]$ and $x_2[n]$ where the sequences have the length N and $0 \leq n \leq N-1$.

$$x_1[n] = \cos^2\left(\frac{2\pi n}{N}\right) \qquad \text{and} \qquad x_2[n] = \cos^3\left(\frac{2\pi n}{N}\right)$$

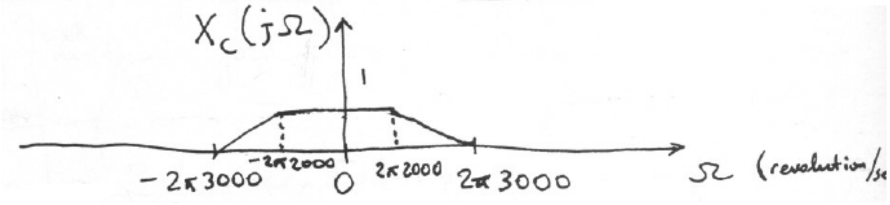
3. $x[n] = \{5, 3, -2, 4, 3, -7, -1, 6\}$ is defined for $0 \leq n \leq 7$. $X[k]$ is the 8−point DFT of $x[n]$. Determine the following values:

a) $X[0]$
b) $X[4]$

4. Given the sequence $x[n] = \delta[n] + 3\delta[n-1] + 2\delta[n-2]$, $X[k]$ is the 6− point DFT of $x[n]$. Moreover, $Y[k] = X^2[k]$ where $Y[k]$ is the DFT of $y[n]$.

a) Find values of $y[n]$ for $0 \leq n \leq 5$.
b) If $X[k]$ is defined to be the $N-$point DFT of $x[n]$, what should be the minimum value of N so that $y[n] = x[n] * x[n]$.

5. Draw the flow-diagram of the $N = 6-$point Decimation-in-time Discrete Fourier Transform algorithm. Show your equation and work.
6. Let $x_c(t)$ be a continuous time signal. $X_c(j\Omega)$ is the continuous time Fourier transform of $x_c(t)$. $x_c(t)$ is sampled: $x[n] \triangleq x_c\left(n\frac{1}{8000}\right)$, $n = 0, \pm1, \pm2, \ldots$



(a) Plot $X(e^{j\omega}) = \mathscr{F}\{x[n]\}$.
(b) Let $v[n]$ be the down-sampled version of $x[n]$ by 2. Plot $X(e^{j\omega})$.
(c) Let $x[n]$, $n = 0, \pm 1, \pm 2, \ldots, \pm 511, \pm 512$ is available. $X[k]$ is the 1024-point DFT of $[x[-511], x[-510], \ldots, x[512]]$. Approximately plot $|X[k]|$ versus k.
(d) Let

$$u[n] = \begin{cases} v[n/2] & \text{if } n \text{ even} \\ 0 & \text{if } n \text{ odd} \end{cases}$$

Plot $U(e^{j\omega})$.
(e) What is the computational cost of computing $X[k]$ is the Fast Fourier Transform (FFT) is used in part (c)?
7. (a) Describe an $N-$point DFT using two $N/2-$point DFT's (N is divisible by 2).
(b) Let $X(e^{j\omega} = \frac{1}{2} + \frac{1}{2}\cos\omega)$. $X[k]$ is defined as $X[k] = X(e^{j\omega}\big|_{\omega = \frac{2\pi k}{64}}$, $k = 0, 1, \ldots, 63$.
Find $x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi kn}{N}}$, $n = 0, 1, 2, \ldots, 63 = N-1$
8. (a) Draw the flow-graph of $N = 6-$point decimation-in-time Fast Fourier Transform (FFT) algorithm.
(b) Let $x[n] = \left\{ \frac{1}{2}, \underbrace{1}_{n=0}, \frac{1}{2} \right\}$. Calculate the $N = 6-$point DFT of $x[n]$ using the flow-graph in (a).
9. Given a discrete-time sequence $x[n]$, $n = 0, \ldots, 8$; the $9-$point DFT shall be computed.
(a) Derive a decimation-in-time FFT method for this task.
(b) Plot the corresponding FFT flow-graph.
(c) Compare the computational complexity of calculating the FFT method developed in part (a) with regular $9-$point DFT.

Chapter 5
Applications of DFT (FFT)

5.1 Introduction

The FFT algorithm makes the DFT a practical tool in many applications. According to G. Strang FFT is the most important numerical algorithm of the 20th century.

We consider two applications in this chapter:

- Implementation of LTI systems in DFT domain and
- waveform coding.

In Section 2 we study convolution using DFT and show that it may be computationally more efficient to use the DFT domain convolution when the filter order is large. In Section 3, we discuss how streaming data can be filtered in DFT domain and in Section 4, we discuss how DFT can be used in coding of digital waveforms. Actually, DFT is not used in waveform coding but Discrete Cosine Transform (DCT) is used. We establish the relation between DFT and the DCT in Section 4. DCT is the transform used in JPEG image coding and MPEG family of video coding standards.

5.2 Convolution using DFT (FFT)

$$\underbrace{y[n]}_{\text{length } L_1+L_2-1} = \underbrace{h[n]}_{\text{length } L_1} * \underbrace{x[n]}_{\text{length } L_2} \xleftrightarrow{N-DFT} \underbrace{Y[k] = H[k]X[k]}_{N>L_1+L_2-1}$$

Implementation:

1. Compute $H[k]$ (Comp. Cost: $\frac{N}{2}\log_2 N$) (This step may not be necessary in some cases because we can store $H[k]$ instead of $h[n]$ in the memory of the computer.)
2. Compute $X[k]$ (Comp. Cost: $\frac{N}{2}\log_2 N$)
3. Compute $H[k]X[k], \quad k = 0, 1, \ldots, N-1$ (Comp. Cost: N)
4. Compute DFT^{-1} [H[k]X[k]] (Comp. Cost: $\frac{N}{2}\log_2 N$)

Total $= N + 3\frac{N}{2}\log_2 N$ complex $\otimes = 4N + 6N\log_2 N$ real $\otimes$.

For long (or large order filters) perform convolution using FFT.

For each case, carry out a computational cost analysis and decide!

Ex: Filter order $L_1 = 11$ and $L_2 = 900$. $y[n] = \sum_{k=0}^{L_1-1} h[k]x[n-k]$.
For a single $y[n]$, we need to perform 11 $\otimes$. Total cost of time domain convolution $\leq L_1 \cdot (L_1 + L_2 - 1) = 11 \cdot 910 \approx 10000 \otimes$. Frequency domain convolution requires $(N = 1024) \quad 4N + 6N\log_2 N \approx 4000 + 60000 \otimes$. Time domain convolution is better in this example.

Ex: Filter order $L_1 = 101$ and $L_2 = 900$. Time domain convolution $\approx 100 \cdot 910 \approx 90000 \otimes$. Frequency domain convolution $\approx 4N + 6N\log_2 N = 64000 \otimes$. Frequency domain convolution is computationally better.

5.3 Overlap and Add Method

In the previos section we learned how to convolve two finite extent signals in DFT domain. In this section we cover what happens when x is a very long duration signal or a streaming signal, i.e., new samples of x arrive in time.

Let $x[n]$ be a long duration signal (or it may be streaming). We divide the input signal x into small windows and perform convolutions with the windowed data. This means that we need some memmory space to store the streaming data. Whenever we have enough date we perform the convolution in the DFTdomain. As a result we introduce some delay but the delay may be tolerable in some applications.

Overlap and add method is based on the linearity of the convolution:

$$\begin{aligned} y[n] &= (x_1[n] + x_2[n] + x_3[n] + \ldots) * h[n] \\ y[n] &= x_1[n] * h[n] + x_2[n] * h[n] + x_3[n] * h[n] + \ldots \end{aligned}$$

where we divided the signal $x[n]$ into sub-signals

$$x[n] = x_1[n] + x_2[n] + x_3[n] + \ldots$$

and

$$\begin{aligned} x_1[n] &= x[n]w[n] &&= [x[0], x[1], \ldots, x[L-1]] \\ x_2[n] &= x[n]w[n-L] &&= [x[L], x[L+1], \ldots, x[2L-1]] \\ &\text{and} \\ x_3[n] &= x[n]w[n-2L] &&= [x[2L], x[2L+1], \ldots, x[3L-1]] \end{aligned}$$

Notice that $w[n]$ is a rectangular window of length L:

$$w[n] = \begin{cases} 1\,, & n = 0, 1, \ldots, L-1 \\ 0\,, & \text{otherwise} \end{cases}$$

Each convolution

$$y_i[n] = x_i[n] * h[n]\,,\; i = 1, 2, \ldots$$

has a length of $L+M-1$ where $h[n]$ has a length of M.

We can use $N \geq L+M-1$ length DFT to compute $y[n]$ by computing $y_i[n]\,,\; i = 1, 2, \ldots$

$$y[n] = y_1[n] + y_2[n] + y_3[n] + \ldots$$

Therefore, the input signal $x[n]$ is divided into windows of length L. After using $N-$ point DFTs, we obtain $y_1[n], y_2[n], \ldots$.

Since the starting points of $y_1[n]$ is 0, $y_2[n]$ is L, $y_3[n]$ is $2L$, the method is called ”overlap and add” method.

Another related streaming data filtering method is called the overlap and save method. You can find detailed information about the ”overlap and save” method in references [1]-[5].

5.4 Discrete Cosine Transform (DCT)

General Transformation Topic

$$X[k] = \sum_{n=0}^{N-1} x[n]\phi_k^*[n] \quad \text{and} \quad x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]\phi_k[n]$$

Orthogonal Basis

$$\frac{1}{N}\sum_{n=0}^{N-1} \phi_k[n]\phi_m^*[n] = \begin{cases} 1 & \text{if } k = m \\ 0 & \text{if } k \neq m \end{cases}$$

In the case of DCT, we have cosine basis. Cosines are

- real functions
- even symmetric
- periodic

In DFT, periodicity of the transformed signal is assumed.

In DFT, what we do is to form a periodic sequence from the finite length signal.

In DCT, we form an even symmetric and periodic sequence from the finite length signal.

Example: $x[n] = \{\underbrace{a}_{n=0}, b, c, d\}$

$$\begin{aligned}
\tilde{x}_1[n] &= \{a,b,c,d,c,b,a,b,c,d,c,b,a\}\\
\tilde{x}_2[n] &= \{a,b,c,d,d,c,b,a,a,b,c,d,d,c,b,a,a\}\\
\tilde{x}_1[n] &= \{a,b,c,d,0,-d,-c,-b,-a,-b,-c,-d,0,d,c,b,a\}\\
\tilde{x}_1[n] &= \{a,b,c,d,0,-d,-c,-b,-a,-a,-b,-c,-d,0,d,c,b,a,a\}
\end{aligned}$$

All of the above signals are periodic with $N = 16$ or less and they have even symmetry.

The first step in DCT is to form one of these periodic, even symmetric sequences. Therefore, we have four different definitions of DCT. Most commonly used ones are DCT-1 and DCT-2 which includes $\tilde{x}_1[n]$ and $\tilde{x}_2[n]$.

$\tilde{x}_1[n] = x_\alpha[(n)_{2N-2}] + x_\alpha[(-n)_{2N-2}]$ where $x_\alpha[n] = \alpha[n]x[n]$ and

$$\alpha[n] = \begin{cases} 1/2 & \text{if } n = 0,\ n = N-1 \\ 1 & \text{otherwise} \end{cases}$$

denotes the weighting of the endpoints because doubling occurs at the endpoints.

DCT-1 is defined as

$$\begin{aligned}
X^{\mathrm{C1}}[k] &= 2\sum_{n=0}^{N-1} \alpha[n]x[n]\cos\left(\frac{\pi k n}{N-1}\right), \qquad 0 \le k \le N-1\\
x[n] &= \frac{1}{N-1}\sum_{k=0}^{N-1} \alpha[k]X^{\mathrm{C1}}[k]\cos\left(\frac{\pi k n}{N-1}\right), \qquad 0 \le n \le N-1
\end{aligned}$$

$\tilde{x}_2[n] = x[(n)_{2N}] + x[(-n-1)_{2N}]$. No modifications since the end points do not overlap.

DCT-2 is defined as

$$\begin{aligned}
X^{\mathrm{C2}}[k] &= 2\sum_{n=0}^{N-1} \alpha[n]x[n]\cos\left(\frac{\pi k(2n+1)}{2N}\right), \qquad 0 \le k \le N-1\\
x[n] &= \frac{1}{N}\sum_{k=0}^{N-1} \beta[k]X^{\mathrm{C2}}[k]\cos\left(\frac{\pi k(2n+1)}{2N}\right), \qquad 0 \le n \le N-1
\end{aligned}$$

where

$$\beta[k] = \begin{cases} 1/2 & \text{if } k = 0 \\ 1 & \text{if } 1 \le k \le N-1 \end{cases}$$

5.5 Relationship between DFT and DCT

Obviously for different definitions of DCT (as DCT-1 and DCT-2), there exist different relationships.

5.5.1 Relation between DFT and DCT-1

From the former sections, we know that

$$\tilde{x}_1[n] = x_\alpha[(n)_{2N-2}] + x_\alpha[(-n)_{2N-2}] \qquad n = 0, 1, \ldots, 2N-3$$

where $x_\alpha[n] = \alpha[n]x[n]$.

Assume that $X_\alpha[k]$ is the $(2N-2)-$point DFT of $x_\alpha[n]$.

$$\begin{aligned} X_1[k] &= X_\alpha[k] + X_\alpha^*[k] = 2Re\{X_\alpha[k]\} \qquad k = 0, 1, \ldots, 2N-3 \\ &= 2\sum_{n=0}^{N-1} \alpha[n]x[n]\cos\left(\frac{2\pi kn}{2N-2}\right) = X^{\mathrm{C1}}[k] \end{aligned}$$

where $X_1[k]$ is $(2N-2)-$point DFT of $\tilde{x}_1[n]$

Example: $x[n] = \{a, b, c, d\} \;\Rightarrow$

$$\begin{aligned} \tilde{x}_1[n] &= \{a/2, b, c, d/2, 0, 0\} + \{a/2, 0, 0, d/2, c, b\} \\ &= \{a, b, c, d, c, b\} \\ X_1[k] &= \sum_{n=0}^{5} \tilde{x}_1[n] e^{\frac{-j2\pi kn}{6}} \\ &= a + be^{\frac{-j2\pi k}{6}} + ce^{\frac{-j4\pi k}{6}} + d(-1)^k \\ &\qquad + be^{\frac{-j10\pi k}{6}} + ce^{\frac{-j8\pi k}{6}} \\ &= a + 2b\cos\left(\frac{2\pi k}{6}\right) + 2c\cos\left(\frac{4\pi k}{6}\right) + d(-1)^k \end{aligned}$$

Conclusion: DCT-1 of an $N-$point sequence is identical to the $(2N-2)-$point DFT of the symmetrically extended sequence $\tilde{x}_1[n]$, and it is also identical to twice the real part of the first N points of the $(2N-2)-$point DFT of the weighted sequence $x_\alpha[n]$.

5.5.2 Relation between DFT and DCT-2

$$\tilde{x}_2[n] = x[(n)_{2N}] + x[(-n-1)_{2N}]$$

Let $X[k]$ be the $2N-$point DFT of $N-$point signal $x[n]$. Then,

$$\begin{aligned}
X_2[k] &= X[k] + X^*[k]e^{\frac{j2\pi k}{2N}} \\
&= e^{\frac{j\pi k}{2N}} \left[X[k]e^{\frac{-j\pi k}{2N}} + X^*[k]e^{\frac{j\pi k}{2N}} \right] \\
&= e^{\frac{j\pi k}{2N}} 2Re\left\{ X[k]e^{\frac{-j\pi k}{2N}} \right\} \\
&= e^{\frac{j\pi k}{2N}} Re\left\{ 2\sum_{n=0}^{2N-1} x[n]e^{\frac{-j2\pi kn}{2N}} e^{\frac{-j\pi k}{2N}} \right\} \\
&= e^{\frac{j\pi k}{2N}} Re\left\{ 2\sum_{n=0}^{2N-1} x[n]e^{\frac{-j\pi k(2n+1)}{2N}} \right\} \\
&= e^{\frac{j\pi k}{2N}} \underbrace{2\sum_{n=0}^{N-1} x[n]\cos\left(\frac{\pi k(2n+1)}{2N}\right)}_{X^{\mathrm{C2}}[k]} \\
X_2[k] &= e^{\frac{j\pi k}{2N}} X^{\mathrm{C2}}[k], \qquad k = 0, 1, \ldots, N-1
\end{aligned}$$

Chapter 6
FIR Filter Design and Implementation

We first review the Linear-Time Invariant (LTI) systems and convolution. In this chapter we focus on the design of LTI systems which have Finite-extent Impulse Responses (FIR). The goal is to find appropriate LTI system parameters such that the filter meets some pre-specified requirements in the frequency domain.

6.1 Linear-Time Invariant Systems

In LTI systems the relation between the input x and the output y is given by the convolution sum

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k]$$
$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

where h is the impulse response of the system, i.e., the response that we get for the input $\delta[n]$. In FIR systems the above sum becomes a finite sum. Here is an example:
Example: $h[n] = \{\underbrace{1}_{n=0}, 2\} \quad \longleftarrow$ FIR
Find the I/O relation for $h[n]$
FIR: Finite-Extent Impulse Response

$$y[n] = h[0]x[n] + h[1]x[n-1]$$
$$y[n] = 1x[n] + 2x[n-1]$$

In Finite-extent Impulse Response (FIR) systems, the LTI system performs a running average of the input. In general, $h[n] = \{a_{-M}, a_{-M+1}, \ldots, \underbrace{a_0}_{n=0}, a_1, \ldots, a_L, \}$ leads to the input/output (I/O) relation:

$$y[n] = \sum_{k=-M}^{L} a_k\, x[n-k] \tag{6.1}$$

where $h[k] = a_k$ for $k = -M, \ldots, L$ which is an anti-causal FIR filter. In discrete-time domain anti-causality is not problem in many cases. Here are some FIR filters:

$$y[n] = \sum_{k=0}^{M-1} h[k]x[n-k] \qquad \text{Causal FIR}$$

$$y[n] = \sum_{k=0}^{M-1} a_k x[n-k] \qquad \text{where } h[k] = a_k$$

$$y[n] = \sum_{k=-L}^{L} b_k x[n-k] \qquad \text{Non-causal FIR where } h[k] = b_k, \quad k = 0, \pm 1, \ldots, \pm L.$$

6.1.1 Design of FIR Filters Using a Rectangular Window

In this chapter, we study the design of low-pass, band-pass, high-pass, band-stop and notch filters.

Design Problem: Find the impulse response $h[n]$ satisfying some requirements in DTFT domain. We consider the design of a low-pass filter design as an example.

Example: Low-pass filter design:

An ideal low-pass filter $H_{id}(e^{j\omega}) = 1$ for $-\omega_c \le \omega \le \omega_c$ and zero otherwise within $-\pi$ and π. Since the DTFT is 2π periodic $\omega = 0, \pm 2\pi, \pm 4\pi, \ldots$ are all equivalent to each other.

$$h_{id}[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_{id}(e^{j\omega})e^{j\omega n}d\omega$$
$$h_{id}[n] = \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} e^{j\omega n}d\omega = \frac{\sin(\omega_c n)}{\pi n} \qquad n = 0, \pm 1, \pm 2, \ldots$$

Clearly, $h_{id}[n]$ is of infinite extent. Therefore, we cannot implement the convolutional sum to realize this filter. One possible approach is to truncate $h_{id}[n]$ and obtain an FIR filter:

$$h_T[n] = \begin{cases} h_{id}[n], & n = 0, \pm 1, \pm 2, \ldots, \pm L \\ 0, & \text{otherwise} \end{cases} \qquad \longleftarrow \text{FIR non-causal}$$

This is basically rectangular windowing because

$$h_T[n] = h_{id}[n] \times w[n]$$

where w is the anti-causal rectangular window

$$w[n] = \begin{cases} 1, & n = 0, \pm 1, \pm 2, \ldots, \pm L \\ 0, & \text{otherwise} \end{cases}$$

This approach is straightforward but it leads to the well-known Gibbs effect in frequency domain. We have no control over the frequency response.

6.1.2 Second Approach: Least-Squares Approach

Let us try another approach. We try to minimize the mean square error

$$\min \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| H_L(e^{j\omega}) - H_{id}(e^{j\omega}) \right|^2 d\omega$$

The above optimization problem equivalent to minimizing:

$$B(h_L[n]) = \sum_{n=-\infty}^{\infty} |h_L[n] - h_{id}[n]|^2$$

$$\sum_{n=-L}^{L} |h_L[n] - h_{id}[n]|^2 + \sum_{n=L+1}^{\infty} |h_{id}[n]|^2 + \sum_{n=-\infty}^{-L-1} |h_{id}[n]|^2$$

where we used the Parseval relation. The equivalent minimization problem is as follows

$$\min \sum_{n=-L}^{L} \left(\underbrace{h_L[n]}_{\text{real}} - \underbrace{h_{id}[n]}_{\text{real, known}} \right)^2 = \min B(h_L[n])$$

To minimize the above cost function we take the derivative of $B(h_L[n])$ with respect to the filter coefficients and set the result to zero as follows:

$$\frac{\partial B}{\partial h_L[k]} = 0 \qquad k = 0, \pm 1, \pm 2, \ldots, \pm L$$

As a result we get the same solution as the rectangular window based filter design:

$$h_L[k] = \begin{cases} h_{id}[k], & k = 0, \pm 1, \pm 2, \ldots, \pm L \\ 0, & \text{otherwise} \end{cases}$$

This design method produced nothing new! It is the same as the rectangular window (or Truncation) design. This approach causes the Gibbs phenomenon because we try to approximate ideal filters with sudden jumps in frequency domain with smooth functions (sinusoids).

6.1.3 Window-Based FIR Filter Design

$h_w[n] = h_{id}[n] \cdot w[n]$ where $w[n]$ is a window of length $2L+1$ and centered around $n=0$. ($h_{id}[n]$ is symmetric wrt $n=0$ for low-pass,high-pass and band-pass filters).
Example:
Hanning window:

$$w_{hn}[n] = \frac{1}{2} + \frac{1}{2}\cos\frac{2\pi n}{M-1} \qquad M = 2L+1, \quad n = -L,\ldots,0,\ldots,L$$

Hamming window:

$$w_h[n] = 0.54 + 0.46\cos\frac{2\pi n}{M-1} \qquad M = 2L+1, \quad n = -L,\ldots,0,\ldots,L$$

Triangular window:

$$w_T[n] = 1 - \frac{|n|}{L}$$

Bartlett window:

$$w_B[n] = 1 - \frac{|n|}{L+1}$$

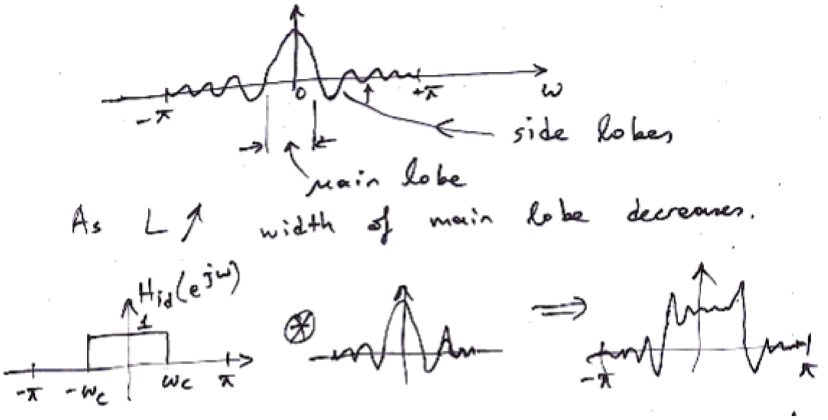
Blackman window (Causal):
$w_b[n] = 0.42 - 0.5\cos\frac{2\pi n}{M-1} + 0.08\cos\frac{4\pi n}{M-1}$
Assume that the window is rectangular.

$$W_r(e^{j\omega}) = \sum_{n=-L}^{L} 1.e^{-j\omega n} = 1 + e^{j\omega} + e^{-j\omega} + e^{j2\omega} + e^{-j2\omega} + \cdots + e^{jL\omega} + e^{-jL\omega}$$
$$= 1 + 2\left(\cos(\omega) + \cos(2\omega) + \cdots + \cos(L\omega)\right)$$

Table 6.1 Window properties

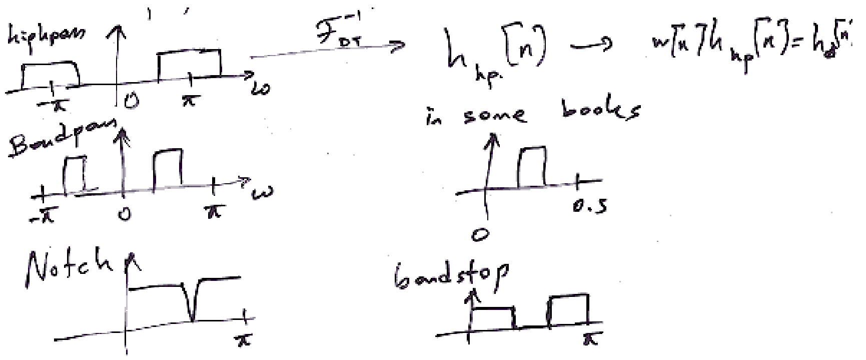
Window Type	Transition width of the mainlobe	Peak sidelobe (dB below the mainlobe)
Rect	$4\pi/2L+1$	-13
Bartlett (Tri.)	$8\pi/2L+1$	-27
Hanning	$8\pi/2L+1$	-32
Hamming	$8\pi/2L+1$	-43
Hanning	$12\pi/2L+1$	-58

As L increases, width of the mainlobe decreases.
Wider Mainlobe $\Longrightarrow$ Slow Transition from Passband to Stopband (If $W(e^{j\omega}) = \delta(\omega) \Longrightarrow$ Instant Transition)
Lower Peak Sidelobe $\Longrightarrow$ Reduced Gibbs Effect!



6.1.4 High-pass, Band-pass, Notch and Band-stop Filter Design

In high-pass, band-pass and band-stop FIR filters, the ideal filter is of infinite extent and anticausal $h_{id}[n] = h_{id}[-n]$ as the low-pass filter. Therefore, window-based FIR filter design method is the same as the low-pass filter design.



6.2 Causal Filter Design

As pointed out in the previous section, $h_{id}[n]$ is infinite-extent, noncausal and symmetric with respect to $n = 0$ in all of the above filters. Let the $(2L+1)$th order anticausal filter be

$$h_d[n] = w[n]h_{id}[n]$$

We can obtain a causal filter $h_c[n]$ from $h_d[n]$ by simply shifting $h_d[n]$ L time units as follows:

$$\begin{aligned} h_c[n] &\triangleq h_d[n-L] \\ h_d[n] &= \{h_d[-L],\ldots,\underbrace{h[0]}_{n=0},\ldots,h_d[L]\} \end{aligned}$$

Therefore

$$h_c[n] = \{\underbrace{h_d[-L]}_{n=0},\ldots,h[0],\ldots,\underbrace{h_d[L]}_{n=2L+1}\}$$

The order of the filter $h_c[n]$ is $M = 2L+1$, which is an odd number. Since $h_{id}[n] = h_{id}[-n]$, it is better to select M an odd number in low-pass, high-pass, ... filters. Let us also establish the relation between $H_c(e^{j\omega})$ and $H_d(e^{j\omega})$:

$$H_c(e^{j\omega}) = H_d(e^{j\omega})e^{-j\omega L} \impliedby \text{Linear phase term}$$

Filters have the same magnitude

$$\left|H_c(e^{j\omega})\right| = \left|H_d(e^{j\omega})\right|, \qquad \left|(e^{j\omega L} = 1\right|$$

but there is a linear phase term due to the shift in time domain:

$$\phi_c(e^{j\omega}) = -\omega L \impliedby \text{Linear phase term}$$

Anticausal filter design is also called the zero-phase design because $H_c(e^{j\omega})$ is real.

In citeOppenheim, windows are defined for causal filters: e.g.,

Blackman: $w_b[n] = 0.42 - 0.5\cos\frac{2\pi n}{M-1} + 0.08\cos\frac{4\pi n}{M-1} \qquad n = 0,1,\ldots,M-1$

and Hanning: $w_{hn}[n] = \frac{1}{2}\left(1-\cos\frac{2\pi n}{M-1}\right) \qquad n = 0,1,\ldots,M-1$

In discrete-time processing, causal FIR filters are not as critically important as continuous-time signal processing. Because, computers or digital signal processors can store future samples of the input and the output $y[n]$ can be computed whenever all of the necessary input samples $x[n-L], x[n-L+1],\ldots,x[-1],x[0],x[1],\ldots,x[n+L]$ are available. In image-processing, causality has no physical meaning. Left and right samples of the input are available.

Increasing the filter order M leads to better approximation but computational cost increases. In practice, one may need to design filters with several M values and check the frequency response until it is satisfactory. Because, we do not have any control on the resulting frequency response, the following filter order

$$\hat{M} = \frac{-20\log_{10}\left(\sqrt{\delta_1\delta_2}\right) - 13}{14.6(\omega_s - \omega_p)/2\pi} + 1$$

where δ_1 and δ_2 are the size of pass-band and stop-band ripples and ω_s, ω_p are approximate cut-off frequencies of pass-band and stop-band, respectively.

We have the following general rules for FIR filter design from pass-band to stop-band

- small, narrow transition region $(\omega_s - \omega_p) \searrow \Rightarrow$ high $M \nearrow$
- $\delta_1, \delta_2 \searrow \Rightarrow M \nearrow$
- $M \nearrow \Rightarrow$ high computational load

Example: Design a high-pass filte from a low-pass filter:
If the low-pass filter has a cut-off frequency of ω_c, then the high-pass filter with cut-off ω_c is given by:

$$H_{hp}(e^{j\omega}) = 1 - H_{lp}(e^{j\omega})$$

Therefore, we compute the inverse Fourier Transform of both sides and obtain:

$$h_{hp}[n] = \delta[n] - h_{lp}[n]\,.$$

Example: The following filter

$$h_{lp}[n] = \left\{\frac{1}{4}, \underbrace{\frac{1}{2}}_{n=0}, \frac{1}{4}\right\}$$

has a cut-off frequency of $\omega = \pi/2$. Here is the impulse response of the corresponding high-pass filter

$$h_{hp}[n] = \{\underbrace{1}_{n=0}\} - \left\{\frac{1}{4}, \underbrace{\frac{1}{2}}_{n=0}, \frac{1}{4}\right\} = \left\{-\frac{1}{4}, \underbrace{\frac{1}{2}}_{n=0}, -\frac{1}{4}\right\}$$

Another way to obtain a high-pass filter from a low-pass filter or vice versa is given by the following relation:

$$h_h[n] = (-1)^n h_l[n]$$

In this case

$$H_h(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h_l[n]e^{-j\pi n}e^{-j\omega n} = \sum_{n=-\infty}^{\infty} h_l[n]e^{-j(\omega+\pi)n}$$

$$H_h(e^{j\omega}) = H_l(e^{j(\omega+\pi)})$$

Example: Consider the following low-pass filter

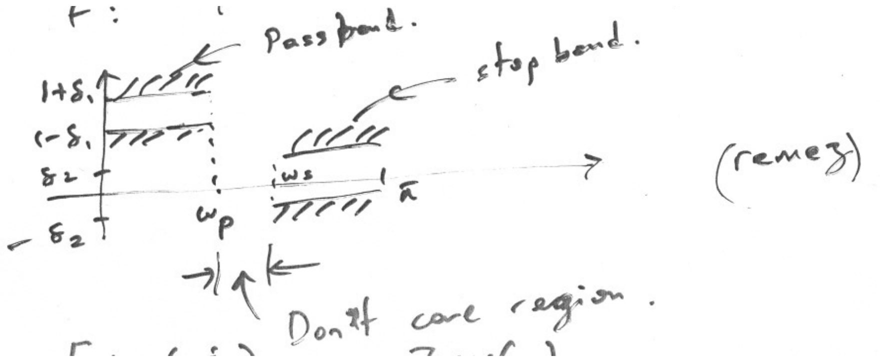
$$h_{lp}[n] = \left\{ \frac{1}{4}, \underbrace{\frac{1}{2}}_{n=0}, \frac{1}{4} \right\}$$

The transformation $h_h[n] = (-1)^n h_l[n]$ produces the high-pass filter

$$h_{hp}[n] = \left\{ -\frac{1}{4}, \underbrace{\frac{1}{2}}_{n=0}, -\frac{1}{4} \right\}.$$

6.3 Equiripple FIR Filter Design

State of the art FIR filter design method is the equiripple FIR filter design method. It produces the lowest order filter satisfying the specifications described in the following figure.



As pointed out in the previous section, purely real desired frequency response of low-pass, high-pass, band-pass and band-stop filters are all symmetric (with respect to $\omega = 0$). This leads to $h_{id}[n] = h_{id}[-n]$. Therefore:

$$H_{id}(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h_{id}[n]e^{-j\omega n} = h_{id}[0] + \sum_{n=1}^{\infty} 2h_{id}[n]\cos\omega n$$

For a $2L+1$ FIR filter:

$$H_L(e^{j\omega}) = \sum_{n=-L}^{L} h_L[n]e^{-j\omega n} = h_L[0] + \sum_{n=1}^{L} 2h_L[n]\cos\omega n$$

Using Chebyshev's relation $T_n(\alpha) = \cos(n\cos^{-1}\alpha)$, we obtain

$$H_L(e^{j\omega}) = \sum_{k=0}^{L} a_k(\cos\omega)^k = A(\omega)$$

One can use the techniques in polynomial approximation theory to design FIR filters.

Parks & McClellan solved the following problem (minimize the maximum error or minimize the maximum possible ripple):

$$\min_{h_L[n]}\left(\max_{\omega\in F}|E(\omega)|\right)$$

where the frequency range F is given by$[0 \le \omega \le \omega_p] \cup [\omega_s \le \omega \le \pi]$

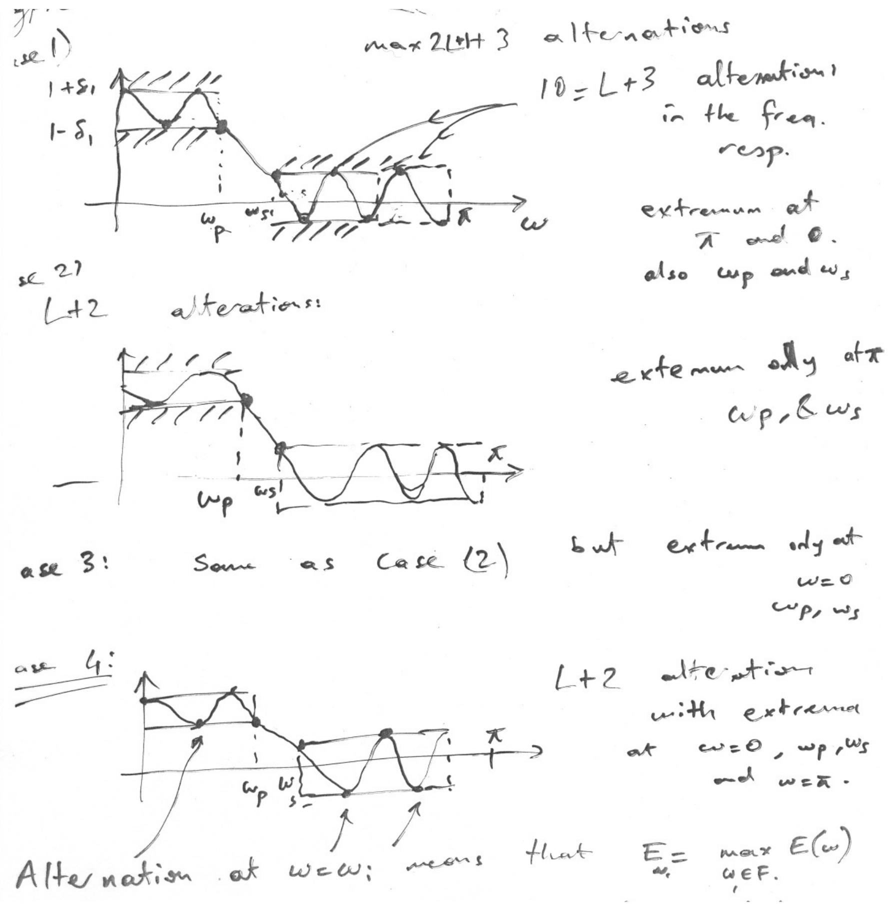
$$E(\omega) = \left[H_{id}(e^{j\omega}) - A(\omega)\right]W(\omega)$$

$$W(\omega) = \begin{cases} \frac{\delta_2}{\delta_1} = \frac{1}{K}, & 0 \le \omega \le \omega_p \\ 1, & \omega_s \le \omega \le \pi \end{cases}$$

You have a complete control over the frequency specs.

firpm is the equiripple FIR filter design tool in Matlab. Matlab's *firpm* requires that you specify ω_p, ω_s, and the filter order and the desired magnitude values in pass-band and stop-band. It produces an equiripple filter based on the specs. However, the pass-band and stop-band ripples cannot be specified. For desired δ_1 and δ_2 levels, one can use Kaiser's formula to estimate the filter order. Therefore, it may be necessary to run *firpm* several times to obtain the best design.

Typical solutions: Filter order $2L+1=7$, maximum $2L+1+3$ alternations.
Case 1): $10 = L+3$ alternations in the frequency response. Extremum at π and 0, also ω_p and ω_s.
Case 2): $L+2$ alternations. Extremum only at $\omega=\pi$, ω_p and ω_s.
Case 3): Same as Case (2) but extremum only at $\omega=0$, ω_p and ω_s.
Case 4): $L+2$ alternations with extrema at $\omega=0$, ω_p, ω_s and $\omega=\pi$.
Alternation at $\omega=\omega_i$ means that $E_\omega = \max\limits_{\omega\in F} E(\omega)$. The maximum number of alternations in a low-pass or high-pass filter design is $2L+1+3 \implies$ Equiripple FIR.



max 2L+1+3 alternations

10 = L+3 alternations in the freq. resp.

extremum at π and 0. also ω_p and ω_s

L+2 alterations:

extemum only at π ω_p, & ω_s

ase 3: Same as Case (2) but extrema only at $\omega=0$ ω_p, ω_s

L+2 alteration with extrema at $\omega=0$, ω_p, ω_s and $\omega=\pi$.

Alternation at $\omega=\omega_i$ means that $E = \max_{\omega \in F} E(\omega)$

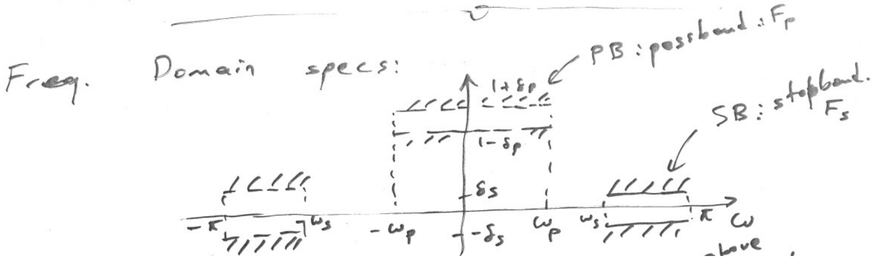
6.3.1 Equiripple FIR Filter Design by using the FFT based method

Another way to obtain equiripple FIR filters is based on typical engineering design approach called the method of projections onto convex sets, whose principles were established by famous 20th century Hungarian-American mathematician John von Neumann and Russian mathematician Bregman. We want the real frequency response $H(e^{j\omega})$ to satisfy the above bounds in pass-band and stop-band. Since $H(e^{j\omega}) = H(e^{-j\omega})$, we have $h[n] = h[-n]$ which is a time domain constraint.

We can express the frequency domain specs as follows

$$H_{id}(e^{j\omega}) - E_d(\omega) \leq H(e^{j\omega}) \leq H_{id}(e^{j\omega}) + E_d(\omega) \text{ for all } \omega.$$

where



$$H_{id}(e^{j\omega}) = \begin{cases} 1 & \omega \in F_p = [-\omega_p, \omega_p] \\ 0 & \omega \in F_s = [-\pi, \omega_s] \cup = [\omega_s, \pi] \end{cases}$$

$$E_d(\omega) = \begin{cases} \delta_p & \omega \in F_p \\ \delta_s & \omega \in F_s \end{cases}$$

We want to have an FIR filter. Therefore, this information introduces a new time domain constraint on the desired impulse response:

$$h[n] = 0 \text{ for } n > L, n < -L, \text{ (or } |n| > L)$$

This means that the filter order is $2L+1$. (In the article by Cetin et al [**?**], filter order is $2N+1$. We use N for the FFT size so I'll use L for the filter size.)

Iterative Procedure:

Initial step: Start with an arbitrary $h_0[n]$ but it should satisfy $h_0[n] = h_0[-n]$.
At the k-th iteration, we have the k-th approximation $h_k[n]$

- Compute the DTFT of $h_k[n]$ and obtain $H_k(e^{j\omega})$ (We have to use FFT).
- Impose the frequency domain constraints on $H_k(e^{j\omega})$ (This is the frequency domain projection operation)

$$G_k(e^{j\omega}) = \begin{cases} H_{id}(e^{j\omega}) + E_d(\omega) & \text{if } H_k(e^{j\omega}) > H_{id}(e^{j\omega}) + E_d(\omega) \\ H_{id}(e^{j\omega}) - E_d(\omega) & \text{if } H_k(e^{j\omega}) < H_{id}(e^{j\omega}) - E_d(\omega) \\ H_k(e^{j\omega}) & \text{otherwise.} \end{cases}$$

  As a result, $G_k(e^{j\omega})$ may be discontinuous.
- Compute the inverse FT of $G_k(e^{j\omega})$ and obtain $g_k[n]$ ($g_k[n] = g_k[-n]$ and real). Use FFT size of $N > 10L$.
- Impose the time-domain constraint (FIR constraint). This is the time-domain projection. The next iterate is given by

$$h_{k+1}[n] = \begin{cases} g_k[n] & \text{for } n = -L, \dots, -1, 0, 1, \dots, L \\ 0 & \text{otherwise} \end{cases}$$

Repeat the above procedure $h_{k+1}[n]$ until $\|h_k[n] - h_{k+1}[n]\| < \varepsilon$ where ε is a small number. $\lim_{k\to\infty} h_k[n]$ is the equiripple solution, if it exists. It gives the same solution as the Parks & McClellan algorithm.

Implementation Issues:

1. Since $g_k[n]$ may be an infinite extent signal in general, FFT size N should be large. The higher N, the better it is. e.g., $N > 10L$.
2. Make sure that ω_p and ω_s are on the FFT grid. $\left(\omega_k = \frac{2\pi k}{N},\ k = 0, 1, \dots, N-1\right)$.
3. Filter order parameter estimate: (lowpass and highpass) (Kaiser)

$$L \approx \frac{-20\log_{10}\sqrt{\delta_p \delta_s} - 13}{14.6(\omega_s - \omega_p)/2\pi}$$

   $\delta_p, \delta_s \searrow$ filter order $L \nearrow$
   $(\omega_s - \omega_p) \searrow$ filter order $L \nearrow$
   Start the iterations with Kaiser's estimate.
4. Here is how this iterative algorithm works: We assume that we have a space of impulse response sequences. we define two sets in this space:
   C_0 : set of frequency domain constraints. Any member of this set satisfies the frequency domain specs.
   C_1 : set of time domain constraints. Any member of this set is an FIR filter of order $2L+1$
   In Figure **??** , the intersection set $C_0 \cap C_1 = \{h^\star : \text{equiripple solution}\}$
   $\lim_{k\to\infty} h_k = h^\star$

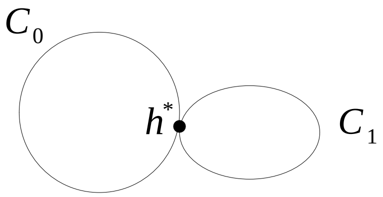


Fig. 6.1 Optimal solution case: The intersection set contains only the optimal solution.

5. Too loose constraints (δ_p, δ_s may be too large or L may be too large or $\omega_s - \omega_p$ too large). We have many solutions satisfying the constraints. $C_0 \cap C_1$ contains many solutions. Result of the iterative algorithm will not be equiripple! $\Longrightarrow$ either δ_p or $\delta_s \searrow$ or $L \searrow$ or $|\omega_s - \omega_p| \searrow$. Start the iterations once again.

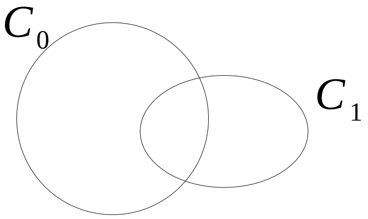


Fig. 6.2 Too loose constraints: intersection set contains many solutions to the filter design problem but the final filter obtained by iterative process may not be optimal.

6. Too tight constraints: Too small L or too small δ_p, δ_s. No solution! In this case, the iterates still converge to a solution $\lim_{k\to\infty} h_k = h_f$ but $H_f(e^{j\omega})$ does not satisfy the frequency domain specs. In this case, either $L \nearrow$ or $\delta_p, \delta_s \nearrow$ or $|\omega_s - \omega_p| \nearrow$ to enlarge the sets such that they have non-empty intersection.

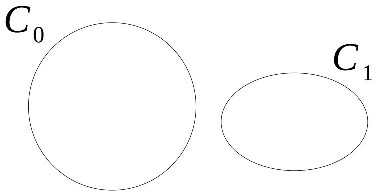


Fig. 6.3 No solution case: Constraint sets are too tight.

7. a. We should compute the DFT of a two-sided sequence because the zero-phase FIR filters have symmetric coefficients with respect to $n = 0$:

$$h[n] = \left\{ \frac{1}{3}, \frac{1}{8}, \underbrace{\frac{1}{2}}_{n=0}, \frac{1}{8}, \frac{1}{3} \right\} \xrightarrow{DFT} H[k] = ?$$

Everything is periodic in DFT implementation. Take the DFT of $\tilde{h}[n] = \left\{ \underbrace{\frac{1}{2}}_{n=0}, \frac{1}{8}, \frac{1}{3} 0, \ldots, \frac{1}{3}, \underbrace{\frac{1}{8}}_{n=N-1} \right\}$
The DFT's are equal to each other:

$$\tilde{H}[k] = H[k]$$

.

b. There is a second way: Shift the anticausal $h[n]$ so that it is causal, then compute the DFT. Because $\tilde{h}[n] = \frac{1}{N}\sum_{k=0}^{N-1}\tilde{H}[k]e^{j\frac{2\pi k}{N}n}$ is periodic with period N. $\Longrightarrow \tilde{h}[n] = h[n]$ for $n = -N/2,\ldots,0,\ldots,N/2-1$

Ex: Time domain projection of $g_k = [a,b,c,0,0,\ldots,0,c,b]$ for $L = 1$ (meaning a third order FIR filter) is $h_{k+1}[n] = [a,b,0,0,\ldots,0,b]$.

8. Translate the frequency domain specs from $[-\pi,\pi]$ to $\omega \in [0,2\pi]$ or $k \in [0,1,\ldots,N-1]$ in the DFT domain because DFT samples $\omega \in [0,2\pi]$ range!

6.4 Exercises

1. Consider the analog system characterized by the differential equation:

$$\frac{dy_a}{dt} = -0.5y_a(t) + x_a(t) \tag{6.2}$$

where $y_a(t)$ is the output and $x_a(t)$ is the input with $BW = 2$ kHz.
(a) Use bilinear transform to approximate this system with discrete-time system. Obtain the I/O relation for the discrete-time system.
(b) Is the discrete-time system stable? Explain.
(c) Define $h[n] = T_s h_a(nT_s)$, $n = 0,1,2,\ldots$ where T_s is the sampling period and $h_a(t)$ is the impulse response of the system given in (**??**). Determine $H(e^{j\omega})$.
(d) Can you obtain a recursive system implementing the impulse response $h[n]$ that you obtained in part (c)? Obtain the I/O relation for this discrete-time system.
2. Consider the analog Butterworth filter

$$|H_a(j\Omega)|^2 = \frac{1}{1+(j\Omega/j\Omega_c)^2}$$

(a) Design a low-pass filter with cut-off normalized angular frequency $\omega_c = \pi/4$.
(b) Design a high-pass filter with cut-off normalized angular frequency $\omega_c = 3\pi/4$.
(c) Are the filters that you realized in part (a) and (b) stable filters? Prove your answer.
3. Consider the following window:

$$w[n] = \left\{\frac{1}{6},\frac{1}{6},\underbrace{\frac{2}{6}}_{n=0},\frac{1}{6},\frac{1}{6}\right\}$$

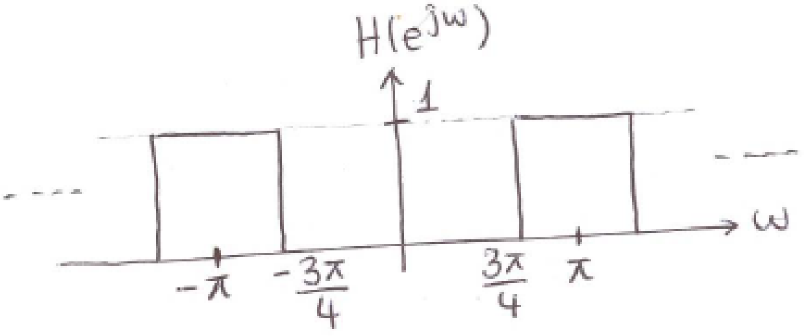
Design a 5-th order FIR filter with passband $PB : [3\pi/4,\pi] \cup [-\pi,-3\pi/4]$ using $w[n]$. This filter is a high-pass filter.
4. Consider the analog Butterworth filter

$$|H_a(j\Omega)|^2 = \frac{1}{1+(j\Omega/j\Omega_c)^2}$$

(a) Design a discrete-time IIR low-pass filter with 3dB cut-off at $\omega_c = \pi/2$.
(b) Design a discrete-time IIR high-pass filter with 3dB cut-off at $\omega_c = \pi/2$ using the above analog prototype filter.
5. Consider the input signal

$$x[n] = \left\{\ldots, 1, \underbrace{1}_{n=0}, 1, 1, 2, 2, 2, 2, \ldots\right\}$$

(a) Filter this signal using the low-pass filter you designed in Question 4.
(b) Filter this signal using the high-pass filter you designed in Question 4.
(c) Comment on filter outputs.
6. (a) Design a 3rd order FIR low-pass filter with cut-off frequency of $\omega_c = \pi/2$ by using any method you like.
(b) Obtain a 3rd order FIR high-pass filter from your design in part (a). Cut-off frequency of the high-pass filter is also $\omega_c = \pi/2$.
(c) Let $x[n] = \left\{\ldots, 1, \underbrace{1}_{n=0}, 1, 1, 2, 2, 2, 2, \ldots\right\}$. Filter $x[n]$ using your low-pass filter.
(d) Filter $x[n]$ using your high-pass filter.
(e) Comment on filter outputs in parts (c) and (d).
7. Let Design a third order FIR high-pass filter whose frequency response is shown

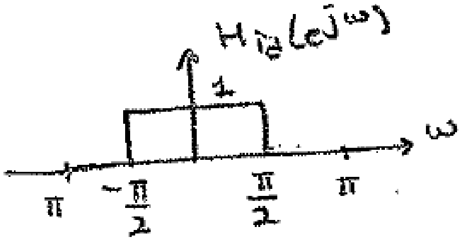


above . Use the triangular window method.
8. Consider the analog Butterworth filter

$$|H_a(j\Omega)|^2 = \frac{1}{1+(j\Omega/j\Omega_c)^2}$$

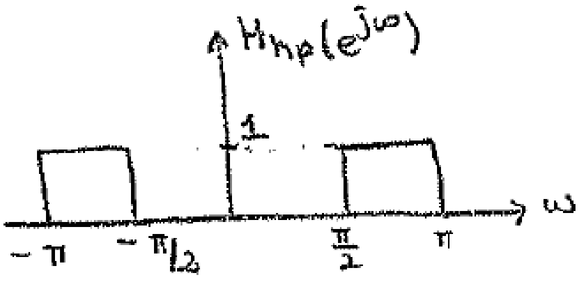
(a) Design a discrete-time IIR low-pass filter with cut-off $\omega_c = \pi/4$.
(b) Design a discrete-time IIR high-pass filter with cut-off $\omega_c = \pi/4$.

9. Given a low-pass filter $H_{id}(e^{j\omega})$ with cut-off frequency ω_c.
(a) Plot $H_{id}(e^{j\omega})$ in the range $[-4\pi, 4\pi]$.
(b) Compute the corresponding impulse response $h_{id}[n]$.
(c) Determine an FIR filter $h_a[n]$ of order $2L+1$, $L = 3$ as an approximation of $h_{id}[n]$ (rectangular windowing method). Specify the rectangular windowing function $w[n]$.
(d) Plot roughly the frequency response $H_a(e^{j\omega})$ of $h_a[n]$. Explain the differences between $H_{id}(e^{j\omega})$ and $H_a(e^{j\omega})$ and the cause of these differences as detailed as possible. Hint: use time-domain and frequency-domain plots or equations of the windowing function $w[n]$.
10. Design a low-pass Butterworth filter with $\omega_c = \pi/2$ and filter order $N = 1$.
(a) Determine the analog prototype filter $H_a(s)$.
(b) Determine all poles of $H_a(s)$.
(c) Apply bilinear transform to determine a stable digital filter $H(z)$.
(d) Determine the Region Of Convergence (ROC) of the digital filter.
11. Design a low-pass filter with these requirements:



$h_{id}[n] = ?$

12. Design a high-pass filter with these requirements:



$h_{hp}[n] = ?$

13. Let $y[n] = x[n-m]$, (a) which FIR filter yields this $h[n]$?, (b) compute $H(e^{j\omega})$.
14. Given $x[n] = \delta[n+1] + 2\delta[n] + \delta[n-1]$ and $y[n] = 3\delta[n-2] - 2\delta[n-3]$.
(a) Compute $z[n] = x[n] * y[n]$.
(b) Compute the 4-point DFT $X[n]$ of $x[n]$.
(c) Compute the 4-point DFT $Y[n]$ of $y[n]$.
(d) Compute $z[n]$ by DFT and compare your result with part (a).

Chapter 7
Recursive IIR Filter Design

In this section, we describe the design of recursive Infinite-extent Impulse Response (IIR) filters. To determine the current output sample the filter uses not only current and past input samples but also past output samples. That is why it is a recursive structure.

This chapter consists of two parts. We first describe approximate implementation of continuous-time systems using discrete-time processing. In the second part of the chapter we use the bilinear transform to design recursive IIR filters from analog (continuous-time) filters.

7.1 Implementation of Analog Systems using Discrete-Time Processing

Consider the analog system described by a first order constant coefficient linear differential equation:

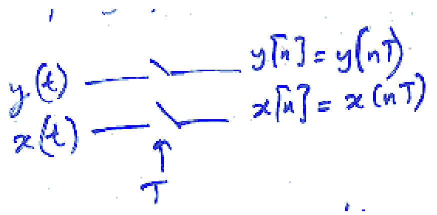
$$\frac{dy_c(t)}{dt} + ay_c(t) = x_c(t) \tag{7.1}$$

where y_c and x_c are continuous-time output and the input signals, respectively. To simulate this system using a digital computer we have to approximate the derivative $\frac{dy_c(t)}{dt}$:

$$\left.\frac{dy_c}{dt}\right|_{t=nT} \simeq \frac{y_c(nT) - y_c(nT-T)}{T} \tag{7.2}$$

where T is the sampling period.

As T goes to zero the above equation converges to the derivative. If x_c is a band-limited signal a good value for the sampling period is the Nyquist rate. We replace the derivative with the differencing operation given by (7.2) to obtain the difference equation approximating (7.1) as follows:



$$\frac{y[n]-y[n-1]}{T}+ay[n]=x[n] \qquad (2) \tag{7.3}$$

where the discrete-time signal $y[n]=y_c(nT), n=0,\pm 1,\pm 2,...$ and $x[n]=x_c(nT), n=0,\pm 1,\pm 2,...$ As a result we obtain the following discrete-time system

$$y[n]\left(\frac{1}{T}+a\right)=\frac{y[n-1]}{T}+x[n] \tag{7.4}$$

which has an infinite-extent impulse response. The above difference equation can be implemented using a computer or a digital signal processor in causal and recursive manner.

The above procedure can be used to simulate any differential equation in a computer. We, now, describe a transform domain method to establish the relation between the continuous-time system and the corresponding discrete-time system. The Laplace Transform of Eq.(7.1) is given by

$$sY_c(s)+aY_c(s)=X_c(s) \tag{7.5}$$

where X_c and Y_c are the Laplace transforms of x_c and y_c, respectively. The Z-Transform of (7.2) is given by

$$\frac{1-z^{-1}}{T}Y(z)+aY(z)=X(z) \tag{7.6}$$

where $X(z)$ and $Y(z)$ are the Z-transforms of the discrete-time signals x and y, respectively. By comparing Eq. 7.5 and 7.6 we observe that we can simply replace s by $\frac{1}{T}(1-z^{-1})$ to obtain the z-transform relation (7.6) of the difference equation (7.3) from the Laplace transform relation given in Eq. 7.5. This defines a transformation

$$s=\frac{1}{T}(1-z^{-1}) \tag{7.7}$$

or $z=\frac{1}{1-sT}$ between the complex s-plane and the complex z-plane.

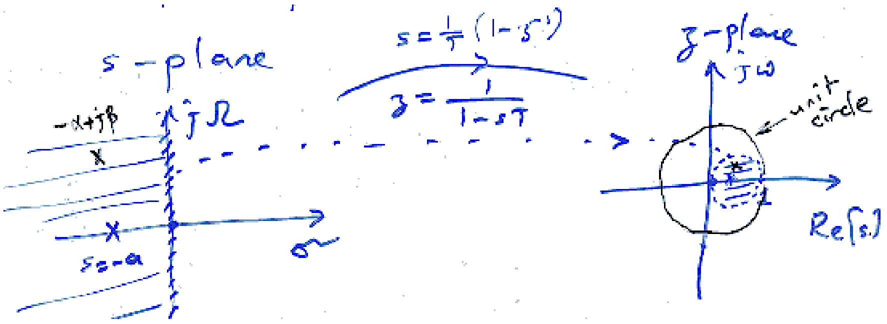
It is possible to generalize this transform domain approach. Given an analog tranfer function $H_c=\frac{Y_c(s)}{X_c(s)}$ it is possible to use Eq. 7.7 to obtain the transfer function $H(z)$ of the approximating difference equation as follows:

$$H_c(s) = \frac{\sum_{k=0}^{M} \beta_k s^k}{\sum_{k=0}^{N} \alpha_k s^k} \xrightarrow{s=\frac{1}{T}(1-z^{-1})} H(z) = \frac{\sum_{k=0}^{M} \beta_k \left(\frac{1-z^{-1}}{T}\right)^k}{\sum_{k=0}^{N} \alpha_k \left(\frac{1-z^{-1}}{T}\right)^k}$$

The discrete-time transfer function $H(z)$ approximates the continuous time transfer function $H_c(s)$ which is the transfer function of linear constant coefficient differential equation.

It is possible to analyze the transform using complex analysis. The transform $s = \frac{1}{T}(1-z^{-1})$ maps the left half plane (LHP) inside the unit circle. This is a good property because it means that a causal stable analog system is mapped to a causal stable discrete-time system. When the poles of $H_c(s)$ are in the LHP, the poles of $H(z)$ are inside the unit circle. Some examples are given in the following table:

	Cont-time	Discrete-time
Pole	$s=-a$	$z=\frac{1}{1+aT}<1$
	$s=0$	1
	$s=\infty+j0$	0
	$s=-\infty+j0$	0
Stable prototype	L.H.P.	circle centered at $(1/2,0)$ with radius $1/2$
	All poles must be in L.H.P. for a stable system	All poles must be inside the unit circle for a stable system



Unfortunately, this transformation is not effective because L.H.P. does not cover the inside of the unit circle! So it does not fully utilize the unit-circle in the z-domain. For example, $s = \pm\infty + j0$ is mapped to z=0 as shown in the above Table. It is not a bad transformation, either. Because a stable analog prototype system produces a stable difference equation. In the next section, we study the bilinear transform which is more efficient than $s = \frac{1}{T}(1-z^{-1})$.

7.2 The Bilinear Transform

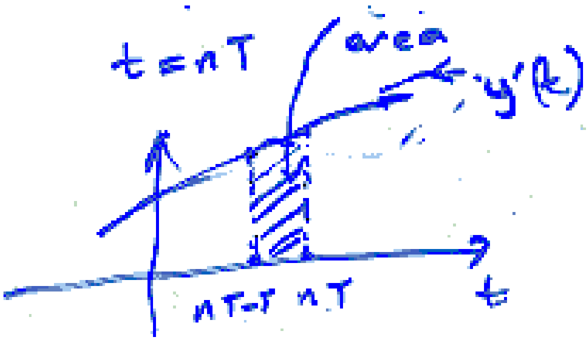
Bilinear transform uses the trapezoidal rule for integration to establish a relation between the s-plane and the z-plane. We consider the same analog system described by a first order constant coefficient linear differential equation once again:

$$\frac{dy_c(t)}{dt} + ay_c(t) = x_c(t) \qquad (7.1)$$

together with the integral relation

$$y_c(t)|_{t=nT} = \int_{nT-T}^{nT} y_c'(\tau)d\tau + y_c(nT-T) \qquad (7.8)$$

where $y_c'(t)$ is the derivative of $y_c(t)$ and T is the sampling period. Equation 7.8



follows from

$$y_c(t) = \int_{-\infty}^{t} y_c'(\tau)d\tau$$

$$y_c(t) = \int_{t_0}^{t} y_c'(\tau)d\tau + \int_{-\infty}^{t_0} y'(\tau)d\tau$$

$$y(t) = \int_{t_0}^{t} y'(\tau)d\tau + y(t_0)$$

When we set $t_0 = nT - T$ we obtain Equation (7.8). To approximate the area under the integral in Eq. 7.8 , we use the trapezoidal approximation:

$$\int_{nT-T}^{nT} y_c'(\tau)d\tau \approx \frac{T}{2}\left(y'(nT) - y'(nT-T)\right) \qquad (7.9)$$

Using Equation (7.9) Equation (7.8) can be approximated as follows

$$\begin{aligned} y_c(nT) = y[n] &\approx \frac{T}{2}\left(y'[n] - y'[n-1]\right) + y[n-1] \\ y_c'(nT) = y'[n] &= -ay(nT) + x(nT) \\ y_c'(nT-T) = y'[n-1] &= -ay(nT-T) + x(nT-T) \end{aligned}$$

We next use the right hand side of the differential equation 7.1 to replace the derivative terms and obtain a difference equation

$$y[n] \approx \frac{T}{2}\left(-ay[n] + x[n] - ay[n-1] + x[n-1]\right) + y[n-1]$$

In z-domain

$$\left(1 + \frac{aT}{2}\right)Y(z) - \left(1 - \frac{aT}{2}\right)Y(z)z^{-1} = \frac{T}{2}(1+z^{-1})X(z)$$

or

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right) + a}$$

The transfer function of the analog system given in Eq. 7.1 is

$$H_c(s) = \frac{1}{s+a}$$

To obtain the transfer function of the corresponding discrete-time system we replace s by the so-called bilinear transformation:

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

and obtain $H(z)$ from $H_c(s)$.

This can be generalized to any linear constant coefficient differential equation with a transfer function $H_a(s)$. It is possible to obtain a discrete-time system approximating the analog system as follows

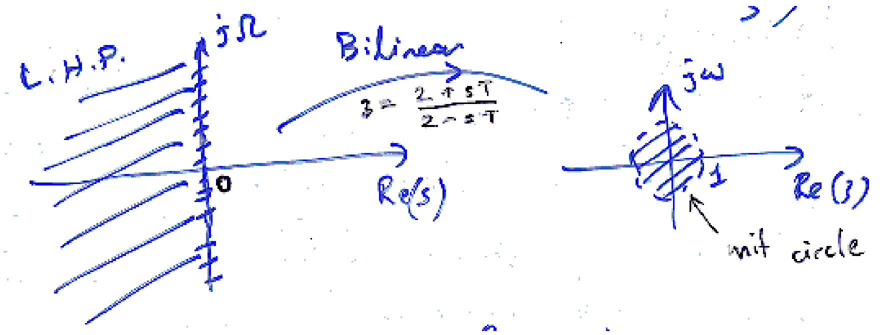
$$H(z) = H_a(s)\big|_{s=\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}$$

where T is the sampling period.

The bilinear transformation

$$z = \frac{2+sT}{2-sT} \qquad (7.10)$$

maps the left half plane (LHP) in s-domain to unit circle centered at the origin in z-domain. The LHP covers the entire unit disc therefore poles of the analog system can go to anywhere in the unit circle. Therefore, an analog stable system or a filter is mapped to a stable discrete-time system. As a result this is a more efficient transfor-



mation than the transformation that we studied in Section 7.1. Furthermore, it maps the $j\Omega$ axis onto unit circle. This follows from the previous Equation 7.x. When we set $s = j\Omega$ in Eq. (**??**) we get

$$|z| = |(2 + j\Omega T)/(2 - j\Omega T)| = 1$$

for all Ω. The origin s=0 goes to z=1 and $s = \pm j\infty$ go to $z = -1$. We can obtain an analytic relation between Ω and ω by using the relation

$$e^{j\omega} = (2 + j\Omega T)/(2 - j\Omega T) \qquad (7.11)$$

which we will use in the next section in recursive IIR filter design.

7.3 IIR Filter Design using the Bilinear Transform

The bilinear transform can not only be used for simulating discrete-time systems but also to design recursive IIR filters.

As pointed out in the previous section the transform

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

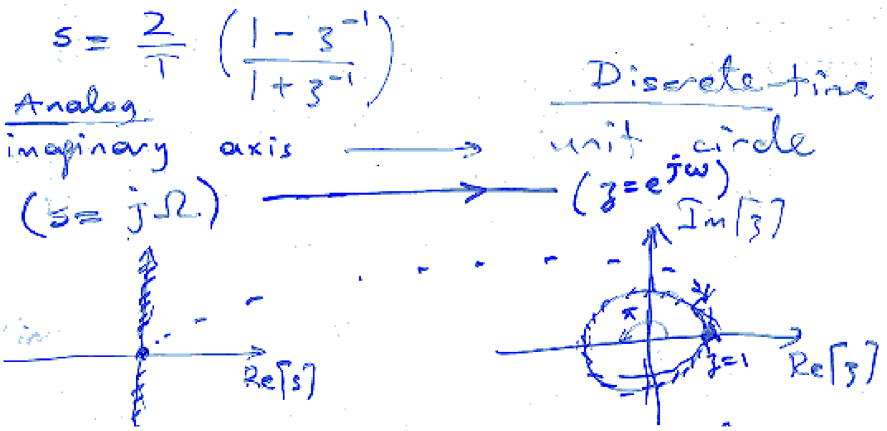
maps the imaginary axis $s = j\Omega$ onto the unit circle $z = e^{j\omega}$ in discrete time domain.

Using Eq. (**??**), we obtain a relation between the normalized angular frequency and the actual angular frequency as follows

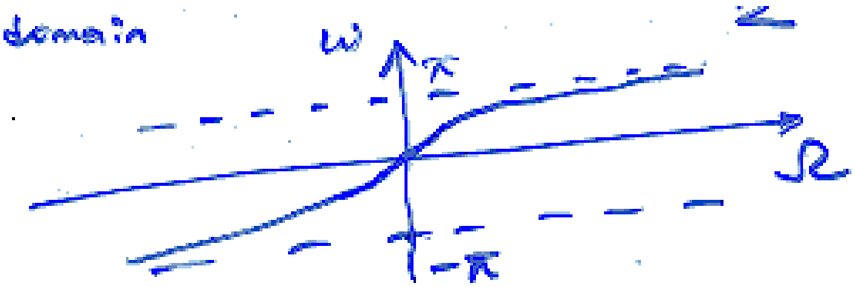
$$\omega = 2\tan^{-1}\frac{\Omega T}{2} \qquad (7.12)$$

or

$$\Omega = \frac{2}{T}\tan\frac{\omega}{2}$$



.



In the following example we use the above equations to design a recursive IIR filter using an analog prototype $H_a(s)$.

Example: Given the transfer function

$$H_a(s) = \frac{\Omega_c}{s + \Omega_c}$$

of a low-pass filter with 3dB BW at Ω_c. Design a discrete-time low-pass filter with 3dB cut-off at the normalized angular frequency $\omega_c = 0.2\pi$.

We use Equation (**??**) to obtain the corresponding 3dB cut-off frequency Ω_c as follows.

$$\Omega_c = \frac{2}{T}\tan\frac{\omega_c}{2} = \frac{2}{T}\tan(0.1\pi) \approx \frac{0.65}{T} \tag{7.13}$$

which determines the analog prototype system

$$H_a(s) = \frac{0.65/T}{s + 0.65/T}$$

This filter is stable because its pole $s = -0.65/T$ is in the LHP for all T values. In filter design we can set $T = 1$ without loss of generality because we do not try to simulate an actual continuous-time system in discrete-time domain.

Once we have the analog prototype system we can determine the corresponding discrete-time system using the bilinear transform as follows

$$H(z) = H_a(s)\Big|_{s=\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}$$

The transfer function of the IIR discrete-time filter is given by

$$H(z) = \frac{0.65/T}{\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right) + 0.65/T} = \frac{0.245(1+z^{-1})}{1-0.509z^{-1}}$$

and the corresponding frequency response is given by

$$H(e^{j\omega}) = \frac{0.245(1+e^{-j\omega})}{1-0.509e^{-j\omega}}$$

We obtain the time-domain recursive filter from the transfer function by setting

$$\frac{Y(z)}{X(z)} = H(z)$$

and

$$Y(z)(1-0.509z^{-1}) = 0.245X(z)(1+z^{-1})$$

which produces the time-domain relation

$$y[n] - 0.509y[n-1] = 0.245(x[n] + x[n-1])$$

This recursive filter turns out to be stable because its pole $z = 0.509$ is inside the unit circle and it is straightforward to implement it in a computer or a digital signal processor using the relation

$$y[n] = 0.509y[n-1] + 0.245(x[n] + x[n-1]) \tag{7.14}$$

in a causal manner.

The impulse response of this filter is of infinite-extent:

$$y[0] = h[0] = 0.509 \times 0 + 0.245(\delta[n] + \delta[n-1]) = 0.245, h[1] = 0.509 \times 0.245 + 0.245,$$

$$h[2] = 0.509h[1], h[3] = 0.509h[2], ..$$

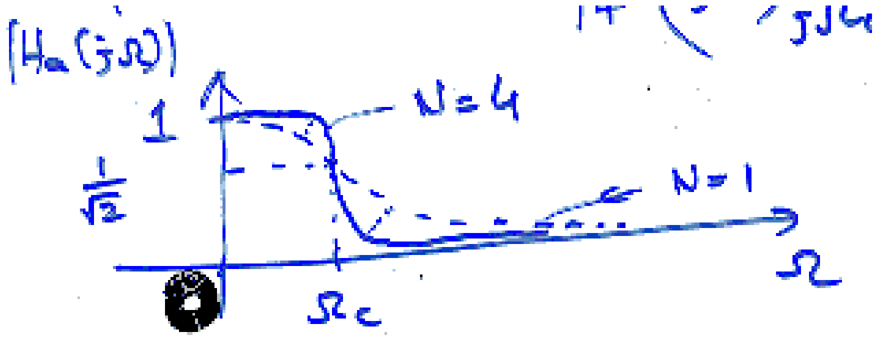
Do not try to use the convolution sum to implement this filter, because it is computationally inefficient compared to the recursive I/O relation given in Eq. **??**.

7.4 Butterworth Filters

The magnitude response of analog Butterworth filters are given by the following relation

$$|H_a(j\Omega)|^2 = \frac{1}{1+(j\Omega/j\Omega_c)^{2N}} = H_a(j\omega)H_a^*(j\omega)$$

where Ω_c is the 3dB cut-off frequency and N is the order of the filter.



We have to determine $H_a(s)$ from the above expression or from the following equation

$$H_a(s)H_a(-s) = \frac{1}{1+(s/j\Omega_c)^{2N}}$$
$$H_a(s)H_a(-s)|_{s=j\Omega} = H_a(j\Omega)H_a^*(j\Omega)$$

for each specific case. The poles of $H_a(s)H_a(-s)$ are $s_k = \Omega_c e^{j\frac{\pi}{2N}(2k+N-1)} = (-1)^{1/2N}j\Omega_c, \quad k = 0,1,2,\ldots,2N-1$. They are located on the circle $|s| = \Omega_c$ and N of them are on the LHP and the remaining ones are on the right half plane (RHP). Among these poles we pick the ones in the left half plane to construct $H_a(s)$ because we want to have a stable analog system to start with. For example for N=3 we pick the poles s_a, s_b and s_c to form

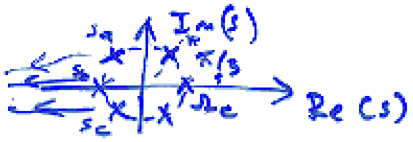
$$H_a(s) = \frac{K}{(s-s_a)(s-s_b)(s-s_c)}$$

Once we have the above analog system we can design $H(z)$ from $H_a(s)$ using the bilinear transformation:

$$H(z) = H_a(s)|_{s=\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}$$

We can determine the gain K when $s = 0$ or $\Omega = 0$,

For N=3
poles of H(s)



RHP poles are
the " of Ha(-j

$$H_a(j0) = 1$$

$$H_a(j0) = \frac{K}{(-s_a)(-s_b)(-s_c)} = 1$$

$$\Rightarrow K = [(-s_a)(-s_b)(-s_c)]$$

so that $H_a(j0) = 1$.

$$|H_a(j\Omega)|^2 = \frac{1}{1+(j\Omega/j\Omega_c)^{2N}}, \quad H_a(s)H_a(-s) = \frac{1}{1+(s/j\Omega_c)^{2N}}$$

Low-pass filter with cut-off at Ω_c

$$\underset{\text{Select the poles in the L.H.P.}}{H_a(s)} \xrightarrow[s=\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)]{\text{Bilinear Transform}} \underset{\text{Poles in the unit circle}}{H(z)}$$

Ex:

$$1 \geq \left|H(e^{j\omega})\right| \geq 0.89125(-1dB) \qquad 0 \leq \omega \leq 0.2\pi,\ (P.B.)$$

$$\left|H(e^{j\omega})\right| \leq 0.17783(-15dB) \qquad 0.3\pi \leq \omega \leq \pi,\ (S.B.)$$

Use bilinear transformation and Butterworth filter to design the filter. Actual frequency $\Omega = \frac{2}{T}\tan\frac{\omega}{2}$ where ω is the normalized frequency used in D.T. domain.

$$1 \geq |H_a(j\Omega)| \geq 0.89125 \quad 0 \leq \Omega \leq \frac{2}{T}\tan\frac{0.2\pi}{2}$$

$$|H_a(j\Omega)| \leq 0.17783 \quad \frac{2}{T}\tan\frac{0.3\pi}{2} \leq \omega \leq \infty$$

Let $T = 1$, $|H(j\Omega)| = \sqrt{\frac{1}{1+(\Omega/\Omega_c)^{2N}}}$

$$\left.\begin{array}{l} 1+\left(\dfrac{2\tan 0.1\pi}{\Omega_c}\right)^{2N} = \left(\dfrac{1}{0.89}\right)^2 \\ 1+\left(\dfrac{2\tan 0.15\pi}{\Omega_c}\right)^{2N} = \left(\dfrac{1}{0.178}\right)^2 \end{array}\right\} \Longrightarrow N = 5.3046, \text{N must be an integer } N = 6$$

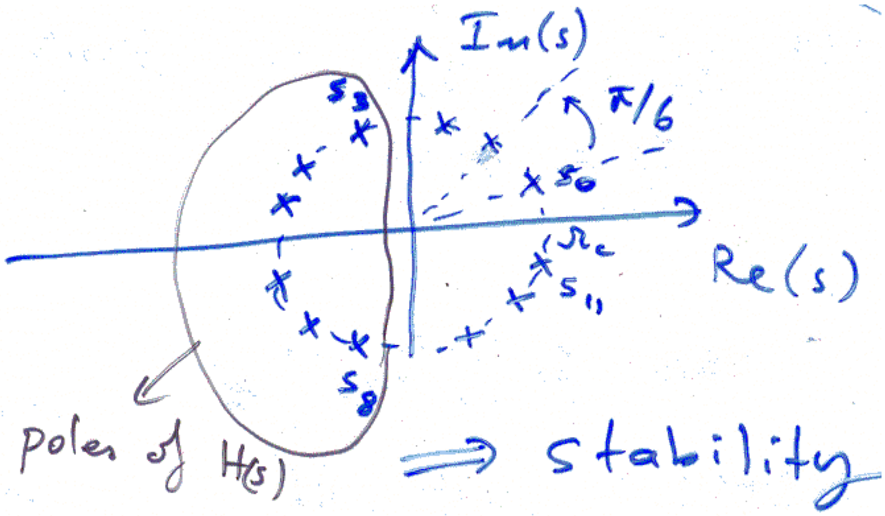
$$N = \frac{\log\left(((1/0.17)^2-1)/((1/0.89)^2-1)\right)}{2\log\left(\tan(0.15\pi)/\tan(0.1\pi)\right)} = 5.3046\ldots$$

Substitute $N = 6$ to find $\Omega_c = 0.7662$. Analog prototype:

$$H_a(s)H_a(-s) = \frac{1}{1 + (s/j0.7662)^{12}}$$

12 poles.

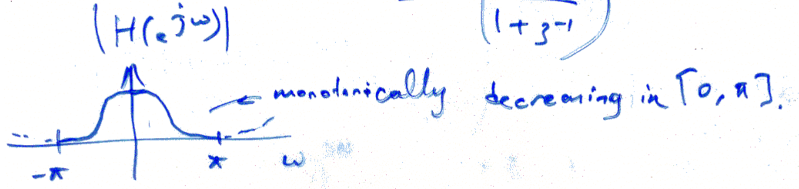
$$s_k = \Omega_c e^{j\frac{\pi}{12}(2k+N-1)} \qquad k = 0, 1, 2, \ldots, 11$$



$$H(s) = \frac{C}{(s-s_3)(s-s_4)\cdots(s-s_8)}$$
$$H(j0) = 1$$

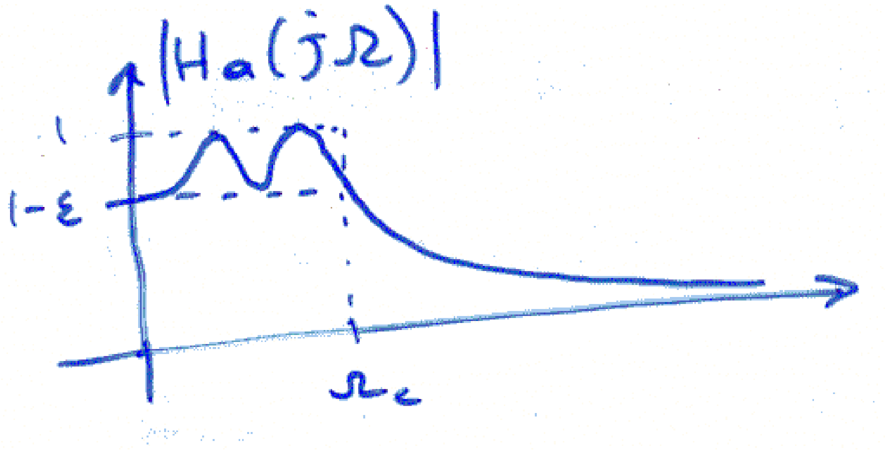
gives you C. Discrete-time IIR filter:

$$H(z) = H(s)\Big|_{s = 2\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}$$



$$H(z) = H(s)|_{s=2\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}$$

7.5 Chebyshev Filters



Analog low-pass Chebyshev filters have

- equiripple response in the passband,
- monotonically decreasing response in the stopband, and
- they usually lead to lower order filter than Butterworth.

We have a compact formula for the magnitude response

$$|H_a(j\Omega)|^2 = \frac{1}{1+\varepsilon^2 V_N^2(j\Omega/j\Omega_c)}$$

where $V_N(\theta) = \cos(N\cos^{-1}\theta)$, which is the N-th order Chebyshev polynomial.

Chebyshev polynomials have a recursive formula. The first three Chebyshev polynomials are given by

$$\begin{aligned} N = 0, &\quad V_0(\theta) = 1 \\ N = 1, &\quad V_1(\theta) = \cos(\cos^{-1}\theta) = \theta \\ N = 2, &\quad V_2(\theta) = \cos(2\cos^{-1}\theta) = 2\theta^2 - 1 \end{aligned}$$

Higher order Cheybyshev polynominals can be obtained from lower order ones using the recursive relation:

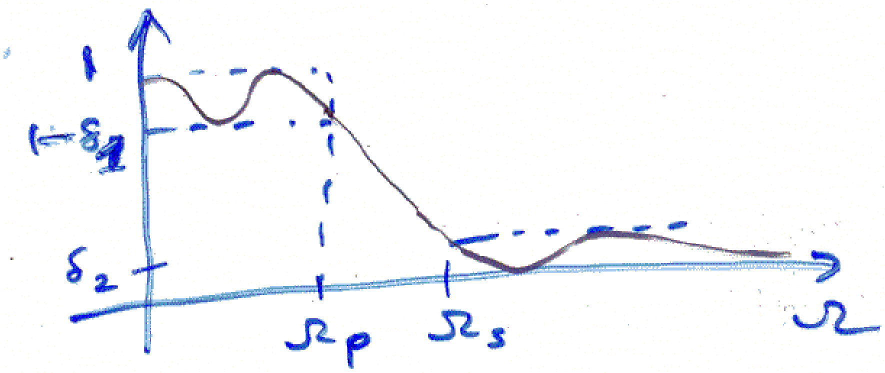
$$V_{N+1}(\theta) = 2\theta V_N(\theta) - V_{N-1}(\theta)$$

Example: Previous example frequency domain requirements, use analog Chebyshev prototype:

$$20\log|H_a(j0.2\pi)| \geq -1 \text{ at the p.b.}$$
$$20\log|H_a(j0.3\pi)| \leq -15 \text{ at the s.b.}$$
$$\Longrightarrow N = 4$$

Analog prototype filter turns out to have 4 poles. This leads to a lower order discrete-time filter. However, phase response of this filter is not as linear as the Butterworth filter.

7.6 Elliptic Filters



Analog elliptic filters are usually the

- lowest order filters satisfying the frequency domain requirements,
- they have sharp transition bands,
- they are equiripple both in the passband and stopband, but
- they have nonlinear phase response.

Magnitude response of an analog elliptic filter has the following compact formula:

$$|H_a(j\Omega)|^2 = \frac{1}{1+\varepsilon^2 V_N^2(\Omega)}$$

where V_N is a Jacobian Elliptic function.

Example: Frequency domain specs are the same as the previous example. In this case the analog prototype filter has only 3 poles. Therefore, the discrete-time elliptic filter turn out to be the lowest order filter achieving the specs with the least amount of multiplications per output sample. However, this filter has the worst phase response compared to Butterworth and the Chebyshev filters that we designed in this chapter.

In early days of DSP computers and digital signal processors were not as powerful as todays computers. Therefore saving a couple of multiplications per output sample was important. However, the computational gain that we achieve by using a lower order IIR recursive filter is not that important today in many applications.

7.7 Phase Response of Recursive IIR Filters

Recursive IIR filters cannot have zero-phase or linear phase as FIR filters. This is an important disadvatage compared to FIR filters because human eye is very sensitive to phase distortions in images and video. Human ear is less sensitive to phase distortions in audio and speech. In both image processing and speech and audio processing we should try not to distort the phase of the input signal while performing filtering.

Let us prove the following statement. Statement 1: Why do we have nonlinear phase response in IIR filters (recursive filters)?

In ideal low-pass, high-pass, band-pass and band-stop filters the ideal impulse response is symmetric with respect to n=0 and it extends from infinity to +infinity. It is very easy to have FIR filters satisfying this condition which is called the zero-phase condition:

$$h[n] = h[-n], \qquad n = 0, 1, \ldots, L.$$

The DTFT of $h[n]$ is real that is why the condition is called the zero-phase condition. If a filter obeys this rule it does not distort the phase of the input signal.

Consider the following FIR low-pass filter $h[n] = \left\{ -1/32, 0, 9/32, \underbrace{1/2}_{n=0}, 9/32, 0, -1/32 \right\} \longleftarrow$ anti-causal.

Its DFT is real: $H(e^{j\omega}) = 1/2 + 9/32\left(e^{-j\omega} + e^{j\omega}\right) - 1/32\left(e^{-3j\omega} + e^{j3\omega}\right)$

We can construct a causal filter from this zero-phase filter which turns out to have linear-phase: (FIR - Causal) $h_{lp}[n] = h[n-L] = \left\{ \underbrace{-1/32}_{n=0}, 0, \ldots, -1/32 \right\}$, $L = 3$.

$H_{lp}(e^{j\omega}) = H(e^{j\omega})e^{-j\omega 3} \longleftarrow$ linear phase.

- Causal & Stable

  IIR filters have infinite-extent impulse response. $h[n] = \left\{ \underbrace{h[0]}_{n=0}, h[1], h[2], \ldots \right\}$

  Symmetry is impossible to achieve $\Longrightarrow$ No linear phase.

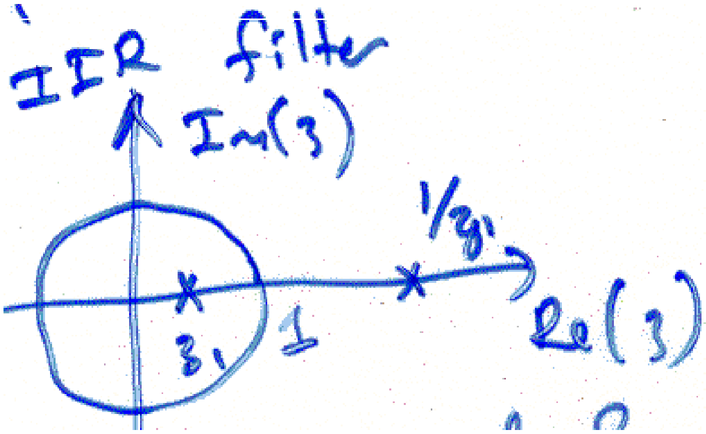
- In Butterworth and Chebyshev filters, we have almost linear phase response in the passband.
- Elliptic filters have bad phase response. Don't use it in image processing.

Proof of Statement 1: Z-domain analysis:
Zero-phase condition: $h[n] = \pm h[-n]$. In FIR filters,

$$\begin{aligned} H(z) &= h[-L]z^{L} + \cdots + h[0] + \cdots + h[L]z^{-L} \\ H(z^{-1}) &= h[-L]z^{-L} + \cdots + h[0] + \cdots + h[L]z^{L} \\ H(z) &= \pm H(z^{-1}) \end{aligned}$$

If you have z_1 as the root (pole) of $H(z)$, then $\frac{1}{z_1}$ must be also a root (pole) of $H(z)$.

Consider an IIR filter:



If the filter is causal and stable, this is not possible because you violate the stability condition.

7.8 Implementation of Recursive IIR Filters

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} = \underbrace{H_1(z)}_{\text{all zero part}} \underbrace{H_2(z)}_{\text{all pole part}}$$

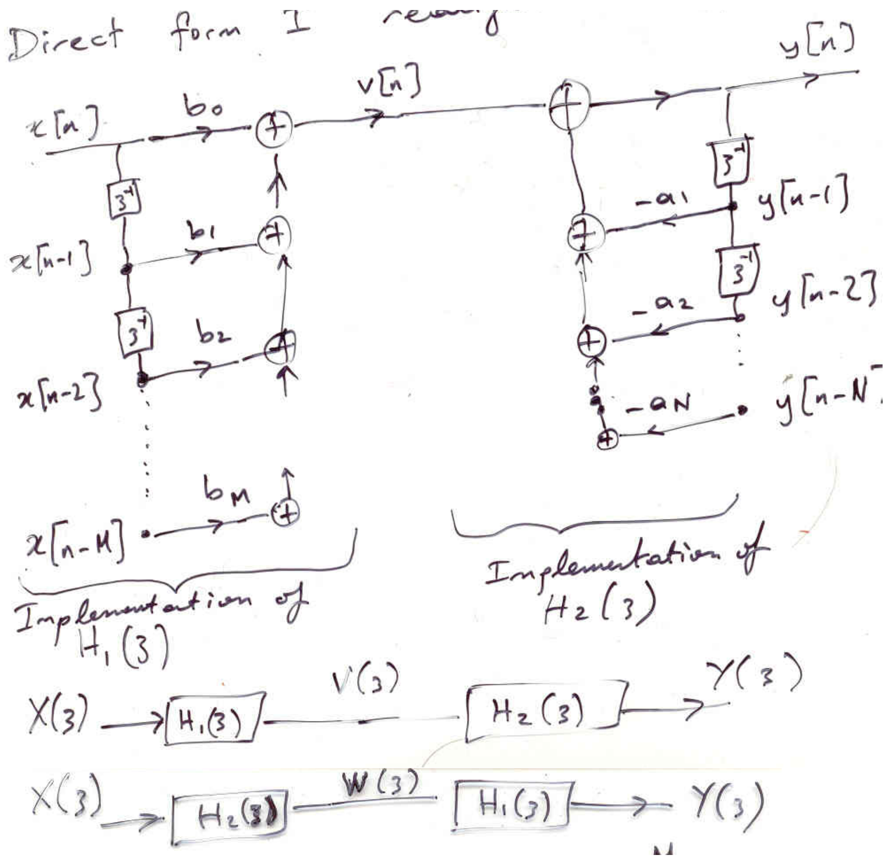
$$Y(z) + \sum_{k=1}^{N} a_k Y(z) z^{-k} = \sum_{k=0}^{M} b_k z^{-k} X(z)$$

Recursive I/O:

$$\underbrace{y[n]}_{\text{current output sample}} = -\sum_{k=1}^{N} a_k \underbrace{y[n-k]}_{\text{past output samples}} + \sum_{k=0}^{M} b_k \underbrace{x[n-k]}_{\text{current and past input samples}}$$

Memory requirement: $N+M$ registers
Computational requirement: $N+M+1$ multiplications per output; $N+M$ additions per output

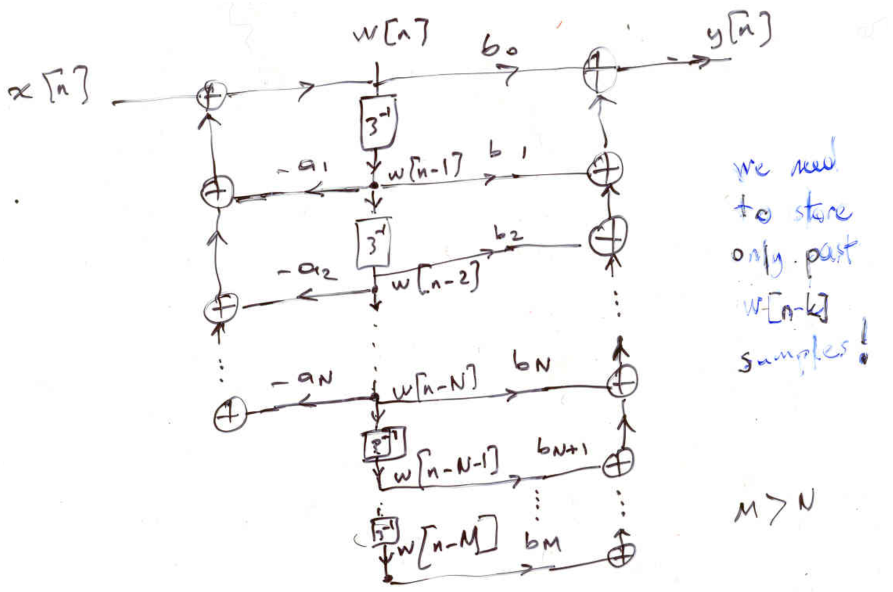
7.8.1 Direct Form I Realization (Signal Flow Diagram):



$$W(z) = \frac{1}{1+\sum_{k=1}^{N} a_k z^{-k}} X(z) \qquad Y(z) = \sum_{k=0}^{M} b_k z^{-k} W(z)$$

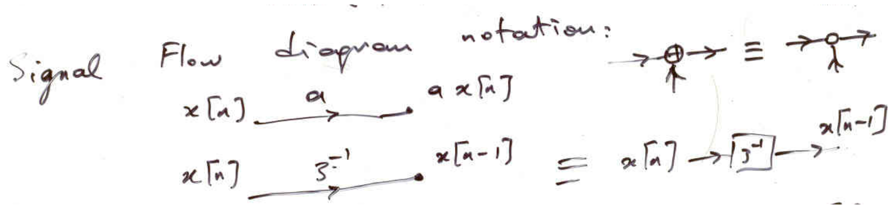
$$w[n] = -\sum_{k=1}^{N} a_k w[n-k] + x[n] \qquad y[n] = \sum_{k=0}^{M} b_k w[n-k]$$

We need to store only past $w[n-k]$ samples!

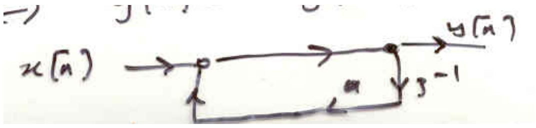


7.8.2 Direct Form II Realization (Signal Flow Diagram):

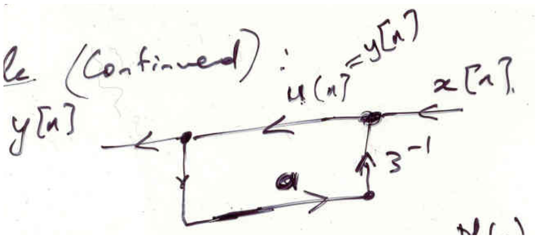
Total cost: $N+M+1$ multiplications; $N+M$ additions; $\max(N,M) < N+M$ memory locations. *Ex:*



$$H(z) = \frac{1}{1-az^{-1}} \Leftrightarrow y[n] = ay[n-1] + x[n]$$



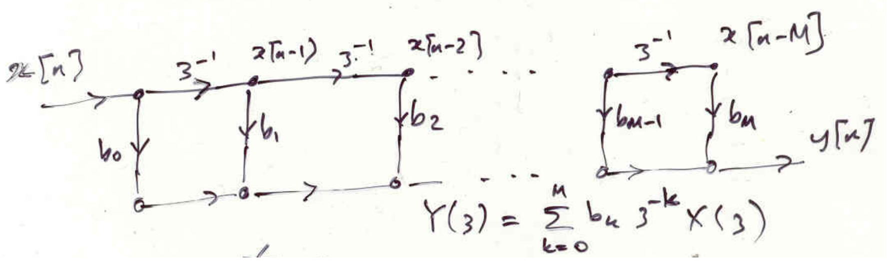
Flow-Graph Reversal Theorem: Change the direction of arrows and interchange the input and the output in a graph ⇒ Transfer function remains the same!
Ex (continued):



$$Y(z) = X(z) + az^{-1}Y(z) \Longrightarrow \frac{Y(z)}{X(z)} = \frac{1}{1-az^{-1}}$$

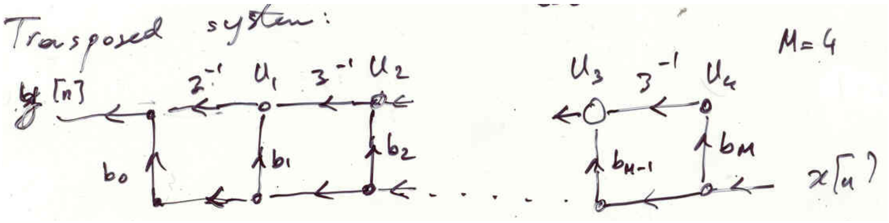
Ex: FIR Filter: $y[n] = \sum_{k=0}^{M} b_k x[n-k] \Longrightarrow Y(z) = \sum_{k=0}^{M} b_k z^{-k} X(z)$

$$H(z) = \sum_{k=0}^{M} b_k z^{-k} \text{ or } h[n] = \begin{cases} b_n & n = 0,1,\ldots,M \\ 0 & \text{ow.} \end{cases}$$



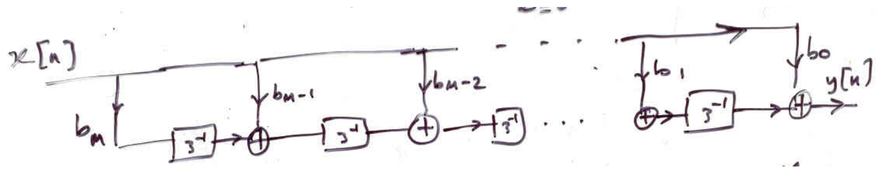
Transposed system:

$$U_1(z) = z^{-1}U_2(z) + b_1X(z), \qquad U_2(z) = z^{-1}U_3(z) + b_2X(z)$$
$$Y(z) = z^{-1}U_1(z) + b_0X(z), \qquad U_3(z) = z^{-1}U_4(z) + b_3X(z)$$
$$U_4(z) = b_4X(z)$$



7.8.3 Lattice Filter:

$Y(z) = \sum_{k=0}^{M} b_k z^{-k} X(z)$



It is more robust to quantization effects compared to direct realization.
Given an IIR filter, it is better to realize it using 2nd order systems:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} = H_1(z) \cdot H_2(z) \cdots H_K(z)$$

where

$$H_i(z) = \frac{b_{0i} + b_{1i}z^{-1} + b_{2i}z^{-2}}{1 + a_{1i}z^{-1} + a_{2i}z^{-2}}, \quad i = 1, 2, \ldots, K$$

Realize $H_1, H_2, \ldots, H_K$ and cascade them to get a computationally more efficient structure.

7.9 IIR All-Pass Filters

Recursive all-pass filters have the interesting property that their magnitude response is flat for all frequencies. An all-pass filter has the following transfer function:

$$H_{ap}(z) = \frac{z^{-1} - a^*}{1 - az^{-1}} \tag{7.15}$$

where a^* denotes the complex conjugate of a.

$$H_{ap}(e^{j\omega}) = \frac{e^{-j\omega} - a^*}{1 - ae^{-j\omega}} = e^{-j\omega}\frac{1 - a^* e^{j\omega}}{1 - ae^{-j\omega}}$$

Since $1 - a^* e^{j\omega}$ is the complex conjugate of $1 - ae^{-j\omega}$

$$\left|H_{ap}(e^{j\omega})\right| = \underbrace{\left|e^{-j\omega}\right|}_{=1} \frac{\left|1 - a^* e^{j\omega}\right|}{\left|1 - ae^{-j\omega}\right|} = 1 \text{ , for all } \omega.$$

Therefore the magnitude response of the all-pass filter is equal to 1 for all frequency values. However, the phase response turns out to be non-linear. All-pass filters can be used for altering the phase response of their inputs. They are also used in bilinear transforms to design bandpass, bandstop and high-pass filters as discussed in Section 7.10.

7.9.1 Input/Output relation

Time domain I/O relation of the all-pass filter can be obtained from the transfer function as follows

$$\frac{Y(z)}{X(z)} = \frac{z^{-1} - a^*}{1 - az^{-1}} \Longrightarrow y[n] - ay[n-1] = -a^* x[n] - x[n-1]$$

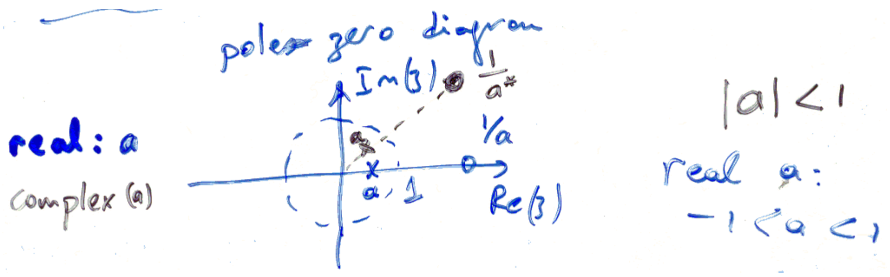
If a is real, we have a real filter.

By concatening first order all-pass sections we can build higher order all-pass filters.

7.9.2 Stability condition

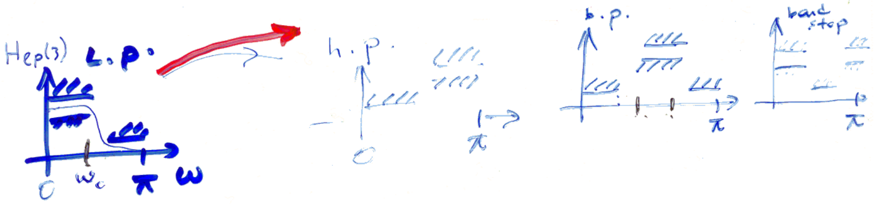
Since all-pass filters are recursive filters we have to satisfy the stability requirements. The pole of the tranfer function given in Eq. **??**)is a. Therefore $|a| < 1$ for stability.

N-th order All-pass filter:



$$G(z) = \prod_{i=1}^{N} \left(\frac{z^{-1} - a_i^*}{1 - a_i z^{-1}} \right), \quad |G(e^{j\omega})| = \prod_{i=1}^{N} \overbrace{\left| \frac{z^{-1} - a_i^*}{1 - a_i z^{-1}} \right|}^{=1} = 1$$

7.10 Frequency Transformation of Low-Pass IIR Filters to Other Filters



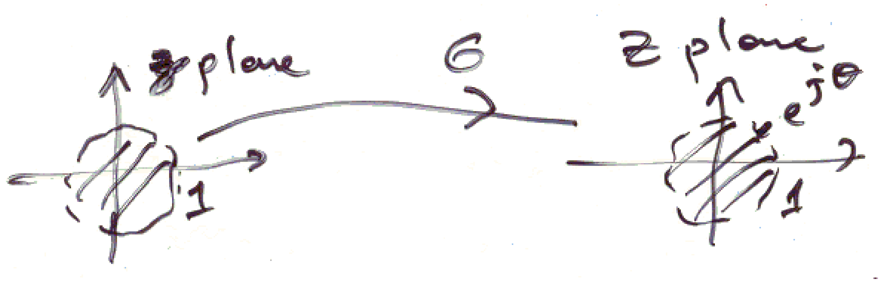
Let $H_{LP}(z)$ be given. It is possible to obtain another low-pass, band-pass, band-stop or a high-pass filter $H_d(z)$ from $H_{LP}(z)$. We can use a transformation $z^{-1} = G(z^{-1})$ or $z^{-1} = G^{-1}(z^{-1})$ to design new recursive filters from $H_{LP}(z)$:

$$H_d(z) = H_{LP}(z)|_{z^{-1}=G(z^{-1}) \text{ or } z^{-1}=G^{-1}(z^{-1})}$$

We wish to have:

1. $G(z^{-1})$ to be a rational function of z^{-1} (So that $H_d(z)$ is also a recursive IIR filter.)
2. To preserve stability
3. Unit circle must be mapped onto unit circle.

(3) $\Longrightarrow \quad e^{-j\theta} = G(e^{-j\omega}) = |G(e^{-j\omega})|e^{j\angle G(\omega)}$. Compute the magnitude of both sides:



$(3) \Longrightarrow \quad |G(e^{-j\omega})| = 1,\ \theta = -\angle G(\omega)$ for all ω.
$(1)\&(3) \Longrightarrow G$ is an all-pass filter.
In general,

$$z^{-1} = G(z^{-1}) = \pm \prod_{k=1}^{N} \left(\frac{z^{-1} - \alpha_k^*}{1 - \alpha_k z^{-1}} \right)$$

$(2) \Longrightarrow |\alpha_k| < 1$

7.10.1 Low-pass filter to low-pass filter

We can use the following transformation to obtain a new low-pass filter from the prototype lowpass filter :

$$z^{-1} = \frac{z^{-1} - \overbrace{\alpha}^{\text{real}}}{1 - \alpha z^{-1}} \text{ or } e^{-j\theta} = \frac{e^{-j\omega} - \alpha}{1 - \alpha e^{-j\omega}}$$

where $-1 < \alpha < 1$. The cut-off frequency of the new filter will be different from the prototype filter. The relation between the new angular frequency ω and the old angular frequency θ are given according to the following relation:

$$\omega = \arctan \left(\frac{(1 - \alpha^2) \sin \theta}{2\alpha + (1 + \alpha^2) \cos \theta} \right) \tag{7.16}$$

The bilinear transformation warps the frequency scale. To transfer the cut-off frequency from θ_p to ω_p, use the following α value

$$\alpha_0 = \frac{\sin((\theta_p - \omega_p)/2)}{\sin((\theta_p + \omega_p)/2)}$$

which follows from Eq. (**??**). The transfer function of the new filter is obtained using the bilinear transform:

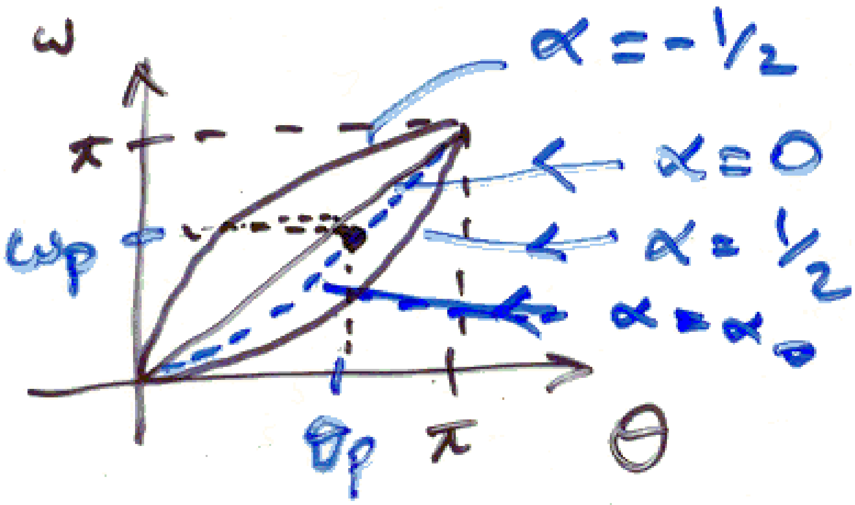


Fig. 7.1 The relation between ω and θ.

$$H_d(z) = H_{LP}(z)|_{z^{-1}=(z^{-1}-\alpha_0)/(1-alpha_0 z^{-1})}$$

7.10.2 Low-pass to high-pass transformation

Similar to the previous case, it is possible to obtain a high-pass filter from a low-pass filter using the bilinear transformation:

$$z^{-1} = -\frac{z^{-1}+\alpha}{1+\alpha z^{-1}},$$

where $-1 < \alpha < 1$ and given by

$$\alpha = -\frac{\cos((\theta_p + \omega_p)/2)}{\cos((\theta_p - \omega_p)/2)}$$

where ω_p is the cut-off frequency of the high-pass filter and θ_p is the cut-off frequency of the low-pass filter.

7.10.3 Low-pass to band-pass filter

To obtain a band-pass filter from a prototype low-pass filter with cut-off frequency θ_p we use a second-order all-pass section to transform the low-pass filter:

$$z^{-1} = -\left(\frac{z^{-2} - \frac{2\alpha k}{k+1} z^{-1} + \frac{k-1}{k+1}}{\frac{k-1}{k+1} z^{-2} - \frac{2\alpha k}{k+1} z^{-1} + 1} \right) = G(z^{-1})$$

where

$$\alpha = \frac{\cos\left((\omega_{p2} + \omega_{p1})/2\right)}{\cos\left((\omega_{p2} - \omega_{p1})/2\right)}, \quad k = \cot\left((\omega_{p2} - \omega_{p1})/2\right) \tan\left(\theta/2\right)$$

where ω_{p1} is the lower cut-off frequency and ω_{p2} is upper cut-off frequency of the band-pass filter, respectively.

7.10.4 Low-pass to band-stop filter

To obtain a band-stop filter from a prototype low-pass filter with cut-off frequency θ_p we also use a second-order all-pass section to transform the low-pass filter:

$$z^{-1} = -G(z^{-1}) = \left(\frac{z^{-2} - \frac{2\alpha k}{k+1} z^{-1} + \frac{k-1}{k+1}}{\frac{k-1}{k+1} z^{-2} - \frac{2\alpha k}{k+1} z^{-1} + 1} \right)$$

where alpha and k are the same as the band-pass filter case described in Sec. 7.10.3.

7.11 Exercises

1. Show that the following is an all pass filter for real b_i:

$$H(z) = \frac{z^{-L} + b_1 z^{-L+1} + \cdots + b_L}{1 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_L z^{-L}}$$

2. Transformation $Z = -z$ maps a low-pass filter to a high-pass filter. Let

$$h_{lp}[n] = \mathscr{Z}^{-1}\left\{H_{lp}(z)\right\} \tag{7.17}$$

$$h_d[n] = \mathscr{Z}^{-1}\left\{H_d(z)\right\} \tag{7.18}$$

Show that $h_d[n] = h_{lp}[n](-1)^n$.
Transformation:

$$H_d(z) = H_{lp}(z)\big|_{Z=-z}$$

3. Given the transfer function of a system

$$H(z) = \frac{-0.8 + z^{-1}}{1 - 0.8z^{-1}}$$

(a) Plot $|H(e^{j\omega})|$. What type of a filter is this?
(b) Find the I/O relation corresponding to this filter.
(c) Is this filter stable when the recursion you obtained in part (b) is implemented in a (i) causal manner (ii) anticausal manner?
(d) Let the input $x[n] = \left\{ \ldots, 1, \underbrace{1}_{n=0}, 1, 1, 2, 2, 2, 2, \ldots \right\}$. Find $y[n]$ using causal recursion (assume $y[-1] = 0$).
(e) Comment on the shape of output $y[n]$.

Chapter 8
Random Signals, Wiener Filtering and Speech Coding

8.1 Introduction

Up to now, we studied deterministic signals. In this chapter we introduce random signals. In many practical applications including speech, audio and image signals it is possible to model signals as random signals.

Let us assume that we have a sensor, e.g., a microphone, an ordinary camera or an infrared imaging system. We can model the observed sensor signal as a realization of a random process (r.p.).

A discrete-time random process is a sequence of random variables (r.v.). Let $\mathbf{x}[n]$ be a sequence of random variables. I will use bold face letters for r.v.'s to distinguish it from deterministic real signals. Each $\mathbf{x}[n]$ has a probability density function (pdf) $f_{x_n}(x)$. Furthermore, we have the joint pdfs:

$$
\begin{aligned}
&[f_{x_0x_1}(x_1,x_2),...,f_{x_{n_1}x_{n_2}}(x_1,x_2),\ldots]\\
&[f_{x_{n_1}x_{n_2}x_{n_3}}\ldots]\\
&[f_{x_{n_1}x_{n_2}x_{n_3}\ldots x_{n_L}}(x_1,x_2,\ldots,x_L),\ldots]
\end{aligned}
$$

In practical applications, it is not possible to know all the above joint pdf's. It may not even be possible to know $f_{x_n}(x)$. However, it may be possible to know some other statistical measures and parameters about the discrete-time random signal. For example, we can define a deterministic sequence based on the mean of $\mathbf{x}[n]$ as follows

$$
\mathrm{E}[\mathbf{x}_n] = \int_{-\infty}^{\infty} x f_{X_n}(x)dx = m_n, \quad n = 0, \pm 1, \ldots
$$

If $f_{X_i}(x) = f_{X_j}(x)$ for all $x[i]$ and $x[j]$, then $\mathrm{E}[\mathbf{x}_n] = m_x$ is constant.

We can observe deterministic signals $x[n]$ which are called realizations of the random process $\mathbf{x}[n]$. Infinitely many different realizations of a typical random process is possible but we may only observe a single relaziation of a random process in many practical applications.

It is possible to estimate the mean from a set of observations $x[n]$, $n = 0, 1, \dots, N-1$ as follows:

$$\hat{m}_X = \frac{1}{N} \sum_{n=0}^{N-1} x[n], \qquad f_{X_i}(x) = f_{X_j}(x) \text{ for all } x_i, x_j.$$

As N tends to infinity the estimate $\hat{m}_x = m_x$, if the r.p. is ergodic. In this course (book), we assume ergodicity.

For ergodic r.p.'s, it is possible to estimate the marginal pdf from the observations with the assumption that $f_{X_i} = f_{X_j}$ for all i, j. We first construct the histogram of the data:

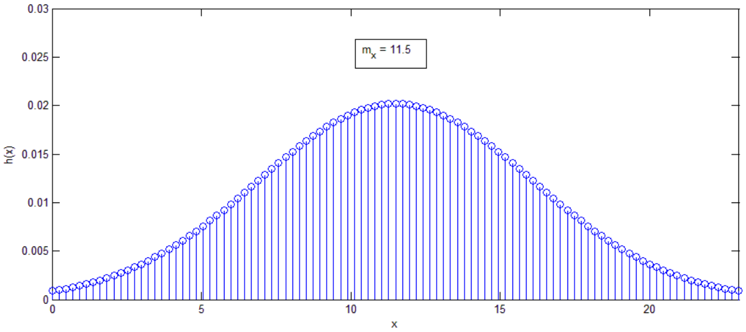


Fig. 8.1 Example data histogram plot

Let us assume that **x** can take discrete values $0, 1, \dots, x_0$. The O-th value of the histogram $h_x(0)$ represents the number of observations taking zero. Similarly, $h_x(1)$ represents the number of observations taking one, etc. We can estimate the pdf from the histogram of the data as follows:

$$\hat{f}_x(x) = \frac{h(x)}{\text{\# of observations}}$$

The variance of $\mathbf{x}[n]$ is defined as follows:

$$\sigma_{x_n}^2 = \int_{-\infty}^{\infty} (x_n - m_n)^2 f_{x_n}(x_n) dx_n\,, \quad n = 0, \pm 1, \pm 2, \dots$$

If $f_{x_i}(x) = f_{x_j}(x)$ for all $x[i]$ and $x[j]$, then

$$\sigma_x^2 = \int_{-\infty}^{\infty} (x - m_x)^2 f_x(x) dx$$

Estimate of the variance from the observed data $x[n],\ n = 0,1,\ldots,N-1$, is given by

$$\hat{\sigma}^2_{x_n} = \frac{1}{N}\sum_{n=0}^{N-1}(x[n]-\hat{m}_x)^2$$

with the assumption that the underlying pdf of $\mathbf{x}[n]$'s are the same for all n.

The auto-correlation sequence of the random process $\mathbf{x}[n]$ is defined as follows

$$r_X(n_1,n_2) = \mathrm{E}\left[X[n_1]X[n_2]\right], \quad n_1 = 0,\pm1,\pm2,\ldots \text{ and } n_2 = 0,\pm1,\pm2,\ldots$$

Similarly, the auto-covariance sequence is defined as follows:

$$\begin{aligned} c_X(n_1,n_2) &= \mathrm{E}\left[(X[n_1]-m_{n_1})(X[n_2]-m_{n_2})\right] \\ &= \mathrm{E}\left[X[n_1]X[n_2]-X[n_1]m_{n_2}-m_{n_1}X[n_2]+m_{n_1}m_{n_2}\right] \\ &= \mathrm{E}\left[X[n_1]X[n_2]\right]-m_{n_2}\underbrace{\mathrm{E}\left[X[n_1]\right]}_{m_{n_1}}-m_{n_1}\underbrace{\mathrm{E}\left[X[n_2]\right]}_{m_{n_2}}+m_{n_1}m_{n_2} \end{aligned}$$

Therefore,

$$c_X(n_1,n_2) = r_X(n_1,n_2) - m_{n_1}m_{n_2}$$

If $m_{n_1} = m_{n_2} = m_X$ or $f_{X_1} = f_{X_2}$, then $c_X(n_1,n_2) = r_X(n_1,n_2) - m_X^2$

Example: $c_x(0,0) = \sigma_X^2$

It is very difficult to deal with an arbitrary discrete-time random signal (= a random process) $\mathbf{x}[n]$ because we have to know all the marginal and joint pdf's. In practice, we have only a single realization $x[n]$ or only a part of the realization. Therefore, we have to have some other tools to deal with them. Luckily, some practical random processes have some structure that we can take advantage of. In this book, we assume that we only have wide-sense stationary ergodic random signals.

8.2 Stationary Random Processes

A random signal is called *Strict-Sense Stationary (SSS)* if all the underlying statistical properties are time-independent. This means that $x[n]$ and $x[n+n_0]$ have the same statistics for any integer n_0. Therefore, the $L-$th order density for any n_0 is as follows:

$$f_{X_{n_1},X_{n_2},\ldots,X_{n_L}}(x_1,x_2,\ldots,x_L) = f_{X_{n_1+n_0},X_{n_2+n_0},\ldots,X_{n_L+n_0}}(x_1,x_2,\ldots,x_L)$$

As a result, the first order pdf is independent of time:

(i) $f_{X_i}(x) = f_{X_j}(x) = f_X(x)$ for all i, j. Similarly,

(ii) $f_{X_1,X_2}(x_1,x_2) = f_{X_2,X_3}(x_1,x_2) = \cdots = f_{X_n,X_{n+1}}(x_1,x_2)$ for all $x[n]$, $x[n+1]$
$f_{X_1,X_{1+k}}(x_1,x_2) = f_{X_2,X_{2+k}}(x_1,x_2) = \cdots = f_{X_n,X_{n+k}}(x_1,x_2)$ for all $x[n]$ and k.

(iii) $f_{X_1,X_2,X_3}(x_1,x_2,x_3) = f_{X_2,X_3,X_4}(x_1,x_2,x_3) = \cdots = f_{X_n,X_{n+1},X_{n+2}}(x_1,x_2,x_3)$
for all $x[n]$, $x[n+1]$ $x[n+2]$

(iv) $f_{X_1,X_{1+k},X_{1+l}} = f_{X_m,X_{m+k},X_{m+l}}(x_1,x_2,x_3)$ for all $x[m]$, l, k

$\vdots$

The strict-sense stationarity is still too strict. We have to know the pdfs to characterize a SSS r.p. A random signal is called *Wide-Sense Stationary (WSS)* if

1. its mean is constant: $\mathrm{E}[x[n]] = m_X$ for all n, and
2. its auto-correlation depends only on $k = n_1 - n_2$:

$$\mathrm{E}[x[n_1]x[n_2]] = \mathrm{E}[x[n]x[n+k]] = r_X[k] \text{ for all } n, n_1, n_2 \text{ and } k \text{ where } k = |n_2 - n_1|$$

$$(2) \Longrightarrow \mathrm{E}[x[n]x[n+k]] = r_X[n,n+k] = r_X[1,1+k] = r_X[0,k] = \cdots = r_X[-k,0].$$

Therefore, the auto-correlation function is symmetric wrt $k = 0$:

$$(2) \Longrightarrow \mathrm{E}[x[n]x[n+k]] = r_X[k] = r_X[-k]$$

Similarly, the auto-covariance function is also symmetric wrt $k = 0$:

$$c_X[k] = c_X[-k]$$

where a new notation is adopted for the auto-correlation function as the difference between the indices matters.

Strict-Sense Stationarity implies WSS but a WSS r.p. need not be SSS.
SSS $\Rightarrow$ WSS but WSS $\nRightarrow$ SSS.

What is nice about WSS is that it is defined based on second order statistics (mean and autocorrelation) of a random process. It is possible to estimate the mean and autocorrelation from realizations of a random process and guess if a random process is WSS or not. Let us assume that we have the observations $x[n], n = 0, 1, \ldots, N-1$. The auto-correlation and auto-covariance sequences are estimated as follows:

$$\hat{r}_X[k] = \frac{1}{N}\sum_{n=0}^{N-1-k} x_n x_{n+k} \text{ and } \hat{c}_X[k] = \frac{1}{N}\sum_{n=0}^{N-1-k} (x_n - \hat{m}_X)(x_{n+k} - \hat{m}_X),$$

respectively.

$$\hat{c}_X[k] = \hat{r}_X[k] - \hat{m}_X^2, \text{ and}$$
$$\hat{c}_X[0] = \sigma_X^2$$

If $x[n]$ is a zero-mean $(\hat{m}_X = 0)$ WSS random process, then $\hat{c}_X[k] = \hat{r}_X[k]$.

In signal processing, we only have a single realization of a random process in most problems. So, we assume ergodicity to replace ensemble averages with time averages.

Example: In this example we observe a realization of a random process. In fact we assume that we have finitely many observations. We will estimate various statistical parameters from the observations.

Given the observations $x[n] = \{1,0,2,1,0,2,2,0,1,1,1,0,0,2,2\}$, estimate the average value (or mean).

$$\hat{m}_X = \frac{\sum_{n=0}^{N-1} x[n]}{N} = \frac{1+0+2+1+\cdots+2+2}{15} = 1$$

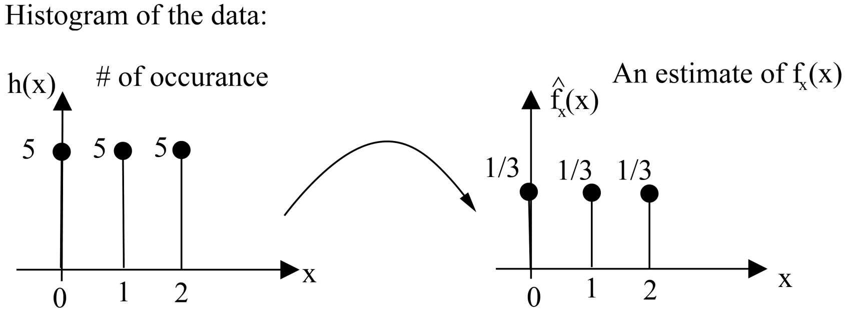


Fig. 8.2 Histogram of the observations and the estimated pdf.

Since **x** is a discrete r.v., its probability mass function (p.m.f.) can be estimated as follows:

$$p_x(x) = \frac{h(x)}{N} \quad \text{where } N = \#\text{ of observations}$$

$$\begin{aligned}\mathrm{E}[X] &= \int_{-\infty}^{\infty} x f_x(x) dx = \int_{-\infty}^{\infty} x \left(\frac{1}{3}\delta(x) + \frac{1}{3}\delta(x-1) + \frac{1}{3}\delta(x-2) \right) dx \\ &= \int_{-\infty}^{\infty} \frac{0}{3}\delta(x) dx + \int_{-\infty}^{\infty} \frac{1}{3}\delta(x-1) dx + \int_{-\infty}^{\infty} \frac{2}{3}\delta(x-2) dx = \frac{1}{3} + \frac{2}{3} = 1\end{aligned}$$

or

$$\mathrm{E}[X] = \sum_{x=0}^{2} x p_x(x) = 0 \cdot \frac{1}{3} + 1 \cdot \frac{1}{3} + 2 \cdot \frac{1}{3} = 1$$

Either use the *p.m.f.* or *p.d.f.* containing impulse functions. Results will be the same in both cases.

Variance:

$$\mathrm{E}[(X - \overbrace{\hat{m}_X}^{=1})^2] = \int_{-\infty}^{\infty} (x - \hat{m}_X)^2 f_X(x) dx$$

where the p.d.f. is

$$f_X(x) = \frac{1}{3}\delta(x) + \frac{1}{3}\delta(x-1) + \frac{1}{3}\delta(x-2)$$

and the equivalent p.m.f. is

$$p_X(x) = \left\{ \underbrace{\frac{1}{3}}_{x=0}, \frac{1}{3}, \frac{1}{3} \right\}$$

Let us calculate the variance of the r.v. X using the p.d.f. and p.m.f., respectively:

$$\sigma_X^2 = \mathrm{E}[(X-1)^2] = \int_{-\infty}^{\infty} (x-1)^2 \frac{\delta(x)}{3} dx + \int_{-\infty}^{\infty} (x-1)^2 \frac{\delta(x-1)}{3} dx + \int_{-\infty}^{\infty} (x-1)^2 \frac{\delta(x-2)}{3} dx$$

$$\mathrm{E}[(X-1)^2] = \frac{1}{3} + 0 + \int_{-\infty}^{\infty} 1^2 \frac{\delta(x-2)}{3} dx = \frac{2}{3} = \sigma_X^2$$

We get the same result using the p.m.f.:

$$\sigma_X^2 = \sum_{x=0}^{2} (x-1)^2 p_X(x) = (-1)^2 p_X(0) + 0^2 p_X(1) + 1^2 p_X(2) = \frac{1}{3} + 0 + \frac{1}{3}$$

we can estimate the variance from the observations

$$\hat{\sigma}_X^2 = \frac{\sum_{n=0}^{N-1} (x[n] - \hat{m}_X)}{N} = \frac{0^2 + (-1)^2 + 1^2 + \cdots + 1^2 + 1^2)}{15} = \frac{10}{15} = \frac{2}{3}$$

For a given r.v., the estimate $\hat{\sigma}_X^2$ may not be equal to the true variance σ_X^2 , in general. They will be close to each other but may not be equal.
Standard deviation σ_X is simply the square root of variance σ_X^2.
Example 2: Observations: $x[n] = \{0, 1, 1, 1.5, 2, 2, 1.5, 0, 1\}$, $N = 9$

To estimate the p.d.f. from the above data we need to normalize the histogram by $\sum_i x_i = 9$.

Estimate of the mean is :

$$\hat{m}_X = \frac{1}{N} \sum_i x_i = \frac{0+1+1+1.5+2+2+1.5+0+1}{9} = \frac{10}{9} \tag{8.1}$$

We can also use the estimated p.d.f. to calculate the mean.

$$\hat{m}_X = \int x f_X(x) dx = \int x \left(\frac{2}{9}\delta(x) + \frac{3}{9}\delta(x-1) + \frac{2}{9}\delta(x-1.5) + \frac{2}{9}\delta(x-2) \right) dx$$

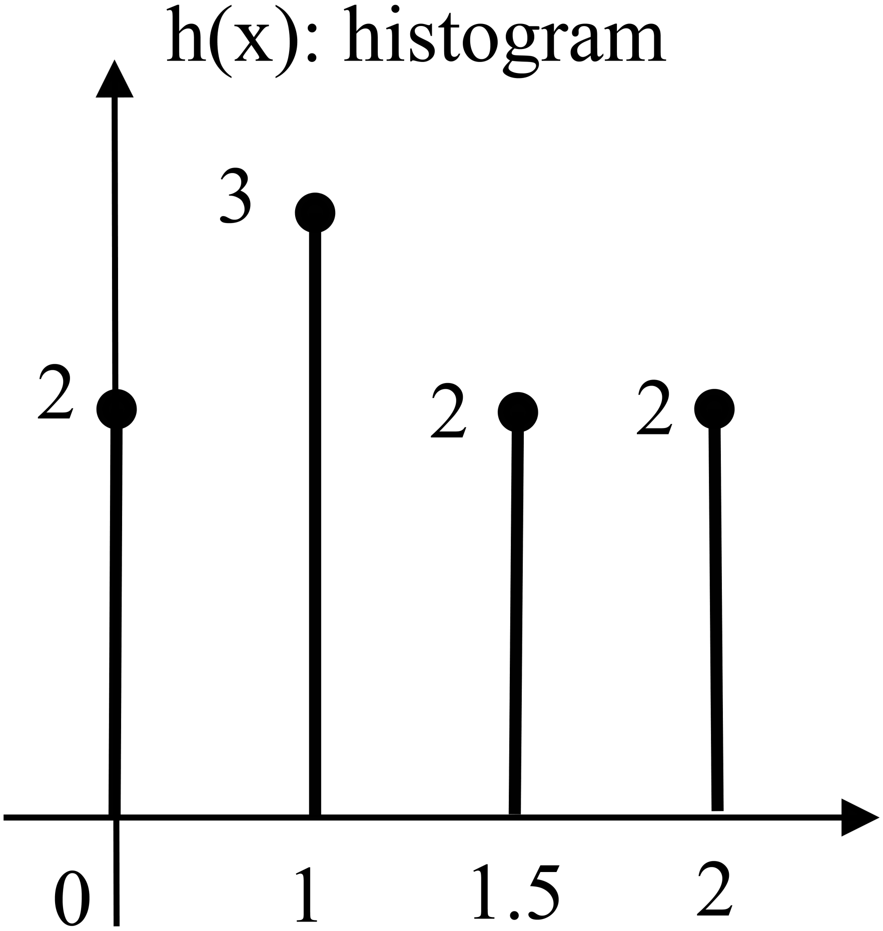


Fig. 8.3 Histogram of the data in Example 2.

$$\hat{m}_X = 0\frac{2}{9} + 1\frac{3}{9} + 1.5\frac{2}{9} + 2\frac{2}{9} = \frac{10}{9} \tag{8.2}$$

In this case, the mean estimate turns out to be the same as the one calculated from the estimated p.d.f. This is because the p.d.f. is directly formed from the observed data. In general, this may not be true for an arbitrary p.d.f.. They can be close to each other but may not be equal. As the number of observations increase, we expect the estimated mean to converge to the true mean.

We can also use the p.m.f to compute the mean.

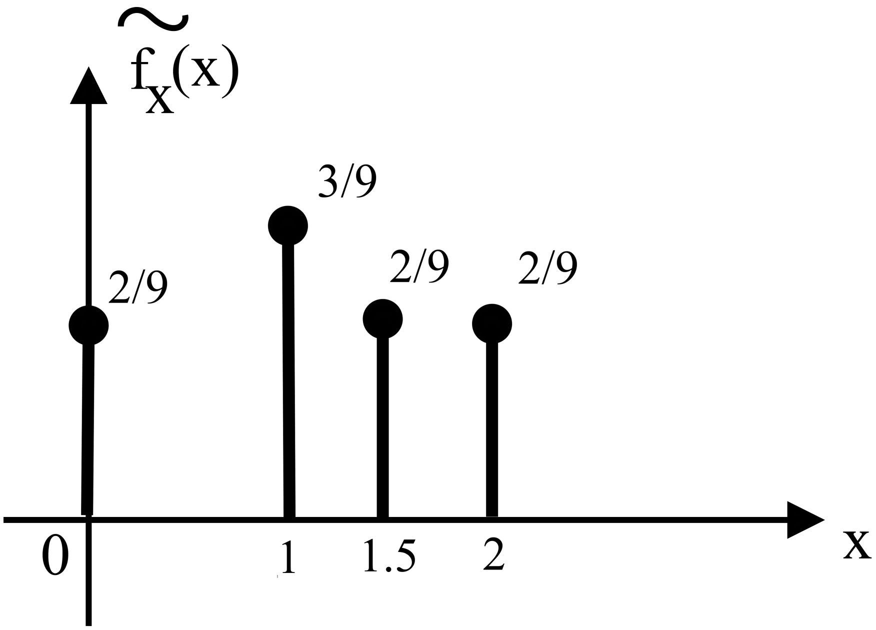


Fig. 8.4 Estimated p.d.f. in Example 2.

$$\hat{m}_X = \sum_i x_i p(X = x_i) = 0\frac{2}{9} + 1\frac{3}{9} + 1.5\frac{2}{9} + 2\frac{2}{9} = \frac{10}{9}$$

Variance Estimation:

$$\hat{\sigma}_X^2 = \frac{1}{N}\sum_i (x_i - \hat{m}_X)^2 = \frac{1}{9}\left(\left(0 - \frac{10}{9}\right)^2 + \left(1 - \frac{10}{9}\right)^2 + \cdots\right)$$
$$\hat{\sigma}_X^2 = \sum_i (x_i - \hat{m}_X)^2 p(X = x_i)$$

Example 3: (Given the Gaussian p.d.f.,)

$$fx(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu_0}{\sigma}\right)^2}$$

Let $\sigma^2 = 1, \mu_0 = 0$.

The following data is observed according to this p.d.f.:

$$x[n] = \{-0.4, -0.73, -0.87, -0.42, -0.94, 1.34, -0.99, 1.82, -0.37, -1.45, -0.62, 0.93, 1.06, 0.16, 0.29\}$$

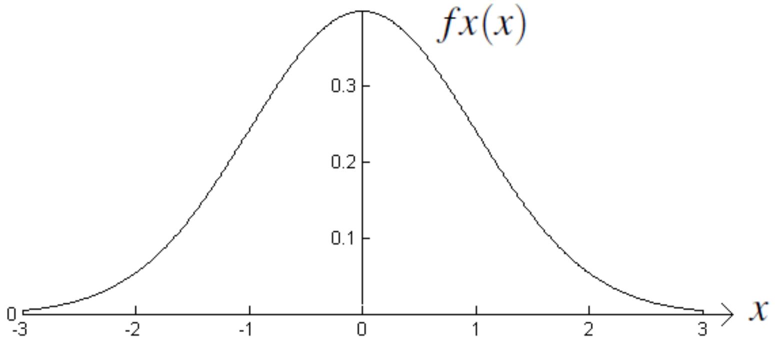


Fig. 8.5 Gaussian p.d.f.

The estimated mean is -0.0793 and the estimated variance is 0.9447. As you can see, the estimated mean and variance are not perfect.

8.2.1 Estimation of Autocorrelation from Data

Similar to mean and variance estimation, we can estimate autocorrelation values from the observed data [**?**]. The estimates can be obtained as follows based on the data given in Example 2:

$$\hat{r}_X[0] = \frac{1}{N}\sum_{i}^{N-1} x_i^2 = \frac{1}{9}\left(0^2+1^2+1^2+1.5^2+\cdots\right)$$

$$\hat{r}_X[1] = \frac{1}{N}\sum_{i}^{N-1-1} x_i x_{i+1} = \frac{1}{9}\left(0\times 1+1\times 1+1\times 1.5+\cdots\right)$$

$$\hat{r}_X[2] = \frac{1}{N}\sum_{i}^{N-1-2} x_i x_{i+2}$$

There are two methods to estimate the auto-correlation sequence:

$$\hat{r}_x[k] = \frac{1}{N}\sum_{n=0}^{N-|k|-1} x[n]x[n+k], \tag{8.3}$$

or

$$\hat{r'}_x[k] = \frac{1}{N-|k|}\sum_{n=0}^{N-|k|-1} x[n]x[n+k] \tag{8.4}$$

In both Eq. (**??**) and (**??**), there are $N-|k|-1$ terms inside the summation. In (**??**) the summation is normalized by N because, $\hat{r}_x(0)$ is more reliable than $\hat{r}_x(1)$ which is in turn more reliable than $\hat{r}_x(2)$ etc. By dividing the sum by N we emphasize the fact that we use more samples to estimate $\hat{r}_x(0)$ compared to $\hat{r}_x(1)$ etc.

The estimate $\hat{r}_x[k]$ is biased but it is preferred when N is much larger than k. This estimate corresponds to the triangular windowed version of $\hat{r'}_x[k]$.

Estimate $\hat{c}_x[k]$ of the auto-covariance $c_x[k] = \mathrm{E}\left[(x[n]-\hat{m}_X)(x[n+k]-\hat{m}_X)\right]$ for a WSS random process is given by

$$\hat{c}_x[k] = \frac{1}{N}\sum_{n=0}^{N-|k|-1}(x[n]-\hat{m}_X)(x[n+k]-\hat{m}_X),$$

or

$$\hat{c'}_x[k] = \frac{1}{N-|k|}\sum_{n=0}^{N-|k|-1}(x[n]-\hat{m}_X)(x[n+k]-\hat{m}_X)$$

and $c_x[0] = \sigma_x^2$ is the variance (or the power) of the WSS random process.

Auto-covariance: $c_x[m] = r_x[m] - m_x^2$ with the assumption that we have a WSS r.p.

8.3 Linear Minimum Mean-Squared-Error (LMMSE) Predictor

In this section we introduce a new filter design method. We do not have any frequency domain specifications as in Chapter 6. We want to predict $x[n]$ using L past observations,$x[n-i],\ i = 1,2,...,L:$

$$\hat{x}[n] = a_1x[n-1] + a_2x[n-2] + \cdots + a_Lx[n-L] \tag{8.5}$$

where $\hat{x}[n]$ is the estimate of $x[n]$ and a_i's are the weights that we should determine to obtain a reasonable estimate.

The filter design problem is to determine $a_1, a_2, \ldots, a_L$ similar to the FIR filter design problem that we studied in Chapter 6, however there are no frequency domain specifications. The goal is to estimate the next $x[n]$ value given in the past observations. We define the error $e(n) = x[n] - \hat{x}[n]$ and design the filter by minimizing the mean-squared-error (MSE)

$$\mathrm{E}\left[e^2[n]\right] = \mathrm{E}\left[(x[n]-\hat{x}[n])^2\right] \tag{8.6}$$

Although $e(n)$ is available when $x[n]$ is available and to determine the MSE we need to know the joint pdf of $x[n-i],\ i = 0,1,2,...,L$ it is possible to find a practical solution to the problem, if we assume that the r.p. $\mathbf{x}[n]$ is wide-sense stationary.

To solve the filter design problem, we take the partial derivative of the MSE defined in Eq. (8.1) with respect to the unknowns $a_1, a_2, \ldots, a_L$ and set the derivatives to zero:

$$\frac{\partial}{\partial a_i}\mathrm{E}\left[(x[n]-\hat{x}[n])^2\right]=0\,,\quad i=1,2,\ldots,L$$
$$\mathrm{E}\left[\frac{\partial}{\partial a_i}(x[n]-\hat{x}[n])^2\right]=0$$
$$\mathrm{E}\left[2\,(x[n]-\hat{x}[n])\,\frac{\partial}{\partial a_i}\left(x[n]\underbrace{-a_1x[n-1]-a_2x[n-2]-\cdots-a_Lx[n-L]}_{-\hat{x}[n]}\right)\right]=0$$
$$\mathrm{E}\left[2\,(x[n]-\hat{x}[n])\,(-1)x[n-i]\right]=0\,,\quad i=1,2,\ldots,L$$

Therefore we obtain the so-called “orthogonality condition“ for the optimal filter design:

$$\mathrm{E}\left[(x[n]-\hat{x}[n])\,x[n-i]\right]=0\,,\quad i=1,2,\ldots,L$$
$$\mathrm{E}\left[x[n]x[n-i]\right]=\mathrm{E}\left[\underbrace{(a_1x[n-1]+a_2x[n-2]+\cdots+a_Lx[n-L])}_{=\hat{x}[n]}x[n-i]\right]$$

In other words, the error must be orthogonal to the past observations. This leads to the following equations

$$r_x[i]=a_1\mathrm{E}\left[x[n-1]x[n-i]\right]+a_2\mathrm{E}\left[x[n-2]x[n-i]\right]+\cdots+a_L\mathrm{E}\left[x[n-L]x[n-i]\right]$$
$$r_x[i]=a_1r_x[i-1]+a_2r_x[i-2]+\cdots+a_Lr_x[i-L]\,,\quad i=1,2,\ldots,L$$

where r_x is the autocorrelation sequence of the WSS random process $\mathbf{x}[n]$. The optimal predictor coefficients satisfy the following set of equations

$$\begin{aligned}
r_x[1]&=a_1r_x[0]+a_2r_x[1]+\cdots+a_Lr_x[L-1]\\
r_x[2]&=a_1r_x[1]+a_2r_x[2]+\cdots+a_Lr_x[L-2]\\
&\vdots\\
r_x[L]&=a_1r_x[L-1]+a_2r_x[L-2]+\cdots+a_Lr_x[0]
\end{aligned}$$

This set of linear equations are called the Auto-correlation Normal Equations (ACNE).

$$\underbrace{\begin{bmatrix} r_X[1]\\ r_X[2]\\ \vdots\\ r_X[L]\end{bmatrix}}_{\mathbf{r}_X}=\underbrace{\begin{bmatrix} r_X[0] & r_X[1] & \cdots & r_X[L-1]\\ r_X[1] & r_X[0] & & \\ \vdots & & \ddots & \vdots\\ r_X[L-1] & & \cdots & r_X[0]\end{bmatrix}}_{\mathbf{R}_X}\underbrace{\begin{bmatrix} a_1\\ a_2\\ \vdots\\ a_L\end{bmatrix}}_{\mathbf{a}}$$

where **a** represent a vector containing the filter coefficients. Solution of the linear predictor design problem:

$$\mathbf{a} = \mathbf{R}_X^{-1}\mathbf{r}_X$$

It is always possible to find the inverse of the autocorrelation matrix for WSS random processes, e.g., see the textbook by Papoulis [**?**].

In practice, we may not know the p.d.f.s of the random process. As a result we cannot compute the autocorrelation values for a given random process X. In such cases we have to estimate the autocorrelation sequence $r_x[k]$ from past observations (past data) using the formula:

$$\hat{r}_x[k] = \frac{1}{N}\sum_{i=0}^{N-1-k} x[i]x[i+k]$$

for k=0,1,2,...,L. After we estimate the autocorrelation sequence from past observations we plug them into the ACNE to estimate the filter coefficients.

8.4 White Noise and MA and AR Random Processes

White noise is a wide-sense stationary random process. It is widely used in electronics systems to model noise.

Zero-mean white noise has the following autocorrelation sequence:

$$\begin{aligned} r_u[0] &= \sigma_u^2 \neq 0 \\ r_u[k] &= 0 \text{ for } k = \pm 1, \pm 2, \dots \end{aligned}$$

In other words,

$$r_u[k] = \sigma_u^2[k]\delta[k]$$

It means that there is no correlation between the samples of $u[n]$.

When each sample of a random process has the same p.d.f. but they are all independent of each other we use the term the independent identically distributed (i.i.d.).

For i.i.d. random process,

$$f_{u_j,u_k}(t_1,t_2) = f_{u_j}(t_1)f_{u_k}(t_2), j \neq k$$

Then,

$$\mathrm{E}\left[U[j]U[k]\right] = \mathrm{E}\left[U[j]\right]\mathrm{E}\left[U[k]\right]$$

Independence implies uncorrelatedness but the converse is not true in general.

Theorem: Stable LTI systems preserve wide-sense stationarity.

For any WSS input $u[n]$, the output $y[n]$ is also WSS in stable LTI systems. When a realization of the random process $u[n]$ is the input to an LTI system, the output becomes a realization of the random process $y[n]$.

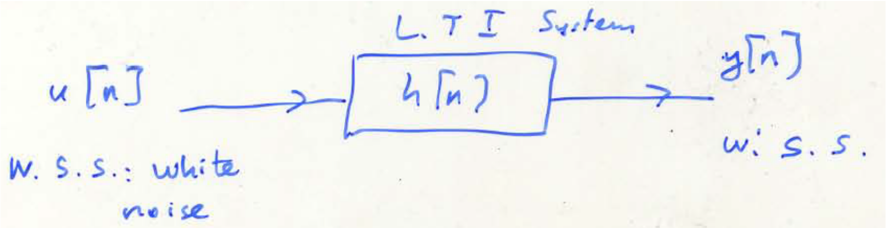


Fig. 8.6 An LTI system driven by white noise.

Definition: Given a zero-mean WSS random process $\mathbf{x}[n]$ with auto-correlation $r_x[k]$. The spectrum of $\mathbf{x}[n]$ is defined as the DTFT of $r_x[k]$:

$$S_x(e^{j\omega}) \triangleq \sum_{n=-\infty}^{\infty} r_x[k]e^{-j\omega k}$$

Since Since the autocorrelation sequence is symmetric w.r.t. $k=0$: $r_x[k] = rxX[-k]$, the spectrum $S_X(e^{j\omega})$ is real for the real w.s.s. random process, i.e., it does not have any phase! When the random process is not zero mean the spectrum is defined as the DTFT of the autocovariance sequence.
Example: Spectrum of white noise:

Given the autocorrelation sequence

$$r_U[k] = \begin{cases} \sigma_U^2 & \text{if } k=0 \\ 0 & \text{if } k \neq 0 \end{cases}$$

$$S_U(e^{j\omega}) = r_U[0]e^{j\omega 0} + 0e^{-j\omega} + 0e^{j\omega} + \dots$$
$$= \sigma_U^2 \text{ for all } \omega$$

The spectrum of white noise is flat. That is why we call it white noise because it contains all the spectral components as “white light“.

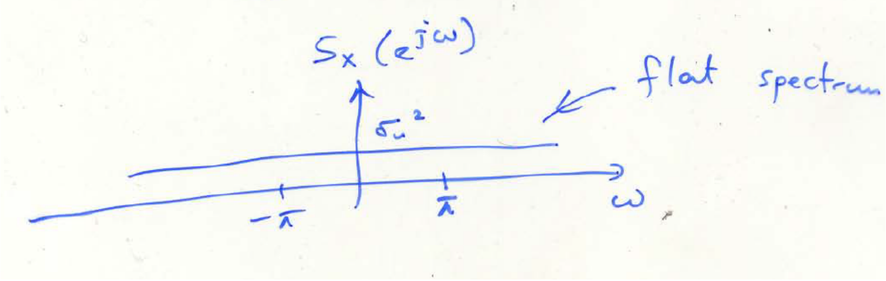
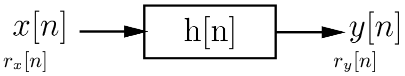


Fig. 8.7 Spectrum of white noise.

Theorem: Let Y represent the output of the stable LTI system $h[n]$. The spectrum $S_Y(e^{jw})$ is given by:,

$$S_Y(e^{j\omega}) = \left|H(e^{j\omega})\right|^2 S_X(e^{j\omega})$$

where X is WSS input to the stable LTI system whose frequency response is $H(e^{j\omega})$.



When the input is white noise, the spectrum of the output is given by

$$S_Y(e^{j\omega}) = \left|H(e^{j\omega})\right|^2 \sigma_U^2$$

Example: (FIR or Moving Average (MA) system)

Let $y[n] = \frac{1}{2}x[n] + \frac{1}{2}x[n-1]$. This is a simple FIR filter. It is also called a moving average system of order 2 because the filter has only two nonzero coefficients. Calculate $r_y[k]$ and $S_y(e^{j\omega})$ given that the input x is white, zero-mean random process with variance σ^2.

The autocorrelation sequence X has to be calculated one by one. We first start with $r_x[0]$:

$$\begin{aligned}
r_Y[0] &= \mathrm{E}\left[\left(\frac{1}{2}x[n] + \frac{1}{2}x[n-1]\right)\left(\frac{1}{2}x[n] + \frac{1}{2}x[n-1]\right)\right] \\
&= \mathrm{E}\left[\frac{1}{4}x^2[n] + 2\frac{1}{4}x[n]x[n-1] + \frac{1}{4}x^2[n-1]\right] \\
&= \frac{1}{4}\mathrm{E}\left[x^2[n]\right] + \frac{1}{2}\underbrace{\mathrm{E}\left[x[n]x[n-1]\right]}_{=0} + \frac{1}{4}\mathrm{E}\left[x^2[n-1]\right] \\
r_Y[0] &= \frac{1}{4}\sigma^2 + \frac{1}{4}\sigma^2 = \frac{1}{2}\sigma^2
\end{aligned}$$

Next let us determine $r_Y[1]$:

$$
\begin{aligned}
r_Y[1] &= \mathrm{E}\left[y[n]y[n-1]\right] \\
&= \mathrm{E}\left[\left(\frac{1}{2}x[n]+\frac{1}{2}x[n-1]\right)\left(\frac{1}{2}x[n-1]+\frac{1}{2}x[n-2]\right)\right] \\
&= \mathrm{E}\left[\frac{1}{4}x[n]x[n-1]+\frac{1}{4}x[n]x[n-2]+\frac{1}{4}x^2[n-1]+\frac{1}{4}x[n-1]x[n-2]\right] \\
&= \frac{1}{4}\underbrace{\mathrm{E}\left[x[n]x[n-1]\right]}_{=0}+\frac{1}{4}\underbrace{\mathrm{E}\left[x[n]x[n-2]\right]}_{=0}+\frac{1}{4}\mathrm{E}\left[x^2[n-1]\right]+\frac{1}{4}\underbrace{\mathrm{E}\left[x[n-1]x[n-2]\right]}_{=0}
\end{aligned}
$$

We do not need to compute $r_Y[1]$ because $r_Y[1] = \frac{1}{4}\sigma^2 = r_Y[-1]$ from the symmetry property of the auto-correlation sequence. Next we compute $r_Y[2]$:

$$
\begin{aligned}
r_Y[2] &= \mathrm{E}\left[y[n]y[n-2]\right] \\
&= \mathrm{E}\left[\left(\frac{1}{2}x[n]+\frac{1}{2}x[n-1]\right)\left(\frac{1}{2}x[n-2]+\frac{1}{2}x[n-3]\right)\right] \\
r_Y[2] &= 0 = r_Y[-2] \\
r_Y[3] &= r_Y[-3] = 0 \\
&\ldots
\end{aligned}
$$

In a MA(p) system $r_Y[p] = r_Y[p+1] = 0 = r_Y[p+2] = \cdots$
Example: (IIR, recursive, all-pole or Auto-Regressive (AR) random process)

Let $y[n] = \alpha y[n-1] + u[n]$ where $u[n]$ is a zero-mean, white and WSS random process with variance σ_u^2. This filter is an all-pole recursive system. It is also called an AR(1) random process because it has a single pole. In this case the pole is α because the transfer function of the system is:

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

Determine the first order predictor $\hat{y}[n] = a_1 y[n-1]$.

First calculate the autocorrelation sequence of $y[n]$

$$
\begin{aligned}
r_Y[0] &= \mathrm{E}\left[y[n]y[n]\right] \\
&= \mathrm{E}\left[(\alpha y[n-1]+u[n])\,(\alpha y[n-1]+u[n])\right] \\
&= \alpha^2 \mathrm{E}\left[y[n-1]y[n-1]\right] + 2\alpha \underbrace{\mathrm{E}\left[y[n-1]u[n]\right]}_{=0} + \mathrm{E}\left[u^2[n]\right] \\
&= \alpha^2 r_Y[0] + 0 + \sigma_U^2 \\
r_Y[0] &= \frac{\sigma_U^2}{1-\alpha^2} \text{ where } \sigma_U^2 \text{ is the variance of } u[n]. \\
r_Y[1] &= \mathrm{E}\left[y[n]y[n-1]\right] \\
&= \mathrm{E}\left[(\alpha y[n-1]+u[n])\,y[n-1]\right] \\
&= \alpha \mathrm{E}\left[y^2[n-1]\right] + \underbrace{\mathrm{E}\left[u[n]y[n-1]\right]}_{=0} \\
r_Y[1] &= \alpha r_Y[0] = \alpha \frac{\sigma_U^2}{1-\alpha^2} \\
r_Y[2] &= \mathrm{E}\left[y[n]y[n-2]\right] \\
&= \mathrm{E}\left[(\alpha y[n-1]+u[n])\,y[n-2]\right] \\
&= \alpha \mathrm{E}\left[y[n-1]y[n-2]\right] + \underbrace{\mathrm{E}\left[u[n]y[n-2]\right]}_{=0} \\
r_Y[2] &= \alpha r_Y[1] = \alpha^2 \frac{\sigma_U^2}{1-\alpha^2}
\end{aligned}
$$

where $E[y[n-1]u[n]] = 0$ because $y[n-1]$ does not contain $u[n]$. It is formed as a linear combination of $u[n-1]$, $u[n-2], \ldots$ which are all uncorrelated with $u[n]$. In general, $r_y[k] = \alpha r_y[k-1]$ for this WSS random process.

A.C.N.E. becomes (the autocorrelation matrix turns out to be a 1×1 matrix):

$$
\begin{aligned}
r_y[1] &= r_y[0]a_1 \\
a_1 &= \frac{r_Y[1]}{r_Y[0]} = \alpha
\end{aligned}
$$

Hence, the first-order predictor is $\hat{y}[n] = \alpha y[n-1]$. This predictor makes sense. If we do not know the value of $u[n]$ we can put the mean value of the process, that is zero into the predictor.

8.5 Quantization and A to D Conversion

Sampling an analog signal is not enough to process the signal using a computer because samples of an analog signal may take real values. We have to quantize them to process the data using digital signal processors and computers. The quantization process introduces noise.

Practical A to D converters provide 8 bit per sample, 16 bit per sample or 32 bit sample data.

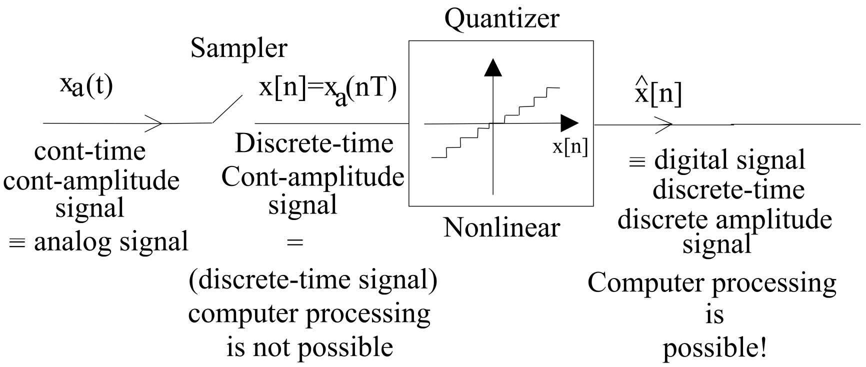


Fig. 8.8 Quantization is part of the standard A to D converter systems.

Example: 8-level quantizer:

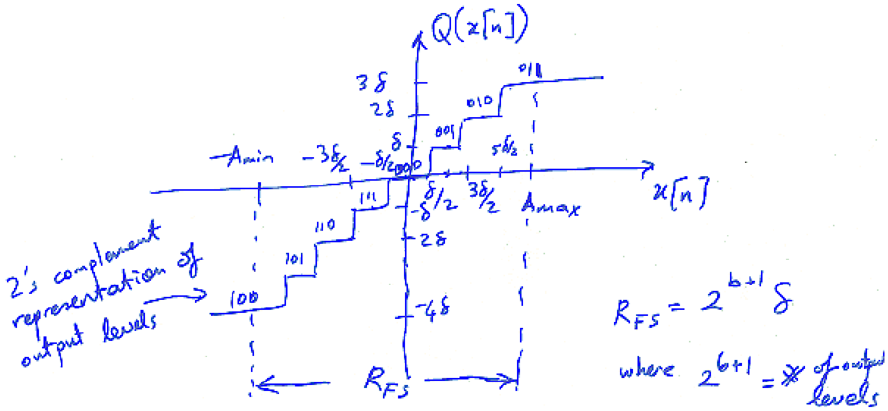


Fig. 8.9 A uniform 8-level quantizer.

$b+1$: output word length including the sign bit ($b+1=3$ in this example).

Error: $e[n] = Q(x[n]) - x[n] = \hat{x}[n] - x[n]$ and $-\delta/2 \le e[n] < \delta/2$ if the input is bounded by A_{min} and A_{max}.

Let $x[n]$ be a random process with $x[n] \sim \mathscr{U}\,[-A_{min}, A_{max}]$ (uniform pdf).

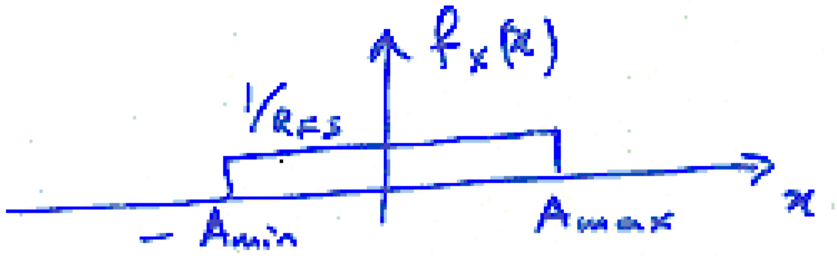


Fig. 8.10 PDF of the input x.

$R_{FS} = A_{min} + A_{max}$
Signal power: $A_{min} \cong A_{max} = A$ (number of quantization levels are high).
Power of the input $x[n]$:

$$\sigma_X^2 = \int_{-\infty}^{\infty} x^2 f_X(x) dx = \int_{-A}^{A} x^2 \frac{1}{R_{FS}} dx = \left.\frac{x^3}{3R_{FS}}\right|_{-A}^{A} = \frac{A^3}{3R_{FS}} - \frac{-A^3}{3R_{FS}}$$

$$\sigma_X^2 = \frac{2A^3}{6A} = \frac{A^2}{3} = \frac{R_{FS}^2}{12} = \frac{\left(2^{b+1}\delta\right)^2}{12}$$

Noise power: $e[n] = Q(x[n]) - x[n] = \hat{x}[n] - x[n]$

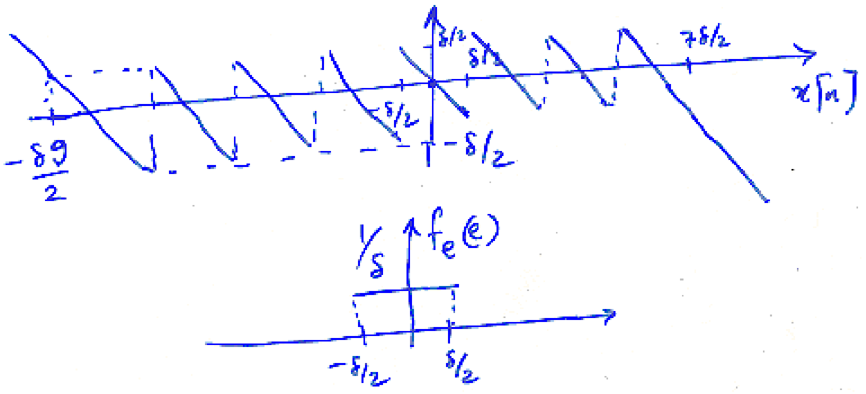
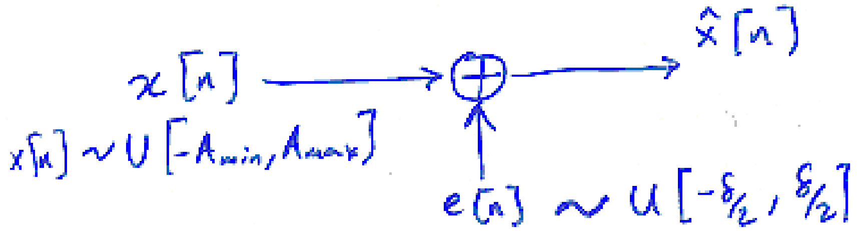


Fig. 8.11 PDF of the quantizer error.

$$\sigma_e^2 = \int_{-\delta/2}^{\delta/2} e^2 \frac{1}{\delta} de = \frac{\delta^2}{12}$$

Statistical model of the quantizer:

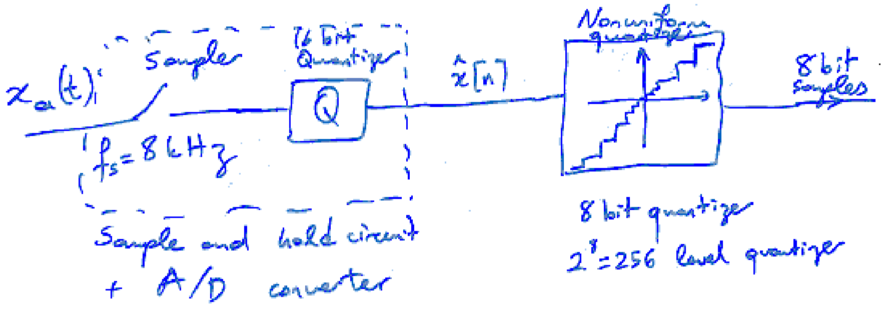


Signal to Noise Ratio (SNR):

$$\mathrm{SNR} = 10\log_{10}\left(\frac{\sigma_X^2}{\sigma_e^2}\right)\ \mathrm{dB} = 20\log_{10}\left(\frac{2^{b+1}\delta}{\delta}\right) = 20\log_{10}(\#\text{ of levels})$$

If the number of levels is increased, SNR increases and δ decreases resulting in smaller error.

In speech processing:



PCM (Pulse Code Modulation) quantizer: 8000 samples/sec × 8 bits/sample = 64 kbits/sec (Transmission rate).

8.6 LPC-10 Algorithm (Earliest Voice Coder / Vocoder)

Linear Predictive Coding (LPC) is the earliest speech vocoder algorithm and LPC-10 is a NATO speech coding standard for military applications. It is the mother (or father) of the GSM speech coding algorithm.

Input comes from lungs and vocal cords.

System consists of nasal cavity and vocal tract.

In voiced segments we assume that the input is periodic and the speech is also almost periodic.

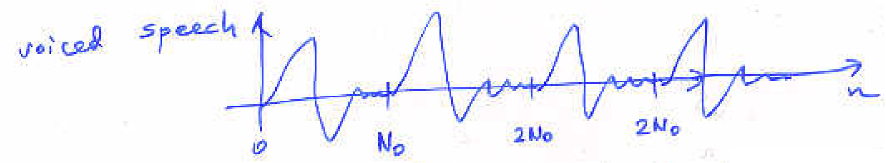


Fig. 8.12 A typical voiced speech segment.

N_0 : pitch period, pitch $= \frac{1}{N_0}$ is the frequency of oscillation of vocal cords.
In unvoiced segments we assume that the input is white noise.

$$\mathrm{E}\left[w(i)w(i+m)\right] = \sigma_w^2\delta(m) \text{ and } r_w[m] = \begin{cases} \sigma_w^2 & \text{if } m = 0 \\ 0 & \text{if } m \neq 0 \end{cases}$$

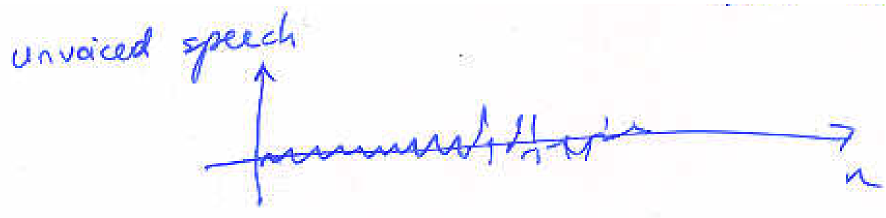


Fig. 8.13 Unvoiced speech segment

Speech is not a WSS r.p. but each phoneme can be assumed to be a WSS rp. We assume that the all-pole system is excided by white noise in unvoiced phoneme segments and it is excited by a periodic pulse-train in voiced segments as shown in Figure **??**.

In LPC-10 standard speech is divided into segments (frames) containing 180 samples in 8kHz sampled speech. In each segment, the speech is assumed to be WSS and we can design LMMSE predictors.

In each frame, we have to estimate the auto-correlation values $\hat{r}_{frame\ 1}[m] \neq \hat{r}_{frame\ 2}[m]$ $m = 0, 1, 2, \ldots, 10$, because each frame can have different statistical properties.

Solve ACNE in each frame $\mathbf{a} = \mathbf{R}_X^{-1}\mathbf{r}_X$ to get the predictor coefficients of each frame.

Determine if a frame is voiced or unvoiced. If voiced, send the pitch period N. If unvoiced, do nothing.

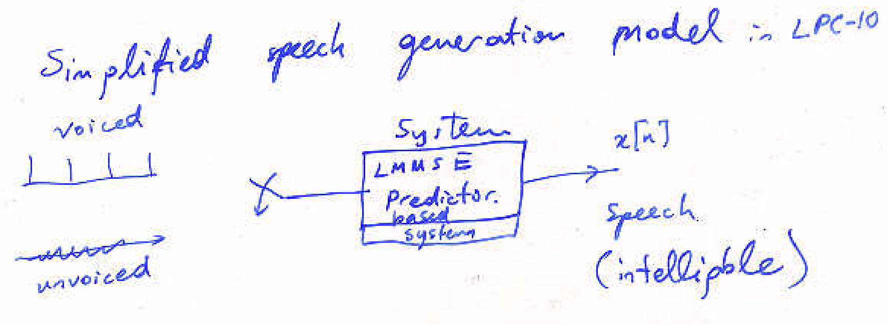
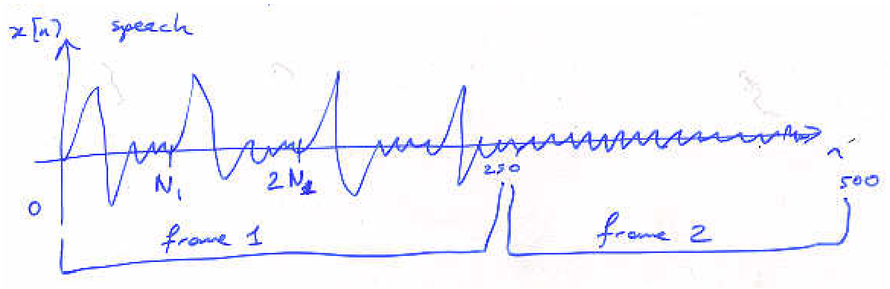


Fig. 8.14 Simplified speech model used in the LPC-10 algorithm



Estimate $\mathrm{E}\left[e^2[n]\right] = \mathrm{E}\left[(x[n]-\hat{x}[n])^2\right] = \sigma_e^2\ \sigma_e^2 = \sum_{i=1}^{M} a_i r_X[i]$

Model parameters are transmitted. Transmit $\left[\{a_i\}_{i=1}^{M=10}, \sigma_e^2, V/UV, N_1\right]$ to the receiver for each frame.

At the receiver

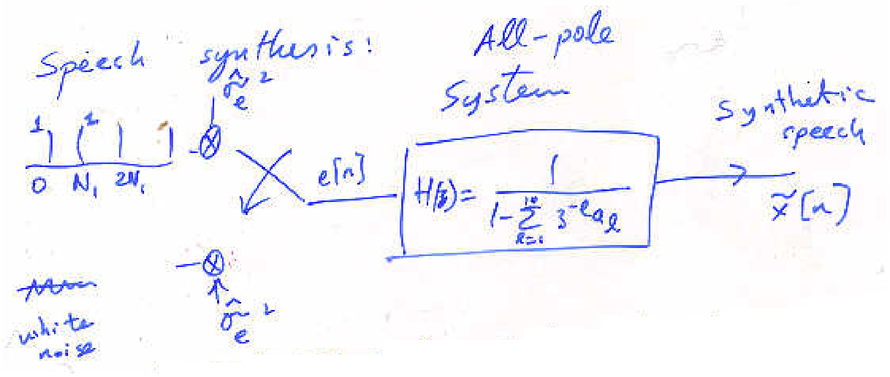
$$\hat{x}[n] = a_1x[n-1] + a_2x[n-2] + \ldots + a_Mx[n-M]$$
$$x[n]-\hat{x}[n] = e[n] = -\left(a_1x[n-1] + a_2x[n-2] + \ldots + a_Mx[n-M]\right) + x[n]$$

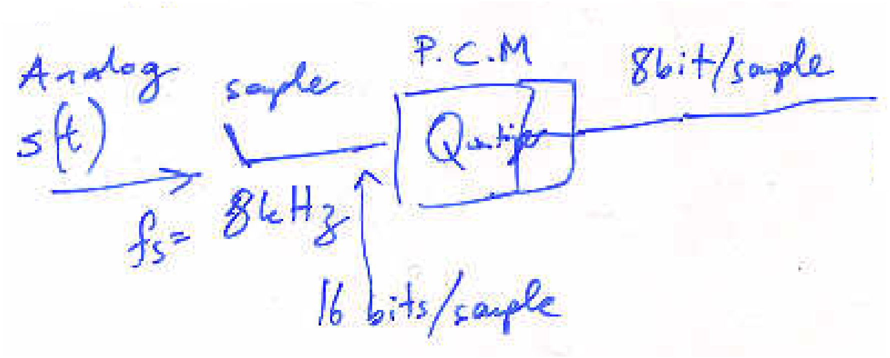
IIR recursive system

$$x[n] = a_1x[n-1] + a_2x[n-2] + \ldots + a_Mx[n-M] + e[n]$$
$$X(z) = \frac{1}{1-\sum_{l=1}^{10} z^{-l}a_l}E(z)$$

For a frame of 250 samples, we transmit 13 parameters. Data transmission rate: 2.4 kbit/sec. PCM rate is 64 kbit/sec.

8 kHz × 8 bit/sample = 64 kbit/sec in ordinary telephone system.





8.7 Exercises

1. Find a formula for $\mathrm{E}\left[e^2[n]\right]$ in LMMSE analysis.

2. Develop a fourth order predictor for the value of the U.S. dollar against TL. Use at least 30 data samples to estimate the autocorrelation and predict the value of U.S. dollar on 29^{th} of December.

3. Let $y[n] = 0.9y[n-1] + u[n]$ where $u[n] = \left\{ \underbrace{1}_{n=0}, -1, 1, 1, -1, -1 \right\}$ and $y[-1] = 0$.

(a) Calculate $y[0], y[1], y[2], y[3], y[4]$ and $y[5]$.

(b) Calculate the mean, variance and the first two autocorrelation values for the data that you obtained in part (a).

(c) Estimate $y[6]$ using the A.C.N.E.

(d) Determine the spectrum of $y[n]$ if $u[n]$ is white, zero-mean with variance $\sigma_u^2 = 1$.

4. (a) How do we model the speech signal in LPC-10 vocoder?

(b) What does the 'pitch' period refer to in speech processing?

(c) Let $y[n] = 0.9y[n-1] + u[n]$ where $u[n]$ is white, zero-mean with variance $\sigma_u^2 =$

0.1. Can $y[n]$ be a voiced speech signal? Explain your answer.
(d) In vocoders (including the GSM vocoder) do we transmit actual speech samples?
(e) Do we use uniform or non-uniform quantizers in PCM speech coding standard? Explain your answer.
5. Consider the I/O relation:

$$y[n] = 0.8y[n-1] + x[n-1] - 0.8x[n]$$

where $x[n]$ is the input and $y[n]$ is the output. Recursion is implemented in a causal manner.
(a) Find the frequency response and plot $|H(e^{j\omega})|$.
(b) Is this system stable? Prove your answer.
(c) Will this system be stable if the recursion is implemented in an anti-causal manner?
(d) Let the input $x[n]$ be a white, zero-mean random process with variance 1. Will $y[n]$ be a wide sense stationary random process? Determine the spectrum of $y[n]$, i.e., $S_Y(e^{j\omega})$.
6. Given the following data:

$$\left\{ \underbrace{1}_{n=0}, 0.5, 1.25, 0.8, 0.4, -0.3, -0.9, -1, -1.5, -0.9, 0.65 \right\}$$

(a) Estimate the mean, variance and the first two autocorrelation values for this data.
(b) Determine the first and second order LMMSE predictors for this data. (First order: $\hat{x}[n] = a_1 x[n-1]$ and second order predictor: $\hat{x}[n] = b_1 x[n-1] + b_2 x[n-2]$)
(c) Determine the minimum mean square error for these predictors.
7. Consider an AR(1) random process, $x[n]$ generated using a recursive manner as follows:

$$x[n] = \alpha x[n-1] + u[n]$$

where $u[n]$ is a white random process with zero-mean with variance $\sigma_u^2 = 1$.
(a) What is $r_U[n]$? What is $r_X[0], r_X[1]$ and $r_X[2]$.
(b) LMMSE optimal predictor for this random process is obtained using the 'orthogonality principle'. What is the orthogonality principle?
(c) The LMMSE predictor is $\hat{x}[n] = 0.8x[n-1]$. What should be the value of the parameter α? Use the orthogonality principle to determine α?
(d) Let $\tilde{x}[n] = a_1 x[n-1] + a_2 x[n-2]$. Determine a_1 and a_2 using the orthogonality principle.
(e) What are the impulse responses of the predictors in parts (c) and (d)?
8. Let $x[n] = 0.9x[n-1] + u[n]$, where $u[n]$ is white, zero-mean r.p. with variance 1.
(a) Is $x[n]$ a wide-sense stationary r.p.? Explain.
(b) Determine the auto-correlation function of $x[n]$.
(c) Determine the first order LMMSE predictor for $x[n]$ ($\hat{x}[n] = ax[n-1]$).
(d) Given a realization of $u[n] = \left\{ \underbrace{1}_{n=0}, 0, -1, 2, 0 \right\}$. Obtain the corresponding re-

alization of $x[n]$. Assume $x[-1] = 0$.
(e) Estimate $x[5]$ using the predictor obtained in part (c) and from the predictor obtained from the data in (d). Compare the results of the two predictors.
9. Let $x[n] = \{-1, 2, 4, 0, 2, -1, 4, 4\}$.
(a) Plot the histogram of $x[n]$.
(b) Determine the PDF and PMF of $x[n]$.
(c) Estimate the mean m_X and variance of σ_X^2 of $x[n]$.
(d) Calculate $r_X[k]$ for $k = 0, 1, 2$.
(e) Repeat parts (a)-(d) for $x[n] = \{1, 0, -1, 2, 2, 0, 2, 1\}$.
10. Given $x[n] = \{1, 4, 1, 4, 1, 1\}$ starting at index zero.
(a) Calculate the estimated mean $\hat{m}_x$ and estimated variance $\hat{\sigma}_x^2$ of $x[n]$.
(b) Propose a third order anticausal FIR filter $h[n]$ with $h[0] = 1$ so that the output signal $y[n], n = 1, 2, 3, 4$ has smaller variance than the input signal $x[n]$. Calculate (approximately) the variance $\hat{\sigma}_y^2$ of $y[n], n = 1, 2, 3, 4$. Specify the type of your proposal filter $h[n]$.
(c) Propose a third order anticausal FIR filter $g[n]$ with $g[0] = 1$ so that the output signal $z[n] = g[n] * x[n], n = 1, 2, 3, 4$ is zero-mean. Specify the type of your proposal filter $g[n]$.
11. Given $x_1[n] = \{-1, -2, 0, 1, -1, 3\}$ and $x_2[n] = \{1, 1, -3, 2, 1, -3, 1\}$, both W.S.S. and starting at index zero.
(a) Which signal is more likely white noise? Explain your answer by computing a suitable statistical measure for both signals and comparing it.
(b) Consider $y_1[n] = h[n] * x_1[n]$, where $h[n]$ is an FIR filter. Is $y_1[n]$ W.S.S.? Explain your answer.
(c) Let $w[n]$ be a W.S.S. zero mean white noise signal with variance 1. Compute its spectrum $S_w(e^{j\omega})$. Comment on your result.
12. Given the sequence of W.S.S. random numbers $x[n] = \{1, 1, 2, 3, 5\}$ starting at index zero.
(a) A second order LMMSE predictor $\hat{x}[n] = a_1 x[n-1] + a_2 x[n-2]$ shall be determined by following method: Find a_1, a_2 such that $g(a_1, a_2) = \sum_{n=2}^{4} (x[n] - \hat{x}[n])^2$ is minimized. Do NOT use the ACNE (no autocorrelations!). Hint: Use derivatives $\frac{dg(a_1,a_2)}{da_1}$ and $\frac{dg(a_1,a_2)}{da_2}$.
(b) Compute the predictions $\hat{x}[2], \hat{x}[3], \hat{x}[4]$ and $\hat{x}[5]$ using the predictor from part (a).
(c) Consider $y[n] = \frac{\sin n}{n}, n \geq 0$. Is $y[n]$ W.S.S.? Explain your answer.

References

References

1. W. K. Chen, Fundamentals of Circuits and Filters, Series: The Circuits and Filters Handbook, Third Edition, CRC Press, 2009.
2. A. V. Oppenheim, R. W. Schafer, J.R. Buck, Discrete-Time Signal Processing, Prentice Hall, 1989.
3. James H. McClellan, Ronald W. Schafer, Mark A. Yoder, Signal Processing First, Prentice-Hall, Inc., 22003.
4. S. Mitra, Digital Signal Processing, McGraw Hill, 2001.
5. J. G. Proakis, D. G. Manolakis, Digital Signal Processing: Principles, Algorithms, and Applications, Prentice Hall, 1996.
6. R. J. Larsen, M. L. Marx, An Introduction to Mathematical Statistics and Its Applications, Prentice-Hall, Inc., 1981.
7. Gilbert Strang and T. Nguyen, Wavelets and Filter Banks, Welleslay-Cambridge Press, 2000.
8. A. E. Cetin, Omer N. Gerek, Y. Yardimci, ”Equiripple FIR Filter Design by the FFT Algorithm,” *IEEE Signal Processing Magazine*, Vol. 14, No. 2, pp.60-64, March 1997.
9. Athanasios Papoulis, S. Unnikrishna Pillai, Probability, Random Variables and Stochastic Processes, Foruth Edition, 2002.