Digital Signal Processing

DR DENNIS DENG

Department of Electronic Engineering

La Trobe University

Preface

This is the first draft of the lecture notes for ELE32DSP. Your comments and feedbacks are welcome. Send your comments/corrections to: d.deng@latrobe.edu.au

The first draft was finished on August 17, 2001. In version 1.1, some typing errors identified by Mr Finn Thain (ELE32DSP student) are fixed. Last revised: July 2008.

I intend to use it as the key teaching material for this subject. However, it is *not* a text book. It can not replace the recommended text book.

The textbook for this subject is "DSP First A Multimedia Approach", by J. H. McClellan, R. W. Schafer and M. A. Yoder. This book is published by Prentice Hall.

Contents

1 Introduction

In this lecture, we try to answer the following questions relating to digital signal processing (DSP). What is it about? What is the advantage of using DSP? What are the major applications of DSP? Can you show me a few simple examples?

1.0.1 Signals and systems[1]

What are signals and systems? There are different type of signal processing systems. In this subject, we are only concerned with linear time-invariant systems.

1.0.2 Advantages of using DSP

Compared with analog signal processing, DSP has a number of advantages:

- flexibility in system reconfiguration
- better control of accuracy
- perfectly reproducible
- no performance drift with temperature or age
- signal processors are becoming more powerful and cheaper

1.0.3 Applications of DSP

Digital signal processing technologies have found a wide range of applications in telecommunications, defense, biomedical engineering, aviation, home electronic appliances and many other areas. In telecommunications, DSP is used in applications such as echo cancellation, ADPCM transcoders and video conferencing. In biomedical engineering, DSP techniques have been developed for patient monitoring, EEG brain signal mapping, ECG analysis and the storage and enhancement of X-ray images.

We also use DSP technologies in our daily life. Digital mobile phone and DVD player are two examples. The digital mobile phone involves, among other complicated signal processing tasks, converting the voice signal into digital data and performing data compression. It must also perform data de-compression and convert the digital data into the voice signal. The key element of a DVD player is the decoder that decodes the digital audio and video data encoded according to the MPEG-2 standard.

1.0.4 Notations and representations

A discrete signal is denoted $x[n]$ where $n$is an interger. This a general representation. When n represents a specific number, then $x[n]$ is the value of the signal at location n. A discrete signal is sometimes called a sequence or a vector. For example, a data sequence is given by

$$x[n] = \begin{cases} \cos(\frac{\pi}{3}n), & n = 0,1,...9 \\ 0, & n < 0 \text{ and } n > 9 \end{cases}$$

We can also represent signal by its numerical values

$$x[n] = \{1, 0.5, -0.5, -1, -0.5, 0.5, 1, 0.5, -0.5, -1\}$$

[1] This part is still being written.

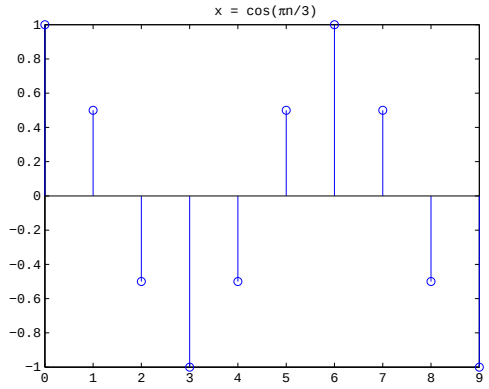


Figure 1: A matlab plot of the cosine signal.

In this case it is important to indicate the location where $n = 0$. This is an important reference point. I will use an up-arrow to point to that signal value. When the signal is represented in this way, it is assumed that signal values at unspecified locations are zero. We make this assumption throughtout this unit. We can also visualize the signal in Matlab by using the following codes:

```
n = 0 : 9;
x = cos( pi * n / 3 );
figure;
stem( n, x );
title(' x = cos(\pin/3) ' ); % \pi is π
```

The figure is shown in fig. 1below.

Therefore, a discrete signal can be represented by using an equation, by specifically listing signal values and by using a figure.

1.0.5 Examples

A simple linear filter is the moving average filter that is defined as:

$$y[n] = \frac{1}{3}\sum_{k=-1}^{1} x[n-k]$$

where $x[n]$ and $y[n]$ represents the input and the output signals of the filter (system) at index n. A simple nonlinear filter is the median filter that is defined as:

$$y[n] = med\{x[n-1], x[n], x[n+1]\}$$

where the output signal $y[n]$ is the median value of the three input samples. For example, if $x[n-1] = 7$, $x[n] = 2$, $x[n+1] = 11$, then $y[n] = med\{7,2,11\} = 2$.

If a data sequence, $u[n] = \{0,0,0,0,0,5,5,5,5,5\}$, is corrupted by noise and the resulting noisy data sequence is given by $x[n] = \{0,7,0,0,0,10,5,5,-3,5\}$. We can apply the above two filters to the noisy signal to see if they can remove the noise. Figure 2 shows A plot of the original signal, the noisy signal and the results of the two simple filters.

We can also apply these two filters to digital images. Results are shown in Figure 3. We can see that the median filter is useful for filtering out impulse noise.

The moving average filter is a useful filter for a number of noise filtering applications. For example, a cosine signal, given by $u[n] = cos(2\pi\omega_0 + 0.1\pi)$, is corrupted by Gaussian noise. The noisy

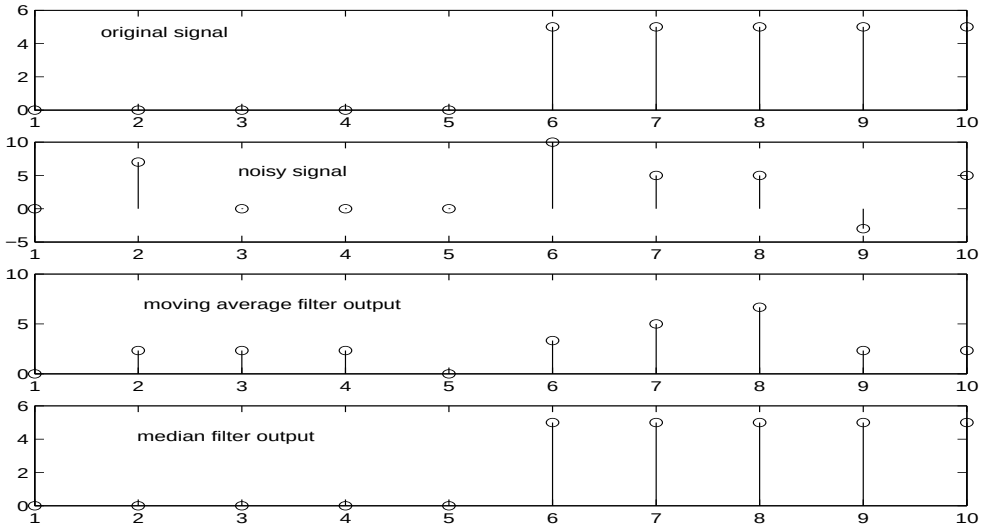


Figure 2: A plot of the original signal, the noisy signal and the results of the two simple filters.

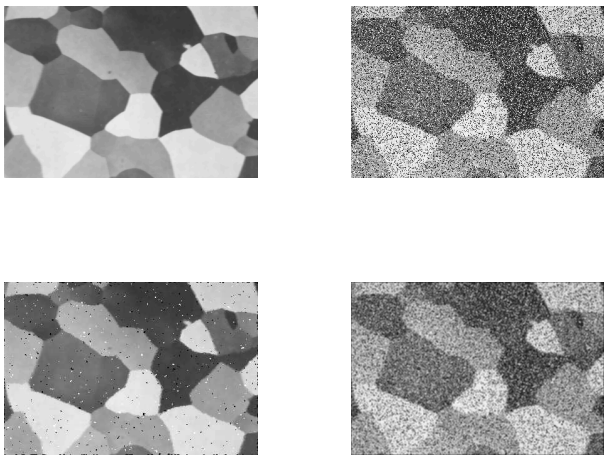


Figure 3: From left to right, top to bottom, the original image, the noisy image, output of a median filter, output of a moving average filter.

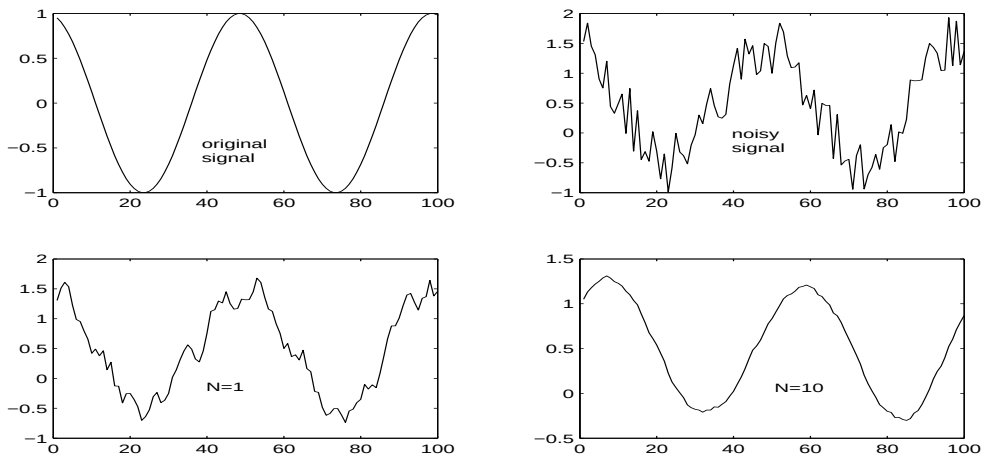


Figure 4: The outputs of the moving average filter with N=1 (bottom left) N=10 (bottom right)

signal is $x[n] = u[n] + r[n]$, where $r[n]$ represents the noise signal. We can use the moving average filter to remove the noise. A generalized form of the filter is

$$y[n] = \frac{1}{2N+1} \sum_{k=-N}^{N} x[n-k]$$

where N controls the number of samples involves in the averaging operation. Figure 4 shows two cases with $N = 1$ and $N = 10$, respectively.

1.1 Text book and reference books

The text book for this subject is:

J. H. McClellan, R. W. Schafer and M. A. Yoder, *DSP First A Multimedia Approach,* Prentice-Hall, 1998

There are many good books on DSP. The following is a list of a few of them

- R. D. Strum and S. E. Kirk, *First Principles of Discrete Systems and Digital Signal Processing*, Addison Wesley, 1988
- A. V. Oppenheim and R. W. Schafer, Discrete-time Signal Processing, Prentice-Hall, 1989
- S. K. Mitra, *Digital Signal Processing A Computer Based Approach*, McGraw Hill, 1998
- J. G. Proakis and D. G. Manolakis, *Digital Signal Processing Principles, Algorithms and Applications,* 3rd ed., Prentice-Hall, 1996

1.2 How to study

To pass the subject, you must do all assignments and laboratory exercises. They are designed to reinforce your understanding of the basic concepts and techniques of DSP. They are also directly related to the exam questions. The CDROM bundled with the text book contains many interesting demonstrations and exercises. You should go through some of them (at least those suggested by the lecturer). At the end of each lecture, there are a few simple self-test questions to remind you the major concepts or techniques introduced in the lecture. You should also go through those questions. Finally, there is an old Chinese proverb:

I hear, I forget
I see, I remember
I do, I understand

1.3 Self-test Exercises

A limitation of the simple moving average filter is that in order to calculate the "current output" at index n, "future" input samples are used (from index n to n+N).

- Modify the filter such that only the past and current input samples are used.
- The moving average filter can also be implemented in a *recursive* form: $y[n] = ay[n-1] + b(x[n] - x[n-N])$. Determine the two constants a and b.
- Compare the number of operations required for both implementations.

Answer:

- $y[n] = \frac{1}{N}\sum_{k=N-1}^{0} x[n-k]$
- $Ny[n-1] = \sum_{k=N-1}^{0} x[n-k-1]$, $N(y[n] - y[n-1]) = x[n] - x[n-N]$, $a = 1$, $b = \frac{1}{N}$
- Direct implementation: (N-1) addition and one multiplication. Recursive implementation: 2 addition, one multiplication.

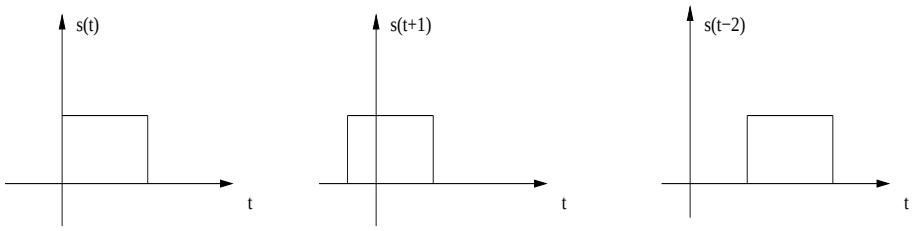


Figure 5: The time-shifting of a signal

2 A review of cosine signals and complex numbers

This lecture is a review of cosine signals and complex numbers and is based on Chapter 2 of the text book. It is assumed that you have a basic knowledge of trigonometric functions. This lecture is thus focused on the concepts and a number of Matlab commands that are related to signal processing.

2.1 Cosine signals

A cosine signal is defined as:

$$x(t) = A\cos(\Omega_0 t + \phi)$$

where A is called the amplitude, Ω_0 is called the radian frequency[2] and ϕ the phase shift. The cosine signal oscillates between +A and -A. Its radian frequency is related to the cyclic frequency:

$$\Omega_0 = 2\pi f_0$$

which determines how fast it oscillates. The period of the signal is $T_0 = \frac{1}{f_0}$. For example, when the frequency is 10 kHz, its radian frequency is 20,000π rad/sec and the period is $T_0 = 0.1$ ms. Note that this cosine signal is an *analog* signal. Its frequency can go very high. Later on we will see that for a discrete cosine signal, its highest frequency is π.

The phase shift is related to the time-shift of a signal. Let $s(t)$ represent a signal. Then $s(t-t_0)$ represents a time-shifted version of the original signal. When $t_0 > 0$, the signal is delayed in time (shifted to the right). When $t_0 < 0$, the signal is advanced in time (shifted to the left).

It is easy to determine the relationship between the time-shift and the phase shift of a cosine signal $x(t) = \cos(\Omega_o t)$. A time-shifted signal is

$$x(t-\tau) = \cos(\Omega_0(t-\tau)) = \cos(\Omega_0 t - \Omega_0\tau)$$

Therefore the phase shift is: $\phi = -\Omega_0\tau$. Another interpretation is that given the phase shift, the time delay is:

$$\tau = -\frac{\phi}{\Omega_0} = -\frac{\phi}{2\pi f_0}$$

Note that if the phase shift caused by the time shift is greater than 2π, then it must be converted to the principal value between $-\pi$ and π.

A discrete cosine signal can be generated by evaluating the analog cosine signal at a discrete set of times, $t_n = nT_s$, where n is an integer, T_s is called the sampling period. By sampling, we obtain a sequence of data:

$$x[n] = x(nT_s) = \cos(2\pi f_0 nT_s + \phi)$$

The index n represents the order of the data samples. Unless we know T_s, the time between any two samples, for example, between $x[n-1]$ and $x[n+3]$ is unknown.

[2]We use Ω to represent the radian frequency of an analog signal and ω to represent that of a discrete time signal. This notation is different from that used in the text book.

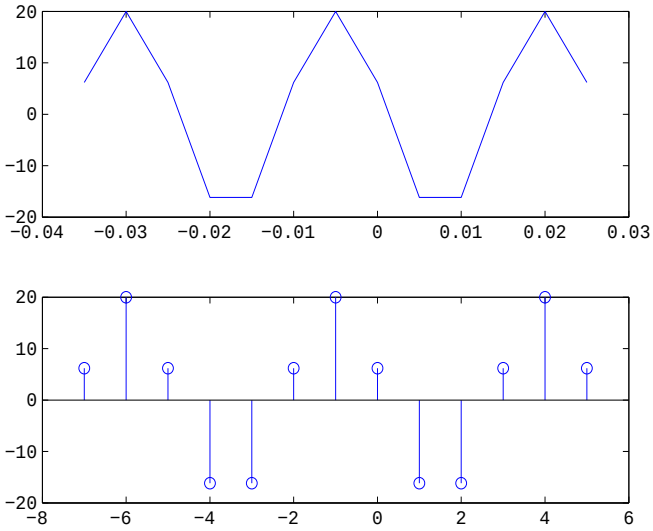


Figure 6: Discrete cosine signal, sampling period = 0.005 sec. Top: using the plot command. Bottom: using the stem command. Note that when we use the stem command, we plot the signal against the location index n.

In Matlab, a sequence of data is usually stored as a row or a column vector. Several related or unrelated data sequences can be stored in a matrix. The following Matlab code shows you how to generate and plot cosine signals:

```
n = -7 : 5 ;                          %index
Ts = 0.005;                           %sampling period
tn = n * Ts ;
xn = 20 * cos ( 80 * pi * tn + 0.4*pi);   % f0 = 40, φ = 0.4π
subplot ( 211 )                       %put more than one signal in on figure
plot ( tn , xn )                      %plot the signal against time
subplot ( 212 )
stem ( n, xn )                        %it is different from plot
```

2.2 Complex numbers

There are two representations for complex numbers.

- Rectangular form: $z = x + jy$
- Polar form: $z = re^{j\theta}$

where x, y, r and θ are real numbers and $j = \sqrt{-1}$. These two representations are related by:

$$r = \sqrt{x^2 + y^2}$$

and

$$\theta = \arctan(\frac{y}{x})$$

A geometric view is shown in Figure 7.

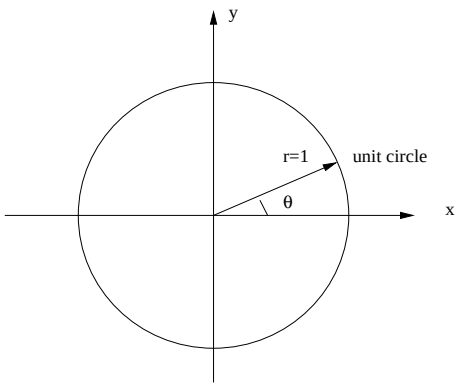


Figure 7: A geometric view of a complex number.

addition	$z_1 + z_2 = (x_1 + x_2) + j(y_1 + y_2)$
multiplication	$z_1 z_2 = r_1 r_2 e^{j(\theta_1+\theta_2)}$
conjugate	$z^* = re^{-j\theta} = x - jy$
power	$z^n = r^n e^{jn\theta}$
roots of unity	$z^N = 1$, $z_n = e^{j2\pi n/N}$, $n = 0, 1, ..., N-1$
unit circle	$\|z\| = 1$

Euler's formula is also very useful for manipulating complex numbers:

$$e^{j\theta} = \cos\theta + j\sin\theta$$

For example, it is easy to see that:

$$\cos\theta = \frac{e^{j\theta} + e^{-j\theta}}{2}$$

and

$$\sin\theta = \frac{e^{j\theta} - e^{-j\theta}}{2j}$$

The following is a list of operational rules for complex numbers.

From Fig. 7, we can see that a complex number corresponds to a point in the 2-D space using x and y as the coordinates. Remember the y-axis has a unit $j = \sqrt{-1}$. A complex number also represents a vector that points from the origin $(0,0)$ to the point (x,y). The length of the vector is r which is always greater than or equal to zero. The angle is θ. If r is fixed, say $r = 1$, then the vector is determined by the angle. Two complex numbers, denoted by $z_1 = re^{j\theta_1}$ and $z_2 = re^{j\theta_2}$ are the same if $\theta_1 = 2k\pi + \theta_2$ for $k = 0, 1, 2, ...$

Here is the geometric interpretation. If we modify a vector z_2 by only adding θ_0 to its angle, then resulting vector is rotated from its orginal position to a new position where the angle is $\theta_0 + \theta_1$. Now if $\theta_0 = 2\pi$, then there will be an anti-clockwise full circle rotation. The new vector is in the same position as the original vector. You can imagine that $\theta_0 = 2k\pi$ corresponds k-times of such rotations. Hence, we can regard θ_1 and θ_2 as the same. We will come back to this point when we discuss the concept of a frequency for a discrete signal.

Taking this geometrical view also helps us to understand better about the multiplication of two vectors. It has two effects: the length of one is scaled by that of the other and the angle is rotated. What can you say about the unit circle? Well, it is simply formed by rotating a vector $z = 1$ from $\theta = 0$ to 2π. Therefore, any point at the unit circle satisfies $|z| = 1$. An equivalent representation is $z = e^{-j\theta}$ for $\theta = 0$ to 2π. (The circle has a riduis of a unit, thus it is called a unit-circle). Later on we will see that the unit circle plays a very important role in signal processing.

Addition, subtraction and division of two complex numbers have interesting geometric interpretations.

In DSP, we are interested in the frequency response of a filter. The frequency response, denoted by $H(\omega)$, is the Fourier transform of the impulse response, denoted by $h[n]$. The frequency response is usually a function of the complex variable $e^{j\omega}$. As such, it can be represented as $H(\omega) = |H(\omega)|e^{j\angle H(\omega)}$, where the amplitude response is $|H(\omega)|$ and the phase response is $\angle H(\omega)$. For the amplitude response, it can be calculated as follows

$$|H(\omega)|^2 = H(\omega)H^*(\omega)$$

where $H^*(\omega)$ is the complex conjugate of $H(\omega)$. For example, if we have $H(\omega) = 1 - e^{-j\omega}$, then $H^*(\omega) = 1 - e^{j\omega}$ and

$$\begin{aligned} |H(\omega)|^2 &= H(\omega)H^*(\omega) \\ &= 2(1-\cos(\omega)) \end{aligned}$$

2.3 Self-test Exercises

- Determine and plot the roots of the equation: $z^6 = 1$.
- Determine the amplitude and phase for the complex number $z = 1 + e^{-j2\omega_0}$

Answer:

- $z_n = e^{jn\pi/3}$, $n = 0, 1, ..., 5$.
- $z = 2e^{-j\omega_0}\frac{e^{j\omega_0}+e^{-j\omega_0}}{2} = 2\cos\omega_0 e^{-j\omega_0}$

3 Matlab

This lecture is based on Appendix B of the text book, where you can find more examples.

3.1 Matrix operations

A good way to learn to program in Matlab is to use it. Matlab provides a very useful on-line help system. When you are unsure about a command, just type **help**[3] command_name. The basic variable type in Matlab is a matrix. A signal such as digital audio is stored as a vector. A digital image is stored as matrix. Elements of a matrix should be separated by a space and different rows separated by a semicolon. An element or part of a matrix can be accessed by giving the row and column index. The following is a segment of a Matlab code that illustrates how to create and manipulate a matrix.

```
% comments                    % comments starts with a "%"
t = 0 : 0.001 : 1;            % create a row vector of 1001 elements whose
                              %values ranges from 0 to 1
y = cos ( 2 * pi * 50 * t);   %create another row vector of a cosine signal
                              %sampled at 1000 Hz.
x = rand ( 1 , length ( t ) ); %rand is a command that generates random numbers.
yn = y + 0.5 * x ;            %we add noise to the cosine signal.
plot ( y ( 1 : 100 ) , ' r ' ) %we want to look at the first 100 samples
                              %ploted in red color
hold on                       %next plot will be in the same figure
plot ( yn ( 1 : 100 ) , ' y ' )
figure ( 2 )                  %want to plot in a new figure
plot ( y ( 1 : 4 : 1000 ) )   %we are only interested in samples indexed 1,5,9,...etc
z = [ y ; yn ];               %creat a matrix of size 2 x 1001.
                              %don't forget to put a "; "at the end !
z ( 2 , 33 )                  %what is the value of the element at (2,33)
whos                          %similar to "dir" in DOS
M = [ 1 2 3 ; 4 5 6 ; 7 8 9 ]; % a new (3 x 3)matrix
E = M' ; F = inv ( M );       %transpose and inverse
F = M .* E;                   %element-wise multiplication
G = M * F;                    %matrix multiplication
H = abs ( F ) > 3 ;           %an element of H is 1 if the absolute
                              % value of the respective element in F > 3
```

So you can see some of the data processing features of Matlab. For example, to implement the last operation, a threshold operation on each element of a matrix, with other high level languages such as C, you would need to use two for-loops. But in Matlab, it is just a command line.

Actually, Matlab is optimized for such "vectorized" operation. A vectorized operation such as this last operation runs in most case faster (or a lot more faster in some cases) than using the for-loop. Therefore, you should avoid using for-loops in Matlab. Here is one more example that shows the vectorized operation. The task is to calculate the *entropy* of an information source that contains N symbols:

$$E = -\sum_{n=1}^{N} p[n]\log_2(p[n])$$

where $p[n]$ is the probability of the nth symbol. The Matlab code is:

[3] A Matlab command is shown in `typewriter` font.

```
%define the probability pn first
pn = [ 0.1 0.2 0.05 0.05 0.3 0.3 ];
E = -sum ( pn .* log2 ( pn ) )
```

Matlab also provides program flow functionality such as **`if`** statements, **`while`** loops and **`for`** loops.

3.2 File input and output

Reading a data file into a matrix (vector) or writing the data to a file are similar to what you would do in the C language. The following is a list of commands: **`fopen, fread, fwrite, fscanf, fprintf`**. There are two other commands which are very handy for loading data from or save variable(s) to a file: **`load, save`**.

For example, a data file named "speech.dat" contains raw digital speech data sampled at 11 kHz and quantized at 8 bits/sample. You can use the following Matlab code to load the data into a Matlab variable:

```
a = fopen ( 'speech.dat', 'r')          %returns -1 if failed
b = fread ( a , 'uchar');               %the data is stored as unsigned char
sound ( b, 11000 )                      %can you hear the sound ?
```

3.3 Matlab script and function

You can edit and save your Matlab code into a text file by using a text editor. The file extension must be .m and the script can be run by simply typing the file name. You can also write your own Matlab function. The file name must be the same as the function name, and it must be with an extension .m . The following is an example. The function, called threshold, takes a vector or a matrix and a value as the input and produces a vector or a matrix as the output. This function is saved as a text file named threshold.m .

```
function y = threshold(x, t)
%threshold y = x>=t ;
%usage:
%       y = threshold(x, t)
%where:
%       y = output
%       x = input
%       t = threshold
y = ( x >= t ) ; %note >= is one operation, no space in between
```

3.4 Self-test Exercises

- Generate a discrete random signal of 500 samples. Break the signal into 5 segments of equal length, ie, the first segment is with data sample from index 1 to 100, etc. Plot these segments in the same figure. Calculate the mean square value of each segment. The mean square value is defined as: $\frac{1}{N}\Sigma_{n=1}^{N} x^2[n]$. You can use commands: **`mean, subplot, rand`**.
- Write a Matlab function to calculate the mean square value of a vector or a matrix.

4 Signal and system representation in the time domain

We will spend two lectures on this section. In the first lecture, we will discuss three topics:

- the representation of a discrete time signal by the unit sample sequence,
- the frequency of a discrete complex exponential signal, and
- some properties of a discrete time system.

In the second lecture, we will cover another three topics:

- the LTI system,
- unit sample response and convolution, and
- linear constant-coefficient difference equation.

4.1 Discrete time signal

A discrete time signal (we will call it a signal later on if no confusion arises) is a function of an independent variable that is an integer. For example, the signal given by:

$$x[n] = \cos(2\pi f_0 n T_s + \phi), \;\; -\infty < n < \infty$$

defines a cosine signal where the index is n. If the signal is of finite duration (or with finite number of samples), then we can use the representation such as

$$x[n] = \{10, \underset{\uparrow}{1}, 5, -9\}$$

In this case, unless the zero position of the sample is indicated, we assume that the first sample is $x[0] = 10 \;\; (n = 0)$.

A signal can be shifted (delayed or advanced). For example, when the signal is delayed by one sample, it is represented as $y_1[n] = x[n-1] = \{0, 10, \underset{\uparrow}{1}, 5, -9\}$. Note that in this case, the first sample of the signal becomes $y_1[0] = x[0-1] = x[-1] = 0$. Here we assume that $x[-1] = 0$ because it is not specified in the original signal. In fact, through out this semester we will use the assumption that if a signal sample is not specified then it is assumed zero value.

Similarly, an advance of the signal by one sample is represented as: $y_2[n] = x[n+1]$. It is important to note that in this case, we have $y_2[-1] = x[-1+1] = x[0] = 10$,$y_2[0] = 1$, $y_2[1] = 5$, and $y_2[2] = -9$.

Here is another example. We define a new signal as $y_3[n] = x[-n]$. We have $y_3[0] = x[0] = 10$, $y_3[1] = x[-1] = 0$, $y_3[-1] = x[1] = 1$,$y_3[-2] = x[2] = 5$ and $y_3[-3] = x[3] = -9$. A graphical representation of this signal together with other examples are shown in Figure 8. Here is a question for you. Can you determine the signal[4] given by $y_4[n] = x[1-n]$?

A unit sample sequence is defined as:

$$\delta[n] = \begin{cases} 0, & n \neq 0 \\ 1, & n = 0 \end{cases}$$

The unit sample sequence, like the impulse signal in analog signal processing, plays an important role in the representation and analysis of signals and linear time invariant systems. For signal representation, we consider a simple case where a signal has only two data samples: $x[0] = 3$, $x[1] = 7$. It can easily be seen that this signal can be represented as:

$$x[n] = 3\delta[n] + 7\delta[n-1]$$

[4] ANSWER: $y[0] = x[1]$, $y_4[1] = x[0]$, $y_4[-1] = x[2]$, $y_4[-2] = x[3]$. It is a delay by one sample version of the signal given by $y_3[n] = x[-n]$.

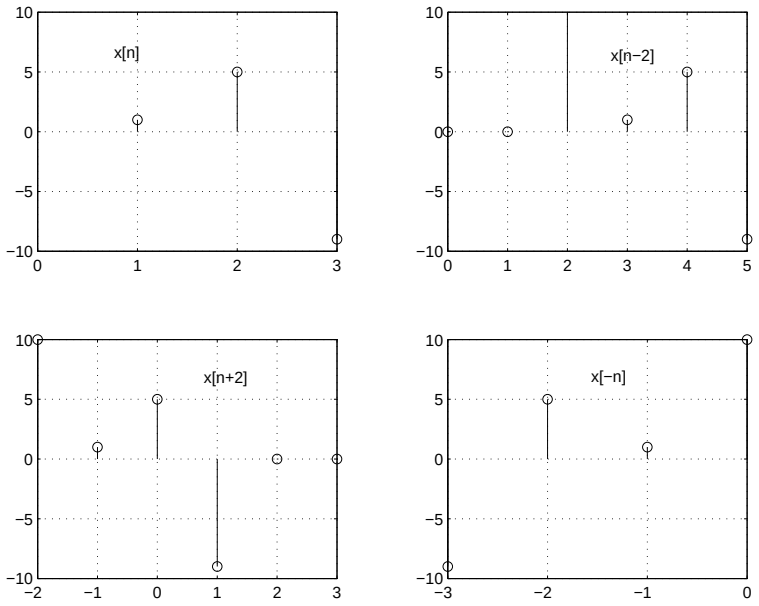


Figure 8: A plot of four signals, top left $x[n]$, top right $x[n-2]$, bottom left $x[n+2]$, bottom right $x[-n]$.

because according to the definition of the unit sample sequence, we have

$$x[0] = 3\delta[0] + 7\delta[-1] = 3*1 + 7*0 = 3$$

In general, a discrete signal can be represented as a linear combination of unit sample sequences:

$$x[n] = \sum_{m=-\infty}^{\infty} x[m]\delta[n-m]$$

Note that $x[m]$ inside the summation represents a data sample at location (index) m and $\delta[n-m]$ is a delayed version of the unit sample sequence. It is delayed by m samples such that the only non-zero sample is at index m. For the above example, we have:

$$x[n] = \Sigma_{m=0}^{1} x[m]\delta[n-m]$$

Another useful data sequence is the unit step sequence:

$$u[n] = \begin{cases} 0, & n < 0 \\ 1, & n \geq 0 \end{cases}$$

4.2 Frequency and periodic signal

In this section, we discuss the concept of frequency for the discrete complex exponential signals which are closely related to discrete cosine or sine signals. The discrete complex exponential signal is defined as:

$$x[n] = Ae^{j(\omega n+\phi)} = A\cos(\omega n+\phi) + jA\sin(\omega n+\phi)$$

where the quantities A, ω and ϕ are called the amplitude, frequency and phase-shift, respectively. An important property of the discrete complex exponential signal is that when its frequency adds $2k\pi$ the signal remains the same. This can be seen by considering a new signal $x_1[n]$ with frequency $\omega_1 = \omega + 2k\pi$:

$$x_1[n] = Ae^{j(\omega_1 n+\phi)} = Ae^{j(\omega n+\phi)}e^{j(2kn\pi)} = x[n]$$

since

$$e^{j(2kn\pi)} = \cos(2kn\pi) + j\sin(2kn\pi) = 1$$

Therefore, we only need to consider frequency in the interval of $0 \leq \omega < 2\pi$ for the discrete complex exponential signal. This is different from analog signals, where a very high frequency is possible.

Next we take a look at the cosine signal

$$x[n] = \cos(\omega n + \phi)$$

If we make a new cosine signal, denoted by $x_1[n]$, with the same phase shift and the frequency $\omega_1 = \omega + 2k\pi$, where k is an integer. We can easily see that $x[n] = x_1[n]$, the two signals are exactly the same. This implies that the two frequencies are indistinguishable. What is the interpretation of a high frequency and a low frequency for the discrete time signal?

Since frequency is used to indicate the rate of oscillation, a higher frequency means faster oscillation. When the frequency is zero, there is no oscillation at all. So the lowest frequency is zero ($\omega = 2k\pi$). This can be verified by:

$$x[n] = \cos(2kn\pi + \phi) = \cos\phi = constant.$$

What is the fastest oscillation for a cosine signal? From Figure 9, which shows cosine signals with different frequencies, we can see that π (or $\pi + 2k\pi$) is the highest frequency. This is because when the frequency gradually increases from 0 to π, the oscillation becomes faster. You can imagine (show it with Matlab, see self-test exercise) that as the frequency gradually increases from π to 2π, the oscillation becomes slower. In fact, let us consider the cosine signal $x[n] = \cos(\omega n)$ and let $\omega = \pi + \omega_0$ ($0 < \omega_0 \leq \pi$). We can rewrite it as $\omega = 2\pi - (\pi - \omega_0) = 2\pi - \omega_1$ where $\omega_1 = \pi - \omega_0$. Therefore

$$\begin{aligned} x[n] &= \cos(\omega n) \\ &= \cos((\pi + \omega_0)n) \\ &= \cos(2\pi n - (\pi - \omega_0)n) \\ &= \cos((\pi - \omega_0)n) \\ &= \cos(\omega_1 n) \end{aligned}$$

As such, we can say the two frequencies ω and ω_1 are the same. When ω is increased from π to 2π, ω_0 is increased from 0 to π, and ω_1 is decreased from π to zero. Therefore, the highest frequency is π and the lowest frequency is 0.

One student asked me the following question: I can understand the development of the idea using the cosine signal, but what would be the conclusion if we use a sine signal? This is a good question. Actually, using the cosine signal is a way to illustrate different speeds of oscillation relating to different settings of ω. It is intuitively understandable that the fastest oscillation is represented by the signal $x[n] = \{1, -1, 1, -1, 1, -1, ...\}$ which is the same as the cosine signal $cos(\pi n)$. However, when we use a sine signal, $x[n] = sin(\omega n)$, the fastest oscillation we can get is by setting $\omega = \pi/2$. This results in a signal $\{0, 1, 0, -1, 0, 1, 0, -1, ...\}$ which is clearly not oscillating as fast as $cos(\pi n)$. We will have a deeper understanding of this issue after we discuss the sampling theorem.

Another issue is related to periodic signals. In the continuous case, a cosine signal, denoted by $x(t) = \cos(\Omega t + \phi)$, is always a periodic signal with a period of $2\pi/\Omega$. Is this true in the discrete case ? Let us define a discrete periodic signal as:

$$x[n] = x[n + N]$$

where N is an integer and is called the period of the signal. To see if a discrete cosine signal $x[n] = \cos(\omega n + \phi)$ is periodic, we perform the following operation (replacing n by $n = N$) :

$$\cos(\omega(n + N) + \phi) = \cos(\omega n + \phi + \omega N)$$

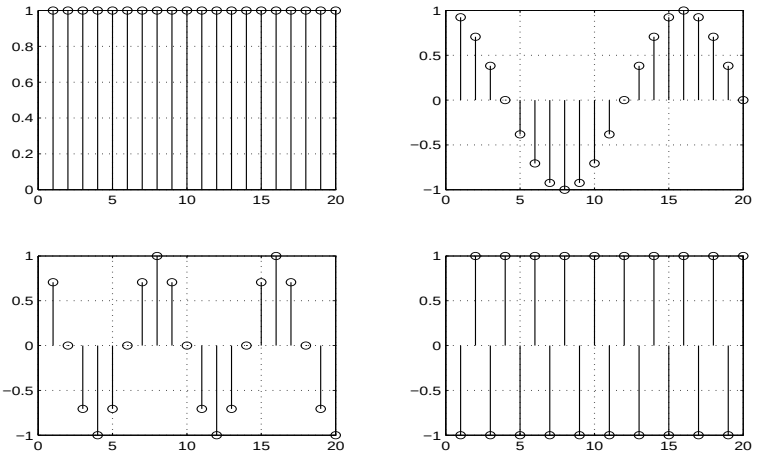


Figure 9: Plots of $cos(\omega n)$. Top left $\omega = 0$. Top right $\omega = \pi/8$. Bottom left $\omega = \pi/4$. Bottom right $\omega = \pi$.

If the signal is indeed a periodic signal, then it requires that:

$$\omega N = 2k\pi$$

where k is an integer. The smallest integer value of N satisfying this condition is called the fundamental period of the signal. Therefore, for a cosine signal to be a periodic signal with a period N, its frequency must be:

$$\omega = \frac{2k}{N}\pi$$

where $k = 0, 1, ..., N-1$. So there is a set of N different frequencies (ω_k) that satisfies the condition. In other words, for a given period N, there are only N different frequencies given by the above equation such that the cosine signal is periodic. Therefore, cosine signals or a complex exponential signals are not necessarily periodic with a period of $2\pi/\omega$, and depending on the value of the frequency, they may not be periodic at all.

For example, when $N = 4$, the frequencies are: $\omega_0 = 0$, $\omega_1 = \pi/2$, $\omega_2 = \pi$, $\omega_3 = 3\pi/2$. You can verify that a cosine signal with one of these frequencies is indeed a periodic signal with a period of 4. When $\omega = \pi/7$, the smallest non-zero value of N that satisfies the condition is 14. This corresponds to $k = 1$.

On the other hand, when $\omega = 1$, we can easily see that the corresponding cosine signal is not periodic at all, because there is no such integer value of N that satisfies

$$\cos(\omega(n+N)+\phi) = \cos(n+N+\phi) = \cos(n+\phi)$$

Since the index n is dimensionless, the unit of the frequency is radians. However, if n is designated a unit of samples, then the unit for the frequency is radians/sample. If, just as in the analog case, we define $\omega = 2\pi f$, then f is the frequency in cycles/sample. For example, a periodic cosine signal with a frequency of $\omega = \pi/5$ corresponds to $f = 1/10$ cycles/sample. This means that for every 10 samples, the signal repeats itself. In other words, the fundamental period is $N = 1/f = 10$.

Summary

The lowest and highest frequency of a discrete signal is 0 and π. A discrete cosine is not necessarily a periodic signal. It is periodic with a period of N only when its frequency satisfies $\omega = \frac{2k}{N}\pi$ for $k = 0, 1, ..., N-1$.

4.3 Discrete time system

A discrete time system is defined mathematically as a transformation or an operator that maps an input signal $x[n]$ into an output signal $y[n]$:

$$y[n] = H\{x[n]\}$$

An ideal delay system produces an output which is a delayed version of the input

$$y[n] = x[n-N]$$

The moving average filter and the median filter are other examples of a system.

A system is *memoryless* if the output $y[n]$ at index n only depends on the input $x[n]$ at the same index. For example, the moving average filter and the median filter are said to have memory, because the output depends on the input sample $x[n]$ as well as other input samples. The system given by:

$$y[n] = ax[n] + b$$

is memoryless.

A system is *causal* if for every choice of n_0 the output sequence value at index $n = n_0$ depends only on the input sequence value for $n \leq n_0$. This implies that if the sequence value at index n_0 is regarded as the "current" value, then the current output value depends only on the past and current input sequence values. For example, the following moving average filter is non-causal, because the current output depends on the "future" input value $x[n+1]$.

$$y[n] = \frac{1}{3}(x[n-1] + x[n] + x[n+1])$$

However, we can change it into causal in this way:

$$y[n] = \frac{1}{3}(x[n-2] + x[n-1] + x[n])$$

In this case, the current output $y[n]$ only depends on the past input $(x[n-2], x[n-1])$ and the current input $x[n]$[5].

A system is *stable* in the bounded-input bounded-output (BIBO) sense, if and only if every bounded input signal produces a bounded output signal. A bounded signal is defined as:

$$|x[n]| < \infty \quad for\ all\ n$$

For example, the unit step signal $u[n]$ is bounded. If it is used as an input to a system defined as:

$$y[n] = \sum_{k=-\infty}^{n} u[k] = \begin{cases} 0, & n < 0 \\ n+1, & n \geq 0 \end{cases}$$

then we can easily see that the output signal is unbounded. Thus the system is unstable.

[5]Please note that there may be a confusion due to the notation that we use to represent the signal. The term $x[n-3]$ has two meanings: (1) if it appears in an equation like $y[n] = x[n-3]$, then it means a delayed version of the WHOLE signal, (2) if it appears in an equation like $y[n] = \frac{1}{3}(x[n-3] + x[n-1] + x[n])$, then it means the value of the signal sample at index $(n-3)$.

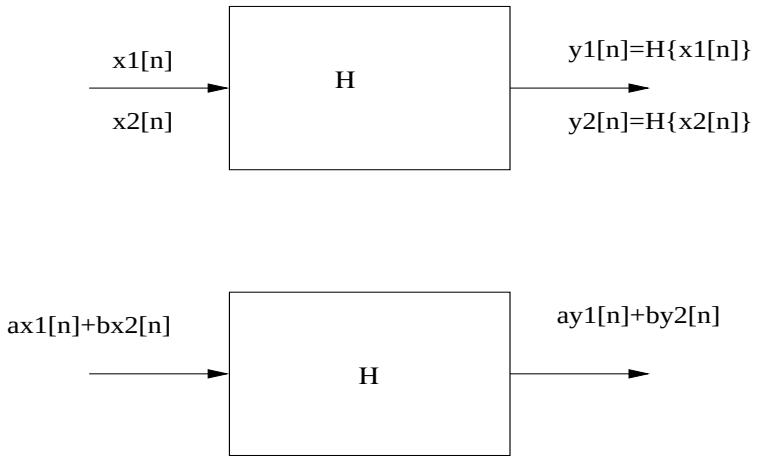


Figure 10: A linear system

4.4 Linear time-invariant system

A system is *linear* if its outputs for two signals are

$$y_1[n] = H\{x_1[n]\}$$

and

$$y_2[n] = H\{x_2[n]\}$$

then for an input which is a linear combination of these two signals:

$$x[n] = ax_1[n] + bx_1[2]$$

its output is

$$\begin{aligned} y[n] = &\quad H\{x[n]\} \\ = &\quad H\{ax_1[n]\} + H\{bx_2[n]\} \\ = &\quad ay_1[n] + by_2[n] \end{aligned}$$

This is shown in Figure 10. In other words, for a linear system, a linear combination of the input signals results in a linear combination of the respective outputs.

For example, suppose the output signals of a linear system for two input signals $x_1[n]$ and $x_2[n]$ are $y_1[n] = \{1, 2, 3\}$ and $y_2[n] = \{0, 4, 5, 6\}$, respectively. Then for a new input signal

$$x[n] = 5x_1[n] - 3x_2[n]$$

the system output is

$$y[n] = 5y_1[n] - 3y_2[n] = \{5, -2, 0, -18\}$$

We can generalize this case to a combination of N input signals. If the input signal is given by:

$$x[n] = \sum_{k=1}^{N} a_k x_k[n]$$

then the output of a linear system is:

$$y[n] = \sum_{k=1}^{N} a_k y_k[n]$$

where $y_k[n]$ is the output signal for the the input signal $x_k[n]$. You can verify that the median filter described in Section 1.0.4 is not a linear system.

A time invariant system is one for which a time shift of the input signal causes a corresponding shift in the output signal. For example, if the output is $y[n]$ for an input $x[n]$, then the output for a delayed input $x_1[n] = x[n-m]$ (delayed by m samples) is $y_1[n] = y[n-m]$. As an example, we consider a system with the input-output relationship: $y[n] = nx[n]$. Is it time invariant? The response of the system to a delayed signal $x[n-k]$ is $y_1[n] = nx[n-k]$. However, a delayed output $y[n-k]$ is due to $(n-k)x[n-k]$. Therefore, the system is time variant since $y_1[n] \neq y[n-k]$. Following the same procedure, we can determine that the system $y[n] = x[-n]$ is also a time variant system.

A linear time-invariant (LTI) system satisfies both requirements. Let us take a look at another example. A system is defined by

$$y[n] = x[n] + x[n-1]$$

Is this an LTI system? To test if it is time-invariant, we use a new signal as the input $x_1[n] = x[n-m]$, where m is an constant integer. The output is

$$y_1[n] = x_1[n] + x_1[n-1] = x[n-m] + x[n-m-1]$$

On the other hand, we can easily see that

$$y[n-m] = x[n-m] + x[n-m-1]$$

Therefore, we have verified that $y_1[n] = y[n-m]$. The system is time-invariant.

To test if the system is linear, we use a combination of two input signals

$$x_3[n] = ax_1[n] + bx_2[n]$$

as the input, the output is

$$\begin{aligned} y_3[n] = &\quad x_3[n] + x_3[n-1] \\ = &\quad a(x_1[n] + x_1[n-1]) + b(x_2[n] + x_2[n-1]) \end{aligned}$$

We can easily see that

$$y_3[n] = ay_1[n] + by_2[n]$$

Therefore, the system is linear. So this is an LTI system.

Let us study another interesting example. We want to determine if the system given by: $y[n] = ax[n] + b$ is an LTI system. Here we assune both a and b are constants. From the definition of an LTI system, we first test if the system is shift invariant. Suppose we have an input signal $x_1[n] = x[n-k]$ where k is a constant and is an integer. Then we have $y_1[n] = ax_1[n] + b = ax[n-k] + b = y[n-k]$. Therefore the system is shift invariant. To test if the system is linear, we use two input signals $x_1[n]$ and $x_2[n]$ to make a new input signal $x_3[n] = \alpha x_1[n] + \beta x_2[n]$. We call the corresponding outputs $y_1[n]$, $y_2[n]$ and $y_3[n]$, respectively. If the system is indeed linear, then $y_3[n] = \alpha y_1[n] + \beta y_2[n]$. You can show that this condition is satisfied only when $b = 0$. Thus, the system is a linear system when $b = 0$. Otherwise, it is not linear system.

4.5 Convolution and unit sample response of an LTI system

4.5.1 Convulution

Let us consider the following simple problem. Suppose the system $y[n] = H\{x[n]\}$ is an LTI system and the output for an input $x_1[n] = \delta[n]$ is $y_1[n] = 2\delta[n] + 4\delta[n-1] = \{2, 4\}$. What is the output of the system for the given input $x[n] = \delta[n] + 2\delta[n-1] = \{1, 2\}$? We can write the input as

$$\begin{aligned} x[n] &= \delta[n] + 2\delta[n-1] \\ &= x_1[n] + 2x_1[n-1] \end{aligned}$$

Because the system is LTI, the output is given by

$$\begin{aligned} y[n] &= y_1[n] + 2y_1[n-1] \\ &= 2\delta[n] + 4\delta[n-1] + 2y_1[n-1] \\ &= 2\delta[n] + 4\delta[n-1] + 2(2\delta[n-1] + 4\delta[n-2]) \\ &= 2\delta[n] + 8\delta[n-1] + 8\delta[n-2]) \end{aligned}$$

We now study a general case for an LTI system. If the input signal is a unit sample sequence $\delta[n]$, the output is called the unit sample response $h[n]$ and is represented in the following general form

$$h[n] = H\{\delta[n]\}$$

What is the output for the input $x_m[n] = a_m\delta[n-m]$ where m is an integer and a_m is a scaling factor? Because the system is LTI, the output denoted $y_m[n]$ is given by

$$\begin{aligned} y_m[n] &= H\{a_m\delta[n-m]\} \\ &= a_m H\{\delta[n-m]\} \\ &= a_m h[n-m] \end{aligned}$$

Once we know the unit sample response $h[n]$, can we determine the output signal for an arbitrary input signal? In other words, can the unit sample response uniquely specify a system? For an LTI system the answer is YES.

To establish the input-output relationship, we use the signal representation

$$x[n] = \sum_{m=-\infty}^{\infty} x[m]\delta[n-m]$$

An interpretation for this representation is that an arbitrary signal $x[n]$ is now presented as a linear combination of many unit sample sequences with different delays. In terms of our previous example, we can write $x[n] = \sum_{m=-\infty}^{\infty} x_m[n]$ where $x_m[n] = a_m\delta[n-m]$ and $a_m = x[m]$. Here is a potentially confusing point. In the equation $a_m = x[m]$, the term $x[m]$ is used to represent the value of the signal $x[n]$ at a particular location index m. It is a number but is not referred to the whole signal! With this interpretation and the properties of an LTI system, we can determine the system output

$$\begin{aligned} y[n] = &\ H\{x[n]\} \\ = &\ H\left\{\sum_{m=-\infty}^{\infty} x[m]\delta[n-m]\right\} \\ = &\ \sum_{m=-\infty}^{\infty} x[m]H\{\delta[n-m]\} \\ = &\ \sum_{m=-\infty}^{\infty} x[m]h[n-m] \end{aligned}$$

Therefore, once $h[n]$ is known, the output can be calculated by the convolution operation

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

For example, if you want to calculate a particular output sample at index, say $n = 3$, then you use

$$y[3] = \sum_{m=-\infty}^{\infty} x[m]h[3-m]$$

This is a multiplication-and-sum operation, where corresponding samples of the two sequences, $x[m]$ and $h[3-m]$, are multiplied then summed together. To understand how to perform convolution, you need to understand the sequence denoted by $h[3-m]$. This sequence can be regarded as being generated by first folding $h[n]$ about $n = 0$ ($h[-m]$), then shifting it to the left by three samples.

The following two examples shows you how to perform the convolution operation. In the first example, we would like to find the input-output relationship of an LTI system given the unit sample response: $h[0] = 1$, $h[1] = 2$, and $h[2] = 1$. In this case, m can only take one of the three values: $m = n$, $m = n-1$ and $m = n-2$. Therefore the convolution can be performed as

$$y[n] = \sum_{m=n-2}^{n} x[m]h[n-m] = h[0]x[n] + h[1]x[n-1] + h[2]x[n-2]$$

In the second example, we would like to perform the convolution operation for the two sequences given by

$$x[n] = \delta[n] + 2\delta[n-1]$$

and

$$h[n] = \delta[n+1] - \delta[n-1]$$

unit sample reponse is: $h[-1] = 1$, $h[0] = 0$ and $h[1] = -1$, the index m can only take three values: $m = n-1$, $m = n$ and $m = n+1$. Therefore, the system output is given by

$$y[n] = \sum_{m=n-1}^{n+1} x[m]h[n-m] = h[1]x[n-1] + h[0]x[n] + h[-1]x[n+1] = x[n+1] - x[n-1]$$

Now we caluclate the output for this particular input $x[n] = \delta[n] + 2\delta[n-1]$ as follows

$$\begin{aligned} y[n] &= x[n+1] - x[n-1] \\ &= \delta[n+1] + 2\delta[n] - (\delta[n-1] + 2\delta[n-2]) \\ &= \delta[n+1] + 2\delta[n] - \delta[n-1] - 2\delta[n-2] \end{aligned}$$

Using the definition of $\delta[n]$, we can easily calculate the following

$y[-2] = \delta[-2+1] + 2\delta[-2] - \delta[-2-1] - 2\delta[-2-2] = 0$
$y[-1] = \delta[-1+1] + 2\delta[-1] - \delta[-1-1] - 2\delta[-1-2] = 1$
$y[0] = \delta[0+1] + 2\delta[0] - \delta[0-1] - 2\delta[0-2] = 2$
$y[1] = \delta[1+1] + 2\delta[1] - \delta[1-1] - 2\delta[1-2] = -1$
$y[2] = \delta[2+1] + 2\delta[2] - \delta[2-1] - 2\delta[2-2] = -2$
$y[3] = \delta[3+1] + 2\delta[3] - \delta[3-1] - 2\delta[3-2] = 0$
...

Of course, we can calculate the convolution operation to determine the output. The folding and shifting of $h[n]$ is shown in the following table. In this table, empty cells indicate zero-value for both sequences.

m	-2	-1	0	1	2	3
x[m]			1	2		
h[m]		1	0	-1		
h[-m] (n = 0), y[0] = 2		-1	0	1		
h[-1-m] (n = -1), y[-1] = 1	-1	0	1			
h[1-m] (n = 1), y[1] = -1			-1	0	1	
h[2-m] (n = 2), y[2] = -2				-1	0	1

In Matlab, you use the command **conv**(x,h) to perform the convolution operation.

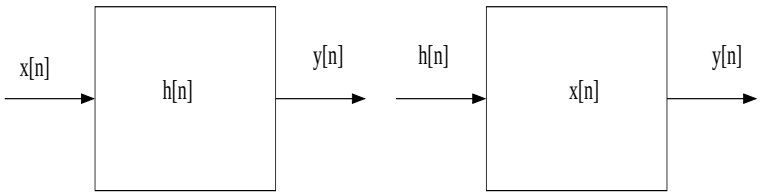


Figure 11: The two systems are equivalent.

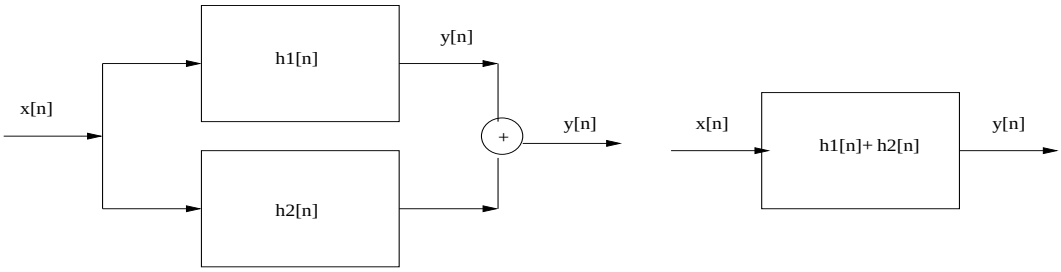


Figure 12: The two systems are equivalent.

4.5.2 Properties of the convultion operation

- The convolution operation is commutative:

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

  This means that you can regard $h[n]$ as the signal and $x[n]$ as the unit sample response. For a cascade system where the output of the first system serves as the input to the second system

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

  the order of processing can be interchanged

- The convolution operation is also distributive

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

  This means a parallel connection of two LTI systems is equivalent to another system of which unit sample response is the sum of the two respective systems.

4.5.3 Stability and causuality

Using the unit sample response, it is easy to test if an LTI system is stable and causal. A stable system must satisfy the condition:

$$\sum_{m=-\infty}^{\infty} |h[m]| < \infty$$

A causal system must satisfy the condition:

$$h[n] = 0, \quad for\ n < 0$$

This condition is easy to understand if we consider a simple system for which $h[-1] = 1$, $h[0] = 1$ and $h[1] = 1$. The input-output relationship is:

$$\begin{aligned} y[n] = & \sum_{m=-1}^{1} x[m]h[n-m] \\ = & x[n-1] + x[n] + x[n+1] \end{aligned}$$

So we see that this is a non-causal system because the current output sample depends on the "future" input sample $x[n+1]$.

4.5.4 Summary

In summary, the output of an LTI system is determined by the convolution of the input and the unit sample response. The convolution operation is both commutative and distributive. The stability and the causality of an LTI system can be determined from its unit sample response.

4.6 Linear constant-coefficient difference equations and LTI systems

An important subclass of LTI systems can be characterized by an Nth-order linear constant-coefficient difference equation:

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{m=0}^{M} b_k x[n-m]$$

For it to represent an LTI and causal system, it must satisfy the initial-rest condition, which states that:

if

$$x[n] = 0 \ for \ n < n_0$$

then

$$y[n] = 0 \ for \ n < n_0$$

We have used the difference equation representation for LTI systems in the above examples. Here is one more example:

$$y[n] = y[n-1] + \frac{1}{N}(x[n] - x[n-N])$$

This is a moving average filter in its recursive form (see self-test exercise in Section 1). Its equivalent implementation is

$$y[n] = \frac{1}{N}\sum_{m=1}^{N} x[n]$$

The difference equation representation is useful as it directly reveals the input-output relationship of a system.

4.7 Self-test Exercises

- A signal is given by $x[n] = \{1, 3, 5\}$. Define a new signal $y[n] = x[n-2]$. Write an equation that expresses $y[n]$ in terms of $\delta[n]$. Answer: $y[n] = \delta[n-2] + 3\delta[n-1] + 5\delta[n]$
- Plot a cosine signal to show that when the frequency gradually increases from π to 2π, the oscillation becomes slower.
- A system is given by $y[n] = x[n] + x[n-1]$. Is this a causal, stable and LTI system ? Answer: Yes
- The unit sample response is given by $h[0] = 1$, $h[1] = -2$, $h[2] = 1$. Determine the difference equation representation of the system. Answer: $y[n] = x[n] - 2x[n-1] + x[n-2]$
- An LTI system is given by: $h[0] = 1$, $h[1] = -2$, $h[2] = 1$. Calculate the system output for the input $x[n] = \{1, 3, 5, 7\}$. Answer:{1 1 0 0 -9 7}
- Assume $x[n] = \{1, 2, 3, 4, 5\}$, $h_1[n] = \{1, 1\}$ and $h_2[n] = \{1, -2, 1\}$. Calculate: (1) $x[n] * h_1[n]$, (2) $x[n] * h_1[n] * h_2[n]$, and (3) $h_1[n] * x[n] * h_2[n]$. Use Matlab to verify your results. Answer (1) {1 3 5 7 9 5} (2) {1 1 0 0 0 -6 -1 5 } (3) same as (2)

5 Signal and system representation in the frequency domain

5.1 Introduction

We will spend two lectures on the following topics

- the Fourier series of discrete time periodic signals
- the Fourier transform of discrete-time aperiodic signals
- properties of the Fourier transform
- the frequency domain representation of the LTI system

If you are not familiar with the frequency domain representation of signals and systems, you should read Chapter 3 of the text book.

A basic idea of the frequency domain representation is that a signal can be represented as a linear combination of cosine signals (or complex exponentials)

$$\begin{aligned} x(t) = & A_0 + \sum_{k=1}^{N} A_k \cos(2\pi f_k t + \phi_k) \\ = & X_0 + \sum_{k=1}^{N} \Re e\left\{X_k e^{j2\pi f_k t}\right\} \\ = & X_0 + \frac{1}{2}\sum_{k=1}^{N}\left\{X_k e^{j2\pi f_k t} + X_k^* e^{-j2\pi f_k t}\right\} \end{aligned}$$

where $X_k = A_k e^{j\phi_k}$ is a complex number that represents the amplitude and phase of the kth cosine signal with frequency f_k. X_k^* represents the complex conjugate of X_k. Now you see where the *negative* frequency comes from. Once we know the set of pairs $\{X_k, f_k\}$, we can generate the signal. With the frequency domain representation, we have another representation which is easier to handle than its original form $x(t)$ that could be very complicated.

5.2 The Fourier series of discrete time period signals

A periodic signal, $x[n] = x[n+N]$ where N is the period, can be represented by the discrete time Fourier series (DTFS):

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

the coefficient c_k is given by

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

Using the DTFS, a periodic signal is represented by a linear combination of the complex exponential signals $e^{j\omega_k n}$, where the frequency is

$$\omega_k = \frac{2\pi}{N}k$$

There are N frequency components (complex exponential signals), their frequencies range from

$$\omega_0 = 0$$

to

$$\omega_{N-1} = \frac{2\pi}{N}(N-1)$$

The quantity c_k represents the "amount" of a particular frequency component that the periodic signal contains. For example, a periodic signal with $N = 3$ is given by: $x[0] = 1$, $x[1] = 2$ and $x[2] = 2$. The DTFS representation is:

$$x[n] = \frac{5}{3} - \frac{1}{3}\left\{e^{j\frac{2\pi}{3}n} + e^{j\frac{4\pi}{3}n}\right\}$$

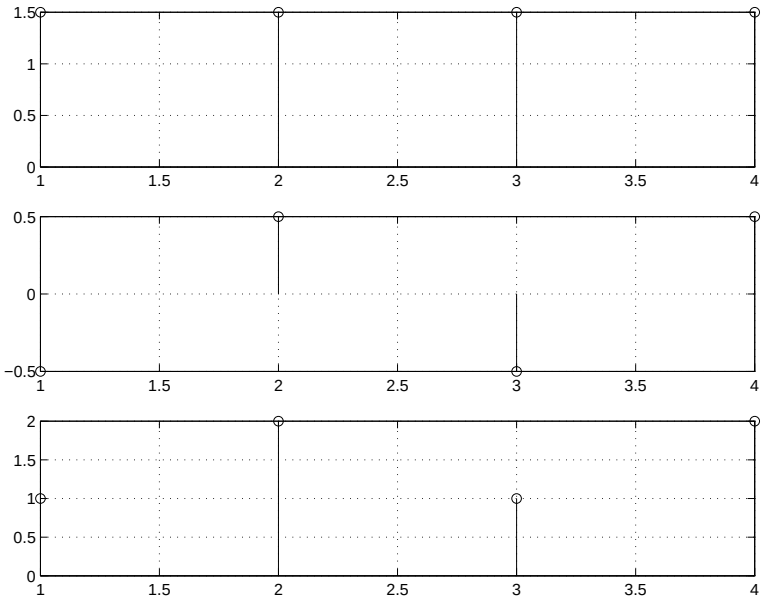


Figure 13: The Fourier representation of a periodic signal $x[n] = \{1, 2\}$ with a period of 2. Only two periods of the signal and its frequency components are shown in this figure. Top: the DC component. Middle: the high frequency component. Bottom: the original signal. We can see easily that the original signal can be represented as a summation of the two frequency components.

where[6] $c_0 = \frac{5}{3}$, $c_1 = c_2 = -\frac{1}{3}$. We can verify that:

$$x[0] = \frac{5}{3} - \frac{1}{3}\left\{e^{j\frac{2\pi}{3}0} + e^{j\frac{4\pi}{3}n}\right\} = \frac{5}{3} - \frac{2}{3} = 1$$

Therefore, the periodic signal can be regarded as being generated by a combination of three complex exponential signals whose frequencies are 0, $2\pi/3$ and $4\pi/3$, respectively. These complex exponential signals are usually referred to as the frequency components of the original signal.

Here is another example. Suppose a periodic signal is given by $x[n] = \{1, 2\}$. The Fourier series coefficients are $c_0 = \frac{3}{2}$ and $c_1 = -\frac{1}{2}$. The signal is represented by

$$x[n] = c_0 + c_1 e^{-jn\pi}$$

which is a combination of two signals: a DC signal represented by c_0 and a high frequency signal (with a frequency π) represented by $c_1 e^{-jn\pi}$. In other words, we can decompose the signal into two frequency components. This is shown in figure 13.

5.3 The Fourier transform of discrete-time aperiodic signals

In general, a signal is aperiodic. There are two solutions. If the signal is time-limited, we can make a periodic signal by repeating the signal. So we can use the DTFS. On the other hand, we can use the Fourier transform:

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

where

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

[6] $c_1 = \frac{1}{3}\left\{x[0] + x[1]e^{-j\frac{2}{3}\pi} + x[2]e^{-j\frac{4}{3}\pi}\right\} = -\frac{1}{3}$

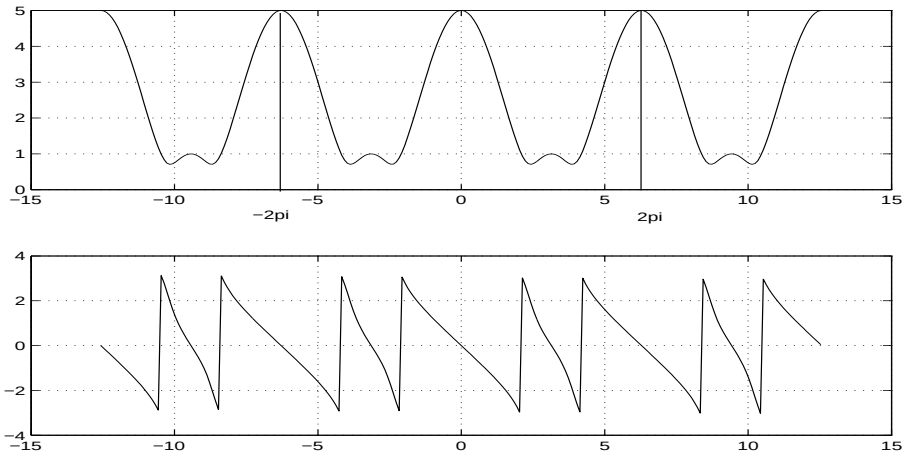


Figure 14: The amplitude (top) and phase (bottom) of the spectrum. Note that both are periodic functions of ω and the amplitude is an even function while the phase is an odd function.

For example, the Fourier transform of an aperiodic signal given by:

$$x[n] = \delta[n] + 2\delta[n-1] + 2\delta[n-2]$$

is

$$X(\omega) = \sum_{n=0}^{2} x[n]e^{-j\omega n} = 1 + 2\left\{e^{-j\omega} + e^{-j2\omega}\right\}$$

Tthe amplitude and phase[7] of $X(\omega)$ is shown in Figure 14. Note that a simple aperiodic signal is now transformed into a continuous complex function

$$X(\omega) = X_R(\omega) + jX_I(\omega)$$

where $X_R(\omega)$ and $X_I(\omega)$ are the real and imaginary part of $X(\omega)$. $X(\omega)$ is called the spectrum of the signal. It is a periodic function of the frequency variable ω with a fundamental period of 2π

$$X(\omega) = X(\omega + 2\pi) = X(\omega + 2k\pi)$$

The amplitude and the phase of the spectrum is given by:

$$|X(\omega)| = \sqrt{X_R^2(\omega) + X_I^2(\omega)}$$

and

$$\angle X(\omega) = \arctan\left(\frac{X_I(\omega)}{X_R(\omega)}\right)$$

We can easily see that the amplitude is an even function while the phase is an odd function.

5.4 Properties of the Fourier transform

The Fourier transform has some useful properties. In the following we use the notation to represent the Fourier transform pair

$$x[n] \longleftrightarrow X(\omega)$$

[7] In the text book, this is represented as $X(e^{j\omega})$

$ax[n]+by[n]$	$aX(\omega)+bY(\omega)$	linearity
$x[n-n_0]$	$e^{-j\omega n_0}X(\omega)$	delay
$e^{j\omega_0 n}x[n]$	$X(\omega-\omega_0)$	modulation
$x[-n]$	$X(-\omega)$	folding
$y[n]=x[n]*h[n]$	$Y(\omega)=X(\omega)H(\omega)$	convolution
$y[n]=x_1[n]x_2[n]$	$Y(\omega)=\frac{1}{2\pi}\int_{-\pi}^{\pi}X_1(\theta)X_2(\omega-\theta)d\theta$	multiplication
$\sum_{n=-\infty}^{\infty}x^2[n]$	$\frac{1}{2\pi}\int_{-\pi}^{\pi}\lvert X(\theta)\rvert^2 d\theta$	Parseval's theorem
$\sum_{n=-\infty}^{\infty}x[n]y[n]$	$\frac{1}{2\pi}\int_{-\pi}^{\pi}X(\theta)Y^*(\theta)d\theta$	

Here are two examples that illustrate how to use these properties. In the first example, we consider a pure delay system given by

$$y[n]=x[n-3]$$

The Fourier transform of the output signal is:

$$Y(\omega)=e^{-j3\omega}X(\omega)=|X(\omega)|e^{j(\angle X(\omega)-3\omega)}$$

Therefore, a time delay corresponds to a linear phase-shift. In the second example, we consider performing the convolution operation in the frequency domain. The convolution of two signals given by: $x_1\{1,\,1\}$ and $x_2\{1,\,-1\}$[8] is

$$y[n]=x_1[n]*x_2[n]=\delta[n]-\delta[n-2]$$

In the frequency domain, we have:

$$X_1(\omega)=1+e^{-j\omega}$$

and

$$X_2(\omega)=1-e^{-j\omega}$$

Therefore we have

$$Y(\omega)=X_1(\omega)X_2(\omega)=1-e^{-j2\omega}$$

Compared this result with the definition of the Fourier transform, we see that

$$y[n]=\delta[n]+\delta[n-2]$$

[8]This means $x_1[0]=1$, $x_1[1]=1$ and $x_2[0]=1$, $x_2[1]=-1$.

5.5 The frequency domain representation of the LTI system

The frequency domain

The convolution property of the Fourier transform suggests another way to represent the LTI system. In the frequency domain, where the signal and the unit sample response are represented by their respective Fourier transform $X(\omega)$ and $H(\omega)$, the Fourier transform of the output $Y(\omega)$ is given by:

$$Y(\omega) = X(\omega)H(\omega)$$

Since $H(\omega)$ is the Fourier transform of $h[n]$, it is a complex function whose amplitude is an even function and its phase is an odd function. It is also a periodic function with a fundamental period of 2π. Therefore, we only need to consider the frequency in range of $[0, \pi]$. The quantity $H(\omega)$ is called the frequency response of the LTI system. We know that, as in the time domain case, once we know the frequency response, we can determine the output of the system for *any* input signal.

To see the implication of the frequency domain representation, let us consider the input-output relationship of an LTI system, with an complex exponential input signal:

$$x[n] = Ae^{j\omega_0 n}$$

The output (in the time domain) is given by

$$\begin{aligned} y[n] = & \sum x[m]h[n-m] \\ = & \sum h[m]x[n-m] \\ = & A\sum h[m]e^{j\omega_0(n-m)} \\ = & Ae^{j\omega_0 n}\sum h[m]e^{-j\omega_0 m} \\ = & Ae^{j\omega_0 n}H(\omega)|_{\omega=\omega_0} \\ = & Ae^{j\omega_0 n}H(\omega_0) \end{aligned}$$

where the notation $H(\omega)|_{\omega=\omega_0}$ represents the frequency response evaluated at a particular frequency. The above calculation shows that the output signal is modified in two ways: the amplitude and the phase. However, the frequency remains the same. The amplitude of the output signal is $A|H(\omega_0)|$, and the phase is (assume A is a real number) $\angle H(\omega_0)$. Therefore, the output signal is:

$$y[n] = A|H(\omega_0)|e^{j(\omega_0 + \angle H(\omega_0))}$$

Here is an example. Suppose the impulse response of an LTI system is $h[n] = \{1, 1\}$. The frequency response is $H(\omega) = 1 + e^{-j\omega}$. The amplitude response is $|H(\omega)| = 2|cos(\omega/2)|$ and the phase response is $\angle H(\omega) = -\omega/2$. As such, we can write $H(\omega) = |H(\omega)|e^{j\angle H(\omega)}$. Let us first consider the output for the input signal $x[n] = cos(\omega_0 n)$. To use our previous results, we can write $x[n] = (e^{j\omega_0 n} + e^{-j\omega_0 n})/2 = x_1[n] + x_2[n]$, where $x_1[n] = \frac{1}{2}e^{j\omega_0 n}$ and $x_2[n] = \frac{1}{2}e^{-j\omega_0 n}$. Therefore the output of the system is

$$\begin{aligned} y[n] &= y_1[n] + y_2[n] \\ &= \frac{1}{2}|H(\omega_0)|e^{j(\omega_0 n + \angle H(\omega_0))} + A|H(-\omega_0)|e^{j(\omega_0 n + \angle H(-\omega_0))} \\ &= \frac{1}{2}|H(\omega_0)|e^{j(\omega_0 n + \angle H(\omega_0))} + A|H(\omega_0)|e^{-j(\omega_0 n + \angle H(\omega_0))} \\ &= \frac{1}{2}|H(\omega_0)|(e^{j(\omega_0 n + \angle H(\omega_0))} + e^{-j(\omega_0 n + \angle H(\omega_0))}) \\ &= |H(\omega_0)|cos(\omega_0 n + \angle H(\omega_0)) \end{aligned}$$

In this example, we have shown that this simple LTI system modifies the input signal in two ways. It changes the amplitude of the signal and adds a phase angle to the signal. The amount of changes

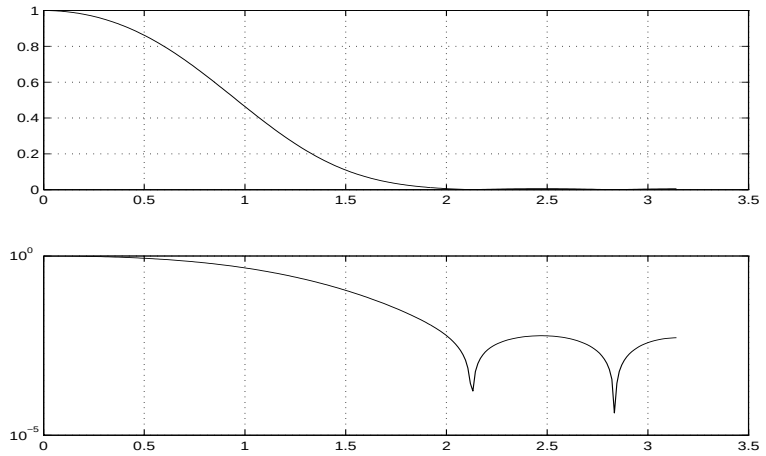


Figure 15: Top: the frequency response (amplitude response) of a low pass filter. Bottom: the same plot in log-scale

is related to the frequency of the signal and frequency response of the system. For example, when $\omega_0 = \frac{\pi}{4}$, we have

$$y[n] = 2\cos(\pi/8)\cos(\pi n/4 - \pi/8).$$

Here is a question for you, what is the output signal if the input signal is $x[n] = \cos(\pi n)$? Answer: $y[n] = 0$. You can verify your result by using convolution: $y[n] = x[n] * h[n] = cos(\pi n) + cos(\pi(n-1)) = 0$. Here is another question for you, what is the output signal is the input signal is given by $x[n] = \cos(\pi n/3) + 0.5\cos(\pi n/4)$.

What is a filter ?

The word *filter* is a key word in digital signal processing. We usually call an LTI system a filter. This is because

- As a result of the Fourier transform, a discrete time signal can be regarded as a linear combination of discrete exponential signals (or cosine signals or frequency components) with different amplitude, frequency and phase.
- An LTI system (a filter) only changes the amplitude and phase of a discrete exponential signal
- By properly designing the frequency response of the LTI system, the amplitude and phase of each frequency component of the input signal are modified. Thus, the input signal is filtered.

For example, to attenuate the high frequency components of a signal, we would use a *low pass filter* which has a unit gain for low frequency components and a smaller gain for high frequency components. In other words, the frequency response of the filter should be similar to Figure 15. We note that the horizontal axis represents the frequency and the vertical axis represents the gain. So this figure shows the system's gain versus the frequency. For example, the gain for a input frequency component whose frequency is $\omega_0 = 0.5\pi$ is 0.1. This means that the amplitude of the respective output signal will be scaled down by a factor of 0.1. In this figure, we only plot the frequency response (or precisely, the amplitude response) in the frequency range $[0, \pi]$. Figure 16 shows a combination of two cosine signals:

$$x[n] = \cos(0.1\pi n) + \cos(0.5\pi n)$$

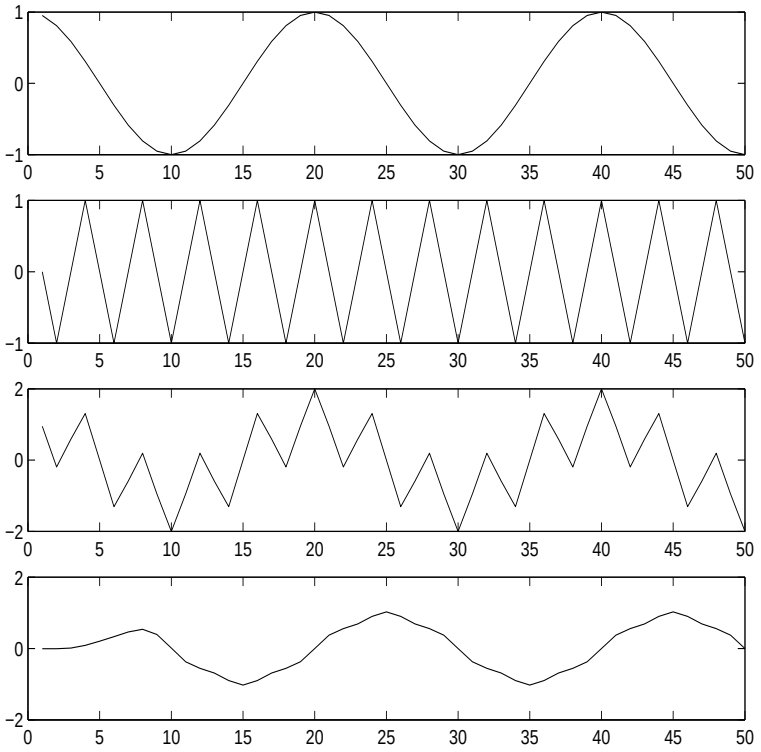


Figure 16: An illustration of a low pass filter. From top to bottom: $cos(0.1\pi n)$, $cos(0.5\pi n)$, $cos(0.1\pi n) + cos(0.5\pi n)$ and the filtering result.

and the result of the previous low pass filter. We can see that the amplitude of the high frequency component becomes about 0.1 after the low pass filter and the amplitude of the low frequency component is almost unchanged. Therefore, the filtering result looks almost the same as the low frequency component In the same way, we can now define a high pass filter, a band pass filter and a band stop filter.

The frequency response characterizes a filter's function. For example, the first-difference system is given by:

$$y[n] = x[n] - x[n-1]$$

To determine its frequency response, we take the Fourier transform on both side of the equation:

$$Y(\omega) = X(\omega) - e^{-j\omega}X(\omega)$$

Therefore we have

$$H(\omega) = \frac{Y(\omega)}{X(\omega)} = 1 - e^{-j\omega}$$

and

$$|H(\omega)| = 2|sin(\omega/2)|$$

Figure 17 shows its magnitude response. We can see that this is a high pass filter. The DC component of a signal will be "nulled" by the filter.

As another example, we consider the frequency response of a moving average filter given by

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

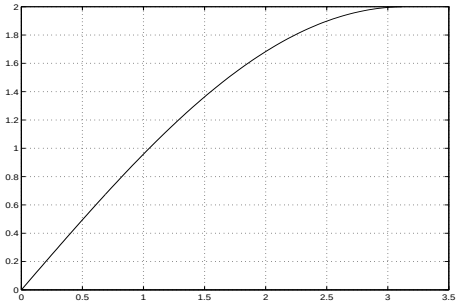


Figure 17: The magnitude response of the first difference system

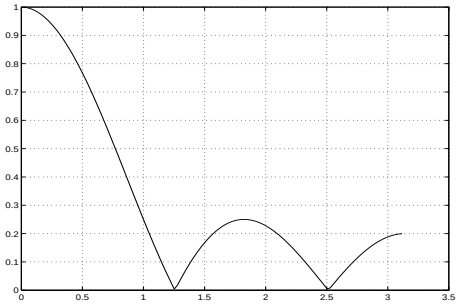


Figure 18: The magnitude response of a moving average filter (N=5)

Its frequency response is given by

$$H(\omega) = \frac{1}{N}\sum_{k=0}^{N-1} e^{-j\omega k} = \left(\frac{sin(\omega N/2)}{Nsin(\omega/2)}\right) e^{-j\omega(N-1)/2}$$

The magnitude response is

$$|H(\omega)| = \frac{1}{N}\sum_{k=0}^{N-1} e^{-j\omega k} = \left|\frac{sin(\omega N/2)}{Nsin(\omega/2)}\right|$$

This is shown in Figure 18. Next, we consider the frequency response of an LTI system (a Butterworth high pass filter) given by

$$y[n] = 0.1311x[n] - 0.2622x[n-1] + 0.1311x[n-2] - 0.7478y[n-1] - 0.2722y[n-2]$$

Its frequency response is

$$H(\omega) = \frac{0.1311 - 0.2622e^{-j\omega} + 0.1311e^{-j2\omega}}{1 + 0.7478e^{-j\omega} + 0.2722e^{-j2\omega}}$$

The magnitude response is shown in Figure 19

5.6 Self-test exercises

- Determine the Fourier transform of a signal $x[n] = \{1, 2, 1\}$. Plot its amplitude and phase response. Verify that the amplitude response is an even function. Why do we only need to consider the frequency response in the frequency range $[0, \pi]$? Answer: $X(\omega] = 1 + e^{-j\omega} + e^{-j2\omega} = e^{-j\omega}(e^{j\omega} + e^{-j\omega} + 2) = e^{-j\omega}(2\cos\omega + 2)$

- Another signal is defined as $y[n] = x[n-2]$. Determine its time domain and frequency domain representations in terms of those of $x[n]$. Answer: $y[n] = \{0, 0, 1, 2, 1\}$. $Y(\omega) = e^{-j2\omega}X(\omega)$

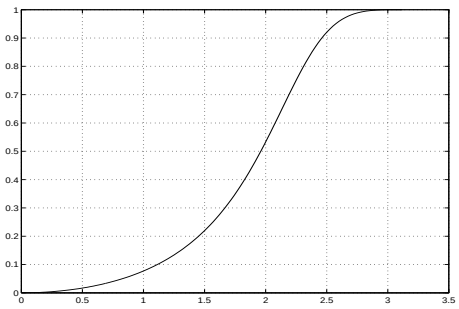


Figure 19: The magnitude response of a Butterworth high pass filter.

- An LTI system is given by: $y[n] = x[n-1] - 2x[n-2] + x[n-3]$. Find its unit sample response, frequency response. Is this a high pass filter? Answer: $h[n] = \delta[n-1] - 2\delta[n-2] + \delta[n-3]$. $H(\omega) = e^{-j\omega}(1 - 2e^{-j\omega} + e^{-j2\omega})$. It is a high pass filter.

- Use the Fourier transform to perform the convolution $x[n] * h[n]$, where $x[n] = \{1, 2, 1\}$ and $h[n] = \{1, -1\}$. Answer: $X(\omega) = 1 + 2e^{-j\omega} + e^{-j2\omega}$, $H(\omega) = 1 - e^{-j\omega}$

- Go through the demos in Chapter 6 of the text book.

6 Problem class-1

We will summarize what we have done so far and briefly discuss problems in Assignment 1

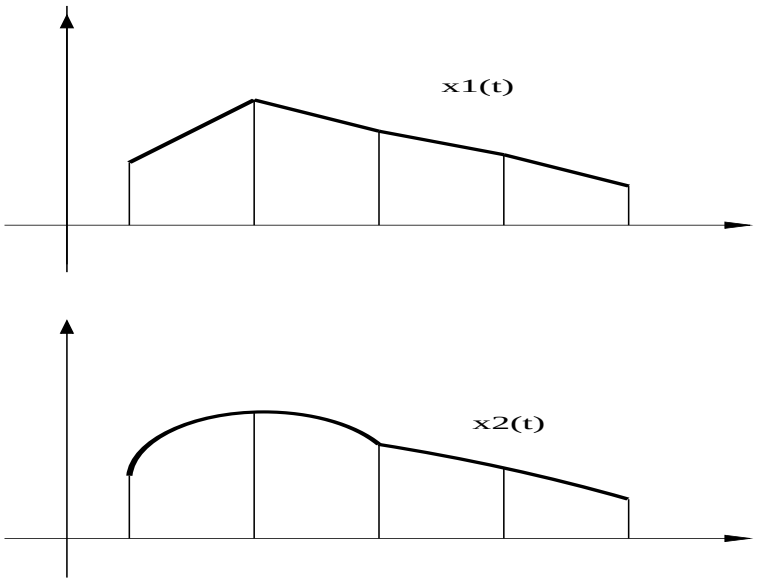


Figure 20: The same discrete signal could be a result of sampling two different analog signals.

7 Sampling and aliasing

This lecture is based on Chapter 4 of the text book.

7.1 Time domain representation

Sampling

Sampling is an essential step in the discrete time processing of analog signals. It is represented by the following equation

$$x[n] = x[nT_s] = x(t)|_{t=nT_s}$$

where $x[n]$ is a short-hand notation for $x[nT_s]$ and $x(t)$ is the analog signal. It is easy to see that given the analog signal and the sampling period T_s, we can obtain a unique discrete signal.

However, the discrete signal does not carry explicit information about the sampling period. The index n only represents the order of the samples, for example, the next sample of $x[n-1]$ is $x[n]$. A more important problem is that even the sampling period is specified, a discrete signal could correspond to *many* different analog signals. This is shown in Figure 20 .

Therefore, we need to study the condition that a discrete signal uniquely represents the original analog signal. Let us study the Fourier series representation of an analog signal:

$$x(t) = A_0 + \sum_{k=1}^{N} A_k \cos(\Omega_k t + \phi_k)$$

where $\Omega_k = 2\pi f_k$. When we sample this signal with a sampling period T_s, the resulting discrete signal is

$$\begin{aligned} x[n] = &\quad A_0 + \textstyle\sum_{k=1}^{N} A_k \cos(\Omega_k n T_s + \phi_k) \\ = &\quad A_0 + \textstyle\sum_{k=1}^{N} A_k \cos(\omega_k n + \phi_k) \end{aligned}$$

where $\omega_k = \Omega_k T_s$. Here we have an important relationship between the frequency of an analog cosine signal and that of a discrete cosine signal:

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

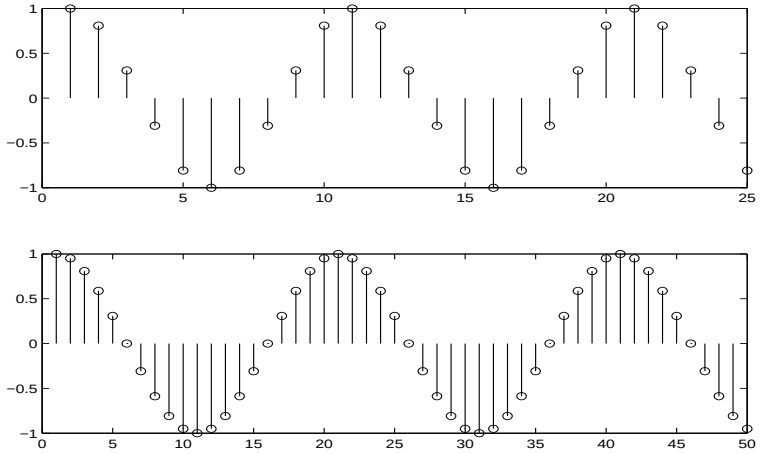


Figure 21: A 200 Hz cosine signal is sampled at 2000 Hz and 4000 Hz respectively. The corresponding frequencies are $\pi/5$ (top) and $\pi/10$ (bottom).

The quantity ω is called the discrete time radian frequency (we will call it discrete frequency or just frequency later on) and it is dimensionless. For example, an analog cosine signal with a frequency of 4 kHz will be converted to a discrete signal with the discrete time radian frequency of $\pi/4$ when it is sampled at $f_s = 32$ kHz. If we increase the sampling frequency to $f_s = 128$ kHz, the resulting discrete time radian frequency will be $\pi/16$. We note that a higher sampling frequency results in a lower discrete time radian frequency and that there are more samples in one period of the signal. A 200 Hz cosine signal is sampled at 2000 Hz and 4000 Hz respectively. The corresponding frequencies are $\pi/5$ and $\pi/10$, respectively. These two discrete signals are shown in Figure 21. If the discrete cosine signal is a periodic signal, then its fundamental period is

$$N = \frac{2\pi}{\omega}$$

For example, the fundamental periods of these two signal are 10 and 20.

From the above discussion, it is evident that we only have to investigate the condition that a sampled cosine signal uniquely represent the original analog cosine signal. Next we take a look at the relationship among the three quantities: the discrete frequency ω_0, the frequency of the signal f_0, and the sampling frequency f_s

Aliasing and folding

We already know that

$$\omega_0 = \frac{2\pi f_0}{f_s}$$

In addition ω_0 is not distinguishable from $\omega_0 + 2k\pi$ when k is an integer. This is because $\cos[(\omega_0 + 2k\pi)n] = \cos(\omega_0 n + 2kn\pi) = \cos(\omega_0 n)$. In the following discussion, we assume the sampling frequency is fixed. Basically, we can regard the sampling process for an analog cosine signal as a mapping from f_0 to ω_0. Because the discrete cosine signal is a periodic function of the variable ω with a fundamental period of 2π, we only have to discuss three cases:

- $\omega_0 \leq \pi$
- $\pi < \omega_0 \leq 2\pi$
- $2\pi < \omega_0 \leq 3\pi$

In the first case, we have $f_0 \leq \frac{f_s}{2}$ and ω_0 is a linear function of f_0. This is a one to one mapping from f_0 to ω_0. In other words, there is no other analog cosine signal (with a different frequency in the range of $[0, f_s/2]$) that would lead to this discrete signal. Therefore, the discrete cosine signal uniquely represents the original signal.

In the second case, we have $\frac{f_s}{2} < f_0 \leq f_s$. Let us define a new signal with a frequency f_1 such that

$$f_0 = f_s - f_1$$

and

$$f_1 \leq \frac{f_s}{2}$$

We can easily see that

$$\omega_0 = \frac{2\pi f_0}{f_s} = \frac{2\pi(f_s - f_1)}{f_s} = 2\pi - \frac{2\pi f_1}{f_s} = 2\pi - \omega_1$$

In this case, the cosine signal $\cos(\omega_0 n)$ is indistinguishable from the other cosine signal $\cos(\omega_1 n)$. Why? Because $\cos[(2\pi - \omega_0)n] = \cos(2n\pi - \omega_0 n) = \cos(\omega_0 n)$. Therefore, we do not know if the discrete cosine signal is due to a signal, given by

$$x_0(t) = \cos(2\pi f_0 t - \phi)$$

or is due to another one, given by

$$x_1(t) = \cos(2\pi f_1 t + \phi)$$

Because when sampled using the same frequency f_s, we have

$$x_0[n] = \cos(\omega_0 n - \phi)$$

and

$$x_1[n] = \cos(\omega_1 n + \phi)$$

You can easily see that $x_0[n] = x_1[n]$. This is called folding, where the two frequencies f_0 and f_1 are mirror images with respect to $f_s/2$. Actually, ω_0 is linearly decreased from π to 0 when f_0 is increased from $f_s/2$ to f_s.

In the third case, $f_s < f_0 \leq \frac{3}{2} f_s$. As in the second case, let us define a new signal with a frequency f_2 such that

$$f_0 = f_s + f_2$$

and

$$f_2 \leq \frac{f_s}{2}$$

Then, we have

$$\omega_0 = \frac{2\pi f_0}{f_s} = \frac{2\pi(f_s + f_2)}{f_s} = 2\pi + \frac{2\pi f_2}{f_s} = 2\pi + \omega_2$$

Again, we see that the signal $\cos(\omega_0 n)$ is indistinguishable from another signal $\cos(\omega_2 n)$. Therefore, in this third case we do not know if the discrete cosine signal is due to a signal, denoted by

$$x_0(t) = \cos(2\pi f_0 t + \phi)$$

or is due to another one, denoted by

$$x_2(t) = \cos(2\pi f_2 t + \phi)$$

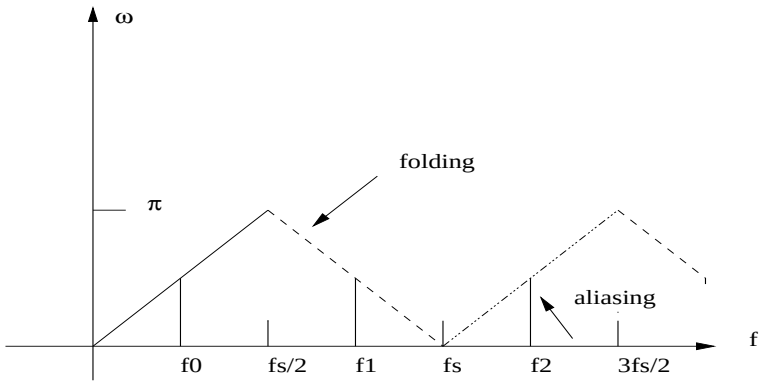


Figure 22: The aliasing and folding as a result of sampling the cosine signal. Three cosine signals with frequencies of f_0, f_1 and f_2 lead to the same discrete-time cosine signal with a frequency of $\pi/2$. f1 is called the folded frequency, because $f_1 = f_s - f_0$. f_2 is called the aliased frequency, because $f_2 = f_s + f_0$.

Because when sampled using the same frequency, we have

$$x_0[n] = \cos(\omega_0 n + \phi)$$

and

$$x_2[n] = \cos(\omega_2 n + \phi)$$

You can easily see that $x_0[n] = x_2[n]$. The frequency f_0 is called the aliased frequency of f_2. ω_0 is linearly increased from 0 to π when f_0 is increased from f_s to $\frac{3}{2}f_s$.

Figure 22 illustrates aliasing and folding as a result of sampling the cosine signal. In this figure, we can see that there are other aliasing and folding frequencies located at:

$$f_k^{aliasing} = kf_s + f_0$$

and

$$f_k^{folding} = kf_s - f_0$$

where k is an integer.

Let us take a look at one example. Suppose a signal is given by the formula

$$x[n] = \cos(0.2\pi n - \pi/7)$$

and that it was obtained by sampling a continuous-time signal at a sampling rate of $f_s = 1000$ samples/sec. We will determine two different continuous-time signals $x_1(t)$ and $x_2(t)$ such that $x[n] = x_1[nT] = x_2[nT]$. Both signals should have a frequency less than 1000 Hz.

From the above discussions, we know that the frequency of $x_1(t)$ should be in the non-aliasing region and the frequency of $x_2(t)$ should be is in the folding region. The frequency for the first signal is given by

$$\frac{2\pi f_1}{f_s} = \omega = 0.2\pi$$

We have $f_1 = 100$ Hz. The frequency for the second signal is the folded frequency of f_1

$$f_2 = f_s - f_1 = 900Hz$$

Therefore, the signals are given by

$$x_1(t) = \cos(2\pi f_1 t - \pi/7)$$

and

$$x_2(t) = \cos(2\pi f_2 t + \pi/7)$$

To verify these results, we sample these two signals with a sampling frequency of 1000 Hz. We obtain

$$x_1[nT] = \cos(2\pi f_1 nT - \pi/7) = \cos(0.2\pi n - \pi/7)$$

and

$$\begin{aligned} x_2[nT] = &\cos(2\pi f_2 nT + \pi/7) \\ = &\cos(2\pi \tfrac{1000-100}{1000} n + \pi/7) \\ = &\cos(2\pi n - (0.2\pi n - \pi/7)) \\ = &\cos(0.2\pi n - \pi/7) \end{aligned}$$

The aliasing and folding frequency can be easily demonstrated by using Matlab. For a 200 Hz cosine signal and a sampling frequency of 1000 Hz, the folding frequency is

$$kf_s - f_0 = 1000k - 200$$

and the aliasing frequency is

$$kf_s + f_0 = 1000k + 200$$

where k is a positive integer. When $k = 1$, these two frequencies are 800 Hz and 1200 Hz respectively. The following is a segment of Matlab code that is used to plot the three discrete cosine signal

```
i = 0 : 1/1000 : 1;          %fs = 1000 Hz, take samples between 0 - 1 second
f1 = 200 ;
s1 = cos ( 2 * pi * f1 * i ) ;
f2 = 800 ;
s2 = cos (2 * pi *f2 * i );
f3 = 1200 ;
s3 = cos (2 *pi *f3 * i );
subplot ( 311 ); stem ( s1 ( 1 : 50 ))
subplot ( 312 ); stem ( s2 ( 1 : 50 ))
subplot ( 313 ); stem ( s3 ( 1 : 50 ))
```

The results are shown in fig. 23. Following this example, let us consider this problem: what would happen if sample the following analog signal with the sampling frequency $f_s = 1000Hz$?

$$x(t) = \cos(2\pi f_1 t) + \cos(2\pi f_2 t) + \cos(2\pi f_3 t)$$

The resulting discrete time signal, as you can see from figure 23, should be

$$x[n] = 3\cos(\omega_1 n)$$

Where have the two other frequency components gone? Because they are the folding (f_2) and aliasing (f_3) frequency of f_1, in the discrete domain they are identical.

To summarize we have demonstrated the following key points for a fixed sampling frequency f_s and a cosine signal $x(t) = \cos(2\pi f t)$.

- When $f \leq \frac{f_s}{2}$, the resulting discrete signal $x[n] = \cos(\omega n)$ ($\omega = \frac{2\pi f}{f_s}$) is a unique representation of $x(t)$.
- When $\frac{f_s}{2} < f \leq f_s$, we can write $f = f_s - f_0$ (for $f_0 \leq \frac{f_s}{2}$) the resulting discrete signal $x[n] = \cos(\omega n) = \cos(\omega_0 n)$ ($\omega_0 = \frac{2\pi f_0}{f_s}$). f_0 is the folding frequency. In other words, when the frequency f is in this range, the resulting discrete signal will be the same as that due to sampling the signal $x_0(t) = \cos(2\pi f_0 t)$.

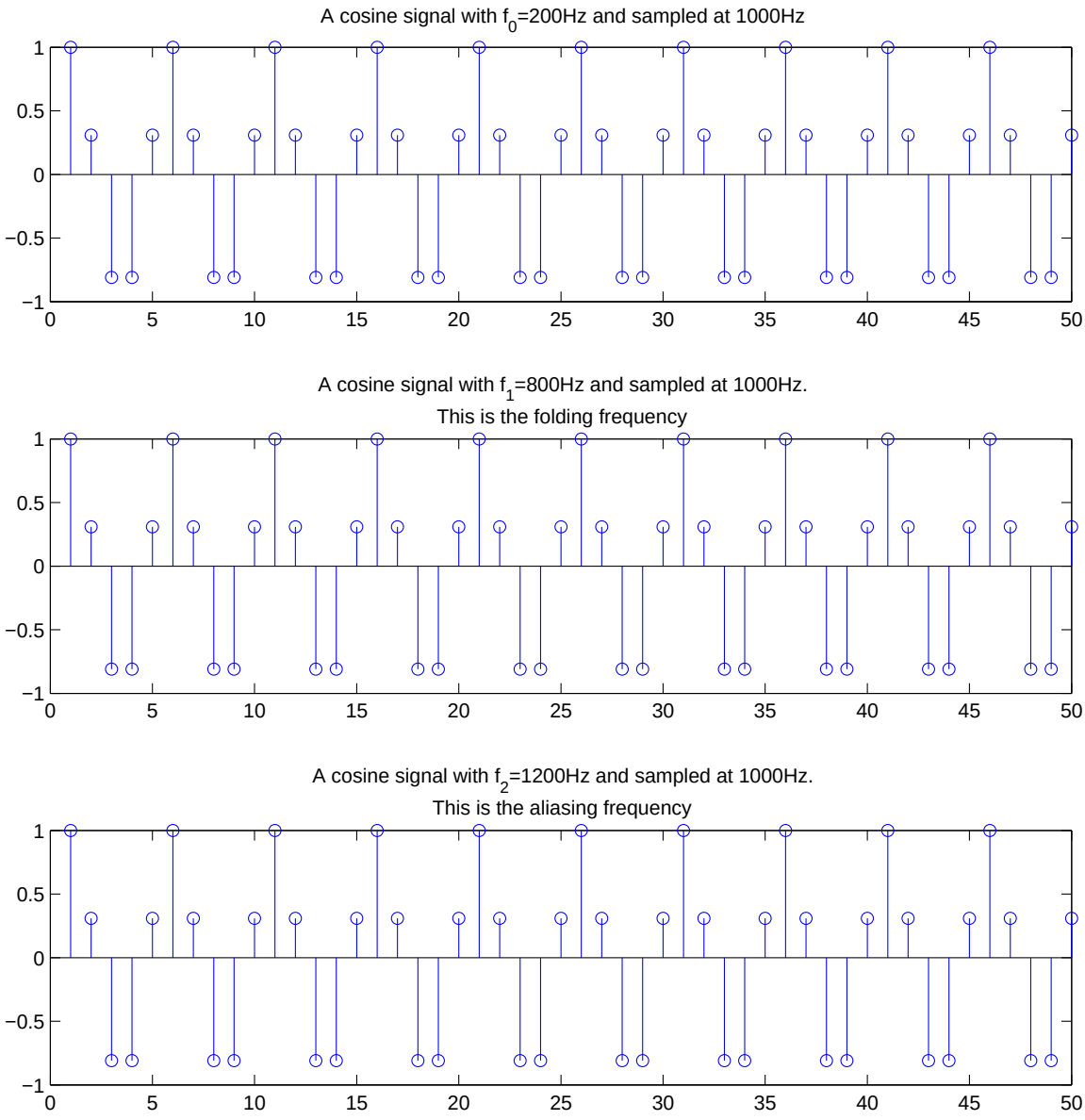


Figure 23: The three figures are results of sampling three cosine signals with frequencies 200Hz, 800 Hz and 1200Hz, respectly. The sampleing frequency is 1000Hz. We can see that the resulting discrere signals are exact the same.

- When $f_s < f \leq \frac{2}{3}f_s$, we can write $f = f_s + f_0$ (for $f_0 \leq \frac{f_s}{2}$) the resulting discrete signal $x[n] = \cos(\omega n) = \cos(\omega_0 n)$ ($\omega_0 = \frac{2\pi f_0}{f_s}$). f_0 is the aliasing frequency. In other words, when the frequency f is in this range, the resulting discrete signal will be the same as that due to sampling the signal $x_0(t) = \cos(2\pi f_0 t)$.

Here is an example to illustrate these points. Suppose we have a signal

$$x(t) = \cos(2\pi f_1 t) + \cos(2\pi f_2 t) + \cos(2\pi f_3 t)$$

with $f_1 = 200Hz$, $f_2 = 700Hz$ and $f_3 = 1400Hz$. Suppose we sample this signal with a sampling frequency $f_s = 1000Hz$. According to the above three points, we have three frequency components. The fisrt is with $\omega_1 = 2\pi f_1/f_s = 2\pi/5$. The second is with the folding frequency for f_2 is 300 Hz (700=1000-300) and $\omega_2 = 2\pi \times 300/1000 = 3\pi/5$ and the third is with the aliasing frequency for f_3 is 400Hz (1400=1000+400) and $\omega_3 = 2\pi \times 400/1000 = 4\pi/5$. So the discrete time signal we have is

$$x[n] = \cos(2\pi n/5) + \cos(3\pi n/5) + \cos(4\pi n/5)$$

If we are given this signal and the sampling frequency f_s, then we may say that the corresponding analog signal has three frequency components with $f_1 = \omega_1 f_s/(2\pi) = 200Hz$, $f_2 = \omega_2 f_s/(2\pi) = 300Hz$ and $f_3 = \omega_3 f_s/(2\pi) = 400Hz$. Note that f_2 is the folding and f_3 is the aliasing frequency.

The sampling theorem

We have seen that if the sampling frequency is properly chosen such that the resulting discrete frequency satisfies: $\omega_0 \leq \pi$, then the discrete cosine signal uniquely represents the original signal. This requires that $f_s \geq 2f_0$. For an arbitrary signal $x(t)$, if we choose a sampling frequency high enough such that neither aliasing nor folding happens to the highest frequency component of the signal, then there will be no aliasing or folding for other frequency components. The resulting discrete signal uniquely represents the original signal. Denoting the highest frequency of an analog signal as f_{max}, the sampling theorem states that:

$$f_s \geq 2f_{max}$$

For example, the bandwidth of human voice is usually regarded as about 4 kHz. The sampling frequency must be at least 8 kHz.

7.2 Frequency domain representation

A detailed mathematical derivation of the frequency domain representation of the sampling operation is beyond the scope of this lecture. Interested students should refer to one of the reference books for more information. Here, we only give a graphical explanation. Our starting point is that the sampling operation maps f to ω. This is shown in Figure 24.

Figure 25 shows the spectrum of the original analog signal and three versions of the corresponding discrete signals sampled at different frequencies. It is easy to understand that the spectrum of the discrete signal is related to the spectrum of the original analog signal in that there are multiple copies of the spectrum, each centered[9] at $2k\pi f_s$, where k is an integer. We make two observations from this figure. Firstly, we observe that if the sampling frequency is less than half of the highest frequency, then there will be overlap of the copies of the spectrum. The smallest sampling frequency that we can use to avoid the spectrum overlapping is $2f_{max}$. This is required by the sampling theorem. Secondly, we observe that we can reconstruct the original signal by applying a low pass filter to the discrete signal to retain the center copy and eliminate other copies of the spectrum. In an extreme case where

[9] If you want to convert this to the discrete domain, then each spectrum copy is centered at $2k\pi (= 2k\pi f_s/f_s)$.

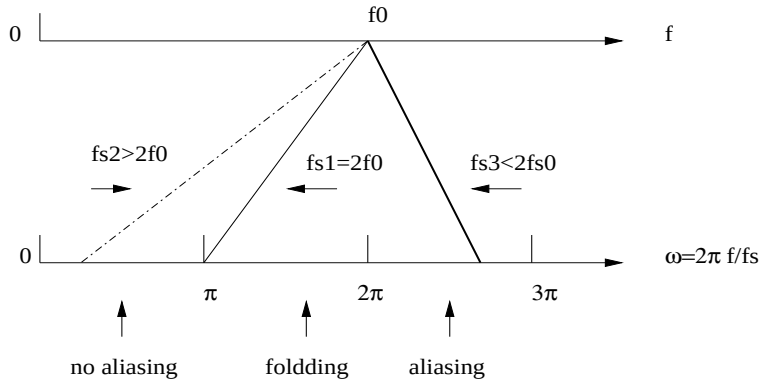


Figure 24: The mapping from f to ω. When $f_s \geq 2f_0$, there is no aliasing. The frequency range $[0, f_0]$ is mapped to $[0, \pi]$.

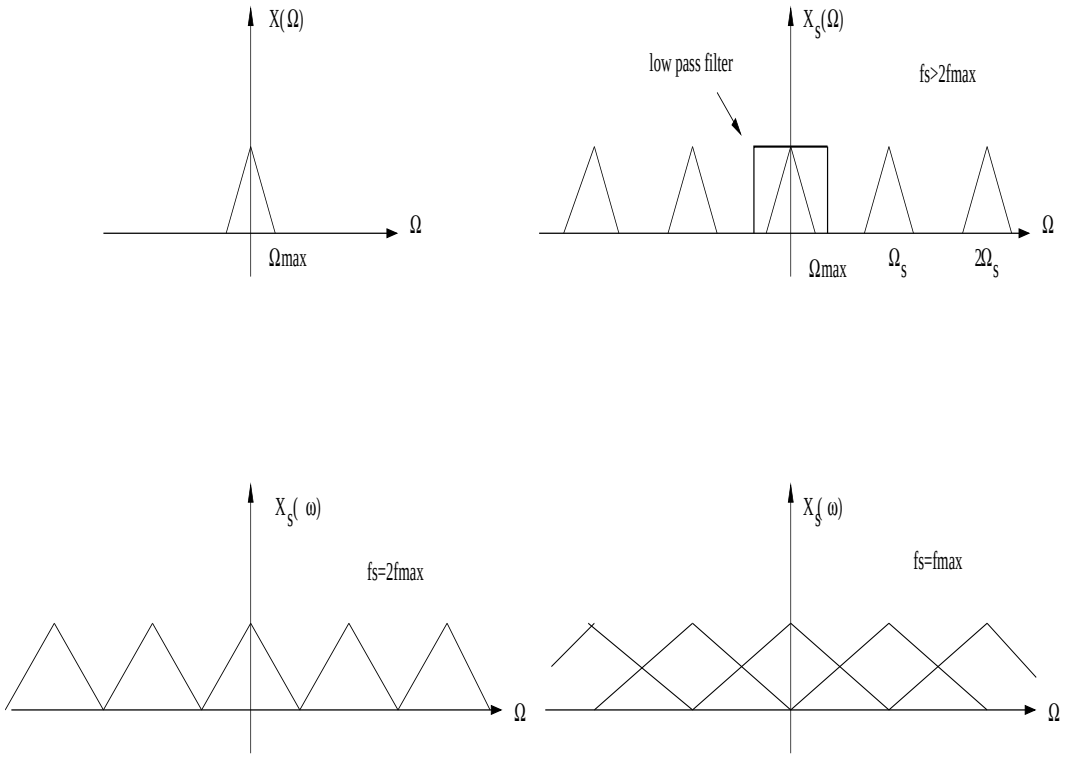


Figure 25: The frequency domain representation of the sampling operation. Note that these figures show the spectrum in the continuous case, where $\Omega_s = 2\pi f_s$. Top left: spectrum of the analog signal with the highest frequency denoted f_{max} . Top right: sample signal spectrum, $f_s > 2f_{max}$. Bottom left: $fs = 2f_{max}$. Bottom right: $f_s = f_{max}$

$\Omega_s = 2\Omega_{max}$, the cut-off frequency is $\Omega_c = \Omega_s/2$. This corresponds to $2\pi f_s/2 = \pi/T_s$. In other cases where $\Omega_s > 2\Omega_{max}$, the cut-off frequency can be selected from the frequency range

$$\Omega_{max} < \Omega_c < (\Omega_s - \Omega_{max})$$

7.3 Discrete-to-continuous conversion

If we take the inverse Fourier transform of the frequency response of an ideal low pass filter where the cut-off frequency is assumed π/T_s, then we have the impulse response of the filter:

$$h(t) = \frac{\sin(\pi t/T_s)}{\pi t/T_s}$$

Therefore, the reconstructed signal is

$$x(t) = \sum_{n=-\infty}^{\infty} x[n] \frac{\sin[\pi(t-nT_s)/T_s]}{\pi(t-nT_s)/T_s}$$

From this equation, we can easily verify that

$$x[n] = x(t)|_{t=nT_s}$$

A general formula for converting a discrete signal into an analog signal is given by:

$$x(t) = \sum_{n=-\infty}^{\infty} x[n]p(t-nT_s)$$

where the function p is the interpolation function which fills the gaps between samples. In the previous case, the interpolation function is a sinc-function. Other interpolation functions, such as a square pulse, a triangular pulse and a linear function can be used. These functions do not usually lead to a perfect reconstruction to the original signal. *Here are two examples using the square pulse and the linear interpolation function to reconstruct the cosine signal.*

7.4 Self-test exercises

- A cosine signal is given by: $x(t) = cos(2\pi ft)$ where $f = 1000Hz$. Plot the sampled signal with the sampling frequency $f_s =$ 200 Hz, 500 Hz, 1000 Hz, 2000 Hz, 4000 Hz and 8000 Hz. Determine the corresponding discrete frequencies. Answer: ω =0, 0, 0, π, $\pi/2$, $\pi/4$.
- If the sampling frequency is 8000 Hz, assuming no aliasing and folding, what is the cut-off frequency for the ideal low pass filter to reconstruct the original signal ? Answer $2\pi \times 4000$ Hz
- Run through demonstrations in Chapter 4.

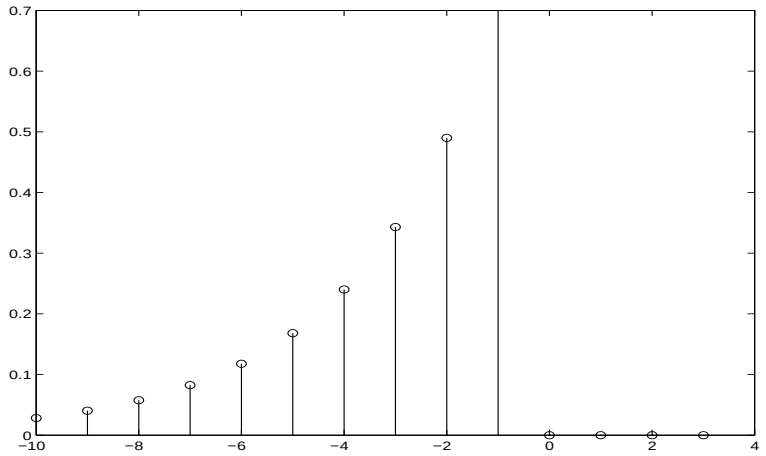


Figure 26: The signal $x[n] = -a^n u[-n-1]$

8 z-Transform

This section covers a little bit more than Chapter 7 of the text book. In particular, we will discuss the region of convergence and the inverse z-transform.

8.1 Definition and properties

The z-transform $X(z)$ of a sequence $x[n]$ is defined as:

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

Note for a finite duration sequence:

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

For example, the z-transform of the sequence $x[n] = \{1, 2, 1\}$ is

$$X(z) = x(0)z^{-0} + x(1)z^{-1} + x(2)z^{-2} = 1 + 2z^{-1} + z^{-2}$$

The z-transform of the unit sample sequence $x[n] = \delta[n]$ and $y[n] = \delta[n-n_0]$ are

$$X(z) = \sum_{n=-\infty}^{\infty} \delta[n]z^{-n} = \delta[0]z^{-0} = 1$$

and

$$Y(z) = \sum_{n=-\infty}^{\infty} y[n]z^{-n} = \delta[n_0]z^{-n_0} = z^{-n_0}$$

These examples are used to show how to perform the z-transform for simple signals. However, more careful calculations are required for other signals. For example, the z-transform of the signal shown in Figure 26 $x[n] = -a^n u[-n-1]$ (where $u[n]$ is the unit step signal) is

$$\begin{aligned} X(z) &= \sum_{n=-\infty}^{\infty} x[n]z^{-n} \\ &= -\sum_{n=-\infty}^{-1} a^n z^{-n} \\ &= 1 - \sum_{n=0}^{\infty} (a^{-1}z)^n \end{aligned}$$

If $|a^{-1}z| < 1$ or equivalently, $|z| < |a|$, then

$$X(z) = 1 - \frac{1}{1-a^{-1}z} = \frac{1}{1-az^{-1}}$$

You see that a signal of infinite duration is now represented by a formula in the z-transform domain. Here is another example, the z-transform of a signal given by: $x[n] = a^n u[n]$ is

$$\begin{aligned} X(z) &= \sum_{n=-\infty}^{\infty} x[n]z^{-n} \\ &= \sum_{n=0}^{\infty} a^n z^{-n} \\ &= \sum_{n=0}^{\infty} (az^{-1})^n \end{aligned}$$

If $|az^{-1}| < 1$ or equivalent, $|z| > |a|$, then

$$X(z) = \frac{1}{1-az^{-1}}$$

An interesting point to note is that in these two examples, although the two signals are different, their z-transforms are the same, except that the conditions are different. We will take a look at this point later on.

The z-transform has some useful properties that are listed below

Linearity	$ax_1[n] + bx_2[n]$	$aX_1(z) + bX_2(z)$
Time shifting	$x[n-n_0]$	$z^{-n_0}X(z)$
Time reversal	$x[-n]$	$X(1/z)$
Convolution	$x[n] * h[n]$	$X[z]H[z]$
Multiplication by an exponential sequence	$z_0^n x[n]$	$X(z/z_0)$
Modulation	$e^{jn\pi}x[n] = (-1)^n x[n]$	$X(-z)$

An important point to note is that for an LTI system, the convolution in the time domain is equivalent to the multiplication in the z-domain. This is similar to the Fourier transform. The Fourier is indeed closely related to the z-transform. For example, we can perform the convolution of two signals, denoted by $x_1[n] = \{1\,2\,2\,1\}$ and $x_2[n] = \{1\,-2\,1\}$ by using the z-transform. Performing the z-transform on both signals we have

$$X_1(z) = 1 + 2z^{-1} + 2z^{-2} + z^{-3}$$

and

$$X_2(z) = 1 - 2z^{-1} + z^{-2}$$

Therefore

$$X_1(z)X_2(z) = 1 - z^{-2} - z^{-3} + z^{-5}$$

The result of the convolution is thus $x_1[n] * x_2[n] = \{1\,0\,-1\,-1\,0\,1\}$. As another example, we take a look at the system function (the z-transform of the impulse response) of the a filter, denoted by $H_1(z)$, expressed in terms of another filter $H_0(z)$ as

$$H_1(z) = (-z)^{-N} H_0(-z^{-1})$$

Assume $h_0[n] = \{h_0\, h_1\, h_2\, h_3\} = \{1\,2\,2\,1\}(N=3)$, determine the impulse response of the filter given by $H_1(z)$. To solve this problem, we determine $H_0(-z^{-1})$ first. Since

$$H_0(z) = 1 + 2z^{-1} + 2z^{-2} + z^{-3}$$

then

$$H_0(-z^{-1}) = 1 - 2z + 2z^2 - z^3$$

Therefore, we have

$$H_1(z) = (-z)^{-3}(1 - 2z + 2z^2 - z^3) = 1 - 2z^{-1} + 2z^{-2} - z^{-3}$$

The impulse response is $h_1[n] = \{1\ -2\ 2\ -1\}$. Actually, $H_0(-z^{-1})$ corresponds to two operations: time reversal indicated by z^{-1} and changing the sign of the odd indexed coefficients indicated by the negative sign. Then z^{-N} is a delay operation to make it a causal filter.

8.2 The z-transform and the Fourier transform

To see their relationship, we note that

$$z = re^{j\omega}$$

Therefore, the z-transform can be re-written as

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

When $r = 1$, the above equation defines the Fourier transform. In other words, when we evaluate the z-transform on the unit circle, we obtain the Fourier transform:

$$X(\omega) = X(z)|_{z=e^{j\omega}}$$

This is because the unit circle is defined as $|z| = 1$ (refer to Section 2 on complex numbers).

The interpretation of variable ω is that it is the angle between a vector that points to a point on the unit circle and the real axis of the complex z-plane. You can refer to Figure 7. With this interpretation, we can see that $\omega = 0$ corresponds to $z = 1$ and $\omega = \pi$ corresponds to $z = -1$. Therefore, interpreting the Fourier transform as the z-transform evaluation on the unit circle corresponds conceptually to wrapping the linear frequency axis around the unit circle. The inherent periodicity property of the frequency ω is captured naturally since a change of an angle by 2π in the z-plane corresponds to traversing the unit circle once and returning to exactly the same location.

There is an interesting demonstration in the CDROM (in section 8, a demo called Z to Freq).

8.3 The z-transform as an operator

The time delay property of the z-transform can be used as an operator. For example, a system given by: $y[n] = x[n] - x[n-1]$ can be written as

$$y[n] = (1 - z^{-1})x[n] = x[n] - z^{-1}x[n] = x[n] - x[n-1]$$

Here we have taken the interpretation that $z^{-1}x[n] = x[n-1]$. In other words, we regard z^{-k} as a delay operator that causes a signal to delay by k samples. Figure 27 shows two equivalent block diagrams that show the implementation of a system.

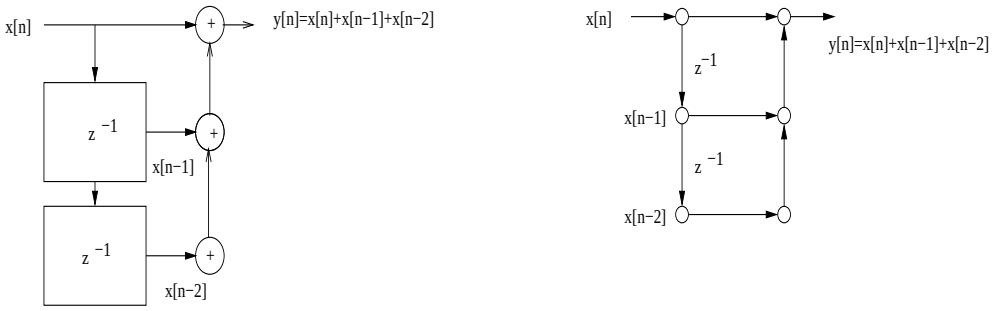


Figure 27: Two equivalent block diagrams that show the implementation of $y[n] = x[n] + x[n-1] + x[n-2]$

8.4 Region of convergence

We have seen in Section 8.1 that for sequences of finite duration, the z-transform always exists. However, for sequences of infinite duration, the summation (the z-transform) *converges* for only the set of values of z. This is called the Region of convergence (ROC). The condition for the z-transform to converge is stated as

$$\sum_{n=-\infty}^{\infty} |x[n]r^{-n}| < \infty$$

Two signals, their z-transforms and ROCs are listed below

Signal	z-transform	ROC
$x[n] = a^n u[n]$	$X(z) = \frac{1}{1-az^{-1}}$	$\|z\| > \|a\|$
$x[n] = -a^n u[-n-1]$	$X(z) = \frac{1}{1-az^{-1}}$	$\|z\| < \|a\|$

We can see that the two different signals have the same z-transform with different ROCs. Therefore, the z-transform together with the ROC uniquely specifies a signal. Figure 28 shows the ROCs.

Let us consider the ROC of a z-transform: $X(z) = 1 + 2z^{-1} + 2z^{-2} + z^{-3}$. Its ROC is the whole z-plane, excluding the origin where $z = 0$. As another example, we want to determine the z-transform of a signal

$$x[n] = (\frac{1}{2})^n u[n] + (-\frac{1}{3})^n u[n]$$

According to the above table and using the linearity property, we obtain

$$X(z) = \frac{1}{1-\frac{1}{2}z^{-1}} + \frac{1}{1+\frac{1}{3}z^{-1}} = \frac{1}{1-\frac{1}{2}z^{-1}} + \frac{1}{1-(-\frac{1}{3})z^{-1}}$$

where the ROC for the first term is $|z| > 1/2$ and the ROC for the second term is $|z| > |-1/3| = 1/3$. The ROC that satisfies both conditions is $|z| > 1/2$.

In general, the problem of finding the ROC is equivalent to determining the values of r, such that

$$|X(z)| \leq \sum_{n=-\infty}^{\infty} |x[n]z^{-n}| = \sum_{n=-\infty}^{\infty} |x[n]r^{-n}e^{-j\omega n}| = \sum_{n=-\infty}^{\infty} |x[n]r^{-n}| < \infty$$

Note that we can decompose our signal into two parts:

$$x[n] = x_1[n] + x_2[n]$$

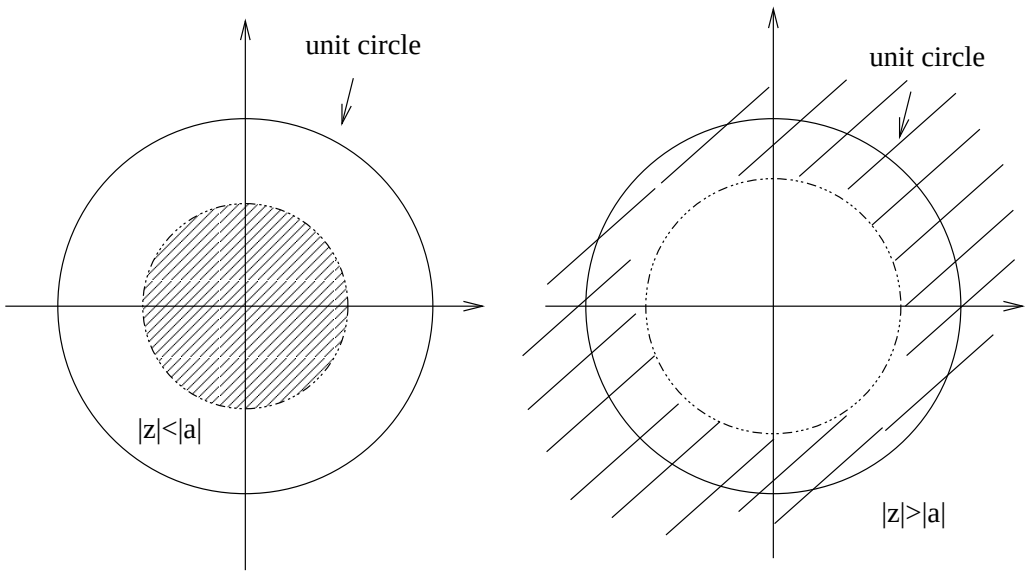


Figure 28: The ROCs

where

$$x_1[n] = \begin{cases} x[-n], & n > 0 \\ 0, & n \leq 0 \end{cases}$$

and

$$x_2[n] = \begin{cases} x[n], & n \geq 0 \\ 0, & n > 0 \end{cases}$$

With these notations, we have[10]

$$\begin{aligned} \sum_{n=-\infty}^{\infty} |x[n]r^{-n}| = & \sum_{n=-\infty}^{-1} |x[n]r^{-n}| + \sum_{n=0}^{\infty} |x[n]r^{-n}| \\ = & \sum_{n=1}^{\infty} |x_1[n]r^{n}| + \sum_{n=0}^{\infty} |x_2[n]r^{-n}| \end{aligned}$$

We can easily see that the ROC for the signal $x_1[n]$ is $|z| < r_1$, while that for $x_2[n]$ is $|z| > r_2$, where r_1 and r_2 are two positive constants that depend on the signal. If $r_1 > r_2$, then the ROC is $r_1 < |z| < r_2$. This is a ring shape region in the z-plane. Otherwise, if $r_1 < r_2$, the z-transform does not exist for the signal.

In summary, the ROC provides useful information about the signal.

- If the signal is of finite duration, then its ROC is the whole z-plane except $z = 0$ or $z = \infty$
- If the signal is a left-sided sequence, i.e., a sequence that is zero for $n < N < \infty$ (N is a positive integer), then its ROC is within the region of a circle, except $z = 0$.
- If the signal is a right-sided sequence, i.e., a sequence that is zero for $n > N > -\infty$ (N is a positive integer), then its ROC is outside a circle.
- If the signal is a two-sided sequence, then its ROC is within a ring-shaped region.

[10]Note, if we let $m = -n$, then $\sum_{n=-\infty}^{-1} |x[n]r^{-n}| = \sum_{m=1}^{\infty} |x[-m]r^{m}| = \sum_{n=1}^{\infty} |x_1[n]r^{n}|$

8.5 The Inverse z-transform

Performing the inverse z-transform is relatively simple in some cases, but it may be quite a complicated task in other cases. In this subject, we only discuss how to use the z-transform table and a number of its properties to perform the inverse transform.

Here are a few examples that show you how to determine the inverse transform

- $X(z) = z^2 + z^{-3} - 0.5z^{-6}$. Answer: $x[n] = \delta[n+2] + \delta[n-3] - 0.5\delta[n-6]$
- $X(z) = \frac{z^{-1}}{1-\frac{1}{4}z^{-1}}$, $|z| > 1/4$. Answer: $X(z) = \frac{z^{-1}}{1-\frac{1}{4}z^{-1}} = z^{-1}\frac{1}{1-\frac{1}{4}z^{-1}}$, $x[n] = (\frac{1}{4})^{n-1}u[n-1]$.
- $X(z) = \frac{1}{(1-az^{-1})(1-z^{-1})}$, $|a| < 1$, $|z| > 1$. Answer: $X(z) = \frac{1}{1-a}\left(\frac{1}{1-z^{-1}} - \frac{a}{1-az^{-1}}\right)$, $x[n] = \frac{1}{1-a}\left(u[n] - a^{n+1}u[n]\right)$

The z-transform of a sequence, denoted by

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{l=0}^{N} a_l z^{-l}}$$

where $N > M$ can be expressed as

$$H(z) = \sum_{k=1}^{N} \frac{A_k}{1 - p_k z^{-1}}$$

where p_k is its kth pole and

$$A_k = H(z)(1 - p_k z^{-1})|_{z=p_k}$$

The inverse z-transform is given by (assuming the ROC of $|z| > max(|p_k|)$)

$$h[n] = \sum_{k=1}^{N} A_k p_k^n u[n]$$

For example, a causal and stable filter is described by the difference equation

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

Its system function is

$$\begin{aligned} H(z) = & \frac{2(1+z^{-1})}{1-z^{-1}+z^{-2}} \\ = & \frac{2p_1^*}{1-p_1 z^{-1}} + \frac{2p_1}{1-p_1^* z^{-1}} \end{aligned}$$

where $p_1 = e^{j\pi/3}$. The impulse response is obtained by taking the inverse z-transform

$$\begin{aligned} h[n] = & 2e^{-j\pi/3}e^{jn\pi/3}u[n] + 2e^{j\pi/3}e^{-jn\pi/3}u[n] \\ = & 4\cos(\tfrac{2\pi}{6}(n-1))u[n] \end{aligned}$$

8.6 Self-test exercises

- Go through the demonstrations in Chapter 7 of the text book
- Use the ZT to calculate the convolution: $x[n] * h[n]$, where $x[n] = \{1, 0, 2, 3\}$ and $h[n] = \{1, -1, -1, 1\}$
- Use an example to show the ZT is a linear transform
- Derive the convolution property
- Use an example to show the time delay property of the ZT

- A system is a cascade of two causal LTI systems: $h_1[n] = \{1, 2, 1\}$ and $h_2[n] = \{1, -2, 1\}$. Determine the system function, the frequency response and the unit sample response of the system.
- Determine the inverse z-transform for the following:

$$H(z) = z^2 + 3z^{-2} - z^{-4}$$

$$H(z) = \frac{0.5z^{-2}}{1 - 0.2z^{-1}}, \ |z| > 0.2$$

$$H(z) = \frac{1 + 0.5z^{-2}}{1 - 0.2z^{-1}}, \ |z| > 0.2$$

Answers

- $Y[z] = X[z]H[z] = (1 + 2z^{-2} + 3z^{-3})(1 - z^{-1} + z^{-2} - z^{-3})$. Perform the inverse z-transform to determine $y[n]$
- Assume $x_1[n] = \{1, 2\}$ and $x_2[n] = \{1, 0, 2\}$, we make a new signal $x[n] = ax_1[n] + bx_2[n]$, you can show that $X(z) = aX_1(z) + bX_2(z)$
- The system function is the z-transform of the unit sample response. In this case, the overall response is $h[n] = h_1[n] * h_2[n]$. So $H(z) = H_1(z)H_2(z)$. $H(\omega) = H(z)|_{z=e^{j\omega}}$
- $H(z) = z^2 + 3z^{-2} - z^{-4}$, $h[n] = \delta[n+2] + 3\delta[n-2] - \delta[n-4]$
- $H(z) = \frac{0.5z^{-2}}{1-0.2z^{-1}}, \ |z| > 0.2$, $h[n] = 0.5(0.2)^{n-2}u[n-2]$
- $H(z) = \frac{1+0.5z^{-2}}{1-0.2z^{-1}}, \ |z| > 0.2$, $h[n] = 0.2^n u[n] + 0.5(0.2)^{n-2}u[n-2]$

9 Implementation of FIR and IIR filters

9.1 Finite impulse response (FIR) filter

An FIR filter is usually represented by a linear constant coefficient difference equation

$$y[n] = \sum_{m=0}^{M} b_m x[n-m]$$

where the parameter M is called the order of the filter. For example, a half-band low pass filter is given by

$$y[n] = \sum_{m=0}^{10} b_m x[n-m]$$

where the filter coefficients are given by

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}
0.005	0	-0.042	0	0.289	0.497	0.289	0	-0.042	0	0.005

The impulse response (unit sample response) of the filter is obtained by using the signal $\delta[n]$ as the input:

$$h[n] = 0.005\delta[n] - 0.042\delta[n-2] + 0.289\delta[n-4] + 0.497\delta[n-5] + 0.289\delta[n-6] - 0.042\delta[n-8] + 0.005\delta[n-10]$$

This can also be represented as:

$$h[n] = \{0.005,\ 0,\ -0.042,\ 0,\ 0.289,\ 0.497,\ 0.289,\ 0,\ -0.042,\ 0,\ 0.005\}$$

We also know that an LTI system can be represented by the convolution

$$y[n] = \sum_{m=0}^{10} h[m]x[n-m]$$

You can easily verify that

$$h[m] = b_m$$

Therefore, given the unit sample response of the FIR filter, you can directly write down the difference equation.

In the frequency domain, an FIR filter is characterized by its system function or its frequency response, defined as

$$H(z) = \sum_{m=0}^{M} h[m]z^{-m}$$

and

$$H(\omega) = \sum_{m=0}^{M} h[m]e^{-jm\omega}$$

The time domain and frequency domain representations of the above half-band filter are shown in Figure 29. The following Matlab codes are used to produce these results

```
b = fir1 (10, 0.5); %design an fir filter
subplot ( 211 ); stem ( 0:10, b ); grid
subplot ( 212 ); plot ( 0 : pi/128 : pi*127/128, abs (freqz ( b, 1, 128 )));
```

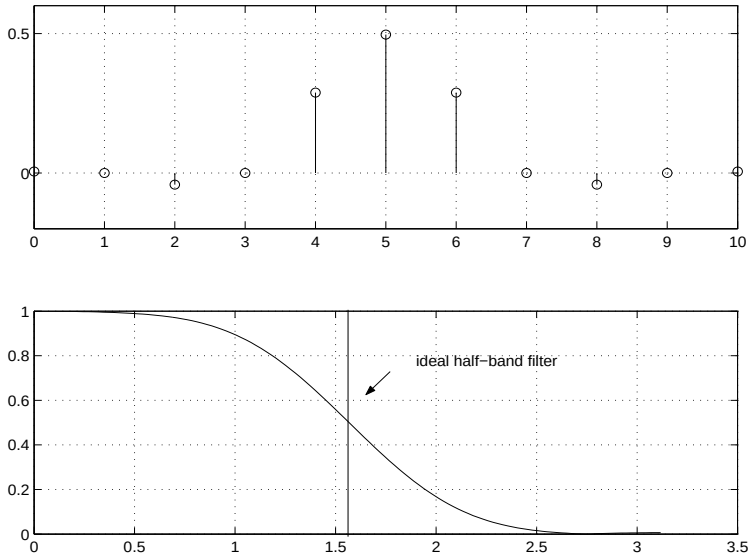


Figure 29: Top: a plot of h[n]. Bottom: the amplitude response of the half-band filter

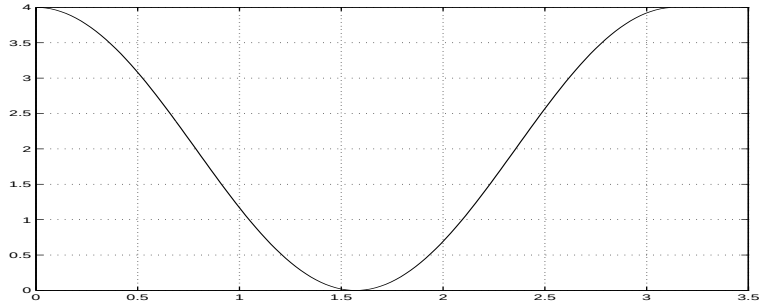


Figure 30: The amplitude response of an FIR filter h[n]={1, 0, 2, 0, 1}

Here is one more example. The unit sample response of the filter is: $h[n] = \{1, 0, 2, 0, 1\}$. The difference equation is

$$y[n] = x[n] + 2x[n-2] + x[n-4]$$

Its frequency response and system function are

$$\begin{aligned} H(\omega) = & \quad 1 + 2e^{-j2\omega} + e^{-j4\omega} \\ = & \quad (1 + e^{-j2\omega})^2 \\ = & \quad e^{-j2\omega} \left(2\frac{e^{j\omega} + e^{-j\omega}}{2}\right)^2 \\ = & \quad 4\cos^2(\omega) e^{-j2\omega} \end{aligned}$$

and

$$H(z) = 1 + 2z^{-2} + z^{-4}$$

The amplitude response is

$$|H(\omega)| = \left|4\cos^2(\omega) e^{-j2\omega}\right| = 4\cos^2\omega$$

This is shown in Figure 30. Note that this filter is neither a low pass nor a high pass filter. It passes the low frequency and high frequency components and attenuates those components whose frequencies are close to $\pi/2$. What is the output for an input signal $x[n] = \cos(0.5n) + \cos(\frac{\pi}{2}n)$?

9.2 Infinite impulse response (IIR) filter[11]

An IIR filter is usually represented by a linear constant coefficient difference equation

$$y[n] = \sum_{k=1}^{N} a_k y[n-k] + \sum_{m=0}^{M} b_m x[n-m]$$

This equation represents an LTI system, only when the initial rest condition is satisfied. The impulse response is obtained when we use the unit sample sequence as the input. For example, we consider the simplest IIR filter given by:

$$y[n] = a_1 y[n-1] + b_0 x[n]$$

Its impulse response is

$$h[n] = a_1 h[n-1] + b_0 \delta[n]$$

The following table shows how to recursively calculate the impulse response.

n	$n<0$	0	1	2	3	4
$\delta[n]$	0	1	0	0	0	0
$h[n-1]$	0	0	b_0	$b_0 a_1$	$b_0 a_1^2$	$b_0 a_1^3$
$h[n]$	0	b_0	$b_0 a_1$	$b_0 a_1^2$	$b_0 a_1^3$	$b_0 a_1^4$

From this table, we can see that for an input signal such as $\delta[n]$ with only one non-zero data sample, the output of the filter is an infinite-length sequence:

$$h[n] = \begin{cases} b_0 a_1^n, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

Therefore, convolution is not a practical way to calculate the output of an IIR filter. In most cases, we use the difference equation. Let us take a look at another example. The IIR filter is given by

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

and the input is $x[n] = \delta[n] + \delta[n-1]$. We can use our previous results by using $a_1 = 1/2$ and $b_0 = 1/2$. The output is given by

$$\begin{aligned} y[n] &= h[n] + h[n-1] \\ &= \tfrac{1}{2}(\tfrac{1}{2})^n + \tfrac{1}{2}(\tfrac{1}{2})^{n-1} \\ &= 1.5 \times 0.5^{n-1} \end{aligned}$$

You can also use the difference equation to obtain the same result. Note that in both examples, we have assumed the initial rest condition.

In the frequency domain, an IIR filter is characterized by its system function, defined as

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{m=0}^{M} b_m z^{-m}}{1 - \sum_{k=1}^{N} a_k z^{-k}}$$

Note that in deriving the above equation, we have used an equivalent representation

$$y[n] = \sum_{k=1}^{N} a_k y[n-k] + \sum_{m=0}^{M} b_m x[n-m]$$

[11] An IIR filter can be motivated from a feedback system point of view. It can be motivated from a computational point of view. Its different from FIR filters in a number of aspects. I will leave these to the next major revision.

For example, the system function of an IIR filter

$$y[n] = 2y[n-1] + 3y[n-2] + 4x[n-1] + 5x[n-2]$$

can be calculated by taking the z-transform of the both side of the equation as

$$Y(z) = 2z^{-1}Y(z) + 3z^{-2}Y(z) + 4z^{-1}X(z) + 5z^{-2}X(z)$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{4z^{-1} + 5z^{-2}}{1 - 2z^{-1} - 3z^{-2}}$$

Matlab provides the functionality to perform the filtering process and to evaluate the frequency response: `filter`(b, a, x) and `freqz`(b, a, N), where a and b are two vectors that store the filter coefficients, x represents the signal to be processed and N the number of points to evaluate the frequency response from 0 to π. For an FIR filter, $a = 1$. For an IIR filter $b = [b_0, b_1, ..., b_M]$ and $a = [1, a_1, a_2, ..., a_N]$. You must be careful when forming the vector a. In the previous example, the vector is $a = [1, -2, -3]$, it is NOT $a = [1, 2, 3]$. To avoid this problem, before forming the vector, you can re-arrange the difference equation so that all the output terms are on the left hand side and all the input terms are on the right hand side. To plot the amplitude response, you can use the following Matlab code:

```
b = [0 4 5] ;

a = [1 -2 -3] ;

n = 0 : pi / 128 : 127 * pi / 128    %plot from 0 to pi, step size =pi/128

r = freqz ( b , a , 128 ) ;          % frequency response, with 128
                                     %points between 0 and pi

plot ( n, abs ( r ) );               %plot the amplitude

semilogy ( n , abs ( r ) )           %plot with log (base 10) scale
```

Figure 31 shows both types of plots.

9.3 Implementations

There are three basic operations involved in implementing an LTI system: addition, multiplication and unit delay. A block diagram and a signal flow graph for an FIR filter

$$y[n] = 0.7x[n] + 0.3x[n-1]$$

are shown in Figure 32.

The implementation of an IIR filter can be shown in the same way. As an example, we consider a second order difference equation

$$y[n] = b_0x[n] + a_1y[n-1] + a_2y[n-2]$$

The corresponding system function is

$$H(z) = \frac{b_0}{1 - a_1z^{-1} - a_2z^{-2}}$$

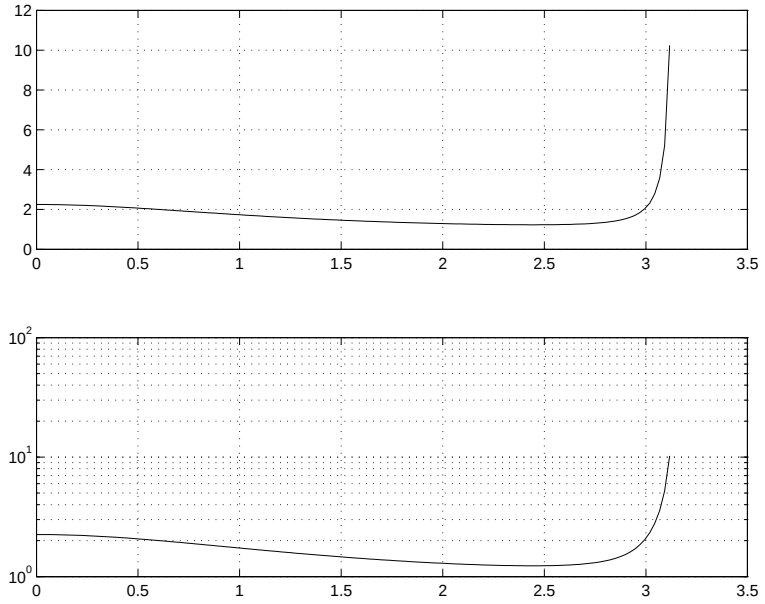


Figure 31: Amplitude response. Bottom plot in log scale

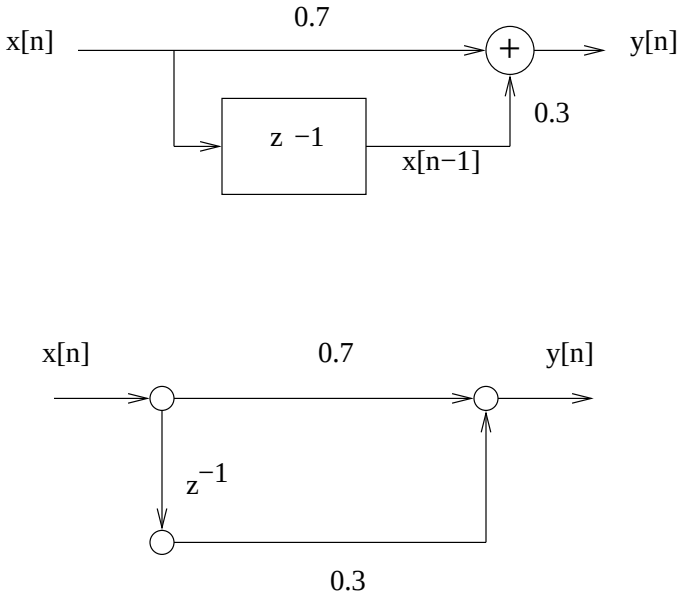


Figure 32: The block diagram (top) and the signal flow graph (bottom) for an FIR filter

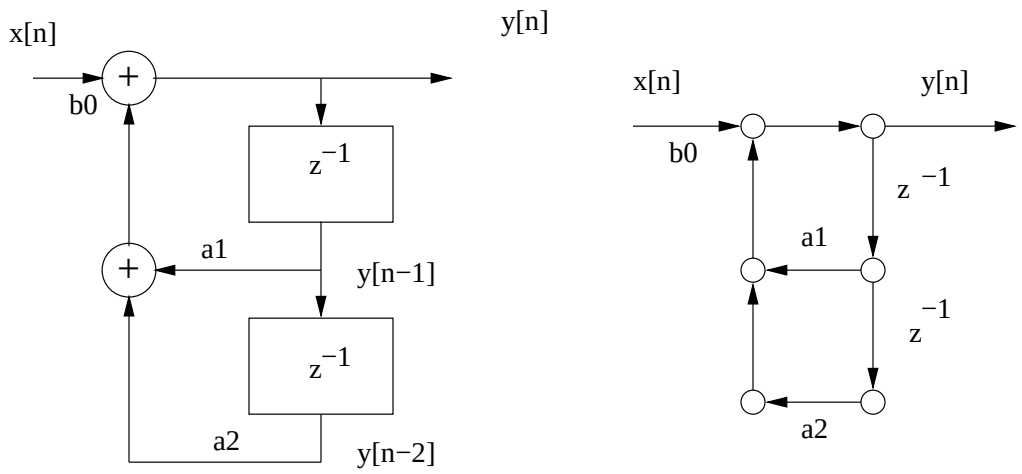


Figure 33: The block diagram (left) and the signal flow graph (right) for an IIR filter

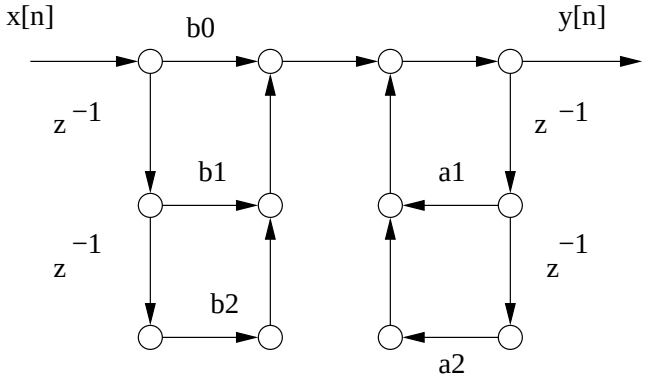


Figure 34: Direct form-I implementation scheme for a second order IIR filter

The block diagram and the signal flow graph are show in Figure 33. A general second order IIR filter is represented by the the difference equation

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] + a_1y[n-1] + a_2y[n-2]$$

Its system function is

$$H(z) = \frac{b_0 + b_1z^{-1} + b_2z^{-2}}{1 - a_1z^{-1} - a_2z^{-2}} = \frac{B(z)}{A(z)}$$

Figure 34 shows its implementation.

The output of the filter can be calculated in two steps:

$$Y[z] = \frac{B(z)}{A(z)} = B(z)\left(\frac{1}{A(z)}X(z)\right)$$

Let

$$W(z) = \frac{1}{A(z)}X(z)$$

The system can be regarded as a cascaded of two sub-systems: one with a system function of $\frac{1}{A(z)}$ and the other with $B(z)$. In the time domain, this system corresponds to two difference equations

$$w[n] = x[n] + a_1w[n-1] + a_2w[n-2]$$

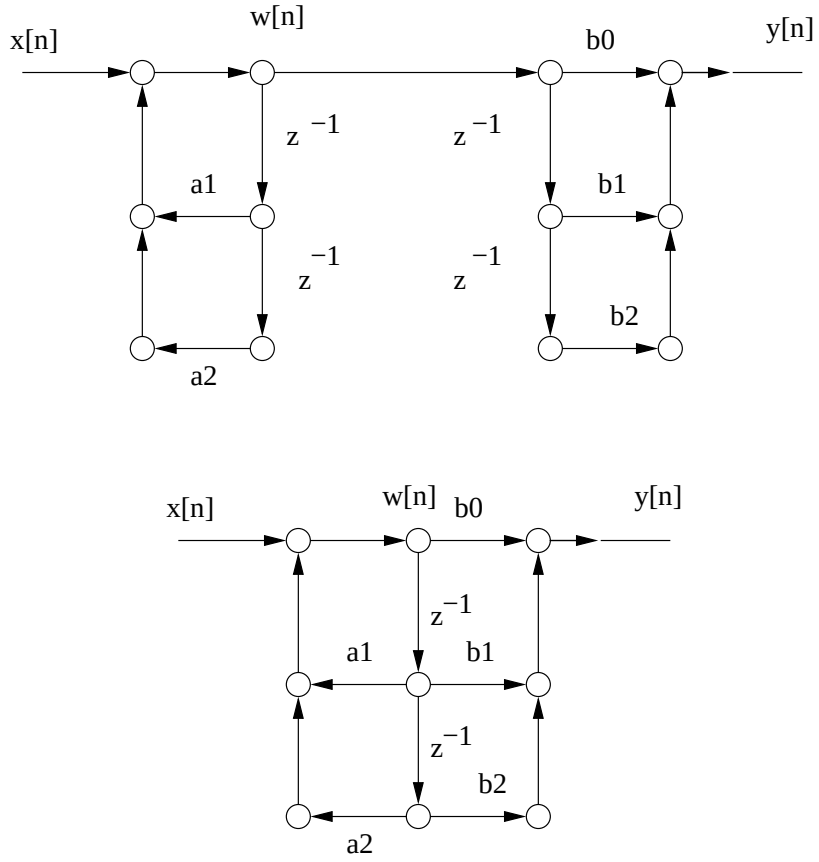


Figure 35: The direct form-II implementation scheme. Top: the two-step implementation. Bottom: the delay units merged together.

and

$$y[n] = b_0 w[n] + b_1 w[n-1] + b_2 w[n-2]$$

These two equations suggest another implementation of an IIR filter, which is called the direct form-II implementation. Compared to direct form-I, it requires smaller number of delay units. Figure 35 shows the above implementation scheme.

9.4 Structures for discrete-time systems[12]

9.4.1 Basic filter structures

$$H(z) = b_0 + b_1 z^{-1}$$

$$H(z) = \frac{1}{1 - a_1 z^{-1}}$$

$$H(z) = \frac{b_0 + b_1 z^{-1}}{1 - a z^{-1}}$$

$$H(z) = \frac{(1 - b_1 z^{-1})(1 - b_2 z^{-1})}{(1 - a_1 z^{-1})(1 - a_2 z^{-1})}$$

[12]This section is still being written. Not required for ELE32DSP, 2001

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}}$$

9.4.2 Structure of FIR filters

For causal FIR systems, the system function is:

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

The structure, which is shown below, is called the transversal filter structure. By properly factorization of the above system function, we have

$$H(z) = \sum_{k=0}^{M} h(k) z^{-k} = \prod_{k=1}^{M_s} \left(b_{0k} + b_{1k} z^{-1} + b_{2k} z^{-2} \right)$$

9.4.3 Structure of IIR filters

A general IIR system has a system function

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

This can be implemented by the direct form I or II. When all coefficients in the above system function are real, then it can be expressed as:

$$H(z) = A \frac{\prod_{k=1}^{M1} (1 - g_k z^{-1}) \prod_{k=1}^{M2} (b_{0k} + b_{1k} z^{-1} + b_{2k} z^{-1})}{\prod_{k=1}^{N1} (1 - c_k z^{-1}) \prod_{k=1}^{N2} (1 - a_{1k} z^{-1} + a_{2k} z^{-1})}$$

Using this factorization, the IIR system can be implemented by a cascade of the basic structures. We can also express the system function in the following way:

$$H(z) = \sum_{k=0}^{M1} c_k z^{-k} + \sum_{k=1}^{N1} \frac{e_{0k} + e_{1k} z^{-1}}{1 - a_{1k} z^{-1} - a_{2k} z^{-2}}$$

Using this factorization, the IIR system can be implemented with the basic structures discussed in section 9.4.1 in a parallel form.

9.5 Self-test exercises

- Draw two implementations for an FIR filter

$$y[n] = 0.1x[n] + 0.25x[n-1] + 0.3x[n-2] + 0.25x[n-3] + 0.1x[n-4]$$

- Draw both direct form-I and II implementations for the following IIR filters

$$y[n] = y[n-1] + 0.5y[n-2] + x[n] + 2x[n-1]$$

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

- Determine the system function of the above three filters

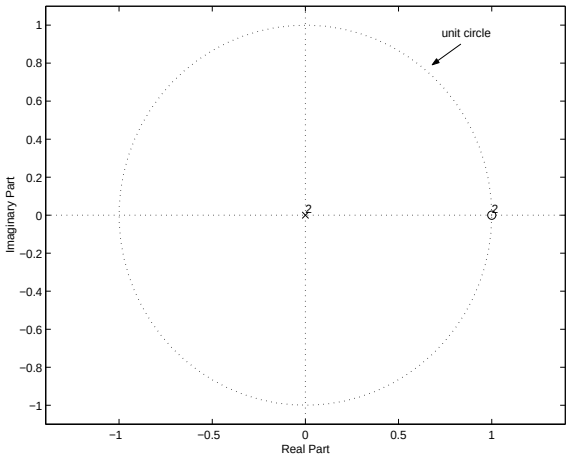


Figure 36: pole-zero plot of $H(z) = 1 - 2z^{-1} + z^{-2}$. This figure is produced by using a Matlab command, `zplane ( [1 -2 1] , [1 0 0] )`.

10 LTI system analysis in the frequency domain

10.1 FIR filters

10.1.1 Zeros and poles

We use a simple example to illustrate the concept of zeros and poles. Consider an FIR filter given by the difference equation

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

Its system function is

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

Therefore, the system function has two zeros

$$z_0 = z_1 = 1$$

and two poles

$$p_0 = p_1 = 0$$

It is useful to display the zeros and poles as points in the complex z-plane. In Matlab, you can use the function `zplane` to generate the pole-zero plot. Figure 36 shows the pole-zero plot of the above system function, where a circle corresponds to a zero and a cross corresponds to a pole. The number beside the circle or the cross indicates the number of zeros and poles at the location. It is very easy to verify that

$$H(z_0) = H(z_1) = 0$$

and

$$H(p_0) = H(p_1) = \infty$$

10.1.2 FIR filters and zeros

Zeros and system function

One important property of zeros is that they are sufficient to determine an FIR system function except for a constant multiplier. In general, a system function can be factorized as

$$H(z) = \sum_{n=0}^{N-1} b_n z^{-n} = A \prod_{n=0}^{N-1} \frac{z - z_n}{z}$$

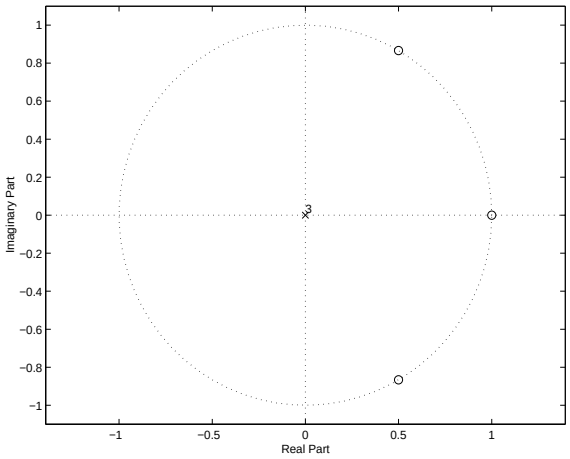


Figure 37: The pole-zero plot of a system.

where A is a constant and z_n is the nth zero of the system function. Therefore, if all zeros of a system function are known, then the system function can be determined to within a scaling constant. For example, a system is given by the pole-zero plot shown in Figure 37. It has three zeros at $z_0 = 1$, $z_1 = e^{j\pi/3}$ and $z_2 = e^{-j\pi/3}$. Its system function can be expressed as (assuming the scaling constant $A = 1$)

$$\begin{aligned} H(z) = & \quad \frac{z-1}{z}\frac{z-e^{j\pi/3}}{z}\frac{z-e^{-j\pi/3}}{z} \\ = & \quad \frac{z^3-2z^2+2z-1}{z^3} \\ = & \quad 1-2z^{-1}+2z^{-2}-z^{-3} \end{aligned}$$

From the system function, we know that in the time domain the system is represented by

$$y[n] = x[n] - 2x[n-1] + 2x[n-2] - x[n-3]$$

We can easily verify that another system given by

$$y[n] = \frac{1}{2}(x[n] - 2x[n-1] + 2x[n-2] - x[n-3])$$

has exactly the same zeros and poles as the previous one.

Zeros and nulling filters

Another important property of zeros is that if the zeros lie on the unit circle, such as those shown in Figures 36 and 37, then certain sinusoidal signals are removed or nulled by the filter. This property is useful in designing an FIR filter to remove sinusoidal interference, such as the 50 Hz signal from the power line. This property is a direct result of the relationship between the frequency response and the system function

$$H(\omega) = H(z)|_{z=e^{j\omega}}$$

When the system function has a zero on the unit circle, denoted by $z_k = e^{j\omega_k}$, the corresponding magnitude of the frequency response at the frequency ω_k is also zero

$$H(\omega_k) = H(z_k) = 0$$

For example, when the input signal is given by

$$x[n] = e^{jn\pi/3}$$

the output signal for the above system is given by[13]

$$\begin{aligned} y[n] = & \quad x[n] - 2x[n-1] + 2x[n-2] - x[n-3] \\ = & \quad e^{jn\pi/3} - 2e^{j(n-1)\pi/3} + 2e^{j(n-2)\pi/3} - e^{j(n-3)\pi/3} \\ = & \quad e^{jn\pi/3}(1 - 2e^{-j\pi/3} + 2e^{-j2\pi/3} - e^{-j\pi}) \\ = & \quad 0 \end{aligned}$$

Therefore, $y[n] = 0$. So we see that a zero, denoted by $z_k = e^{j\omega_k}$, can remove a complex exponential signal

$$x[n] = Ae^{jn\omega_k}$$

Because a cosine signal can be expressed as

$$\cos(\omega_k n) = \frac{1}{2}(e^{j\omega_k n} + e^{-j\omega_k n})$$

to remove this cosine signal, we can use a cascade of two filters, one with a zero z_k and and the other with a zero z_k^* (the conjugate of z_k). Therefore, the system function is given by

$$\begin{aligned} H(z) = & \quad \frac{z-z_k}{z}\frac{z-z_k^*}{z} \\ = & \quad 1 - 2\cos(\omega_k)z^{-1} + z^{-2} \end{aligned}$$

In the time domain, the difference equation is given by

$$y[n] = x[n] - 2\cos(\omega_k)x[n-1] + x[n-2]$$

You can verify that when the input is $x[n] = A\cos(\omega_k n)$, the output of this system is zero.

Pole-zero plot and frequency response

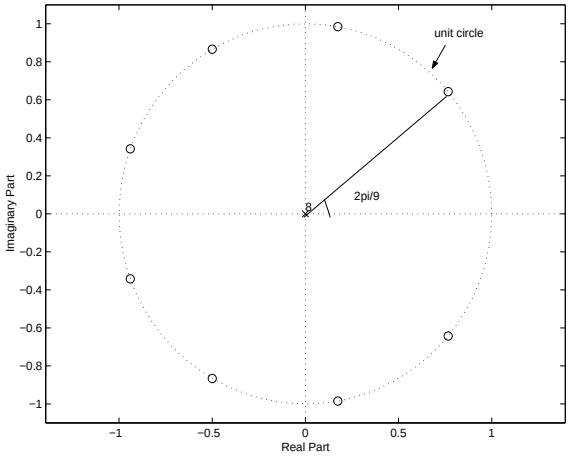
The frequency response of a filter can be obtained by evaluating the system function on the unit circle. Basically, a pole will "push-up" the frequency response and a zero will "pull it down". If a zero is on the unit circle, then it will force the the frequency response to be zero at the corresponding frequency. Figure 38 shows the pole-zero plot of a 9-point running average filter and its magnitude response. From this figure, we can see the relationship between the locations of zeros in the pole-zero plot and the shape of the magnitude response. We can also identify the filter as a low pass filter by inspecting the pole-zero plot. The pass band is between $2\pi/9$ and 0. In the same way, we can identify a high pass or a band pass filter. For example, Figure 37 represents a high pass filter. The pass band is between $\pi/3$ and π.

10.1.3 Linear phase filters

In many applications, it is necessary to ensure that there is no phase distortion of the frequency components of the input signal. In an ideal case, it requires a zero-phase characteristic for the digital filter. However, it is impossible to design a causal FIR filter with zero phase. A linear-phase characteristic represents a pure delay, so a linear-phase FIR filter is desirable in many applications.

[13] It can be shown that

$$1 - 2e^{-j\pi/3} + 2e^{-j2\pi/3} - e^{-j\pi} = 0$$



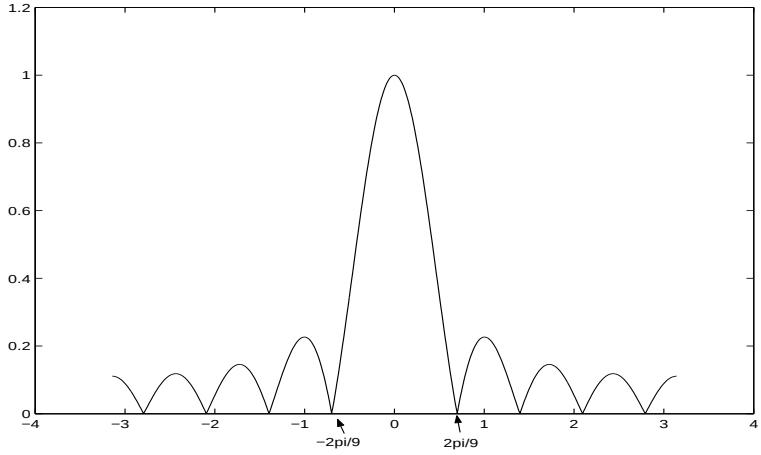


Figure 38: Top: pole-zero plot. Bottom: magnitude response

A special class of FIR filter, denoted by

$$y[n] = \sum_{k=0}^{M} b_k x[n-k]$$

is the linear phase filter satisfying the condition

$$b_k = b_{M-k}$$

for $k = 0, 1, ..., M$. It can be proved that the frequency response of this filter is given by

$$H(\omega) = R(\omega)e^{-j\omega M/2}$$

where $R(\omega)$ is a real valued function given by

$$R(\omega) = \begin{cases} b_{M/2} + \sum_{k=0}^{\frac{M-2}{2}} 2b_k \cos((\frac{M}{2}-k)\omega), & M\,is\,even \\ \sum_{k=0}^{\frac{M-1}{2}} 2b_k \cos((\frac{M}{2}-k)\omega), & M\,is\,odd \end{cases}$$

So the frequency response is a real valued amplitude function times a linear phase factor $e^{-j\omega M/2}$ which corresponds to a delay of M/2 samples. For a discussion of how to interpret a delay of non-integer value, you can refer to page 192-194 of the text book.

If the signal is of finite length and real time processing is not required, then it is possible to have a filtering process with zero-phase. The process involves the following steps

1. $y[n] = h[n] * x[n] \Longleftrightarrow Y(z) = H(z)X(z)$,
2. $u[n] = y[-n] \Longleftrightarrow U(z) = Y(1/z) = H(1/z)X(1/z)$
3. $w[n] = u[n] * h[n] \Longleftrightarrow W(z) = U(z)H(z) = H(z)H(1/z)X(1/z)$
4. $d[n] = w[-n] \Longleftrightarrow D(z) = H(1/z)H(z)X(z)$

In the above steps, we have introduced three internal variables $y[n]$, $u[n]$ and $w[n]$ and used both time domain and z-transform domain representations. The final output is $d[n]$. We can see that it is a filtering output with zero-phase by converting the z-transform representation to the Fourier transform representation

$$D(\omega) = H^*(\omega)H(\omega)X(\omega) = |H(\omega)|^2 X(\omega)$$

In Matlab, you can use the function called **`filtfilt`** to perform the above operations.

10.2 IIR filters

10.2.1 Zeros and poles

An IIR filter is represented in the time domain as

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] + \sum_{l=1}^{N} a_l y[n-l]$$

Its system function is given by

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{l=1}^{N} a_l z^{-l}} = \frac{B(z)}{A(z)}$$

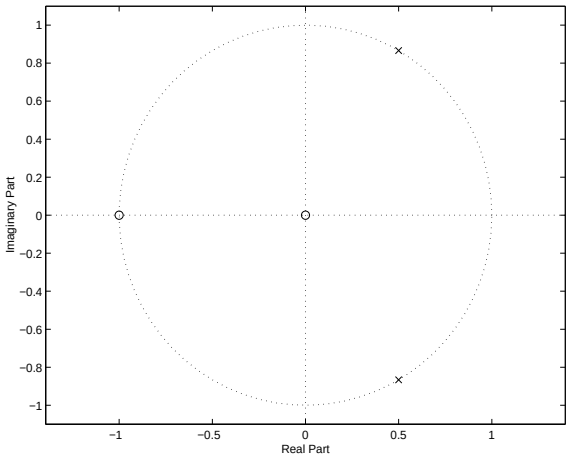


Figure 39: The pole-zero plot of an IIR filter

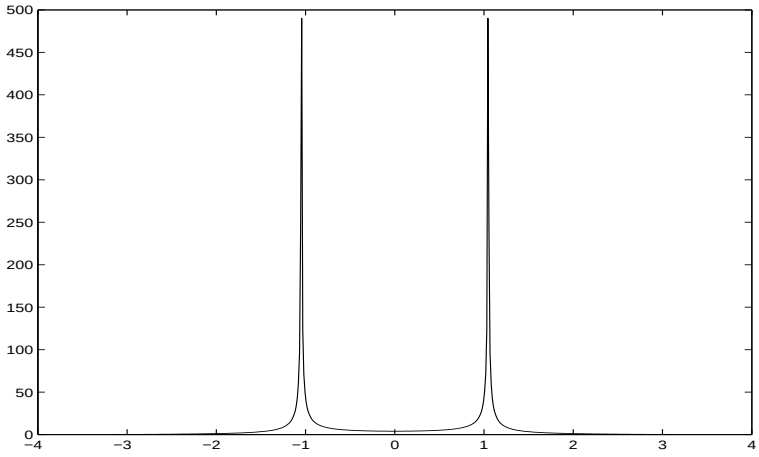


Figure 40: The magnitude response of an IIR filter

Zeros and poles satisfy the condition $A(z) = 0$ and $B(z) = 0$, respectively. For example, the system function

$$H(z) = \frac{2}{1 - 0.5z^{-1}}$$

has a zero $z = 0$ and a pole $p = 0.5$. The system function

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

has two zeros $z_0 = 0$ and $z_1 = -1$ and two poles $p_0 = e^{-i\pi/3}$ and $p_1 = e^{j\pi/3}$. As in the FIR filter case, we can plot the pole-zero plot using the `zplane` command in Matlab. Figure 39 shows the pole-zero plot for the last system function.

We can also look for connections between zeros and poles and the shape of the frequency response. We use the fact that a pole pushes the frequency response up and a zero pulls it down. For the pole-zero plot shown in Figure 39, we know that the filter is a band pass filter with a center frequency at $\pi/3$, this is because the zero (z_1) pull it down at frequency π. The magnitude response of this filter is shown in Figure 40.

Here is an example of how to calculate the amplitude response from the locations of zeros and

poles. We can write

$$\begin{aligned} H(z) &= \frac{2+2z^{-1}}{1-z^{-1}+z^{-2}} \\ &= 2\frac{z+1}{z^2-z+1} \\ &= 2\frac{(z-z_0)(z-z_1)}{(z-p_0)(z-p_1)} \end{aligned}$$

Since $H(\omega) = H(z)|_{z=e^{j\omega}}$, we have the following

$$H(\omega) = 2\frac{(e^{j\omega}-z_0)(e^{j\omega}-z_1)}{(e^{j\omega}-p_0)(e^{j\omega}-p_1)}$$

and

$$|H(\omega)| = 2\frac{|(e^{j\omega}-z_0)||(e^{j\omega}-z_1)|}{|(e^{j\omega}-p_0)||(e^{j\omega}-p_1)|}$$

What does the term $|(e^{j\omega}-z_0)|$ mean? It is the length of the vector that points from the point $e^{j\omega}$ to the point z_0. Figure 41 illustrates this interpretation.

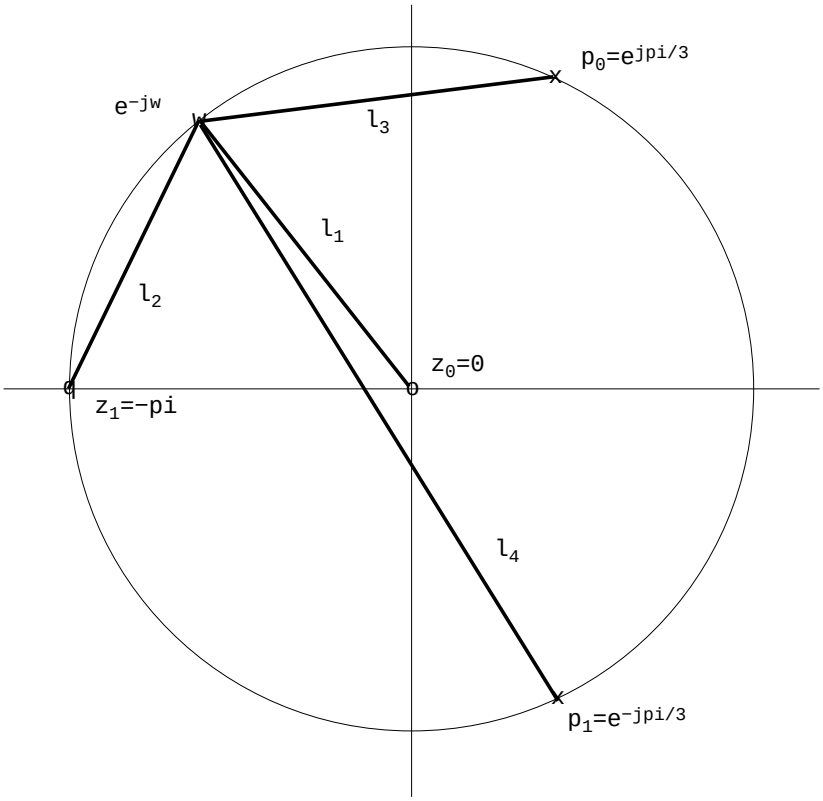


Figure 41: Using the locations of zeros and poles to calculate the amplitude response.

With this interpretation, we can simplify our notation and use $l_1(\omega) = |(e^{j\omega}-z_0)|$, $l_2(\omega) = |(e^{j\omega}-z_1)|$, $l_3(\omega) = |(e^{j\omega}-p_0)|$,$l_4(\omega) = |(e^{j\omega}-p_1)|$. As such, we have

$$|H(\omega)| = 2\frac{l_1(\omega)l_2(\omega)}{l_3(\omega)l_4(\omega)}$$

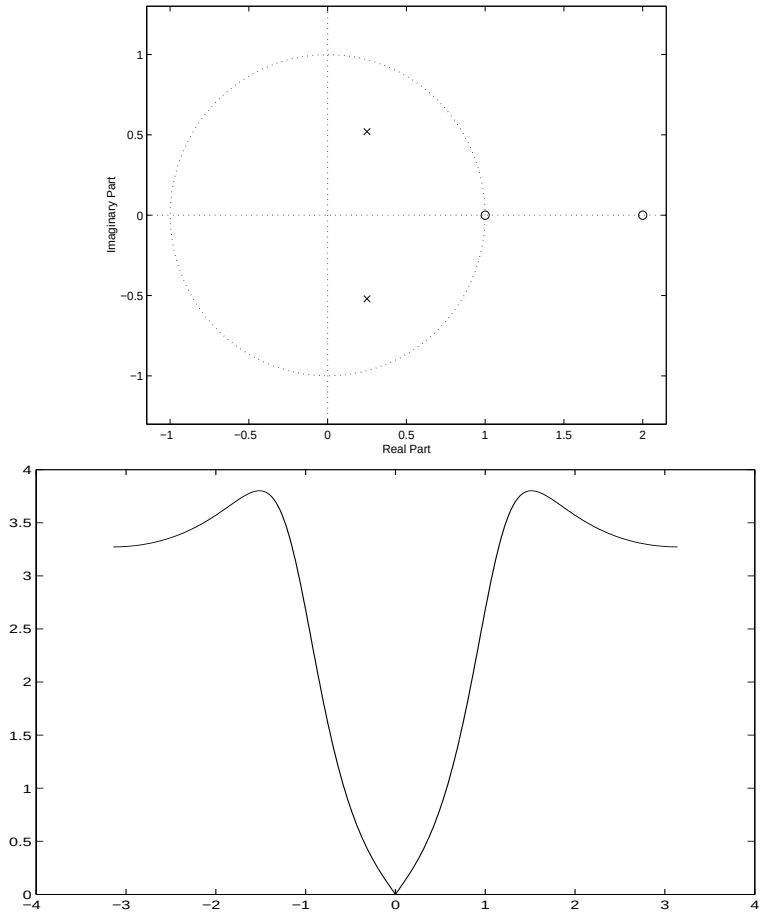


Figure 42: The pole-zero plot and the frequency response of an IIR filter

When we calculate $|H(\omega)|$ by going through the unit circle once (this is the same as $0 \leq \omega < 2\pi$ or $-\pi \leq \omega < \pi$), we have the amplitude response. Now let us calculate the amplitude response for the frequency of π and $\pi/3$. You can easily see that since $l_2(\pi) = 0$, $|H(\pi)| = 0$. Since $l_3(\pi/3) = 0$, $|H(\pi/3)| \to \infty$.

As another example, we take a look at a filter given by

$$y[n] = \frac{1}{2}y[n] - \frac{1}{3}y[n-1] - x[n] + 3x[n-1] - 2x[n-2]$$

Its system function is

$$H(z) = \frac{-1 + 3z^{-1} - 2z^{-2}}{1 - \frac{1}{2}z^{-1} + \frac{1}{3}z^{-2}}$$

The pole-zero plot and the magnitude response is shown in Figure 42. From this figure, you can see the connection between the pole-zero plot and the magnitude response.

10.2.2 Stability condition

A causal IIR filter with initial rest conditions is stable if the poles of its system function lie strictly inside the unit circle. We use a simple example to illustrate this condition. We consider an IIR filter given by

$$y[n] = b_0 x[n] + a_1 y[n-1]$$

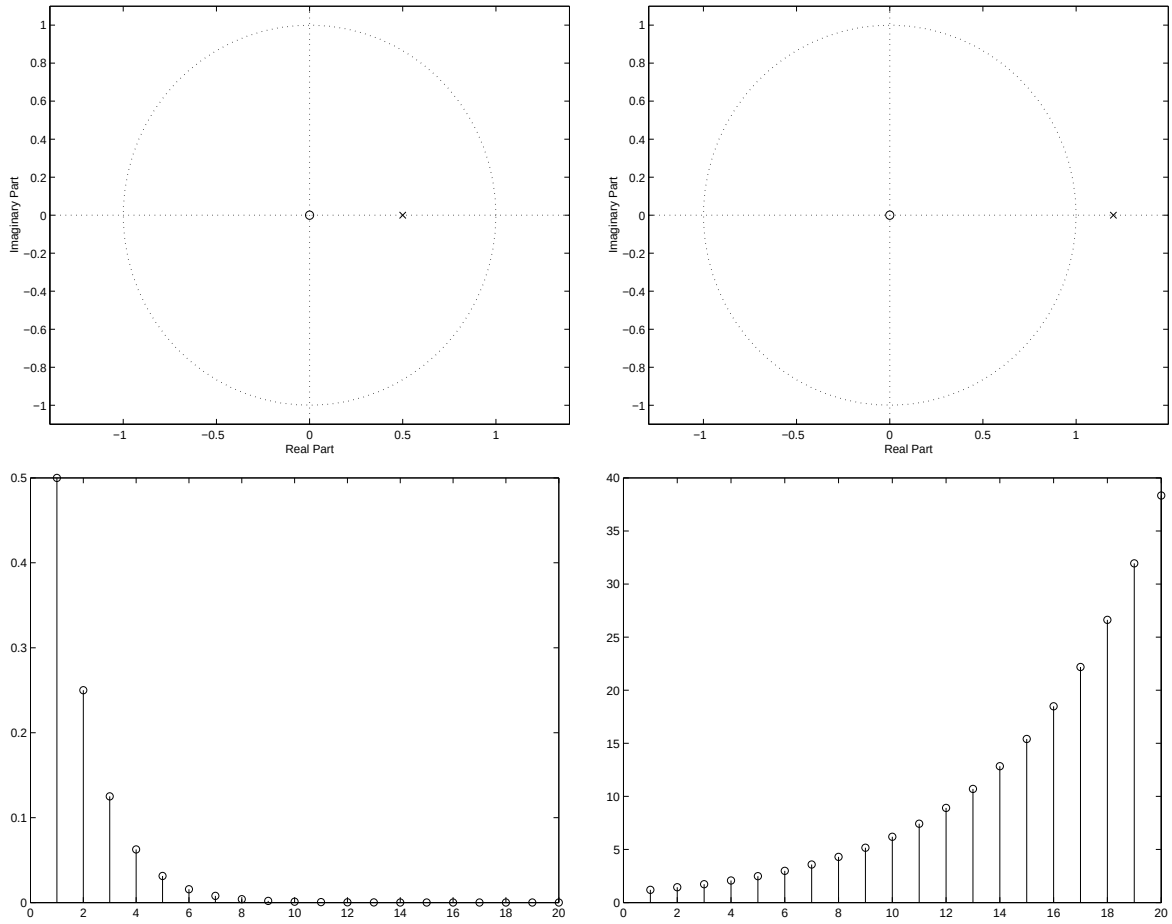


Figure 43: Pole-zero plot and impulse response. Left panel $a_1 = 0.5$, right panel $a_1 = 1.2$.

where a_1 and b_0 are non-zero real numbers. Its system function is

$$H(z) = \frac{b_0}{1 - a_1 z^{-1}} = \frac{b_0 z}{z - a_1}$$

There is a zero $z_0 = 0$ and pole $p_0 = a_1$. The impulse response of this filter can be obtained by taking the inverse z-transform

$$h[n] = b_0 a_1^n u[n]$$

Here we assume $|z| > |a_1|$. If this filter is stable, then it must satisfy the condition

$$\sum |h[n]| < \infty$$

This condition is satisfied only when $|a_1| < 1$. In other words, when the pole is inside the unit circle, the filter is stable. Figure 43 shows two cases where $a_1 = 0.5$ and $a_1 = 1.2$. In both cases, we set $b_0 = 1$. We can see that when the pole is inside the unit circle (left panel), the values of the samples of impulse response are decreasing. When the pole is outside the unit circle, the values of the samples of the impulse response are increasing. Further more, a causal and stable IIR filter requires that the ROC be outside the outermost poles and have all poles inside the unit circle.

10.2.3 Unit sample response

The unit sample response is also related to the locations of poles and zeros. In the CDROM there is a very interesting demonstration using a function called `pez`. We will take a look at the following cases

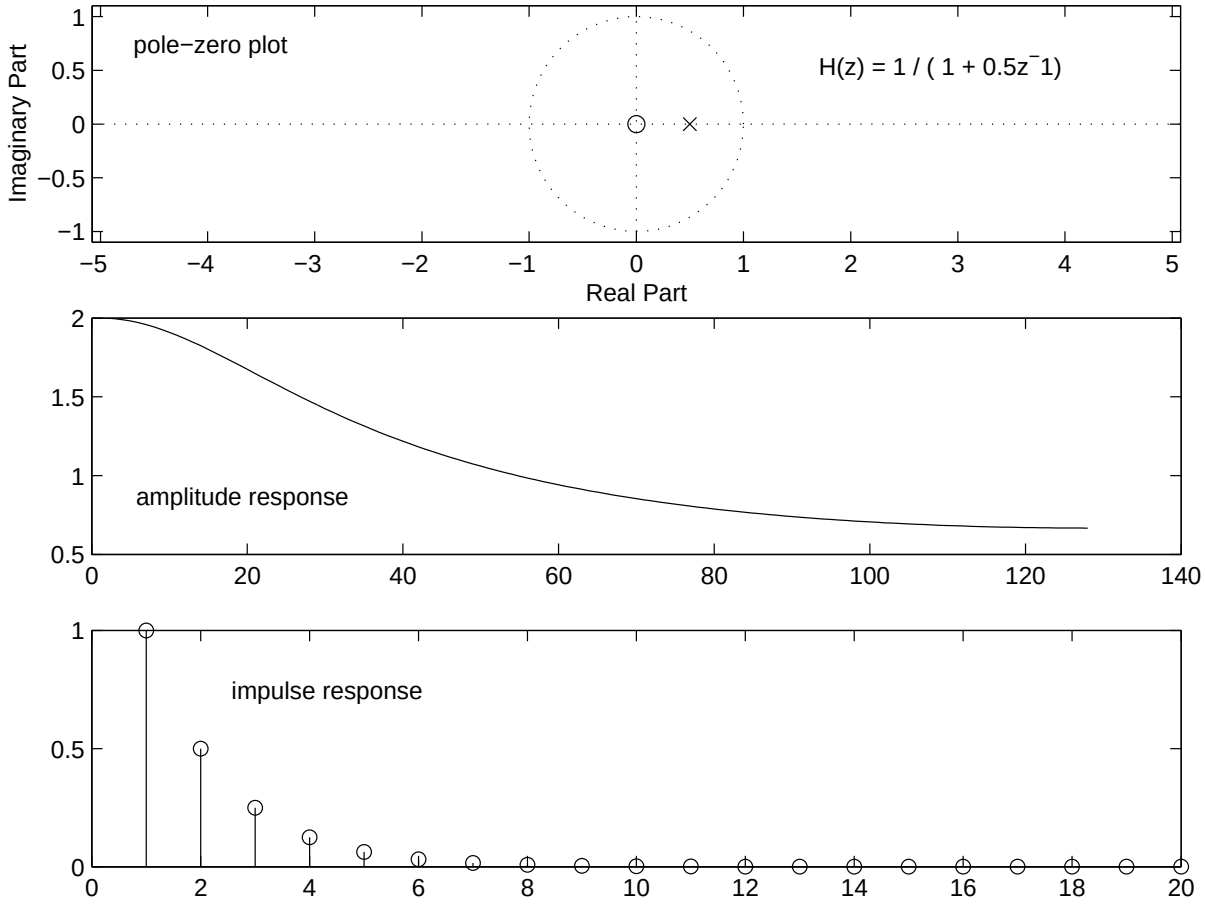


Figure 44: A stable low pass filter.

- A single pole with a zero
- Double poles
- Other configurations of poles and zeros

In the first two cases, we will examine the effects of the locations of poles (inside, on and outside the unit circle). In this demonstration, we can also see that as in the case of an FIR filter, the locations of poles and zeros are also related to the shape of the frequency response. We use a single pole with a zero as an example and plot its pole-zero plot, amplitude response and impulse response. These are shown in figures 44 to47. You can make a number of observations from these figures and see how the three domains of representations of an LTI system are related.

We have already shown that when all poles are inside the unit circle, the filter is stable. What happens when they are on the unit circle? In section 8.5, we studied the IIR filter given by

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

The impulse response of the filter is

$$h[n] = 4\cos(\frac{2\pi}{6}(n-1))u[n]$$

This equation shows that the filter is actually a digital oscillator when the input is an impulse. The pole-zero plot of this filter is shown in Figure 48. There are two poles (located at $\pm\pi/3$) on the unit

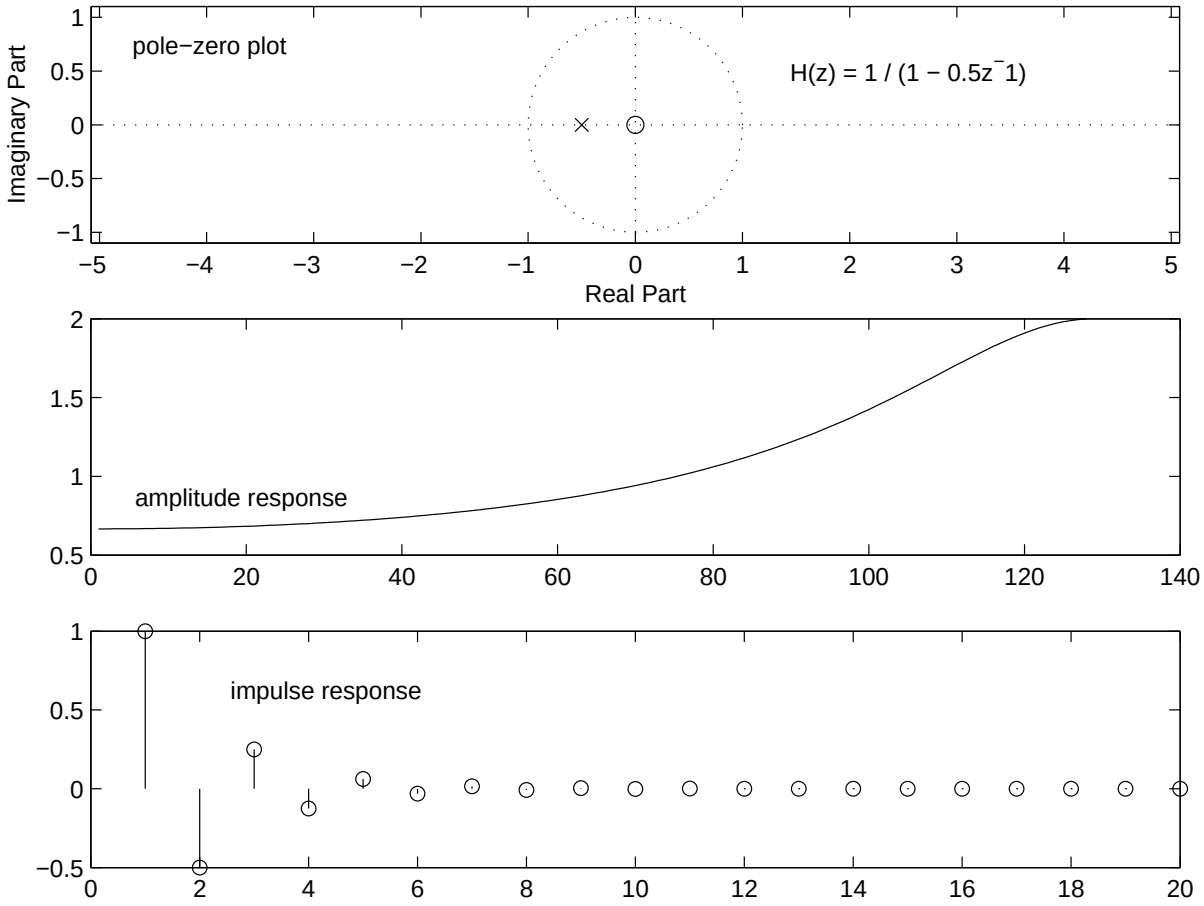


Figure 45: A stable high pass filter.

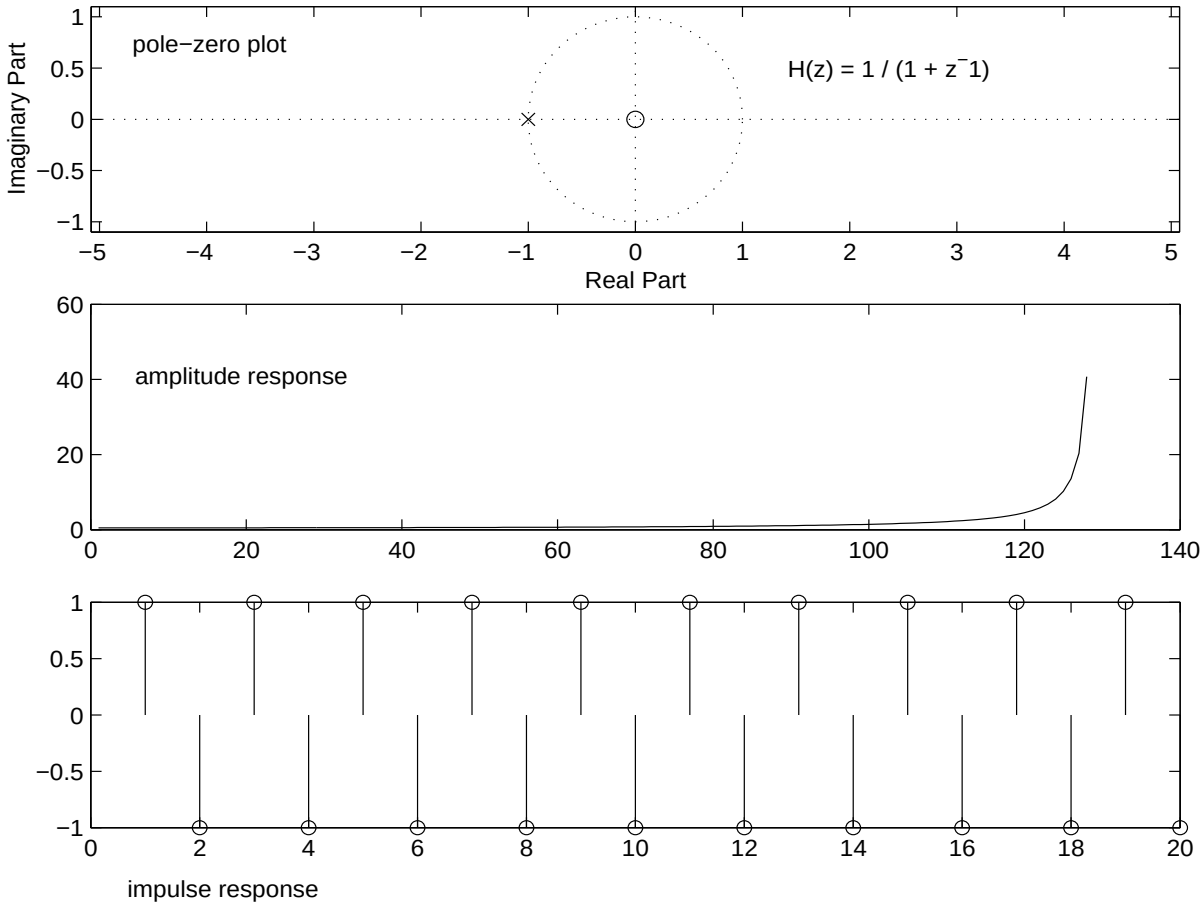


Figure 46: The pole is on the unit circle. The impulse response is a cosine signal with a frequency of π. Can you change the frequency to other values, say $\pi/5$? You need a two-pole filter, why?

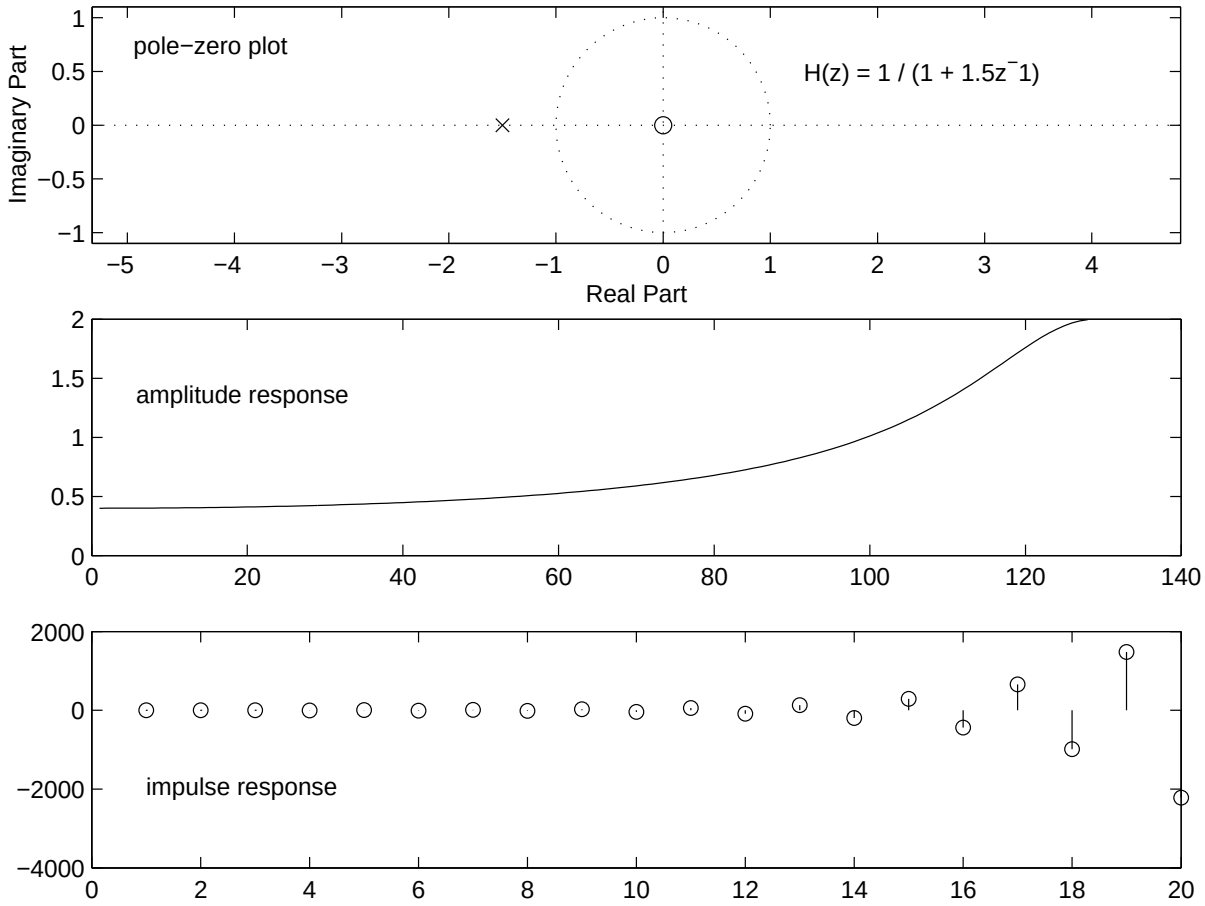


Figure 47: This is an unstable filter. The pole is outside the unit circle. The absolute values of the impulse response increases with the index n.

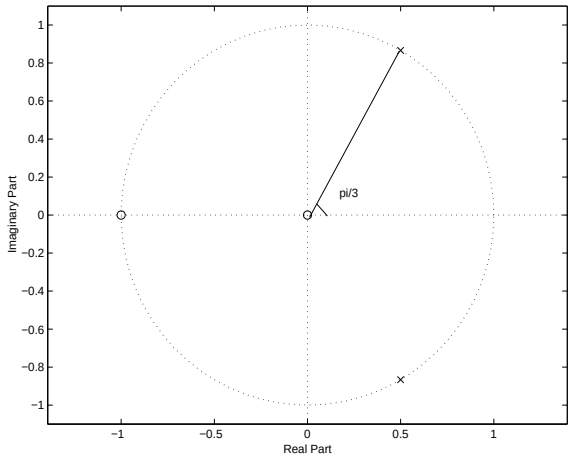


Figure 48: The pole-zero plot for an IIR filter. There are two poles (located at $\pm\pi/3$) on the unit circle. The impulse response of this filter is a discrete cosine signal with a frequency of $\pi/3$.

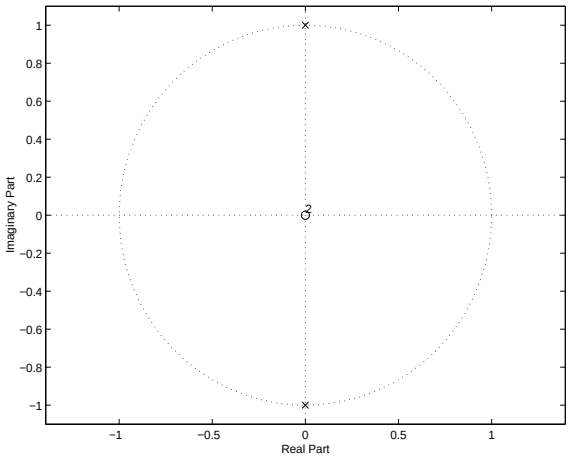


Figure 49: The pole-zero plot of an IIR filter There are two poles (located at $\pm\pi/2$) on the unit circle. The impulse response of this filter is a discrete cosine signal with a frequency of $\pi/2$.

circle. The impulse response of this filter is a discrete cosine signal with a frequency of $\pi/3$. As another example, the impulse response of an IIR filter given by

$$y[n] = -y[n-2] + x[n]$$

is

$$h[n] = \cos(\frac{\pi}{2}n)u[n]$$

The pole-zero plot of its system function is shown in Figure 49.

10.2.4 Inverse filters

Deconvolution techniques have many applications. For example, in signal transmission the distortion due to an imperfect channel can be modelled as

$$y[n] = x[n] * h[n] + r[n]$$

where $x[n]$, $h[n]$, $r[n]$ and $y[n]$ represent the original signal, the impulse response of the channel, the additive noise and the received signal, respectively. If the noise power is relatively low and can be omitted, then we would like to recover the original signal from $y[n]$. More specifically, we want to design a filter to process the received signal such that the output is as close to the original as possible. Such a filter, denoted by $h_1[n]$, is a deconvolution filter

$$v[n] = h_1[n] * y[n] = h_1[n] * h[n] * x[n] + h_1[n] * r[n]$$

In an ideal case, we require $v[n] = x[n-n_d]$ where n_d represents a pure delay by n_d samples. If noise is not considered and the delay is assumed zero, then it requires that

$$h_1[n] * h[n] = \delta[n]$$

or in the z-transform domain

$$H_1(z) = \frac{1}{H(z)}$$

The zeros of $H(z)$ become the poles of $H_1(z)$. Depending on the locations of poles (or the zeros of $H(z)$), the inverse filter may not be stable. For example, the channel effect is modelled as

$$y[n] = x[n] - \frac{5}{6}x[n-1] + \frac{1}{6}x[n-2]$$

The system function of the inverse filter is given by

$$H_1(z) = \frac{1}{1-\frac{5}{6}z^{-1}+\frac{1}{6}z^{-2}} = \frac{1}{(1-\frac{1}{2}z^{-1})(1-\frac{1}{3}z^{-1})}$$

This system function has two poles (both inside the unit circle) so that the inverse filter is stable and is given by

$$v[n] = y[n] + \frac{5}{6}v[n-1] - \frac{1}{6}v[n-2]$$

To obtain its impulse response, the ROC must be $|z| > 0.5$. The system function can be expanded to

$$H_1(z) = \frac{3}{1-\frac{1}{2}z^{-1}} - \frac{2}{1-\frac{1}{3}z^{-1}}$$

Therefore the impulse response is given by

$$h_1[n] = 3(\frac{1}{2})^n u[n] - 2(\frac{1}{3})^n u[n]$$

10.3 Self-test exercises

- Design an FIR filter to remove the 50 Hz sinusoidal interference in a data processing system with a sampling frequency of 1 kHz. Plot the magnitude response to verify you result. Answer: the corresponding digital frequency for the sinusoidal signal is $\omega = \pi/10$. $\cos(0.1\pi) = 0.9511$. The filter is given by $y[n] = x[n] - 1.9021x[n-1] + x[n-2]$

- An FIR filter is given by

  $$y[n] = x[n] - 2x[n-1] + 2x[n-2] - x[n-3]$$

  Determine the frequency of a cosine signal that will be "nulled" by this filter. Answer: $x[n] = \cos(n\pi/3)$

- Use the method described in Section 8.5 to determine the impulse response of the filter

  $$y[n] = -y[n-2] + x[n]$$

- An IIR filter is given by

  $$H(z) = \frac{1+0.8z^{-1}}{1-0.9z^{-1}}$$

  Is this filter stable ? Determine its magnitude response $|H(\omega)|^2$. Plot the pole-zero plot of the system function and determine if this is a low pass or a high pass filter based on the locations of poles and zeros. Determine two different impulse responses that correspond to this system function

11 Problem class-2 A review of signal and system representation methods

11.1 Signal representation

- In the time domain:
$$x(n) = x(n) * \delta(n) = \sum x(m)\delta(n-m)$$
- In the Fourier transform domain:
$$X(\omega) = \sum x(n)e^{-j\omega n}$$
- In the z-transform domain:
$$X(z) = \sum x(n)z^{-n}$$

11.2 LTI system representation

- Impulse response: $h(n)$
- In the time domain (*convolution*):
$$y(n) = x(n) * h(n) = \sum x(m)h(n-m)$$
- In the Fourier transform domain :
$$Y(\omega) = X(\omega)H(\omega),\ H(\omega) = \frac{Y(\omega)}{X(\omega)}$$
- $H(\omega)$ is called the *frequency response* of the filter
- In the z-transform domain:
$$Y(z) = X(z)H(z),\ H(z) = \frac{Y(z)}{X(z)}$$
- $H(z)$ is called the *system function.* The z transform together with its ROC uniquely specifies a filter. when performing the inverse z transform, the ROC must be considered.
- The relationship between the Fourier transform and the z-transform is:
$$H(\omega) = H(z)|_{z=e^{j\omega}}$$
- Poles and zeros are related to the frequency response. We can design a filter by properly placing zeros and poles in the z-plane.
- For a stable filter, all poles must be inside the unit circle
- An FIR filter is determined by its zeros and a scaling constant

11.3 Sampling and reconstruction of analog signals

- Sampling is a process to convert an analog signal into a discrete signal with the sampling interval T_s or the sampling frequency f_s.
- The relationship between the analog frequency and the corresponding digital frequency is given by:

$$\omega = \Omega T_s = \Omega / f_s$$

- The minimum sampling frequency (the Nyquist frequency) is:

$$f_s \geq 2 f_{max}$$

- A low pass filter is used to reconstruct the analog signal from the digital form. The cutoff frequency of the low pass filter should satisfy:

$$\Omega_{max} < \Omega_c < (\Omega_s - \Omega_{max})$$

where Ω_{max} is the maximum frequency of the signal, $\Omega_s = 2\pi f_s$ and $\Omega_c = 2\pi f_c$ is cutoff frequency of the filter.

12 FIR filter design

12.1 Design of FIR filters using windows

To design a filter, the desired filter characteristics are usually specified in the frequency domain in terms of the magnitude and phase response. A digital filter can be either an FIR or an IIR filter. The designer's task is then to determine the type of the filter and the filter coefficients. In this section, we discuss using the window method to determine the coefficients of an FIR filter of which the frequency response is an approximation to the desired frequency response. Other methods, such as frequency sampling and optimal FIR filter design, will not be discussed.

The task of designing an FIR filter can be described as: given the desired frequency response, denoted by $H_d(\omega)$, determine the filter coefficients $h[n]$ such that the resulting frequency response, denoted by $H(\omega)$ is a good approximation of $H_d(\omega)$.

Usually, $H_d(\omega)$ is defined as piece-wise constant with discontinuity at the boundaries between bands. As a result, the corresponding unit sample response is non-causal and is of infinite length. For example, an ideal low pass filter is given by

$$|H_d(\omega)| = \begin{cases} 1, & 0 \le \omega \le \omega_c \\ 0, & \omega_c < \omega \le \pi \end{cases}$$

We assume a linear phase response ($\angle H_d(\omega) = -m\omega$). To simplify the analysis, this phase characteristic is neglected temporarily by assuming zero delay ($m = 0$). The unit sample response is determined by

$$\begin{aligned} h_d[n] = & \ \tfrac{1}{2\pi}\int_{-\pi}^{\pi} H_d(\omega)e^{j\omega n}d\omega \\ = & \ \tfrac{1}{2\pi}\int_{-\omega_c}^{\omega_c} H(\omega)e^{j\omega n}d\omega \\ = & \ \tfrac{1}{2\pi}\int_{-\omega_c}^{\omega_c} e^{j\omega n}d\omega \\ = & \ \tfrac{1}{\pi n}\sin(n\omega_c), \quad n = 0, \pm 1, \pm 2, ... \end{aligned}$$

Therefore, the unit sample response of an ideal low pass filter is of infinite length and is non-causal. To make it an FIR filter, we can truncate it to a length of $2N+1$ by using

$$h[n] = \begin{cases} h_d[n], & |n| \le N \\ 0, & |n| > N \end{cases}$$

There are two implications for this process. The first implication is that truncation in this way can be regarded as multiplication of $h_d[n]$ by a window function, denoted by $w[n]$

$$h[n] = h_d[n]w[n]$$

where

$$w[n] = \begin{cases} 1, & |n| \le N \\ 0, & |n| > N \end{cases}$$

The second implication is that the resulting frequency response is no longer the same as the ideal one. It is actually a convolution of $H_d(\omega)$ with $W(\omega)$ which is the Fourier transform of the window function $w[n]$. This is a direct result of the property of the Fourier transform which states that the multiplication of two sequences in the time domain corresponds to the convolution of their respective Fourier transform in the frequency domain. Figure 50 shows the Fourier transform of the window function as well as the result of the convolution.

We can see from Figure 50 that the "distortions" in the resulting frequency response are

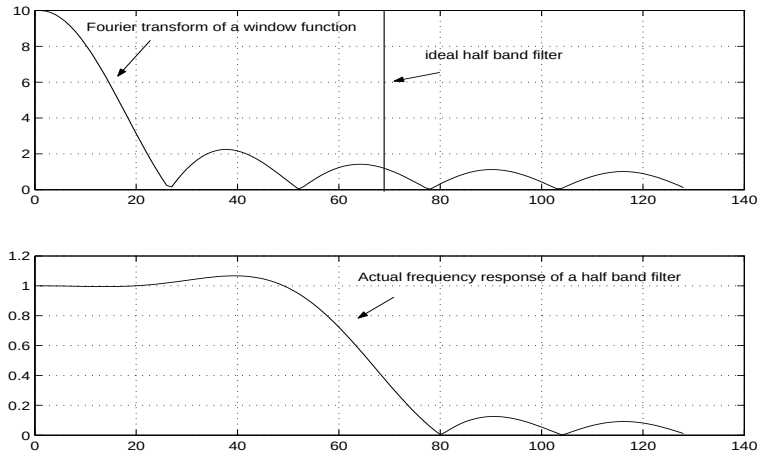


Figure 50: The actual frequency response is a result of the convolution of the Fourier transform of the window function and the ideal frequency response.

- a smooth transition between the pass band and the stop band, which is largely due to the width of the main lobe of the Fourier transform of the window function, and
- the ripples in both pass band and stop band which are due to the main lobe and the side lobes.

Since our task is to design a filter such that its frequency response is a good approximation to the desired one, we must optimize the window function. In an ideal case, the Fourier transform of the window function should be an impulse. This is not possible, because it requires that the window be of infinite length. Generally, a longer window leads to a longer unit sample response and a better approximation to the desired frequency response. Figure 51 shows the Fourier transform of a rectangular window of different lengths as well as the resulting frequency response of the half band filter. We can see from this figure that as the length of the window increases, the width of main lobe and the magnitude of the side lobes decrease. The resulting frequency response better approximates the ideal one.

A practical requirement is that we should choose a window function such that its Fourier transform has a very narrow main lobe and a very small side lobe magnitude. However, these are usually conflicting requirements for a fixed window length. For example, by tapering the window smoothly to zero at each end, the magnitude of the side lobes becomes smaller at the cost of a wider main lobe. There is a trade-off between the width of the main lobe and magnitude of the side lobes. Figure 52 shows a number of window functions and their Fourier transforms.

So what is a good window function? We can see from Figure 52 that if a sharp transition between the pass band and the stop band is desired, then for a given filter length the rectangular window is most suitable as its main lobe is the narrowest among other window functions. However, it should be noted that the magnitude of its side lobes is also the largest.

12.2 Low pass filter design

We have shown that the unit sample response of an ideal low pass filter with a cut off frequency of ω_c is given by:

$$h_d[n] = \frac{1}{\pi n}\sin(n\omega_c), \quad n = 0, \pm 1, \pm 2, ...$$

It is truncated by using a window function and is made causal by shifting to the right hand side.

For example, assume the cutoff frequency of an ideal low pass filter is $\pi/4$ and a rectangular window (21 coefficients) is used. Then the filter coefficients are given in the following table

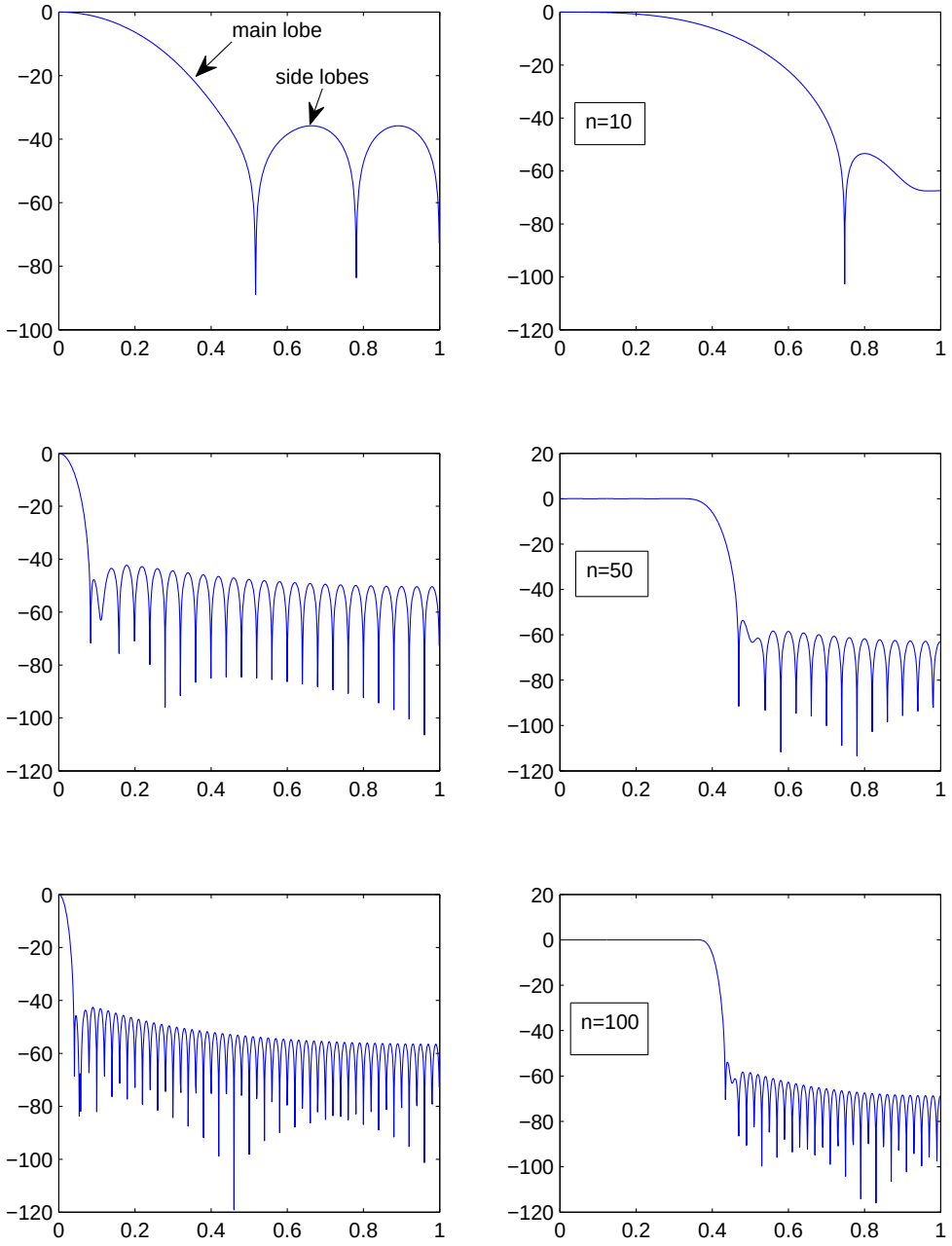


Figure 51: Left panel: the Fourier transform of Hamming window with N=10, 50 and 100. Right panel: the frequency response of an FIR filter (cutoff frequency $\omega_c = 0.4\pi$) designed using the Hamming window with N=10, 50, 100.

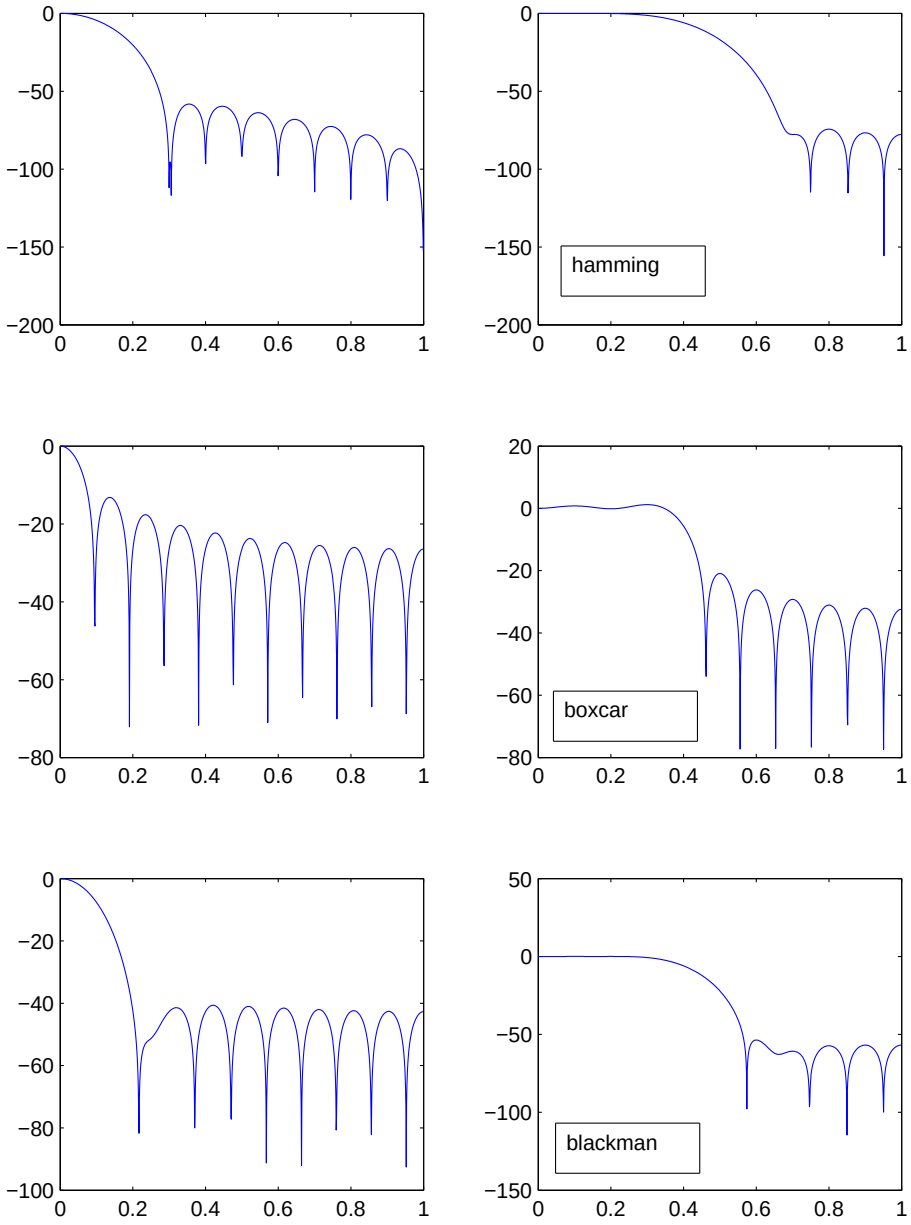


Figure 52: This figure shows the different characteristics of window functions and the FIR filters designed using thes window functions. Left panel: the Fourier transform of three window functions (top to bottom) Hamming, boxcar and Blackman. Right penal: the amplitute response of the FIR filters designed using these window functions.

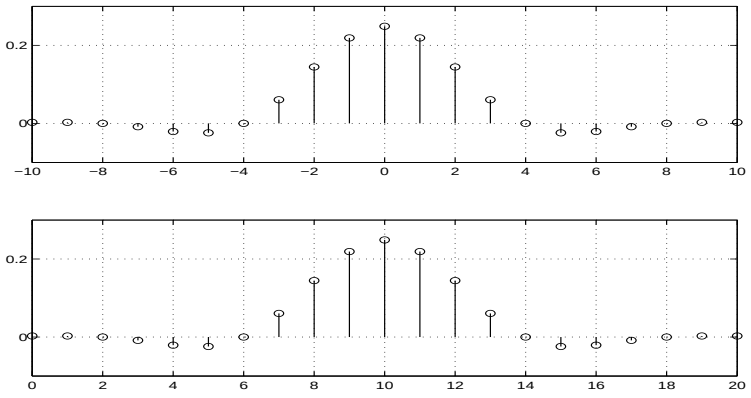


Figure 53: The unit sample response before and after shifting

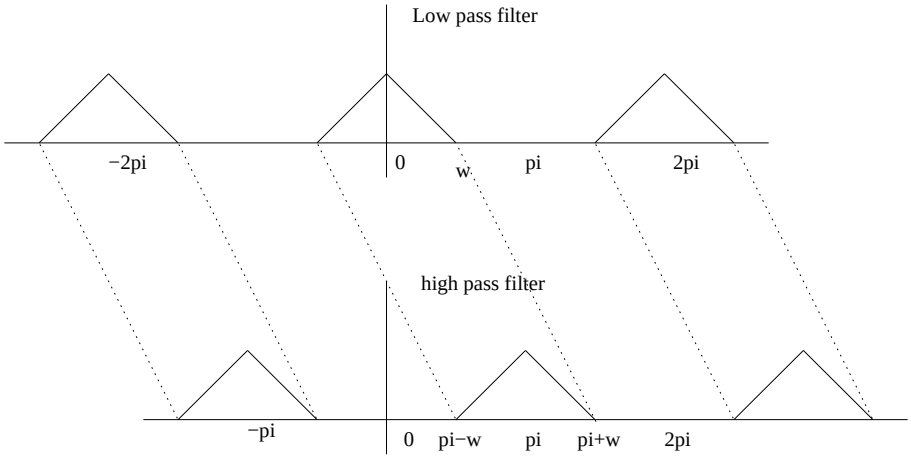


Figure 54: Magnitude relationship between a low pass filter and a high pass filter.

n	0	±1	±2	±3	±4	±5	±6	±7	±8	±9	±10
h[n]	.25	.225	.159	.075	0	-.045	-.053	-.032	0	.025	.032

To make this filter causal, the unit sample response is shifted to the right side by 10 samples:

$$h(n) = \frac{1}{\pi(n-10)} \sin((n-10)\omega_c), \quad 0 \leq n \leq 20, ...$$

The shifting introduces delay in the system. Figure 53 shows the unit sample response before and after shifting.

12.3 High pass filter

A high pass filter can be easily designed using a low pass filter. Recall that if the frequency response of a low pass filter is shifted by π, then a high pass filter with cutoff frequency of $\pi - \omega_c$ is obtained. This is illustrated in Figure 54.

Therefore, an ideal high pass filter can be specified in terms of a low pass filter

$$H_H(\omega) = H_L(\omega - \pi)$$

Here we use the subscripts H and L to represent the high pass and low pass filter, respectively. According to the definition, we have

$$H_H(\omega) = \sum_{n=-\infty}^{\infty} h_H(n) e^{-j\omega n}$$

and

$$H_L(\omega) = \sum_{n=-\infty}^{\infty} h_L(n) e^{-j\omega n}$$

then

$$\begin{aligned} H_H(\omega) = & \quad H_L(\omega - \pi) \\ = & \quad \sum_{n=-\infty}^{\infty} h_L(n) e^{-j(\omega-\pi)n} \\ = & \quad \sum_{n=-\infty}^{\infty} h_L(n) e^{j\pi n} e^{-j\omega n} \end{aligned}$$

Therefore, we have established the relationship between the low pass and high pass filters in the time domain

$$h_H(n) = h_L(n) e^{jn\pi} = (-1)^n h_L(n)$$

To design a high pass filter with a cut off frequency of 0.7π, we can design a low pass filter with a cut off frequency of $\pi - 0.7\pi = 0.3\pi$. Then use the the above relationship to obtain the unit sample response.

12.4 Band pass and band stop filters design

The band pass and band stop filters can be designed using a similar method. The design methods are summarized in the following

$$\begin{aligned} \text{band pass filter} \quad & h_{BP}[n] = 2\cos(n\omega_0) h_L[n], \\ & \omega_u - \omega_l = 2\omega_c \\ & \tfrac{\omega_u + \omega_l}{2} = \omega_0 \\ \text{band stop filter} \quad & h_{BS}[0] = 1 - h_{BP}[0] \\ & h_{BS}[n] = -h_{BP}[n],\ n \neq 0 \\ & \omega_u - \omega_l = 2\omega_c \\ & \tfrac{\omega_u + \omega_l}{2} = \omega_0 \end{aligned}$$

Figure 55 shows the magnitude responses for both type of filters. To design a band pass filter based on the above method is simple. For example, a 11th order digital filter is required to meet the following specifications: sampling frequency 10 kHz, pass band 1kHz to 2 kHz. We need to convert the specification into the discrete radian frequency by using

$$\omega = \Omega / f_s = 2\pi f / f_s$$

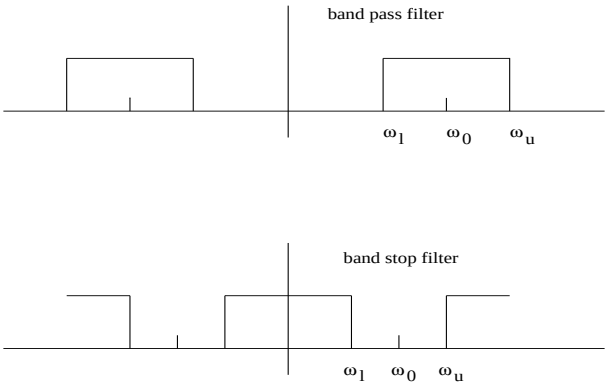


Figure 55: The magnitude response. Top : band pass filter, bottom: band stop filter

Therefore, we have

$$\omega_u = 2\pi \times 2/10 = 0.4\pi$$

and

$$\omega_l = 2\pi \times 1/10 = 0.2\pi$$

The corresponding cut off frequency of the low pass filter is

$$\omega_c = \frac{\omega_u - \omega_l}{2} = 0.1\pi$$

and the center frequency of the pass band is

$$\omega_0 = \frac{\omega_u + \omega_l}{2} = 0.3\pi$$

Therefore, the unit sample response of the band pass filter is given by

$$h_{BP}[n] = 2\cos(n\omega_0)h_L[n] = 2\cos(0.3\pi n)h_L[n]$$

where

$$h_L[n] = \frac{1}{\pi n}\sin(n\omega_c) = \frac{1}{\pi n}\sin(0.1\pi n), \quad n = 0, \pm 1, \pm 2, ..., \pm 5$$

To make this filter a causal filter, the unit sample response must be shifted to the right hand side by 5 samples

$$h[n] = h_{BP}[n-5]$$

12.5 Self-test exercises

- You can use a Matlab command `fir1` to design FIR filters. Type `help fir1` to see how to use it. Design an 11th order low pass filter with cut off frequency of 0.5π by using the window method described in section 12.2 and by using the command `fir1`. Compare the unit sample response and the frequency response for both methods.

- How do you convert the above low pass filter into a high pass filter with a cutoff frequency of 0.5π?

- A 7th order digital filter is required to meet the following specifications: sampling frequency 40 kHz, stop band from 4kHz to 8 kHz. Design a causal digital band stop filter to meet these requirements. Plot the magnitude response to verify your design. (hint: refer to the example in section 12.4)

13 IIR filter design

This section is divided into three parts: analog prototype filters, IIR filter design using the bi-linear transform, and IIR filter design using Matlab.

13.1 Analog Prototype Filters

13.1.1 Laplace transform

The one-sided Laplace transform of a continuous function is defined as:

$$X(s) = L[x(t)] = \int_0^\infty x(t)e^{-st}dt$$

where s is a complex variable that is independent of t. The transfer function for a continuous system is defined as:

$$H(s) = \frac{Y(s)}{X(s)}$$

The frequency response is given by substituting $s = j\Omega$ into the transfer function.

$$H(\Omega) = H(s)\Big|_{s=j\Omega}$$

13.1.2 Butterworth low pass prototype design

The frequency response of a Butterworth low pass filter is:

$$|H(\Omega)|^2 = \frac{1}{1+\varepsilon^2\left(\Omega/\Omega_c\right)^{2N}}$$

where ε is the parameter that controls the ripple (usually ε=1), Ω_c is the cutoff frequency and N is the order of the filter. The Butterworth filter is maximally-flat and it passes through -3dB at the cut-off frequency.

When the cut off frequency is 1, the filter is called the prototype (since filters with different cut off frequency can be easily obtained from the prototype by a scaling in the frequency domain).

$$|H(\Omega)|^2 = \frac{1}{1+\Omega^{2N}}$$

The only parameter is the order of the filter. It is usually determined by the specification of the signal processing system, ie, the attenuation at a given frequency Ω_a . For example, if an attenuation of M dB is required, then

$$M = 10\log_{10}|H(\Omega_a)|^2 = 10\log_{10}\frac{1}{1+\Omega_a^{2N}}$$

and

$$N = \frac{\log_{10}\left(10^{-\frac{M}{10}}-1\right)}{2\log_{10}\Omega_a}$$

13.1.3 Chebyshev low pass prototype filter design

A class of filters are called equal-ripple filters. The Chebyshev filter has equal-ripple in the pass band, the inverse Chebyshev filter has equal-ripple in the stop band. By allowing ripples in the filter frequency response, a sharper transition from pass band to stop band than that of the Butterworth filter can be achieved.

The Chebyshev low pass prototype is defined as:

$$|H(\Omega)|^2 = \frac{1}{1+\varepsilon^2 C_N^2(\Omega)}$$

where $C_N(\Omega)$ is the N-th order Chebyshev polynomial It can be defined by the recursion function

$$C_{N+1}(\Omega) = 2\Omega C_N(\Omega) - C_{N-1}(\Omega)$$

where

$$C_0(\Omega) = 1$$

and

$$C_1(\Omega) = \Omega$$

The prototype ripple bandwidth is from $0 \leq \Omega \leq 1$. In $0 \leq \Omega \leq 1$, there are N maximum and minimum points. The ripple amplitude in dB is given by

$$r_{dB} = -10\log_{10}(\frac{1}{1+\varepsilon^2}) = 10\log_{10}(1+\varepsilon^2)$$

The -3dB cutoff frequency is greater than 1.

To design a Chebyshev filter, two parameters need to be determined. The desired ripple is set by ε and the order N is found by knowing a desired half-power or -3dB frequency and/or the required stop band characteristics. This is usually done by consulting a filter design handbook (using tables or monograms)

13.1.4 Elliptic low pass prototype design

An elliptic filter has an equal-ripple response in both its pass band and its stop band. This leads to even sharper cut off characteristics than the Chebyshev filters. An elliptic filter is defined as:

$$|H(\Omega)|^2 = \frac{1}{1+\varepsilon^2 Z_N^2(\Omega)}$$

where $Z_N(\Omega)$ is a Chebyshev rational function determined from the specified ripple characteristics. An elliptic filter has three parameters (which can be defined in different ways). One set of parameters are the order N, the acceptable pass band ripple and the acceptable stop band gain.

The magnitude responses for 4th-order digital IIR filters (Butterworth, Chebyshev-I and elliptic filter) are shown in Figure 56.

13.2 IIR Filter Design Using Bilinear Transform

Because the design methods for analog filters are well established, it is useful to study how to covert an analog filter into a digital filter. One of the most effective way is by means of a *bilinear transform*:

$$s = \frac{z-1}{z+1}$$

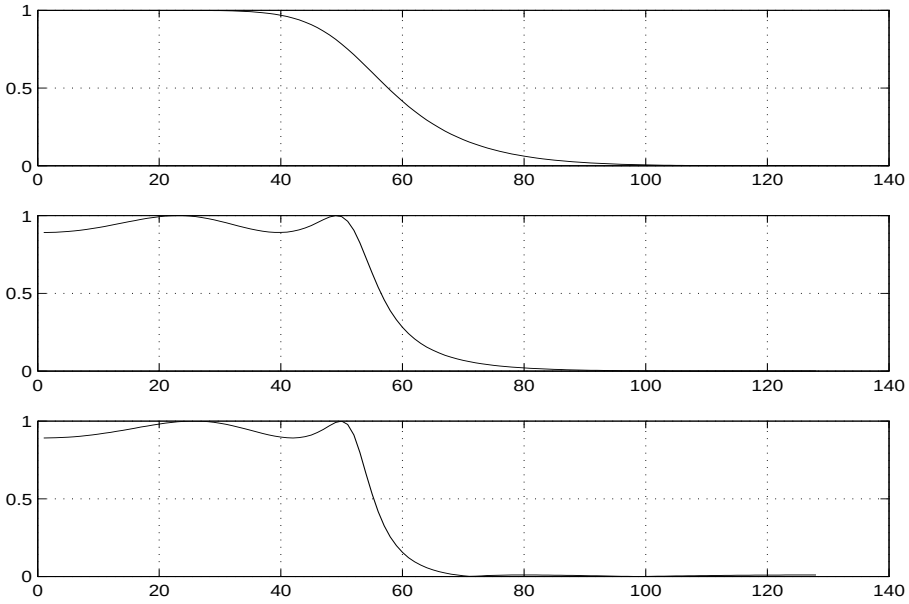


Figure 56: Magnitude response of Butterworth filter (top), Chebyshev-I filter (middle) and elliptic filter (bottom)

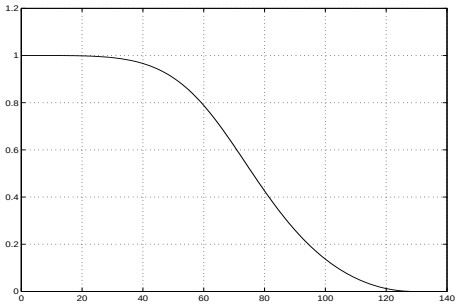


Figure 57: The magnitude response of a second order digital Butterworth filter designed using bilinear transform.

The transfer function of an analog filter $H(s)$ is converted to that of a digital filter with the transfer function

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

For example, the transfer function of the second order Butterworth prototype filter and its corresponding digital filter are:

$$H(s) = \frac{1}{1+\sqrt{2}s+s^2}$$

and

$$H(z) = \frac{1}{1+\sqrt{2}\frac{z-1}{z+1}+\left(\frac{z-1}{z+1}\right)^2} = 0.293\frac{1+2z^{-1}+z^{-2}}{1+0.172z^{-2}}$$

The frequency response of this digital filter is shown in Figure 57.

It can be shown that using the bilinear transform, the right/left half of the s-plane is mapped to the exterior/interior of the unit circle of the z-plane, and the entire imaginary axis of the s-plane is mapped

to the unit circle in the z-plane. Therefore, a stable analog filter (the poles of which must be in the left-half of the s-plane) is converted to a stable digital filter (the poles of which must be inside the unit circle).

How does the system function of $H(s)$ relate to that of $H(z)$? The frequency responses of the analog and the digital filter can be calculated by

$$H(\Omega) = H(s)|_{s=j\Omega}$$

and

$$H(\omega) = H(z)|_{z=e^{j\omega}}$$

Therefore the bilinear transform corresponds to the following relationship in frequency:

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

The above equation is equivalent to

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

One can easily see that the range of the discrete radian frequency [0, π] is mapped to the analog frequency range $[0, \infty)$. Therefore, the bilinear transform corresponds to a nonlinear compression of the analog frequency range to the discrete radian frequency range.

In general, the design of a digital IIR filter using the bilinear transform involves the following steps

1. Determine the filter specification (such as the cutoff frequency) according to a problem
2. Prewarp the discrete radian frequencies using the above equation. This step converts the discrete radian frequencies into the analog frequencies.
3. Design an analog filter based on the analog frequencies specified in step 2.
4. Use the bilinear transform to determined the transfer function of the digital filter.

13.3 IIR filter design using Matlab

13.3.1 Filter specifications and types

You need a filter for a specific application. There are different applications, so there are different ways to specify a filter. A loose specification might ask for a filter to remove the noise above 30 Hz with a 100 Hz sampling frequency. A more rigorous specification might call for a specific amount of pass band ripple, stop band attenuation, or transition width. A very precise specification could ask to achieve the performance goals with the minimum filter order, or it could call for an arbitrary magnitude shape, or it might require an FIR filter.

In the following, we will specify a filter by four parameters: the pass band ripple, the stop band attenuation, the pass band edge frequency and the stop band edge frequency. This is illustrated in Figure 58. Note that although in this figure, a low pass filter is used, these four parameters are also applicable to high pass filters and band pass filters where the pass band and stop band frequencies are two-element vectors.

There are three types of IIR filters. The Butterworth filter has a maximally flat frequency response which passes through -3 dB at the cut-off frequency. For the same filter order, the Chebyshev filter has a sharper transition between the pass band and the stop band than the Butterworth filter. The Chebyshev type-I filter has ripples in the pass band, while type-II has ripples in the stop band. For the same filter order, the elliptic filter has even sharper transition than the Chebyshev filter. It has ripples

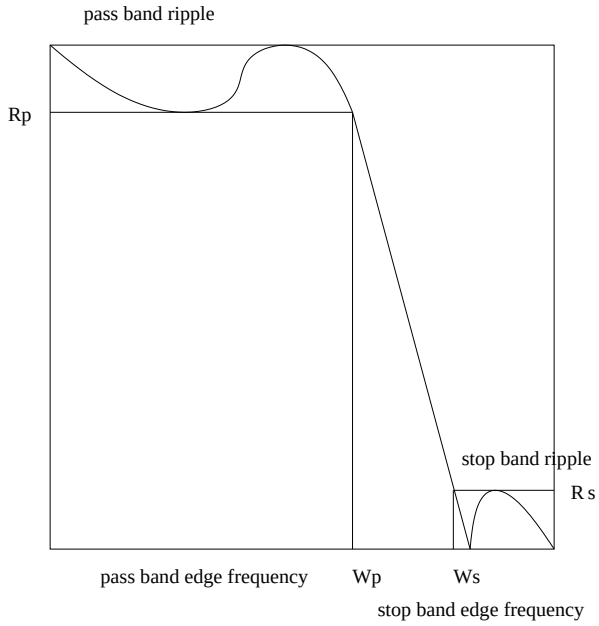


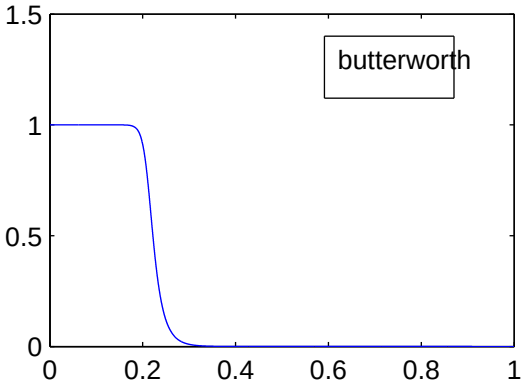
Figure 58: The specification of an IIR filter.

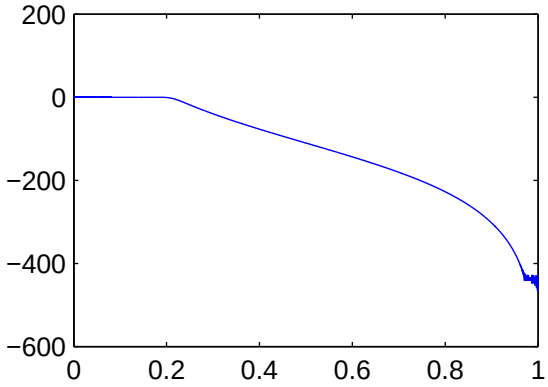
at both pass band and stop band. So there is a trade-off between the computational complexity (order of the filter) and the frequency response.

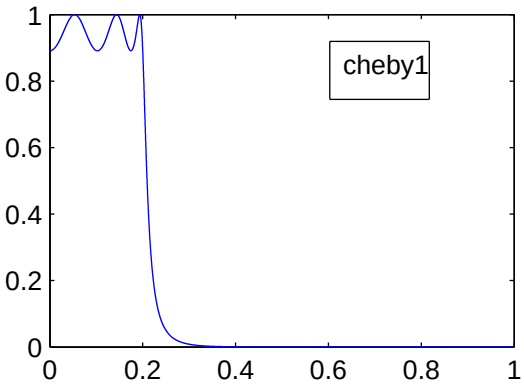
Here is an example[14] of the 4 types of IIR filters with $\omega_p = 0.2$, $\omega_s = 0.3$, $R_p = 1$ and $R_s = 40$. The amplitude responses are roughly the same, when we ignor the effects of ripples. However, the order of these filters are not the same, Butterworth 12, Chebyshev I and II 6, and elliptic 4. In other words, to have roughly the same amplitude response, the elliptic filter is most computational efficient among the filters. This is at the costs of having ripples in both the pass band and stop band. The butterorth filter, on the other hand, is most computationally demanding. It has the advantage of being max-flat, i.e., no ripples. The computational requirement for the Chebyshev typy I and II filters is between the elliptic and butterworth filters. What will happen if we use the same order for the filter types? Fig. 61 shows the amplitude repsonses of using N = 4. We can clearly see that the elliptic filter has the fastest

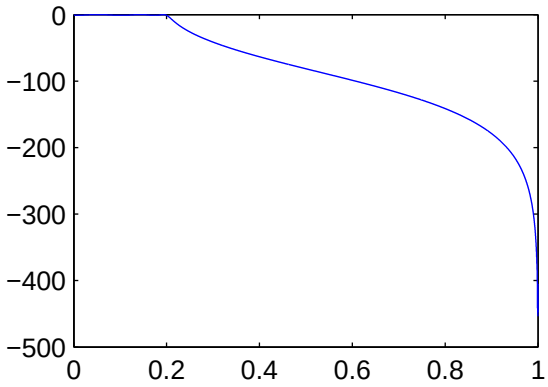
[14]Here is the matlab code.

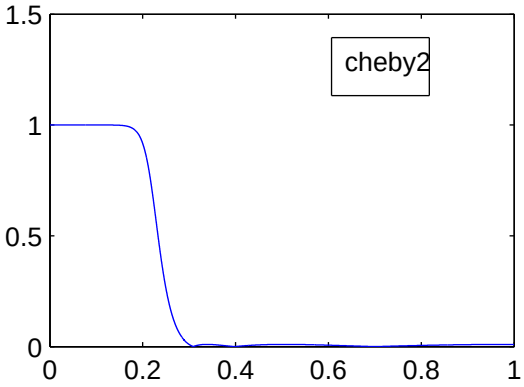
```
wp=0.2; ws=0.3; rp=1; rs=40;
[bN,bw] = buttord(wp,ws,rp,rs);
[c1N,c1w] = cheb1ord(wp,ws,rp,rs);
[c2N,c2w] = cheb2ord(wp,ws,rp,rs);
[eN,ew] = ellipord(wp,ws,rp,rs);
[bb,ba]=butter(bN,bw);
[c1b,c1a]=cheby1(c1N,rp,c1w);
[c2b,c2a]=cheby2(c2N,rs,c2w);
[eb,ea]=ellip(eN,rp,rs,ew);
[bH,W]=freqz(bb,ba,10000);
[c1H,W]=freqz(c1b,c1a,10000);
[c2H,W]=freqz(c2b,c2a,10000);
[eH,W]=freqz(eb,ea,10000);
subplot(421);plot(W/pi,abs(bH))
subplot(422);plot(W/pi,20*log10(abs(bH)))
subplot(423);plot(W/pi,abs(c1H))
subplot(424);plot(W/pi,20*log10(abs(c1H)))
subplot(425);plot(W/pi,abs(c2H))
subplot(426);plot(W/pi,20*log10(abs(c2H)))
subplot(427);plot(W/pi,abs(eH))
subplot(428);plot(W/pi,20*log10(abs(eH)))
```

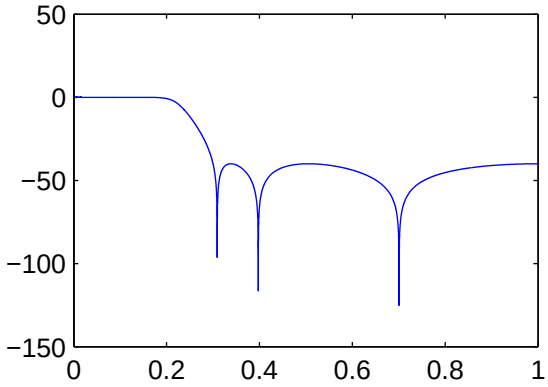

butterworth
1.5
1
0.5
0
0
0.2
0.4
0.6
0.8
1

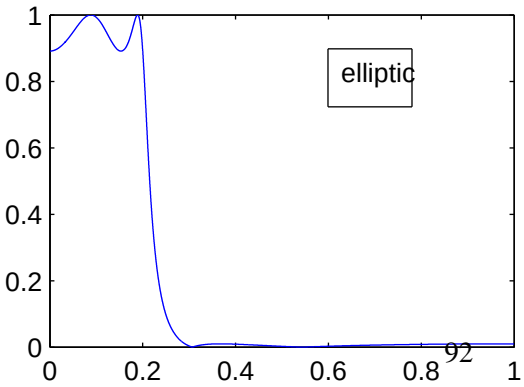

200
0
−200
−400
−600
0
0.2
0.4
0.6
0.8
1

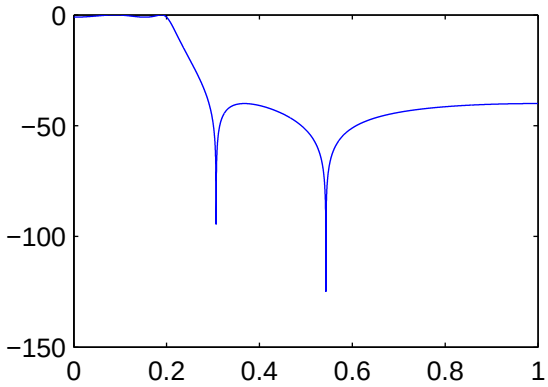

cheby1
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1

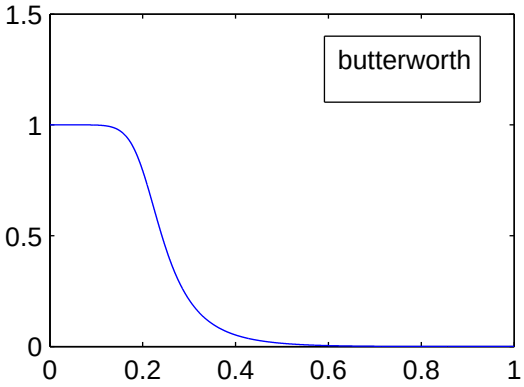

0
−100
−200
−300
−400
−500
0
0.2
0.4
0.6
0.8
1

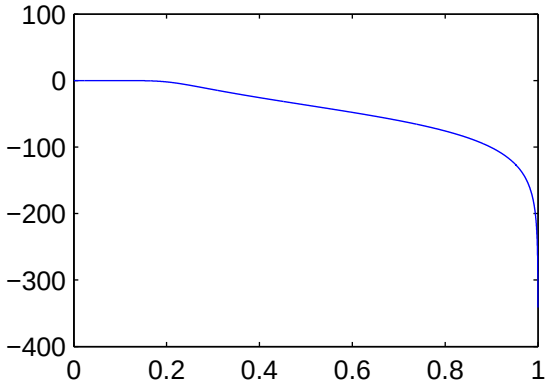

cheby2
1.5
1
0.5
0
0
0.2
0.4
0.6
0.8
1

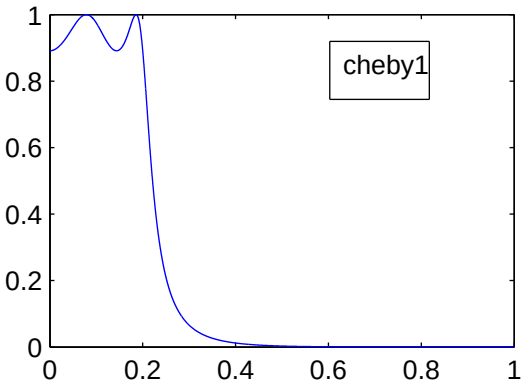

50
0
−50
−100
−150
0
0.2
0.4
0.6
0.8
1

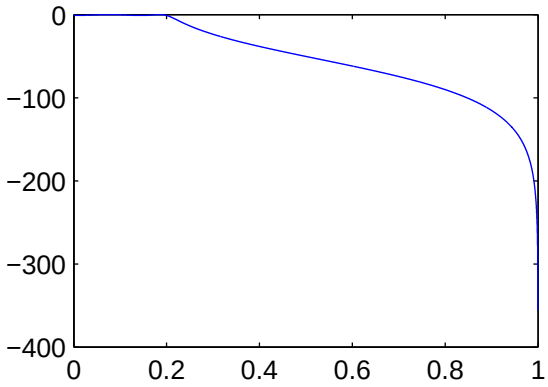

elliptic
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1

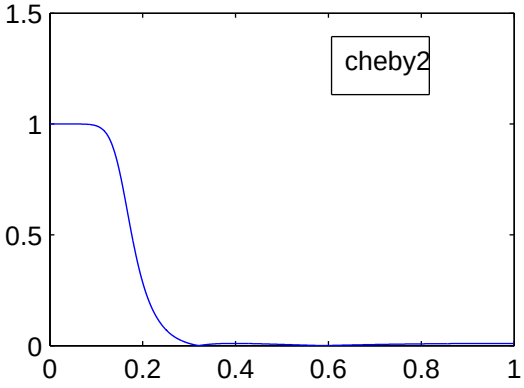

0
−50
−100
−150
0
0.2
0.4
0.6
0.8
1

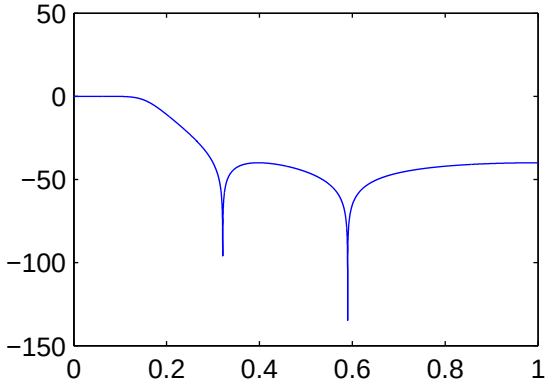

butterworth
1.5
1
0.5
0
0
0.2
0.4
0.6
0.8
1

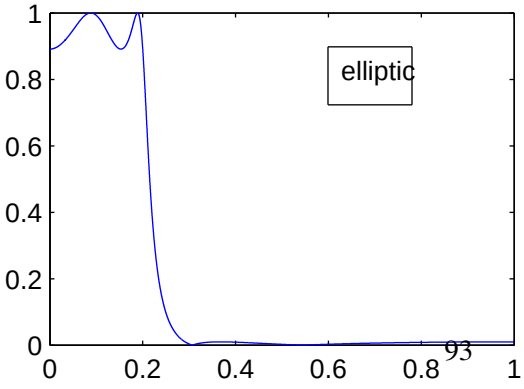

100
0
−100
−200
−300
−400
0
0.2
0.4
0.6
0.8
1

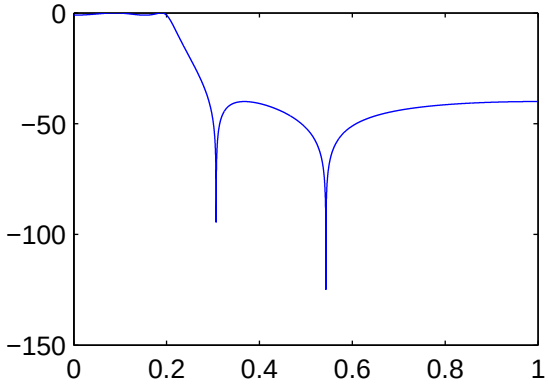

cheby1
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1


0
−100
−200
−300
−400
0
0.2
0.4
0.6
0.8
1


cheby2
1.5
1
0.5
0
0
0.2
0.4
0.6
0.8
1


50
0
−50
−100
−150
0
0.2
0.4
0.6
0.8
1


elliptic
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1


0
−50
−100
−150
0
0.2
0.4
0.6
0.8
1

change from pass band to stop band, while the Butterworth has the slowest changes.

13.3.2 Design procedures

Once the filter is specified, the design of the filter involves

1. Choose the type of the filter that best fits the application.
2. Determine the order of the filter by using the following Matlab functions. Note that the frequency specifications must be ***normalized discrete radian frequency*** in the range [0, 1], which corresponds to [0,π]. If the filter is a band pass or band stop filter, then the frequency specifications are two-element vectors.

Butterworth filter	$[N, \omega_n] = buttord(\omega_p, \omega_s, R_p, R_s)$
Chebyshev I filter	$[N, \omega_n] = cheb1ord(\omega_p, \omega_s, R_p, R_s)$
Chebyshev II filter	$[N, \omega_n] = cheb2ord(\omega_p, \omega_s, R_p, R_s)$
Elliptic filter	$[N, \omega_n] = ellipord(\omega_p, \omega_s, R_p, R_s)$

3. Determine the filter coefficients. In Matlab, the filter coefficients are stored in two vectors called *b* and *a*.

$$H(z) = \frac{B(z)}{A(z)} = \frac{b(1) + b(2)z^{-1} + ... + b(n+1)z^{-n}}{1 + a(2)z^{-1} + ... + a(n+1)z^{-n}}$$

   To design a low pass filter, you use:

Butterworth filter	$[b, a] = butter(N, \omega_n)$
Chebyshev I filter	$[b, a] = cheby1(N, R_p, \omega_n)$
Chebyshev II filter	$[b, a] = cheby2(N, R_s, \omega_n)$
Elliptic filter	$[b, a] = ellip(N, R_p, R_s\omega_n)$

4. Test your design. You may plot the frequency response of the filter to see if it matches the specification. A more important task is to see the quantization effect of the filter coefficients.

13.3.3 Design examples

Example 1

A second order IIR digital filter is required to implement the same processing function of an analog high pass filter with a cut off frequency of 2 rad/sec. Assume the sampling interval T = 0.2 second and a pass band ripple of 3 dB.

To design this digital filter, we convert the filter specification into the normalized discrete radian frequency by using

$$\omega_c = \frac{\Omega_c T}{\pi} = 0.1247$$

The filter coefficients are then calculated by using

$$[b, a] = cheby1(2, 3, 0.1247, 'high');$$

And the system function is given by

$$H(z) = \frac{0.5697 - 1.1394z^{-1} + 0.5687z^{-2}}{1 - 1.516z^{-1} + 0.7028z^{-2}}$$

Example 2

The signal is sampled at 1000 samples/sec. Design a 10th-order band pass digital Butterworth filter with a pass band from 100 to 200 Hz. As in example 1, we first convert the filter specification to normalized discrete radian frequency by using

$$\omega_c = \frac{\Omega_c T_s}{\pi} = \frac{2\pi f_c}{\pi f_s} = \frac{2 f_c}{f_s}$$

The filter coefficients are then calculated by using

$$[b, a] = butter(5, [100, 200]/500);$$

Note that in the above Matlab function [b, a] = butter(n, w), when the cut off frequency is a two-element vector, it will produce a filter with an order of 2n.

Example 3

Design a Chebyshev type-I filter with the following specifications

- 1 dB ripple in the range 600-900 Hz
- the sampling frequency is 3000 Hz
- Max. gain is -40 dB in the frequency ranges $0 \leq f \leq 200$ Hz and $f \geq 1300$ Hz

We can see that this is a band pass filter. We need to determine the filter specification first

$w_p = [600, 900]/1500$
$w_s = [200, 1300]/1500$
$R_p = 1$
$R_s = 40$; %not -40
$[n, w] = cheb1ord(w_p, w_s, R_p, R_s)$

The filter coefficients are obtained by using

$$[b, a] = cheby1(n, R_p, w);$$

The amplitude response[15] is plotted in both linear scale (left) and log-scale (dB). Note that the unit of the horizontal axis is now coverted to Hertz.

Example 4

Design a max-flat low pass filter with the following specification

- sampling frequency is 20 kHz

[15] Matlab code is here

```
wp = [600 900] / 1500;
ws = [200 1300] / 1500;
rp = 1;
rs = 40;
[n, w] = cheb1ord(wp, ws, rp, rs);
[b, a] = cheby1(n, rp, w);
subplot(121); plot(1500*W/pi, abs(H));
subplot(122); plot(1500*W/pi, 20*log10(abs(H)))
```

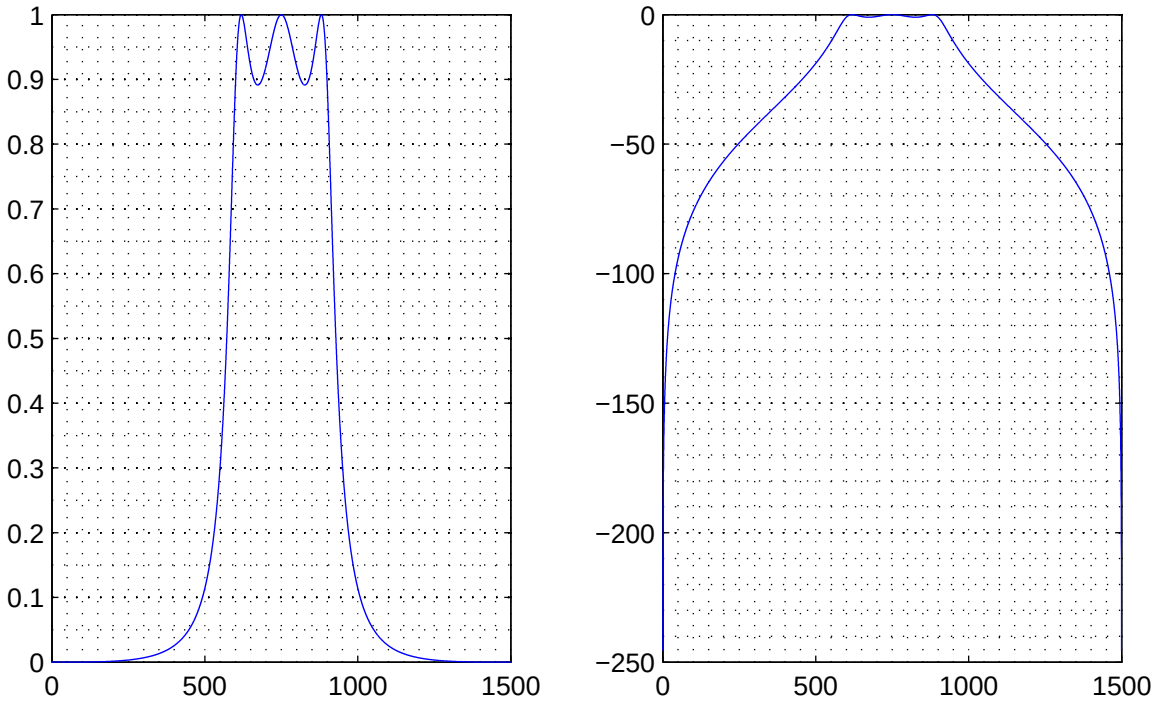


Figure 61: Amplitude response of the Chebyshev type I filter in linear scale and log scale (dB).

- -3 dB gain at cut off frequency 2 kHz
- max. gain of -10 dB at frequency 4 kHz

According to the filter requirements, we use a Butterworth filter and follow exactly the same procedure as example 3 to design the filter. This is shown in the following

$w_p = 2/10;$
$w_s = 4/10;$
$R_p = 3;$
$R_s = 10;$ %not -10
$[n, w] = buttord(w_p, w_s, R_p, R_s);$
$[b, a] = butter(n, w);$

13.4 Self-test exercises

- Go through all filter design examples in this section.
- For a sampling frequency of 400 Hz, what is the minimum order for an elliptic filter that you can use to separate the 80 Hz cosine signal from its mixture with a 50 Hz cosine signal.

14 Applications of DSP[16]

14.1 Audio signal processing

14.1.1 Reverberation effects

14.1.2 Equalizers

14.1.3 Digital FM stereo generation

14.2 Number representations in digital signal processors

14.2.1 Fix point representation

The simplest representation is to use a format: sign+value. In this section, this format is referred to as the signed-binary format. For example, the number (-10) can be represented as: 10001010, where the MSB is the sign bit and the rest bits represent the value. In this case, 1 is used to represent the negative sign and 0 is used to represent the positive sign.

Fix point DSP devices use a format called the two's complement, where a positive number is represented in exactly the same as the above example. The negative number, however, is represented in a different format. The procedure to convert a negative number from the signed-binary format to the two's complement format involves three simple steps:

- change the sign bit to zero,
- invert every bit, and
- add 1. An example: -10 = 10001010, its two's complement representation is 11110110.

One advantage of two's complement format is that one adder is required for both positive and negative numbers. Another advantage is that if the final result is known to be within the processor's number range, then an intermediate overflow can be ignored as the final correct result will be produced. A 16-bit integer in two's complement format can vary between +32767 and -32768. This is known as the 16.0 format for the Analog Devices DSP processor ADSP 219x/2191. The value of the number is calculated as: $x = (-2^{15})a_{15} + \sum_{i=0}^{14} a_i 2^i$

In DSP, it is also common to represent numbers as fractions. The reason for this is that when multiplying fractions (e.g. 0.99×0.98) the result will be always less than one and there is never an overflow. For a fix point processor this is obvious an advantage. The TMS320c5x processor uses a format call Q15 which is show below:

-2^0	2^{-1}	2^{-2}											2^{-15}
a_0	a_1	a_2											a_{15}

The Q15 format is also known as the 1.15 format for the Analog Devices DSP processor 219x/2191, which supports both formats: 1.15 and 16.0. The user must specify which format is being used by setting a bit in the `MSTAT` register.

Using the Q15 format, the number, $x = (-1)a_0 + \sum_{i=1}^{15} a_i 2^{-i}$, varies from -1 to 0.99996948242188. Addition and subtraction are exactly the same as for binary integers. The multiplications of two Q15 numbers results in a Q30 number which is also a fraction. It has 30 fractional bits, 2 sign bits and no integer bits. To store the result as a Q15 number, a left shift of one bit is performed to eliminate the extract sign bit and the resulting left-most 16 bits are stored. An interesting property of the two's

[16] This section is still being written.

complement format is that for a negative number, if you add a "1" in front of the MSB, the value of the number remains the same. For example, -4 = 1100 (-8+4) = 11100 (-16+8+4) = 111100 (-32+16+8+4).

Let us use a few simple examples to illustrate the addition and mutilication of numbers in two's complement format. To simplify our example, we will use 4.0 format which can represent an integer ranging from -8 (1000) to 7 (0111).

Example 1

3 - 4 = 0011 + 1100 = 1111 = -1

Example 2

In this example, we have to left shift one bit to get rid of the extra sign bit to get the correct result.

5 - 4 = 0101 + 1100 = 10001 = 0001 = 1

Example 3

$3 \times (-4) = 0011 \times 1100 = 1110100 = -12$. The actual calculation is shown below

Example 4

$(-3) \times (-4) = 1101 \times 1100 = 0001100 = 12$

Using a fixed point DSP processor such as ADSP 219x, if both operands are in 1.15 format, the multiplication of these two numbers results in 2.30 format (2 sign bits and 30 fractional bits). The processor can automatically left shift one bit to get rid of the extra sign bit and the result is a 1.31 format which is then truncated to 1.15 format.

14.2.2 Floating point representation

The industry standard floating-point number format is defined by IEEE Standard 754.1985. However, the TI's TMS320C3x and TMS320C4x devices use a special form of floating point representation for internal calculations. The reason for the difference is that using the TI's format the hardware multiplier is easier to design and smaller (therefore cheaper). Conversion between these two formats can be achieved by either using a single instruction in TMS320c4x or using an external conversion ASIC (application specific integrated circuit). In either case the overhead for conversion is not significant when compared to overall algorithm execution time.

The IEEE 754 floating point format is 32 bits long and is shown below:

31	30 23	22 0
s	EXPONENT(E)	FRACTION(F)
	MSB LSB	MSB LSB

The interpretation of the format is as following:

- If $E = 255$ and $F \neq 0$, then $x = NaN$.
- If $E = 255$ and $F = 0$, then $x = (-1)^s \infty$.
- If $0 < E < 255$ and $F \neq 0$, then $x = (-1)^s 2^{E-127}(1.F)$.
- If $E = 0$ and $F \neq 0$, then $x = (-1)^s 2^{-126}(0.M)$.

- If $E = 0$ and $F = 0$, then $x = 0$.

The TMS320 format is also 32 bits long and is shown below:

EXPONENT(E)		s	FRACTION(F)	
31	24	23	22	0

The interpretation is: $x = \{(-2)^s + (.F)\}2^E$

14.3 Quantization effects of filter coefficients

Filter coefficients designed by softwares such as Matlab are usually represented by floating point numbers. To implement an FIR filter in a fix point processor such as TMS320c50, the filter coefficients must be quantized. The simplest way is to multiply them with a scaling factor. This is shown in the following equation:

$$c_q(i) = floor(c(i)2^n)$$

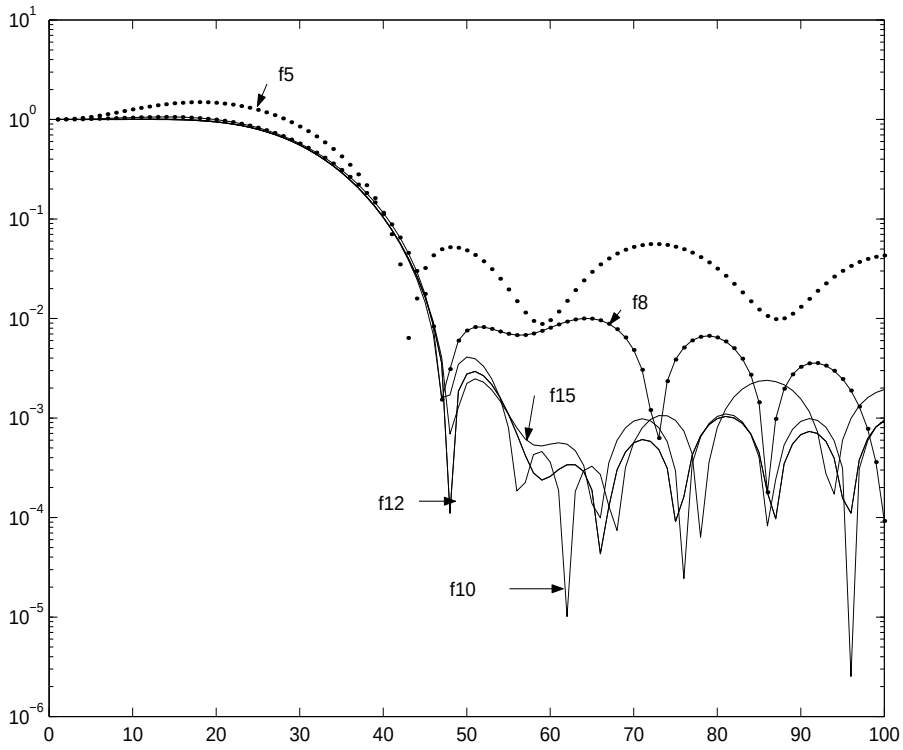
where $c_q(i)$ and $c(i)$ are the original and quantized coefficient, respectively, n is the number of bits for the quantization. Note that if $max(c(i)) > 1$, then it may be necessary to scale the filter coefficients. Scaling can be avoided by simple trick described at the end of this section.

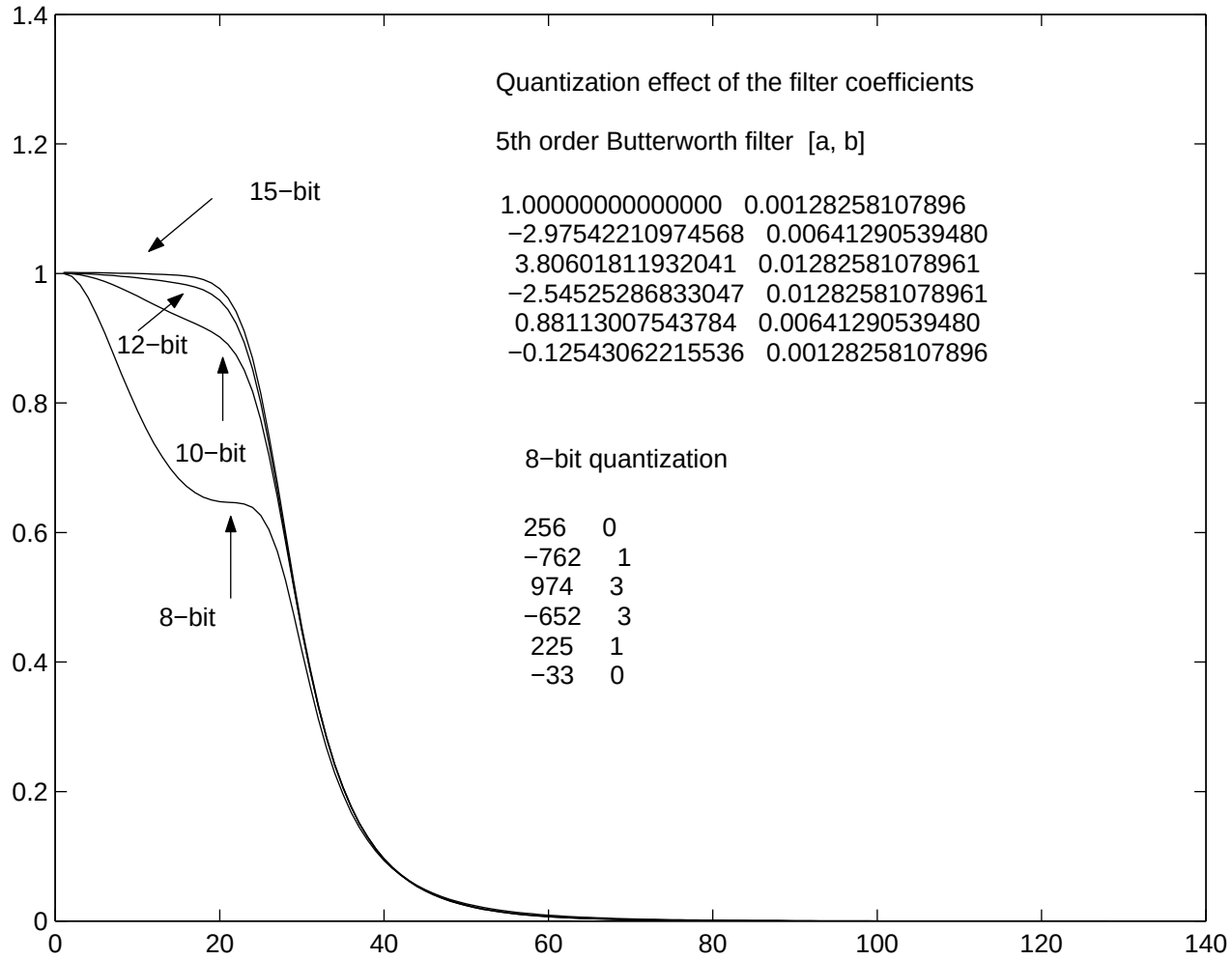
Quantization introduces distortions to the desired frequency response. The following Matlab codes are used to show the effect of quantization.

```
f=fir1(20,0.3);          %design a 21-tap FIR filter, with cut-off frequency of 0.3
f5=floor(f*2^5);         %quantize them to 5-bit
f8=floor(f*2^8);         %quantize them to 8-bit
f10=floor(f*2^10);       %quantize them to 10-bit
f12=floor(f*2^12);       %quantize them to 12-bit
semilogy(abs(freqz(f5/sum(f5),1,100)));
hold on
semilogy(abs(freqz(f8/sum(f8),1,100)));
semilogy(abs(freqz(f10/sum(f10),1,100)));
semilogy(abs(freqz(f12/sum(f12),1,100)));
```

So we see that the performance of the filter changes a lot as the number of bits used to represent the filter varies from 5 to 15. The quantization effect is more significant for an IIR than an FIR filter. FIR filters are always stable, while quantization could turn a stable IIR filter into a unstable one since the poles of the filter with quantized coefficients may be outside the unit circle. The following figure shows the quantization effect of a 5th order Butterworth filter.

Figure 62: Quantization effects





Implementing an IIR filter in a fix point processor needs more careful efforts. For example to implement an IIR filter described the difference equation:

$$y(n) = 3.125x(n) + 1.17x(n-1) + 0.9y(n) - 2.11y(n-1)$$

one can rewrite the equation in the following way:

$$y(n) = 3x(n) + 0.125x(n) + x(n-1) + 0.17x(n-1) + 0.9y(n) - 2y(n-1) - 0.11y(n-1)$$

In this equation, variables multiplied by an integer can be implemented by addition. Those fractional multiplication factors must be quantized.

14.4 Fix point digital signal processor-TMS320c5x

The aim of this section is to introduce the key elements in the implementation of a real time digital filter in the TMS320C50 DSK. The operations of the instruction MACD is explained in detail. You should have a better understanding of this lecture and the processor as well as its assembly language after you have finished the laboratory session. More information about the processor and the DSK can be found in the "TMS320C5x User's Guide" and the "TMS320C5x DSP Starter Kit User's Guide"

14.4.1 Addressing modes

The following is a brief description of two addressing modes used in the implementation of the FIR filter.

Direct addressing

In the direct memory addressing mode, the instruction contain the lower 7-bit of the data memory address (dma). The 7-bit dma is concatenated with the 9-bit of the data memory page pointer (DP) in status register 0 to form the full 16-bit address. The DP bits can be loaded by using the LDP instruction. It should be noted that the DP is not initialized by reset and , therefore, is undefined after power on. It is critical that all programs initialized the DP in software.

Indirect addressing

The C50 processor has 8 16-bit auxiliary registers(AR0-AR7) that provide flexible and powerful indirect addressing. Any location in the 64K-word data memory space cane be accessed using a 16-bit address contained in an AR. For example, to select a AR0 as the current AR and load it with the address 0f00, you can use the following code :

```
MAR     *, AR0
LAR     AR0 #0f00
```

The following indirect-addressing symbols are used in the C5x assembly language instructions

*	no increment or decrement. content of the current AR is used as the dma
*+	content of the current AR is used as the dma, after memory access, content of the current AR is increment by 1
*-	content of the current AR is used as the dma, after memory access, content of the current AR is decrement by 1

14.4.2 Implementing an FIR filter

An N*th* order FIR filter is represented by the following equation:

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

The real time implementation of this filter involves: (1) storing the coefficients in memory, (2) storing N samples of the data in memory and (3) performing the multiplication-and-sum operations and updating the data as the filtering operation finishes. For simplicity, we will use a 3rd order FIR filter to illustrate the implementation. The filter coefficients are $a_0 = 1$, $a_1 = 2$, and $a_2 = 1$. They are stored in memory location starting from 0xf280. The three data samples are stored in in memory location starting from 0xf380 and they initialized as zeros. The newest sample $x(n)$ is always stored in the location 0xf380. This is illustrated in the following figure:

				$\downarrow x(n+1)$	
f280	a_2	1	f380	$x(n)$	0
f281	a_1	2	f381	$x(n-1)$	0
f282	a_0	1	f382	$x(n-2)$	0

In this figure, the arrow indicates that when a new sample $x(n+1)$ is available the data memory must be moved up by one sample so that the next filtering operation can be performed. The source code for specifying the filter coefficients and initialization of data memory is shown below:

```
            .ds     0f28    ;the address of coeff
coefftab    .word   1       ;a2
            .word   2       ;a1
            .word   1       ;a0
            .ds     0f38    ;the address of data
XN          .word   0
            .word   0
XNLAST      .word   0
```

The implementation of the FIR filter is based on a powerful instruction called *MACD*. The source code is listed blow.

RECEIVE:			
	LDP	#XN	;load DP register
	CLRC	INTM	;clear interup mode
	LAMM	DRR	;read a new sample from DRR
	SACL	XN	;store new sample to location XN
;start FIR			
	ZAP		;ACC=0,P=0
	LAR	ar0, #XNLAST	;load the address of XNLAST to ar0
	MAR	*,ar0	;content of ar0 is used as data memory address
	RPT	#taps-1	;repeat MACD until repeat counter=0
	MACD	#coefftab,*-	;multiplication and sum and data move
	APAC		; ACC = ACC +P
	SACH	Output, 1	;get rid of extra sign bit
	LACC	Output	;load output to ACC
	SFL		;left shift ACC one bit
	AND	#0fffch	;get rid of last 2-bits for a 14-bit DAC
	SAMM	DXR	;send to output register
	RETE		;return

After reading and storing the new data sample XN, the ACC and P register are set to zero, and the address of the data memory is pointed to the last data sample. The execution of the the next two instructions (RPT and MACD) is shown the following steps:

if(repeat_counter !=0)

{

1. (ACC) = (ACC) + (shifted P register); //the shift is defined by the PM status bit
2. (P register) =(coefftab) * (current data address); //the current coefficient is multiplied by the the content of the data memory whose address is pointed to by the current AR
3. current data address = current data address -1; //point to next data
4. PFC = (PFC) + 1; //coefftab point to next coefficient
5. (current data address +1) = (current data address); //move the processed sample to next data memory location
6. repeat counter =repeat counter -1

else

{perform the above steps 1, 2,3, 5}

The following figure shows the actual implementation using a 3rd order FIR filter. In the figure, the symbol "#" represents the current data/coefficient.

step 1		step 2		step 3		step 4	
$\#a_2$	$x(n)$	a_2	$x(n)$	a_2	$\#x(n)$	$\#a_2$	$x(n+1)$
a_1	$x(n-1)$	$\#a_1$	$\#x(n-1)$	a_1	$x(n-1)$	a_1	$x(n)$
a_0	$\#x(n-2)$	a_0	$x(n-2)$	$\#a_0$	$x(n-1)$	a_0	$\#x(n-1)$
			$x(n-2)$		$x(n-2)$		$x(n-1)$

14.4.3 The assembler source code

This section is taken from the TMS320C50 DSK User's Guide.

15 Spectrum analysis

16 Problem class-3 23

17 Exam Tutorial 24

18 Appendix-I Past Assignment Questions

Due Date 2 pm Monday

LA TROBE UNIVERSITY
DEPARTMENT OF ELECTRONIC ENGINEERING
ELE32DSP: DIGITAL SIGNAL PROCESSING

(Attach the underneath to the front of your assignment)

Name Student No.
Subject Digital Signal Processing Assignment
Lecturer Dr Dennis Deng

DECLARATION

I certify that the above assignment is my original work and that no part of it has been copied or reproduced from any other person's work without acknowledgment.

Signed: Date:

1. A signal is given by x[-1] = 1, x[0] = 2, x[1] = 2, and x[2] = 1. Sketch the following signals and represent them as a linear combination of $\delta[n]$. (1.1) x[n], (1.2) x[3-n], (1.3) x[n-2].

2. Use the signal $x[n] = cos(\omega n)$ to show that the discrete radian frequency is in the range $[0, 2\pi]$. Assume this signal is a periodic signal with a period of 7. Determine all possible distinctive frequencies for this periodic signal.

3. A signal is given by x[0] = 1, x[1] = 2, x[2] = 2, and x[3] = 1. Perform the convolution operations $x[n] * h[n]$ for the following unit sample responses. (3.1) h[0] = 1, h[1] = -1, (3.2) h[-1] = -1, h[0] = 2, h[1] = -1, (3.3) h[0] = 1, h[1] = 0, h[2] = -2, h[3] = 0, h[4] =1.

4. The signal specified in question 3 is processed by a system which is a cascade of two sub-systems specified by (3.1) and (3.2) respectively. Determine and sketch the output of the whole system. Is this a causal system ?

5. An LTI system is given by $y[n] = x[n] - 2x[n-1] + x[n-2]$. Determine the unit sample response and frequency response of the system.

6. A system is given by $y[n] = ax[n] + b$ where a and b are two constants. Determine if this is an LTI system

Due Date 2 pm Monday

LA TROBE UNIVERSITY
DEPARTMENT OF ELECTRONIC ENGINEERING
ELE32DSP: DIGITAL SIGNAL PROCESSING

(Attach the underneath to the front of your assignment)

Name Student No.
Subject Digital Signal Processing Assignment
Lecturer Dr Dennis Deng

DECLARATION

I certify that the above assignment is my original work and that no part of it has been copied or reproduced from any other person's work without acknowledgment.

Signed: Date:

1. (a) Find the frequency response of the LTI system whose input and output satisfy the difference equation

$$y[n] = 0.5y[n-1] + x[n] + 2x[n-1] + x[n-2]$$

   (b) Write a difference equation that characterizes s system whose frequency response is

$$H(\omega) = \frac{1 - 0.5e^{-j\omega} + e^{-j3\omega}}{1 + 0.5e^{-j\omega} + 0.75e^{-j2\omega}}$$

2. An LTI system is given by:

$$y[n] = x[n-1] - 2x[n-2] + x[n-3]$$

   Find its unit sample response and frequency response. Is this a high pass filter ?

3. Use the Fourier transform to perform the convolution $x[n] * h[n]$, where $x[n] = \{1, 2, 1\}$ and $h[n] = \{1, -1\}$.

4. An LTI system is given by the difference equation

$$y[n] = x[n] - 3x[n-1] + 3x[n-2] + x[n-3]$$

   (a) Obtain an expression for the frequency response of this system. Hint:

$$(1-a)^3 = 1 - 3a + 3a^2 - a^3$$

(b) What is the output if the input is

$$x[n] = 10 + 4cos(0.5\pi n)$$

(c) What is the output if the input is

$$x[n] = \delta[n]$$

(d) What is the output if the input is

$$x[n] = 10 + 4cos(0.5\pi n) + 2\delta[n-3]$$

5. Let $x(t) = 7sin(11\pi t)$. In each of the following cases, the discrete time signal $x[n]$ is obtained by sampling $x(t)$ at a frequency f_s and the resultant $x[n]$ can be written as

$$x[n] = Acos(2\pi f_0 n + \phi)$$

So for each case below, determine the values of A, f_0 and ϕ. In addition, state whether the sampling theorem is satisfied. (a) $f_s = 5$ Hz, (b) $f_s = 10$ Hz (c) $f_s = 15$ Hz

6. 2.3 The sequence, $x[n] = \cos(\frac{\pi}{7}n), \quad -\infty < n < \infty$, was obtained by sampling an analog signal $x_c(t) = \cos(\Omega t), \quad -\infty < t < \infty$, at a sampling rate of 2000 samples/second. What are the three smallest possible values of Ω ($\Omega > 0$) that could have resulted in the same discrete signal ?

7. The band width of an analog signals 8 kHz. This signal is processed by an analog low pass filter whose cutoff frequency is 3 kHz. If this signal processing task is performed by a digital signal processing system, determine the minimum sampling frequency and the corresponding cut off frequency of the digital low pass filter.

Due Date 2 pm Monday

LA TROBE UNIVERSITY
DEPARTMENT OF ELECTRONIC ENGINEERING
ELE32DSP: DIGITAL SIGNAL PROCESSING

(Attach the underneath to the front of your assignment)

Name Student No.
Subject Digital Signal Processing Assignment
Lecturer Dr Dennis Deng

DECLARATION

I certify that the above assignment is my original work and that no part of it has been copied or reproduced from any other person's work without acknowledgment.

Signed: Date:

Question 1

Design a 9-tap FIR low pass filter with a rectangular window to separate two sinusoidal signals whose frequencies are 500 Hz and 50 Hz, respectively.

- Determine the specification of the filter.

(4 marks)

- Write an expression for a seven term casual impulse response

(6 marks)

- How would you modify the filter coefficients to make it a high pass filter that can be used to separate these two signals ?

(6 marks)

Question 2

Briefly describe the frequency response of the following types of IIR filters

- Butterworth filter
- Chebyshev filter
- Elliptic filter

(9 marks)

The following questions are related to filter design using Matlab. Some useful Matlab functions are `semilogy`, `fir1, freqz` and those listed in Section 13.3 of the lecture notes

Question 3

Design a low pass FIR whose cut off frequency is $\pi/4$.

(a) Determine the filter coefficients of a 25-tap filter using the following windows: rectangular, Hamming and Bartlett window.

(9 marks)

(b) Plot the magnitude response of each filter

(9 marks)

(c) Compare the magnitude responses and identify their differences in terms of the transition between the pass band and the stop band and the ripples.

(7 marks)

Question 4

Determine the transfer function of a third order low pass Butterworth filter whose cutoff frequency is $\pi/4$. Plot its amplitude response. If the same amplitude response is produced by an FIR filter, estimate the order of that FIR filter by trying different order and plotting the corresponding amplitude response.

(10 marks)

Question 5

Determine the transfer function of a Chebyshev type 1 filter with the following specifications:

(a) 1 dB ripple in the range 600-900 Hz (b) the sampling frequency is 3000Hz

(c) a maximum gain of -40 dB at frequency range $[0, 200]$ Hz and $[1300, \infty)$ Hz.

Draw a direct form-2 implementation of this filter.

(10 marks)

Due Date 2 pm, Monday Oct. 30, 2006

LA TROBE UNIVERSITY
DEPARTMENT OF ELECTRONIC ENGINEERING
ELE32DSP: DIGITAL SIGNAL PROCESSING

(Attach the underneath to the front of your assignment)

Name Student No.
Subject Digital Signal Processing Assignment
Lecturer Dr Dennis Deng

DECLARATION

I certify that the above assignment is my original work and that no part of it has been copied or reproduced from any other person's work without acknowledgment.

Signed: Date:

Question 1

We want to use DSP to separate two sinusoidal signals whose frequencies are 500 Hz and 50 Hz, respectively. Assume the sampling frequency is 4000Hz. You need to design a 7-tap FIR low pass filter with a rectangular window.

(a) Determine the specification of the filter.

(b) Write an expression for a seven term casual impulse response

(c) How would you modify the filter coefficients to make it a high pass filter that can be used to separate these two signals ?

(10 marks)

Question 2

We want to use DSP to implement the following signal processing task.

(a) 1 dB ripple in the range 600-900 Hz

(b) the sampling frequency is 3000Hz

(c) a maximum gain of -40 dB at frequency range $[0, 200]$ Hz and $[1300, \infty)$ Hz.

Determine the transfer function of a Chebyshev type-1 filter and raw a direct form-2 implementation of this filter.

(10 marks)

Question 3

An LTI system is described by the difference equation

$$y[n] = 0.8y[n-1] - 0.8x[n] + x[n-1]$$

(a) Determine the system function $H(z)$.
(b) Plot the poles and zeros in the z-plane and determine if the system is stable.
(c) Obtain the frequency response and show ???that $|H(\omega)|^2 = 1$ for all ω.

(15 marks)

Question 4

The pole-zero plot of an LTI system is shown in the following figure.
(a) Determine the system function.
(b) Determine the impulse response.

(15 marks)

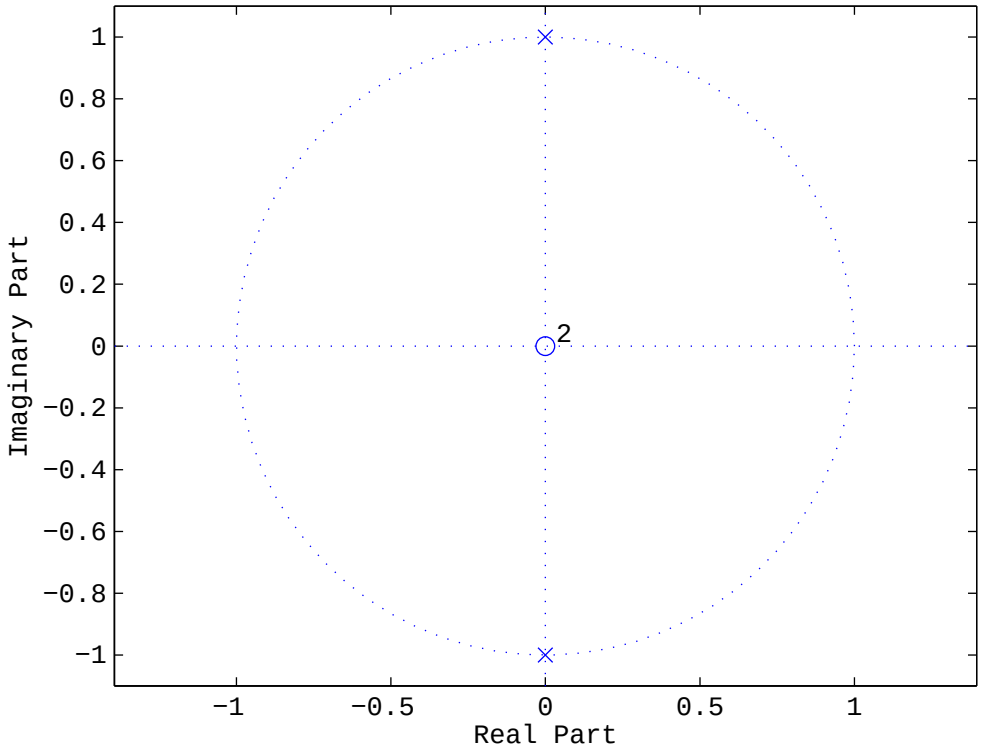


Figure 63: The pole-zero plot of an LTI system

Due Date 2 pm, Monday Oct. 30, 2006

Late submission will NOT be accepted

LA TROBE UNIVERSITY

DEPARTMENT OF ELECTRONIC ENGINEERING

ELE32DSP: DIGITAL SIGNAL PROCESSING

(Attach the underneath to the front of your assignment)

Name Student No.
Subject Digital Signal Processing Assignment
Lecturer Dr Dennis Deng

DECLARATION

I certify that the above assignment is my original work and that no part of it has been copied or reproduced from any other person's work without acknowledgment.

Question 1

We want to use DSP to implement the following signal processing task.
(a) 1 dB ripple in the range 600-900 Hz
(b) the sampling frequency is 3000Hz
(c) a maximum gain of -40 dB at frequency range $[0, 200]$ Hz and $[1300, \infty)$ Hz.

Determine the corresponding specifications for the digital filter and use Matlab to design a Chebyshev type-1 filter. Express the filter using a difference equation and draw a direct form-II implementation of this filter.

(15 marks)

Question 2

An LTI system is described by the difference equation

$$y[n] = 0.8y[n-1] - 0.8x[n] + x[n-1]$$

(a) Determine the system function $H(z)$.
(b) Plot the poles and zeros in the z-plane and determine if the system is stable.
(c) Determine the frequency response $H(\omega)$ and show that $|H(\omega)|^2 = 1$ for all ω.

(15 marks)

Question 3

The pole-zero plot of an LTI system is shown in the following figure.
(a) Determine the system function.
(b) Determine the impulse response.

(15 marks)

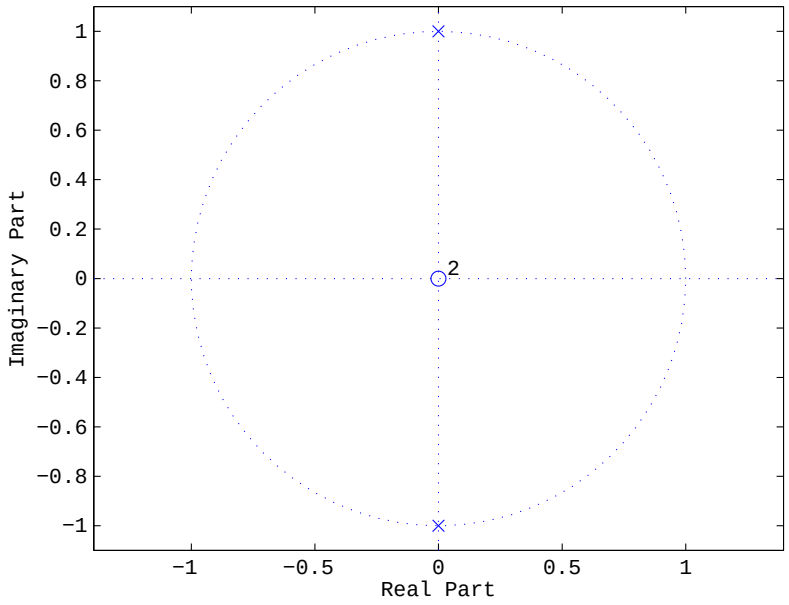


Figure 64: The pole-zero plot of an LTI system

Question 4

Match the frequency responses (A-E) with the correct pole-zero plots (PZ 1-6). Poles are denoted by x and zeros by o.

(15 marks)

LA TROBE UNIVERSITY
DEPARTMENT OF ELECTRONIC ENGINEERING
ELE31DSP: DIGITAL SIGNAL PROCESSING

(Attach the underneath to the front of your assignment)

Name Student No.
Subject Digital Signal Processing Assignment
Lecturer Dr Dennis Deng

DECLARATION

I certify that the above assignment is my original work and that no part of it has been copied or reproduced from any other person's work without acknowledgment.

Signed: Date:

The following questions are related to window methods for FIR filter design which is cover in the lecture notes, in section 8.2.2 of Proakis' book, and in section 7.4 of Oppenheim's book. Other DSP books also cover this topic (but not DSP First). You need to use Matlab to do the simulation. Matlab is available in the PC's in the CAD lab.

1. Derive the impulse response for an ideal low pass filter and an ideal high pass filter whose cut-off frequency is ω_c. Briefly describe the window method for FIR filter design.

2 Based on your results in question 1, design an FIR filter approximating the ideal frequency response

(a) Determine the filter coefficients of a 25-tap filter using a rectangular window
(b) Plot the magnitude response of the filter

Use Matlab function ***fir1*** to design the same FIR filter

(c) Using the Hamming window
(d) Using the a Bartlett window
(e) Compare the magnitude responses and identify their differences.

Note: In Matlab function ***fir1,*** *the highest frequency is* **1**. *So the cutoff frequency of* $\pi/6$ *corresponds to 1/6.*

3. Based on your results in question 1, design an FIR filter approximating the ideal frequency response

(a) Determine the filter coefficients of a 25-tap filter using a rectangular window
(b) Plot the magnitude response of the filter

4. An analog signal is given by $s(t) = \cos(10000t)$.

(a) Determine the Nyquist sampling frequency for this signal and the expression for the corresponding digital signal.

(b) If this analog signal is sampled with a period T=10^{-6} second, then the corresponding digital signal is :

$s(n) = \cos(0.01n)$. What is the digital frequency of this signal ? Assume that the signal is corrupted by noise :

$x(n) = s(n) + r(n)$, where r(n) is the noise. In Matlab, this can be simulated by:

n=0:1023; %assume we have 1024 data

*x=cos(0.01*n)+rand(size(n)); %create the noise corrupted signal*

plot(x)

(c) Design a low pass filter to filter out noise. This can be easily done using Matlab commands ***fir1*** and ***filter***. Use the ***help*** command to learn how to use these two commands. Justify the cut-off frequency you use in the low pass filter. Using a Hamming window and a fixed cut-off frequency, compare the filtering results by using filters of different length (4, 14, 24, 104).

A sample Matlab code to implement the filtering process is given below:

f=fir1(4,0.4); %design a 5-tap low pass filter, cutoff frequency=0.4

y=filter(f, 1, x); %filter the signal

plot(y) %plot the output

(d) Repeat part (c) using a rectangular window.

(e) Compare the results in parts (c) and (d). For the same window, what do you observe as the length of the filter increases ? For the same filter length, which window produces a better filtering result ? The above questions can be answered by plotting the magnitude of the frequency response of different filters, ie, *plot(abs(freqz(f,1,128)))*.

LA TROBE UNIVERSITY
DEPARTMENT OF ELECTRONIC ENGINEERING
ELE31DSP: DIGITAL SIGNAL PROCESSING

(Attach the underneath to the front of your assignment)

Name Student No.
Subject Digital Signal Processing Assignment
Lecturer Dr Dennis Deng

DECLARATION

I certify that the above assignment is my original work and that no part of it has been copied or reproduced from any other person's work without acknowledgment.

Signed: Date:

0. The following questions are related to IIR filter design using Matlab. Matlab is available in the PC's in the CAD lab. Section 13.3 of the lecture notes provides a brief summary of all functions you need. Use help function-name to learn how to use them.

1. Determine the transfer function of a third order low pass Butterworth filter whose cutoff frequency is $\pi/4$. Plot its amplitude response. If the same amplitude response is produced by an FIR filter, estimate the order of that FIR filter by trying different order and plotting the corresponding amplitude response.

2 Determine the transfer function of a Chebyshev type 1 filter with the following specifications:
(a) 1 dB ripple in the range 600-900 Hz (b) the sampling frequency is 3000Hz
(c) a maximum gain of -40 dB at frequency range $[0, 200]$ Hz and $[1300, \infty)$ Hz.
Draw a direct form-2 implementation of this filter.

3. Determine the transfer function of a low pass elliptic filter with the following specifications:
(a) pass band ripple 0.5 dB (b) stop band gain -30 dB
(c) pass band ripple-edge frequency = 2 kHz
(d) stop band ripple-edge frequency = 2.5 kHz
Draw a direct form-2 implementation of this filter

4. Design a second order digital IIR filter to remove the $\cos 5t$ component from the following signal:

$$x(t) = 1 + \cos t + \cos 5t$$

You should clearly specify the sampling frequency, the cutoff frequency and the type of the filter you use. Suppose the filter coefficients are stored in *b* and *a,* and the sampling interval is T, then the following Matlab code first generates a digitized signal then filters it:

*n=0:1023; T=0.1; x=1+cos(T*n)+cos(5*T*n); y=filter(b,a,x); %sample only*

Show your results by plotting the original and the filtered signals.

19 Appendix-II Laboratory Instructions

A Discrete signals and systems

LA TROBE UNIVERSITY

DEPARTMENT OF ELECTRONIC ENGINEERING

ELE32DSP LABORATORY

Preliminary work

The following is a list of preliminary work associated with using Matlab. You should go through the following exercises in your spare time before the lab. You should be able to show your work upon request.

Defining and plotting signals

- It is an easy task to define a signal. For example, we can defined
  $h = [1\,2\,1]$;
  $n = 0:127$;
  $s = sin(pi * n/10)$;

- You can also form a new signal by using the following
  $s1 = [hs(1:40)\, zeros(1,10)\, ones(1,10)]$;
  `stem`(s1)

- To visualize the signal, we can use the following
  `figure`(1)
  `stem`(h)
  `figure`(2)
  `plot`(n,s)

- If you want to have two figures in one "big" figure, you use the command `hold on`

  ```
  figure(1);clf
  subplot(211);stem(h);
  subplot(212);plot(n,s);
  figure(2);clf
  subplot(121);stem(h);
  subplot(122);plot(n,s);
  ```

  Look at figure 1 and figure 2 carefully, you will see how to use the command `subplot` properly.

- To plot two signals in one figure, you use the command `hold on`
  `figure`(3)
  $c = cos(pi * n/10)$;
  `plot`(n,c,'b');
  `hold on`
  `plot`(n,s,'k')

- You should know how to use the functions: **`plot, subplot, stem, figure`** and **`clf`**.

The frequency response, amplitude and phase

- The frequency response of an LTI system with the finite impulse response of $h[n]$ is calculated by using the Maltab `function freqz.` It produces a complex valued vector.

  $h = [1\ 1\ 1]$;%define the impulse response
  M = 128; %take 128 samples from 0 to π
  [H,W] = `freqz`(h,1,M); %h is the impulse response, 1 for an FIR filter, M is the number of samples
  Ha = `abs`(H); %the amplitude
  Hp = `angle`(H); %the phase
  `semilogy`(W,Ha); %similar to plot, but use log-scale in the y-axis.
  plot(W,Hp); %plot the phase response

  Here H is a complex valued vector. The kth element of H, denoted $H[k]$, is the frequency response $H(\omega)$ evaluated at $\omega_k = \frac{\pi}{M}(k-1)$, i.e., $H[k] = H(\omega_k)$. The amplitude of the frequency response at the particular frequency ω_k is given by $abs(H(k))$ and the phase is $angle(H(k))$. When k takes the value of 0,1,...,M, ω_k takes the value of 0, $\frac{\pi}{M}$,...,$\frac{(M-1)}{M}\pi$. A larger value of M leads to a better approximation of $H(\omega)$ by the vector H.
  The other vector W represents the frequency. For example, the kth element of W, denoted $W[k]$, is $W[k] = \omega_k = \frac{\pi}{M}(k-1)$. You should use **`help freqz`** to learn more about the function.

- You should know how to use the function **`freqz`** and **`semilogy`** to visualize the frequency response of an FIR filter.

1. Introduction

In this lab you will use Matlab to study the frequency response of simple filters and to simulate an image sharpening algorithm using convolution and the FFT. After this lab you should gain a deeper understanding of certain important concepts such as convolution, frequency, amplitude and phase response, and the Fourier transform. You should also learn numerical simulation techniques for system analysis and implementation in the time domain and the frequency domain.

This lab is divided into three parts. The first part is designed for a review of some important concepts. The second and third parts are designed for system analysis and implementation.

You should ask a tutor to verify what you have done as required in the verification sheet.

Lab Report

A lab report is due on Monday, Sept. 11, 2006. In the report, you should briefly comment on what you have done and learned. You should also answer all questions.

2. A review of signals and systems

In this section, we assume that the first element of the unit sample response of a filter is in the zero position, for example, given $h_0[n] = \{1, -2, 1\}$ we assume $h_0[0] = 1$.

2.1 The unit sample response

Plot the amplitude and phase responses of the filter given by $h_0[n]$ using the following Matlab code

```
h0 = [ 1 -2 1];
H0 = freqz ( h0, 1, 128, 'whole');   %calculate freq response from 0 to 2π
plot ( abs ( H0 ) )                  %amplitude
figure ( 2 )
plot ( angle ( H0 ))                 %phase
```

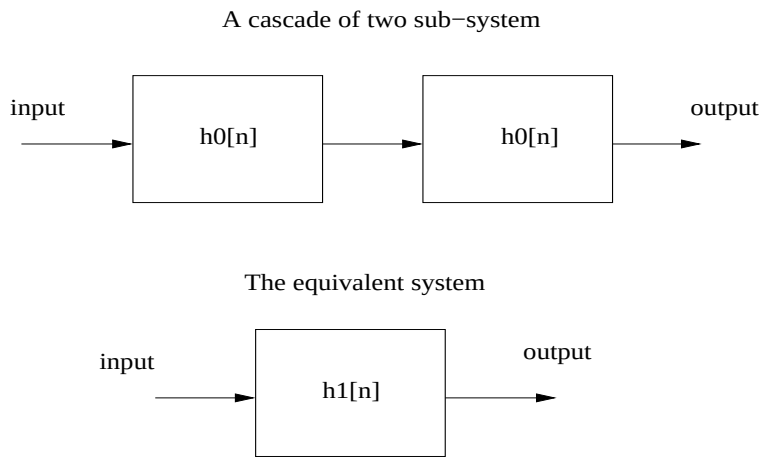
If you are not familiar with the above Matlab commands, you can use the "`help`" command, eg, `help freqz`. Observe that the amplitude is an even function and the phase is an odd function.

- Explain why the frequency response is usually specified in the range $[0, \pi]$?

2.2 A cascade of two filters

An LTI system consists of a cascade of two identical sub-systems (shown in the following figure) specified by the difference equation

$$y[n] = x[n-1] + x[n] + x[n+1]$$



- Determine the unit sample response of the sub-system, denoted by $h_0[n]$ and the whole system, denoted by $h_1[n]$. Verify your results by using the convolution command: conv.
- Determine the system function of the system which is the z-transform of the impulse response. You can skip this step if you do not know the z-transform.
- Plot the amplitude response of the system.
- Is this system a causal filter ? If not, what is the simplest way to make it causal ?
- Explain that in the frequency domain the relationship between the magnitude responses of the whole system and sub-system is $|H_1(\omega)| = |H_0^2(\omega)|$. You can verify this relationship in Matlab by using: plot(abs(H1)-abs(H0.*H0)).

2.3 Adding a zero to the unit sample response

- Plot the amplitude and phase responses of the following three filters: h1=[1 1 1], h2=[0 1 1 1] and h3=[1 1 1 0].
- Discuss the effects of adding a zero (or zeros) before (or after) the unit sample response. The phase response of h3 should be the same as that of h1. To see the relationship between phase response of h0 and that of h2, you can use the following Matlab code:

```
H1 = freqz (h1, 1, 128) ;
H2 = freqz (h2, 1, 128) ;
H3 = freqz (h3, 1, 128) ;
p1 = angle ( H1 );
p2 = angle ( H2 );
p3 = angle ( H3 );                  %p3 should be the same as p1
p4 = unwrap ( p2 );                 %use help to see what 'unwrap' does
i = 0 : 127; p5 = p1 - i'*pi/128    % subtracting a phase angle ω from p1
plot ( p5 - p4 )                    % shows the phase relationship between h1 and h2
```

- Note that according to the Fourier transform $p2 = p1 - \omega$. Derive this equation.

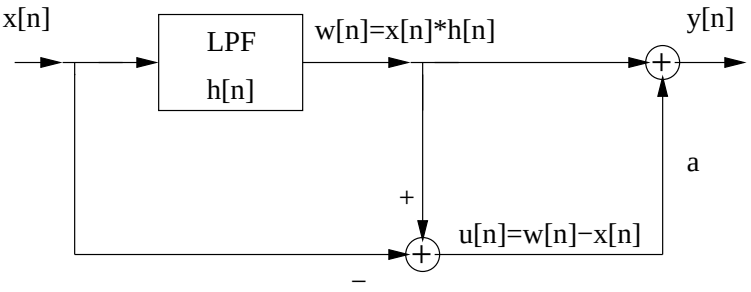


Figure 65: The image processing filter.

2.4 Optional

Use the following example to study the range of the that the discrete radian frequency.

```
clear
n = 0 : 100;
x1 = cos ( 13 * n * pi / 3 );
x2=cos ( n * pi / 3 );
figure ( 1 )
clf
subplot ( 211 ); plot ( x1 )
subplot ( 212 ); plot ( x2 )
```

- What are the frequencies for both signals?
- A discrete sinusoidal signal is not necessary periodic. Generate and plot such a signal to show that it is not periodic. (Hint: follow the above example to generate the signal).

3. System analysis

A simple and effective image sharpening filter is shown in Figure 1. In this filter, a is a constant.

- Assume the low pass filter (LPF) is fixed and its frequency response is denoted by $H_0(\omega)$. Derive the frequency response of this filter, denoted by $H(\omega)$, in terms of $H_0(\omega)$.
- Assume the LPF is given by $h_0[n] = \{1/3, 1/3, 1/3\}$, plot the amplitude response of $H(\omega)$ using $a = 0.1$ and $a = 3$.
- Assume the LPF is $h_0[n] = \{1/5, 1/5, 1/5, 1/5, 1/5\}$, plot the amplitude response of $H(\omega)$ using a=0.1 and a=3. Study the difference in the amplitude response for these two different low pass filters.

4. System implementation

Try the following Matlab code, which reads an image into an array and displays it.

```
y = imread ( ' cameraman.tif ' ) ; %read a (256*256) image into array y
clg %clear figure
colormap ( gray ( 256 ) ); %set the display to 256 graylevel
```

```
image ( y ) %display image
y = double ( y ); % make it a double array required by conv
size(y) % the size of the image
```

4.1 Matlab script file

Matlab code can be saved as a .m-file, which can be used by simply calling the file name. In the Matlab window, click on File/New/M-file. Type in the following Matlab code and save it as `abc.m`, then in the Matlab command window type: `abc`. It is not important how you name the file. But you can not name it by using a number. For example, naming the file as `2.3.m` or `2.m` is not allowed.

```
h0 = [ 1 1 1] / 3;
a = input ( ' Input the sharpening factor, a>1 sharpening, a<1 blurring');
%The following part implements the image sharpening filter in the time domain
for i = 1 : 256
z ( i , 1 : 256 ) = (1 + a) * conv (y ( i ,1 : 254 ) , h0) - a * y ( i , 1 : 256);
end

%The following part limits the output pixel value to the range [0, 255]
z =z.*( z >= 0 ) ;
z = z.*(z <= 255 ) +255 * (z > 255 );
figure ( 1 );
colormap ( gray ( 256 ) );
image ( z' )
```

Using the above Matlab code, the image is processed line by line. It is important to note that the length of the signal resulted from a convolution of two signals of M and N elements is $N+M-1$. Therefore, to keep the processed image the same size as that of the original (in this case 256 pixels/line), a simple way is to use part of the signal such that after the convolution the resulted signal has 256 pixels. That is why we have the following: `conv (y ( i ,1 : 254 ) , h0)` in the Matlab code. For the *i*th line, we take the first 254 pixels as the signal. Why use 254 pixels? Because h0 has 3 samples, the length of the signal from convolution is 254+3-1=256. Therefore, if the length of h0 changes to 5, then we only take 252 pixels. However, I should emphasize that this is NOT a very good way to deal with the problem. A better way is to perform the convolution is take only the first 256 pixels of the result. The Matlab code is as follows

```
for i = 1 : 256
tmp = conv (y ( i ,:) , h0);
tmp1 = tmp(1 : 256 ); %taking the first 256 pixels
z ( i , 1 : 256 ) = (1 + a) * tmp1 - a * y ( i , 1 : 256);
end
```

4.2 Image processing using convolution

Try two settings $a = 0.1$ and $a = 3$ to process the image and compare the input and output images. Explain what you have observed using the results in section 3.1. Repeat this experiment by using h_0=[1 1 1 1 1]/5;

4.3 Image processing using the FFT

Convolution can also be implemented in the frequency domain using the fast Fourier transform (FFT). The convolution part of the above m-file can be implemented as:

```
h = [ h0 zeros ( 1, 253 ) ];
H0 = fft ( h ); %convert h to the frequency domain
for i = 1 : 256
Y = fft ( y ( i , 1 : 256 ) );
Z = ( ( 1 + a ) * H0 - a ) . * Y;
z ( i , 1 : 256 ) = real ( ifft ( Z ) );
end
```

Here we make another impulse response h by appending 253 zeros to h0. As you have seen in section 2, appending zeros to the end of a signal does not change the signal. The reason for doing so is because in the frequency domain we need to perform the multiplication operation of H0 and Y. They must be of the same size.

Here is your task: modify your m-file by replacing the convolution part with the above FFT section. Save it as a new m-file. Run this m-file. You should compare the results of this section with those of section 4.2.

4.4 Noise filtering

Now let us add some noise to the image by using

```
y = imread (' cameraman.tif ' ) ;
y = y + floor ( k * randn ( 256 ) ) ;
```

where k is a constant that controls the amount of noise added to the image. Try k=10, 20, 30. By displaying y, you can see that the quality of the image is degraded.

For $k = 30$, repeat experiments specified in section 3.1 by setting $a = 0.1$ and $a = 3$, respectively. Which filter produces a better image ? An important lesson you can learn from this simple experiment is that a low pass filter will filter out noise and blur the image, while a high frequency emphasis filter will sharpen the image and make the noise more visible.

Instructor Verification

Name

Student Number

Your laboratory work is divided into three parts. In each part, your work must be marked by the demonstrator. The mark is from 1 (poor) to 5 (excellent).

Preliminary work

Signals and frequency response	

Part 1. A review of signals and systems

2.1 The unit sample response	
2.2 A cascade of two filters	
2.3 Adding a zero to the unit sample response	

Part 2. System analysis

System analysis	

Part 3. System implementation

4.2 Image processing using convolution	
4.3 Image processing using fft	
4.4 Noise filtering	

B Applications of digital filters

LA TROBE UNIVERSITY

DEPARTMENT OF ELECTRONIC ENGINEERING

ELE31DSP LABORATORY

Preliminary work

Here are some simple preliminary work which should be finished before the laboratory class. You should be able to show your results to a demonstrator upon request.

- To design an FIR filter and check the amplitude response, you can use the Matlab functions: `fir1` and `freqz`. Here is an example.

  help `fir1`
  h = `fir1`(12,0.3);
  H = `abs`(`freqz`(h,1,128));
  `semilogy`(H)

  What is the order of the FIR filter? What is the cutoff frequency of filter? In Matlab the frequency is normalized by π. Thus, if the frequency is 0.2π, then you should enter it in a particular filter design function as 0.2.
  Design a bandpass filter with passband [0.2π, 0.5π]. You can vary the order of the filter to see the effects. Check your design by plotting the amplitude response.

- To design an IIR filter (a Butterworth filter), you can use the Matlab function: butter. Here is an example.
  help butter
  [B,A] = `butter`(2,0.3);%a second order Butterworth lowpass filter with cutoff freq. = 0.3π
  H = `abs`(`freqz`(B,A,128));
  semilogy(H)
  Design a third order high pass Butterworth filter with cutoff frequency 0.8π and check its amplitude response.
  Design a third order band stop Butterworth filter with cutoff frequency [0.5π, 0.8π] and check its amplitude response.

- To perform the filtering operation, you can use the Matlab command `filter`. Here is am example.
  n = 1 : 1000;
  s = `cos`(pi * n /100) + 0.5 * `randn`(1,1000);
  `plot`(`abs`(`fft`(s))
  w = 0.01;
  [B,A] = `butter`(3,w);
  y = `filter`(B,A,s);
  figure
  subplot(211);plot(s)
  subplot(212);plot(y)

You should study the spectrum of the noisy corrupted cosine signal and determine a reasonable cutoff frequency. You should refer to section 6 for an interpretation of the spectrum using the `fft`.

Note that if you are using an FIR filter then A = 1.

1. Introduction

In this laboratory you will investigate the use of digital filters to recover signals that have been corrupted by noise and 50 Hz sinusoidal mains interference.

2. Signal corrupted by noise

Figure 1 shows a good ECG signal. This data was collected using a sampling rate of 1 kHz. This signal is available in the Matlab file *ecgsig.dat*. It can be loaded into Matlab by using *load ecgsig.dat*, it will then be available as th variable *ecgsig*. Try loading the ECG signal and plotting the data.

Unfortunately, when another set of ECG measurements were made, there was a problem in the digital data collection system, and the data was corrupted by high-frequency noise. This noise corrupted ECG signal is shown in Figure 2.

3. Signals corrupted by mains interference

In another set of ECG measurements, the engineers who designed the digital data collection system did not graduate from La Trobe. As a result the ECG data collected on this system was corrupted by 50 Hz interference from the power mains. The corrupted signal is shown in Figure 3.

4. Digital filters

There are two type of digital filters: finite impulse response (FIR) and infinite impulse response (IIR). In the time domain, FIR and IIR filters are represented by difference equations:

$$\text{FIR filter} \quad y(n) = \sum_{i=0}^{N-1} b_i x(n-i)$$

$$\text{IIR filter} \quad y(n) = \sum_{i=0}^{N-1} b_i x(n-i) + \sum_{j=1}^{M-1} a_j y(n-j)$$

where b_i and a_j are called the filter coefficients and N is the order of the FIR filter. The filter coefficients are usually stored in two vectors in Matlab.

A filter is usually specified in the frequency domain. For example, a low pass filter is specified by a cut-off frequency of ω_c. Matlab provides a number of functions that you can used to design an FIR or an IIR filter.

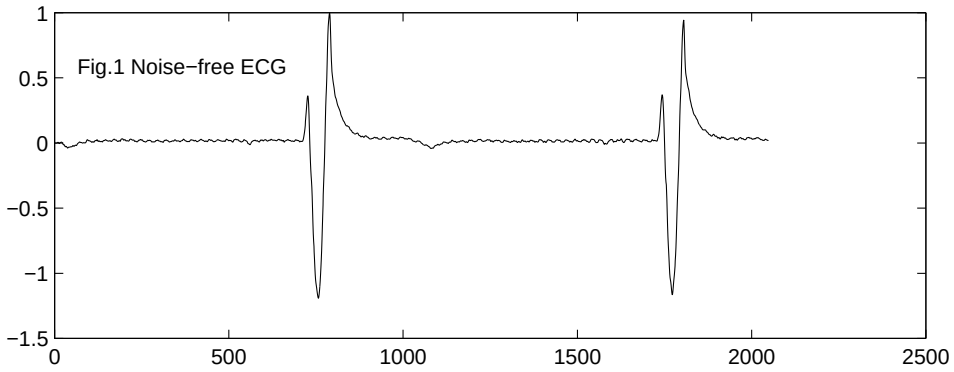


Fig.1 Noise−free ECG
1
0.5
0
−0.5
−1
−1.5
0
500
1000
1500
2000
2500

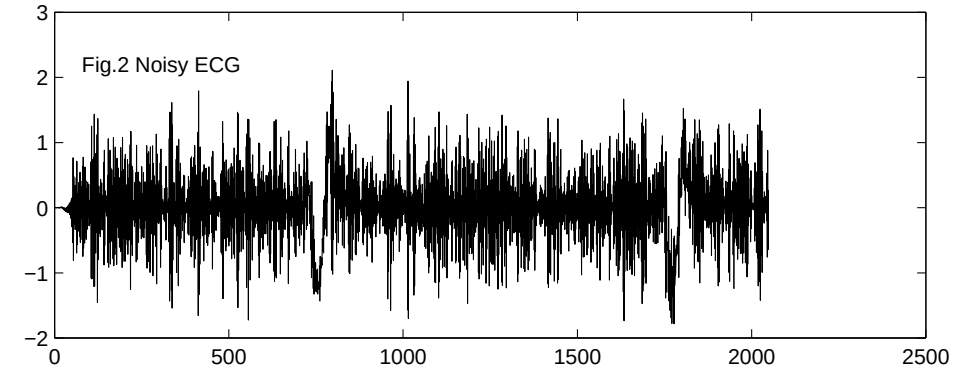


Fig.2 Noisy ECG
3
2
1
0
−1
−2
0
500
1000
1500
2000
2500

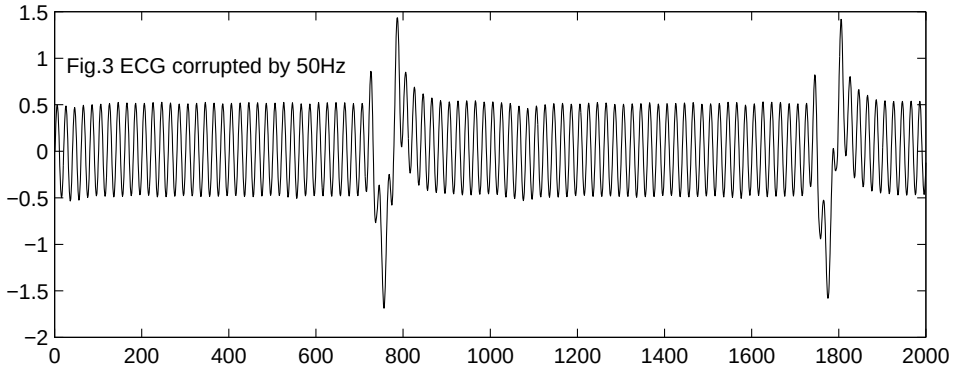


Fig.3 ECG corrupted by 50Hz
1.5
1
0.5
0
−0.5
−1
−1.5
−2
0
200
400
600
800
1000
1200
1400
1600
1800
2000

Matlab function name	Description
fir1	FIR filter design
butter	IIR filter design, Butterworth response
cheby1	IIR filter design, Chebyshev I response
cheby2	IIR filter design, Chebyshev II response
ellip	IIR filter design, elliptic response

Matlab also provides a function called *filter* to implement the filter. In this laboratory, you will learn how to design FIR filters and Butterworth IIR filters to recover the ECG signal.

NOTE Due to a problem with Matlab, when you design an IIR bandstop filter with a very sharp transition between pass band and stop band, the result may not be the same as what you specified. You should try to design IIR filters with different orders and pick the one that is closest to your filter specification. To do this, you may try the following:

```
[B, A] = butter(N,[w1,w2], 'stop');   %B and A are tow vectors
                                      %that store the filter coefficients
plot(abs(freqz(B, A, 128)))           %N is the order, w1 and w2
                                      %are two cut-off frequencies
```

5. Your task

Your task is to use both FIR and IIR filters to remove the high frequency noise and the 50 Hz interference from the ECG signals.

- To design your filter, you must look at the spectrum of the good ECG signal and the noisy ECG signal. For example[17], suppose the signal (with M data samples) is stored in a vector called s, the spectrum can be obtained by using

  $$S = fft(s, N)$$

  where N is the number of points between 0 to 2π. N is usually chosen to be a number of times larger than M. You can plot the spectrum by using

  $$plot(abs(S))$$

  The horizontal axis represents the frequency, the vertical axis represents the magnitude. Note $|S|$ is a vector. The kth element of this vector is the magnitude of the spectrum at a frequency $\omega_k = 2\pi k/N$. If the discrete signal is a result of sampling an analog signal, then this frequency[18] corresponds to $f_k = kf_s/N$ which is the frequency of the analog signal.

- You can use the Matlab function `fft` to calculate the spectrum and plot its amplitude. Based on your interpretation of the plot, determine the filter specifications and design an FIR and an IIR (Butterworth) filter to remove the noise.

- Repeat the above steps to design an FIR and an IIR (Butterworth) filter to remove the 50 Hz interference.

[17] This is only a rough estimation of the spectrum. In ELE41ASP, we will have detailed discussion on this topic.

[18] $\omega_k = \frac{2\pi f_k}{f_s} \Longrightarrow f_k = kf_s/N$

- In both cases, compare the frequency response of these two filters and their order. Which one requires less computational operations to implement ? You could also measure the time (use Matlab function *tic* and *toc*) that is required for the filtering process. What is minimum order for the FIR filter (IIR filter) to work reasonably well?
- In your lab book, you should clearly document your design method and justify the filter specifications. You should include relevant plots to support your justifications and to show your results.

Instructor Verification

Name

Student Number

Your laboratory work is divided into two parts. In each part, your work must be marked by the demonstrator. The mark is from 1 (poor) to 5 (excellent).

Part 1. Filtering out high frequency noise

Interpretation of the spectrum	
Justification of the digital filter	
Results	

Part 2. Filtering out 50Hz interference

Interpretation of the spectrum	
Justification of the digital filter	
Results	

C Sampling Frequency, Quantization Noise and Digital Filters

LA TROBE UNIVERSITY

DEPARTMENT OF ELECTRONIC ENGINEERING

ELE31DSP LABORATORY

1. Introduction

The sampling frequency and the quantization levels are two important issues in digital processing of analog signals. The purpose of this laboratory is to investigate these issues. A further purpose is to give you hands-on experience of using a digital signal processor.

The TMS320c50 DSP starter Kit (DSK) is used in the laboratory. The DSK has a fix point processor (c50), the A/D and D/A converters and other parts such as RAM, ROM and the RS232 port that is used to communicate between the DSK and the host PC. The DSK also has an analog input and an analog output. It is powered by 9v AC. A detailed description of the DSK can be found in its User's Guide.

An analog signal is converted to digital form through sampling and quantization. The digital signal is processed by the processor according to the DSP program. The processed signal is then converted into analog form.

2. Sampling

1. Connect the output of the signal generator to the input of the DSK.

2. In the DOS PROMPT, type OSCOPE to run the program which contains two parts. The DSK part converts the input analog signal into digital and send it to the PC through the RS232. The PC part is responsible for displaying the signal as well as responding to user commands such as changing the time base of the display and the quantization level.

3. Set the frequency of the signal generator to 250 Hz and make sure that the PC scope has a time base of 1 ms and a resolution of 8 bits. You should see on the PC an approximation of the same signal that can be seen on the analog oscilloscope. The period of the sine wave should be 4 ms. The red lines indicate the samples received from the DSK's ADC. A white line joins the samples to give a rough approximation of the input signal.

4. You can determine the sampling frequency of the ADC by dividing the period of the sine wave from the number of samples in one period. To count the samples, press PAUSE button on the keyboard. This makes counting easier. Press ENTER to resume counting.

5. Increase the input frequency to 1 kHz. If the display looks cluttered, the time base of the PC oscilloscope can be altered using the left and right cursor keys. The representation of the sine wave is now much poorer since there are less samples to represent it. However, there is still theoretically enough information to reconstruct the signal to its original form.

6. Turn the input frequency to 2.5 kHz. Draw the sampled sine wave. Is is still theoretically possible to reconstruct the signal to its original form ?

7. Slowly increase the frequency from 2.5 kHz to 5 kHz. Explain what you observed.

8. Set the input frequency to 3.5 kHz. Measure the frequency of the signal appeared on the PC screen and relate it to the sampling frequency and the frequency of the input signal.

9. Explain what happens when the input frequency is set at 10 kHz, 20 kHz, 30 kHz, etc.

3. Quantization

1. Set the signal generator 250 Hz and a 5 v peak-to-peak output. Set the time base to 1 ms/div. You should see a since wave. Press the 1 key. This sets the quantization level to 1 bit. The sine wave becomes a square wave. Is the frequency of the square wave the same as that of the sine wave ?

2. Set the quantization level to 2 bits by pressing the 2 key. How many levels do we have to represent the input signal ? Draw the waveform for future reference.

3. Continue increasing the quantization levels up to the maximum of 8 bits. Draw the wave forms for each quantization levels.

4. Digital Filters

4.1 How to run a real time digital filter in the DSK

1. You need to work out the specification of the filter, then design the filter, convert the filter coefficients to suitable representation.

2. Use a text editor to edit your program.

3. The source file (file_name.asm) must be assembled to (file_name.dsk) by using the DOS command: *dsk5a file_name.asm.*

4. Note: the above three steps have been done. Two 85-tap FIR filter have been designed and assembled. All filters have a sampling frequency of 10 kHz. The low pass and high pass filter have a cut-off frequency of 2 kHz. They are: *demo3_lp.dsk* and *demo3_hp.dsk*.

5. Run the C50 debugger by using: *dsk5d.*

6. Type LD to load the file (file_name.dsk).

7. Press enter to select the non-fast mode.

8. Press enter again to return to the debugger display.

9. Type XG to run the program.

10. Press ESC to stop the DSP.

4.2 Experiment setup

1. Set the frequency of the signal generator to 50 Hz.

2. Connect the signal generator output to one trace of the scope and the input of the DSK. Connect the output of the DSK to another trace of the scope.

4.3 The low pass filter

1. Run the low pass filter. Observe both traces on the scope. Increase the frequency of the input waveform slowly. Around 2 kHz, the output should begin to attenuate. The cut-off frequency is at a point where the amplitude of the output waveform is about half that of the input waveform. Change the frequency of the input signal from 50 Hz to 4.5 kHz, take amplitude readings of the output signal at intervals of 250 Hz. Plot the frequency response of the filter.

4.4 The high pass filter

1. Repeat the same steps for the high pass filter. Plot the frequency response of the filter.

20 Appendix-III Sample exam questions

1. Answer the following questions

1.1 Determine the convolution $y(n) = x_1(n) * x_2(n)$,where

$$x_1(n) = \delta(n) + 2\delta(n-1) + \delta(n-2)$$

and

$$x_2(n) = \delta(n-1) + \delta(n) + \delta(n+1)$$

.

(5 marks)

1.2 Determine the Z-transform of the following two signals

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

and

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

(5 marks)

1.3 The output *y(n)* of a system is related with the input *x(n)* by the following equation

$$y(n) = ax(n) + b$$

where a and b are non-zero constants. Is this system an LTI system ?

(5 marks)

1.4 Determine the discrete time signal whose Fourier transform is

$$X(\omega) = \frac{1}{3}e^{-j\omega} + \frac{2}{3}e^{j2\omega} + 1$$

(5 marks)

1.5 Determine if the following two LTI systems are stable:

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

and

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

(5 marks)

2. Answer the following questions

2.1 Briefly describe the frequency response of the following types of IIR filters

- Butterworth filter,
- Chebyshev filter, and
- Elliptic filter

(9 marks)

2.2 Draw a block diagram of a general system which uses digital signal processing to process continuous time signals.

(5 marks)

2.3 The sequence, $x(n) = \cos(\frac{\pi}{7}n), \quad -\infty < n < \infty$ **, was obtained by sampling an analog signal** $x_c(t) = \cos(\Omega t), \quad -\infty < t < \infty$ **, at a sampling rate of 2000 samples/second. What are two possible values of** Ω **that could have resulted in the same x(n) ?**

(6 marks)

2.4 An analog signal whose band width is 8 kHz is processed by an analog low pass filter whose cutoff frequency is 3 kHz. If this signal processing task is performed by a digital signal processing system, determine the minimum sampling frequency and the corresponding cutoff frequency of the digital low pass filter.

(5 marks)

3. Answer the following questions about the system whose system function is

$$H(z) = \frac{1 + 0.8z^{-1}}{1 - 0.9z^{-1}}$$

3.1 Determine the poles and zeros of H(z). Is this filter a low pass filter or a high pass filter ? Explain your answer in terms of the locations of the pole and zero.

(6 marks)

3.2 Determine the difference equation relating the input and output of this filter.

(2 marks)

3.3 Draw a direct form-I and a direct form-II implementation of this filter.

(6 marks)

3.4 Determine the impulse response of this filter. (Hint: the Z-transform of the signal $a^n u(n), a < 1$**) is** $1/(1 - az^{-1})$ **)**

(6 marks)

3.5 Determine the filter output *y(n)* for the input signal $x(n) = \delta(n) + \delta(n-1)$ **.**

(5 marks)

4. The following diagram depicts a cascade connection of two LTI systems

4.1 Suppose that system 1 and system 2 both have the same impulse response of the form $h_1(n) = h_2(n) = \delta(n) + \delta(n-1) + \delta(n-2) + \delta(n-3)$ **Determine the system function** $H_1(z)$ **.**

(3 marks)

4.2 Determine the system function of the overall system and obtain a single difference equation that relates y(n) with x(n) in the cascade system.

(7 marks)

4.3 Show that $H_1(z)$ can be expressed as the following.Plot the poles and zeros of this system function. Draw a direct form-II implementation of this system function.

$$H_1(z) = \frac{1 - z^{-4}}{1 - z^{-1}}$$

(15 marks)