OBJECTIVES:

- To demonstrate the concept of digital filter design.
- To use a MATLAB Filter design and Analysis Tool to design and analyze digital filters.
- To know the use of filters in noise filtering.

Theoretical Background

Filtering is one of the most useful signal processing operations. Digital signal processors are now available to implement digital filters in real time. The TMS320C6x instruction set and architecture makes it well suited for such filtering operations. An analog filter operates on continuous signals and is typically realized with discrete components such as operational amplifiers, resistors, and capacitors. However, a digital filter, such as a finite impulse response (FIR) filter, operates on discrete-time signals and can be implemented with a digital signal processor such as the TMS320C6x. This involves use of an ADC to capture an external input signal, processing the input samples, and sending the resulting output through a DAC. Within the last few years, the cost of digital signal processors has been reduced significantly, which adds to the numerous advantages that digital filters have over their analog counterparts. These include higher reliability, accuracy, and less sensitivity to temperature and aging.

Stringent magnitude and phase characteristics can be realized with a digital filter. Filter characteristics such as center frequency, bandwidth, and filter type can readily be modified. A number of tools are available to design and implement within a few minutes an FIR filter in real time using the TMS320C6x-based DSK. The filter design consists of the approximation of a transfer function with a resulting set of coefficients. Different techniques are available for the design of FIR filters, such as a commonly used technique that utilizes the Fourier series, as discussed in later on.

General difference equation for any digital system:

\[ y(n) + a_1y(n - 1) + a_2y(n - 2) + \cdots + a_Ny(n - (M - 1)) = b_0x(n) + b_1x(n - 1) + b_2x(n - 2) + \cdots + b_Mx(n - (N - 1)) \]

\[ y(n) = \sum_{k=1}^{M-1} -a_ky(n - k) + \sum_{k=0}^{N-1} b_kx(n - k) \]
The impulse response is given by:

\[ h(n) = \sum_{k=1}^{M-1} -a_k h(n-k) + \sum_{k=0}^{N-1} b_k \delta(n-k) \]

\[ h(n) = \sum_{k=1}^{M-1} -a_k h(n-k) + \sum_{k=0}^{N-1} b_k \delta(n-k) \]

\[ H(z) = -\sum_{k=1}^{M-1} a_k z^{-k} H(z) + \sum_{k=0}^{N-1} b_k z^{-k} \]

Which give a transfer function of a form:

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

For FIR Filter, \( h(n) = \{b_0, b_1, \ldots, b_{N-1}\} \) or \( H(z) = \sum_{k=0}^{N-1} b_k z^{-k} \)

then, \( h(n) = \sum_{k=0}^{N-1} b_k \delta(n-k) = \sum_{k=0}^{N-1} h(k) \delta(n-k) \)

\[ h(n) = h(0) \delta(n) + h(1) \delta(n-1) + h(2) \delta(n-2) + \cdots + h(N-1) \delta(n-(N-1)) \]

\[ h(n) = \sum_{k=0}^{N-1} h(k) \delta(n-k) \]

but, \( y(n) = x(n) \ast h(n) = x(n) \ast \sum_{k=0}^{N-1} h(k) \delta(n-k) \)

\[ y(n) = \sum_{k=0}^{N-1} h(k) x(n-k) \delta(n-k) \]

\[ y(n) = \sum_{k=0}^{N-1} h(k) x(n-k) \quad \ldots \quad \ldots \quad \ldots \quad (10.1) \]
The convolution equation (10.1) is very useful for the design of FIR filters, since we can approximate it with a finite number of terms.

If the input is a unit impulse \( x(n) = \delta(0) \), the output impulse response will be \( y(n) = h(n) \). We will see how to design an FIR filter with \( N \) coefficients \( h(0), h(1), \ldots, h(N - 1) \), and \( N \) input samples \( x(n), x(n - 1), \ldots, x(n - (N - 1)) \).

The input sample at time \( n \) is \( x(n) \), and the delayed input samples are \( x(n - 1), \ldots, x(n - (N - 1)) \). Equation (10.1) shows that an FIR filter can be implemented with knowledge of the input \( x(n) \) at time \( n \) and of the delayed inputs \( x(n - k) \). It is non-recursive and no feedback or past outputs are required. Filters with feedback (recursive) that require past outputs are Infinite Impulse Response (IIR).

Other names used for FIR filters are transversal and tapped-delay filters.

We can write equation (10.1) as following:

\[
y(n) = h(0)x(n) + h(1)x(n - 1) + h(2)x(n - 2) + \cdots + h(N - 1)x(n - (N - 1))
\]

The \( z \)-transform of (10.1) with zero initial conditions yields

\[
Y(z) = h(0)X(z) + h(1)z^{-1}X(z) + h(2)z^{-2}X(z) + \cdots + h(N - 1)z^{-(N-1)}X(z)
\]  
\[\cdots \cdots \cdots \text{(10.2)}\]

Equation (10.1) represents a convolution in time between the coefficients and the input samples, which is equivalent to a multiplication in the frequency domain, or

\[
Y(z) = H(z)X(z)
\]  
\[\cdots \cdots \cdots \text{(10.3)}\]

Where \( H(z) = ZT [h(k)] \) is the transfer function, or

\[
H(z) = h(0) + h(1)z^{-1} + h(2)z^{-2} + \cdots + h(N - 1)z^{-(N-1)}
\]

\[
= \frac{h(0)z^{(N-1)} + h(1)z^{N-2} + h(2)z^{N-3} + \cdots + h(N-1)}{z^{N-1}}
\]  
\[\cdots \cdots \cdots \text{(10.4)}\]

which shows that there are \( N - 1 \) poles, all of which are located at the origin. Hence, this FIR filter is inherently (always) stable, with its poles located only inside the unit circle.

We usually describe an FIR filter as a filter with “no poles.” Figure 10.1 shows an FIR filter structure representing (10.1) and (10.2).
A very useful feature of an FIR filter is that it can guarantee linear phase. The linear phase feature can be very useful in applications such as speech analysis, where phase distortion can be very critical. For example, with linear phase, all input sinusoidal components are delayed by the same amount. Otherwise, harmonic distortion can occur.

The Fourier transform of a delayed input sample \( x(n - k) \) is \( e^{-j\omega k T} X(j\omega) \), yielding a phase of \( \theta = -\omega k T \), which is a linear function in terms of \( \omega \). Note that the group delay function, defined as the derivative of the phase, is a constant, or \( d\theta/d\omega = -kT \).

**Window Design Techniques**

The basic idea behind the window design is to choose a proper ideal frequency-selective filter (which always has a noncausal, infinite-duration impulse response) and then truncate (or window) its impulse response to obtain a linear-phase and casual FIR filter.

Consider an ideal LPF which has a unity magnitude gain and linear-phase characteristics over its passband, and zero response over its stopband.

\[
H_d(\omega) = \begin{cases} 
1 \cdot e^{-j\alpha \omega}, & |\omega| \leq \omega_c \\
0, & \omega_c < |\omega| \leq \pi
\end{cases}
\]

Where \( \omega_c \) is called the cutoff frequency, and \( \alpha \) is called the sample delay (\( e^{-j\alpha \omega} \) implies shift in the positive \( n \) direction or delay).

The impulse response of this filter is of infinite duration and is given by

\[
h_d(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(\omega) e^{j\omega n} d\omega = \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} e^{-j\alpha \omega} e^{j\omega n} d\omega = \frac{\sin(\omega_c (n-\alpha))}{\pi (n-\alpha)}
\]

Note that \( h_d(n) \) is symmetric with respect to \( \alpha \), a fact useful for linear-phase FIR filters.

To obtain an FIR from \( h_d(n) \), one has truncated \( h_d(n) \) on both sides. To obtain a causal and linear-phase FIR filter \( h(n) \) of length \( M \), we must have

\[
h(n) = \begin{cases} 
h_d(n), & 0 \leq n \leq M - 1 \\
0, & \text{elsewhere}
\end{cases}
\]

This operation is called "windowing."
In general: \[ h(n) = h_d(n)w(n) \] , \( w(n) \) is a window function.

Rectangular window
This is the simplest window function but provides the worst performance from the viewpoint of stopband attenuation. It is defined by:

\[
w(n) = \begin{cases} 
1, & 0 \leq n \leq M - 1 \\
0, & \text{elsewhere}
\end{cases}
\]

And its frequency response function

\[
W(\omega) = \sum_{n=-\infty}^{\infty} w(n)e^{-j\omega n} = \sum_{n=0}^{M-1} e^{-j\omega n} = \sum_{n=0}^{M-1} (e^{-j\omega})^n = \frac{1 - e^{-j\omega N}}{1 - e^{-j\omega}}
\]

\[
= \frac{e^{-j\omega N/2} - e^{-j\omega N/2} \sin(\omega M)}{e^{-j\omega/2} - e^{-j\omega/2}}
\]

\[
|W(\omega)| = \frac{\sin(\omega M/2)}{\sin(\omega/2)}
\]

In frequency domain the causal FIR filter response \( H(\omega) \) is given by the periodic convolution of \( H_d(\omega) \) and the window response \( W(\omega) \), that is (as shows in Figure 10.2)

\[ H(\omega) = H_d(\omega) \otimes W(\omega) \]

**FIGURE 10.2.** Windowing operation in frequency domain
From the observation of plots in figure 10.3 we can make several observations.

**Figure 10.3.** Rectangular window: M=45

1- The amplitude response \( W(\omega) \) has the first zero at \( \omega = \omega_1 \), where

\[
\frac{\omega_1 M}{2} = \pi \quad \text{or} \quad \omega_1 = \frac{2\pi}{M}
\]

Hence the width of the main lobe is \( 2\omega_1 = \frac{4\pi}{M} \). Therefore the approximate transient bandwidth is \( \frac{4\pi}{M} \).

2- The magnitude of the first side lobe (which is also the peak side lobe magnitude) is approximately at \( \omega = \frac{3\pi}{M} \) and is given by:

\[
\left| W\left(\frac{3\pi}{M}\right)\right| = \left| \frac{\sin\left(\frac{3\pi}{M}\frac{M}{2}\right)}{\sin\left(\frac{3\pi}{M}\frac{3\pi}{2}\right)} \right| = \left| \frac{\sin\left(\frac{3\pi}{2}\right)}{\sin\left(\frac{3\pi}{M}\frac{3\pi}{2}\right)} \right| \approx \frac{2M}{3\pi} \quad \text{for} \ M \gg 1
\]

Comparing this with the main lobe amplitude, which is equal to \( M \), the peak side lobe magnitude is:
$\frac{2}{3\pi} = 21.22\% = 13$ dB of the main lobe amplitude.

3- The amplitude response of $H(\omega)$ has the first side lobe magnitude at 21 dB. This results in the minimum stopband attenuation of 21 dB irrespective of the window length M.

This is a simple window operation in the time domain and an easy function to analyze in the frequency domain. However, there are two main problems. First, the minimum stopband attenuation of 21 dB is insufficient in practical applications. Second, the rectangular windowing suffers from the Gibbs phenomenon.

If we increase $M$, the width of each side lobe will decrease, but the area under each lobe will remain constant. Therefore the relative amplitudes of side lobes will remain constant, and the minimum stopband attenuation will remain constant, and the minimum stopband attenuation will remain at 21 dB. This implies that all ripples will bunch up near the band edges.

FIGURE 10.4. Gibbs phenomenon.

Since the rectangular window is impractical in many applications, we consider other window functions.
**Bartlett Window**

Bartlett suggested a more gradual transition in the form of a triangle window, which is given by

\[
w(n) = \begin{cases} 
\frac{2n}{M-1}, & 0 \leq n \leq \frac{M-1}{2} \\
2 - \frac{2n}{M-1}, & \frac{M-1}{2} \leq n \leq M-1 \\
0, & \text{otherwise}
\end{cases}
\]

This window and its frequency domain response are shown in figure 10.5

![Bartlett Window](image)

**FIGURE 10.5.** Bartlett (triangular) window : M=45

**Hanning Window**

This is a raised cosine window with function given by

\[
w(n) = \begin{cases} 
0.5 \left[1 - \cos \left(\frac{2\pi n}{M-1}\right)\right], & 0 \leq n \leq M-1 \\
0, & \text{otherwise}
\end{cases}
\]

This window and its frequency domain response are shown in figure 10.6
Hamming Window

This window is similar to the Hanning window except that it has a small amount of discontinuity and is given by

\[ w(n) = \begin{cases} 
0.54 - 0.46 \cos \left( \frac{2\pi n}{M-1} \right), & 0 \leq n \leq M - 1 \\
0, & \text{otherwise}
\end{cases} \]

This window and its frequency domain response are shown in figure 10.7

---

FIGURE 10.6. Hanning window : M=45

FIGURE 10.7. Hamming window : M=45
**Blackman Window**

This window is also similar to the previous two but contains a second harmonic term and is given by

\[
   w(n) = \begin{cases} 
   0.42 - 0.5 \cos \left( \frac{2\pi n}{M-1} \right) + 0.08 \cos \left( \frac{4\pi n}{M-1} \right), & 0 \leq n \leq M - 1 \\
   0, & \text{otherwise} 
   \end{cases}
\]

This window and its frequency domain response are shown in figure 10.8.

**FIGURE 10.8.** Blackman window : M=45

Note Hamming window appears to be the best choice for many applications.

**Kaiser Window**

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

\[
   w(n) = \frac{I_0 \left[ \beta \sqrt{1 - \left( 1 - \frac{2n}{M-1} \right)^2} \right]}{I_0 [\beta]}, \quad 0 \leq n \leq M - 1
\]

This window can provide different transition widths for the same M, which is something other windows lack. For example:

- If \( \beta = 5.658 \), then the transition width is equal to \( 7.8\pi/M \) and the minimum stopband attenuation is equal to 60 dB. This is shown in Figure 10.9.
If $\beta = 4.538$, then the transition width is equal to $5.8\pi/M$ and the minimum stopband attenuation is equal to 50 dB.

Hence the performance of this window is comparable to that of Hamming window. In addition, the Kaiser window provides flexible transition bandwidths.

**FIGURE 10.9.** Kaiser window: $M=45, \beta = 5.658$

---

**FIR design by MATLAB:**

The Filter Design and Analysis Tool (FDATool) is a powerful user interface for designing and analyzing filters. FDATool enables you to quickly design digital FIR or IIR filters by setting filter performance specifications, by importing filters from your MATLAB workspace, or by directly specifying filter coefficients. FDATool also provides tools for analyzing filters, such as magnitude and phase response plots and pole-zero plots. This background contains the following topics:

1. **Opening the Filter Design and Analysis Tool**

To open the Filter Design and Analysis Tool, type `fdatool`, The Filter Design and Analysis Tool opens in the default design mode.
2. Choosing a Filter Type

You can choose from several filter types:

- Lowpass
- Highpass
- Bandpass
- Bandstop
- Differentiator
- Hilbert transformer
- Multiband
- Arbitrary magnitude
- Arbitrary group delay
- Raised cosine

For example, to design a bandpass filter, select the radio button next to Bandpass in the Filter Type region of the GUI.

3. Choosing a Filter Design Method

You can use the default filter design method for the filter type that you’ve selected, or you can select a filter design method from the available FIR and IIR methods listed in the GUI. To select the Remez algorithm to compute FIR filter coefficients, select the FIR radio button and choose Equiripple from the list of methods.
You can also use the **Method** menu to select a filter design method.

4. Setting the Filter Design Specifications

The filter design specifications that you can set vary according to filter type and design method. For example, to design a bandpass filter, you can enter:

- Bandpass Filter Frequency Specifications
- Bandpass Filter Magnitude Specifications
- Filter Order

The display region illustrates filter specifications when you select **Filter Specifications** from the **Analysis** menu.

### Bandpass Filter Frequency Specifications

For a bandpass filter, you can set:

- Units of frequency: (Hz, kHz, MHz)
- Normalized (0 to 1)
- Sampling frequency
- Passband frequencies
- Stopband frequencies

You specify the passband with two frequencies. The first frequency determines the lower edge of the passband, and the second frequency determines the upper edge of the passband. Similarly, you specify the stopband with two frequencies. The first frequency determines the upper edge of the first stopband, and the second frequency determines the lower edge of the second stopband.

For this example:

- Keep the units in Hz (default).
- Set the sampling frequency (Fs) to 2000 Hz.
- Set the end of the first stopband (Fstop1) to 200 Hz.
- Set the beginning of the passband (Fpass1) to 300 Hz.
- Set the end of the passband (Fpass2) to 700 Hz.
- Set the beginning of the second stopband (Fstop2) to 800 Hz.
Bandpass Filter Magnitude Specifications

For a bandpass filter, you can specify the following magnitude response characteristics:

- Units for the magnitude response (dB or linear)
- Passband ripple
- Stopband attenuation

For this example:
- Keep the units in **dB** (default).
- Set the passband ripple (**A_{pass**}) to 0.1 dB.
- Set the stopband attenuation for both stopbands (**A_{stop1}, A_{stop2}**) to 75 dB.

Filter Order

You have two mutually exclusive options for determining the filter order when you design an equiripple filter

- **Minimum order**: The filter design method determines the minimum order filter.
- **Specify order**: You enter the filter order in a text box.

Select the **Minimum order** radio button for this example.

Note that filter order specification options depend on the filter design method you choose. Some filter methods may not have both options available.

5. Computing the Filter Coefficients

Now that you’ve specified the filter design, select the **Design Filter** button to compute the filter coefficients. Notice that the **Design Filter** button is dimmed once you’ve computed the coefficients for your filter design. This button is enabled again once you make any changes to the filter specifications.

6. Analyzing the Filter

Once you’ve designed the filter, you can view the following filter response characteristics in the display region or in a separate window:

- Magnitude response
- Phase response
- Overlaid magnitude and phase responses
- Group delay
• Impulse response
• Step response
• Pole-zero plot
You can also display the filter coefficients in this region.
You can access the analysis methods as items in the Analysis menu, or by using the toolbar buttons.

For example, to look at the filter’s magnitude response, select the **Magnitude Response** button on the toolbar.

![Magnitude Response](image)

To display the filter response characteristics in a separate window, select **Full View Analysis** from the Analysis menu. The Filter Visualization Tool (fvtool) opens. You can use this tool to annotate your design, view other filter analyses, and print your filter response.

![Full View Analysis](image)

You can click on the response to add a data marker that displays information about the particular point on the response. Right-clicking displays a menu where you can adjust the appearance of the data markers or delete them.

**7. Saving and Opening Filter Design Sessions**

You can save your filter design session as a MAT-file and return to the same session another time. Select the **Save session** button to save your session as a MAT-file. The first time you save a session, a **Save Filter Design File** browser opens, prompting you for a session name. For example, save this design session as TestFilter.fda.
in your current working directory by typing TestFilter in the **File name** field. The .fda extension is added automatically to all filter design sessions you save.

You can load existing sessions into the Filter Design and Analysis Tool by selecting the **Open session** button. A **Load Filter Design File** browser opens that allows you to select from your previously saved filter design sessions.

To export filter coefficients to MATLAB workspace, go to File>Export, enter the numerator variable name (Num) and click OK.

Go to workspace and type: Num; you should see the filter coefficients printed.

**Application Exercise: Filtering a noise signal from a sinusoidal signal**

A sinusoidal signal having $f_0=75\text{Hz}$ and amplitude of $1\text{v}$ is sampled at a rate of $1000$ sample/sec. The discrete time sinusoidal is transmitted through a twisted pair cable. During its transmission, the signal is corrupted by a noise signal. The noise signal can be visualized as uniformly distributed random numbers with amplitude ranging from $0$ to $4\text{v}$ and sampled at a rate of $1000$ samples/sec. This random signal will be filtered to remove the noise signal in the band above $250$ Hz. At the receiver side, we need to isolate the sinusoidal signal from the noisy signal.

a) Using simulink, design a model that can generate the noise signal described in the above problem.
b) Suggest the type of the filter that can be used at the receiver side.
c) Specify the parameters of the filter you choose in part b.
d) Try to visualize the input signal, corrupted signal and the reconstructed signal on the same figure.