

**Daltech, Dalhousie University**  
**Department of Electrical and Computer Engineering**  
**ECED 4502 – Digital Signal Processing**

**Lab 3 –Finite Impulse Response Filtering in MatLab**

- I. For this lab you need the DSP System and Data Acquisition Toolboxes installed on your MatLab. To check installed toolboxes in your MatLab type “ver” in your command window.

Please see Appendix B learn how to install it if you do not have it.

- II. MatLab 2018 release or newer versions is recommended.

### **Objectives**

The objectives of this lab are to observe the characteristics of Finite Impulse Response (FIR) filters and to design your own FIR filters in MatLab.

In the first section, Background, you will become familiar with using the Filter Designer toolbox. In the second section, an audio signal is generated by the signal generator and acquired using your computer sound card.

You are expected to design different FIR filters with specified parameters, comment on your design and answer questions about it.

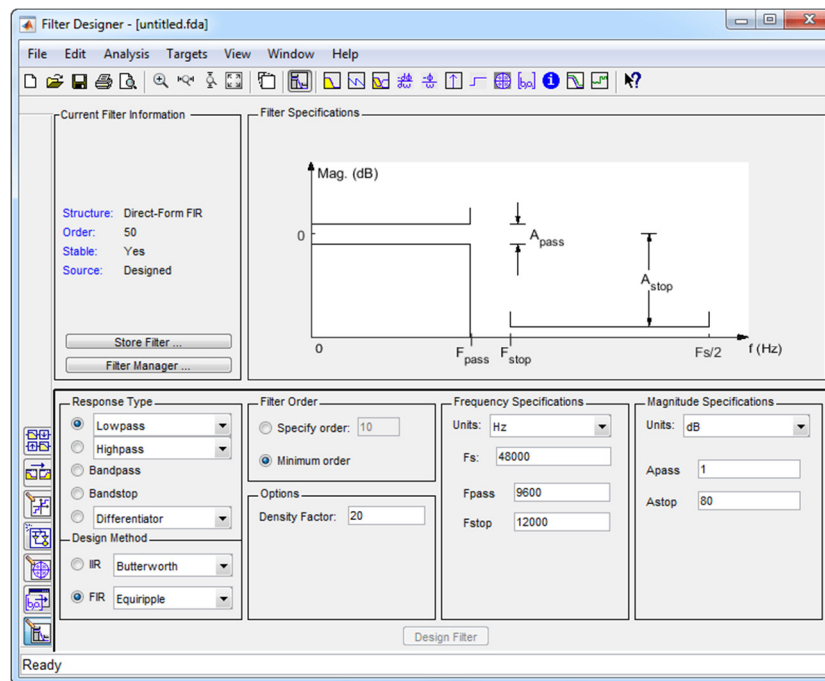
#### **1- Background: Using Filter Designer**

---

To open filter designer, type

filterDesigner

at the MATLAB<sup>®</sup> command prompt. The filter designer opens with the Design filter panel displayed below.



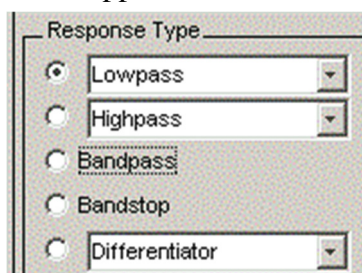
Note that when you open filter designer, **Design Filter** is not enabled. You must make a change to the default filter design in order to enable **Design Filter**. This is true for each time you want to change the filter design. Changes to radio button items or drop down menu items such as those under **Response Type** or **Filter Order** enable **Design Filter** immediately. Changes to specifications in text boxes such as **F<sub>s</sub>**, **F<sub>pass</sub>**, and **F<sub>stop</sub>** require you to click outside the text box to enable **Design Filter**.

## Choosing the Frequency Response Type

You can choose from several response types:

- Lowpass
- Raised cosine
- Highpass
- Bandpass
- Bandstop
- Differentiator

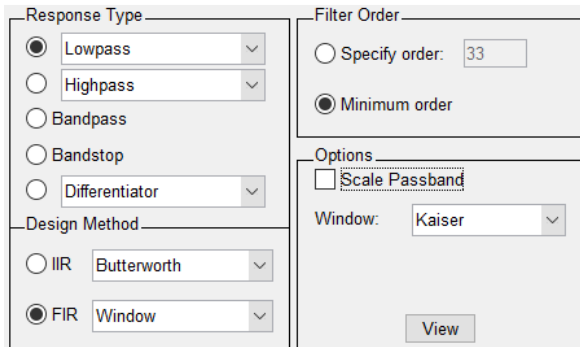
To design a Lowpass filter, select the radio button next to Lowpass in the Response Type region of the app, as shown below.



## Choosing a Filter Design Method

---

You can select the FIR windowing method as shown below.



The screenshot shows a dialog box for filter design with the following settings:

- Response Type:** Lowpass (selected), Highpass, Bandpass, Bandstop, Differentiator.
- Design Method:** IIR (Butterworth), **FIR (Window)** (selected).
- Filter Order:** Specify order: 33, **Minimum order** (selected).
- Options:** Scale Passband (unchecked), Window: Kaiser (selected).
- View** button.

### *Viewing Filter Specifications*

The filter design specifications that you can set vary according to response type and design method. The display region illustrates filter specifications when you select **Analysis > Filter Specifications** or when you click the **Filter Specifications** toolbar button.

You can also view the filter specifications on the Magnitude plot of a designed filter by selecting **View > Specification Mask**.

### *Filter Order*

You have two mutually exclusive options when it comes to the filter order:

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

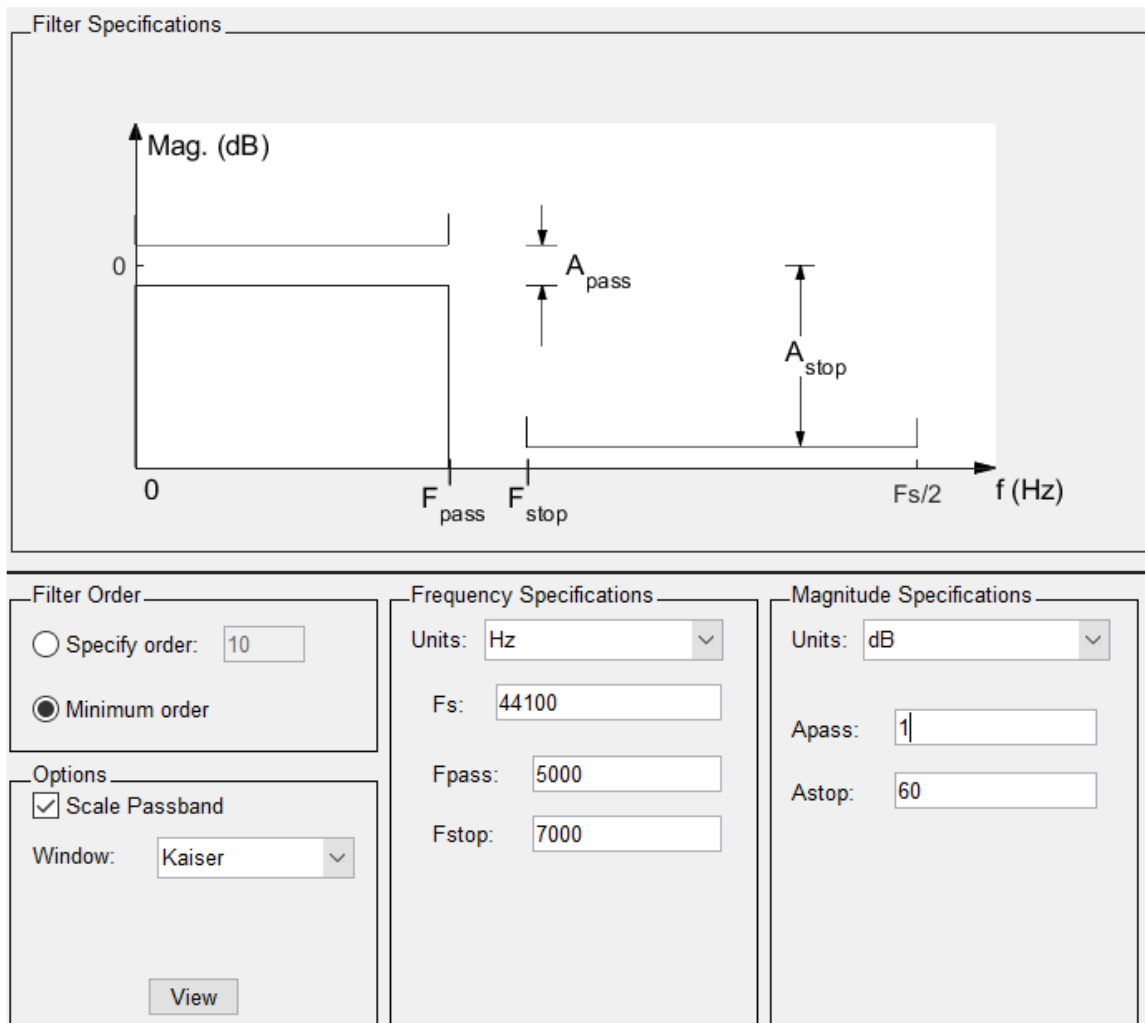


Figure 1. Minimum order filter

- The parameters of your filter are given in the “**Frequency Specification**” part of the screen where you can select:
- the filter type – you can try first the low-pass filter.
- the order of your filter (the higher the order, the better approximation of the ideal filter you will get)
- the cut-off frequency of your filter for the passband  $F_p$  and the cut-off frequency of your filter for the stopband  $F_s$  – both of them should be always smaller than  $f_s/2$  with  $f_s$  sampling frequency selected in the **Acquire** part.

Once the filter is designed you can observe some of its characteristics by using one of the buttons along the top, as shown in figure 2 below.

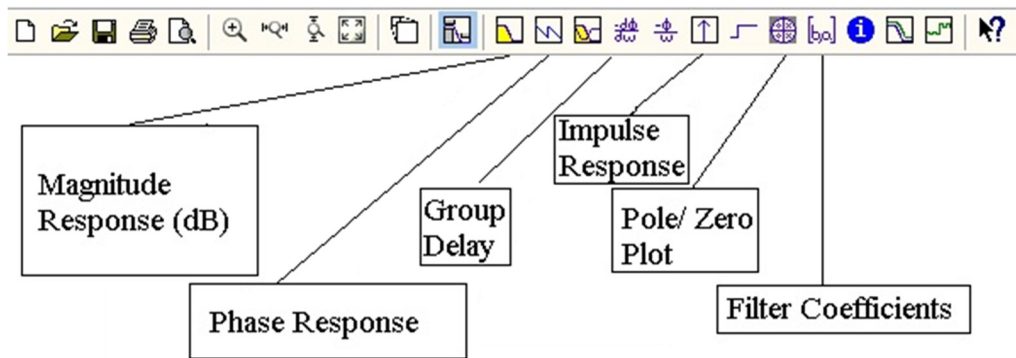


Figure 2. Design output buttons

Now that you are familiar with the Filter Design toolbox, we acquire an audio tone from the signal generator

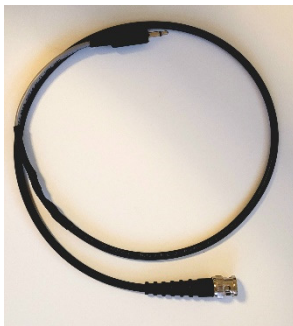


Figure 3. Custom Cable



Figure 4. A 4 kHz sinusoidal signal generation

## 2. Signal Generator setup

In this part, you will set the signal generator to generate the required waveform. First, before connecting the custom cable to the signal generator, create a 4 kHz sinusoidal waveform with a 200 mV peak to peak signal. Then use the custom BNC to 3.5 mm audio jack to connect the signal generator to the MIC input on your computer. You can also connect the other port of the BNC splitter to the oscilloscope to see the signal generator output on the oscilloscope screen. A built-in protection diode inside the custom cable clamps voltages over 300 mV peak to peak to safeguard the audio card input. However, to avoid any damage to the sound card avoid voltages higher than 200 mV peak to peak and make sure that the signal generator is set to 200 mV peak to peak before applying the signal to the sound card MIC.

### 3. Audio Signal Acquisition

Here we will use the *Digital\_filter\_input.m* script to acquire an audio signal from the signal generator. Open the script in the MatLab and run the code.

Enter the sampling frequency for the acquired signal using the audio card. Remember that the audio card can sample with the maximum rate of 44100 Hz, then enter an arbitrary frame size, a frame size of 1024 samples is recommended for better visualization of the signal.

After signal acquisition you can see the generated plots for the sampled signal as well as its spectrum. Then, press enter in the command window to open the filter design toolbox.

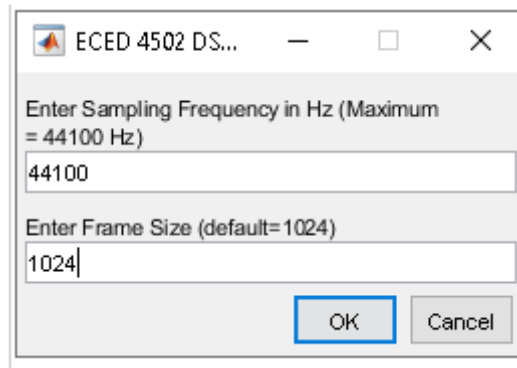


Figure 5. Signal Acquisition Dialogue Box

### 4. Filter Design

Change the filter type to the lowpass filter. Change the sampling rate to the sampling rate entered in section 3 and the cutoff frequency  $F_C$  to **5,000 Hz**. Choose a filter order, say **33** taps, and a shape factor  $\beta$  corresponding to the Hann window, this is **3.86** (see Figure 6 next). This lab focus will be on the FIR filter design using the Kaiser window.

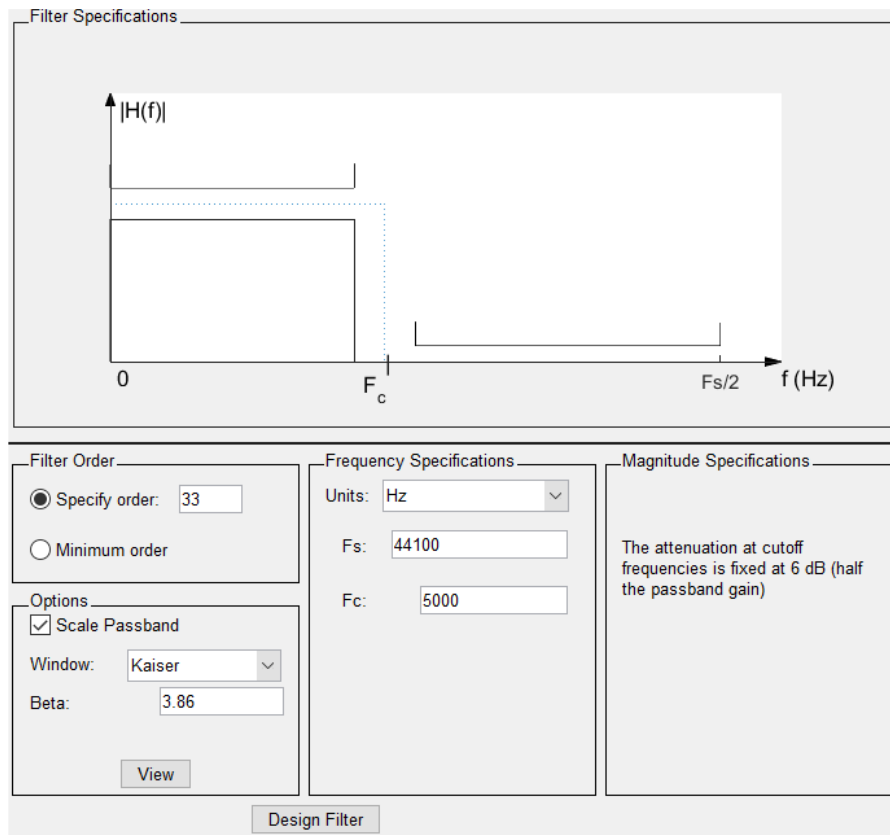


Figure 6. Approximation of ideal LP filter specifying window shape and length (order)

Once you have obtained the filter coefficients by clicking on the Design Filter button, make sure to record your filter frequency response for later comparison. A detailed view of it can be obtained, after the design is done, by clicking the Full View Analysis button, see Figure 7 below.



Figure 7. Full View Analysis button

Next, go to **File** and export these filter coefficients to the workspace, as indicated in Figure 8 below. The filters coefficients show up in your workspace as parameter Num after you press Enter in the command window.

After pressing Enter in the workspace the designed filter will be applied to the sampled signal input from the generator. You can see the time domain and frequency domain plots of the input signal and the filtered version.

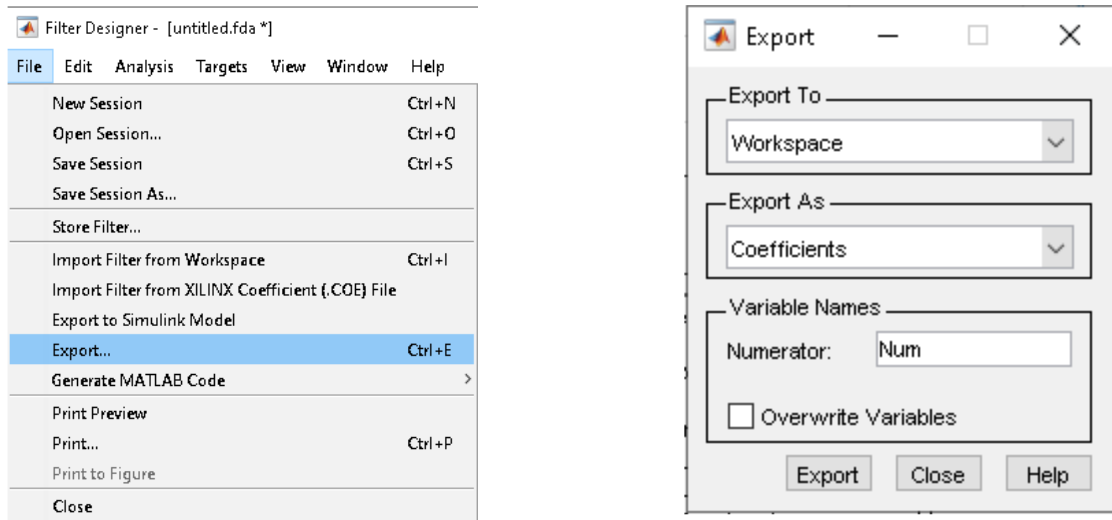


Figure 8. Exporting filter coefficients

## 5. Procedure

### Part 1

- A) In order to verify the frequency response obtained using the filter parameters in Figure 6, page 7, we would like to acquire sinusoidal signals of **4, 5 & 6 kHz**. Obtain the filter outputs and record their amplitudes.
- B) We would like to repeat **A)** for filter orders not only of 33 as indicated in Figure 6, page 7, but also for filter orders of **9** and **5** taps.
- C) Repeat **A)** for values of  $\beta$  equivalent to the Rectangular (**0**) and the Blackman (**7.04**) windows.

### Part 2

- A) In order to verify the frequency response obtained using the filter specifications shown in Figure 1, page 4, we would like to acquire sinusoidal signals of **5, 6 & 7 kHz**. Obtain the filter outputs and record their amplitudes.
- B) We would like to repeat **A)** for stopband attenuation values  $A_{stop}$  of not only of 60 dB as indicated in Figure 1, but also for stopband attenuation values  $A_{stop}$  of **40** and **80** dB.
- C) Repeat **A)** & **B)**, this is  $A_{stop}$  taking values of 40, 60 & 80 dB for an stopband corner frequency  $F_{stop}$  of **6,000** Hz instead of 7,000 Hz as shown in Figure 1.



## 6. Discussion

In your lab report please comment on the following:

1. (a) Your observations resulting from comparing the filtered signal amplitude and the input signal amplitude for the different frequencies 4, 5, 6 & 7 kHz, depending on the Part of the Procedure.  
  
(b) How did different number of taps in Part 1.B) affect the filtered signal amplitudes?  
  
(c) How does the number of taps affect the frequency response of your FIR filters?
  
2. (a) How does the value of  $\beta$ , in Part 1.C), affects the amplitude of the filtered outputs?  
  
(b) Which factors contribute to the differences observed for the 3 values of  $\beta$  used:  
  - i) the passband/stopband ripple,
  - ii) the transition band width, or
  - iii) both?  
Explain
  
3. Increasing the attenuation of the stopband  $A_{\text{stop}}$  in Parts 2.A) & 2.B) of the Procedure, results in more, less or the same number of filter coefficients? Explain why.
  
4. (a) How reducing the stopband corner frequency  $F_{\text{stop}}$  in Part 2.C) affected the number of filter coefficients required? Explain  
  
(b) Which frequency component, 5, 6 or 7 kHz, was affected the most by reducing the stopband corner frequency  $F_{\text{stop}}$  from 7 to 6 kHz in Part 2.C)?
  
5. In which part of the Procedure, Part 1, Part 2, or both, are we designing a filter as discussed in classes? Explain.

## Appendix A

### Downloading the DSP System Toolbox

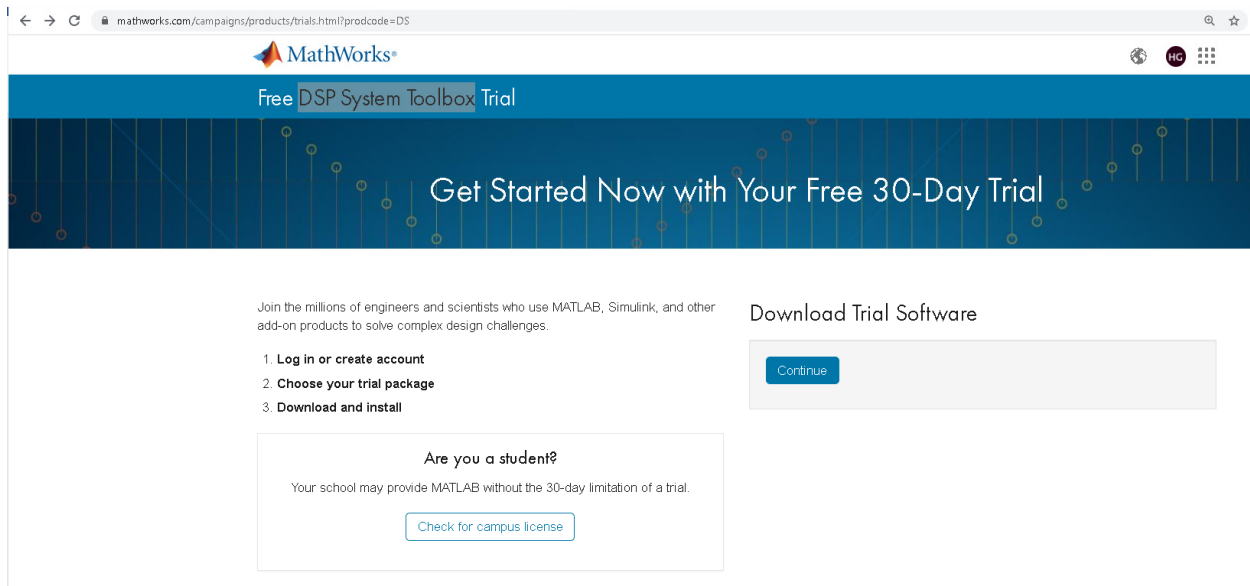
With DSP System Toolbox you can design and analyze FIR, IIR filters.

<https://www.mathworks.com/products/dsp-system.html>

You can stream signals from variables, data files, and network devices for system development and verification. The Time Scope, Spectrum Analyzer, and Logic Analyzer let you dynamically visualize and measure streaming signals. For desktop prototyping and deployment to embedded processors, including ARM® Cortex® architectures, the system toolbox supports C/C++ code generation. It also supports bit-accurate fixed-point modeling and HDL code generation from filters, FFT, IFFT, and other algorithms.

Use the following link to download the toolbox

<https://www.mathworks.com/campaigns/products/trials.html?prodcode=DS>



If you do not have the Academia license. Click on **Check for the campus license**.

← → ↻ mathworks.com/academia/tah-support-program/eligibility.html

MathWorks®

Academia

## Campus-Wide License

See if your school has a MATLAB campus license

The Campus-Wide License offers an effective way for students, faculty, and researchers to get access to a comprehensive set of MATLAB and Simulink products. To find out if your school is covered under a Campus-Wide License, complete the form below.

\* Indicates required information

Information

\* University

Dalhousie University

Enter the official name.

\* Email

Use your official university email address, which is required for license verification.

This field is required.

Submit

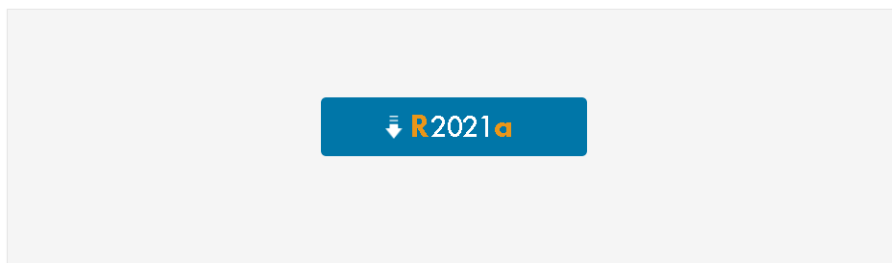
After entering your email you will receive an email to add Dalhousie license into your Mathworks account

After getting the academic license choose your MatLab release version

[FAQ](#) | [Download & Install Troubleshooting](#)

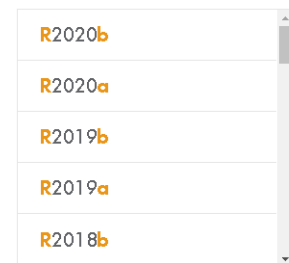
[Contact support](#)

Choose trial release



R2021a  
Released: 2021-03-10  
[System Requirements](#)  
[Release Highlights](#)

Choose earlier release



Then choose DSP System Toolbox corresponding to your MatLab version. From the download page

← → C mathworks.com/downloads/web\_downloads/select\_products?dl\_action=download\_installer&platform=win64&release\_name=R2021a&tab=f

MathWorks®

Get MATLAB HG

## Downloads

FAQ | Download & Install Troubleshooting

Contact support

### Trial R2021a Products

Sort by: Category Platform: Windows

- ☐ Polyspace Code Prover Server (10.4)
- ☐ Polyspace Client for Ada (6.21)
- ☐ Polyspace Server for Ada (6.21)
- Signal Processing**
  - ☐ Signal Processing Toolbox (8.6)
  - ☐ Phased Array System Toolbox (4.5)
  - ☐ Radar Toolbox (1.0)
  - ☒ DSP System Toolbox (9.12)
  - ☐ Audio Toolbox (3.0)
  - ☐ Wavelet Toolbox (5.6)
- Image Processing and Computer Vision**
  - ☐ Image Processing Toolbox (11.3)

**Trial R2021a Products**

In the next page choose your operating system and download the installer

MathWorks®

Get MATLAB HG

## Downloads

FAQ | Download & Install Troubleshooting

Contact support

### Download R2021a

#### Download and run the Installer

Windows

macOS

Linux

- When prompted, sign in as h.ghannadrezai@dal.ca
- Select license id 8726418
- Choose the products, toolboxes, and blocksets that you want to install

#### Getting Started Using Your Software Trial

Thank you for requesting a trial!

Your trial for these products expires in **30 days**.

If this is your first time using MATLAB, you may wish to check out the following resources available from our Web site:

- Getting Started with MATLAB
- Getting Started with Simulink
- Documentation (Help Desk)

#### Learn MATLAB Now

Learn core MATLAB functionality with this free, interactive, self-paced course.

» Get Started

#### Discover Live Editor

Create scripts with code, output, and formatted text in a single executable document.

» Learn About Live Editor

## Appendix B

### Introduction

A digital filter is a basic building block in any Digital Signal Processing (DSP) system. The frequency response of the filter depends on the value of its coefficients, or taps. Many software programs can compute the values of the coefficients based on the desired frequency response. In our course, the students are using Labview programs to obtain the filter coefficients. These values are typically floating point numbers and they are represented with a fairly high degree of precision. However, when a digital filter is implemented in hardware, such as the TI TMS320C26 boards as it is a case in the second part of this laboratory experiment, the designer wants to represent the coefficients (and also the data) with the smallest number of bits that still gives acceptable resolution for the numbers. This is because representing a number with excess bits increases the size of the registers, buses, adders, multipliers and other hardware used to process that signal. The bigger sizes result in a chip with a larger die size, which translates into increased power consumption and a higher chip price. Thus, the bit precisions used to represent numbers are important in the performance of real-world signal processing designs.

### Filtering with FIR systems

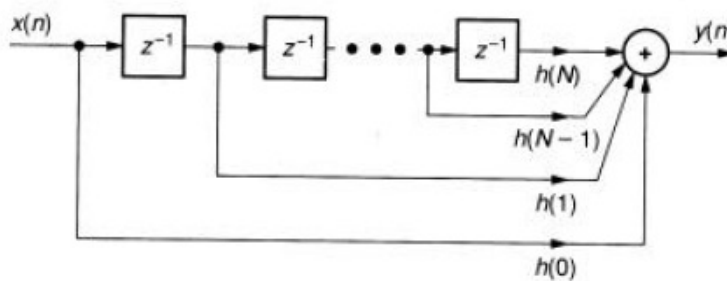
A Finite Impulse Response (FIR) digital filter is one whose impulse response is of finite duration. This can be stated mathematically as

$$h[n] = 0 \Rightarrow n < 0 \text{ or } n > N$$

where  $h[n]$  denotes the impulse response of the digital filter,  $n$  is the discrete time index, and  $N$  is the length of the filter. A difference equation is the discrete time equivalent of a continuous time differential equation. The general difference equation for a FIR digital filter is

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

where  $y(n)$  is the filter output at discrete time instance  $n$ ,  $h[k]$  is the  $k$ th feedforward tap, or filter coefficient, and  $x[n-k]$  is the filter input delayed by  $k$  samples. The  $\sum_{k=0}^N$  denotes summation from  $k = 0$  to  $k = N$  where  $N$  is the number of feedforward taps in the FIR filter. The calculations of the output sample in terms of the input samples and the  $h[k]$  coefficients are represented in the figure below with the  $z^{-1}$  representing the delays (of one sampling period):



**Figure 3:** FIR structure

Note that the FIR filter output depends only on the previous  $N$  inputs. This feature is why the impulse response for a FIR filter is finite.

When the input to a FIR filter is the unit impulse  $\delta[n]$ , the output of the filter is the impulse response  $h[n]$ .

## Advantages of FIR filters

FIR filters are simple to design and they are guaranteed to be bounded input-bounded output (BIBO) stable. By designing the filter taps to be symmetrical about the center tap position, a FIR filter can be guaranteed to have linear phase. This is a desirable property for many applications such as music and video processing. FIR filters also have a low sensitivity to filter coefficient quantization errors. This is an important property to have when implementing a filter on a DSP processor or on an integrated circuit.

## FIR Filter Design by Windowing

The most commonly used techniques for design of infinite impulse response (IIR) filters are based on transformations of continuous-time systems into discrete-time IIR systems. This is partly because continuous-time filter design was a highly advanced art before discrete-time filters were of interest and partly because of the difficulty of implementing a noniterative direct design method for IIR filters.

In contrast, FIR filters are almost entirely restricted to discrete-time implementations. Consequently the design techniques for FIR filters are based on directly approximating the desired frequency response of the discrete-time system. Furthermore, most techniques for approximating the magnitude response of an FIR system assume a linear phase constraint, thereby avoiding the problem of spectrum factorization that complicates the direct design of IIR filters.

The simplest method of FIR filter design is called the window method. This method generally begins with an ideal desired frequency response that can be represented as

$$H_d(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} h_d[n] e^{j\omega n} \quad (0)$$

where  $h_d[n]$  is the corresponding impulse response sequence, which can be expressed in terms of  $H_d(e^{j\omega})$  as

$$h_d[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(e^{j\omega}) \cdot e^{-j\omega n} d\omega \quad (1)$$

Many idealized systems are defined by piecewise-constant or piecewise-functional frequency responses with discontinuities at the boundaries between bands. As a result, they have impulse responses that are noncausal and infinitely long. The most straightforward approach to obtaining a causal FIR approximation to such systems is to truncate the ideal response. Equation (1) can be thought of as a Fourier series representation of the periodic frequency response  $H_d(e^{j\omega})$ , with the sequence  $h_d[n]$  playing the role of the Fourier coefficients. Thus, the approximation of an ideal filter by truncation of the ideal impulse response is identical to the issue of the convergence of Fourier series, a subject that has received a great deal of study. A particularly important concept from this theory is the Gibbs phenomenon, which will be covered in the example with this Appendix.

The simplest way to obtain a causal FIR filter from  $h_d[n]$  is to define a new system with impulse response  $h[n]$  given by

$$h[n] = \begin{cases} h_d[n], & 0 \leq n \leq M \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

More generally, we can represent  $h[n]$  as the product of the desired impulse response and a finite-duration "window"  $w[n]$ ; i.e.,

$$h[n] = h_d[n] \cdot w[n] \quad (3)$$

where for simple truncation as in Eq. (2), the window is the rectangular window



$$w[n] = \begin{cases} 1, \dots, 0 \leq n \leq M \\ 0, \dots, \text{otherwise} \end{cases} \quad (4)$$

It follows from the windowing theorem that

$$H(e^{j\omega}) = \frac{1}{2 \cdot \pi} \cdot \int_{-\pi}^{\pi} H_d(e^{j\theta}) \cdot W(e^{j(\omega-\theta)}) d\omega \quad (5)$$

That is,  $H(e^{j\omega})$  is the periodic convolution of the desired ideal frequency response with the Fourier transform of the window. Thus, the frequency response  $H(e^{j\omega})$  will be a "smeared" version of the desired response  $H_d(e^{j\omega})$ . Figure 2(a) depicts typical functions  $H_d(e^{j\omega})$  and  $W(e^{j\omega})$  as required in Eq.(5).

If  $w[n] = 1$  for all  $n$  (i.e., we do not truncate at all),  $W(e^{j\omega})$  is a periodic impulse train with period  $2 \cdot \pi$ , and therefore  $H_d(e^{j\omega}) = H(e^{j\omega})$ .

This interpretation suggests that if  $w[n]$  is chosen so that  $W(e^{j\omega})$  is concentrated in a narrow band of frequencies around  $\omega=0$ , then  $H(e^{j\omega})$  will "look like"  $H_d(e^{j\omega})$  except where  $H_d(e^{j\omega})$  changes very abruptly. Thus, the choice of window is governed by the desire to have  $w[n]$  as short as possible in duration so as to minimize computation in the implementation of the filter while having  $W(e^{j\omega})$  approximate an impulse; that is, we want  $W(e^{j\omega})$  to be highly concentrated in frequency so that the convolution of Eq. (5) faithfully reproduces the desired frequency response. These are conflicting requirements, as can be seen in the case of the rectangular window of Eq. (4), where

$$W(e^{j\omega}) = \sum_{n=0}^M h_d e^{-j\omega n} = \frac{1 - e^{-j\omega(M+1)}}{1 - e^{-j\omega}} = e^{-j\omega M/2} \cdot \frac{\sin(\omega \cdot (M+1)/2)}{\sin(\omega/2)} \quad (6)$$

The magnitude of  $W(e^{j\omega})$  is plotted in Fig. 8 for the case  $M = 7$ . Note that  $W(e^{j\omega})$  for the rectangular window has generalized linear phase. As  $M$  increases, the width of the "main lobe" decreases. The main lobe is usually defined as the region between the first zero crossings on either side of the

origin. For the rectangular window, the mainlobe width is  $\Delta\omega_m = \frac{4 \cdot \pi}{M+1}$ . However, for the rectangular window, the sidelobes are significantly large and in fact, as  $M$  increases, the peak amplitudes of the mainlobe and the sidelobes grow in a manner such that the area under each lobe is a constant while the width of each lobe decreases with  $M$ . Consequently, as  $W(e^{j(\omega-\theta)})$  "slides by" a discontinuity of  $H_d(e^{j\omega})$  with increasing  $\omega$ , the integral of  $W(e^{j(\omega-\theta)}) \cdot H_d(e^{j\omega})$  will oscillate as each sidelobe of  $W(e^{j(\omega-\theta)})$  moves past the discontinuity. This result is depicted in Fig.7(b). Since the area under each lobe remains constant with increasing  $M$ , the oscillations occur more rapidly but do not decrease in amplitude as  $M$  increases.

In the theory of Fourier series, it is well known that this nonuniform convergence, the Gibbs phenomenon, can be moderated through the use of a less abrupt truncation of the Fourier series by tapering the window smoothly to zero.

**Figure 7** (a) Convolution process implied by truncation of the ideal impulse response.  
(b) Typical approximation resulting from windowing the ideal impulse response.

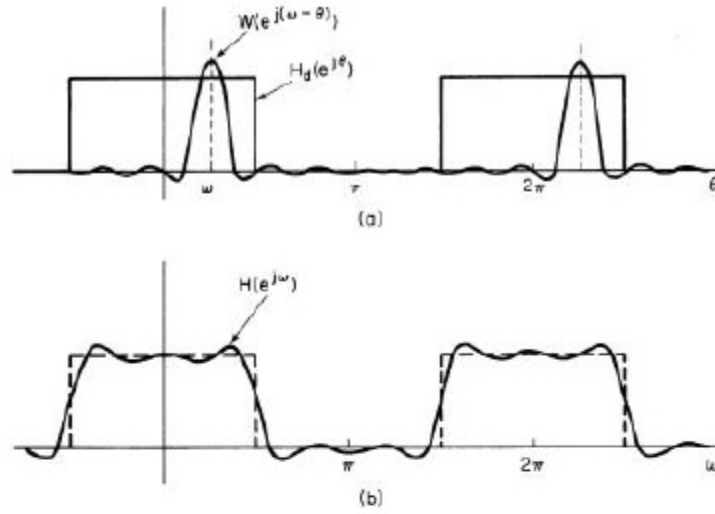
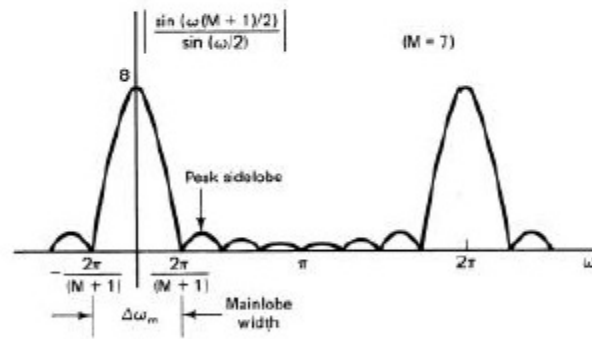


Figure 8 Magnitude of the Fourier transform of a rectangular window ( $M = 7$ ).



## Design Example

The desired frequency response is defined as

$$H_{lp}(e^{j\omega}) = \begin{cases} e^{-j\omega \cdot M/2}, & |\omega| \leq \omega_c \\ 0, & \omega_c < |\omega| \leq \pi \end{cases} \quad (86)$$

where the generalized linear phase factor has been incorporated into the definition of the ideal lowpass filter. The corresponding ideal impulse response is

$$h_{lp}[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-j\omega \cdot M/2} \cdot e^{j\omega \cdot n} d\omega = \frac{\sin(\omega_c \cdot (n - M/2))}{\pi \cdot (n - M/2)} \quad (87)$$



for  $-\infty < n < +\infty$ . It is easily shown that  $h_p[M - n] = h_p[n]$ , so if we use a symmetric window in the equation

$$h[n] = \frac{\sin(\omega_c \cdot (n - M/2))}{\pi \cdot (n - M/2)} \cdot w[n] \quad (88)$$

then a linear phase system will result.

The upper part of Fig. 31 depicts the character of the amplitude response that would result for the typical windows used in your design. Figure 31 displays the important properties of window method approximations to desired frequency responses that have step discontinuities. It applies accurately when  $\omega_c$  is not close to zero or to  $\pi$  and when the width of the mainlobe is smaller than  $2 \cdot \omega_c$ . At the bottom of the figure is a typical Fourier transform for a symmetric window. This function should be visualized in different positions as an aid in understanding the shape of the approximation  $H(e^{j\omega})$  in the vicinity of  $\omega_c$ .

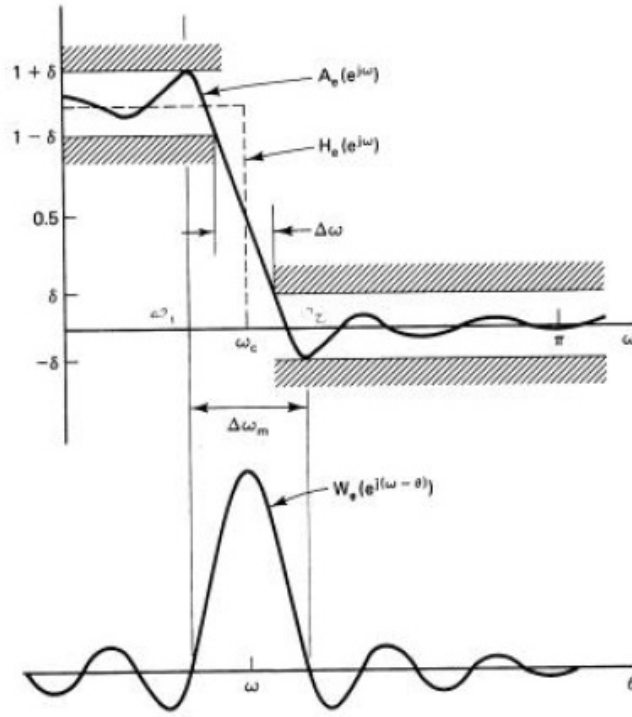


Figure 31 Illustration of type of approximation obtained at a discontinuity of the ideal frequency response.