Filter Design and Analysis using FDATool of MATLAB

The Filter Design and Analysis Tool (FDATool) is a powerful user interface for designing and analyzing filters quickly. FDATool enables you to design digital FIR or IIR filters by setting filter specifications, by importing filters from your MATLAB workspace, or by adding, moving or deleting poles and zeros. FDATool also provides tools for analyzing filters, such as magnitude and phase response and pole-zero plots. FDATool seamlessly integrates additional functionality from other MathWorks products as described in the following table.

Use FDATOOL in matlab.
If you type
>>fdatool
in command window, FDAtool will be opened. There you can select FIR or IIR filter, order of filter and cutoff frequency of a filter (either HPF, LPF or BPF). That code will automatically generate .m file for you.

Record your voice. Analyze the effect of lowpass filtering on speech signal. Use the Filter Design and Analysis Tool (FDATool) of MATLAB for the purpose of designing LPF filter. Consider separately FIR and IIR filter.

A detail demo of the FDATool is available in the website of The Mathworks which I recommend you to visit for acquiring more knowledge in Matlab.

http://www.mathworks.com/products/demos/shipping/signal/introfdatooldemo.html?product=SG

Introduction to the Filter Design and Analysis Tool (FDATool)

The Filter Design and Analysis Tool (FDATool) is a powerful graphical user interface (GUI) in the Signal Processing Toolbox™ 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 adding, moving or deleting poles and zeros. FDATool also provides tools for analyzing filters, such as magnitude and phase response plots and pole-zero plots.

You can use FDATool as a convenient alternative to the command line filter design functions.

Contents

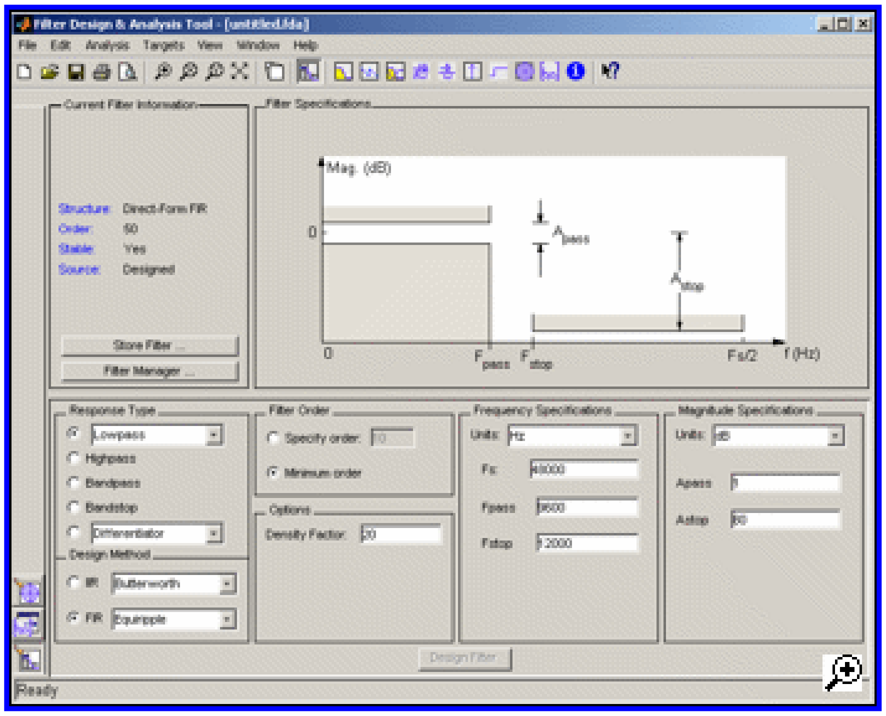
- Getting Started
- Designing a Filter
- Viewing other Analyses
- Comparing the Design to Filter Specifications
- Changing Axes Units
- Marking Data Points
- Optimizing the Design
- Using a Different Filter Structure
- Changing Analyses Parameters
- Exporting the Filter
- Generating an M-File
- Quantizing a Filter
- Targets
- Additional Features

Getting Started

Type fdatool at the MATLAB command prompt:

```
>>fdatool
```

A **Tip of the Day** dialog displays with suggestions for using FDATool. Then, the GUI diplays with a default filter.



Structure: Direct-Form FIR
Order: 50
Stable: Yes
Source: Designed

Response Type: Lowpass
Design Method: FIR Equiripple
Filter Order: Minimum order
Density Factor: 20

Units: Hz
Fs: 48000
Fpass: 9600
Fstop: 12000

Units: dB
Apass: 1
Astop: 80

The GUI has three main regions:

- The Current Filter Information region
- The Filter Display region and
- The Design panel

The upper half of the GUI displays information on filter specifications and responses for the current filter. The Current Filter Information region, in the upper left, displays filter properties, namely the filter structure, order, number of sections used and whether the filter is stable or not. It also provides access to the Filter manager for working with multiple filters.

The Filter Display region, in the upper right, displays various filter responses, such as, magnitude response, group delay and filter coefficients.

The lower half of the GUI is the interactive portion of FDATool. The Design Panel, in the lower half is where you define your filter specifications. It controls what is displayed in the other two upper regions. Other panels can be displayed in the lower half by using the sidebar buttons.

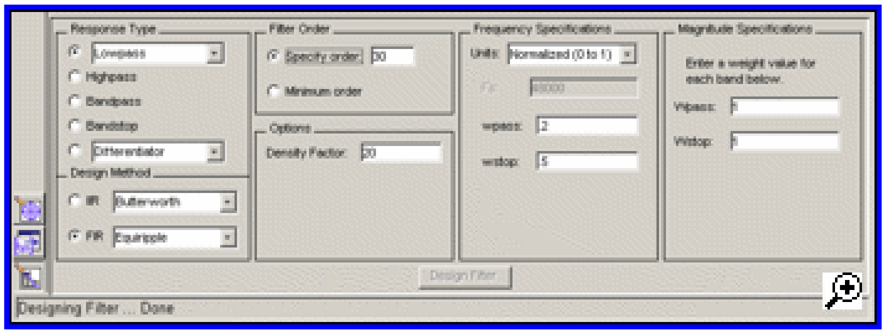
The tool includes Context-sensitive help. You can right-click or click the **What's This?** button to get information on the different parts of the tool.

Designing a Filter

We will design a low pass filter that passes all frequencies less than or equal to 20% of the Nyquist frequency (half the sampling frequency) and attenuates frequencies greater than or equal to 50% of the Nyquist frequency. We will use an FIR Equiripple filter with these specifications:

- Passband attenuation 1 dB
- Stopband attenuation 80 dB
- A passband frequency 0.2 [Normalized (0 to 1)]
- A stopband frequency 0.5 [Normalized (0 to 1)]

To implement this design, we will use the following specifications:



Response Type: Lowpass [✓] / Highpass / Bandpass / Bandstop / Differentiator
Design Method: IIR Butterworth / FIR Equiripple [✓]
Filter Order: Specify order [✓] 30 / Minimum order
Options: Density Factor: 20
Frequency Specifications: Units: Normalized (0 to 1); Fs: 48000; wpass: .2; wstop: .5
Magnitude Specifications: Enter a weight value for each band below. Wpass: 1; Wstop: 1
Designing Filter ... Done

1. Select **Lowpass** from the dropdown menu under **Response Type** and **Equiripple** under **FIR Design Method**. In general, when you change the Response Type or Design Method, the filter parameters and Filter Display region update automatically.

2. Select **Specify order** in the **Filter Order** area and enter **30**.

3. The FIR Equiripple filter has a **Density Factor** option which controls the density of the frequency grid. Increasing the value creates a filter which more closely approximates an ideal equiripple filter, but more time is required as the computation increases. Leave this value at 20.

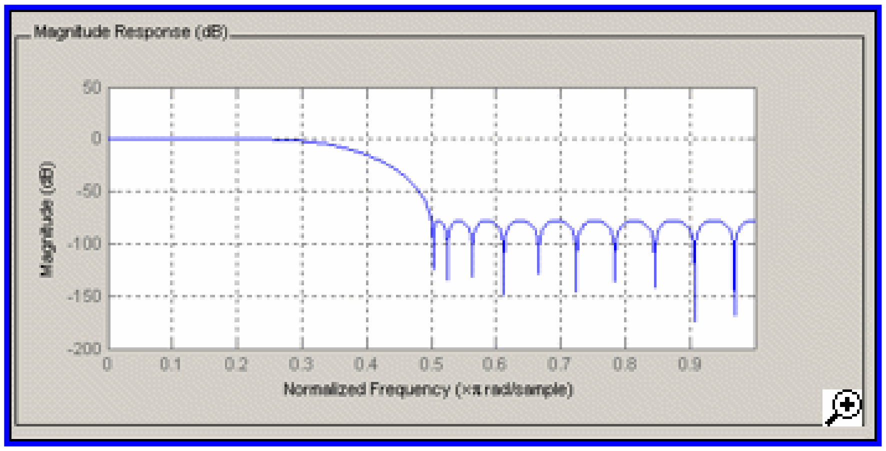
4. Select **Normalized (0 to 1)** in the Units pull down menu in the **Frequency Specifications** area.

5. Enter **0.2** for **wpass** and **0.5** for **wstop** in the **Frequency Specifications** area.

6. **Wpass** and **Wstop**, in the **Magnitude Specifications** area are positive weights, one per band, used during optimization in the FIR Equiripple filter. Leave these values at 1.

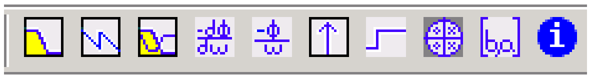
7. After setting the design specifications, click the **Design Filter** button at the bottom of the GUI to design the filter.

The magnitude response of the filter is displayed in the Filter Analysis area after the coefficients are computed.



Viewing other Analyses

Once you have designed the filter, you can view the following filter analyses in the display window by clicking any of the buttons on the toolbar:



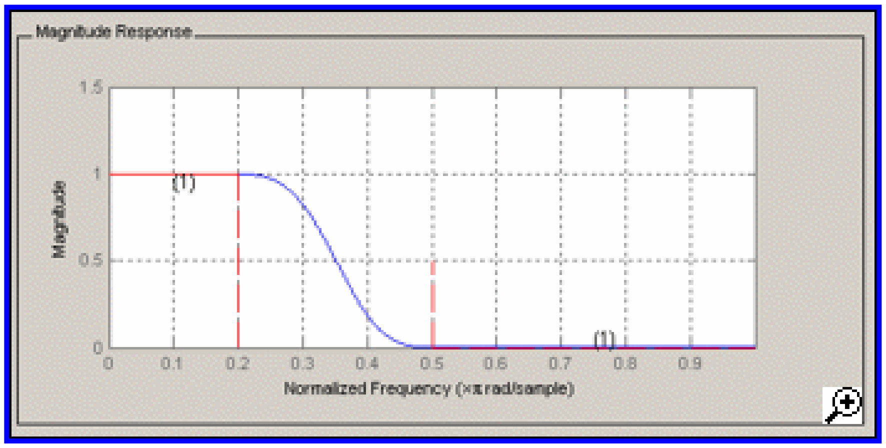
In order from left to right, the buttons are

- Magnitude response
- Phase response
- Magnitude and Phase responses
- Group delay response
- Phase delay response
- Impulse response
- Step response
- Pole-zero plot
- Filter Coefficients
- Filter Information

Comparing the Design to Filter Specifications

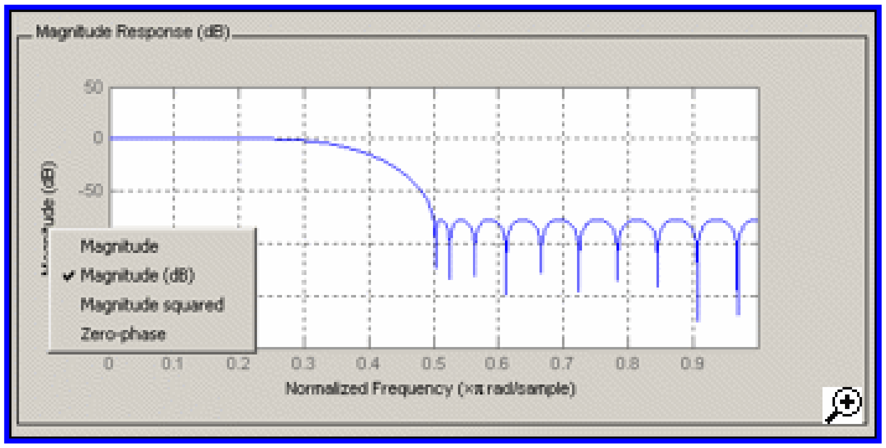
FDATool allows you to measure how closely your design meets the filter specifications by using Specification masks which overlay the filter specifications on the response plot. In the Display Region, when the Magnitude plot is displayed, select **Specification Mask** from the **View** menu to overlay the filter specifications on the response plot.

The magnitude response of the filter with Specification mask is shown below:



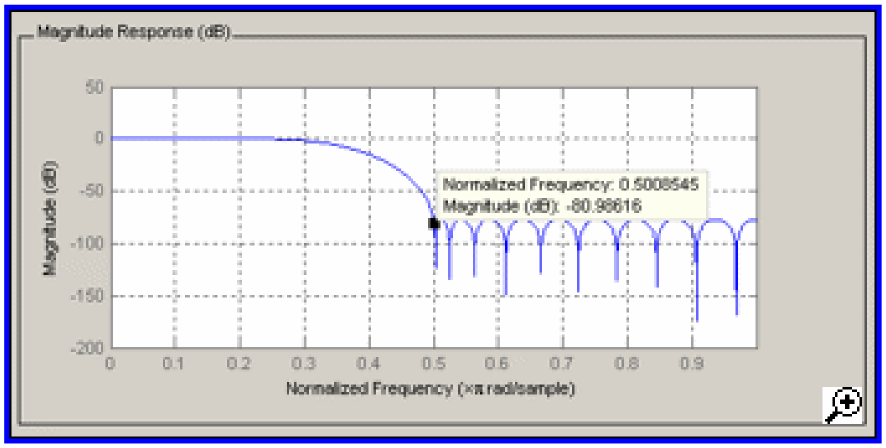
Changing Axes Units

You can change the x- or y-axis units by right-clicking the mouse on an axis label and selecting the desired units. The current units have a checkmark.



Marking Data Points

In the Display region, you can click on any point in the plot to add a data marker, which displays the values at that point. Right-clicking on the data marker displays a menu where you can move, delete or adjust the appearance of the data markers.

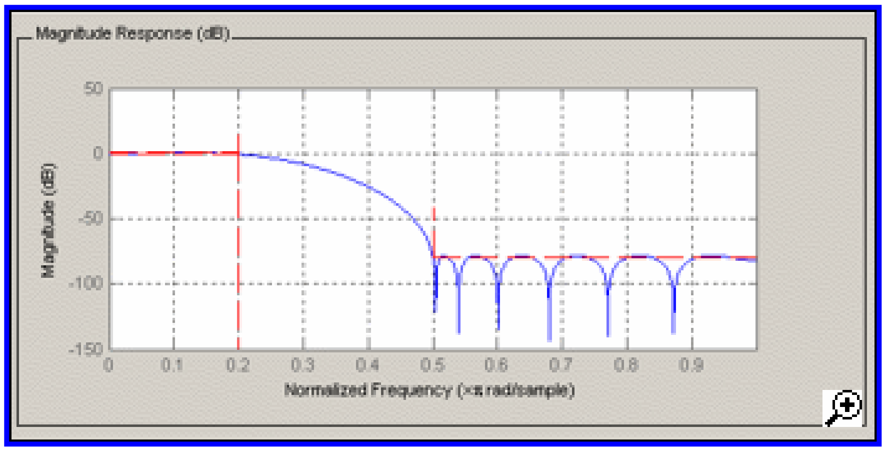


Optimizing the Design

To minimize the cost of implementation of the filter, we will try to reduce the number of coefficients by using **Minimum Order** option in the design panel.

Change the selection in **Filter Order** to **Minimum Order** in the Design Region and leave the other parameters as they are.

Click the **Design Filter** button to design the new filter.



As you can see in the Current Filter Information area, the filter order decreased from 30 to 16, the number of ripples decreased and the transition width became wider. The passband and the stopband specifications still meet the design criteria.

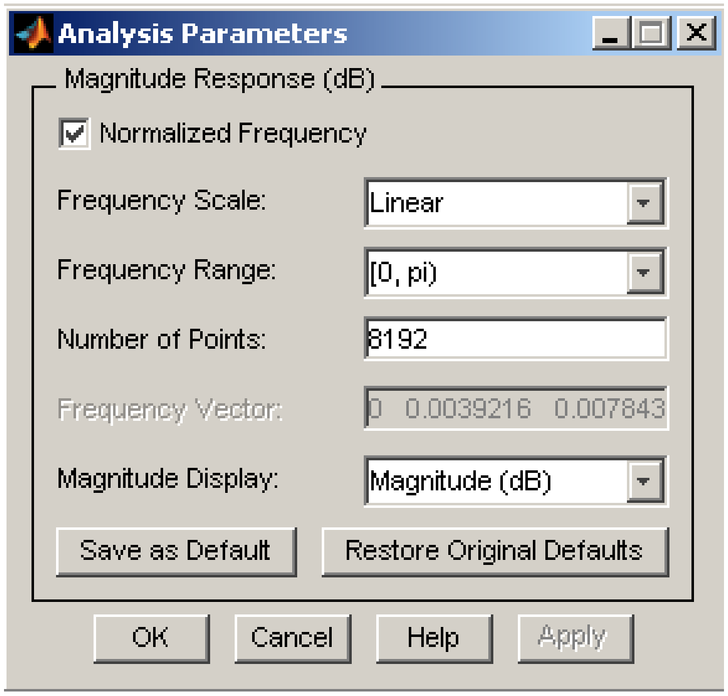
Using a Different Filter Structure

Our filter is a Direct-form FIR. Typically, the Direct-Form FIR transposed structure is implemented in hardware. You can use Convert Structure dialog from the Edit menu to change the current filter to a new structure. Filters can be converted to the following representations:

- State-Space
- Direct-Form FIR
- Direct-Form FIR Transposed
- Direct-Form Symmetric FIR

Changing Analyses Parameters

By right-clicking on the plot and selecting Analysis Parameters, you can display a dialog box for changing analysis-specific parameters. (You can also select Analysis Parameters from the Analysis menu.)



✓ Normalized Frequency	
Frequency Scale:	Linear
Frequency Range:	[0, pi)
Number of Points:	8192
Frequency Vector:	0 0.0039216 0.007843
Magnitude Display:	Magnitude (dB)

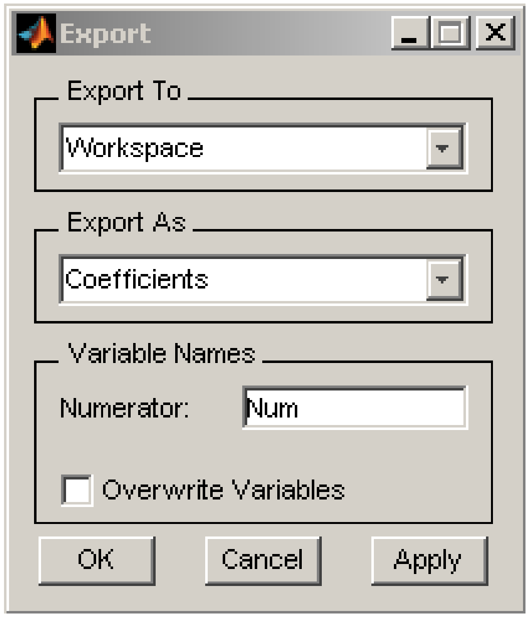
To save the displayed parameters as the default values, click **Save as Default**. To restore the MATLAB-defined default values, click **Restore Original Defaults**.

Exporting the Filter

Once you are satisfied with your design, you can export your filter to the following destinations:

- MATLAB workspace
- MAT-file
- Text-file

Select **Export** from the **File** menu.



Export To: Workspace

Export As: Coefficients

Variable Names

Numerator: Num

Overwrite Variables

If exporting to the MATLAB workspace, you can export as coefficients or as an object by selecting from the **Export** from the pulldown menu.

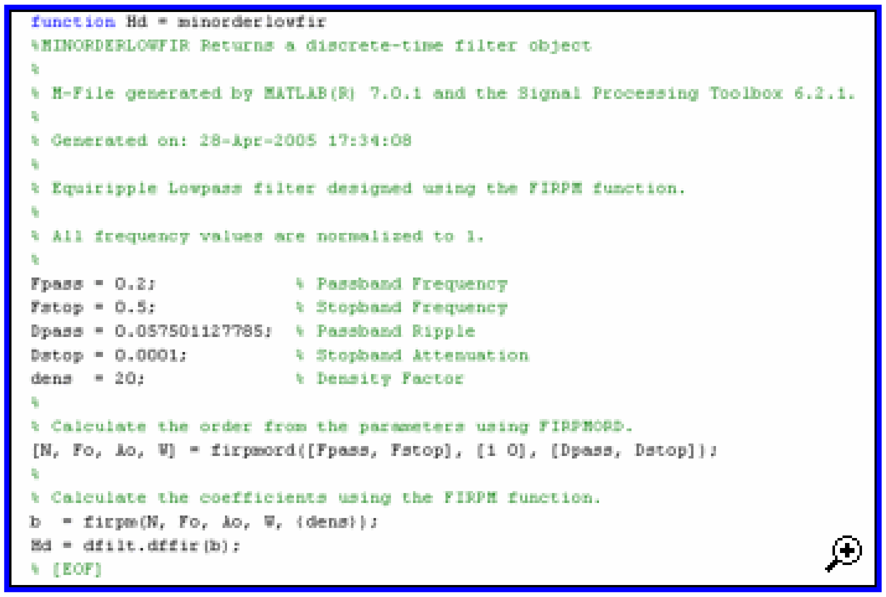
If you want to export as an **object**, the object's properties control many aspects of its apearance and behaviour. You can use **GET** and **SET** commands from the MATLAB command prompt to have access and manipulate the property values of the object.

Generating an M-File

FDATool allows you to generate M-code to re-create your filter. This enables you to embed your design into existing code or automate the creation of your filters in a script.

Select **Generate M-file** from the **File** menu and specify the filename in the Generate M-file dialog box.

The following code was generated from the minimum order filter we designed above:



```
function Hd = minorderlowfir
%MINORDERLOWFIR Returns a discrete-time filter object
%
% M-File generated by MATLAB(R) 7.0.1 and the Signal Processing Toolbox 6.2.1.
%
% Generated on: 28-Apr-2005 17:34:08
%
% Equiripple Lowpass filter designed using the FIRPM function.
%
% All frequency values are normalized to 1.
%
Fpass = 0.2;               % Passband Frequency
Fstop = 0.5;               % Stopband Frequency
Dpass = 0.057501127785;    % Passband Ripple
Dstop = 0.0001;            % Stopband Attenuation
dens  = 20;                % Density Factor
%
% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fpass, Fstop], [1 0], [Dpass, Dstop]);
%
% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);
% [EOF]
```

Quantizing a Filter

If you have the Filter Design Toolbox™ installed, the **Set quantization parameters** panel is available on the sidebar:

You can use this panel to quantize and analyze double-precision filters. With the Filter Design Toolbox you can quantize from double-precision to single-precision. If you have the Fixed Point Toolbox, you can quantize filters to fixed-point precision. Note that you can not mix floating-point and fixed-point arithmatic in your filter.

Targets

The **Targets** menu of the FDATool allows you to generate various types of code representing your filter. For example, you can generate C header files, XILINX coefficients(COE) files (with the Filter Design Toolbox) and VHDL, Verilog along with test benches (with Filter Design HDL Coder™).

Additional Features

FDATool also integrates additional functionality from these other MathWorks products:

- **Embedded Target for Texas Instruments™ C6000™ DSP-** Generates downloadable code for C6000 DSP target board.
- **Filter Design HDL Coder-** Generates synthesizable VHDL or Verilog code for fixed-point filters
- **Filter Design Toolbox-** Adds advanced FIR and IIR design techniques (i.e. Filter transformations,

Multirate filters)

```
% * *Embedded IDE Link(TM) CC Development Tool*- Exports code

%   usable by Code Composser Studio
%
% * *Signal Processing Blockset(TM)*- Generates equivalent Signal Processing
 Blockset
%   block for the filter
%
% * *Simulink(R)*- Generates filters from atomic Simulink blocks
%
```

Chapter 4

Quantization

Quantization

The goal of quantization is to take an input (either a single value or a signal [which is all we shall talk of from now on]) which can take on many values and generate an output which can take on fewer values. The input of a quantizer may be analog (taking on a continuum of values for example any real number between 0 and 1) or it may be discrete (taking only a countable number of values; often only a finite number of values). The output will always be discrete.

The case of an analog input is found inside any non-ideal D/A converter. The case of discrete input are found in any digital computer after for example the multiplication of two 32 bit words to result in a 32 bit product.

Quantization is done by passing the signal through a memoryless non-linearity which is piece wise constant and non-decreasing.

Prelab Quantizer Design

Before coming to the lab design a uniform 8 bit quantizer whose input is a guassian random variable with mean 0 and variance 1. Use a 4σ loading rule.

Your design and the decisions you have made should be written up and handed in with the lab write up.

The mean of the quantization noise should be zero. Explain.

Include in your design calculations the probability that the input will fall in a range where one only has granular noise.

Include in your design calculations the expected quantization noise variance due to granular noise. Take into effect the probability of granular noise.

Include in your design calculations the probability of overload noise.

Include in your design the overall quantization noise including overload and granular noise.

In your calculation and design give details and clearly state any assumptions you have made.

Simulation of Quantizers using Matlab

Write a piece of MATLAB code that implements the quantizer you designed in the prelab. (The MATLAB functions floor, ciel and round may be useful.)

Provide the code you wrote in your write up.

Test your calculations by generating 100,000 gaussian random variables and putting them through the quantizer. Generate a vector which contains all the quantization errors. Count how many times the quantization error falls outside the 4σ area. Does this make sense compared to your calculations above?

What conclusions can you draw about the importance of granular versus overload noice in this example?

Calculate the sample mean and sample variance of your quantization noise. Use the following formula:

$$\bar{E} = \frac{\sum_{i=1}^{N} E_i}{N}$$

Here $N = 100,000$ and E_i is the error from the ith input. $\bar{E}$ is the sample mean.

Also use

$$S^2 = \frac{\sum_{i=1}^{N} (E_i - \bar{E})^2}{N - 1}$$

To measure the sample variance.

Compare your measured mean and variance to what you calcuated. How do they compare?

Quantization Noise in FIR Systems

In this we will examine the effect of noise on an FIR structure. There are three ways that quantization effects the output of a filter:

- Quantization error on the input signal
- Quantization at the internal arithmetic
- Coefficent roundoff effects

The first one is an effect before filtering takes place. The input itself is altered in quantization and this in turn effects the output of the filter.

Quantization error at the internal arithmetic can be be explained as follows. When only B bits of arithmetic are allowed, this means that there are only B lines to carry the information between operations (like adds or multiplies). However after any multiplication and some adds the output may require more than B bits. Thus a quantizer is placed after each multiplier and adder.

Coefficient round off effects come about when the coefficents we use are not those generated by the filter design. Instead we used quantized versions of these coefficents. We will not study these effects here. Coefficent round off has the effect of slightly moving poles and zeros in unpredictable ways. This may result in filters that were stable becoming unstable.

Rather than use the MATLAB routine filter first write a piece of code that implements an FIR filter. Your code should be able to handle fitler of length 6 and 12.

Next design a lowpass FIR filter of length 6 and 12 using any design routine you wish (use the same routine for both designs). Note which routine you used. Your cutoff frequency should be $\pi/2$.

Using the $100,000$ gaussian random variable realizations you used above input these into your length 6 filter. Obtain the output signal.

Now go over the code you wrote to implement the filter. After each multiplication and each add quantize result using the quantizer you designed in the prelab. Insert the *quantized* guassian variables (simulating quantization on the input signal) into this filter.

Now comparing the output from the filter that had no quantization error and the one with quantization error obtain the variance of the error of the filter output.

Repeat this for the length 12 filter. What do you notice about the error variance between these examples? Comment. What are the implications for implementing very long filters?