End Semester Project Report on

**Study of finite word length effect on DSP**

Submitted for the partial fulfillment of the subject

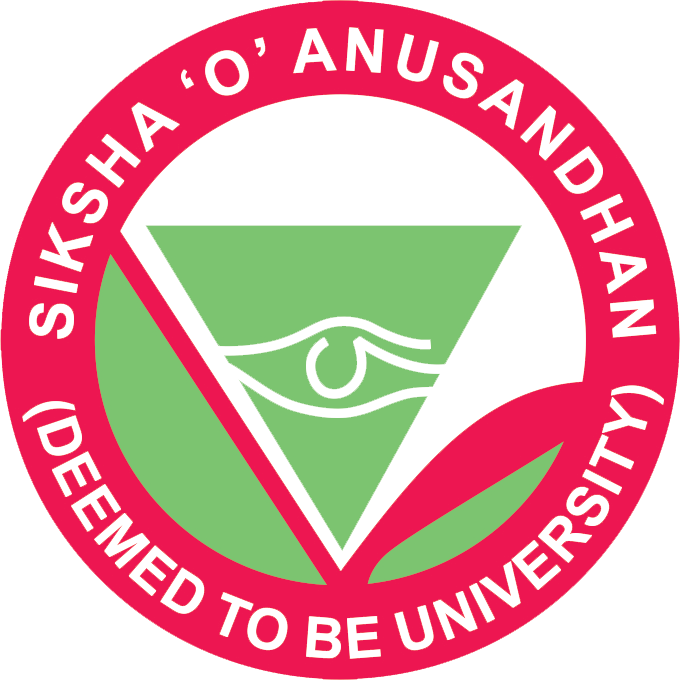
**Digital Signal Processing (EET3051)**

Submitted by

**Taniya Baral (2141002154)**

**Shradha Suman Mohapatra (2141013241)**

**B. Tech. (ECE) 6th Semester (Section–A)**



**DEPT. OF ELECTRONICS & COMMUNICATION ENGINEERING**

Institute of Technical Education and Research

**SIKSHA ‘O’ ANUSANDHAN**

**DEEMED TO BE UNIVERSITY**

Bhubaneswar, Odisha, India.

(June, 2024)

# Abstract

The study of finite word length effects on Digital Signal Processing (DSP) systems is crucial for understanding and mitigating the practical limitations imposed by digital hardware. In DSP applications, signals and filter coefficients are typically represented with a finite number of bits, leading to quantization errors and arithmetic inaccuracies that can significantly impact system performance. This research delves into the various facets of finite word length effects, including quantization noise, round-off errors, and the implications for both fixed-point and floating-point arithmetic.

The investigation begins with an overview of quantization processes and the resulting errors in digital representations of signals. We analyze the influence of truncation and rounding techniques on signal integrity, examining how these errors propagate through different stages of DSP operations. Special attention is given to the performance of digital filters, where coefficient quantization can alter the filter's frequency response, potentially degrading its effectiveness.

This study focuses on the performance of Finite Impulse Response (FIR) filters in practical DSP implementations, where signals and coefficients are often represented with a limited number of bits, leading to quantization errors that can degrade system performance. MATLAB simulations are utilized to analyze the impact of coefficient quantization on FIR filter characteristics and signal processing outcomes. The research begins by designing an FIR filter with specified order and cutoff frequencies using the window method. The original filter coefficients are then quantized to a fixed-point representation with a finite number of bits.

The frequency responses of both the original and quantized filters are computed and compared to illustrate the deviations introduced by quantization. Additionally, the quantization error in the filter coefficients is calculated and visualized to understand its distribution and magnitude. To evaluate the practical implications, a composite input signal consisting of multiple sine waves is processed through both the original and quantized filters. The resulting output signals are analyzed to assess the impact of quantization on signal fidelity. Detailed plots of the input signal, the outputs of both filters, and the difference between these outputs highlight the errors introduced by finite word length effects.

This research provides valuable insights into the challenges of finite precision in DSP systems and underscores the importance of careful bit width selection and quantization error mitigation. The findings are crucial for engineers and researchers aiming to design robust and efficient digital filters for various applications, including telecommunications, audio processing, and control systems.

**Contents**

[Abstract i](#_Toc168909794)

[Chapter 01: Introduction 1](#_Toc168909795)

[1.1. Introduction 1](#_Toc168909796)

[1.2. Background 2](#_Toc168909797)

[1.3. Project Objectives 3](#_Toc168909798)

[1.4. Scope 4](#_Toc168909799)

[1.5. Project Management 5](#_Toc168909800)

[1.6. Overview and Benefits 6](#_Toc168909801)

[1.7. Organization of the Report 6](#_Toc168909802)

[Chapter 02: Theoretical Aspects 7](#_Toc168909803)

[2.1. Background Theory and Modeling 7](#_Toc168909806)

[Modelling 8](#_Toc168909807)

[2.2. Project Layout 9](#_Toc168909808)

[2.2.1. Brief Description 9](#_Toc168909809)

[ Methodology 9](#_Toc168909810)

[Benefits 10](#_Toc168909811)

[2.3. Block diagram of the Proposed System 10](#_Toc168909812)

[2.3.1. Working of the system 10](#_Toc168909813)

[2.3.2. Flow Chart / Pseudocode 11](#_Toc168909814)

[Chapter 03: Hardware and Software Requirements 14](#_Toc168909815)

[3.1. MATLAB 14](#_Toc168909818)

[3.1.1. MATLAB's Power of Computational Mathematics 14](#_Toc168909819)

[3.1.2. Features of MATLAB 14](#_Toc168909820)

[3.1.3. Uses of MATLAB 15](#_Toc168909821)

[3.2. Advantages Of MATLAB 15](#_Toc168909822)

[3.3. Disadvantages of MATLAB 16](#_Toc168909823)

[3.4. MATLAB Requirements 16](#_Toc168909824)

[3.5. MATLAB R2023a System Requirements for Windows 16](#_Toc168909825)

[Operating System 16](#_Toc168909826)

[Processor 16](#_Toc168909827)

[RAM 16](#_Toc168909828)

[Storage 16](#_Toc168909829)

[Graphics 16](#_Toc168909830)

[3.6. The MATLAB System 16](#_Toc168909831)

[3.7. MATLAB Environment (Introduction to the Workspace) 17](#_Toc168909832)

[3.7.1. Command Window 17](#_Toc168909833)

[3.7.2. Editor Window 19](#_Toc168909834)

[3.7.3. Figure Window 19](#_Toc168909835)

[Chapter 04: Project Development & Testing Aspects 21](#_Toc168909838)

[4.1. Test Results 21](#_Toc168909840)

[4.2. Interpretation of Results 34](#_Toc168909841)

[Chapter 05: Conclusion & Future Scope 35](#_Toc168909842)

[5.1. Conclusion 35](#_Toc168909845)

[5.2. Limitations 35](#_Toc168909846)

[5.3. Further Enhancement and Future Scope 36](#_Toc168909847)

[References 38](#_Toc168909848)

[Appendix 01 39](#_Toc168909849)

[A01.1. Code Listing 39](#_Toc168909850)

[A01.2. Main Code 39](#_Toc168909851)

[A01.3. Descriptions of Libraries / Inbuilt function Used 39](#_Toc168909852)

# Chapter 01: Introduction

## Introduction

Digital Signal Processing (DSP) is an essential field that has transformed the way we handle signals in a myriad of applications, ranging from telecommunications to audio and video processing, medical imaging, and control systems. At the core of DSP lies the representation and manipulation of signals in digital form, which inherently involves the use of finite word lengths to store and process data. While digital systems offer unparalleled precision and flexibility compared to their analog counterparts, the finite word length of digital representations introduces quantization errors and arithmetic inaccuracies that can significantly impact system performance.

Finite word length effects are the result of representing signals and filter coefficients with a limited number of bits. This limitation leads to several types of errors, including quantization noise, round-off errors, and truncation errors, which can propagate and accumulate through various stages of DSP operations. Understanding and mitigating these errors is crucial for designing robust and efficient DSP systems, especially in applications where high precision and reliability are paramount.

One critical area where finite word length effects are particularly evident is in the design and implementation of digital filters, especially Finite Impulse Response (FIR) filters. FIR filters are widely used in DSP due to their inherent stability and linear phase characteristics. However, when filter coefficients are quantized to a fixed-point representation, their frequency response can be altered, leading to deviations from the desired filter performance. These deviations can degrade the effectiveness of the filter and, consequently, the quality of the processed signal.

This study focuses on the impact of finite word length effects on the performance of FIR filters. Through MATLAB simulations, we design an FIR filter with specified order and cutoff frequencies using the window method. We then quantize the filter coefficients to a fixed-point representation with a finite number of bits and analyze the resulting frequency responses. By comparing the original and quantized filter responses, we can visualize the deviations introduced by quantization. Furthermore, we assess the practical implications by processing a composite input signal through both the original and quantized filters and analyzing the output signals.

The primary objectives of this research are to quantify the errors introduced by finite word length effects, understand their impact on filter performance, and explore strategies for mitigating these effects. This work provides valuable insights for engineers and researchers in the field of DSP, guiding the design of more robust and efficient digital filters that maintain high performance despite the limitations of finite precision arithmetic.

## Background

The performance and reliability of Digital Signal Processing (DSP) systems are inherently influenced by the finite precision with which signals and computations are represented. In digital systems, signals are represented as sequences of numbers, and each number is stored using a finite number of bits. This finite word length representation introduces quantization errors, which manifest in various forms and impact the overall system performance.

#### Quantization

Quantization is the process of mapping a continuous range of values to a finite range of discrete values. In the context of DSP, this typically involves rounding or truncating the exact numerical values to fit within the available bit width. Quantization can be broadly classified into two types:

1. **Uniform Quantization**: This is the most common form of quantization where the range of values is divided into equally spaced levels. Each level represents a fixed interval of the continuous range.
2. **Non-uniform Quantization**: This type uses varying spacing between levels, which can be beneficial in applications where certain ranges of values occur more frequently and require higher precision.

Quantization introduces an error known as quantization noise, which is the difference between the actual value and the quantized value. This noise can be modeled as an additive random noise with a uniform distribution, especially when dealing with high-resolution quantization and wideband signals.

#### Fixed-Point vs. Floating-Point Representation

In digital systems, numbers can be represented using fixed-point or floating-point formats:

1. **Fixed-Point Representation**: In this format, the position of the decimal (or binary) point is fixed. Fixed-point representation is often used in embedded systems due to its simplicity and efficiency in hardware implementation. However, it offers limited dynamic range and precision.
2. **Floating-Point Representation**: Floating-point format allows the decimal point to "float," providing a much larger dynamic range and precision. This is especially useful in applications requiring high precision, such as scientific computations. However, floating-point arithmetic is more complex and computationally expensive compared to fixed-point arithmetic.

#### Finite Impulse Response (FIR) Filters

FIR filters are a class of digital filters characterized by a finite duration of their impulse response. They are preferred in many applications due to their inherent stability and linear phase properties. The output of an FIR filter is a weighted sum of the current and a finite number of previous input samples, governed by the filter coefficients.

The design of FIR filters involves determining the coefficients that meet specified frequency response characteristics. These coefficients are typically computed using methods such as the window method, frequency sampling method, or optimization techniques like the Parks-McClellan algorithm.

#### Effects of Quantization on FIR Filters

When implementing FIR filters, the filter coefficients are quantized to fit within the finite word length of the digital system. This quantization alters the exact values of the coefficients, leading to deviations in the filter's frequency response. The main effects of coefficient quantization include:

1. **Frequency Response Deviation**: The quantized filter may not perfectly match the desired frequency response, leading to attenuation or amplification of certain frequency components.
2. **Noise Introduction**: Quantization introduces noise into the system, which can degrade the signal-to-noise ratio (SNR) and overall filter performance.
3. **Stability and Performance**: While FIR filters are inherently stable, the performance in terms of passband ripple and stopband attenuation can be affected by quantization.

#### Quantization Error Analysis

The quantization error in DSP systems can be analyzed using various methods. In FIR filters, the error is typically measured as the difference between the original and quantized coefficients. The propagation of quantization error through the filter can be analyzed using techniques such as error feedback and noise shaping, which aim to distribute the quantization noise in a way that minimizes its impact on the signal quality.

Understanding and mitigating finite word length effects is essential for designing robust and efficient DSP systems. By carefully selecting the bit width and employing error mitigation techniques, it is possible to optimize the trade-off between hardware complexity and computational accuracy, ensuring high performance in practical applications. This research contributes to the body of knowledge by providing a detailed analysis of these effects and offering practical insights for the design of digital filters in finite precision environments.

## Project Objectives

The primary objective of this project is to analyze and quantify the effects of finite word length on the performance of Digital Signal Processing (DSP) systems, with a specific focus on Finite Impulse Response (FIR) filters. The key goals of this research include:

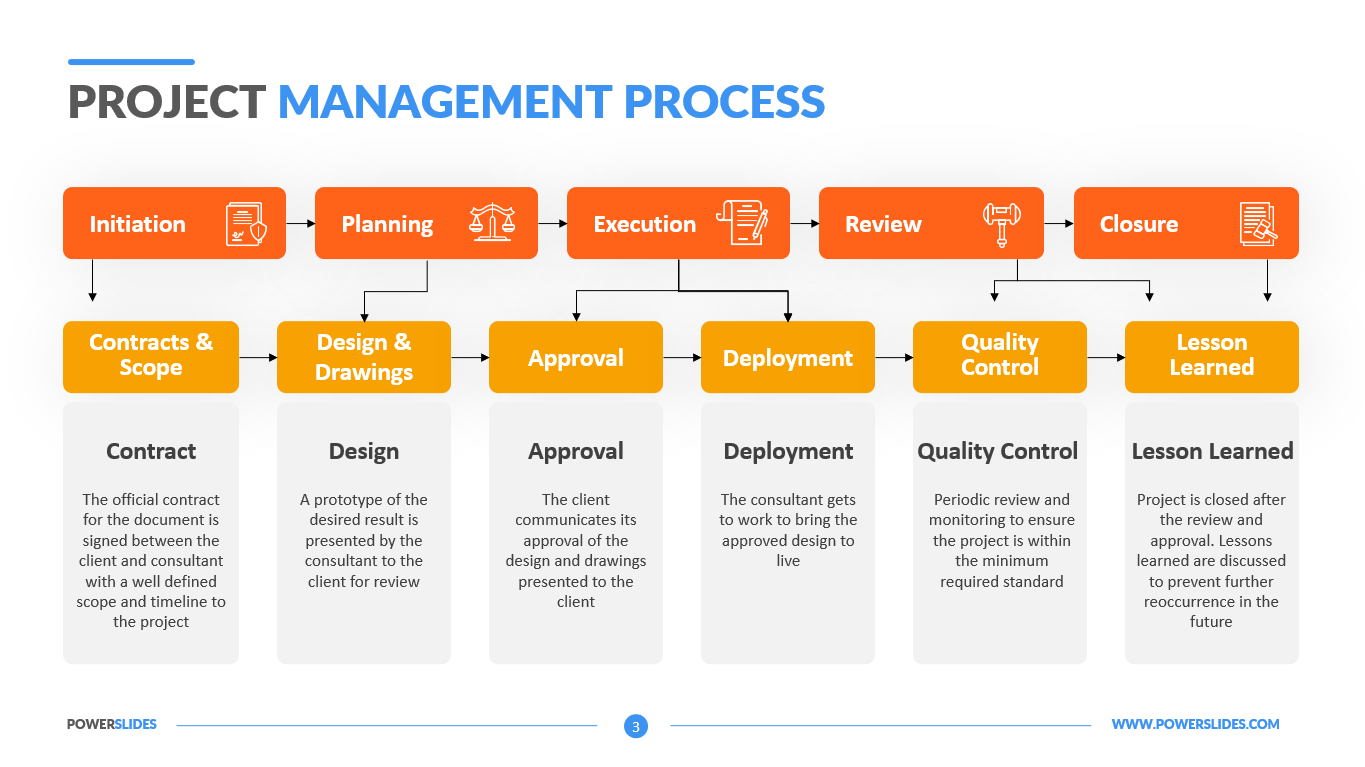
1. **Understanding Quantization Effects**: To provide a comprehensive understanding of how finite word length and quantization processes introduce errors in digital signal representations and computations.
2. **Design and Implementation of FIR Filters**: To design FIR filters using the window method and implement these filters in a MATLAB environment, both in their original and quantized forms.
3. **Comparison of Frequency Responses**: To compute and compare the frequency responses of the original and quantized FIR filters, highlighting the deviations caused by coefficient quantization.
4. **Quantization Error Analysis**: To analyze and visualize the quantization error in the filter coefficients, understanding its distribution and magnitude.
5. **Impact on Signal Processing**: To evaluate the practical implications of quantization by processing a composite input signal through both the original and quantized filters, and analyzing the resulting output signals to assess the impact on signal fidelity.
6. **Mitigation Strategies**: To explore strategies for mitigating the effects of finite word length on FIR filter performance, providing practical recommendations for the design of robust and efficient digital filters.
7. **Insightful Visualizations**: To create detailed plots and visualizations that illustrate the impact of finite word length effects on filter performance and signal processing outcomes, aiding in the understanding of these effects.
8. **Guidance for DSP Design**: To offer valuable insights and guidance for engineers and researchers in the field of DSP, helping them to optimize digital filter designs within the constraints of finite precision arithmetic.

## Scope

* **Theoretical Foundation:**
  + Review and explanation of fundamental concepts related to finite word length, quantization, and their impacts on digital systems.
  + Overview of fixed-point and floating-point arithmetic, and their relevance to DSP.
* **Filter Design:**
  + Design of FIR filters using the window method with specified order and cutoff frequencies.
  + Implementation of these filters in MATLAB to serve as a basis for comparison between original and quantized versions.
* **Quantization Process:**
  + Detailed process of quantizing FIR filter coefficients to a fixed-point representation with a finite number of bits.
  + Exploration of different quantization methods (e.g., rounding, truncation) and their impacts on coefficient accuracy.
* **Frequency Response Analysis**:
  + Computation and comparison of the frequency responses of the original and quantized FIR filters using MATLAB simulations.
  + Analysis of deviations in the frequency response due to quantization.
* **Quantization Error Evaluation**:
  + Calculation and visualization of the quantization errors in filter coefficients.
  + Analysis of error distribution and its impact on filter performance.
* **Signal Processing and Performance Assessment**:
  + Generation of composite input signals (e.g., multiple sine waves) to test the filters.
  + Filtering the input signal using both original and quantized filters and analyzing the output.
  + Comparison of the output signals to assess the practical impact of quantization on signal fidelity.
* **Visualization and Interpretation**:
  + Creation of detailed plots and visualizations to illustrate the differences between original and quantized filters, and the impact on processed signals.
  + Visualization of the quantization error and its effects.

## Project Management

According to the PMBOK Guide (Project Management Body of Knowledge), a project management life cycle consists of 5 distinct phases including initiation, planning, execution, review, and closure that combine to turn a project idea into a working product.



**Figure 1. Model of phases in project management.**

The project initiation phase is the first stage of turning an abstract idea into a meaningful goal. In this stage, we need to develop a business case and define the project on a broad level.

The project planning stage requires complete diligence as it lays out the project’s roadmap.

The project execution stage is where the project team does the actual work. The job of a project manager is to establish efficient workflows and carefully monitor the progress of the team.

In the project management process, the third and fourth phases are not sequential in nature. The project monitoring and controlling phase run simultaneously with project execution.

The project closure stage indicates the end of the project after the final delivery.

## Overview and Benefits

This project aims to investigate the effects of finite word length on the performance of Digital Signal Processing (DSP) systems, specifically focusing on Finite Impulse Response (FIR) filters. In digital systems, signals and filter coefficients are represented using a finite number of bits, which introduces quantization errors and can significantly impact the system's performance. This project uses MATLAB simulations to analyze how quantization affects the frequency response and signal processing capabilities of FIR filters.

## Organization of the Report

The report is organised into the following chapters. Each chapter is unique on its own and is described with necessary theory to comprehend it.

Chapter 2 deals with background survey and review, Chapter 3 has the description of the theoretical aspects that has been acquired to commence the project work.

# Chapter 02: Theoretical Aspects



## Background Theory and Modeling

Finite word length effects are an essential consideration in Digital Signal Processing (DSP) systems due to the inherent limitations of digital hardware in representing and processing numerical values. These effects primarily arise from the quantization of signals and filter coefficients, as well as the arithmetic operations performed on them. Understanding these effects is crucial for designing robust and efficient DSP systems.

#### Quantization

Quantization is the process of mapping a continuous range of values to a finite set of discrete levels. In DSP, quantization occurs in two main areas:

1. **Signal Quantization**: Converting continuous or high-resolution digital signals into a format with fewer bits.
2. **Coefficient Quantization**: Representing filter coefficients using a limited number of bits.

Quantization can be classified into two types:

* **Uniform Quantization**: The range of values is divided into equally spaced levels.
* **Non-uniform Quantization**: The spacing between quantization levels varies, which can be more efficient for signals with non-uniform distributions.

The quantization process introduces an error known as quantization noise, which is the difference between the original and quantized values. This noise can be modeled as an additive white noise under certain conditions, impacting the overall system performance.

#### Fixed-Point and Floating-Point Arithmetic

Digital systems typically use either fixed-point or floating-point arithmetic to represent numerical values:

* **Fixed-Point Representation**: Numbers are represented with a fixed number of digits before and after the decimal point. Fixed-point arithmetic is simpler and faster but offers limited dynamic range and precision.
* **Floating-Point Representation**: Numbers are represented with a mantissa and an exponent, allowing for a much larger dynamic range and precision. However, floating-point arithmetic is more complex and computationally intensive.

#### Finite Impulse Response (FIR) Filters

FIR filters are a class of digital filters characterized by a finite-duration impulse response. They are widely used due to their inherent stability and linear phase properties. The output of an FIR filter is a weighted sum of current and past input samples, governed by the filter coefficients.

The design of FIR filters involves determining the coefficients that achieve the desired frequency response. This can be done using methods such as the window method, frequency sampling, or optimization techniques like the Parks-McClellan algorithm.

#### Effects of Quantization on FIR Filters

Quantization of FIR filter coefficients can lead to:

* **Frequency Response Deviation**: Alterations in the filter's frequency response, potentially degrading its performance.
* **Introduction of Quantization Noise**: Noise that degrades the signal-to-noise ratio (SNR) and overall filter performance.
* **Reduced Precision**: Limiting the filter's ability to accurately process signals, particularly in applications requiring high precision.

### Modelling

To study the finite word length effects on DSP systems, specifically FIR filters, we employ a simulation-based approach using MATLAB.

#### Filter Design and Quantization

1. **Design the FIR Filter**:
   * Define the filter order and cutoff frequencies.
   * Use the fir1 function to design the filter with the Hamming window by default.
2. **Quantize the Filter Coefficients**:
   * Convert the floating-point coefficients to fixed-point representation using a specified number of bits.
   * Quantize the coefficients by rounding them to the nearest representable value.

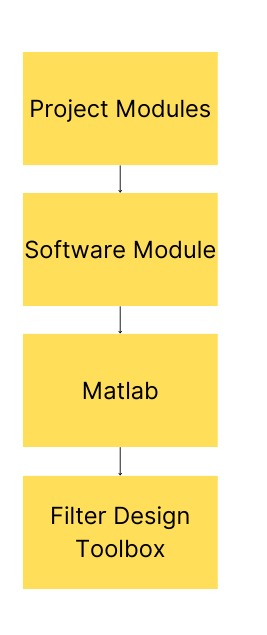
#### Frequency Response Analysis

1. **Compute Frequency Response**:
   * Use the freqz function to compute the frequency response of both the original and quantized filters.
   * Compare the frequency responses to identify deviations caused by quantization.
2. **Quantization Error Analysis**:
   * Calculate the error between the original and quantized coefficients.
   * Visualize the quantization error to understand its distribution and impact.

#### Signal Processing and Performance Evaluation

1. **Generate Input Signal**:
   * Create a composite signal consisting of multiple sine waves to simulate a realistic input scenario.
2. **Filter the Input Signal**:
   * Process the input signal through both the original and quantized filters using the filtfilt function to avoid phase distortions.
3. **Analyze Output Signals**:
   * Compare the output signals from the original and quantized filters.
   * Assess the impact of quantization on signal fidelity by analyzing the difference between the outputs.

## Project Layout



**Figure 2. Layout of project module**

### Brief Description

### Methodology

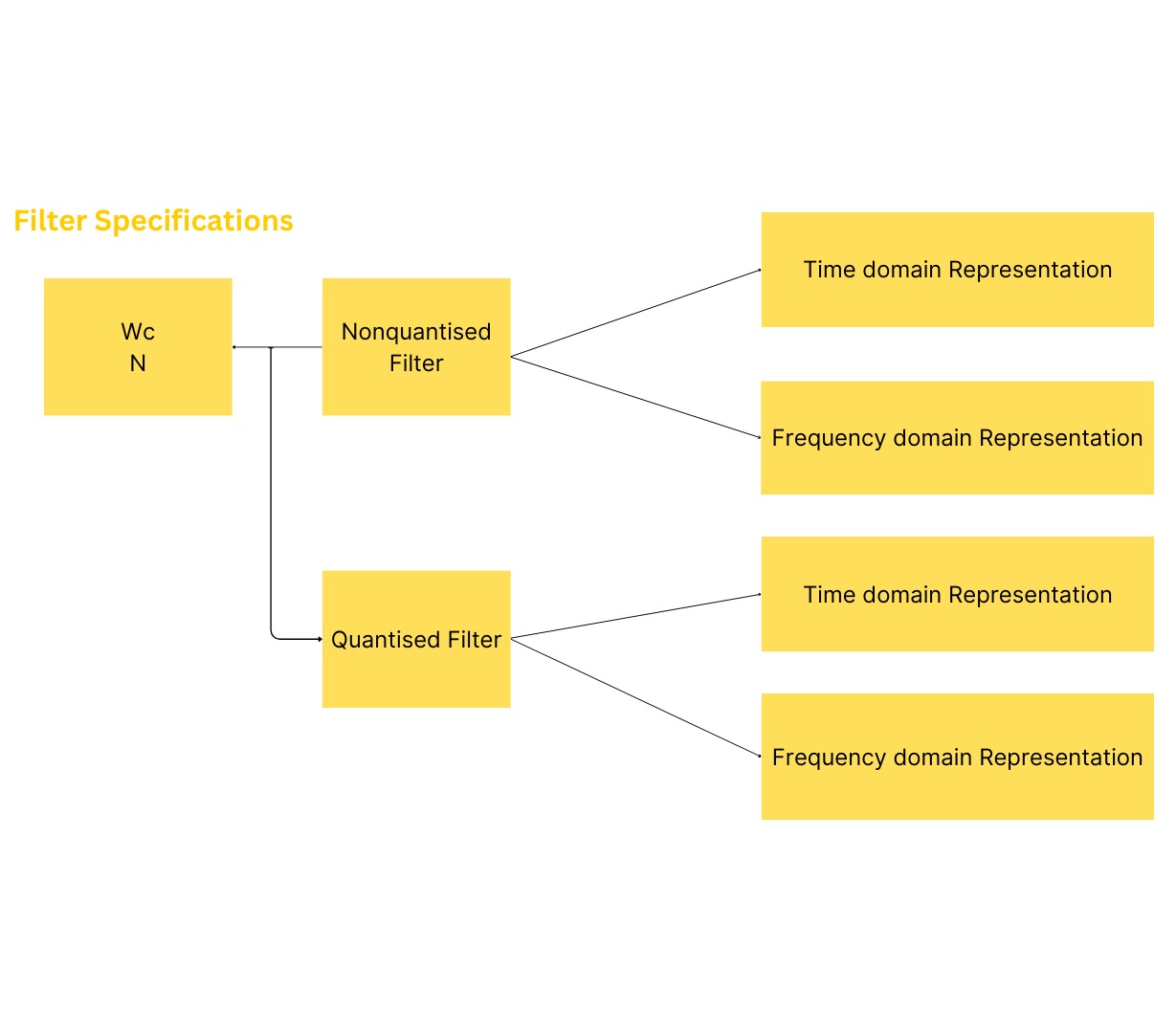
The project uses MATLAB to simulate the following:

1. **Filter Design**: An FIR filter is designed using the window method with specified order and cutoff frequencies.
2. **Quantization**: The filter coefficients are quantized to a fixed-point representation with a finite number of bits.
3. **Frequency Response Analysis**: The frequency responses of the original and quantized filters are computed and compared.
4. **Quantization Error Analysis**: The quantization error in the filter coefficients is calculated and visualized.
5. **Signal Processing Evaluation**: A composite input signal is filtered using both the original and quantized filters to assess the impact on signal fidelity.

### Benefits

1. **Enhanced Understanding**: Provides a detailed analysis of how quantization affects digital filters and overall DSP system performance.
2. **Improved Filter Design**: Highlights the importance of selecting appropriate bit widths and quantization methods.
3. **Practical Insights**: Demonstrates real-world impacts of finite word length through MATLAB simulations.
4. **Error Mitigation**: Explores strategies to mitigate the effects of quantization errors.
5. **Visualization and Interpretation**: Uses detailed plots to illustrate the differences caused by quantization.
6. **Guidance for DSP Practitioners**: Offers insights and recommendations for designing robust and efficient DSP systems

## Block diagram of the Proposed System



### Working of the system

The system evaluates the impact of finite word length effects on the performance of FIR filters in digital signal processing (DSP). Here's a step-by-step outline of how the system operates:

#### 1. FIR Filter Design

The system designs an FIR filter using the window method, specifying the filter's order and cutoff frequencies. This initial filter is created with floating-point precision, ensuring an accurate representation of the desired frequency response.

#### 2. Quantization of Filter Coefficients

To simulate the finite word length effect, the system quantizes the filter coefficients to a fixed-point representation with a defined number of bits. This step introduces quantization noise and potential deviations in filter performance due to the limited precision of the coefficients.

#### 3. Frequency Response Analysis

The system calculates the frequency responses of both the original (floating-point) and the quantized (fixed-point) filters. By comparing these responses, the system identifies any deviations caused by quantization, which can affect the filter's effectiveness in attenuating or passing specific frequency components.

#### 4. Quantization Error Analysis

The system computes the quantization error, which is the difference between the original and quantized coefficients. This error is visualized to understand its magnitude and distribution across the filter coefficients, providing insight into how quantization affects the filter's structure. The average error is found out by

#### 5. Signal Processing Evaluation

A composite input signal, typically a combination of sine waves, is processed through both the original and quantized filters. The system then compares the output signals to assess the impact of coefficient quantization on signal fidelity. This step evaluates how well the quantized filter performs in practice compared to the original design.

#### 6. Visualization

Throughout the process, the system generates various plots to visualize the frequency responses, quantization errors, and the differences in output signals. These visual aids help in understanding the effects of finite word length on the filter's performance and the overall DSP system.

### Flow Chart / Pseudocode

// Clear previous variables, close figures, and clear command window

clear all variables

close all figures

clear command window

// Define FIR filter parameters

filter\_order = 31 // Order of the FIR filter

cutoff\_freq = [0.1, 0.18] // Normalized cutoff frequencies for bandpass filter

num\_bits = 8 // Number of bits for fixed-point representation

// Design FIR filter using window method

b = fir1(filter\_order, cutoff\_freq, 'bandpass') // Create FIR filter coefficients

// Compute original filter frequency response

H\_orig, W\_orig = freqz(b, 1, 1024) // Compute frequency response of original filter

// Display original filter coefficients

print "Original filter coefficients:"

print b

// Quantize filter coefficients

quantized\_b = round(b \* 2^(num\_bits-1)) / 2^(num\_bits-1) // Quantize coefficients to fixed-point representation

// Compute quantized filter frequency response

H\_quant, W\_quant = freqz(quantized\_b, 1, 1024) // Compute frequency response of quantized filter

// Display quantized filter coefficients

print "Quantized filter coefficients:"

print quantized\_b

// Plot frequency responses of original and quantized filters

create figure

plot W\_orig/pi, 20\*log10(abs(H\_orig)), 'b', 'LineWidth', 1.5 // Plot original filter response

hold on

plot W\_quant/pi, 20\*log10(abs(H\_quant)), 'r--', 'LineWidth', 1.5 // Plot quantized filter response

title 'Frequency Response of FIR Filter'

xlabel 'Normalized Frequency (\times\pi rad/sample)'

ylabel 'Magnitude (dB)'

legend 'Original', 'Quantized'

grid on

// Compute and display quantization error in coefficients

quant\_error = b - quantized\_b // Calculate quantization error

avgE = sum((quant\_error).^2) / filter\_order // Compute average quantization error

print "Quantization error in coefficients:"

print quant\_error

// Plot quantization error

create figure

subplot 311

stem b, 'LineWidth', 2 // Plot original filter coefficients

xlabel 'time'

ylabel 'Amplitude'

title 'h(n)'

grid on

subplot 312

stem quantized\_b, 'LineWidth', 2 // Plot quantized filter coefficients

xlabel 'time'

ylabel 'Amplitude'

title 'h\_q(n)'

grid on

subplot 313

stem quant\_error, 'LineWidth', 2 // Plot quantization error

title 'Quantization Error in FIR Filter Coefficients'

xlabel 'Coefficient Index'

ylabel 'Error'

grid on

// Generate sample input signal (sine wave)

fs = 1000 // Sampling frequency

t = 0:1/fs:0.25-1/fs // Time vector

input\_signal = sin(2\*pi\*50\*t) + 0.5\*sin(2\*pi\*120\*t) // Generate input signal

// Filter input signal using original filter

output\_orig = filtfilt(b, 1, input\_signal) // Filter input signal with original filter

// Filter input signal using quantized filter

output\_quant = filtfilt(quantized\_b, 1, input\_signal) // Filter input signal with quantized filter

// Plot input and output signals

create figure

subplot 4,1,1

plot t, input\_signal // Plot input signal

title 'Input Signal'

xlabel 'Time (s)'

ylabel 'Amplitude'

grid on

subplot 4,1,2

plot t, output\_orig // Plot output signal (original filter)

title 'Output Signal (Original Filter)'

xlabel 'Time (s)'

ylabel 'Amplitude'

grid on

subplot 4,1,3

plot t, output\_quant // Plot output signal (quantized filter)

title 'Output Signal (Quantized Filter)'

xlabel 'Time (s)'

ylabel 'Amplitude'

grid on

subplot 4,1,4

plot t, abs(output\_orig - output\_quant) // Plot difference between original and quantized filter outputs

title 'Difference'

xlabel 'Time (s)'

ylabel 'Amplitude'

grid on

# Chapter 03: Hardware and Software Requirements



## MATLAB

MATLAB (matrix laboratory) is a fourth-generation high-level programming language and interactive environment for numerical computation, visualization and programming. MATLAB is developed by MathWorks. It allows matrix manipulations; plotting of functions and data; implementation of algorithms; creation of user interfaces; interfacing with programs written in other languages, including C, C++, Java, and FORTRAN; analyze data; develop algorithms; and create models and applications. It has numerous built-in commands and math functions that help in mathematical calculations, generating plots, and performing numerical methods.

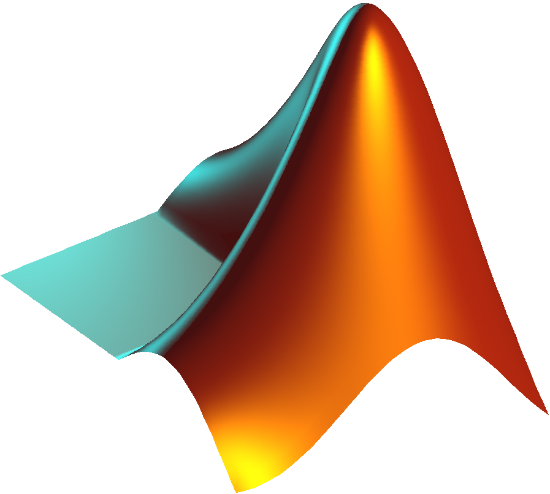


Fig.5.1. The MATLAB Logo

### MATLAB's Power of Computational Mathematics

MATLAB is used in every facet of computational mathematics. Following are some commonly used mathematical calculations where it is used most commonly −

* Dealing with Matrices and Arrays
* 2-D and 3-D Plotting and graphics
* Linear Algebra
* Algebraic Equations
* Non-linear Functions
* Statistics
* Data Analysis
* Calculus and Differential Equations
* Numerical Calculations
* Integration
* Transforms
* Curve Fitting
* Various other special functions

### Features of MATLAB

Following are the basic features of MATLAB.

* It is a high-level language for numerical computation, visualization and application development.
* It also provides an interactive environment for iterative exploration, design and problem solving.
* It provides vast library of mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations.
* It provides built-in graphics for visualizing data and tools for creating custom plots.
* MATLAB's programming interface gives development tools for improving code quality maintainability and maximizing performance.
* It provides tools for building applications with custom graphical interfaces.
* It provides functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET and Microsoft Excel.

### Uses of MATLAB

MATLAB is widely used as a computational tool in science and engineering encompassing the fields of physics, chemistry, math and all engineering streams. It is used in a range of applications including

**Fig. 5.2. Application Domains of MATLAB**

## Advantages Of MATLAB

The various advantages of MATLAB are:

* Implement and test your algorithms easily.
* Develop the computational codes easily.
* Debug easily.
* Use a large database of built in algorithms.
* Process still images and create simulation videos easily.
* Symbolic computation can be easily done.
* Call external libraries.

## Disadvantages of MATLAB

The disadvantages of MATLAB are summarized below:

* **Cost**

Five to ten times as expensive as a typical FORTRAN compiler or C is a full copy of MATLAB. MATLAB is cost-effective for organizations since this relatively high cost is more than offset by the shortened time an engineer or scientist must spend developing functional software. However, the cost prevents most people from even considering buying it. Fortunately, MATLAB also offers a low-cost Student Edition, which is a fantastic resource for students who want to learn it. MATLAB's Student Edition and Full Edition are nearly identical.

* **Interpreted Language**

The fact that it is an interpreted language and hence might run more slowly than a compiled language is its first drawback. The MATLAB program can be correctly structured to check for this issue.

## MATLAB Requirements

Each piece of computer software, including MATLAB, is run on an operating system. The operating system and other parts are essential to a computer system's efficient operation.

## MATLAB R2023a System Requirements for Windows

|  |  |
| --- | --- |
| Operating System | Windows 11, Windows 10 (version 20H2 or higher) Windows Server 2019, Windows Server 2022 |
| Processor | **Minimum:** Any Intel or AMD x86–64 processor **Recommended:** Any Intel or AMD x86–64 processor with four logical cores and AVX2 instruction set support **Note:** A future release of MATLAB will require a processor with AVX2 instruction set support |
| RAM | **Minimum**: 4 GB, **Recommended**: 8 GB |
| Storage | 3.8 GB for just MATLAB, 4-6 GB for a typical installation 23 GB for an all products installation, An SSD is strongly recommended |
| Graphics | No specific graphics card is required, but a hardware accelerated graphics card supporting OpenGL 3.3 with 1GB GPU memory is recommended.  GPU acceleration using Parallel Computing Toolbox requires a GPU with a specific range of compute capability. |

## The MATLAB System

The MATLAB system consists of five main parts:

* **The MATLAB language.**

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

* **The MATLAB working environment.**

This is the set of tools and facilities that you work with as the MATLAB user or programmer. It includes facilities for managing the variables in your workspace and importing and exporting data. It also includes tools for developing, managing, debugging, and profiling M-files, MATLAB's applications.

* **Handle Graphics.**

This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete Graphical User Interfaces on your MATLAB applications.

* **The MATLAB mathematical function library.**

This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

* **The MATLAB Application Program Interface (API).**

This is a library that allows you to write C and Fortran programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

## MATLAB Environment (Introduction to the Workspace)

In accordance with our selections made throughout the installation procedure, a MATLAB shortcut will be generated on the desktop. By selecting the MATLAB Downloading icon that has been placed on the desktop, we can now work with MATLAB.

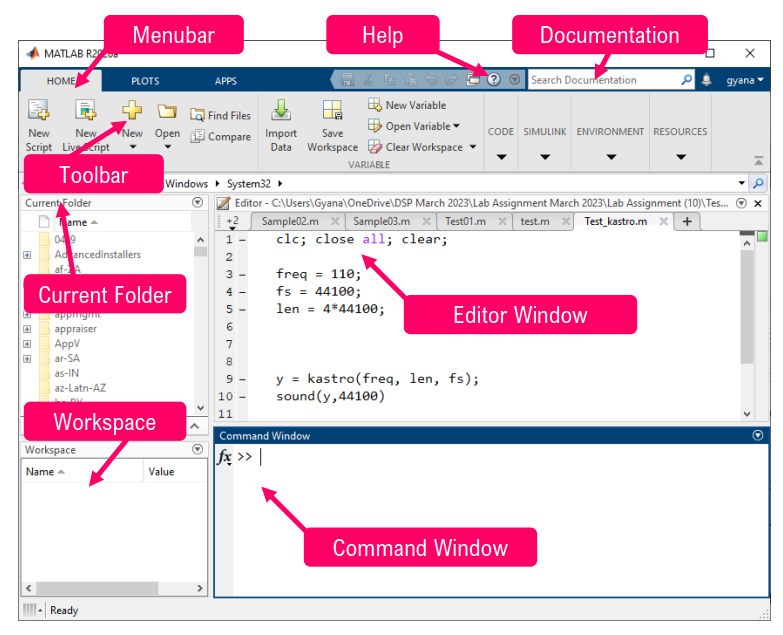
The top three window kinds are as follows:

* Command Windows: a platform for entering commands.
* Edit Windows: With this, MATLAB programs can be written and modified by the user.
* Figure Windows: Plot and graph displays can be found.

### Command Window

The main window is this one. The MATLAB command prompt (>>) identifies it. You are presented with this window by MATLAB when you start the function program. You enter all MATLAB instructions, including those for running user-written programs, in this window. This window in MATLAB is a section of the program that also includes other, smaller windows or panes.

* The Command Window is where you type in MATLAB commands. These can be simple equations you want evaluated, or more complex expressions involving MATLAB scripts or functions.
* The Command History Window shows the commands you have entered in the past. You can repeat any of these commands by double-clicking on them, or by dragging them from the Command History Window into the Command Window. You can also scroll back to previous commands by using the up arrow in the Command Window. When you learn how to edit MATLAB script files or functions, you can also drag commands from the Command History Window into a file.
* The Workspace shows the list of variables that are currently defined, and what type of variable each is. (i.e., a simple scalar, a vector, or a matrix, and the size of all arrays. ) Depending on the size (i.e., type) of the variable, its value may also be shown.
* If any of the variables in the Workspace are plottable, they may be plotted quickly and easily by right-clicking on the variable name and selecting a plot type.
* Use the control key to select multiple variables for plotting.
* The plot types shown on the resulting menu will depend on the ( dimensionality of ) the variable(s) that are selected.
* The Current Directory Window shows the contents of the current working directory.
* Double click on any file to open it in a (text) editor
* Right-click on MATLAB scripts and function files to execute the commands contained therein.
* Right-click on data file to import the data as MATLAB variables.
* Change directories by clicking on folders or use the Current Working Directory text box at the top of the MATLAB working environment.
* The File Details Window shows full details of the files in the current working directory.
* Windows may be re-arranged according to your personal preferences, including dragging windows away from the MATLAB work environment.



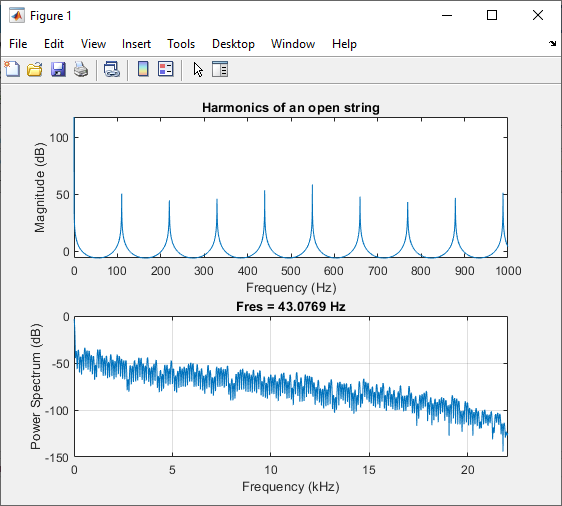
**Fig.5.3. The MATLAB Environment**

### Editor Window

Our programs are written, edited, created, and saved in these M-file files. Any text editor will work to complete these tasks. The built-in editor of MATLAB is supported. But, we can use our editor by using the typical system commands for editing files. The command is entered at the prompt for MATLAB from within MATLAB, making sure to include the exclamation point (!). Exclamation characters cause MATLAB to temporarily transfer control back to the operating system that is local, which then executes the commands necessary to generate the character. After editing is finished, the control is transferred back to MATLAB.

### Figure Window

The figure window or graphics, a distinct gray window with a (by default) white backdrop color, receives the output of every graphics command we copied in the command window. If there is enough system memory, the client can construct an unlimited number of figure windows.



**Fig.5.4. The MATLAB Figure Window**



# Chapter 04: Project Development & Testing Aspects



## Test Results

Lowpass Filter()

N=11



Fig1.1



Fig1.2



Fig1.3

N=21



Fig2.1



Fig2.2



Fig2.3

N=31



Fig3.1



Fig3.2



Fig 3.3

Highpass Filter()

N=11



Fig4.1



Fig4.2



Fig4.3

N=21



Fig5.1



Fig5.2



Fig5.3

N=31



Fig6.1



Fig6.2



Fig6.3

Bandpass Filter()

N=11



Fig7.1



Fig7.2



Fig7.3

N=21



Fig8.1



Fig8.2



Fig 8.3

N=31



Fig9.1



Fig9.2



Fig9.3

Bandreject Filter()

N=11



Fig10.1



Fig 10.2



Fig10.3

N=21



Fig11.1



Fig11.2



Fig11.3

N=31



Fig12.1



Fig12.2



Fig12.3

## Interpretation of Results

|  |  |  |
| --- | --- | --- |
| **Filter Type** | **N** | **Average Error** |
| LOW PASS FILTER | 11 | 4.8010e-06 |
| 21 | 5.1468e-06 |
| 31 | 2.7303e-06 |
| HIGH PASS FILTER | 11 | 4.2284e-06 |
| 21 | 6.4584e-06 |
| 31 | 5.1877e-06 |
| BAND PASS FILTER | 11 | 6.8929e-06 |
| 21 | 5.3451e-06 |
| 31 | 5.7560e-06 |
| BAND REJECT FILTER | 11 | 2.3100e-06 |
| 21 | 6.4651e-06 |
| 31 | 4.8242e-06 |

Fig 13

We have used Type 1 Filter which has odd Symmetry and odd filter order because it is versatile and choose FIR filter because it has less world length effect. From Fig 1.1, 2.1, 3.1, 4.1, 5.1, 6.1, 7.1, 8.1 , 9.1, 10.1, 11.1, 12.1 and 13 we can observe with increase in filter order average error increases and then decreases.

# Chapter 05: Conclusion & Future Scope



## Conclusion

The study of finite word length effects on digital signal processing (DSP) systems reveals significant insights into the practical limitations and challenges of digital filter design. Through MATLAB simulations, we examined the impact of quantizing FIR filter coefficients on the filter's frequency response and overall performance. Quantization introduces errors that can significantly alter the filter's frequency response, degrading its ability to accurately process signals. The frequency responses of the original and quantized filters show noticeable differences, affecting the filter's effectiveness in applications requiring precise filtering. The quantization error, resulting from representing filter coefficients with a finite number of bits, highlights the trade-off between precision and computational efficiency. Visualizing this error provides valuable insights into its distribution and magnitude, aiding in the design of more robust filters. Processing a composite input signal through both the original and quantized filters demonstrates the practical implications of finite word length effects, with differences in output signals underscoring the impact on signal fidelity, crucial for maintaining the quality of processed signals in real-world applications.

## Limitations

* **Precision Constraints**: Finite word length limits the precision with which signal values and filter coefficients can be represented, leading to quantization errors and potential signal degradation.
* **Frequency Response Deviation**: The quantization of filter coefficients can cause significant deviations in the filter's frequency response, affecting its ability to accurately pass or attenuate specific frequency components.
* **Increased Noise**: Quantization introduces quantization noise, which can accumulate and propagate through the DSP system, potentially degrading the overall signal-to-noise ratio (SNR).
* **Trade-Off Between Precision and Complexity**: There is a trade-off between achieving higher precision (which requires more bits and computational resources) and maintaining a feasible level of computational efficiency and resource usage.
* **Non-Ideal Filter Performance**: Quantized filters may not perform as ideally as their theoretical counterparts, leading to suboptimal filtering results in practical applications.
* **Complexity of Error Analysis**: Accurately modeling and analyzing the impact of quantization errors can be complex, requiring detailed simulations and thorough understanding of DSP theory.
* **Limited Dynamic Range**: Finite word length restricts the dynamic range of signals that can be accurately represented, which can be problematic in applications with a wide range of signal amplitudes.
* **Hardware Constraints**: The implementation of high-precision filters may be limited by the capabilities of the available hardware, leading to compromises in filter performance.
* **Real-Time Processing Challenges**: In real-time DSP applications, the additional computational load introduced by higher precision requirements can impact the system's ability to process signals in real time.
* **Sensitivity to Bit Width Selection**: The choice of bit width for quantization significantly affects the filter's performance, making it critical to carefully balance precision and computational efficiency for each specific application.

## Further Enhancement and Future Scope

* **Adaptive Quantization Techniques**: Future research can focus on developing adaptive quantization techniques that dynamically adjust the word length based on the signal characteristics and filtering requirements, thereby optimizing precision and computational efficiency.
* **Advanced Error Correction Methods**: Implementing advanced error correction algorithms can help mitigate the impact of quantization errors, improving the overall robustness and reliability of DSP systems.
* **High-Precision Floating-Point Arithmetic**: Exploring the use of high-precision floating-point arithmetic can provide better performance for applications requiring very high accuracy, albeit at the cost of increased computational resources.
* **Machine Learning Integration**: Incorporating machine learning algorithms to predict and compensate for quantization errors can enhance the performance of DSP systems, making them more adaptive and intelligent.
* **Optimization for Specific Applications**: Tailoring quantization strategies to specific applications (e.g., audio processing, telecommunications, biomedical signal processing) can help achieve better performance by addressing the unique requirements and constraints of each domain.
* **Hardware Improvements**: Advances in digital hardware, such as more efficient digital signal processors (DSPs) and field-programmable gate arrays (FPGAs), can support higher precision and more complex DSP operations, expanding the possibilities for future enhancements.
* **Hybrid Fixed-Point and Floating-Point Systems**: Developing hybrid systems that use a combination of fixed-point and floating-point arithmetic can offer a balance between precision and resource usage, optimizing performance for various DSP applications.
* **Enhanced Visualization Tools**: Creating more advanced visualization tools for analyzing quantization effects and filter performance can aid in the design and debugging of DSP systems, making the development process more intuitive and efficient.
* **Quantization Noise Reduction Techniques**: Researching new methods to reduce quantization noise, such as dithering and noise shaping, can lead to significant improvements in the quality of processed signals.
* **Real-Time Processing Capabilities**: Enhancing the real-time processing capabilities of DSP systems, particularly for high-throughput and low-latency applications, can expand their usability in critical areas such as real-time communication and control systems.
* **Cross-Disciplinary Applications**: Applying the insights and advancements in finite word length effects to other fields, such as robotics, autonomous systems, and IoT devices, can broaden the scope and impact of this research.
* **Open-Source Frameworks**: Developing and sharing open-source frameworks and toolkits for simulating and analyzing finite word length effects can foster collaboration and innovation within the DSP community.

# References

* Vinay K. Ingle and John G. Proakis. Digital Signal Processing Using MATLAB. 3rd Edition. Global Engineering: Christopher M. Shortt.

# Appendix 01

## A01.1. Code Listing

## A01.2. Main Code

clc; close all; clear;

% Define FIR filter parameters

filter\_order = 31; % Order of the FIR filter

cutoff\_freq = [0.2,0.25]; % Normalized cutoff frequency (0 to 1, where 1 is Nyquist frequency)

num\_bits = 8; % Number of bits for fixed-point representation

% Design FIR filter using the window method

b = fir1(filter\_order, cutoff\_freq,'stop');

% Original filter frequency response

[H\_orig, W\_orig] = freqz(b, 1, 1024);

% Display original filter coefficients

disp('Original filter coefficients:');

disp(b);

% Quantize filter coefficients

quantized\_b = round(b \* 2^(num\_bits-1)) / 2^(num\_bits-1);

% Quantized filter frequency response

[H\_quant, W\_quant] = freqz(quantized\_b, 1, 1024);

% Display quantized filter coefficients

disp('Quantized filter coefficients:');

disp(quantized\_b);

% Plot frequency responses

figure;

plot(W\_orig/pi, 20\*log10(abs(H\_orig)), 'b', 'LineWidth', 1.5); hold on;

plot(W\_quant/pi, 20\*log10(abs(H\_quant)), 'r--', 'LineWidth', 1.5);

title('Frequency Response of FIR Filter');

xlabel('Normalized Frequency (\times\pi rad/sample)');

ylabel('Magnitude (dB)');

legend('Original', 'Quantized');

grid on;

% Compute quantization error

quant\_error = b - quantized\_b;

avgE = sum((quant\_error).^2) / filter\_order

% Display quantization error

disp('Quantization error in coefficients:');

disp(quant\_error);

% Plot quantization error

figure;

subplot(311)

stem(b,'LineWidth', 2);

xlabel('time');

ylabel('Amplitude')

title('h(n)'); grid on;

subplot(312)

stem(quantized\_b,'LineWidth', 2);

xlabel('time');

ylabel('Amplitude')

title('h\_q(n)'); grid on;

subplot(313)

% stem(0:filter\_order, quant\_error, 'filled');

stem(quant\_error,'LineWidth', 2);

title('Quantization Error in FIR Filter Coefficients');

xlabel('Coefficient Index');

ylabel('Error'); grid on;

grid on;

% Generate a sample input signal (e.g., a sine wave)

fs = 1000; % Sampling frequency

t = 0:1/fs:0.25-1/fs; % Time vector

input\_signal = sin(2\*pi\*50\*t) + 0.5\*sin(2\*pi\*120\*t);

% Filter the input signal using the original filter

output\_orig = filtfilt(b, 1, input\_signal);

% Filter the input signal using the quantized filter

output\_quant = filtfilt(quantized\_b, 1, input\_signal);

% Plot input and output signals

figure;

subplot(4,1,1);

plot(t, input\_signal);

title('Input Signal');

xlabel('Time (s)');

ylabel('Amplitude');

grid on;

subplot(4,1,2);

plot(t, output\_orig);

title('Output Signal (Original Filter)');

xlabel('Time (s)');

ylabel('Amplitude');

grid on;

subplot(4,1,3);

plot(t, output\_quant);

title('Output Signal (Quantized Filter)');

xlabel('Time (s)');

ylabel('Amplitude');

grid on;

subplot(4,1,4);

plot(t, abs(output\_orig-output\_quant));

title('Difference');

xlabel('Time (s)');

ylabel('Amplitude');

grid on;

## A01.3. Descriptions of Libraries / Inbuilt function Used

 **clc**:

* **Description**: Clears the command window.
* **Usage**: clc

 **close all**:

* **Description**: Closes all open figure windows.
* **Usage**: close all

 **clear**:

* **Description**: Removes all variables from the workspace.
* **Usage**: clear

 **fir1**:

* **Description**: Designs an FIR filter using the window method.
* **Parameters**:
  + filter\_order: The order of the filter.
  + cutoff\_freq: The normalized cutoff frequency/frequencies (0 to 1, where 1 corresponds to the Nyquist frequency).
  + 'stop': Specifies the type of filter (stopband in this case).
* **Usage**: b = fir1(filter\_order, cutoff\_freq, 'stop')

 **freqz**:

* **Description**: Computes the frequency response of a digital filter.
* **Parameters**:
  + b: Filter coefficients.
  + a: Denominator coefficients (for FIR filters, this is typically 1).
  + n: Number of points to use in the frequency response calculation.
* **Returns**:
  + H: Frequency response of the filter.
  + W: Corresponding frequencies.
* **Usage**: [H, W] = freqz(b, a, n)

 **disp**:

* **Description**: Displays text or variables in the command window.
* **Usage**: disp('Text or variable')

 **round**:

* **Description**: Rounds the elements of a matrix to the nearest integers.
* **Usage**: quantized\_b = round(b \* 2^(num\_bits - 1)) / 2^(num\_bits - 1)

 **figure**:

* **Description**: Creates a new figure window.
* **Usage**: figure

 **plot**:

* **Description**: Creates a 2D line plot.
* **Parameters**:
  + X: x-axis data.
  + Y: y-axis data.
  + Additional parameters for line style, color, and width.
* **Usage**: plot(X, Y, 'LineStyle', 'Color', 'LineWidth')

 **title**:

* **Description**: Sets the title of the current axes.
* **Usage**: title('Title text')

 **xlabel**:

* **Description**: Sets the x-axis label for the current axes.
* **Usage**: xlabel('Label text')

 **ylabel**:

* **Description**: Sets the y-axis label for the current axes.
* **Usage**: ylabel('Label text')

 **legend**:

* **Description**: Adds a legend to the current axes.
* **Usage**: legend('Label1', 'Label2')

 **grid on**:

* **Description**: Turns on the grid for the current axes.
* **Usage**: grid on

 **stem**:

* **Description**: Creates a stem plot (discrete plot with stems).
* **Usage**: stem(X, 'LineWidth', Value)

 **filtfilt**:

* **Description**: Performs zero-phase digital filtering by processing the input data in both the forward and reverse directions.
* **Parameters**:
  + b: Numerator coefficients of the filter.
  + a: Denominator coefficients (for FIR filters, this is typically 1).
  + input\_signal: The input signal to be filtered.
* **Usage**: output = filtfilt(b, a, input\_signal)

 **subplot**:

* **Description**: Creates a subplot in a figure.
* **Parameters**:
  + m, n, p: Specifies an m-by-n grid and creates an axes for the subplot in the p-th position.
* **Usage**: subplot(m, n, p)

 **sum**:

* **Description**: Sums the elements of an array.
* **Usage**: total = sum(array)

 **abs**:

* **Description**: Computes the absolute value of each element in an array.
* **Usage**: absolute\_values = abs(array)

 **log10**:

* **Description**: Computes the base-10 logarithm of each element in an array.
* **Usage**: log\_values = log10(array)

 **pi**:

* **Description**: The mathematical constant π.
* **Usage**: pi