**Title**

Study of Analog to Digital Conversion using MATLAB

**Abstract**

This experiment is designed to-

1.To understand the use of MATLAB for solving communication engineering problems.

2.To develop understanding of Digital to Analog conversion using MATLAB.

**Introduction**

1. ANALOG TO DIGITAL CONVERSION (ADC): Is the process of converting continuous analog signals into discrete digital data. Key factors in ADC include resolution and sampling rate (how often the signal is measured). ADCs are used in many applications like audio, sensors, and communication systems.
2. PULSE CODE MODULATION (PCM): The most common technique to change an analog signal to digital data (digitization) is called pulse code modulation (PCM). A PCM encoder has three processes, as shown in the following figure.

a. The analog signal is sampled.

b. The sampled signal is quantized.

c. The quantized values are encoded as streams of bits

|  |
| --- |
| A diagram of a computer network  AI-generated content may be incorrect. |

Fig 1: Components of PCM encoder

1. SAMPLING: The first step in PCM is sampling. The analog signal is sampled every Ts s, where Ts is the sample interval or period. The inverse of the sampling interval is called the sampling rate or sampling frequency and denoted by fs, where fs = 1/Ts. There are three sampling methods—ideal, natural, and flat-top—as shown in the following figure.

|  |
| --- |
| A diagram of a signal  AI-generated content may be incorrect. |

Fig 2: Three different sampling methods for PCM

1. QUANTIZATION:

The following are the steps in quantization:

a. We assume that the original analog signal has instantaneous amplitudes between Vmin and Vmax

b. We divide the range into L zones, each of height Δ (delta) Δ= 𝑉𝑚𝑎𝑥−𝑉𝑚𝑖𝑛 𝐿

c. We assign quantized values of 0 to L − 1 to the midpoint of each zone.

d. We approximate the value of the sample amplitude to the quantized values. As a simple example, assume that we have a sampled signal and the sample amplitudes are between −20 V and +20 V. We decide to have eight levels (L = 8). This means that Δ = 5 V

1. QUANTIZATION ERROR:

The contribution of the quantization error to the SNRdB of the signal depends on the number of quantization levels L, or the bits per sample nb, as shown in the following formula: � �𝑁𝑅𝑑𝐵 = 6.02𝑛𝑏 +1.76 dB

1. ENCODING:

The last step in PCM is encoding. After each sample is quantized and the number of bits per sample is decided, each sample can be changed to an nb-bit code word. Note that the number of bits for each sample is determined from the number of quantization levels. If the number of quantization levels is L, the number of bits is n𝑏 = 𝑙𝑜𝑔2(𝐿)

Index for Encoding, i = 𝑟𝑜𝑢𝑛𝑑((𝑥 − 𝑥𝑚𝑖𝑛)/𝛥)

Required bit rate (BR) for the encoding scheme can be determined as

B𝑅 = 𝑓𝑠 × 𝑛𝑏

|  |
| --- |
| A graph with numbers and lines  AI-generated content may be incorrect. |

Fig 3: Quantization and encoding of a sampled signal

**Results and Discussion**

**AB-CDEFG-H**

**ID: 23-50922-1**

**(a)** Show analog signal, sampled signal, and quantized signal.

|  |
| --- |
| **Code:** |
| >> clc;  clear;  %// Define parameters  F = 2;  G = 2;  %// Amplitude values  a1 = F + 1; %// 3  a2 = F + 3; %// 5  a3 = F + 2; %// 4  a4 = F + 4; %// 6  %// Frequency values  f1 = G + 5; %// 7 Hz  f2 = G + 7; %// 9 Hz  f3 = G + 1; %// 3 Hz  f4 = G + 2; %// 4 Hz  %// Time parameters  fs = 50; %// Sampling frequency (Hz)  T = 1; %// Signal duration (1 second)  t = linspace(0, T, 1000); %// Continuous time for analog signal  %// Define the analog signal  analog\_signal = a1 \* sin(2 \* pi \* f1 \* t) + ...  a2 \* cos(2 \* pi \* f2 \* t) + ...  a3 \* sin(2 \* pi \* f3 \* t) + ...  a4 \* sin(2 \* pi \* f4 \* t);  %// Sampling  t\_sampled = 0:1/fs:T;  sampled\_signal = a1 \* sin(2 \* pi \* f1 \* t\_sampled) + ...  a2 \* cos(2 \* pi \* f2 \* t\_sampled) + ...  a3 \* sin(2 \* pi \* f3 \* t\_sampled) + ...  a4 \* sin(2 \* pi \* f4 \* t\_sampled);  %// Quantization (8-bit, 256 levels)  quantization\_levels = 256;  min\_val = min(sampled\_signal);  max\_val = max(sampled\_signal);  quantized\_signal = round(((sampled\_signal - min\_val) / (max\_val - min\_val)) \* (quantization\_levels - 1));  quantized\_signal = (quantized\_signal / (quantization\_levels - 1)) \* (max\_val - min\_val) + min\_val;  %// Plot results  figure;  %// Plot Analog Signal  subplot(3,1,1);  plot(t, analog\_signal, 'b');  title('Analog Signal');  xlabel('Time (s)');  ylabel('Amplitude');  grid on;  %// Plot Sampled Signal  subplot(3,1,2);  plot(t, analog\_signal, 'b', 'LineWidth', 0.5);  hold on;  stem(t\_sampled, sampled\_signal, 'r', 'filled');  title('Sampled Signal');  xlabel('Time (s)');  ylabel('Amplitude');  grid on;  hold off;  %// Plot Quantized Signal  subplot(3,1,3);  stairs(t\_sampled, quantized\_signal, 'g', 'LineWidth', 1.5);  title('Quantized Signal');  xlabel('Time (s)');  ylabel('Amplitude');  grid on; |

|  |
| --- |
| **Result and Discussion:** |
| **Fig4: analog signal, sampled signal, and quantized signal.** |
|  The experiment demonstrated the process of converting an analog signal into a discrete digital representation using Pulse Code Modulation (PCM).   The analog signal was sampled at a specific frequency, and the sampled values were then quantized to discrete levels.   The results showed how increasing the sampling rate preserves more details of the original signal, while a lower sampling rate can lead to loss of information due to aliasing. |

**(b)** Show the digital data from the analog signal.

|  |
| --- |
| **Code:** |
| clc;  clear;  %// Define parameters  F = 2;  G = 2;  %// Amplitude values  a1 = F + 1; %// 3  a2 = F + 3; %// 5  a3 = F + 2; %// 4  a4 = F + 4; %// 6  %// Frequency values  f1 = G + 5; %// 7 Hz  f2 = G + 7; %// 9 Hz  f3 = G + 1; %// 3 Hz  f4 = G + 2; %// 4 Hz  %// Time parameters  fs = 50; %// Sampling frequency (Hz)  T = 1; %// Signal duration (1 second)  t\_sampled = 0:1/fs:T;  %// Compute sampled signal  sampled\_signal = a1 \* sin(2 \* pi \* f1 \* t\_sampled) + ...  a2 \* cos(2 \* pi \* f2 \* t\_sampled) + ...  a3 \* sin(2 \* pi \* f3 \* t\_sampled) + ...  a4 \* sin(2 \* pi \* f4 \* t\_sampled);  %// Quantization (8-bit, 256 levels)  quantization\_levels = 256;  min\_val = min(sampled\_signal);  max\_val = max(sampled\_signal);  quantized\_signal = round(((sampled\_signal - min\_val) / (max\_val - min\_val)) \* (quantization\_levels - 1));  %// Convert quantized values to 8-bit binary  binary\_data = dec2bin(quantized\_signal, 8);  %// Display first few binary values  disp('First 10 binary values from quantization:');  disp(binary\_data(1:10, :)); |
| **Result and Discussion:** |
| **Fig5: First 10 binary values from quantization** |
|  After quantization, the signal was converted into binary representation, demonstrating how digital data is stored and transmitted.   The higher the number of quantization levels, the better the approximation of the analog signal, but this also increases the bit rate required for transmission. |

**(c)** What are the appropriate values of sampling frequency and number of levels of quantization if minimum required SNR and bandwidth of the channel are 25 dB and 150 Hz respectively.

|  |
| --- |
| **Code:** |
| clc;  clear;  % Given parameters  SNR\_required = 25; % Minimum required SNR in dB  BW = 150; % Channel bandwidth in Hz  % Calculate minimum sampling frequency using Nyquist Theorem  fs\_min = 2 \* BW; % Minimum sampling frequency must be ≥ 2 × Bandwidth  % Solve for number of bits using the formula:  % SNR(dB) ≈ 6.02 \* N + 1.76  N = ceil((SNR\_required - 1.76) / 6.02); % Number of bits per sample  % Calculate number of quantization levels  L = 2^N;  % Display results  fprintf('Minimum Sampling Frequency: %.2f Hz\n', fs\_min);  fprintf('Number of Bits per Sample: %d bits\n', N);  fprintf('Number of Quantization Levels: %d levels\n', L); |
| **Result and Discussion:** |
| **Fig6: Calculate the number of quantization levels & Sampling frequency** |
|  The required sampling frequency was calculated using the Nyquist theorem to ensure accurate signal reconstruction.   The number of quantization levels was determined based on the required Signal-to-Noise Ratio (SNR) and bandwidth constraints.   The experiment highlighted the trade-off between quantization levels, bit rate, and channel bandwidth, which is crucial for efficient data transmission. |

**Conclusion**

In this experiment, we successfully implemented and analyzed the process of Analog-to-Digital Conversion (ADC) using MATLAB. We explored key concepts such as sampling, quantization, and encoding, which are essential for digital communication systems. The results demonstrated the importance of selecting an appropriate sampling rate and quantization levels to achieve a balance between signal accuracy and transmission efficiency. Understanding these concepts is fundamental for designing reliable digital communication and signal processing systems.