EXPERIMENT NO: 3

1. Aim: To study the process of Pulse Code Modulation (PCM) and Differential pulse code modulation and analyse the effects of on audio signals.

2. Software Used: MATLAB R2024b

3. Theory:

Pulse Code Modulation (PCM): Pulse Code Modulation (PCM) of an audio signal is a method to convert an analog audio waveform into a digital signal by sampling the waveform at regular intervals, quantizing the amplitude of each sample to a discrete level, and then representing each quantized value with a binary code, essentially translating the continuous audio signal into a series of digital bits that can be stored or transmitted digitally

- **Sampling:** The input analog signal is sampled at a fixed rate (Nyquist rate or higher) to obtain discrete-time signals.
- **Quantization:** The sampled values are rounded off to the nearest quantization level to reduce the infinite possibilities to a finite set of values.
- **Encoding:** Each quantized value is assigned a binary representation for digital storage or transmission.
- **Reconstruction (Decoding):** The binary data is converted back to an analog signal using a digital-to-analog converter (DAC).

To improve the quantization efficiency, **µ-law compression** is applied before quantization, which helps in reducing the quantization error for lower amplitude signals.

- **µ-law compression** is a non-linear method used to improve the efficiency of **Pulse Code Modulation (PCM)** by reducing the number of bits needed to accurately represent audio signals, particularly in telephony and audio systems.

$$F(x) = \text{sign}(x) \cdot \frac{\log(1+\mu|x|)}{\log(1+\mu)}$$

- x is the input signal (normalized between -1 and 1).
- µ is a constant.
- sign(x) preserves the sign of the input.

After transmission, the signal is **expanded (decompressed)** using the inverse function:

$$x = \text{sgn}(y)\frac{(1+\mu)^{|y|}-1}{\mu}$$

where y is the compressed signal.

Differential Pulse Code Modulation (DPCM): Differential Pulse Code Modulation (DPCM) is a signal encoding technique that encodes the difference between successive signal samples rather than encoding the absolute values of the samples themselves. DPCM is particularly effective when there is high correlation between consecutive samples, as it can significantly reduce the amount of data required to represent a signal.

In DPCM, instead of encoding the actual signal values (as done in Pulse Code Modulation, PCM), the encoder computes the difference between successive samples. These differences are typically smaller than the original samples, so fewer bits are required to represent the signal.

Encoding Process:

- **First Sample Transmission:** The first sample of the original signal is transmitted directly. This serves as the reference for the decoding process.
- **Difference Calculation:** The difference between successive samples is computed

$$\text{Difference} = d[n] = y[n] - y[n-1]$$

where y[n] is the current sample and y[n−1] is the previous sample

Quantization: The difference values are quantized to a finite set of values to reduce bit requirements.
Transmission: The quantized difference values along with the first sample are transmitted.
Decoding Process:

1. **Restoring the First Sample:** Since only differences are transmitted, the decoder starts with the first sample received.
2. **Signal Reconstruction:** Using the first sample and the received quantized difference values, the original signal is reconstructed iteratively:

$$y[n] = y[n-1] + d[n]$$

where y[n] is the reconstructed signal, and d[n] is the decoded difference.

The steps are:

- Load the audio signal:
- Initialize parameters:
- Apply μ-law compression:

$$compressed\_audio = sign(audio) \times \frac{\log(1 + \mu \times |audio|)}{\log(1 + \mu)}$$

- Quantization and Expansion for Different Bit Depths:
- Compute the number of quantization levels

$$bitdepth = 2^{(b-1)}$$

- Perform quantization using floor rounding

$$quantized\_audio = \frac{\lfloor compressed\_audio \times bitdepth \rfloor}{bitdepth}$$

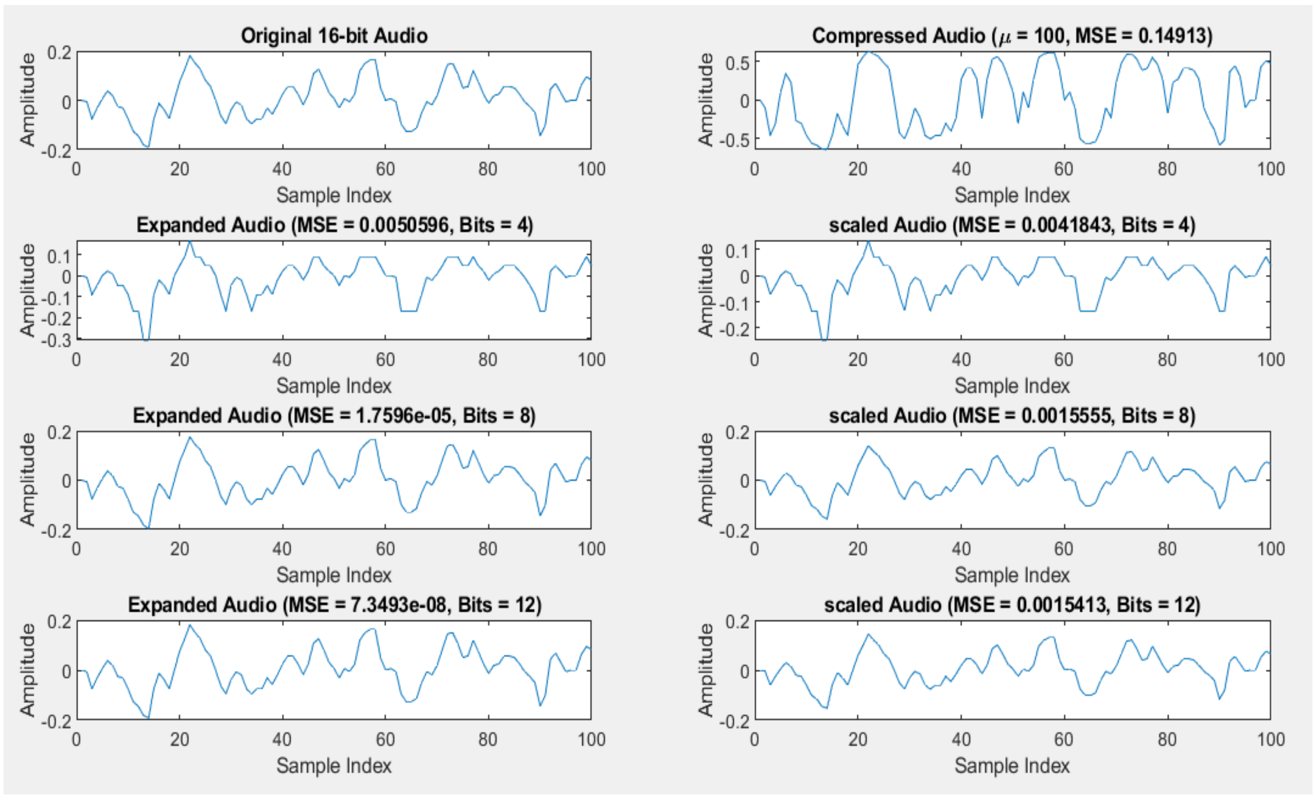
- Apply μ-law expansion using the formula:

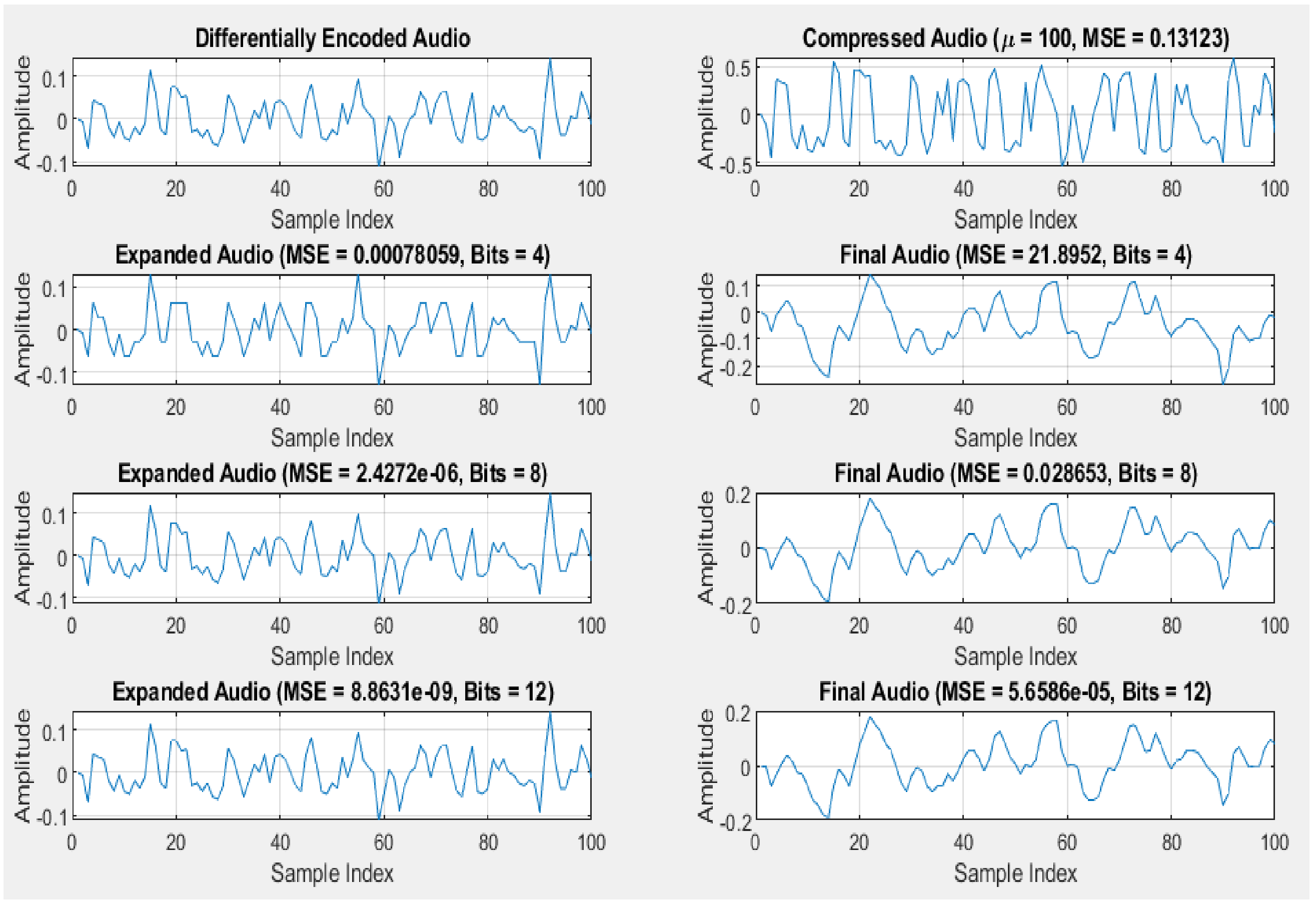
$$expanded\_audio = sign(quantized\_audio) \times \frac{(1 + \mu)^{|quantized\_audio|} - 1}{\mu}$$

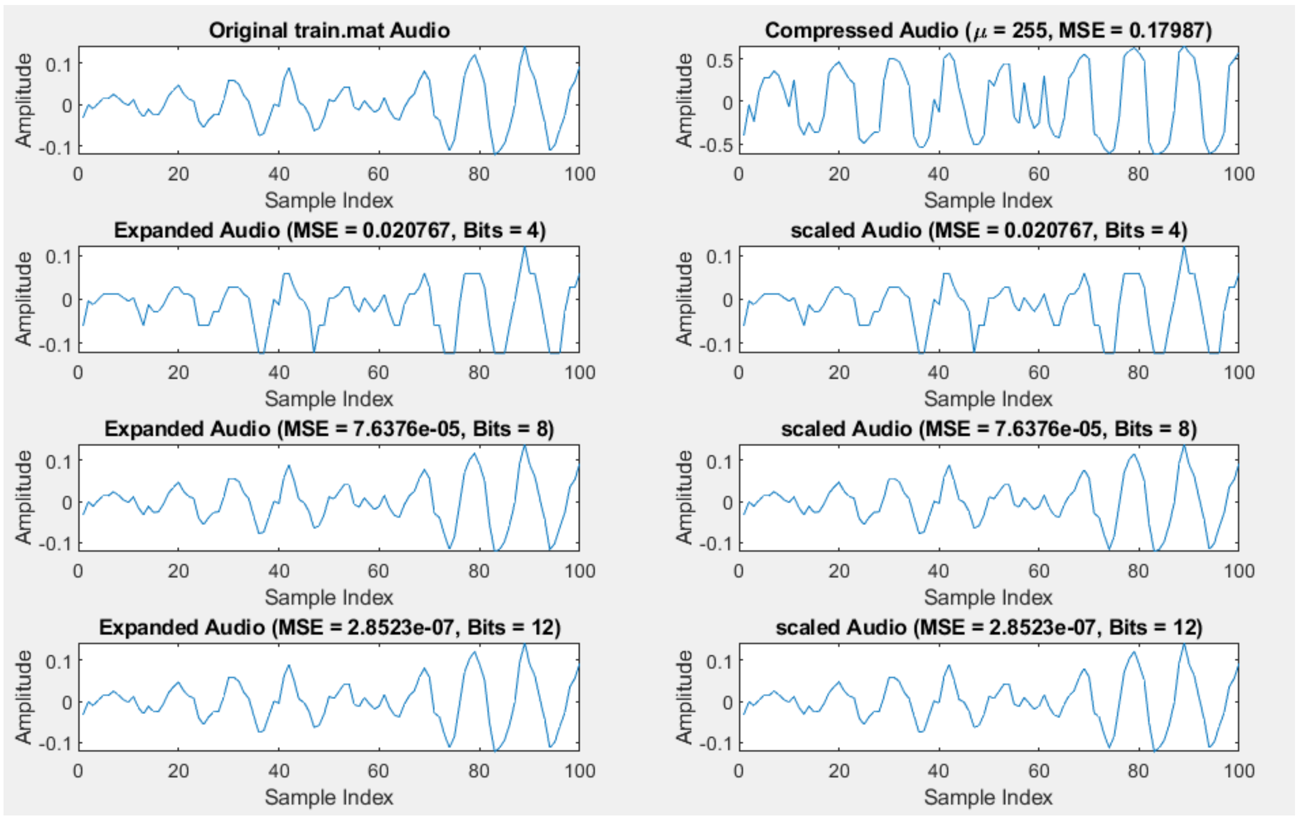
- Scale back to original amplitude

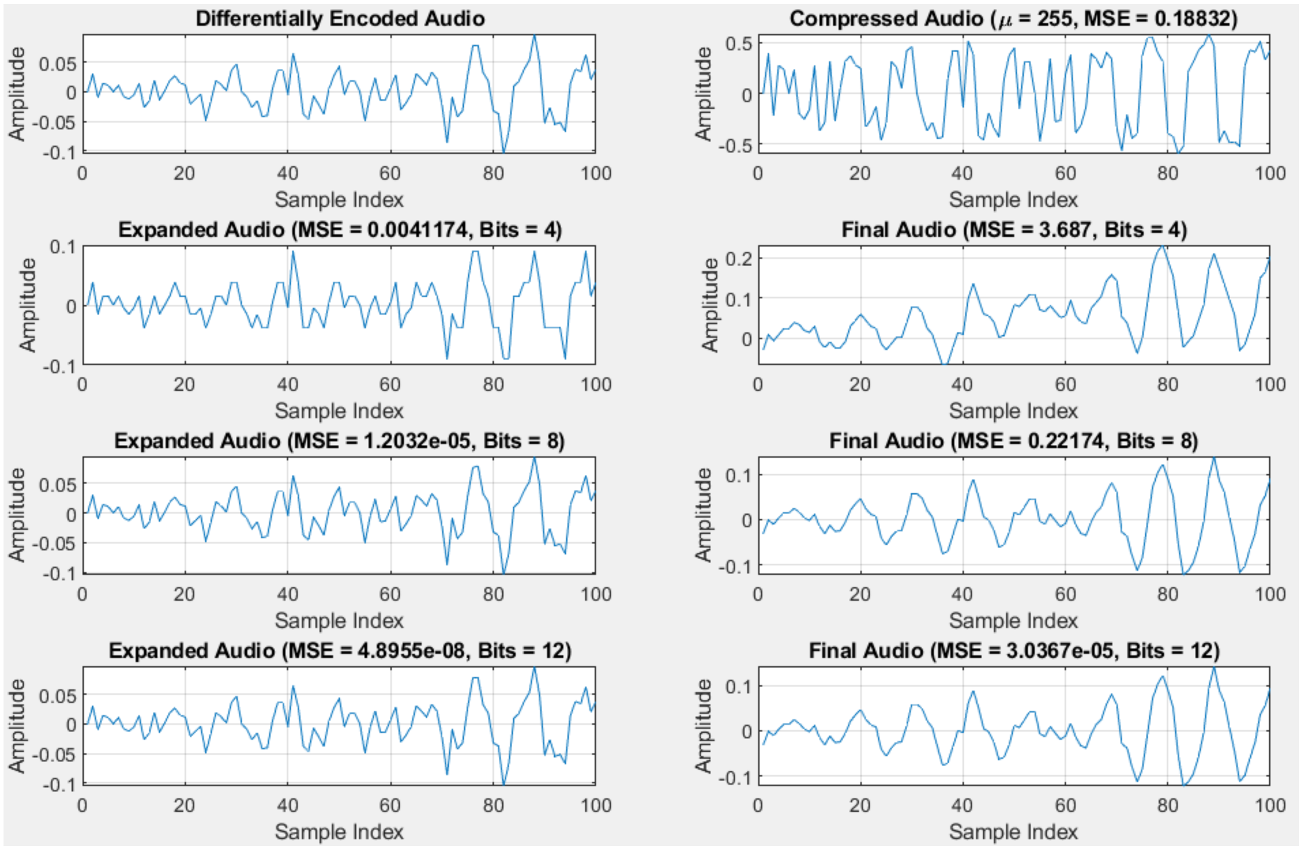
$$scaled\_audio = expanded\_audio \times \max(|audio|)$$

4.Result:


Original 16-bit Audio
Compressed Audio (μ = 100, MSE = 0.14913)
Expanded Audio (MSE = 0.0050596, Bits = 4)
scaled Audio (MSE = 0.0041843, Bits = 4)
Expanded Audio (MSE = 1.7596e-05, Bits = 8)
scaled Audio (MSE = 0.0015555, Bits = 8)
Expanded Audio (MSE = 7.3493e-08, Bits = 12)
scaled Audio (MSE = 0.0015413, Bits = 12)
Amplitude
Sample Index


Differentially Encoded Audio
Compressed Audio (μ = 100, MSE = 0.13123)
Expanded Audio (MSE = 0.00078059, Bits = 4)
Final Audio (MSE = 21.8952, Bits = 4)
Expanded Audio (MSE = 2.4272e-06, Bits = 8)
Final Audio (MSE = 0.028653, Bits = 8)
Expanded Audio (MSE = 8.8631e-09, Bits = 12)
Final Audio (MSE = 5.6586e-05, Bits = 12)
Amplitude
Sample Index





5. Discussion: We analyse the effects of varying the compression factor and quantization bit levels.

Pulse code modulation:

- As bit-depth increases, PCM can represent a wider range of amplitudes, leading to finer quantization and lower distortion, which is reflected in the decreasing MSE values as bit-depth increases.
- The low MSE values suggest that **PCM** is better suited for applications where high-fidelity reproduction of the audio signal is required.

DPCM (Differential Pulse Code Modulation):

- At lower bit-depths, quantization errors become more noticeable, and DPCM may struggle with accurately encoding and reconstructing signals with larger differences between adjacent samples.
- The **high MSE for 4-bit DPCM** suggests that the prediction model used in DPCM (typically based on the previous sample) is not sufficient to minimize errors, especially when there is a large variation between samples.

Impact of μ (Mu) Value:

- The value of **μ** (used for compression) plays a significant role in both methods:
  - **Higher μ values (255)** would result in stronger compression for both PCM and DPCM, potentially leading to greater differences between quantized values and the original signal. This could increase distortion, but the overall impact would vary depending on the method used.
  - For **DPCM**, increasing μ might worsen the prediction accuracy due to the larger compression ratio applied to the differences between samples, exacerbating the MSE.
  - In contrast, **PCM** might perform slightly better with higher μ values as it directly quantizes individual samples, but large μ values could still lead to a loss of fidelity.

Comparison of PCM and DPCM (Point-wise Summary)

- **Audio Quality:** PCM provides better reconstruction accuracy than DPCM, as indicated by lower MSE values.
- **Distortion:** DPCM suffers from higher distortion, especially at lower bit-depths (e.g., 4 bits), due to its predictive nature.
- **Bit-depth Effect:** Increasing the bit-depth reduces MSE for both PCM and DPCM, improving signal accuracy.
- **PCM Advantage:** PCM shows a more consistent and significant reduction in MSE as bit-depth increases.
- **DPCM Limitation:** DPCM struggles with predicting differences accurately at lower bit-depths, leading to poor fidelity.
- **Compression Benefit:** DPCM encodes differences between consecutive samples, reducing data rate requirements.

- **Efficiency Trade-off:** While DPCM offers compression benefits, it sacrifices reconstruction accuracy, especially at low bit-depths.
- **High Bit-depth Impact:** At higher bit-depths, DPCM improves but still does not match PCM's accuracy.
- **Application Suitability:** PCM is preferred for high-fidelity applications due to its superior accuracy.

6. Conclusion:

- Based on the **MSE values**, **PCM** is the **better method** for audio compression and reconstruction, especially at **higher bit-depths**. It produces consistently low MSE values and maintains good quality in the reconstructed audio signal, making it suitable for applications where audio fidelity is important.
- **DPCM**, while offering potential compression advantages, does not perform well at lower bit-depths, as seen by the **high MSE values** at 4 bits. At low bit-depths, the method struggles to predict the differences accurately, leading to **significant reconstruction errors**.

In summary, PCM offers higher fidelity at the cost of larger file sizes and higher bit rates, making it ideal for high-quality audio applications. However, DPCM is preferred when compression efficiency is more important than absolute fidelity, such as in telecommunications, low-bit-rate audio coding, and real-time audio transmission. The choice between PCM and DPCM ultimately depends on the specific requirements of the application, including available bandwidth, storage capacity, and quality expectations.

7. Code:

PCM:

```
load train.mat
audiowrite('train.wav', y, Fs);
mu = 255;   audio = y ;
figure;
subplot(4, 2, 1);     plot(audio(1:100));
title('Original train.mat Audio');   xlabel('Sample Index');   ylabel('Amplitude');
bit_values = [ 4, 8, 12];
compressed_audio = sign(audio) .* (log(1 + mu * abs(audio)) / log(1 + mu));
mse_compressed = mean((compressed_audio - audio).^2);
subplot(4, 2, 2);   plot(compressed_audio(1:100));
title(['Compressed Audio (\mu = ', num2str(mu),', MSE = ', num2str(mse_compressed), ')']);
xlabel('Sample Index');   ylabel('Amplitude');
for i = 1: 3
   bitdepth = 2^(bit_values(i)-1);
   floor_r = floor(compressed_audio * bitdepth);
   quantized_audio = floor_r / bitdepth;
   expanded_audio = sign(quantized_audio) .* ((1 + mu).^abs(quantized_audio) - 1) / mu;
   scaled_audio = expanded_audio * max(abs(audio));
   mse_expanded = mean((expanded_audio - audio).^2);
```

```
    mse_scaled = mean((scaled_audio - audio).^2);
    subplot(4, 2, i*2 +1);       plot(expanded_audio(1:100));
    title(['Expanded (MSE = ', num2str(mse_expanded),', Bits = ', num2str(bit_values(i)), ')']);
    xlabel('Sample Index');      ylabel('Amplitude');
    subplot(4, 2, i*2 +2);       plot(scaled_audio(1:100));
    title(['scaled Audio (MSE = ', num2str(mse_scaled),', Bits = ', num2str(bit_values(i)), ')']);
    xlabel('Sample Index');      ylabel('Amplitude');
end
```

DPCM:

```
load train.mat
audiowrite('train.wav', y, Fs);
mu = 255;   bit_values = [4, 8, 12];
audio = zeros(length(y), 1);
for i = 2:length(y)
    audio(i) = y(i) - y(i-1);
end
figure;    subplot(4, 2, 1);   plot(audio(1:100));
title('Differentially Encoded Audio');   xlabel('Sample Index');  ylabel('Amplitude');
compressed_audio = sign(audio) .* (log(1 + mu * abs(audio)) / log(1 + mu));
mse_compressed = mean((compressed_audio - audio).^2);
subplot(4, 2, 2);   plot(compressed_audio(1:100));
title(['Compressed Audio (\mu = ', num2str(mu),', MSE = ', num2str(mse_compressed), ')']);
xlabel('Sample Index');   ylabel('Amplitude');
for i = 1:3
    bitdepth = 2^(bit_values(i) - 1);
    quantized_audio = round(compressed_audio * (bitdepth - 1)) / (bitdepth - 1);
    expanded_audio = sign(quantized_audio) .* ((1 + mu).^abs(quantized_audio) - 1) / mu;
    final = zeros(length(y), 1);
    final(1) = y(1);
    for j = 2:length(y)
        final(j) = expanded_audio(j) + final(j-1);
    end
    mse_expanded = mean((expanded_audio - audio).^2);
    mse_final = mean((final - y).^2);
    subplot(4, 2, i*2 + 1);       plot(expanded_audio(1:100));
    title(['Expanded (MSE = ', num2str(mse_expanded), ', Bits = ', num2str(bit_values(i)), ')']);
    xlabel('Sample Index');      ylabel('Amplitude');
    subplot(4, 2, i*2 + 2);      plot(final(1:100));
    title(['Final Audio (MSE = ', num2str(mse_final), ', Bits = ', num2str(bit_values(i)), ')']);
    xlabel('Sample Index');
    ylabel('Amplitude');
end
```