Cairo University - Faculty Of Engineering
Computer Engineering Department
Digital Communication - Spring 2025

Digital Communications: Assignment 1 (Quantization)

Submitted to

Dr. Mai Badawi
Dr. Hala
Eng. Mohamed Khaled

Submitted by

Akram Hany Sec 1 BN 14
Amir Anwar Sec 1 BN 15

Contents

List of Figures

1 Introduction

In this report, we provide our solution / implementation for Quantization. Covering concepts like midtread and midrise, SNR, uniform and non-uniform quantizers.

2 General Theoretical Background

2.1 Uniform Quantization

- Number of levels: $L = 2^n$ where n is the number of bits
- Level amplitude: $\Delta = \frac{A_{max} - A_{min}}{L}$
- Quantization error: $q = \frac{\Delta}{2}$

The quantization error is bounded by: $-\frac{\Delta}{2} < q \leq \frac{\Delta}{2}$

2.2 Signal-to-Noise Ratio (SNR)

For uniform quantization, the theoretical SNR can be calculated as:

$$SNR_{avg} = \frac{3\alpha^2 L^2}{\Delta^2/12} = 3\alpha^2 2^{2n}$$

Where α is a signal-dependent parameter. In decibels:

$$SNR_{Peak} = 10\log_{10}(3\alpha^2) + 6n$$

2.3 Midtread and Midrise Quantizers

The types of uniform quantizers are midtread and midrise:

- **Midtread Quantizer**: voltage levels includes zero
- **Midrise Quantizer**: voltage levels doesn't include a level at zero

2.4 Non-uniform Quantization

For non-uniform signals in uniform quantizers, the SNR performance degrades. To improve performance, non-uniform quantization through using a compressor and an expander can be used, the SQNR given by:

$$SQNR_{non-uniform} = \frac{3L^2}{\ln^2(1+\mu)}$$

Where μ is the compression parameter.

3 Results and comments

3.1 Midtread and Midrise Quantization (part 1,2,3)

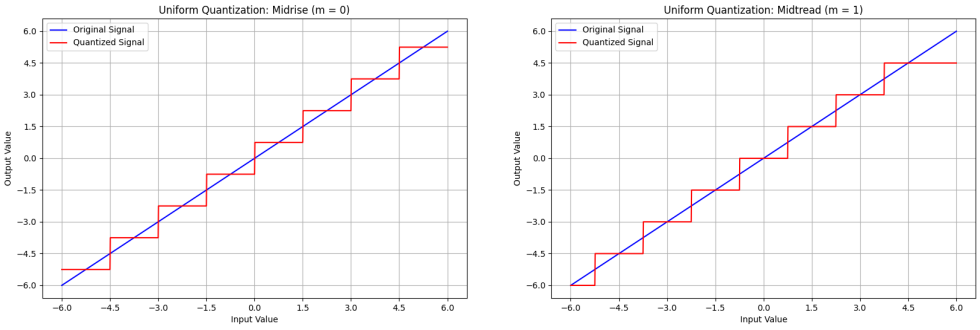

Figure 1: Original (input) signal and its dequantized version using midtread (m=1) and midrise (m=0)

Figure 1 shows the original signal and its quantized version for both midtread (m=1) and midrise (m=0) quantizers. The midtread quantizer includes a level at zero, while the midrise quantizer has levels at half-integer values.

Note: We adjusted the lower bound clipping in the quantizer implementation to allow the -6 level to appear in the output, which causes the quantization to appear asymmetric (showing 7 levels instead of the expected 8 levels). This adjustment was made to better demonstrate the behavior of the quantizer at the signal extremes. And to match our tutorial quantizer graphs.

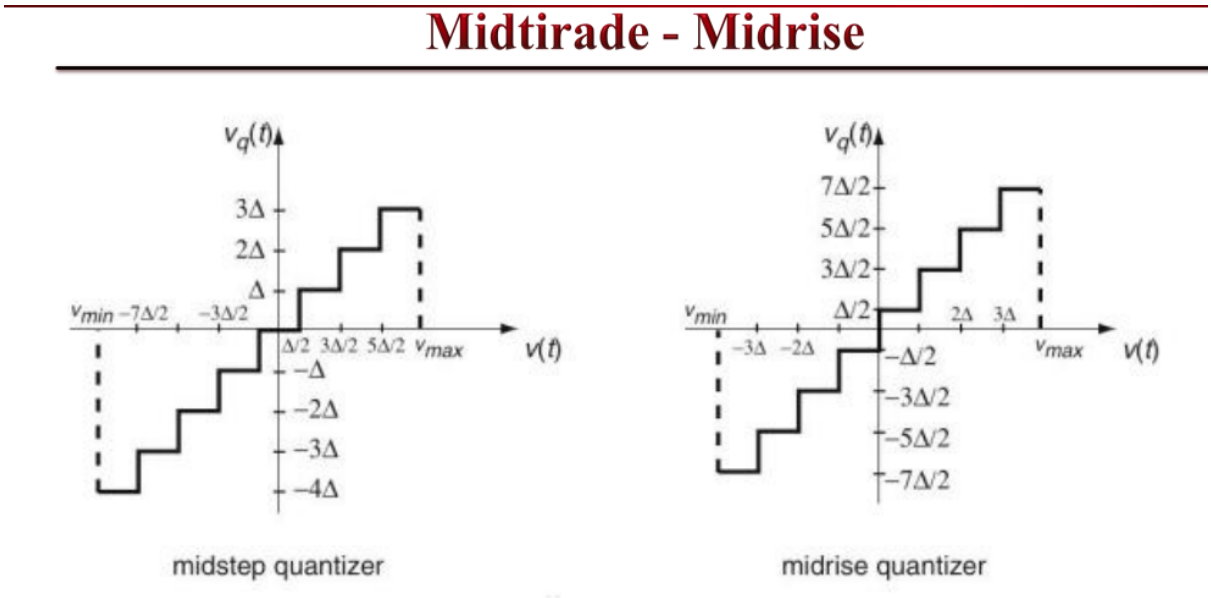

Figure 2: Tutorial midrise and midtread demonostration graph

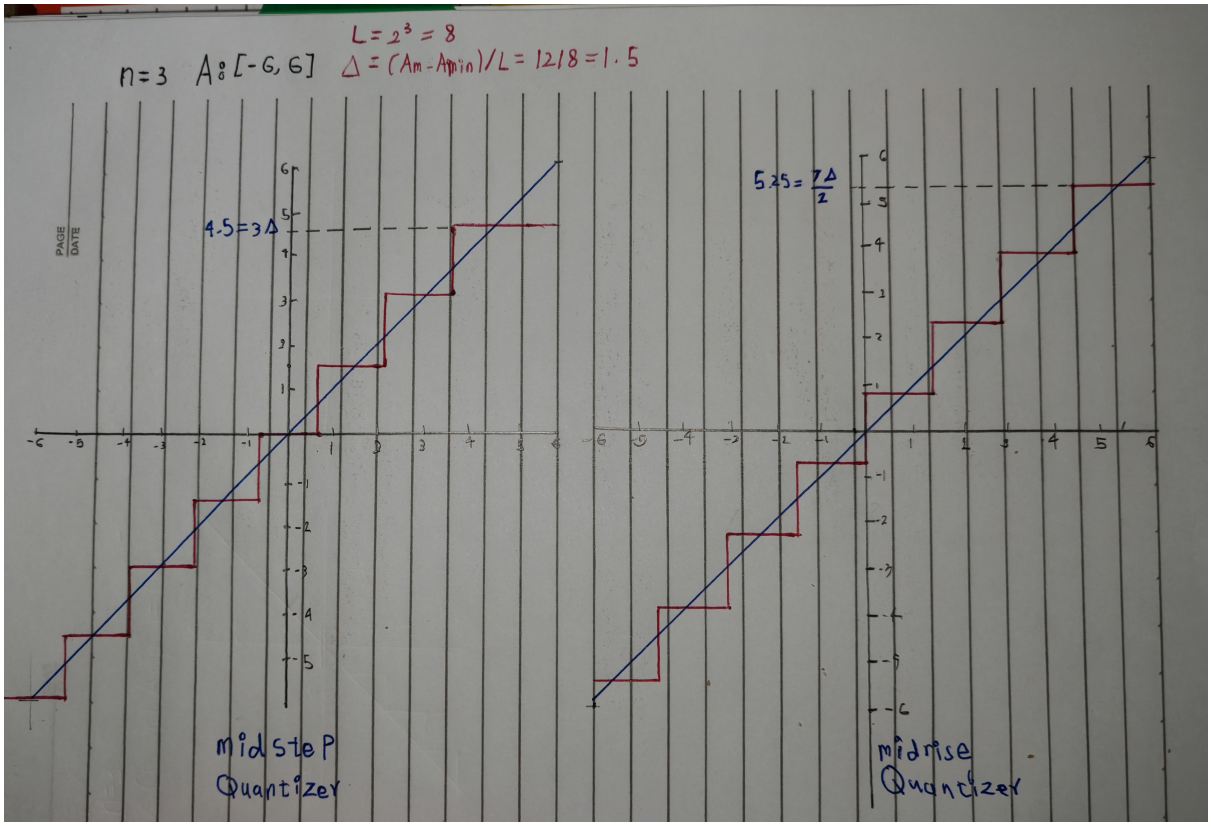

Figure 3: Manual calculation and hand-drawing of quantization processes for verification

Figure 3 presents a manual calculation and hand-drawing of the same quantization processes to verify the results. Showing on the 2 graphs:

1. the signal maximum value for both
2. midtrade have a level at 0 while midrise doesn't
3. midrise have equal number of levels above and under 0
4. midtrade levels are integer multiple of Δ
5. midrise step (input level boundaries) are integer multiple of Δ

Note: All equations for this part are in the *General Theoretical Background* Section.

3.2 SNR Analysis for uniform signal (part 4)

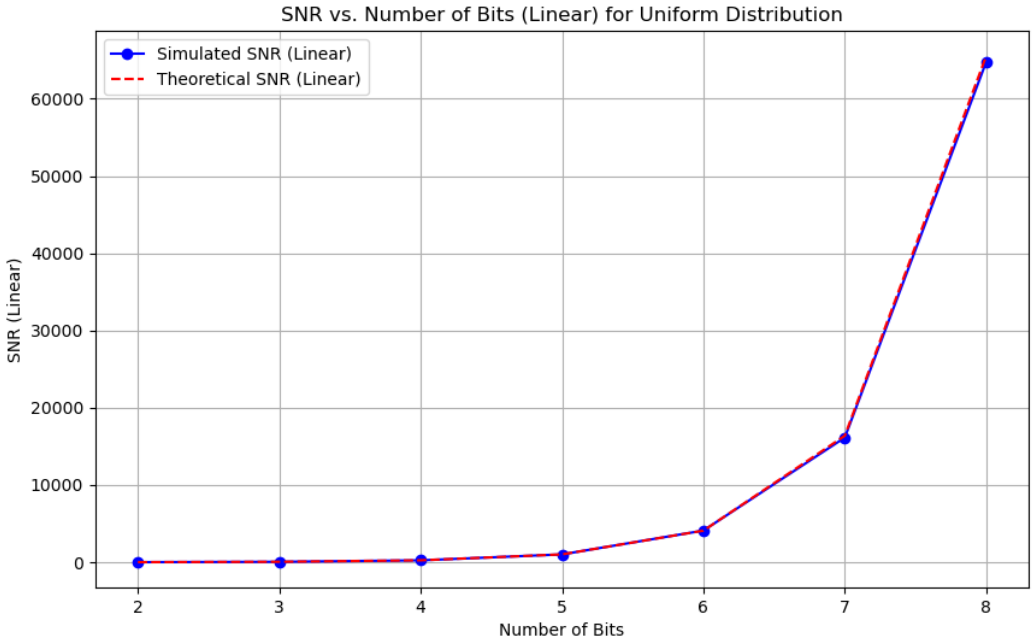

Figure 4: SNR across different bit depths (n) - Linear scale

Figure 4 displays the SNR across different bit depths (n) for both theoretical and calculated values in linear scale. As expected, the SNR increases linearly with the number of bits.

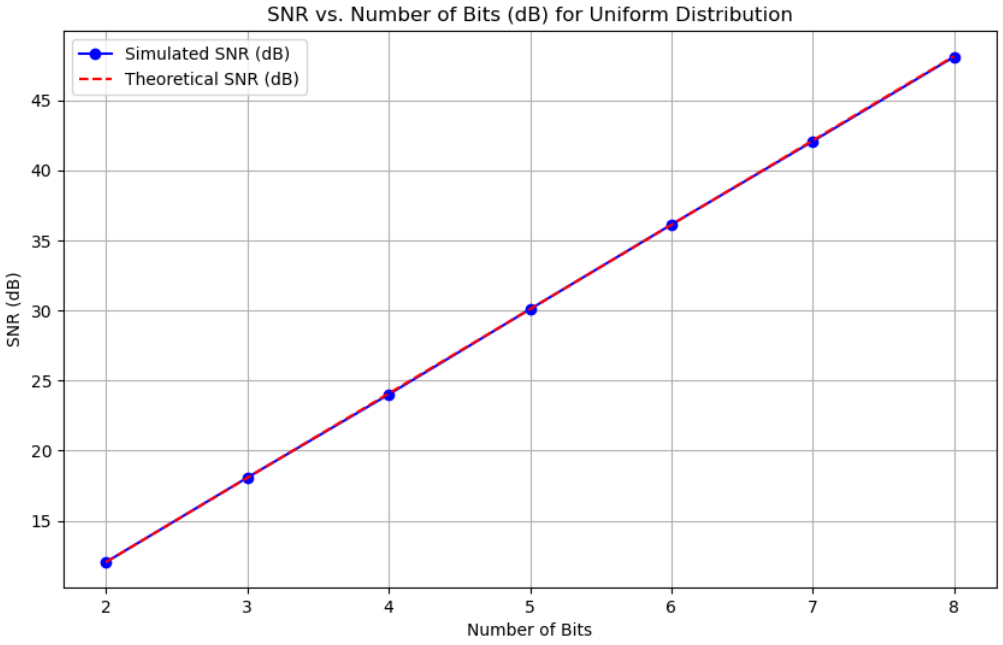

Figure 5: SNR across different bit depths (n) - Decibel scale

Figure 5 presents the same data in decibel scale, showing the expected 6 dB improvement in SNR for each additional bit of quantization. Additionaly because we have a uniform signal the Simulated/-calculated SNR almost matches with the theoretical SNR.

3.3 SNR Analysis for nonuniform signal (part 5)

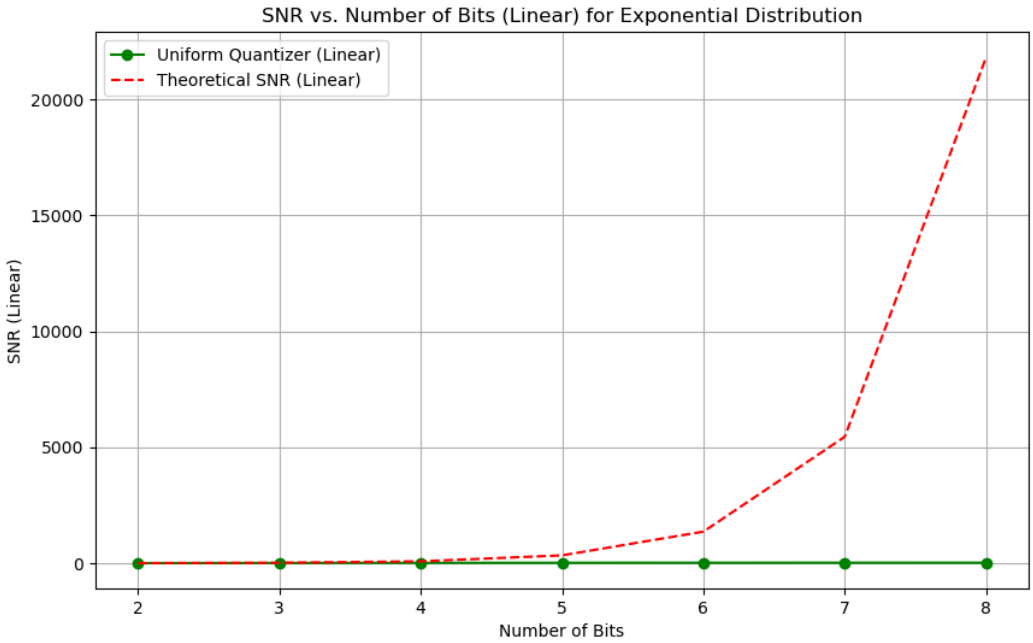

Figure 6: SNR for non-uniform signals in uniform quantizers - Linear scale

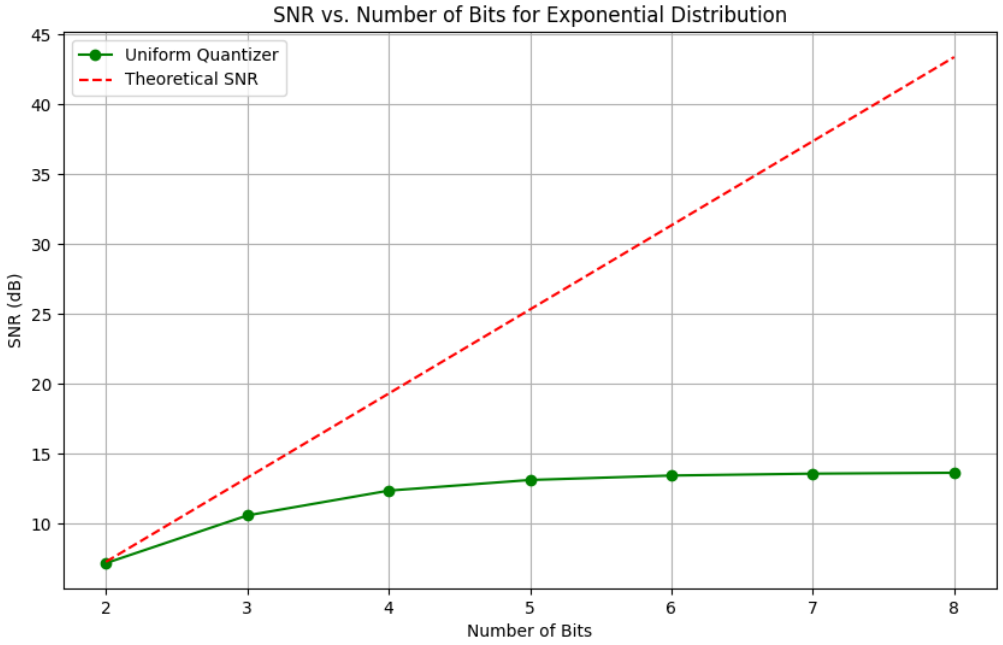

Figure 7: SNR for non-uniform signals in uniform quantizers - Decibel scale

Figures 6 and 7 show the SNR performance for non-uniform signals processed through uniform quantizers, highlighting the degradation in performance compared to uniform signals.

This happens because the signal is not equally distributed between the levels like in uniform signals (for example all signals most of the signals should be centered in one level increasing the error significantly). Even though this happens uniform quantizer could still be used using compressor and expander before and after the quantization and dequantization.

3.4 Compressor Expander to use uniform quantizer with nonuniform signals

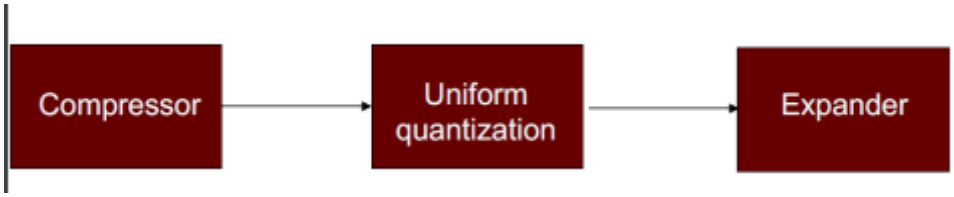

Figure 8: Compressor Quantizer Expander block diagram

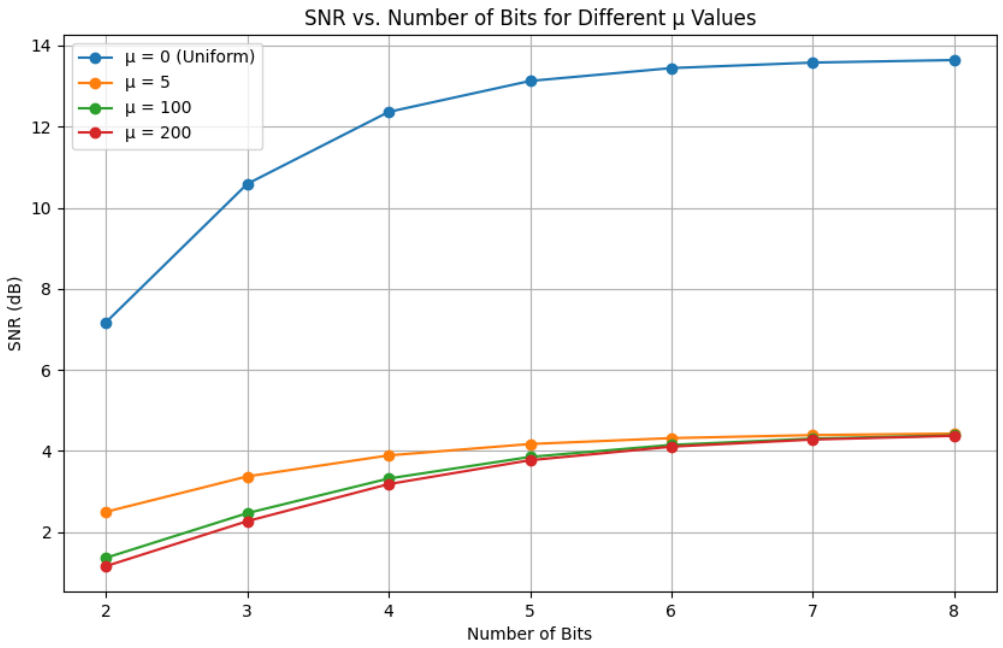

Figure 9: Effect of compression parameter (μ) on SNR performance

Figure 9 illustrates the effect of different compression parameters (μ) on the SNR performance of the compander-expander system. Appropriate selection of μ can optimize performance for specific signal distributions.

4 Conclusion

Our analysis confirms the theoretical expectations regarding quantization performance:

1. The SNR improves by approximately 6 dB for each additional bit in the quantizer
2. Midtread and midrise quantizers show different characteristics, particularly in how they handle signals near zero
3. Non-uniform signals experience degraded performance in uniform quantizers
4. using (compressing and expanding) can significantly improve performance for non-uniform signals when the compression parameter is properly selected

A Python Implementation

A.1 Quantizer and Dequantizer Implementation

```python
import numpy as np

def uniform_quantizer(in_val, n_bits, xmax, m):
    L = 2 ** n_bits
    delta = (2 * xmax) / L

    lower_bound = -xmax
    higher_bound = m * delta / 2 + xmax

    in_val_clipped = np.clip(in_val, lower_bound, higher_bound)
    q_ind = np.floor((in_val_clipped + (m * delta / 2) + xmax) / delta)
    q_ind = np.clip(q_ind, 0, L - 1).astype(int)
```

```
    return q_ind

def uniform_dequantizer(q_ind, n_bits, xmax, m):
    L = 2 ** n_bits
    delta = (2 * xmax) / L

    deq_val = (q_ind * delta) - xmax + ((1 - m) * delta / 2)

    return deq_val
```

Listing 1: Uniform Quantizer and Dequantizer Functions

A.2 Midtread and Midrise Visualization

```
def plot_midtread_midrise_comparison():
    x = np.arange(-6, 6.01, 0.01)
    n_bits = 3
    xmax = 6

    # Test for m = 0 (midrise)
    m = 0
    q_ind = uniform_quantizer(x, n_bits, xmax, m)
    y_midrise = uniform_dequantizer(q_ind, n_bits, xmax, m)

    # Test for m = 1 (midtread)
    m = 1
    q_ind = uniform_quantizer(x, n_bits, xmax, m)
    y_midtread = uniform_dequantizer(q_ind, n_bits, xmax, m)

    plt.figure(figsize=(20, 6))

    plt.subplot(1, 2, 1)
    plt.plot(x, x, "b-", linewidth=1.5, label="Original Signal")
    plt.plot(x, y_midrise, "r-", linewidth=1.5, label="Quantized Signal")
    plt.title("Uniform Quantization: Midrise (m = 0)")
    plt.xlabel("Input Value")
    plt.ylabel("Output Value")
    plt.legend()
    plt.grid(True)
    plt.xticks(np.arange(-6, 7, 1.5))
    plt.yticks(np.arange(-6, 7, 1.5))

    plt.subplot(1, 2, 2)
    plt.plot(x, x, "b-", linewidth=1.5, label="Original Signal")
    plt.plot(x, y_midtread, "r-", linewidth=1.5, label="Quantized Signal")
    plt.title("Uniform Quantization: Midtread (m = 1)")
    plt.xlabel("Input Value")
    plt.ylabel("Output Value")
    plt.legend()
    plt.grid(True)
    plt.xticks(np.arange(-6, 7, 1.5))
    plt.yticks(np.arange(-6, 7, 1.5))

    plt.savefig("quantization_comparison.png")
    plt.show()
```

Listing 2: Midtread and Midrise Quantization Visualization

A.3 Uniform Signal SNR Calculation

```
def calculate_uniform_signal_snr():
    np.random.seed(42)
    num_samples = 10000
    x_unif = -5 + 10 * np.random.rand(num_samples)
    xmax = 5
    m = 0

    bits_range = np.arange(2, 9)
    snr_sim_db = np.zeros_like(bits_range, dtype=float)
```

```
    snr_theory_db = np.zeros_like(bits_range, dtype=float)
    snr_sim_ln = np.zeros_like(bits_range, dtype=float)
    snr_theory_ln = np.zeros_like(bits_range, dtype=float)

    for i, n_bits in enumerate(bits_range):
        q_ind = uniform_quantizer(x_unif, n_bits, xmax, m)
        y = uniform_dequantizer(q_ind, n_bits, xmax, m)

        error = x_unif - y
        signal_power = np.mean(x_unif**2)
        noise_power = np.mean(error**2)
        snr_sim_db[i] = 10 * np.log10(signal_power / noise_power)
        snr_sim_ln[i] = signal_power / noise_power

        snr_theory_db[i] = 6.02 * n_bits
        snr_theory_ln[i] = 2 ** (2 * n_bits)

    return bits_range, snr_sim_db, snr_theory_db, snr_sim_ln, snr_theory_ln
```

Listing 3: SNR Calculation for Uniform Signal

A.4 Uniform Signal SNR Visualization

```
def plot_uniform_signal_snr(bits_range, snr_sim_db, snr_theory_db, snr_sim_ln,
    snr_theory_ln):
    plt.figure(figsize=(20, 6))

    plt.subplot(1, 2, 1)
    plt.plot(bits_range, snr_sim_db, "bo-", linewidth=1.5, label="Simulated SNR (dB)")
    plt.plot(bits_range, snr_theory_db, "r--", linewidth=1.5, label="Theoretical SNR (dB
    )")
    plt.title("SNR vs. Number of Bits (dB) for Uniform Distribution")
    plt.xlabel("Number of Bits")
    plt.ylabel("SNR (dB)")
    plt.legend()
    plt.grid(True)

    plt.subplot(1, 2, 2)
    plt.plot(bits_range, snr_sim_ln, "bo-", linewidth=1.5, label="Simulated SNR (Linear)
    ")
    plt.plot(bits_range, snr_theory_ln, "r--", linewidth=1.5, label="Theoretical SNR (
    Linear)")
    plt.title("SNR vs. Number of Bits (Linear) for Uniform Distribution")
    plt.xlabel("Number of Bits")
    plt.ylabel("SNR (Linear)")
    plt.legend()
    plt.grid(True)

    plt.savefig("uniform_snr.png")
    plt.show()
```

Listing 4: Uniform Signal SNR Visualization

A.5 Non-uniform Signal SNR Calculation

```
def calculate_nonuniform_signal_snr():
    np.random.seed(42)
    num_samples = 10000
    mag = np.random.exponential(scale=1.0, size=num_samples)
    polarity = np.sign(np.random.rand(num_samples) - 0.5)
    x_exp = polarity * mag

    xmax = 3
    m = 0

    bits_range = np.arange(2, 9)
    snr_exp = np.zeros_like(bits_range, dtype=float)
    snr_theory = np.zeros_like(bits_range, dtype=float)

    for i, n_bits in enumerate(bits_range):
```

```
        q_ind = uniform_quantizer(x_exp, n_bits, xmax, m)
        y = uniform_dequantizer(q_ind, n_bits, xmax, m)

        error = x_exp - y
        signal_power = np.mean(x_exp**2)
        noise_power = np.mean(error**2)
        snr_exp[i] = 10 * np.log10(signal_power / noise_power)
        snr_theory[i] = 6.02 * n_bits - 4.77

    return bits_range, snr_exp, snr_theory
```

Listing 5: Non-uniform Signal SNR Calculation

A.6 Non-uniform Signal SNR Visualization

```

def plot_nonuniform_signal_snr(bits_range, snr_exp, snr_theory):
    plt.figure(figsize=(10, 6))
    plt.plot(bits_range, snr_exp, "go-", linewidth=1.5, label="Uniform Quantizer")
    plt.plot(bits_range, snr_theory, "r--", linewidth=1.5, label="Theoretical SNR")
    plt.title("SNR vs. Number of Bits for Exponential Distribution")
    plt.xlabel("Number of Bits")
    plt.ylabel("SNR (dB)")
    plt.legend()
    plt.grid(True)
    plt.savefig("nonuniform_snr.png")
    plt.show()
```

Listing 6: Non-uniform Signal SNR Visualization

A.7 μ-law Functions

```

def mu_law_compressor(x, mu):
    return np.sign(x) * np.log(1 + mu * np.abs(x)) / np.log(1 + mu)

def mu_law_expander(y, mu):
    return np.sign(y) * (1 / mu) * ((1 + mu) ** np.abs(y) - 1)
```

Listing 7: μ-law Functions

A.8 μ-law SNR Analysis

```
def analyze_mu_law_performance():
    np.random.seed(42)
    num_samples = 10000
    mag = np.random.exponential(scale=1.0, size=num_samples)
    polarity = np.sign(np.random.rand(num_samples) - 0.5)
    x_exp = polarity * mag

    xmax = 3
    m = 0

    mu_values = [0, 5, 100, 200]
    bits_range = np.arange(2, 9)

    snr_mu = np.zeros((len(mu_values), len(bits_range)))

    for mu_idx, mu in enumerate(mu_values):
        for bit_idx, n_bits in enumerate(bits_range):
            if mu == 0:
                q_ind = uniform_quantizer(x_exp, n_bits, xmax, m)
                y = uniform_dequantizer(q_ind, n_bits, xmax, m)
            else:
                x_expanded = mu_law_compressor(x_exp, mu)
                scale_factor = xmax
                x_expanded_scaled = x_expanded * scale_factor

```

```
                q_ind = uniform_quantizer(x_expanded_scaled, n_bits, xmax, m)
                y_expanded = uniform_dequantizer(q_ind, n_bits, xmax, m)

                y_expanded_scaled = y_expanded / scale_factor
                y = mu_law_expander(y_expanded_scaled, mu)

            error = x_exp - y
            signal_power = np.mean(x_exp**2)
            noise_power = np.mean(error**2)
            snr_mu[mu_idx, bit_idx] = 10 * np.log10(signal_power / noise_power)

    return bits_range, mu_values, snr_mu
```

Listing 8: μ-law SNR Analysis

A.9 μ-law Visualization

```
def plot_mu_law_performance(bits_range, mu_values, snr_mu):
    plt.figure(figsize=(10, 6))
    for mu_idx, mu in enumerate(mu_values):
        label = "$\mu$ = " + str(mu)
        if mu == 0:
            label += " (Uniform)"
        plt.plot(bits_range, snr_mu[mu_idx, :], "o-", linewidth=1.5, label=label)

    plt.title("SNR vs. Number of Bits for Different $\mu$ Values")
    plt.xlabel("Number of Bits")
    plt.ylabel("SNR (dB)")
    plt.legend()
    plt.grid(True)
    plt.savefig("mu_law_comparison.png")
    plt.show()
```

Listing 9: μ-law Visualization

A.10 μ-law Characteristic Curve

```
def plot_mu_law_characteristic():
    x_range = np.linspace(-1, 1, 1000)
    mu_demo = 255
    y_compressed = mu_law_compressor(x_range, mu_demo)

    plt.figure(figsize=(10, 6))
    plt.plot(x_range, x_range, 'r--', label='Linear (No Compression)')
    plt.plot(x_range, y_compressed, 'b-', label=f'$\mu$-law Compressed ($\mu$={mu_demo})
    ')
    plt.grid(True)
    plt.xlabel('Input Signal Amplitude')
    plt.ylabel('Output Signal Amplitude')
    plt.title('$\mu$-law Compression Characteristic')
    plt.legend()
    plt.savefig('mu_law_characteristic.png')
    plt.show()
```

Listing 10: μ-law Characteristic Curve