Name : SahilYadav
Roll No. : 21ECE1039
Course : Digital Communication Assignment

Experiment No. - 3

Generation of Delta Modulation and its demodulation

Aim :-

Introduction of basic concepts of Delta Modulation (DM) system with simulation.

Tools Used :-

Anaconda's Spyder Python Simulation Software.

Theory :-

Delta Modulation is a special case of DPCM. In DPCM scheme if the base band signal is sampled at a rate much higher than the Nyquist rate purposely to increase the correlation between adjacent samples of the signal, so as to permit the use of a simple quantizing strategy for constructing the encoded signal, Delta modulation (DM) is precisely such a scheme. Delta Modulation is the one-bit (or two-level) versions of DPCM

DM provides a staircase approximation to the over sampled version of an input baseband signal. The difference between the input and the approximation is
quantized into only two levels, namely, ±δ corresponding to positive and negative differences, respectively, Thus, if the approximation falls below the signal at any sampling epoch, it is increased by δ. Provided that the signal does not change too rapidly from sample to sample, we find that the staircase approximation remains within ±δ of the input signal. The symbol δ denotes the absolute value of the two representation levels of the one-bit quantizer used in the DM

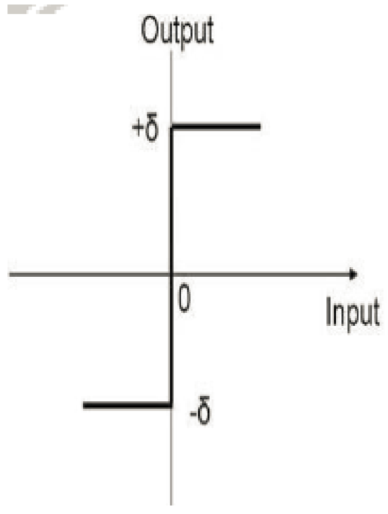


Fig 2.2.3: Input and output characteristics of Delta Modulator
(Source: Tutorials Point)

Let the input signal be x(t) and the staircase approximation to it is u(t)
In the receiver the staircase approximation u(t) is reconstructed by passing the incoming sequence of positive and negative pulses through an accumulator in a manner similar to that used in the transmitter. The out-of –band quantization noise in the high frequency staircase waveform u(t) is rejected by passing it through a low- pass filter with a band-width equal to the original signal bandwidth. Delta modulation offers two unique features
. A block diagram of DM system is given in Figure 3.1

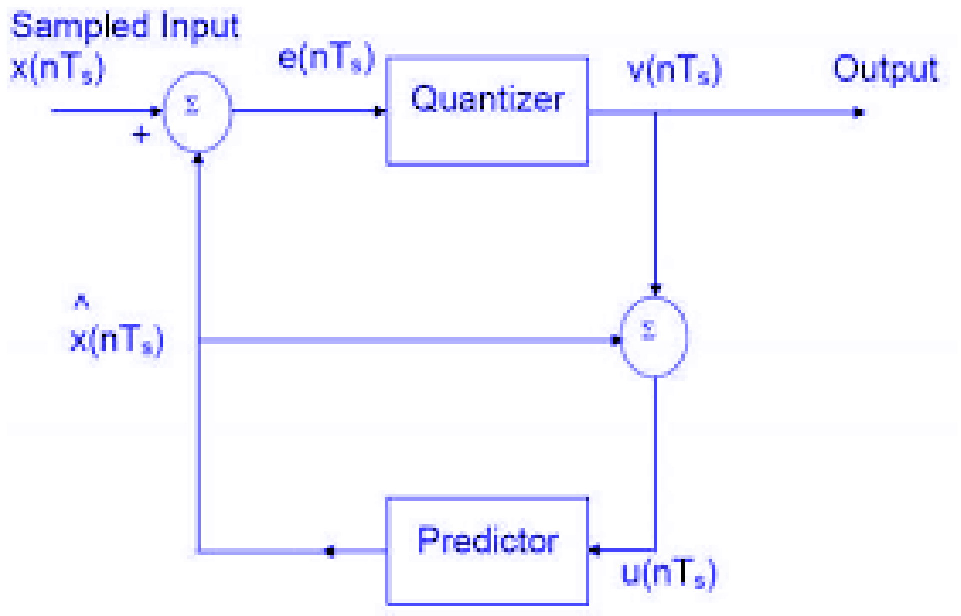


Fig 3.1: Block Diagram of DPCM Transmitter
(Source: Electronics Post)

Advantage of DM:

1. No need for Word Framing because of one-bit code word.
2. Simple design for both Transmitter and Receiver

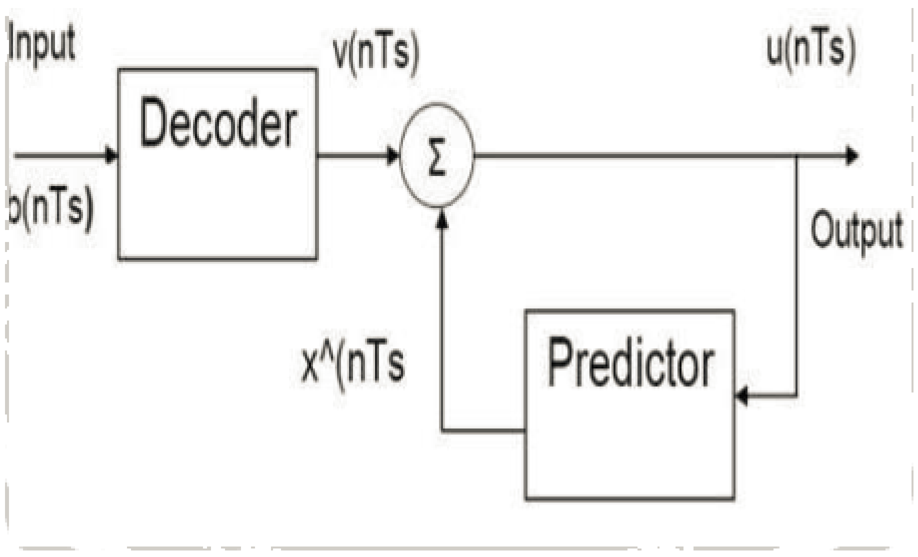


Fig 2.2.2: Block diagram of DPCM Receiver
(Source: Electronics Post)

Disadvantage of DM:
Delta modulation systems are subject to two types of quantization error:

1. slope –overload distortion, and
2. granular noise

Adaptive Delta Modulation:
The performance of a delta modulator can be improved significantly by making the size of the modulator assumes a time-varying form. In particular, during a step segment of the input signal the step size is increased. Conversely, when the input signal is varying slowly, the step size is reduced. In this way, the size is adapted to the level of the input signal. The resulting method is called adaptive delta modulation (ADM). There are several types of ADM, depending on the type of scheme used for adjusting the step size. In this ADM, a discrete set of values is provided for the step size

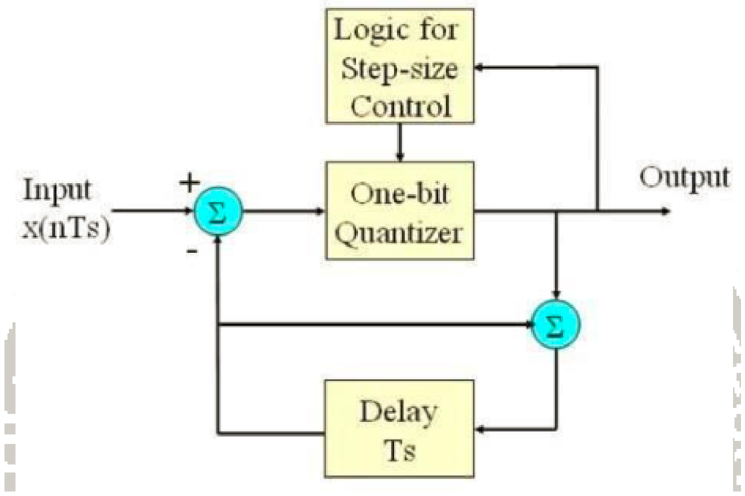


Fig 3.2 Block diagram of ADM Transmitter
(Source: ElProCus)

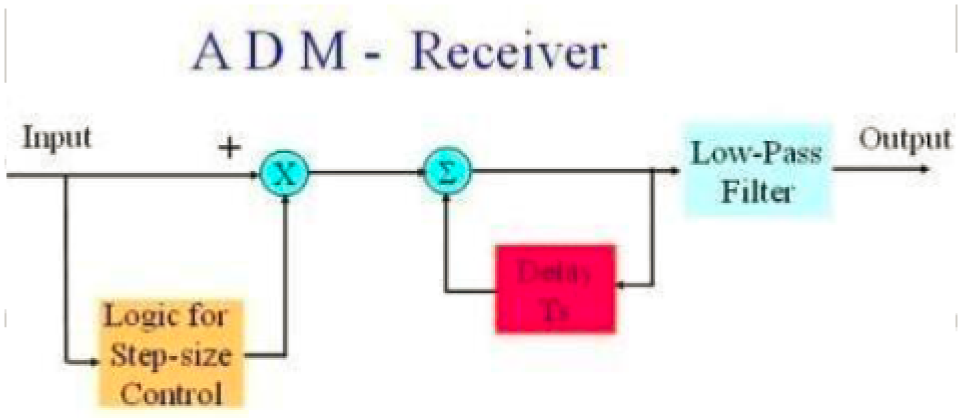
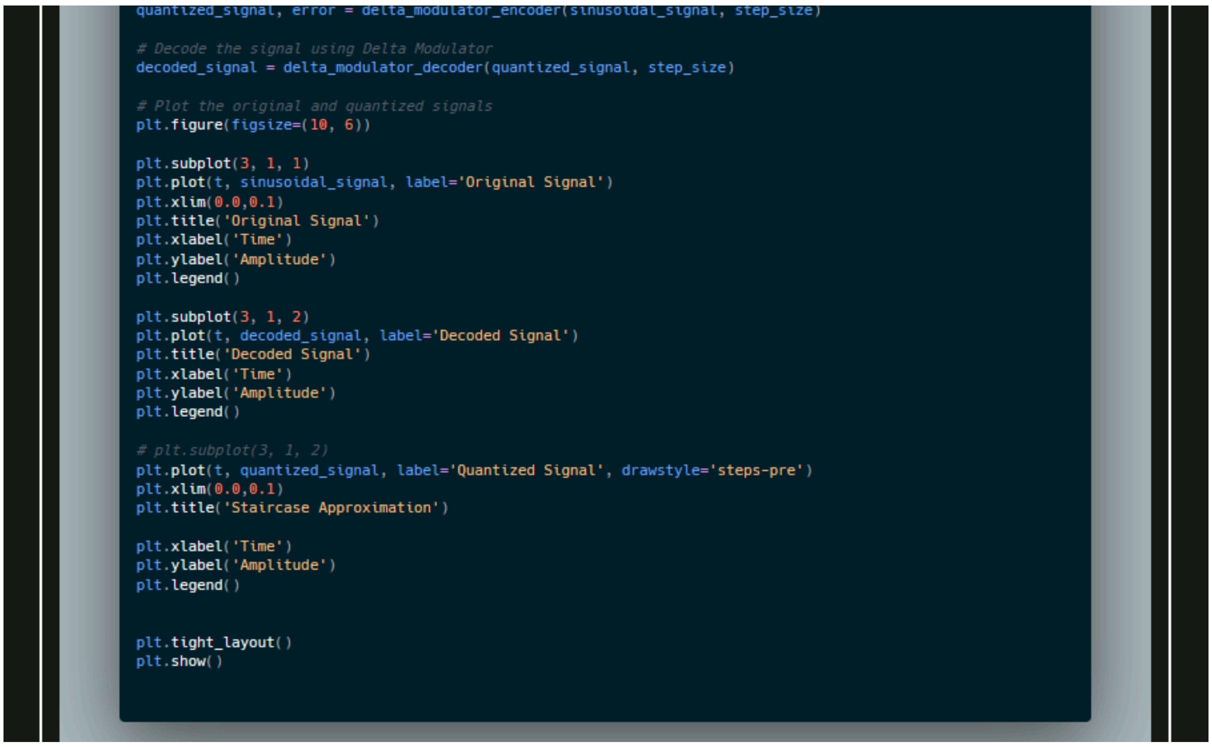
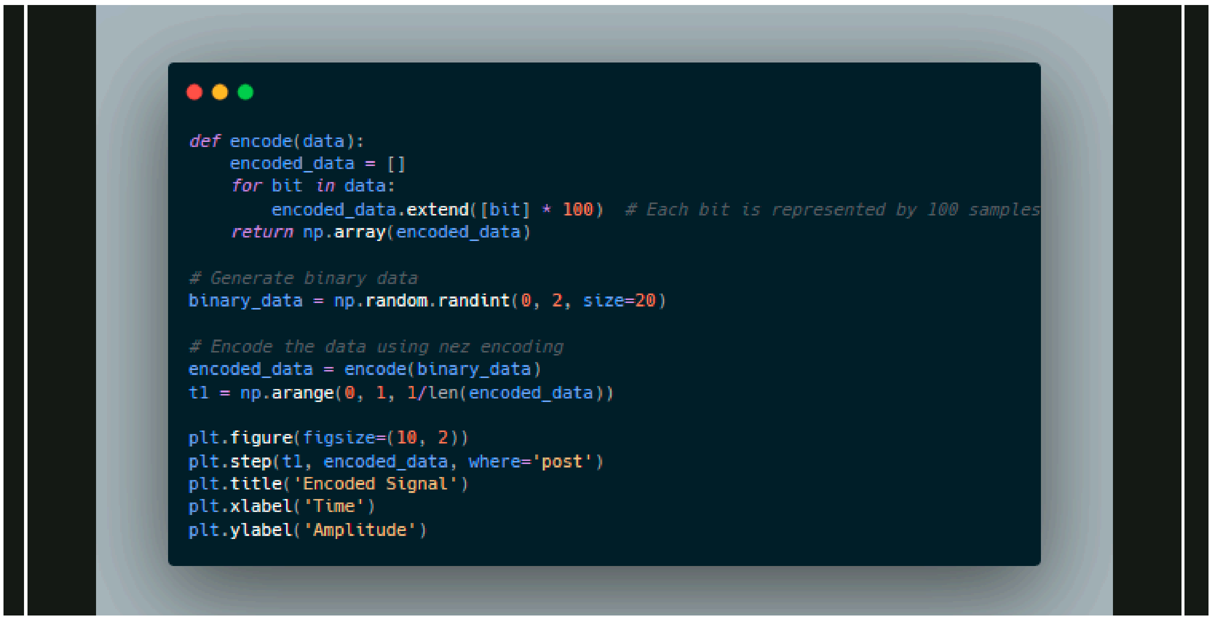


Fig 3.3: Block Diagram of ADM Receiver
(Source: ElProCus)

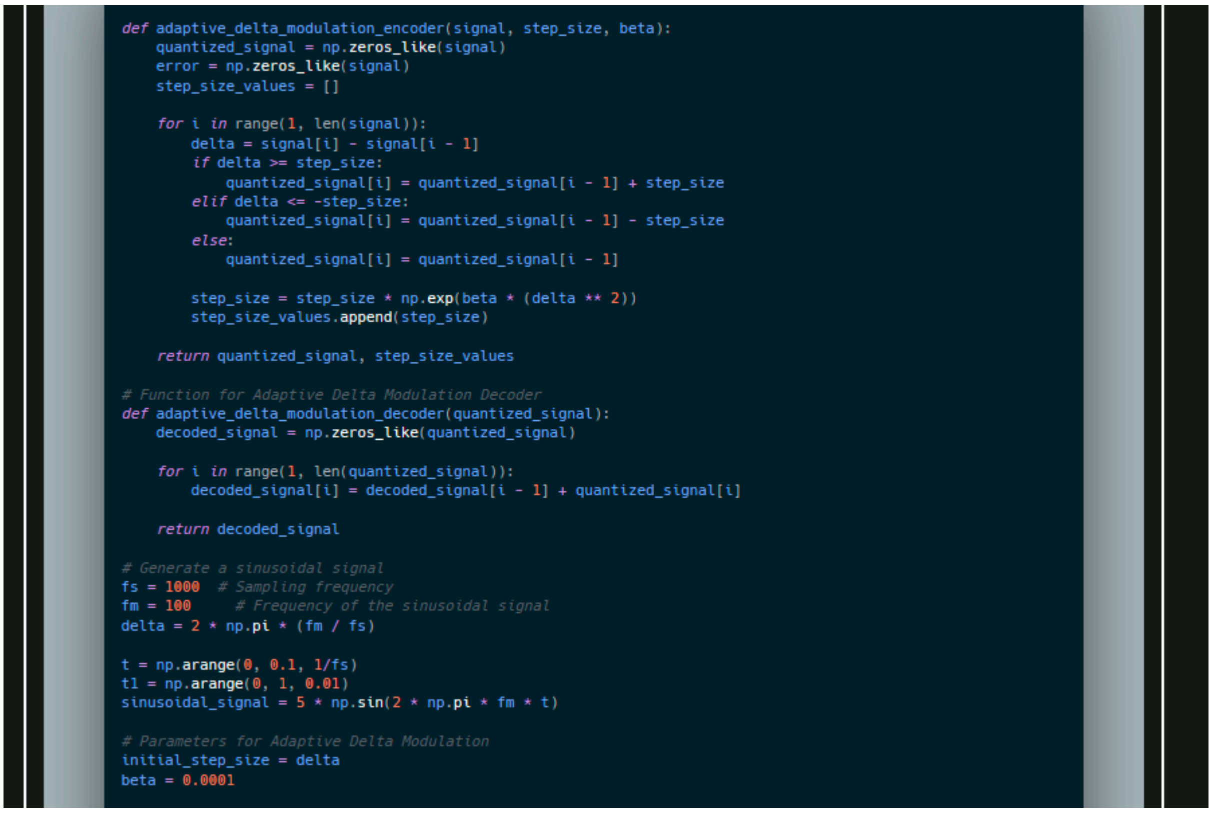
Code :-



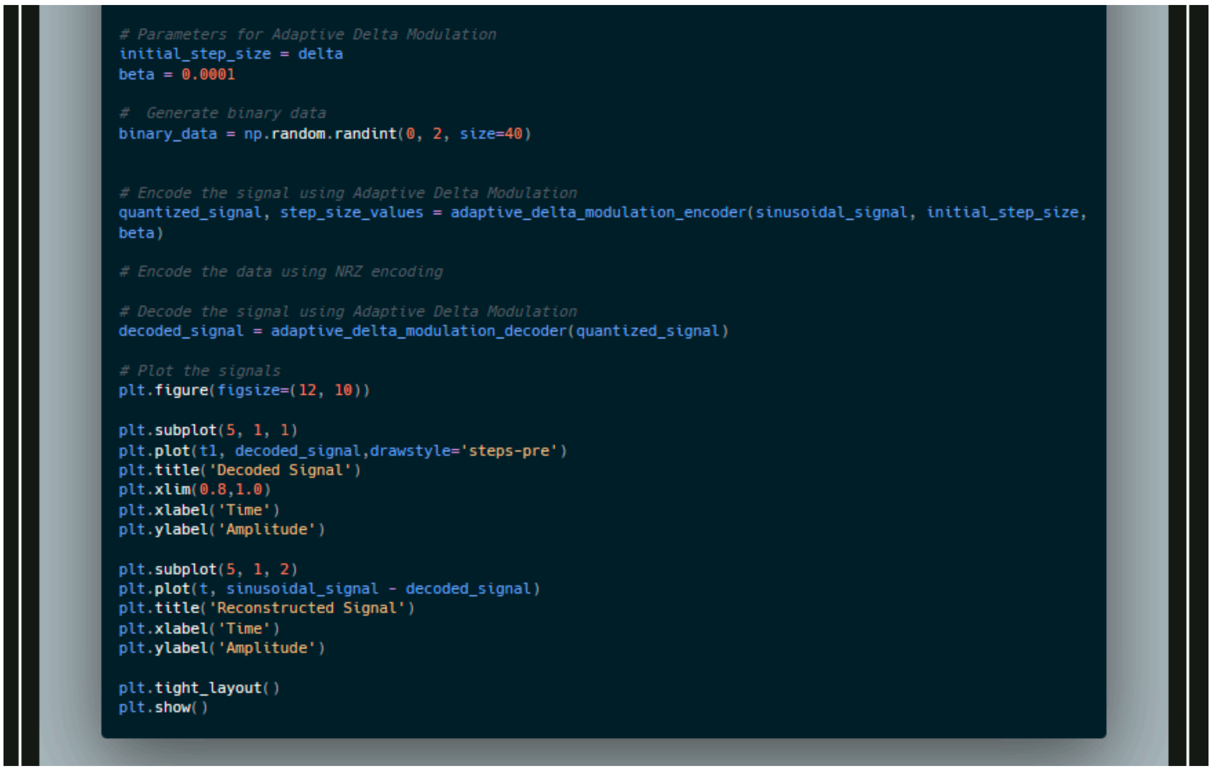
```
quantized_signal, error = delta_modulator_encoder(sinusoidal_signal, step_size)

# Decode the signal using Delta Modulator
decoded_signal = delta_modulator_decoder(quantized_signal, step_size)

# Plot the original and quantized signals
plt.figure(figsize=(10, 6))

plt.subplot(3, 1, 1)
plt.plot(t, sinusoidal_signal, label='Original Signal')
plt.xlim(0.0,0.1)
plt.title('Original Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.legend()

plt.subplot(3, 1, 2)
plt.plot(t, decoded_signal, label='Decoded Signal')
plt.title('Decoded Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.legend()

# plt.subplot(3, 1, 2)
plt.plot(t, quantized_signal, label='Quantized Signal', drawstyle='steps-pre')
plt.xlim(0.0,0.1)
plt.title('Staircase Approximation')

plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.legend()


plt.tight_layout()
plt.show()
```



```
def encode(data):
    encoded_data = []
    for bit in data:
        encoded_data.extend([bit] * 100)  # Each bit is represented by 100 samples
    return np.array(encoded_data)

# Generate binary data
binary_data = np.random.randint(0, 2, size=20)

# Encode the data using nez encoding
encoded_data = encode(binary_data)
t1 = np.arange(0, 1, 1/len(encoded_data))

plt.figure(figsize=(10, 2))
plt.step(t1, encoded_data, where='post')
plt.title('Encoded Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')
```



```python
def adaptive_delta_modulation_encoder(signal, step_size, beta):
    quantized_signal = np.zeros_like(signal)
    error = np.zeros_like(signal)
    step_size_values = []

    for i in range(1, len(signal)):
        delta = signal[i] - signal[i - 1]
        if delta >= step_size:
            quantized_signal[i] = quantized_signal[i - 1] + step_size
        elif delta <= -step_size:
            quantized_signal[i] = quantized_signal[i - 1] - step_size
        else:
            quantized_signal[i] = quantized_signal[i - 1]

        step_size = step_size * np.exp(beta * (delta ** 2))
        step_size_values.append(step_size)

    return quantized_signal, step_size_values

# Function for Adaptive Delta Modulation Decoder
def adaptive_delta_modulation_decoder(quantized_signal):
    decoded_signal = np.zeros_like(quantized_signal)

    for i in range(1, len(quantized_signal)):
        decoded_signal[i] = decoded_signal[i - 1] + quantized_signal[i]

    return decoded_signal

# Generate a sinusoidal signal
fs = 1000   # Sampling frequency
fm = 100     # Frequency of the sinusoidal signal
delta = 2 * np.pi * (fm / fs)

t = np.arange(0, 0.1, 1/fs)
t1 = np.arange(0, 1, 0.01)
sinusoidal_signal = 5 * np.sin(2 * np.pi * fm * t)

# Parameters for Adaptive Delta Modulation
initial_step_size = delta
beta = 0.0001
```



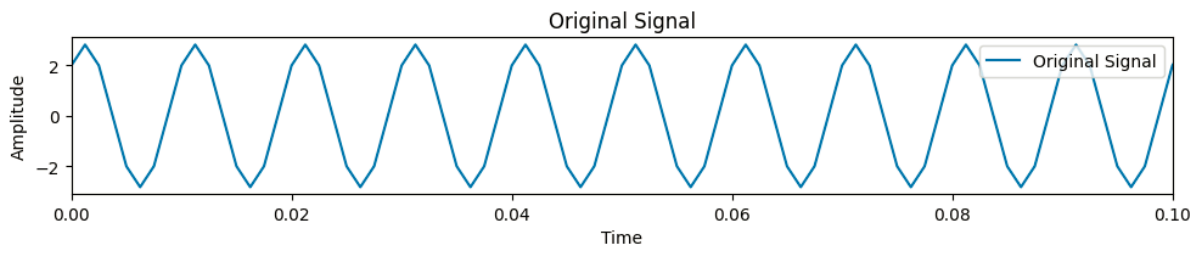
```python
# Parameters for Adaptive Delta Modulation
initial_step_size = delta
beta = 0.0001

#  Generate binary data
binary_data = np.random.randint(0, 2, size=40)


# Encode the signal using Adaptive Delta Modulation
quantized_signal, step_size_values = adaptive_delta_modulation_encoder(sinusoidal_signal, initial_step_size,
beta)

# Encode the data using NRZ encoding

# Decode the signal using Adaptive Delta Modulation
decoded_signal = adaptive_delta_modulation_decoder(quantized_signal)

# Plot the signals
plt.figure(figsize=(12, 10))

plt.subplot(5, 1, 1)
plt.plot(t1, decoded_signal,drawstyle='steps-pre')
plt.title('Decoded Signal')
plt.xlim(0.8,1.0)
plt.xlabel('Time')
plt.ylabel('Amplitude')

plt.subplot(5, 1, 2)
plt.plot(t, sinusoidal_signal - decoded_signal)
plt.title('Reconstructed Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')

plt.tight_layout()
plt.show()
```

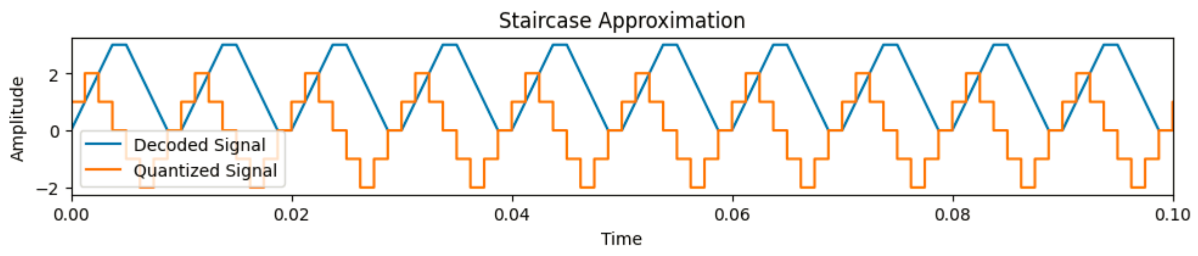
Procedure :-

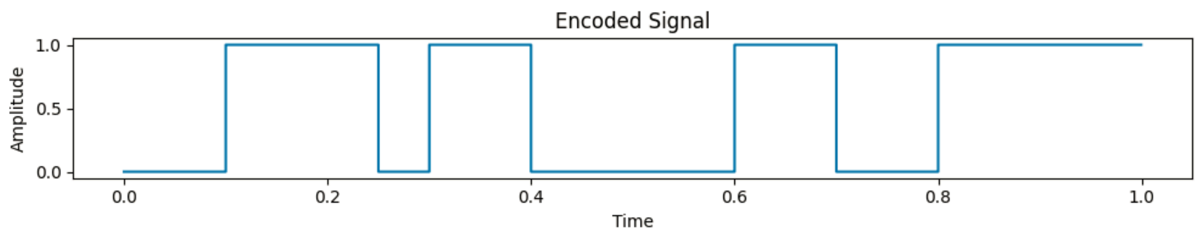
1. Generate a sinusoidal signal of amplitude 5 and frequency 100 Hz. Choose appropriate sampling frequency (usually 4- or 5-times Nyquist rate) and plot for two cycles.
2. Generate the staircase approximation to the sinusoidal signal.
3. Plot the original signal and staircase approximation on the same graph.
4. Change the step size and observe slope-overloading issue in Delta Modulation.
5. Generate the DM wave -transmitted binary stream of ±Δ. Encode +Δ to 1 and -Δ to zero.Receiver
6. Decode the received binary stream and pass through an accumulator.
7. Pass through a low pass filter to reconstruct the signal.

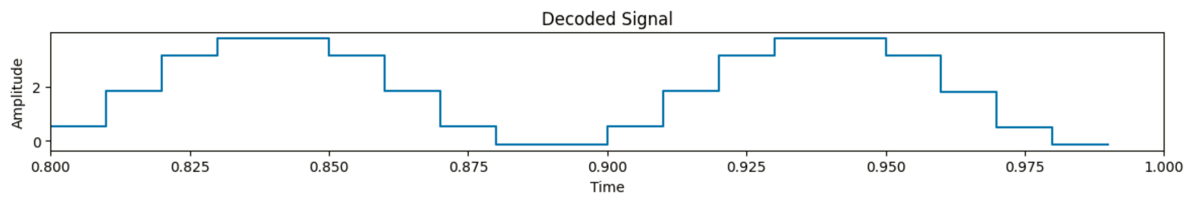
Graphs :-

(1) Case 1 : $\Delta = 2 * \pi * fm/fs$



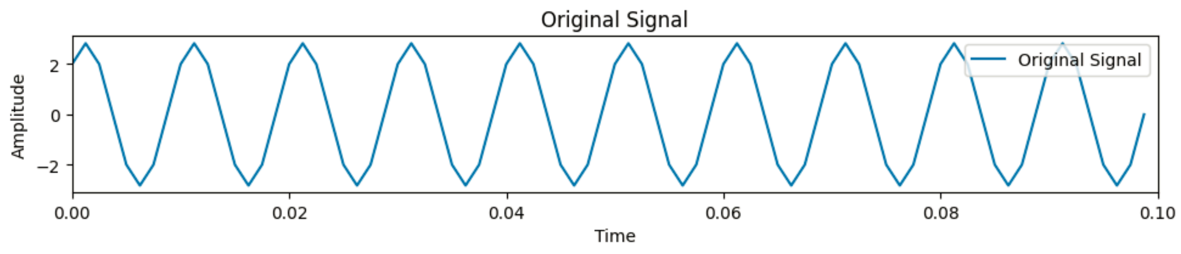


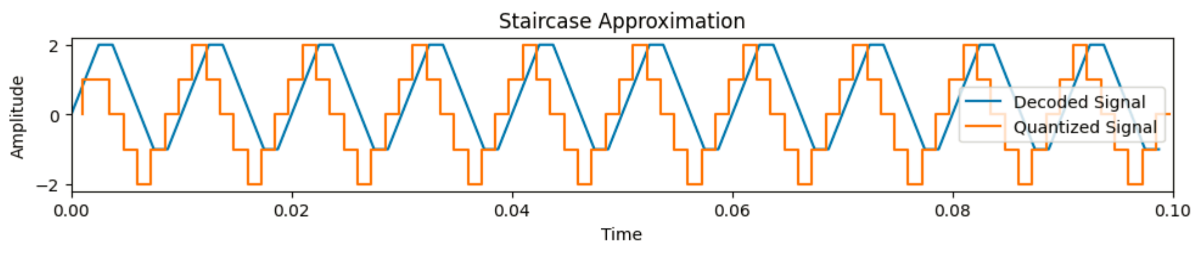


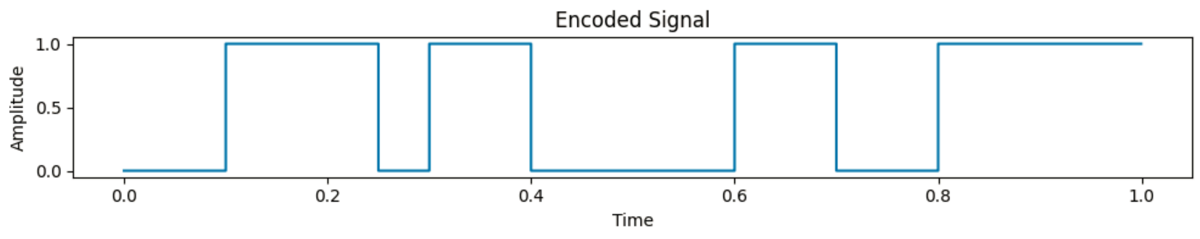


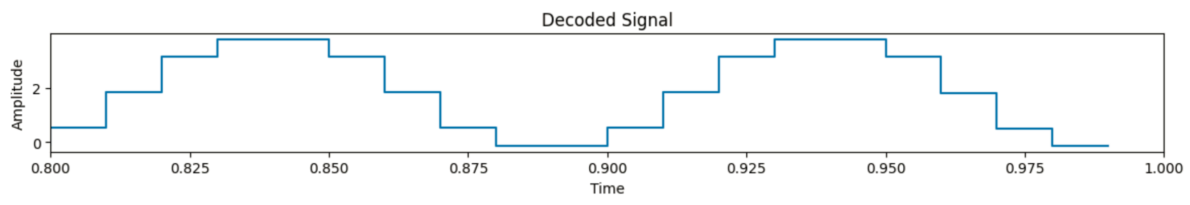
Reconstructed Signal

(2) <u>Case 2</u> : $\Delta < 2 * \pi * fm/fs$









Reconstructed Signal

Observations :-

In this simulation experiment the output graphs that were obtained, the following observations can be made

- Delta Modulation:
  - **Advantages:**
    - Simple implementation.
    - Low complexity.
  - **Disadvantages:**
    - High quantization noise, especially for rapidly changing signals.
- Adaptive Delta Modulation:
  - **Advantages:**
    - Better performance for signals with varying dynamics.
  - **Disadvantages:**
    - Increased complexity due to adaptation mechanism.

Results :-

Delta modulation provides a simple approach for analog-to-digital conversion, while adaptive delta modulation adapts to varying signal dynamics. The choice between these techniques depends on the application requirements regarding signal fidelity and complexity.

.

Conclusion :-

In conclusion , this experiment was done to learn how to generate a signal, obtain its plots, sampled version, fourier transform, reconstructed signal, and finally, fourier transform of the reconstructed signal, a Python program was successfully written and implemented in Anaconda's Spyder Software Simulator.