Experiment No. 01

1.1 Experiment Name

Study of sampling error vs frequency graph and quantization error vs ADC graph using Python Code

1.2 Objectives

- To get a better understanding of sampling error and quantization error procedure
- To get familiar with the procedure of sampling any signal using python

1.3 Apparatus

- Jupyter Notebook

1.4 Theory

1.4.1 Sampling & sampling error

Sampling is the digitization of the coordinate value. Sampling is done prior to the quantization process.

In digital signal processing, sampling error refers to the discrepancy between the original analog signal and the digital signal obtained by sampling the analog signal at a finite rate. Aliasing and quantization noise are two examples of the different distortions that can be caused by sampling inaccuracy.

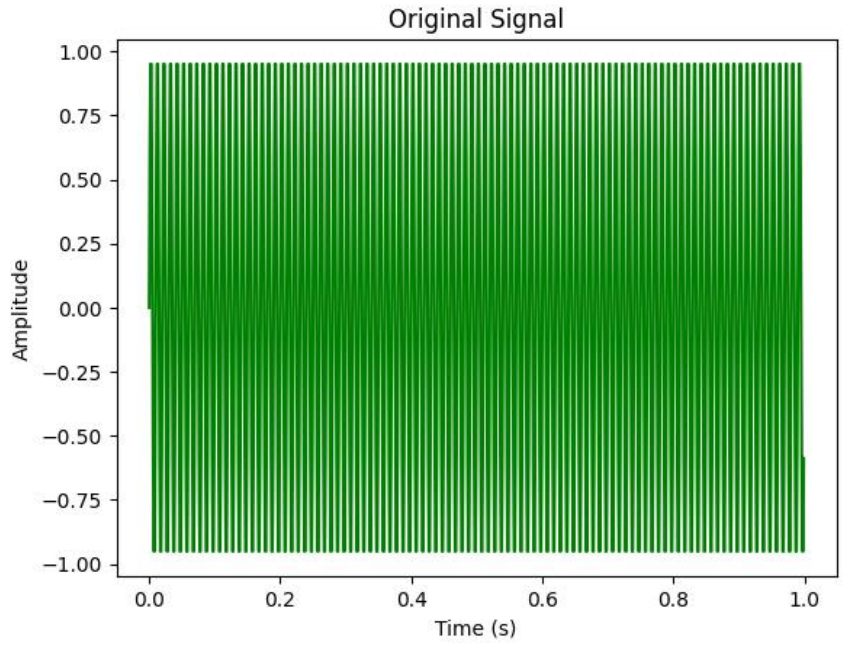
1.4.2 Quantization & quantization error

Quantization is the digitization of the amplitude value. Quantization is done after the sampling process.

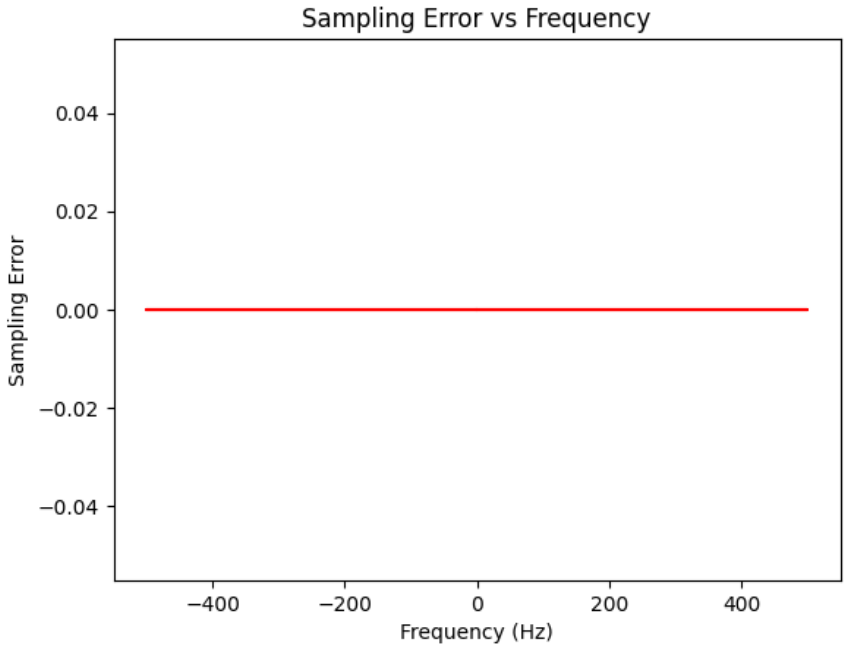
Quantization error in digital signal processing is the difference between the continuous signal's analog and digital components after being rounded to the nearest quantization level. The quantization error is proportional to the quantization step size, which is the distance between adjacent quantization levels.

1.5 Python code & graph

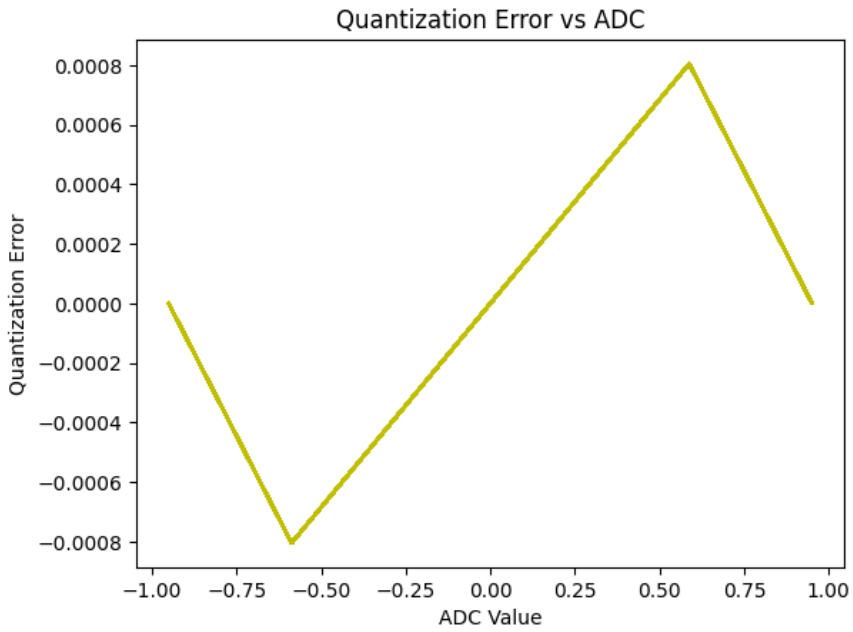
```
import numpy as np
import matplotlib.pyplot as plt

# Set up parameters
Fs = 1000  # Sampling frequency (Hz)
Ts = 1/Fs  # Sampling period (s)
T = 1      # Signal duration (s)
N = int(T/Ts)  # Number of samples

# Generate a sinusoidal signal
f = 100  # Signal frequency (Hz)
t = np.arange(N)*Ts  # Time vector
x = np.sin(2*np.pi*f*t)

# Plot original signal
fig, ax = plt.subplots()
plt.plot(t, x,'g')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Original Signal')
```




```
# Sample the signal
n = np.arange(N)
xn = x[n]

# Calculate sampling error vs frequency
f_sample = np.fft.fftfreq(N, Ts)
X = np.fft.fft(x)/N  # Fourier transform of original signal
Xn = np.fft.fft(xn)/N  # Fourier transform of sampled signal
sampling_error = np.abs(X - Xn)

# Plot sampling error vs frequency
fig, ax = plt.subplots()
plt.plot(f_sample, sampling_error,'r')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Sampling Error')
plt.title('Sampling Error vs Frequency')
```



```
# Quantize the signal
n_bits = 8  # Number of bits for ADC
Vmax = np.max(np.abs(xn))  # Maximum signal amplitude
q_step = Vmax/(2**(n_bits-1))  # Quantization step size
xnq = np.round(xn/q_step)*q_step  # Quantized signal

# Calculate quantization error vs ADC
quantization_error = xn - xnq

# Plot quantization error vs ADC
fig, ax = plt.subplots()
plt.plot(xn, quantization_error,'y')
plt.xlabel('ADC Value')
plt.ylabel('Quantization Error')
plt.title('Quantization Error vs ADC')

plt.show()
```



1.6 Discussion & Conclusion

In this experiment, we used the python to analyze sampling error vs frequency and quantization error vs ADC. The signal is then sampled by selecting every Nth sample, where N is the number of samples. The sampling error vs. frequency is then calculated by taking the absolute difference between the Fourier transform of the original signal and the Fourier transform of the sampled signal, using NumPy functions. The signal is then quantized by converting the signal values to a limited number of discrete levels using an analog-to-digital converter (ADC). The number of bits used for the ADC is set to 8, and the maximum signal amplitude is calculated.