MULTIMEDIA UNIVERSITY OF KENYA

Riding on Technology, Inspiring Innovation

ANALYSIS OF SAMPLING THEOREM USING PYTHON JUPYTER NOTEBOOK

Course Code and Name

CHARLES GITHINJI

ENG-219-017/2021

ECE 2414: DIGITAL COMMUNICATION 1

Date of Submission

7TH NOV 2024

LECTURER

MARTIN WAFULA

Contents

1 Abstract

This lab report explores the Sampling Theorem, which asserts that a continuous-time signal can be perfectly reconstructed from its samples if the sampling frequency is at least twice the maximum frequency of the signal (the Nyquist rate). The lab includes practical analysis of the theorem through the sampling and reconstruction of signals in both the time and frequency domains. Using Python and Jupyter Notebook, signal sampling, frequency spectrum analysis, and signal reconstruction were carried out to observe the effects of different sampling rates

2 Introduction

The Sampling Theorem is a fundamental principle in digital signal processing that provides guidelines for converting a continuous-time signal into discrete samples while maintaining the ability to reconstruct the original signal without loss of information. According to this theorem, the sampling rate must be at least twice the highest frequency component present in the signal. This minimum rate, known as the Nyquist rate, prevents aliasing—a phenomenon where higher frequency components appear as lower frequencies in the sampled data. This lab investigates the theorem using sinusoidal signals with 1 Hz and 3 Hz components.

3 Objective

The objectives of this lab are:

- To demonstrate the Sampling Theorem and understand its implications.
- To sample a continuous-time signal at various rates and observe aliasing when the sampling rate is below the Nyquist rate.

- To reconstruct the original signal from sampled data using a Low Pass Filter (LPF).

4 Theory

The Sampling Theorem states that a signal can be completely reconstructed from its samples if the sampling frequency is at least twice the maximum frequency in the signal. If the sampling frequency is lower than this rate, aliasing occurs, causing distortions in the signal. Aliasing appears as overlapping in the frequency spectrum and makes it impossible to distinguish between different frequency components.

For example, a sinusoidal signal with frequencies of 1 Hz and 3 Hz can be sampled at a minimum of 6 Hz (Nyquist rate) to prevent aliasing. When the sampling rate falls below this threshold, higher frequencies appear distorted in the sampled signal.

Figure 1: Test results for circuit 1

5 Procedure

Experiment 1: Analysis of Sampling Theorem

1.Define the Message Signal:

- Created a message signal composed of 1 Hz and 3 Hz sinusoidal components using Python code in Jupyter Notebook.

2. **Plot the Message Signal**:

- The continuous-time representation of the signal was plotted to visualize its behavior in the time domain.

3. **Compute and Plot the Spectrum:**

- Used the Fast Fourier Transform (FFT) to observe the frequency spectrum of the message signal.

4. **Sample the Message Signal:**

Sampled the signal at a rate of 50 Hz (sampling period of 0.02 seconds) and then at lower rates to observe aliasing effects.

5. Plot the Sampled Signal:Plotted the sampled signal in the time domain using stem plotting to represent discrete samples.
6. Compute and Plot the Spectrum of the Sampled Signal:Applied FFT to the sampled signal to analyze the frequency components and observe the aliasing effect.

Experiment 2: Reconstruction from Sampled Signal

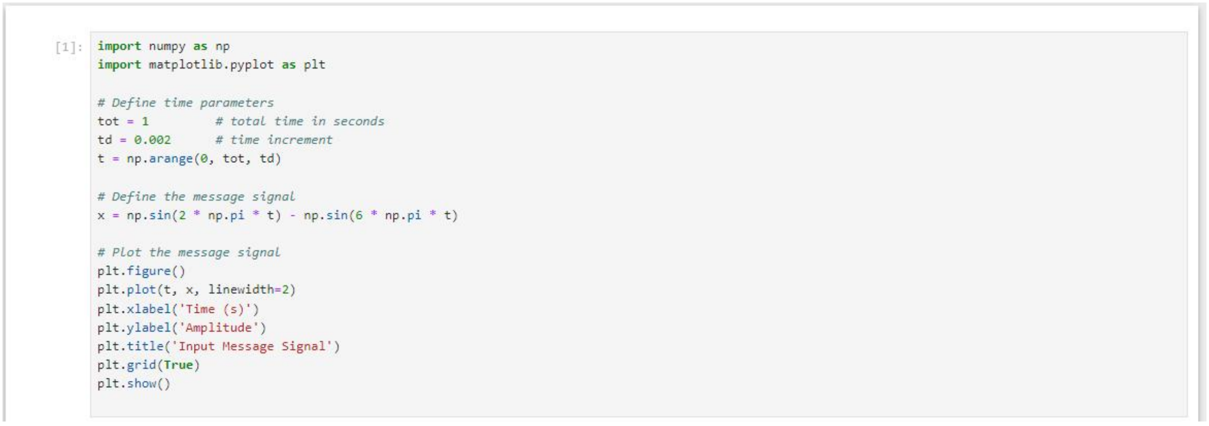
1. Upsample and Zero-fill the Sampled Signal:
   Increased the sample rate by inserting zeros between samples, preparing the signal for reconstruction.
2. Frequency Spectrum of the Sampled Signal: Calculated the FFT of the upsampled signal to observe its spectrum.
3. Design and Apply a Low Pass Filter (LPF): Designed an LPF to retain frequencies from -10 Hz to 10 Hz and filtered the upsampled signal.
4. Inverse FFT for Time Domain Representation: Used inverse FFT to convert the filtered signal back to the time domain, comparing it with the original signal.

6 RESULTS

Message Signal and Spectrum

❖ **Figure 1**: Original Message Signal in the time domain

Description: This plot shows the continuous-time message signal with 1 HZ and 3 HZ component



```
[1]: import numpy as np
     import matplotlib.pyplot as plt

     # Define time parameters
     tot = 1         # total time in seconds
     td = 0.002      # time increment
     t = np.arange(0, tot, td)

     # Define the message signal
     x = np.sin(2 * np.pi * t) - np.sin(6 * np.pi * t)

     # Plot the message signal
     plt.figure()
     plt.plot(t, x, linewidth=2)
     plt.xlabel('Time (s)')
     plt.ylabel('Amplitude')
     plt.title('Input Message Signal')
     plt.grid(True)
     plt.show()
```

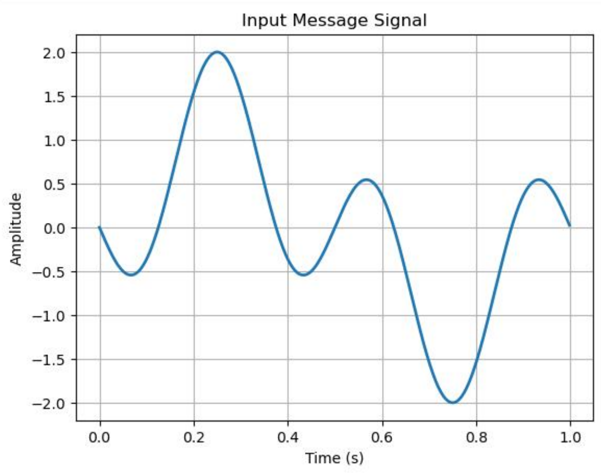
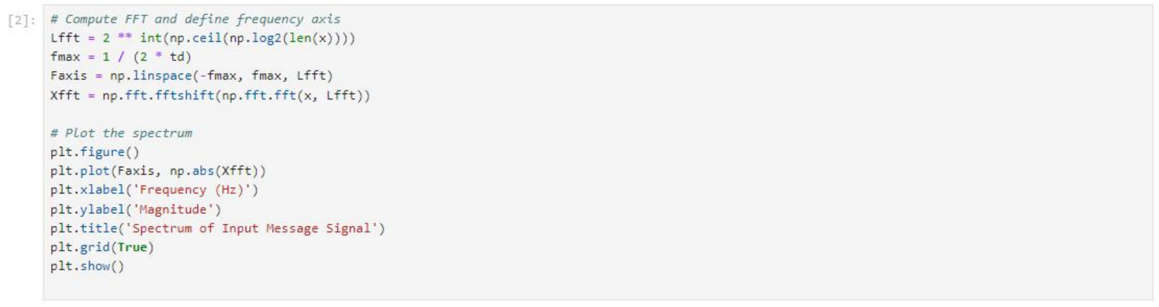
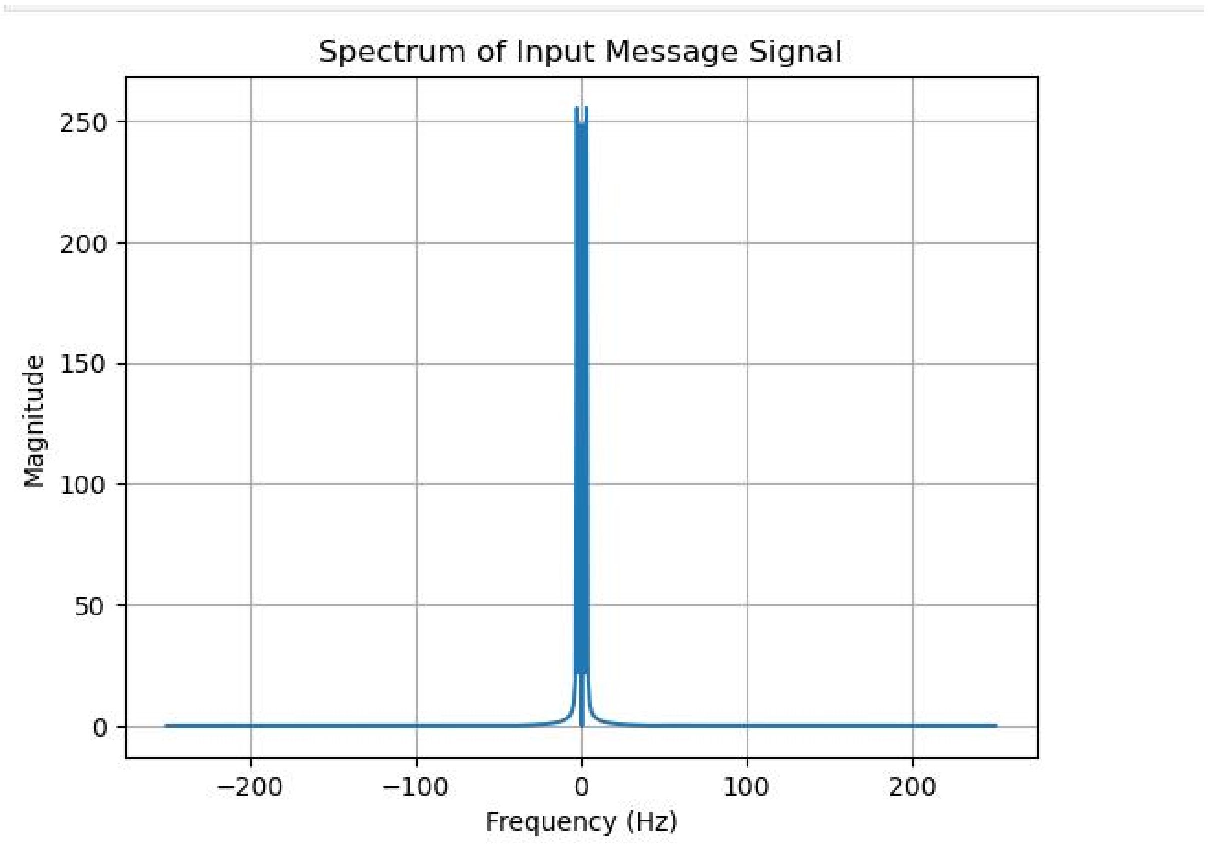


Figure 2: Spectrum of the Original Message Signal

Description: FFT analysis of the message signal reveals distinct peaks at 1 Hz and 3 Hz, confirming the frequency components of the signal.



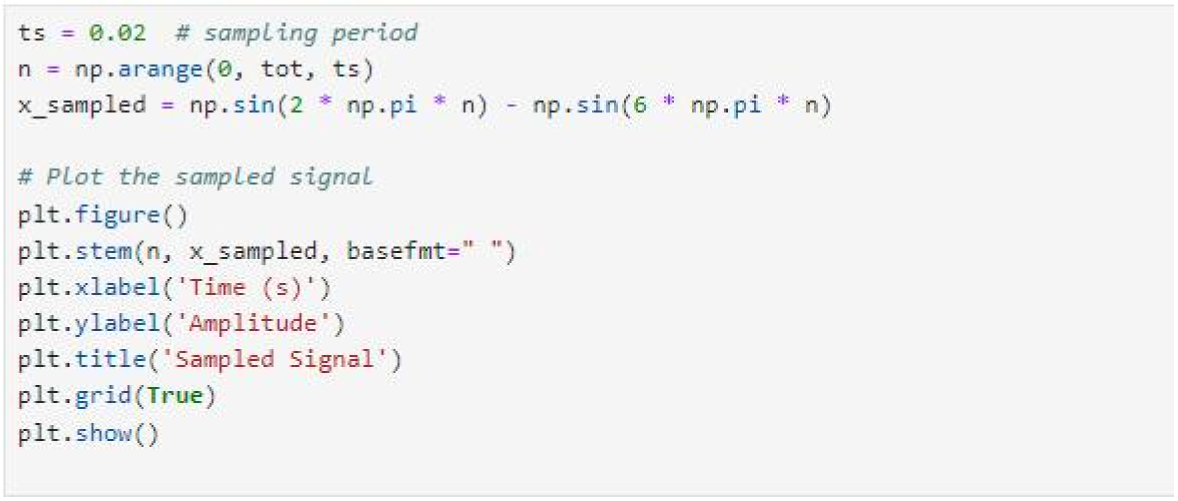
```
[2]: # Compute FFT and define frequency axis
     Lfft = 2 ** int(np.ceil(np.log2(len(x))))
     fmax = 1 / (2 * td)
     Faxis = np.linspace(-fmax, fmax, Lfft)
     Xfft = np.fft.fftshift(np.fft.fft(x, Lfft))

     # Plot the spectrum
     plt.figure()
     plt.plot(Faxis, np.abs(Xfft))
     plt.xlabel('Frequency (Hz)')
     plt.ylabel('Magnitude')
     plt.title('Spectrum of Input Message Signal')
     plt.grid(True)
     plt.show()
```



Sampling Results

Figure 3: Sampled Signal in the Time Domain (50 Hz Sampling Rate)

Description: The sampled signal at a 50 Hz rate accurately represents the original signal without aliasing, due to the high sampling frequency.



```
ts = 0.02  # sampling period
n = np.arange(0, tot, ts)
x_sampled = np.sin(2 * np.pi * n) - np.sin(6 * np.pi * n)

# Plot the sampled signal
plt.figure()
plt.stem(n, x_sampled, basefmt=" ")
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Sampled Signal')
plt.grid(True)
plt.show()
```

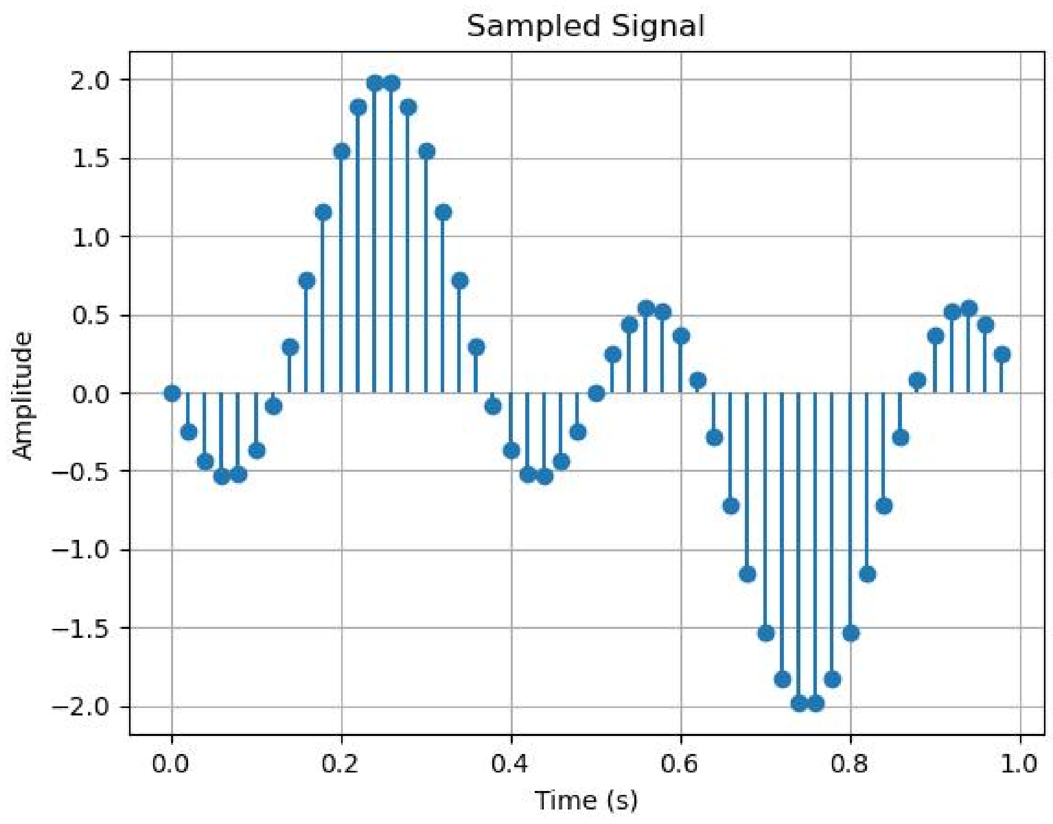
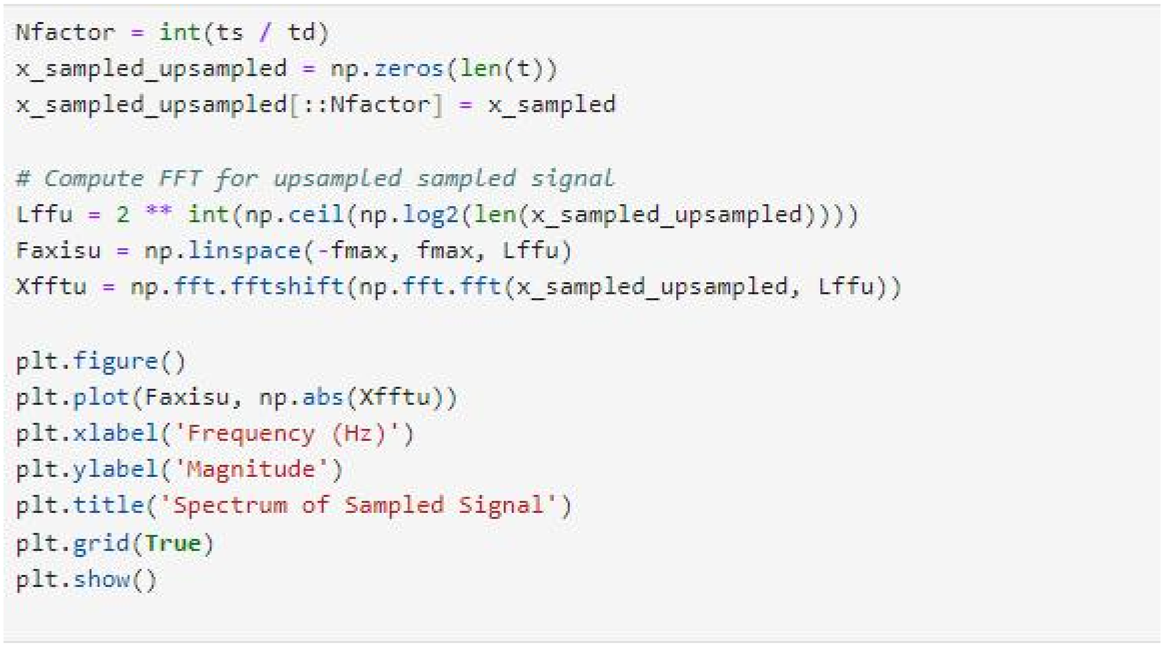


Figure 4: Spectrum of the Sampled Signal (50 Hz Sampling Rate)

Description: The frequency spectrum of the sampled signal closely matches the original spectrum, indicating no loss of information.



```python
Nfactor = int(ts / td)
x_sampled_upsampled = np.zeros(len(t))
x_sampled_upsampled[::Nfactor] = x_sampled

# Compute FFT for upsampled sampled signal
Lffu = 2 ** int(np.ceil(np.log2(len(x_sampled_upsampled))))
Faxisu = np.linspace(-fmax, fmax, Lffu)
Xfftu = np.fft.fftshift(np.fft.fft(x_sampled_upsampled, Lffu))

plt.figure()
plt.plot(Faxisu, np.abs(Xfftu))
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')
plt.title('Spectrum of Sampled Signal')
plt.grid(True)
plt.show()
```

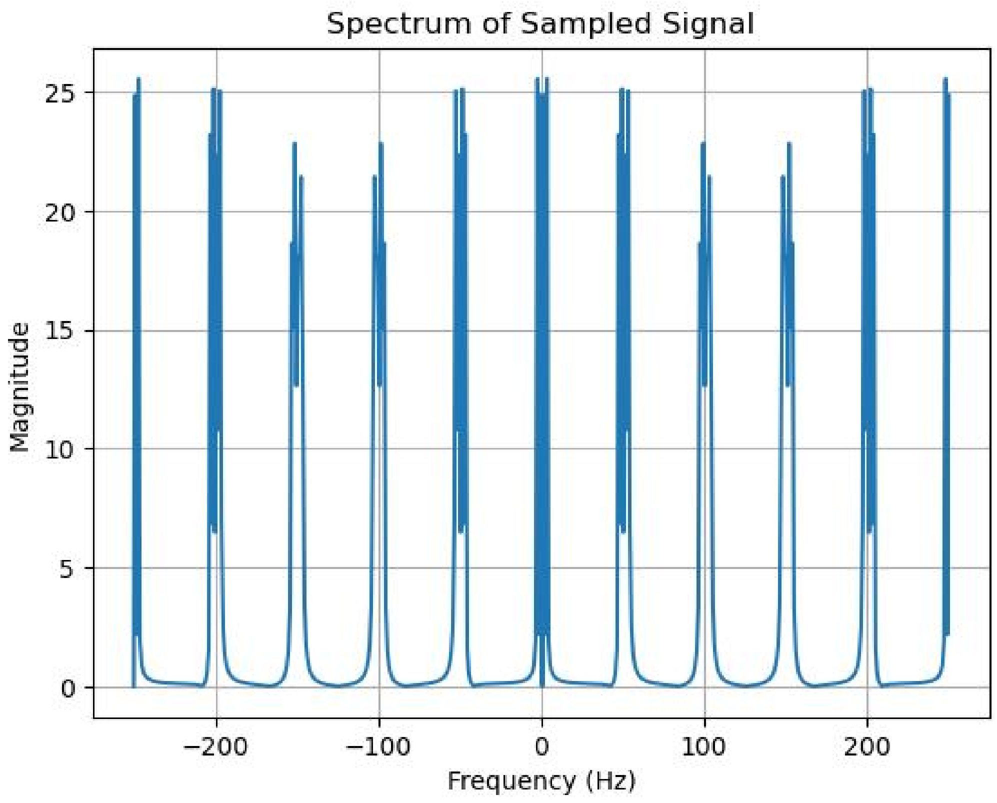
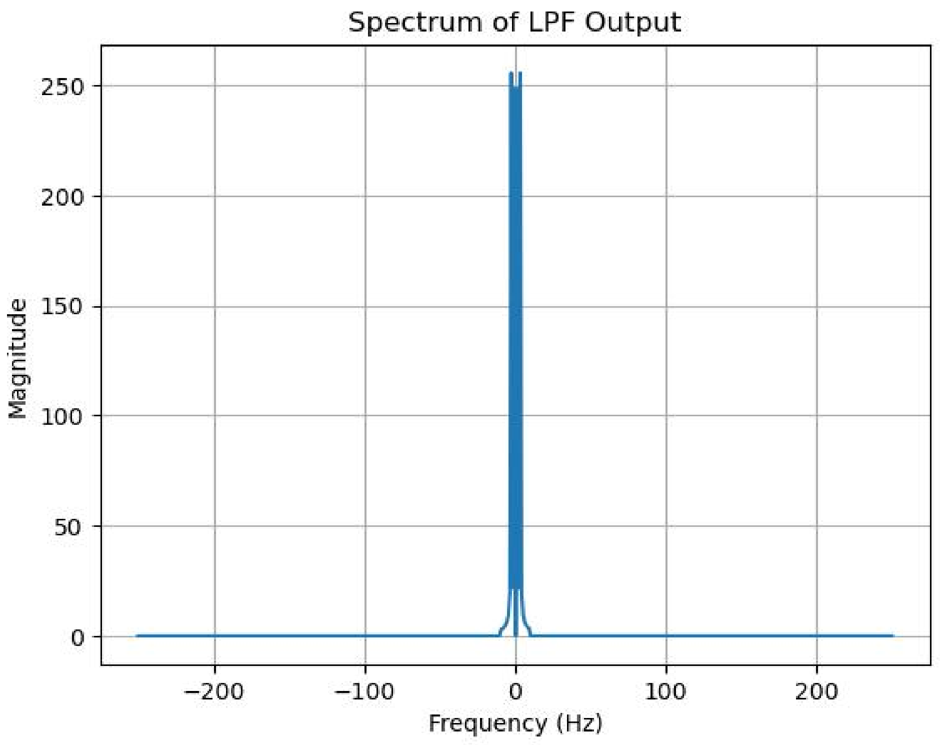


Figure 5: Sampled Signal in the Time Domain (Below Nyquist Rate)

Description: When sampled below the Nyquist rate, the signal exhibits aliasing, causing high frequencies to fold into lower frequencies and creating a distorted signal.

```
BW = 10
H_lpf = np.zeros(Lffu)
H_lpf[Lffu // 2 - BW:Lffu // 2 + BW] = 1

# Apply the LPF to the FFT of the upsampled signal
x_recv = Nfactor * Xfftu * H_lpf

# Plot the spectrum after filtering
plt.figure()
plt.plot(Faxisu, np.abs(x_recv))
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')
plt.title('Spectrum of LPF Output')
plt.grid(True)
plt.show()
```



Reconstruction Results

Figure 6: Spectrum after Low Pass Filtering

Description: The low-pass filtered spectrum retains only the necessary frequencies from -10 Hz to 10 Hz, successfully eliminating unwanted aliasing effects.

```
x_recv_time = np.real(np.fft.ifft(np.fft.fftshift(x_recv)))

plt.figure()
plt.plot(t, x, 'r', label='Original Signal')
plt.plot(t, x_recv_time[:len(t)], 'b--', label='Reconstructed Signal')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Original vs. Reconstructed Signal')
plt.legend()
plt.grid(True)
plt.show()
```

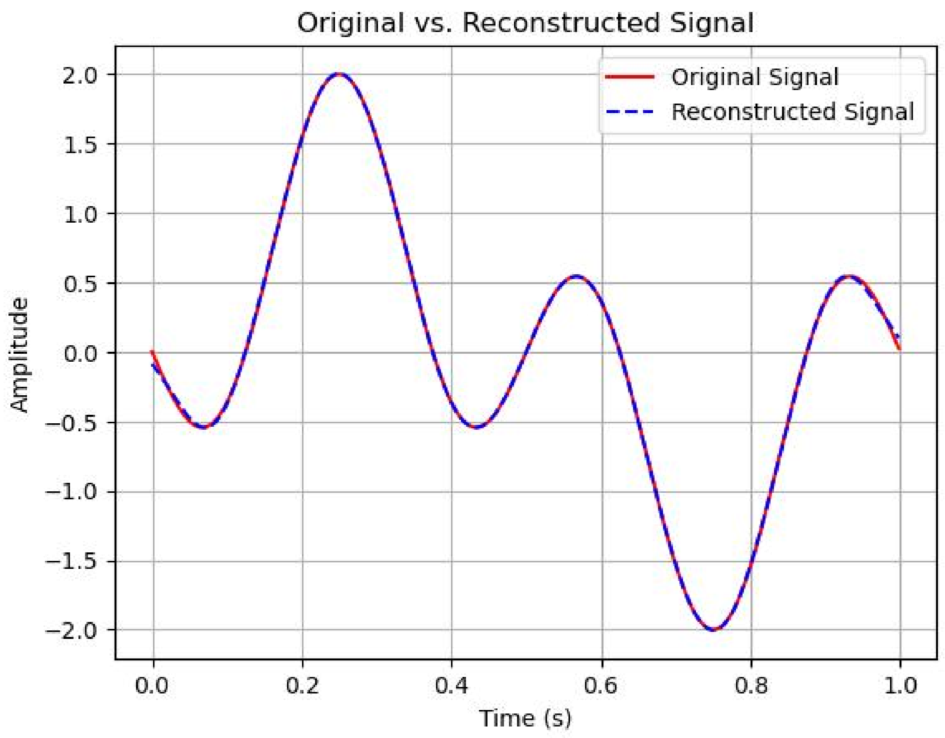
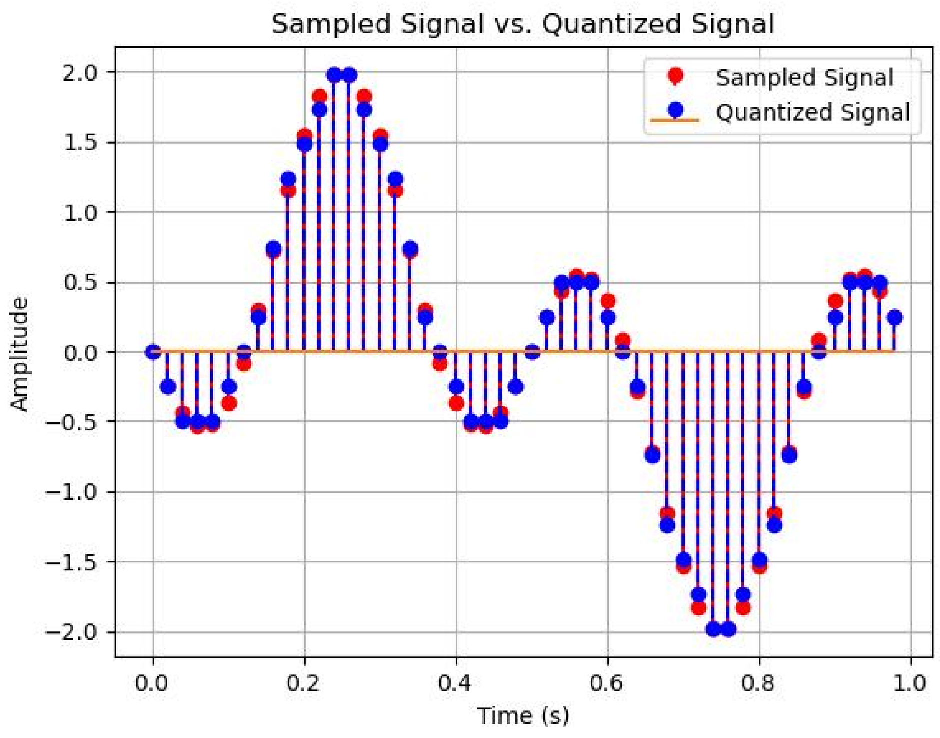


Figure 7: Original vs. Reconstructed Signal

Description: A side-by-side comparison between the original and reconstructed signals, showing a high degree of accuracy in reconstruction, validating the effectiveness of the low-pass filter.

```
levels = 16
x_min, x_max = np.min(x_sampled), np.max(x_sampled)
step = (x_max - x_min) / levels

# Quantize the sampled signal
x_quantized = step * np.round((x_sampled - x_min) / step) + x_min

# Plot quantized vs. sampled signal
plt.figure()
plt.stem(n, x_sampled, 'r', markerfmt='ro', basefmt=" ",  label='Sampled Signal')
plt.stem(n, x_quantized, 'b--', markerfmt='bo', basefmt="", label='Quantized Signal')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Sampled Signal vs. Quantized Signal')
plt.legend()
plt.grid(True)
plt.show()
```



Discussion

The results from this lab underscore the critical role that the Sampling Theorem plays in digital signal processing. Sampling is essential in converting analog signals into digital data that can be processed and transmitted in modern communication systems. Through this lab, we explored the direct impact of sampling rates on the integrity of a signal's representation and demonstrated the consequences of violating the Nyquist rate.

Sampling Above and Below the Nyquist Rate

When sampling was performed at a rate above the Nyquist threshold (in this case, at 50 Hz for a signal with a maximum frequency of 3 Hz), the signal retained its original structure in both time and frequency domains. This confirms the theorem's assertion that a sampling rate at least twice the maximum signal frequency enables accurate representation and reconstruction without distortion.

However, when the sampling rate was reduced below the Nyquist rate, aliasing effects became evident. Aliasing caused higher frequency components to overlap into lower frequencies, resulting in a misrepresentation of the signal. In the frequency spectrum, aliasing appeared as additional, unintended peaks, which made it impossible to distinguish the original frequencies accurately. This visualizes a common issue in digital communication systems where inadequate sampling leads to data loss and misinterpretation, as critical information can be masked or distorted.

Reconstruction Using Low-Pass Filtering

The reconstruction of the signal through low-pass filtering (LPF) demonstrated an effective method to recover the original signal from sampled data. By using an LPF with a bandwidth matching the original signal's range (e.g., -10 Hz to 10 Hz), frequencies outside the Nyquist range were suppressed. This process allowed only the intended frequency components to pass through, enabling accurate reconstruction in the time domain.

The choice of filter bandwidth plays a crucial role in signal fidelity during reconstruction. A narrower bandwidth would risk cutting out essential parts of the signal, whereas a wider bandwidth could allow unwanted noise or aliasing artifacts to remain. By carefully selecting the LPF characteristics, we achieved a reconstructed signal closely matching the original, highlighting the importance of appropriate filter design in communication systems.

Real-World Implications

In real-world digital communication systems, adherence to the Sampling Theorem is essential to ensure that signals are captured accurately and remain useful after transmission and reconstruction. Applications such as audio processing, image processing, and telecommunications rely on this theorem to prevent aliasing and ensure data integrity. For instance, digital audio signals are typically sampled at 44.1 kHz to avoid aliasing in the audible frequency range (up to 20 kHz), while MRI systems in medical imaging use high sampling rates to capture complex biological signals accurately. This lab reinforces the need for careful consideration of sampling rates and appropriate filtering to maintain signal quality across various applications.

---

7. Conclusion

This lab successfully illustrated the principles of the Sampling Theorem and the practical considerations necessary for effective signal processing in digital communications. Key findings include:

1. **Confirmation of the Sampling Theorem**: The experiments validated that sampling a signal at or above twice its maximum frequency enables accurate representation and reconstruction, aligning with the theoretical predictions.
2. **Impact of Aliasing**: When sampled below the Nyquist rate, the signal experienced aliasing, which distorted the frequency components and rendered the sampled data inaccurate. This phenomenon is a significant challenge in digital communications, as it can lead to misinterpreted data and lost information.
3. **Importance of Low-Pass Filtering**: The low-pass filter played a crucial role in reconstructing the original signal from its samples. By allowing only the necessary frequencies to pass through, the LPF effectively removed aliasing artifacts, underscoring the role of filtering in accurate signal reconstruction.
4. **Application to Digital Communication Systems**: The lab findings have direct applications in modern digital communication systems. Proper sampling and filtering techniques are essential to preserving data integrity in fields ranging from audio processing to telecommunications and biomedical imaging.

In summary, the Sampling Theorem provides the foundation for digital signal processing by establishing the necessary conditions for capturing and reconstructing signals without information loss. This lab emphasizes the need for careful attention to sampling rates and filter design to ensure accurate digital representations of analog signals, making these principles indispensable in the design and analysis of digital communication systems.

8. Acknowledgments

Special thanks to Dr. Sudip Mandal, Assistant Professor at Jalpaiguri Government Engineering College, West Bengal, India, for his MATLAB code resources. The MATLAB code provided by Dr. Mandal was referenced and adapted for this experiment, as seen on his YouTube channel: Digital Communication Labs using MATLAB.

9. References

- **Jupyter Notebook:**

1. *Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., ... & Willing, C. (2016). Jupyter Notebooks – a publishing format for reproducible computational workflows. In F. Loizides & B. Schmidt (Eds.), Positioning and Power in Academic Publishing: Players, Agents and Agendas (pp. 87–90). IOS Press.*

- **Digital Signal Processing with Python:**

2. *Sundararajan, V. (2021). Digital Signal Processing Using Python Programming. Springer.*

- **Matplotlib for Plotting in Python:**

3. *Hunter, J. D. (2007). "Matplotlib: A 2D graphics environment." Computing in Science & Engineering, 9(3), 90-95.*

- **NumPy for Scientific Computing:**

4. *Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). "Array programming with NumPy." Nature, 585(7825), 357-362.*