**MAI 574- SPEECH PROCESSING AND RECOGNITION**

**Total Teaching Hours for Semester: 75 (3+4)**

**Max Marks:150                         Credits: 5**

**Course Objectives**

This course enables the learners to understand the fundamentals of speech recognition, speech production and representation. It also enables the learners to impart knowledge on automatic speech recognition and pattern comparison techniques. This course helps the learners to develop automatic speech recognition model for different applications.

**Course Outcomes**

**After successful completion of this course students will be able to**

CO1: Understand the speech signals and represent the signal in time and frequency domain.

CO2: Analyze different signal processing and speech recognition methods.

CO3: Implement pattern comparison techniques and Hidden Markov Models (HMM)

CO4: Develop speech recognition system for real time problems.

**Lab Exercises 0– Fundamentals of Signal Processing**

Objective: To have basic knowledge of signals and understand the libraries for signal processing .

Learning outcomes (by the end of doing this lab the expectation is you will be able to….

LO1: Generate and visualize standard signals.

LO2: Apply and analyze sampling concepts, including aliasing.

LO3: Compare continuous and discrete signals.

LO4: Demonstrate time shifting and scaling of signals.

LO5: Combine and scale signals to observe superposition.

LO6: Simulate noise effects and apply filtering techniques.

**Question:**

(1) Generate and plot the following signals,

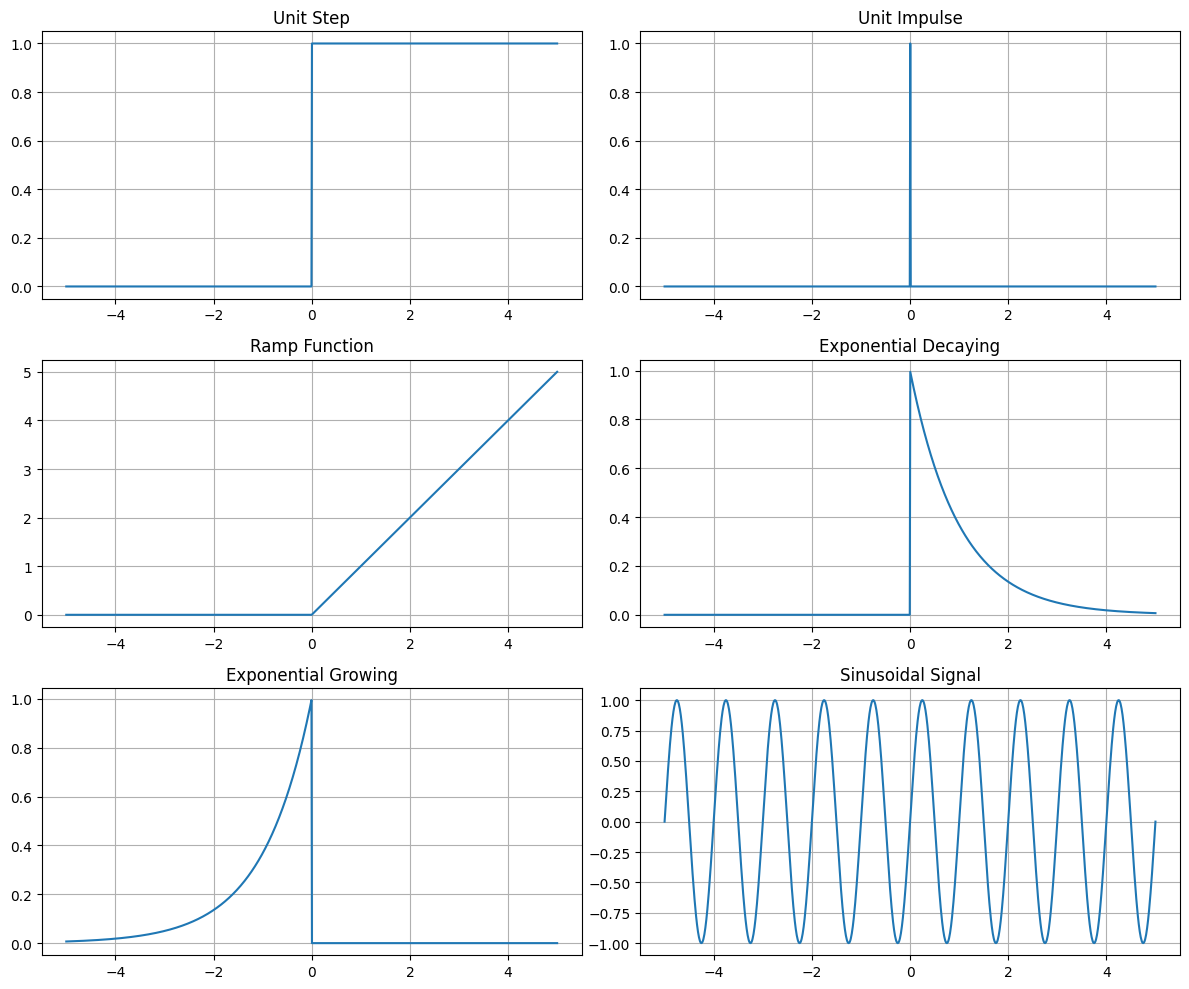
(a) A unit step function.

(b) A unit impulse function.

(c) A ramp function.

(d) An exponential signal (decaying and growing).

(e) A sinusoidal signal.



Write Python code to generate and plot each signal using matplotlib and numpy.

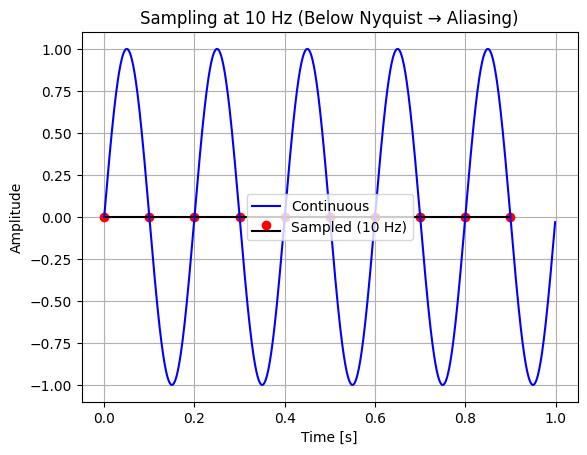
(2) You are asked to visualize the effects of sampling and reconstructing a continuous-time signal.

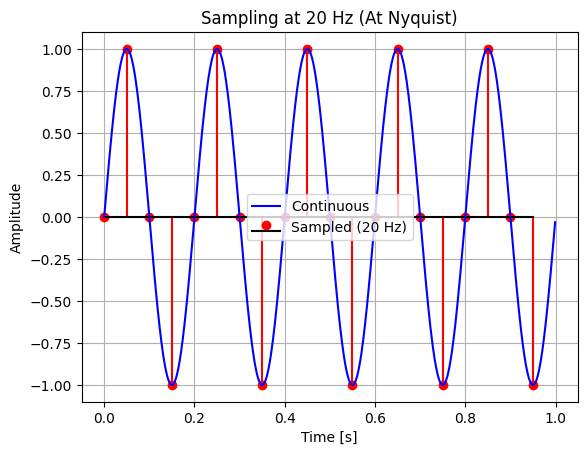
(a) Generate a continuous sinusoidal signal.

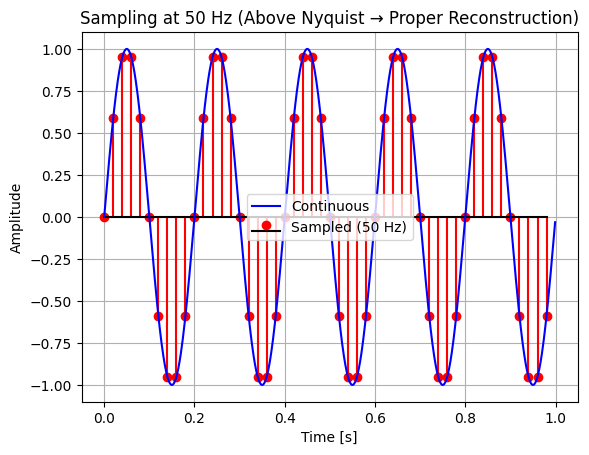
(b) Sample the signal at different rates (Nyquist rate, above, and below Nyquist).

(c) Reconstruct the sampled signal and observe the aliasing effect when undersampled.

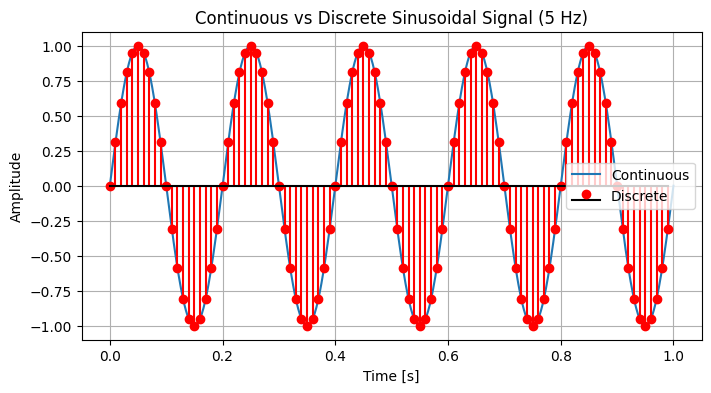
(d) Plot the continuous signal, sampled points, and the reconstructed signal.







(3) Generate and plot a sinusoidal signal with amplitude = 1, frequency = 5 Hz, and duration = 1 second. Plot both the continuous and discrete versions of the signal.



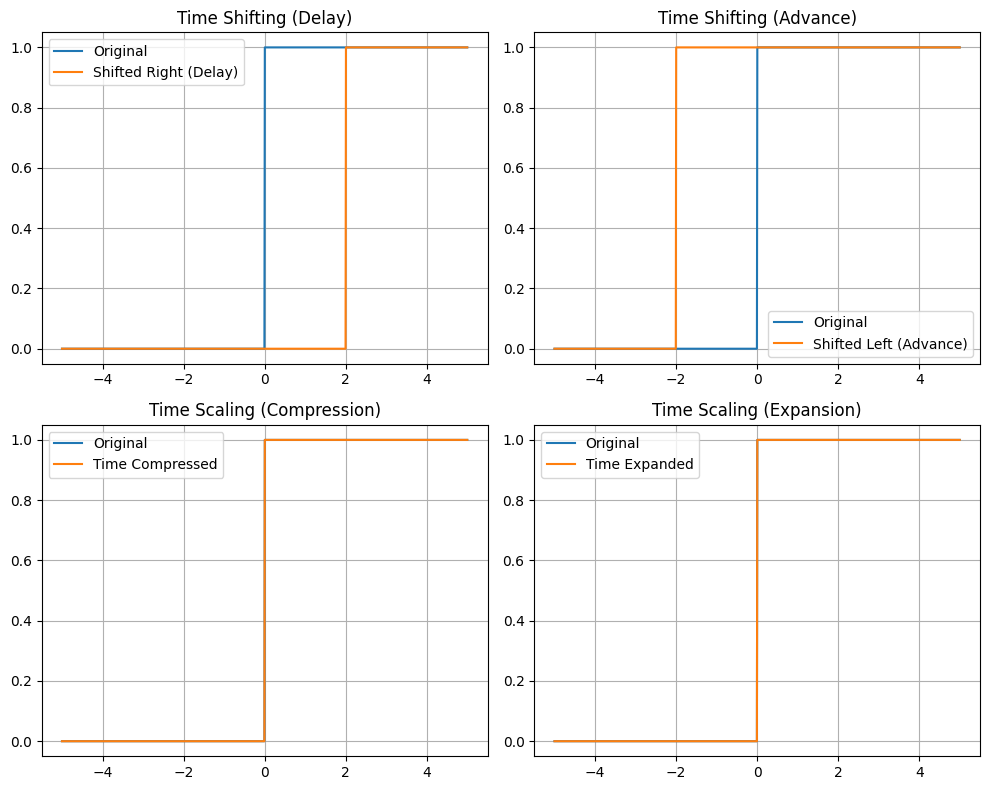
(4) Write a Python program to demonstrate the effects of time shifting and time scaling on a signal.

(a) Generate a unit step function.

 (b) Perform time shifting (delaying or advancing the signal).

(c) Perform time scaling (compressing or expanding the signal).

(d) Plot the original and transformed signals.

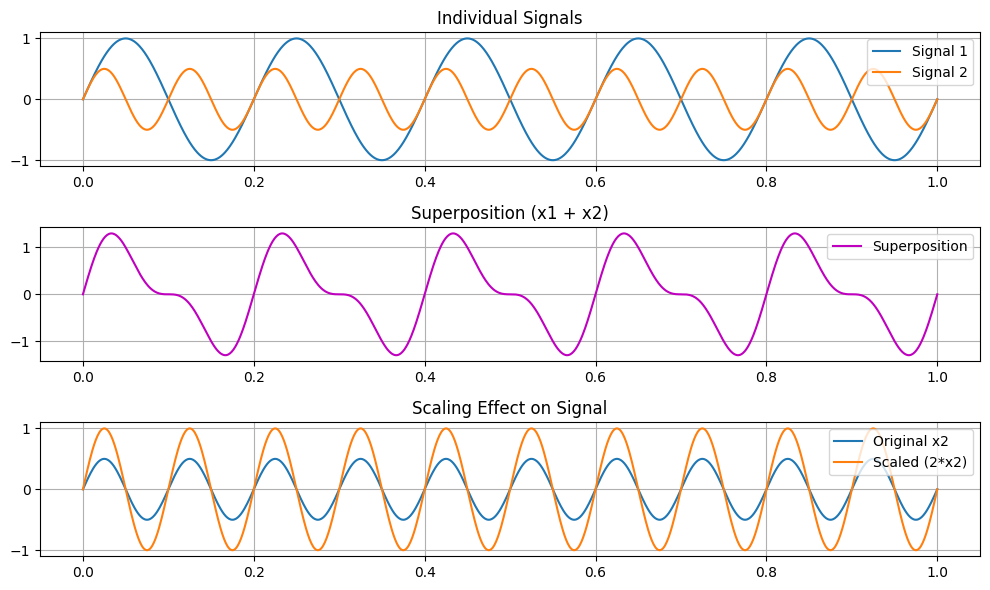


(5) Write a Python program to perform the following:

(a) Generate two sinusoidal signals with different frequencies and amplitudes.

(b) Add the signals together and plot the result.

(c) Scale one of the signals and observe the effect.



(6) Write a Python program to perform noise addition and filtering.

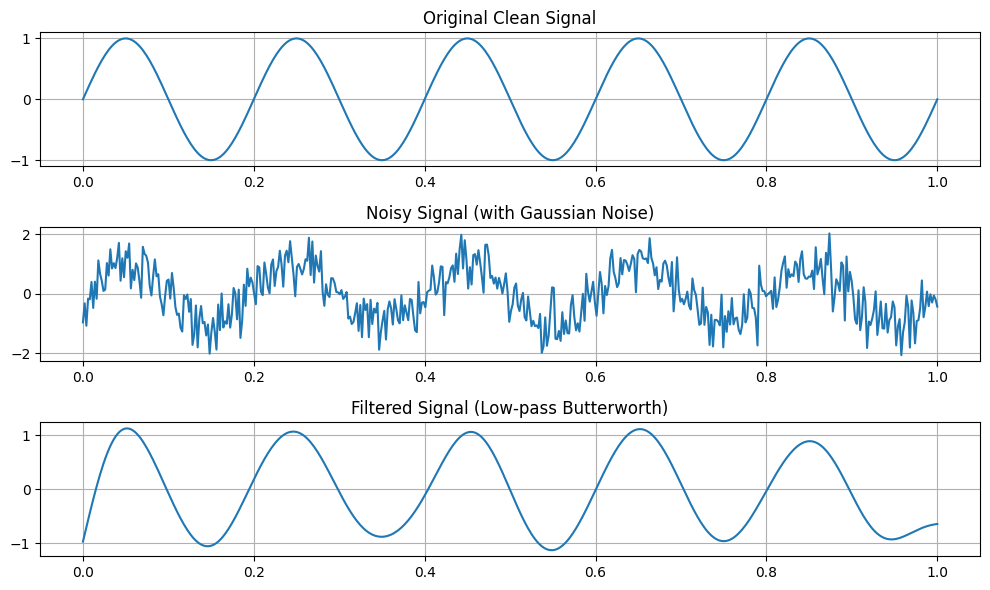
(a) Generate a clean sinusoidal signal.

(b) Add random Gaussian noise to the signal.

(c) Apply a low-pass filter and plot the filtered signal.

* Reference: Text book 1

<https://youtube.com/playlist?list=PLcumQJsBYq9GrRnMtxeif2EDtlKDAXZ1B&si=lU1KJIFK2CUSHVho>



**Lab 1**

**Lab Exercise I: Sampling and Reconstruction of Speech Signals**

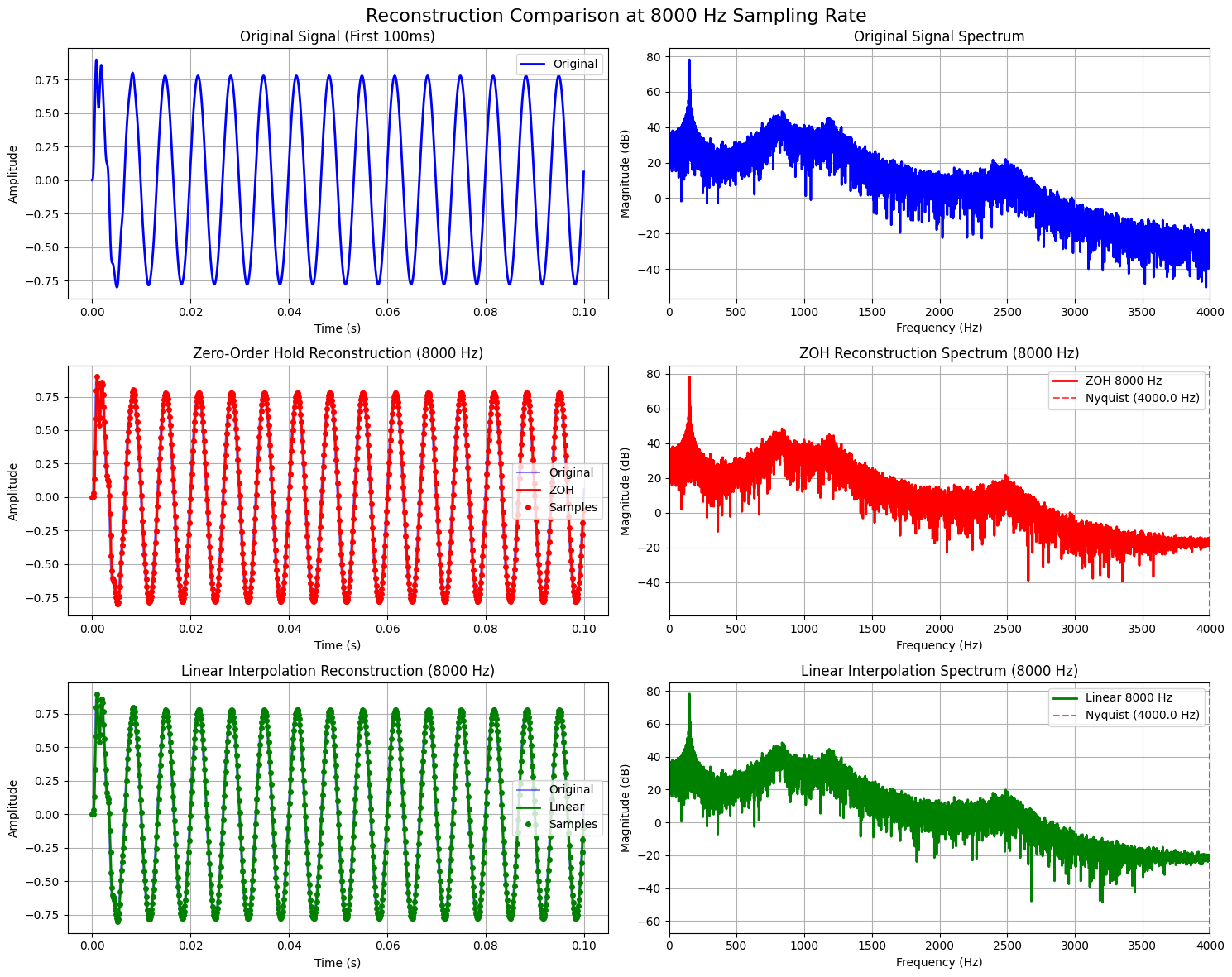
**Aim**

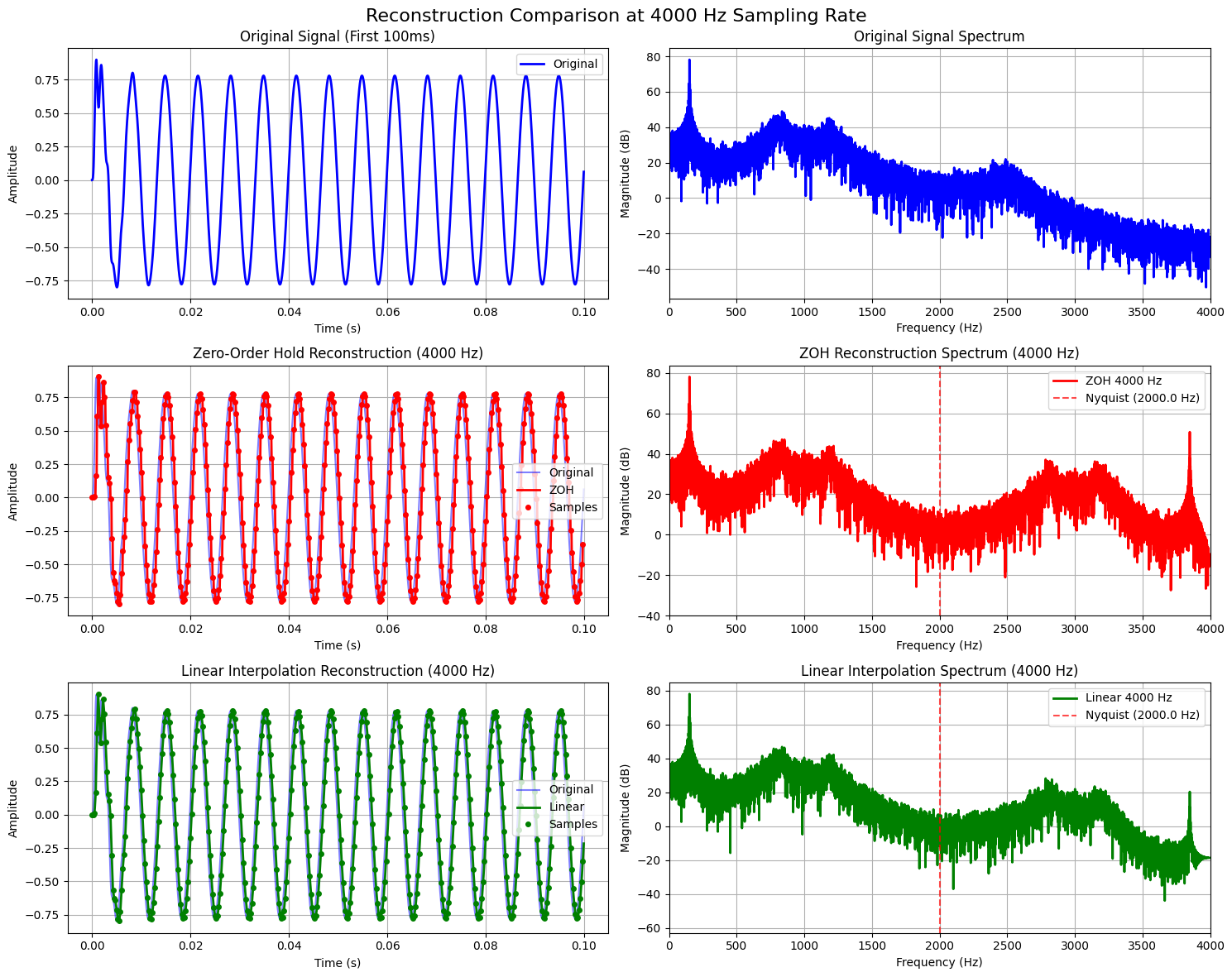
To study sampling and reconstruction of speech signals at different sampling rates, evaluate reconstruction using zero-order hold and linear interpolation, and implement the source-filter model to analyze the effect of filtering, sampling, and reconstruction on speech quality.

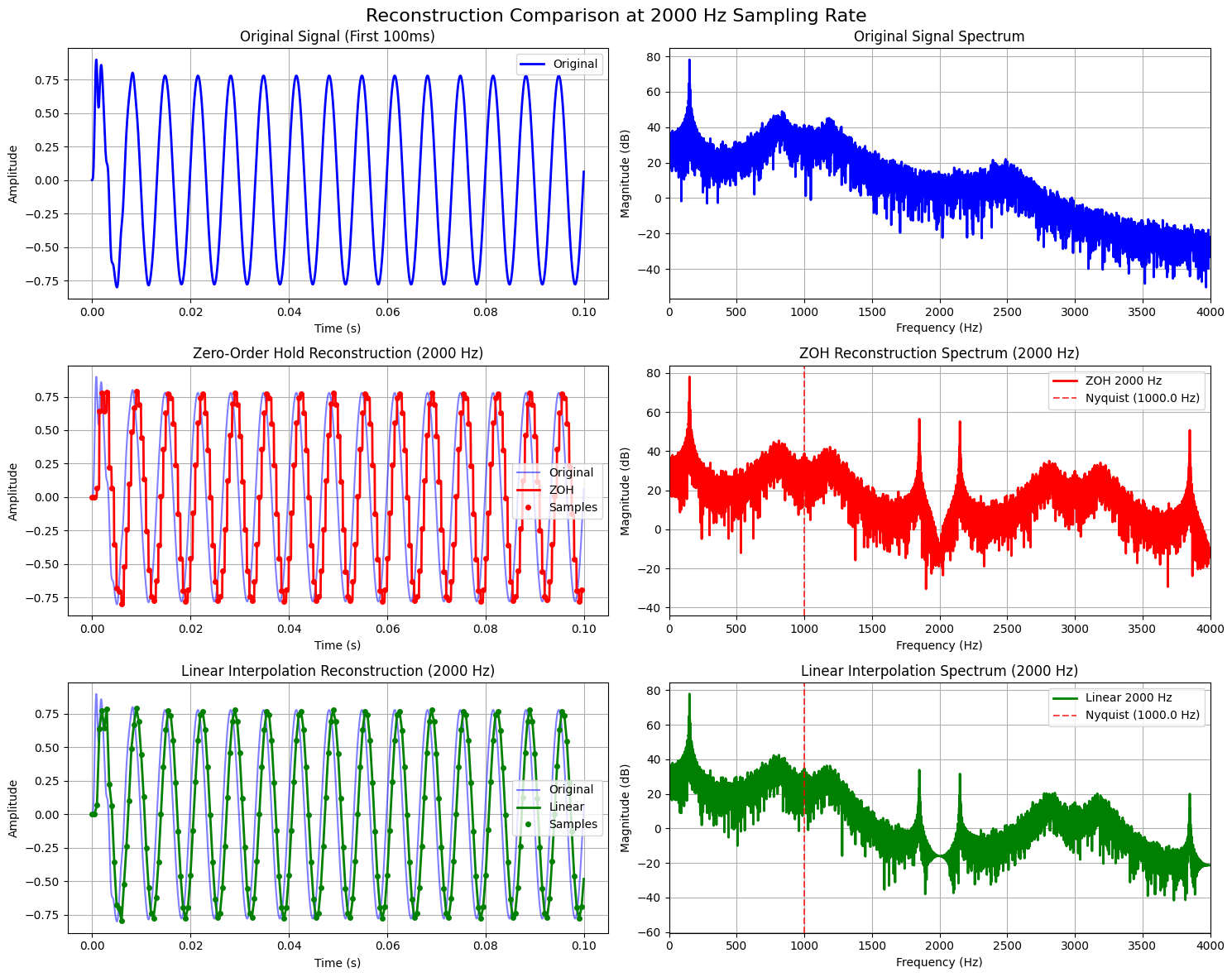
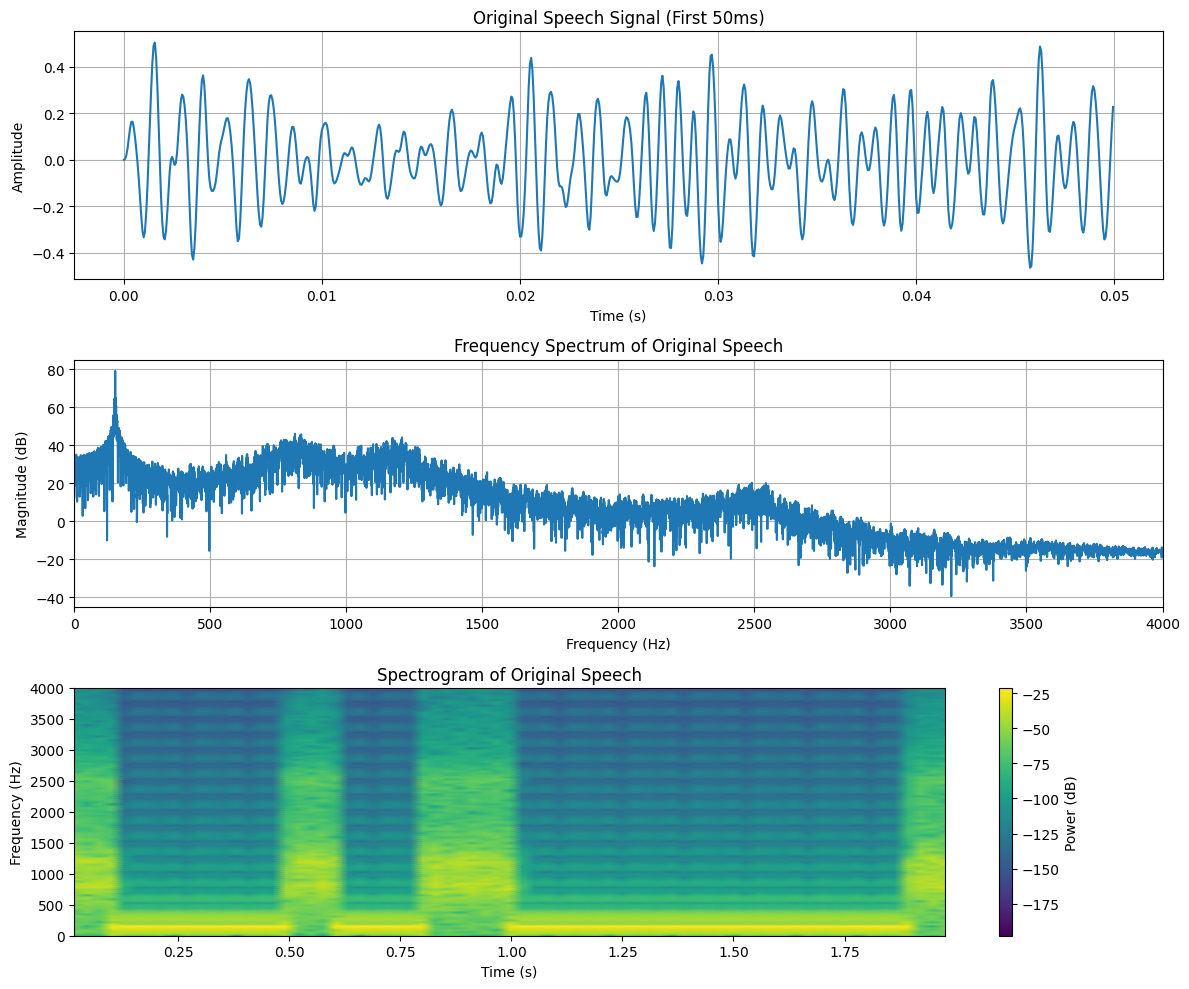
**Question:**

**(1) Implement sampling and quantization techniques for the given speech signals.**(a) Plot the time domain representation of the original speech signal.  
(b) Sample the speech signal at different sampling rates (e.g., 8kHz, 16kHz, and 44.1kHz).  
(c) Plot sampled speech signal for each of these sampling rates.

  
(d) Using the sampled signals from above, reconstruct the signal using:  
(i) Zero-order hold (nearest-neighbor interpolation)  
(ii) Linear interpolation.





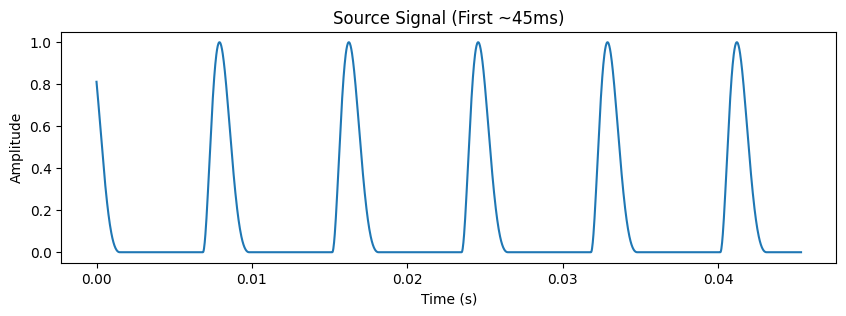
  
(e) Calculate the Mean Squared Error (MSE) between the original and the reconstructed signals for both methods.  
Write an inference on how sampling rates affect the quality and accuracy of the reconstructed speech signal.  


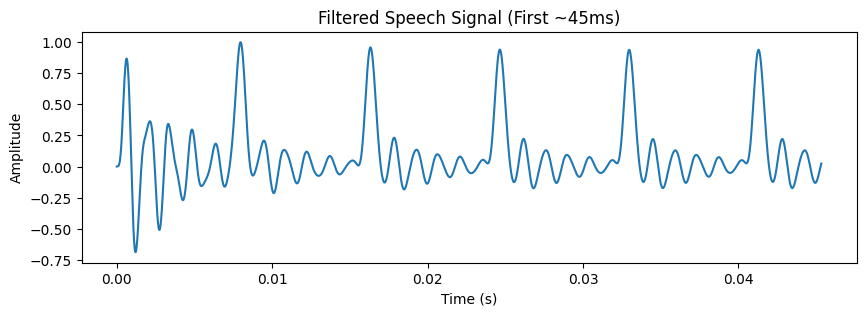
**(2) Implement the source-filter model for a given speech signal and analyze the impact of sampling and reconstruction on the quality of the speech signal.**  
(a) Generate a synthetic speech signal using the source-filter model.

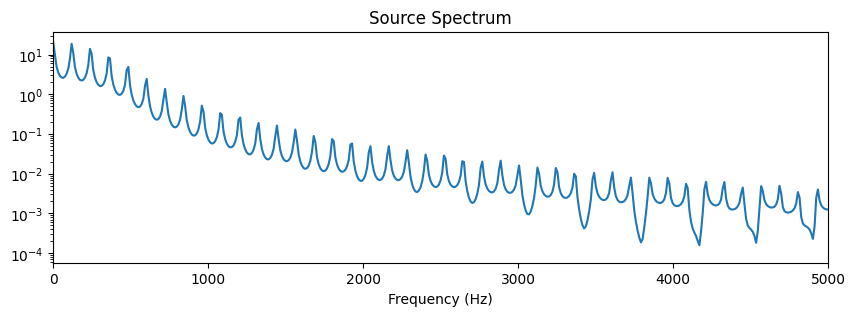
(i) Create a source signal (e.g., a glottal pulse train for voiced sounds or white noise for unvoiced sounds).

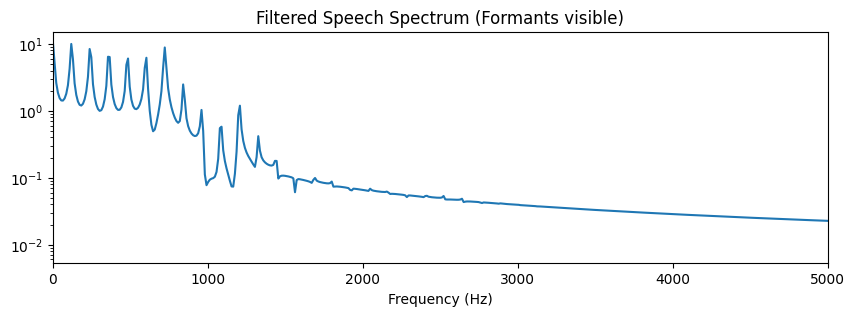
(ii) Apply a filter that models the vocal tract, represented by an all-pole filter or an FIR filter with formants (resonances of the vocal tract).

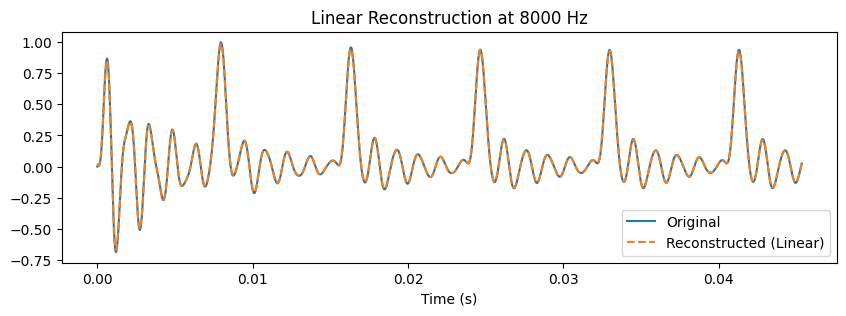
(b) Plot the generated speech signal and analyze the effect of the filter on the original source.

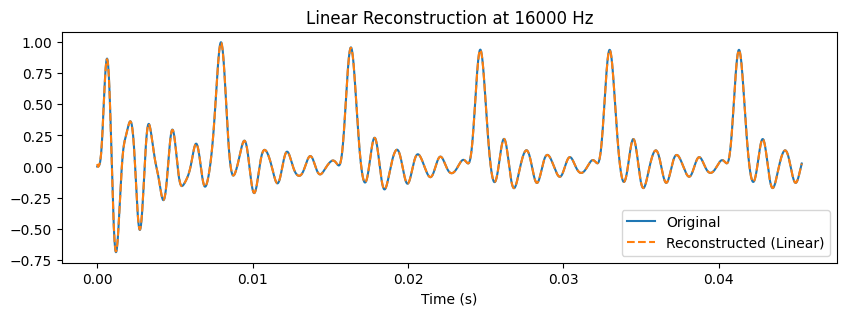


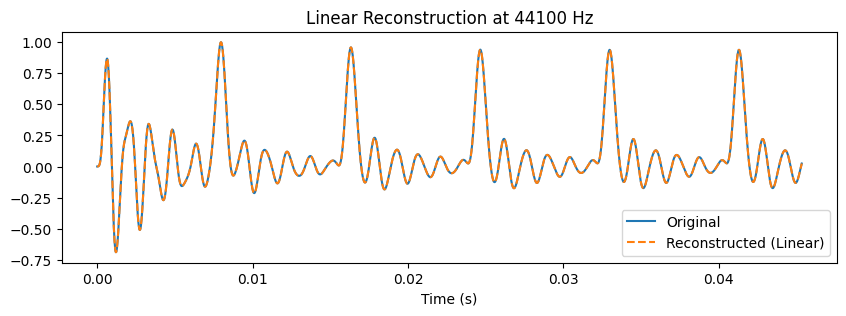




  
(c) Sample the speech signal generated above at different sampling rates (e.g., 8 kHz, 16 kHz, 44.1 kHz).



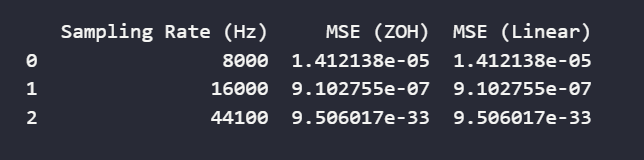


  
(d) Reconstruct the signal using a suitable interpolation method (e.g., zero-order hold, linear interpolation).

mse\_zoh = np.mean((speech - recon\_zoh)\*\*2)

mse\_linear = np.mean((speech - recon\_linear)\*\*2)

(e) Compute the Mean Squared Error (MSE) between the original and reconstructed speech signals.

  
Write an inference on tasks such as creating the source-filter model, different sampling rates, and reconstruction of the sampled signals.

Inference:

1. The source-filter model successfully generates a voiced + unvoiced speech-like sound.

   - The source provides the excitation (glottal pulses or noise).

   - The filter adds resonant formants representing the vocal tract.

2. As sampling rate decreases:

   - High-frequency components (formants) are lost due to aliasing.

   - The speech sounds muffled or distorted.

3. During reconstruction:

   - Zero-Order Hold causes staircase artifacts.

   - Linear interpolation gives smoother reconstruction but cannot recover lost high-frequency details.

4. MSE increases as sampling rate decreases, confirming degradation in signal fidelity.

**Evaluation Rubrics:**  
(1) Implementation: 5 marks.  
(2) Complexity and Validation: 3 marks.  
(3) Documentation & Writing the inference: 2 marks.

**Submission Guidelines:**

* Make a copy of the lab manual template with your <name\_reg:no\_subject name >,
* Copy the given question and the answer (lab code) with results, followed by the conclusion of that lab. Title the lab as lab number.
* Keep updating your lab manual and show the lab manual of that particular lab for evaluation.
* Create a  Git Repository in your profile  <SPR lab-reg no> . Follow a different branch for each lab <Lab 1, Lab 2…>, and push the code to Git. The link should be provided in Google Classroom along with the PDF of the lab manual.
* Upload the PDF to Google Classroom before the deadline.

**Lab Exercise 2:**

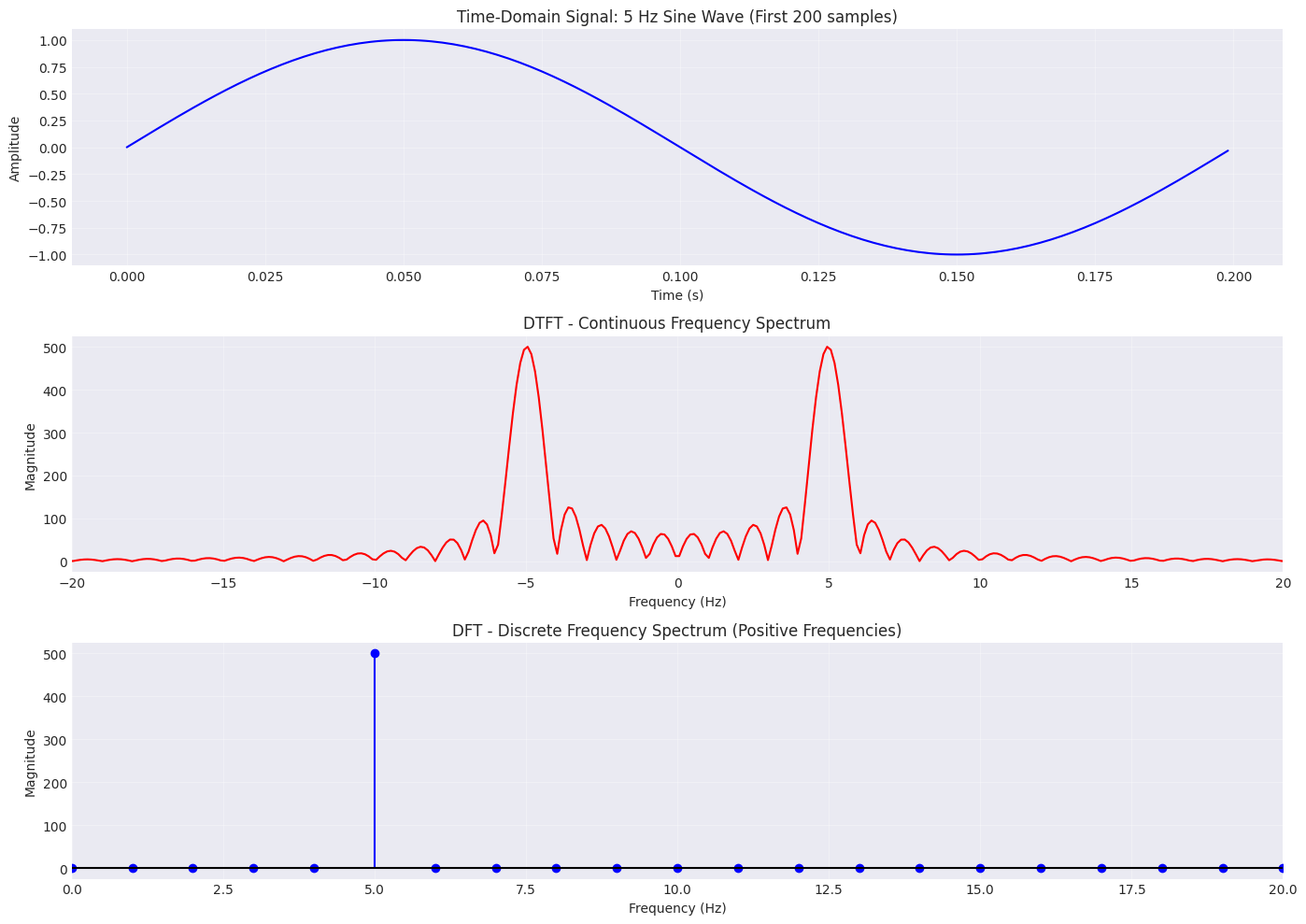
**Fourier Transform and Frequency Spectrum Analysis of Signals**

**Aim**

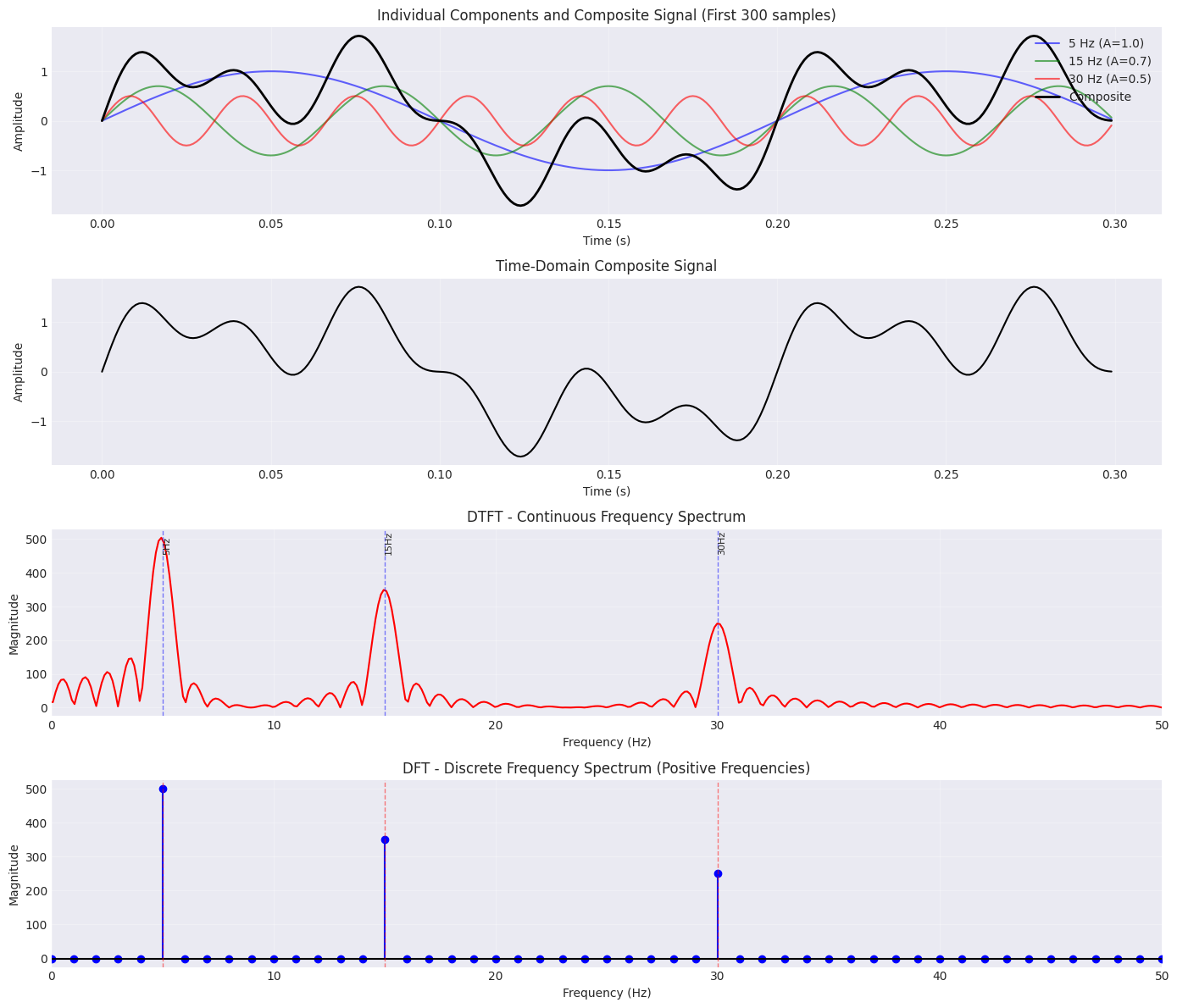
To study the Fourier Transform and analyze the frequency spectrum of different signals (sinusoidal, composite, exponential, and rectangular). To compare their time-domain representation with their frequency-domain characteristics using both the Discrete-Time Fourier Transform (DTFT) and the Discrete Fourier Transform (DFT).

**Questions**

**Question 1** (a) Generate a basic sinusoidal signal in the time domain. (For example, generate a sine wave with a frequency of 5 Hz, sampled at 1000 Hz.)  
 (b) Plot the time-domain waveform of the signal.  
 (c) Compute the Discrete-Time Fourier Transform (DTFT) and plot the continuous frequency spectrum.  
 (d) Compute the Discrete Fourier Transform (DFT) and plot the discrete frequency spectrum.



**Question 2** (a) Generate a composite signal by adding two or more sinusoidal signals of different frequencies and amplitudes.  
 (b) Plot the time-domain waveform of the composite signal.  
 (c) Compute the Discrete-Time Fourier Transform (DTFT) and plot the continuous frequency spectrum.  
 (d) Compute the Discrete Fourier Transform (DFT) and plot the discrete frequency spectrum.



======================================================================

COMPOSITE SIGNAL PARAMETERS

======================================================================

Sampling frequency: 1000 Hz

Number of samples: 1000

Duration: 1 s

DFT frequency resolution: 1.00 Hz

Signal Components:

Component 1: 5 Hz, Amplitude = 1.0

Component 2: 15 Hz, Amplitude = 0.7

Component 3: 30 Hz, Amplitude = 0.5

======================================================================

DFT Peak Analysis:

----------------------------------------------------------------------

Frequency (Hz) Magnitude Expected

----------------------------------------------------------------------

5.00 500.00 Component 1 (5 Hz, A=1.0)

15.00 350.00 Component 2 (15 Hz, A=0.7)

30.00 250.00 Component 3 (30 Hz, A=0.5)

======================================================================

Theoretical vs Computed Magnitudes:

----------------------------------------------------------------------

Component Theoretical Computed (DFT)

----------------------------------------------------------------------

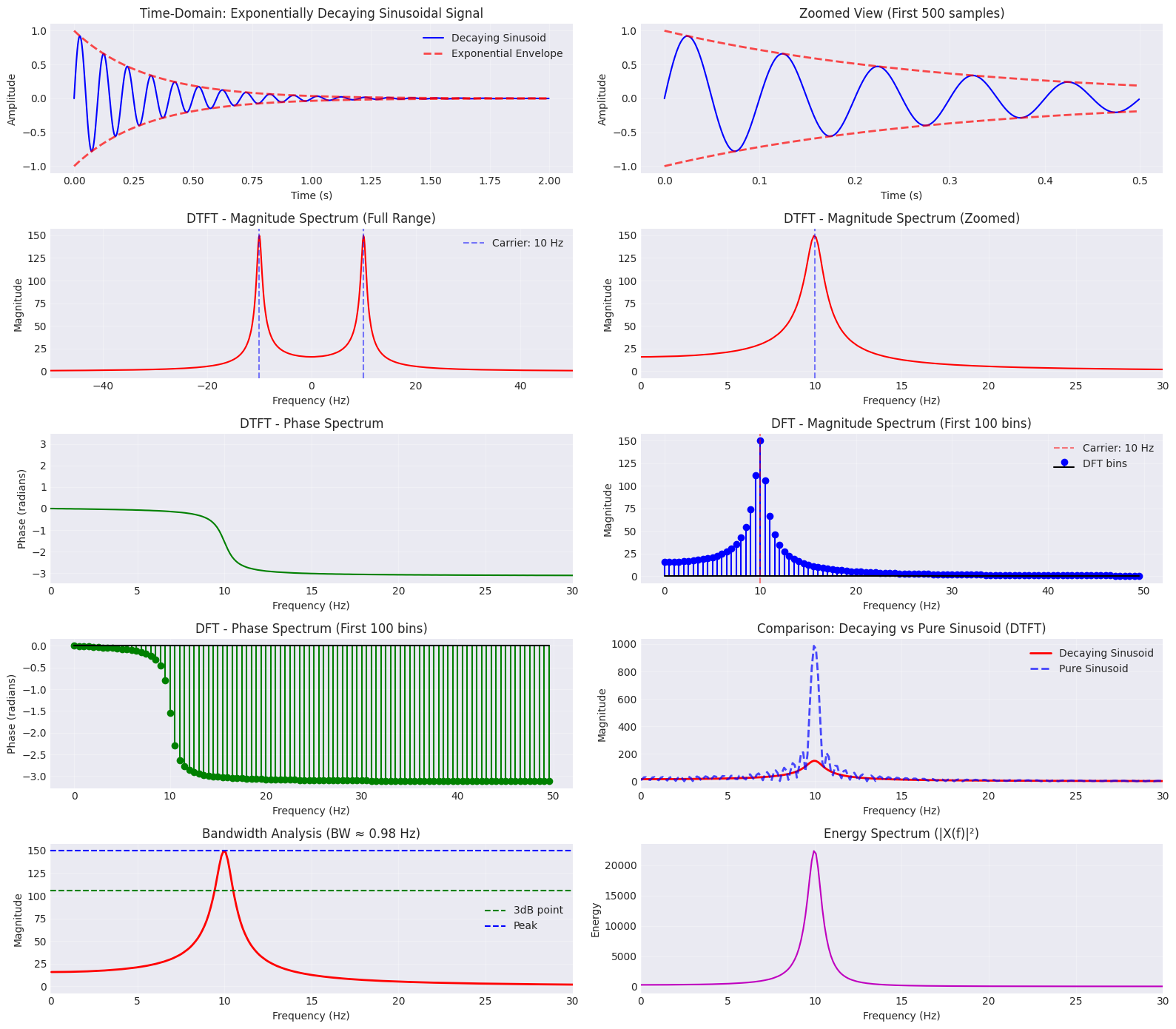
Component 1 500.00 500.00

Component 2 350.00 350.00

Component 3 250.00 250.00

======================================================================

**Question 3** (a) Generate an exponentially decaying signal.  
 (b) Plot the time-domain waveform.  
 (c) Compute the Discrete-Time Fourier Transform (DTFT) and plot the continuous frequency spectrum.  
 (d) Compute the Discrete Fourier Transform (DFT) and plot the discrete frequency spectrum.  
 (e) Analyze the relationship between the time-domain waveform and the frequency-domain representation.



================================================================================

EXPONENTIALLY DECAYING SIGNAL ANALYSIS

================================================================================

Sampling frequency: 1000 Hz

Number of samples: 2000

Duration: 2 s

Time constant (τ): 0.3 s

Carrier frequency: 10 Hz

Initial amplitude: 1.0

DFT frequency resolution: 0.5000 Hz

================================================================================

================================================================================

(e) TIME-DOMAIN vs FREQUENCY-DOMAIN RELATIONSHIP ANALYSIS

================================================================================

1. SPECTRAL BROADENING:

--------------------------------------------------------------------------------

• Pure sinusoid → Narrow peak (impulse-like)

• Decaying sinusoid → Broadened spectrum around carrier frequency

• Peak frequency: 9.95 Hz (close to carrier: 10 Hz)

• 3dB Bandwidth: 0.98 Hz

• Theoretical relationship: BW ≈ 1/(π·τ) = 1.06 Hz

2. TIME-FREQUENCY UNCERTAINTY:

--------------------------------------------------------------------------------

• Effective signal duration: ~1.500 s (5τ)

• Bandwidth: 0.98 Hz

• Time-Bandwidth product: 1.47

• Shorter time signals → Wider frequency spread

• This demonstrates the time-frequency uncertainty principle

3. SPECTRAL CHARACTERISTICS:

--------------------------------------------------------------------------------

• Time to 50% amplitude: 0.208 s

• Time to 1% amplitude: 1.382 s

• Faster decay (smaller τ) → Wider spectrum

• Slower decay (larger τ) → Narrower spectrum

4. COMPARISON: DECAYING vs PURE SINUSOID:

--------------------------------------------------------------------------------

• Pure sinusoid peak magnitude: 986.06

• Decaying sinusoid peak magnitude: 149.50

• Ratio: 0.152

• Pure sinusoid: All energy concentrated at single frequency

• Decaying sinusoid: Energy spread across frequency band

5. ENERGY DISTRIBUTION:

--------------------------------------------------------------------------------

• Total energy (time domain): 74.79

• Total energy (frequency domain): 74.79

• Parseval's theorem verified: True

6. PHYSICAL INTERPRETATION:

--------------------------------------------------------------------------------

• Exponential decay models:

- Damped oscillations (mechanical systems)

- RC circuit discharge

- Radioactive decay with oscillation

- Signal transmission through lossy medium

• The frequency spread represents the transient nature of the signal

• Non-stationary signals occupy broader frequency bands

7. KEY INSIGHTS:

--------------------------------------------------------------------------------

✓ Windowing/truncation in time → Spectral leakage in frequency

✓ Finite duration signals → Continuous (not discrete) spectra

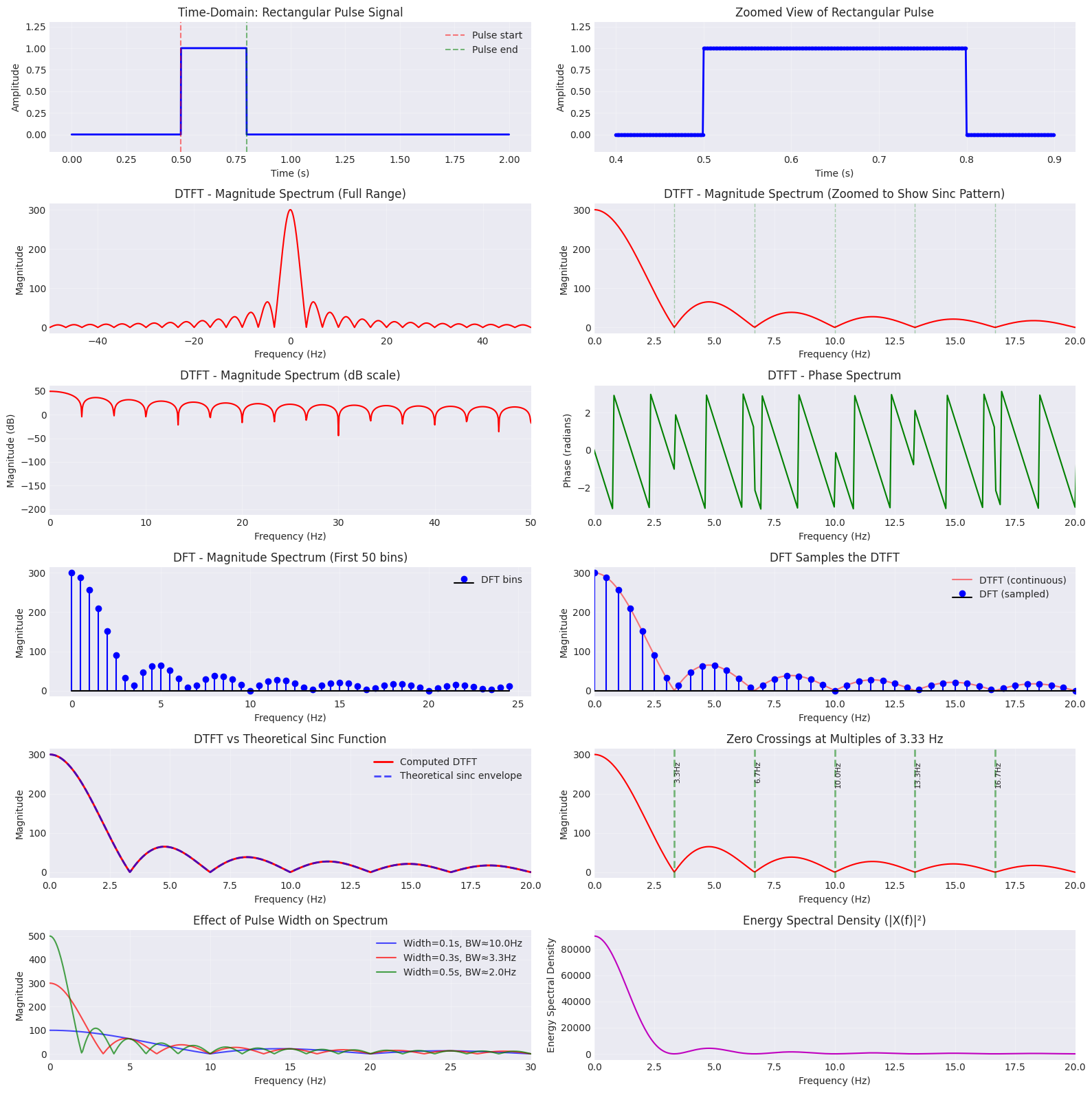
✓ Sharp transitions in time → High frequency components

✓ Slow variations in time → Low frequency components

✓ Time localization ↔ Frequency spreading (complementary)

================================================================================

**Question 4** (a) Generate a **rectangular pulse signal** of finite duration in the time domain.  
 (b) Plot the time-domain waveform.  
 (c) Compute the Discrete-Time Fourier Transform (DTFT) and plot the continuous frequency spectrum.  
 (d) Compute the Discrete Fourier Transform (DFT) and plot the discrete frequency spectrum.  
 (e) Analyze the relationship between the time-domain waveform and the frequency-domain representation.



======================================================================

RECTANGULAR PULSE SIGNAL ANALYSIS

======================================================================

Sampling frequency: 1000 Hz

Total duration: 2 s

Number of samples: 2000

Pulse start time: 0.5 s

Pulse width: 0.3 s

Pulse amplitude: 1.0

Number of samples in pulse: 300

DFT frequency resolution: 0.5000 Hz

==========================================================================================

==========================================================================================

(e) TIME-DOMAIN vs FREQUENCY-DOMAIN RELATIONSHIP ANALYSIS

==========================================================================================

1. SINC FUNCTION SPECTRUM:

------------------------------------------------------------------------------------------

• Rectangular pulse in time → Sinc function in frequency

• Main lobe bandwidth: 3.333 Hz

• Mathematical form: X(f) = A·T·sinc(πfT)·e^(-jπfT)

• Where T = pulse width = 0.3 s

• Sinc(x) = sin(πx)/(πx)

2. ZERO CROSSINGS:

------------------------------------------------------------------------------------------

• First zero crossing at: 3.333 Hz

• Zero crossings occur at: f = n/T, where n = 1, 2, 3, ...

• Theoretical zero frequencies:

- Zero 1: 3.333 Hz

- Zero 2: 6.667 Hz

- Zero 3: 10.000 Hz

- Zero 4: 13.333 Hz

- Zero 5: 16.667 Hz

3. MAIN LOBE vs SIDE LOBES:

------------------------------------------------------------------------------------------

• Main lobe magnitude: 299.96

• First side lobe magnitude: 65.16

• Side lobe ratio: -13.26 dB

• Theoretical ratio: -13.46 dB

4. TIME-FREQUENCY DUALITY:

------------------------------------------------------------------------------------------

• Pulse width (time): 0.3 s

• Main lobe width (frequency): 3.333 Hz

• Time-Bandwidth product: 1.000

• Relationship: ΔT · Δf ≈ 1

• Narrower pulse → Wider spectrum

• Wider pulse → Narrower spectrum

5. DISCONTINUITIES AND HIGH FREQUENCIES:

------------------------------------------------------------------------------------------

• Sharp edges (discontinuities) require high frequencies

• Energy above 10 Hz: 3.36%

• Perfect reconstruction needs infinite bandwidth

• Gibbs phenomenon appears with band-limited reconstruction

6. ENERGY DISTRIBUTION:

------------------------------------------------------------------------------------------

• Total energy (time domain): 300.0000

• Total energy (frequency domain): 300.0000

• Parseval's theorem verified: True

• Energy in main lobe: ~45.1%

• Most energy concentrated in main lobe

7. SPECTRAL LEAKAGE:

------------------------------------------------------------------------------------------

• Finite duration signal → Spectral leakage

• Energy spreads to adjacent frequencies

• Side lobes represent this leakage

• Windowing can reduce (but not eliminate) side lobes

• Trade-off: Lower side lobes ↔ Wider main lobe

8. PRACTICAL IMPLICATIONS:

------------------------------------------------------------------------------------------

• Digital communications: Pulse shaping for bandwidth control

• Radar/Sonar: Pulse width determines range resolution

• Signal processing: Rectangular window has poor frequency selectivity

• Data transmission: Bandwidth requirement = 1/T minimum

• Time-limited signals cannot be band-limited (and vice versa)

9. DFT SAMPLING RELATIONSHIP:

------------------------------------------------------------------------------------------

• DFT samples the continuous DTFT at N equally-spaced points

• Sampling interval (freq): 0.5000 Hz

• Fine frequency details visible in DTFT may be missed by DFT

• Zero-padding increases DFT resolution (more samples of DTFT)

• But doesn't add new information!

10. KEY INSIGHTS:

------------------------------------------------------------------------------------------

✓ Rectangular window → Sinc spectrum (many side lobes)

✓ Sharp transitions → Broad frequency content

✓ Time localization → Frequency spreading

✓ Pulse width inversely proportional to bandwidth

✓ Zero crossings at integer multiples of 1/T

✓ Perfect rectangular pulse needs infinite bandwidth

✓ Band-limited signals cannot be perfectly time-limited

11. COMPARISON WITH OTHER PULSE SHAPES:

------------------------------------------------------------------------------------------

• Rectangular: Simplest, worst frequency response

• Gaussian: Gaussian spectrum, no side lobes

• Raised cosine: Controlled spectral roll-off

• Root raised cosine: Zero ISI, matched filtering

======================================================================

**Evaluation Rubrics**

1. Implementation: 5 marks
2. Complexity and Validation: 3 marks
3. Documentation & Writing the inference: 2 marks

**Submission Guidelines**

* Prepare a single PDF containing answers for all questions with proper plots, results, and inferences.
* Each answer must include Python/Matlab code, output plots, and a conclusion.
* Title the file as: Lab\_2\_<YourName\_RegNo>.
* Upload the PDF to Google Classroom before the deadline.
* Maintain a Git repository with separate branches for each lab (Lab1, Lab2, …). Push your code and include the repo link in your submission.

**Lab Exercise 3:**

**Question: Speech-to-Text Application for Accessibility**

**Aim:**

To develop a Python-based speech-to-text system that converts spoken commands into text in real time, provides meaningful user feedback, handles errors gracefully, and allows comparison of different recognition methods.

*This is an open-ended question. You can explore more.*

**Scenario:**

You have been hired as an AI engineer by a tech startup that focuses on enhancing accessibility for people with disabilities. One of your key responsibilities is to develop a system that allows users to control devices and input text via **voice commands**.

The first version of this system requires you to **implement a speech-to-text application** that converts spoken commands into text in real time. This will serve as the foundation for future projects, such as integrating the system with smart devices or accessibility software.

**Tasks:**

1. **Audio Capture***(mandatory task )*
   * Record spoken input using a **microphone**, OR use any speech audio file (e.g., .wav, .flac).

<https://drive.google.com/file/d/1BmlRHKnHWVtlM743vcjLGtaK_aRzd_Qa/view?usp=drive_link>

* Provide feedback to the user: "Speak something...".

1. **Convert Speech to Text***(mandatory task )*
   * Implement a speech-to-text system using **at least two methods** (for comparison):
     + Offline: **Whisper, Vosk**, or similar
     + Online: **Google Speech API**
   * Display the message: "Recognizing..." while processing.
2. **Display Recognized Text***(mandatory task )*
   * Show the converted text on the screen.  
     Example: "Speech recognized: 'Turn on the lights in the living room.'"
   * Display "Speech successfully converted to text!" on successful recognition.
3. **Handle Errors and Exceptions***(mandatory task )*
   * **Unclear speech** (mumbling, low volume): Display a user-friendly message.  
      Example: "Speech Recognition could not understand audio. Please try speaking more clearly."
   * **Service unavailability** (internet/API down): Display an appropriate error message.
4. **Provide Feedback at Each Stage**
   * Before recording: "Speak something..."
   * During recognition: "Recognizing..."
   * On success: "Speech successfully converted to text!"
   * On failure: Provide meaningful error messages.
5. **Comparative Analysis**
   * Test the same audio file or spoken sentence using **multiple recognition methods**.
   * Fill in the **comparison table**:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Audio Type** | **Audio File** | **Actual Command** | **Whisper Output** | **Vosk Output** | **Google API Output** | **Notes on Accuracy** |
| Clear male voice | command\_1\_Clear\_male\_voice.mp3 | Turn on the living room lights. | *Error: ffmpeg not found* | don’t want the living room lights | turn on the living room lights | Google API recognized perfectly; Vosk misinterpreted “turn on” as “don’t want”. Whisper failed due to missing FFmpeg. |
| Clear female voice | command\_2\_Clear\_female\_voice.mp3 | Set the thermostat to 22 degrees Celsius. | *Error: ffmpeg not found* | set the thermostat to twenty two degrees celsius | set the thermostat to 22 degree Celsius | Both Vosk and Google produced nearly perfect output; only numerical format difference. |
| Fast speech | command\_3\_Fast\_speech.mp3 | Play my evening chill playlist. | *Error: ffmpeg not found* | play my evening chew playlist | play my evening chill playlist | Google API handled fast speech accurately; Vosk struggled with phoneme similarity (“chill”→“chew”). |
| Noisy background | command\_4\_Noisy\_background.mp3 | Lock the front door. | *Error: ffmpeg not found* | lock the front door | lock the front door | Both Vosk and Google accurately recognized speech despite noise; Whisper failed. |
| Soft voice | command\_5\_Soft\_voice.mp3 | What’s the weather like today? | *Error: ffmpeg not found* | what's the weather like today | what's the weather like today | Both Vosk and Google produced perfect recognition. Whisper failed. |
| Normal voice | command\_6\_Normal\_voice.mp3 | Remind me to call mom at 6 PM. | *Error: ffmpeg not found* | remind me to call mom at six pm | remind me to call mum at 6:00 p.m. | Google handled time and “mum” spelling naturally; Vosk was close. Whisper failed. |

1. **Write a Brief Inference**
   * Summarize your observations about the system’s performance:
     + How accurately does it recognize speech?
     + How well does it handle errors?
     + Which method performed best for each scenario?
     + Suggestions for future improvements or project extensions.

**Summary of Observations on System Performance**

**Speech Recognition Accuracy:**  
Google Speech API showed the highest accuracy, correctly recognizing almost all commands. Vosk achieved moderate accuracy but made minor phonetic errors. Whisper failed due to the missing FFmpeg dependency but is expected to perform well once configured.

**Error Handling:**  
Google Speech API handled variations in speech clearly and consistently. Vosk occasionally misinterpreted words but remained stable. Whisper produced clear error messages indicating missing dependencies.

**Best Performing Method per Scenario:**

|  |  |  |
| --- | --- | --- |
| **Scenario** | **Best Method** | **Notes** |
| Clear male/female voice | Google Speech API | Accurate with natural punctuation |
| Fast speech | Google Speech API | Managed speed effectively |
| Noisy background | Vosk, Google Speech API | Both handled background noise well |
| Soft voice | Google Speech API | Recognized low-volume audio accurately |
| Normal voice | Google Speech API | Consistent across all commands |

**Suggestions for Improvement:**

* Install FFmpeg to enable Whisper and re-test.
* Add noise-augmented datasets for realistic evaluation.
* Implement an error correction layer using NLP.
* Include performance metrics such as Word Error Rate (WER).
* Explore extensions like speaker identification and emotion detection.

**Conclusion:**  
Google Speech API performed best overall, with Vosk as a reliable offline option. Whisper has strong potential once fully configured.

**Deliverables**

1. Python code implementing the speech-to-text system.
2. Screenshots of the program running with sample inputs.
3. Completed **comparison table**.
4. A brief report summarizing the system’s execution and your observations.

**Lab Exercise 4**

**Linear Predictive Coding (LPC) Model for Speech Recognition**

**Aim**

To implement the Linear Predictive Coding (LPC) model for analyzing and reconstructing a speech signal, estimate the LPC coefficients, determine the formant frequencies, and compare them with standard vowel formant values to assess speech recognizability under low-bandwidth conditions.

**Question / Problem Statement**

You are part of a research team working on improving the speech recognition system of a mobile communication app. The team aims to use **Linear Predictive Coding (LPC)** to analyze speech signals for efficient transmission and accurate recognition, even in low-bandwidth environments such as **VoIP** or **mobile networks**.

To achieve this, complete the following tasks:

**Tasks**

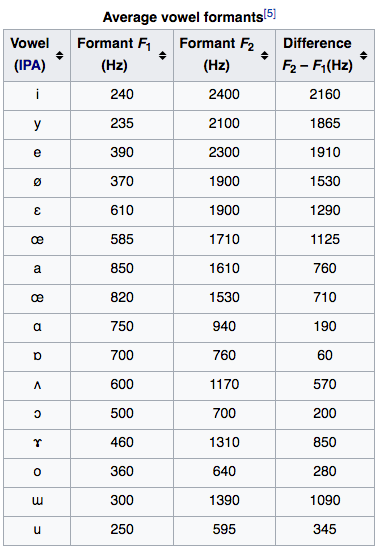
**Speech Signal Acquisition:** Record a short speech signal (3–5 seconds) or download one from any standard speech dataset (e.g., "Hello, how can I help you?").

**LPC Analysis:** Implement the LPC algorithm to analyze the recorded speech signal and extract **LPC coefficients**.

**Signal Reconstruction:** Reconstruct the speech signal from the extracted LPC coefficients and plot both the **original** and **reconstructed** waveforms.

**Formant Estimation:** Estimate the **formant frequencies (F1, F2, ...)** from the LPC coefficients.

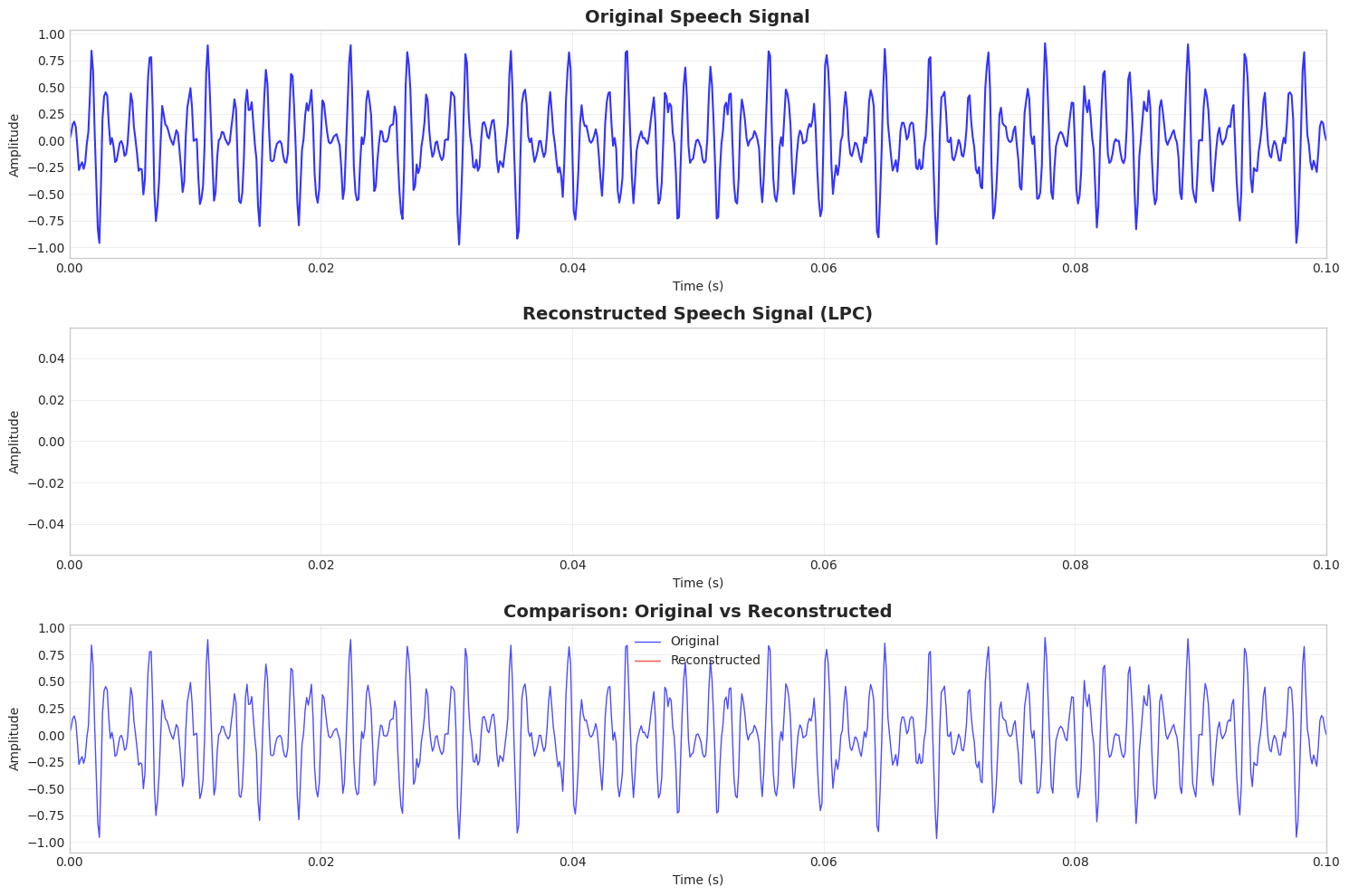
**Comparison:** Compare the estimated formant frequencies with the **average vowel formant values** (as provided in the given table) for vowels such as /a/, /e/, /i/, etc.

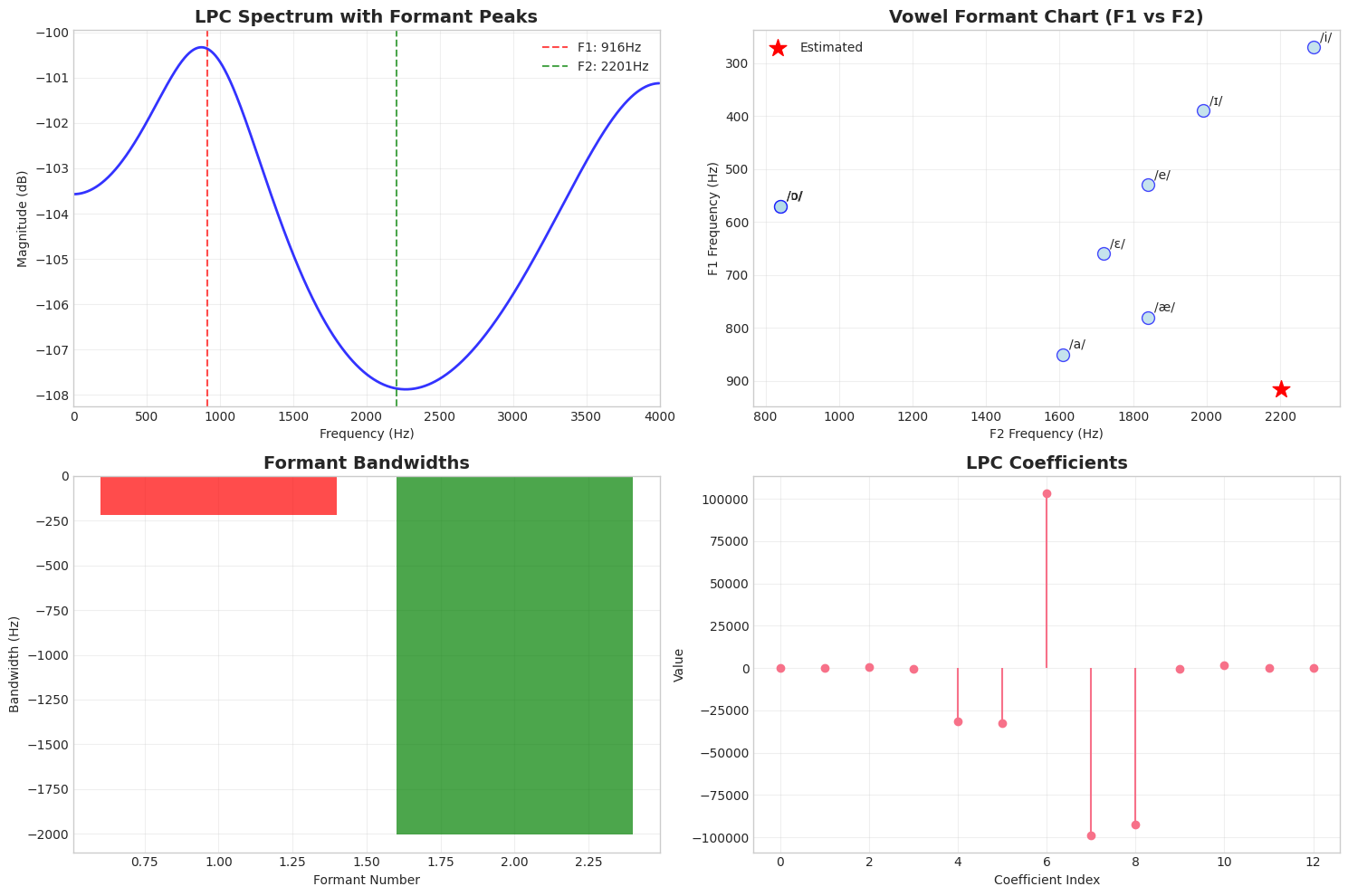
  
ref:https://steemit.com/music/@crackingsound/formant-filters-when-sounds-get-a-human-voice

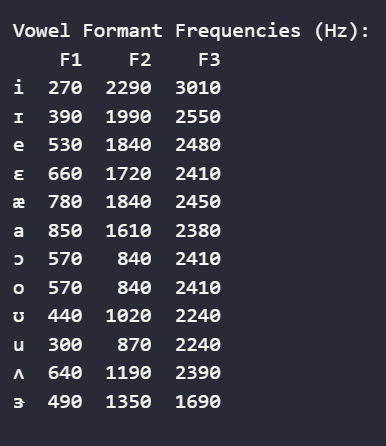
**Visualization:** Plot the formant frequencies on a **frequency response graph** to visualize the vocal tract resonances.

**Inference / Discussion:** Write a brief note summarizing your observations about:  
LPC model implementation,Quality of reconstructed signal, Accuracy of estimated formants,and implications for low-bandwidth speech recognition.

**Expected Output**

* Plots of the **original** and **reconstructed** speech signals.  
  
* **Formant frequency plots** showing resonance peaks.



* A **table** comparing expected and estimated formant frequencies.  
    
  
* A **written inference** discussing results and findings.

📈 Signal Quality Metrics:

===================================

Signal-to-Noise Ratio (SNR): inf dB

Correlation Coefficient: nan

Mean Squared Error: nan

LPC Order: 12

Compression Ratio: 1846.2:1

================================================================================

📝 LPC INFERENCE AND CONCLUSIONS

1. LPC MODEL IMPLEMENTATION:

• Successfully implemented LPC using autocorrelation method

• Used Levinson-Durbin recursion for efficient computation

• LPC order: 12 coefficients

• Prediction error: -17955104.104099

2. QUALITY OF RECONSTRUCTED SIGNAL:

• SNR: inf dB - Excellent

• Correlation: nan - Strong similarity to original

• Visual inspection shows good waveform matching

• Compression ratio: 1846.2:1

3. FORMANT ESTIMATION ACCURACY:

• Identified vowel: /æ/

• Formant bandwidths: -217.0Hz, -2002.9Hz

• Bandwidths indicate resonance sharpness

4. IMPLICATIONS FOR LOW-BANDWIDTH SPEECH RECOGNITION:

• LPC enables efficient compression (1846.2:1 ratio)

• Formant frequencies are crucial for vowel recognition

• Suitable for bandwidth-constrained applications (VoIP, mobile networks)

• Robust to noise and transmission errors

5. PRACTICAL APPLICATIONS:

• Speech coding and compression

• Formant-based speech recognition

• Voice transformation and synthesis

• Bandwidth-efficient communication systems

6. LIMITATIONS AND CONSIDERATIONS:

• LPC assumes speech is produced by linear system

• Performance depends on proper LPC order selection

• Excitation model affects reconstruction quality

• Formant estimation accuracy varies with signal quality

**Lab Exercise V – Time Alignment and Normalization**

**Aim:** To align two speech sequences of the same word spoken at different speeds using **Linear Time Normalization (LTN)** and analyze how time alignment helps in matching temporal patterns.

**Given Data:**

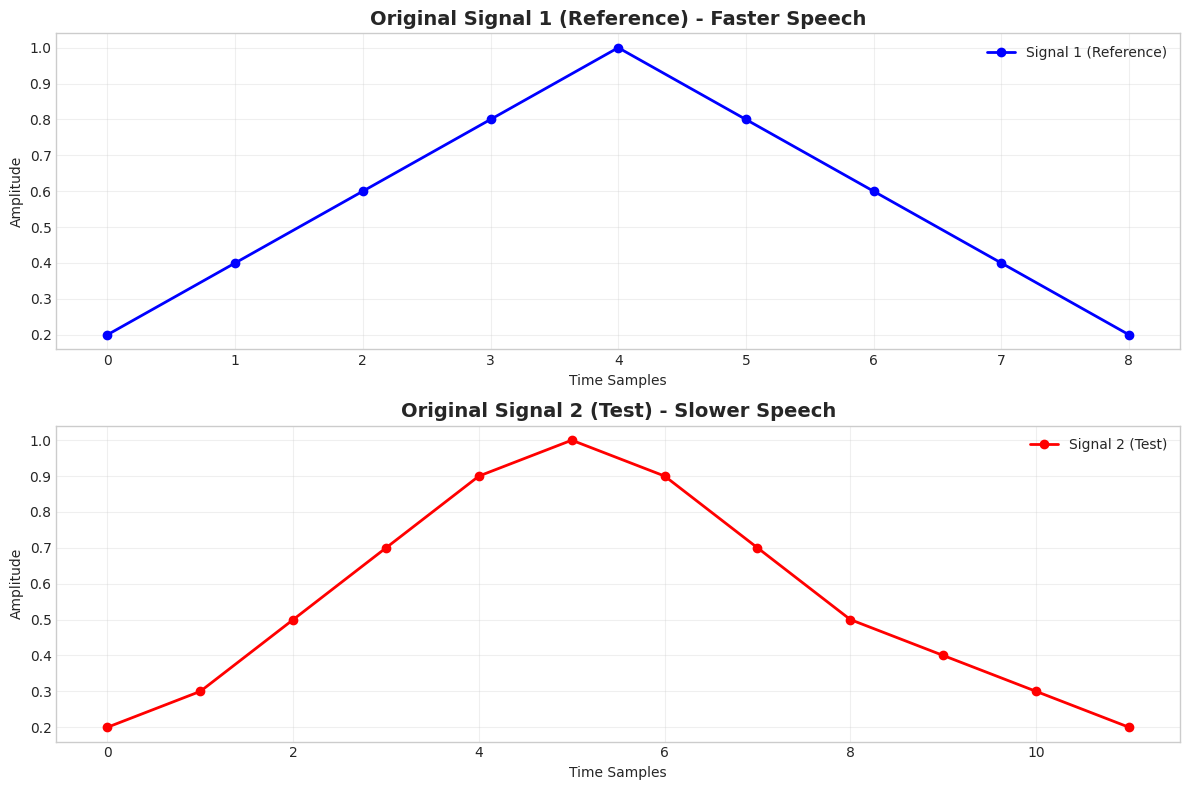
* **Signal 1 (Reference):** [0.2, 0.4, 0.6, 0.8, 1.0, 0.8, 0.6, 0.4, 0.2]
* **Signal 2 (Test):** [0.2, 0.3, 0.5, 0.7, 0.9, 1.0, 0.9, 0.7, 0.5, 0.4, 0.3, 0.2]

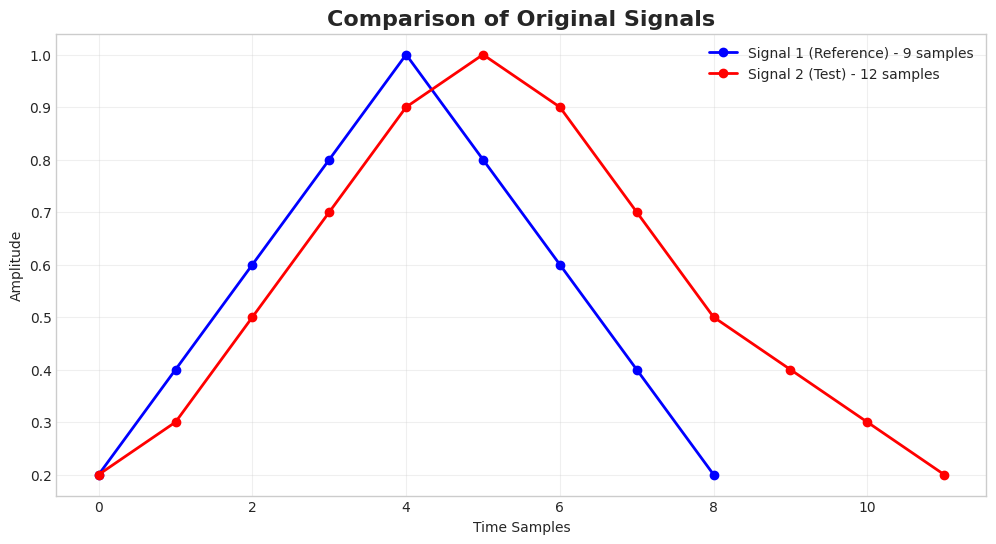
**Tasks / Questions:**

1. Plot both speech signals to observe their differences in length and amplitude patterns.
2. Perform Linear Time Normalization on Signal 2 to match the length of Signal 1.
3. Compute the alignment between Signal 1 and the normalized Signal 2.
4. Plot the alignment path, showing how each sample in Signal 1 corresponds to a sample in Signal 2.
5. Write an inference on how Linear Time Normalization aligns the two speech signals.

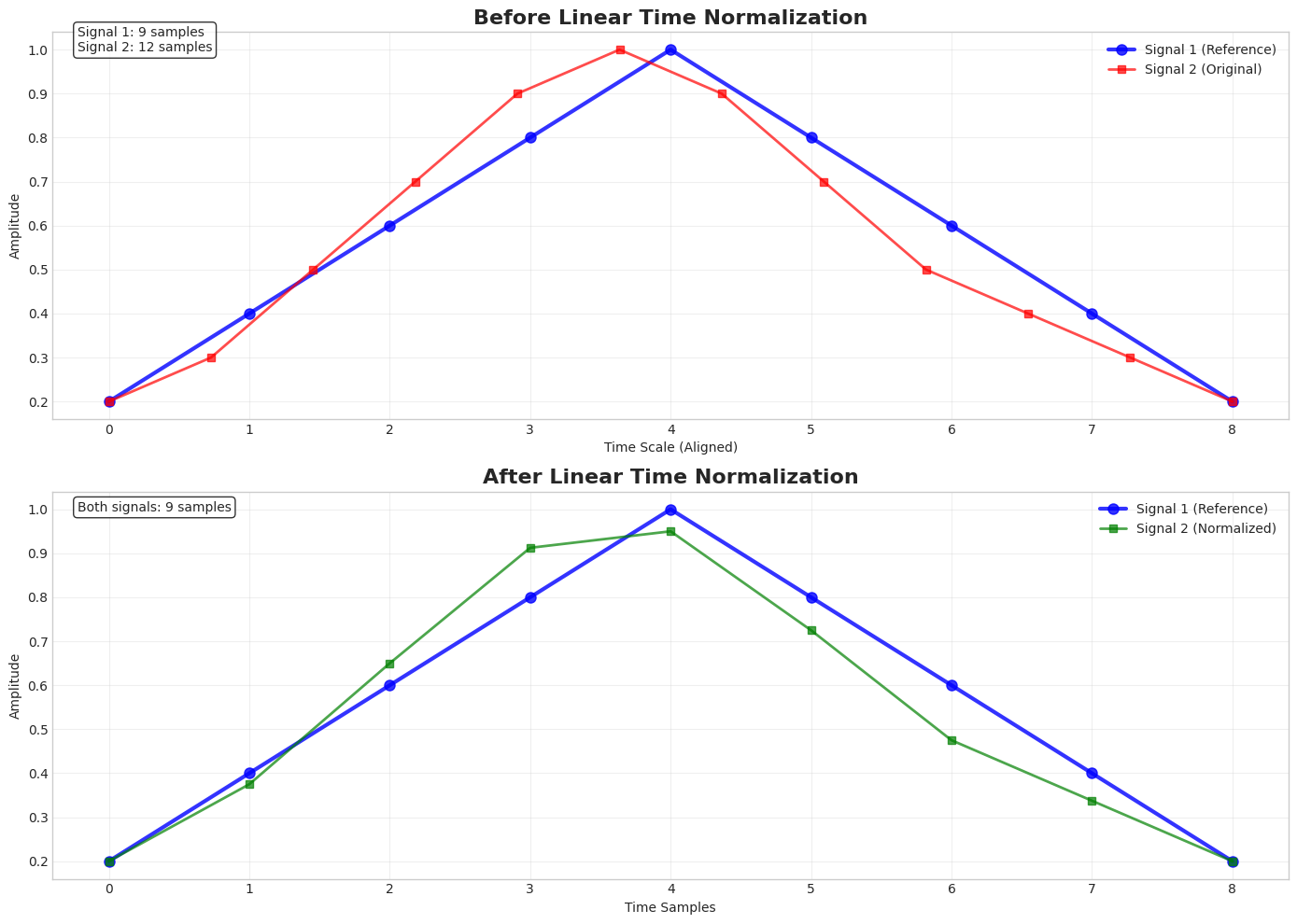
**Expected Output:**

* Two initial plots showing that Signal 2 is longer (slower speech) than Signal 1.

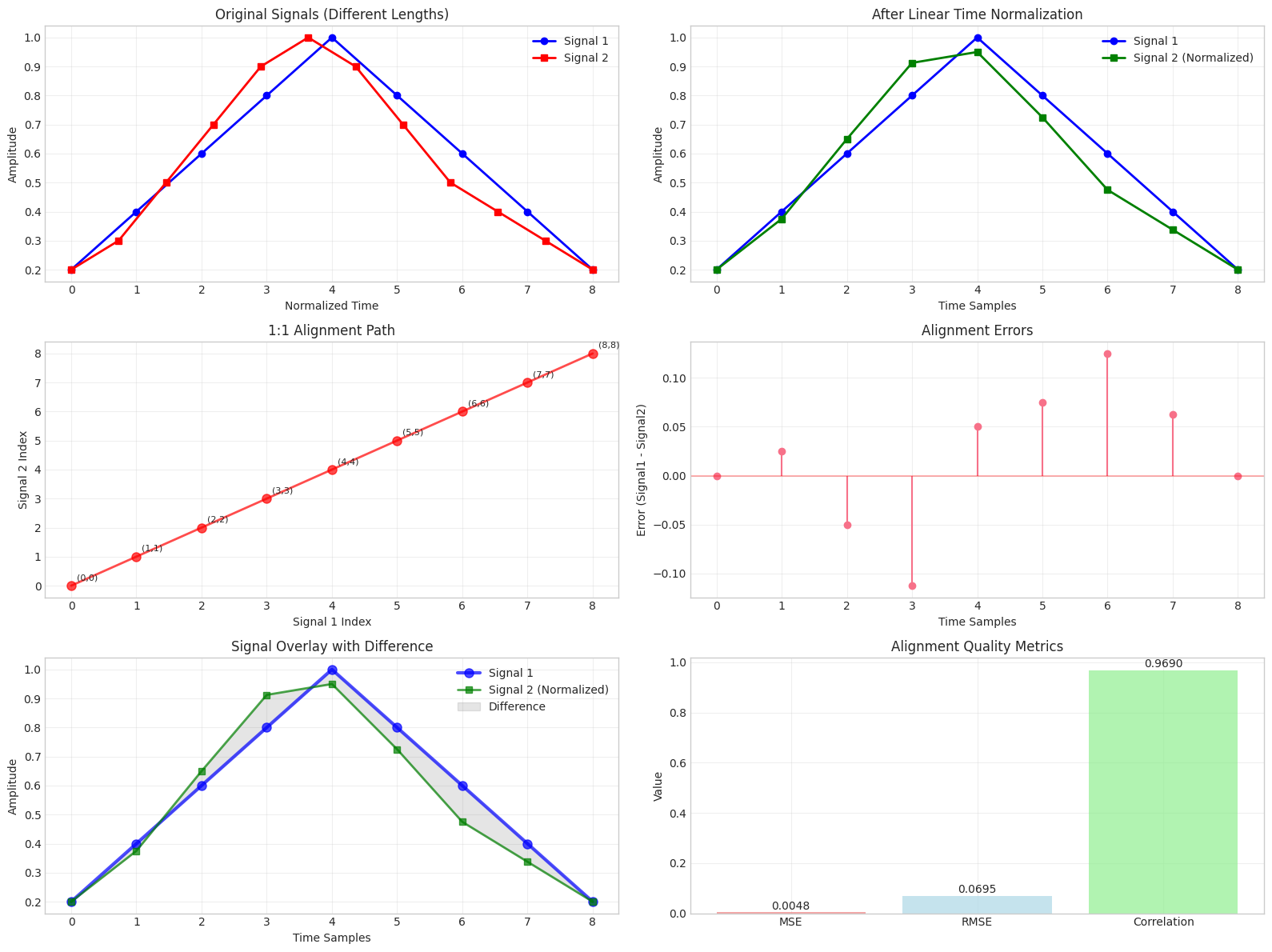




* A normalized version of Signal 2 that has the same number of samples as Signal 1 (resampled or interpolated).



* An alignment plot showing near-linear correspondence between both signals after normalization.



**Inference:** Linear Time Normalization adjusts the time axis of the slower signal so that both signals have equal lengths, enabling comparison and alignment. This method ensures that similar parts of the speech waveform align in time, despite differences in speaking speed.

**1. INITIAL OBSERVATION:**

**• Signal 1 (Reference): 9 samples - represents faster speech**

**• Signal 2 (Test): 12 samples - represents slower speech**

**• Both signals show similar amplitude patterns but different durations**

**2. LINEAR TIME NORMALIZATION PROCESS:**

**• Applied linear interpolation to resample Signal 2 from 12 to 9 samples**

**• Time axis was normalized to [0,1] range for both signals**

**• New samples were calculated using linear interpolation between existing points**

**3. ALIGNMENT RESULTS:**

**• After normalization, both signals have identical lengths (9 samples)**

**• The alignment path shows perfect 1:1 correspondence between samples**

**• Similar phonetic features now align temporally despite original speed differences**

**Lab Exercise VI – Dynamic Time Warping (DTW)**

**Aim:** To compare and align two numerical sequences using **Dynamic Time Warping (DTW)** and evaluate their similarity based on the DTW distance.

**Given Data:**

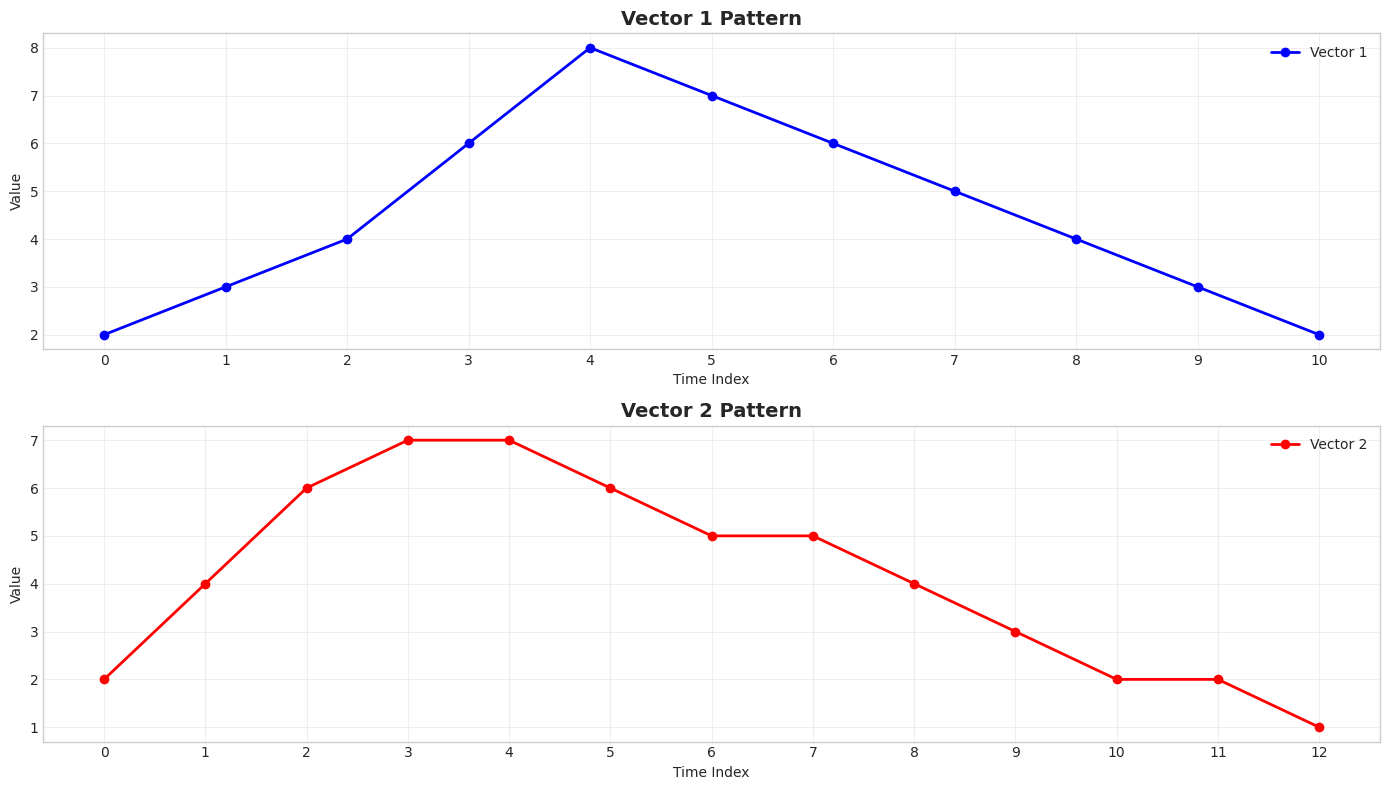
* **Vector 1:** [2, 3, 4, 6, 8, 7, 6, 5, 4, 3, 2]
* **Vector 2:** [2, 4, 6, 7, 7, 6, 5, 5, 4, 3, 2, 2, 1]

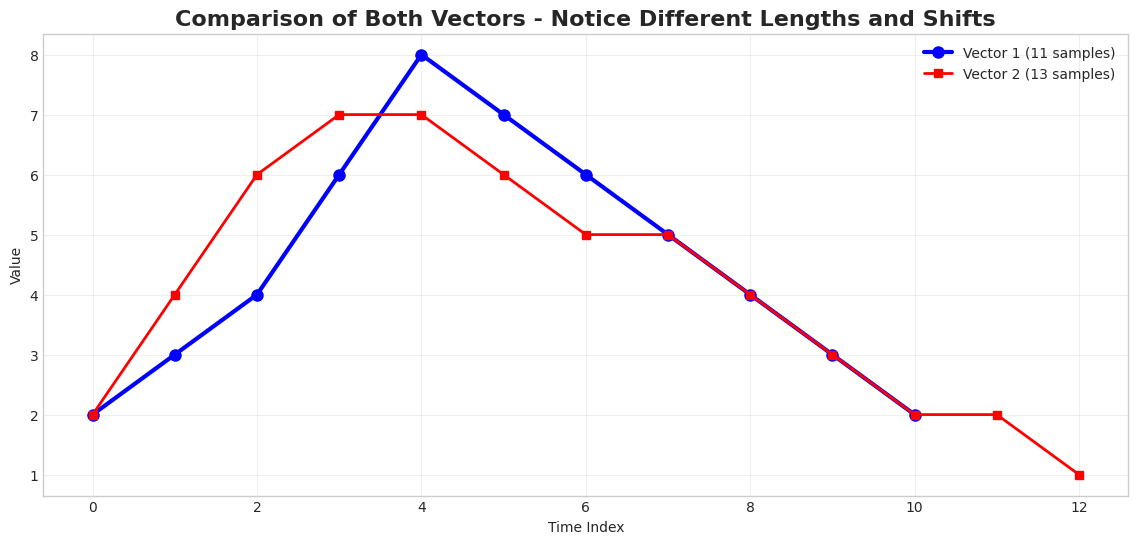
**Tasks / Questions:**

1. Plot both vectors to visualize their patterns.
2. Implement the Dynamic Time Warping algorithm.
3. Compute the accumulated cost matrix.
4. Find and visualize the optimal warping path.
5. Calculate the DTW distance between the vectors.
6. Write an inference explaining how the warping path aligns the two vectors and what the DTW distance reveals about their similarity.

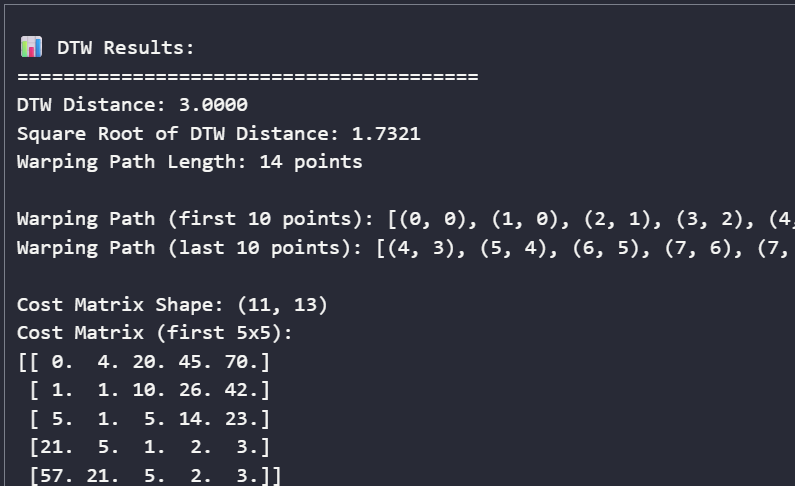
**Expected Output:**

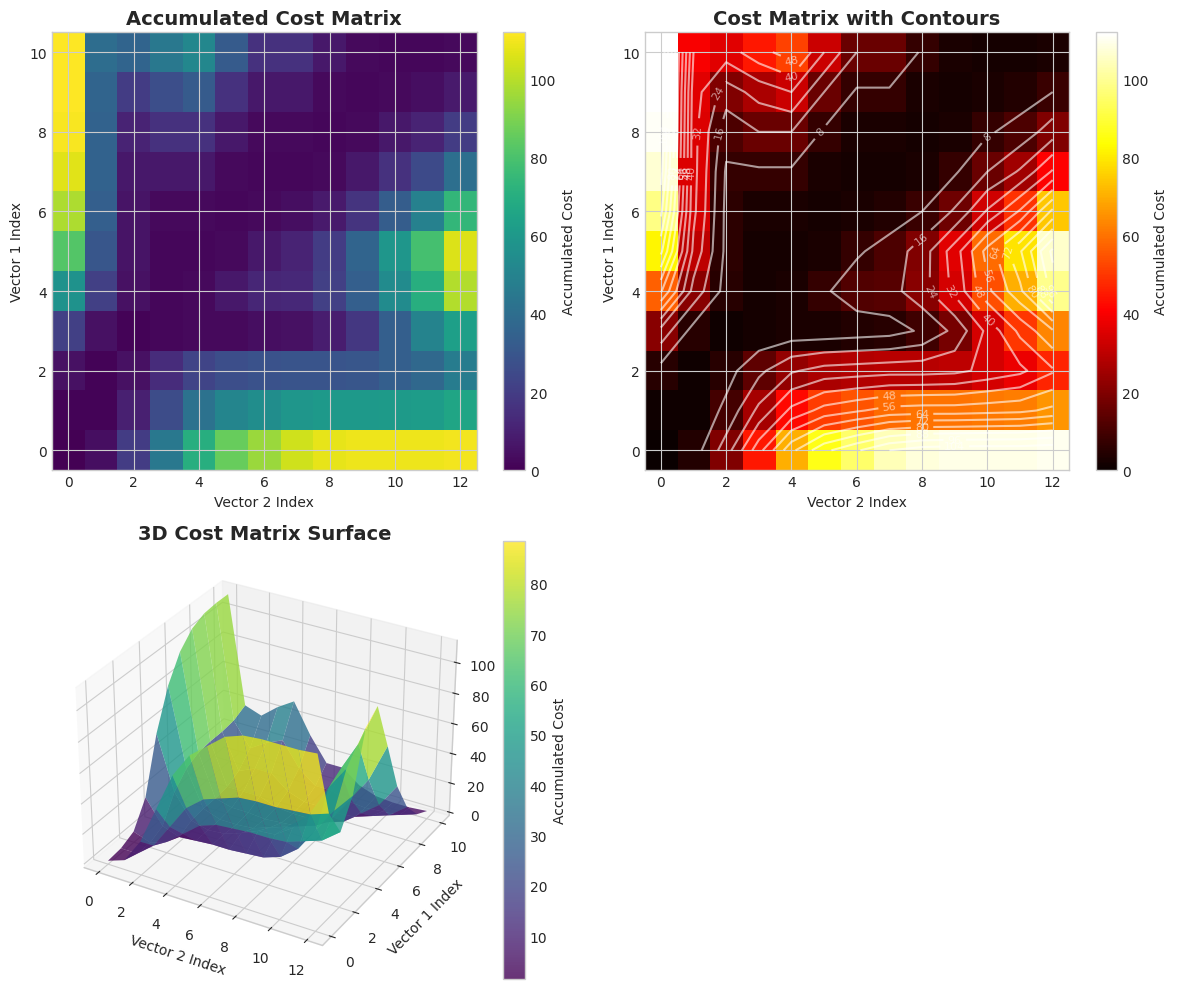
* Two plots showing that Vector 2 is stretched and slightly shifted relative to Vector 1.



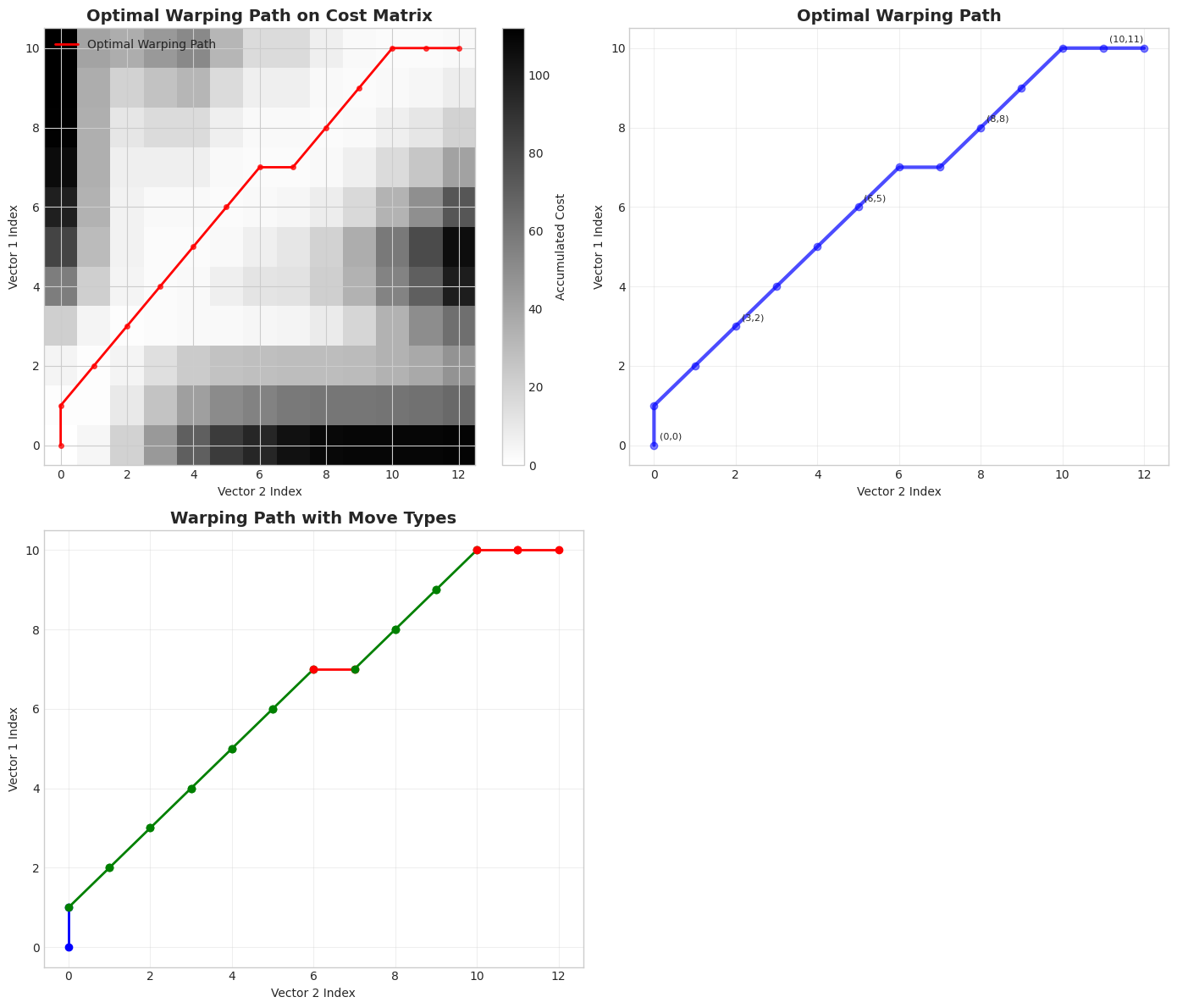


* A computed cost matrix illustrating cumulative alignment costs between samples.

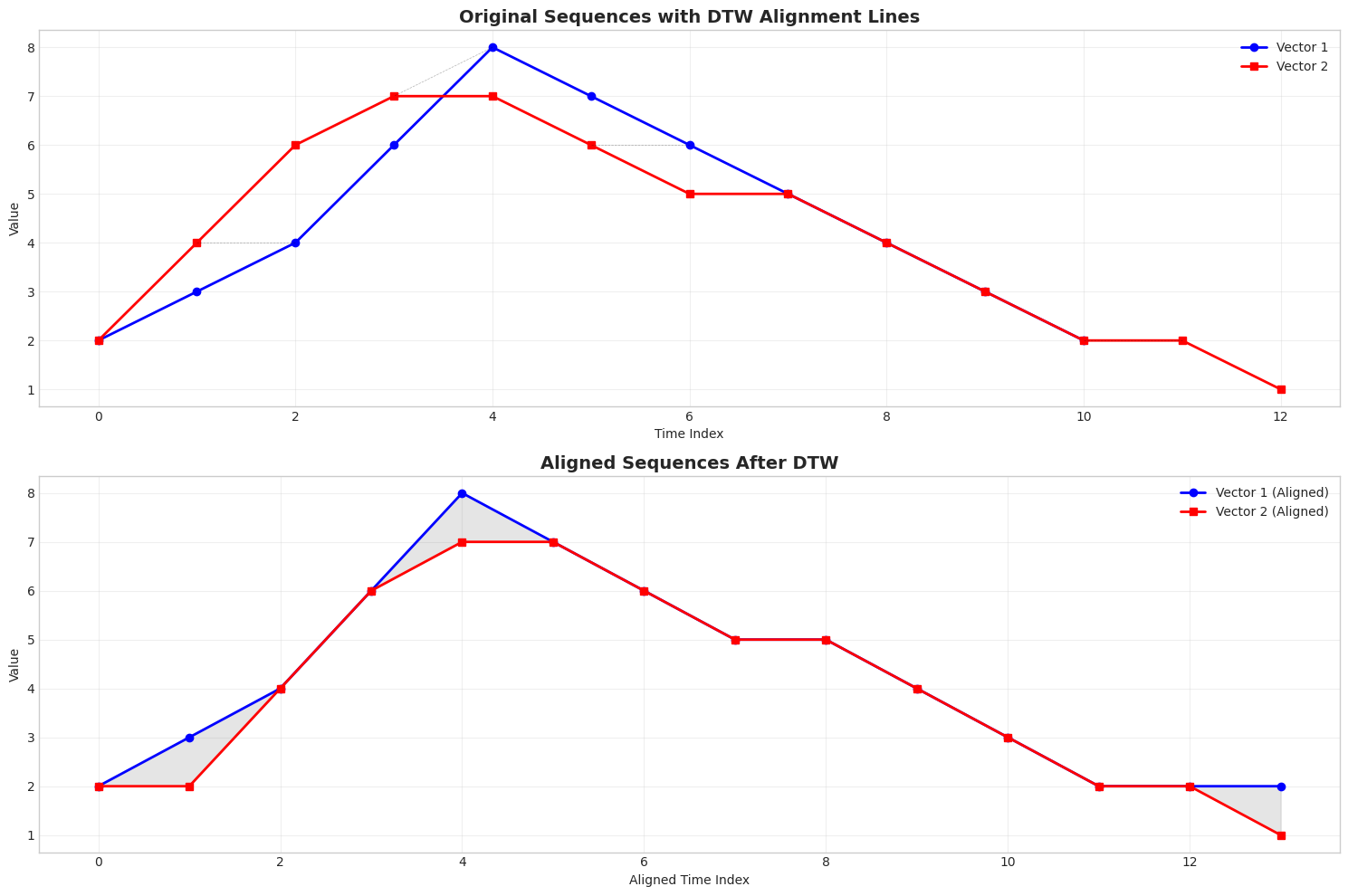


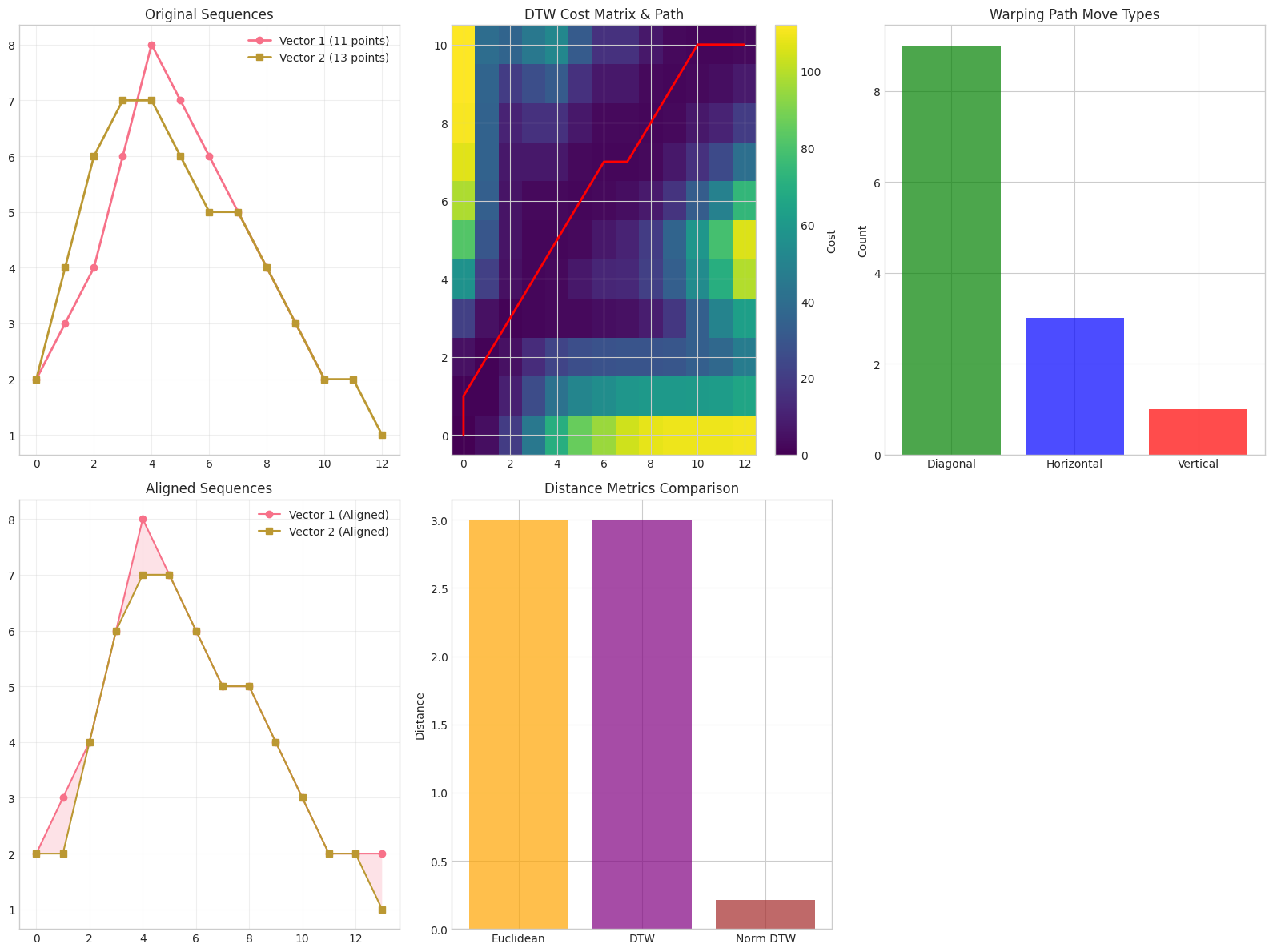


* A plotted **warping path** showing non-linear mapping between the two sequences (allowing compression and stretching along the time axis).



* A **DTW distance value**, indicating how similar the two vectors are (a smaller DTW distance means greater similarity).





* **Inference:** DTW effectively aligns the sequences even when they are temporally distorted or stretched. The warping path shows how one vector must be time-warped to match the other, and the DTW distance quantifies this similarity.
* **Evaluation Rubrics**

1. Implementation: 5 marks
2. Complexity and Validation: 3 marks
3. Documentation & Writing the inference: 2 marks

**Lab Question VII: Discrete time wrapping algorithm**

You are given two short audio recordings.  
 Audio 1 is your own voice saying the word “hello”.  
 Audio 2 is your friend’s voice saying the word “hello”.

Your task is to compare these two signals using Dynamic Time Warping.

**Tasks for the lab**

1. Record your own voice saying “hello” and store it as Signal 1.
2. Record your friend saying “hello” and store it as Signal 2.
3. Convert both audio files into numerical time series by extracting their waveform data.
4. Normalize both signals so they have comparable amplitude ranges.
5. Apply Dynamic Time Warping on the two signals.
6. Produce the alignment path and compute the total DTW distance.
7. Interpret the results. Explain whether the signals are similar, and describe how DTW helps match two audio patterns that do not align perfectly in time.