Determining Time Shift Between Two Signals Using Cross-Correlation

Shubhankar Mahanta

28-03-2025

Abstract

Analyzing the shift between two signals with similar wave patterns presents challenges in accurately determining time delays. Traditional methods, such as visually dragging one wave over another for overlap inspection, are inefficient and prone to human error. These approaches fail to provide precise alignment, especially in cases where damping, noise, or phase shifts are involved. A more reliable alternative is mathematical cross-correlation, which quantifies the time shift by computing the similarity between original and shifted waveforms, offering a robust and automated solution to signal alignment issues.

Steps to Find the Shift

To determine the time shift between an original wave and a damped, shifted wave, follow these steps:

1. Compute the cross-correlation of the two signals.
2. Identify the lag at which the cross-correlation is maximum.
3. The corresponding lag value indicates the time shift of the second wave relative to the first.

Mathematical Approach

Let:

- $x(t)$ represent the original wave.
- $y(t)$ represent the damped and shifted wave.

The cross-correlation function is defined as:

$$R_{xy}(\tau) = \int_{-\infty}^{\infty} x(t) \cdot y(t+\tau)\, dt$$

The peak of $R_{xy}(\tau)$ occurs at $\tau = \tau_{\max}$, which provides the best alignment between $x(t)$ and $y(t)$, thereby revealing the time shift.

Python Implementation

The following Python code demonstrates how to compute the time shift between two signals using the cross-correlation method:

Listing 1: Compute time shift using cross-correlation

```
1
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5
6
7  # Parameters
8  sampling_rate = 1000  # Hz
9  duration = 2.0        # seconds
10 frequency = 5.0       # Hz
11 true_shift = 0.5      # seconds
12 damping_factor = 0.1  # Damping coefficient
13
14 # Time vector
15 t = np.arange(0, duration, 1/sampling_rate)
16
17
18 # Original wave: Sine wave
19 original_wave = np.sin(2 * np.pi * frequency * t)
20
21 # Damped and shifted wave
22 damped_wave = np.exp(-damping_factor * t) * np.sin(2 * np.pi *
       frequency * (t - true_shift))
23
24
25 # Compute cross-correlation
26 correlation = np.correlate(original_wave, damped_wave, mode='
       full')
27 lags = np.arange(-len(original_wave) + 1, len(original_wave))
28
29
30
31 # Find the lag with the maximum correlation
32 max_corr_index = np.argmax(correlation)
33 time_shift = lags[max_corr_index] / sampling_rate
34
35
36 # Final Result
37 print(f"Estimated time shift: {time_shift:.4f} seconds")
38
39 #
40 #
41 #
```

Listing 2: Plotting for Visualization of Results

```
1
2
3  # Plotting
4  plt.figure(figsize=(12, 6))
5
6  # Plot original and damped waves
7  plt.subplot(2, 1, 1)
8  plt.plot(t, original_wave, label='Original Wave')
9  plt.plot(t, damped_wave, label='Damped & Shifted Wave')
10 plt.xlabel('Time (s)')
11 plt.ylabel('Amplitude')
12 plt.legend()
13 plt.title('Original and Damped Waves')
14
15
16 # Plot cross-correlation
17 plt.subplot(2, 1, 2)
18 plt.plot(lags / sampling_rate, correlation)
19 plt.axvline(time_shift, color='r', linestyle='--', label=f'
       Estimated Shift = {time_shift:.4f} s')
20 plt.xlabel('Lag (s)')
21 plt.ylabel('Cross-Correlation')
22 plt.legend()
23 plt.title('Cross-Correlation Function')
24
25 plt.tight_layout()
26 plt.show()
27
28
29
```