UNIVERSIDAD DE VALPARAÍSO
Escuela de Ingeniería Civil Biomédica
CBM414 Procesamiento digital de señales biomédicas

Laboratory 4

Maximiliano Antonio Gaete Pizarro

Contents

1 Time-Domain Comparison of Windows

The following figure shows the time-domain comparison of the Rectangular, Hamming, and Hann windows of length $N = 128$.

Python Code:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import get_window
from scipy.fft import fft, fftshift, fftfreq

# Window size
N = 128

# Generate the windows
rectangular_window = get_window('boxcar', N)
hamming_window = get_window('hamming', N)
hann_window = get_window('hann', N)

# Create the time axis
n = np.arange(N)

# Plot the windows
plt.figure(figsize=(10, 6))
plt.plot(n, rectangular_window, label='Rectangular (boxcar)', linestyle='-', marker=
    'o')
plt.plot(n, hamming_window, label='Hamming', linestyle='-', marker='x')
plt.plot(n, hann_window, label='Hann', linestyle='-', marker='d')

# Configure labels and legends
plt.title('Comparison of Windows: Rectangular, Hamming, and Hann')
plt.xlabel('Samples')
plt.ylabel('Amplitude')
plt.legend()
plt.grid(True)
plt.show()
```

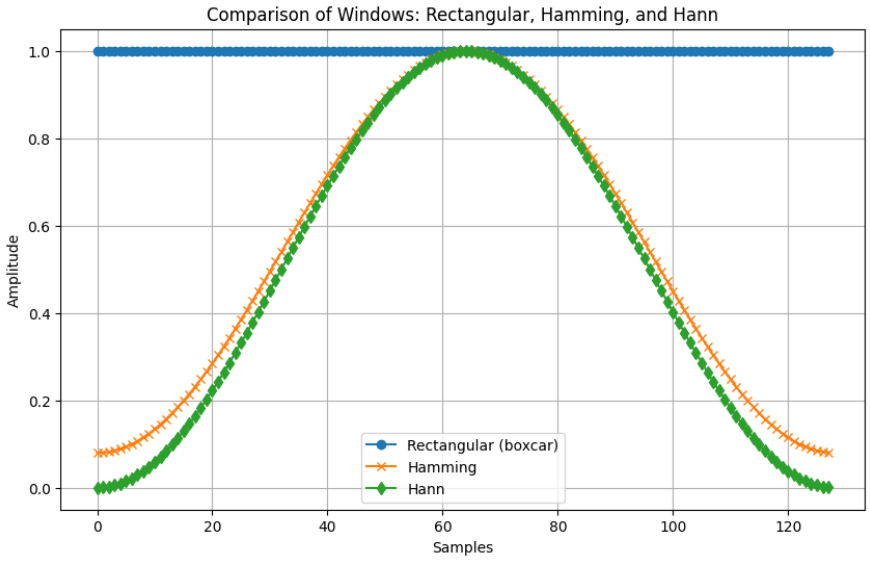

Figure 1: Comparison of Rectangular, Hamming, and Hann windows in the time domain.

Observations

In the time domain, the three windows show distinct characteristics, summarized as follows:

- **Rectangular Window:** The window maintains a constant amplitude of 1 across its entire length, which results in a sharp cut-off at the edges. This behavior leads to significant spectral leakage in the frequency domain.
- **Hamming Window:** The Hamming window tapers smoothly towards the edges, although it does not reach zero. This gradual tapering helps in reducing spectral leakage while maintaining relatively good frequency resolution.
- **Hann Window:** The Hann window also tapers smoothly, but unlike the Hamming window, it reaches an amplitude of zero at the edges. This smooth tapering to zero provides better suppression of leakage, though at the cost of wider main lobes in the frequency domain.

1.1 Frequency-Domain Comparison of Windows

The Fourier Transforms of the Rectangular, Hamming, and Hann windows are computed, and their magnitude spectra are shown below.

Python Code:

```
# Zero padding for FFT
zero_pad = 10000

# Apply FFT with zero padding and avoid log10(0) by adding a small epsilon value
epsilon = 1e-10  # Small value to avoid log(0)
rectangular_fft = 10 * np.log10(2 / zero_pad * (np.abs(fftshift(fft(rectangular_window,
    zero_pad))) + epsilon))
hamming_fft = 10 * np.log10(2 / zero_pad * (np.abs(fftshift(fft(hamming_window, zero_pad)))
    + epsilon))
hann_fft = 10 * np.log10(2 / zero_pad * (np.abs(fftshift(fft(hann_window, zero_pad))) +
    epsilon))

# Generate the corresponding frequencies
freqs = fftshift(fftfreq(zero_pad))

# Plot the spectra
plt.figure(figsize=(10, 6))
plt.plot(freqs, rectangular_fft, label='Rectangular (boxcar)')
plt.plot(freqs, hamming_fft, label='Hamming')
plt.plot(freqs, hann_fft, label='Hann')

# Configure labels and legend
plt.title('Magnitude Spectrum of Windows: Rectangular, Hamming, and Hann')
plt.xlabel('Normalized Frequency')
plt.ylabel('Magnitude (dB)')
plt.legend()
plt.grid(True)
plt.xlim([-0.05, 0.05])  # Limit the X-axis to better view the main lobe
plt.show()
```

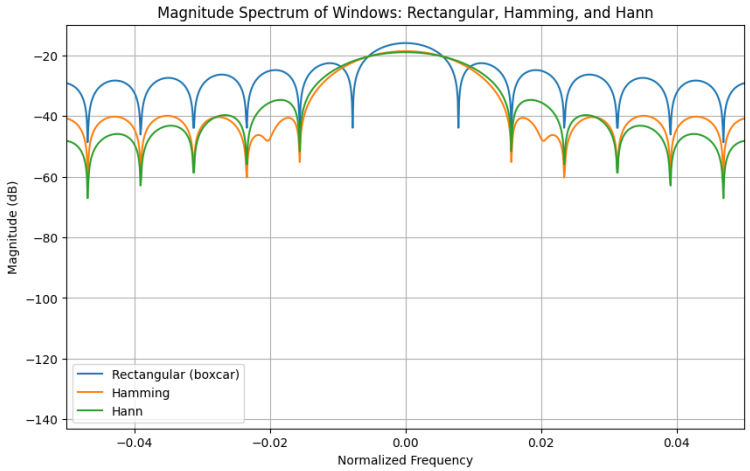

Figure 2: Magnitude spectrum of Rectangular, Hamming, and Hann windows.

Observations

The analysis of the magnitude spectrum of the Rectangular, Hamming, and Hann windows leads to the following key observations:

- **Rectangular Window:** This window exhibits the narrowest main lobe, which provides superior frequency resolution. However, it also presents the highest side lobes, causing significant spectral leakage. This makes it less suitable when side-lobe suppression is important.
- **Hamming Window:** The main lobe of the Hamming window is wider compared to the Rectangular window, which results in slightly lower frequency resolution. Nevertheless, the side lobes are considerably smaller, leading to reduced spectral leakage.
- **Hann Window:** Among the three, the Hann window has the widest main lobe, thus offering the lowest frequency resolution. However, its side lobes are the smallest, providing the best suppression of spectral leakage. This makes it particularly effective in applications where leakage minimization is prioritized.

1.2 Discussion of Time-Domain and Frequency-Domain Representation

The time-domain shape of a window has a direct influence on its behavior in the frequency domain:

- A **sharp cutoff** in the time domain, as seen with the Rectangular window, results in a narrow main lobe but produces large side lobes in the frequency domain. This translates into higher spectral leakage.
- **Smoother transitions**, as observed with the Hamming and Hann windows, lead to significantly better suppression of the side lobes in the frequency domain. However, this improvement in leakage suppression comes at the cost of a wider main lobe, which reduces frequency resolution.

Thus, there is a clear trade-off between leakage suppression and frequency resolution. The Hann window provides the best side-lobe suppression, while the Rectangular window excels in frequency resolution. The Hamming window offers a balanced compromise between these two factors.

1.3 Trade-offs in Window Selection

When comparing the windows, the trade-off between frequency resolution and spectral leakage becomes evident:

- The **Rectangular window** has the narrowest main lobe, giving it the best frequency resolution, but it also suffers from the worst spectral leakage due to high side lobes.
- The **Hamming window** offers a moderate compromise, with reduced spectral leakage at the expense of some frequency resolution. This balance makes it suitable for many practical signal processing tasks.
- The **Hann window**, with the widest main lobe, provides the best suppression of spectral leakage. However, the trade-off is that it offers the least frequency resolution. It is ideal for applications where leakage minimization is more important than precise frequency resolution.

In most practical applications, the **Hamming window** is often preferred as it strikes a good balance between frequency resolution and spectral leakage suppression, making it versatile for a variety of signal processing needs.

2 Time-Domain Signal Visualization

We start by visualizing both the resting and sleep signals in the time domain to observe their characteristics.

```
# Function to load a signal from a .txt file
def load_signal_from_txt(filename, total_time=10):
        # Load the signal from the .txt file
        signal = np.loadtxt(filename, delimiter=';')[0, :]
        fs = np.round(len(signal)) / total_time
        # Return the loaded signal
        return signal, int(fs)

reposo, fs_reposo = load_signal_from_txt('./reposo.txt')

plt.figure(figsize=(10, 6))
plt.plot(np.linspace(0, 10, 10 * fs_reposo), reposo, label=r'$\sigma␣=␣1$')
plt.title('Rest␣Signal␣in␣Time␣Domain')
plt.xlabel('Time␣(s)')
plt.ylabel('Amplitude')
plt.legend()

sueno, fs_sueno = load_signal_from_txt('./sueno.txt')

plt.figure(figsize=(10, 6))
plt.plot(np.linspace(0, 10, 10 * fs_sueno), sueno, label=r'$\sigma␣=␣1$')
plt.title('Sleep␣Signal␣in␣Time␣Domain')
plt.xlabel('Time␣(s)')
plt.ylabel('Amplitude')
plt.legend()
plt.show()
```

In the time domain, we can observe the amplitude and variability of the cardiac signals. Below are two figures representing the signals during rest and sleep.

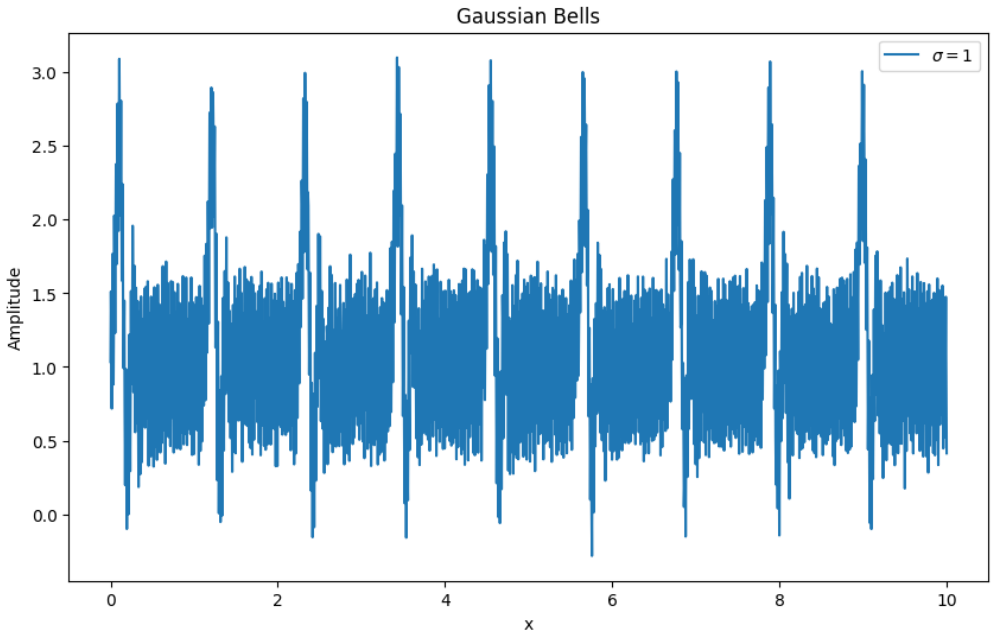

Figure 3: Cardiac signal during rest (Time domain).

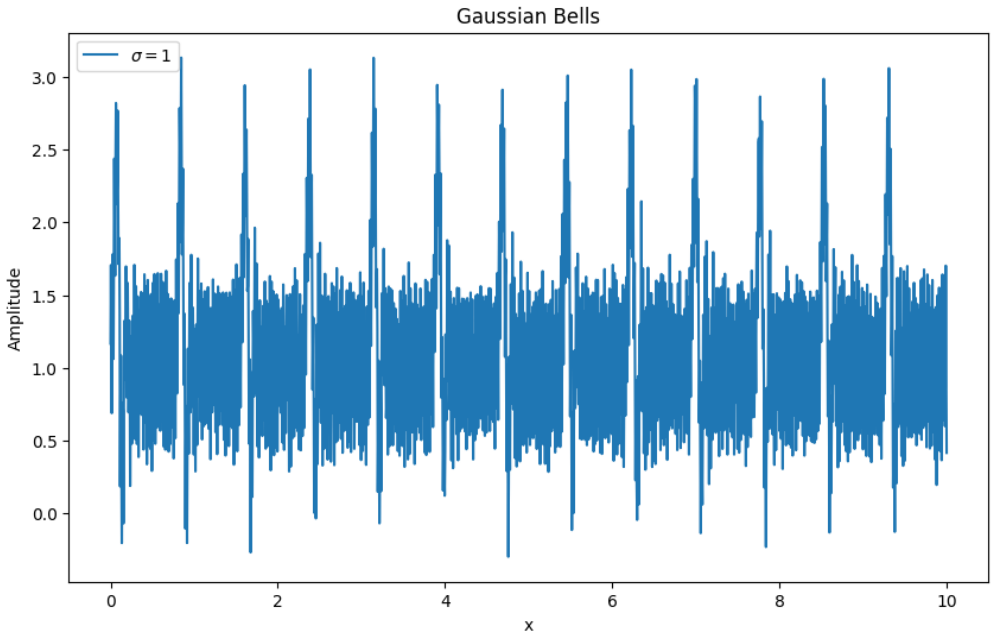

Figure 4: Cardiac signal during sleep (Time domain).

From the figures, we can observe the following:

- **Amplitude and variability:** The rest signal appears to have more regular amplitudes, while the sleep signal exhibits more irregularities in amplitude, which may correspond to different stages of sleep.
- **Peak frequency:** Peaks in the rest signal seem more evenly spaced compared to the sleep signal, indicating a more regular heartbeat during rest.

While time-domain analysis provides valuable information regarding amplitude and rhythm, it is insufficient for understanding the frequency characteristics of the cardiac signal. To overcome this limitation, frequency-domain analysis is necessary.

2.1 Limitations of Time-Domain Analysis

The time-domain analysis is limited in its ability to decompose the signal into its constituent frequencies. Cardiac signals are complex, comprising multiple frequency components. These components can change dynamically over time, especially during different sleep phases. The time-domain approach does not allow for the clear identification of these dominant frequencies.

For example, while the rest state may be dominated by lower frequencies corresponding to a steady heart rhythm, during sleep, certain stages (such as REM sleep) may introduce higher frequency components. These frequency variations are not easily visible in the time-domain representation.

2.1.1 Frequency-Domain Analysis: The Spectrogram

To fully understand the cardiac signal and its frequency components, we employ the spectrogram, which represents how the frequencies in the signal evolve over time.

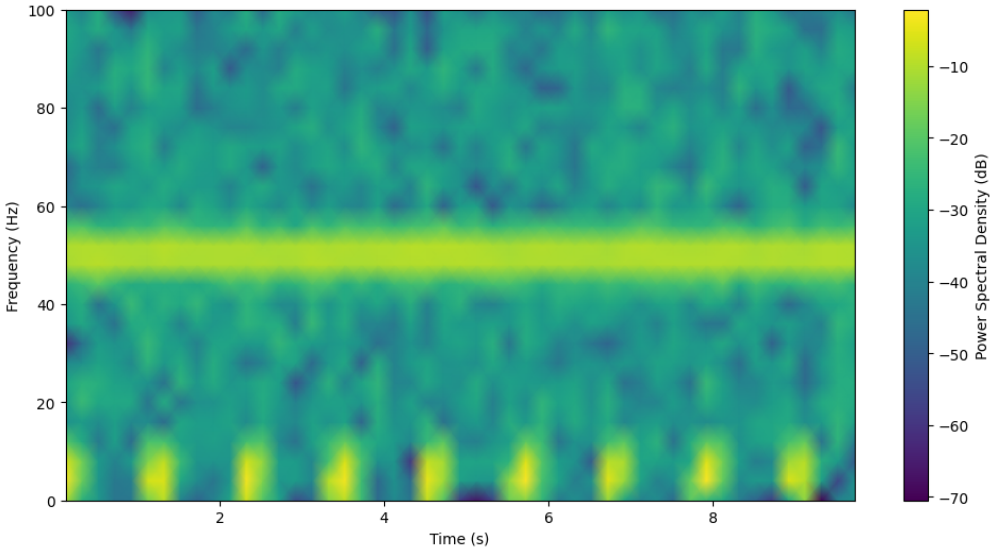

Figure 5: Spectrogram of the cardiac signal during rest.

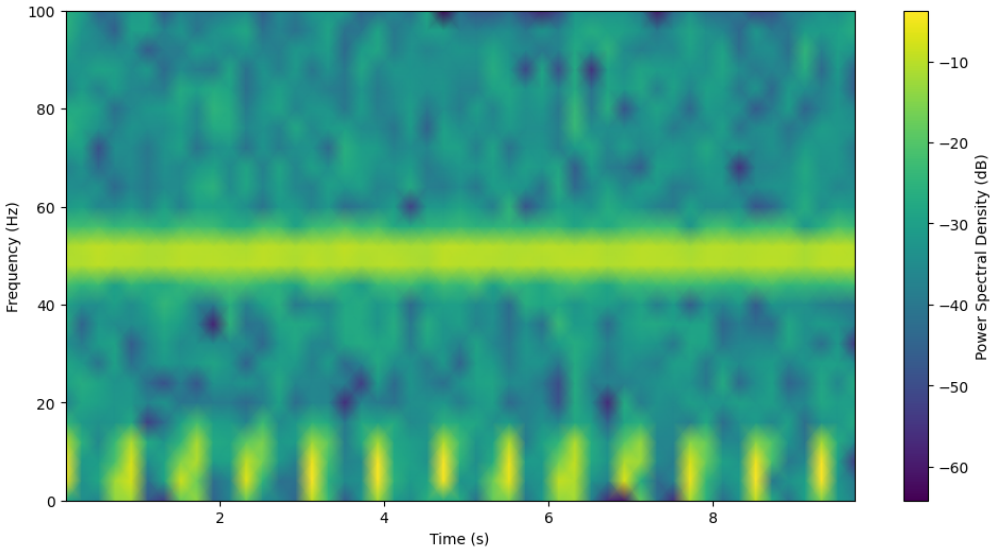

Figure 6: Spectrogram of the cardiac signal during sleep.

The spectrogram provides the following advantages:

- **Frequency evolution over time:** The spectrogram shows how the dominant frequencies change over time, a feature that is not accessible through time-domain analysis.
- **Power spectral density:** It displays the power of each frequency component, which is crucial for distinguishing relevant features of the signal from noise.
- **Identification of transitions:** The spectrogram allows for the identification of transitions between different phases, such as REM sleep, where the frequency content may change significantly.

2.1.2 Low-Pass Filtering

By observing the spectrogram, it becomes clear that a low-pass filter can be applied to isolate the lower frequencies associated with heartbeats. This filtering process can help in eliminating high-frequency noise or components that are not related to the heart's activity.

- **Noise reduction:** By applying a low-pass filter, we can eliminate unwanted high-frequency noise, allowing for a clearer analysis of the heartbeats.
- **Heartbeat isolation:** The low-pass filter helps isolate the peaks and variability corresponding to heartbeats, making it easier to identify and analyze features such as heart rate variability (HRV).
- **Sleep phase analysis:** During different sleep phases, heart rhythms may change, but heartbeats remain the dominant feature in the lower frequency range. Applying a low-pass filter highlights these patterns more clearly and eliminates artifacts that could distort the signal interpretation.

2.1.3 Conclusion

In conclusion, while time-domain analysis gives a general overview of the signal's amplitude and rhythm, it is insufficient for understanding the signal's frequency characteristics. The spectrogram provides a more complete picture by revealing how the frequency content changes over time. Additionally, applying a low-pass filter further enhances the analysis by isolating the frequencies corresponding to heartbeats and reducing noise.

2.2 Spectrograms with Different Window Types

Now we compute and visualize the spectrograms using three window types: Rectangular, Hamming, and Hann. We use a window size of 256 samples and a 50% overlap.

```
from scipy import signal

# Define function to calculate and display the spectrogram
def plot_spectrogram(sig, fs, window_type='hann', window_size=256, overlapping=0.5, title=''
    ):
    f, t, Sxx = signal.spectrogram(sig, fs, window=window_type, nperseg=window_size,
                                   noverlap=int(window_size * overlapping), scaling='
                                       spectrum')
    plt.figure(figsize=(12, 6))
    plt.pcolormesh(t, f, 10 * np.log10(Sxx), shading='gouraud')
    plt.colorbar(label='Power␣Spectral␣Density␣(dB)')
    plt.ylabel('Frequency␣(Hz)')
    plt.xlabel('Time␣(s)')
    plt.title(title)
    plt.show()

# Common parameters
window_size = 256
overlapping = 0.5

# List of windows
windows = ['boxcar', 'hamming', 'hann']
window_names = ['Rectangular', 'Hamming', 'Hann']

# Signal at rest
for window, name in zip(windows, window_names):
    plot_spectrogram(reposo, fs_reposo, window_type=window, window_size=window_size,
                     overlapping=overlapping, title=f'Spectrogram␣(Rest)␣-␣Window␣{name}')

# Signal during sleep
for window, name in zip(windows, window_names):
    plot_spectrogram(sueno, fs_sueno, window_type=window, window_size=window_size,
                     overlapping=overlapping, title=f'Spectrogram␣(Sleep)␣-␣Window␣{name}')
```

Rest Signal Spectrograms

2.2.1 Rectangular Window

The rectangular window is the simplest, with no weighting applied. As a result, it suffers from aliasing and spectral leakage, meaning it is less effective at separating closely spaced frequency components. In the spectrogram, this is reflected in less defined frequency lines and more "noise" surrounding the actual frequencies.

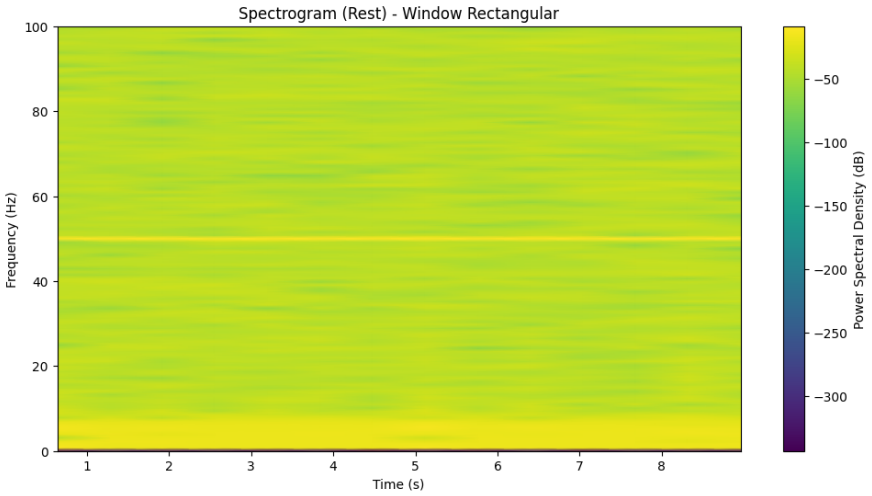

Figure 7: Spectrogram (Rest) using Rectangular Window

2.2.2 Hamming Window

The Hamming window smooths the edges, reducing spectral leakage. This means that the frequency components in the spectrogram are more defined compared to the rectangular window but at the cost of lower time resolution. The energy of the frequency components is more concentrated, making individual frequencies easier to identify.

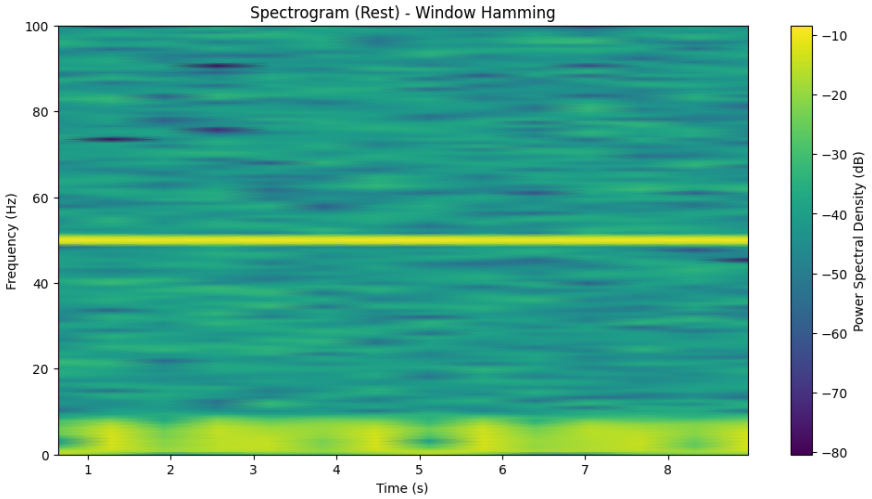

Figure 8: Spectrogram (Rest) using Hamming Window

2.2.3 Hann Window

The Hann window is similar to the Hamming window but with a steeper fall-off at the edges, making it ideal for periodic signals. The spectrogram shows an energy distribution that is intermediate between the rectangular and Hamming windows, with well-defined frequency components but slightly worse time resolution than the rectangular window.

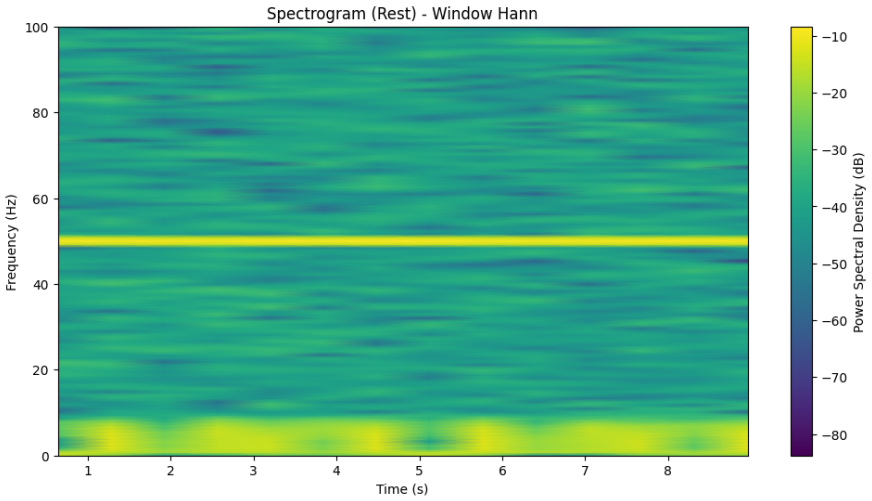

Figure 9: Spectrogram (Rest) using Hann Window

2.3 Sleep Signal Spectrograms

2.3.1 Rectangular Window

Due to high spectral leakage, the rectangular window tends to show additional frequency components that are not actually present in the signal. This occurs because the transitions between different frequency components are not smoothed, leading to artifacts.

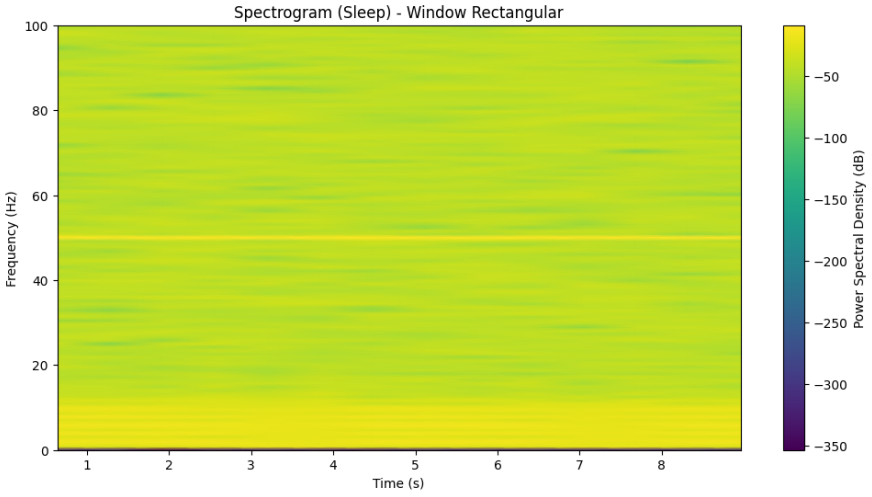

Figure 10: Spectrogram (Sleep) using Rectangular Window

2.3.2 Hamming Window

This window minimizes the appearance of spurious components, making the spectrogram more faithful to the actual frequency components of the signal. The primary frequencies will be better differentiated and less affected by artifacts.

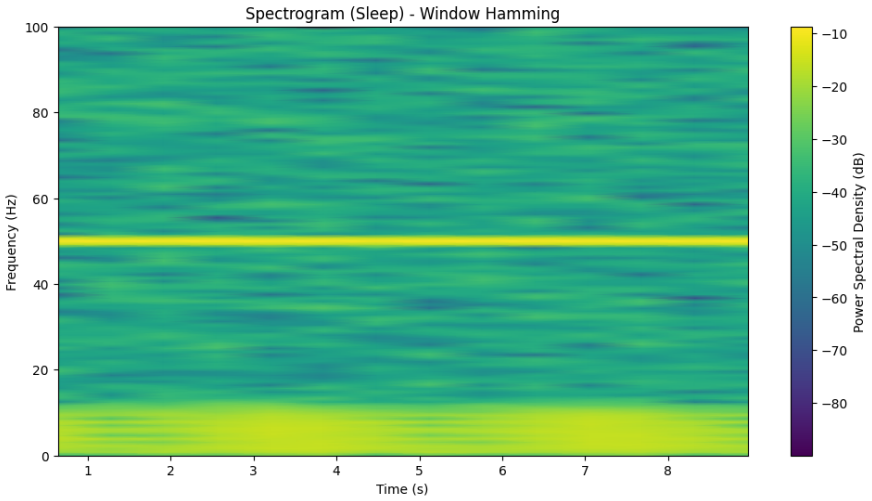

Figure 11: Spectrogram (Sleep) using Hamming Window

2.3.3 Hann Window

Like the Hamming window, the Hann window suppresses spurious components, but its faster fall-off at the edges means that frequencies very close together may appear more overlapped compared to the Hamming window.

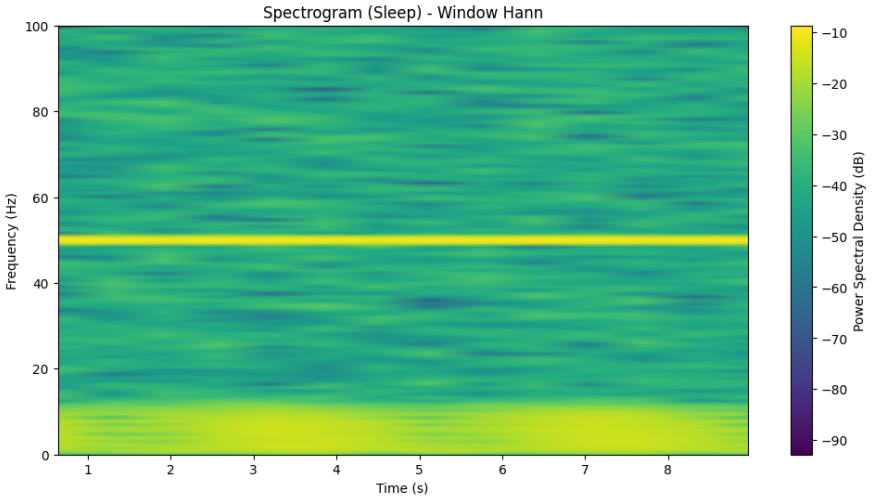

Figure 12: Spectrogram (Sleep) using Hann Window

2.4 Energy Distribution

2.4.1 Rectangular Window

The energy distribution is more dispersed with the rectangular window. Spectral leakage is more noticeable, leading to a less accurate representation of the frequency components. This can make adjacent frequencies appear more diffuse.

2.4.2 Hamming Window

With this window, the energy is more concentrated around the main frequencies. In the spectrogram, this results in more defined frequency components and less noise between them.

2.4.3 Hann Window

The Hann window provides a concentration of energy that is slightly less than the Hamming window but better than the rectangular window. The frequency components in the spectrogram are clear, though the additional smoothness of the window may cause some detail to be lost in rapid frequency transitions.

Conclusion

Each window affects both the time and frequency resolution of the spectrograms. The Rectangular window offers the best time resolution but poor frequency resolution, introducing more artifacts. On the other hand, the Hamming and Hann windows provide better frequency resolution, which allows for clearer spectral components at the expense of time resolution. For signals with well-defined frequency components where spectral noise is a concern, Hamming or Hann windows are preferable, while the rectangular window may be chosen when time resolution is more critical.

3 ffect of Different Overlaps with Hamming Window

Next, we analyze the effect of different overlap percentages: 0%, 50%, and 75%, while using the Hamming window and a window size of 256 samples.

```
# List of overlap percentages
overlaps = [0.0, 0.5, 0.75]
overlap_percentages = ['0%', '50%', '75%']

# Signal at rest with different overlaps
for overlap, percentage in zip(overlaps, overlap_percentages):
    plot_spectrogram(reposo, fs_reposo, window_type='hamming', window_size=256,
                     overlapping=overlap, title=f'Spectrogram␣(Rest)␣-␣Overlap␣{percentage}'
                         )

# Signal during sleep with different overlaps
for overlap, percentage in zip(overlaps, overlap_percentages):
    plot_spectrogram(sueno, fs_sueno, window_type='hamming', window_size=256,
                     overlapping=overlap, title=f'Spectrogram␣(Sleep)␣-␣Overlap␣{percentage}
                         ')
```

3.1 Rest Spectrogram Analysis

3.1.1 Overlap 0%

In the spectrogram with no overlap (0%), we observe that the data representation is rougher and less smooth. Transitions between different frequency bands and power levels appear less continuous, which results in a higher capacity to capture sudden changes in the signal over time. However, this increased temporal resolution comes at the cost of lower frequency resolution, meaning finer details in the frequency domain might not be as prominent.

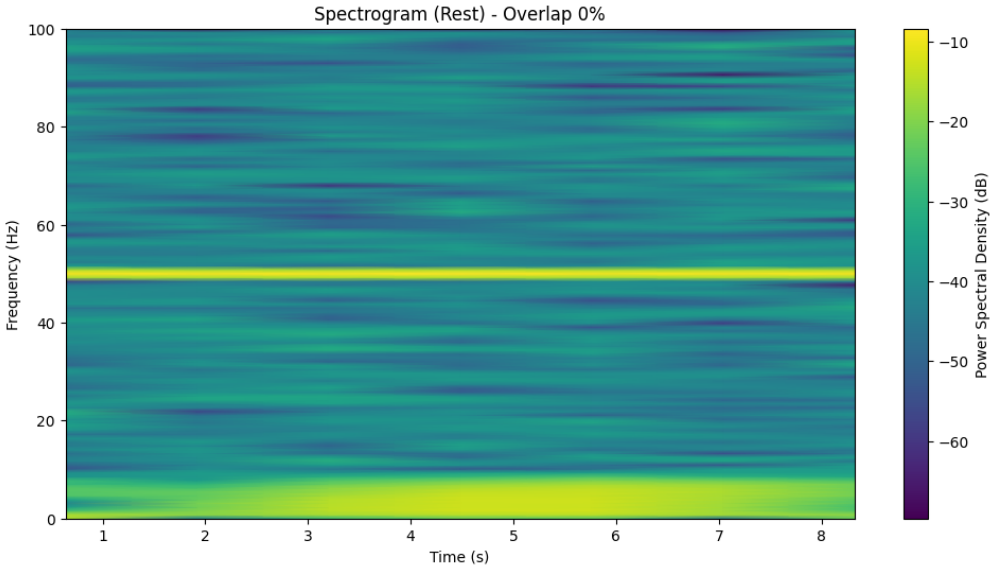

Figure 13: Spectrogram (Rest) with 0% Overlap

3.1.2 Overlap 50%

As the overlap increases to 50%, we observe a balance between temporal resolution and smoothness. Changes in the signal are captured more precisely in both domains without sacrificing too much temporal or frequency resolution. This overlap is typically seen as a good compromise, as it improves the continuity of transitions in the spectrogram without losing too much temporal information.

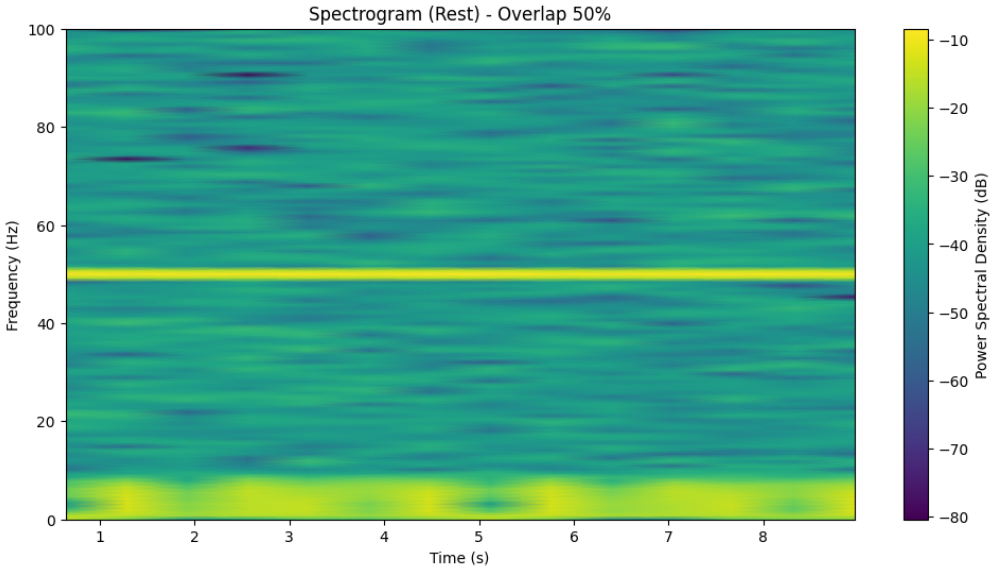

Figure 14: Spectrogram (Rest) with 50% Overlap

3.1.3 Overlap 75%

With a 75% overlap, the spectrogram appears much smoother. This results in a more continuous representation of frequency bands, making it easier to visualize more persistent or longer-duration frequency components. However, this smoothness reduces the ability to detect rapid changes in time, as temporal reso-

lution is compromised. Nevertheless, frequency resolution is improved, focusing on stationary characteristics of the signal.

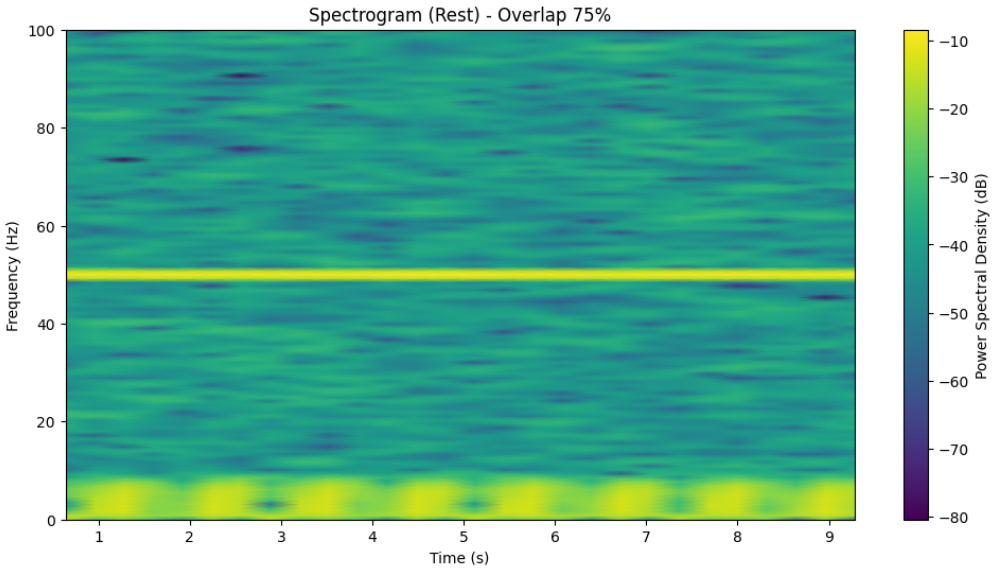

Figure 15: Spectrogram (Rest) with 75% Overlap

3.2 Sleep Spectrogram Analysis

To further understand this effect, we see the spectrogram for a sleep condition with varying overlaps.

3.2.1 Overlap 0%

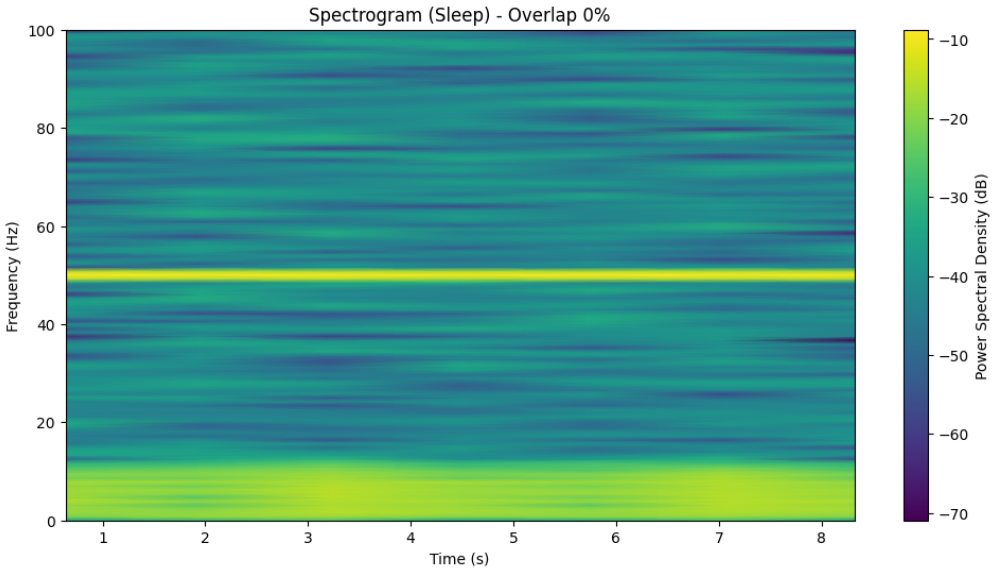

Figure 16: Spectrogram (Sleep) with 0% Overlap

3.2.2 Overlap 50%

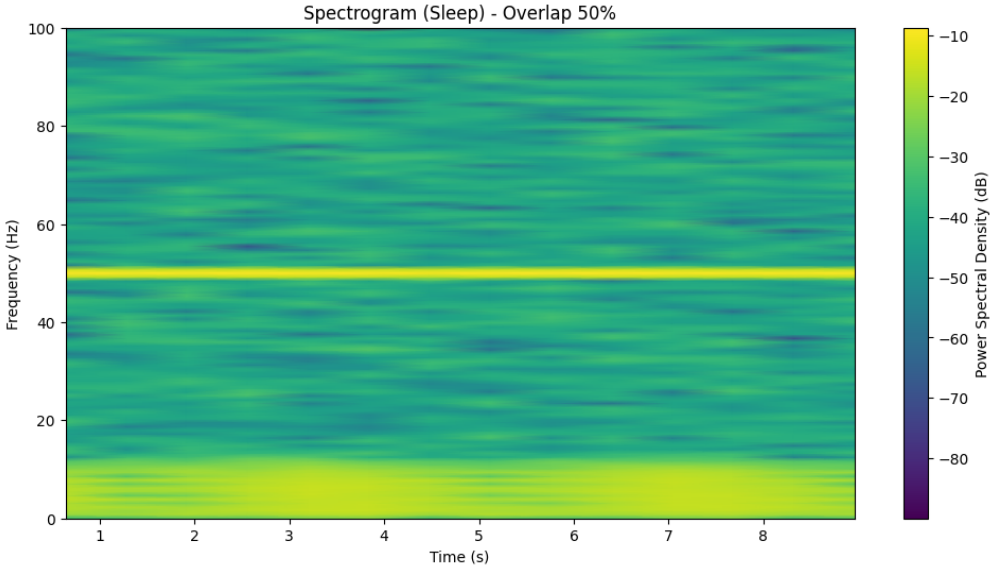

Figure 17: Spectrogram (Sleep) with 50% Overlap

3.2.3 Overlap 75%

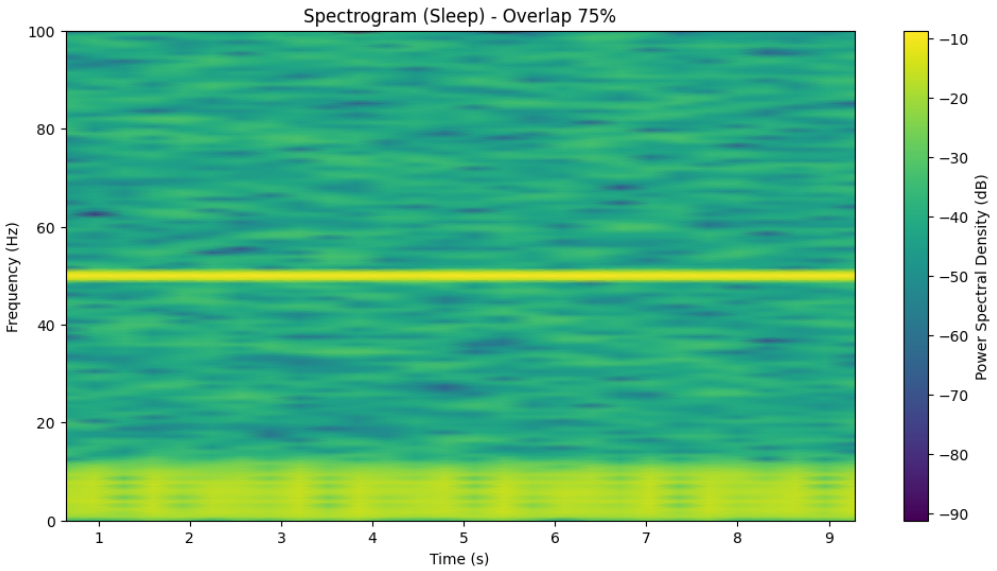

Figure 18: Spectrogram (Sleep) with 75% Overlap

3.3 Relation to the Heisenberg Uncertainty Principle

The Heisenberg Uncertainty Principle, when applied to signal analysis, states that there is a trade-off between time and frequency resolution. It is not possible to improve both simultaneously; increasing the resolution in time results in a loss of frequency precision, and vice versa.

Temporal Resolution

Temporal resolution refers to the ability to identify changes in the signal over short time intervals. A lower overlap (0%) enhances temporal resolution, allowing transient events to be captured more clearly, but sacrifices the ability to represent frequencies accurately.

Frequency Resolution

Frequency resolution represents the ability to distinguish between nearby frequencies. A higher overlap (50% or 75%) improves frequency resolution, allowing more persistent or subtle frequency components to be identified. However, this comes at the cost of detecting rapid changes in time.

The window size (256 samples in this case) plays a crucial role in this trade-off. Larger windows offer better frequency resolution at the cost of temporal resolution, and vice versa. By applying overlap, especially in combination with smaller windows, it is possible to improve frequency resolution without losing too much temporal information.

Conclusion

In summary, overlap affects the smoothness of the spectrogram and the ability to detect rapid changes in the signal as follows:

- Lower overlap (0%) prioritizes temporal resolution at the expense of a noisier or less continuous frequency representation.
- Intermediate overlap (50%) achieves a good compromise between temporal and frequency resolution.
- Higher overlap (75%) improves smoothness and frequency precision but reduces the ability to capture transient events.

This analysis directly reflects the Heisenberg Uncertainty Principle, where increasing the resolution in one domain leads to a decrease in resolution in the other. The choice of overlap percentage and window size must be carefully selected based on the goal of the analysis, whether to detect transient events or highlight persistent frequency characteristics.

4 Effect of Window Size on Spectrogram Resolution

We now compare the spectrograms using different window sizes: 128, 256, and 512 samples, with a 50% overlap.

```
# List of window sizes
window_sizes = [128, 256, 512]
window_size_labels = ['128 samples', '256 samples', '512 samples']

# Spectrograms with different window sizes
for size, label in zip(window_sizes, window_size_labels):
    plot_spectrogram(reposo, fs_reposo, window_type='hamming', window_size=size,
                     overlapping=0.5, title=f'Spectrogram (Rest) - Window Size {label}')
```

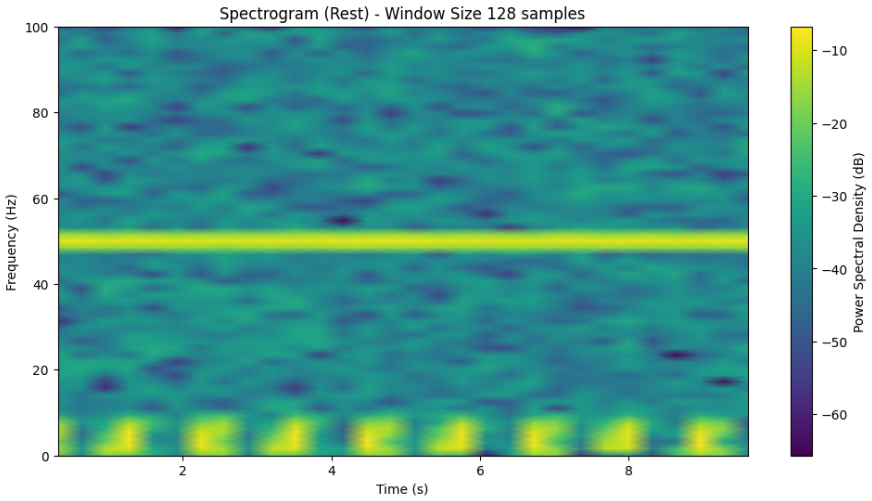

Figure 19: Spectrogram with Window Size 128 samples

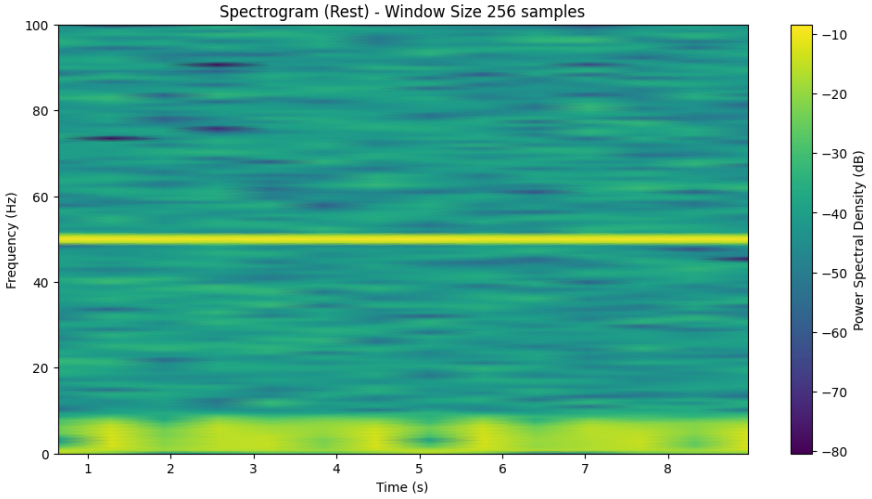

Figure 20: Spectrogram with Window Size 256 samples

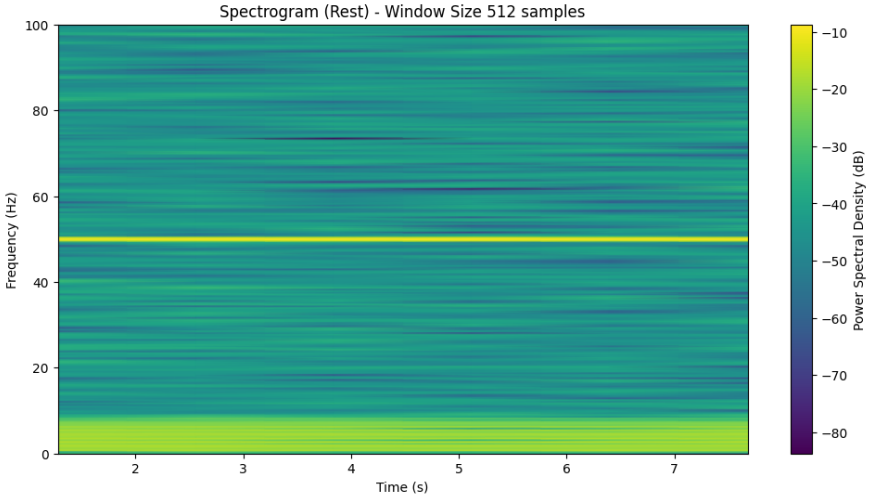

Figure 21: Spectrogram with Window Size 512 samples

4.1 Temporal Resolution

Temporal resolution refers to the ability of the spectrogram to distinguish events over time. A smaller window size leads to better temporal resolution, meaning that it is easier to detect rapid changes in the signal.

- With a window size of **128 samples**, we observe the highest temporal resolution. The variability along the time axis is more defined, and rapid changes in the signal are better captured, especially in the lower frequency ranges.
- Increasing the window size to **256 samples** reduces temporal resolution. Rapid transitions or changes in the signal become more diffuse, smoothing out variations.
- At a window size of **512 samples**, the temporal resolution is the lowest. The signal's rapid changes are significantly smoothed out and are less distinguishable.

4.2 Frequency Resolution

Frequency resolution refers to the ability of the spectrogram to distinguish between closely spaced frequencies. A larger window size leads to better frequency resolution.

- With a window size of **128 samples**, the frequency resolution is limited. The frequency bands appear wider, making it harder to distinguish between adjacent frequencies.
- As the window size increases to **256 samples**, the frequency resolution improves, and closely spaced frequencies become more distinguishable.
- With a window size of **512 samples**, the frequency resolution is the highest, as the frequency bands are more sharply defined, particularly around 60 Hz.

4.3 Ability to Detect Rapid Changes in Time

- The spectrogram with a **128-sample window** size offers the best ability to detect rapid changes in time, as the small window captures the quick transitions in the signal with high detail.
- As the window size increases to **256 and 512 samples**, the ability to detect rapid changes decreases. The signal becomes smoother over time, and fast transitions are averaged out, making them less noticeable.

Conclusion

- A smaller window size (128 samples) provides better **temporal resolution**, allowing the detection of rapid events in the signal at the cost of **frequency resolution**.
- A larger window size (512 samples) improves **frequency resolution**, enabling better distinction between closely spaced frequencies but at the expense of losing details of rapid changes over time.
- The intermediate window size (256 samples) balances both temporal and frequency resolution, but still sacrifices the ability to capture rapid temporal events.

5 Best Combination of Parameters

The analysis of cardiac signals using spectrograms requires a careful selection of parameters, such as the window type, window size, and overlap. These parameters directly influence the trade-off between temporal and frequency resolution, and thus the type of information that can be extracted from the spectrogram.

5.1 Selection of Parameters

For this analysis, the following parameters were chosen based on the optimal balance between temporal and frequency resolution:

- **Window type**: Hamming
- **Window size**: 256 samples
- **Overlap**: 75%

The Hamming window was selected due to its ability to provide a good compromise between minimizing spectral leakage and maintaining accurate frequency representation. A window size of 256 samples was chosen as it offers a good balance between capturing the variations in heart rate (temporal resolution) and resolving the frequency components of interest (frequency resolution). The 75% overlap ensures that no information is lost between successive windows and that temporal continuity is preserved.

5.2 Balancing Temporal and Frequency Resolution

The choice of these parameters directly affects the resolution of the spectrogram:

- **Temporal resolution**: A smaller window size improves temporal resolution, allowing the detection of rapid changes in the signal, such as individual heartbeats.
- **Frequency resolution**: A larger window size improves frequency resolution, allowing better identification of specific frequency components, such as heart rate variability.

With the selected window size of 256 samples, an appropriate balance is achieved, ensuring that both rapid changes and frequency components are accurately captured.

5.3 Differences between Rest and Sleep States

The cardiac signals during rest and sleep typically exhibit different characteristics. During rest, the signal is more stable, with less variability in heart rate. In contrast, during sleep, especially in REM sleep, the cardiovascular system shows higher variability, with fluctuations in heart rate being more pronounced.

5.3.1 Resting Signal

During rest, the spectrogram tends to show a clear dominant frequency corresponding to the average heart rate, with less variability over time. The selected parameters provide sufficient resolution to capture this steady behavior.

5.3.2 Sleep Signal

During sleep, the signal exhibits greater variability, particularly in the REM phase. The spectrogram shows transient increases in higher frequencies, reflecting increased sympathetic activity. The selected window size and overlap allow for the capture of these rapid changes, especially during the transition between sleep stages.

5.4 Interpretation of Spectrograms

The spectrograms obtained with the selected parameters reveal the following patterns:

- **Rest**: A stable frequency component corresponding to the average heart rate, with low variability.
- **Sleep**: Higher variability, particularly during REM sleep, with transient increases in higher frequency components.

These changes in frequency components during sleep are related to the regulation of the autonomic nervous system, with increased sympathetic activity during REM sleep leading to higher heart rate variability.

Conclusion

The chosen combination of window type (Hamming), window size (256 samples), and overlap (75%) provides an optimal balance for analyzing cardiac signals in both rest and sleep states. The parameters allow for accurate capture of both the temporal dynamics of heart rate and its frequency components, enabling a comprehensive understanding of the cardiovascular system's behavior in different physiological states.

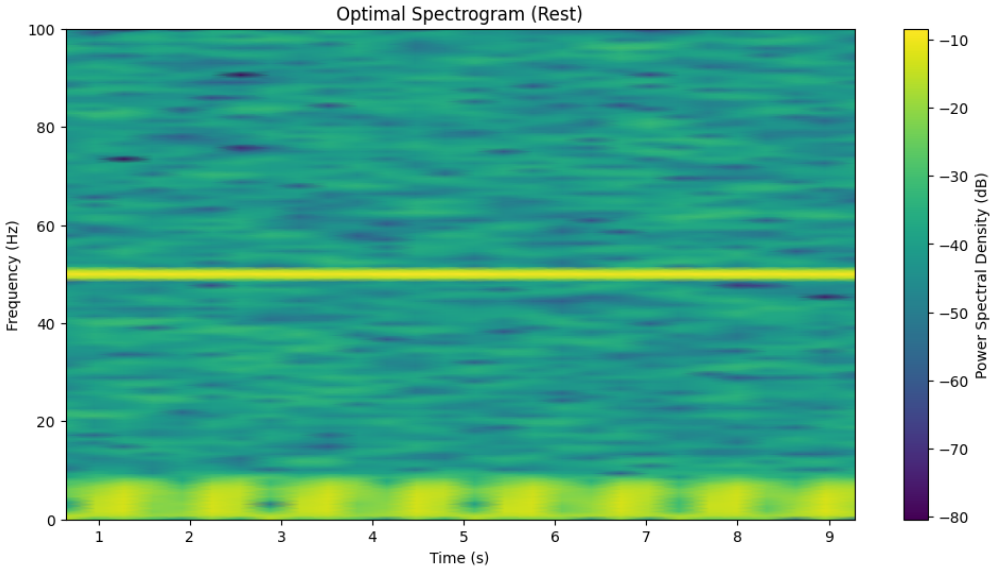

Figure 22: Spectrogram of the cardiac signal during rest.

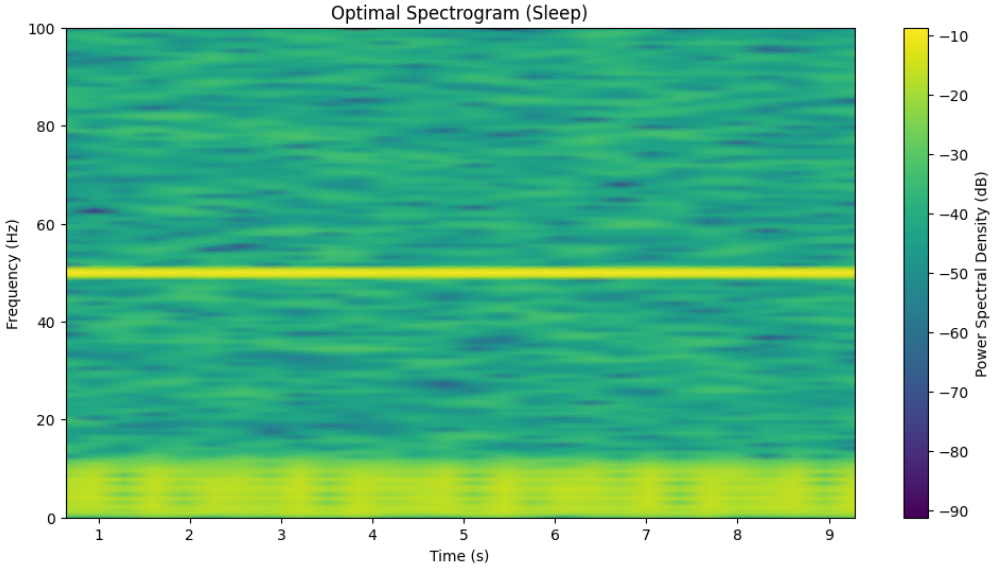

Figure 23: Spectrogram of the cardiac signal during sleep.

6 Brain Response for the hemodynamic response function (HRF)

We analyze the response of a linear and time-invariant (LTI) system using the hemodynamic response function (HRF). The system's behavior is explored through various types of inputs, including pulses and sinusoidal signals, and the properties of linearity and time invariance are verified.

The HRF is mathematically described as:

$$h(t) = \left(\frac{t}{\tau_1}\right)^{\delta_1} \exp\left[-\frac{\delta_1}{\tau_1}(t-\tau_1)\right] - c\left(\frac{t}{\tau_2}\right)^{\delta_2} \exp\left[-\frac{\delta_2}{\tau_2}(t-\tau_2)\right], \tag{0.1}$$

where the parameters $\{\tau_1, \tau_2, \delta_1, \delta_2, c\}$ control the shape of the HRF. The typical parameter values are $\tau_1 = 5.4$, $\tau_2 = 10.8$, $\delta_1 = 6$, $\delta_2 = 12$, and $c = 0.35$.

6.1 Hemodynamic Response Function (HRF) in Python

The HRF function can be implemented in Python as follows:

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

def hrf(t, tau1=5.4, delta1=6.0, tau2=10.8, delta2=12.0, c=0.35):
    h = ((t / tau1) ** delta1) * np.exp(-delta1 * (t - tau1) / tau1) - \
        c * ((t / tau2) ** delta2) * np.exp(-delta2 * (t - tau2) / tau2)
    h[t < 0] = 0  # Ensure h(t) = 0 for t < 0
    return h

TR = 0.1  # sampling interval
N_pts = 30  # number of HRF points
timeline = np.arange(0, N_pts, TR)
h = hrf(timeline)

plt.figure(figsize=(20, 5))
plt.plot(timeline, h)
plt.xlabel('seconds')
plt.ylabel('amplitude')
plt.title('Hemodynamic Response Function (HRF)')
plt.grid(True)
plt.show()
```

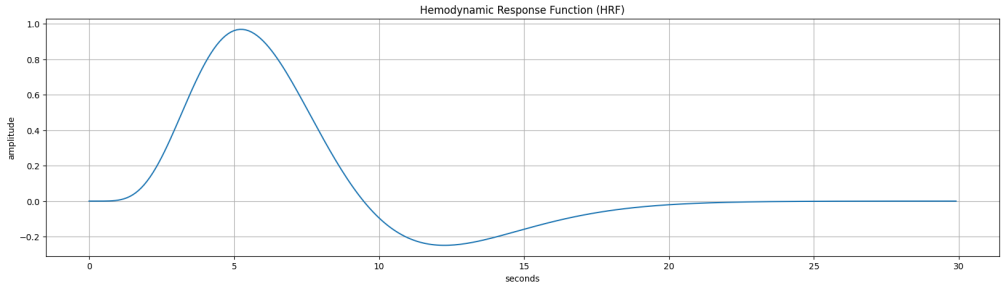

Figure 24: Hemodynamic Response Function (HRF) in Python.

6.2 Brain Response to Inputs

We calculated the brain's response to three different input signals: a 3-second pulse, a 10-second pulse, and a 0.5 Hz sine wave.

Response

The following Python code was used to generate the response.

```
e_pulse3 = np.zeros_like(timeline)
e_pulse3[(timeline >= 0) & (timeline < 3)] = 1

e_pulse10 = np.zeros_like(timeline)
e_pulse10[(timeline >= 0) & (timeline < 10)] = 1

#sinusoidal input 0.5 Hz
e_sin = np.sin(2 * np.pi * 0.5 * timeline)

# Responses to the inputs
response_pulse3 = signal.convolve(e_pulse3, h, mode='full')[:len(timeline)] * TR
response_pulse10 = signal.convolve(e_pulse10, h, mode='full')[:len(timeline)] * TR
response_sin = signal.convolve(e_sin, h, mode='full')[:len(timeline)] * TR

plt.figure(figsize=(12, 8))

plt.subplot(3, 1, 1)
plt.plot(timeline, e_pulse3, label='Input: 3s Pulse')
plt.plot(timeline, response_pulse3, label='Response')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Response to 3-second Pulse')
plt.grid(True)
plt.legend()

plt.subplot(3, 1, 2)
plt.plot(timeline, e_pulse10, label='Input: 10s Pulse')
plt.plot(timeline, response_pulse10, label='Response')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Response to 10-second Pulse')
plt.grid(True)
plt.legend()

plt.subplot(3, 1, 3)
plt.plot(timeline, e_sin, label='Input: 0.5 Hz Sine Wave')
plt.plot(timeline, response_sin, label='Response')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Response to 0.5 Hz Sine Wave')
plt.grid(True)
plt.legend()

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

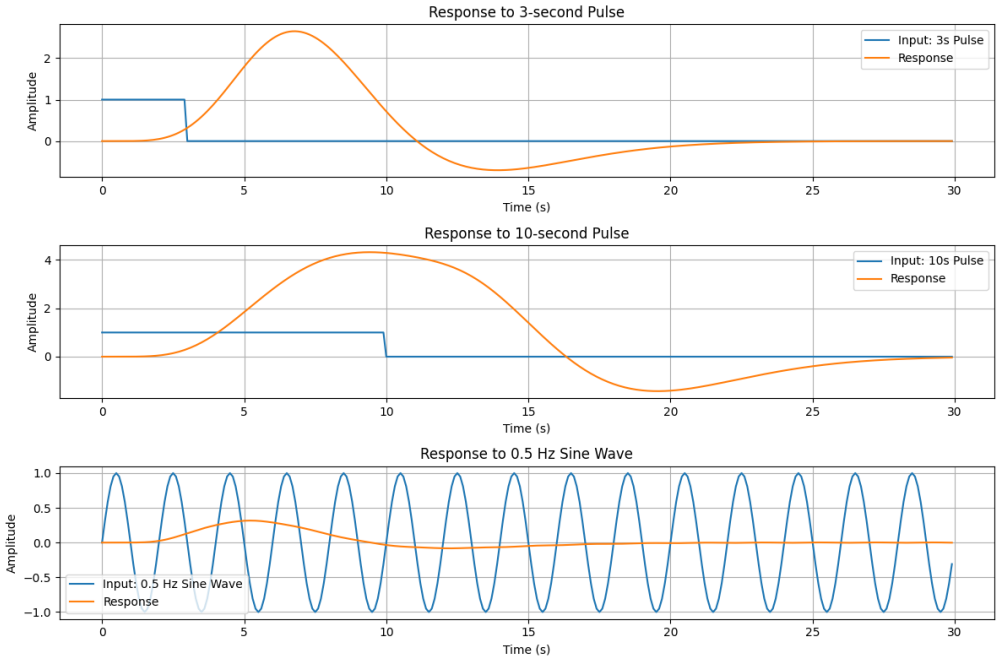

Figure 25: Response to a 3-second pulse, 10-second pulse and 0.5 Hz sine wave.

6.2.1 Interpretation

The responses to the different inputs — a 3-second pulse, a 10-second pulse, and a 0.5 Hz sinusoidal wave — highlight how the Hemodynamic Response Function (HRF) models brain activity as a linear and time-invariant system.

For the **3-second pulse**, the response shows a rapid increase, peaking around 5 seconds, followed by a smooth decay. This demonstrates how the HRF models the brain's reaction to brief stimuli with an initial sharp rise in activity and a gradual return to baseline.

For the **10-second pulse**, the response is more pronounced in both amplitude and duration. The longer input generates a larger peak, reflecting the accumulation of energy over time. Although the profile of the response is similar to that of the 3-second pulse, the extended duration of the input causes a higher overall response.

For the **0.5 Hz sinusoidal input**, the HRF generates a damped response. This occurs because the HRF acts like a low-pass filter, attenuating the higher frequency components and preserving the low-frequency ones. The response follows the input oscillation but is smoothed out, indicating the system's filtering characteristics.

In summary, these results illustrate how the HRF serves as a temporal filter that smooths out incoming signals, providing a biologically plausible model of how the brain responds to various stimuli. The HRF effectively modulates the shape and amplitude of the input signals, particularly when the inputs vary in frequency or duration.

6.3 e1 and e2 Inputs

We define two input signals e_1 and e_2 as follows:

```
TR = 1  # Sampling interval
N_pts = 30  # Number of points
timeline = np.arange(0, N_pts, TR)
h = hrf(timeline)

e1 = np.zeros(N_pts)
e1[10:13] = 1
e1[20:30] = 1
```

```
e2 = np.zeros(N_pts)
e2[5:8] = 1
e2[15:25] = 1

plt.figure(figsize=(20, 5))
plt.stem(timeline, e1, label='e1', linefmt='C0-', markerfmt='C0o', basefmt='C0-')
plt.stem(timeline, e2, label='e2', linefmt='C1-', markerfmt='C1o', basefmt='C1-')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Inputs e1 and e2')
plt.grid()
plt.legend()
plt.show()
```

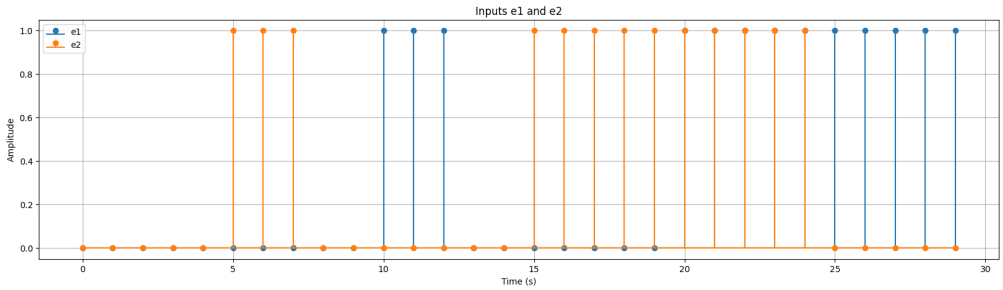

Figure 26: Inputs e_1 and e_2.

6.4 e1 and e2 Responses

We calculate the responses of the system to the inputs e_1 and e_2 using the convolution operation.

```
# Convolucion de e1 con h
response_e1 = convolve(e1, h, method='fft')[:len(e1)] * TR

# Convolucion de e2 con h
response_e2 = convolve(e2, h, method='fft')[:len(e2)] * TR

# Graficar las respuestas
plt.figure(figsize=(12, 8))

plt.subplot(2, 1, 1)
plt.plot(timeline, e1, label='Input: e1')
plt.plot(timeline, response_e1, label='Response to e1')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Response to e1')
plt.grid(True)
plt.legend()

plt.subplot(2, 1, 2)
plt.plot(timeline, e2, label='Input: e2')
plt.plot(timeline, response_e2, label='Response to e2')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Response to e2')
plt.grid(True)
plt.legend()

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

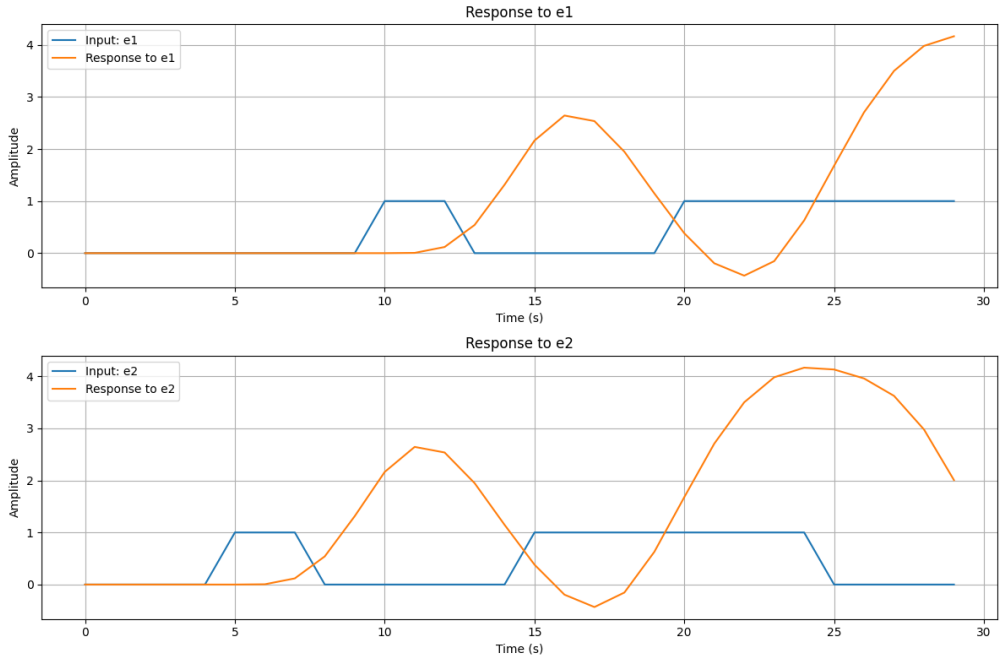

Figure 27: Response to inputs e_1 and e_2.

6.5 Linearity and Time-Invariance Verification

Verification

We combine two inputs e_1 and e_2 and Time invariance is verified by shifting the input signal e_1 by 5 seconds.:

```
# Convolucion de e1 y e2 con h
response_e1 = convolve(e1, h, method='fft')[:len(e1)] * TR
response_e2 = convolve(e2, h, method='fft')[:len(e2)] * TR

# Verificar la propiedad de linealidad
a, b = 2, 3
combined_input = a * e1 + b * e2
combined_response = convolve(combined_input, h, method='fft')[:len(combined_input)] * TR
expected_combined_response = a * response_e1 + b * response_e2

# Verificar la propiedad de invariancia temporal
shift = -5
shifted_input = np.roll(e1, shift)
shifted_response = convolve(shifted_input, h, method='fft')[:len(shifted_input)] * TR
expected_shifted_response = np.roll(response_e1, shift)

# Graficar las respuestas para verificar visualmente
plt.figure(figsize=(12, 8))

plt.subplot(3, 1, 1)
plt.plot(timeline, combined_response, label='Combined Response')
plt.plot(timeline, expected_combined_response, '--', label='Expected Combined Response')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Verification of Linearity')
plt.legend()
plt.grid(True)

plt.subplot(3, 1, 2)
plt.plot(timeline, shifted_response, label='Shifted Response')
plt.plot(timeline, expected_shifted_response, '--', label='Expected Shifted Response')
plt.xlabel('Time (s)')
```

```
plt.ylabel('Amplitude')
plt.title('Verification␣of␣Time␣Invariance')
plt.legend()
plt.grid(True)

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

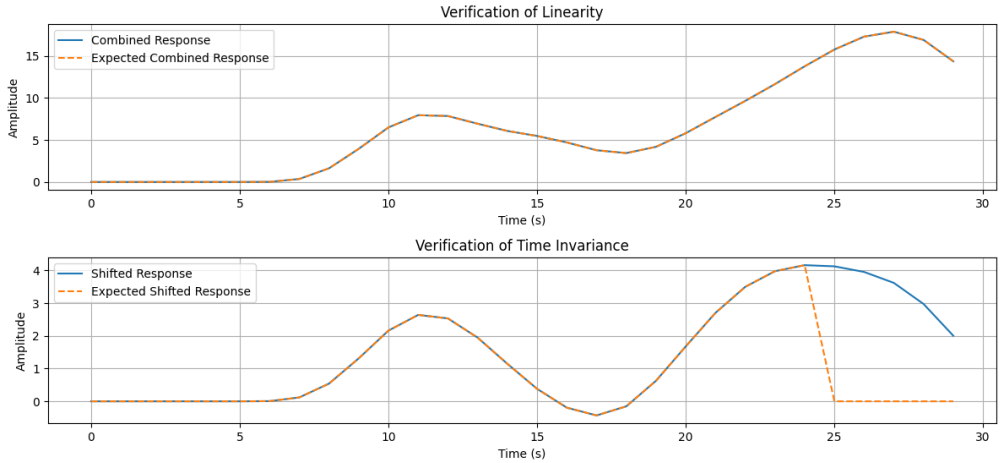

Figure 28: Linearity and Time invariance verification between e_1 and e_2.

Conclusion

Through the analysis, we verified that the HRF can be considered as a model for a linear and time-invariant system. The system fulfills the properties of linearity and time invariance, making it suitable for modeling biological responses to various types of stimuli.