Wavelet Transform - EEG Denoising and Applications

Riccardo Scibetta
BAINSA - EEG Robotic Arm Project

November 25, 2024

Contents

1 What is Wavelet Transform

1.1 Introduction

Signal processing often starts with the Fourier Transform, a powerful method for frequency analysis. However, when working with EEG data, Fourier's inability to localize time-specific features becomes a limitation. For this reason, we introduce the Wavelet Transform—a method that analyses frequency component and their location in time as well. By using compact, oscillatory waveforms known as wavelets, this technique provides a more informative perspective of signal structures. For these reasons Wavelet Transform is optimal for the analysis on Non-Stationary signals such as EEG.

1.2 Time-Frequency Duality

The French mathematician Fourier introduced the concept that, given a continuous function, you can decompose it as the sum of sine waves with different frequencies. The Fourier Transform is the process of dividing a function into the sine waves that compose it, assigning to each frequency a weight representing how much it contributes to the original function.

We say that the original function (or signal) lies in the time domain, while the information about how much different frequencies contribute to it lies in the frequency domain.

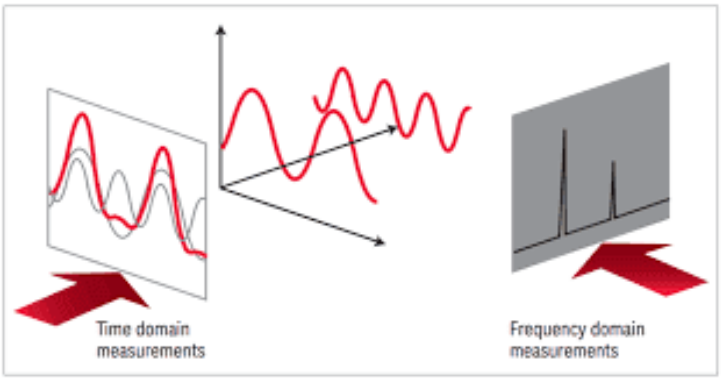

Figure 1: Time and Frequency domains

In this image, for example, the original signal is on the left, while the amplitude of the different frequencies it is decomposed into are on the right.

The inverse process, going from various sines into a function, hence from the frequency into the time domain, is called the Inverse Fourier Transform. Some applications are easier to implement in the frequency domain, others in the time domain. In any case, thanks to these instruments, it is easy to switch between the two.

The problem with the Fourier Transform is that to access the information about frequency, we lose all information about time.

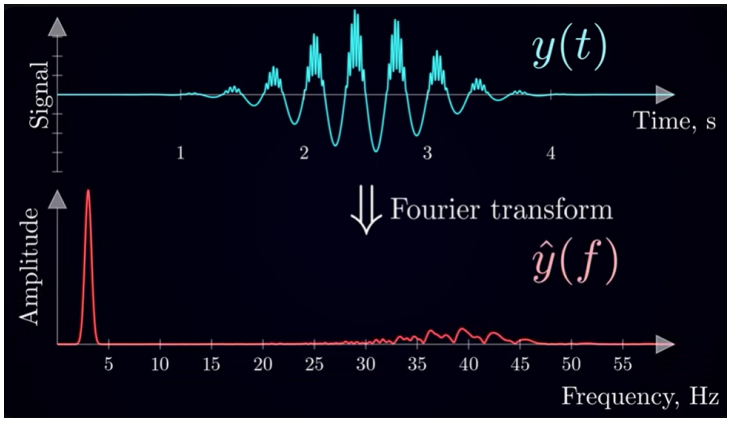

Figure 2: Fourier Tranform on a signal

We, in fact, gain the knowledge of which frequencies are more present in the signal, but lose the information on when they are present. For example, if we had a 10-second signal composed of a 1 Hz sine wave for 5 seconds, followed by a 50 Hz sine wave for the other 5 seconds, by simply looking at the Fourier Transform, we would not be able to distinguish it from another signal in which the 50 Hz comes before the 1 Hz, or even a signal in which the two alternate every two seconds.

This, however, is not a flaw of the Fourier Transform; it is impossible to have perfect information of both the time and frequency domains. There is always a trade-off between the two. This fact follows as a manifestation of the Heisenberg uncertainty principle, a fundamental concept from quantum mechanics stating that the more precisely you measure a particle's position, the less you can know its momentum and vice versa. It's application is signal processing states

$$\Delta t \cdot \Delta f \geq \frac{1}{4\pi}$$

where Δt is uncertainty about time and Δf uncertainty about frequency.

Hence, you can either know the precise value of the function at each point in time without any knowledge of the underlying frequencies, or know exactly which frequencies are present in the signal but have no idea of their temporal dynamics.

Isn't there a compromise?

1.3 Wavelets

We have seen the Fourier Transform decomposes the signal into sine and cosine functions, which are infinitely periodic in time; as anticipated earlier, this is not optimal for time series analysis in which trends are non stationary. It would be optimal if we had some kind of function that oscillates up and down (as this is the foundation of frequency representation) but somehow restrained it in time. This is where wavelets come in;

as their name from French indicates, they are like small waves: short-lived oscillations localized in time.

More properly, wavelets are not just one function, but a family of functions that satisfies certain parameters:

A proper wavelet function $\Psi(t)$ satisfies:

1.3.1 Has Zero Mean

$$\int_{-\infty}^{\infty} \Psi(t)\, dt = 0$$

This is called the admissibility condition.

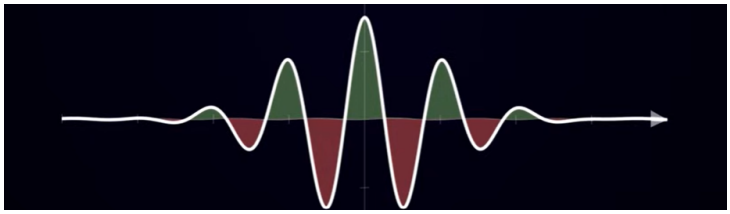

Figure 3: the sum of positive and negative areas is 0

1.3.2 Finite Energy

$$\int_{-\infty}^{\infty} |\Psi(t)|^2\, dt < \infty$$

This property imposes that it is localized in time.

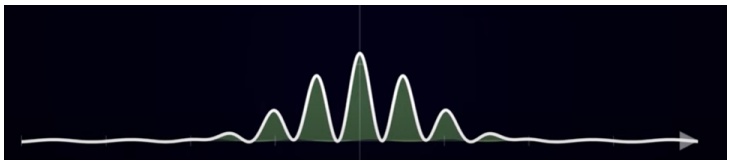

Figure 4: Finite energy

For example, the sine wave function would pass the first requirement but fail the second, as its mean is 0 but it is not localized in time.

Many wavelets have been created, each being optimal for some kind of application.

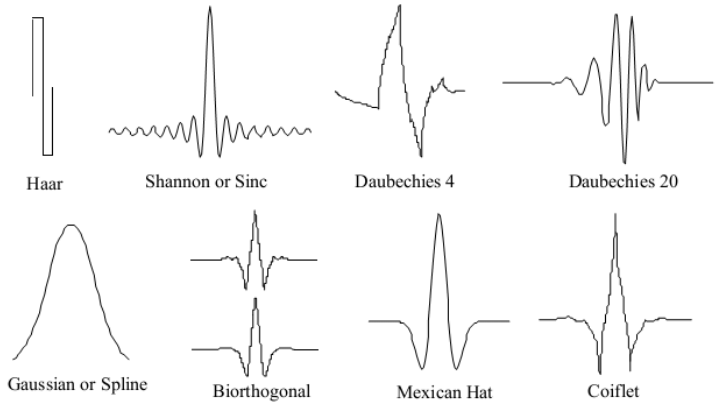

Figure 5: Different families of wavelets

1.4 Wavelet Transform

So we have seen that the Fourier Transform turns our original signal, a function of time $y(t)$, into a 1D function of frequency $Y(f)$. Instead, the Wavelet Transform turns our signal into a 2D function of both time and frequency $T(t, f)$.

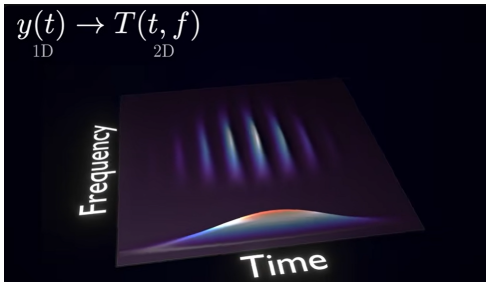

Figure 6: 3D graph of $T(t, f)$

The value of $T(t, f)$ corresponds to the contribution of the frequency f at time t in the considered signal.

1.5 Mother Wavelet Modifications

We start by taking into consideration a mother wavelet $\Psi(t)$, hence a specific wavelet that we consider the most fit for our purposes (e.g., Debauchy, Morlet, etc.). Further, we'll discuss which ones are more fit for EEG analysis. For the sake of this mathematical explanation, we'll consider the Real Component of Morlet's wavelet:

$$\Psi(t) = k_0 \cos(\omega t) e^{-\frac{t^2}{2}}$$

which is defined as the cosine wave of frequency ω dampened by a standard Gaussian.

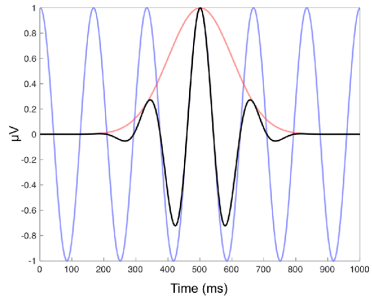

Figure 7: Morlet Wavelet in black

We now want to see how much this wave fits our signal at different times and at different frequencies. To translate our wavelet in time, we add a constant b to its argument, $\Psi(t-b)$, hence by changing b we can "slide" the wavelet in time.

To change its frequency, we divide its argument by a scale a, which has the effect of "shrinking" the wavelet as a grows, and since frequency is defined as the number of oscillations in a unit of time, the bigger a the more oscillations in one unit of time, hence the bigger the frequency.

So we obtain our scaled and translated wavelet $\Psi_{a,b}(t) = \Psi\left(\frac{t-b}{a}\right)$, which we'll call the "daughter wavelet".

$T(t,f)$ is a measure of how well this daughter wavelet "fits" with our signal at the defined time and frequency.

1.6 Computing Local Similarity

We can see a similarity when the signal and the wavelet match in signs: they match at time t if they both have a positive or negative value (green area) in t, they do not match if their signs are opposite (red area).

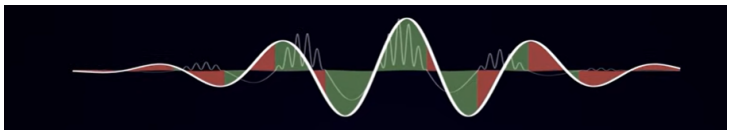

Figure 8: Wavelet is overlayed on the signal and its area is divided between matching (green) and non matching (red) signs sections

Then, multiplying the signal and the wavelet, the areas of the same sign will become positive ($+\times+=+$, $-\times-=+$) while the areas of different signs will become negative ($-\times+=-$).

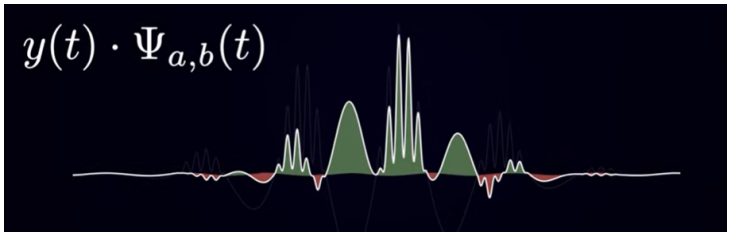

Figure 9: Multiplication of wavelet and signal

Taking an integral of this quantity is like summing the green areas (good fit) and the red areas (bad fit), effectively giving us a measure of how well the wavelet and the signal fit at that point in time.

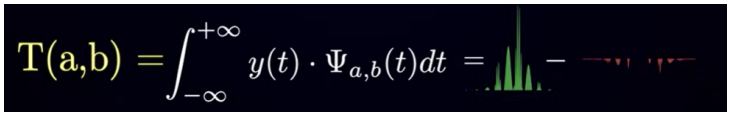

Figure 10: Total measure of "fit"

This operation can be thought of as taking the dot (inner) product of two functions, as we are taking the product of both functions at each point in time and then summing all the products together.

1.7 Convolution and Scalograms

We are now going to translate the wavelet along the signal and calculate the previously defined quantity at each step. I suggest this short video (< 1 min) to get a more visual sense of this concept. This sliding dot product is what is generally referred to as convolution.

We then proceed to repeat this method of convolution for the whole set of frequencies we are considering in our analysis.

However, we can note that the output of the convolution (blue) is a wave-like oscillation, and we'd actually be more interested in the "envelope" of it (green).

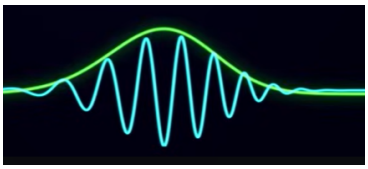

Figure 11: Convolution of the real component (blue) vs its "envelope" (green)

This is due to the fact that we only used the real component of the Morlet Wavelet Function, while the actual complete function lies in the Complex Numbers domain.

$$\Psi(t) = k_e e^{i\omega t} e^{-\frac{t^2}{2}} \quad \text{(a very nice visualization here (< 1 min))}$$

The key idea is to calculate the convolution with both the real and the imaginary components of the wavelet. To do so, we consider the contribution of the real part and the contribution of the imaginary part as coordinates of a vector in the imaginary plane, and consider the total contribution as the modulus of this vector.

The modulus of this vector will then correspond to the complex $T(a, b)$, which itself represents how well the wavelet of frequency f at time t approximates the signal. The graph of $T(a, b)$ is in 3D but can be represented through color in 2D, producing what is called the Wavelet Scalogram (the graph in the bottom section of the image).

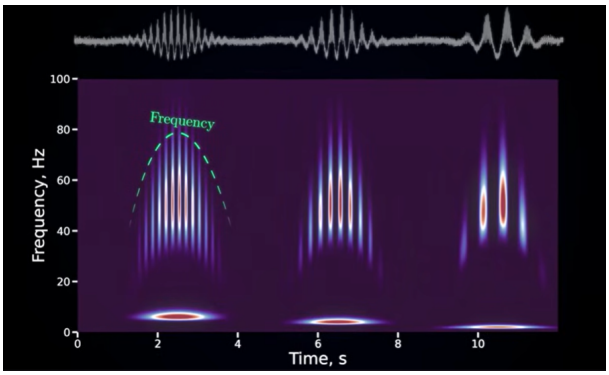

Figure 12: A signal from the mouse brain and its scalogram

We can objectively quantify all sorts of parameters, such as the frequency values, durations, the decay of frequency modulation, and so on.

1.8 Uncertainty

As previously mentioned, there is a trade-off between the uncertainty about time and that about frequency, and the Wavelet Transform is not an exception. Look, for example, at the following scalogram of the simplest signal, a pure wave:

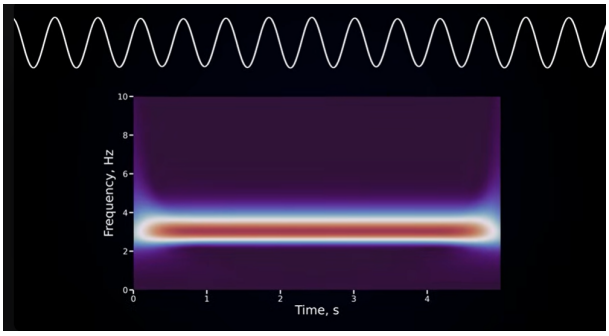

Figure 13: A pure wave and its scalogram

Even if there is only one single frequency in the signal, the scalogram does not look like an infinitely narrow bar at the height of the wave's frequency, as it would for a Fourier Transform. Instead, it looks kind of blurred around the signal's frequency. That is because we lose resolution in both time and frequency, to know something about both.

Luckily for us, the Wavelet Transform is designed in such a way that the uncertainty is distributed in a smart way:

- At **lower frequencies**, we have more resolution on frequency and less on time. If you think about it, it makes perfect sense: if we are measuring a 2 Hz wave, we want to be very sure about its frequency, as an error of even ± 1 Hz would be a pretty big relative error. At the same time, lower frequencies by definition take more time to perform an oscillation, hence we do not need very high time precision.
- Vice versa, at **higher frequencies**, we have more resolution on time than on frequency.

You can notice it by looking at the image above, as the higher frequencies are characterized by a higher level of uncertainty and hence are more blurred. You can also notice that there is a spike of uncertainty at the edges; this is intrinsic to the process of convolution. To account for it in applications, we add extra time at the beginning and at the end of the considered epoch (for example, 400 ms on each side). This process of adding time at the edges to account for avoiding edge effects or distortions is called **padding**.

2 Types of Wavelet Transform for Denoising

Different types of wavelet transforms offer various advantages and limitations in denoising applications. This section discusses the most commonly used wavelet transforms for denoising, including their methodologies, pros and cons, and how they can be optimized for better performance. We will also discuss how to choose the best mother wavelet and how to apply thresholding, a key step in the denoising process.

2.1 Continuous Wavelet Transform (CWT)

The Continuous Wavelet Transform (CWT) is what we have seen so far: a mathematical tool that decomposes a continuous-time signal into wavelets, which are scaled and translated versions of a mother wavelet. The CWT provides a highly redundant and detailed time-frequency representation of the signal, making it suitable for analyzing non-stationary signals with complex structures.

2.1.1 Pros in Denoising

- **High Time-Frequency Resolution**: CWT offers excellent time and frequency localization, which is beneficial for identifying and isolating transient features and

artifacts in signals.

- **Detailed Analysis**: The continuous nature of CWT allows for a fine-grained analysis of the signal across a continuous range of scales and translations.

2.1.2 Cons in Denoising

- **Computational Complexity**: The CWT is computationally intensive due to its continuous scaling and translation parameters, making it less practical for real-time applications or large datasets.
- **Redundancy**: The high redundancy in CWT representations can make it challenging to apply thresholding techniques effectively for denoising, as noise may not be easily distinguishable from the signal across scales.

2.1.3 Mentioning of Fast Continuous Wavelet Transform (fCWT)

The Fast Continuous Wavelet Transform (fCWT) is an optimized computational algorithm that significantly accelerates the calculation of the CWT. It enables real-time, high-resolution time-frequency analysis of non-stationary signals by separating scale-independent and scale-dependent operations and utilizing optimized Fast Fourier Transforms (FFTs) with downsampled wavelets.

While fCWT excels in providing rapid and detailed analysis, it is not optimal for denoising for the following reasons:

- **Not Suited for Thresholding**: The redundancy and dense representation in fCWT make it less effective for applying traditional thresholding techniques used in denoising.
- **Complexity**: The algorithm's complexity requires a substantial investment of time to understand and implement, which may not be justified if denoising is the primary objective.

Therefore, while fCWT is advantageous for time-frequency analysis, other wavelet transforms may be more suitable for denoising purposes.

[So, even if it is not good for denoising, this tool might become useful when we approach time-series analysis]

2.2 Discrete Wavelet Transform (DWT)

The most important type of Wavelet Transform is the Discrete Wavelet Transform. Not to prolong ourselves we will now explain DWT by comparing it with CWT, which is more thoroughly explained in the first chapter.

2.2.1 Nature of the Transformation

CWT:

- Provides a continuous representation of the signal. It computes the wavelet coefficients for all possible scales (frequencies) and translations (time shifts), creating a highly detailed, redundant map of the signal.
- The output is a function of two continuous variables: scale (or frequency) and time.
- This redundancy makes it useful for detailed signal analysis but computationally expensive.

DWT:

- Provides a discrete representation of the signal. It samples scales and translations at discrete intervals, often chosen to create an orthogonal or biorthogonal wavelet basis. This process is performed recursively, at each step separating the signal into **approximation** (low-frequency) and **detail** (high-frequency) coefficients.
- The output is a set of discrete wavelet coefficients at specific scales and time positions.
- This makes the representation compact, nonredundant, and computationally efficient.

Note: In essence an orthogonal wavelet basis (their dot product is zero) ensures that all wavelets carry distinct information, maximizing efficiency.

NB: As we perform more **levels** (recursive steps), we will increase frequency resolution, which can be beneficial for denoising as it facilitates the application of thresholding techniques. At the same time, more levels lead to a lower time resolution and higher computational costs. Typically, denoising benefits from moderate levels (3-6), a good compromise for having a good identification of noise without excessively compromising time resolution. In any case, selecting the right number of levels often involves empirical testing and validation to balance the trade-offs based on performance metrics, like Signal-to-Noise Ratio (SNR), Mean Squared Error (MSE), or classification accuracy.

2.2.2 Sampling

CWT:

- No explicit sampling; it essentially evaluates wavelet coefficients for every possible combination of scale and translation.
- The result is a "redundant" transform because the same information can be expressed more concisely.

DWT:

- Samples scales and translations at predefined intervals, often powers of two (dyadic sampling), ensuring an efficient and compact representation.
- It avoids redundancy by carefully choosing scales and shifts to form an orthogonal or biorthogonal basis.

2.2.3 Output Representation

CWT:

- Produces a scalogram: a 2D map of wavelet coefficients as a function of scale and time. This provides a detailed, continuous picture of the signal's frequency content over time.
- The redundancy makes it visually and intuitively informative, useful for pattern recognition and qualitative analysis.

DWT:

- Produces a set of wavelet coefficients corresponding to discrete time and scale values. These coefficients are compact and suitable for storage, reconstruction, and further processing.
- Commonly used for quantitative tasks like compression, noise reduction, and efficient signal representation.

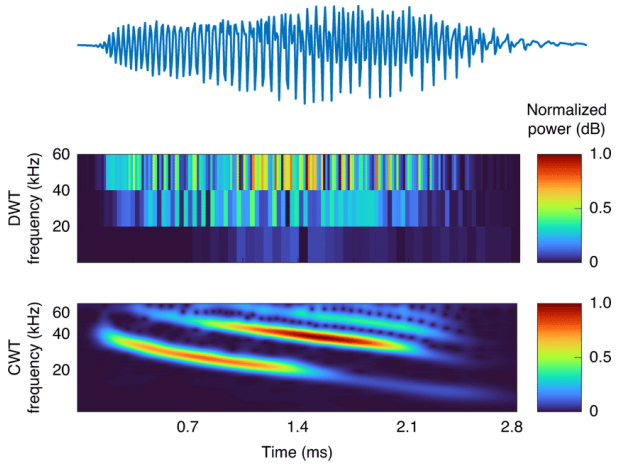

Figure 14: A signal with the graph of DWT coefficients and its scalogram (CWT)

2.2.4 Pros in Denoising

- **Computational Efficiency**: DWT is less computationally intensive compared to CWT, making it suitable for real-time denoising applications.
- **Effective Thresholding**: The orthogonality and sparsity facilitate the application of thresholding techniques to suppress noise while preserving important signal features.

2.2.5 Cons in Denoising

- **Shift Variance**: DWT is not shift-invariant; small shifts in the input signal can cause large changes in the wavelet coefficients, potentially affecting denoising performance.
- **Edge Effects**: DWT can introduce artifacts near the boundaries of the signal due to the finite-length filters and signal extension methods.

2.3 Stationary Wavelet Transform (SWT)

2.3.1 What It Is

The Stationary Wavelet Transform (SWT) is a version of the Discrete Wavelet Transform (DWT) that solves one of its main issues: sensitivity to shifts in the input signal (shift invariance).

- **No Downsampling**: Unlike the DWT, which reduces the length of the signal by removing some of its samples, the SWT does not downsample. This means that the signal's length stays the same throughout all levels of the transformation.
- **What This Means**: By avoiding downsampling, the SWT ensures that the information from the signal aligns perfectly across all levels of the decomposition. This makes it "shift-invariant," meaning the results of the transformation don't change if the signal is shifted slightly in time.

2.3.2 Pros in Denoising Compared to Other Techniques

- **Shift Invariance**: SWT's shift invariance leads to more consistent denoising results.
- **Improved Edge Preservation**: By avoiding downsampling, SWT reduces artifacts near the signal boundaries.
- **Enhanced Noise Reduction**: The redundancy in SWT can improve denoising performance by averaging out noise.

2.3.3 Cons

- **Increased Computational Load**: SWT is computationally more intensive than DWT due to the absence of downsampling and the resulting redundancy.

2.4 Hybrid Methods

Hybrid methods combine wavelet transforms with other signal processing techniques to enhance denoising performance. Two notable combinations are discussed below.

2.4.1 Incorporation of Independent Component Analysis (ICA) and Wavelet Transform

What ICA Is: Independent Component Analysis (ICA) is a computational method for separating a multivariate signal into additive, statistically independent components. (For more info, refer to Anna's document on Blind Source Separation.)

Combination with Wavelet Transform for Denoising: By integrating ICA with wavelet transforms, one can leverage the strengths of both methods:

1. **ICA Decomposition**: Apply ICA to multichannel EEG data to separate the signals into independent components (ICs).
2. **Identification of Artifacted Components**: Identify ICs that represent artifacts or noise based on statistical properties or spatial distributions.
3. **Wavelet Denoising of Artifacted ICs**: Apply wavelet transform (DWT or SWT) to the artifacted ICs and perform thresholding to remove noise.
4. **Reconstruction**: Reconstruct the denoised signal by performing the inverse ICA using the cleaned ICs.

2.4.2 Pros

- **Reducing Computational Complexity**: Applying wavelet transform to each channel individually is computationally intensive. By first decomposing the data with ICA, wavelet denoising is applied only to selected ICs, reducing the overall computational load.
- **Improving Accuracy**: Preservation of Neural Information: By denoising at the IC level, important signal features are better preserved.

2.4.3 Downsides

- **Assumptions Required**:
  - **Statistical Independence**: The sources must be statistically independent.
  - **Non-Gaussian Distribution**: Each independent component should have a non-Gaussian distribution.
  - **Determined Mixing System**: The mixing system (number of observed mixtures equal to the number of source signals) must be determined.
- **Limitations in Single-Channel Data**: ICA is not effective for single-channel signals or datasets with fewer channels than sources.

2.4.4 Wavelet Transform Combined with Empirical Mode Decomposition (EMD)

Empirical Mode Decomposition (EMD), unlike traditional methods like Fourier or wavelet transforms, does not rely on predefined basis functions. Instead, it adapts to the signal's intrinsic characteristics by extracting Intrinsic Mode Functions (IMFs): oscillatory components capturing different frequency bands. (For more info, refer to Samu's document on EMD.)

Combination with Wavelet Transform:

1. **EMD Decomposition**: Decompose the noisy signal into IMFs using EMD.
2. **Wavelet Denoising of IMFs**: Apply wavelet transform and thresholding to the IMFs to remove noise.
3. **Reconstruction**: Sum the denoised IMFs to reconstruct the denoised signal.

2.4.5 Advantages of Combining EMD with Wavelet Transform

- **Improved Denoising**: Wavelet thresholding can effectively denoise individual IMFs, enhancing overall signal quality.
- **Complementary Strengths**: EMD provides adaptive decomposition, while wavelet transform offers efficient denoising techniques.

2.4.6 Limitations

- **Complexity**: Combining EMD with wavelet transform increases computational complexity.
- **Parameter Selection**: Requires careful selection of parameters for both EMD and wavelet thresholding to achieve optimal results.

2.5 How to Choose the Correct Mother Wavelet

Selecting an appropriate mother wavelet is crucial for effective denoising. The choice depends on the characteristics of the signal:

- **Similarity to Signal Features**: The shape of the mother wavelet should resemble the features present in the signal for better representation (for example, spikes, sharp waves, and other non-stationary components).
- **Vanishing Moments**: Vanishing moments refer to the number of polynomial moments that a wavelet can integrate to zero; a wavelet with N vanishing moments is orthogonal to all polynomials of degree less than N. Therefore, wavelets with higher vanishing moments can effectively capture polynomial trends in the data, which is useful for denoising smooth signals and better distinguishing between signal and noise. Lower vanishing moments instead are better at detecting abrupt changes and edges in signals.

2.5.1 Examples of Wavelets with Different Vanishing Moments

Haar Wavelet:

- **Vanishing Moments**: 1
- **Characteristics**: Piecewise constant, good for detecting sharp changes but not ideal for smooth signal approximation.

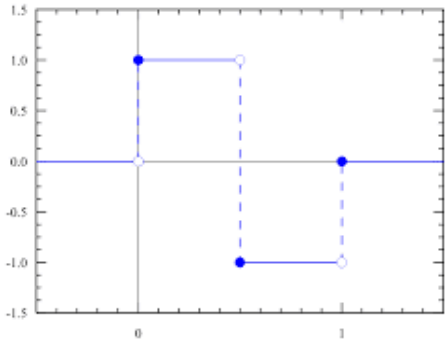

Figure 15: Haar Wavelet

Daubechies (db):

- **Features**: Orthogonal, compact support, varying degrees of smoothness.
- **Usage**: db4 and db6 are popular for EEG denoising. Good for detecting sharp changes and transient events.

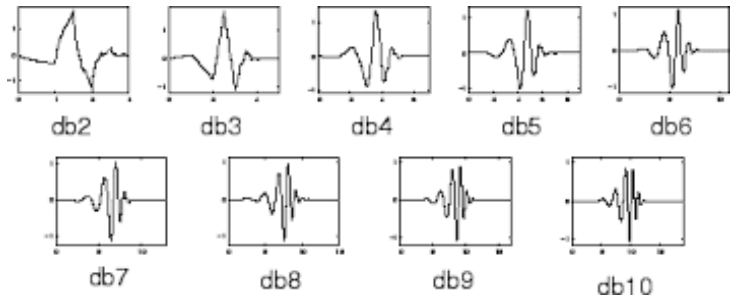

Figure 16: Daubechie Wavelets

Coiflets (coif):

- **Features**: Orthogonal, higher number of vanishing moments.
- **Usage**: Capture fine details and smoother signal components. Useful when preserving low-frequency trends.

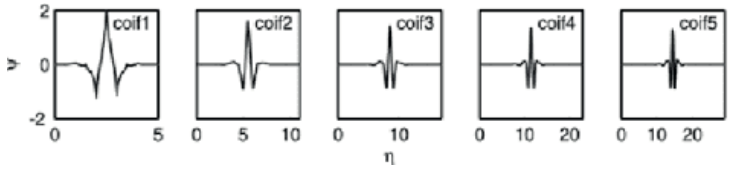

Figure 17: Coiflet Wavelets

Symlets (sym):

- **Features**: Nearly symmetrical, modified Daubechies wavelets.
- **Usage**: sym5 and sym8 offer a balance between symmetry and compact support. Suitable for minimizing phase distortion.

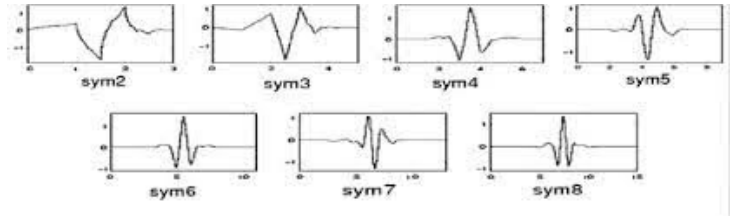

Figure 18: Symlet Wavelets

2.5.2 Choosing

Considering what was mentioned above, a good process for choosing the right wavelet is the following:

1. Start with commonly used wavelets like db4, sym5, and coif5. Perform denoising and evaluate the results visually and quantitatively.
2. Calculate SNR, MSE, or other relevant statistics to compare wavelets.
3. Fine-tune the process by adjusting decomposition levels.
4. Validate the denoising performance on different EEG datasets to ensure generalizability.

2.6 Thresholding Techniques

Finally, we come to consider the issue of Thresholding. Thresholding is a method used to distinguish between significant signal components and noise by applying a threshold value. Components of the signal that fall below this threshold are considered noise and are either reduced or removed, while those above the threshold are retained or minimally altered. This is based on the intuition that the signals we want to observe are localized in certain frequencies, which in turn will have high coefficients, while the frequencies that have low coefficients are just a byproduct of the brain and are not of interest to us.

Hard Thresholding : In hard thresholding, coefficients with absolute values below the threshold are set to zero, effectively removing them. Coefficients above the threshold remain unchanged.

$$T_{\text{hard}}(x, \lambda) = \begin{cases} x & \text{if } |x| \geq \lambda, \\ 0 & \text{otherwise.} \end{cases}$$

It is the simplest to implement and completely removes noise components below the threshold. The downside is that it can introduce discontinuities, leading to artifacts like the "staircase effect" and potentially distorting the original signal.

Soft Thresholding : Soft thresholding not only sets coefficients below the threshold to zero but also shrinks the remaining coefficients towards zero by the threshold value.

$$T_{\text{soft}}(x, \lambda) = \begin{cases} \text{sign}(x)(|x| - \lambda) & \text{if } |x| \geq \lambda, \\ 0 & \text{otherwise.} \end{cases}$$

This technique produces smoother results with fewer artifacts, better preserving the overall shape of the signal. However, it can slightly alter significant signal components by shrinking them.

2.6.1 Workflow

The general workflow of thresholding is the following:

1. **Wavelet Decomposition**: The signal is decomposed into wavelet coefficients at various scales.
2. **Thresholding**: Apply hard or soft thresholding to the detail coefficients to eliminate or reduce noise.
3. **Wavelet Reconstruction**: Reconstruct the denoised signal from the modified coefficients.

2.6.2 Choosing the Right Threshold

Selecting an appropriate threshold is critical for effective denoising. Several methods exist to determine the optimal threshold:

Universal Threshold (Donovan's Threshold)

Formula:

$$\lambda = \sigma\sqrt{2\ln(N)}$$

Where:

- σ = Estimated noise standard deviation.
- N = Number of data points.

Usage: Suitable for signals with Gaussian noise.

VisuShrink

Characteristics: Utilizes the universal threshold.

- **Pros**: Simple and widely used.
- **Cons**: May overshrink coefficients, leading to loss of signal details.

SureShrink (Stein's Unbiased Risk Estimate)

Characteristics: Selects a threshold based on minimizing the mean squared error.

- **Pros**: Balances noise reduction and signal preservation better than the universal threshold.
- **Cons**: More computationally intensive.

Adaptive Thresholding

Description: Instead of using a single global threshold, adaptive methods adjust the threshold based on local signal characteristics to enhance denoising performance.

- **Pros**: Can provide better denoising performance.
- **Cons**: Complexity varies depending on the specific technique used, but generally increases.

3 Code Implementation

In this section, we provide a brief Python code example demonstrating the denoising of EEG signals using Wavelet Transform techniques. The process involves loading necessary libraries, processing EEG data, applying wavelet decomposition, thresholding, reconstructing the signal, and evaluating the denoising quality through visual inspection and metric calculations.

3.1 Loading Useful Libraries

First, install and import the required Python libraries.

```
# pip install pywavelets  # Library for handling wavelets
import pywt
import numpy as np
import mne
from mne.datasets import eegbci
import matplotlib.pyplot as plt
```

Listing 1: Installing and Importing Libraries

3.2 Loading the EEG Signal Using MNE

Next, we load the EEG signal using the MNE library.

```
# Download EEGBCI dataset
eegbci_files = eegbci.load_data(subject=1, runs=[1])

# Get the path to the EEGBCI dataset
eegbci_path = eegbci_files[0]

# Load the raw EEG data file
raw = mne.io.read_raw_edf(eegbci_path, preload=True)
```

Listing 2: Loading EEG Data

For simplicity, we consider a single channel and apply wavelet decomposition on it.

```
eeg_channel = 'Fc6.'  # We chose a single channel to simplify the
    process

# Get data for the selected channel
eeg_signal, times = raw.get_data(picks=eeg_channel, return_times=
   True)
eeg_signal = eeg_signal[0]  # Extract the signal array
```

Listing 3: Selecting a Single EEG Channel

3.3 Wavelet Decomposition

Choose an appropriate mother wavelet and decomposition level before performing the wavelet decomposition.

```
# Choose wavelet and decomposition level
wavelet = 'db4'  # Options: 'db4', 'sym5', 'coif5', etc.
level = 4  # Generally between 3-6

# Perform wavelet decomposition
coeffs = pywt.wavedec(eeg_signal, wavelet, level=level)
```

Listing 4: Wavelet Decomposition

3.4 Thresholding

Thresholding is applied to the detail coefficients to eliminate or reduce noise.

```
# Estimate noise sigma from detail coefficients at level 1
sigma = np.median(np.abs(coeffs[-1])) / 0.6745

# Define universal threshold
threshold = sigma * np.sqrt(2 * np.log(len(eeg_signal)))

# Apply soft thresholding to detail coefficients
thresholded_coeffs = [coeffs[0]]  # Approximation coefficients
    remain unchanged
thresholded_coeffs += [pywt.threshold(c, threshold, mode='soft')
    for c in coeffs[1:]]
```

Listing 5: Applying Thresholding

3.5 Reconstructing and Plotting the Denoised Signal

Reconstruct the denoised signal and plot both the original and denoised signals to visually inspect the denoising results.

```
# Reconstruct the denoised signal
denoised_eeg = pywt.waverec(thresholded_coeffs, wavelet)

# Plot original EEG signal
plt.figure(figsize=(12, 6))
plt.plot(times, eeg_signal, label='Original Signal', alpha=0.7)
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.legend()
plt.title(f'Denoising EEG Signal - Channel {eeg_channel}')
plt.show()
```

```
12
13 # Plot denoised EEG signal
14 plt.figure(figsize=(12, 6))
15 plt.plot(times[:len(denoised_eeg)], denoised_eeg, label='Denoised
       Signal', alpha=0.7)
16 plt.xlabel('Time (s)')
17 plt.ylabel('Amplitude')
18 plt.legend()
19 plt.title(f'Denoising EEG Signal - Channel {eeg_channel}')
20 plt.show()
```

Listing 6: Reconstructing and Plotting Signals

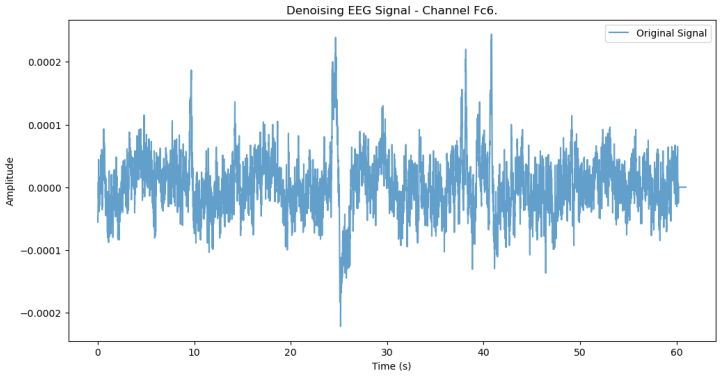

Figure 19: Original EEG data

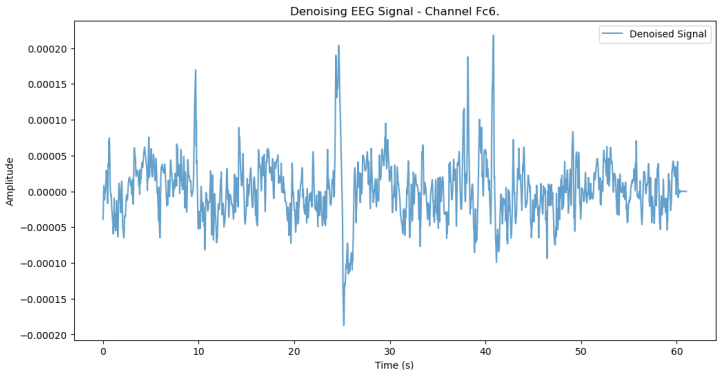

Figure 20: Denoised EEG data

3.6 Calculating Denoising Metrics

To quantitatively assess the denoising quality, we calculate the Mean Squared Error (MSE) and Signal-to-Noise Ratio (SNR).

```
1 # Calculate Mean Squared Error (MSE)
2 mse = np.mean((eeg_signal[:len(denoised_eeg)] - denoised_eeg) **
     2)
3 print(mse)
```

```
4
5 # Calculate Signal-to-Noise Ratio (SNR)
6 signal_power = np.mean(eeg_signal ** 2)
7 noise_power = np.mean((eeg_signal[:len(denoised_eeg)] -
     denoised_eeg) ** 2)
8 snr = 10 * np.log10(signal_power / noise_power)
9 print(snr)
```

Listing 7: Calculating Denoising Metrics

Which outputs: 2.682791301180141e-10 (MSE) 8.670166630009573 (SNR)

3.7 Running the Code and Exploring Parameters

You can now execute this code on your laptop to observe how the plots and metrics change when you navigate through the permutations of the different parameters:

- Mother Wavelet
- Decomposition Level
- Thresholding Technique

Disclaimer

A fair warning to the mathematically inclined reader: the world of Wavelet Transform is a wonderfully complex beast. While I've done my best to tame it for this introduction, some concepts have been approximated or simplified to avoid getting lost in the maths. Think of this as your friendly guide to WT - comprehensive enough to get you oriented, but light in formal concept definitions.

References

- Huge shoutout to Artem Kirsanov for this awesome video `https://www.youtube.com/watch?v=jnxqHcObNK4&t=1685s`
- Shoutout to 3BlueOneBrown as well `https://www.youtube.com/watch?v=spUNpyF58BY`
- `https://www.scirp.org/journal/paperinformation?paperid=19`
- `https://it.mathworks.com/help/wavelet/gs/continuous-and-discrete-wavelet-transfo html`
- `https://www.nature.com/articles/s43588-021-00183-z`

- On various WT Denoising techniques and their comparison: https://www.mdpi.com/2624-6120/3/3/35