Analyzing EEG Signal

Surname/Name: Podimarakkala Gurunnanselage Kasun Achintha Perera
(Dated: July 20, 2025)

Electroencephalography (EEG) is a non-invasive tool used to diagnose epilepsy by analyzing the brain's complex dynamics as a system of coupled oscillators. Their reliability is governed by a critical coupling strength, ϵ_c, which is inversely proportional to the frequency distribution $g(\omega)$. This relationship was analyzed using various signal processing methods, including Hilbert Transform, EMD, FFT, ZCR, PSD, and AR models, each of which yielded distinct results. As the parameter ϵ increased, the Hilbert Transform (from 0.1618 to 1.4560) and ZCR (from 0.3857 to 3.4714) methods showed a transition to a low-variation, clustered state, while the EMD (from 0.2274 to 2.7294) and AR models (from 0.1618 to 19.8135) transitioned to a single, continuous phase arc. The FFT method (from 0.1325 to 1.1923) moved to a lower-variation state with a dominant cluster from 0° to 90°. A direct correlation was observed between the estimated $g(\omega)$ and the critical coupling ϵ_c: the FFT method had the highest $g(\omega)$ of 0.4898 and the lowest ϵ_c of 1.3179, while the AR model had the lowest $g(\omega)$ of 0.0292 and the highest ϵ_c of 22.0116. In terms of classification accuracy, the FFT method was the most reliable, with 101 correct classifications and an accuracy of 59.06%, while the AR model was the least accurate with 56 correct classifications and an accuracy of 32.75%. It is noted that the results for the EMD and AR methods might be affected by their use of a reduced dataset of 27,000 data points due to high computational costs.

INTRODUCTION

Electroencephalography (EEG) is a non-invasive neuro diagnostic technique that records the electrical activity of the brain by placing electrodes on the scalp. They integrate the diagnosis of epilepsy by detecting epileptic abnormalities, such as spike-wave discharges. It also plays a critical role in classifying seizure types and informing evidence based treatment strategies. However, as EEG captures only the brain activity occurring during the recording period, it is required to integrate with a comprehensive clinical history and additional diagnostic tools to ensure a thorough assessment. Ongoing researchers are aimed to enhance the accuracy and effectiveness, including advancements in image based analysis and the application of electrocorticogram of EEG in the management of epilepsy.

The brain operates as a complex dynamical system of interacting neural networks, and interpreting EEG patterns are essential to understand its properties of stability. Dynamical systems theory provides a framework for analyzing this phenomena. It provides an insight into how nonlinear systems like brain behave under normal and pathological conditions. In the dynamical systems, Chaos are defined by positive Lyapunov exponents(TLE). They reflect a sensitive dependence on initial conditions and identical systems (replicas) with slightly different starting states diverge over time and losing their correlation. Reliability and anti-reliability describe how these replicas respond to external forces, such as noise or chaotic driving. A relation between reliability and correlations emerges in the Kuramoto model of coupled oscillators. It is interpreted as a dissipation fluctuation principle with the concept of extending replicated subnetworks. These findings deepen our understanding of synchronized neural activity, and enhance the interpretation of EEG patterns in epilepsy, while supporting the advancements in diagnosis and treatment.

THEORY

The **Kuramoto model** provides a foundational framework for studying internal reliability in networks of coupled oscillators. Each of the N oscillators has a natural frequency ω_k, drawn from a distribution $g(\omega)$, and interacts with all others through sinusoidal coupling. These systems evolve according to the equation given below.

$$\dot{x}_k = \omega_k + \frac{\varepsilon}{N}\sum_{j\neq k}\sin(x_j - x_k), \quad k = 1, \ldots, N,$$

where ε is the global coupling strength. This formulation captures the essential dynamics driving synchronization among oscillators. The evolution of small perturbations are examined to assess reliability. v_k is transverse to the trajectory and governed by the variational equation

$$\dot{v}_k = -\frac{\varepsilon}{N}\sum_{j\neq k}\cos(x_j - x_k)v_k.$$

In the thermodynamic limit ($N \to \infty$), the mean fields stabilize the dynamics to stationary or periodic behavior. As a result non-positive transverse Lyapunov exponents (TLEs) show overall reliability. But For finite networks, fluctuations cause some units to have positive TLEs and indicating anti-reliability. At weak coupling ($\varepsilon \to 0$), all TLEs approach zero, but finite coupling leads to heterogeneous reliability showing some units to become reliable (negative TLEs), and others are to anti-reliable (positive TLEs). When coupling exceeds the critical threshold $\varepsilon_c = \frac{2}{\pi g(0)}$, global synchronization emerges and all

units become reliable. Additionally, an oscillator's reliability depends on its natural frequency ω_k, with units near the center of $g(\omega)$ being most reliable and those in the distribution tails more likely anti-reliable. [1]

METHODS

Signal Analysis Techniques

I. Hilbert Transform

The Hilbert transform facilitates the extraction of frequency and phase by transforming a real-valued signal, $x(t)$, into a complex-valued analytic signal, $Z(t) = x(t) + j\mathcal{H}[x(t)]$, where $\mathcal{H}[x(t)]$ is the Hilbert transform. This process allows the signal's properties to be expressed in polar form as $Z(t) = A(t)e^{j\phi(t)}$, where the instantaneous amplitude $A(t) = \sqrt{x(t)^2 + (\mathcal{H}[x(t)])^2}$ and the instantaneous phase $\phi(t) = \arctan 2(\mathcal{H}[x(t)], x(t))$ can be directly computed. While the phase provides the signal's angular information, the instantaneous frequency, a measure of how the signal's frequency changes over time, is subsequently derived by taking the time derivative of the unwrapped phase, $\omega(t) = \frac{d\phi(t)}{dt}$. This technique provides a rigorous, time-localized measure of both amplitude and frequency, essential for analyzing non-stationary data.[2],[3]

Python script is used analyze the initial phase distribution of multiple EEG channels using the Hilbert transform. The process begins with a `for` loop that iterates through each column of the `EEG` DataFrame. For each channel's signal data, the script uses `scipy.signal.hilbert` to compute the complex-valued analytic signal. The instantaneous phase is then extracted from this analytic signal using `np.angle`, and the value at the very first time point (`phase[0]`) is stored. After the loop, the collected initial phases are converted into a `numpy` array `x`, and the `np.mod` function is used to wrap all phase values to the standard range of $[0, 2\pi)$. Finally, `matplotlib.pyplot` is utilized to visualize this phase distribution through polar plot(FIG. 1). It shows the points distributed on a unit circle from 0 to 2π.

Next step analyzes and visualizes the instantaneous frequency and dominant "natural" frequencies of EEG signals using the Hilbert transform. A function, `full_instantaneous_frequency`, is defined to perform the core calculation: it applies `scipy.signal.hilbert` to a signal, unwraps the phase using `np.unwrap(np.angle())`, and then computes the instantaneous frequency as the time derivative of the phase using `np.diff`. The main part of the script iterates through each EEG channel. For each channel, it computes the instantaneous frequency spectrum(FIG. 2) and generates a histogram(FIG. 3) to represent its natural frequency. It then uses `scipy.signal.find_peaks` to automatically identify the dominant frequency, or "natural frequency," which corresponds to the highest peak in the histogram. These dominant frequencies are collected in a list and finally visualized in a separate histogram plot to show their overall distribution across all channels.

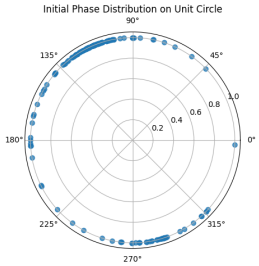


FIG. 1: Intial Phase Distribution obtained under Hilbert Transform method

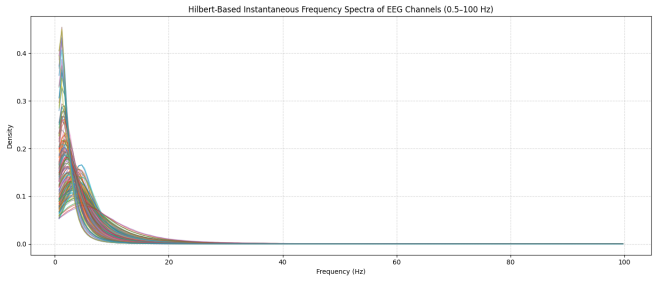


FIG. 2: Frequency Spectrum obtained under Hilbert Transform method

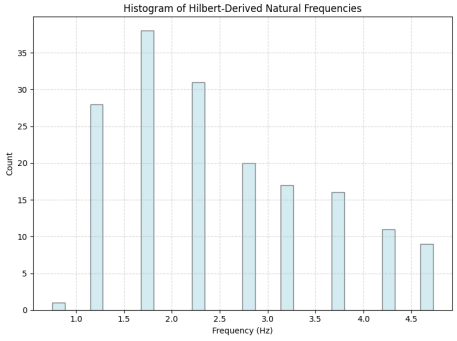


FIG. 3: Natural Frequencies obtained under Hilbert transform method

II. Hilbert-Huang Transform (HHT) and Empirical Mode Decomposition (EMD)

The Hilbert-Huang Transform (HHT) is a powerful and data-adaptive method for analyzing non-stationary signals such as signals with frequency content changes

over time. It operates in two main steps. First, Empirical Mode Decomposition (EMD) breaks down the complex signal into a collection of simpler, oscillatory components called Intrinsic Mode Functions (IMFs). Each IMF represents a distinct, well-behaved oscillatory mode present in the data. Second, the Hilbert Transform is applied to each IMF to create an analytic signal. From this analytic signal, the instantaneous phase and amplitude of each IMF can be directly extracted. By taking the time derivative of the instantaneous phase, the instantaneous frequency is obtained. This two-stage process allows HHT to provide a precise, time-varying representation of a signal's frequency and phase content, offering insights that are often obscured by traditional methods like Fourier analysis.[2]

Python script is used to analyze the initial phase of EEG signals by combining Empirical Mode Decomposition (EMD) and the Hilbert transform. The core functionality is contained within the `extract_initial_phase` function, which first uses `PyEMD.EMD()` to decompose a given signal into its Intrinsic Mode Functions (IMFs). For the first IMF (the highest frequency component), it applies `scipy.signal.hilbert` to compute the analytic signal and then uses `np.angle` to extract its phase at the initial time point. To efficiently process each EEG channel, the script leverages `joblib.Parallel` and `joblib.delayed`, enabling concurrent execution of the analysis across all available CPU cores. The results are stored in a `pandas.Series` and then visualized using `matplotlib.pyplot`: a polar scatter plot(FIG. 4) provides a clear and circular representation of the initial phases.

Then parallelized the Hilbert-Huang Transform (HHT) to extract key frequency metrics from EEG data. The core logic resides in the `hht_analysis` function, which first employs `PyEMD.EMD()` to decompose a signal into Intrinsic Mode Functions (IMFs). For each IMF, `scipy.signal.hilbert` is applied to derive the instantaneous phase, from which the instantaneous frequency is computed using `np.unwrap` and `np.diff`. The script then calculates a `weighted_freq` by combining the instantaneous frequencies and amplitudes of the IMFs, and it identifies a `dominant_freq` from the peak of its histogram. The entire process is efficiently parallelized across all EEG channels using `joblib.Parallel` and `joblib.delayed`. Finally, `matplotlib.pyplot` is used to generate two visualizations: an overlaid plot of the frequency spectrum(FIG. 5) for all channels and a histogram(FIG. 6) of the extracted dominant frequencies.

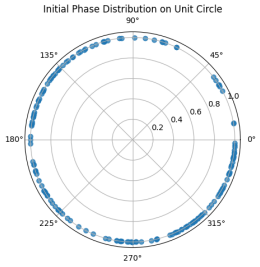


FIG. 4: Intial Phase Distribution obtained under EMD method

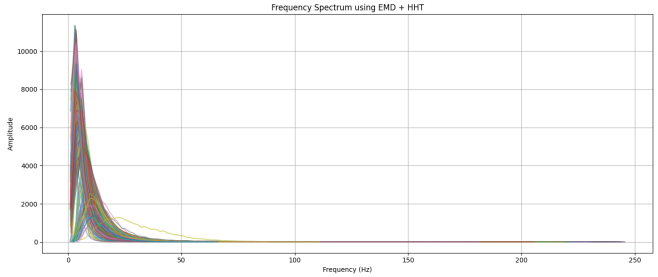


FIG. 5: Frequency Spectrum obtained under EMD method

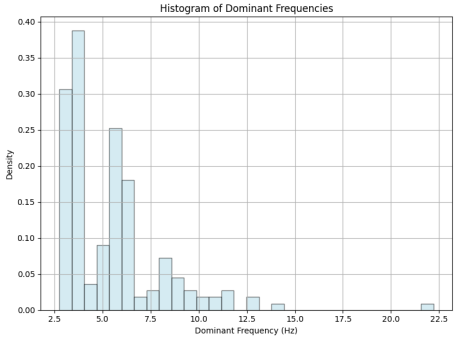


FIG. 6: Natural Frequencies obtained under EMD method

III. Fast Fourier Transform

The Fast Fourier Transform (FFT) is an efficient algorithm used to convert a discrete signal from the time domain to the frequency domain by representing it as a sum of sinusoidal components. The algorithm processes a sequence of time-domain samples and outputs a sequence of complex numbers, and each complex number corresponds to a specific frequency bin. The amplitude of each frequency component is then extracted as the magnitude of its associated complex number, while its phase is given by the angle of the complex number in the complex plane. This process, often visualized as a power spectrum, provides a static spectral representation of the

signal, revealing the strength and initial phase of all frequency components present over the entire captured time window.[4],[3]

A Python script is analyzes the initial phase of EEG signals using a Fast Fourier Transform (FFT)-based method. The process begins with a loop that iterates over each channel of the `EEG` DataFrame. For each channel, the script computes the FFT of the signal using `np.fft.fft` and determines the corresponding frequencies with `np.fft.fftfreq`. It then isolates a specific frequency band (0.5 to 100 Hz) and identifies the dominant frequency component by finding the index with the maximum magnitude. The initial phase is extracted from this dominant component's complex value using `np.angle`, and subsequently wrapped to the $[0, 2\pi)$ range with `np.mod`. These initial phases are collected into a `numpy` array, which is then visualized using `matplotlib.pyplot` in a polar scatter plot(FIG. 7) to represent the phases on a unit circle.

Then Fast Fourier Transform (FFT) is used to extract and visualize the dominant "natural" frequencies of EEG signals. The process iterates through each EEG channel, and for each, it computes the frequency-domain representation using `np.fft.fft` and `np.fft.fftfreq`. After calculating the amplitude of the spectrum, the script applies a frequency mask to focus on the 0.5-100 Hz band relevant to EEG data. Within this band, `np.argmax` is used to find the index of the highest amplitude, which corresponds to the dominant frequency for that channel. These dominant frequencies are collected and then visualized using `matplotlib.pyplot` in two ways: first, as an overlaid plot showing the frequency spectrum(FIG. 8) for each individual channel, and second, as a histogram(FIG. 9) displaying the overall distribution of the extracted natural frequencies.

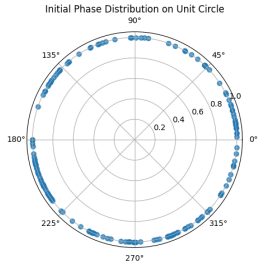


FIG. 7: Intial Phase Distribution obtained under FFT method

IV. Zero Crossing Rate(ZCR) and Wavelet Transform(WT)

The Zero-Crossing Rate (ZCR) is a simple method primarily used to extract frequency, with limited ability to extract phase. It works on the fundamental principle that for a simple, periodic signal (like a sine wave), the

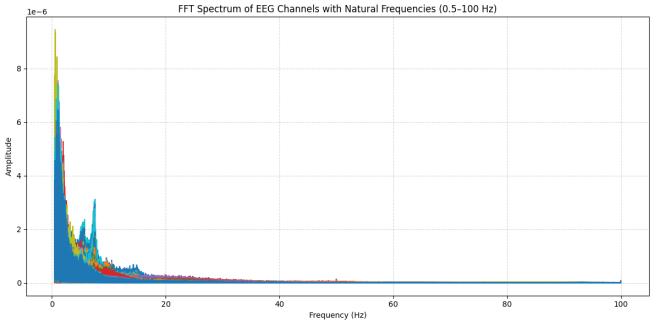


FIG. 8: Frequency Spectrum obtained under FFT method

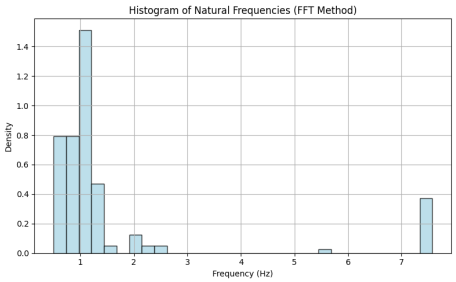


FIG. 9: Natural Frequencies obtained under FFT method

number of times it crosses the zero-amplitude axis is directly related to its frequency. To find the frequency, one counts the number of zero-crossings within a specific time window and divides that by the time period. For example, a 100 Hz sine wave will cross zero 200 times per second (once from positive to negative, and once from negative to positive, per cycle). The ZCR is a good, low-complexity method for analyzing narrow-band signals but can be unreliable for complex or noisy signals that may have multiple zero-crossings per cycle. While not designed to directly measure phase, the initial phase of a simple sine wave can be inferred from the time of the first zero-crossing relative to the start of the time window.

The Wavelet Transform (WT) is a powerful tool for time-frequency analysis that can accurately extract both frequency and phase from a signal, even if it is non-stationary. Unlike the FFT, which provides a global frequency spectrum, the WT uses "wavelets"—short-lived, oscillating functions—to analyze a signal at different scales (frequencies) and positions (time). The WT computes a set of "wavelet coefficients" that represent the similarity between the wavelet and the signal at each scale and time.

The frequency information is encoded in the scale of the wavelet. Large scales correspond to low frequencies (stretched wavelets), and small scales correspond to high frequencies (compressed wavelets). By plotting the

wavelet coefficients against both time and scale, one obtains a time-frequency representation that shows how the frequency content of the signal changes over time.

Phase information is directly embedded in the complex-valued wavelet coefficients. For a complex wavelet transform, the phase at any given time and scale is simply the angle of the complex wavelet coefficient. This allows for the extraction of the instantaneous phase of a signal's components at any point in time.[3]

Python script analyzes the initial phase of EEG signals using the Continuous Wavelet Transform (CWT) with a complex Morlet wavelet. The process iterates through each EEG channel, and for each signal, it applies `scipy.signal.cwt` with the `morlet2` wavelet, centered around a `target_freq`. The phase is extracted from the resulting complex wavelet coefficients using `np.angle`, from which the initial phase at the first time point is isolated. This value is then wrapped to the standard $[0, 2\pi)$ range using `np.mod`. The collected initial phases are stored in a `numpy` array, which is then visualized with `matplotlib.pyplot` in a polar scatter plot(FIG. 10) to represent the phases as points on a unit circle.

After that, Python script analyzes EEG signals using a Zero-Crossing Rate (ZCR) method to estimate frequency and explores properties related to synchronization. The core frequency estimation is performed by the `natural_frequency_zcr` function, which counts the number of times a signal crosses zero to derive its frequency. The script then iterates through each EEG channel and applies this ZCR method to overlapping windows of the signal, building a frequency distribution for each channel. `scipy.signal.find_peaks` is used to identify the dominant "natural" frequency from the most prominent peak in each channel's frequency spectrum. Furthermore, the script analyzes the collected natural frequencies using `scipy.stats.gaussian_kde` to estimate the probability density function $g(\omega)$, which is a crucial parameter for calculating the theoretical critical coupling strength (ϵ_c) in a Kuramoto model of coupled oscillators. Finally, `matplotlib.pyplot` is used to visualize the individual channel spectrum(FIG. 11) and a histogram of the natural frequencies(FIG. 12).

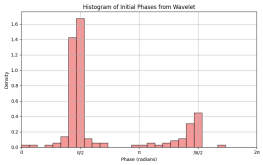


FIG. 10: Intial Phase Distribution obtained under Wavelet Transform method

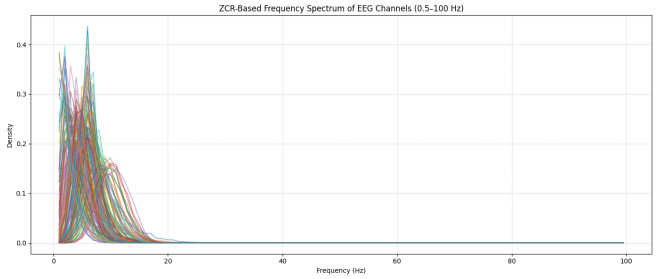


FIG. 11: Frequency Spectrum obtained under ZCR method

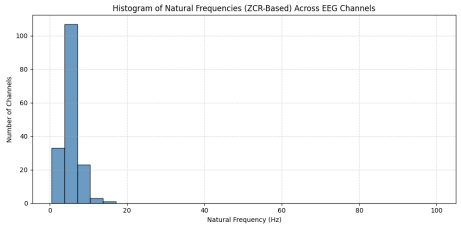


FIG. 12: Natural Frequencies obtained under ZCR method

V. Power Spectral Density(PSD) and Phase Locked Loop(PLL)

Power Spectral Density is a mathematical function that describes how the power of a signal is distributed over frequency. It is typically calculated using the Fast Fourier Transform (FFT) of the signal, and its magnitude at each frequency gives a measure of the power or energy of that frequency component, effectively revealing the signal's dominant frequencies. However, PSD is an average over the entire time window and does not directly provide instantaneous phase information. In contrast, a Phase-Locked Loop is a feedback control system used to track both the frequency and phase of a real-time signal. A PLL works by continuously comparing the phase of a reference signal with the phase of its own internal voltage-controlled oscillator (VCO). Any phase difference (or "phase error") generates a control voltage that is fed back to the VCO, adjusting its frequency until its phase locks onto the reference signal's phase. When the loop is locked, the VCO's output signal tracks the input signal's frequency and phase, and the control voltage to the VCO serves as a proxy for the instantaneous frequency.[5]

The Python script extracts the initial phase of EEG signals by simulating a digital Phase-Locked Loop (PLL), with performance accelerated by the `@njit` decorator from `numba`. The core logic is contained within the `pll_frequency` function, which iteratively updates a voltage-controlled oscillator's (VCO's) frequency and phase by comparing it to the input signal to generate a phase error. The main script then iterates through each

EEG channel, using a pre-calculated dominant frequency as the PLL's center frequency for each channel. After the PLL has locked onto the signal, the initial phase is approximated by taking the cumulative sum of the instantaneous frequency series and extracting the phase at the last time point, which is then wrapped to the $[0, 2\pi)$ range. Finally, `matplotlib.pyplot` is used to visualize the distribution of these initial phases with a polar scatter plot(FIG.13).

Then Python script analyzes the dominant "natural" frequencies of EEG signals using the Power Spectral Density (PSD) method. The process involves iterating through each EEG channel, where the script uses `scipy.signal.welch` to compute the PSD of the signal. The dominant frequency within a specified 0.5 to 100 Hz band is identified by using `scipy.signal.find_peaks` to locate the highest power peak. If no clear peaks are found, a power-weighted average frequency is calculated. These natural frequencies are collected into a `numpy` array, and the script then uses `matplotlib.pyplot` to generate two plots: a `semilogy` plot displaying the individual PSD spectrun(FIG. 14) of all channels and a histogram(FIG. 15) showing the overall density distribution of the extracted natural frequencies.

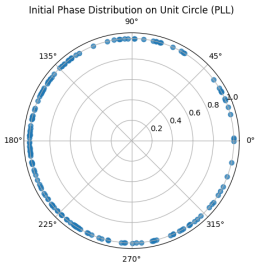


FIG. 13: Intial Phase Distribution obtained under PLL method

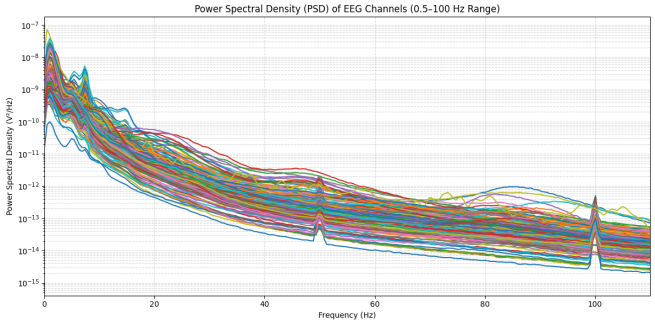


FIG. 14: Frequency Spectrum obtained under PSD method

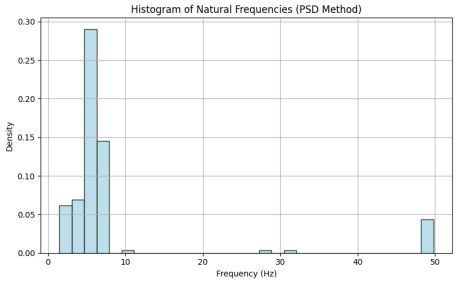


FIG. 15: Natural Frequencies obtained under PSD method

VI. Auto Regressive (AR) Model

In an AutoRegressive (AR) model, extracting frequency and phase is an indirect process based on the model's coefficients. The model works by representing a signal as a weighted sum of its past values. The weights, or coefficients, hold the key to the signal's underlying dynamics. To find the frequencies and phases, the model's characteristic polynomial is constructed from these coefficients, and its roots are computed. These roots are complex numbers, and their position on the complex plane reveals the oscillatory behavior of the signal. Specifically, the angle of each root corresponds to a specific frequency component, while the magnitude of the root is related to that component's amplitude. By analyzing the angles and magnitudes of these roots, the AR model can estimate the key frequencies and their associated phases that make up the original signal.

A Python script is used to extract natural frequencies and initial phases from EEG signals by modeling them with an AutoRegressive (AR) process. The core functionality is contained within the `extract_ar_features` function, which first automatically selects an optimal model order using `statsmodels.tsa.ar_model.ar_select_order`. After fitting the AR model, it calculates the roots of the characteristic polynomial and derives the frequencies and phases from their angles and magnitudes, filtering for stable roots with a magnitude less than one. The script then computes a full amplitude spectrum and uses the `scipy.signal.find_peaks` function to identify the most prominent peak, which is defined as the natural frequency. This entire process is executed in parallel for all EEG channels using `joblib.Parallel` for efficiency, and the results are then visualized with a polar plot of the initial phases(FIG. 16), a collective plot of the AR frequency spectrum in Fig. 17, and a histogram of the natural frequencies(FIG. 18).

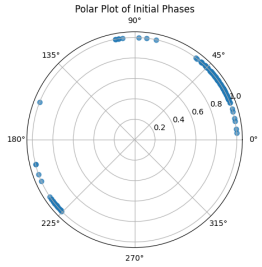


FIG. 16: Intial Phase Distribution obtained under AR Model

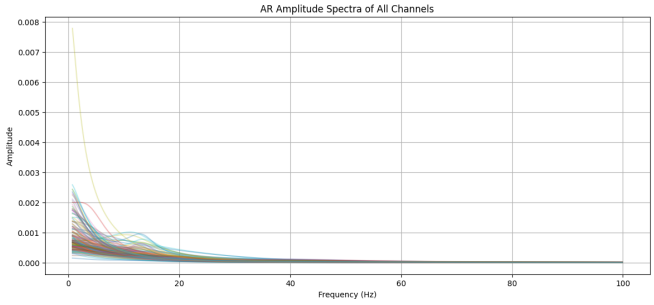


FIG. 17: Frequency Spectrum obtained under AR Model

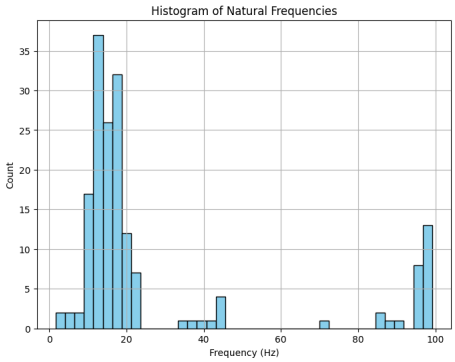


FIG. 18: Natural Frequencies obtained under AR Model

Kuramato Model and Transverse Lyaponov Exponents(TLE) values

critical coupling strength is estimated (ϵ_c) for a Kuramoto model based on the distribution of natural frequencies, denoted by the `numpy` array `omega`. First, `scipy.stats.gaussian_kde` is used to create a Kernel Density Estimate (KDE) of the frequency distribution, which serves as a smooth representation of the density function $g(\omega)$. The code then evaluates this density function and finds its peak value using `np.max`, which is assigned to the variable `g_omega`, corresponding to the maximum density. Finally, the `epsilon_critical` value is calculated using the analytical result from Kuramoto model theory, $\epsilon_c = 2/(\pi g(\omega))$, representing the theoretical threshold for the onset of global synchronization. The provided Python code generates a `numpy` array named `epsilons` to define a range of coupling strengths for a Kuramoto model simulation. It uses the `np.linspace` function to create an array of 5 evenly spaced values. The range of these values is defined from 10% of a pre-calculated `epsilon_critical` value to 90% of that value. This approach sets up a specific range of sub-critical coupling strengths, allowing for a subsequent analysis of the system's behavior as it approaches the theoretical threshold for global synchronization without fully reaching it.

Then classical Kuramoto model is simulated to observe the effect of coupling strength (ε) on oscillator synchronization. The simulation is driven by an outer `for` loop that iterates through a range of `epsilon` values. For each value, the script runs a nested `for` loop for `T_sim` time steps. Inside this loop, it updates the phase of each oscillator by adding its natural frequency to a coupling term, which is efficiently computed using `np.sum` on the sinusoidal phase differences between all oscillators. The phase state of all oscillators at each time step is stored in a `phase_history` array. After the simulation, the final phase distribution is extracted from the last row of this array, and `matplotlib.pyplot` is used to create a polar scatter plot. The phases are first wrapped to the $[0, 2\pi)$ range using the modulo operator to accurately visualize their final arrangement on a unit circle.

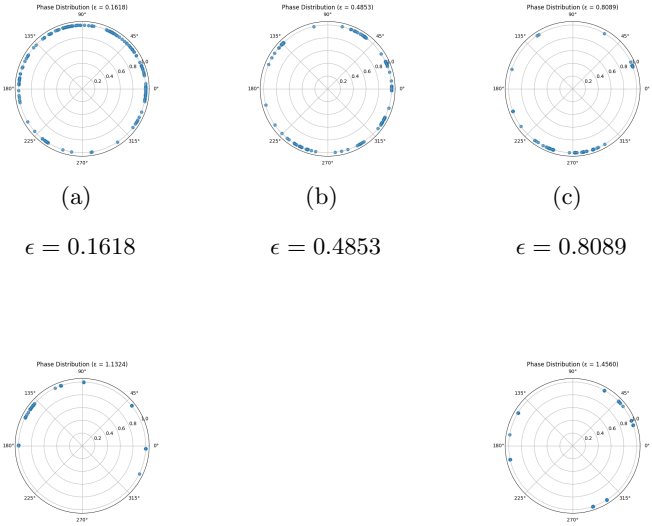
The simulation was conducted over a total of $T_{sim} = 1500$ time steps, each with a size of $dt = 0.01$ units. This results in a total simulation time of $1500 \times 0.01 = 15$ units.

Then Python script calculates a set of values, likely corresponding to the transient Lyapunov exponents (TLEs) of a Kuramoto model, by iterating through a network of oscillators. The code initializes a `numpy` array named `TLEs` and then uses a nested `for` loop structure. The outer loop iterates through each oscillator `k`, while the inner loop computes a sum of the cosines of the phase differences between oscillator `k` and every other oscillator `j`. An `if` condition ensures that an oscillator is not included in its own sum. Finally, the value for each oscillator `k` is determined by multiplying this sum by `-(epsilon / N)` and for this `epsilon` is the half of the `epsilon_critical` value is used. The TLE are stored in the `TLEs` array, providing a measure of the collective influence of all other oscillators on its phase dynamics.

RESULTS

Phase Variation Under Kuramato Model

I. Hilbert Transform



(a) $\epsilon = 0.1618$ (b) $\epsilon = 0.4853$ (c) $\epsilon = 0.8089$

(d) $\epsilon = 1.1324$ (e) $\epsilon = 1.4560$

FIG. 19: Phase variation of the Kuramato model simulation under signal analysis done by Hilbert Transform

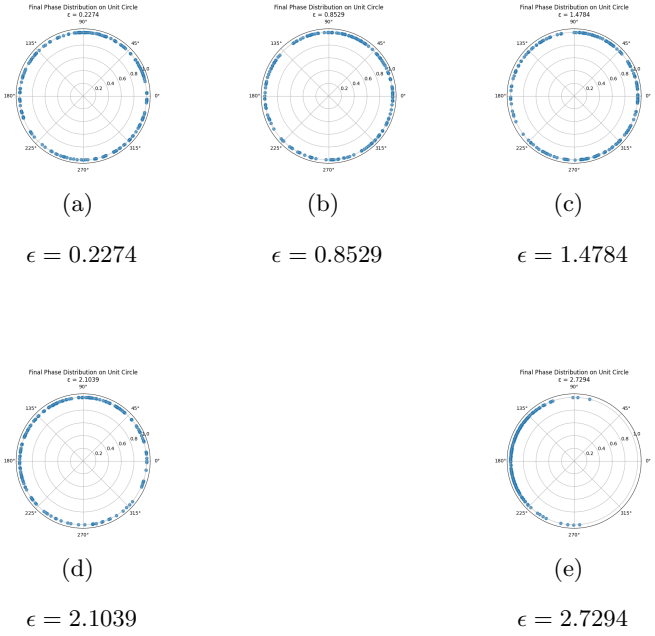
As shown in FIG. 19 the parameter ϵ increases from 0.1618 to 1.4560, the phase distribution undergoes a clear transition from a high distributed state to low variation state to distinct phase clusters.

II. Hilbert-Huang Transform (HHT) and Empirical Mode Decomposition (EMD)

FIG. 20 shows as ϵ increases from 0.2274 to 2.7294, the phase variation changes, but in a different way than the Hilbert plots. Instead of increasing randomness across the entire circle, the EMD plots show a transition from a multi cluster distribution to a distribution concentrated in one large, continuous arc.

III. Fast Fourier Transform

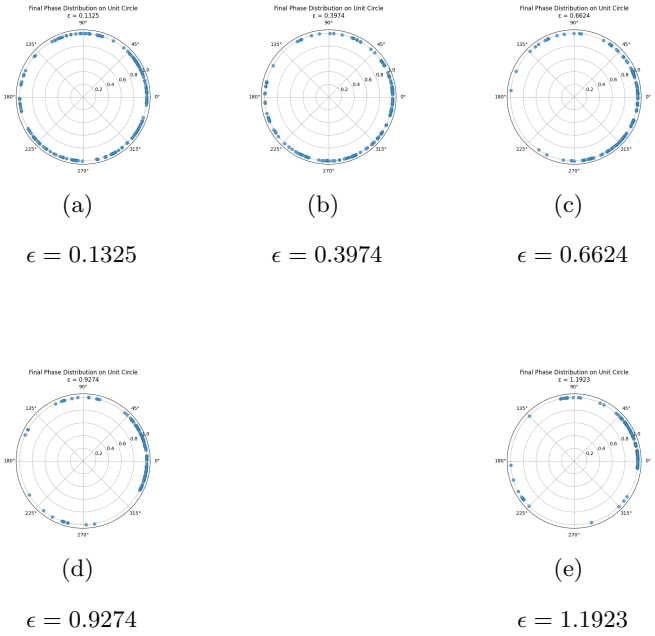
As FIG. 21 the parameter ϵ increases from 0.1325 to 1.1923, the phase distribution transitions from a highly ordered, multi-clustered state to a state with lower phase variation. While most of the phases are diffuse and random, a key feature is that one specific 0° to 90° phase



(a) $\epsilon = 0.2274$ (b) $\epsilon = 0.8529$ (c) $\epsilon = 1.4784$

(d) $\epsilon = 2.1039$ (e) $\epsilon = 2.7294$

FIG. 20: Phase variation of the Kuramato model simulation under signal analysis done by EMD and HHT method



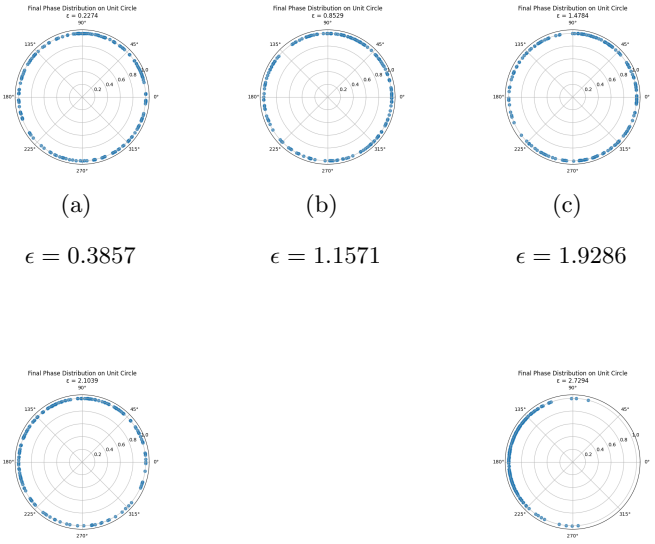
(a) $\epsilon = 0.1325$ (b) $\epsilon = 0.3974$ (c) $\epsilon = 0.6624$

(d) $\epsilon = 0.9274$ (e) $\epsilon = 1.1923$

FIG. 21: Phase variation of the Kuramato model simulation under signal analysis done by FFT method

value remains a dominant, tightly clustered state at the highest ϵ value.

IV. Zero Crossing Rate(ZCR) and Wavelet Transform(WT)



(a) $\epsilon = 0.3857$ (b) $\epsilon = 1.1571$ (c) $\epsilon = 1.9286$

(d) $\epsilon = 2.7000$ (e) $\epsilon = 3.4714$

FIG. 22: Phase variation of the Kuramato model simulation under signal analysis done by ZCR and WT method

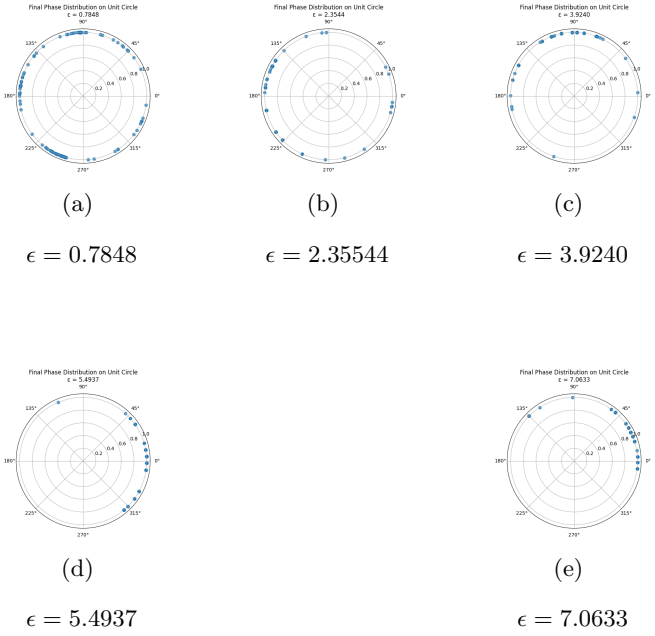
As appear in FIG. 22 the parameter ϵ increases from 0.3857 to 3.4714, the phase distribution undergoes a clear transition from a highly ordered state on to a clustered state. The phase values become less confined to specific 90° to 270° region.

V. Power Spectral Density(PSD) and Phase Locked Loop(PLL)

FIG. 23 shows the parameter ϵ increases from 0.7848 to 7.0633, the phase distribution undergoes a significant change. The system transitions from a multi-stable state with numerous clusters to an irregular phase distribution with several clusters, and also the number of active phase values appears to decrease.

VI. Auto Regressive (AR) Model

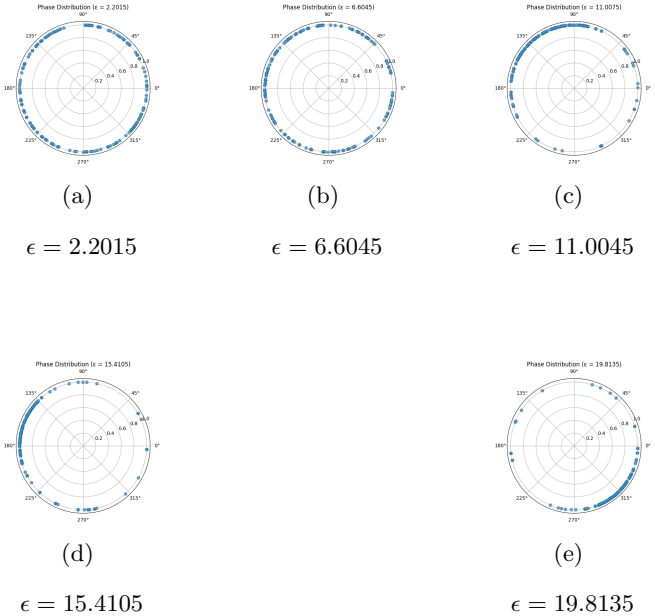
As in the FIG. 24 the progression of phase variation as ϵ increases from 0.1618 to 19.8135, the plots show a clear transition from an ordered, multi-clustered state to



(a) $\epsilon = 0.7848$ (b) $\epsilon = 2.35544$ (c) $\epsilon = 3.9240$

(d) $\epsilon = 5.4937$ (e) $\epsilon = 7.0633$

FIG. 23: Phase variation of the Kuramato model simulation under signal analysis done by PSD and PLL method



(a) $\epsilon = 2.2015$ (b) $\epsilon = 6.6045$ (c) $\epsilon = 11.0045$

(d) $\epsilon = 15.4105$ (e) $\epsilon = 19.8135$

FIG. 24: Phase variation of the Kuramato model simulation under signal analysis done by AR Model

a state where the phase is highly constrained to a single, large, continuous arc in the region of 270° to 360°.

The initial phase of a system is not the primary factor in determining critical transitions. Instead, the critical epsilon value, ϵ_c, is governed by frequency variation. The provided formula, $\epsilon_c = \frac{2}{\pi g(\omega)}$, highlights this relationship. Here, ω represents the frequency and $g(\omega)$ is the frequency distribution function. This equation shows that the critical epsilon value is inversely proportional to the frequency distribution. Therefore, for a system to undergo a critical transition (indicated by a change in phase behavior from ordered to disordered), the coupling strength, ϵ, must reach a specific threshold defined by the system's inherent frequency characteristics. The more concentrated the frequency distribution (a large $g(\omega)$), the lower the required coupling strength (ϵ_c) to induce a transition. Conversely, a broad frequency distribution (a small $g(\omega)$) requires a higher coupling strength to achieve a similar transition. This means that the diversity of frequencies within a system is the key determinant of its stability against external perturbations or internal coupling. Table I show resultant $g(\omega)$) and (ϵ_c) values.

TABLE I: Estimated Critical Coupling Values from Different Signal Analysis Methods

Method	**Estimated** $g(\omega)$	**Critical Coupling** ϵ_c
Hilbert Transform	0.3972 ± 0.0310	1.6124 ± 0.1262
EMD and HHT	0.3163 ± 0.0274	2.0277 ± 0.1752
FFT	0.4898 ± 0.0589	1.3179 ± 0.1534
ZCR and WT	0.1695 ± 0.0137	3.7801 ± 0.3003
PSD and PLL	0.0831 ± 0.0119	7.8037 ± 1.0345
AR Model	0.0292 ± 0.0028	22.0116 ± 2.0803

Transverse Lyaponov Exponent(TLE) Value and Epileptogenic Results Comparison

The following table presents the results of the different signal processing methods, focusing on their ability to accurately identify Epilepsy. It highlights the number of correct and accuracy percentage for each method.

TABLE II: Performance Metrics of Classification Methods

Method	**Correctly Classified**	**Accuracy (%)**
Hilbert Transform	74	43.27
EMD and HHT	80	46.78
FFT	101	59.06
ZCR and WT	71	41.52
PSD and PLL	85	49.71
AR Model	56	32.75

The provided data reveals significant differences in the performance of various signal processing methods for classifying Epilepsy. The analysis of the accuracy and error percentages, as summarized in Table II, highlights the effectiveness of each approach.

The FFT method demonstrates the highest classification accuracy at 59.06%, with a corresponding error rate of 40.94%. This method achieved the best balance of correct classifications, indicating its superior ability to accurately identify both Epileptogenic and Non-Epileptogenic cases compared to the other techniques.

Conversely, the AR Model shows the lowest performance, with a strikingly low accuracy of 32.75% and a very high error rate of 67.25%. A closer look at its cross-tabulation reveals a major weakness: it has an exceptionally high number of false positives (104 cases), meaning it frequently misclassifies Non Epileptogenic instances as Epileptogenic.

The other methods fall in the middle of this range. The EMD and HHT method and the PSD and PLL method show moderate performance, with accuracies of 46.78% and 49.71%, respectively. Both methods still exhibit a high number of false positives, which negatively impacts their overall accuracy. The Hilbert Transform and the ZCR and WT method also struggle, with accuracies of 43.27% and 41.52%, respectively, primarily due to a large number of false positive classifications.

In conclusion, while all methods attempt to classify the condition based on the computed critical coupling values, the FFT method is the most reliable for this specific classification task. The results suggest a strong correlation between the method used for frequency characterization and the resulting classification accuracy, with methods like the AR model proving to be highly unreliable due to their tendency for high error rates.

While all methods attempt to classify the condition based on the computed critical coupling values, the FFT method is the most reliable for this specific classification task. It is important to note, however, that due to high computational costs, the AR Model and EMD and HHT methods were run on a reduced dataset of only 27,000 data points. This constraint likely affects their results, potentially leading to a less robust or representative model and an underestimation of their true performance. Despite this limitation, the AR model's exceptionally low accuracy suggests it may be fundamentally less suited for this task. The results as presented suggest a strong correlation between the method used for frequency characterization and the resulting classification accuracy, with the FFT method providing the most promising results.

CONCLUSIONS

The analysis reveals that as the parameter ϵ increases, the phase distribution of the system undergoes distinct

transitions depending on the signal analysis method used. For the Hilbert Transform, increasing ϵ from 0.1618 to 1.4560 results in a transition from a highly distributed state to a low-variation state with distinct phase clusters, whereas for the EMD method, an increase in ϵ from 0.2274 to 2.7294 leads to a transition from a multi-cluster distribution to a single, continuous arc. Similarly, the FFT method shows a transition from a multi-clustered state to a lower-variation, diffuse state where a dominant cluster remains in the 0° to 90° region as ϵ increases from 0.1325 to 1.1923. The ZCR and WT method shows a transition from an ordered to a clustered state as ϵ increases from 0.3857 to 3.4714, with phase values becoming less confined to the 90° to 270° region. For the PSD and PLL method, as ϵ increases from 0.7848 to 7.0633, the system transitions to an irregular, sparse distribution. Finally, the AR model shows a transition from an ordered state to a single, continuous arc in the 270° to 360° region as ϵ increases from 0.1618 to 19.8135. The critical epsilon value (ϵ_c) is found to be inversely proportional to the frequency distribution function $g(\omega)$, with the FFT method having the highest $g(\omega)$ (0.4898) and lowest ϵ_c (1.3179), while the AR model has the lowest $g(\omega)$ (0.0292) and highest ϵ_c (22.0116). In terms of classification accuracy for identifying epileptogenicity, the FFT method proved most reliable with 101 correct classifications and an accuracy of 59.06%, while the AR model was the least accurate with only 56 correct classifications and 32.75% accuracy. It is important to note that the EMD and AR methods were evaluated using a reduced dataset of 27,000 data points due to high computational costs, which may have impacted their results and potentially underestimated their true performance.

REFERENCES

[1] T. Matteuzzi et al., *arXiv preprint arXiv:2501.00079* **2024**, Accessed July 2025.

[2] N. E. Huang et al., *Proceedings of the Royal Society of London. Series A: Mathematical Physical and Engineering Sciences* **1998**, *454*, 903–995.

[3] A. A. Torres-García et al. in Insert Publisher Name, Insert Address, **Insert Year**, Chapter 4.

[4] M. Jones in *Building Valve Amplifiers*, Newnes, **2014**, pp. 235–380.

[5] N. Haque et al. in Proceedings of the World Congress on Engineering 2010, *Vol. II*, London, U.K., **2010**.