Report – Research Work from 8th January to 10th January, 2024, upto 15th January

Contents

Feedback Received on 8th Jan, 2024.:

- Generate Synthetic Timeseries Sinusoidal Data and record 1000 time points.
- Keep Nyquist Theorem in mind when sampling.
- Generate Welsh plots (Power Spectrums) for this compound data. Analyse this and understand what the meaning of the plot is.
- Give a write up.
- Extract 21 channel data of annotated regions of 1 subject; concatenate and generate Welsh plots.
- Write an analysis.
- Report on 10-01-2024.
- Read up on the 4 research papers. Make more slides. Write down the theory with references.

1. Generate Synthetic Time-Series Sinusoidal Data

- **Objective:** Create a dataset with 1000 time points, adhering to the Nyquist Theorem.

- **Steps:**

Determine the frequency range for sinusoidal data.

For EEG data, typical brainwave frequencies range from 0.5 Hz (delta waves) to around 60 Hz (beta waves). Let's assume we want to cover this range.

Ensure the sampling rate is at least twice the highest frequency (Nyquist Theorem).

Sampling rate is set to nearest century greater than or equal to twice the maximum frequency

60*2 = 120; sampling set to 200hz.

Use Python to generate the sinusoidal data.

Code was written. Included with report.

Save the data for further analysis.

Was done.

Synthetic EEG Data Generation: Report

- *Introduction*

This report outlines the methodology and implementation details for generating synthetic Electroencephalogram (EEG) data. The objective was to create a realistic simulation of EEG signals for 21 channels, each comprising a mixed signal of multiple pure sinusoidal waves. This synthetic data serves as a critical tool for testing EEG analysis methods, particularly focusing on the power spectrum.

- *Objectives*

➢ **Simulate Realistic EEG Data:** Generate data that closely mimics the characteristics of real EEG signals.
➢ **Multiple Channels:** Produce signals for 21 distinct channels, reflecting standard multi-channel EEG recordings.
➢ **Frequency and Amplitude Variation:** Each channel's signal consists of 12 to 200 sinusoidal waves with varied frequencies and amplitudes.
➢ **Data Accessibility:** Store the generated data in a format that allows for easy access and analysis.

- *Methodology*

Channel Configuration

- **Total Channels:** 21
- **Sinusoidal Components per Channel:** Randomly chosen between 12 and 8000.

Frequency and Amplitude Selection

- **Frequency Range:** Each sinusoidal wave within a channel has a frequency randomly chosen between 0.5 Hz and 60 Hz, encompassing typical EEG brainwave frequencies.
- **Amplitude Range:** Amplitudes are randomly assigned values between 0.1 and 1.0, introducing variability in signal strength.

Signal Generation

- A compound signal for each channel is generated by summing its sinusoidal components, each with its frequency and amplitude.

Sampling Rate and Duration

- **Sampling Rate:** Set at 200 Hz, ensuring the capture of fine details within the signal and adherence to the Nyquist Theorem.
- **Signal Duration:** 10 minutes, resulting in `120,000 data points` per channel.

Data Storage and Format

- The generated data is stored in a JSON file, named **synthetic_eeg_data.json**.
- This file includes the signal, frequencies, and amplitudes for each channel.
- A JSON schema is provided to define the structure of the stored data.

Json Schema Used :

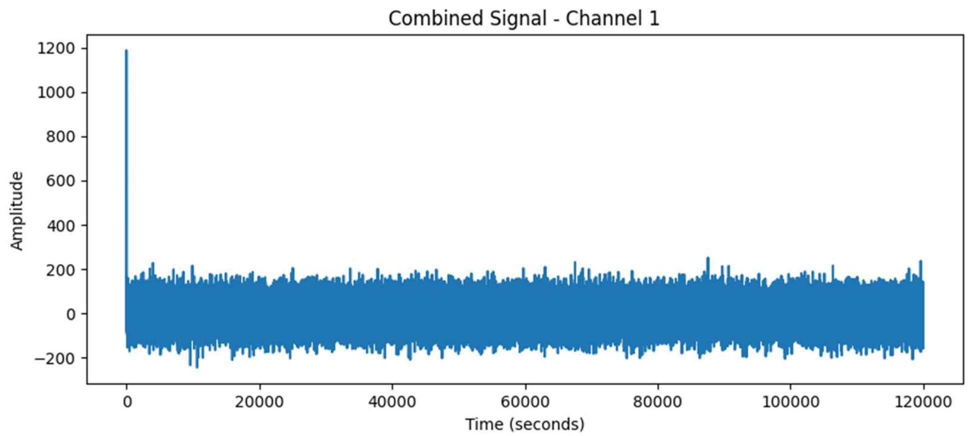
```
{
  "type": "object",
  "properties": {
    "Channel_1": {
      "type": "object",
      "properties": {
        "Signal": { "type": "array", "items": { "type": "number" } },
        "Frequencies": { "type": "array", "items": { "type": "number" } },
        "Amplitudes": { "type": "array", "items": { "type": "number" } }
      },
      "required": ["Signal", "Frequencies", "Amplitudes"]
    },
    "...": {
      "...": "..."
    },
    "Channel_21": {
      "type": "object",
      "properties": {
        "Signal": { "type": "array", "items": { "type": "number" } },
        "Frequencies": { "type": "array", "items": { "type": "number" } },
        "Amplitudes": { "type": "array", "items": { "type": "number" } }
      },
      "required": ["Signal", "Frequencies", "Amplitudes"]
    }
  },
  "required": ["Channel_1", "...", "Channel_21"]
}
```

Further Directions

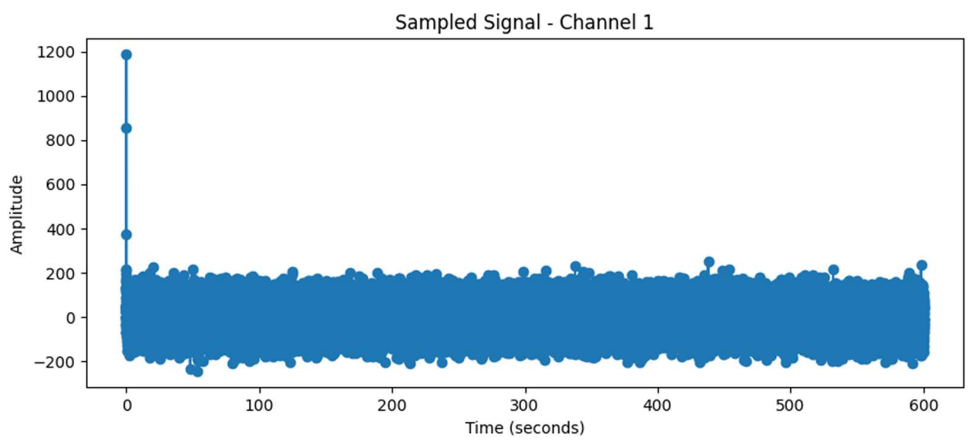
Further enhancements can include the introduction of noise elements and the simulation of specific EEG patterns related to various brain states or disorders. This would provide an even more robust dataset for testing and research purposes.

Outputs of Script

Combined Pure Signal per Channel:



Sampled Signal Per Channel: (200hz Sampling)



Json Files of the Sampled Signal per Channel.

2. Generate and Analyze Welch Plots (log Power spectra)

- **Objective:** Analyze the power spectrum of the synthetic data.
- **Steps:**
  1. Use Python's **scipy.signal.welch** function to compute the power spectral density.
  2. Plot the Welch power spectrum.
  3. Analyze the plot to validate the power spectrum function.

Report : Welch Power Spectrum Analysis of Synthetic EEG Data:

Introduction

This report details the methodology and implementation of a Python script designed to analyze the power spectrum of synthetic EEG data using Welch's method. The goal is to validate the power spectrum function and understand the characteristics of the generated EEG signals.

Objectives:

1. **Compute Power Spectral Density:** Use Welch's method to estimate the power spectral density (PSD) of synthetic EEG signals.
2. **Overlay Plotting:** Compare the generated power spectrum with an expected power spectrum.
3. **Residual Analysis:** Analyze the residuals between the expected and generated power spectra.

Methodology:

Data Loading

- **Source:** The synthetic EEG data is loaded from a JSON file.
- **Format:** Each channel's data includes the signal, frequencies, and amplitudes.

Power Spectrum Analysis

- **Welch's Method:** Employed to estimate the PSD of each channel's signal.
- **Parameters:**
  - **fs**: Sampling frequency set at 200 Hz.
  - **nperseg**: Length of each segment for Welch's method, set at 1024.
- **Advantages of Welch's Method:**
  - Reduces noise in the estimated power spectra.
  - Suitable for analyzing non-stationary and/or short-lived signals, making it ideal for EEG data.

Expected Power Spectrum Computation

- Calculated based on the frequencies and amplitudes provided in the synthetic EEG data.
- Utilizes the square of the amplitude to represent power, adhering to the physical representation of power in signals.

Overlay Plotting

- **Purpose:** To visually compare the generated power spectrum with the expected power spectrum.
- **Implementation:** Plots are created with both spectra overlaid, facilitating direct comparison.

Residuals Analysis

- **Objective:** To quantify the difference between the expected and generated power spectra.

- **Method:** Residuals (differences) are computed and plotted for each channel.
- **Significance:** Helps in identifying discrepancies and validating the accuracy of Welch's method.

Implementation

Script Structure

- Modular design with distinct functions for loading data, computing spectra, plotting, and analysis.
- Use of **if __name__ == "__main__"** for script execution control.
- Documentation comments included for clarity and understanding of each function.
- Progress printouts for tracking the execution and analysis process.

Design Decisions

- **Global Parameters:** Defined for ease of modification and readability.
- **Frequency Range Alignment:** In overlay and residual plots, frequencies are aligned to a full range (0.5 to 60 Hz) for accurate comparison.
- **Residuals Computation:** Highlights any deviation between expected and actual spectra, crucial for validating spectral analysis methods.

Conclusion

The Python script effectively analyzes the power spectrum of synthetic EEG data using Welch's method. By comparing the generated and expected power spectra and analyzing the residuals, the script provides insights into the accuracy and characteristics of the synthetic EEG signals. This approach is instrumental in validating spectral analysis techniques used in EEG studies.

Future Work

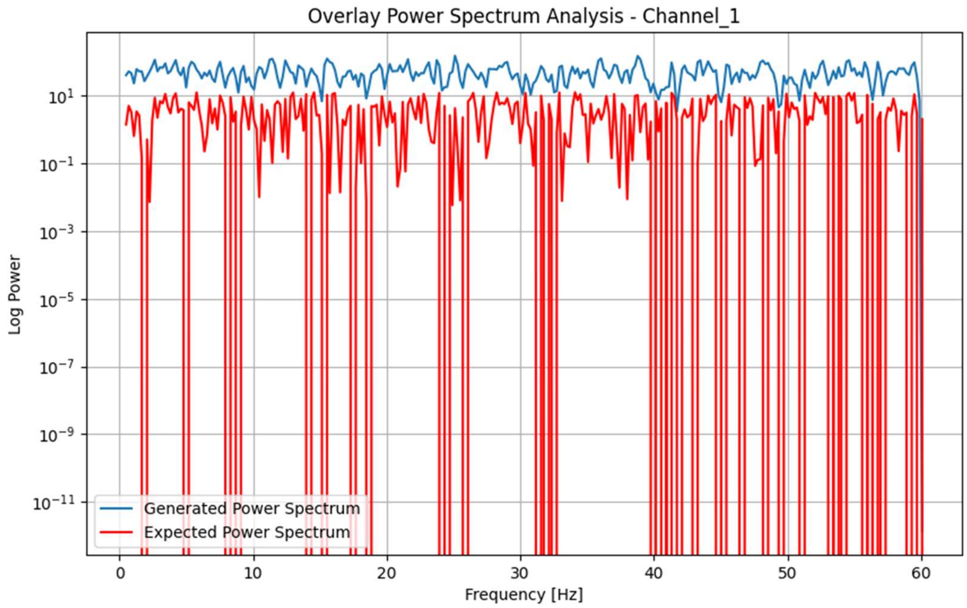
- Incorporating noise elements to simulate more realistic EEG data.
- Extending the analysis to specific EEG patterns related to different brain states or disorders.

Execution

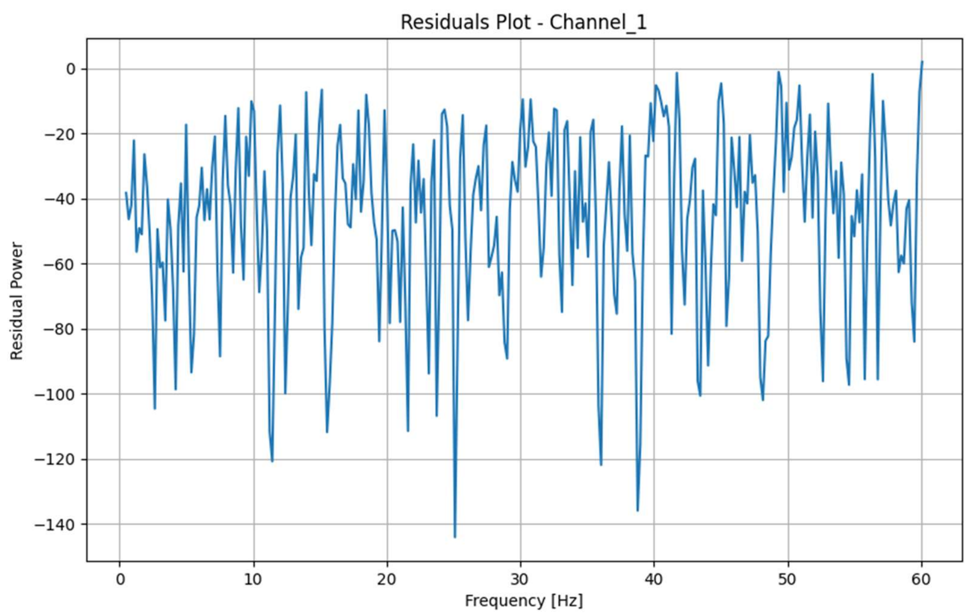
- Ensure the presence of the input JSON file in the specified directory.
- Run the script in a Python environment with necessary libraries installed.
- Output plots and data files are generated for each channel, providing comprehensive insights into the power spectrum analysis.

Outputs of Script

Overlay Plots per channel



Residuals Plot (Expected – Generated) per channel



Json file with Residuals in the same schema per channel

3. Extract and Concatenate EEG Data

- **Objective:** Process real EEG data from sleeping adults and infants.
- **Steps:**
  1. Extract 21-channel data from the provided EEG dataset.
  2. Concatenate the data from different annotated regions for one subject.
  3. Generate Welch plots for this concatenated data.

4. Comparative Analysis

- **Objective:** Compare EEG signals of sleeping adults with infants.
- **Steps:**
  1. Apply the same extraction and concatenation process for both adult and infant data.
  2. Perform a comparative Welch analysis.
  3. Document differences and similarities in the power spectrum.

5. Report Writing

- **Objective:** Compile findings and analysis into a report.
- **Deadline:** 13-01-2024.
- **Steps:**
  1. Summarize the results from synthetic and real EEG data analysis.
  2. Write down the interpretation of Welch plots.
  3. Include comparisons between adult and infant EEG data.
  4. Refer to and cite the four research papers in your analysis.

6. Additional Research

- **Objective:** Enhance understanding and presentation.
- **Steps:**
  1. Read and summarize key points from the four research papers.
  2. Create slides to visually represent the findings and theories.
  3. Include references and citations in the slides.

Tools and Languages:

- **Programming Languages:** Python (for data generation and analysis).
- **Libraries:** NumPy (for data handling), SciPy (for signal processing), Matplotlib (for plotting).
- **Software:** EEGLAB (for EEG data analysis), any slide presentation software.