Sharif University of Technology

Department of Electrical Engineering

Signals and Systems Project

Analysis of Phase Locking Value during Olfactory Stimulation as a Biomarker for Alzheimer's Disease in EEG Signals

---

Instructor: Prof. Hamid Aghajan

Nikoo Moradi

400101934

Spring 2023

2 Electroencephalography (EEG)

2.1 What is EEG?

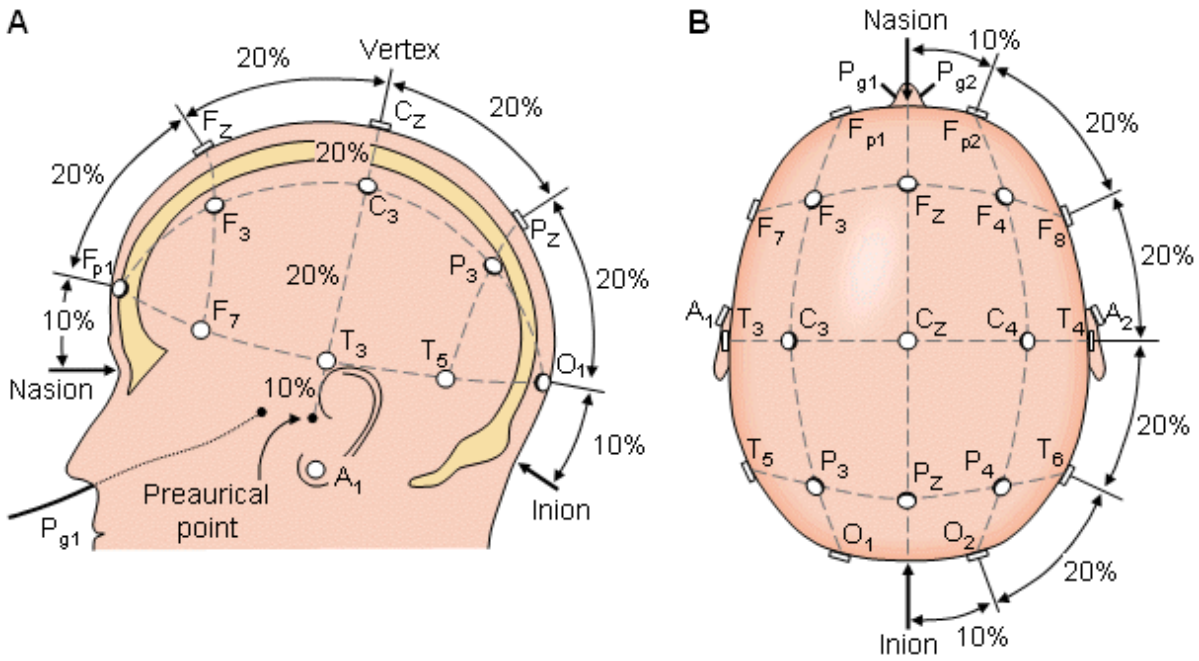


Figure 1: EEG 10-20 Electrode Placement System

- Based on the picture above, What does each electrode's name stand for? Explain the naming method used in the 10-20 EEG system.

The electrodes are labeled with a combination of letters and numbers. The letter indicates the region of the brain where the electrode is located, and the number indicates its specific location within that region. The letters F, T, C, P, and O represent the frontal, temporal, central, parietal, and occipital lobes, respectively. The numbers 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10 are used to denote specific positions within each region. Odd numbers (e.g., F3, C1, P7) typically indicate positions on the left side of the head, while even numbers (e.g., F4, C2, P8) indicate positions on the right side.

2.3 Frequency Bands of EEG

In the frequency domain, EEG signals are divided into 5 bands.

- Determine the activities each frequency band is associated with.

1. Delta (0 - 4 Hz): Delta waves are the slowest brainwaves and are commonly observed during deep sleep or unconscious states. They are associated with restorative processes, physical healing, and the release of

growth hormones. Delta waves are also sometimes seen in certain meditative and deep relaxation states.

2. Theta (4 - 7 Hz): Theta waves are present during light sleep, deep relaxation, daydreaming, and creative states. They are also associated with the early stages of sleep and the REM (rapid eye movement) sleep cycle. Theta waves are often observed during meditation and hypnosis and are linked to memory formation, learning, and emotional processing.

3. Alpha (8 - 12 Hz): Although not mentioned in the frequency ranges provided, it's important to note the presence of the alpha band. Alpha waves are prominent during relaxed wakefulness and are often seen when the eyes are closed but the individual is not asleep. They are associated with a calm and relaxed state of mind, increased creativity, and a focused yet relaxed attention.

4. Beta (12 - 30 Hz): Beta waves are typically present during active and alert mental states. They are associated with focused attention, cognitive tasks, problem-solving, decision-making, and active thinking. Higher frequencies within the beta range are often associated with increased mental activity or stress.

5. Gamma (30 - 100 Hz): Gamma waves are the fastest brainwaves and are associated with high-level cognitive processes, information processing, memory recall, and perception. They have been linked to higher mental functions such as learning, attention, and consciousness. Gamma waves are also observed during states of heightened awareness and intense focus.

2.4 Sampling frequency

- Based on frequency bands and Nyquist criterion, which sampling frequencies are preferred for EEG signals?

For EEG signals, it is recommended to choose sampling frequencies that satisfy the Nyquist criterion to accurately capture the frequency content of the signals. The Nyquist criterion states that the sampling frequency should be at least twice the maximum frequency present in the signal to avoid aliasing.

Considering the frequency bands of EEG signals, the highest frequency of interest is typically in the gamma range, which is around 30 to 100 Hz. To accurately capture these high-frequency components and avoid aliasing, a sampling frequency of at least twice the maximum frequency should be used. Therefore, a conservative approach would be to choose a sampling

frequency of at least 200 Hz or higher for EEG recordings. This allows for accurate representation of the gamma band while providing sufficient sampling resolution for lower-frequency components such as delta, theta, and beta waves. Higher sampling frequencies, such as 500 Hz or even 1000 Hz, can also be used to capture finer details in the EEG signals if required for specific analysis purposes.

3 EEG Signal Processing

3.3 Pre-Processing

For pre-processing I followed the provided pipeline.

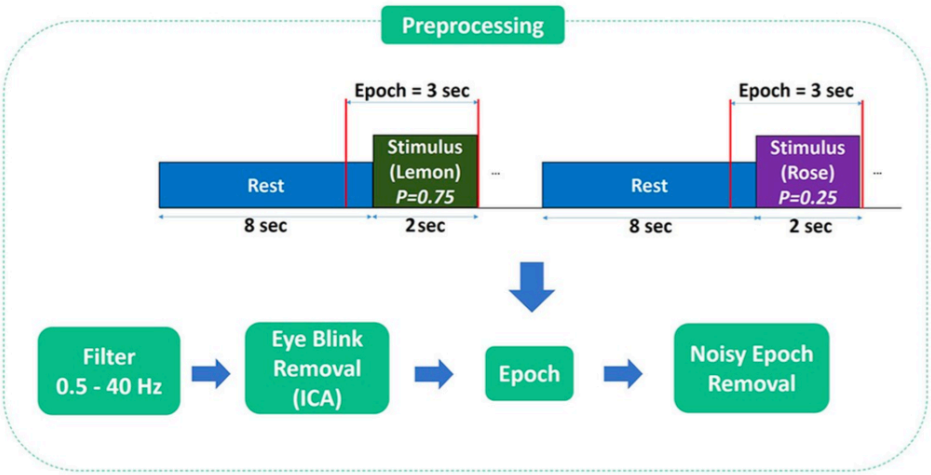


Figure 3: Task and Preprocessing Steps [5]

- **Step 1:** To pre-process using EEGLAB, first re-reference data to the mean of the channels. Then use a bandpass filter to filter 0.5 - 40.5 Hz frequencies. As we have filtered to 40.5 Hz, there is no need to apply a 49.9 - 50.1 Hz notch filter to remove the line noise (However, keep this step in mind as this is a crucial step in EEG signal preprocessing!). Using FFT function or EEGLAB, plot the frequency spectrum of Fz channel data. (Just to note, your data will be saved at EEG struct in MATLAB workspace.)

At first we should cast each subject from table to array in order to be able to use theme in EEGLAB, then we should remove the 20th channel as it is not useful for us. The written code are provided in the next page. After that we go to EEGLAB environment and load our Subjects and then, we delete the first few seconds of each subject using EEGLAB tools (14s for Subject1 and 16.2s for Subject2). After that, we're ready to follow **Step 1** , after re-referencing and bandpassing, we plot the frequency spectrum of Fz channel data for each subject. The results are provided in the next page.

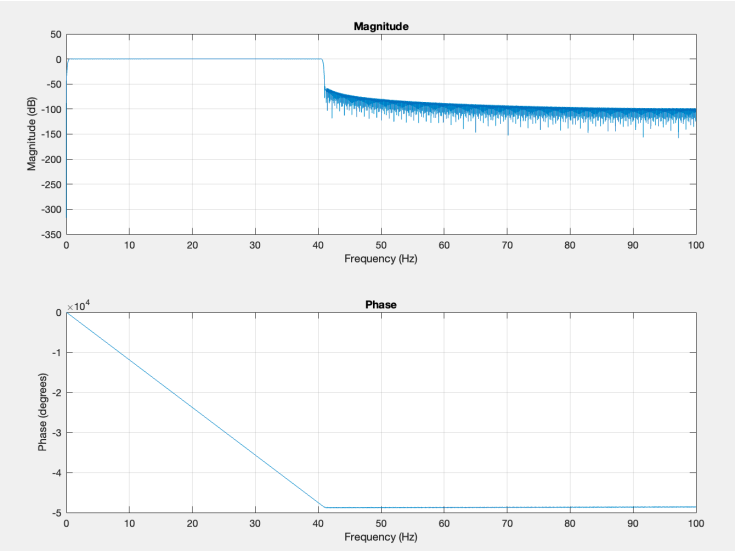
Code for casting Subjects from table to array

```
clear;clc;
% Load the .mat file
load('Subject2.mat');
% Check if the loaded data is a table
if istable(subject2)
    % Convert the table to an array
    data_array = table2array(subject2);
    disp('done');
else
    error('The loaded variable is not a table.');
end
% Compute transpose
transposed_array = transpose(data_array);
% Save transposed array to a new .mat file
save('Subject2_array.mat', 'transposed_array');
```

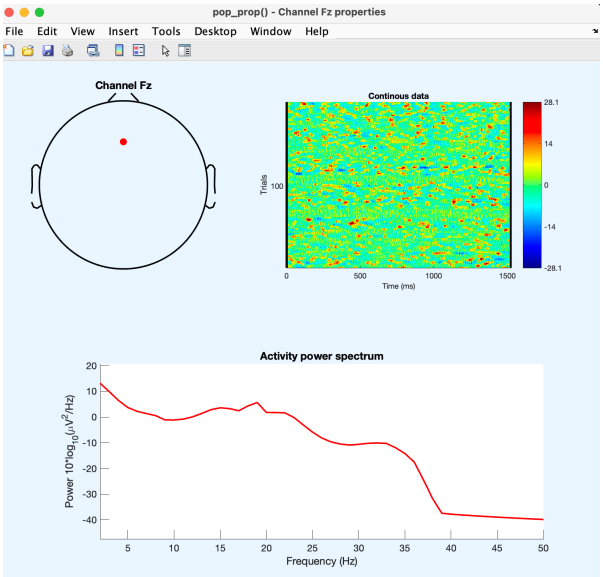
Code for removing the 20th channel

```
clear;clc;
% Load the .mat file
load('Subject2_array.mat');
if exist('transposed_array', 'var')
    % Check the size of the data array
    [numChannels, ~] = size(transposed_array);
    if numChannels == 20
        % Remove the 20th channel
        transposed_array(20, :) = [];
        % Save the modified data to a new .mat file
        save('subject2_array_selected.mat', ...
            'transposed_array');
        disp('done')
    else
        error(['The loaded variable does not ' ...
            'contain 20 channels of data.']);
    end
else
    error('The loaded variable does not exist in the .mat file.');
end
```

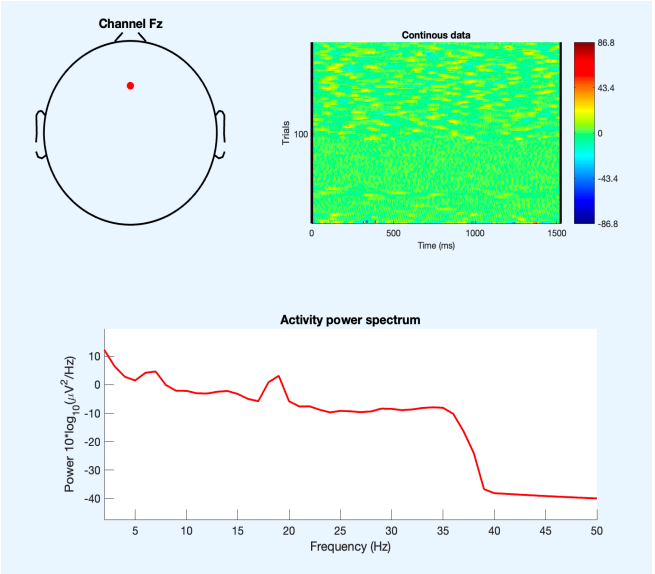
The bandpass filter to filter 0.5 - 40.5 Hz frequencies



frequency spectrum of Fz channel data for Subject 1



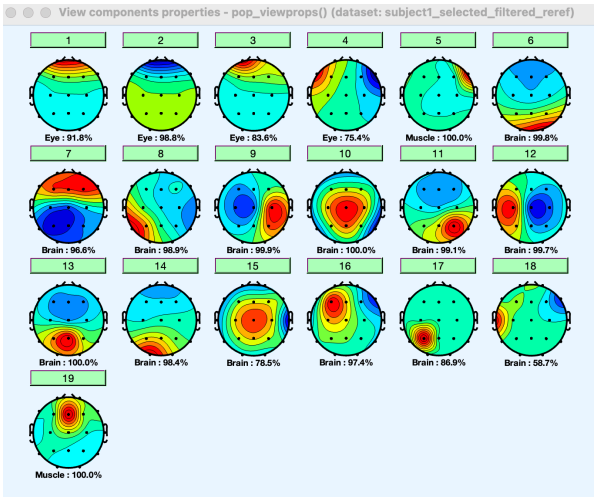
frequency spectrum of Fz channel data for Subject 2



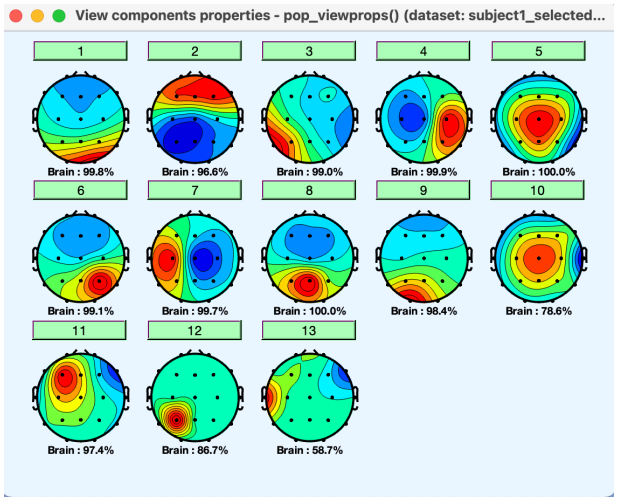
- **Step 2**: In this part you would remove the artifacts of the signal. Artifacts include blinking, eye movement, muscle movement, heart rate and etc. For this, load your data at EEGLAB. Now load Standard-10-20-Cap19.loc file from edit-channel locations menu that contains locations of channels. Then run ICA (Independent Component Analysis) algorithm from tools-decompose data by ICA menu. Please note that this part would probably takes more time. Then you will have the a figure like Figure 4 by running tools → classify components using ICLabel-label components. By clicking on
each component, you can some details about it as well. Present a figure from one of the brain components with its details.
Now remove all non-brain components. For this purpose, from Tools-remove components from data enter the number of components that must be removed.

We follow as was explained in **Step 2**. The results for each subject are shown below :

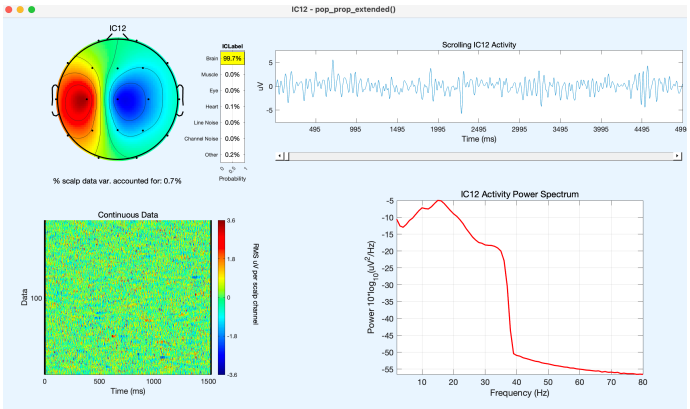
Subject 1 components before removal



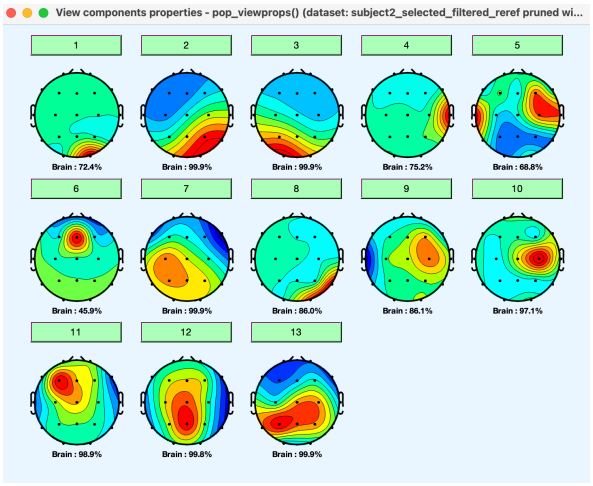
Subject 1 components after removal



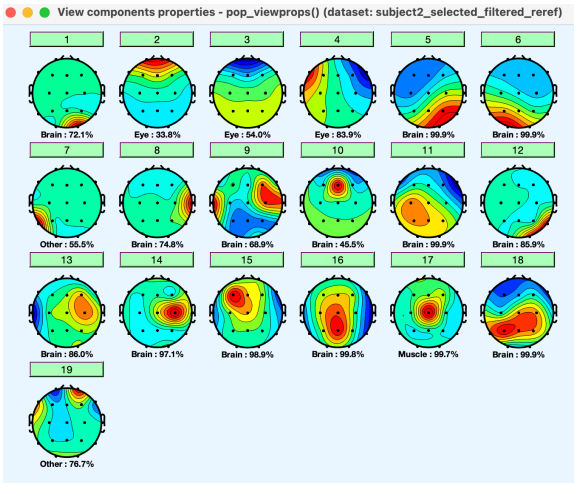
one of the brain components of Subject 1



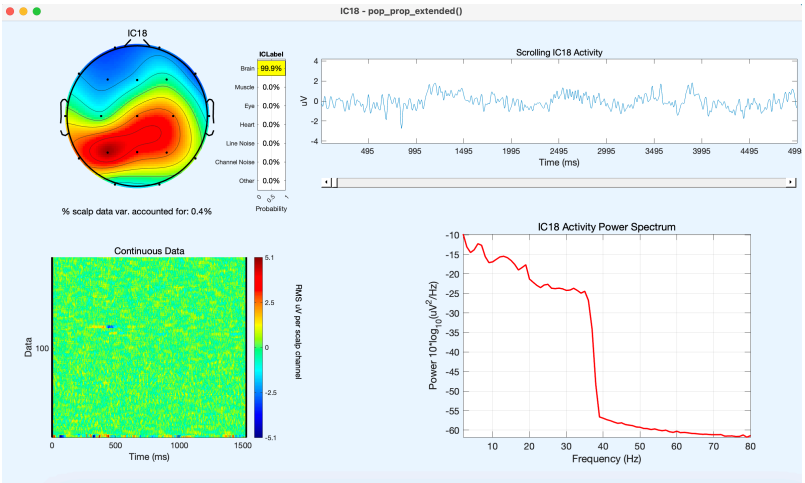
Subject 2 components after removal



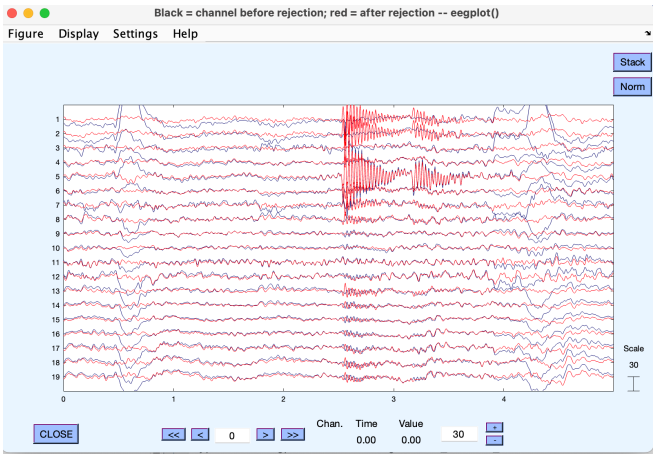
Subject 2 components before removal



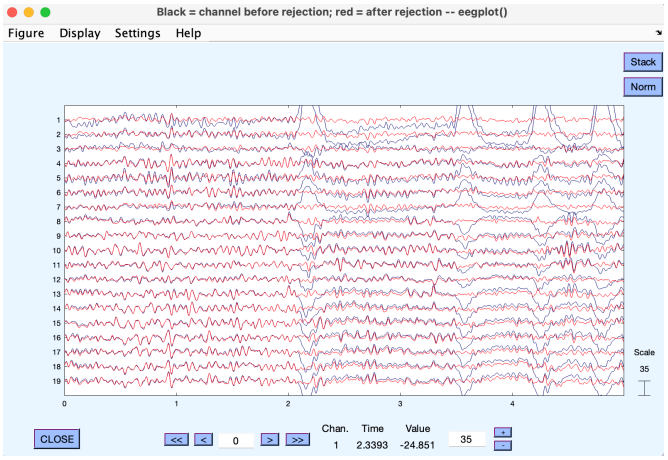
one of the brain components of Subject 2



Subject 2



Subject 1



- Step 3: Epoch the data of each subject. Epoch is a 3D matrix of the shape {Num Channels × Samples × Num Trials}. In fact, all data must be reshaped as the following figure suggests:

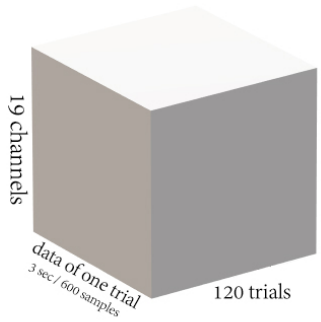


Figure 5: Epoch

  For epoching the data, starting point of the experiment is required. This is provided in the help file for each subject. Please note that you must epoch the data by considering this time as the start. Also, the data after 120 trials should be neglected as well.

For the epoching part, we load the data of each subject (separately) from EEG struct in MATLAB workspace, and then use the following code :

Written code for epoch

```
x = EEG.data;

result = zeros(19, 600, 120);
for i = 1 : 120
    startIndex = (i - 1) * 2000 + 1400;
    endIndex = startIndex + 600 - 1;
    result(:, :, i) = x(:,startIndex:endIndex);
end
save('Subject2_epoched.mat', 'result');
```

As we mentioned before, each trial is a 10min process (10min × 200 = 2000 sample)(first 8min rest, then 2min stimulus, Figure 3). But we just want one last minute of rest and 2 minutes of stimulus Which is 3min × 200 = 600 sample. So we should throw away the first 1400 samples, and store the 600 samples left in a matrix until we have 120 of these 600 samples, which is exactly what we have done in the code snippet above.

- **Step 4:** In this step, you need to remove noisy trials. There are two ways to achieve this:

1. Observe data at EEGLAB and remove any trial that seems noisy. (Preferred!)

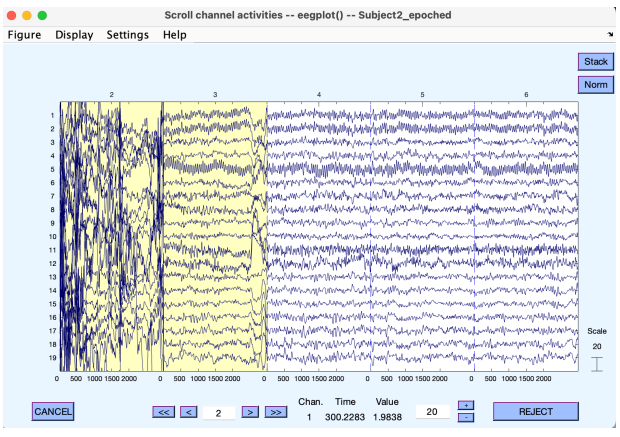
2. Using power spectrum of each trial, remove trials that their standard deviation of their power spectrum is bigger than 3.5 .Create a 3D matrix by each trial's power spectrum for each channel using `pspectrum` in MATLAB. You can use the following commands to find noisy trials:

```
vr = sum(nanstd(p,[],2).^2,2);
noisy_trials = find(abs(zscore(vr)) >3.5);
```

In these commands, p is a matrix of frequency spectrums of all trials of a channel. noisy trials contains the number of noisy trials of that channel. These commands must run for each channel individually and the resultant noisy trials must be accumulated over all channel. Then remove all noisy trials from your epoch.

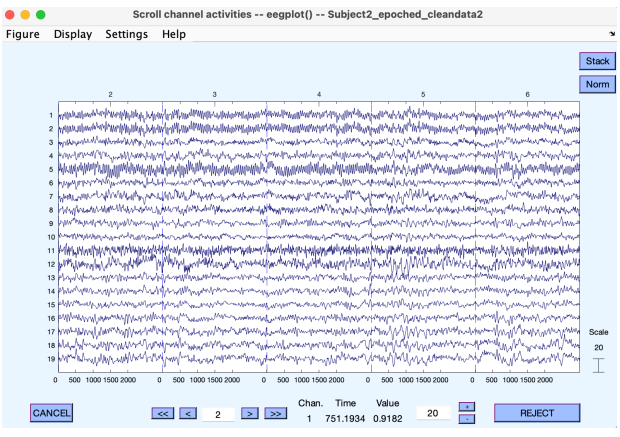
For this part, we removed noisy trials using the first method. We observe each trial and Reject any trial that seems noisy.
We also wrote a code using the the second method to remove noisy trials, but the first method is much easier so we don't explore that here, but you can find the code in *"PreProcessing_maincode.m / step4(optional) section"*
Now we plot both epoched subjects before and after noise removal in EEGLAB and compare the results :

Subject2 before noise removal



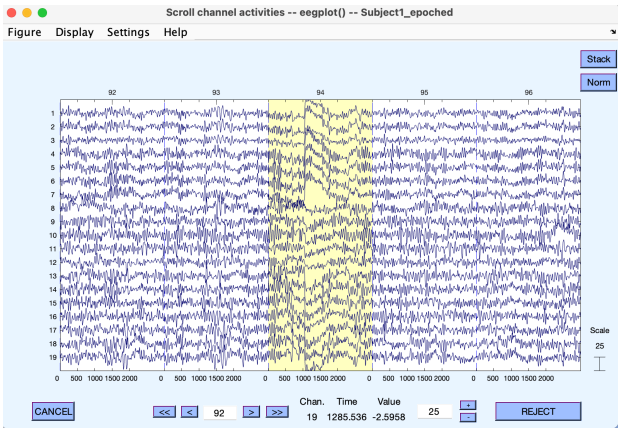
The two noisy trials are selected to be removed

Subject2 after noise removal



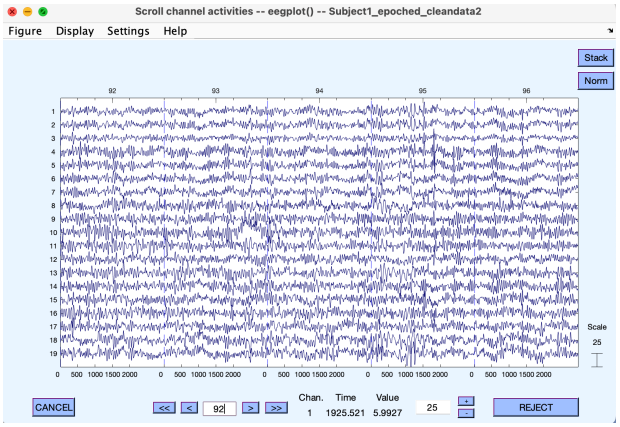
The two noisy trials are removed

Subject1 before noise removal



The noisy trial is selected to be removed

Subject1 after noise removal



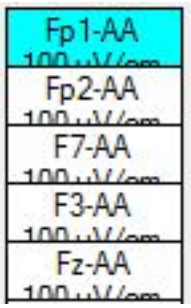
The noisy trial is removed

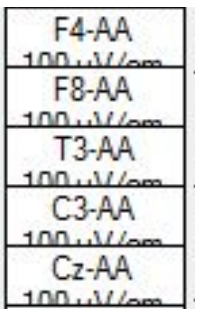
- **Step 5:** In the final step, only subsample the data corresponding to the Fp1, Fz, Cz & Pz channels. You can find the channels' orders in the Channels.jpg.

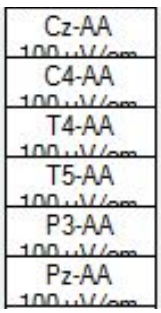
```
%% Step 5: Select and Save channels
Subject1_epoched_cleandata2 = load('Subject1_epoched_cleandata2.mat');
Subject2_epoched_cleandata2 = load('Subject2_epoched_cleandata2.mat');
Subject1_epoched_cleandata2 = Subject1_epoched_cleandata2.x;
Subject2_epoched_cleandata2 = Subject2_epoched_cleandata2.x;
% Define the channels to keep
channelsToKeep = [1, 5, 10, 15];
numChannels = 19;
% Initialize the channels index array with zeros
channelsIdx = zeros(1, numChannels);
% Set the selected channels to 1 in the index array
for i = channelsToKeep
    channelsIdx(i) = 1;
end
% Keep only the desired channels
selectedEpochData = Subject1_epoched_cleandata2(logical(channelsIdx),:,:);
% Save the selected epoch data
save('Subject2_final.mat', 'selectedEpochData');
disp(['Epoch data has been saved to ', 'Subject2_final.mat']);
```

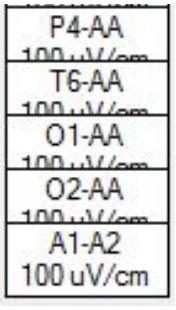
We wrote the following code for this part :

We extracted index of the channels we want to keep, using this map of channels.









Do these 5 steps for each subject and save the final data through an struct with the same format as described in Table 1. Also, consider the order of odor being the same as the ones used for normal participants.

And finally for the last part, we want to save our clean-filtered data as an struct with the same format as described in the table below :

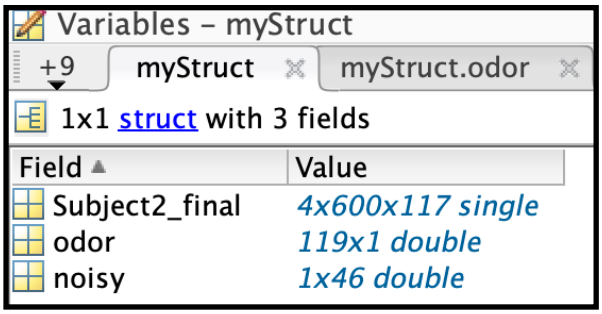
Field	Description
epoch	This is a 3D array structured as 4 × 600 × Num_trials. The first dimension indicates EEG channels respectively from the first column as Fp1, Fz, Cz, and Pz. The second dimension contains EEG samples from 1 s pre stimulus to 2 s post stimulus, which at a 200 Hz sampling rate amounts to 600 samples. The last dimension shows the number of trials. This could be different for each participant as some trials were deleted during preprocessing.
odor	This is a 2D binary array shaped as Num_trial × 1. This array shows the odorant type (lemon/rose) the participant was exposed to in each trial. The value = 1 indicates the rose odor and the value = 0 indicates the lemon odor.
noisy	This is a 2D array with the size 1 × Num_noisy. This array indicates noisy trials identified based on comparing the instantaneous and average trial amplitudes. These noisy trails can be ignored in processing and were included for the dataset completeness.

Table 1: Description of each structure array (.mat file) in the dataset.

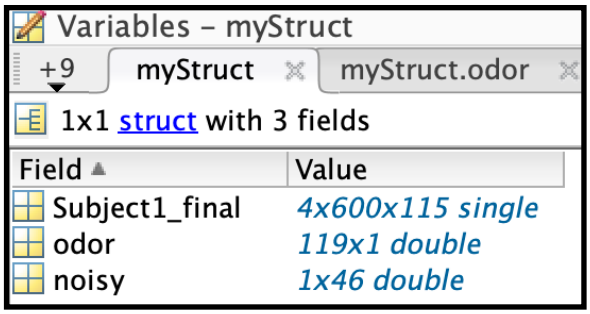
We wrote this code in order to achieve the desired struct :

```
%% Step 5 (Struct): Save Selected Epoch Data
Subject1_final = load('Subject1_final.mat');
Subject2_final = load('Subject2_final.mat');
Subject1_final = Subject1_final.selectedEpochData;
Subject2_final = Subject2_final.selectedEpochData;

Normal = load('Normal.mat','normal');
odor = Normal.normal.odor;
noisy = Normal.normal.noisy;
% Create a struct and assign the matrices as fields
myStruct = struct(...
    'Subject2_final', Subject2_final, ...
    'odor', odor, ...
    'noisy',noisy);

% Save the selected epoch data
save('subject2_struct.mat', 'myStruct');
disp(['Selected epoch data has been saved to ', 'subject2_struct.mat']);
```

We can see that The format of our struct is just as desired:



Subject2 struct



Subject1 struct

3.4 Phase Locking Value (PLV)

- What does phase synchronization indicate from a functional point of view? Discuss its importance with valid references.

Phase synchronization, as indicated by high phase locking value (PLV), has important functional implications in the brain. It reflects the coordination and communication between different brain regions or within a single region. Some key points include:

1. Integration of information: Phase synchronization suggests that brain regions are actively exchanging and integrating information, facilitating efficient communication and collaboration between regions involved in specific cognitive processes or tasks.

2. Neural network dynamics: Phase synchronization enables the formation of transient functional networks, where different brain regions come together to perform specific tasks and then disengage. This flexible coordination is crucial for adaptive cognition and behavior.

3. Cognitive processes: Phase synchronization is involved in various cognitive processes such as perception, attention, memory, and consciousness. It supports perceptual binding, attentional selection, and memory encoding and retrieval.

Studying phase synchronization helps researchers understand the underlying mechanisms of brain function, cognitive processes, and potential applications in clinical settings.

References are at the end of the file.

- Formulate the definition of PLV and briefly discuss the mathematical tools needed to calculate it.

Phase Locking Value (PLV) is a metric used to quantify the degree of phase synchronization or phase consistency between two oscillatory signals. It measures the relationship between the phases of the signals at a specific frequency range.

To calculate PLV, several mathematical tools are required:

1. Fourier Transform: The signals need to be transformed from the time domain to the frequency domain using the Fourier Transform. This allows us to examine the oscillatory components present in the signals.

2. Phase Extraction: The phases of the oscillatory components are extracted from the Fourier-transformed signals. This can be done using techniques such as the Hilbert Transform or complex demodulation.

3. Phase Differences: The phase differences between the two signals at each time point or frequency bin are calculated. This provides information on the relative phase relationship between the signals.

4. Averaging: The phase differences are then averaged over multiple time points or frequency bins to obtain a single value that represents the degree of phase synchronization. This can be done using methods like vector averaging or circular statistics.

5. Normalization: Finally, the calculated phase differences are typically normalized to a range between 0 and 1, where 0 represents a lack of synchronization and 1 indicates perfect phase synchronization.

By applying these mathematical tools, PLV provides a quantitative measure of the extent to which the phases of two signals are consistently aligned or coupled, indicating the level of synchronization between them.

- Implement a function which finds the PLV between two channels in a specific frequency range. This function is going to be needed in the section 4. (Note: You are allowed to define this function with any required input arguments.)

```
function plv = calculatePLV(signal1, signal2, fs, frequencyRange)
    % signal1: Time series data for channel 1
    % signal2: Time series data for channel 2
    % fs: Sampling frequency (in Hz)
    % frequencyRange(Hz):Frequency range of interest[lowerBound,upperBound]

    % Extract frequency range of interest using a bandpass filter
    filteredSignal1 = bandpass(signal1, frequencyRange, fs);
    filteredSignal2 = bandpass(signal2, frequencyRange, fs);
    % Compute the Hilbert transform to obtain instantaneous phases
    phase1 = angle(hilbert(filteredSignal1));
    phase2 = angle(hilbert(filteredSignal2));
    % Compute phase differences
    phaseDiff = phase2 - phase1;
    % Compute PLV
    plv = abs(mean(exp(1i * phaseDiff)));
end
```

4 Results

- In this section, you need to present the required results to assess the difference of Phase Locking Values (PLV) among two groups, namely AD and Normal in the slow gamma frequency range, which is 35 to 40 Hz.

  To fairly compare your results in this part, you do not need to use your preprocessing data from section 3.3 and the preprocessed data of 15 healthy (normal) (age = 69.27±6.65, female = 53.33%) individuals and 13 AD patients (age = 75.31±9.90, female = 61.54%) are availabe through Dataset/Normal.mat and Dataset/AD.mat.

4.1 Values

- Find the PLV for all participants of both groups on both frequent and rare odors between the Fz and Cz channels using the function you implemented in section 3.4 .

The code proceeds to loop through each participant in the normal group and the AD group separately. For each participant, it accesses the epoch data and odor information. It then calculates the PLV for each trial by extracting specific signals from the epoch data (based on the odor of that trial) and

applying a function called `calculatePLV` with certain parameters. The PLV values are accumulated for each participant and odor condition.

After the inner loop completes for each participant, the code calculates the average PLV for each odor condition within the normal and AD groups. Finally, it displays the results (the PLV values) for each group and saves the results in separate .mat files named 'normalPLV.mat' and 'adPLV.mat', respectively.

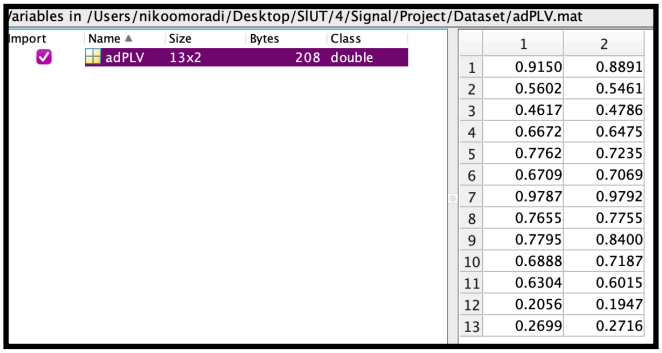
Here is our code for Normal group (the code for AD group is exactly the same):

```
% Load data files
normalData = load('Normal.mat');
adData = load('AD.mat');
% Define frequency range of interest
frequencyRange = [35, 40]; % Hz
% Initialize arrays to store PLV values
normalPLV = zeros(15, 2); % 15 people x 2 odors
adPLV = zeros(13, 2);
% Loop through each participant in the Normal group
for person = 1:15
    % Access the epoch
    [~ , ~ , numTrials] = size(normalData.normal(person).epoch);
    % 4 x 600 x NumTrials matrix
    epochData = normalData.normal(person).epoch;
     % NumTrials x 1 binary array
    odorData = normalData.normal(person).odor;
    % Loop through each trial and calculate PLV
    for trial = 1 : numTrials
        numLemon = sum(odorData(:) == 0);
        numRose = sum(odorData(:) == 1);
        odor = odorData(trial,1);
        signal1_Fz = epochData(2,:,trial);
        signal2_Cz = epochData(3,:,trial);
        normalPLV(person,odor+1) = normalPLV(person,odor+1) + ...
        calculatePLV(signal1_Fz, signal2_Cz, 200, frequencyRange);
    end
    % Average PLV of each odor
    normalPLV(person,1) = normalPLV(person,1)/ numLemon;
    normalPLV(person,2) = normalPLV(person,2)/ numRose;
end
disp('Normal Group PLV:');
disp(normalPLV);
save('normalPLV.mat', 'normalPLV');
```

This code takes a while to run as it goes through each trial for each person and calculates the PLV of that trial. In the end we take the average of the

PLV's. In the end, we have two PLV number for each person in each group, one for the Lemon odor and one for the Rose odor.

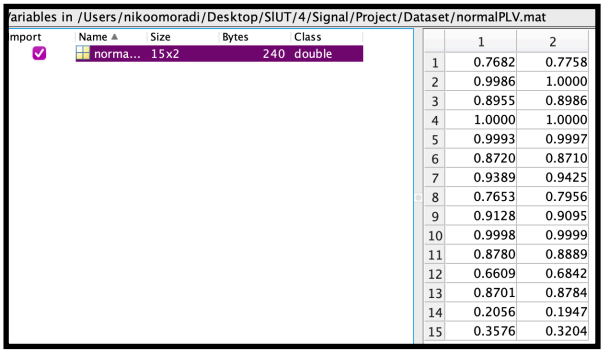
Here are the results :



Variables in /Users/nikoomoradi/Desktop/SIUT/4/Signal/Project/Dataset/adPLV.mat

Import	Name	Size	Bytes	Class
☑	adPLV	13x2	208	double

	1	2
1	0.9150	0.8891
2	0.5602	0.5461
3	0.4617	0.4786
4	0.6672	0.6475
5	0.7762	0.7235
6	0.6709	0.7069
7	0.9787	0.9792
8	0.7655	0.7755
9	0.7795	0.8400
10	0.6888	0.7187
11	0.6304	0.6015
12	0.2056	0.1947
13	0.2699	0.2716



Variables in /Users/nikoomoradi/Desktop/SIUT/4/Signal/Project/Dataset/normalPLV.mat

Import	Name	Size	Bytes	Class
☑	norma...	15x2	240	double

	1	2
1	0.7682	0.7758
2	0.9986	1.0000
3	0.8955	0.8986
4	1.0000	1.0000
5	0.9993	0.9997
6	0.8720	0.8710
7	0.9389	0.9425
8	0.7653	0.7956
9	0.9128	0.9095
10	0.9998	0.9999
11	0.8780	0.8889
12	0.6609	0.6842
13	0.8701	0.8784
14	0.2056	0.1947
15	0.3576	0.3204

4.2 Distributions

- Draw the box plots of PLVs you found in the previous part among two groups and two odors. Also, fit a gaussian distribution on these PLVs and present you results. You need to specify the corresponding p-values to evaluate the statistical significance of your findings.

Here is our code for this part (as everything is clear we avoid unnecessary explanation):

Code for drawing box plots

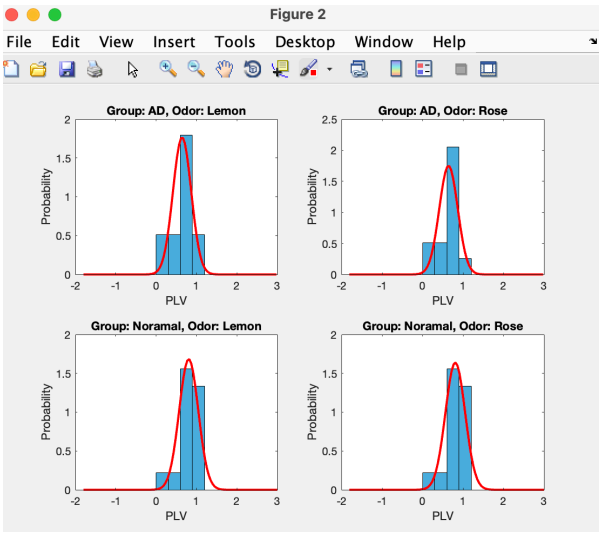
```
%% 4.2
clear; clc;
adPLV = load('adPLV.mat');
normalPLV = load('normalPLV.mat');
adPLV = adPLV.adPLV;
normalPLV = normalPLV.normalPLV;
% Create group and odor labels
groupLabels = {'AD', 'Noramal'};
odorLabels = {'Lemon', 'Rose'};
figure;
for i = 1:2
    for j = 1:2
        subplot(2, 2, (i-1)*2+j);
        if (i == 1 && j == 1)
            data = adPLV(:,1);
        elseif (i == 1 && j == 2)
            data = adPLV(:,2);
        elseif (i == 2 && j == 1)
            data = normalPLV(:,1);
        elseif (i == 2 && j == 2)
            data = normalPLV(:,2);
        end
        boxplot(data);
        title(sprintf('Group: %s, Odor: %s', ...
        groupLabels{i}, odorLabels{j}));

    end
end
```

Code for fitting Gaussian

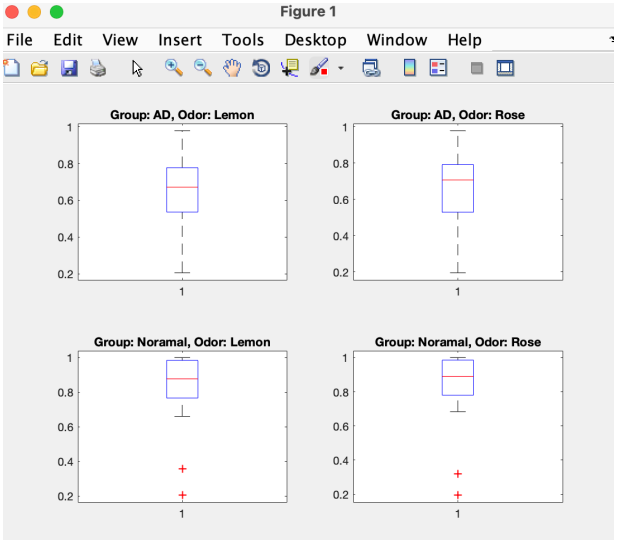
```
% Fit Gaussian distributions
figure;
for i = 1:2
    for j = 1:2
        subplot(2, 2, (i-1)*2+j);
        if (i == 1 && j == 1)
            data = adPLV(:,1);
        elseif (i == 1 && j == 2)
            data = adPLV(:,2);
        elseif (i == 2 && j == 1)
            data = normalPLV(:,1);
        elseif (i == 2 && j == 2)
            data = normalPLV(:,2);
        end
        histogram(data,'Normalization', 'pdf');
        hold on;
        x = linspace(min(data)-2, max(data)+2, 100);
        mu = mean(data);
        sigma = std(data);
        y = normpdf(x, mu, sigma);
        plot(x, y, 'r-', 'LineWidth', 2);
        hold off;
        xlabel('PLV');
        ylabel('Probability');
        title(sprintf('Group: %s, Odor: %s',...
        groupLabels{i}, odorLabels{j}));

    end
end
```

Code for calculating p-value using ttest2 function

```
% Perform statistical tests (t-test) between groups and odors
alpha1 = abs(mean(normalPLV(:, 1))- mean(adPLV(:, 1))); % Significance level
alpha2 = abs(mean(normalPLV(:, 2))- mean(adPLV(:, 2))); % Significance level
% Comparison between groups for Lemon odor
[h1 , pval_group_lemon] = ttest2(normalPLV(:, 1), adPLV(:, 1), 'Alpha', alpha1);
% Comparison between groups for Rose odor
[h2 ,pval_group_rose] = ttest2(normalPLV(:, 2), adPLV(:, 2), 'Alpha', alpha2);
disp('Statistical Significance (p-values):');

disp(['Group Comparison - Lemon Odor: p = ', num2str(pval_group_lemon)]);
if h1 == 1
    disp('Lemon trials are significantly different between AD and normal groups');
else
    disp('not significantly different');
end

disp(['Group Comparison - Rose Odor: p = ', num2str(pval_group_rose)]);
if h2 == 1
    disp('Rose trials are significantly different between AD and normal groups');
else
    disp('not significantly different');
end
```



Fitting Gaussian



Box plot

```
Statistical Significance (p-values):
Group Comparison - Lemon Odor: p = 0.072726
Lemon trials are significantly different between AD and normal groups
Group Comparison - Rose Odor: p = 0.07436
Rose trials are significantly different between AD and normal groups
```

P-values and statistical Significance

4.3 Statistical Significance

- Based on the p-values you founded in the previous part, discuss whether we could state that the ”PLV is significantly different among AD and Normal subjects in the slow gamma frequency range”.

In order to see whether the PLV is significantly different among AD and Normal subjects in the slow gamma frequency range or not, we need to consider the significance threshold or alpha level. Here , as you can see in the provided code , we considered alpha the absolute difference between the average of PLVs of each group for each odor. And then in the t-test we give this alpha as an input argument to the ttest2 function , and we get h as the result.

If p-value is smaller than alpha , then h = 1 meaning that there’s a significant difference between the PLV of two groups with the same odor trial.

And If p-value is bigger than alpha , then h = 0 meaning that there isn’t a significant difference between the PLV of two groups with the same odor trial.

```
alpha1 (Significance level for Group Comparison - Lemon Odor) = 0.16435

alpha2 (Significance level for Group Comparison - Rose Odor) = 0.16654

Statistical Significance (p-values):

Group Comparison - Lemon Odor: p = 0.072726
Lemon trials are significantly different between AD and normal groups

Group Comparison - Rose Odor: p = 0.07436
Rose trials are significantly different between AD and normal groups
```

P-values and statistical Significance results

In this case, as you can see in the results, both alpha1 and alpha2 are bigger than calculated p-values, so there’s a significant difference between the two groups, in other words we can state that ”PLV is significantly different among AD and Normal subjects in the slow gamma frequency range”.

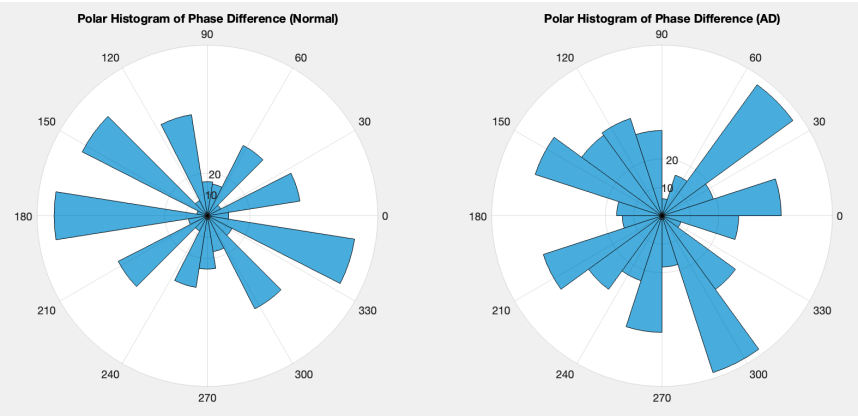
4.4 Phase Difference

- Draw a polar histogram of the phase difference between Fz and Cz channels during frequent odor trials for a random subject in each group and compare the results. Also, plot the mean value of this quantity among all the subjects of each group and discuss the results.

Our code for this part is a bit too long so here we just have Normal group:

Code for plotting polar hist of the phase diff between Fz and Cz during Lemon odor (Normal)

```
% Generate a random number between a and b
randomNormalPerson = floor(1 + (15 - 1) * rand);
randomADPerson = floor(1 + (13 - 1) * rand);

%  Normal group ========================================================
% Access the epoch
    [~ ,~ , numTrials] = size(normalData.normal(randomNormalPerson).epoch);
    % 4 x 600 x NumTrials matrix
    epochData = normalData.normal(randomNormalPerson).epoch;
    % NumTrials x 1 binary array
    odorData = normalData.normal(randomNormalPerson).odor;
    for trial = 1 : numTrials
        numLemon = sum(odorData(:) == 0);
        if odorData(trial,1) == 0 % we just want to do it for lemon odor
        signal1_Fz = epochData(2,:,trial);
        signal2_Cz = epochData(3,:,trial);
        % Calculate phase difference
        phase_diff = abs(angle(signal1_Fz) - angle(signal2_Cz));
        % Convert phase difference to degrees
        phase_diff_deg = rad2deg(phase_diff);
        temp = temp + phase_diff_deg;
        end
    end
    avrg_phase_diff_deg = temp / numLemon;
   % Create polar histogram
   figure;
        subplot(1,2,1);
        num_bins = 20;
        polarhistogram(avrg_phase_diff_deg, num_bins);
```

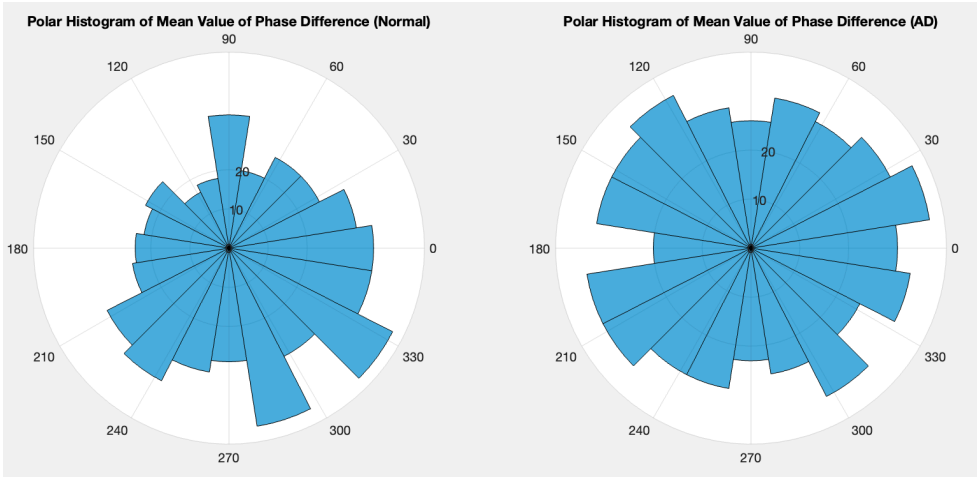


results

And for mean value of the phase difference between Fz and Cz we have :

Code for plotting polar hist of the Mean value of phase diff between Fz and Cz during Lemon odor (Normal)

```
% ================================================================
temp = 0;
avrg_phase_diff_deg =0;
% Mean Phase diff for Normal group ================================
for person = 1 : 15
 [~ , ~ , numTrials] = size(normalData.normal(person).epoch);
 % 4 x 600 x NumTrials matrix
 epochData = normalData.normal(person).epoch;
 % NumTrials x 1 binary array
 odorData = normalData.normal(person).odor;
 for trial = 1 : numTrials
     numLemon = sum(odorData(:) == 0);
     % we just want to do it for lemon(frequent) odor
     if odorData(trial,1) == 0
     signal1_Fz = epochData(2,:,trial);
     signal2_Cz = epochData(3,:,trial);
     % Calculate phase difference
     phase_diff = abs(angle(signal1_Fz) - angle(signal2_Cz));
     % Convert phase difference to degrees
     phase_diff_deg = rad2deg(phase_diff);
     temp = temp + phase_diff_deg;
     end
 end
 avrg_phase_diff_deg = avrg_phase_diff_deg + temp / numLemon;
 temp = 0;
end
% Create polar histogram
figure;
     subplot(1,2,1);
     num_bins = 20;
     polarhistogram(avrg_phase_diff_deg / 15, num_bins);
```



results

4.5 Heatmaps

- Now you need to plot a heatmap which has the PLVs between each pair of the channels. Find whether PLV between other channel pairs are significantly different among two groups in the slow gamma frequency range and test your results. (Note: You need to provide p-values for your hypothesis if you found any significantly different channel pairs apart from (Fz,Cz).)

Our code for this part is the dirtiest code you can imagine. :) so we avoid bringing the code here, instead ***we explain its functionality.***

The code first initializes arrays to store PLV values and matrices to store the PLV values between different pairs of channels. It then loops through each participant in each group.

For each participant, the code performs the following steps:

1. Accesses the participant's epoch data, which consists of a 4 x 600 x NumTrials matrix representing brain signal data.

2. Accesses the participant's odor data, which is a NumTrials x 1 binary array indicating the odor presented during each trial.

3. Averages the epoch data across trials separately for each odor (lemon and rose) to obtain averagedTrials matrix (4 x 600 x 2) for each odor.

4. Extracts the signals from specific channels (Fp1, Fz, Cz, and Pz) for each odor.

5. Calculates the PLV between pairs of signals using the calculatePLV() function, within the specified frequency range of interest.

6. Stores the PLV values in the normalPLV array.

After looping through all participants, the code calculates the mean PLV values across participants for each pair of channels and odors. These mean PLV values are stored in the plv_normal_lemon , plv_normal_rose , plv_AD_lemon and plv_AD_lrose matrices.

Finally, the code generates a figure with 4 subplots.

In summary, this code , calculates PLV values between pairs of channels and odors, and visualizes the results in heatmaps. The PLV values can provide insights into the synchronization of neural activity between different brain regions in response to different odors.

The results are shown in the next page :

results

```
Statistical Significance (p-values):

Group Comparison - Fp1 & Fz : p = 0.90673     0.76592
not significantly different

Group Comparison - Fp1 & Cz : p = 0.58951     0.84699
not significantly different

Group Comparison - Fp1 & Pz : p = 0.91814     0.79607
not significantly different

Group Comparison - Fz & Cz : p = 0.15787    0.040343
significantly different

Group Comparison - Fz & Pz : p = 0.75285     0.94352
not significantly different

Group Comparison - Cz & Pz : p =  0.85143     0.35815
not significantly different
```

5 *Bonus

5.1.1 Additional Information

- Describe the relationship between MCI and AD. Explain whether MCI would always result in AD and briefly investigate the causes of MCI.

Mild Cognitive Impairment (MCI) is often considered a transitional stage between normal age-related cognitive decline and Alzheimer's disease (AD). While MCI increases the risk of developing AD, it does not always progress to this more severe form of dementia. Studies have shown that individuals with MCI have a higher likelihood of developing AD compared to those without MCI, but not everyone with MCI will develop AD. Other factors, such as genetics, lifestyle, and overall health, play a role in determining the progression from MCI to AD.

The causes of MCI can vary and are often multifactorial. Some common factors contributing to MCI include neurodegenerative diseases like AD, vascular issues that affect blood flow to the brain, certain medications, depression, anxiety, thyroid problems, and chronic conditions such as diabetes and hypertension. Additionally, genetic factors, including the presence of the apolipoprotein E (APOE) ε4 allele, have been associated with an increased risk of MCI and AD. However, it's important to note that MCI can also be caused by non-pathological factors and may not always progress to dementia.

5.1.2 MCI Data Processing

- In the provided dataset, you can find MCI.mat file. This dataset contains preprocessed cleaned EEG recording of the same task described in sections 3.1 and 3.2 for 7 MCI patients.

  Based on the significantly different coupled channels you found for differentiation between AD and Normal groups, find the Phase-Locking-Value (PLV) for the MCI subjects and provide the required results by comparing all the the 3 states (Normal, MCI, AD). Your findings must include the significance testing by providing the corresponding p-values.

Here we have not much to add because we are technically repeating every thing for MCI data. But here’s our results for PLV of MCI subjects and significance testing for all the 3 states :

PLV of MCI subjects for each odor

Variables in /Users/nikoomoradi/Desktop/SIUT/4/Signal/Project/codes/mciPLV.mat

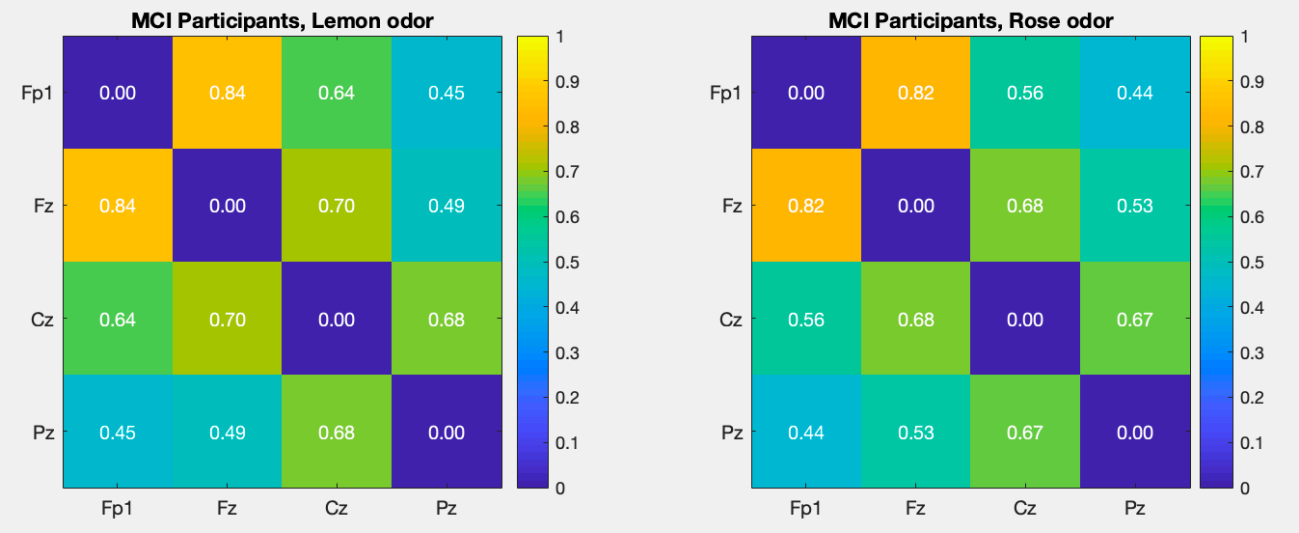
Import	Name	Size	Bytes	Class
☑	mciPLV	7x2	112	double

	1	2
1	0.7703	0.7434
2	0.9128	0.9236
3	0.7716	0.8002
4	0.5822	0.5834
5	0.4029	0.3883
6	0.7532	0.7140
7	0.7051	0.7217

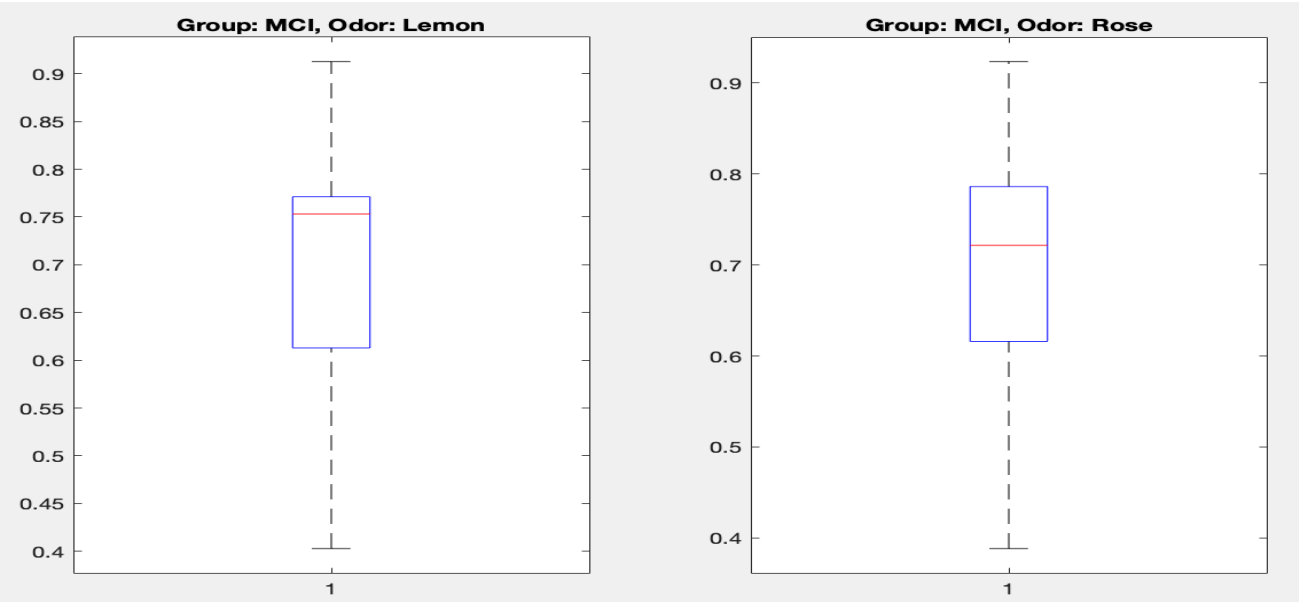
Significance testing and p-values between Normal and AD and MCI

```
Statistical Significance (p-values):

Comparison between Normal & AD - Lemon Odor: p = 0.072726
Lemon trials are significantly different between AD and Normal groups

Comparison between Normal & AD - Rose Odor: p = 0.07436
Rose trials are significantly different between AD and Normal groups

Comparison between Normal & MCI - Lemon Odor: p = 0.28908
not significantly different

Comparison between Normal & MCI - Rose Odor: p = 0.27828
not significantly different

Comparison between MCI & AD - Lemon Odor: p = 0.57192
not significantly different

Comparison between MCI & AD - Rose Odor: p = 0.60237
not significantly different
```

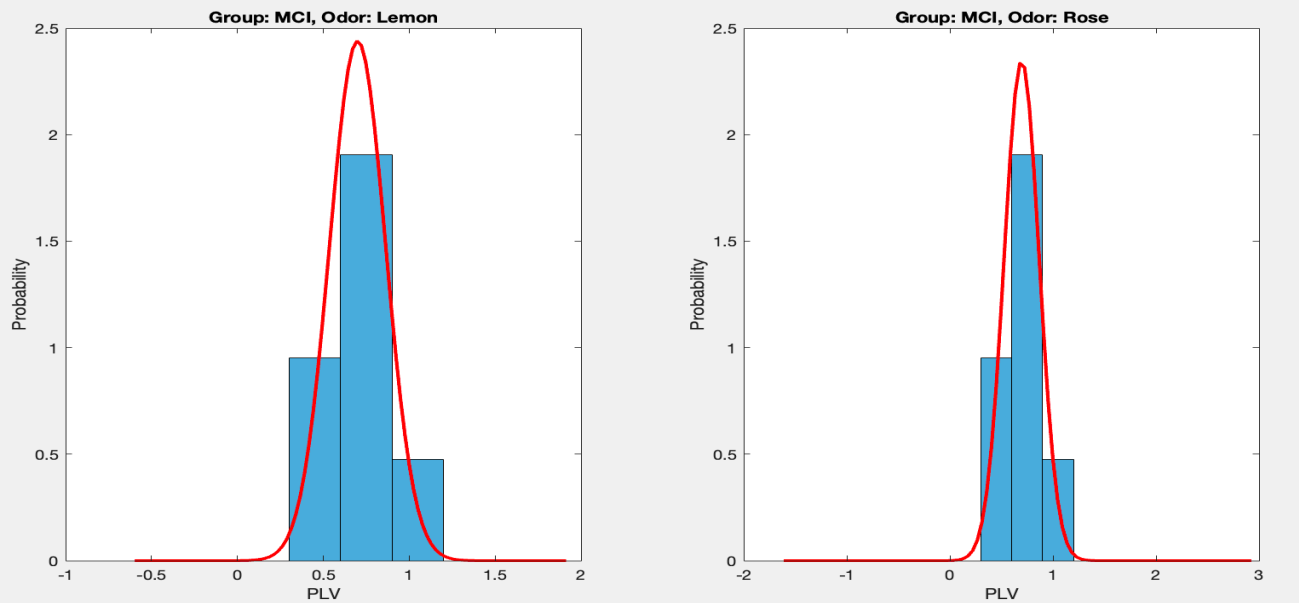
From the significance test, we can conclude that MCI is somehow a stage between Normal and AD. That Is why there's no significant difference between MCI and neither Normal nor AD group.



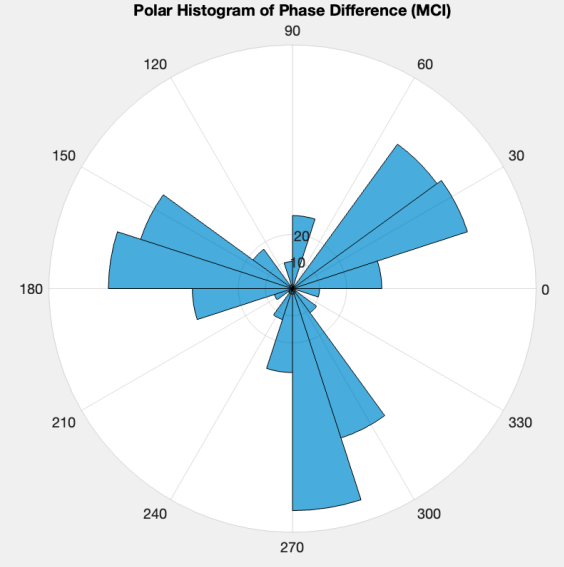
Heat map for MCI

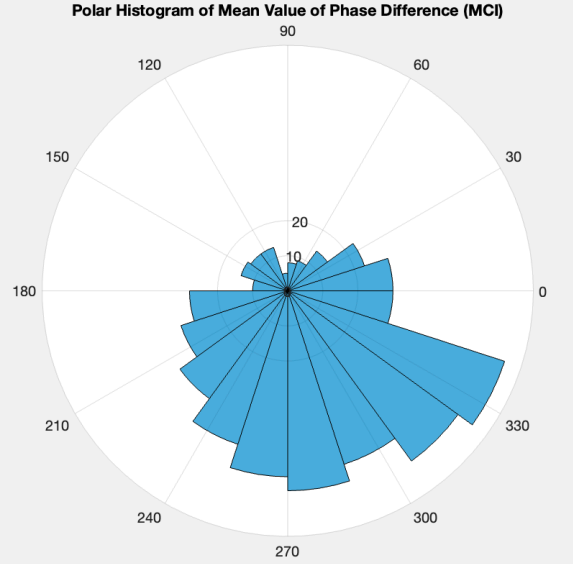


Box plot for MCI



Fitting Gaussian for MCI





5.2 Phase-Amplitude Coupling (PAC)

- PLV was just one instance of the Phase-Amplitude Coupling (PAC) metrics. PAC is a form of cross-frequency coupling where the amplitude of a high frequency signal is modulated by the phase of low frequency oscillations. PAC is the most-studied type of cross-frequency coupling and is thought to be responsible for integration across populations of neurons. Low frequency brain activity controls the information exchange between brain regions by modulating the amplitude of the high frequency oscillations. [6]

5.2.1 Metrics

- Conduct a search about other PAC measures and briefly provide an explanation about two of them.

Phase-Amplitude Coupling (PAC) is a widely studied phenomenon in neuroscience that describes the relationship between the phase of low-frequency oscillations and the amplitude of high-frequency oscillations. In addition to Phase-Locking Value (PLV), there are several other measures that have been developed to quantify PAC. Here are brief explanations of two other commonly used PAC measures:

1. Modulation Index (MI): The Modulation Index is a PAC measure that quantifies the strength of phase-amplitude coupling between two frequency bands. It calculates the degree to which the amplitude of the high-frequency oscillations is modulated by the phase of the low-frequency oscillations. The MI is based on comparing the observed amplitude distribution with the

expected amplitude distribution under the assumption of no coupling. A higher MI value indicates stronger PAC.

2. Normalized Amplitude-Modulation Index (nAMI): The Normalized Amplitude-Modulation Index is a PAC measure that aims to address the bias of the Modulation Index towards higher amplitude oscillations. It normalizes the MI by dividing it by the average amplitude of the high-frequency oscillations. This normalization ensures that the PAC measure is not biased by the overall amplitude of the high-frequency oscillations and allows for a more accurate comparison across different frequency bands and recording conditions.

It is important to note that PAC measures can vary in their mathematical formulation and the specific assumptions they make. Different measures may be suitable for different experimental contexts or research questions, and researchers often choose the most appropriate measure based on their specific needs and the characteristics of their data.

5.2.2 Implementation

- Implement one of the metrics mentioned earlier as a biomarker for distinguishing between AD and Normal groups. Present the relevant results through plots and provide a discussion regarding the efficacy of the selected metric.

For this part we use Modulation Index (MI) metric, for calculating this metric we used a GitHub link that you can find in the **References** part at the end of the report. (We don't explore the details of this function here)
We calculate this metric for each person in Normal and AD group, and between Fz and Cz channels.

After that we apply the Significance test and measure p-value between the two groups for the 2 odors, to see whether the two groups have a significant difference in this specific metric.

We also draw box plots of MIs and fit a gaussian distribution on these MIs, just like what we did for PLV.

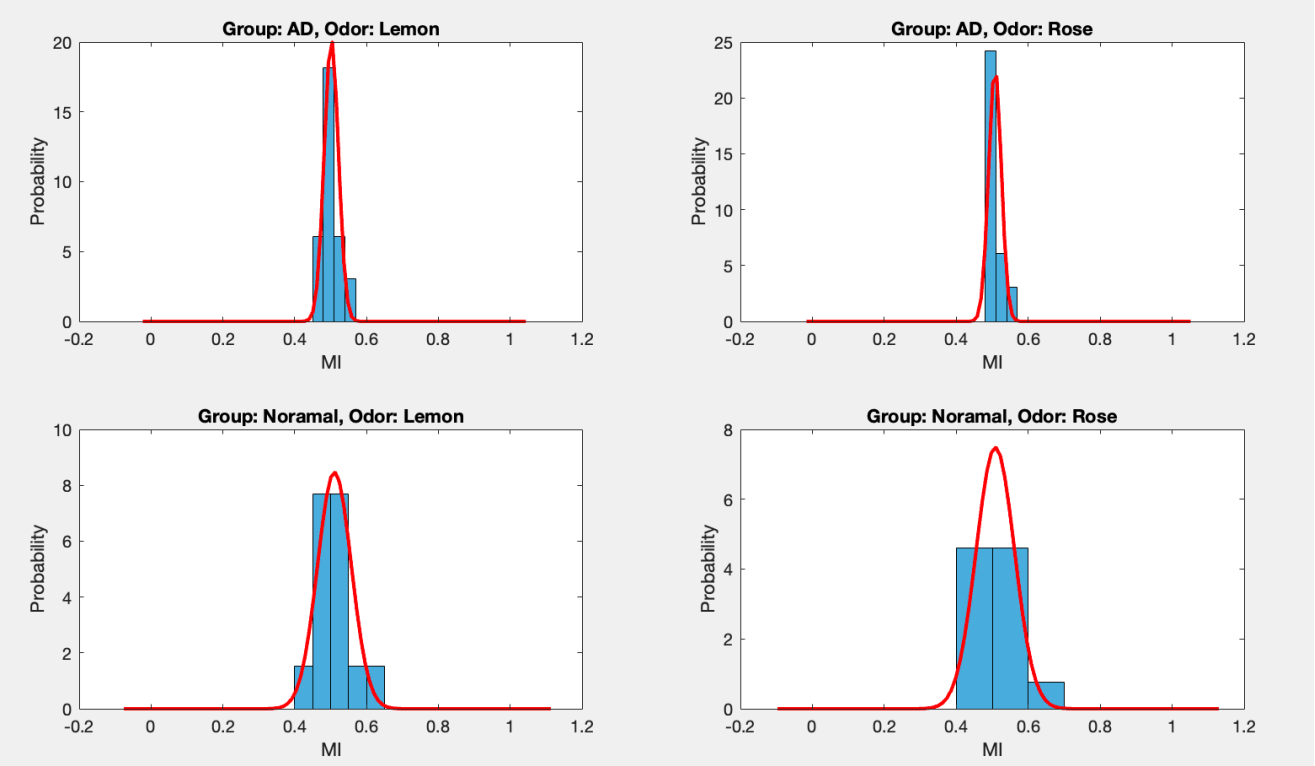
Calculating MI for Normal group and for each odor

```
% Load data files
normalData = load('Normal.mat');
adData = load('AD.mat');
nbins = 18;
% helpMI = zeros( 600 , 600); % 15 people x MI param(600x600)
normalMI = zeros(13, 2);

% Loop through each participant in the Normal group
for person = 1:13
    % Access the epoch
    [~ , ~ , numTrials] = size(normalData.normal(person).epoch);
    % 4 x 600 x NumTrials matrix
    epochData = normalData.normal(person).epoch;
     % NumTrials x 1 binary array
    odorData = normalData.normal(person).odor;
    % Loop through each trial and calculate PLV
    for trial = 1 : numTrials
        numLemon = sum(odorData(:) == 0);
        numRose = sum(odorData(:) == 1);
        odor = odorData(trial,1);

        signal1_Fz = epochData(2,:,trial);
        signal2_Cz = epochData(3,:,trial);

        temp = get_mi(angle(signal1_Fz), abs(signal2_Cz), nbins);
        normalMI(person,odor + 1) = normalMI(person,odor + 1) + ...
        mean(mean(temp.MI));

    end
    % Average PLV of each odor
    normalMI(person, 1) = normalMI(person,1)/ numLemon;
    normalMI(person, 2) = normalMI(person,2)/ numRose;

end
```

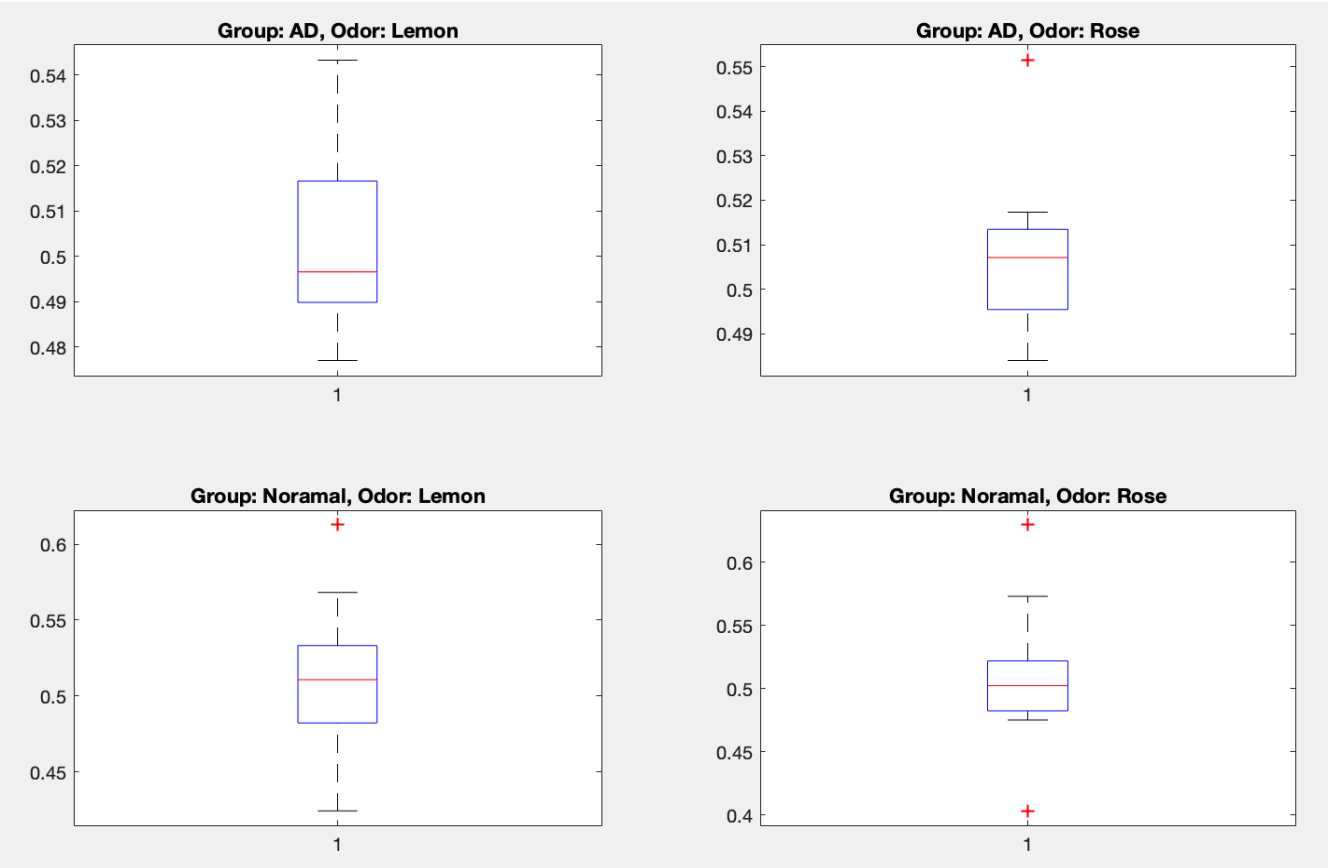
```
Normal Group MI:
    0.4729    0.4938
    0.4791    0.4814
    0.4990    0.5049
    0.5108    0.4827
    0.5436    0.5426
    0.4877    0.4998
    0.5297    0.4750
    0.4242    0.4031
    0.5682    0.5729
    0.5131    0.5023
    0.6130    0.6296
    0.5144    0.5149
    0.4833    0.5094
```

```
AD Group MI:
    0.5433    0.5515
    0.4942    0.5072
    0.4891    0.4936
    0.5080    0.5092
    0.4770    0.5058
    0.4965    0.4840
    0.4784    0.5072
    0.4920    0.4928
    0.5055    0.5173
    0.5198    0.5012
    0.5194    0.5149
```

Fitting Guassian for MI



Box plot for MI



Significance testing and p-values for MI

```
alpha1 (Significance level for Group Comparison - Lemon Odor) = 0.0085767

alpha2 (Significance level for Group Comparison - Rose Odor) = 0.00095574

Statistical Significance (p-values):

Group Comparison - Lemon Odor: p = 0.57943
not significantly different

Group Comparison - Rose Odor: p = 0.95521
not significantly different
```

From the p-values and significance testing and, we can conclude that there isn't a significant difference between this metric of the two groups (Noramal and AD), so we can say that this metric isn't efficient in this matter.

6 Conclusion

Throughout this project, we have gone through various stages of processing brain signals. We initially familiarized ourselves with EEG signals and data acquisition methods. We then proceeded with the step-by-step preprocessing of the laboratory data and recognized its significance in data processing. We utilized the powerful toolbox, EEG Lab, for preprocessing and gained familiarity with its different components and functionalities. Additionally, we discovered that preprocessing tasks can also be accomplished without using EEG Lab by coding in the MATLAB environment.

Furthermore, we concluded that the preprocessing stages and different pipeline configurations vary depending on the type of experiment, data recording session, required information, and more. We processed the raw data by applying various filters, performing epoching, eliminating noisy segments, and ultimately retaining the relevant data from the desired 4 channels, leading us to obtain clean preprocessed data.

In the subsequent stage, we delved into the concept of Phase-Locking Value (PLV) and its importance. We initially presented a function for measuring PLV between two signals to enable us to utilize it for data analysis throughout the project.

In Part 4 of the project, using the preprocessed signals, we calculated PLV for the Fz and Cz channels in both the Alzheimer's patient group and the healthy control group. We then computed various statistics to analyze these PLV values, including box plots, Gaussian curve fitting, hypothesis testing, and p-values.

By employing hypothesis testing, we deduced that a significant difference exists in PLV between the Alzheimer's patient group and the healthy control group. This finding highlights the importance of Olfactory Dysfunction as a crucial factor in analyzing Alzheimer's patients, and further research in this area may potentially lead to prediction and even prevention of Alzheimer's disease.

Furthermore, we observed differences not only in PLV between the Fz and Cz channels but also among other channels, suggesting the need for further investigation and exploration.

Finally, we applied the aforementioned steps to analyze the MCI (Mild Cognitive Impairment) data as well. Additionally, we explored another type of PAC, MI and conclude that this metric isn't efficient in this study.

In conclusion, this project provided valuable insights into the processing and analysis of brain signals. By leveraging techniques such as preprocessing, PLV calculation, statistical analysis, and hypothesis testing, we shed light on the significance of olfactory dysfunction in neurodegenerative diseases, particularly Alzheimer's disease. The findings indicate that PLV analysis can serve as a potential diagnostic tool and contribute to early detection and intervention in neurodegenerative diseases. This project sets the stage for further research and exploration in this field, ultimately aiming to improve the understanding, prediction, and management of neurodegenerative disorders.

• References:

- Babiloni, C., Vecchio, F., Bultrini, A., Luca Romani, G., & Rossini, P. M. (2009). Pre-and poststimulus alpha rhythms are related to conscious visual perception: A high-resolution electroencephalographic study. Cerebral Cortex, 18(3), 676-685.

- Engel, A. K., Fries, P., & Singer, W. (2001). Dynamic predictions: oscillations and synchrony in top–down processing. Nature Reviews Neuroscience, 2(10), 704-716.

- Lisman, J. E., & Jensen, O. (2013). The theta–gamma neural code. Neuron, 77(6), 1002-1016.

- Sauseng, P., Klimesch, W., Schabus, M., & Doppelmayr, M. (2007). Fronto-parietal EEG coherence in theta and upper alpha reflect central executive functions of working memory. International Journal of Psychophysiology, 57(2), 97-103.

- https://github.com/bvoloh/cfc_analysis