Aim

The aim of this project is to analyze neural signals for phase synchronization using Inter-Site Phase Clustering (ISPC) and other related techniques such as the Laplacian method on EEG data.

Objective

1. To understand and implement phase synchronization analysis using ISPC.
2. To apply the Laplacian method on simulated and real EEG data.
3. To compare phase synchronization metrics (ISPC and PLI) across different conditions and data types.
4. To visualize the results using various plotting techniques in MATLAB.

Summary

This project involves the analysis of neural signals, focusing on phase synchronization between different channels using the Inter-Site Phase Clustering (ISPC) method. The code loads neural signal data from mice and EEG data, processes the signals using wavelets and Fourier transforms, and calculates phase synchronization metrics. The results are visualized through various plots, including time-frequency representations and polar plots. The Laplacian method is also applied to both simulated and real EEG data to highlight spatial frequency features.

Tools and Libraries Used

-Matlab
-EEG data
-Morlet wavelet transform
-FFT (Fast Fourier Transform)
-ISPC (Inter-site Phase Clustering)
-PLI (Phase Lag Index)
-Topoplot functions for visualization

- Built-in functions: fft, ifft, polar, imagesc, etc.
- Custom functions: laplacian_perrinX, topoplotIndie

Procedure

1. Loading Data

CODE:

% Load V1 mouse data

load v1_laminar

The data is loaded into the workspace for analysis.

2. Setting Up Connectivity Parameters

CODE:

```
% Channels for connectivity
chan1idx = 1;
chan2idx = 8;

% Create a complex Morlet wavelet
cent_freq = 8;
time      = -1.5:1/srate:1.5;
s         = 8/(2*pi*cent_freq);
wavelet   = exp(2*1i*pi*cent_freq.*time) .* exp(-time.^2./(2*s^2));
half_wavN = (length(time)-1)/2;

% FFT parameters
nWave = length(time);
nData = size(csd,2);
nConv = nWave + nData - 1;

% FFT of wavelet
waveletX = fft(wavelet,nConv);
waveletX = waveletX ./ max(waveletX);
```

The code sets up parameters for connectivity analysis, including defining channels and creating a Morlet wavelet for signal processing.

3. Analytic Signal Extraction

CODE:

```
% Analytic signal of channel 1
dataX = fft(squeeze(csd(chan1idx,:,1)),nConv);
as = ifft(waveletX.*dataX,nConv);
as = as(half_wavN+1:end-half_wavN);

% Collect real and phase data
```

phase_data(1,:) = angle(as); % Extract phase angles

real_data(1,:) = real(as); % Extract the real part (projection onto real axis)

The analytic signal is extracted from the data using FFT and inverse FFT, and phase angles and real parts are stored for further analysis.

4. Visualization Setup

CODE:

% Setup figure and define plot handles

figure(1), clf

set(gcf,'NumberTitle','off','Name','Movie magic minimizes the magic.');

% Draw the filtered signals

subplot(221)

filterplotH1 = plot(timevec(1),real_data(1,1),'b');

hold on

filterplotH2 = plot(timevec(1),real_data(2,1),'m');

set(gca,'xlim',[timevec(1) timevec(end)],'ylim',[min(real_data(:)) max(real_data(:))])

xlabel('Time (ms)')

ylabel('Voltage (\muV)')

title(['Filtered signal at ' num2str(cent_freq) ' Hz'])

The code sets up plots for visualizing the filtered signals, phase angles, and their differences.

5. Updating Plots in Real-Time

CODE:

% Update plots at each timestep

for ti=1:5:length(timevec)

  set(filterplotH1,'XData',timevec(1:ti),'YData',real_data(1,1:ti))

  set(filterplotH2,'XData',timevec(1:ti),'YData',real_data(2,1:ti))

  % Cartesian plot of phase angles

  set(phaseanglesH1,'XData',timevec(1:ti),'YData',phase_data(1,1:ti))

```
    set(phaseanglesH2,'XData',timevec(1:ti),'YData',phase_data(2,1:ti))

    % Update polar plots
    subplot(223), cla
    polar([zeros(1,ti) phase_data(1,1:ti)]',repmat([0 1],1,ti)','b');
    hold on
    polar([zeros(1,ti) phase_data(2,1:ti)]',repmat([0 1],1,ti)','m');

    subplot(224), cla
    polar([zeros(1,ti) phase_data(2,1:ti)-phase_data(1,1:ti)]',repmat([0 1],1,ti)','k');

    drawnow
end
```

The code updates plots in real-time to visualize the evolution of filtered signals and phase angles over time.

6. Quantifying Phase Synchronization

CODE:

```
% Quantify phase synchronization
phase_angle_differences = phase_data(2,:)-phase_data(1,:);

% Euler representation of angles
euler_phase_differences = exp(1i*phase_angle_differences);

% Mean vector in complex space
mean_complex_vector = mean(euler_phase_differences);

% Length of mean vector as measure of phase synchronization
phase_synchronization = abs(mean_complex_vector);
```

```
disp([ 'Synchronization between ' num2str(chan1idx) ' and ' num2str(chan2idx) ' is ' num2str(phase_synchronization) '!' ])
```

Phase synchronization is quantified using the mean vector length in complex space, representing the degree of synchronization between channels.

7. Applying Laplacian to Simulated EEG Data

CODE:

```
clear

load sampleEEGdata.mat


chan1 = 'pz';

chan2 = 'cz';


chan1idx = strcmpi(chan1,{EEG.chanlocs.labels});

chan2idx = strcmpi(chan2,{EEG.chanlocs.labels});


% Compute inter-channel distances

eucdist = zeros(2,64);

for chani=1:EEG.nbchan

  eucdist(1,chani) = sqrt( (EEG.chanlocs(chani).X-EEG.chanlocs(chan1idx).X)^2 + ...

                (EEG.chanlocs(chani).Y-EEG.chanlocs(chan1idx).Y)^2 + ...

                (EEG.chanlocs(chani).Z-EEG.chanlocs(chan1idx).Z)^2 );

  eucdist(2,chani) = sqrt( (EEG.chanlocs(chani).X-EEG.chanlocs(chan2idx).X)^2 + ...

                (EEG.chanlocs(chani).Y-EEG.chanlocs(chan2idx).Y)^2 + ...

                (EEG.chanlocs(chani).Z-EEG.chanlocs(chan2idx).Z)^2 );

end


% Generate low- and high- spatial frequency activity

lo_spatfreq = 50*exp(- (eucdist(1,:).^2)/ 300000 );

hi_spatfreq =    exp(- (eucdist(2,:).^2)/   3000 );
```

```
% Compute Laplacian of summed activity
surf_lap =
laplacian_perrinX(hi_spatfreq+lo_spatfreq,[EEG.chanlocs.X],[EEG.chanlocs.Y],[EEG.chanlocs.Z]);


% Plot results
figure(2), clf
subplot(221)
topoplotIndie(lo_spatfreq,EEG.chanlocs,'numcontour',0);
title('Low spatial frequency feature')

subplot(222)
topoplotIndie(hi_spatfreq,EEG.chanlocs,'numcontour',0);
title('High spatial frequency features')

subplot(223)
topoplotIndie(hi_spatfreq+lo_spatfreq,EEG.chanlocs,'numcontour',0);
title('Low+high features')

subplot(224)
topoplotIndie(surf_lap,EEG.chanlocs,'numcontour',0);
title('Laplacian of low+high features')
```

The Laplacian method is applied to simulated EEG data, highlighting low and high spatial frequency features and their combination.

8. Applying Laplacian to Real EEG Data

CODE:

```
clear
load sampleEEGdata.mat
```

```
time2plot = 250; % ms
chan2plot = 'cz';

% Convert to indices
tidx = dsearchn(EEG.times',time2plot);
chanidx = strcmpi({EEG.chanlocs.labels},chan2plot);

% Compute the Laplacian
EEG.lap = laplacian_perrinX(EEG.data,[EEG.chanlocs.X],[EEG.chanlocs.Y],[EEG.chanlocs.Z]);

% Normalize data
voltERP = mean(EEG.data(:,:,tidx),3);
lapERP = mean(EEG.lap(:,:,tidx),3);

voltERP = voltERP / max(abs(voltERP));
lapERP  =  lapERP / max(abs(lapERP));

% Plot results
figure(3), clf
subplot(231)
topoplotIndie(voltERP,EEG.chanlocs,'numcontour',0);
title([ 'Voltage at ' num2str(EEG.times(tidx)) ' ms' ])

subplot(232)
topoplotIndie(lapERP,EEG.chanlocs,'numcontour',0);
title([ 'Laplacian at ' num2str(EEG.times(tidx)) ' ms' ])

subplot(233)
plot(EEG.times,squeeze(mean(EEG.data(chanidx,:,:),3)),'k','linew',3)
```

```
set(gca,'xlim',[-300 1000])

xlabel('Time (ms)'), ylabel('Voltage (\muV)')

title([ 'Voltage at electrode ' chan2plot ])


subplot(235)

plot(EEG.times,squeeze(mean(EEG.lap(chanidx,:,:),3)),'k','linew',3)

set(gca,'xlim',[-300 1000])

xlabel('Time (ms)'), ylabel('Voltage (\muV)')

title([ 'Laplacian at electrode ' chan2plot ])
```

The Laplacian method is applied to real EEG data, and the results are visualized, showing the spatial distribution of voltage and Laplacian values.

Conclusion

The project demonstrates the analysis of neural signals using phase synchronization metrics like ISPC and PLI. The Morlet wavelet transform and FFT are used to extract analytic signals, and the results are visualized in various plots. The Laplacian method is applied to both simulated and real EEG data to highlight spatial frequency features, providing a comprehensive understanding of neural synchrony and signal processing techniques in EEG data analysis.