**Aim:**

To perform time-frequency analysis on neural signals using complex Morlet wavelets, extracting power and phase information from EEG data, and comparing different methods of spectral analysis.

**Objective:**

1. To analyze time-frequency power and phase for specific EEG trials.
2. To compare results across multiple trials and visualize the effects.
3. To explore edge effects in time-frequency analysis.
4. To improve spectral precision and compare wavelet convolution with filter-Hilbert methods.
5. To perform wavelet convolution on multiple channels and visualize the results.

**Summary:**

The project involves conducting time-frequency analysis on EEG data using complex Morlet wavelets. Key tasks include:

- Creating a family of complex Morlet wavelets.
- Performing convolution to extract power and phase information.
- Comparing results for different frequencies and trials.
- Investigating edge effects and spectral precision.
- Comparing complex Morlet wavelet convolution with filter-Hilbert analysis.
- Extending analysis to multiple EEG channels and visualizing the results.

**Tools and Libraries Used:**

- **MATLAB**: For data processing and visualization.
- **Complex Morlet Wavelets**: For time-frequency decomposition.
- **FFT/IFFT**: For Fourier transforms and signal processing.
- **EEG Data**: For neural signal analysis.

**Procedure:**

**Power and Phase Extraction (Trial 10):**

- Load the V1 laminar dataset.
- Define frequency range (10-100 Hz) and number of frequencies (43).
- Create complex Morlet wavelets.
- Perform convolution between the wavelets and V1 data from trial 10.

- Extract and store power and phase in a 4D matrix.

**Visualization (Trial 10):**

- Plot time-frequency power and phase from electrode 6.

**Edge Effects Analysis:**

- Create a square-wave time series.
- Perform time-frequency analysis.
- Visualize the effects of edges on power and phase.

**Comparison with Filter-Hilbert:**

- Compute ERP from channel 7.
- Perform wavelet convolution and filter-Hilbert methods.
- Compare the results.

**1. Power and Phase Extraction from EEG Data**

**Code Explanation:**

1. **Load Data:**

CODE:

```
load v1_laminar.mat
```

This line loads the EEG data from a .mat file.

2. **Set Parameters:**

CODE:

```
freqrange  = [10 100];
numfrex    = 43;
whichTrial = 10;
```

Defines the frequency range for the analysis, the number of frequencies, and the trial to analyze.

3. **Convolution Setup:**

CODE:

```
wavtime = -2:1/srate:2-1/srate;
frex    = linspace(freqrange(1),freqrange(2),numfrex);
nData   = length(timevec);
```

```
nKern   = length(wavtime);
nConv   = nData + nKern - 1;
halfwav = (length(wavtime)-1)/2;
```

Sets up parameters for convolution including the time vector and frequency vector.

4. **Create Wavelets:**

CODE:

```
cmwX = zeros(numfrex,nConv);
for fi=1:numfrex
   twoSsquared = 2 * (numcyc(fi)/(2*pi*frex(fi))) ^ 2;
   cmw = exp(2*1i*pi*frex(fi).*wavtime) .* exp( (-wavtime.^2) / twoSsquared );
   cmwX(fi,:) = fft(cmw,nConv);
   cmwX(fi,:) = cmwX(fi,:) ./ max(cmwX(fi,:));
end
```

Generates complex Morlet wavelets for each frequency and normalizes them.

5. **Perform Convolution:**

CODE:

```
tf = zeros(size(csd,1),numfrex,length(timevec),2);
for chani=1:size(csd,1)
   eegX = fft( squeeze(csd(chani,:,whichTrial)) ,nConv);
   for fi=1:numfrex
      as = ifft( cmwX(fi,:).*eegX ,nConv );
      as = as(halfwav+1:end-halfwav);
      tf(chani,fi,:,1) = abs(as).^2;
      tf(chani,fi,:,2) = angle(as);
   end
end
```

Convolves each wavelet with the EEG data and extracts power and phase information.

6. **Plot Results:**

CODE:

```
figure(2), clf
subplot(121)
contourf(timevec,frex,squeeze(tf(chan2plot,:,:,1)),40,'linecolor','none')
xlabel('Time (s)'), ylabel('Frequencies (Hz)'), title([ 'Power from trial 10 at contact ' num2str(chan2plot) ])
set(gca,'xlim',[-.2 1],'clim',[0 80000])
subplot(122)
contourf(timevec,frex,squeeze(tf(chan2plot,:,:,2)),40,'linecolor','none')
xlabel('Time (s)'), ylabel('Frequencies (Hz)'), title([ 'Phase from trial 10 at contact ' num2str(chan2plot) ])
set(gca,'xlim',[-.2 1],'clim',[-pi pi])
```

Plots time-frequency power and phase for a specific electrode.

**2. Convolution with All Trials**

**Code Explanation:**

1. **Prepare Data:**

CODE:

```
nData   = length(timevec)*size(csd,3);
nConv   = nData + nKern - 1;
cmwX = zeros(numfrex,nConv);
for fi=1:numfrex
   twoSsquared = 2 * (numcyc(fi)/(2*pi*frex(fi))) ^ 2;
   cmw = exp(2*1i*pi*frex(fi).*wavtime) .* exp( (-wavtime.^2) / twoSsquared );
   cmwX(fi,:) = fft(cmw,nConv);
   cmwX(fi,:) = cmwX(fi,:) ./ max(cmwX(fi,:));
end
```

Recomputes convolution parameters and creates wavelets for all trials.

2. **Compute Time-Frequency Representation:**

CODE:

```
eegX = fft( reshape(csd,size(csd,1),[]) ,nConv,2);
for fi=1:numfrex
   as = ifft( eegX.*repmat(cmwX(fi,:),size(csd,1),1) ,nConv,2 );
```

```
    as = as(:,halfwav+1:end-halfwav);
    as = reshape(as,size(csd));
    tf(:,fi,:,1) = mean( abs(as).^2 ,3);
    tf(:,fi,:,2) = abs(mean( exp(1i*angle(as)) ,3));
end
```

Computes time-frequency power and phase for all trials and channels, averaging across trials for ITPC (Inter-Trial Phase Coherence).

3. **Plot Results:**

CODE:

```
figure(4), clf
subplot(121)
contourf(timevec,frex,squeeze(tf(chan2plot,:,:,1)),40,'linecolor','none')
xlabel('Time (s)'), ylabel('Frequencies (Hz)'), title([ 'Power from trial 10 at contact ' num2str(chan2plot) ])
set(gca,'xlim',[-.2 1],'clim',[0 80000])
subplot(122)
contourf(timevec,frex,squeeze(tf(chan2plot,:,:,2)),40,'linecolor','none')
xlabel('Time (s)'), ylabel('Frequencies (Hz)'), title([ 'Phase from trial 10 at contact ' num2str(chan2plot) ])
set(gca,'xlim',[-.2 1],'clim',[0 .5])
```

Plots the averaged time-frequency power and phase for a specific electrode.

**3. Exploring Edge Effects**

**Code Explanation:**

1. **Create and Plot Square Wave:**

CODE:

```
srate = 999;
npnts = srate*1;
time  = (0:npnts-1)/srate;
squarets = zeros(1,npnts);
squarets( round(npnts*.4:npnts*.6) ) = 1;
subplot(311)
```

```
plot(time,squarets,'linew',3)
title('Box-care time series')
```

2. **Perform Time-Frequency Analysis:**

CODE:

```
freqrange  = [1 100];
numfrex    = 83;
wavtime = -2:1/srate:2;
frex    = linspace(freqrange(1),freqrange(2),numfrex);
nKern   = length(wavtime);
nConv   = npnts + nKern - 1;
halfwav = (length(wavtime)-1)/2;
numcyc = linspace(3,15,numfrex);
impfunX = fft( squarets ,nConv);
tf = zeros(numfrex,npnts,2);
for fi=1:numfrex
   cmw = exp(2*1i*pi*frex(fi).*wavtime) .* exp( (-wavtime.^2) / (2 * (numcyc(fi)/(2*pi*frex(fi))) ^ 2) );
   cmwX = fft(cmw,nConv);
   as = ifft( cmwX.*impfunX / max(cmwX) ,nConv );
   as = as(halfwav+1:end-halfwav);
   tf(fi,:,1) = abs(as).^2;
   tf(fi,:,2) = angle(as);
end
subplot(312)
imagesc(time,frex,squeeze(tf(:,:,1)))
set(gca,'clim',[0 .001],'ydir','normal')
title('Time-frequency power (square wave)')
subplot(313)
imagesc(time,frex,squeeze(tf(:,:,2)))
set(gca,'clim',[-pi pi],'ydir','normal')
```

```
title('Time-frequency phase (square wave)')
```

**4. Comparing Wavelet Convolution and Filter-Hilbert Method**

**Code Explanation:**

1. **Define Parameters and Create Wavelet:**

CODE:

```
fwhm    = .2;
wavetime= (0:2*srate-1)/srate;
wavetime= wavetime - mean(wavetime);
cmw = exp(1i*2*pi*42*wavetime) .* exp(-4*log(2)*wavetime.^2 / fwhm^2);
```

2. **Compute Time-Frequency Representation Using Wavelet Convolution:**

CODE:

```
erp = mean(csd(7,:,:), 3);
nData = length(erp);
nConv = nData + length(cmw) - 1;
cmwX = fft(cmw, nConv);
erpX = fft(erp, nConv);
as = ifft(erpX .* cmwX, nConv);
as = as((length(cmw)-1)/2 + 1:end-(length(cmw)-1)/2);
figure(5), clf
subplot(121)
plot(timevec,abs(as))
title('Wavelet Convolution')
```

3. **Filter-Hilbert Method:**

CODE:

```
filter_width = 3;
center_freq = 42;
fbounds = [center_freq-filter_width center_freq+filter_width] / (srate/2);
filtkern = fir1(order, fbounds, 'bandpass');
filtsig = filtfilt(filtkern, 1, double(erp));
```

```
fh_amp = abs(hilbert(filtsig));

subplot(122)

plot(timevec, fh_amp)

title('Filter-Hilbert Method')
```

**Highlights:**

- **Complex Morlet Wavelets**: Used to extract power and phase information from neural signals.
- **Time-Frequency Analysis**: Performed on both single-trial and multi-trial data.
- **Edge Effects Exploration**: Analyzed using a square-wave time series.
- **Comparison**: Between wavelet convolution and filter-Hilbert methods for ERP data.

**Explanation of Results:**

- **Wavelet Convolution vs. Filter-Hilbert**: The wavelet convolution method extracts time-frequency information by convolving the signal with Morlet wavelets, allowing precise frequency resolution. The filter-Hilbert method uses band-pass filtering followed by Hilbert transform to get the instantaneous amplitude and phase. Both methods provide insights into the signal's frequency content, but they differ in terms of temporal and spectral precision.

**Conclusion:**

The time-frequency analysis using complex Morlet wavelets provided detailed insights into the power and phase of neural signals across different frequencies. The comparison with the filter-Hilbert method demonstrated that while both approaches are effective, Morlet wavelets offer more flexibility in terms of frequency resolution. The exploration of edge effects highlighted the importance of handling signal boundaries carefully.

The project successfully demonstrated the extraction of power and phase information from neural signals using complex Morlet wavelet convolution. The visualizations provided insights into the time-frequency characteristics of the signals. The exploration of edge effects highlighted the importance of considering artifacts in time-frequency analysis. The comparison with filter-Hilbert methods showed that both approaches can yield similar results, but parameters need to be carefully adjusted for accurate analysis.