*# a.*

A screenshot of a graph

Description automatically generated

b) Convolve each wavelet with EEG data from all electrodes and from only the first trial

In [18]:

*# b.*

convolved\_data

array([[-16.947456 , -23.503107 , -17.0633 , ..., -26.82681 ,

-30.02092 , -17.705837 ],

[-20.31652 , -22.39365 , -10.985493 , ..., 0.3493317 ,

-0.21125636, -0.7934704 ],

[-33.983974 , -27.592262 , -4.465775 , ..., -27.515846 ,

-49.51342 , -43.288696 ],

...,

[-19.17071 , -7.238967 , 10.825006 , ..., -13.468891 ,

-30.038723 , -28.459368 ],

[-20.99644 , -7.128301 , 12.264906 , ..., 20.106596 ,

1.5189717 , -15.421003 ],

[-27.2869 , -21.165905 , -0.8333369 , ..., -31.996515 ,

-49.65603 , -38.924156 ]], dtype=float32)

c) Extract power and phase from the result of the complex wavelet convolution and store in a matrix

**more details of the calculation for part b & c are provided in the code at the end of the report.**

d) Make topographical plots of power and phase at 180 ms at all frequencies

A group of circles with different colored circles

Description automatically generated

e) Repeat step (d) for activity at 360 ms, and 650 ms.

A group of circles with different colors

Description automatically generated

In [29]:

*# e. 650*

A group of circles with different colors

Description automatically generated

**f)**

**Power Topographies:**

Power at 2 Hz (dB): The entire scalp appears to have elevated power, with the central region showing the most pronounced activity. Power at 9 Hz (dB): There's elevated activity at the central and left lateral regions. Power at 16 Hz (dB): The central region, particularly the fronto-central area, exhibits a pronounced decrease in power, surrounded by areas of higher power. Power at 23 Hz (dB): A mixed pattern with some spots of increased power in the central and left lateral regions and decreased power in the fronto-central region. Power at 30 Hz (dB): A pronounced central region with decreased power and elevated power at the peripheral regions. Phase Topographies:

Phase at 2 Hz: A fairly uniform phase distribution with slight phase reversals at the left and right lateral regions. Phase at 9 Hz: Strong phase reversals are observed in the central and frontal regions. Phase at 16 Hz: The frontal and central regions exhibit pronounced phase reversals. Phase at 23 Hz: There's a strong phase reversal in the centro-parietal region. Phase at 30 Hz: The pattern is more complex, with multiple regions showing phase reversals, particularly in the central, parietal, and frontal regions.

**Are there any prominent topographical features in power or in phase?**

Yes, the topographical maps show distinct patterns for both power and phase across various scalp regions.

**Do these differ for different frequencies?**

Yes, the topographical patterns vary across different frequencies. For example, at 2 Hz, power is elevated across the entire scalp, while at 30 Hz, central regions show decreased power. Phase distributions also differ, with distinct phase reversals seen at different frequencies.

**Do power and phase have similar topographical distributions?**

Not entirely. While some regions show overlapping features in power and phase, distinct differences are evident in various frequencies and scalp regions. both power and phase maps exhibit distinct topographical patterns at different frequencies. The power maps tend to have more localized features of high or low power, while the phase maps demonstrate several regions of phase reversals.

**Is there any reason to suspect that they might have similar or different topographies?**

Yes, power and phase often represent different neural dynamics and processes, which can lead to varied topographical distributions. Power typically captures the amplitude of oscillatory activity, while phase reflects the timing or synchrony of neural events. Given these inherent differences, one would expect them to exhibit distinct topographical patterns.

**g)**

A diagram of a graph

Description automatically generated with medium confidence

A close-up of a graph

Description automatically generated

**g) comparison:**

Variable cycles provide a more adaptive and arguably intuitive view of the data, especially when interested in a broad range of frequencies. In contrast, fixed cycles offer a consistent view across frequencies, which can be useful when one desires uniform time and frequency resolution.

1. Time-Frequency Distribution:

Variable Cycles:

The time-frequency representation appears smoother, with broader frequency bands and more gradation between areas of activation. A prominent activity is seen around 25 Hz and between 400-600 ms, appearing as a dark blue patch indicating high power. Additional areas of increased power are seen around 10-15 Hz spanning various time points.

Fixed Cycles:

The representation is more striated, with pronounced vertical stripes throughout. This suggests that there's consistent power across multiple frequencies at specific time points. Like the variable cycles, there's a strong activity around 25 Hz and 400-600 ms, but the contour appears sharper. Elevated power around 10-15 Hz is also evident but appears more intermittent due to the vertical striations.

2. Resolution & Precision:

Variable Cycles:

Offers a more adaptable representation where the width of the wavelet (and thus time and frequency resolution) changes depending on the frequency. This allows for a more precise view at lower frequencies and a broader view at higher frequencies.

Fixed Cycles:

The fixed number of cycles across all frequencies can lead to a compromise between time and frequency resolution. This results in the prominent vertical patterns seen in the heat map due to this fixed-width wavelet convolution at all frequencies.

1. Clarity & Interpretation:

Variable Cycles:

The smoother gradient makes it more intuitive to interpret regions of high and low activity and how they evolve.

Fixed Cycles:

The striated pattern can make it slightly harder to pinpoint exact regions of activity. However, the sharp contours can also be beneficial when trying to identify the exact timings of specific events.

# codes in python

*# a.*

**def** morlet\_wavelet(frequency, num\_cycles, sampling\_rate, duration**=**2):

t **=** np**.**linspace(**-**duration**/**2, duration**/**2, int(sampling\_rate **\*** duration), endpoint**=False**)

sine\_wave **=** np**.**exp(2j **\*** np**.**pi **\*** frequency **\*** t)

amplitude\_envelope **=** np**.**exp(**-**t**\*\***2 **\*** (np**.**pi **\*** frequency **/** num\_cycles)**\*\***2)

wavelet **=** sine\_wave **\*** amplitude\_envelope

**return** wavelet

*# Wavelet parameters*

frequencies **=** [2, 9, 16, 23, 30]

n\_cycles **=** 4

sampling\_freq **=** 256

time\_window **=** 2

wavelets **=** [] *# Empty list to store wavelets*

*# Plotting*

fig, axs **=** plt**.**subplots(len(frequencies), 1, figsize**=**(6, 10), sharex**=True**)

**for** i, freq **in** enumerate(frequencies):

wavelet **=** morlet\_wavelet(freq, n\_cycles, sampling\_freq, time\_window)

wavelets**.**append(wavelet) *# Append wavelet to the list*

axs[i]**.**plot(np**.**real(wavelet), label**=**'Real Part')

axs[i]**.**plot(np**.**imag(wavelet), label**=**'Imaginary Part', linestyle**=**'--')

axs[i]**.**set\_title(f"Morlet Wavelet at {freq} Hz")

axs[i]**.**legend()

plt**.**xlabel("Time (samples)")

fig**.**suptitle('Morlet Wavelets at Different Frequencies', fontsize**=**16)

plt**.**tight\_layout()

plt**.**show()

b) Convolve each wavelet with EEG data from all electrodes and from only the first trial

In [18]:

*# b.*

*# We already have wavelets and their corresponding time vector 't'*

convolved\_data **=** np**.**zeros\_like(First\_trial\_eeg) *# To store the convolved data*

t **=** np**.**linspace(**-**time\_window**/**2, time\_window**/**2, int(sampling\_freq **\*** time\_window), endpoint**=False**)

*# Length of the convolved data*

len\_convolved **=** First\_trial\_eeg**.**shape[1] **+** len(t) **-** 1

print('len\_convolved:', len\_convolved) *#len\_convolved: 1151*

**for** i, wavelet **in** enumerate(wavelets):

*# FFT of the wavelet*

wavelet\_fft **=** fft(wavelet, n**=**len\_convolved)

*# Convolve with each channel*

**for** ch **in** range(First\_trial\_eeg**.**shape[0]):

*# FFT of the EEG data*

eeg\_fft **=** fft(First\_trial\_eeg[ch, :], n**=**len\_convolved)

*# Multiply in frequency domain and then IFFT*

conv\_result **=** ifft(wavelet\_fft **\*** eeg\_fft)

*# Cut the convolved data to original EEG data length*

cut\_length **=** len(t) **//** 2

convolved\_data[ch, :] **=** np**.**real(conv\_result[cut\_length:cut\_length **+** First\_trial\_eeg**.**shape[1]])

*# convolved\_data now contains the result of the convolution of each wavelet with each channel's data*

print('convolved\_data.shape:', convolved\_data**.**shape) *# convolved\_data.shape: (64, 640)*

len\_convolved: 1151

convolved\_data.shape: (64, 640)

In [20]:

convolved\_data

Out[20]:

array([[-16.947456 , -23.503107 , -17.0633 , ..., -26.82681 ,

-30.02092 , -17.705837 ],

[-20.31652 , -22.39365 , -10.985493 , ..., 0.3493317 ,

-0.21125636, -0.7934704 ],

[-33.983974 , -27.592262 , -4.465775 , ..., -27.515846 ,

-49.51342 , -43.288696 ],

...,

[-19.17071 , -7.238967 , 10.825006 , ..., -13.468891 ,

-30.038723 , -28.459368 ],

[-20.99644 , -7.128301 , 12.264906 , ..., 20.106596 ,

1.5189717 , -15.421003 ],

[-27.2869 , -21.165905 , -0.8333369 , ..., -31.996515 ,

-49.65603 , -38.924156 ]], dtype=float32)

c) Extract power and phase from the result of the complex wavelet convolution and store in a matrix

In [23]:

*# c.*

*# Initialize the 4D array to store power and phase results*

results\_matrix **=** np**.**zeros((convolved\_data**.**shape[1], len(frequencies), convolved\_data**.**shape[0], 2))

**for** f\_idx, wavelet **in** enumerate(wavelets):

wavelet\_length **=** len(wavelet)

*# Calculate the length of the convolved data*

n\_convolution **=** 640 **+** wavelet\_length

*# FFT of the wavelet*

wavelet\_fft **=** np**.**fft**.**fft(wavelet, n**=**n\_convolution)

**for** ch\_idx **in** range(64):

*# EEG signal for this channel*

eeg\_signal **=** First\_trial\_eeg[ch\_idx, :]

*# FFT of the EEG signal*

eeg\_fft **=** np**.**fft**.**fft(eeg\_signal, n**=**n\_convolution)

*# Convolution in the frequency domain*

conv\_result\_fft **=** wavelet\_fft **\*** eeg\_fft

*# Inverse FFT to return to the time domain*

conv\_result **=** np**.**fft**.**ifft(conv\_result\_fft)

*# Correcting the length of the convolved data*

*# Ensure integer division and proper trimming*

half\_wavelet **=** int(np**.**ceil((wavelet\_length **-** 1) **/** 2))

conv\_result **=** conv\_result[half\_wavelet:**-**half\_wavelet **or** **None**]

*# Check and adjust the length if necessary*

**if** conv\_result**.**size **!=** 640:

**raise** ValueError(f"Unexpected size of convolved result: {conv\_result**.**size}")

*# Calculate power and phase*

power **=** np**.**abs(conv\_result) **\*\*** 2

phase **=** np**.**angle(conv\_result)

*# Store in the matrix*

results\_matrix[:, f\_idx, ch\_idx, 0] **=** power

results\_matrix[:, f\_idx, ch\_idx, 1] **=** phase

print('results\_matrix.shape:', results\_matrix**.**shape)

results\_matrix.shape: (640, 5, 64, 2)

In [24]:

*# Shape of the entire results\_matrix*

print("Shape of the results\_matrix:", results\_matrix**.**shape)

*# To check the shape of power and phase for a specific frequency and electrode*

f\_idx **=** 0 *# For example, the first frequency*

ch\_idx **=** 0 *# For example, the first electrode*

*# Extracting power and phase for this specific frequency and electrode*

power\_example **=** results\_matrix[:, f\_idx, ch\_idx, 0]

phase\_example **=** results\_matrix[:, f\_idx, ch\_idx, 1]

print("Shape of power for frequency index", f\_idx, "and electrode index", ch\_idx, ":", power\_example**.**shape)

print("Shape of phase for frequency index", f\_idx, "and electrode index", ch\_idx, ":", phase\_example**.**shape)

Shape of the results\_matrix: (640, 5, 64, 2)

Shape of power for frequency index 0 and electrode index 0 : (640,)

Shape of phase for frequency index 0 and electrode index 0 : (640,)

d) Make topographical plots of power and phase at 180 ms at all frequencies

In [27]:

*# d.*

*# Find the index for 180 ms*

time\_idx **=** np**.**argmin(np**.**abs(times **-** 180)) *#302*

*# Define frequencies*

frequencies **=** [2, 9, 16, 23, 30] *# Adjust according to your frequencies*

*# Calculate the global minimum and maximum for power (in dB) and phase*

power\_min **=** 10 **\*** np**.**log10(results\_matrix[:, :, :, 0])**.**min()

power\_max **=** 10 **\*** np**.**log10(results\_matrix[:, :, :, 0])**.**max()

phase\_min **=** results\_matrix[:, :, :, 1]**.**min()

phase\_max **=** results\_matrix[:, :, :, 1]**.**max()

*# Set up figure*

fig, axs **=** plt**.**subplots(2, len(frequencies), figsize**=**(20, 6)) *# Adjust the size as needed*

*# Loop over frequencies to plot power and phase*

**for** i, freq **in** enumerate(frequencies):

*# Power (applying 10log10 and normalizing)*

power\_data **=** 10 **\*** np**.**log10(results\_matrix[time\_idx, i, :, 0])

im, \_ **=** mne**.**viz**.**plot\_topomap(power\_data, info, axes**=**axs[0, i], show**=False**, cmap**=**'jet')

axs[0, i]**.**set\_title(f'Power at {freq} Hz (dB)')

*# Phase (normalizing)*

phase\_data **=** results\_matrix[time\_idx, i, :, 1]

im, \_ **=** mne**.**viz**.**plot\_topomap(phase\_data, info, axes**=**axs[1, i], show**=False**, cmap**=**'jet')

axs[1, i]**.**set\_title(f'Phase at {freq} Hz')

*# Adding a common colorbar for power*

cax1 **=** fig**.**add\_axes([0.92, 0.55, 0.01, 0.3]) *# Adjust these values as needed for positioning*

norm **=** Normalize(vmin**=**power\_min, vmax**=**power\_max)

cb1 **=** plt**.**colorbar(ScalarMappable(norm**=**norm, cmap**=**'jet'), cax**=**cax1, orientation**=**'vertical')

cb1**.**set\_label('Power (dB)')

*# Adding a common colorbar for phase*

cax2 **=** fig**.**add\_axes([0.92, 0.15, 0.01, 0.3]) *# Adjust these values as needed for positioning*

norm **=** Normalize(vmin**=**phase\_min, vmax**=**phase\_max)

cb2 **=** plt**.**colorbar(ScalarMappable(norm**=**norm, cmap**=**'jet'), cax**=**cax2, orientation**=**'vertical')

cb2**.**set\_label('Phase')

plt**.**tight\_layout(rect**=**[0, 0, 0.9, 1])

plt**.**show()

C:\Users\Maryam\AppData\Local\Temp\ipykernel\_3936\1682916526.py:42: UserWarning: This figure includes Axes that are not compatible with tight\_layout, so results might be incorrect.

plt.tight\_layout(rect=[0, 0, 0.9, 1])

e) Repeat step (d) for activity at 360 ms, and 650 ms.

In [28]:

*# e. 360ms*

*# Find the index for 360ms*

time\_idx **=** np**.**argmin(np**.**abs(times **-** 360))

*# Define frequencies*

frequencies **=** [2, 9, 16, 23, 30] *# Adjust according to your frequencies*

*# Calculate the global minimum and maximum for power (in dB) and phase*

power\_min **=** 10 **\*** np**.**log10(results\_matrix[:, :, :, 0])**.**min()

power\_max **=** 10 **\*** np**.**log10(results\_matrix[:, :, :, 0])**.**max()

phase\_min **=** results\_matrix[:, :, :, 1]**.**min()

phase\_max **=** results\_matrix[:, :, :, 1]**.**max()

*# Set up figure*

fig, axs **=** plt**.**subplots(2, len(frequencies), figsize**=**(20, 6)) *# Adjust the size as needed*

*# Loop over frequencies to plot power and phase*

**for** i, freq **in** enumerate(frequencies):

*# Power (applying 10log10 and normalizing)*

power\_data **=** 10 **\*** np**.**log10(results\_matrix[time\_idx, i, :, 0])

im, \_ **=** mne**.**viz**.**plot\_topomap(power\_data, info, axes**=**axs[0, i], show**=False**, cmap**=**'jet')

axs[0, i]**.**set\_title(f'Power at {freq} Hz (dB)')

*# Phase (normalizing)*

phase\_data **=** results\_matrix[time\_idx, i, :, 1]

im, \_ **=** mne**.**viz**.**plot\_topomap(phase\_data, info, axes**=**axs[1, i], show**=False**, cmap**=**'jet')

axs[1, i]**.**set\_title(f'Phase at {freq} Hz')

*# Adding a common colorbar for power*

cax1 **=** fig**.**add\_axes([0.92, 0.55, 0.01, 0.3]) *# Adjust these values as needed for positioning*

norm **=** Normalize(vmin**=**0, vmax**=**50)

cb1 **=** plt**.**colorbar(ScalarMappable(norm**=**norm, cmap**=**'jet'), cax**=**cax1, orientation**=**'vertical')

cb1**.**set\_label('Power (dB)')

*# Adding a common colorbar for phase*

cax2 **=** fig**.**add\_axes([0.92, 0.15, 0.01, 0.3]) *# Adjust these values as needed for positioning*

norm **=** Normalize(vmin**=-**3, vmax**=**3)

cb2 **=** plt**.**colorbar(ScalarMappable(norm**=**norm, cmap**=**'jet'), cax**=**cax2, orientation**=**'vertical')

cb2**.**set\_label('Phase')

plt**.**tight\_layout(rect**=**[0, 0, 0.9, 1])

plt**.**show()

C:\Users\Maryam\AppData\Local\Temp\ipykernel\_3936\3697760241.py:42: UserWarning: This figure includes Axes that are not compatible with tight\_layout, so results might be incorrect.

plt.tight\_layout(rect=[0, 0, 0.9, 1])

In [29]:

*# e. 650*

*# Find the index for 650 ms*

time\_idx **=** np**.**argmin(np**.**abs(times **-** 650))

*# Define frequencies*

frequencies **=** [2, 9, 16, 23, 30] *# Adjust according to your frequencies*

*# Calculate the global minimum and maximum for power (in dB) and phase*

power\_min **=** 10 **\*** np**.**log10(results\_matrix[:, :, :, 0])**.**min()

power\_max **=** 10 **\*** np**.**log10(results\_matrix[:, :, :, 0])**.**max()

phase\_min **=** results\_matrix[:, :, :, 1]**.**min()

phase\_max **=** results\_matrix[:, :, :, 1]**.**max()

*# Set up figure*

fig, axs **=** plt**.**subplots(2, len(frequencies), figsize**=**(20, 6)) *# Adjust the size as needed*

*# Loop over frequencies to plot power and phase*

**for** i, freq **in** enumerate(frequencies):

*# Power (applying 10log10 and normalizing)*

power\_data **=** 10 **\*** np**.**log10(results\_matrix[time\_idx, i, :, 0])

im, \_ **=** mne**.**viz**.**plot\_topomap(power\_data, info, axes**=**axs[0, i], show**=False**, cmap**=**'jet')

axs[0, i]**.**set\_title(f'Power at {freq} Hz (dB)')

*# Phase (normalizing)*

phase\_data **=** results\_matrix[time\_idx, i, :, 1]

im, \_ **=** mne**.**viz**.**plot\_topomap(phase\_data, info, axes**=**axs[1, i], show**=False**, cmap**=**'jet')

axs[1, i]**.**set\_title(f'Phase at {freq} Hz')

*# Adding a common colorbar for power*

cax1 **=** fig**.**add\_axes([0.92, 0.55, 0.01, 0.3]) *# Adjust these values as needed for positioning*

norm **=** Normalize(vmin**=**0, vmax**=**50)

cb1 **=** plt**.**colorbar(ScalarMappable(norm**=**norm, cmap**=**'jet'), cax**=**cax1, orientation**=**'vertical')

cb1**.**set\_label('Power (dB)')

*# Adding a common colorbar for phase*

cax2 **=** fig**.**add\_axes([0.92, 0.15, 0.01, 0.3]) *# Adjust these values as needed for positioning*

norm **=** Normalize(vmin**=-**3, vmax**=**3)

cb2 **=** plt**.**colorbar(ScalarMappable(norm**=**norm, cmap**=**'jet'), cax**=**cax2, orientation**=**'vertical')

cb2**.**set\_label('Phase')

plt**.**tight\_layout(rect**=**[0, 0, 0.9, 1])

plt**.**show()

C:\Users\Maryam\AppData\Local\Temp\ipykernel\_3936\619713156.py:42: UserWarning: This figure includes Axes that are not compatible with tight\_layout, so results might be incorrect.

plt.tight\_layout(rect=[0, 0, 0.9, 1])

f)

In [35]:

*#g.*

**def** convolve\_with\_wavelet(data, wavelet):

n\_signal **=** data**.**shape[1]

n\_wavelet **=** len(wavelet)

n\_convolution **=** n\_signal **+** n\_wavelet **-** 1

data\_fft **=** np**.**fft**.**fft(data, n\_convolution, axis**=**1)

wavelet\_fft **=** np**.**fft**.**fft(wavelet, n\_convolution)[np**.**newaxis, :]

convolution\_result\_fft **=** data\_fft **\*** wavelet\_fft

convolution\_result **=** np**.**fft**.**ifft(convolution\_result\_fft, axis**=**1)

*# Cut the data*

start **=** (n\_wavelet **-** 1) **//** 2

end **=** start **+** n\_signal

**return** convolution\_result[:, start:end]

*# Function to calculate power for variable or fixed cycles*

**def** calculate\_power(cycles):

power\_matrix **=** []

**for** freq, cycle **in** zip(frequencies, cycles):

*# Extract data for 'FCz'*

data\_fcz **=** eeg\_data[fcz\_index, :, :] *# Assuming trials are on the second dimension*

*# Calculate power for each trial*

power\_trials **=** []

**for** trial\_idx **in** range(data\_fcz**.**shape[1]):

trial\_2D **=** data\_fcz[:, trial\_idx]**.**reshape(1, **-**1) *# Making trial 2D for the convolution function*

wavelet **=** morlet\_wavelet(freq, cycle, sampling\_rate)

convolution **=** convolve\_with\_wavelet(trial\_2D, wavelet)

power **=** np**.**abs(convolution)**\*\***2

power\_trials**.**append(power[0])

*# Average power across trials*

avg\_power **=** np**.**mean(power\_trials, axis**=**0)

power\_matrix**.**append(avg\_power)

**return** np**.**array(power\_matrix)

*# Baseline correction function*

**def** baseline\_correction(power\_matrix, times, baseline\_range):

baseline\_timepoints **=** np**.**logical\_and(times **>=** baseline\_range[0], times **<=** baseline\_range[1])

baseline\_mean **=** np**.**mean(power\_matrix[:, baseline\_timepoints], axis**=**1, keepdims**=True**)

**return** power\_matrix **/** baseline\_mean *# Division for baseline correction*

*# Constants*

baseline\_range **=** (**-**500, **-**200) *# in milliseconds*

frequencies **=** np**.**arange(2, 41, 2) *# 2 to 40 Hz*

cycles **=** np**.**linspace(3, 10, len(frequencies)) *# Variable cycles from 3 to 10*

sampling\_rate**=**256

*# Find the channel index for 'FCz'*

fcz\_index **=** raw**.**ch\_names**.**index('Fcz')

*# Calculate power for variable and fixed cycles*

power\_variable\_cycles **=** calculate\_power(cycles)

power\_fixed\_cycles **=** calculate\_power([4] **\*** len(frequencies)) *# Fixed cycle: 4*

*# Apply baseline correction*

corrected\_variable **=** baseline\_correction(power\_variable\_cycles, times, baseline\_range)

corrected\_fixed **=** baseline\_correction(power\_fixed\_cycles, times, baseline\_range)

*# Plotting function*

**def** plot\_time\_frequency(data, title):

plt**.**figure(figsize**=**(6, 4))

plt**.**imshow(10 **\*** np**.**log10(data), aspect**=**'auto', cmap**=**'jet', origin**=**'lower',

extent**=**[times[205], times[512], frequencies[0], frequencies[**-**1]],

vmin**=-**3, vmax**=**3) *# Log transform of power; color limits as specified*

plt**.**colorbar(format**=**'%+2.0f dB')

plt**.**xlabel('Time (ms)')

plt**.**ylabel('Frequency (Hz)')

plt**.**title(title)

plt**.**tight\_layout()

plt**.**show()

*# Plot the results*

plot\_time\_frequency(corrected\_variable, 'Time-Frequency (Variable Cycles)')

plot\_time\_frequency(corrected\_fixed, 'Time-Frequency (Fixed Cycles)')