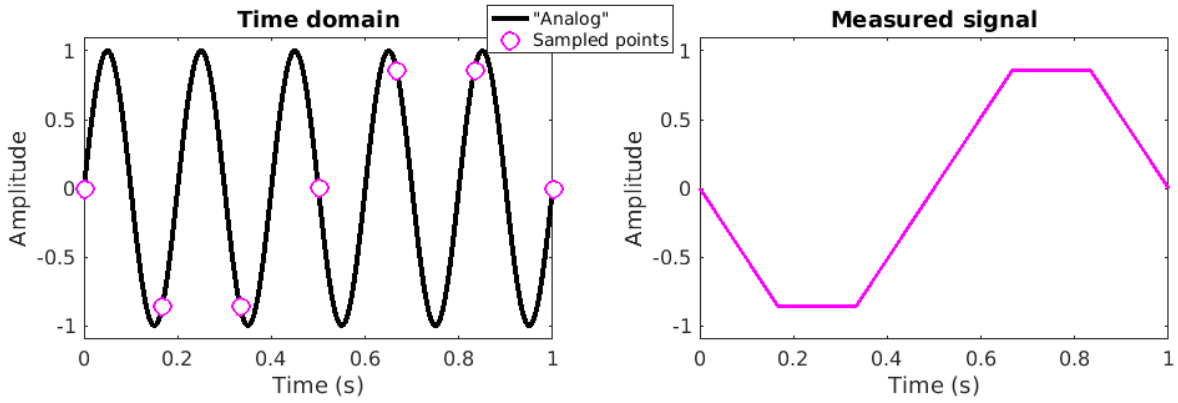
Course reader: *Aliasing, stationarity, and violations*

- Aliasing is the phenomenon that a feature of a signal or image that is higher than the Nyquist frequency, appears at a lower frequency in the measurement. You can see an example of aliasing below.



- The best way to avoid aliasing is to have a Nyquist rate above the highest feature in the signal. Some digitizers or other measurement devices have a built-in low-pass filter that will remove any features of the signal above the Nyquist. Ultimately, understanding the signal you want to measure is the best way to determine the appropriate sampling rate.

- Stationarity refers to the reproducibility of the descriptive statistics of a signal, such as mean, variance, frequency, and so on. For example: You measure variance in different time windows in different regions of the signal; if the variance measurements are all statistically non-significantly different, then the signal is stationary.
- If the signal violates stationarity, then it is a non-stationary signal.

- There are three sources of ambiguity in this definition:
  1. **Many features of a signal**: A signal could be mean-stationary but frequency-non-stationary.
  2. **Window size and placement**: A signal might look stationary or non-stationary depending on the size of the time windows (e.g., 100 ms vs. 10 seconds).
  3. **Threshold-dependence**: In sampled data, particularly in the presence of noise, there is no precise thresholding for calling different values of descriptive statistics "different." Therefore, thresholds must be selected, which could entail bias. And the appropriate threshold might be different in different kinds of signals.

  There is no single, analytically precise solution to this problem; instead, different disciplines (e.g., econometrics vs. neuroscience) have developed solutions that are appropriate given the characteristics of the signals under investigation.

- But the important point is how non-stationarities affect the results of the Fourier transform: Non-stationary signals have power spectra that can be difficult to interpret, and the stronger the non-stationarity, the more difficult the power spectrum could be.
- Let me be clear about this: The Fourier transform is a perfect (lossless) transform, and its validity does not depend on the features of the signal. Instead, I'm referring here only to the ability to look at a power spectrum and infer the important features of the time-domain signal. Non-stationarities

are really detrimental for this latter point.

- This is an important limitation, because many real signals have lots of non-stationarities. For example, the brain is able to produce thought, memories, sensations, and so on, only because of its non-stationarities.

- Many solutions to the non-stationarity issue involve *time-frequency analyses*, which involve examining how the spectral structure of a signal changes over time or space. This topic is too big to cover here, although I show a few examples in the "Solutions" video.

Exercises

1. This question is similar to exercise #3 from the FFT section, but here using real data. The online material includes a file called LFPdata.mat, which contains 200 repetitions of brain electrical activity (LFP=local field potential), where time=0 is when a picture appeared on a computer screen. Compute and show the following power spectra:

1. Average the time-domain repetitions together, and then compute the power spectrum.
2. Compute the FFT for each repetition, then average the complex Fourier coefficients together, and then extract the power spectrum.
3. Compute the FFT for each repetition, then extract the power spectrum separately for each repetition, and then average the power spectra.

2. Create a multipolar chirp using frequencies that change over time in a triangular fashion. The following steps will help you:

1. Create a triangle time series. There are several ways to compute a triangle time series; one equation you can use is $a\|m\mathbf{t} \mod 2 - 1\|$, where a is an amplitude modulating factor, $\mathbf{t}$ $\mod 2$ means the modulus, or remainder, of dividing time vector $\mathbf{t}$ by 2, m is a frequency modulator, and $\|\|$ indicates the absolute value.
   This triangle time series is the vector of frequencies at each time point.
2. Compute a vector $\mathbf{y}$ as the mean-centered cumulative sum of the triangle computed above.
3. Finally, the signal can be defined as
   $s = \sin(2\pi ft + y2\pi/q)$
   where f is the average of the triangle time series, t is time, and q is the sampling rate in Hz (assuming t is in seconds).

3. Show the amplitude spectrum of the chirp. Is it interpretable? Does that depend on the parameters you select (a and m are the primary parameters to focus on changing)?

4. Clearly, this signal is non-stationary. Is it possible to get a more visually interpretable amplitude spectrum by taking the FFT of a subset of the time series?

5. I didn't talk in much detail about the mechanisms of time-frequency analyses, but the short-time FFT is just a minor extension of the "full-time" FFT. Copy/paste the code from the "Solutions..." video to implement a time-frequency analysis on this signal. Is this result more interpretable? Try it again with different parameters for the time series.

6. If you copy/paste the STFFT code with minimal adjustments, does the time-frequency power plane match the temporal dynamics of the signal? The correct answer is "yeah sort of, but not exactly." Your goal is to explain why this happens.

MATLAB Answers

Scroll down for Python solutions.

1. My code is below. Note that the time-domain averaging and coefficients averaging approaches are identical, because both involve averaging phases together. It's interesting that this situation with real data is closer to the "random-phase" simulation from the FFT section (question #5).

```
% load the LFP data
load LFPdata.mat
n = length(timevec);

% option 1
pow1 = (2*abs(fft( mean(lfp,2) )/n)).^2;

% option 2
pow2 = (2*abs( mean(fft(lfp,[],1)/n ,2) )).^2;

% option 3
pow3 = mean( (2*abs(fft(lfp,[],1)/n)).^2 ,2);

% frequencies
hz = linspace(0,srate/2,floor(n/2)+1);

% now plot
clf
plot(hz,pow1(1:length(hz)),'b','linew',2), hold on
plot(hz,pow2(1:length(hz)),'r','linew',2)
plot(hz,pow3(1:length(hz)),'k','linew',2)

legend({'Time-domain ave';'Fourier coefs ave';'Power spect ave'})
set(gca,'xlim',[0 100])
xlabel('Frequency (Hz)'), ylabel('Amplitude')
```

2. One solution:

```
% simulation details
srate  = 1000;
t      = 0:1/srate:5;
n      = length(t);
a      = 10; % amplitude modulator
m      = 2; % frequency modulator
freqTS = a*abs(mod(m*t,2)-1);

% create signal
cf = mean(freqTS);
k = 2*pi/srate;
```

```
y = sin(2*pi.*cf.*t + k*cumsum(freqTS-cf));

% plot instantaneous frequencies
subplot(211), plot(t,freqTS)
xlabel('Time (s)'), ylabel('Frequency (Hz)')

% plot sinusoidal signal derived from time-varying instantaneous frequencies
subplot(212), plot(t,y)
xlabel('Time (s)'), ylabel('Amplitude')
```

3. I know I often "cheat" by multiplying the DC by 2. Sadly, many people do. Here I do it the right way. The answer to the interpretability issue is that if you select a and m such that there is little frequency non-stationarity, then (not surprisingly) the static amplitude spectrum becomes more interpretable.

```
hz = linspace(0,srate/2,floor(n/2)+1);
amp = abs(fft(y)/n);
amp(2:end) = 2*amp(2:end);

stem(hz,amp(1:length(hz)),'ks-','linew',2)
set(gca,'xlim',[0 30])
```

4. Again, this depends on the simulation parameters and on the time window. The point is to appreciate (1) the relationship between visual features of the amplitude spectrum and stationarity, and (2) the importance of time window selection for the interpretation of stationarity.

5. You can just copy/paste the code from the short-time FFT section of the *Fourier_aliasStation.m* script. You may need to make a few minor adjustments to variable names.

6. There is a phase shift in the time-frequency plane. For example, the maximal frequency at the first time point is 5 Hz, but the vector of instantaneous frequencies starts at 10. The reason why this happens is that the window for the short time FFT always goes forward by 500 points (which is 500 ms if you use a sampling rate of 1000 Hz). That means the spectrum at the first time point will be a reflection of the first 500 ms of the signal, and the first 500 ms includes frequencies ranging from 10 Hz to 0 Hz, which means the average is 5 Hz.

Python Answers

These answers are identical to those above, but with Python code.

1. My code is below. Note that the time-domain averaging and coefficients averaging approaches are identical, because both involve averaging phases together. It's interesting that this situation with real data is closer to the "random-phase" simulation from the FFT section (question #5).

```
import numpy as np
import matplotlib.pyplot as plt

# load the LFP data
import scipy.io
mat = scipy.io.loadmat('LFPdata.mat')
timevec = mat['timevec'][0]
lfp = mat['lfp']

n = len(timevec)

# option 1
pow1 = (2*np.abs(np.fft.fft( np.mean(lfp,axis=1) )/n))**2

# option 2
pow2 = (2*np.abs( np.mean(np.fft.fft(lfp,axis=0)/n ,axis=1) ))**2

# option 3
pow3 = np.mean( (2*np.abs(np.fft.fft(lfp,axis=0)/n))**2 ,axis=1)

# frequencies
hz = np.linspace(0,srate/2,int(np.floor(n/2)+1))

# now plot
plt.plot(hz,pow1[:len(hz)],'b')
plt.plot(hz,pow2[:len(hz)],'r')
plt.plot(hz,pow3[:len(hz)],'k')

plt.legend(['Time-domain ave','Fourier coefs ave','Power spect ave'])
plt.xlim([0,100])
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude')
plt.show()
```

2. One solution:

```
# simulation details
srate  = 1000
t      = np.arange(0,5,1/srate)
```

```
n      = len(t)
a      = 10 # amplitude modulator
m      =  2 # frequency modulator
freqTS = a*np.abs(np.mod(m*t,2)-1)

# create signal
cf = np.mean(freqTS)
k = 2*np.pi/srate
y = np.sin(2*np.pi*cf*t + k*np.cumsum(freqTS-cf))

# plot instantaneous frequencies
_,axs = plt.subplots(2,1)
axs[0].plot(t,freqTS)
axs[0].set_xlabel('Time (s)')
axs[0].set_ylabel('Frequency (Hz)')

# plot sinusoidal signal derived from
#  time-varying instantaneous frequencies
axs[1].plot(t,y)
axs[1].set_xlabel('Time (s)')
axs[1].set_ylabel('Amplitude')

plt.tight_layout()
plt.show()
```

3. I know I often "cheat" by multiplying the DC by 2. Sadly, many people do. Here I do it the right way. The answer to the interpretability issue is that if you select a and m such that there is little frequency non-stationarity, then (not surprisingly) the static amplitude spectrum becomes more interpretable.

```
hz = np.linspace(0,srate/2,int(np.floor(n/2)+1))
amp = np.abs(np.fft.fft(y)/n)
amp[1:] = 2*amp[1:]

plt.stem(hz,amp[:len(hz)],'ks-',use_line_collection=True)
plt.xlim([0,30])
plt.show()
```

4. Again, this depends on the simulation parameters and on the time window. The point is to appreciate (1) the relationship between visual features of the amplitude spectrum and stationarity, and (2) the importance of time window selection for the interpretation of stationarity.

5. You can just copy/paste the code from the short-time FFT section of the *Fourier_aliasStation.m* script. You may need to make a few minor adjustments to variable names.

6. There is a phase shift in the time-frequency plane. For example, the maximal frequency at the first time point is 5 Hz, but the vector of instantaneous frequencies starts at 10. The reason why this happens is that the window for the short time FFT always goes forward by 500 points (which is 500 ms if you use a sampling rate of 1000 Hz). That means the spectrum at the first time point will be a reflection of the first 500 ms of the signal, and the first 500 ms includes frequencies ranging from 10 Hz to 0 Hz, which means the average is 5 Hz.