# Course reader: *Frequency resolution and zero padding*

- The sampling theorem describes the frequency limit that you can measure, given your sampling rate. There are two important take-home messages about sampling:
  1. The *Nyquist frequency* is one-half of the sampling rate, and is the highest frequency in the signal you can—in theory—measure. Dynamics in the signal above that rate will be lost in the measurement, and could become aliased into lower frequencies (more on aliasing in the next section).
  2. The Nyquist limit is not a good practical limit. If you are interested in 500 Hz dynamics, then sampling at 1 kHz is a bad idea, because the quality of the measured signal will depend on the phase relationship between the signal and the sample points. A good guideline is to sample at least 5 times the highest frequency you think will be in the signal.

- *Frequency resolution* (sometimes also called spectral resolution) refers to the spacing between successive frequencies. For example, if the vector of frequencies is 0, 1, 2, 3, ..., then the frequency resolution is 1 Hz.

- Frequency resolution can be calculated by taking the distance between two successive frequencies (including the first two, and the first frequency is always 0, so the resolution is actually just the first non-zero frequency), or by the equation $r = s/n$ where $r$ is the frequency resolution, $s$ is the sampling rate, and $n$ is the number of time points in the FFT.

- That equation makes it clear that resolution is related to both sampling rate and N.
- However, in applications it is more common to keep the sampling rate fixed. Thus, for a fixed sampling rate, the frequency resolution is determined by the number of time points: More time points $\rightarrow$ higher resolution.

- The relationships among sampling rate, signal length, frequency resolution, and temporal resolution can get confusing. It's kindof opposite in the time vs. frequency domains:
  - In the time domain, the resolution is determined by the sampling rate, not the signal length.
  - In the frequency domain, the resolution is determined by the signal length for a fixed sampling rate.

- Therefore, you can increase frequency resolution by adding zeros to the end of the signal. This procedure is called *zero padding*. The zeros don't add any new information; they just increase the N.

- In implementation, you don't need to add the zeros manually; you can specify an optional input into the `fft` function. Just make sure to input the **total N** (length of signal + zeros to pad), not the number of zeros to pad.

- There are three reasons to zero-pad a signal:
  1. To obtain a specific frequency that is not present in the native resolution.
  2. To match FFT lengths for some signal-processing methods like convolution. You'll see an example of this in the video "Solutions for non-stationary time series." There, the Fourier spectrum of a wavelet is point-wise multiplied by the Fourier spectrum of a signal. But the signal and wavelet have different lengths, and so they must be zero-padded to the same length

for the multiplication to be valid.

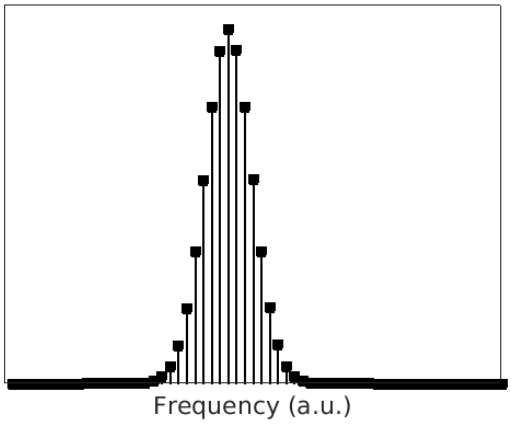
3. To make the power spectral plot look smoother (a.k.a. nicer).

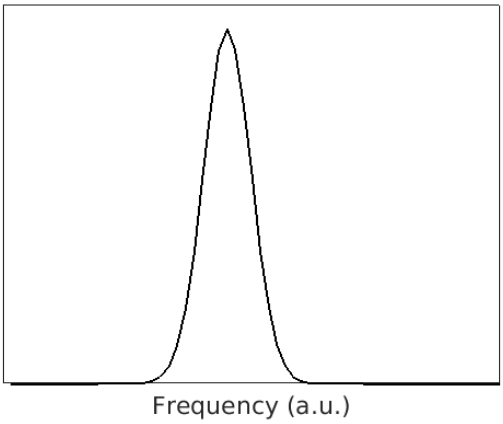
- Zero-padding in the time domain increases frequency resolution. Zero-padding in the frequency domain increases temporal resolution. This is stated by the zero-padding theorem. In the frequency domain, the zeros are added at the Nyquist, i.e., the middle of the spectrum. Again, you don't do this manually — the `ifft` function will add the zeros if you specify that as an additional input.

# Exercises

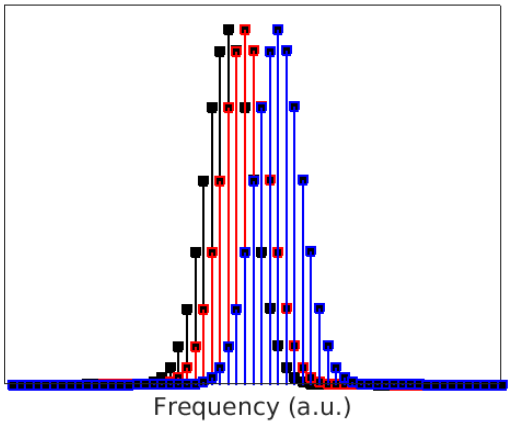
**1.** Create a 2-second sine wave at 11 Hz. Compute and plot its amplitude spectrum. Next, recompute the Fourier transform after zero-padding to a total of 4 seconds. Plot the amplitude spectrum after (1) dividing the Fourier coefficients by the number of time points in the original sine wave and (2) dividing the coefficients by the number of time points including the padded zeros. Which normalization returns the accurate amplitude?

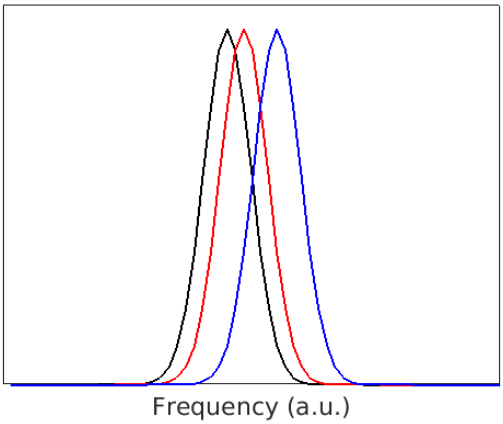
**2.** Which of the following two plots is "better" in the sense of providing a more accurate depiction of the data, and why?




**3.** Which of the following two plots is "better" in the sense of being able to understand the prominent features of the signals?




**4.** Could you apply the Fourier transform to a signal with non-regularly sampled points? For example, if the measurements took place at samples 1, 2, 5, 8, 13, 22, etc.

# MATLAB Answers

Scroll down for the Python solutions.

**1.** My code is below. The correct answer depends on what you want to normalize for: Scaling by the N of the original signal normalizes for the original frequency component, whereas scaling by the zero-padded N normalizes for the new signal distribution, possibly including nonstationarities introduced by the extra zeros.

```
srate = 1000;
time = 0:1/srate:2-1/srate;
s = sin(2*pi*10*time);


sx1 = 2*abs( fft(s)/length(time) );
sx2 = 2*abs( fft(s,4*srate)/length(time) );
sx3 = 2*abs( fft(s,4*srate)/(4*srate) );

hz1 = linspace(0,srate/2,floor(length(time)/2)+1);
hz2 = linspace(0,srate/2,floor((4*srate)/2)+1);

clf
subplot(311)
stem(hz1,sx1(1:length(hz1)),'ks-','linew',3)
set(gca,'xlim',[5 15],'ylim',[0 1])
xlabel('Frequency (Hz)'), ylabel('Amplitude (a.u.)')

subplot(312)
stem(hz2,sx2(1:length(hz2)),'ks-','linew',3)
set(gca,'xlim',[5 15],'ylim',[0 1])
xlabel('Frequency (Hz)'), ylabel('Amplitude (a.u.)')

subplot(313)
stem(hz2,sx3(1:length(hz2)),'ks-','linew',3)
set(gca,'xlim',[5 15],'ylim',[0 1])
xlabel('Frequency (Hz)'), ylabel('Amplitude (a.u.)')
```

**2.** The stem plot (left) is better. The smooth line between frequencies implies that you know what the amplitude is between each measured frequency, but that's not the case in discretely digitized samples.

**3.** This one is a bit more subjective. An old-school purist would say the stem plot is better for the reason I wrote above. However, when comparing spectra from multiple signals, stem plots can become indecipherable to the point of obscuring the information it is supposed to reveal! So in applications, line plots are often used.

**4.** Strictly speaking, the answer is No, you cannot apply the Fourier transform to a non-regularly sampled signal. The best approach here would be to interpolate the signal to regularly spaced points, and then apply the Fourier transform to the interpolated signal.

# Python Answers

Answers here are identical to those above but with Python code.

**1.** My code is below. The correct answer depends on what you want to normalize for: Scaling by the N of the original signal normalizes for the original frequency component, whereas scaling by the zero-padded N normalizes for the new signal distribution, possibly including nonstationarities introduced by the extra zeros.

```
import numpy as np
import matplotlib.pyplot as plt
srate = 1000
time  = np.arange(0,2-1/srate,1/srate)
s = np.sin(2*np.pi*10*time)


sx1 = 2*np.abs( np.fft.fft(s)/len(time) )
sx2 = 2*np.abs( np.fft.fft(s,4*srate)/len(time) )
sx3 = 2*np.abs( np.fft.fft(s,4*srate)/(4*srate) )

hz1 = np.linspace(0,srate/2,int(np.floor(len(time)/2)+1))
hz2 = np.linspace(0,srate/2,int(np.floor((4*srate)/2)+1))

_,axs = plt.subplots(3,1,figsize=(5,10))

axs[0].stem(hz1,sx1[:len(hz1)],'ks-',use_line_collection=True)
axs[0].set_xlim([5,15])
axs[0].set_ylim([0,1])
axs[0].set_xlabel('Frequency (Hz)')
axs[0].set_ylabel('Amplitude (a.u.)')

axs[1].stem(hz2,sx2[:len(hz2)],'ks-',use_line_collection=True)
axs[1].set_xlim([5,15])
axs[1].set_ylim([0,1])
axs[1].set_xlabel('Frequency (Hz)')
axs[1].set_ylabel('Amplitude (a.u.)')

axs[2].stem(hz2,sx3[:len(hz2)],'ks-',use_line_collection=True)
axs[2].set_xlim([5,15])
axs[2].set_ylim([0,1])
axs[2].set_xlabel('Frequency (Hz)')
axs[2].set_ylabel('Amplitude (a.u.)')

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

**2.** The stem plot (left) is better. The smooth line between frequencies implies that you know what the amplitude is between each measured frequency, but that's not the case in discretely digitized samples.

**3.** This one is a bit more subjective. An old-school purist would say the stem plot is better for the reason I wrote above. However, when comparing spectra from multiple signals, stem plots can become indecipherable to the point of obscuring the information it is supposed to reveal! So in applications, line plots are often used.

**4.** Strictly speaking, the answer is No, you cannot apply the Fourier transform to a non-regularly sampled signal. The best approach here would be to interpolate the signal to regularly spaced points, and then apply the Fourier transform to the interpolated signal.