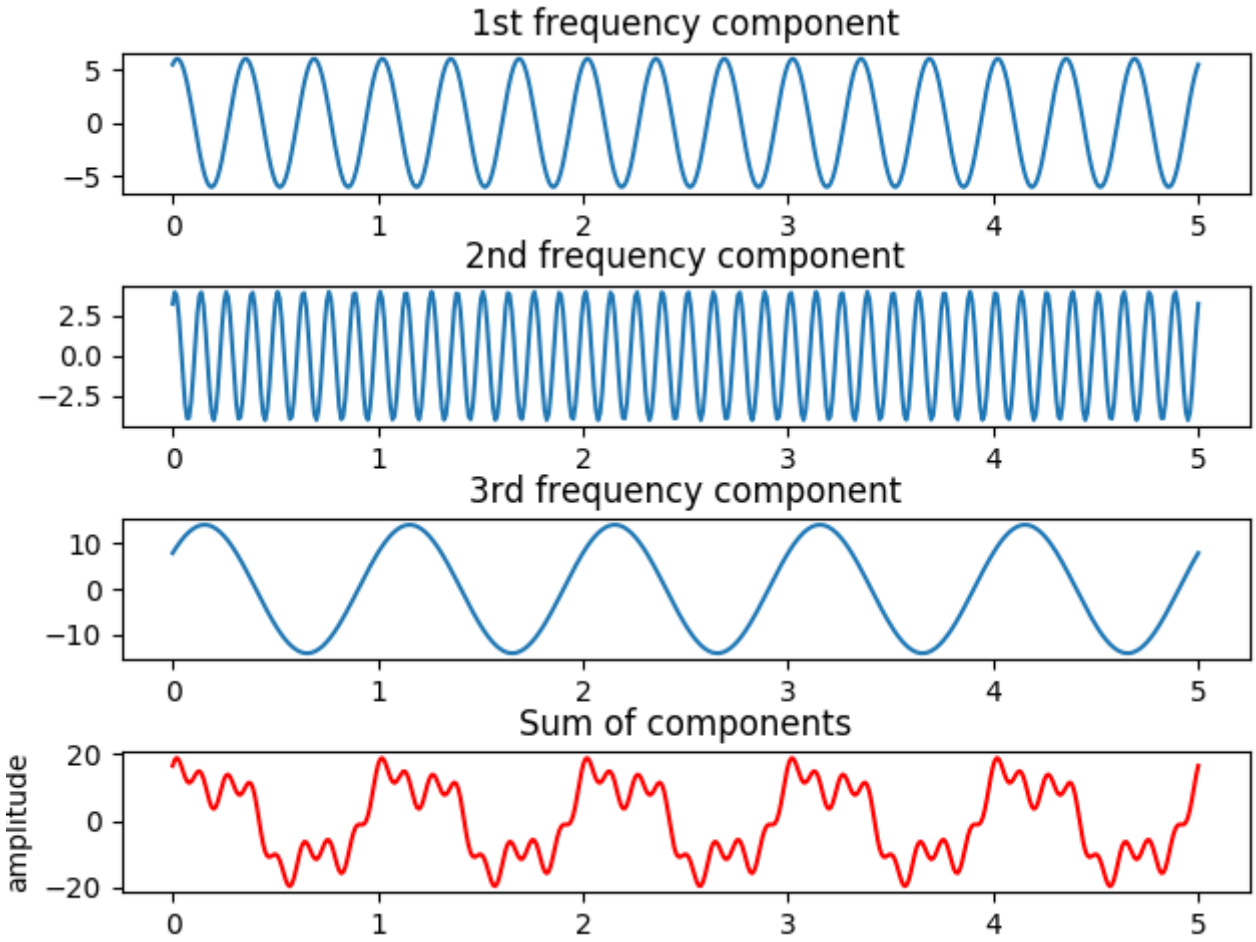
Fast Fourier Transforms

An FFT, or Fast Fourier Transform, is an efficient implementation of a Discrete Fourier Transform. The algorithm transforms a sum of sinusoidal signals into into its pure frequency components. In the following quiz, we will demonstrate adding sinusoidal waves together and then deconstructing them back into their component frequencies.



Instructions

`choose_frequencies`

In the first definition, `choose_frequencies`, choose three frequencies in a range from 1 to 50. These are the number of full cycles each sinusoidal wave will have in one "time unit". Running the definition will produce a visual of the three chosen frequencies as well as a new wave that is the sum of the three. This is similar to an acoustic signal, which is the sum of many sinusoidal waves. The waves may have different phases and amplitudes as well. Here's an example of a choice of (3, 8, 1) for the three frequencies.

`add_the_waves`

The second definition is where the waves are created. For this demonstration, we will only create three. This has been done for you with the `utils.make_waves` function - take a look at to understand how it works. You just need to add them together. This simulates an audio

signal, which is really just sinusoidal waves added together. In audio signals, the sinusoidal waves are created by sound vibrations and may be at varying amplitudes and phase as well as frequency. To simulate this variety, the `utils.make_waves` function provides random amplitudes and phase shifts.

`demo_fft`

An FFT can be created with a variety of library functions including scipy.fftpack.fft , which we'll use in this quiz. Read the linked reference to understand how to use it in code.

This FFT algorithm will create both positive and negative values, but we'll just display the positive ones. When you're done, you should see something like the following, showing peaks at the three frequency values originally provided!