phase

December 31, 2020

1 Hearing the phase of a sound

In this notebook we will investigate the effect of phase on the perceptual quality of a sound. It is often said that the human ear is lergely insensitive to phase and that's why most of the equalization in commercial-grade audio equipment takes place in the magnitude domain only.

But is it really so? Let's find out.

```
In [1]: %matplotlib inline
        import matplotlib
        import matplotlib.pyplot as plt
        import numpy as np
        import IPython
        from scipy.io import wavfile
```

```
In [2]: plt.rcParams["figure.figsize"] = (14,4)
```

We will be synthesizing audio clips so let's set the sampling rate for the rest of the not:

```
In [3]: Fs = 16000 # sampling freqency
        TWOPI = 2 * np.pi
```

We will be synthesizing and playing audio clips so let's define a convenience function to "beautify" the resulting sound: basically, we want a gentle fade-in and fade-out to avoid abrupt "clicks" when the waveform begins and ends.

Also, there is a "bug" in the current version of IPython whereby audio data is forcibly normalized prior to playing (see here for details; this may have been solved in the meantime). On the other hand, we want to avoid normalization in order to keep control over the volume of the sound. A way to do so is to make sure that all audio clips have at least one sample at a pre-defined maximum value, and this value is the same for all clips. To do so we add a slow "tail" to the data which will not result in an audible sound but will set a common maximum value in all clips.

```
In [4]: def prepare(x, max_value = 3):
            N = len(x)
            # fade-in and fade-out times max 0.2 seconds
            tf = min(int(0.2 * Fs), int(0.1 * N))
            for n in range(0, int(tf)):
                s = float(n) / float(tf)
                x[n] *= s
```