

MUSC 3264: Lab Assignment 4 (Due Tuesday, April 1, by 11:59 pm)

You are going to create two functions

makeWaveforms() – to generate a specific type of waveform

visualizeWaveforms() – to call makeWaveforms() to generate a waveform and then plot it in the time and frequency domains

1) In cell 1: import the necessary libraries

import numpy as np

import matplotlib.pyplot as plt

import librosa

import librosa.display

import IPython.display

2) In cells 2 copy plotAudio2()

3) In cell 3 copy plotAudioFreqDomain() that you created in Lab Assignment 3

4) In cell 4 copy additiveSynthesis() from additiveSynthesis.ipynb

5) In cell 5 create a function called makeWaveforms () that inputs

- frequency for the generated waveform (frequency)

- sampling rate for the generated waveform (samplingRate)

- number of harmonics in the generated waveform (numHarmonics)

- type of waveform to be generated (waveType)

- either 'sawtooth', 'triangle', 'square', or 'sine'

- (if nothing " is entered then a sine will be plotted)

The function will

- use an if/elif/else statement to set up the parameters for the different type of waveforms based on the code in additiveSynthesis.ipynb

- call additiveSynthesis() to generate a waveform

- return the generated waveform (signal)

6) In cell 6 create a function called visualizeWaveforms() that inputs

- frequency for the generated waveform (frequency)

- sampling rate for the generated waveform (samplingRate)

- number of harmonics in the generated waveform (numHarmonics)

- type of waveform to be generated (waveType)

- either 'sawtooth', 'triangle', 'square', or 'sine'

- (if nothing " is entered then a sine will be plotted)

- window size for the spectrogram (winSize)

- spectrogram type, 'linear' or 'log' (specType)

The function will

- call makeWaveform() –test whether makeWaveforms() runs without error before you try plotting anything

- call plotAudio2()

- call plotAudioFreqDomain()

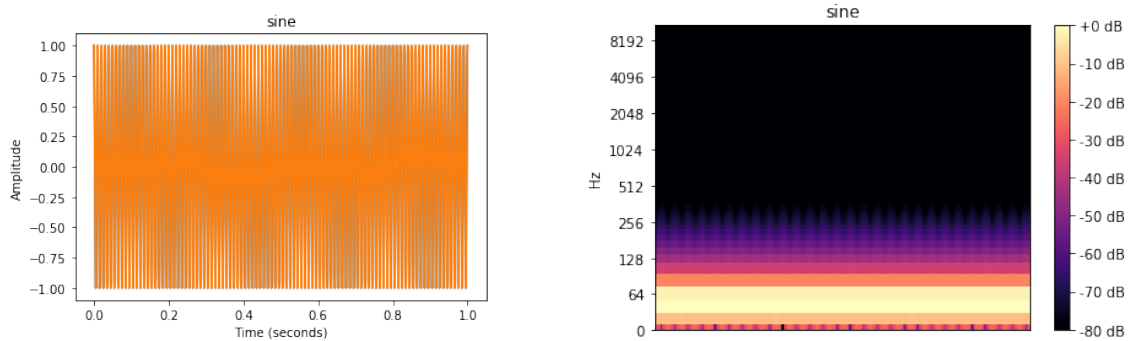
And it will return the output of makeWaveform()

7) In cell 7: call `visualizeWaveforms()` with the following arguments

```
frequency = 100  
samplingRate = 44100  
numHarmonics = 100  
waveType = 'sine'  
winSize = 1024  
specType = 'log'
```

and call `IPython.display.Audio()` with the signal returned from `visualizeWaveforms()`

This should generate the following plots (note that the plots will be above and below each other, not side by side)

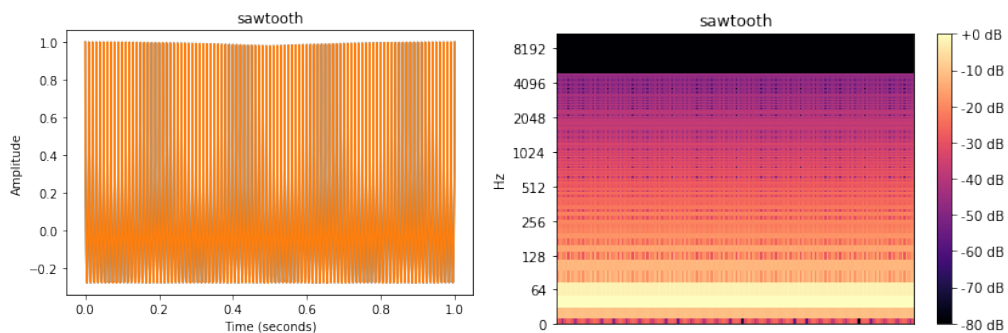


8) In cell 8: call `visualizeWaveforms()` with the following arguments

```
frequency = 100  
samplingRate = 44100  
numHarmonics = 100  
waveType = 'sawtooth'  
winSize = 1024  
specType = 'log'
```

and call `IPython.display.Audio()` with the signal returned from `visualizeWaveforms()`

This should generate the following plots (note that the plots will be above and below each other, not side by side)

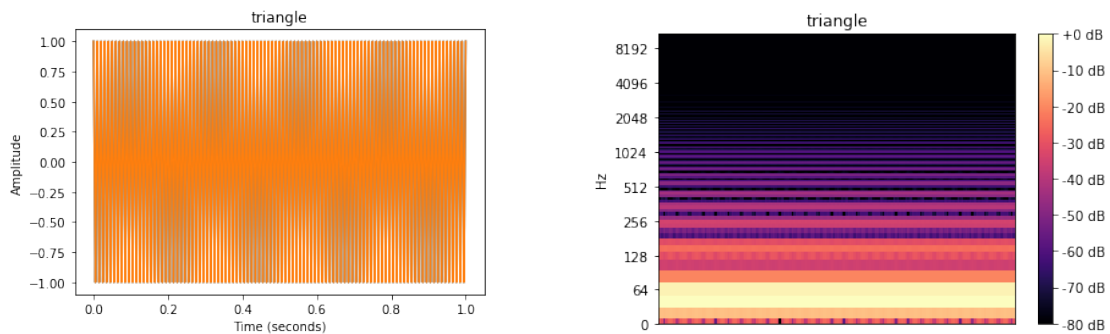


9) In cell 9: call visualizeWaveforms() with the following arguments

```
frequency = 100  
samplingRate = 44100  
numHarmonics = 100  
waveType = 'triangle'  
winSize = 1024  
specType = 'log'
```

and call IPython.display.Audio() with the signal returned from visualizeWaveforms()

This should generate the following plots (note that the plots will be above and below each other, not side by side)



10) In cell 10: call visualizeWaveforms() with the following arguments

```
frequency = 100  
samplingRate = 44100  
numHarmonics = 100  
waveType = 'square'  
winSize = 1024  
specType = 'log'
```

and call IPython.display.Audio() with the signal returned from visualizeWaveforms()

This should generate the following plots (note that the plots will be above and below each other, not side by side)

