Assignment 7

Sounds and Frequency

EC602 Fall 2016

1 Introduction

In this assignment, we explore audio signals in time and frequency.

1.1 Due Date

This assignment is due 2016-10-24 at midnight.

1.2 Submission Link

You can submit here: week 7 submit link

2 Background

2.1 Dual Tone Multi Frequency

Touch-tone phones, or as they are now called, phones, use tones to indicate dialed numbers.

The system started to replace *pulse* or *rotary* phones in North America in the 1960s.

The details are described here: about DTMF

2.2 Definition of Loudest band.

Given a continuous-time signal $s(t)$, the loudest band of bandwidth B is defined as the range $[L, H)$ such that $B = H - L$ and the energy of the signal in the frequency domain in that band is the largest possible value for this signal.

This means that

$$\int_L^H |S(f)|^2 df$$

is maximized over $L > 0$ and $B = H - L$

2.3 Filtering

A signal $x(t)$ can be filtered by a linear system represented by its impulse response $h(t)$ or the Fourier transform of $h(t)$ which is indicated by $H(f)$.

The filtered signal $y(t) = h(t) * x(t)$ which in the frequency domain is equivalent to

$$Y(f) = H(f)X(f)$$

Hence, by designing the shape of $H(f)$, the system can choose frequencies in the input signal $x(t)$ that will *survive* or pass through the system and remain in $y(t)$. Thus, signals can be filtered to extract features. For example, if $H(f) = 1$ whenever $L < |f| < H$, then this represents a bandpass filter that passed all frequencies between L and H and eliminates all other frequency content.

3 Background: sounds in python

The following file demonstrates how to read sounds and listen to them using python.

```
# Support for sound is provided
# in a number of modules. Some
# are part of the standard library,
# but others are not. Everything we
# use is part of the Anaconda distribution
# of python.

# WAV file support
import scipy.io.wavfile as wavfile

# sound playing
import PyQt4.QtGui as qt

# sleep while sound is playing
import time

# arrays
import numpy

# plotting facilities
import matplotlib.pyplot as pyplot

def read_wave(fname,debug=False):
    "return information about and time signal in the WAV file fname"
    frame_rate,music = wavfile.read(fname)
    if debug:
        print(frame_rate,type(music),music.shape,music.ndim)
    if music.ndim>1:
        nframes,nchannels = music.shape
    else:
        nchannels = 1
```

```
        nframes = music.shape[0]
    return music,frame_rate,nframes,nchannels

def wavplay(fname):
    "play a sound, and sleep until it is finished"
    qt.QSound.play(fname)
    music,frame_rate,nframes,nchannels = read_wave(fname)
    time.sleep(nframes/frame_rate)


fname = "bach10sec.wav"

# Plot the sound
music,frame_rate,nframes,nchannels = read_wave(fname,debug=True)
if nchannels > 1:
    music = music.sum(axis=1)

pyplot.plot(music)
pyplot.show()

# Listen to the sound
wavplay('bach10sec.wav')
```

The file is available here: sounds_example.py

3.1 Jupyter Notebook / IPython

Some of the facilities provided by jupyter notebook are explained in the following example notebook: audio_example.ipynb

3.2 Example Audio Files

For your convenience, here are some audio files to play with.

Some of them are Halloween themed.

- bach10sec.wav
- scary.wav
- bachish.wav

4 The assignment

4.1 Part A: telephone dialer.

Write a python function `dialer(file_name,frame_rate,phone,tone_time)` which creates a WAV file of the sound of a telephone number being dialed.

The parameters are

- `file_name`, a string representing the name of the WAV file to be created. Do not append ".wav" to this string.
- `frame_rate`, a number representing the number of samples per second to use in the sound
- `phone`, a string of digits representing a phone number to dial.

- `tone_time`, a number representing the time in seconds of each tone to generate.

The function `dialer` should be defined in an importable python file `w7_dialer.py`

4.2 Part B: loudest band.

Write a python function`loudest_band(music,frame_rate,bandwidth)` which returns a tuple `(low,high,loudest)` containing information about the loudest part of `music`.

The input parameters are:

- `music`: an ndarray of shape `(N,)` representing a continuous time signal which has been digitized. You may assume that `music` is real, and so the Fourier Transform will be conjugate symmetric.
- `frame_rate`: the sampling rate that was used to sample `music`
- `bandwidth`: the width of the frequency band to be selected (the loudest band)

The output parameters `(low,high,loudest)` are:

- `low`: the low end of the loudest frequency band in `music`
- `high`: the high end of the loudest frequency band in `music`
- `loudest`: a time signal extracted (filtered) from `music` which contains only the frequencies of `music` that are in the loudest band.

The function `loudest_band` should be defined in an importable python file `w7_loudest.py`.

4.3 Unittest / checker

The checker uses unittest to check your function. The test cases are available for you to read and attempt on your own computers here: w7_loudest_tester.py