

Python を使った WAVE ファイルの処理

桂田 祐史

2017年1月8日, 2018年1月28日

<http://nalab.mind.meiji.ac.jp/~mk/fourier/python-sound/>

1 Python で WAVE ファイルを扱うには

1.1 Wave モジュール

WAVE ファイルを扱うための wave モジュールが標準で用意されている。

testwavemodule.py

```
# testwavemodule.py
# encoding: utf-8
# これは Python 2.x 用のプログラム
# Python 3.x では print("チャンネル数 = ", numch) のようにする。

import wave

fname = 'guitar-5-3.wav'
wfile = wave.open(fname, 'r')

numch = wfile.getnchannels()
samplewidth = wfile.getsampwidth()
samplerate = wfile.getframerate()
numsamples = wfile.getnframes()

print u"チャンネル数 = ", numch
print u"サンプル幅 (バイト数) = ", samplewidth
print u"サンプリングレート (Hz) =", samplerate
print u"サンプル数 =", numsamples
print u"録音時間 =", numsamples / samplerate
```

実行結果

```
$ ls
guitar-5-3.wav testwavemodule2.py
testwavemodule.py testwavemodule3.py
$ python testwavemodule.py
チャンネル数 = 2
サンプル幅 (バイト数) = 2
サンプリングレート (Hz) = 44100
サンプル数 = 452025
録音時間 = 10
$
```

1.2 Numpy

Numpy は数値データ処理をするためのモジュールで、非常にポピュラーである。Numpy の ndarray (n-dimensional array) や、FFT をする関数 `numpy.fft.fft()` などを使うのがお勧め (Python 標準のリストを使うのは効率上の観点から勧められない)。

Numpy は、macOS に標準装備されている Python にも含まれている。それ以外の Python 処理系を使う場合、その処理系にあったやり方で Numpy をインストールする必要がある。

私は、MacPorts でインストールした Python 2.7, Python 3.6 を使っているが、それぞれ

```
sudo port install py27-numpy
sudo port install py36-numpy
```

とすれば Numpy がインストールされる。

2 プログラム例

2.1 Wave ファイルを読んで離散 Fourier 変換して周波数成分を調べる

授業で、Mathematica を用いて、WAVE ファイル “guitar-5-3.wav” を読み込み、適当な時点からの 1 秒分のデータ (項数 N をサンプリング周波数と同じにとった) を離散 Fourier 変換して、 $|C_0|, |C_1|, \dots, |C_{N-1}|$ をプロットする、という実験を行った ([1] の第 4 章)。

Python を使ってそれと同じことをしてみよう。

次のプログラムでは、グラフ描画のために matplotlib というモジュールを用いている¹。

<http://nalab.mind.meiji.ac.jp/~mk/fourier/testwavemodule3.py>

```
1 # testwavemodule3.py
2 # encoding: utf-8
3 # http://nalab.mind.meiji.ac.jp/~mk/lecture/fourier-2017/testwavemodule3.py
4 # これは Python 3.x 用のプログラム
5
6 import wave
7 import numpy as np
8 import matplotlib.pyplot as plt
9
10 fname = 'guitar-5-3.wav'
11 wfile = wave.open(fname, 'r')
12
13 numch = wfile.getnchannels()
14 samplewidth = wfile.getsampwidth()
15 samplerate = wfile.getframerate()
16 numsamples = wfile.getnframes()
17
18 print("チャンネル数 = ", numch)
19 print("サンプル幅 (バイト数) = ", samplewidth)
20 print("サンプリングレート (Hz) =", samplerate)
21 print("サンプル数 =", numsamples)
22 print("録音時間 =", numsamples / samplerate)
23
24 # すべてのフレームを読み込む (bytes オブジェクトになる)
25 buf = wfile.readframes(numsamples)
26 wfile.close()
27
28 # numpy の ndarray に変換する
29 if samplewidth == 2:
```

¹例えば MacPorts では、`sudo port install py36-matplotlib` のようにしてインストールできる。

```

30     data = np.frombuffer(buf, dtype='int16')
31     data = data / 32768.0
32 elif samplewidth == 4:
33     data = np.frombuffer(buf, dtype='int32')
34     data = data / 2147483648.0
35
36 # ステレオの場合は左チャンネルだけを取り出す
37 # (0 から最後まで 2つおきに、つまり 0,2,4,... 番目を取り出す)
38 if numch == 2:
39     #l_channel = data[0::2]
40     #r_channel = data[1::2]
41     data = data[0::2]
42
43 # 62000 番目から 1 秒分 (samplerate 個) 取り出し、離散フーリエ変換する
44 start = 62000
45 N = samplerate
46 c = np.fft.fft(data[start:start+N])
47 c = abs(c)
48
49 plt.subplot(2,1,1)
50 plt.title('data')
51 plt.plot(range(start, start+N), data[start:start+N])
52
53 plt.subplot(2,1,2)
54 plt.title('frequency spectrum')
55 # 対数目盛りにするには次の 3 行をアンコメントする (注釈を外す)
56 #ax = plt.gca()
57 #ax.set_yscale('log')
58 #ax.set_xscale('log')
59
60 plt.plot(c, linestyle='--')
61 #横軸を周波数 (Hz) には、次の 2 行をアンコメントする
62 #freqList = np.fft.fftfreq(N, d=1.0/samplerate)
63 #plt.plot(freqList, c, linestyle='--')
64
65 plt.tight_layout() # タイトルの被りを防ぐ
66 plt.show()

```

57 行目で単に

```
plt.plot(c, linestyle='--')
```

としているので、授業でやった Mathematica を用いた実験と同様に、 $|C_0|, |C_1|, \dots, |C_{N-1}|$ が順にプロットされるが、横軸の値の説明が少し面倒である (N をサンプリング周波数に等しく取っているため、1 番目は「直流成分」、 n 番目 ($2 \leq n \leq N/2$) は周波数 $n-1$ (Hz) の成分ということになる)。

Numpy で用意されている `numpy.fft.fftfreq()` を使って

```
freqList = np.fft.fftfreq(N, d=1.0/samplerate)
plt.plot(freqList, c, linestyle='--')
```

とすると、横軸の絶対値が周波数 (Hz) を表すようになる (詳しいことは省略するので、必要があれば Numpy のドキュメントを読むこと)。

同様に、1 つめのグラフ (PCM データのプロット) の横軸も、単位が時間になるように直すべきかもしれない (横軸、縦軸の説明は必須で、即答するためには、説明がしやすいものにしておくのが望ましい)。

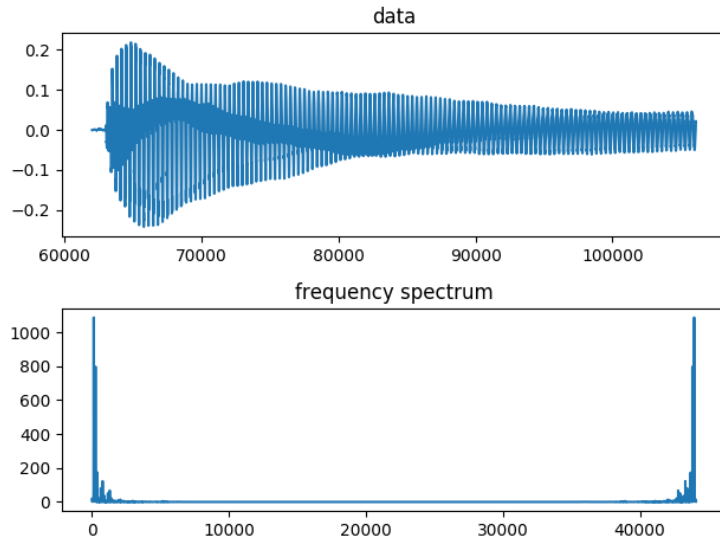


図 1: 下のグラフは $|C_0|, |C_1|, \dots, |C_{N-1}|$ をプロットしたもの

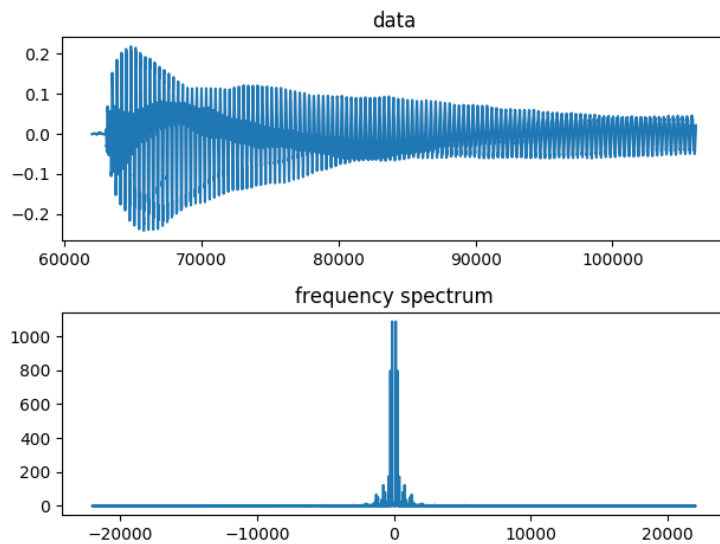


図 2: `numpy.fft.fftfreq()` を利用して横軸を周波数 (ただし符号付き) に直す

2.2 正弦波の WAVE ファイルを作る

A, f を正の定数とするとき、時刻 t (秒) での値が

$$u(t) = A \sin(2\pi ft)$$

である信号は、振幅 A , 周波数 f (Hz) の正弦波である。

次のプログラムを実行すると、 $f = 440$ Hz の信号を 10 秒間記録した WAVE ファイル (サンプリング周波数 44100 Hz モノラル、量子化ビット数 16 ビット、モノラル) “sinwave.wav” が出来る。

<http://nalab.mind.meiji.ac.jp/~mk/fourier/testwritewave.py>

```
1 # testwritewave.py
2 # encoding: utf-8
3 # http://nalab.mind.meiji.ac.jp/~mk/lecture/fourier-2017/testwavemodule3.py
4 # これは Python 3.x 用のプログラム
5
6 import numpy as np
7 import wave
8 import struct
9
10 fname = 'sinwave.wav'
11 wf = wave.open(fname, 'w')
12 ch = 1
13 width = 2
14 samplerate = 44100
15 wf.setnchannels(ch)
16 wf.setsampwidth(width)
17 wf.setframerate(samplerate)
18
19 time = 10
20 numsamples = time * samplerate
21
22 # python 2.x では ( ) を取る
23 print( u"チャンネル数 = ", ch)
24 print( u"サンプル幅 (バイト数) = ", width)
25 print( u"サンプリングレート (Hz) =", samplerate)
26 print( u"サンプル数 =", numsamples)
27 print( u"録音時間 =", time)
28
29 # 信号データを作る (numpy の ndarray で)
30 freq = 440 # 周波数 freq を 440 Hz にする
31 x=np.linspace(0, time, numsamples+1) # 0 ≤ t ≤ time を numsamples 等分
32 y=np.sin(2 * np.pi * freq * x) # 周波数 freq (Hz) の正弦波
33 y=np rint(32767*y/max(abs(y))) # [-32767,32767] の範囲に収める
34 y=y.astype(np.int16) # 16 ビット整数に型変換する
35 y=y[0:numsamples] # numsamples 個のデータに打ち切る
36
37 # ndarray から bytes オブジェクトに変換
38 data=struct.pack("h" * numsamples , *y)
39
40 # データを書き出す
41 wf.writeframes(data)
42 wf.close()
```

A PyAudio

(工事中 — そのうちプログラムを載せます)

A.1 簡単な紹介

PortAudio² と呼ばれる、マルチプラットフォーム (Windows, Mac, Unix) の、C, C++用のオーディオ I/O ライブラリがある。

PyAudio³ は、Python用の PortAudio インターフェイス・ライブラリである。

A.2 インストール

MacPorts でのインストール

```
sudo port install py27-pyaudio  
あるいは  
sudo port install py36-pyaudio
```

Homebrew でのインストール

```
brew install portaudio  
pip install pyaudio  
  
あるいは  
pip3 install pyaudio
```

A.3 ドキュメント

PyAudio Documentation⁴

参考文献

- [1] 桂田祐史：「画像処理とフーリエ変換」講義ノート, <http://nalab.mind.meiji.ac.jp/~mk/fourier/fourier-lecture-notes.pdf> (2014～).

²<http://www.portaudio.com/>

³<https://people.csail.mit.edu/hubert/pyaudio/>

⁴<https://people.csail.mit.edu/hubert/pyaudio/docs/>