2024年度　信号処理特論1

36414067 飯田 諒
メディア情報プログラム
徳田・南角・橋本研究室

2024年6月29日

- 使用した音：自分自身の声
- 使用したソフト：WaveSufer
- 使用した機材：自身の計算機
- 使用した OS：Windows
- プログラミング言語：Python

課題のソースコードは資料の最後尾にまとめて載せる。

1　課題 1

1.1　内容

ヘッダには、ファイルの形式、ファイルサイズ等の、ファイル自体の情報を示すものが書かれていた。具体的には先端から順に以下のとおりである。
52 49 46 46 : ”RIFF”
24 fa 00 00 : ファイルサイズ
57 41 56 45 : ”WAVE”
66 6d 74 20 : ”fmt”
10 00 00 00: fmt チャンクのサイズ
01 00: オーディオフォーマット
01 00: チャンネル数
80 3e 00 00: サンプリングレート
00 7d 00 00: バイトレート
02 00: ブロックアライン
10 00: ビット数
64 61 74 61: ”data”
00 fa 00 00 ... : データの中身

データの中身に置ける 10 番目の値は、00 00 であったため、0 である。

1.2　ソースコード

2　課題 2

図 1 に示すとおりである。

3　課題 3

3.1　方法

正規化は、音声の数値データの最大値と、-3dBFS の最大値である 23170.47 との比を取る。その後、音声データの数値すべてに対して算出した比を掛け算し、int 型にする。

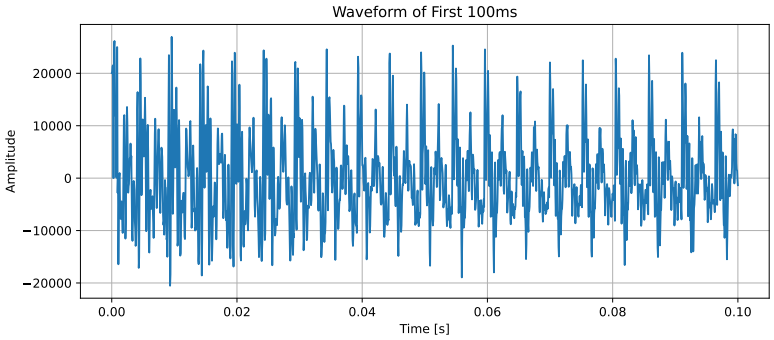

図 1: 先頭 100msec のプロット

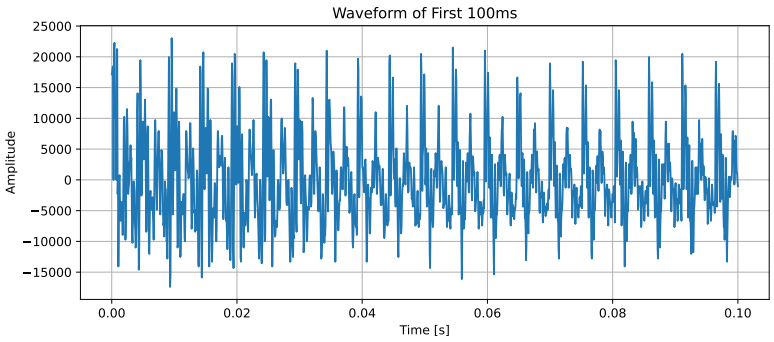

図 2: 量子ビット数を $\frac{1}{2}$ へ

量子ビット数の変更は、データに対してそれぞれ、$2^8, 2^{12}$ で割り、int 型にして、$2^8, 2^{12}$ を掛けた。

3.2　結果

図 2、図 3 に示すとおりである。

4　課題 4

量子ビット数を $\frac{1}{2}$ にしたものは、声の音質の劣化は感じ取ることが出来なかったが、声を出すたびに、「ザッ」というノイズが少し乗っているように聞こえた。
量子ビット数を $\frac{1}{4}$ にしたものは、明らかに音質の劣化が見られ、ノイズが組み合わさってできたような声、ブツブツとした声に感じた。

5　課題 5

図 4、図 5 に示すとおりである。 線形補間のようなことをした。例を挙げると、-8, 2, 2, 2, 2, 2, 8, ... という音声データが存在するとき、連続する値である 2 に注目して、最

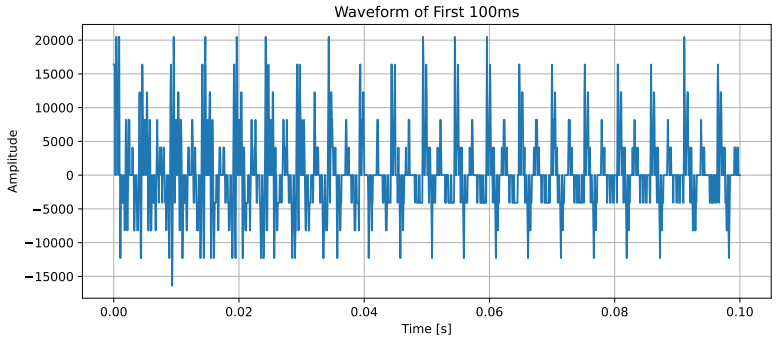

図 3: 量子ビット数を $\frac{1}{4}$ へ

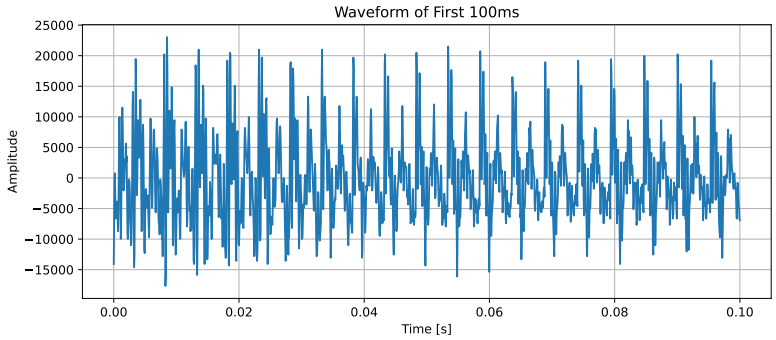

図 4: 量子ビット数を $\frac{1}{2}$ に変換後、線形補間

初に現れた 2 と、連続した 2 の次に現れる 8 を滑らかにつなぐように線形で補間をし、-8, 2, 3, 4, 5, 6, 7, 8, ... となるようにした。
量子ビット数を $\frac{1}{2}$ にしたものは、「ザッ」といったノイズの音自体が低くなり、目立たなくなったように感じた。
量子ビット数を $\frac{1}{4}$ にしたものは、音質の劣化は改善できなかったが、高いノイズの音が、低くなっているように感じた。
線形補間を行った考察として、急激な値の変化に対応する高周波成分を、線形補間を行うことによって緩やかな値の変化に変換したことによって低周波成分に変わったと考える。

6　ソースコードの主要部

6.1　課題 1

Listing 1: binary.py

```
filename= "arayurugennzituwo"
with open('{}.wav'.format(filename), 'rb') as f:
    data = f.read()

mystr = ""
i = 0
```

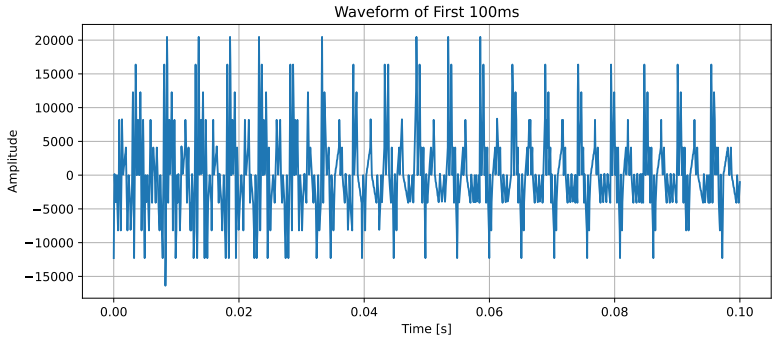

図 5: 量子ビット数を $\frac{1}{4}$ に変換後、線形補間

```
while i < len(data)-1 :
datai = int(data[i])
if i <= 40 :
    if (65 <= datai <= 90 or 97 <= datai <= 122) :
        mystr = mystr + "␣" + chr(datai)
    else :
        mystr = mystr + "␣" + str(datai)
    i += 1
else :
    datai *=256
    datai += int(data[i+1])
    if datai > 32767 :
        datai -= 65536
    mystr = mystr + "␣" + str(datai)
    i += 2

with open('{}.txt'.format(filename), 'w') as output_file:
output_file.write(mystr)
```

6.2　課題 2

Listing 2: dataplot.py

```
sampling_rate = 16000
max_time = 0.1
fleng = int(max_time*sampling_rate)
time = np.linspace(0,max_time,fleng)
fdata_short = fdata[:fleng].copy()
if __name__ == "__main__" :
    plt.figure(figsize=(10,4))
    plt.plot(time,fdata_short)
    plt.title("Waveform␣of␣First␣100ms")
    plt.xlabel("Time␣[s]")
    plt.ylabel("Amplitude")
    plt.grid()
```

```
13      plt.savefig("../figure/dataplot_{}.pdf".format(fname))
```

6.3 課題 3

Listing 3: bitdown.py

```
1  downbit = 12
2
3  def normalization(data) :
4      full_scale = 32767
5      minus_3dbfs = full_scale / np.sqrt(2)
6
7      normal_scale = minus_3dbfs/data.max()
8      norm_data = (data*normal_scale).astype(int)
9      return norm_data
10
11 def down(data, downbit) :
12     size = 1 << downbit
13     down_data = (data/size).astype(int)
14     down_data = down_data*size
15
16     return down_data
```

6.4 課題 4

Listing 4: reconst.py

```
1  import scipy.io.wavfile as wavfile
2
3  downbit = 8
4
5  if __name__ == "__main__" :
6      input_wav_file = fpath + ".wav"
7      output_wav_file = '{}bit_reconst_{}.wav'.format(16-downbit,fname)
8      sampling_rate, data = wavfile.read(input_wav_file)
9
10     norm_data = normalization(data)
11     down_data = down(norm_data,downbit).astype(np.int16)
12
13     wavfile.write(output_wav_file, sampling_rate, down_data)
```

6.5 課題 5

Listing 5: liear_interpolation.py

```
1  li = 0
```

```
2 for ri in range(len(down_data)) :
3     if down_data[li] != down_data[ri] :
4         l = down_data[li]
5         r = down_data[ri]
6         leng = ri - li
7         for j in range(leng) :
8             down_data[li+j] = int(l + (r - l) * j/leng)
9         li = ri
```