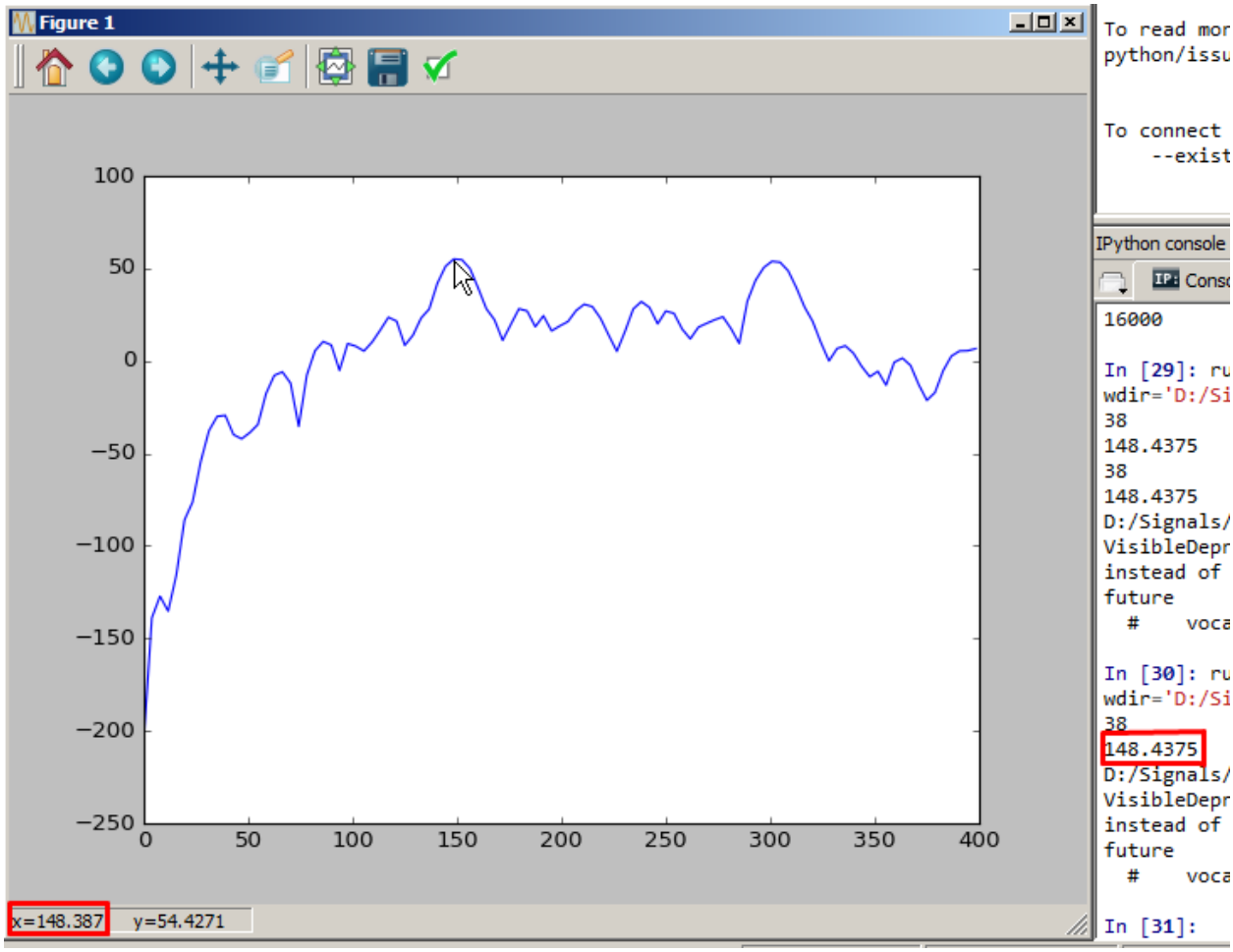
```
# -*- coding: utf-8 -*-
import matplotlib.pyplot as plt
import scipy.io.wavfile as wav
import scipy.io
import scipy.signal
from scipy.signal import lfilter
import numpy as np

def HPS(x, N, R):
    fs = 8000
    k = 2000 * N // fs
    sp = np.fft.fft(x * np.hamming(len(x)), N)
    f = np.arange(len(sp))*(fs/len(sp))
    sp = sp[:k]
    f  = f[:k]

    if R > 1:
        p_sp = []
        for i in range(2, R+1):
            p_sp.append(sp[0:-1:i])

        p_sp_dim = []
        for q in p_sp:
            p_sp_dim.append(q[:len(p_sp[-1])])

        p = np.ones(len(p_sp_dim[0]))
        for q in p_sp_dim:
            p = p * q
    #    print(p)
        f1 = f[0:len(p_sp[-1]):1]

        y = 20 * np.log10(np.abs(p))
        y1 = y[0:len(y)//2]
        m = y1.argmax()
    #
    #    print(m)
    #    print(f1[m])
        return [m, f1[m]]


def main():
# --.1
    data = scipy.io.loadmat('ma1_1', squeeze_me=True, struct_as_record=False)
#    sample_rate, data = wav.read('kdt_413.wav')
    fs = 8000
    t_frame = 0.02
    kol_sampes = fs * t_frame
    signal = data['ma1_1']
    vocal_frame = signal[4160:4160 + kol_sampes]
```

```
51. #    vocal_frame = data[8580:8580 + kol_sampes]
52.
53. #    plt.plot(data)
54. #    plt.plot(vocal_frame)
55.
56. # --.2
57.     sp = np.fft.fft(vocal_frame)
58.     f = np.arange(len(sp))*(fs/len(sp))
59. #    plt.plot(f, 20 * np.log10(np.abs(sp)))
60.
61. # --.3
62.     n = 2048
63.     sp = np.fft.fft(vocal_frame, n)
64.     f = np.arange(len(sp))*(fs/len(sp))
65. #    plt.plot(f, 20 * np.log10(np.abs(sp)))
66.
67. # --.4
68.     sp = np.fft.fft(vocal_frame*np.hamming(len(vocal_frame)), n)
69.     f = np.arange(len(sp))*(fs/len(sp))
70. #    plt.plot(f, 20 * np.log10(np.abs(sp)))
71.
72. # --.5
73.     f1 = f[0:512]
74. #    d = 20 * np.log10(np.abs(sp))
75. #    d = d[0:512]
76.     d = sp[0:512]
77. #    plt.plot(f1, d)
78.
79. # --.6
80.     sp2 = d[0:-1:2]
81. #    plt.plot(f1[0:len(sp2)], sp2)
82.
83. # --.7
84.     sp3 = d[0:-1:3]
85. #    plt.plot(f1[0:len(sp3)], sp3)
86.
87. # --.8
88.     sp4 = d[0:-1:4]
89. #    plt.plot(f1[0:len(sp4)], sp4)
90.
91. # --.9
92.     sp5 = d[0:-1:5]
93. #    plt.plot(f1[0:len(sp5)], sp5)
94.
95. # --.10
96.     p = sp[:len(sp5)] * sp2[:len(sp5)] * sp3[:len(sp5)] * sp4[:len(sp5)] * sp5
97. #    print(p)
98.     f1 = f[0:len(sp5):1]
99.     plt.plot(f1, 20 * np.log10(np.abs(p)))
100.
```

```
101.        # --.11
102.            y = 20 * np.log10(np.abs(p))
103.            y1 = y[0:len(y)//2]
104.            m = y1.argmax()
105.            print(m)
106.            print(f1[m])
107.
108.        # --.12
109.            hps = HPS(vocal_frame, 2048, 5)
110.            print(hps[0])
111.            print(hps[1])
112.
113.
114.        if __name__ == '__main__':
115.            main()
```



kdt_413.wav

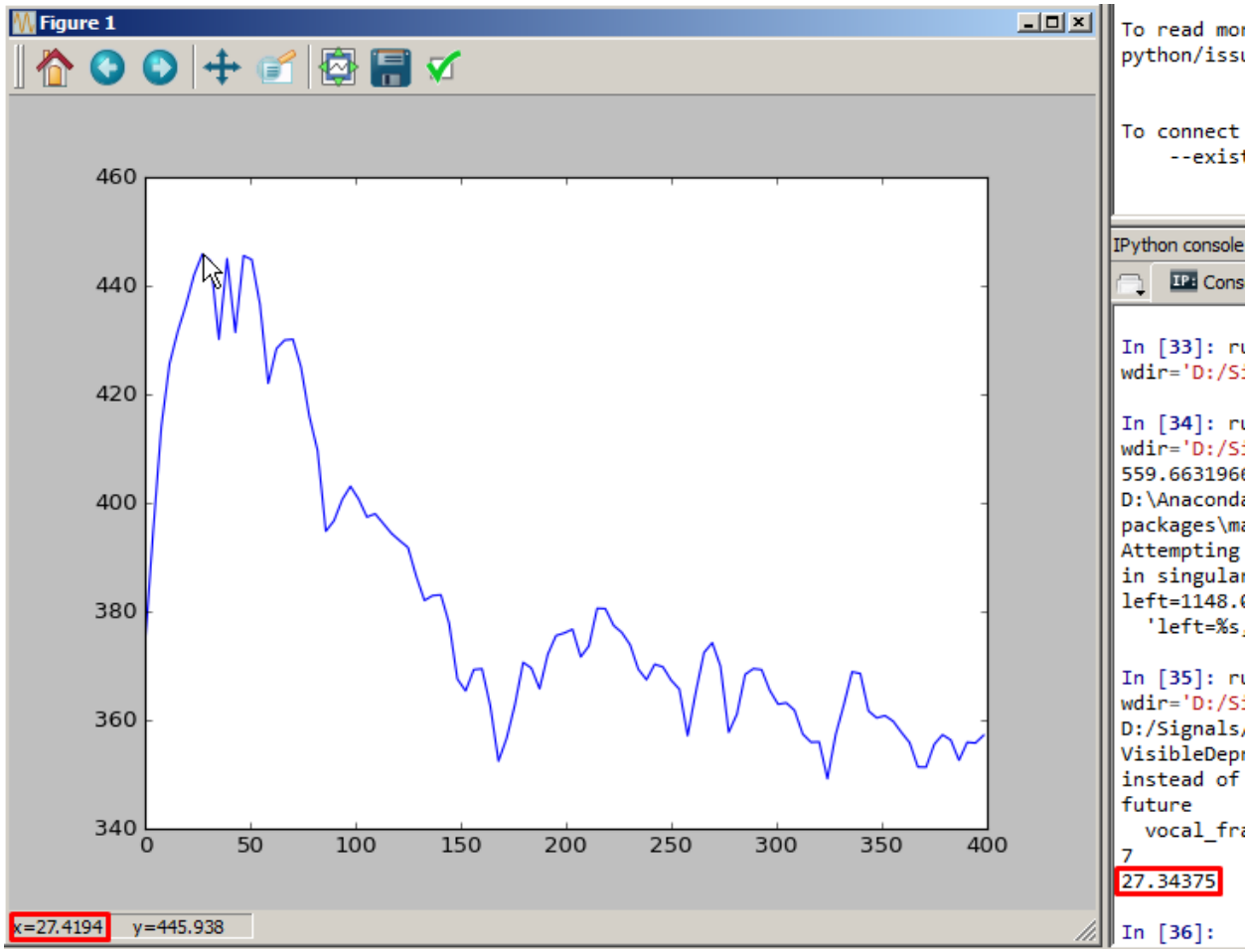


Figure 1
460
440
420
400
380
360
340
0
50
100
150
200
250
300
350
400
x=27.4194
y=445.938
IPython console
In [33]: ru
wdir='D:/Si
In [34]: ru
wdir='D:/Si
559.6631966
D:\Anaconda
packages\ma
Attempting
in singular
left=1148.6
'left=%s,
In [35]: ru
wdir='D:/Si
D:/Signals/
VisibleDepr
instead of
future
vocal_fra
7
27.34375
In [36]:

Для своего варианта (kdt_413.wav) выделить из сигнала вокализованный сегмент. Для этого сегмента найти период и частоту основного тона следующими способами:

1. Анализ во временной области
2. Анализ в частотной области
3. Частотная селекция
4. Корреляционные методы: функция autocorrelation
5. Корреляционные методы: функция pitch
6. Корреляционные методы: функция mdf
7. Кепстральный метод

Заполнить сводную таблицу: метод, частота основного тона.

kdt_413.wav

Метод	1	2	3	4	5	6	7
Частота	125	484	125	102	117	126	126.984126984

1.

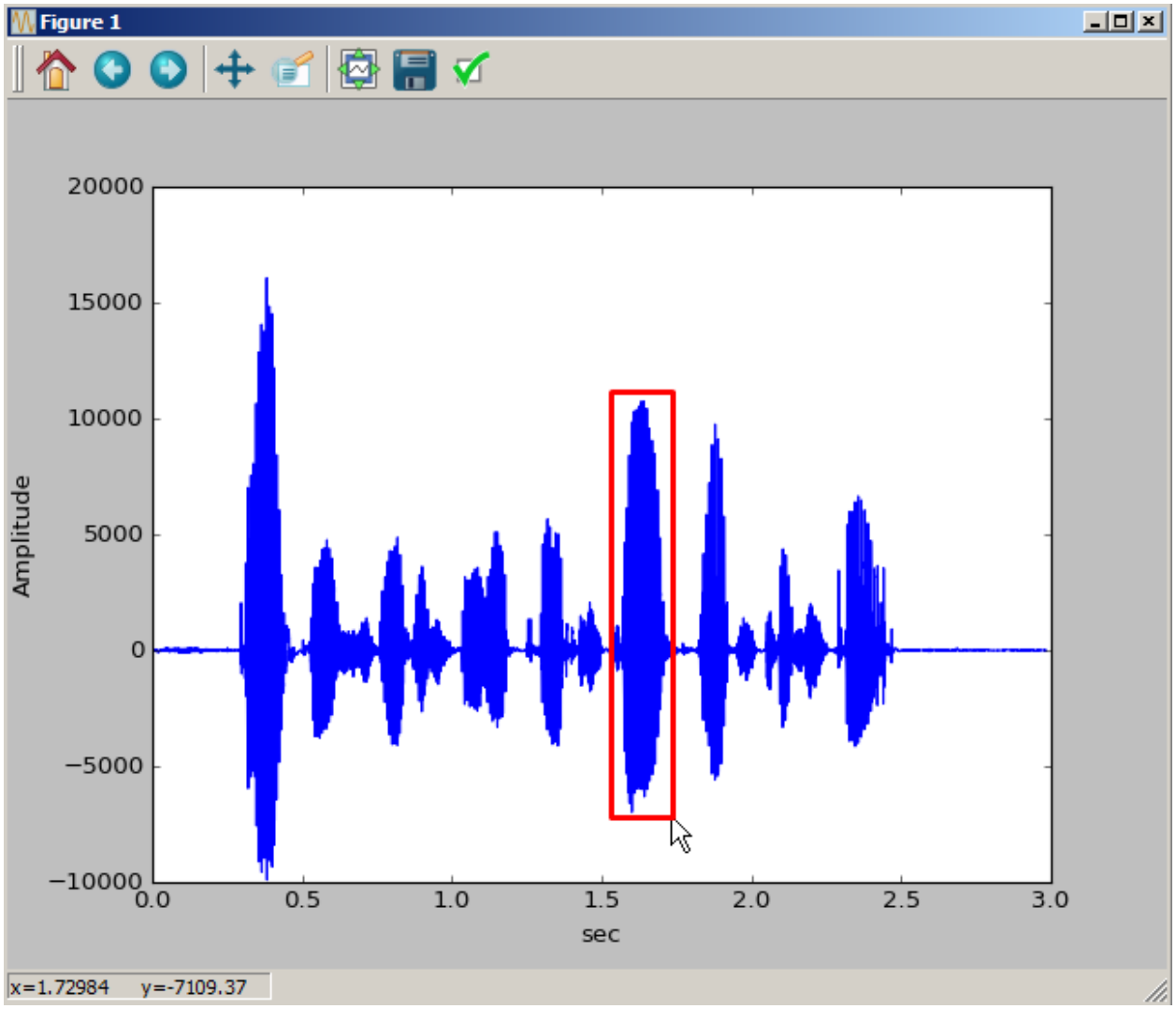
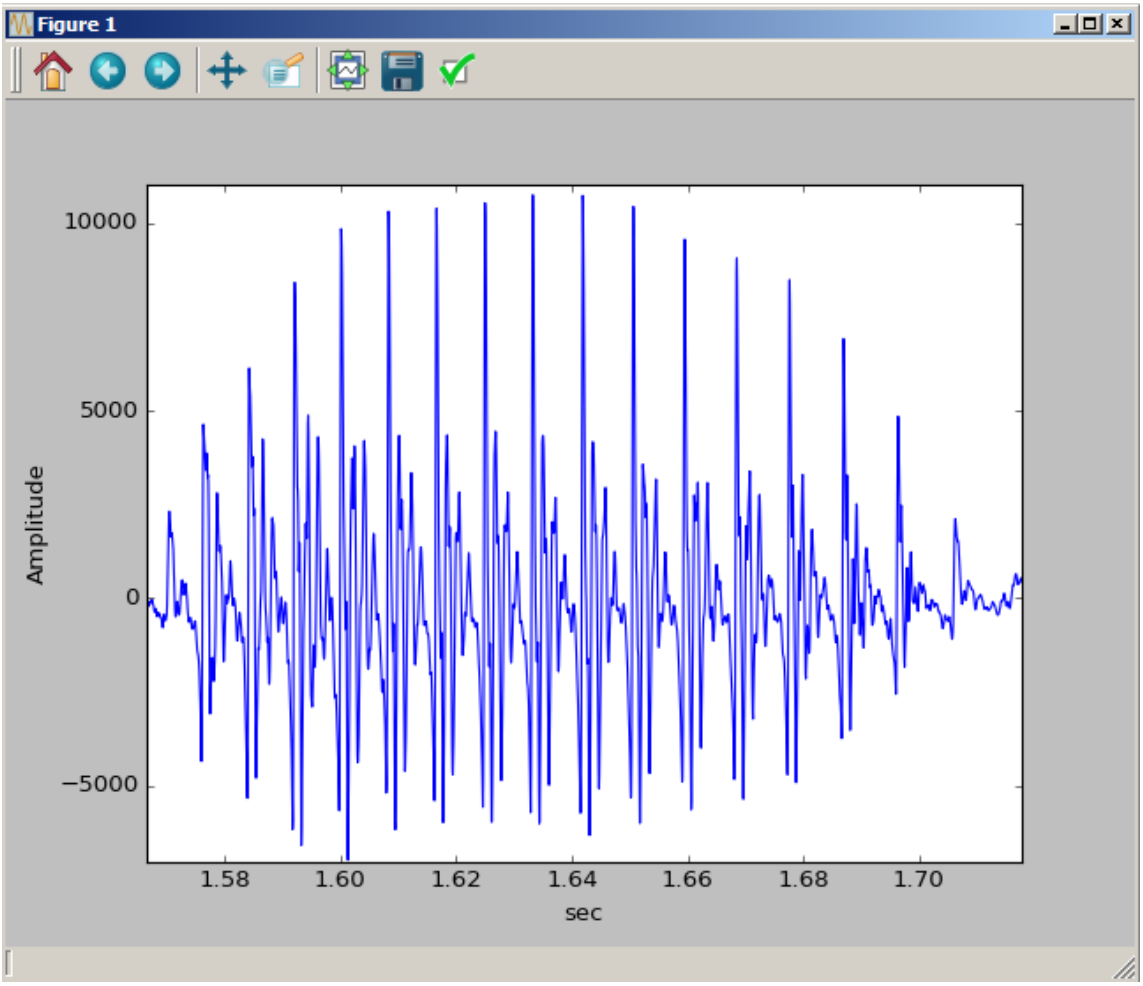
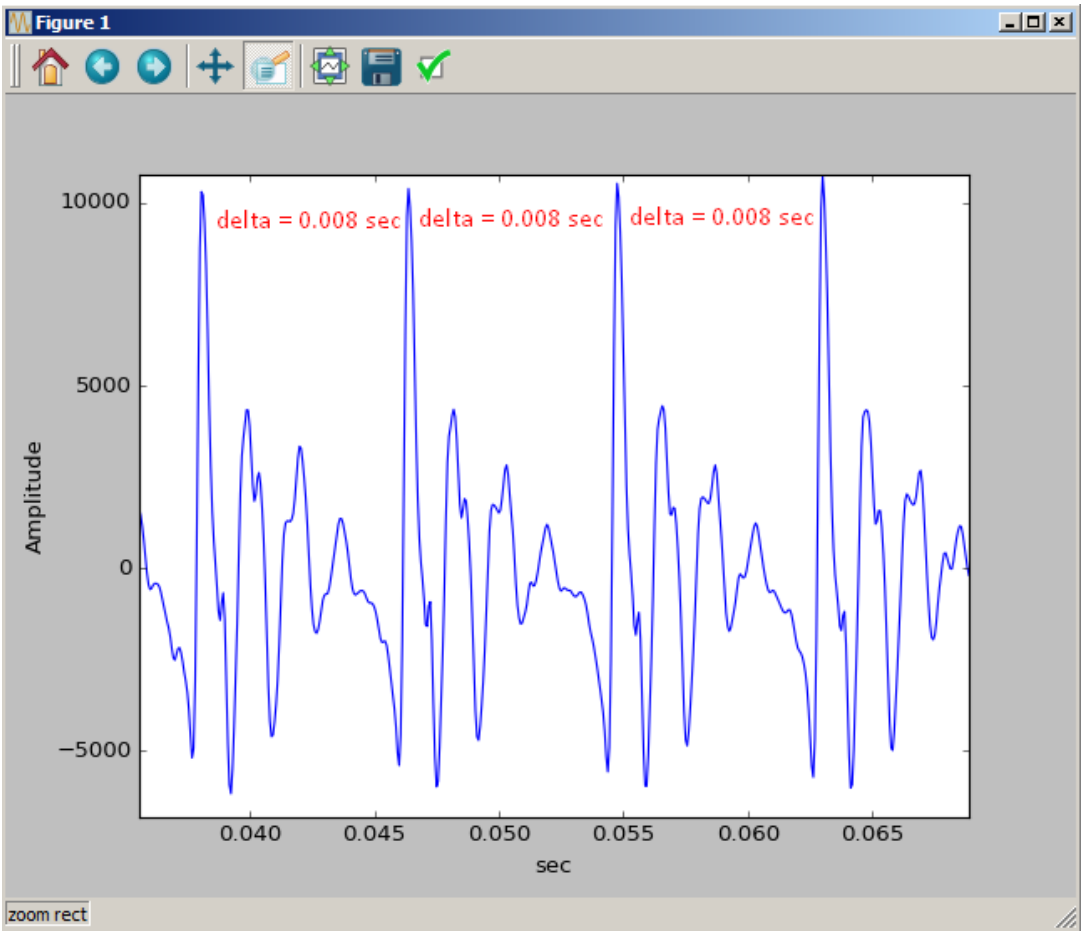


Figure 1
20000
15000
10000
5000
0
−5000
−10000
Amplitude
0.0
0.5
1.0
1.5
2.0
2.5
3.0
sec
x=1.72984 y=-7109.37



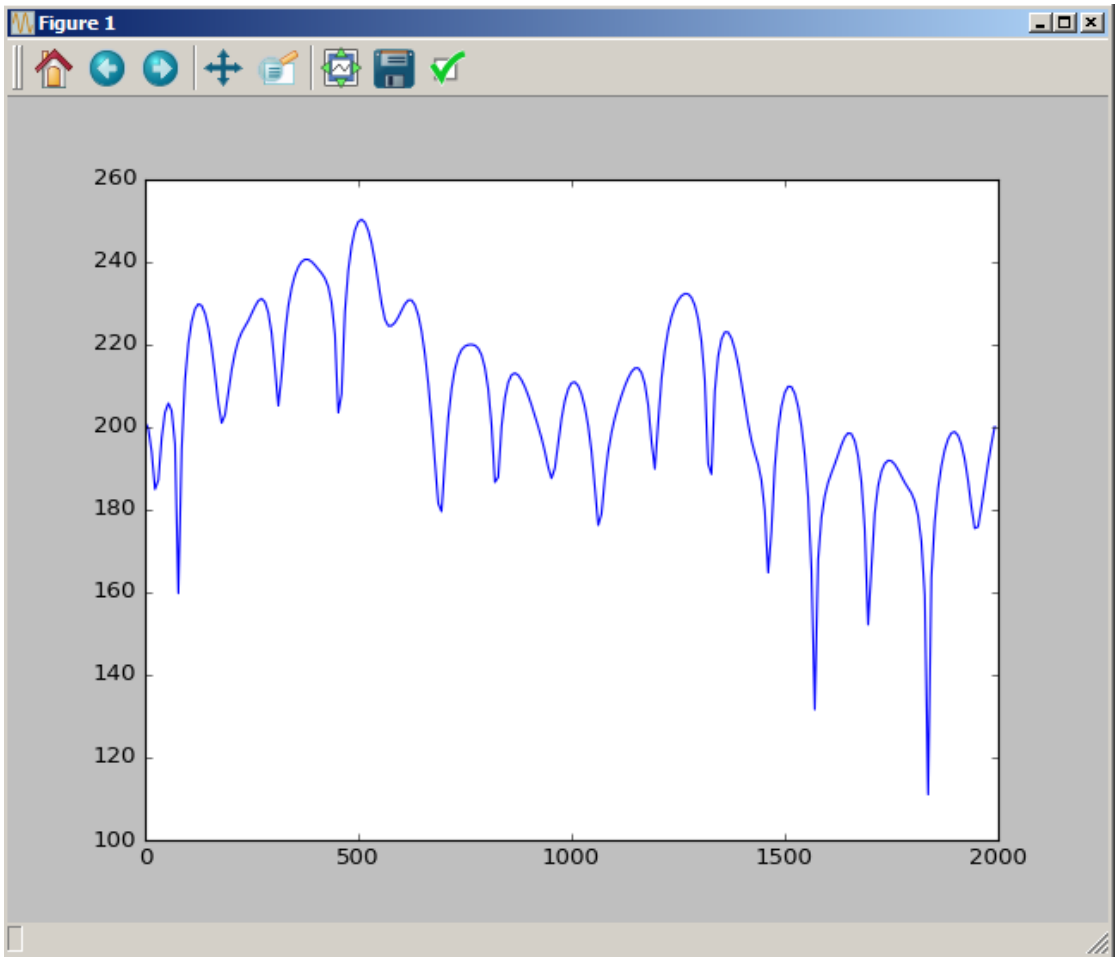
Участок с 1.57с – 1.70с



1 / 0.008 = 125

```
1. sample_rate, data = wav.read('kdt_413.wav')
2. fig, ax = plt.subplots(1, 1)
3. ax.set_xlabel("sec")
4. ax.set_ylabel("Amplitude")
5. ax.plot(np.arange(len(data)) / sample_rate, data)
6. plt.show()
```

2.



Max = 484

```
1. fs = 16000
2. t_frame = 0.02
3. kol_sampes = fs * t_frame
4. vocal_frame = data[25124:25124 + kol_sampes]
5. n = 2048
6. sp = np.fft.fft(vocal_frame, n)
7. f = np.arange(len(sp))*(fs/len(sp))
8. k = 2000 * n // fs
9. fig, ax = plt.subplots(1, 1)
10. ax.plot(f[:k], 20 * np.log(np.abs(sp[:k])))
11. print(f[sp[:k].argmax()])
12. plt.show()
```

3.

```
def HPS(x, N, R):
    fs = 16000
    k = 2000 * N // fs
    sp = np.fft.fft(x * np.hamming(len(x)), N)
    f = np.arange(len(sp))*(fs/len(sp))
    sp = sp[:k]
    f  = f[:k]

    if R > 1:
        p_sp = []
        for i in range(2, R+1):
            p_sp.append(sp[0:-1:i])

        p_sp_dim = []
        for q in p_sp:
            p_sp_dim.append(q[:len(p_sp[-1])])

        p = np.ones(len(p_sp_dim[0]))
        for q in p_sp_dim:
            p = p * q
    #    print(p)
        f1 = f[0:len(p_sp[-1]):1]

        y = 20 * np.log10(np.abs(p))
        y1 = y[0:len(y)//2]
        m = y1.argmax()
    #
    #    print(m)
    #    print(f1[m])
        return [m, f1[m]]
```

4.

```
def autocorrelation(x, fs):
    r = np.correlate(x, x, mode='full')
    r = r[int(r.size/2):]
    ms20 = int(20/1000 * fs)
    ms2 = int(2/1000 * fs)
    i = r[ms2:ms20].argmax()
    print('i={} ms2={} ms20={}'.format(i, ms2, ms20))
    return r, fs / (i + ms2)
```

i = 102

5.

```
def pitch(x, m, N):
    peak = 0
    for l in np.arange(20, 150):
        autoc = 0
        for n in np.arange(m - N + 1, m):
            autoc = autoc + x[n] * x[n-l]
        if autoc > peak :
            peak = autoc
            lag = l
    return lag
```

lag = 117

6.

```
def pitch_md(x, m, N):
    min_ = np.inf
    for l in np.arange(20, 150):
        mdf = 0
        for n in np.arange(m - N + 1, m):
            mdf = mdf + np.abs(x[n] - x[n-l])
        if mdf < min_:
            min_ = mdf
            lag = l
    return lag
```

lag = 126

7.

```
def cepstrum(data):
    fs = 16000
    t_frame = 0.02
    kol_sampes = fs * t_frame
    x = data[25124:25124 + kol_sampes]
    t = np.arange (0, len(x))/fs
    fig, ax = plt.subplots(3, 1)
    ax[0].plot(t, x)

    y = np.fft.fft(x*np.hamming(len(x)),2048)
    fs_05 = fs / 2
    hz5000 = fs_05 * len(y) / fs
    f = np.arange(0, hz5000) * fs / len(y)
```

```
    ax[1].plot(f, 20*np.log10(np.abs(y[0 : len(f)])))
    ax[1].grid()

    C = np.fft.fft(np.log(abs(y)))
    ms2 = int(2/1000*fs)
    ms20 = int(20/1000*fs)
    q = np.arange(ms2, ms20)/ fs
    ax[2].plot(q, np.abs(C[ms2:ms20]))
    ax[2].grid()

    fx = np.abs(C[ms2:ms20]).argmax()
    print("fx=", fs/(ms2+fx-1))
    plt.show()
```

fx = 126.984126984