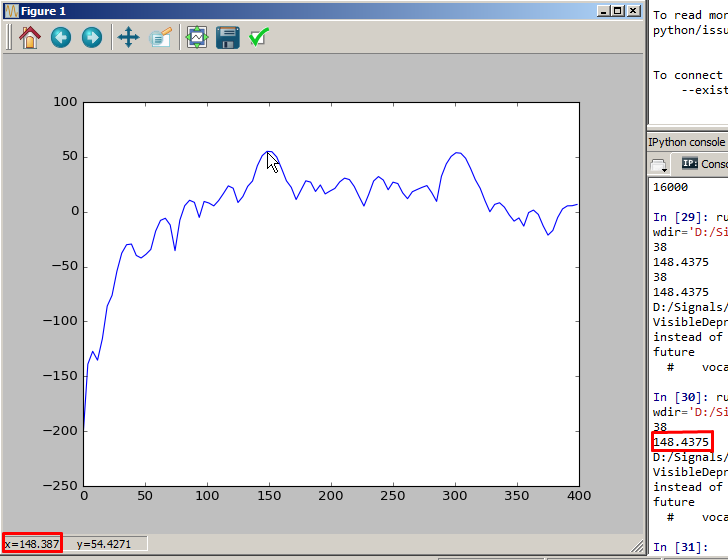
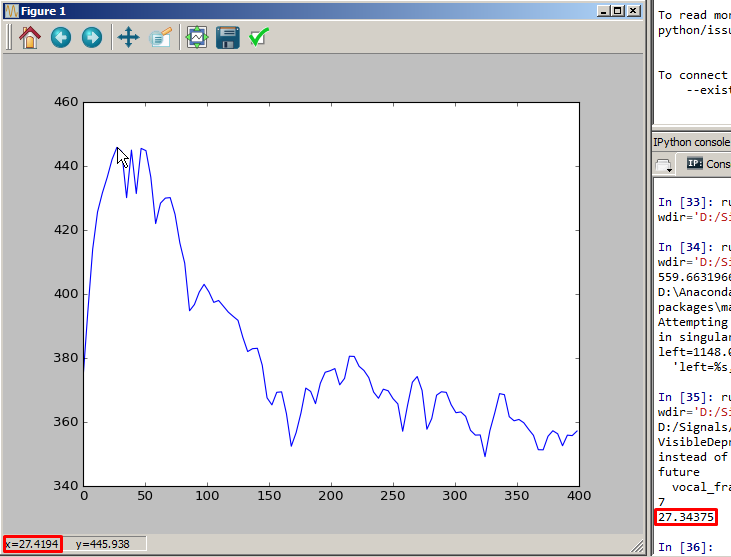
1. *# -\*- coding: utf-8 -\*-*
2. **import** matplotlib.pyplot **as** plt
3. **import** scipy.io.wavfile **as** wav
4. **import** scipy.io
5. **import** scipy.signal
6. **from** scipy.signal **import** lfilter
7. **import** numpy **as** np
9. **def** HPS(x, N, R):
10. fs = 8000
11. k = 2000 \* N // fs
12. sp = np.fft.fft(x \* np.hamming(len(x)), N)
13. f = np.arange(len(sp))\*(fs/len(sp))
14. sp = sp[:k]
15. f  = f[:k]
17. **if** R > 1:
18. p\_sp = []
19. **for** i **in** range(2, R+1):
20. p\_sp.append(sp[0:-1:i])
22. p\_sp\_dim = []
23. **for** q **in** p\_sp:
24. p\_sp\_dim.append(q[:len(p\_sp[-1])])
26. p = np.ones(len(p\_sp\_dim[0]))
27. **for** q **in** p\_sp\_dim:
28. p = p \* q
29. *#    print(p)*
30. f1 = f[0:len(p\_sp[-1]):1]
32. y = 20 \* np.log10(np.abs(p))
33. y1 = y[0:len(y)//2]
34. m = y1.argmax()
35. *#*
36. *#    print(m)*
37. *#    print(f1[m])*
38. **return** [m, f1[m]]


42. **def** main():
43. *# --.1*
44. data = scipy.io.loadmat('ma1\_1', squeeze\_me=True, struct\_as\_record=False)
45. *#    sample\_rate, data = wav.read('kdt\_413.wav')*
46. fs = 8000
47. t\_frame = 0.02
48. kol\_sampes = fs \* t\_frame
49. signal = data['ma1\_1']
50. vocal\_frame = signal[4160:4160 + kol\_sampes]
51. *#    vocal\_frame = data[8580:8580 + kol\_sampes]*
53. *#    plt.plot(data)*
54. *#    plt.plot(vocal\_frame)*
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)))*
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)))*
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)))*
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)*
79. *# --.6*
80. sp2 = d[0:-1:2]
81. *#    plt.plot(f1[0:len(sp2)], sp2)*
83. *# --.7*
84. sp3 = d[0:-1:3]
85. *#    plt.plot(f1[0:len(sp3)], sp3)*
87. *# --.8*
88. sp4 = d[0:-1:4]
89. *#    plt.plot(f1[0:len(sp4)], sp4)*
91. *# --.9*
92. sp5 = d[0:-1:5]
93. *#    plt.plot(f1[0:len(sp5)], sp5)*
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)))
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])
108. *# --.12*
109. hps = HPS(vocal\_frame, 2048, 5)
110. **print**(hps[0])
111. **print**(hps[1])

114. **if** \_\_name\_\_ == '\_\_main\_\_':
115. main()



kdt\_413.wav



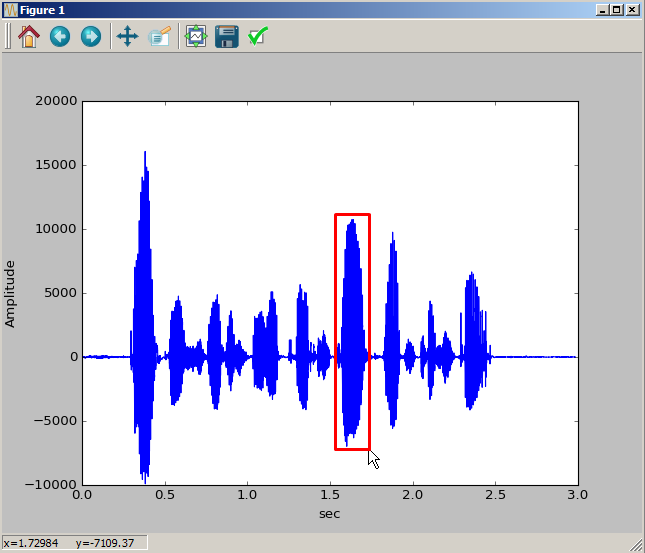
Для своего варианта (kdt\_413.wav) выделить из сигнала вокализованный сегмент. Для этого сегмента найти период и частоту основного тона следующими способами:

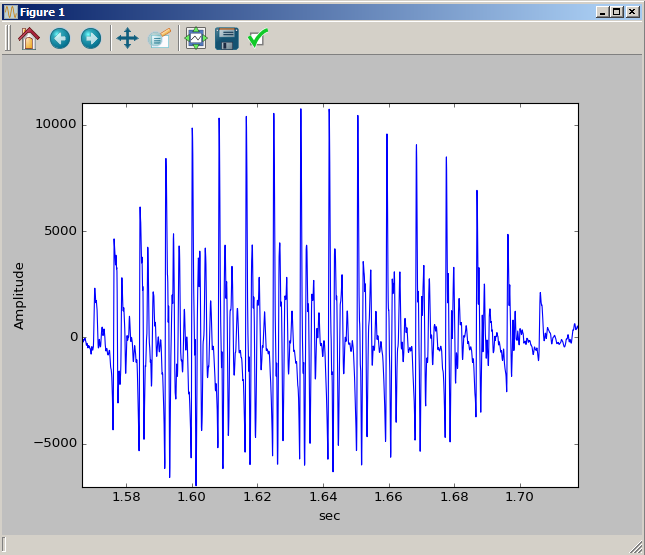
1. Анализ во временной области
2. Анализ в частотной области
3. Частотная селекция
4. Корреляционные методы: функция autocorrelation
5. Корреляционные методы: функция pitch
6. Корреляционные методы: функция mdf
7. Кепстральный метод

Заполнить сводную таблицу: метод, частота основного тона.

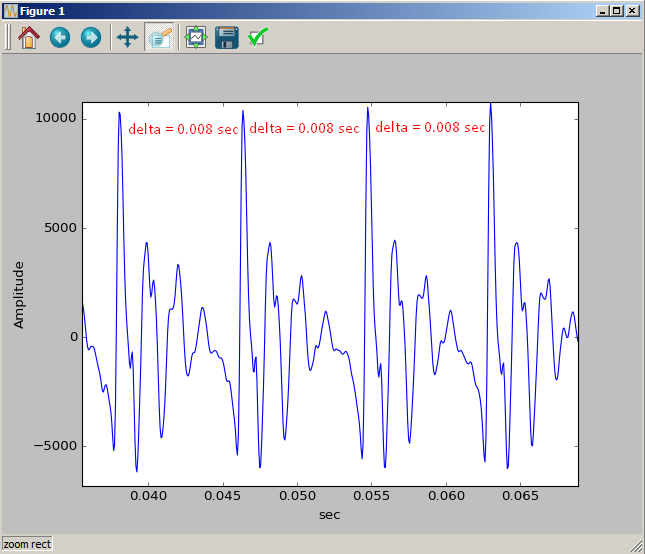
kdt\_413.wav

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

1. 

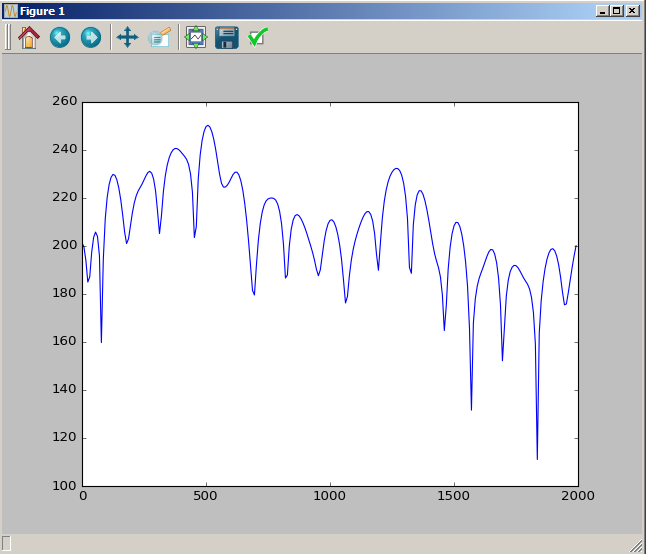


Участок с 1.57с – 1.70с



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()



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()
13. **def** HPS(x, N, R):
14. fs = 16000
15. k = 2000 \* N // fs
16. sp = np.fft.fft(x \* np.hamming(len(x)), N)
17. f = np.arange(len(sp))\*(fs/len(sp))
18. sp = sp[:k]
19. f  = f[:k]
21. **if** R > 1:
22. p\_sp = []
23. **for** i **in** range(2, R+1):
24. p\_sp.append(sp[0:-1:i])
26. p\_sp\_dim = []
27. **for** q **in** p\_sp:
28. p\_sp\_dim.append(q[:len(p\_sp[-1])])
30. p = np.ones(len(p\_sp\_dim[0]))
31. **for** q **in** p\_sp\_dim:
32. p = p \* q
33. *#    print(p)*
34. f1 = f[0:len(p\_sp[-1]):1]
36. y = 20 \* np.log10(np.abs(p))
37. y1 = y[0:len(y)//2]
38. m = y1.argmax()
39. *#*
40. *#    print(m)*
41. *#    print(f1[m])*
42. **return** [m, f1[m]]
43. **def** autocorrelation(x, fs):
44. r = np.correlate(x, x, mode='full')
45. r = r[int(r.size/2):]
46. ms20 = int(20/1000 \* fs)
47. ms2 = int(2/1000 \* fs)
48. i = r[ms2:ms20].argmax()
49. **print**('i={} ms2={} ms20={}'.format(i, ms2, ms20))
50. **return** r, fs / (i + ms2)

i = 102

1. **def** pitch(x, m, N):
2. peak = 0
3. **for** l **in** np.arange(20, 150):
4. autoc = 0
5. **for** n **in** np.arange(m - N + 1, m):
6. autoc = autoc + x[n] \* x[n-l]
7. **if** autoc > peak :
8. peak = autoc
9. lag = l
10. **return** lag

lag = 117

1. **def** pitch\_md(x, m, N):
2. min\_ = np.inf
3. **for** l **in** np.arange(20, 150):
4. mdf = 0
5. **for** n **in** np.arange(m - N + 1, m):
6. mdf = mdf + np.abs(x[n] - x[n-l])
7. **if** mdf < min\_:
8. min\_ = mdf
9. lag = l
10. **return** lag

lag = 126

1. **def** cepstrum(data):
2. fs = 16000
3. t\_frame = 0.02
4. kol\_sampes = fs \* t\_frame
5. x = data[25124:25124 + kol\_sampes]
6. t = np.arange (0, len(x))/fs
7. fig, ax = plt.subplots(3, 1)
8. ax[0].plot(t, x)
10. y = np.fft.fft(x\*np.hamming(len(x)),2048)
11. fs\_05 = fs / 2
12. hz5000 = fs\_05 \* len(y) / fs
13. f = np.arange(0, hz5000) \* fs / len(y)
14. ax[1].plot(f, 20\*np.log10(np.abs(y[0 : len(f)])))
15. ax[1].grid()
17. C = np.fft.fft(np.log(abs(y)))
18. ms2 = int(2/1000\*fs)
19. ms20 = int(20/1000\*fs)
20. q = np.arange(ms2, ms20)/ fs
21. ax[2].plot(q, np.abs(C[ms2:ms20]))
22. ax[2].grid()
24. fx = np.abs(C[ms2:ms20]).argmax()
25. **print**("fx=", fs/(ms2+fx-1))
26. plt.show()

fx = 126.984126984

