In [290]:

```
import plotly.plotly as py
import plotly.graph_objs as go
import plotly.figure_factory as FF
import sys
import math
import struct
import wave
import random
import argparse
import numpy as np
import IPython.display as ipd
import matplotlib.pyplot as plt

import pylab as pl
from itertools import *
from struct import pack
import numpy as np
import pandas as pd
import csv
```

In [ ]:

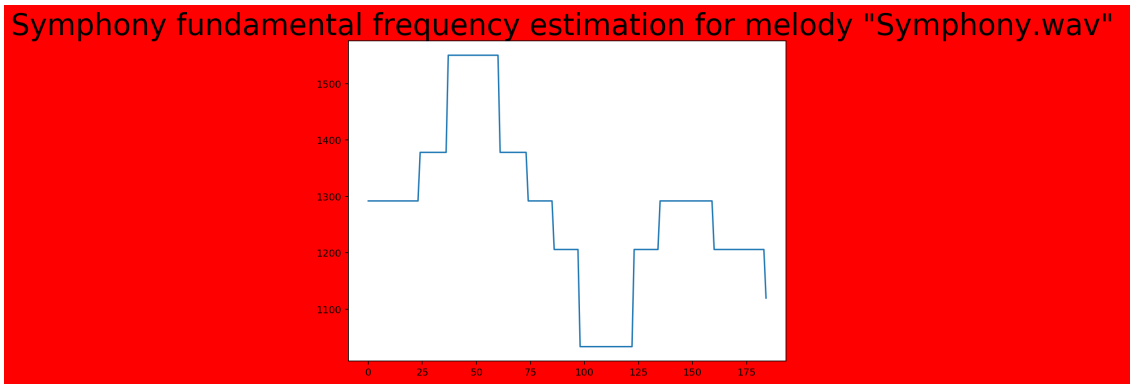
```
Question 1.1
=============================================================
========================
// 1. Take an FFT of the middle third of a recorded
plucked string tone.
// 2. Find the frequencies and amplitudes of the
largest K peaks
// 3. Form a histogram of peak spacing delta fi
// 4. The pitch estimate f0 is defined as the most
common spacing delta fi in the histogram.
=============================================================
========================================
Series {
    // set hop size (size in samples) to 2048
    inSamples = 2048

    // read source file
```

```
    + input_file  = "Symphony.wav"

    // refer to the root of file use /<control
name>, input_file
    -> input: SoundFileSource { filename = /
input_file }

    // apply a windowing function to input signal
    -> Windowing {size = 2048}

    // computing spectrum
    -> Spectrum

    // computing powerSpectrum
    // Controls: "power","magnitude", "decibels",
"logmagnitude", etc..
    -> PowerSpectrum { spectrumType =
"logmagnitude" }

    -> Transposer
    // calculate maximums and return position
    -> maximum : MaxArgMax
    -> Transposer

    // select different observations from input
data
    // Controls: "disable", "enables"
    -> selection: Selector {disable = 0 }

    // audio sink to CSV file named f1.csv
    -> sink : CsvSink { filename = "f1.csv" }

    // end of file is reached, control becomes
false
    + done = ( input/hasData == false )
}
```

In [291]:

```
list1 = []
```

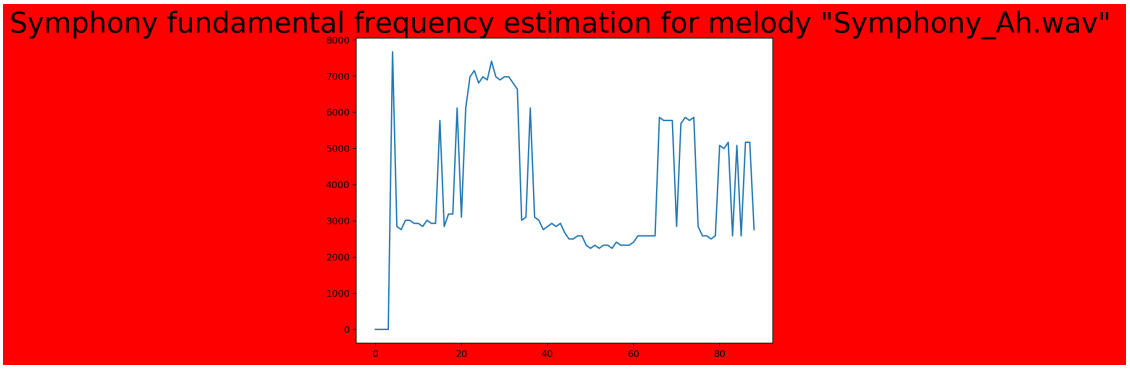
```
with open('f1.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list1.append(int(', '.join(row)))

w = [44100/512*x for x in list1]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
plt.title('Symphony fundamental frequency
estimation for melody "Symphony.wav" ', fontsize =
30)
plt.plot(w)
plt.show()
```



In [292]:

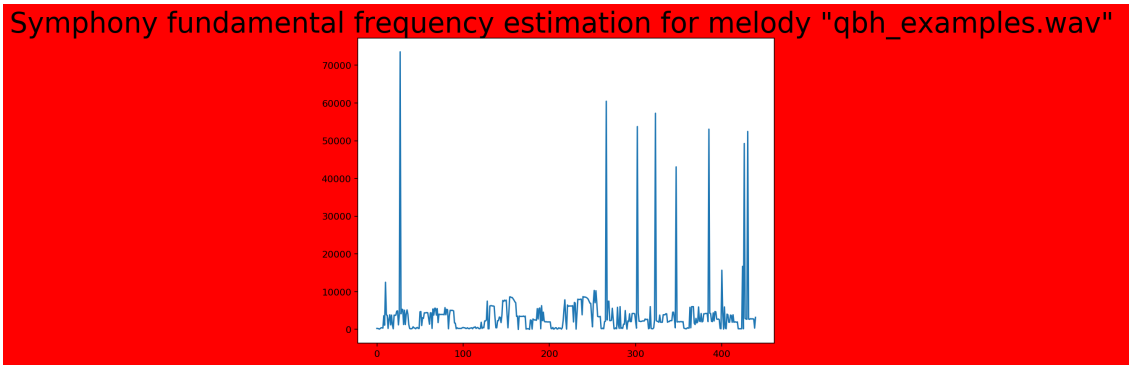
```
list1 = []
with open('f2.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list1.append(int(', '.join(row)))

w = [44100/512*x for x in list1]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
plt.title('Symphony fundamental frequency
estimation for melody "Symphony_Ah.wav" ', fontsize
= 30)
plt.plot(w)
```

```
plt.show()
```



In [293]:

```
list1 = []
with open('f3.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list1.append(int(', '.join(row)))

w = [44100/512*x for x in list1]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
plt.title('Symphony fundamental frequency
estimation for melody "qbh_examples.wav" ',
fontsize = 30)
plt.plot(w)
plt.show()
```



In [ ]:

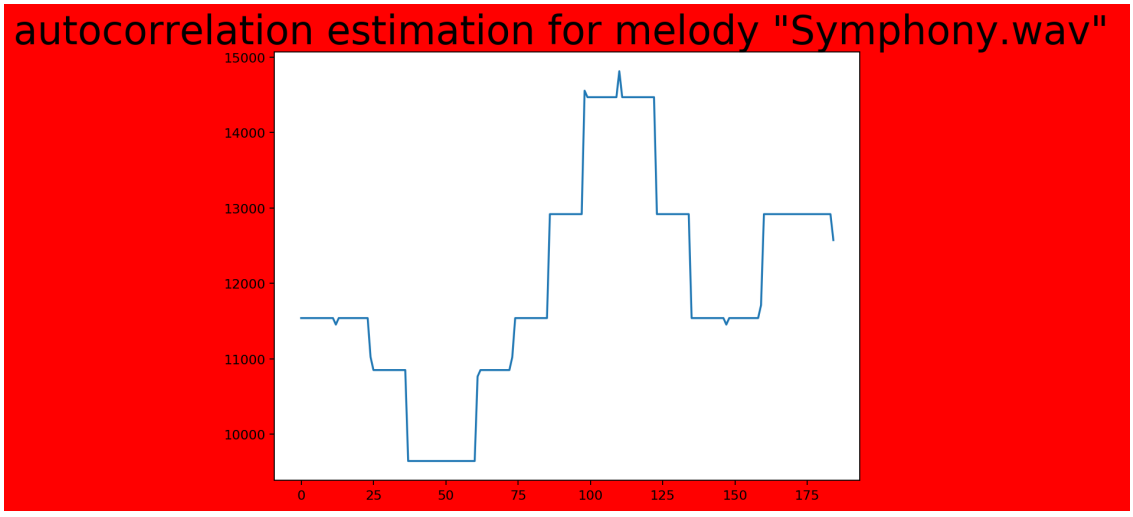
```
Question 1.2
==================================================
```

```
================
Series {
    // set hop size (size in samples) to 2048
    inSamples = 2048

    // read source file
    + input_file  = "qbh_examples.wav"

    // refer to the root of file use /<control
name>, input_file
    -> input: SoundFileSource { filename = /
input_file }

    // apply a windowing function to input signal
    -> Windowing {size = 2048}

    -> AutoCorrelation { setr0to1 = true
aliasedOutput = false}

    // pick peaks out of signal
    -> Peaker

    // calculate maximums and return position
    -> maximum : MaxArgMax

    // switch samples and observations
    -> Transposer

    // select different observations from input
data
    // Controls: "disable", "enables"
    -> selection: Selector {disable = 0 }

    // audio sink to CSV file named a3.csv
    -> sink : CsvSink { filename = "a3.csv" }

    // end of file is reached, control becomes
false
```

```
    + done = ( input/hasData == false )
}
```

In [294]:

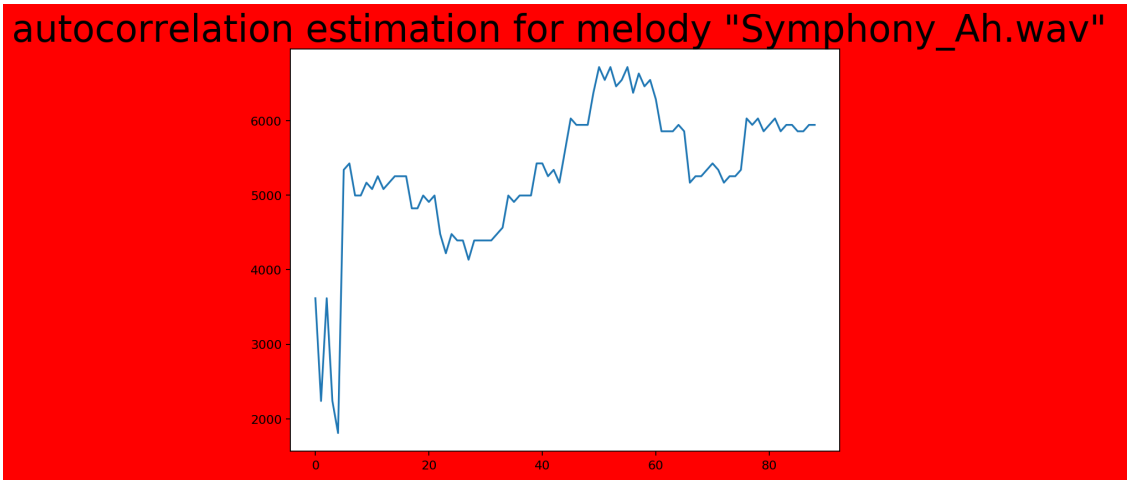
```
list2 = []
with open('a1.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list2.append(int(', '.join(row)))

w = [44100/512*x for x in list2]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
plt.title('autocorrelation estimation for melody
"Symphony.wav" ', fontsize = 30)
plt.plot(w)
plt.show()
```



In [295]:

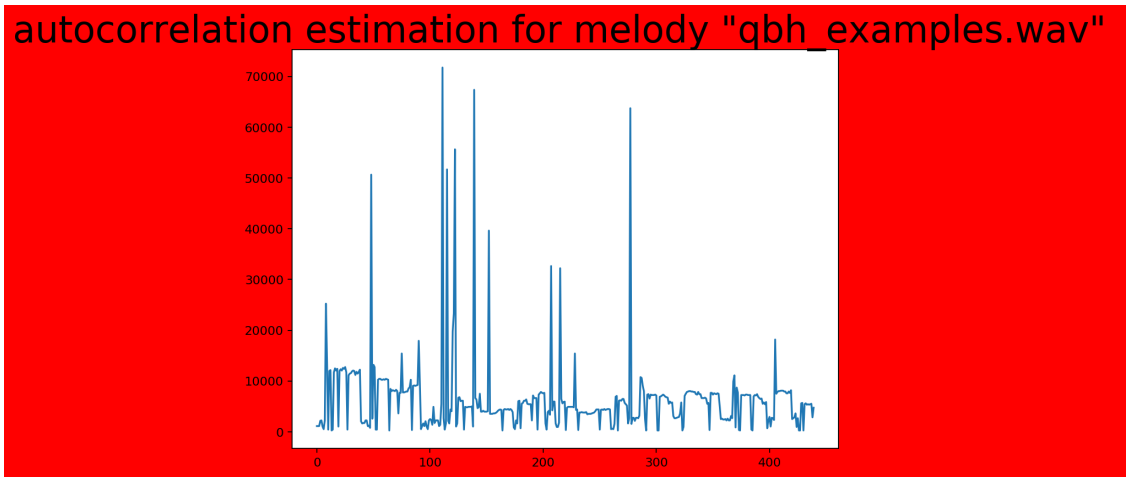
```
list2 = []
with open('a2.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list2.append(int(', '.join(row)))

w = [44100/512*x for x in list2]
```

```
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
plt.title('autocorrelation estimation for melody
"Symphony_Ah.wav" ', fontsize = 30)
plt.plot(w)
plt.show()
```



In [296]:

```
list2 = []
with open('a3.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list2.append(int(', '.join(row)))

w = [44100/512*x for x in list2]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
plt.title('autocorrelation estimation for melody
"qbh_examples.wav" ', fontsize = 30)
plt.plot(w)
plt.show()
```



In [ ]:

Question 1.3.1

==========

```
#By Comparing the fundamental frequency contour,
from the DFT approach I
#noticed that it obtained a similar shape contour
with both wave
#"Symphony" generated by program and the one sang
by myself. Some fluctuation
#could also be observed from the contour. From
autocorellation approach,
#the "Symphony" wave sang by my self is looking
more up and down. More
#fluctuations are noticable by comparing with DFT
method. There is also an
#error which their is two or three small peaks
seeing from autocorelation
#method of wave "Symphony" generated by program.
However, I also find that
#even though there are a lot of error and
fluctuations, but the fluctuation
#standard error is way smaller than DFT method (in
other words: stable).
#So I will asssume autocorelation is a better
choice for the most of time.
```

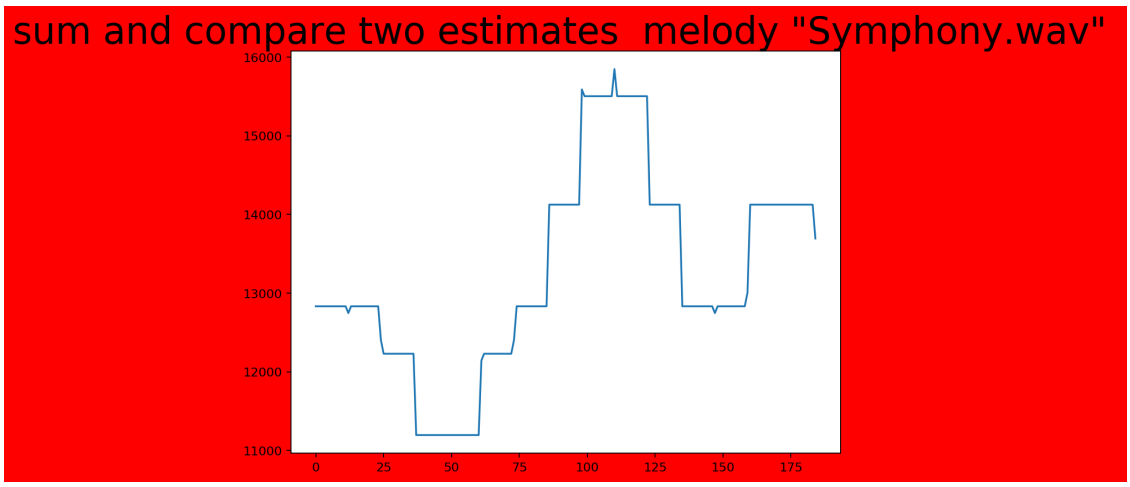
```
Question 1.3.2
================================================================
==================
Series {
    // set hop size (size in samples) to 2048
    inSamples = 2048

    // read source file
    + input_file  = "qbh_examples.wav"

    // refer to the root of file use /<control
name>, input_file
    -> input: SoundFileSource { filename = /
input_file }

    // apply a windowing function to input signal
    -> Windowing {size = 2048}

    // In Marsyas use a Fanout after the Windowing
followed by a Sum
    -> Fanout {
        -> Series{
            // computing spectrum
            -> Spectrum
            // Controls: "power","magnitude",
"decibels", "logmagnitude", etc..
            -> PowerSpectrum { spectrumType =
"logmagnitude" }
            -> Transposer
            // calculate maximums and return
position
            -> maximum1 : MaxArgMax
            -> Transposer
            // Controls: "disable", "enables"
            -> selection1: Selector {disable = 0 }
        }

        -> Series{
```

```
            // AutoCorrelation setting
            -> AutoCorrelation { setr0to1 = true
aliasedOutput = false}

            -> Peaker
            // calculate maximums and return
position
            -> maximum2 : MaxArgMax
            -> Transposer

            // select different observations from
input data
            // Controls: "disable", "enables"
            -> selection2: Selector {disable = 0 }
        }
    }

    -> Sumup: Sum
    // audio sink to CSV file named s3.csv
    -> sink : CsvSink { filename = "s3.csv" }

    // end of file is reached, control becomes
false
    + done = ( input/hasData == false )
}
```

In [297]:

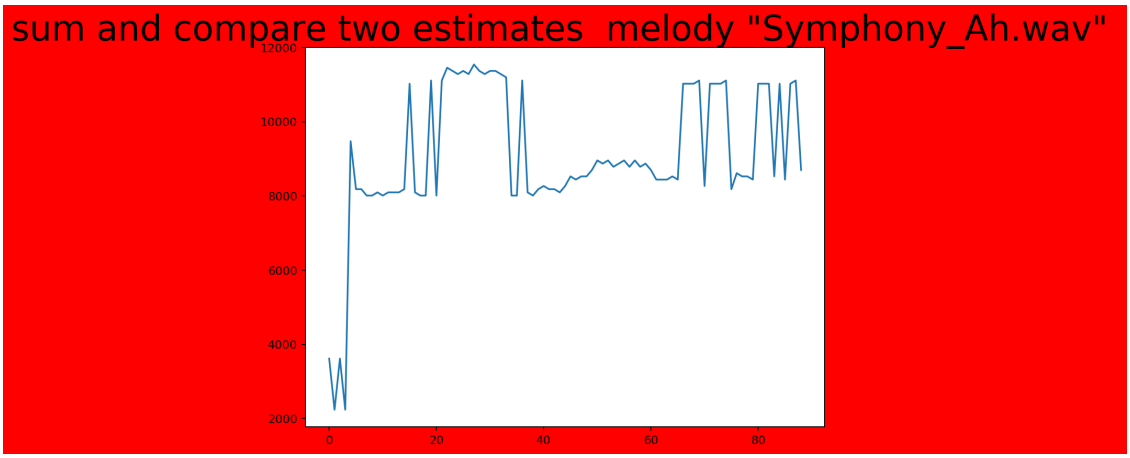
```
list3 = []
with open('s1.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list3.append(int(', '.join(row)))

w = [44100/512*x for x in list3]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
plt.title('sum and compare two estimates  melody
"Symphony.wav" ', fontsize = 30)
```

```
plt.plot(w)
plt.show()
```



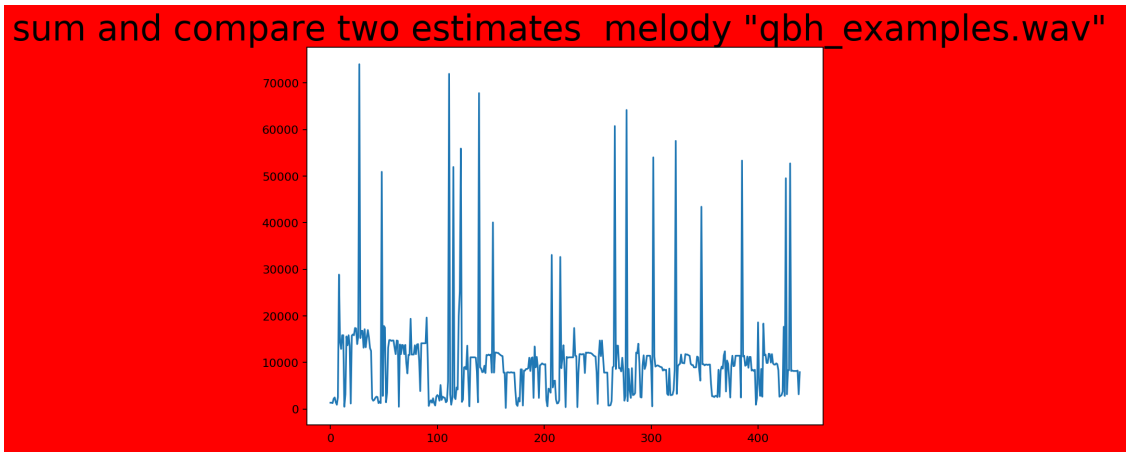
In [298]:

```
list3 = []
with open('s2.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list3.append(int(', '.join(row)))

w = [44100/512*x for x in list3]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
plt.title('sum and compare two estimates  melody
"Symphony_Ah.wav" ', fontsize = 30)
plt.plot(w)
plt.show()
```



In [299]:

```
list3 = []
with open('s3.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list3.append(int(', '.join(row)))

w = [44100/512*x for x in list3]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
plt.title('sum and compare two estimates  melody
"qbh_examples.wav" ', fontsize = 30)
plt.plot(w)
plt.show()
```



In [ ]:

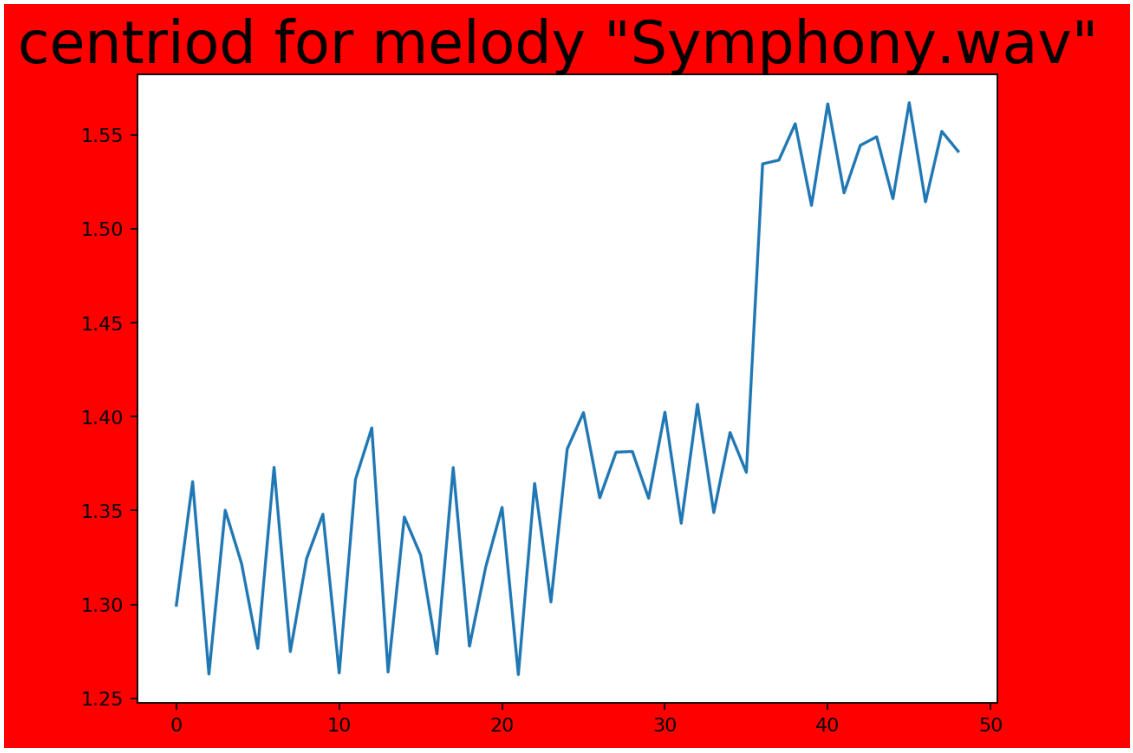
```
Question 2.1
```

```
================================================
===============
#As I heard from the audio generated by centriod, I
heard sometime buzzy,
#I think what happening is that the audio tells me
how good the quality of
#the wave file which I imported. How the sound
timbres behave for different
#music files. Like was said in class "brightness of
sound".

##################################################
##############NON-SMOOTHED CONTOUR################
##################################################
================================================
===============
Series {
    inSamples = 2048
    + input_file  = "classical.00001.wav"
    -> input: SoundFileSource { filename = /
input_file }
    //-> Windowing {size = 2048}

    // passes a complete copy of its input
    // to each of its children, and combine outputs
to multichannel
    -> Fanout {
        -> Series {
             // Perform the Centriod on system
             // Get a value of power spectral bin
number
            -> Spectrum
            -> PowerSpectrum
            -> Centroid
            -> Cen: FlowToControl
        }

         // Convert the bin index k to a frequency
```

```
f in Hertz:
         // RELATION FORMULA  "f = k * (Sr/N)",
         // where Sr is the sampling rate, and N is
the FFT size
        -> Series {
             // frequency = ( Censlice *
(44100.0/512.0))
            -> SineSource { frequency = (Cen/value
* (43)) }
             // Gain volumne
             -> Gain {gain = 20.0}
            -> AudioSink
        }
    }

    -> sink : CsvSink { filename = "c11.csv" }

    // end of file is reached, control becomes
false
    + done = (input/hasData == false)
}
```

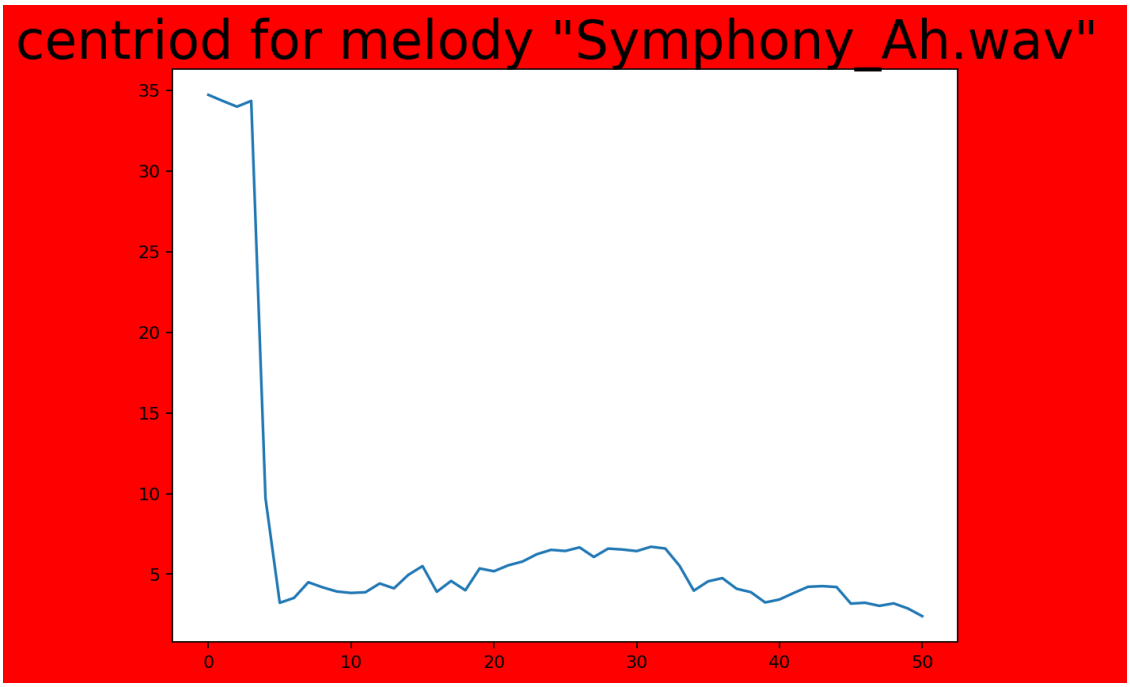
In [300]:

```
list4 = []
with open('c1.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list4.append(float(', '.join(row)))
w = [44100/512*x for x in list4]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
plt.title('centriod for melody "Symphony.wav" ',
fontsize = 30)
plt.plot(w)
plt.show()
```



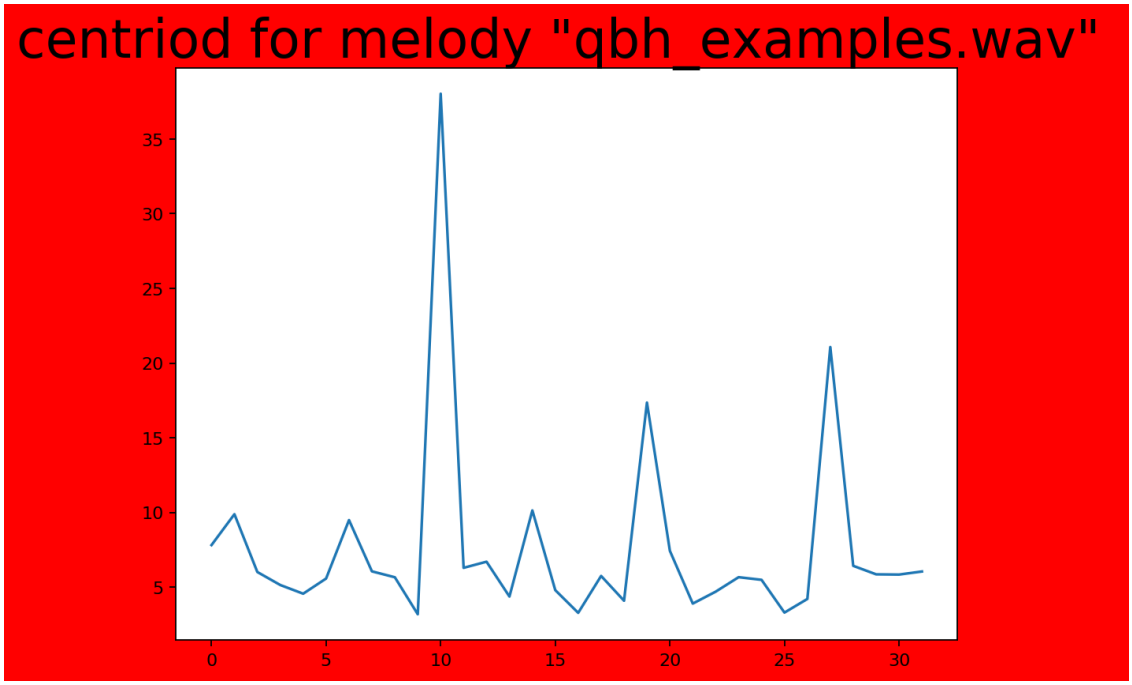
In [301]:

```
list4 = []
with open('c2.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list4.append(float(', '.join(row)))
w = [44100/512*x for x in list4]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
plt.title('centriod for melody "Symphony_Ah.wav" ',
fontsize = 30)
plt.plot(w)
plt.show()
```



In [302]:

```
list4 = []
with open('c3.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list4.append(float(', '.join(row)))
w = [44100/512*x for x in list4]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
plt.title('centriod for melody "qbh_examples.wav"
', fontsize = 30)
plt.plot(w)
plt.show()
```



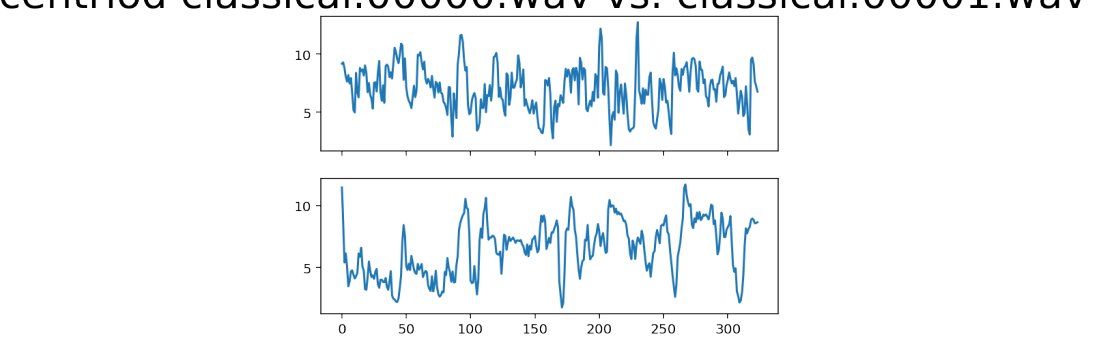
In [303]:

```
list5 = []
with open('cl0.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list5.append(float(', '.join(row)))

x = [44100/512*x for x in list5]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')

list6 = []
with open('cl1.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list6.append(float(', '.join(row)))

y = [44100/512*x for x in list6]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
```

```
f, axarr = plt.subplots(2, sharex=True)
axarr[0].plot(x)
axarr[1].plot(y)
axarr[0].set_title('centriod classical.00000.wav
vs. classical.00001.wav ', fontsize = 30)
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
plt.show()
```

```
<matplotlib.figure.Figure at 0xb545dd0>
```

```
<matplotlib.figure.Figure at 0xb545e10>
```




```
<matplotlib.figure.Figure at 0xd032050>
```

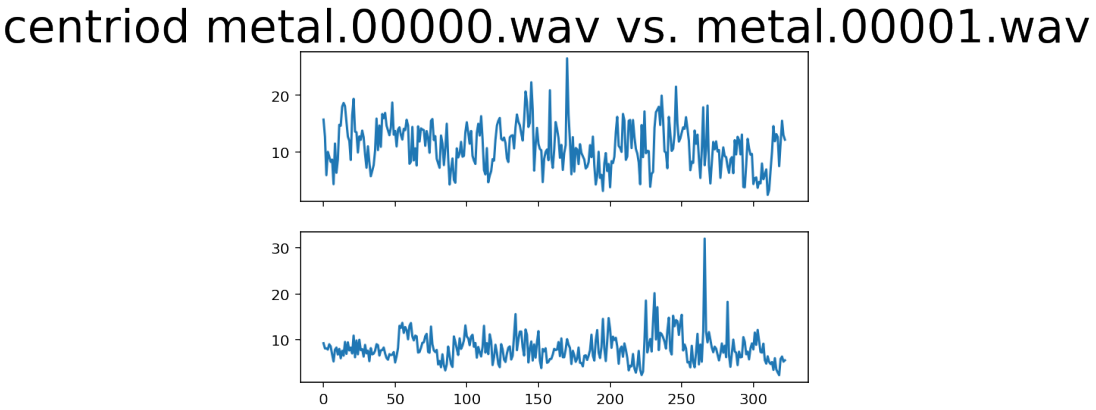
In [304]:

```
list5 = []
with open('m0.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list5.append(float(', '.join(row)))

x = [44100/512*x for x in list5]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')

list6 = []
with open('m1.csv', newline='') as csvfile:
```

```
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list6.append(float(', '.join(row)))

y = [44100/512*x for x in list6]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')


f, axarr = plt.subplots(2, sharex=True)
axarr[0].plot(x)
axarr[1].plot(y)
axarr[0].set_title('centriod metal.00000.wav vs.
metal.00001.wav ', fontsize = 30)
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
plt.show()
```

```
<matplotlib.figure.Figure at 0xb74a1d0>
```

```
<matplotlib.figure.Figure at 0xaf36b30>
```




```
<matplotlib.figure.Figure at 0xb746c90>
```

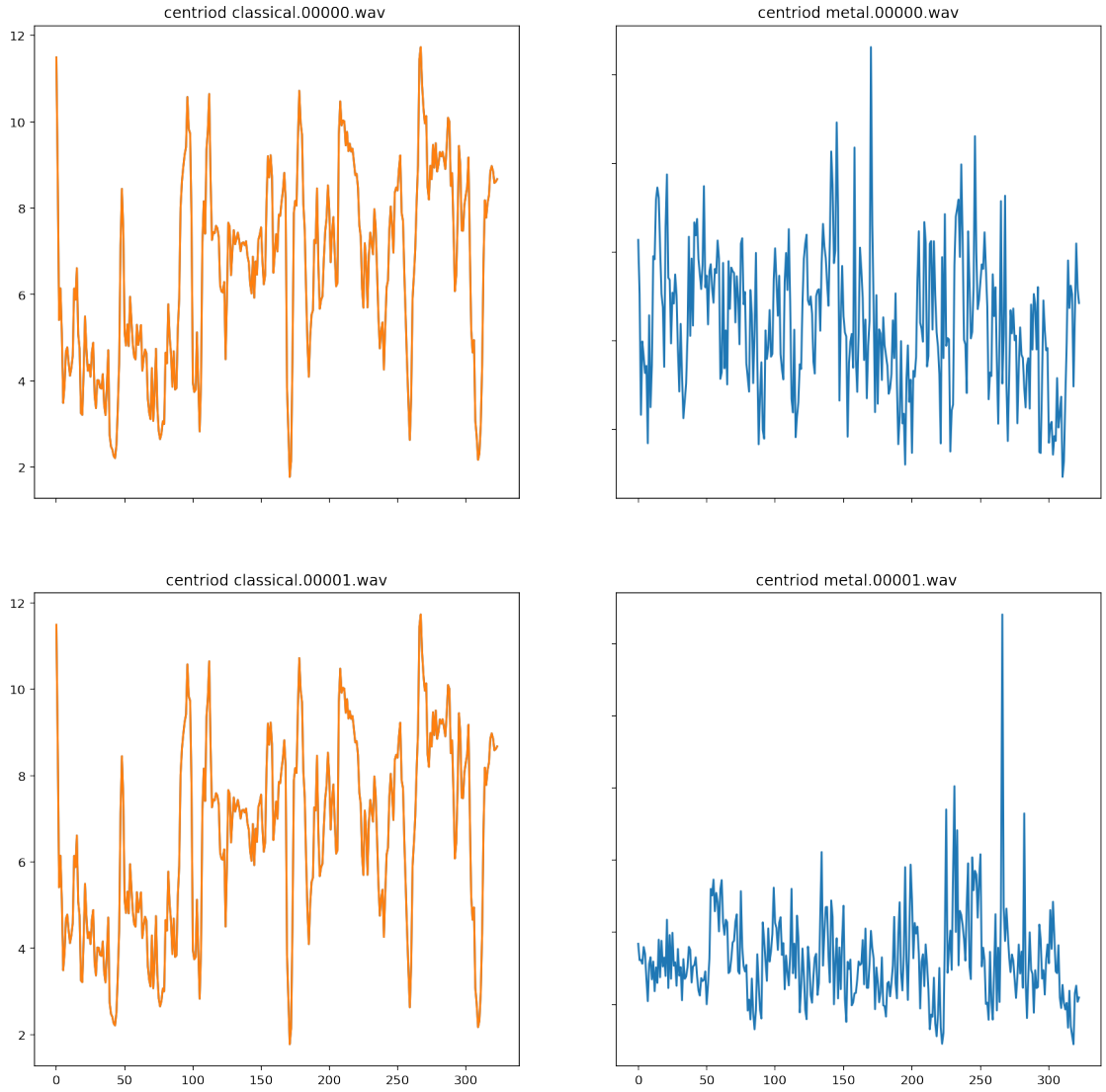
In [327]:

```
##########################################################
##############NON-SMOOTHED CONTOUR###############
##########################################################
```

```
list5 = []
with open('cl0.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list5.append(float(', '.join(row)))
w = [44100/512*x for x in list5]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')

list6 = []
with open('m0.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list6.append(float(', '.join(row)))

x = [44100/512*x for x in list6]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')


list5 = []
with open('cl1.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list5.append(float(', '.join(row)))

y = [44100/512*x for x in list5]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')

list6 = []
with open('m1.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
```

```
    for row in reader:
        list6.append(float(', '.join(row)))

z = [44100/512*x for x in list6]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')


#f, axarr = plt.subplots(2, sharex=True)
#axarr[0].plot(x)
#axarr[1].plot(y)
#axarr[0].set_title('centriod metal.00000.wav vs.
classical.00000.wav ', fontsize = 30)
#plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
#plt.show()

f, axarr = plt.subplots(2, 2,figsize=(15,15))
axarr[0, 0].plot(w)
axarr[0, 0].plot(w)
axarr[0, 0].set_title('centriod classical.00000.wav
')
axarr[0, 1].plot(x)
axarr[0, 1].set_title('centriod metal.00000.wav' )
axarr[1, 0].plot(y)
axarr[1, 0].plot(y)
axarr[1, 0].set_title('centriod classical.
00001.wav' )
axarr[1, 1].plot(z)
axarr[1, 1].set_title('centriod metal.00001.wav ')

# Fine-tune figure; hide x ticks for top plots and
y ticks for right plots
plt.setp([a.get_xticklabels() for a in
axarr[0, :]], visible=False)
plt.setp([a.get_yticklabels() for a in axarr[:,
1]], visible=False)
plt.show()
```

```
<matplotlib.figure.Figure at 0xb3c1ed0>

<matplotlib.figure.Figure at 0xb42fe10>

<matplotlib.figure.Figure at 0xb7e32d0>

<matplotlib.figure.Figure at 0xb8aba70>
```



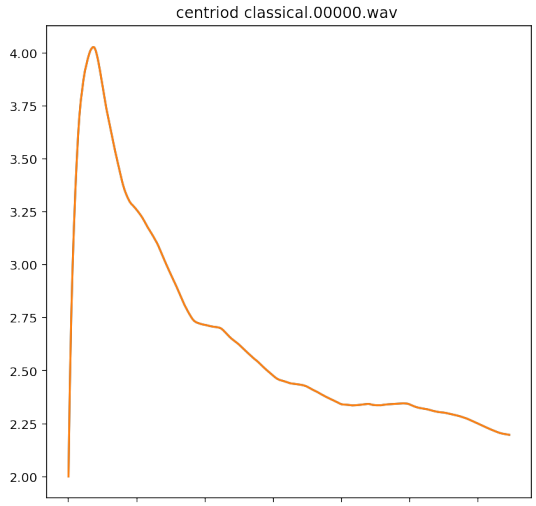
In [329]:

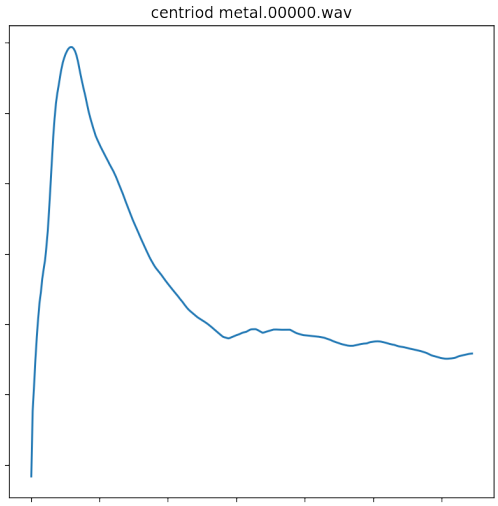
```
##########################################################
##############SMOOTHEER CONTOUR###########################
##########################################################
```

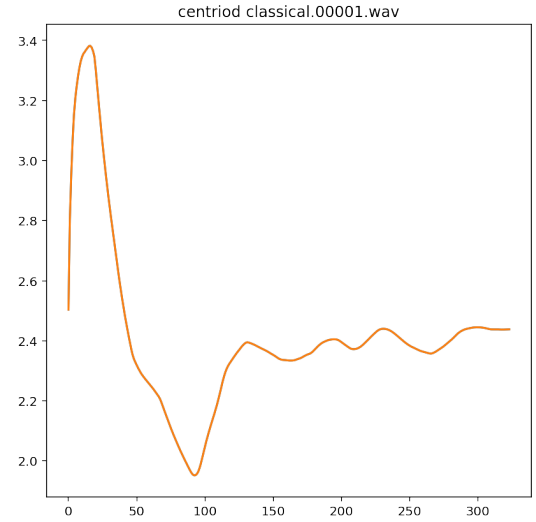
```
list5 = []
with open('cl10.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list5.append(float(', '.join(row)))
w = [44100/512*x for x in list5]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')

list6 = []
with open('m10.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list6.append(float(', '.join(row)))

x = [44100/512*x for x in list6]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')


list5 = []
with open('cl11.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
    for row in reader:
        list5.append(float(', '.join(row)))

y = [44100/512*x for x in list5]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')

list6 = []
with open('m11.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ',
quotechar='|')
```

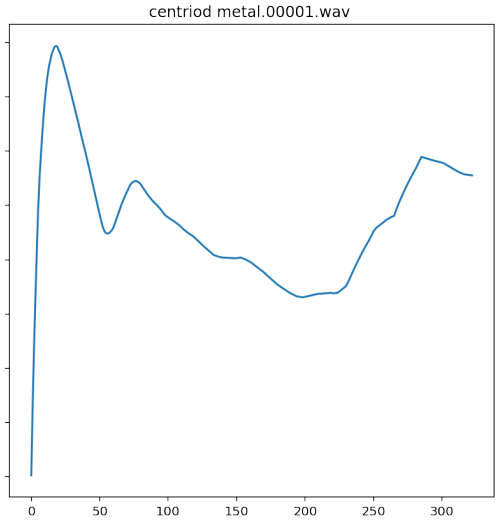
```
    for row in reader:
        list6.append(float(', '.join(row)))

z = [44100/512*x for x in list6]
plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')

#f, axarr = plt.subplots(2, sharex=True)
#axarr[0].plot(x)
#axarr[1].plot(y)
#axarr[0].set_title('centriod metal.00000.wav vs.
classical.00000.wav ', fontsize = 30)
#plt.figure(num=None, figsize=(8, 6), dpi=80,
facecolor='red', edgecolor='k')
#plt.show()

f, axarr = plt.subplots(2, 2,figsize=(15,15))
axarr[0, 0].plot(w)
axarr[0, 0].plot(w)
axarr[0, 0].set_title('centriod classical.00000.wav
')
axarr[0, 1].plot(x)
axarr[0, 1].set_title('centriod metal.00000.wav' )
axarr[1, 0].plot(y)
axarr[1, 0].plot(y)
axarr[1, 0].set_title('centriod classical.
00001.wav' )
axarr[1, 1].plot(z)
axarr[1, 1].set_title('centriod metal.00001.wav ')

# Fine-tune figure; hide x ticks for top plots and
y ticks for right plots
plt.setp([a.get_xticklabels() for a in
axarr[0, :]], visible=False)
plt.setp([a.get_yticklabels() for a in axarr[:,
1]], visible=False)
plt.show()
```

```
<matplotlib.figure.Figure at 0xb46f210>

<matplotlib.figure.Figure at 0xad2f7d0>

<matplotlib.figure.Figure at 0xb463930>

<matplotlib.figure.Figure at 0xb463e70>
```









In [ ]:

```
Question 2.2

##########################################
##############SMOOTHED CONTOUR#############
##########################################
```

```
================================================================
===============
Series {
    inSamples = 2048
    + input_file  = "classical.00001.wav"
    -> input: SoundFileSource { filename = /
input_file }
    //-> Windowing {size = 2048}

    // passes a complete copy of its input
    // to each of its children, and combine outputs
to multichannel
    -> Fanout {
        -> Series {
             // Perform the Centriod on system
             // Get a value of power spectral bin
number
            -> Spectrum
            -> PowerSpectrum
            -> Centroid

            //################### Mean & 20
previous sampels
             -> Memory { memSize = 20}
             -> RunningStatistics {enableMean =
true}


             -> Selection: Selector {disable = 0 }
            -> Cen: FlowToControl
        }

         // Convert the bin index k to a frequency
f in Hertz:
         // RELATION FORMULA  "f = k * (Sr/N)",
         // where Sr is the sampling rate, and N is
the FFT size
        -> Series {
```

```
            // frequency = ( Censlice *
(44100.0/512.0))
            -> SineSource { frequency = (Cen/value
* (43)) }
             // Gain volumne
             -> Gain {gain = 20.0}
             //-> SoundFileSink { filename =
"output.wav" }
            -> AudioSink
        }
    }
    -> sink : CsvSink { filename = "cl11.csv" }

    // end of file is reached, control becomes
false
    + done = (input/hasData == false)
}
================================================
=
#Different genres of music shows similar trend in
centroid graph. For example,
#centroid for classical music is more stablized,
whereas for metal generally more
#fluctuations. For metal genre, the contour is
higher than contour with classical.
#Maybe it is because for classical music sometimes
quiet some times loud, whereas
#for metal it is always loud and high frequent.
```