**Assignment#3 Report**

**Team 9 (Jun Ma, Zhuojian Zhong, Chang Liu)**

**1. Code Package**

<https://github.com/jma19/CS244Fall2017.git>

Our code for this assignment locates in package Assignment#3/caculator.ipynb

**2. Band Pass Filter**

We design bandpass filter based on Butterworth filter. butter\_bandpass\_filter function provides bandpass filter implementation to filter data according to input parameters, and its implementation shown as follows:

# data: input data

# lowcut: low frequency

# highcut: high frequency

# fs: sample frequency

# order: order of Butterworth filter

**def** butter\_bandpass\_filter(data, lowcut, highcut, fs, order):

b, a = butter\_bandpass(lowcut, highcut, fs, order=order)

y = lfilter(b, a, data)

**return** y

Following figure shows the frequency response for our bandpass filter.

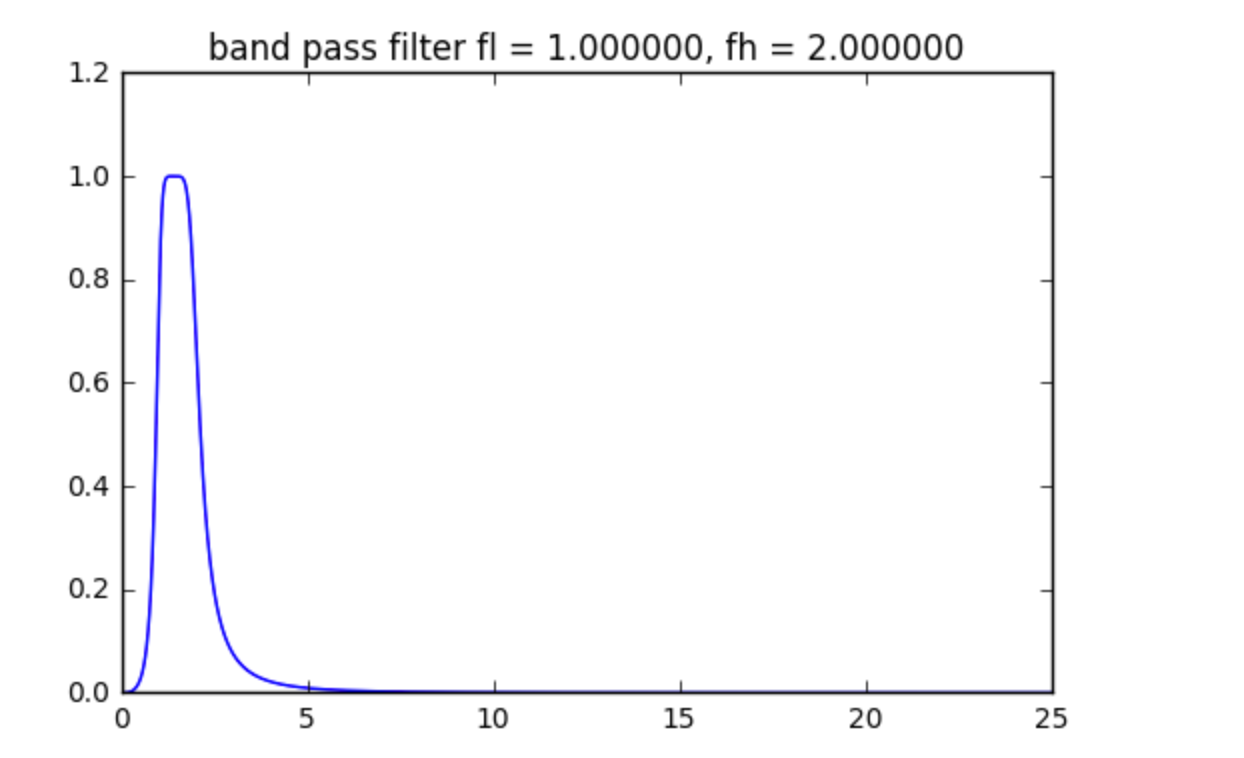


Fig.1 Frequency Response for our band pass filter with lowcut = 1Hz, highcut = 2Hz, order = 3

**2. Preprocessing data**

In order to simply data processing, we can normalize data (IR, RED) into range [0, 1] using following code.

data = np.genfromtxt("takashin\_Homework\_sample.csv", dtype=float, delimiter=',', names=True)

timestamp = data['timesecond']

scaler = preprocessing.MinMaxScaler()

*# normazile data*

IR = scaler.fit\_transform(data['IR'])

RED = scaler.fit\_transform(data['RED'])

**3. Calculate Heart Rate**

The procedure to calculate heart rate is shown as follows:

1. Filter normalized IR data using bandpass filter with lowcut = 1Hz, highcut = 2Hz, order = 3
2. Using Python PeakUtils library to calculate all indexes of positive peak points.
3. Calculate the time interval vector for neighbor positive peaks
4. Calculate the heart rate using formula 60 / time interval

Fig.2 shows the normalized IR data, and Fig.3 shows the filtered heart rate signal with peak detection.

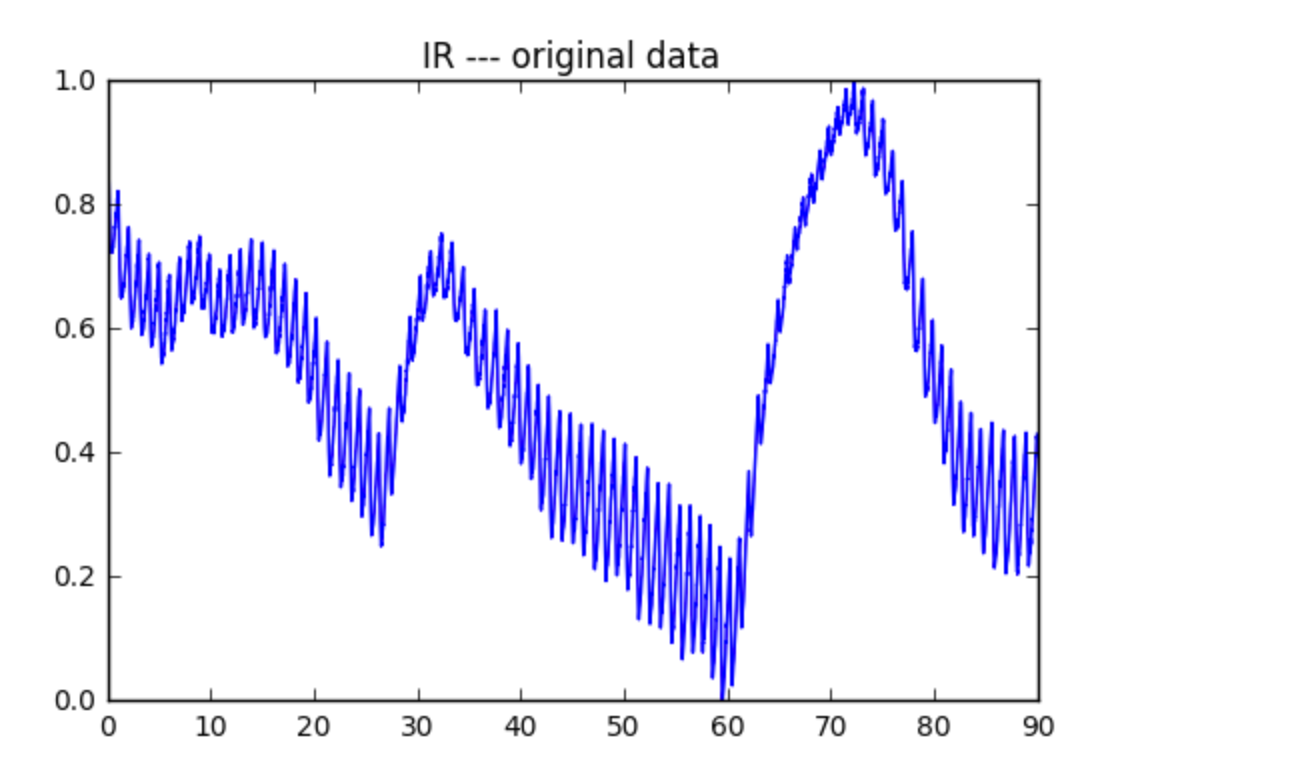


Fig.2 Normalized IR data

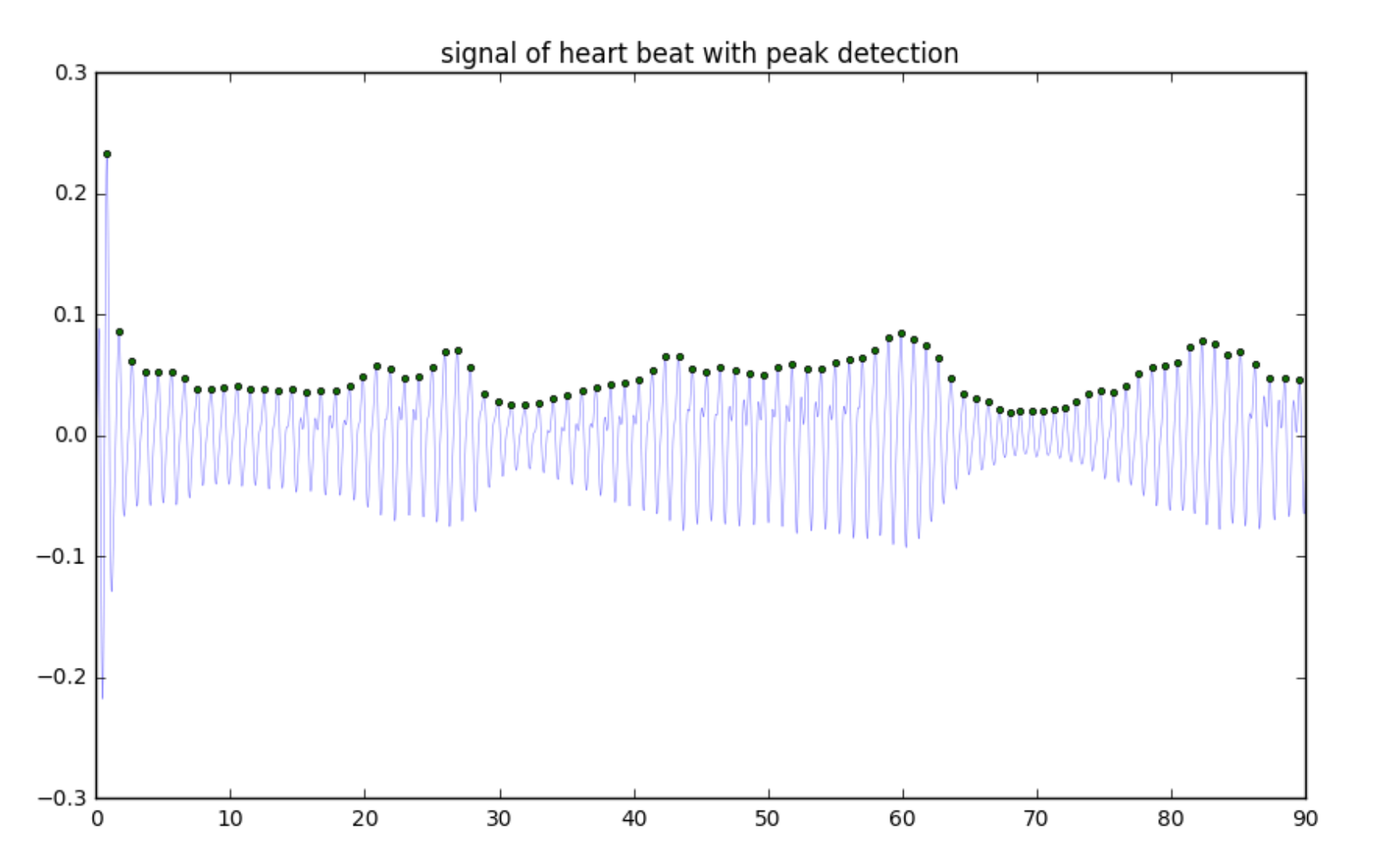


Fig.3 heart beat signal with peak detection.

Its main implementation shows as follows:

**def** caculateRate(timestamp, data, thres, min\_dist, fs):

peakind = peakutils.indexes(data, thres, min\_dist)

numberOfPeak = len(peakind)

**print** "number of peak **%d**" % numberOfPeak

rate = np.zeros(numberOfPeak - 1)

**for** i **in** range(1, numberOfPeak):

interval = timestamp[peakind[i]] - timestamp[peakind[i-1]]

rate[i-1] = 60 / interval

**return** rate

Fig.4 shows the calculate heart rate signal.

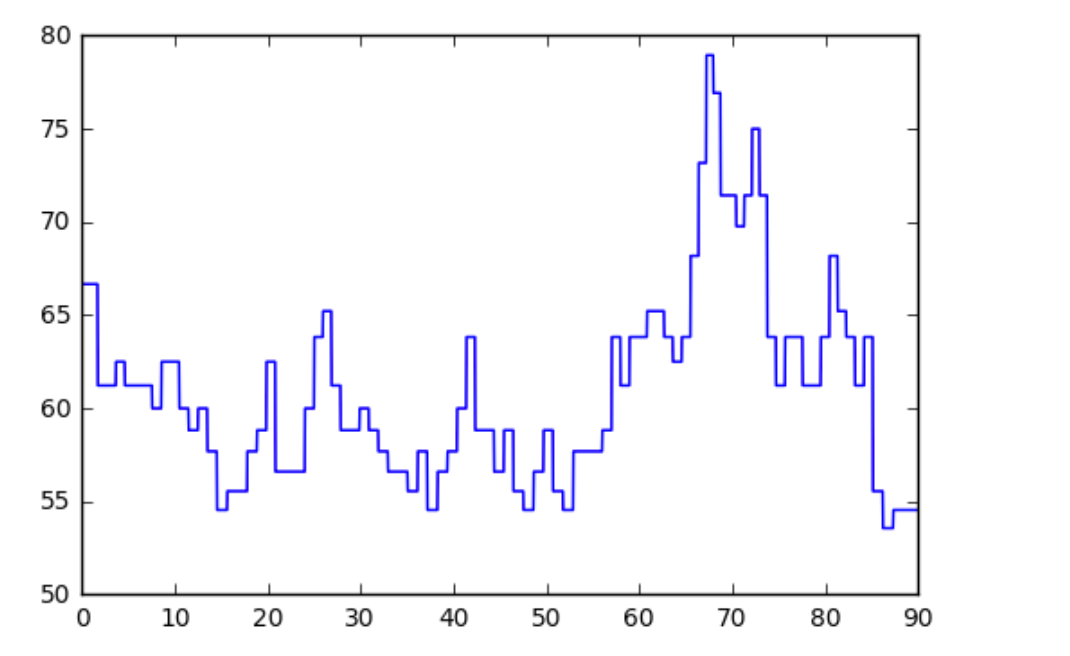


Fig.4 Calculated heart beat rate signal

**3. Calculate Respiration Rate**

The procedure to calculate respiration rate is same as heart rate, but we use lowcut = 0.15Hz, highcut = 0.35Hz to filter data. Fig.5 shows the respiration rate signal with peak detection, and Fig.6 shows the calculated respiration rate.

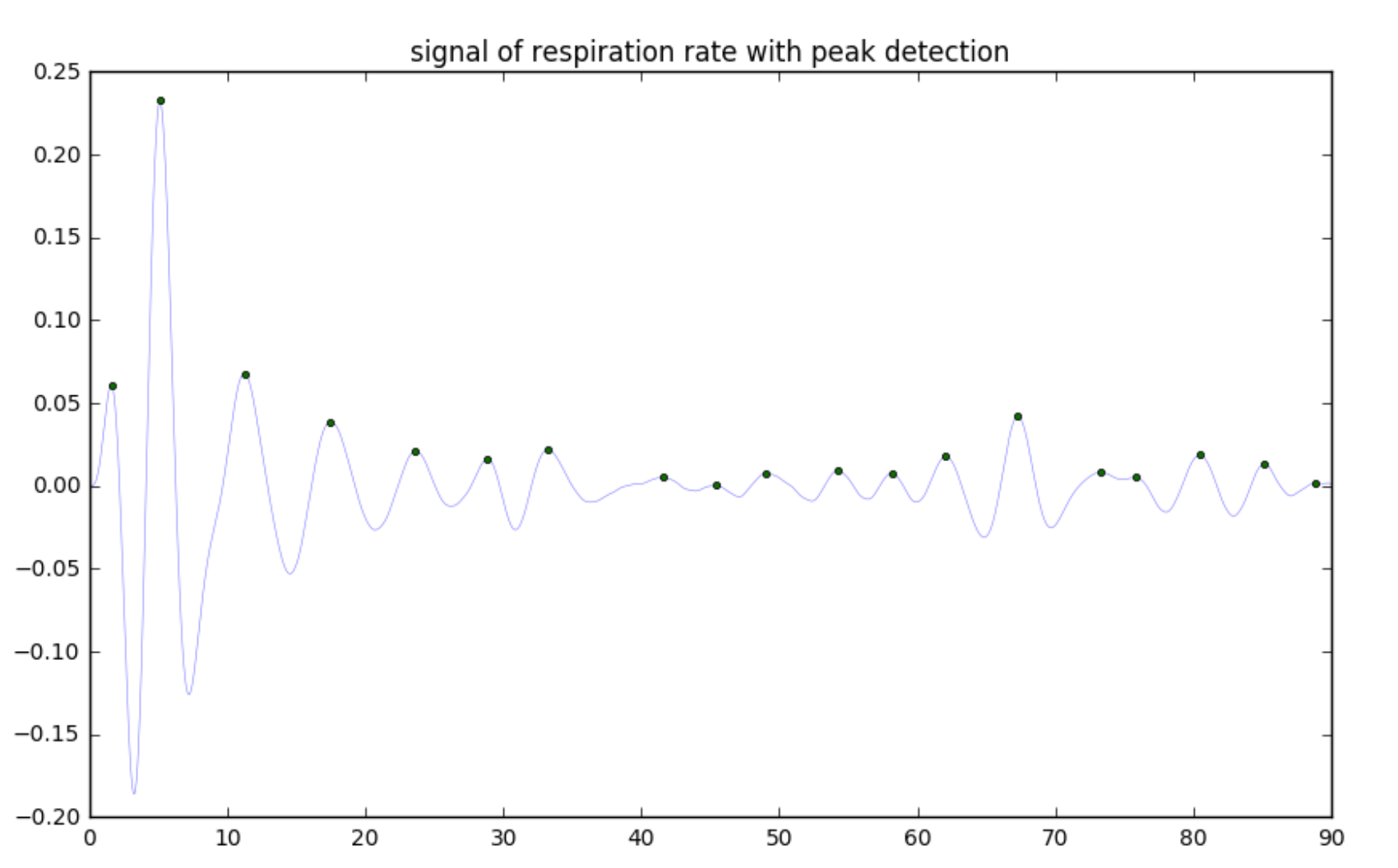


Fig.5 Respiration signal with peak detection.

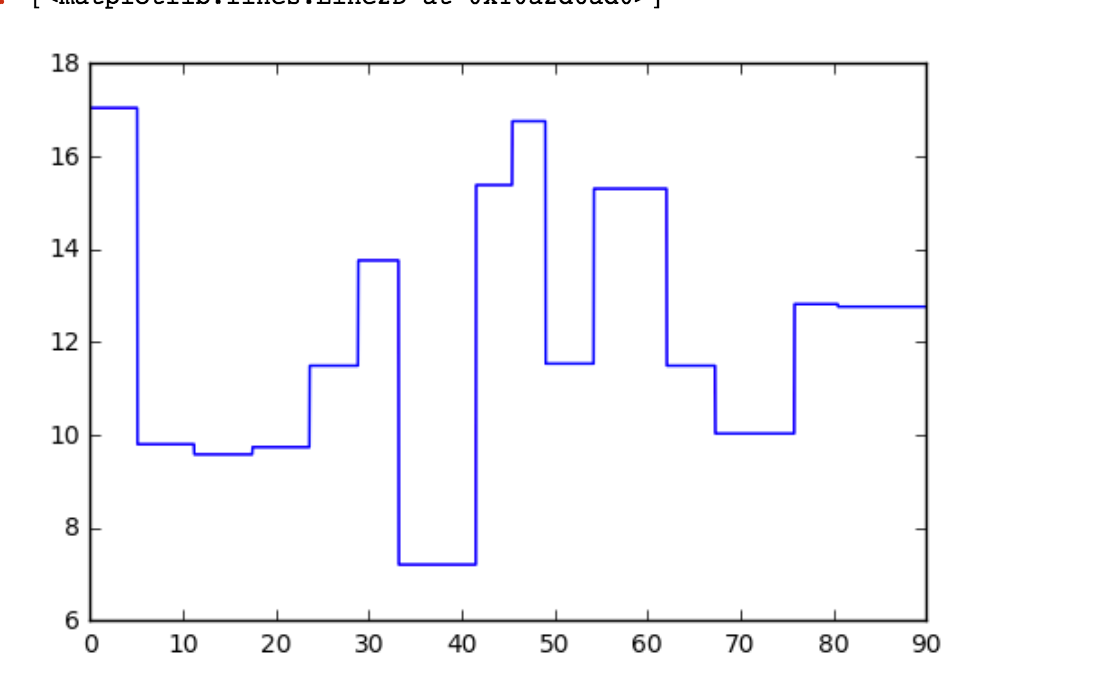


Fig.6 Calculated heart rate signal

**4. Calculate SPO2**

In order to get the negative (low) peak, when we use PeakUtils to get all peak indexs, we need to flip the input data. Fig. 7 shows the IR and RED signal with peak detection including positive and negative peak. We label all position of peaks. Fig.8 shows the calculated SPO2

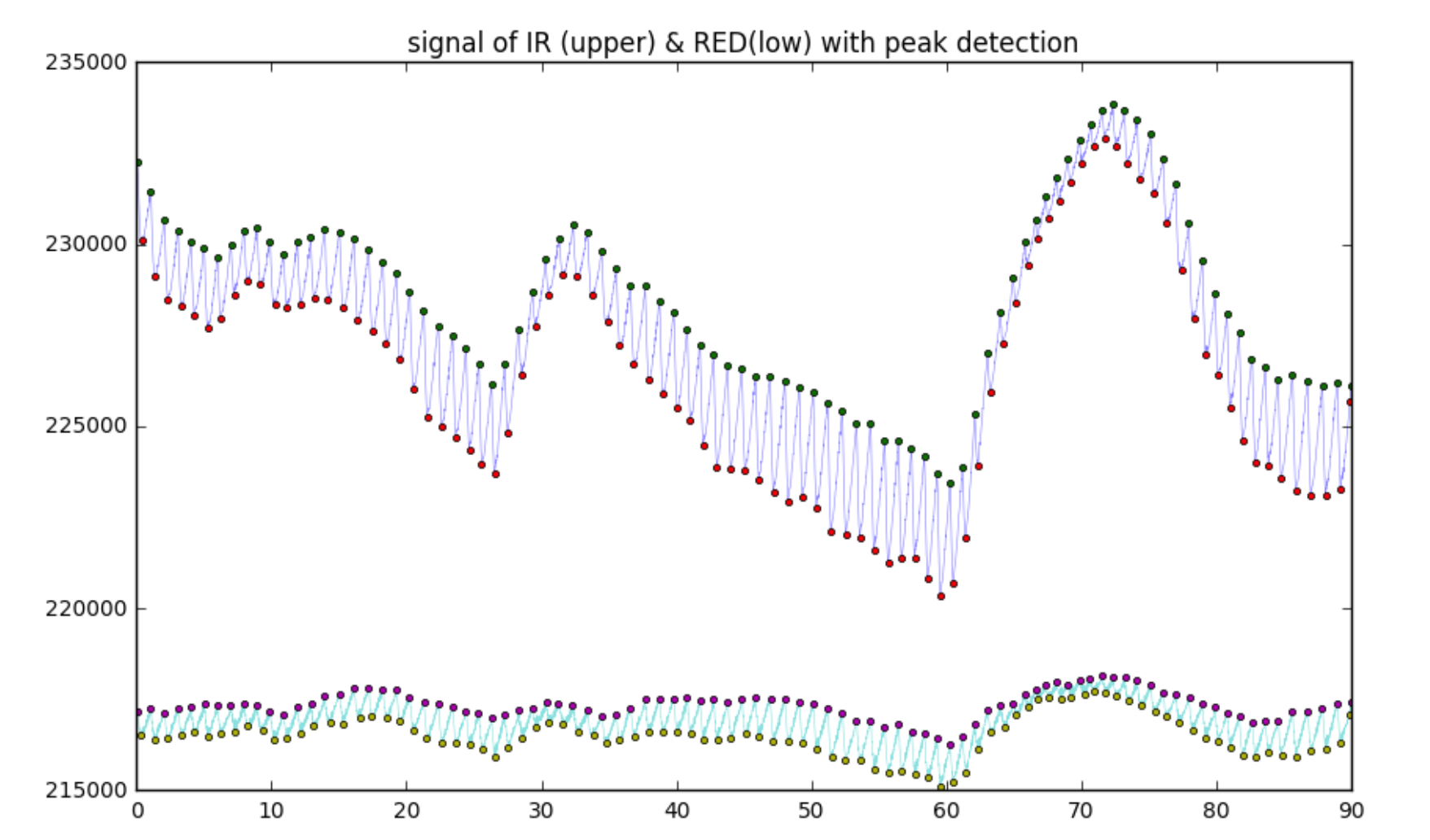


Fig.7 IR and RED signal with peak detection

The following codes shows how to get positive and negative peaks.

*# the index of "postive" peak for IR signal*

indexPositiveIR = peakutils.indexes(IROrgin, thres, min\_dist)

*# the index of "negative" peak for IR signal*

indexNegativeIR = peakutils.indexes(-1\*IROrgin, thres, min\_dist)

*# the index of "postive" peak for RED signal*

indexPositiveRED = peakutils.indexes(REDOrgin, thres, min\_dist)

*# the index of "negative" peak for RED signal*

indexNegativeRED = peakutils.indexes(-1\*REDOrgin, thres, min\_dist)

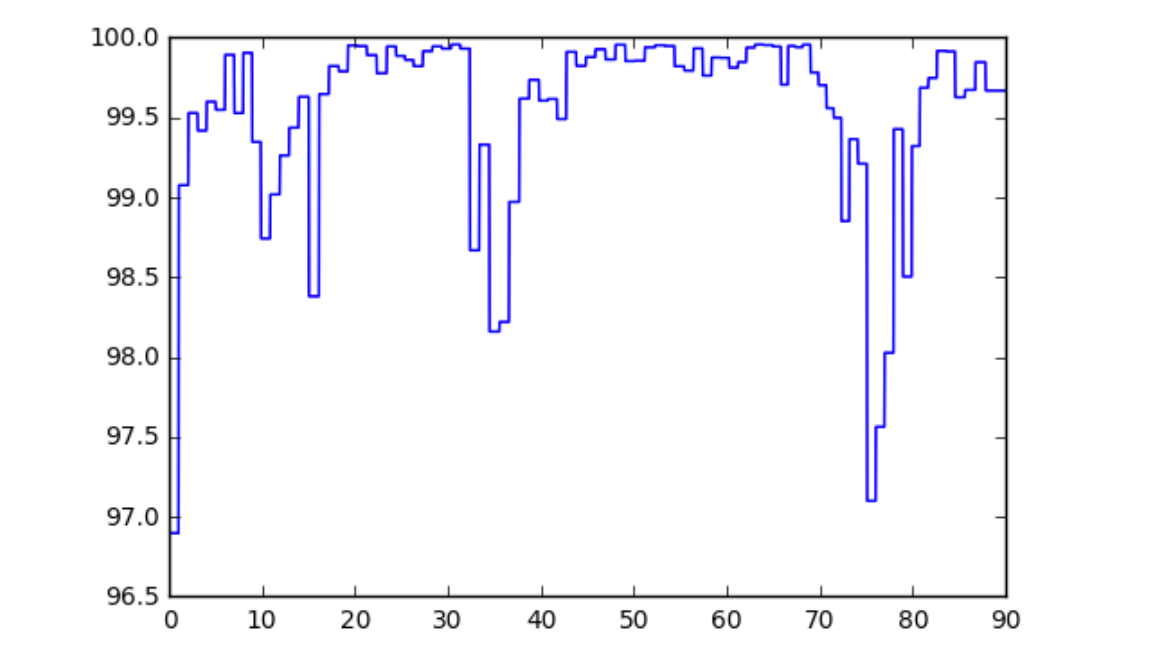
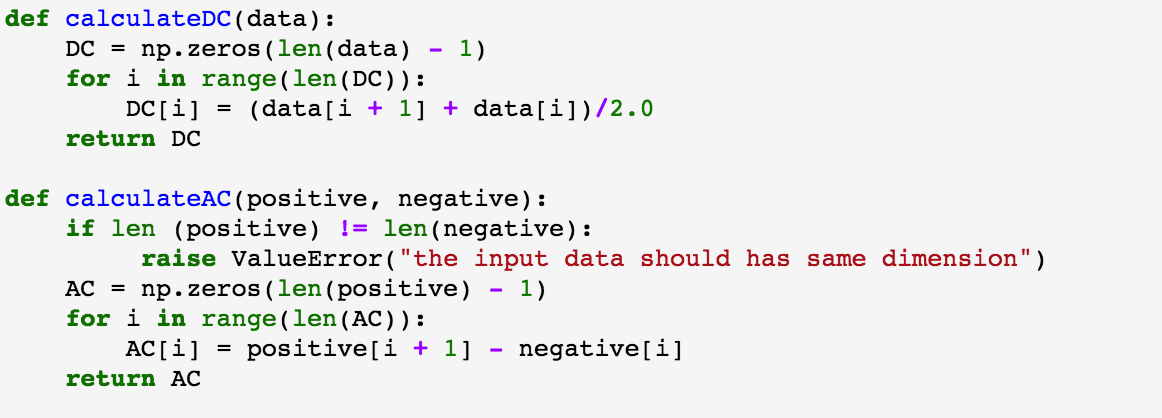


Fig.8 Calculated SPO2 signal

In order to get DC for a signal, we first need to get all indexes of negative peaks for a signal, and then get amplitude interval for neighboring negative peak. In our program, we choose the middle amplitude as the DC value for any neighboring negative peak. For the DC value, we use the positive peak amplitude minus its previous negative neighbor amplitude, and the implementation is shown as follows:



The following code shows how to calculate the SPO2 when we get AC, and DC for IR and RED signal.

