PQRS Complex Detection Algorithm

-By Mosaif Ali

Methodology.

1. Signal Acquisition

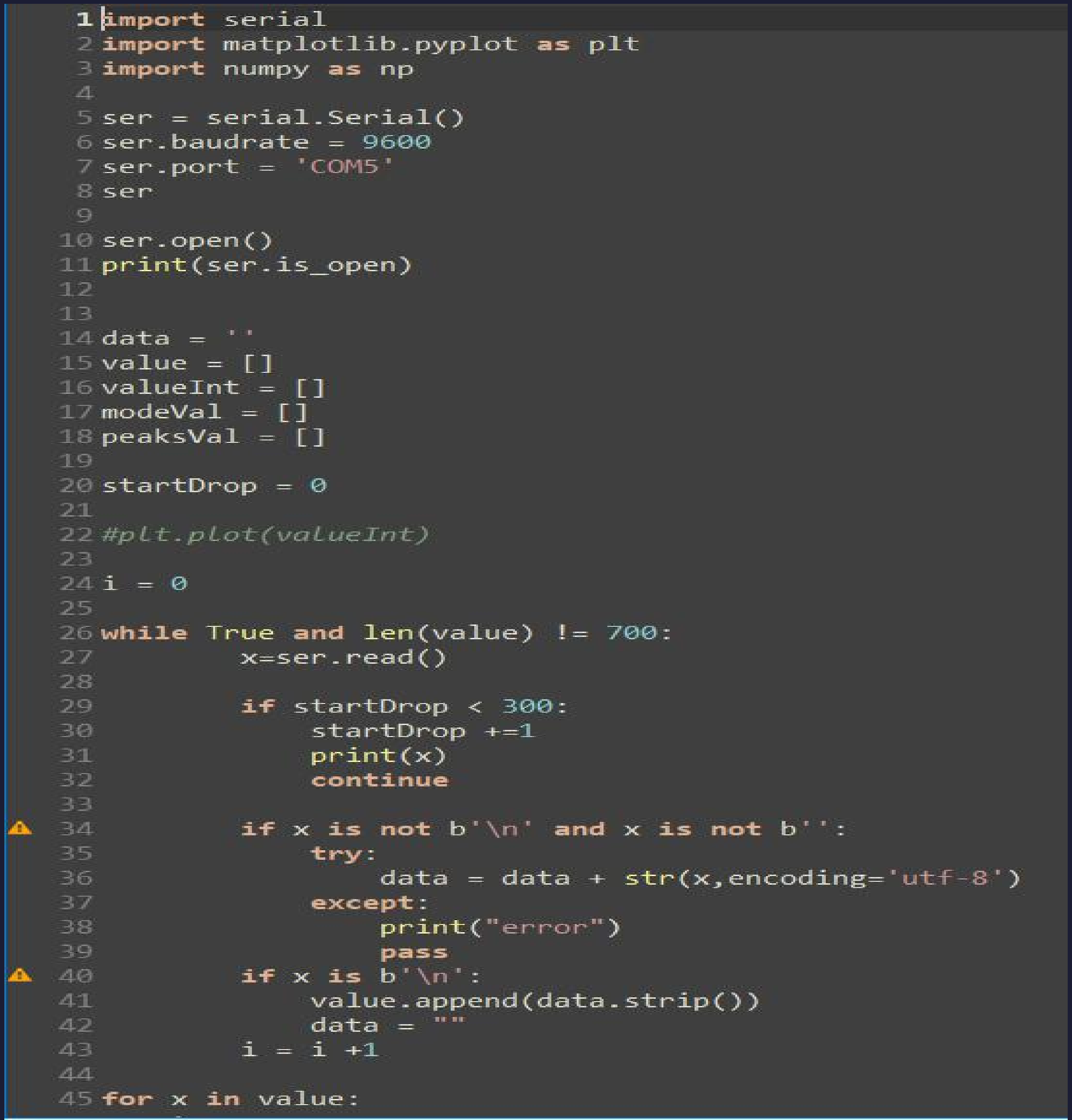
2. Processing of Acquired Data

3. Removal of Noise and refreshment of Signal- Via Discrette Wavelet Transform

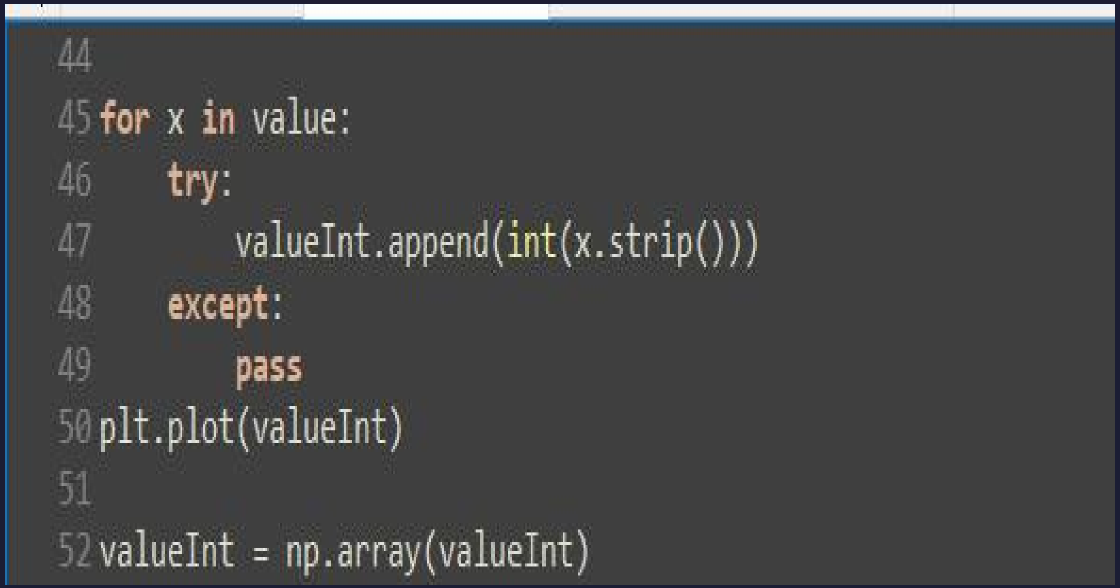
4. Recognition of Resonable R points

5. Using Detected R points as Reference for Detecting PQS points.

1. Signal Acquisition & Processing of Acquired Data.

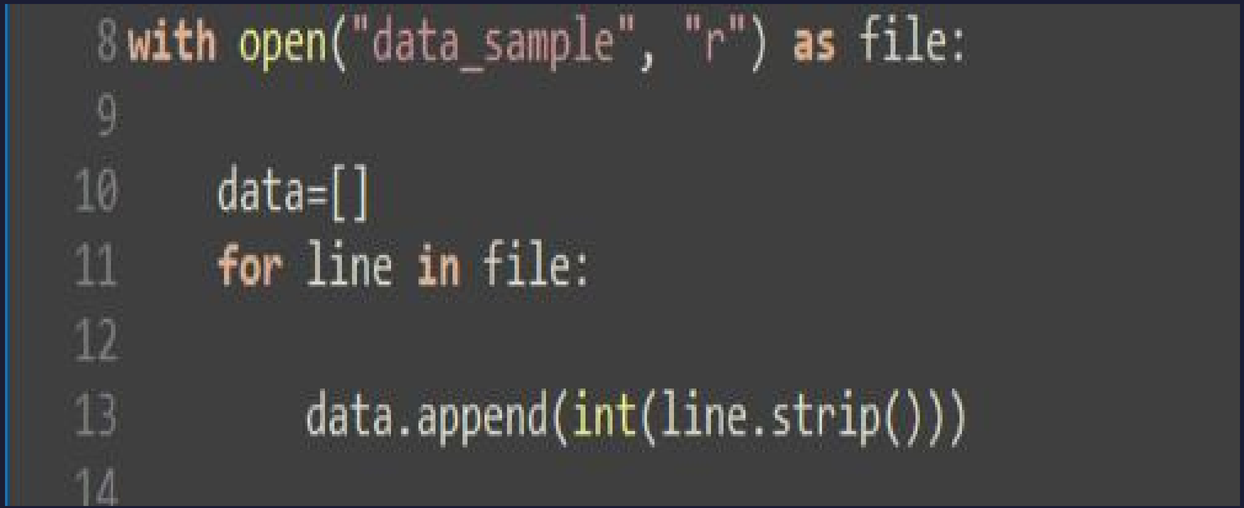


```python
import serial
import matplotlib.pyplot as plt
import numpy as np

ser = serial.Serial()
ser.baudrate = 9600
ser.port = 'COM5'
ser

ser.open()
print(ser.is_open)


data = ''
value = []
valueInt = []
modeVal = []
peaksVal = []

startDrop = 0

#plt.plot(valueInt)

i = 0

while True and len(value) != 700:
        x=ser.read()

        if startDrop < 300:
            startDrop +=1
            print(x)
            continue

        if x is not b'\n' and x is not b'':
            try:
                data = data + str(x,encoding='utf-8')
            except:
                print("error")
                pass
        if x is b'\n':
            value.append(data.strip())
            data = ""
        i = i +1

for x in value:
```



```python

for x in value:
    try:
        valueInt.append(int(x.strip()))
    except:
        pass
plt.plot(valueInt)

valueInt = np.array(valueInt)
```

----------------XXXXXXXXXXXXXXXXXXXXXXXXXXXXX-------------



```python
with open("data_sample", "r") as file:

    data=[]
    for line in file:

        data.append(int(line.strip()))

```

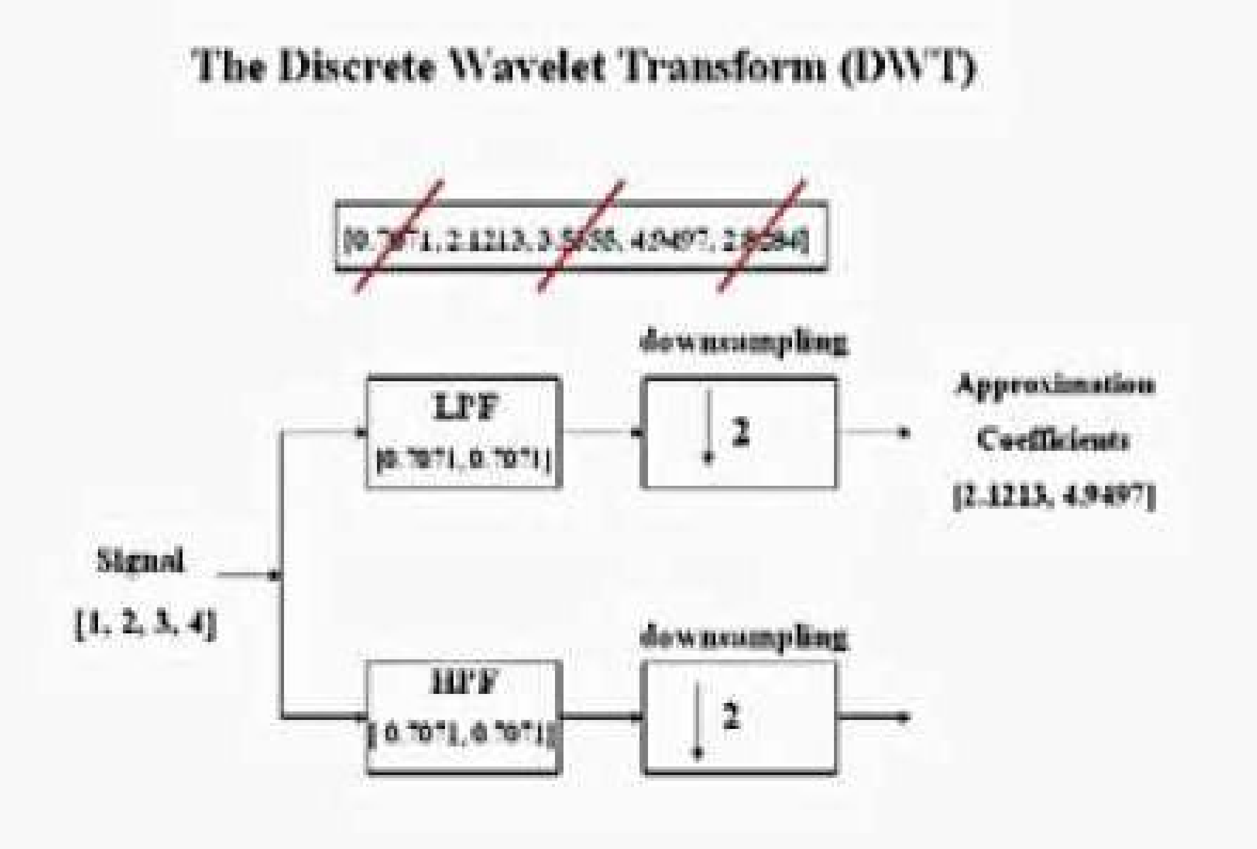
3. Removal of Noise and refreshment of Signal- Via Discrette Wavelet Transform.

- DWT is used to remove Baseline Wander (Noise) in the ECG signal.

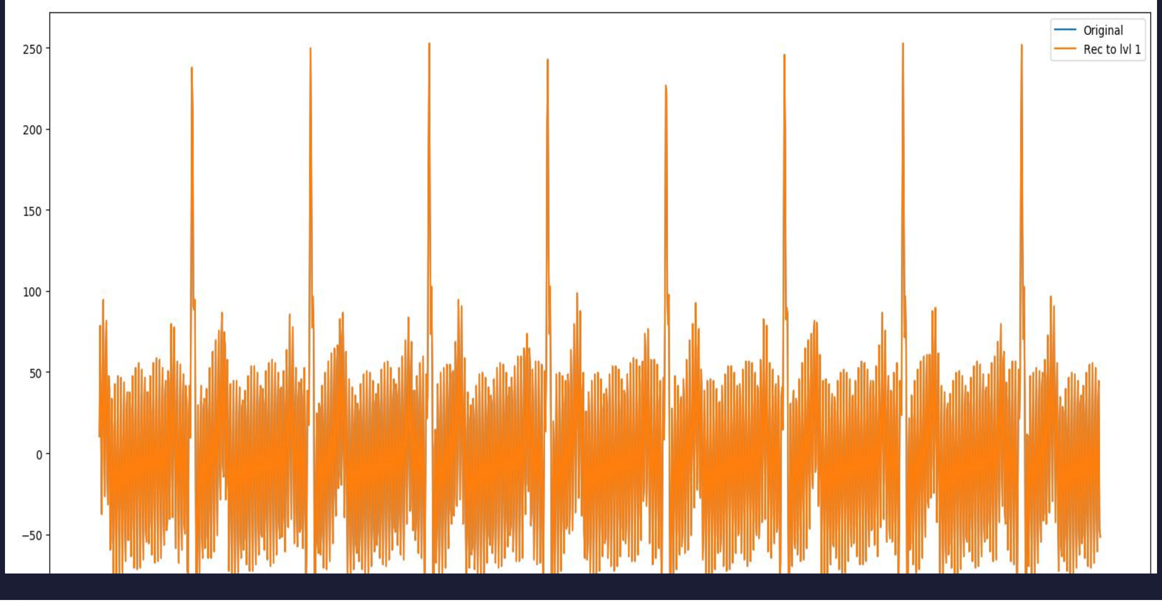
- Uses High and Low Pass Filters

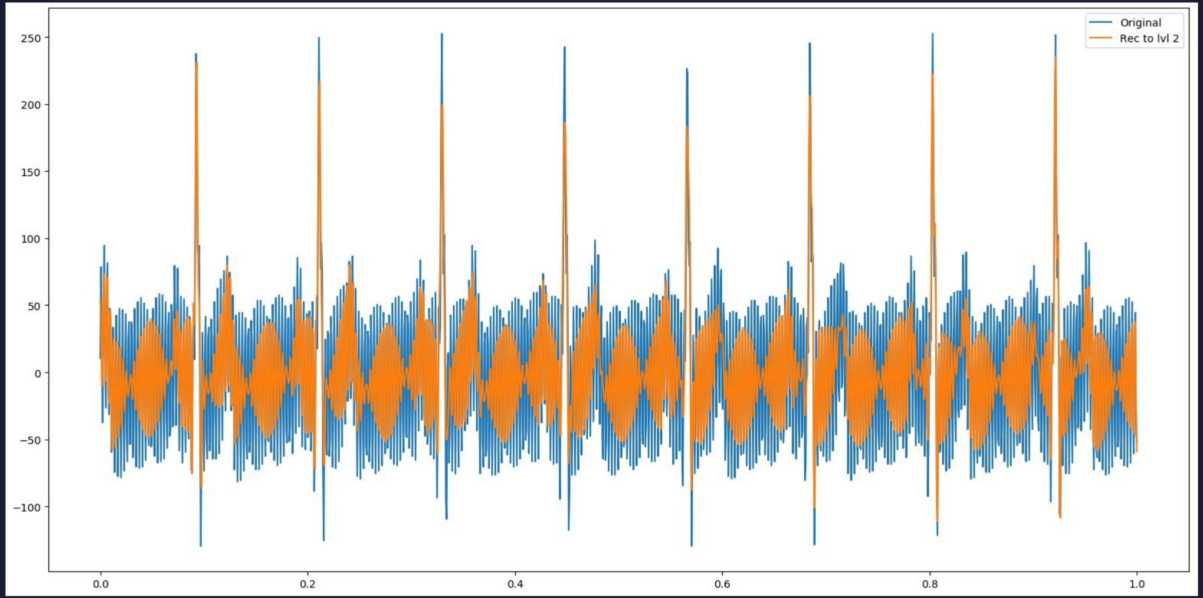
- Removal of Noise is done by decomposition of signal upto 8 levels

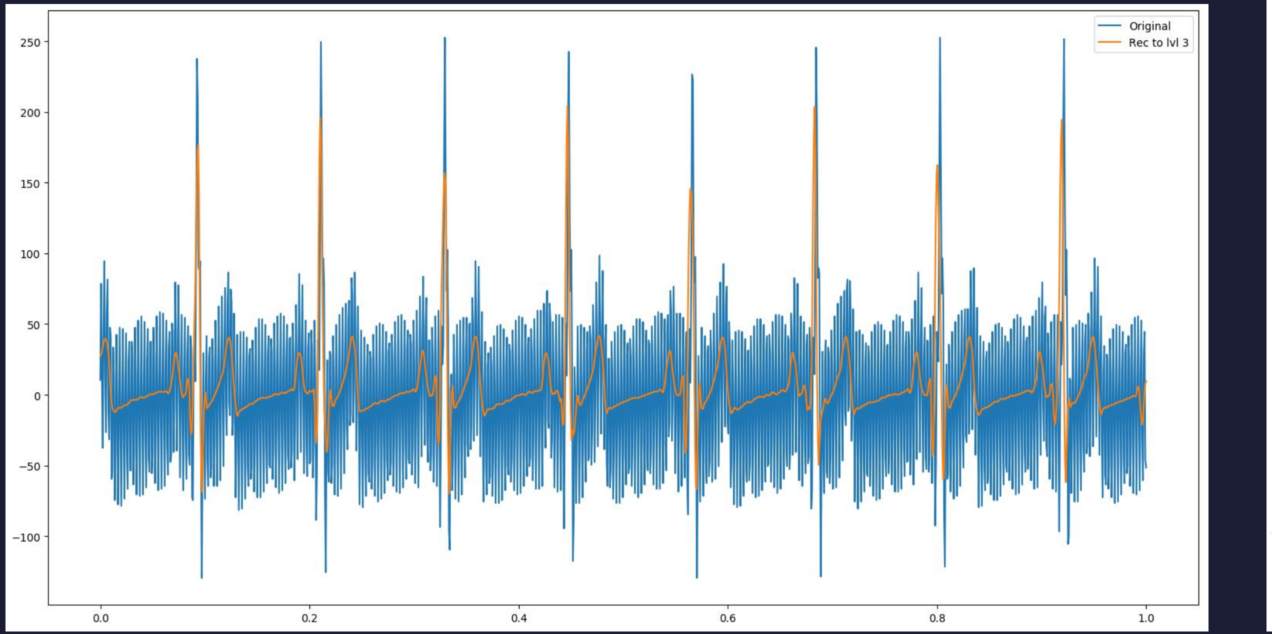
- We are using level 5 reconstruction as Filtered Signal

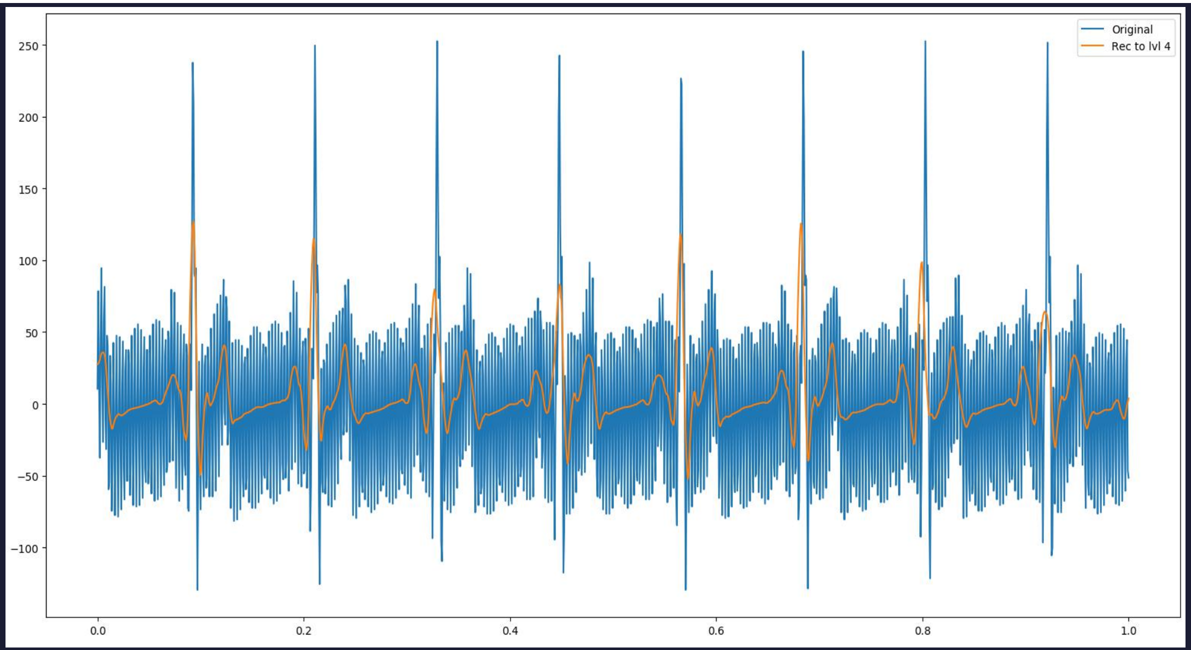


Level Reconstruction of Signal.

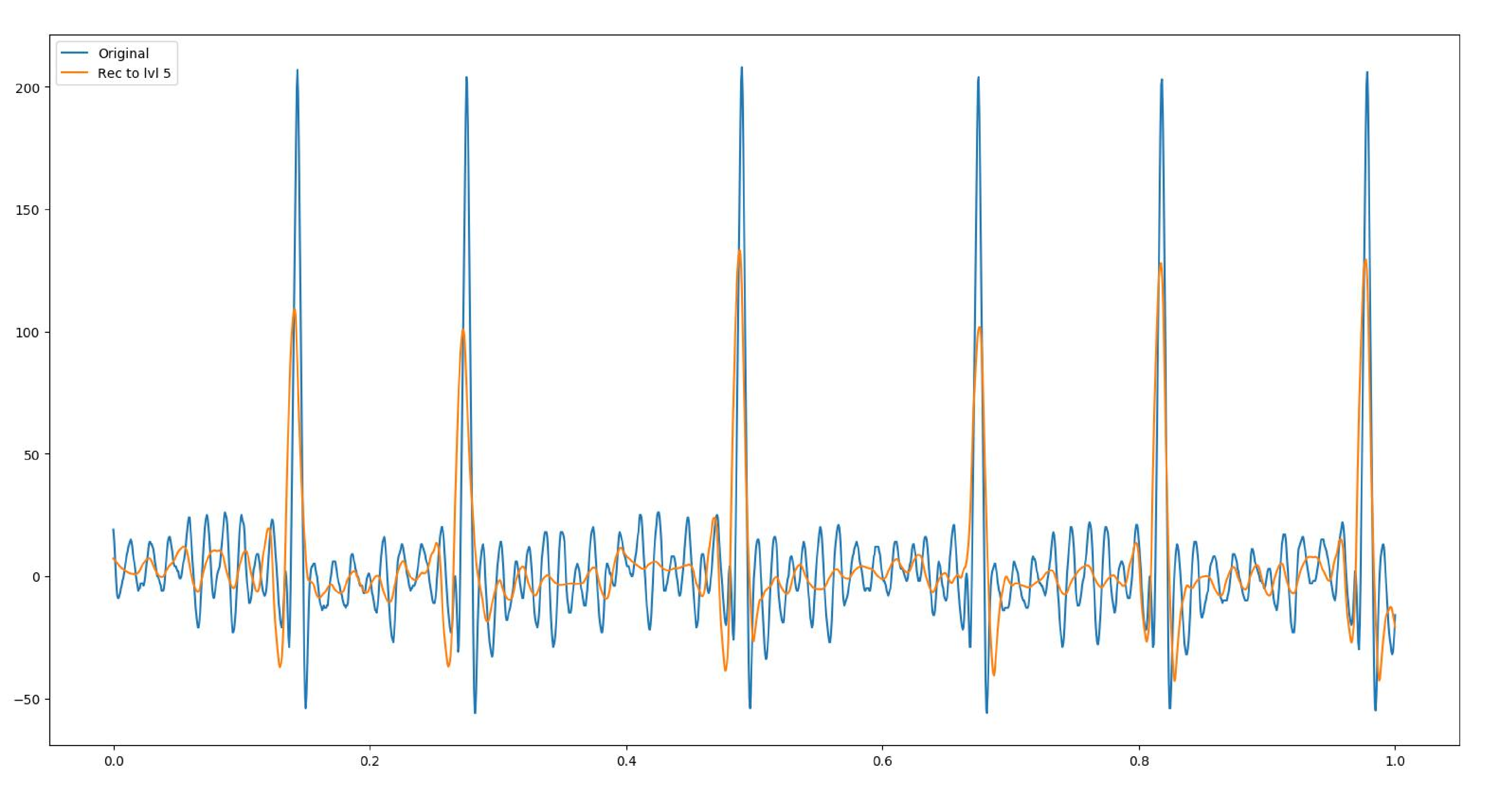




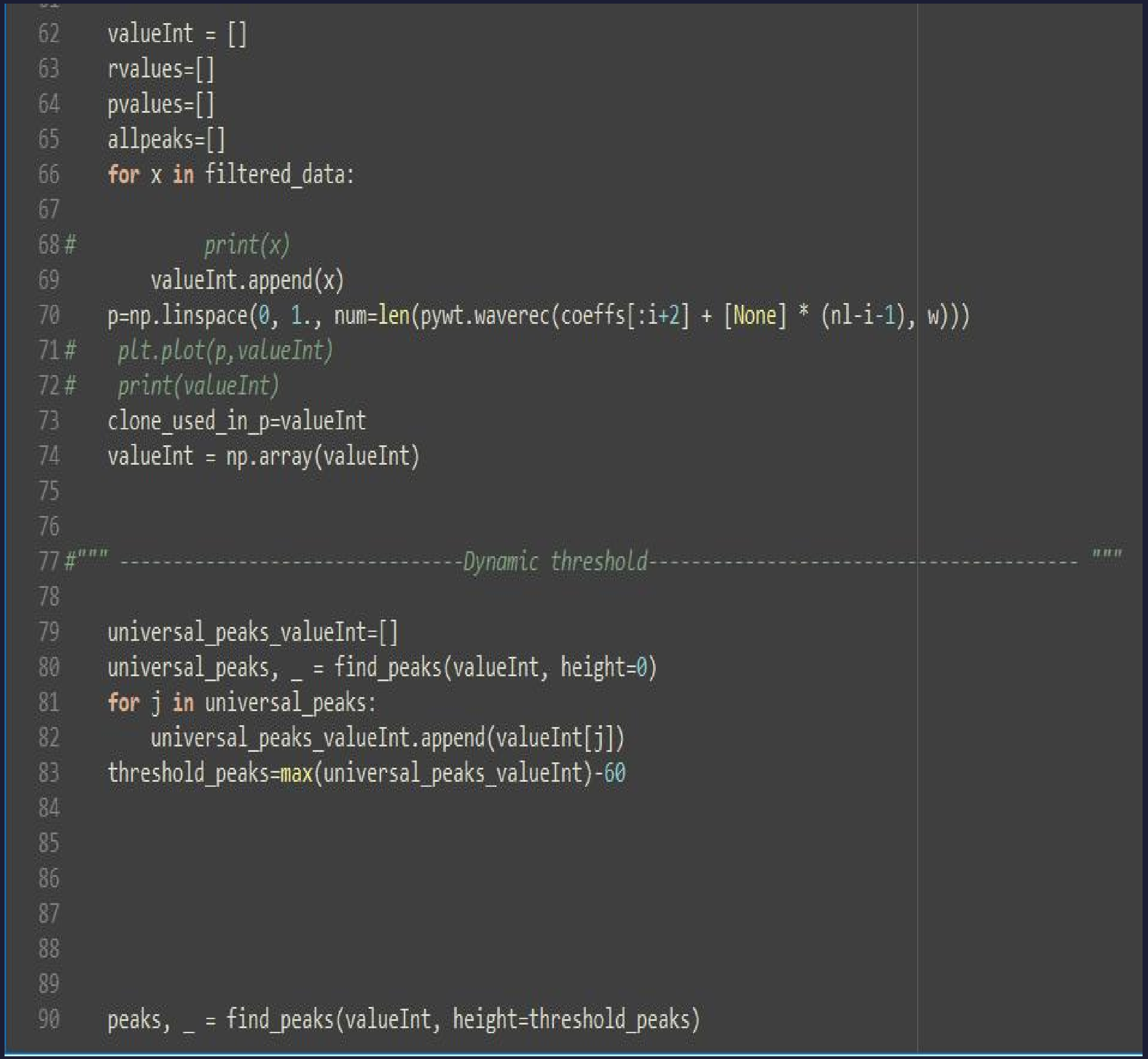




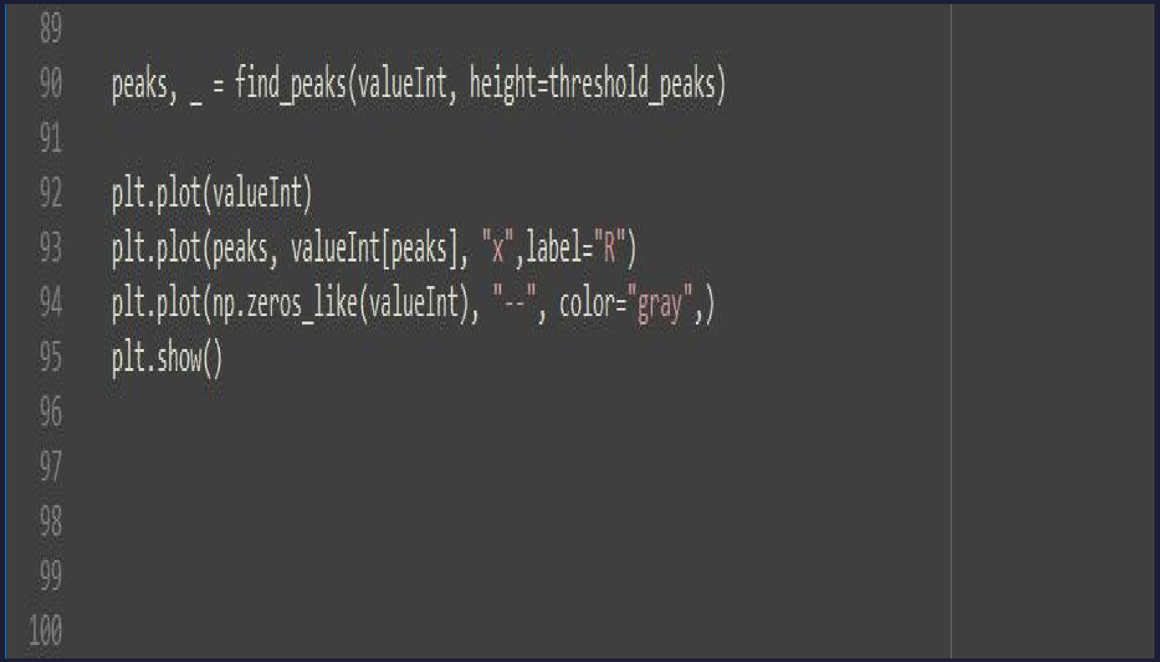
Noisy Signal vs Level 5 Reconstruction by DWT.



4. Recognition of Resonable R points

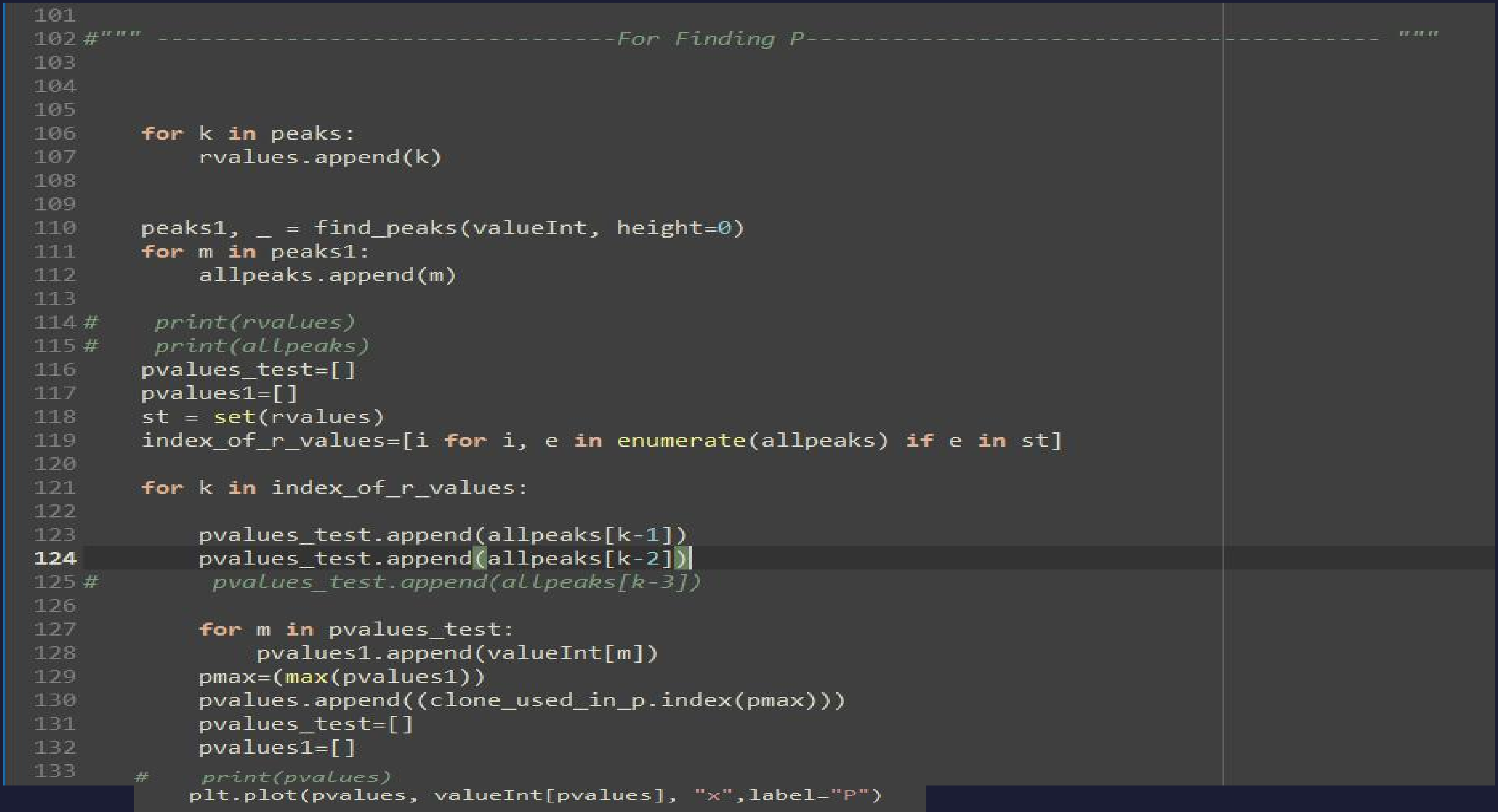


```python
    valueInt = []
    rvalues=[]
    pvalues=[]
    allpeaks=[]
    for x in filtered_data:

#            print(x)
        valueInt.append(x)
    p=np.linspace(0, 1., num=len(pywt.waverec(coeffs[:i+2] + [None] * (nl-i-1), w)))
#    plt.plot(p,valueInt)
#    print(valueInt)
    clone_used_in_p=valueInt
    valueInt = np.array(valueInt)


#"""  -------------------------------Dynamic threshold---------------------------------------  """

    universal_peaks_valueInt=[]
    universal_peaks, _ = find_peaks(valueInt, height=0)
    for j in universal_peaks:
        universal_peaks_valueInt.append(valueInt[j])
    threshold_peaks=max(universal_peaks_valueInt)-60






    peaks, _ = find_peaks(valueInt, height=threshold_peaks)
```

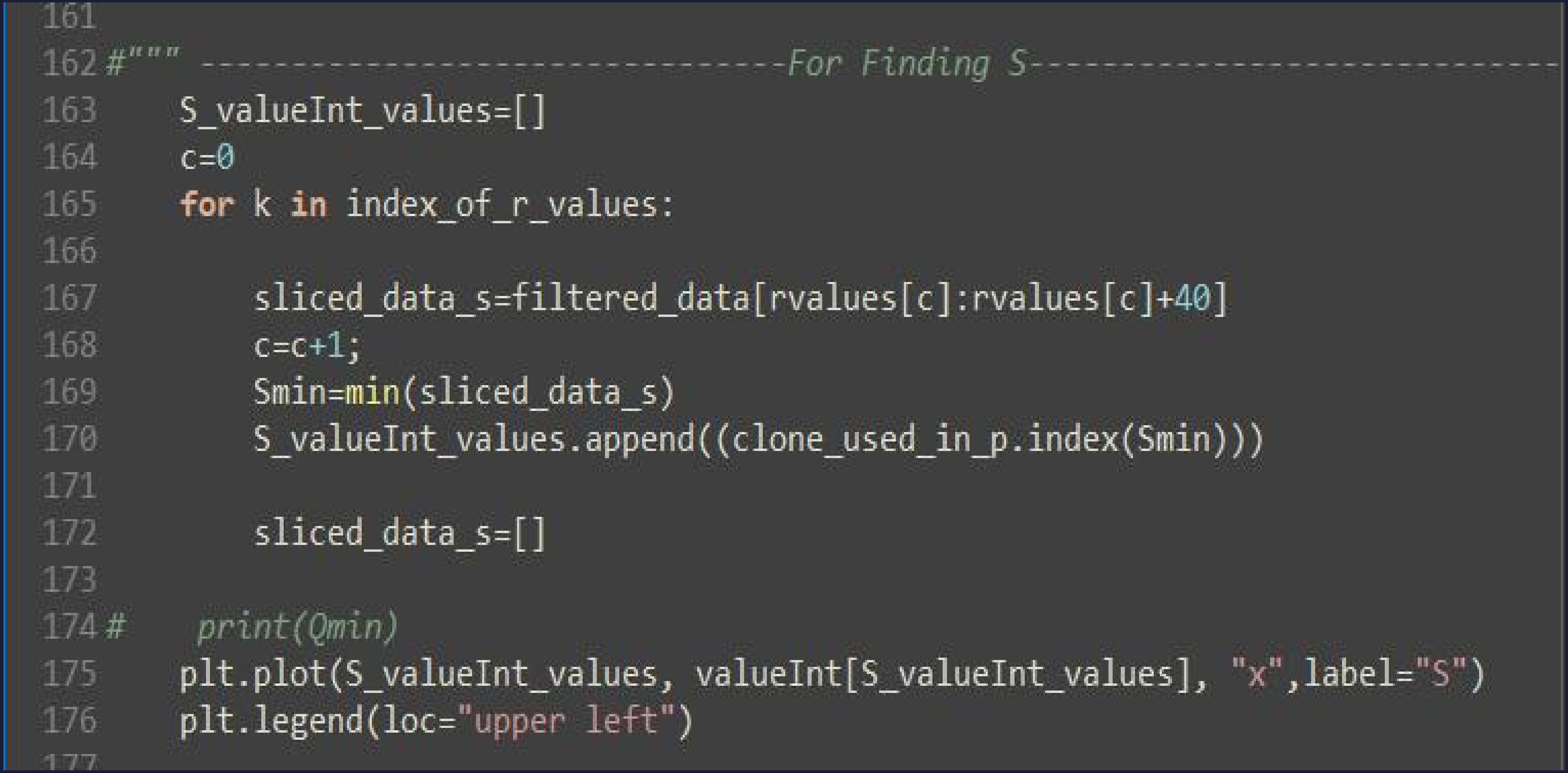


```python

    peaks, _ = find_peaks(valueInt, height=threshold_peaks)

    plt.plot(valueInt)
    plt.plot(peaks, valueInt[peaks], "x",label="R")
    plt.plot(np.zeros_like(valueInt), "--", color="gray",)
    plt.show()
```

5. Using Detected R points as Reference for Detecting PQS .

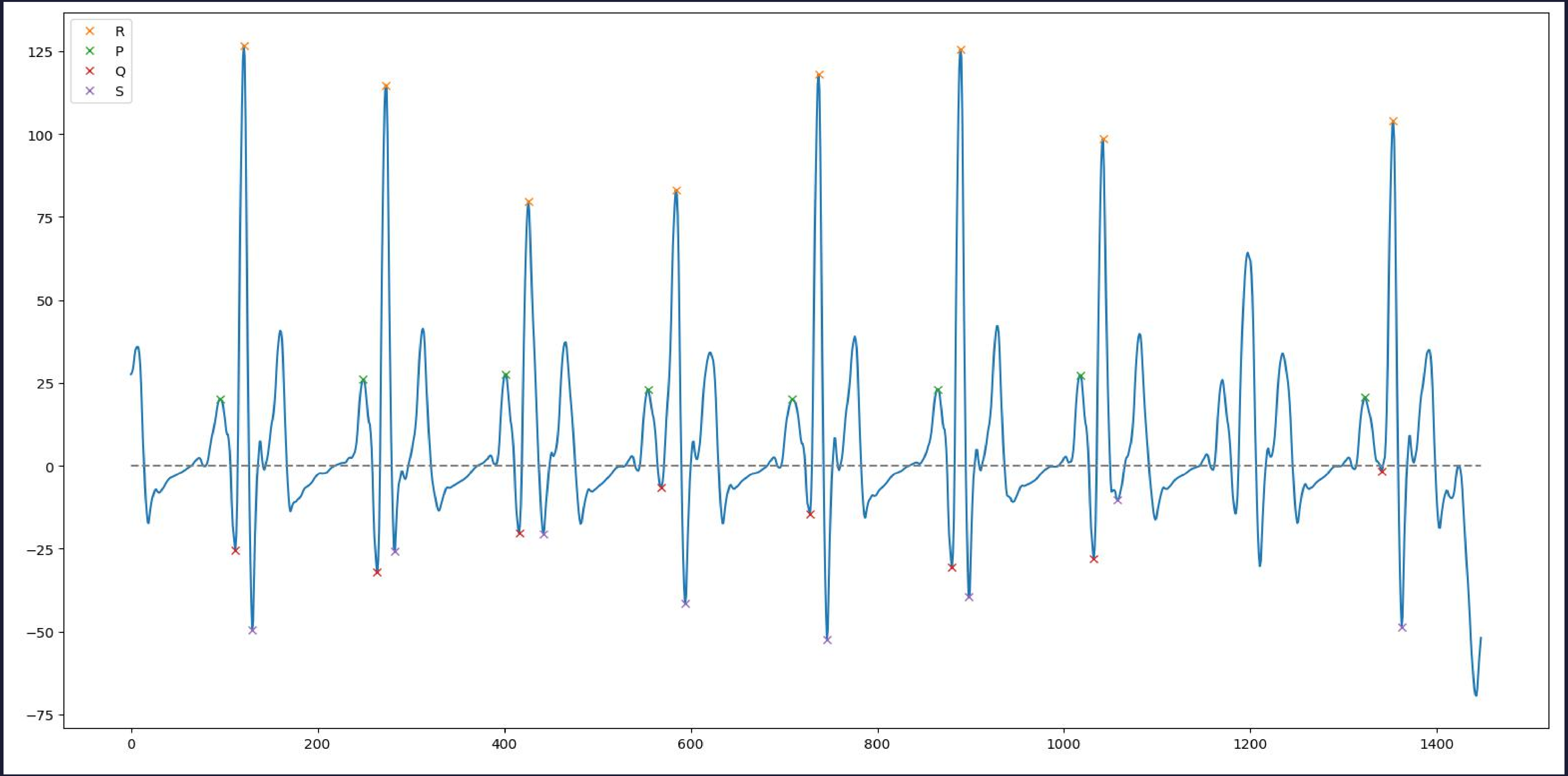


```
#""" --------------------------------For Finding P---------------------------------------- """



    for k in peaks:
        rvalues.append(k)


    peaks1, _ = find_peaks(valueInt, height=0)
    for m in peaks1:
        allpeaks.append(m)

#    print(rvalues)
#    print(allpeaks)
    pvalues_test=[]
    pvalues1=[]
    st = set(rvalues)
    index_of_r_values=[i for i, e in enumerate(allpeaks) if e in st]

    for k in index_of_r_values:

        pvalues_test.append(allpeaks[k-1])
        pvalues_test.append(allpeaks[k-2])
#        pvalues_test.append(allpeaks[k-3])

        for m in pvalues_test:
            pvalues1.append(valueInt[m])
        pmax=(max(pvalues1))
        pvalues.append((clone_used_in_p.index(pmax)))
        pvalues_test=[]
        pvalues1=[]
    #    print(pvalues)
        plt.plot(pvalues, valueInt[pvalues], "x",label="P")
```



```
#""" --------------------------------For Finding S-----------------------------
    S_valueInt_values=[]
    c=0
    for k in index_of_r_values:

        sliced_data_s=filtered_data[rvalues[c]:rvalues[c]+40]
        c=c+1;
        Smin=min(sliced_data_s)
        S_valueInt_values.append((clone_used_in_p.index(Smin)))

        sliced_data_s=[]

#    print(Qmin)
    plt.plot(S_valueInt_values, valueInt[S_valueInt_values], "x",label="S")
    plt.legend(loc="upper left")
```

Algorithm Processed Signal.



Cons of the Algorithm.

- The only Con so far in this Algorithm is setting Dynamic threshold for Detecting R peak.