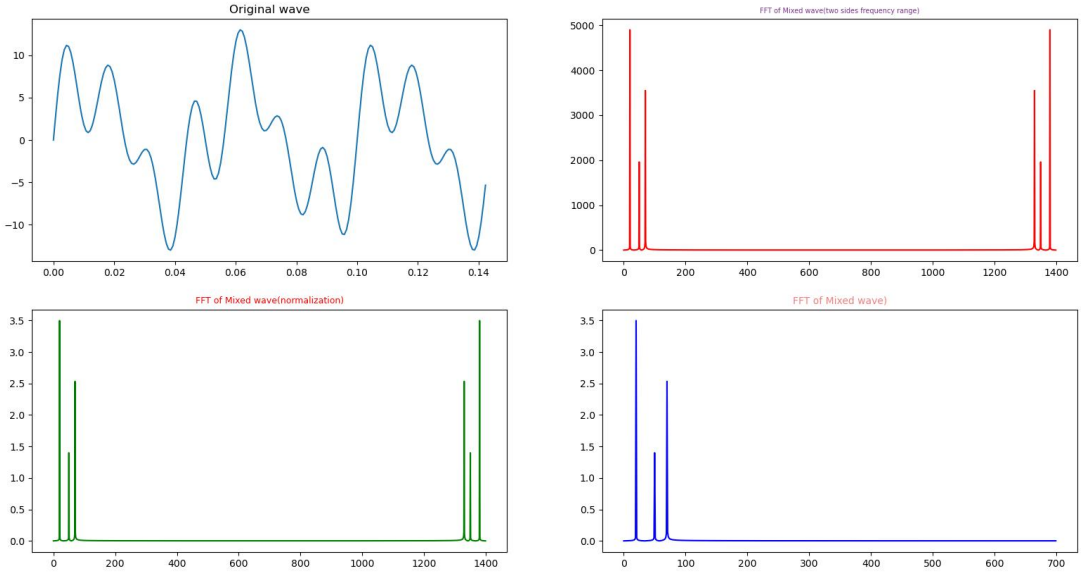
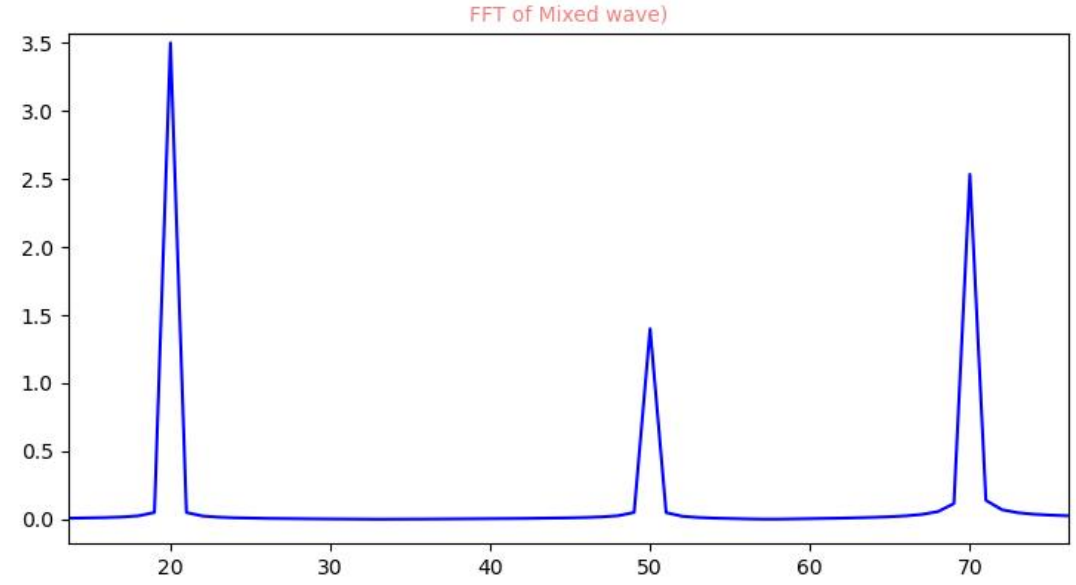
对一段时间序列做了傅里叶变换后得到了频谱图，我们可以将能量的主要部分给分离出来，但此时得到的主要部分所对应的频率可能会比较高，对此考虑做滑窗嵌入，很有可能会出现维数很高的情况。

什么是时间域？
自变量是时间,即横轴是时间,纵轴是信号的变化。其动态信号 x（t）是描述信号在不同时刻取值的函数。
是什么频率域？
自变量是频率,即横轴是频率,纵轴是该频率信号的幅度,也就是通常说的频谱图。

快速傅里叶变换 FFT

$$y = 7\sin 2\pi * 20 * x + 2.8\sin 2\pi * 50 * x + 5.1\sin 2\pi * 70 * x$$

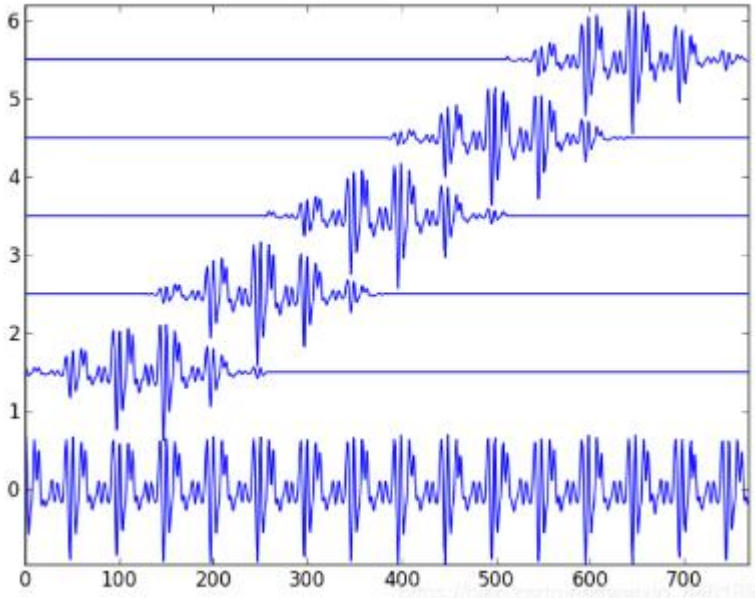




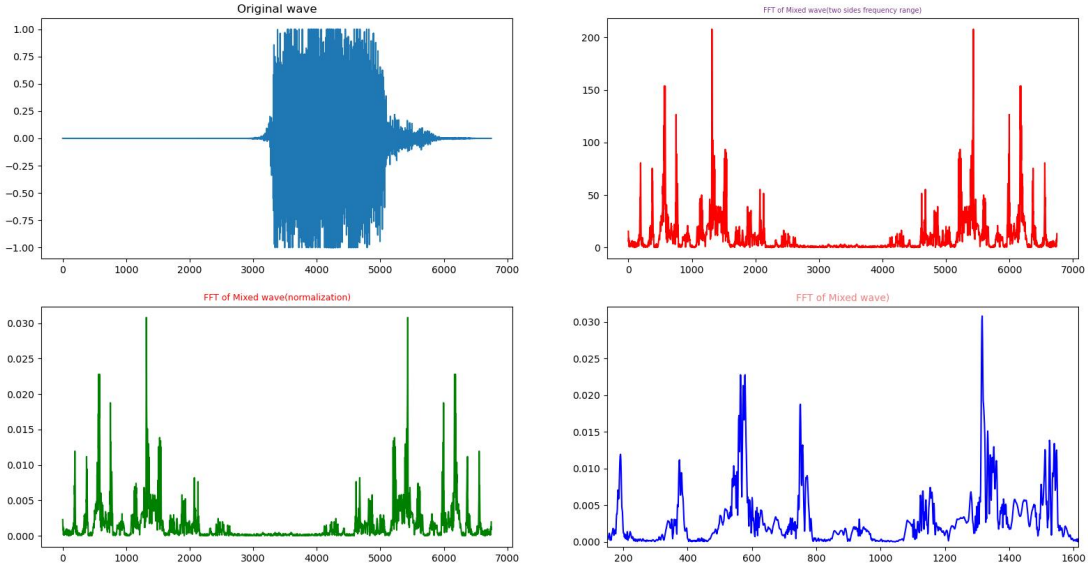
Nyquist–Shannon sampling theorem

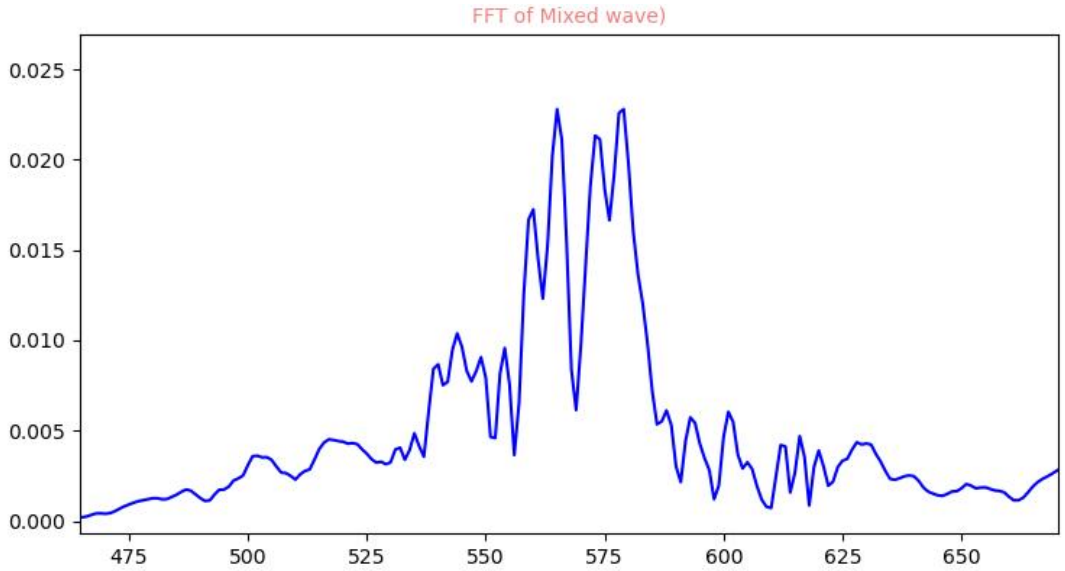
定理内容 ：如果一个系统以超过信号最高频率至少两倍的速率对模拟信号进行均匀采样，那么原始模拟信号就能从采样产生的离散值中完全恢复。

短时傅里叶变换

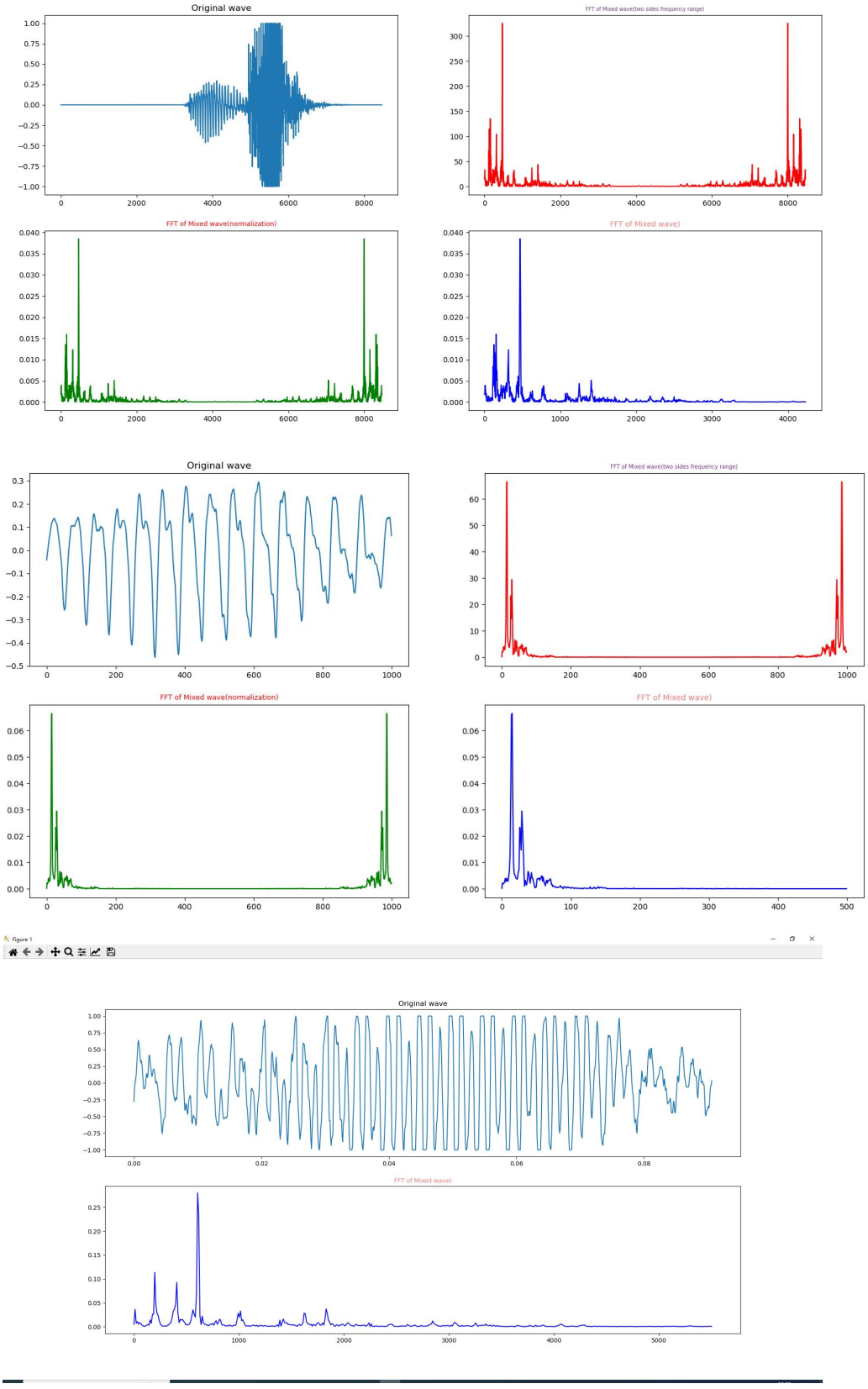


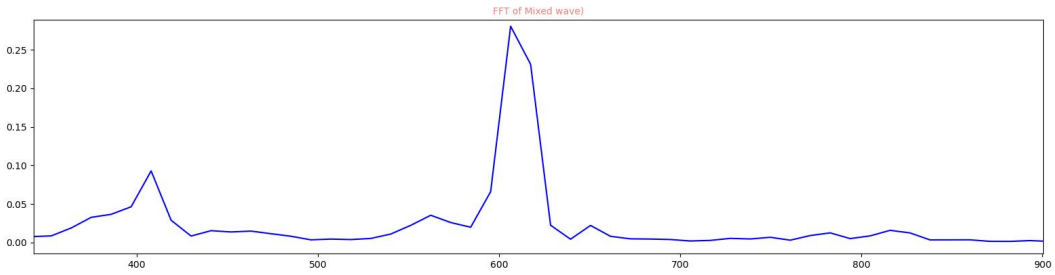
[æ]


Original wave
FFT of Mixed wave(two sides frequency range)
FFT of Mixed wave(normalization)
FFT of Mixed wave)


FFT of Mixed wave)

[b]


Original wave
FFT of Mixed wave(two sides frequency range)
FFT of Mixed wave(normalization)
FFT of Mixed wave)
Original wave
FFT of Mixed wave(two sides frequency range)
FFT of Mixed wave(normalization)
FFT of Mixed wave)
Figure 1
Original wave
FFT of Mixed wave)


FFT of Mixed wave)
0.25
0.20
0.15
0.10
0.05
0.00
400
500
600
700
800
900

[f]

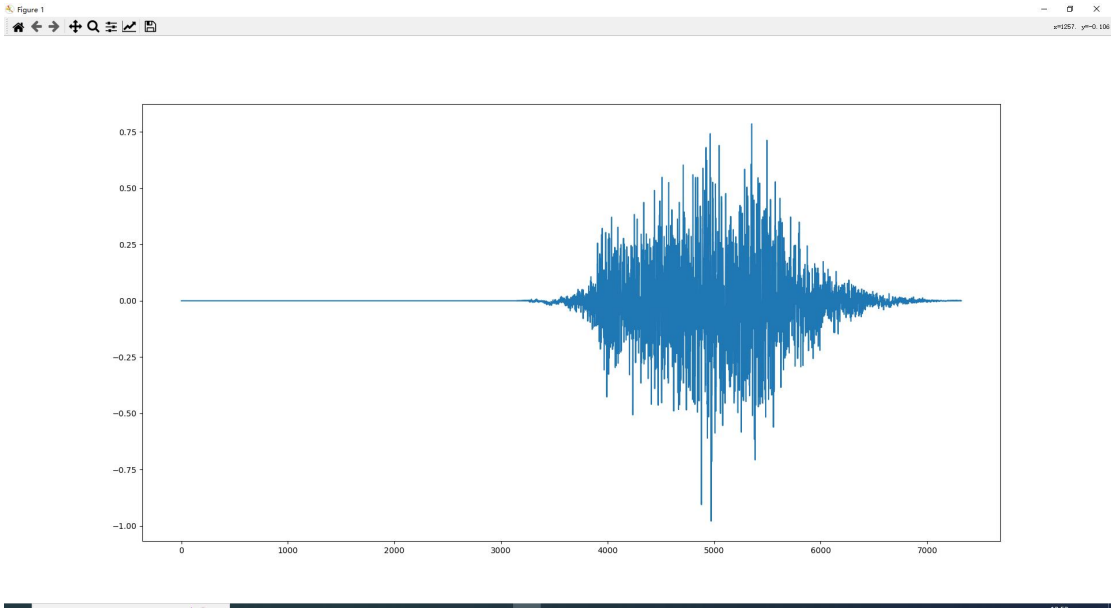


Figure 1
x=1257, y=-0.106
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
1000
2000
3000
4000
5000
6000
7000

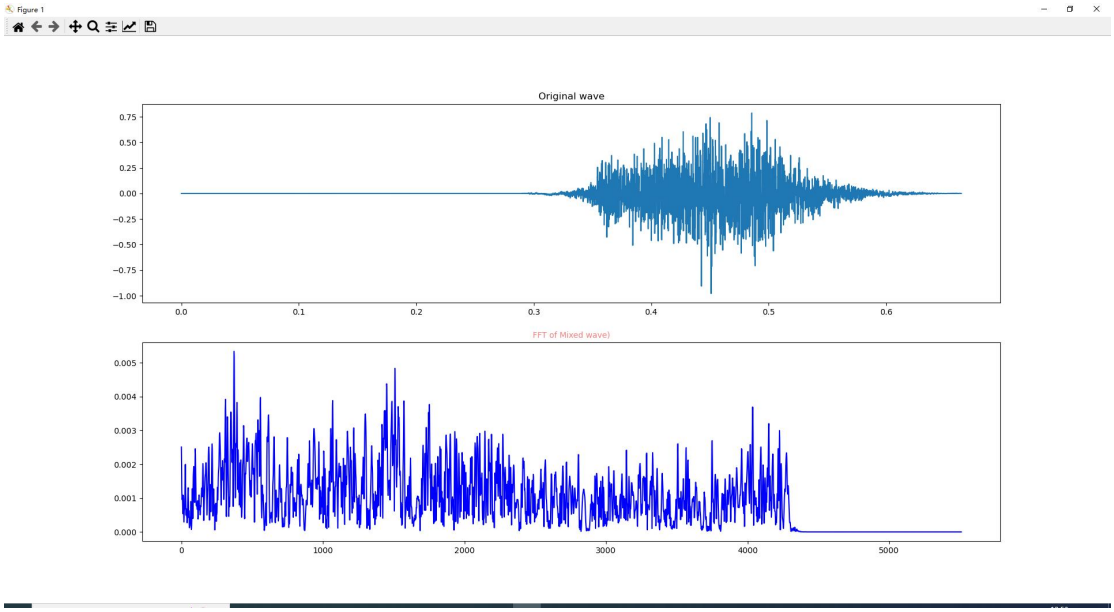


Figure 1
Original wave
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.1
0.2
0.3
0.4
0.5
0.6
FFT of Mixed wave)
0.005
0.004
0.003
0.002
0.001
0.000
0
1000
2000
3000
4000
5000

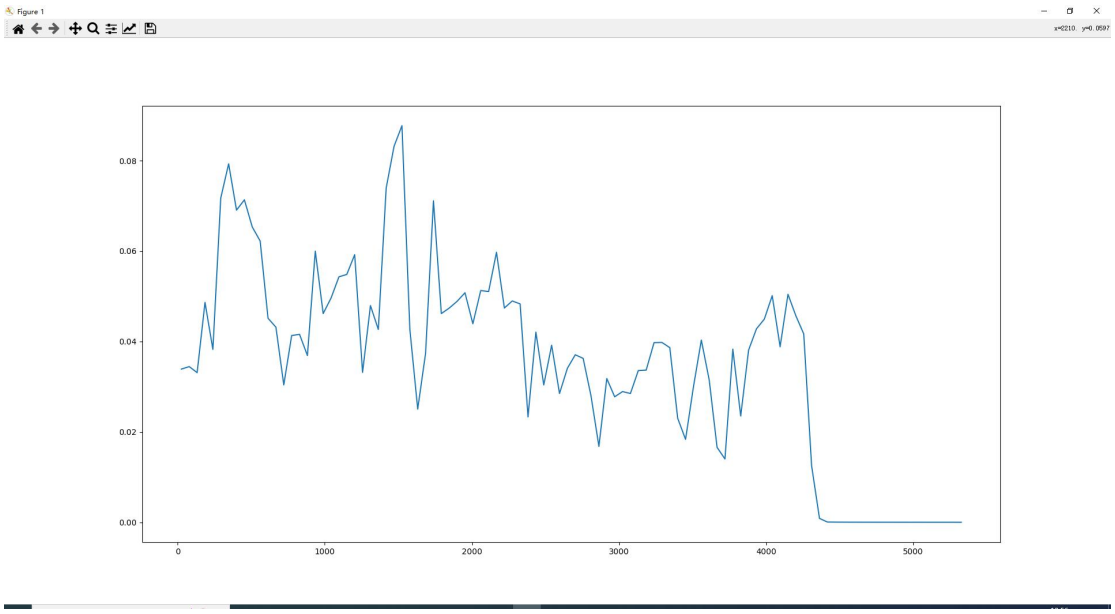


Figure 1
0.08
0.06
0.04
0.02
0.00
0
1000
2000
3000
4000
5000

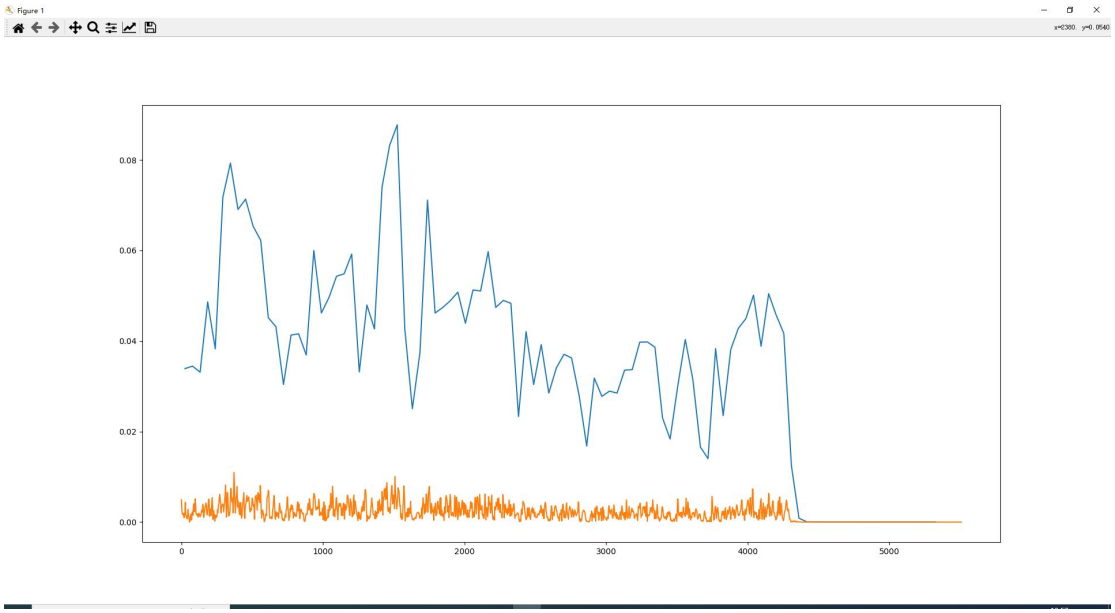


Figure 1
0.08
0.06
0.04
0.02
0.00
0
1000
2000
3000
4000
5000

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jul 10 19:25:12 2022

@author: h'h
"""

import gudhi
import matplotlib.pyplot as plt
import numpy as np

def f_1():

    Points=[[1, 1], [7, 0], [4, 6], [9, 6], [0, 14], [2, 19], [9, 17]]

    x=np.array(Points)[:,0]
    y=np.array(Points)[:,1]

    f=plt.figure()
    plt.scatter(x,y)

    alpha_complex = gudhi.AlphaComplex(points=Points)

    simplex_tree = alpha_complex.create_simplex_tree()

    diag = simplex_tree.persistence(min_persistence=0.01)

    gudhi.plot_persistence_barcode(diag,legend=True)
    gudhi.plot_persistence_diagram(diag,legend=True)
    plt.show()

    return diag

    [(1, (30.25, 37.244897959183675)),
     (1, (10.0, 12.99586776859504)),
     (1, (11.25, 12.5)),
     (0, (0.0, inf)),
     (0, (0.0, 20.0)),
     (0, (0.0, 13.25)),
     (0, (0.0, 9.25)),
     (0, (0.0, 8.5)),
     (0, (0.0, 7.25)),
     (0, (0.0, 6.25))]
```

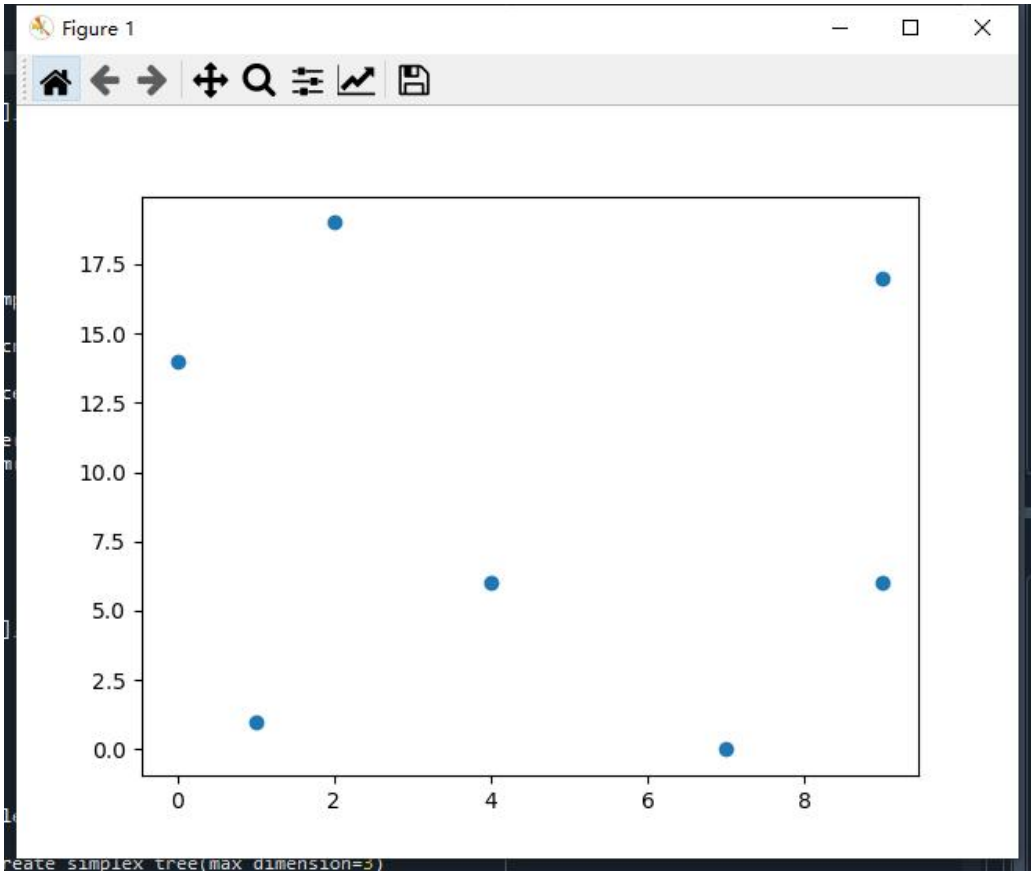


Figure 1
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
2
4
6
8

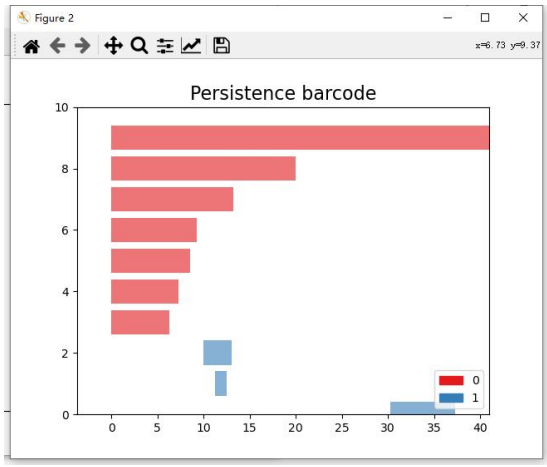


Figure 2
Persistence barcode
0
1

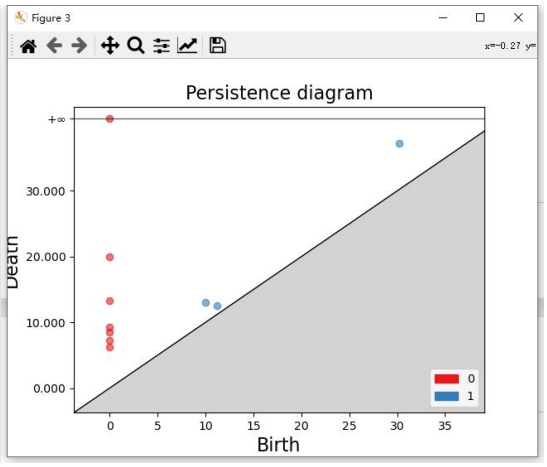


Figure 3
Persistence diagram
Death
Birth
+∞
30.000
20.000
10.000
0.000
0
1

```
def f_2():

    Points=[[1, 1], [7, 0], [4, 6], [9, 6], [0, 14], [2, 19], [9, 17]]

    x=np.array(Points)[:,0]
    y=np.array(Points)[:,1]

    f=plt.figure()
    plt.scatter(x,y)

    rips_complex = gudhi.RipsComplex(points=Points, max_edge_length=12.0)

    simplex_tree = rips_complex.create_simplex_tree(max_dimension=3)

    diag = simplex_tree.persistence(min_persistence=0.01)

    gudhi.plot_persistence_barcode(diag,legend=True)
    gudhi.plot_persistence_diagram(diag,legend=True)

    return diag
```

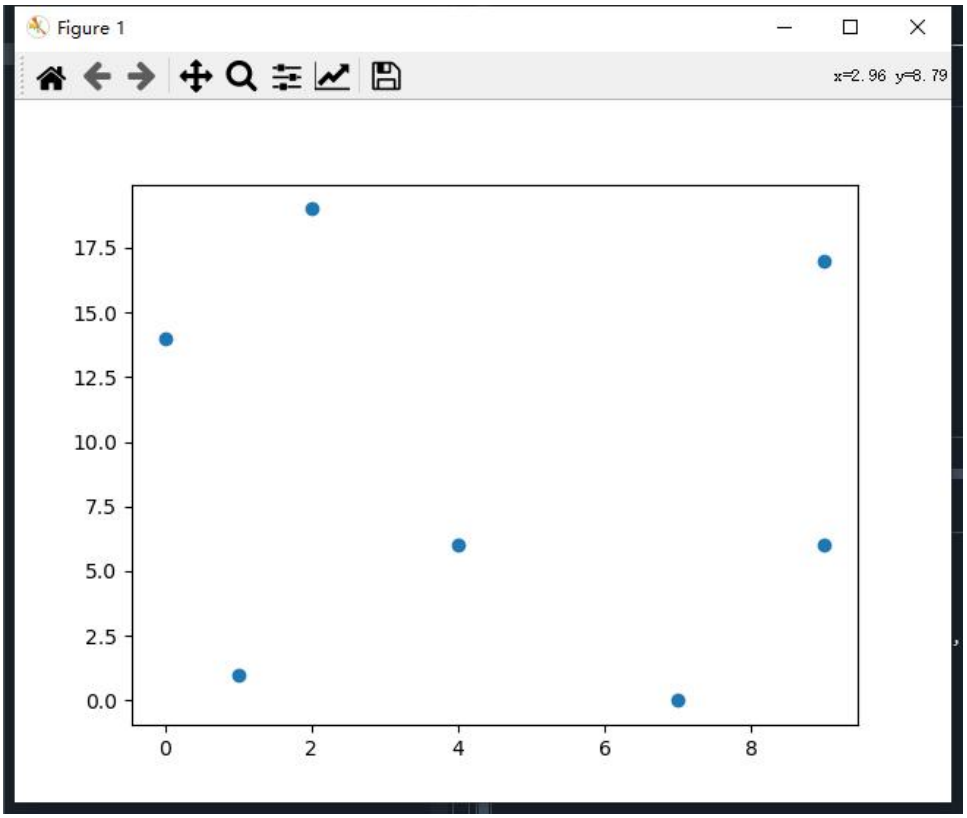


Figure 1
x=2.96 y=8.79
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
2
4
6
8

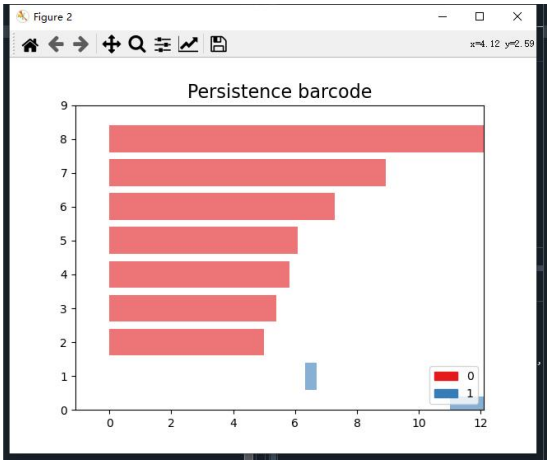


Figure 2
Persistence barcode
0
1
0
2
4
6
8
10
12

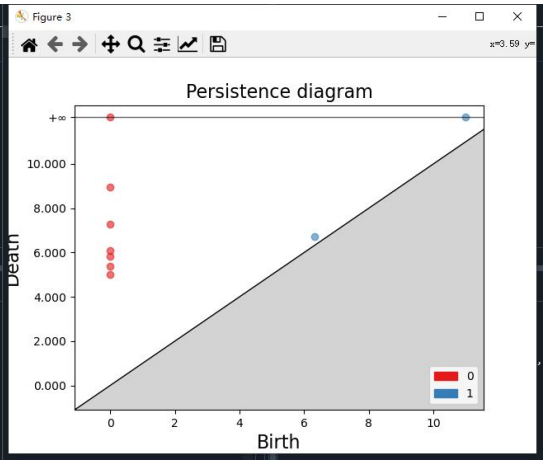


Figure 3
Persistence diagram
Death
Birth
+∞
10.000
8.000
6.000
4.000
2.000
0.000
0
1
0
2
4
6
8
10

```
def f_3():

    rips_complex = gudhi.RipsComplex(distance_matrix=[[],
                                                      [6.0827625303],
                                                      [5.8309518948, 6.7082039325],
                                                      [9.4339811321, 6.3245553203, 5],
                                                      [13.0384048104,    15.6524758425,
8.94427191, 12.0415945788],
                                                      [18.0277563773,    19.6468827044,
13.152946438, 14.7648230602, 5.3851648071],
                                                      [17.88854382,      17.1172427686,
12.0830459736, 11, 9.4868329805, 7.2801098893]],
                                             max_edge_length=12.0)

    simplex_tree = rips_complex.create_simplex_tree(max_dimension=1)

    diag = simplex_tree.persistence(min_persistence=0.01)

    gudhi.plot_persistence_barcode(diag,legend=True)
    gudhi.plot_persistence_diagram(diag,legend=True)

    return diag
```