N9030A_Ez_Analysis

July 5, 2019

In [1]:
```
#-*- coding: utf-8 -*-
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('qt5agg')
#
matplotlib.rcParams['font.sans-serif'] = ['SimHei']
matplotlib.rcParams['font.family']='sans-serif'
#'-'
matplotlib.rcParams['axes.unicode_minus'] = False

#%matplotlib inlinenotebook
%matplotlib
#%matplotlib
```

```
Using matplotlib backend: Qt5Agg
```

In [2]:
```
matplotlib.rcParams['figure.figsize'] = (12.0, 10.0)
```

In [3]:
```
data = pd.read_csv('N9030A_Ez20190429.csv', header=None)
```

In [4]:
```
data.head(8)
```

Out[4]:

	0	1	2	3	4	5	6	\
0	-44.20758	-112.6993	-91.54488	-113.3325	-90.98305	-120.3877	-91.56597	
1	-44.62451	-117.4041	-90.53913	-107.9832	-89.63585	-110.9886	-90.97483	
2	-44.47005	-113.8537	-90.76211	-118.0489	-90.69619	-114.0361	-92.66928	
3	-44.49908	-113.8421	-92.00131	-118.8630	-91.37560	-112.7318	-90.68216	
4	-44.11002	-113.4879	-88.58568	-112.6694	-88.93554	-116.2102	-91.52532	
5	-44.04847	-115.4088	-91.62850	-119.0105	-91.13636	-107.1619	-91.16646	
6	-44.47005	-113.9827	-88.45793	-116.2000	-90.95867	-112.7667	-91.38944	
7	-44.43056	-111.0734	-89.18496	-116.2696	-91.14913	-117.2647	-91.23382	

	7	8	9	...	991	992	993	994	\
0	-111.0566	-91.68183	-110.6295	...	-111.1460	-90.16299	-110.9321	-87.46660	
1	-117.0098	-90.02221	-110.9396	...	-113.7242	-87.68460	-115.9969	-89.08324	

```
2 -121.0340 -91.24515 -115.1252  ... -110.6872 -89.29428 -112.6591 -85.79537
3 -111.4537 -90.05473 -110.5180  ... -107.4272 -86.65446 -112.3606 -86.50614
4 -115.1423 -91.40774 -110.4646  ... -112.6326 -86.79382 -110.4165 -86.99159
5 -114.8264 -91.57150 -110.7375  ... -115.0006 -87.87623 -114.4221 -86.04623
6 -112.0089 -90.66446 -116.5235  ... -118.2792 -86.18178 -112.5987 -87.34117
7 -109.6873 -89.77832 -113.8500  ... -111.9044 -87.58589 -112.8461 -87.30168

       995       996       997       998       999       1000
0 -112.9544 -89.35766 -111.1198 -86.79134 -105.6732 -89.76597
1 -116.6825 -86.46582 -113.4820 -88.87138 -114.6832 -88.10984
2 -115.4642 -88.43785 -111.8108 -88.68091 -112.7972 -86.34222
3 -110.5666 -88.44365 -112.5251 -88.87138 -110.2479 -86.82303
4 -111.8488 -86.56105 -113.6644 -87.21060 -110.4314 -87.52683
5 -108.3716 -87.63069 -111.7272 -87.24893 -110.2224 -87.30965
6 -113.7744 -88.72355 -118.2065 -88.03054 -112.5881 -89.23638
7 -108.8002 -86.57731 -113.2741 -89.80397 -110.6323 -89.55112

[8 rows x 1001 columns]
```

In [5]:
```
len(data.columns)
```

Out[5]:
```
1001
```

In [6]:
```
cols = []
for i in data.columns:
    cols.append('f' + str(i+1))
data.columns = cols
```

In [7]:
```
data.head()
```

Out[7]:
```
         f1        f2        f3        f4        f5        f6        f7  \
0 -44.20758 -112.6993 -91.54488 -113.3325 -90.98305 -120.3877 -91.56597
1 -44.62451 -117.4041 -90.53913 -107.9832 -89.63585 -110.9886 -90.97483
2 -44.47005 -113.8537 -90.76211 -118.0489 -90.69619 -114.0361 -92.66928
3 -44.49908 -113.8421 -92.00131 -118.8630 -91.37560 -112.7318 -90.68216
4 -44.11002 -113.4879 -88.58568 -112.6694 -88.93554 -116.2102 -91.52532

         f8        f9       f10  ...      f992      f993      f994      f995  \
0 -111.0566 -91.68183 -110.6295  ... -111.1460 -90.16299 -110.9321 -87.46660
1 -117.0098 -90.02221 -110.9396  ... -113.7242 -87.68460 -115.9969 -89.08324
2 -121.0340 -91.24515 -115.1252  ... -110.6872 -89.29428 -112.6591 -85.79537
3 -111.4537 -90.05473 -110.5180  ... -107.4272 -86.65446 -112.3606 -86.50614
4 -115.1423 -91.40774 -110.4646  ... -112.6326 -86.79382 -110.4165 -86.99159

       f996      f997      f998      f999     f1000     f1001
0 -112.9544 -89.35766 -111.1198 -86.79134 -105.6732 -89.76597
1 -116.6825 -86.46582 -113.4820 -88.87138 -114.6832 -88.10984
2 -115.4642 -88.43785 -111.8108 -88.68091 -112.7972 -86.34222
3 -110.5666 -88.44365 -112.5251 -88.87138 -110.2479 -86.82303
```

```
4 -111.8488 -86.56105 -113.6644 -87.21060 -110.4314 -87.52683

[5 rows x 1001 columns]
```

```
In [9]: df = np.arange(1,101).reshape(20,5)
        df
```

```
Out[9]: array([[  1,   2,   3,   4,   5],
               [  6,   7,   8,   9,  10],
               [ 11,  12,  13,  14,  15],
               [ 16,  17,  18,  19,  20],
               [ 21,  22,  23,  24,  25],
               [ 26,  27,  28,  29,  30],
               [ 31,  32,  33,  34,  35],
               [ 36,  37,  38,  39,  40],
               [ 41,  42,  43,  44,  45],
               [ 46,  47,  48,  49,  50],
               [ 51,  52,  53,  54,  55],
               [ 56,  57,  58,  59,  60],
               [ 61,  62,  63,  64,  65],
               [ 66,  67,  68,  69,  70],
               [ 71,  72,  73,  74,  75],
               [ 76,  77,  78,  79,  80],
               [ 81,  82,  83,  84,  85],
               [ 86,  87,  88,  89,  90],
               [ 91,  92,  93,  94,  95],
               [ 96,  97,  98,  99, 100]])
```

```
In [131]: df = pd.DataFrame(df)
          df
```

Out[131]:

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25
5	26	27	28	29	30
6	31	32	33	34	35
7	36	37	38	39	40
8	41	42	43	44	45
9	46	47	48	49	50
10	51	52	53	54	55
11	56	57	58	59	60
12	61	62	63	64	65
13	66	67	68	69	70
14	71	72	73	74	75
15	76	77	78	79	80
16	81	82	83	84	85

```
          17  86  87  88  89   90
          18  91  92  93  94   95
          19  96  97  98  99  100
```

```
In [38]: df.iloc[:10,] = df.iloc[:10,][::-1].values
```

```
In [132]: df[0:10].values
```

```
Out[132]: array([[ 1,  2,  3,  4,  5],
                 [ 6,  7,  8,  9, 10],
                 [11, 12, 13, 14, 15],
                 [16, 17, 18, 19, 20],
                 [21, 22, 23, 24, 25],
                 [26, 27, 28, 29, 30],
                 [31, 32, 33, 34, 35],
                 [36, 37, 38, 39, 40],
                 [41, 42, 43, 44, 45],
                 [46, 47, 48, 49, 50]])
```

```
In [133]: df
```

```
Out[133]:      0   1   2   3    4
          0    1   2   3   4    5
          1    6   7   8   9   10
          2   11  12  13  14   15
          3   16  17  18  19   20
          4   21  22  23  24   25
          5   26  27  28  29   30
          6   31  32  33  34   35
          7   36  37  38  39   40
          8   41  42  43  44   45
          9   46  47  48  49   50
          10  51  52  53  54   55
          11  56  57  58  59   60
          12  61  62  63  64   65
          13  66  67  68  69   70
          14  71  72  73  74   75
          15  76  77  78  79   80
          16  81  82  83  84   85
          17  86  87  88  89   90
          18  91  92  93  94   95
          19  96  97  98  99  100
```

```
In [134]: arr = df.values
```

```
In [44]: type(arr)
```

```
Out[44]: numpy.ndarray
```

```
In [46]: np.vsplit?
```

```
In [135]: a = np.vsplit(arr,4)
          a
```

```
Out[135]: [array([[ 1,  2,  3,  4,  5],
                  [ 6,  7,  8,  9, 10],
                  [11, 12, 13, 14, 15],
                  [16, 17, 18, 19, 20],
                  [21, 22, 23, 24, 25]]), array([[26, 27, 28, 29, 30],
                  [31, 32, 33, 34, 35],
                  [36, 37, 38, 39, 40],
                  [41, 42, 43, 44, 45],
                  [46, 47, 48, 49, 50]]), array([[51, 52, 53, 54, 55],
                  [56, 57, 58, 59, 60],
                  [61, 62, 63, 64, 65],
                  [66, 67, 68, 69, 70],
                  [71, 72, 73, 74, 75]]), array([[ 76,  77,  78,  79,  80],
                  [ 81,  82,  83,  84,  85],
                  [ 86,  87,  88,  89,  90],
                  [ 91,  92,  93,  94,  95],
                  [ 96,  97,  98,  99, 100]])]
```

```
In [138]: a_list = []
          i = 0
          for it in a:
              if i % 2 == 0:
                  a_list.append(it[::-1,])
              else:
                  a_list.append(it)
              i = i + 1
```

```
In [139]: a_list
```

```
Out[139]: [array([[21, 22, 23, 24, 25],
                  [16, 17, 18, 19, 20],
                  [11, 12, 13, 14, 15],
                  [ 6,  7,  8,  9, 10],
                  [ 1,  2,  3,  4,  5]]), array([[26, 27, 28, 29, 30],
                  [31, 32, 33, 34, 35],
                  [36, 37, 38, 39, 40],
                  [41, 42, 43, 44, 45],
                  [46, 47, 48, 49, 50]]), array([[71, 72, 73, 74, 75],
                  [66, 67, 68, 69, 70],
                  [61, 62, 63, 64, 65],
                  [56, 57, 58, 59, 60],
                  [51, 52, 53, 54, 55]]), array([[ 76,  77,  78,  79,  80],
                  [ 81,  82,  83,  84,  85],
                  [ 86,  87,  88,  89,  90],
                  [ 91,  92,  93,  94,  95],
                  [ 96,  97,  98,  99, 100]])]
```

```
In [143]: i = 0
          if len(a_list) > 0:
              a_shuffled = a_list[0]
              for it in a_list:
                  if i > 0:
                      a_shuffled = np.vstack((a_shuffled,it))
                  i = i +1
```

```
In [144]: a_shuffled
```

```
Out[144]: array([[ 21,  22,  23,  24,  25],
                 [ 16,  17,  18,  19,  20],
                 [ 11,  12,  13,  14,  15],
                 [  6,   7,   8,   9,  10],
                 [  1,   2,   3,   4,   5],
                 [ 26,  27,  28,  29,  30],
                 [ 31,  32,  33,  34,  35],
                 [ 36,  37,  38,  39,  40],
                 [ 41,  42,  43,  44,  45],
                 [ 46,  47,  48,  49,  50],
                 [ 71,  72,  73,  74,  75],
                 [ 66,  67,  68,  69,  70],
                 [ 61,  62,  63,  64,  65],
                 [ 56,  57,  58,  59,  60],
                 [ 51,  52,  53,  54,  55],
                 [ 76,  77,  78,  79,  80],
                 [ 81,  82,  83,  84,  85],
                 [ 86,  87,  88,  89,  90],
                 [ 91,  92,  93,  94,  95],
                 [ 96,  97,  98,  99, 100]])
```

0.0.1

```
In [8]: data_list = np.vsplit(data.values, 160)
```

```
In [9]: len(data_list)
```

```
Out[9]: 160
```

```
In [10]: data_array_list = []
         i = 0
         for it in data_list:
             if i % 2 == 0:
                 data_array_list.append(it[::-1,])
             else:
                 data_array_list.append(it)
             i = i + 1
```

```
In [11]: len(data_array_list)
```

```
Out[11]: 160

In [12]: i = 0
         if len(data_array_list) > 0:
             data_shuffled = data_array_list[0]
             for it in data_array_list:
                 if i > 0:
                     data_shuffled = np.vstack((data_shuffled,it))
                 i = i +1


In [13]: data_shuffled.shape

Out[13]: (25600, 1001)

In [14]: x_col = []
         for i in range(1,161):
             x_col.append([i]*160)

In [15]: x_col = np.squeeze(np.array(x_col))

In [16]: xx = x_col.ravel()

In [17]: xx

Out[17]: array([  1,   1,   1, ..., 160, 160, 160])

In [18]: y_col = [i for i in range(1,161)] * 160

In [19]: yy = np.array(y_col)

In [20]: data_shuffled_df = pd.DataFrame(data_shuffled)

In [21]: data_xx_yy = data_shuffled_df.copy()

In [22]: data_xx_yy.insert(0,'y',yy)

In [23]: data_xx_yy.insert(0,'x',xx)

In [24]: cols = []
         for i in data_shuffled_df.columns:
             cols.append('f' + str(i+1))
         data_shuffled_df.columns = cols

In [26]: data_shuffled_df.to_csv("N9030A_Ez_shuffled.csv", header=None)

In [25]: cols = ['x','y']
         for i in data_xx_yy.columns:
             if i == 'x' or i =='y':
                 continue
             cols.append('f' + str(i+1))
         data_xx_yy.columns = cols
```

In [26]: data_xx_yy

Out[26]:

	x	y	f1	f2	f3	f4	f5	f6
0	1	1	-41.74312	-119.5852	-91.39158	-120.8374	-91.08758	-117.8814
1	1	2	-41.47252	-116.9360	-90.70404	-111.4290	-91.56723	-114.7608
2	1	3	-41.45278	-112.9002	-92.39501	-119.5529	-91.61136	-117.3843
3	1	4	-41.49111	-112.2719	-88.10023	-118.6307	-91.17468	-114.2056
4	1	5	-41.47020	-110.2488	-89.48924	-108.4443	-90.21306	-118.4110
5	1	6	-41.48646	-112.6842	-89.43930	-114.5496	-91.81228	-109.2918
6	1	7	-41.52711	-114.8456	-90.03509	-117.3323	-89.76012	-114.8734
7	1	8	-41.48762	-113.9780	-90.40905	-110.9459	-89.45700	-110.5264
8	1	9	-41.87785	-114.3369	-90.28130	-112.6368	-90.45230	-110.4579
9	1	10	-41.86391	-111.8480	-91.58437	-112.9191	-90.18054	-121.5572
10	1	11	-41.93475	-117.2589	-90.54842	-111.7856	-89.04006	-117.2473
11	1	12	-41.55614	-115.7828	-90.02464	-113.9353	-91.83086	-111.9386
12	1	13	-41.60724	-112.1872	-89.10482	-109.0586	-91.03996	-114.1371
13	1	14	-41.90456	-114.1116	-90.14774	-118.6702	-89.77406	-114.0988
14	1	15	-41.93243	-115.9663	-90.84457	-112.8645	-90.58122	-114.5564
15	1	16	-41.95914	-112.7144	-90.88986	-113.5880	-91.00628	-115.1858
16	1	17	-41.96843	-115.0743	-90.07922	-117.7794	-88.92509	-110.9735
17	1	18	-41.96611	-114.0128	-91.59831	-118.1418	-90.92499	-119.4307
18	1	19	-41.67112	-119.2216	-90.03741	-111.3349	-91.14216	-112.1512
19	1	20	-42.01837	-114.3694	-91.55998	-114.5299	-88.72997	-113.3323
20	1	21	-42.05670	-114.4042	-89.87249	-110.4476	-89.67998	-116.3240
21	1	22	-42.01257	-120.5015	-91.15234	-116.6575	-89.66605	-112.6111
22	1	23	-42.05786	-108.9759	-92.62032	-119.3113	-91.55097	-115.2509
23	1	24	-42.07296	-110.4358	-88.84816	-116.3126	-90.72058	-116.7409
24	1	25	-42.09270	-112.3265	-91.43804	-115.2581	-88.79966	-113.7330
25	1	26	-42.10083	-116.5539	-87.94228	-112.7704	-91.46851	-112.6331
26	1	27	-42.09618	-114.1034	-91.88401	-113.2373	-89.61030	-115.3821
27	1	28	-42.00211	-118.9754	-88.09907	-113.3534	-90.67180	-110.4393
28	1	29	-41.78377	-113.4473	-90.82715	-115.2918	-91.01093	-115.8652
29	1	30	-41.83371	-111.8445	-88.88532	-109.0331	-90.95170	-112.2696
...	...	...	...	...	...	...	...	...
25570	160	131	-38.62482	-111.4764	-92.32649	-120.1928	-90.42908	-113.6923
25571	160	132	-38.64340	-122.0601	-89.77378	-115.5786	-90.72174	-120.2216
25572	160	133	-38.64224	-111.8213	-91.00833	-117.0408	-90.20841	-114.1743
25573	160	134	-38.82922	-115.9140	-89.58563	-113.7471	-90.33616	-115.8606
25574	160	135	-38.61320	-114.5610	-90.76444	-110.1317	-89.09349	-114.6249
25575	160	136	-38.63643	-111.5844	-92.02337	-125.9079	-89.30602	-117.6131
25576	160	137	-38.60972	-113.0431	-90.20233	-114.3313	-89.12368	-111.8794
25577	160	138	-38.60507	-112.4055	-91.93743	-110.9749	-89.60449	-110.8481
25578	160	139	-38.59578	-111.9015	-90.20000	-113.3267	-90.83788	-112.3335
25579	160	140	-38.61436	-118.1810	-89.81326	-111.5777	-91.40580	-115.4727
25580	160	141	-38.61669	-111.9769	-89.96541	-116.0687	-90.65787	-115.0976
25581	160	142	-38.59927	-116.4134	-90.83180	-113.5450	-90.56960	-115.8885
25582	160	143	-38.79438	-111.6587	-89.93056	-113.5439	-90.23164	-112.3765
25583	160	144	-38.59346	-112.8038	-90.35795	-114.1246	-90.45230	-114.0105

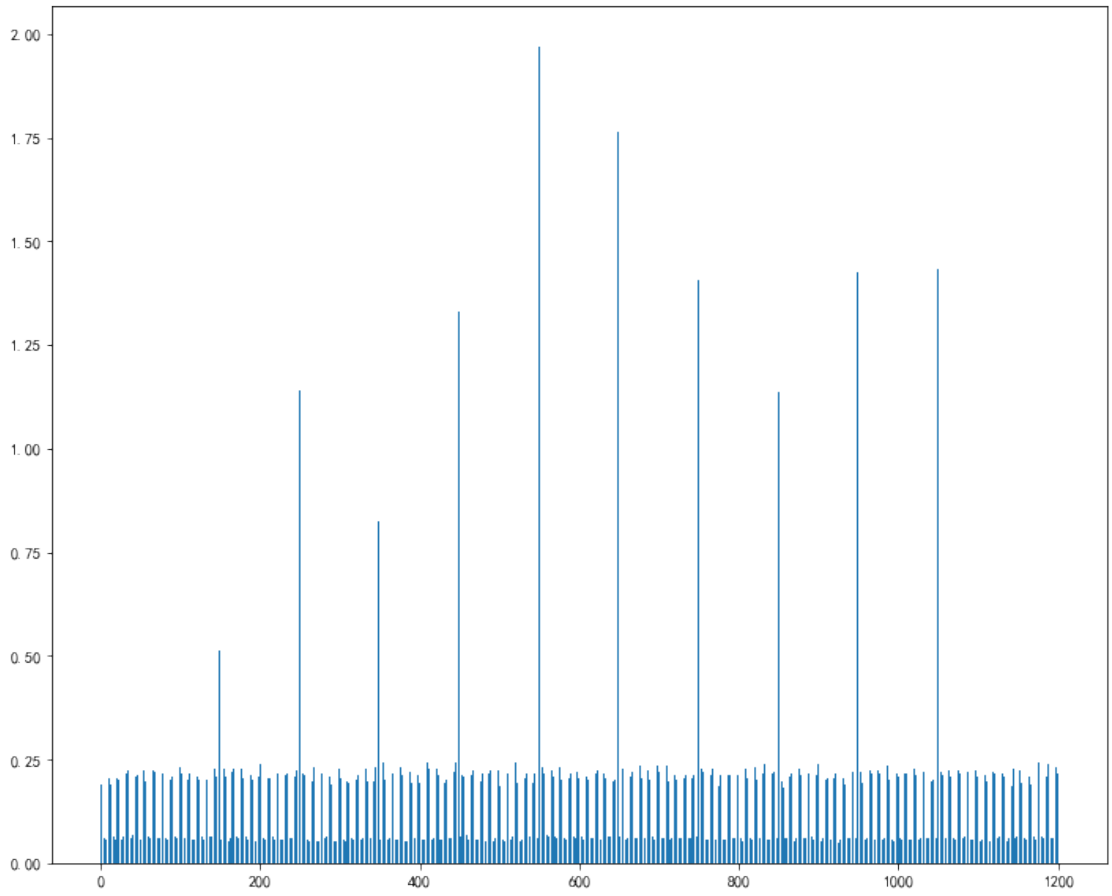
```
25584  160  145 -38.60623 -115.6690 -88.97359 -117.5321 -91.24553 -109.0596
25585  160  146 -38.58649 -114.4681 -90.05948 -115.6878 -90.73684 -116.6852
25586  160  147 -38.58881 -113.6981 -91.00600 -113.8226 -90.03769 -114.7805
25587  160  148 -38.60972 -112.7481 -91.31725 -115.2372 -89.73109 -111.4450
25588  160  149 -38.58185 -111.0281 -91.56927 -120.8873 -88.46286 -114.3102
25589  160  150 -38.81877 -118.5329 -91.63663 -112.8018 -90.92150 -113.8654
25590  160  151 -38.78625 -111.3103 -90.61926 -115.1698 -90.79491 -118.0707
25591  160  152 -38.81296 -115.2973 -90.92354 -113.6182 -90.73103 -120.6641
25592  160  153 -38.80483 -121.4910 -89.33245 -116.2673 -91.05274 -112.4461
25593  160  154 -38.57488 -115.8385 -92.06867 -114.2349 -90.26648 -110.2267
25594  160  155 -38.74328 -115.7061 -91.89910 -114.1153 -89.89716 -112.6842
25595  160  156 -38.77812 -111.7110 -91.70980 -114.0979 -89.85768 -112.0548
25596  160  157 -38.79438 -119.0184 -90.28827 -120.3031 -90.70897 -113.7422
25597  160  158 -38.79554 -113.0094 -92.80614 -112.0097 -89.09697 -114.3659
25598  160  159 -38.76767 -113.6714 -90.68546 -110.3826 -91.85874 -115.8269
25599  160  160 -38.79206 -110.3382 -90.42647 -114.7378 -92.39994 -109.9016

              f7        f8  ...      f992      f993      f994      f995  \
0     -89.32102 -115.4919  ... -113.7846 -86.86931 -112.7648 -88.75225
1     -91.72160 -112.2330  ... -115.0726 -89.50216 -117.2245 -88.00780
2     -90.37672 -109.5096  ... -110.3237 -87.87507 -113.2537 -88.20640
3     -87.82284 -115.1226  ... -113.1110 -88.19096 -108.5629 -88.96710
4     -92.74477 -114.2260  ... -113.8926 -89.10265 -110.6743 -88.13439
5     -90.53815 -113.8497  ... -113.8590 -88.23858 -108.4619 -88.11813
6     -89.08758 -116.6068  ... -106.9755 -89.61946 -107.0276 -88.15646
7     -91.77037 -116.9239  ... -111.7011 -87.98192 -108.2052 -87.74417
8     -91.59036 -112.5327  ... -108.1090 -87.79726 -113.5511 -88.35041
9     -91.76108 -115.4292  ... -110.8417 -88.21652 -112.0970 -89.28532
10    -89.74376 -118.6392  ... -112.9473 -89.64269 -111.4757 -88.17156
11    -89.91449 -117.9168  ... -113.5686 -88.32685 -112.1063 -88.42938
12    -90.86333 -122.1988  ... -112.6209 -88.67526 -111.6940 -89.78239
13    -92.22680 -113.7335  ... -110.2645 -89.93536 -112.8589 -89.12737
14    -90.64383 -115.3339  ... -110.6710 -88.67526 -110.7579 -89.98447
15    -90.10147 -121.2930  ... -112.0054 -87.69157 -112.2503 -87.38530
16    -88.85995 -113.5408  ... -113.8671 -88.10386 -112.4129 -90.05880
17    -90.27916 -114.7718  ... -115.1400 -86.99706 -109.9705 -85.94054
18    -88.03421 -114.2295  ... -114.3467 -88.48131 -112.7752 -87.61293
19    -89.94236 -111.4212  ... -113.9031 -87.57195 -121.7829 -86.52007
20    -89.70195 -115.5592  ... -110.2308 -87.54988 -109.5536 -87.14025
21    -91.39757 -112.0821  ... -110.1762 -86.40244 -111.8485 -87.70352
22    -87.52553 -117.9552  ... -114.5465 -87.43258 -111.1900 -88.32021
23    -88.83092 -110.1635  ... -112.3596 -86.23171 -110.9112 -89.23190
24    -89.67756 -114.4037  ... -111.9915 -86.72530 -108.7801 -87.73371
25    -90.86101 -113.8938  ... -111.5699 -88.26762 -113.9819 -89.03911
26    -92.11763 -114.2283  ... -107.9475 -89.30125 -110.8926 -89.11111
27    -90.69610 -114.2596  ... -113.6929 -89.88426 -110.8590 -86.33309
28    -90.76113 -119.1270  ... -114.2608 -89.12936 -112.2793 -89.06234
29    -90.75765 -110.2099  ... -113.0460 -87.56498 -114.3803 -89.49785
```

```
...         ...       ...  ...       ...       ...       ...       ...
25570 -89.99462 -118.5242  ... -109.7558 -87.38729 -113.7195 -87.85682
25571 -90.05153 -117.7345  ... -113.3445 -89.48242 -108.3632 -88.99614
25572 -90.49169 -115.7276  ... -106.2705 -88.31756 -109.2667 -88.30628
25573 -92.09904 -110.0705  ... -111.5490 -86.74272 -114.0725 -87.00088
25574 -90.86217 -116.3629  ... -111.0728 -87.67880 -107.4329 -88.51881
25575 -91.57758 -114.0715  ... -109.6083 -85.82175 -109.8764 -86.34703
25576 -91.35344 -112.4525  ... -110.6083 -88.08876 -106.3354 -89.66858
25577 -89.88545 -112.3457  ... -110.6373 -87.17359 -108.8138 -88.79406
25578 -89.16888 -114.4838  ... -112.2725 -87.00171 -116.1189 -88.41312
25579 -90.45337 -119.6949  ... -109.7186 -87.00055 -124.0163 -89.26209
25580 -90.46498 -119.6577  ... -113.9716 -87.10507 -111.7637 -87.33885
25581 -91.44170 -112.0135  ... -110.1170 -86.81937 -113.3525 -87.96715
25582 -92.09208 -108.5642  ... -118.3233 -87.24095 -110.4931 -85.77098
25583 -89.48710 -118.2548  ... -108.7175 -87.99934 -106.7500 -89.37126
25584 -89.87616 -120.8180  ... -111.3887 -87.58008 -109.3701 -89.81375
25585 -91.55203 -115.5012  ... -107.1427 -88.65900 -117.1107 -89.48740
25586 -90.65429 -117.0121  ... -114.9797 -87.92733 -105.8081 -88.78593
25587 -90.15954 -116.1585  ... -109.1925 -87.27696 -113.2119 -87.43292
25588 -90.52653 -111.6326  ... -112.5907 -88.44763 -112.7044 -89.18312
25589 -91.39409 -112.1575  ... -110.5966 -88.57887 -113.2630 -87.41434
25590 -89.44877 -116.4558  ... -115.8345 -85.93092 -111.3212 -88.23775
25591 -90.33491 -118.4731  ... -107.8302 -88.33846 -115.4894 -87.93928
25592 -89.85990 -111.1204  ... -110.6315 -88.96096 -110.1250 -88.86838
25593 -89.81577 -114.0111  ... -113.7219 -88.85760 -111.6835 -87.14838
25594 -92.63444 -112.7719  ... -108.1926 -87.03307 -119.1884 -88.94620
25595 -91.60662 -118.5858  ... -111.3550 -88.51499 -119.9526 -89.68948
25596 -90.71584 -118.2107  ... -111.2935 -88.89941 -110.9101 -88.46190
25597 -90.20483 -112.5280  ... -117.3814 -88.34543 -110.4014 -87.65126
25598 -89.79835 -111.6175  ... -107.0672 -88.17703 -116.8110 -88.23195
25599 -90.84127 -121.0549  ... -115.4663 -86.56735 -112.0819 -89.10879

           f996      f997      f998      f999     f1000     f1001
0     -111.3889 -86.16270 -115.3461 -85.26645 -111.1712 -87.92286
1     -110.6061 -87.16730 -116.9836 -87.34881 -110.4790 -88.99714
2     -110.1567 -88.53540 -117.2658 -87.86097 -109.9947 -87.89847
3     -105.6703 -88.37978 -110.1315 -87.32906 -112.1932 -89.93670
4     -108.7549 -89.07893 -110.2801 -87.70419 -111.9563 -89.77875
5     -115.0925 -89.06035 -111.1546 -87.34416 -118.8944 -87.24462
6     -113.8022 -88.00930 -111.8352 -86.77741 -115.1652 -88.91352
7     -110.3715 -89.93254 -113.2962 -87.51953 -110.0702 -88.48729
8     -109.7664 -88.63064 -113.5506 -87.98292 -114.2419 -87.84505
9     -109.2752 -86.80727 -113.8409 -88.75059 -108.7242 -85.98568
10    -110.4459 -88.25667 -108.9887 -88.37547 -116.6355 -87.28410
11    -108.8199 -87.36705 -109.9561 -87.39991 -108.8345 -90.11903
12    -104.6053 -88.20905 -113.0500 -86.95161 -111.9946 -87.18074
13    -108.3728 -88.64573 -116.1010 -87.53114 -113.1200 -87.63368
14    -109.6956 -87.19749 -115.9116 -87.92833 -115.9201 -89.40130
```

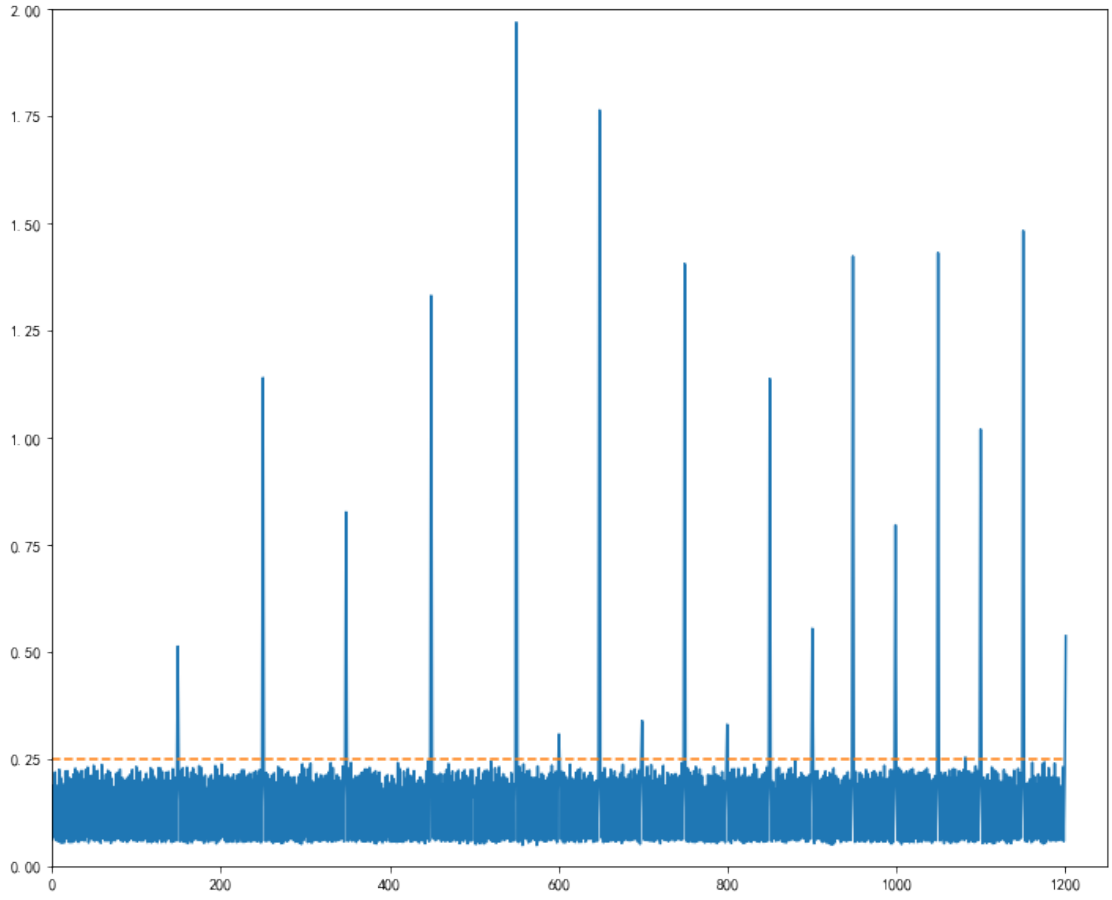
```
15    -112.8720 -88.13240 -106.8587 -85.64390 -112.2722 -88.07036
16    -112.6060 -86.01869 -113.4054 -86.17581 -112.1177 -87.27713
17    -112.3877 -88.30313 -114.4971 -86.92142 -112.5184 -87.31314
18    -111.5015 -88.70961 -108.7854 -87.52534 -112.3210 -88.95765
19    -117.5628 -87.36821 -116.5783 -88.27443 -106.4316 -89.30839
20    -113.3958 -87.14987 -114.7468 -88.03402 -112.3268 -88.59646
21    -110.5028 -89.56787 -113.1092 -87.19434 -110.3350 -87.58490
22    -123.1003 -88.88033 -114.3008 -88.30927 -109.4721 -87.92402
23    -109.9546 -89.25430 -112.8154 -89.14662 -116.8318 -88.19811
24    -113.6280 -88.16841 -111.7620 -87.61476 -113.4591 -87.29223
25    -112.8383 -87.81883 -107.5973 -87.47075 -112.7542 -88.55814
26    -113.0044 -85.61104 -118.5910 -88.96429 -114.2222 -87.16448
27    -114.0252 -88.23809 -111.7748 -86.52190 -106.6546 -88.03435
28    -105.4333 -85.25566 -109.5868 -86.97949 -116.5147 -88.05410
29    -113.8893 -89.17184 -107.6287 -88.59264 -114.0793 -87.31081
...         ...       ...       ...       ...       ...       ...
25570 -109.0882 -89.51329 -111.5994 -88.31391 -114.4812 -88.14352
25571 -111.2495 -89.24152 -110.0467 -87.68793 -107.9879 -87.08086
25572 -108.4018 -88.18699 -113.8850 -88.37663 -109.6545 -89.23871
25573 -108.3089 -88.83852 -108.6379 -86.98645 -109.7125 -87.43508
25574 -111.3088 -86.91528 -116.4668 -86.80296 -117.7295 -88.48962
25575 -109.9372 -87.79328 -108.8609 -87.54276 -109.1632 -88.31309
25576 -109.0394 -88.05111 -115.4552 -88.37779 -111.8204 -88.89494
25577 -109.2299 -90.13927 -116.0951 -89.01655 -109.7601 -89.37227
25578 -113.6118 -87.41351 -115.2497 -89.53220 -111.9900 -87.53148
25579 -110.4749 -87.50874 -109.9224 -88.84931 -111.2363 -87.81485
25580 -111.5526 -87.67366 -110.2081 -88.85860 -112.4522 -87.71497
25581 -114.5571 -88.52495 -107.8412 -88.10719 -108.7974 -89.07495
25582 -112.0149 -88.09175 -115.1846 -88.55200 -113.7588 -88.47568
25583 -109.1950 -89.13584 -110.5844 -86.64501 -112.5091 -88.12378
25584 -110.0684 -88.60160 -112.2974 -89.12223 -118.8166 -88.28405
25585 -115.0635 -87.11619 -110.8863 -87.49166 -112.1839 -87.17377
25586 -109.9964 -87.67714 -109.1884 -87.48004 -112.4929 -88.98204
25587 -119.2631 -88.64457 -113.6284 -89.75635 -124.8186 -87.54193
25588 -116.0658 -87.33453 -112.3810 -87.09795 -112.8912 -88.32935
25589 -112.7895 -86.92573 -112.2243 -87.62754 -113.7170 -87.85202
25590 -111.0509 -87.47158 -109.5252 -89.79003 -110.9657 -87.15983
25591 -110.0951 -88.49127 -115.5086 -89.33825 -120.3519 -87.19816
25592 -108.2021 -87.50642 -112.7608 -88.03867 -111.3013 -87.65342
25593 -113.0671 -86.86301 -115.2415 -90.69939 -111.0748 -87.62323
25594 -113.4666 -88.49359 -108.3116 -86.81922 -119.2707 -87.88337
25595 -109.4715 -87.59120 -108.7053 -87.62173 -109.5081 -90.08419
25596 -118.8124 -87.97097 -108.9317 -87.76690 -112.8552 -90.01219
25597 -109.7432 -88.85014 -114.6980 -89.21979 -112.8006 -88.88332
25598 -105.8735 -88.05343 -110.4311 -88.62865 -109.6057 -87.81485
25599 -116.4792 -87.78050 -109.5426 -89.29528 -107.3723 -87.30617

[25600 rows x 1003 columns]
```

```
In [249]: data_xx_yy.to_csv('data_xx_yy_n9010a.csv',index=None)

In [27]: data_shuffled_df.head()

Out[27]:          f1        f2        f3        f4        f5        f6        f7  \
         0 -41.74312 -119.5852 -91.39158 -120.8374 -91.08758 -117.8814 -89.32102
         1 -41.47252 -116.9360 -90.70404 -111.4290 -91.56723 -114.7608 -91.72160
         2 -41.45278 -112.9002 -92.39501 -119.5529 -91.61136 -117.3843 -90.37672
         3 -41.49111 -112.2719 -88.10023 -118.6307 -91.17468 -114.2056 -87.82284
         4 -41.47020 -110.2488 -89.48924 -108.4443 -90.21306 -118.4110 -92.74477

                  f8        f9       f10  ...       f992      f993      f994      f995  \
         0 -115.4919 -91.61795 -120.9368  ... -113.7846 -86.86931 -112.7648 -88.75225
         1 -112.2330 -89.56928 -115.0985  ... -115.0726 -89.50216 -117.2245 -88.00780
         2 -109.5096 -89.88517 -111.6596  ... -110.3237 -87.87507 -113.2537 -88.20640
         3 -115.1226 -89.22667 -112.2368  ... -113.1110 -88.19096 -108.5629 -88.96710
         4 -114.2260 -91.88972 -112.5376  ... -113.8926 -89.10265 -110.6743 -88.13439

                f996      f997      f998      f999     f1000     f1001
         0 -111.3889 -86.16270 -115.3461 -85.26645 -111.1712 -87.92286
         1 -110.6061 -87.16730 -116.9836 -87.34881 -110.4790 -88.99714
         2 -110.1567 -88.53540 -117.2658 -87.86097 -109.9947 -87.89847
         3 -105.6703 -88.37978 -110.1315 -87.32906 -112.1932 -89.93670
         4 -108.7549 -89.07893 -110.2801 -87.70419 -111.9563 -89.77875

         [5 rows x 1001 columns]

In [28]: data_1001 = [] #160*160
         for f in data_shuffled_df.columns:
             data_1001.append(data_shuffled_df[f].values.reshape(160,160).T)
         data_1001 = np.array(data_1001)

In [29]: data_1001[0].std(axis=0).std()#

Out[29]: 0.06090926029165222

In [30]: std_all = []
         for it in data_1001:
             std_all.append(it.std(axis=0).std())

In [31]: std_all = np.array(std_all)
         std_all[-10:]

Out[31]: array([0.20565058, 0.05982225, 0.20560615, 0.06061483, 0.18633369,
                0.0621454 , 0.23095907, 0.05752638, 0.21568618, 0.53681566])

In [32]: fig, ax = plt.subplots()
         std_x = np.linspace(0,1201,1001)
         plt.bar(std_x, std_all)
         #plt.step(std_x, std_all)
         #plt.stem(std_x, std_all)#
```

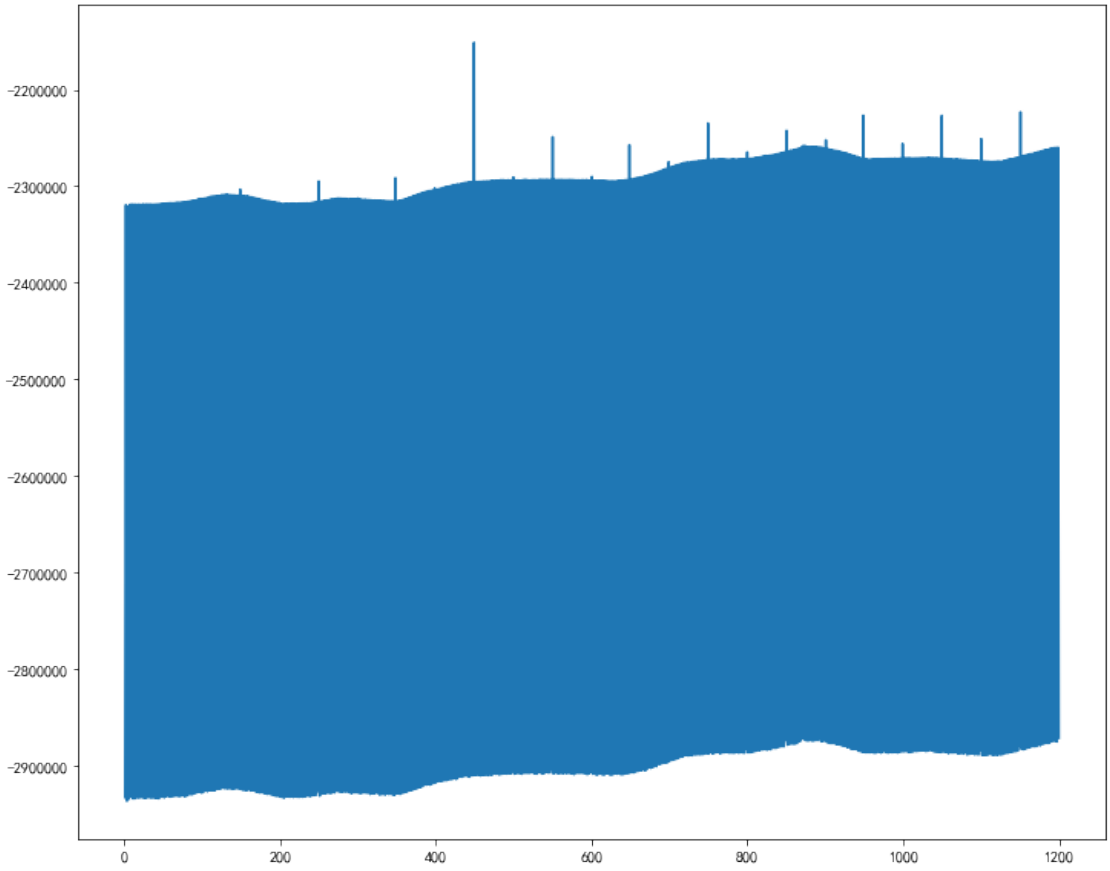
```
Out[32]: <BarContainer object of 1001 artists>
```




```
In [33]: std_all[std_all>=1.0]

Out[33]: array([1.13962112, 1.33132377, 1.96866888, 1.76402775, 1.40561384,
                1.13724405, 1.42333566, 1.43129113, 1.01908209, 1.48262081])

In [34]: fig = plt.figure()

<Figure size 864x720 with 0 Axes>


In [35]: threshold_y = [0.25]*1001

In [36]: plt.plot(std_x, std_all)
         plt.plot(std_x, threshold_y,linestyle='--')
         plt.xlim([0,1250])
         plt.ylim([0.00,2.00])

Out[36]: (0.0, 2.0)
```




```
In [37]: sum_all = []
         for it in data_1001:
             sum_all.append(it.sum(axis=0).sum())#

In [38]: fig, ax = plt.subplots()
         ax.plot(std_x[1:-1], sum_all[1:-1])

Out[38]: [<matplotlib.lines.Line2D at 0x25196283b00>]
```



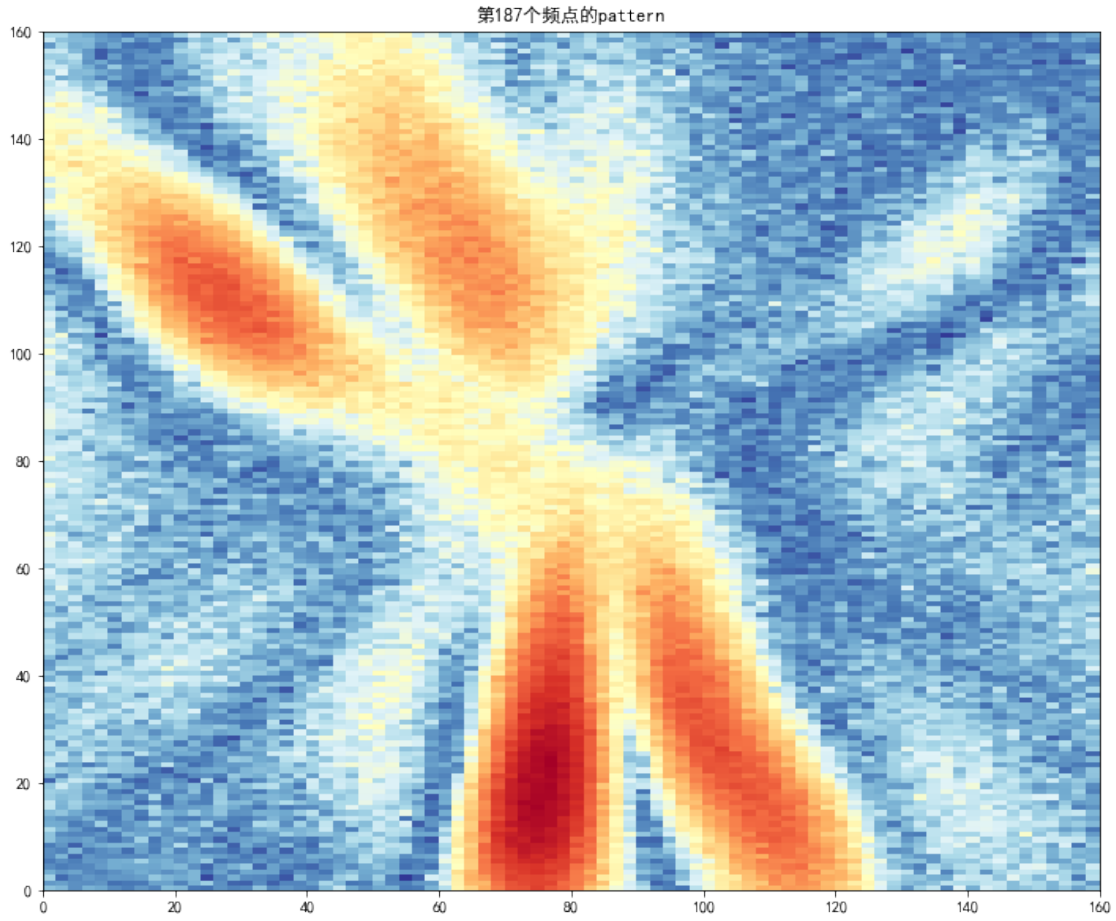

```
In [39]: print(std_all[std_all>0.25])
         print(len(std_all[std_all>0.25]))
```

```
[0.51245104 1.13962112 0.82590883 1.33132377 1.96866888 0.30769138
 1.76402775 0.33898844 1.40561384 0.33031613 1.13724405 0.55432597
 1.42333566 0.79570215 1.43129113 0.25352463 1.01908209 1.48262081
 0.53681566]
19
```

pattern

```
In [42]: a = pd.read_csv('a.csv',header=None)

In [43]: a.shape

Out[43]: (160, 160)

In [44]: plt.subplots()
         #plt.imshow(np.rot90(a.T), cmap=plt.cm.RdYlBu_r)
         plt.pcolor(np.rot90(a.T,3), cmap=plt.cm.RdYlBu_r)
         plt.title('187pattern')
```

Out[44]: Text(0.5, 1.0, '187pattern')



In [45]: std_all>=0.25

Out[45]: array([False, False, False, ..., False, False, True])

In [46]: print(len(data_1001))

1001

In [40]:
```
def find_all_indixes(a, val):
    '''
    find all the indexes of val in a

    parameters
    aan 1D array-like.
    val: the values that is required to find all the indexes.
```

```
            returns:
            a list of
            '''
            ans = [i for i, value in enumerate(a) if value == val]
            #or
            #ans=list(filter(lambda x:a[x] == val, range(len(a))))
            return ans
```

```
In [41]: mask0dot3 = std_all>=0.3
         print(len(mask0dot3))
```

```
1001
```

```
In [47]: mask0dot3[:5]
```

```
Out[47]: array([False, False, False, False, False])
```

```
In [48]: ind0dot3 = find_all_indixes(mask0dot3, True)
```

```
In [49]: len(ind0dot3)
```

```
Out[49]: 18
```

```
In [50]: data_extract_by_std = data_1001[std_all>=0.3]
```

```
In [51]: data_extract_by_std.shape
```

```
Out[51]: (18, 160, 160)
```

```
In [52]: data_extract_by_std[0].shape
```

```
Out[52]: (160, 160)
```

```
In [55]: start_freq = 120 #120Hz
         stop_freq = 6 * 1000 * 1000 * 1000 #6GHz
         span_freq = stop_freq - start_freq
         center_freq = (start_freq + stop_freq) / 2.0
```

```
In [93]: print(span_freq)
         print(center_freq)
```

```
5999999880
3000000060.0
```

start_freq

```
In [94]: center_freq - span_freq / 2
```

```
Out[94]: 120.0
```

span_freqpoint_freqi
freq_i = start_freq + span_freq / point_freq * (i - 1) = center_freq - span_freq / 2 + span_freq / (point_freq - 1) * (i - 1)

```
In [184]: i = 691
          point_freq = 691
          freq_1 = start_freq + span_freq / (point_freq - 1) * (i - 1)
          print(freq_1)
```

```
6000000000.0
```

```
In [96]: a = data_extract_by_std[0]
```

```
In [45]: def get_frequency(num, start_freq, stop_freq, points_freq):
             '''
             num
             freq = start_freq + span_freq / (point_freq - 1) * (num - 1)
             span_freq = stop_freq - start_freq
             center_Freq = (start_freq + stop_freq) / 2.0


             numintnon-negative
                  num
             start_freqfloat

             stop_freqfloat

             points_freqintnon-negative



             floatnum
             '''
             if points_freq == 1:
                 return start_freq
             if num > points_freq:
                 raise ValueError('num:' + str(num) + " must be less equal than " + 'points_fre
             if start_freq > stop_freq:
                 raise ValueError("start_freq:" + str(start_freq) + " must be less equal than s

             span_freq = stop_freq - start_freq
             freq = start_freq + span_freq / (points_freq - 1) * (num - 1)
             return freq
```

```
In [53]: freq_list = [get_frequency(i,0,1200*10**6,1001) for i in ind0dot3]
         print(freq_list)
```

```
[147600000.0, 248400000.0, 346800000.0, 447600000.0, 548400000.0, 598800000.0, 646800000.0, 697
```

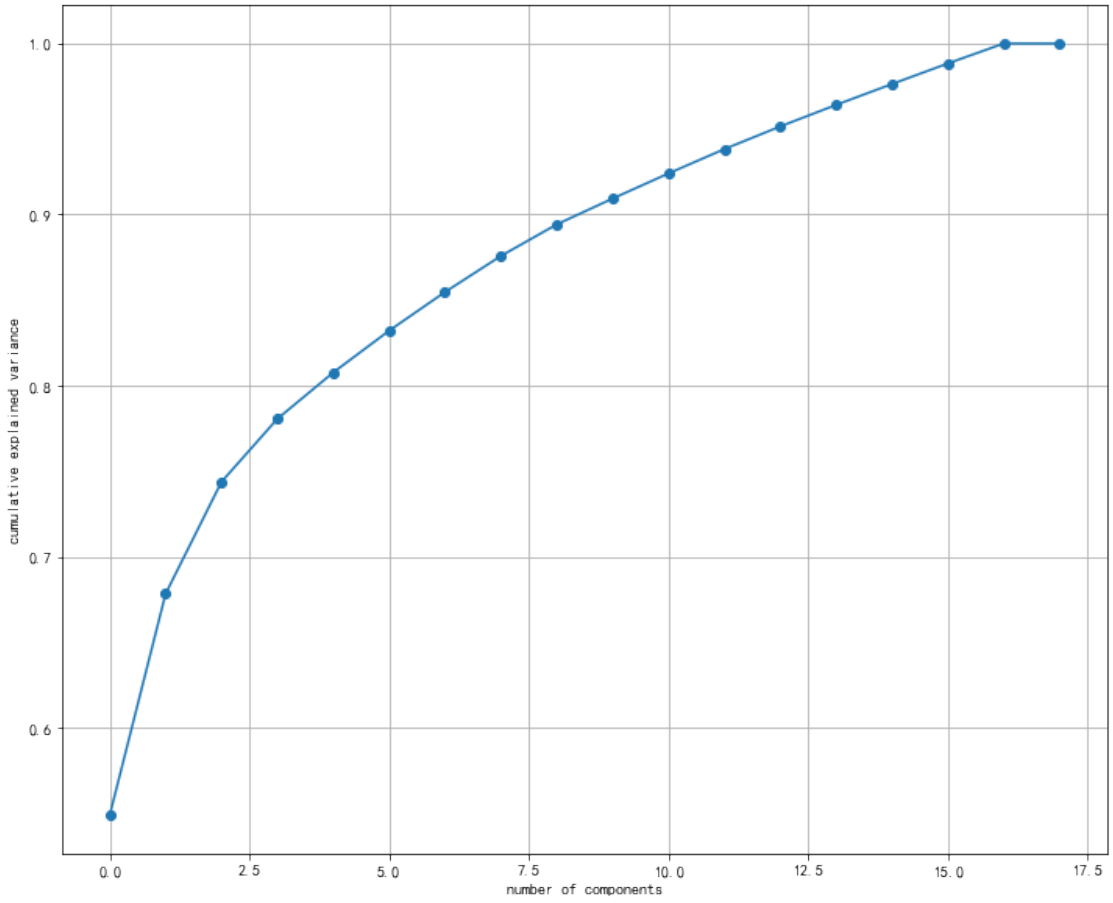
```
In [54]: def approximate_size(size):
             '''

             Arguments:
             size -- the size of the file in bytes
             a_kilobyte_is_1024_bytes -- indicates whether use 1024 or 1000 to compute file si

             Returns:
             a string

             '''

             SUFFIXES = {1000:['KHz', 'MHz', 'GHz', 'THz']}
             if size < 0:
                 raise ValueError("size must be non-negative!")
             multiples = 1000
             for suffix in SUFFIXES[multiples]:
                 size /= multiples
                 if size < multiples:
                     return "{0:.5f}{1}".format(size, suffix)
             raise ValueError('size is too large.')
```

```
In [55]: freq_list_human = [approximate_size(f) for f in freq_list]

         freq_list_human = np.array(freq_list_human)
         print(freq_list_human)
```

```
['147.60000MHz' '248.40000MHz' '346.80000MHz' '447.60000MHz'
 '548.40000MHz' '598.80000MHz' '646.80000MHz' '697.20000MHz'
 '747.60000MHz' '798.00000MHz' '848.40000MHz' '898.80000MHz'
 '946.80000MHz' '997.20000MHz' '1.04760GHz' '1.09800GHz' '1.14840GHz'
 '1.19880GHz']
```

```
In [56]: approximate_size(1200*10**6)

Out[56]: '1.20000GHz'

In [97]: a

Out[97]: array([[-92.09586, -88.73598, -90.84505, ..., -90.58722, -88.96826,
                 -88.94852],
                [-90.6662 , -90.28062, -90.7533 , ..., -89.93801, -90.99835,
                 -90.94725],
                [-90.11106, -89.46301, -90.05299, ..., -91.25966, -91.11681,
                 -90.09596],
```

```
                ...,
                [-89.96937, -88.73714, -88.57803, ..., -88.57223, -90.80324,
                 -87.68145],
                [-90.27481, -91.5744 , -92.04592, ..., -88.71392, -90.85899,
                 -90.76143],
                [-91.10288, -89.94614, -89.90085, ..., -89.06001, -90.47341,
                 -90.04718]])

In [98]: a_flat = a.reshape(1,160*160)

In [99]: a_flat.shape

Out[99]: (1, 25600)

In [100]: a_flat

Out[100]: array([[-92.09586, -88.73598, -90.84505, ..., -89.06001, -90.47341,
                  -90.04718]])

In [57]: data_pca = []
         for it in data_extract_by_std:
             data_pca.append(it.reshape(1, 160**2))
         data_pca = np.array(data_pca)

In [58]: data_pca = np.squeeze(data_pca)

In [59]: data_pca.shape

Out[59]: (18, 25600)

In [60]: from sklearn.decomposition import PCA

In [61]: pca = PCA().fit(data_pca)

In [62]: plt.subplots()
         plt.plot(np.cumsum(pca.explained_variance_ratio_), 'o-')
         plt.xlabel('number of components')
         plt.ylabel('cumulative explained variance')
         plt.grid()
```




```
In [63]: pca1 = PCA(0.98).fit(data_pca)
         pca1.n_components_
```

```
Out[63]: 16
```

```
In [68]: pca_m = PCA(n_components=pca1.n_components_).fit(data_pca)
```

```
In [69]: components = pca_m.transform(data_pca)
```

```
In [70]: components.shape
```

```
Out[70]: (18, 16)
```

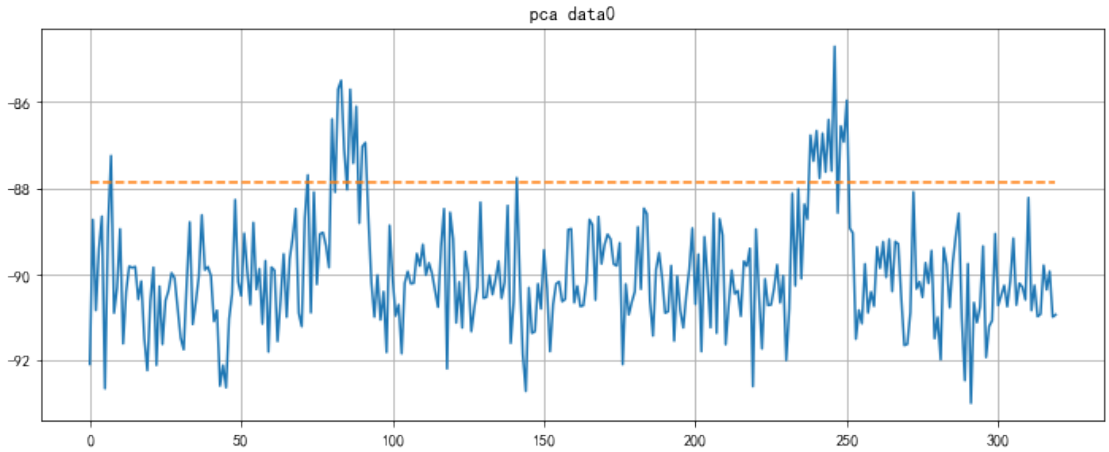
```
In [71]: filter_data = pca_m.inverse_transform(components)#
```

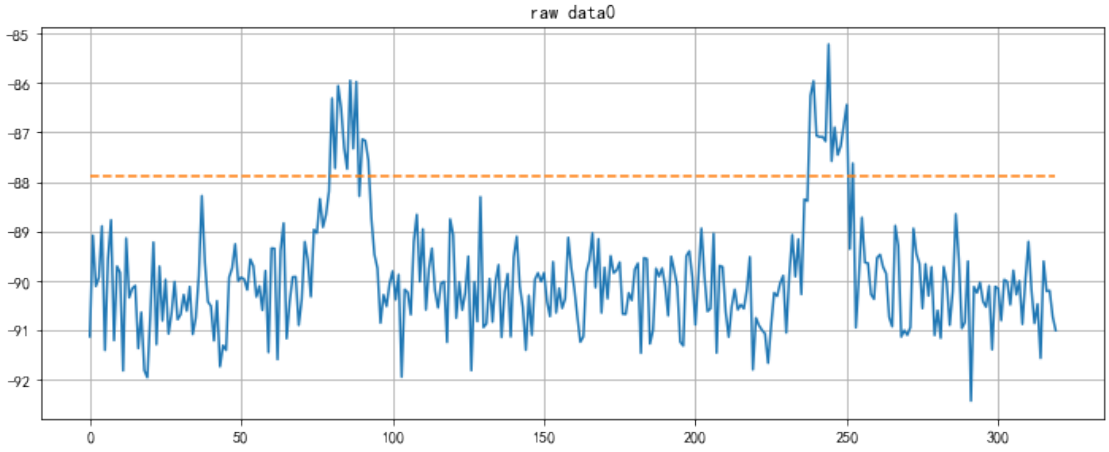
```
In [72]: print(filter_data.shape)
         print(data_pca.shape)
```

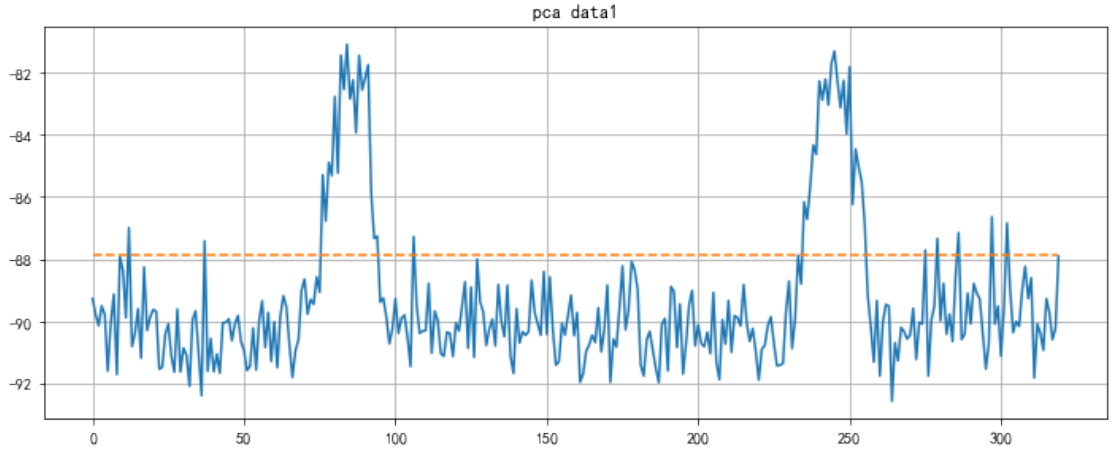
```
(18, 25600)
(18, 25600)
```

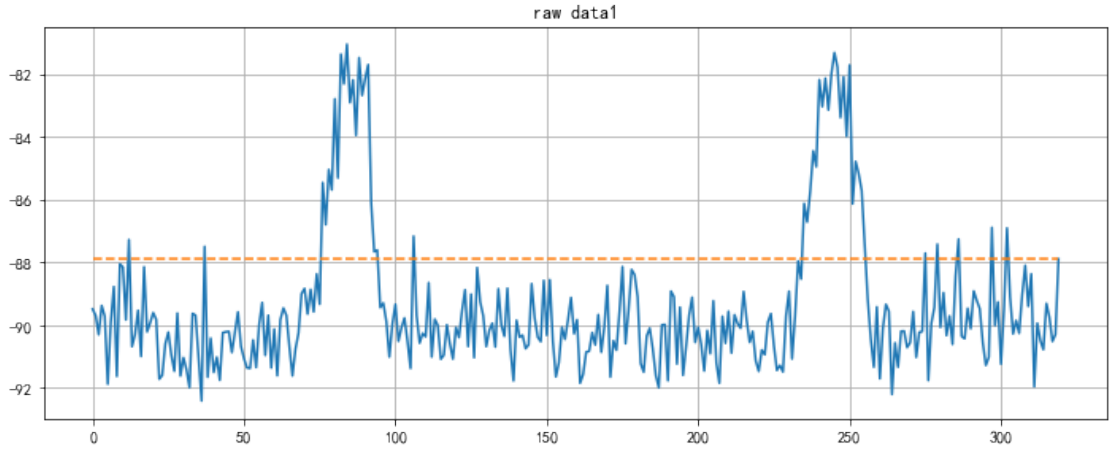
PCA

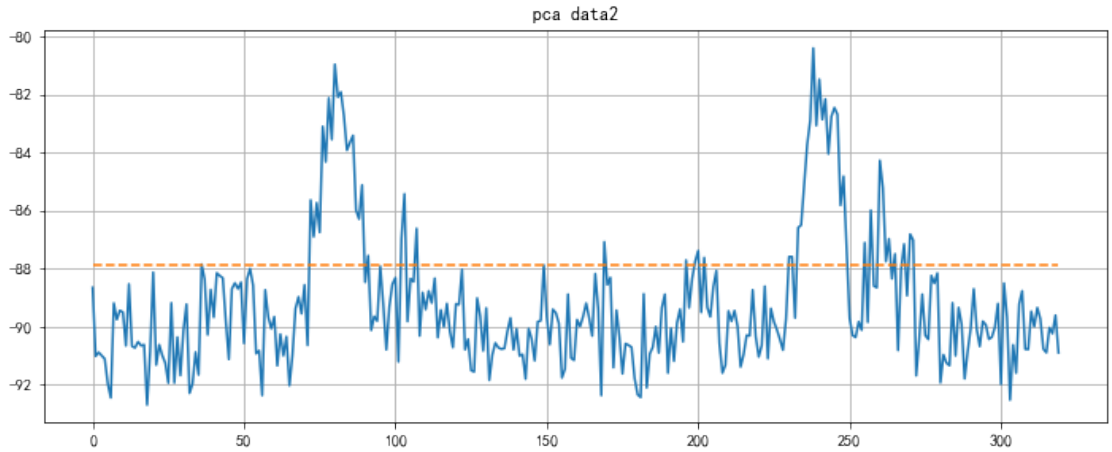
```
In [74]: seen = 320 #320/160=2,
         y_threshold = [-87.88] * seen
         for i in range(components.shape[0]):
             fig = plt.figure()

             ax1 = fig.add_subplot(211)
             ax1.plot(data_pca[i][:seen])#
             ax1.plot(y_threshold,'--')
             ax2 = fig.add_subplot(212)
             ax2.plot(filter_data[i][:seen])#
             ax2.plot(y_threshold,'--')
             ax1.set_title("pca data" + str(i))
             #plt.grid( dict(color='r', linestyle='-', linewidth=2))
             ax2.set_title('raw data' + str(i))
             ax1.grid()
             ax2.grid()



             #plt.grid( dict(color='r', linestyle='-', linewidth=2))
```

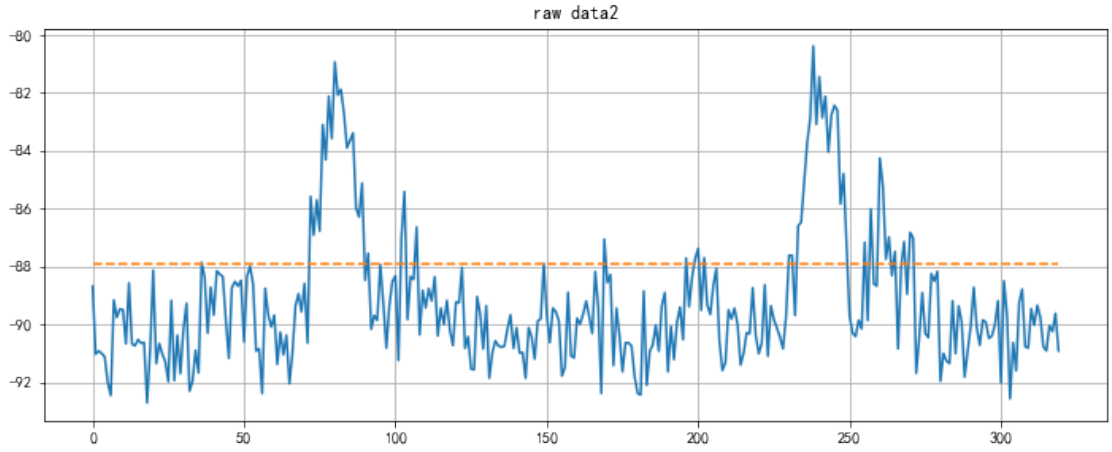

pca data0
-86
-88
-90
-92
0
50
100
150
200
250
300

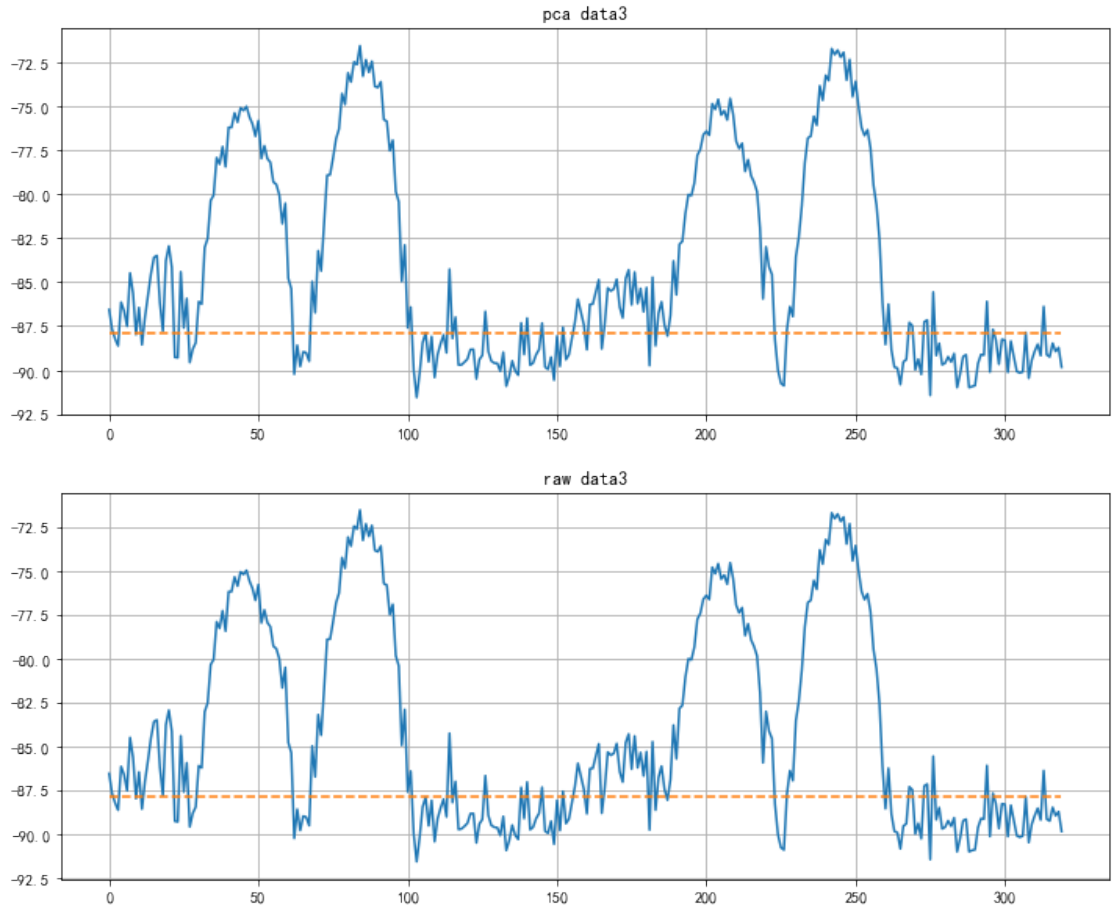

raw data0
-85
-86
-87
-88
-89
-90
-91
-92
0
50
100
150
200
250
300

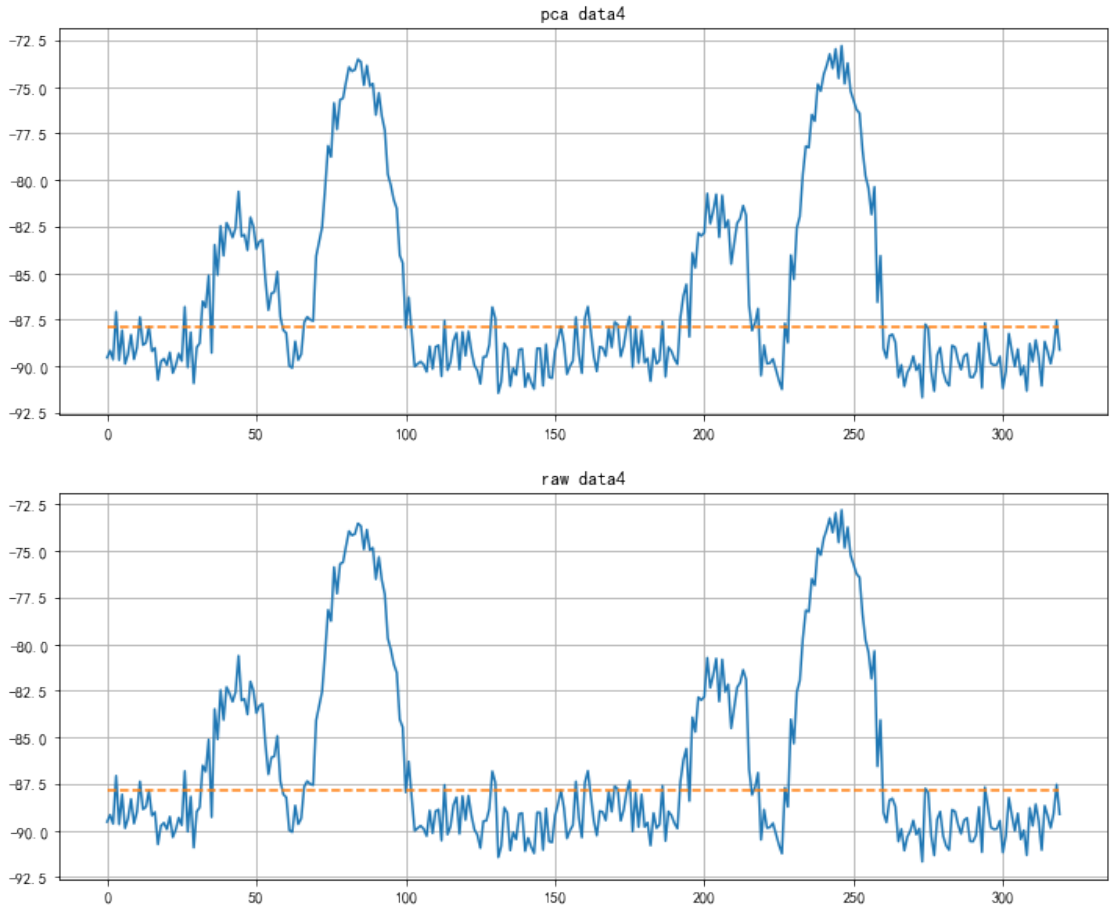

pca data1
-82
-84
-86
-88
-90
-92
0
50
100
150
200
250
300

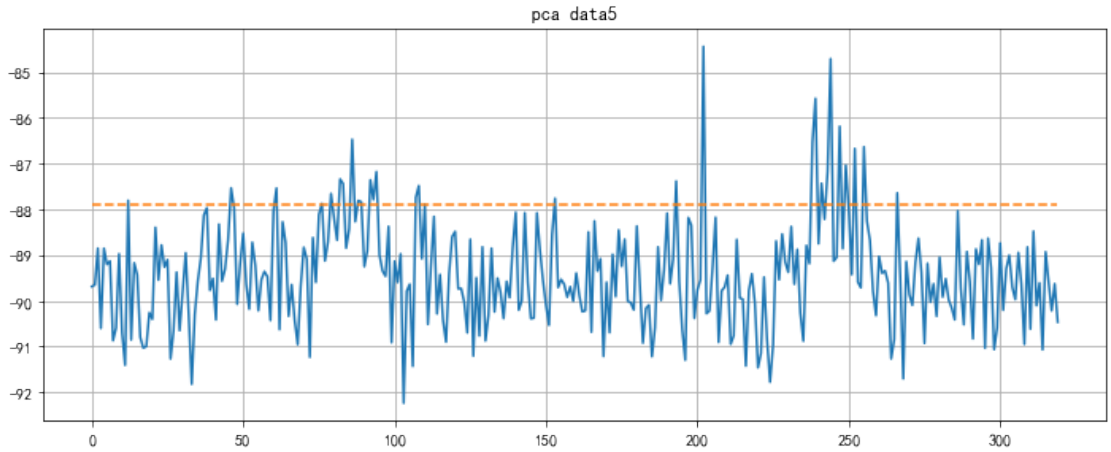

raw data1
-82
-84
-86
-88
-90
-92
0
50
100
150
200
250
300

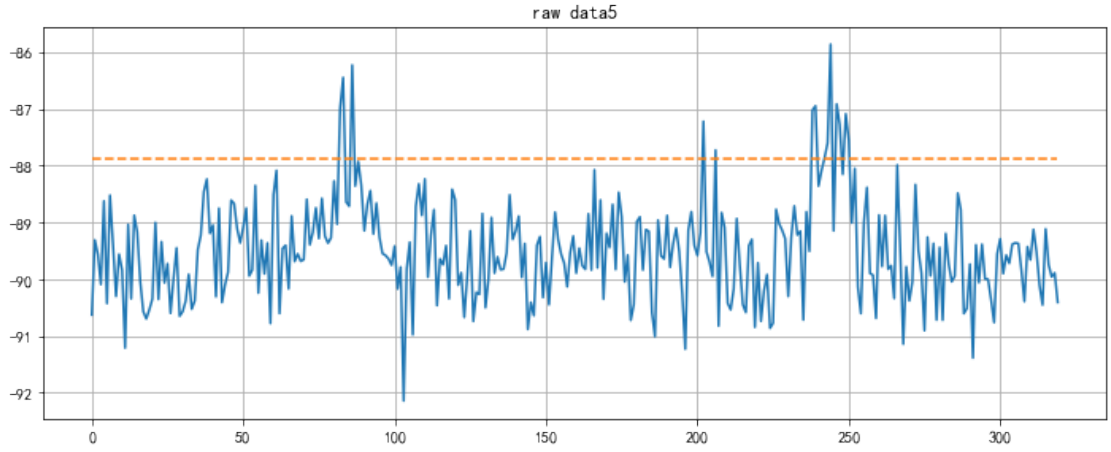

pca data2
-80
-82
-84
-86
-88
-90
-92
0
50
100
150
200
250
300

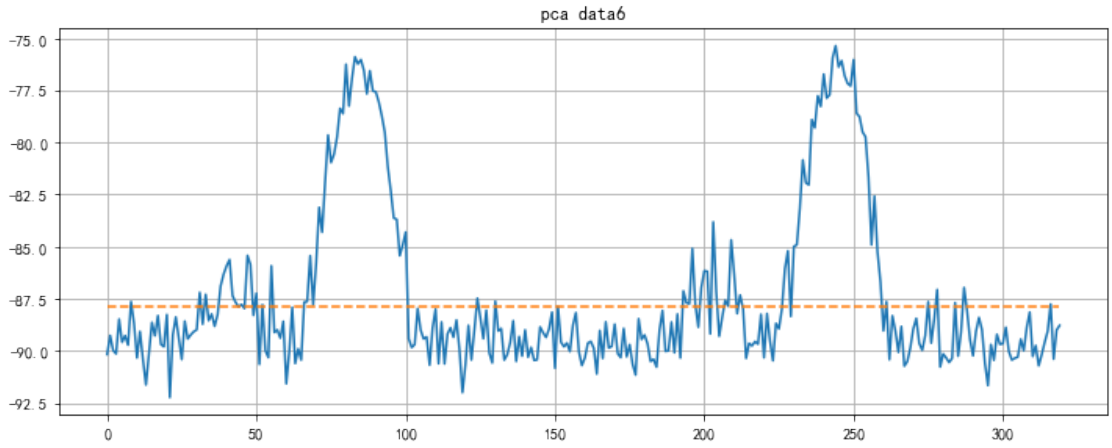

raw data2
-80
-82
-84
-86
-88
-90
-92
0
50
100
150
200
250
300

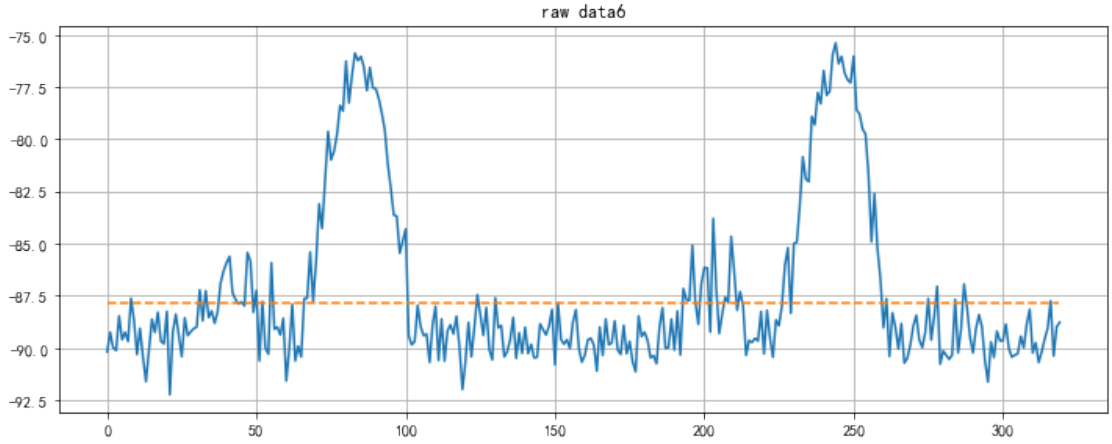

pca data3
-72.5
-75.0
-77.5
-80.0
-82.5
-85.0
-87.5
-90.0
-92.5
0
50
100
150
200
250
300
raw data3
-72.5
-75.0
-77.5
-80.0
-82.5
-85.0
-87.5
-90.0
-92.5
0
50
100
150
200
250
300

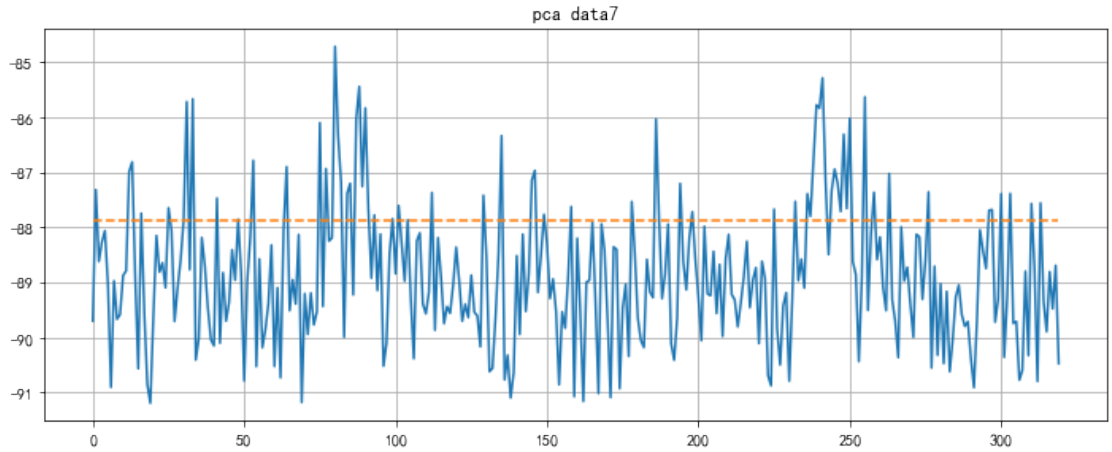

pca data4
-72.5
-75.0
-77.5
-80.0
-82.5
-85.0
-87.5
-90.0
-92.5
0
50
100
150
200
250
300
raw data4
-72.5
-75.0
-77.5
-80.0
-82.5
-85.0
-87.5
-90.0
-92.5
0
50
100
150
200
250
300

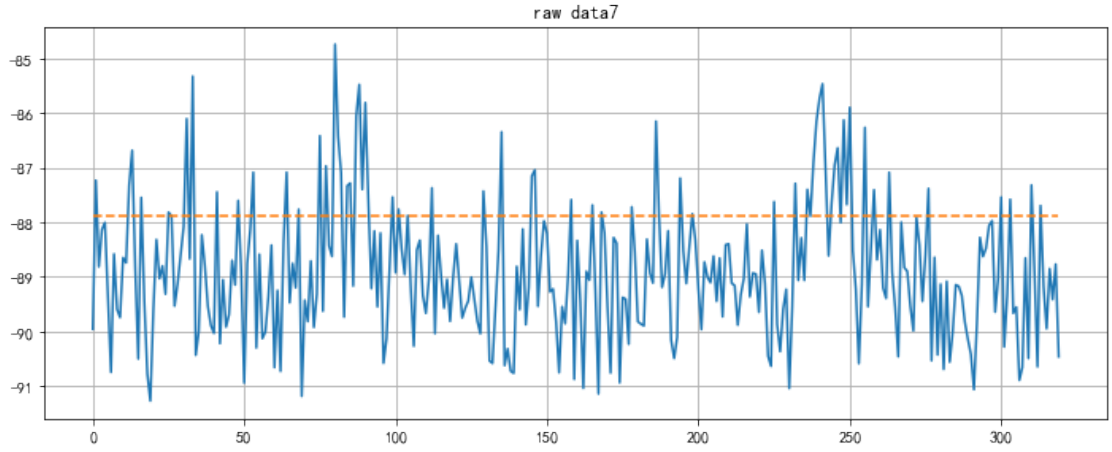

pca data5
-85
-86
-87
-88
-89
-90
-91
-92
0
50
100
150
200
250
300

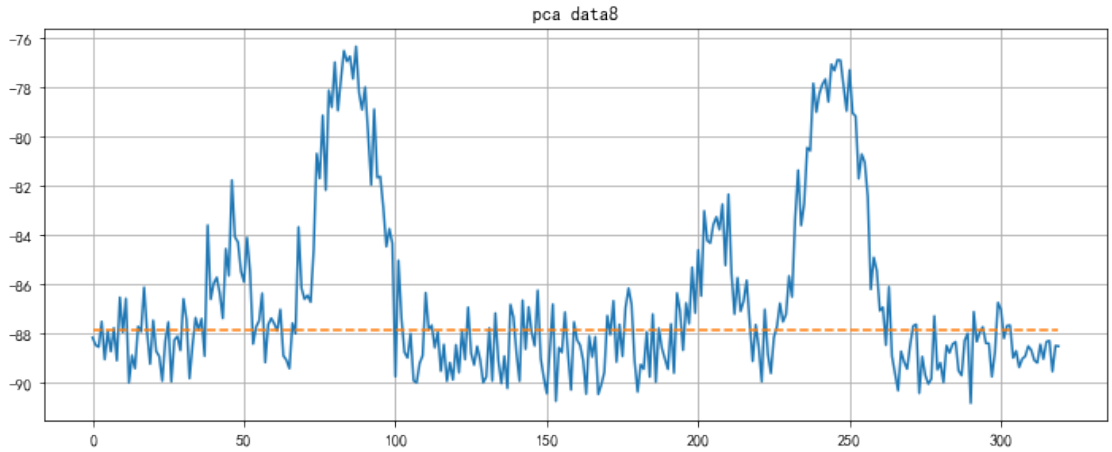

raw data5
-86
-87
-88
-89
-90
-91
-92
0
50
100
150
200
250
300

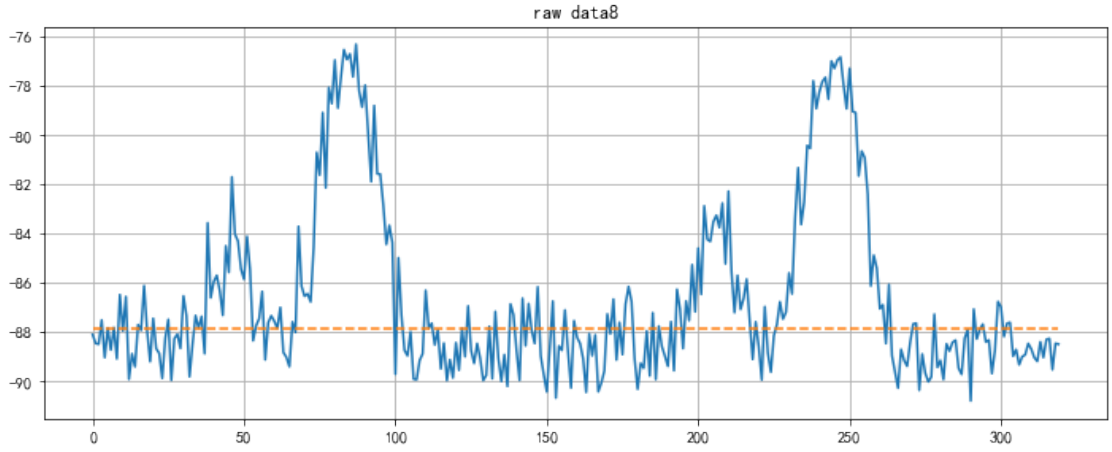

pca data6
-75.0
-77.5
-80.0
-82.5
-85.0
-87.5
-90.0
-92.5
0
50
100
150
200
250
300

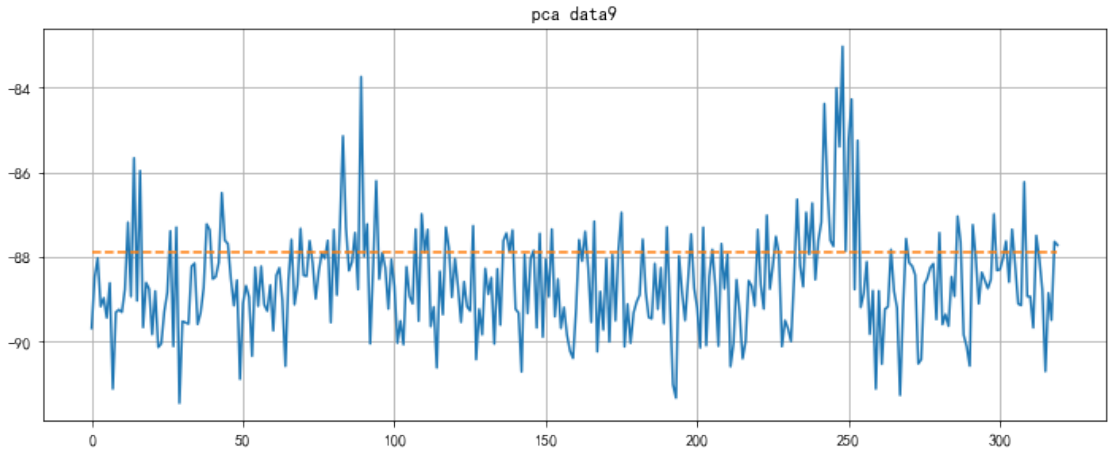

raw data6
-75.0
-77.5
-80.0
-82.5
-85.0
-87.5
-90.0
-92.5
0
50
100
150
200
250
300

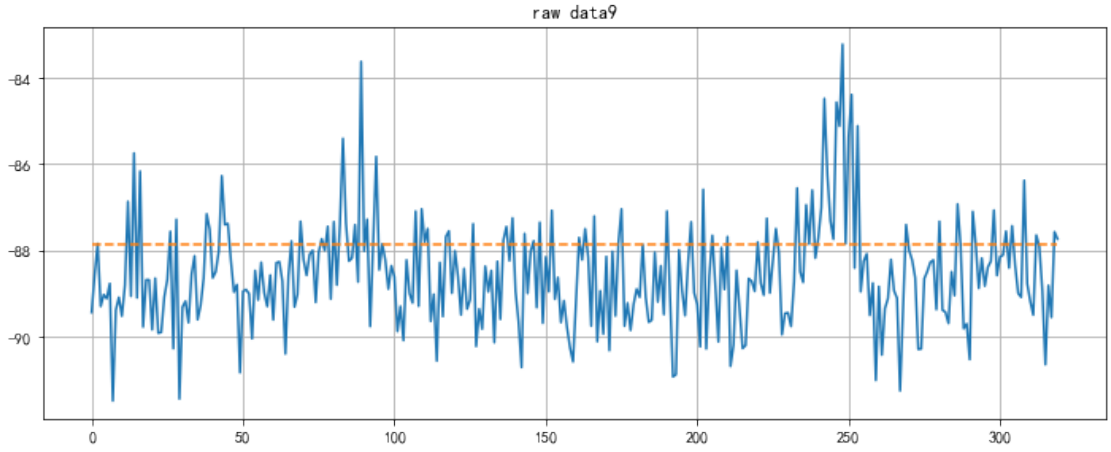

pca data7
-85
-86
-87
-88
-89
-90
-91
0
50
100
150
200
250
300

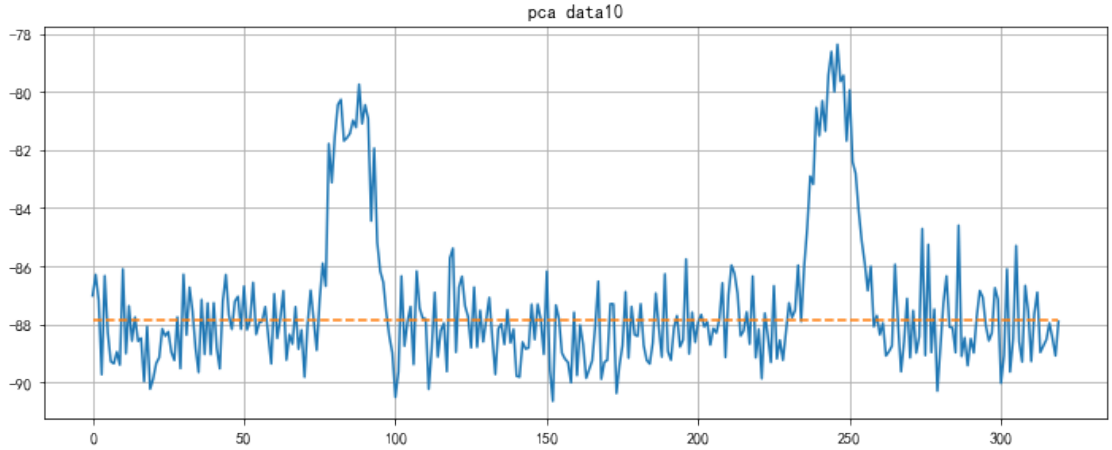

raw data7
-85
-86
-87
-88
-89
-90
-91
0
50
100
150
200
250
300

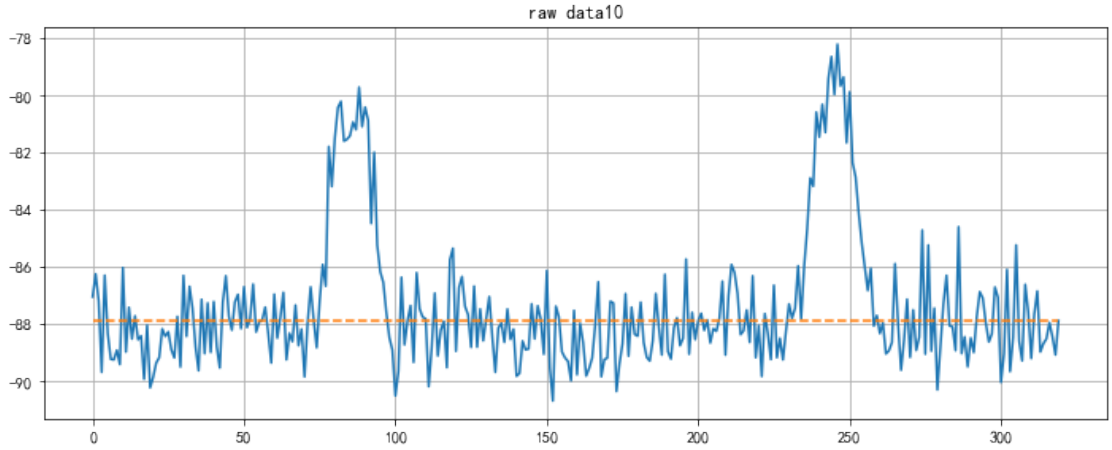

pca data8
-76
-78
-80
-82
-84
-86
-88
-90
0
50
100
150
200
250
300

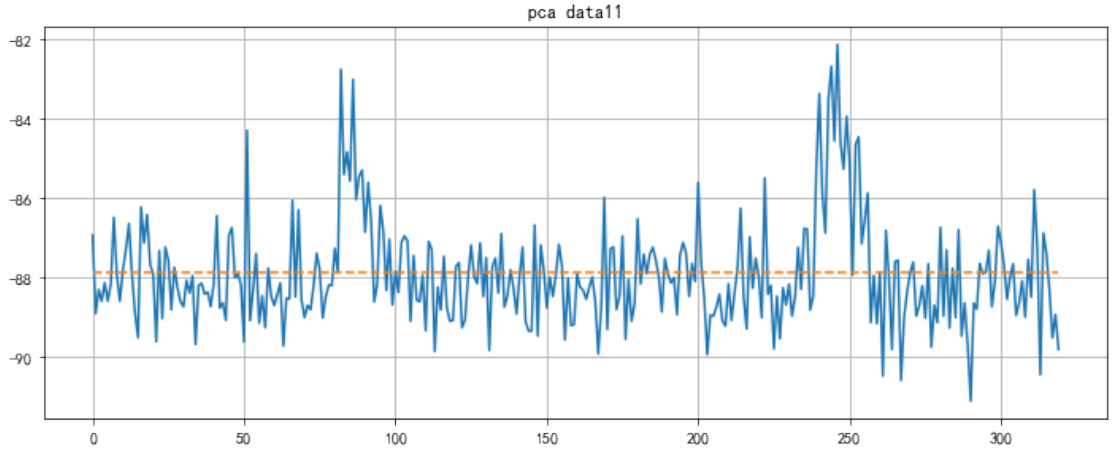

raw data8
-76
-78
-80
-82
-84
-86
-88
-90
0
50
100
150
200
250
300

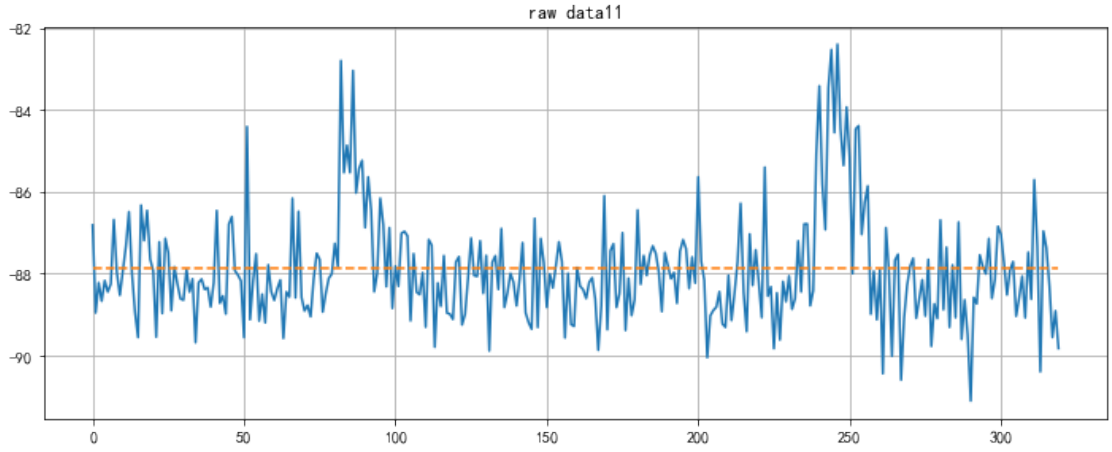

pca data9
-84
-86
-88
-90
0
50
100
150
200
250
300

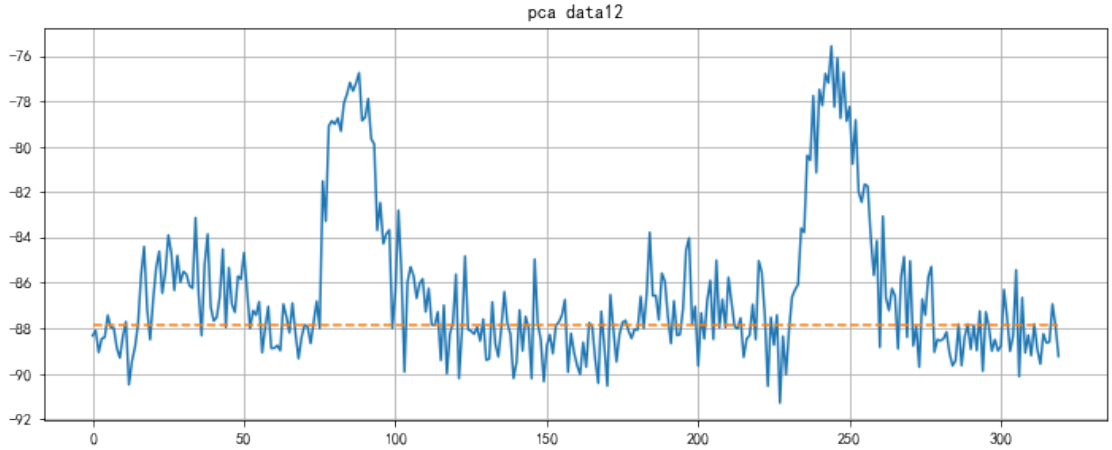

raw data9
-84
-86
-88
-90
0
50
100
150
200
250
300

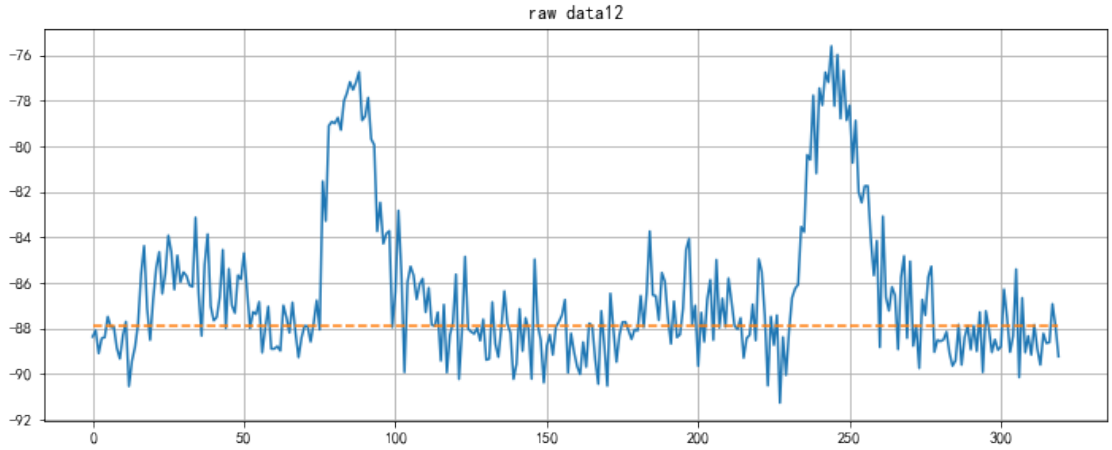

pca data10
-78
-80
-82
-84
-86
-88
-90
0
50
100
150
200
250
300

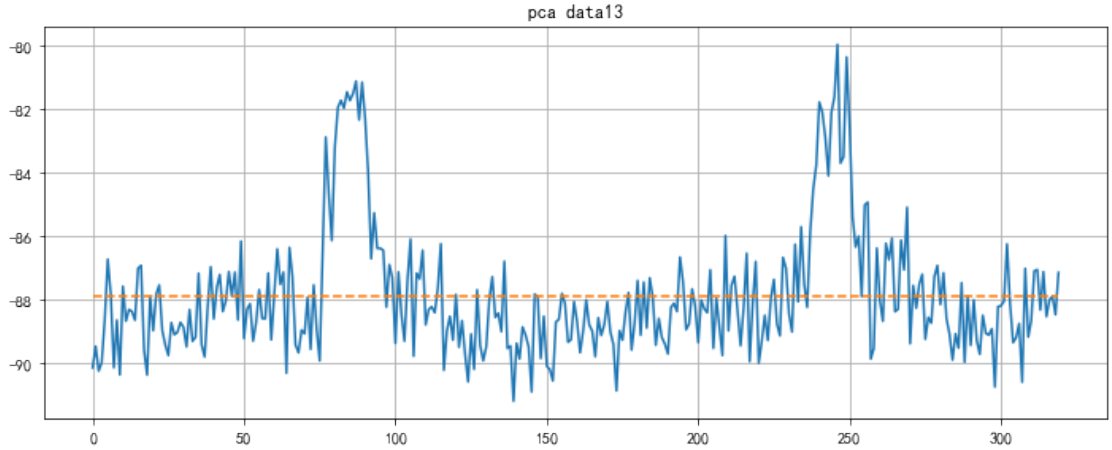

raw data10
-78
-80
-82
-84
-86
-88
-90
0
50
100
150
200
250
300

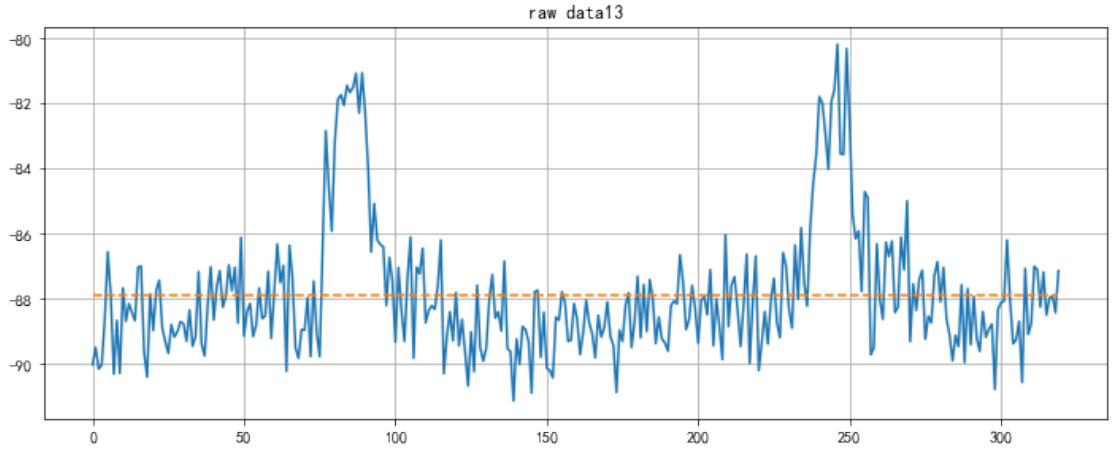

pca data11
-82
-84
-86
-88
-90
0
50
100
150
200
250
300

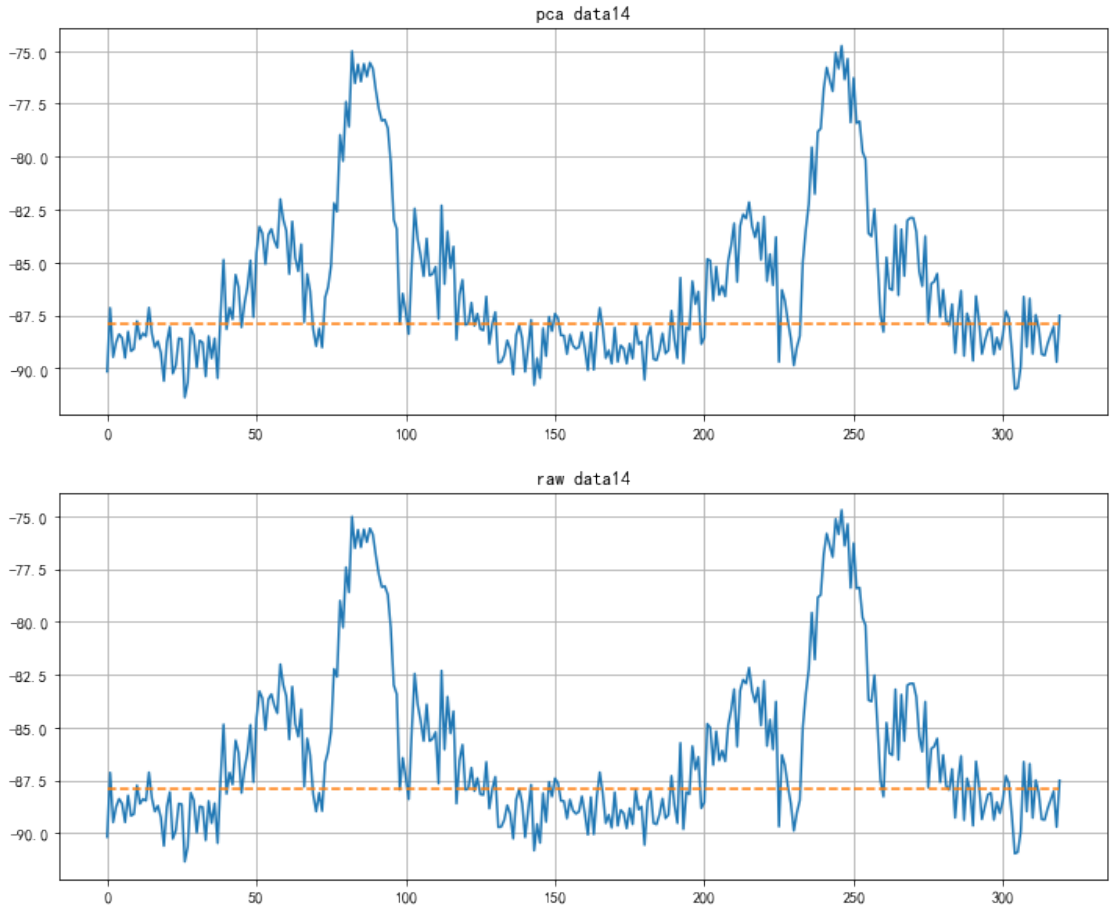

raw data11
-82
-84
-86
-88
-90
0
50
100
150
200
250
300

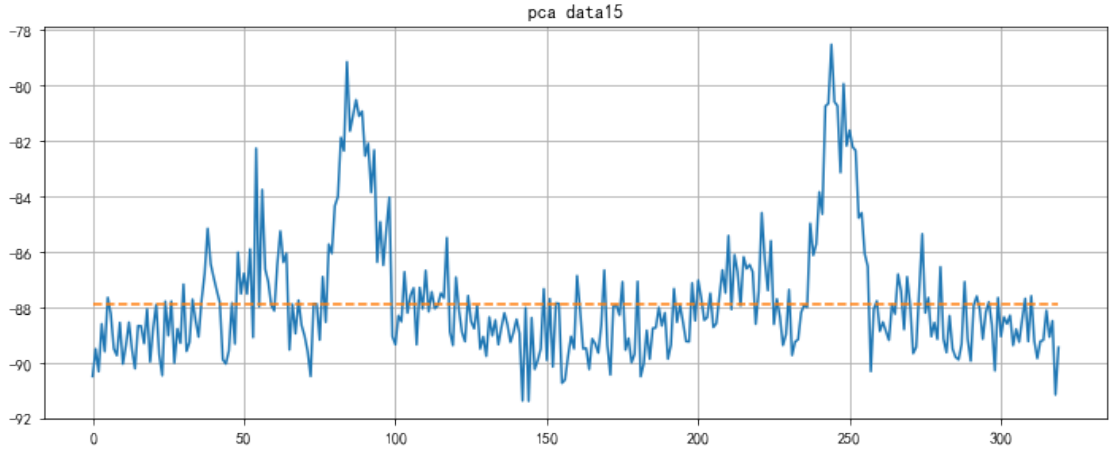

pca data12
-76
-78
-80
-82
-84
-86
-88
-90
-92
0
50
100
150
200
250
300

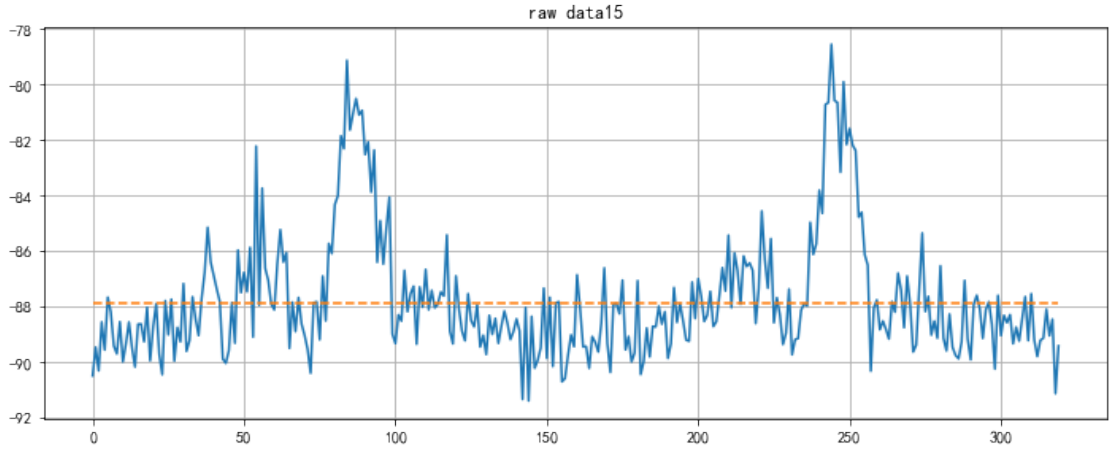

raw data12
-76
-78
-80
-82
-84
-86
-88
-90
-92
0
50
100
150
200
250
300

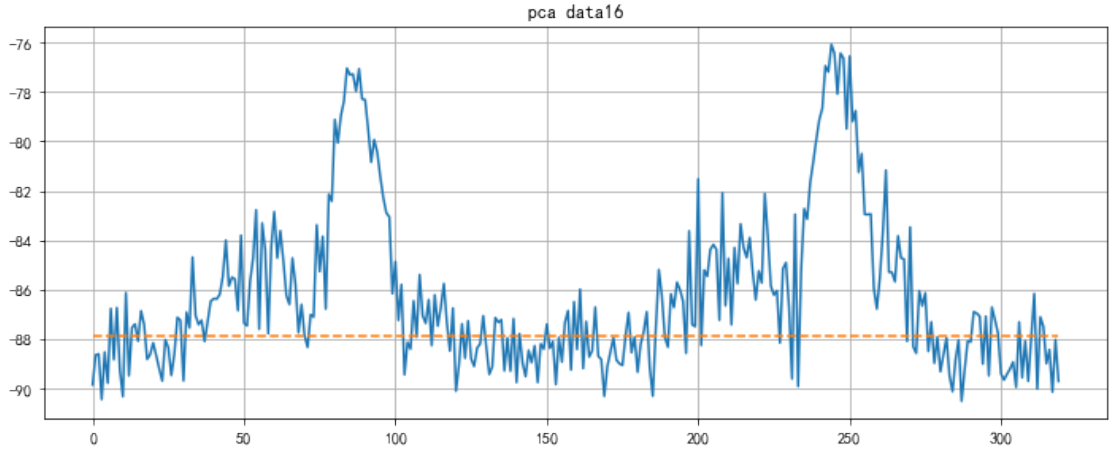

pca data13
-80
-82
-84
-86
-88
-90
0
50
100
150
200
250
300

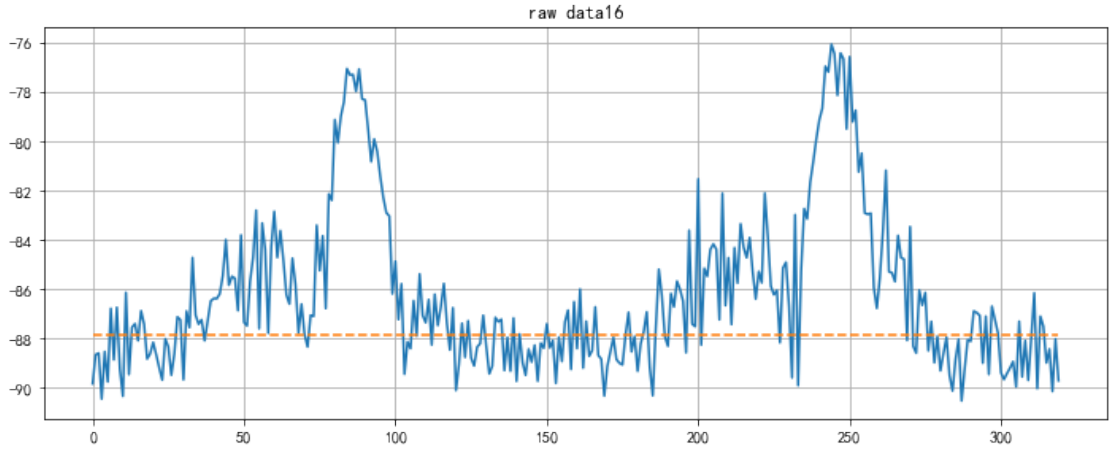

raw data13
-80
-82
-84
-86
-88
-90
0
50
100
150
200
250
300

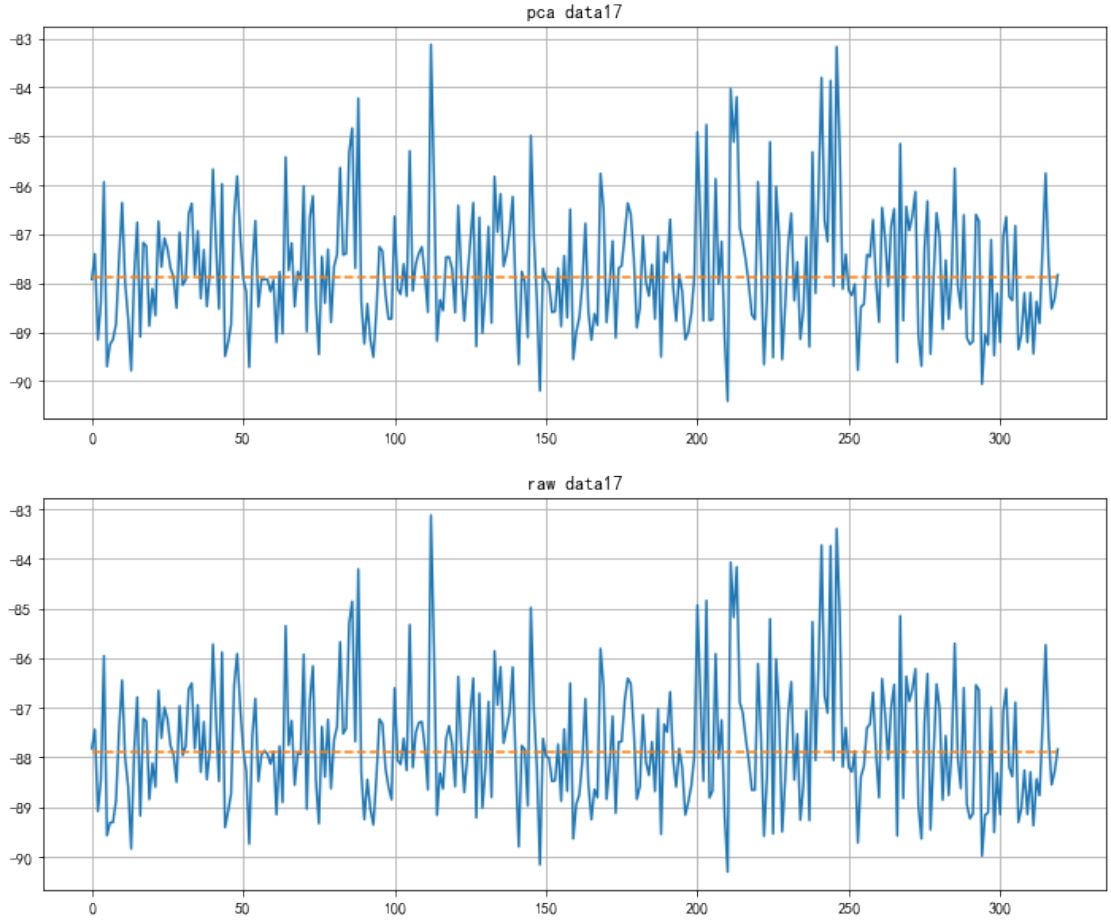

pca data14
-75.0
-77.5
-80.0
-82.5
-85.0
-87.5
-90.0
0
50
100
150
200
250
300
raw data14
-75.0
-77.5
-80.0
-82.5
-85.0
-87.5
-90.0
0
50
100
150
200
250
300


pca data15
-78
-80
-82
-84
-86
-88
-90
-92
0
50
100
150
200
250
300


raw data15
-78
-80
-82
-84
-86
-88
-90
-92
0
50
100
150
200
250
300


pca data16
-76
-78
-80
-82
-84
-86
-88
-90
0
50
100
150
200
250
300


raw data16
-76
-78
-80
-82
-84
-86
-88
-90
0
50
100
150
200
250
300




```
In [65]: from sklearn.cluster import KMeans

In [66]: clusters = 3
         kmeans = KMeans(n_clusters=clusters)

In [73]: y_means = kmeans.fit_predict(components)

In [74]: print(":\n",y_means)
         print("\n:\n",kmeans.cluster_centers_)

:
 [1 1 1 2 0 1 0 1 0 1 0 1 0 1 0 0 0 1]

:
 [[ 1.19392376e+02  1.04550155e+02 -4.21741937e+01 -6.21369056e+00
   7.32094703e+00 -6.74490832e+00 -2.39578351e+00 -6.42715522e-01
   3.62676017e+00 -4.40440807e+00 -9.82641102e-01  3.85552098e+00
  -1.53661184e+01 -5.49355279e+00  3.16247995e+00  1.23961162e+00]
 [-1.99406244e+02 -6.58469066e+01  1.76842214e+01  4.10806914e+00
```

```
  -6.88537197e+00  5.32432109e+00  7.89475509e-01  2.57116031e+00
  -1.51855606e+00  4.76530232e+00  4.59547192e-01 -3.72125190e+00
   1.35509467e+01  5.12210548e+00 -2.80134419e+00 -9.72904882e-01]
 [ 8.39517185e+02 -2.43779077e+02  1.78235557e+02  1.27369022e+01
   3.40077154e+00  6.04037679e+00  1.20609885e+01 -1.79987186e+01
  -1.53470768e+01 -7.65245630e+00  3.72520409e+00  2.64709926e+00
   9.70426557e-01 -2.15052702e+00 -8.77418533e-02 -1.16074900e+00]]
```

In [ ]:

In [117]: print(y_means)

```
[1 1 1 2 0 1 0 1 0 1 0 1 0 1 0 0 0 1]
```

In [75]: find_all_indixes(y_means,1)

Out[75]: [0, 1, 2, 5, 7, 9, 11, 13, 17]

In [76]:
```
clusters_n = []
for i in range(clusters):
    clusters_n.append(np.count_nonzero(i==y_means))
```

In [77]:
```
unit_i = freq_list_human[find_all_indixes(y_means,0)]
print(freq_list_human)
print(unit_i)
```

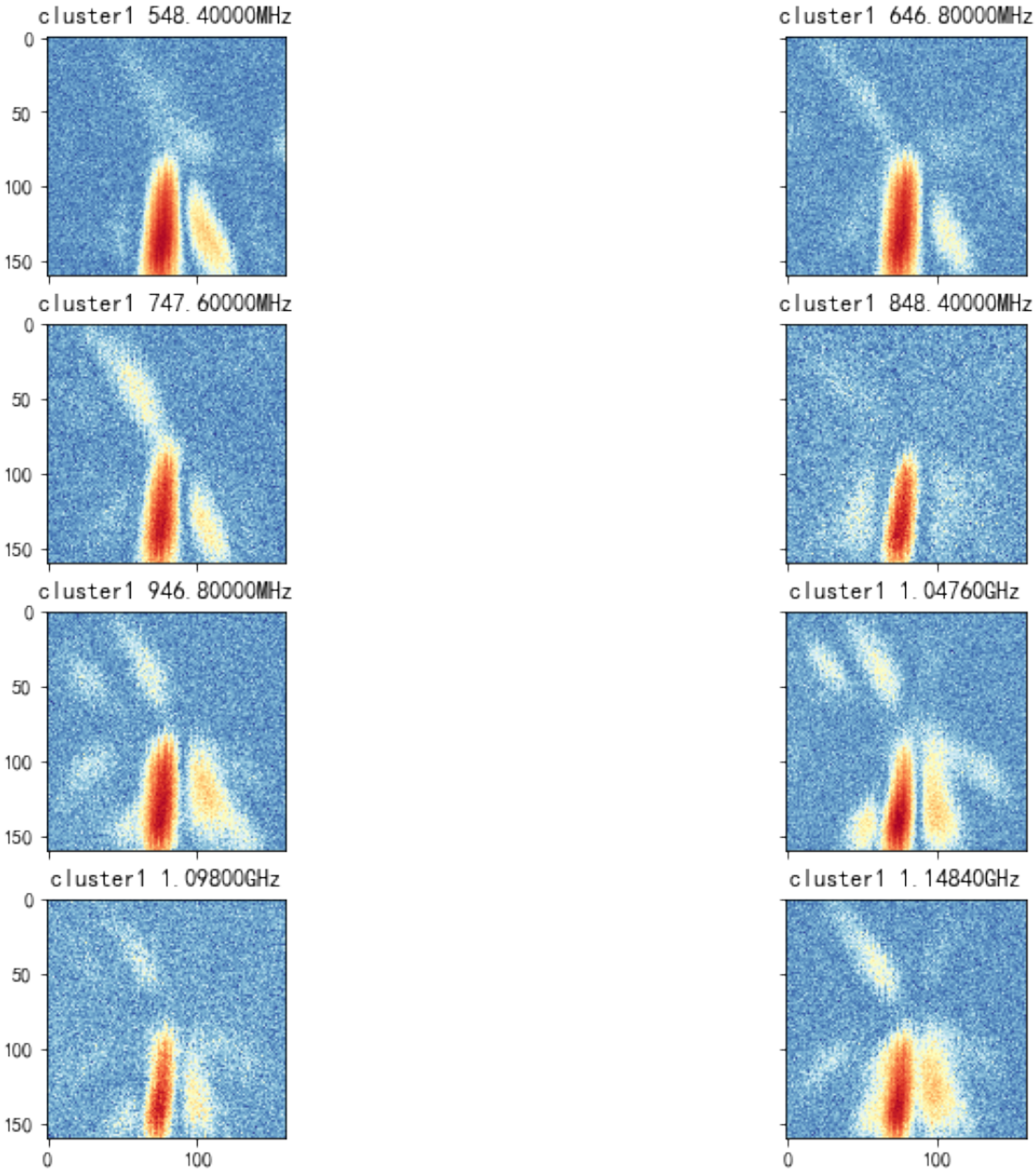
```
['147.60000MHz' '248.40000MHz' '346.80000MHz' '447.60000MHz'
 '548.40000MHz' '598.80000MHz' '646.80000MHz' '697.20000MHz'
 '747.60000MHz' '798.00000MHz' '848.40000MHz' '898.80000MHz'
 '946.80000MHz' '997.20000MHz' '1.04760GHz' '1.09800GHz' '1.14840GHz'
 '1.19880GHz']
['548.40000MHz' '646.80000MHz' '747.60000MHz' '848.40000MHz'
 '946.80000MHz' '1.04760GHz' '1.09800GHz' '1.14840GHz']
```

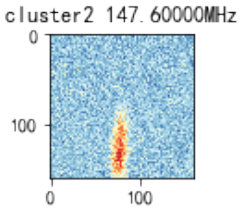
In [78]: clusters_n

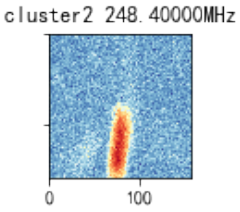
Out[78]: [8, 9, 1]

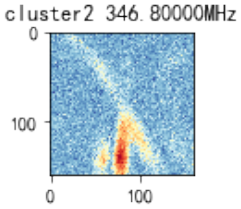
In [79]:
```
axes_list = []
for i in clusters_n:
    if i > 1:
        if i % 2:
            fig, axes = plt.subplots(i//2+1,2,sharey=True,sharex=False)

            axes = axes.reshape(-1)
```

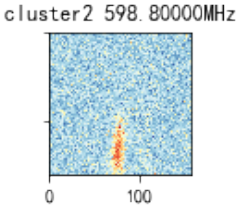
```
                #fig.delaxes(axes[len(axes)-1])
                #plt.draw()
                axes[len(axes)-1].axis('off')
                plt.subplots_adjust(wspace=0.3, hspace=0.5)

            else:
                fig, axes = plt.subplots(int(i/2),2, sharey=True, sharex=True)

        else:
            fig, axes = plt.subplots()


        axes_list.append(axes)

    for it in axes_list:
        for i in range(clusters):

            data_i = data_extract_by_std[find_all_indixes(y_means,i)]
            unit_i = freq_list_human[find_all_indixes(y_means,i)]
            j = 0
            for d in data_i:

                if isinstance(axes_list[i], np.ndarray):

                    ax = axes_list[i].flatten()
                    #ax.set_title('cluster ' + str(i+1))
                    #print(type(ax[j]))
                    #print(data_i.shape)
                    ax[j].imshow(np.rot90(d,2), cmap=plt.cm.RdYlBu_r)
                    ax[j].set_title('cluster' + str(i+1) + " " + unit_i[j])
                    j = j + 1

                else:
                    ax = axes_list[i]
                    ax.imshow(np.rot90(d, 90), cmap=plt.cm.RdYlBu_r)
                    ax.set_title('cluster' + str(i+1) + " " + unit_i[0])
```

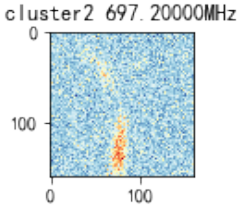

cluster1 548.40000MHz
cluster1 646.80000MHz
cluster1 747.60000MHz
cluster1 848.40000MHz
cluster1 946.80000MHz
cluster1 1.04760GHz
cluster1 1.09800GHz
cluster1 1.14840GHz
0
50
100
150
0
100

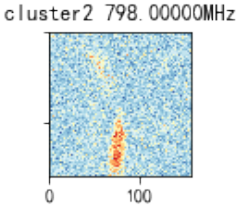

cluster2 147.60000MHz
0
100
0
100

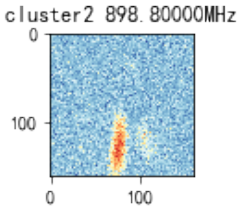

cluster2 248.40000MHz
0
100

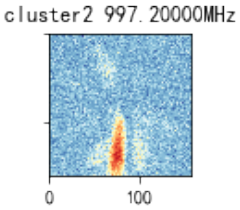

cluster2 346.80000MHz
0
100
0
100

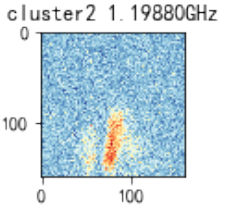

cluster2 598.80000MHz
0
100

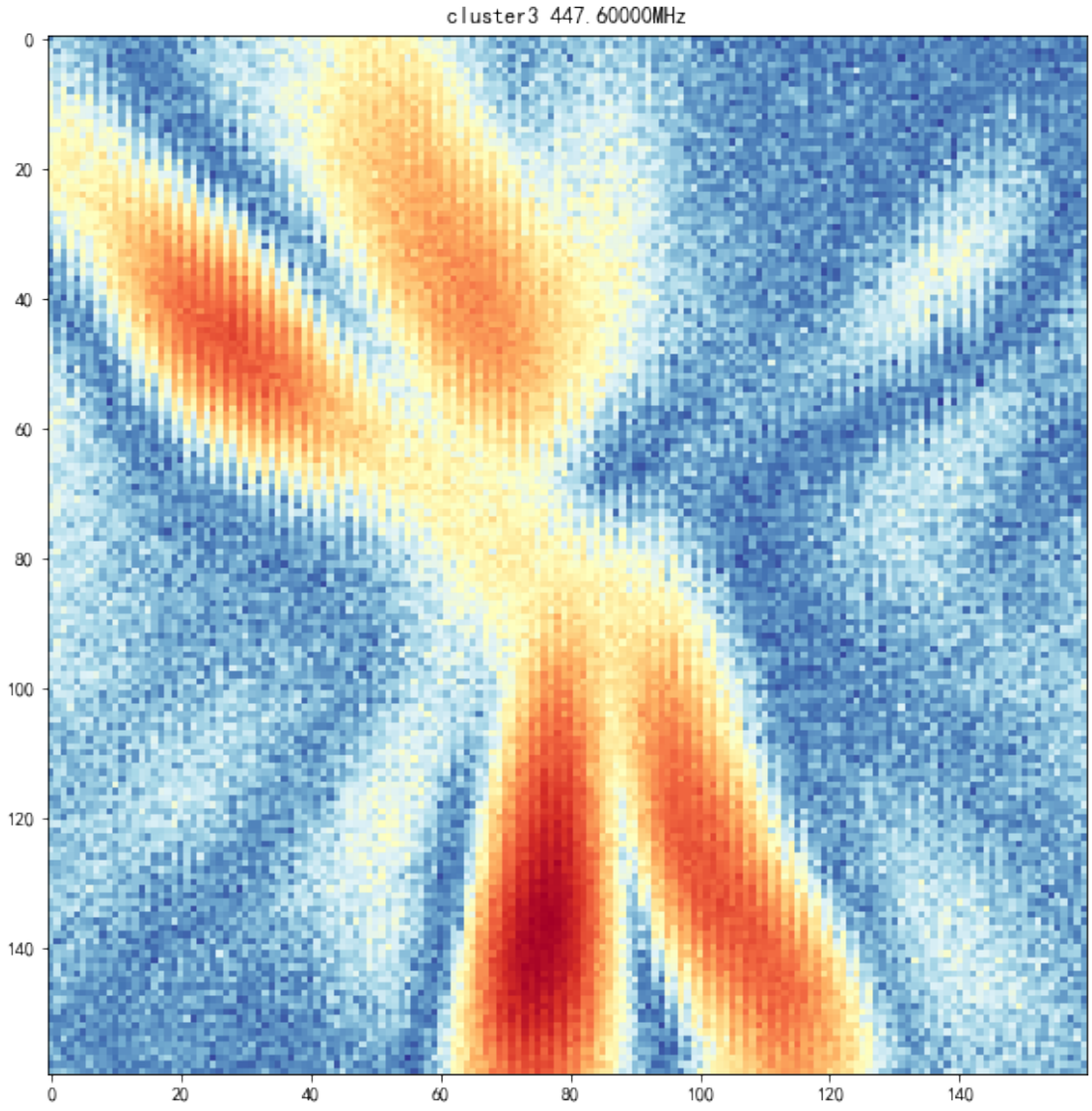

cluster2 697.20000MHz
0
100
0
100


cluster2 798.00000MHz
0
100

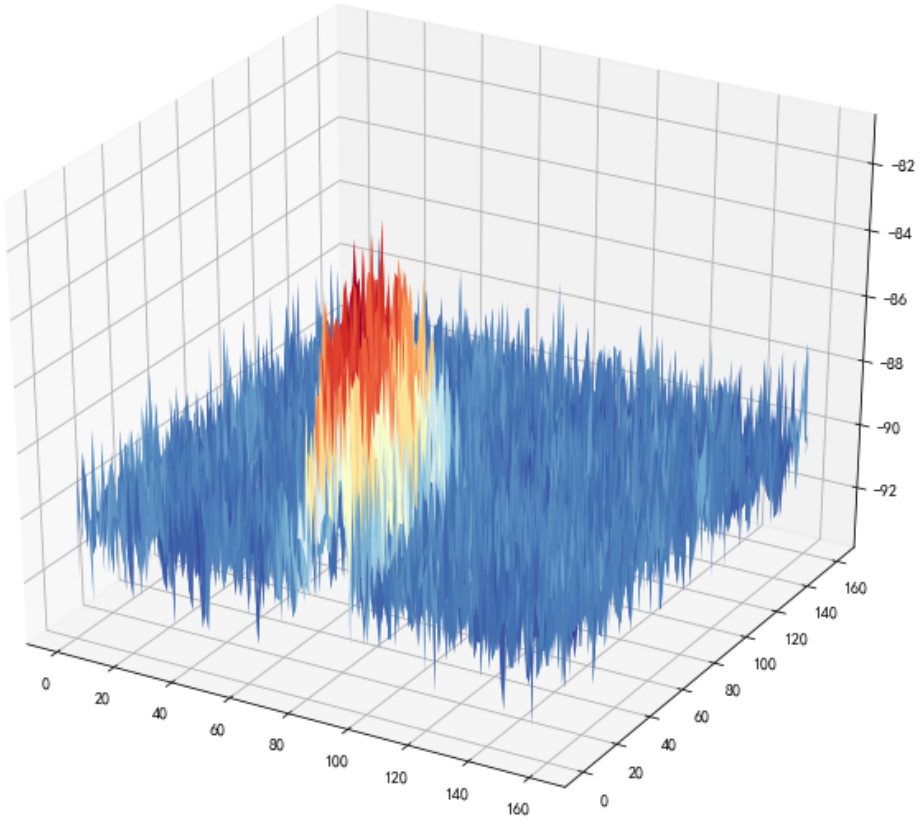

cluster2 898.80000MHz
0
100
0
100


cluster2 997.20000MHz
0
100


cluster2 1.19880GHz
0
100
0
100




```
In [116]: np.save('data_extract_by_std.npy', data_extract_by_std)

In [161]: x_len = np.arange(1, 161, 1)
          y_len = np.arange(1, 161, 1)
          xx, yy = np.meshgrid(x_len, y_len)

In [108]: from mpl_toolkits import mplot3d

In [163]: plt.figure('plot_surf')
          ax_surf = plt.axes(projection='3d')
          ax_surf.plot_surface(xx, yy, data_extract_by_std[0],cmap=plt.cm.RdYlBu_r)

Out[163]: <mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x21d8fa165f8>
```




```
In [111]:
```

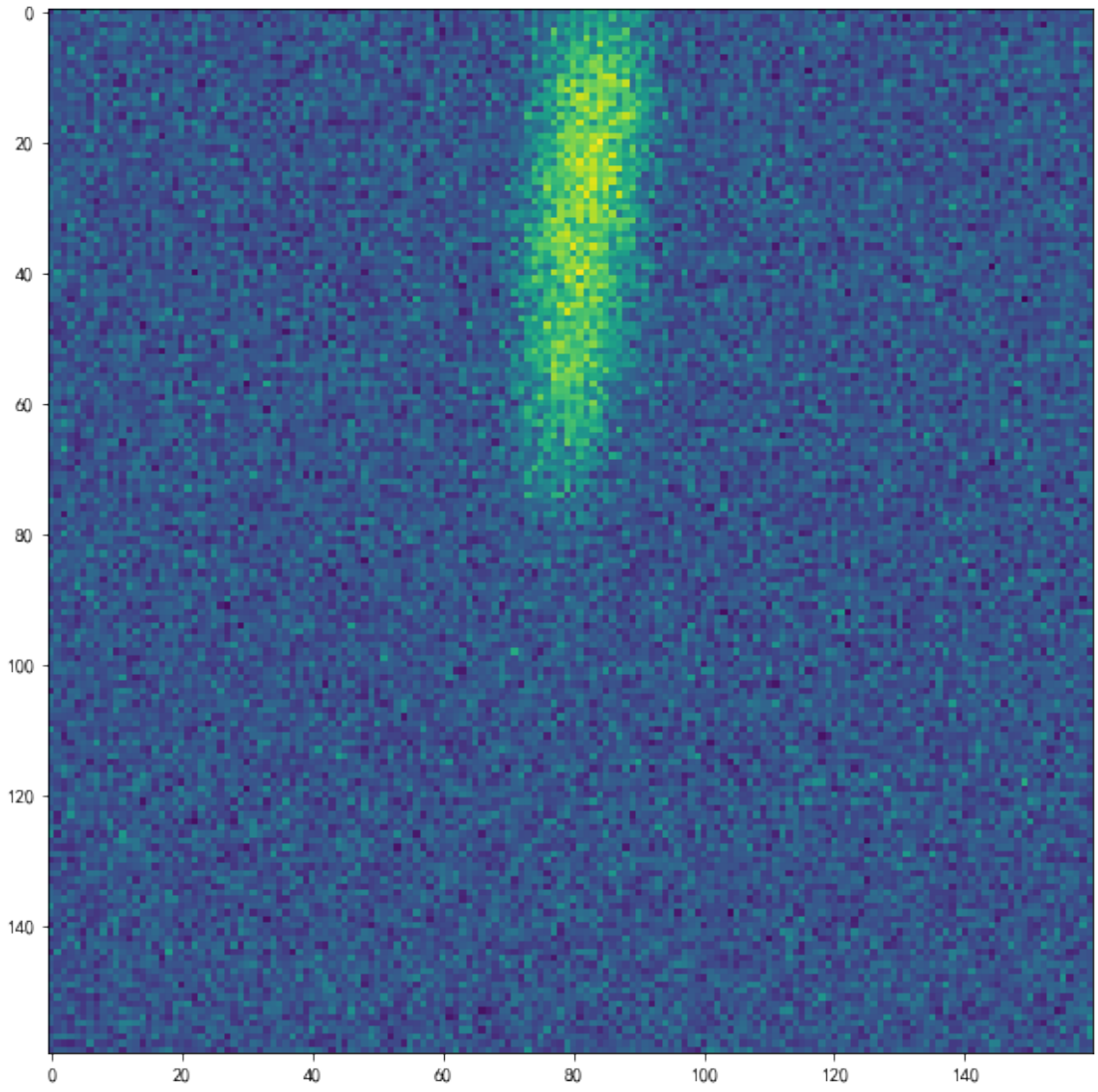
```
Out[111]: <mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x21d6d78da58>
```

```
In [113]: plt.draw()
```

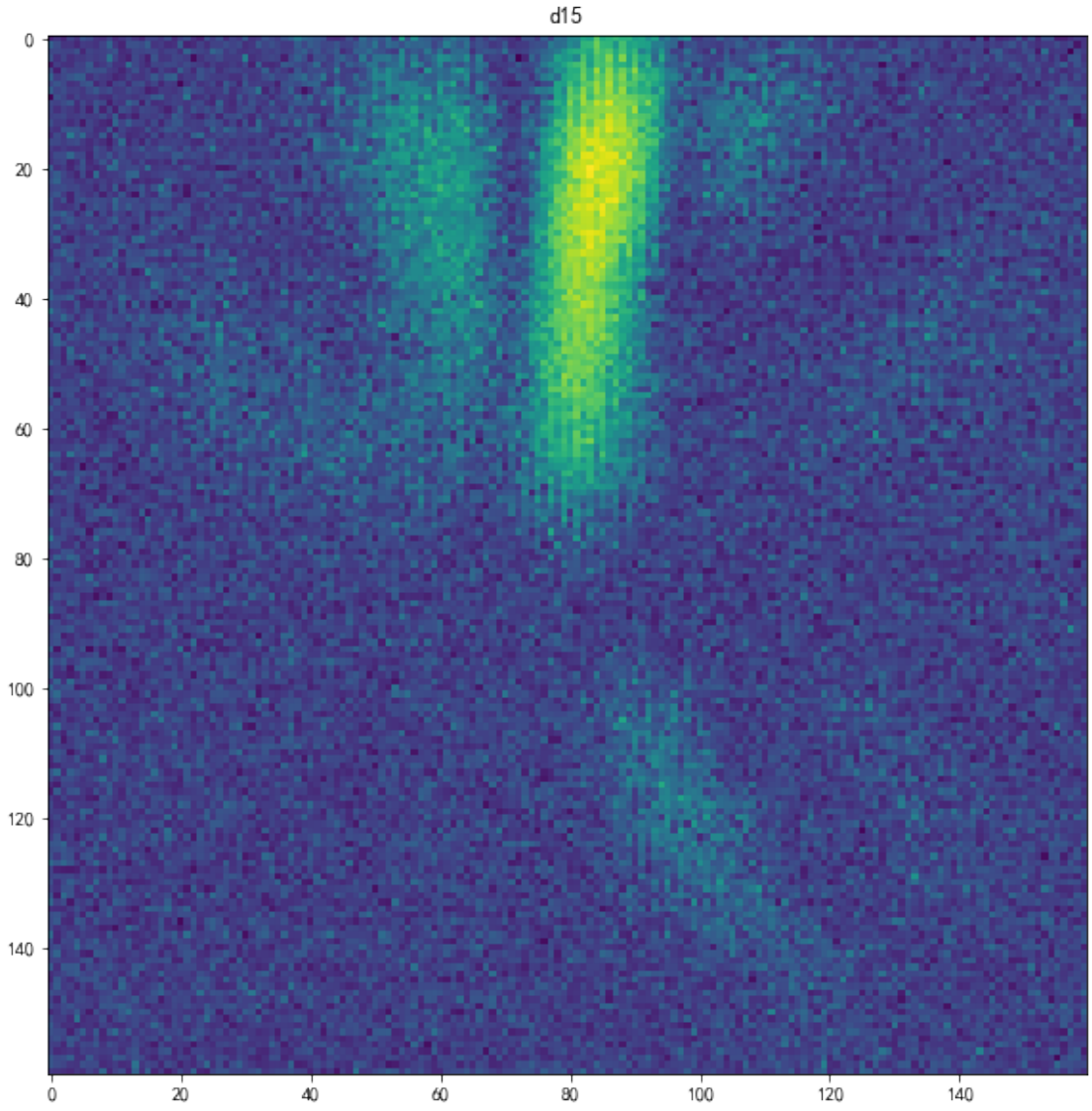
```
<Figure size 864x720 with 0 Axes>
```

```
In [99]: fig, ax = plt.subplots()
         ax.imshow(data_extract_by_std[0])
```

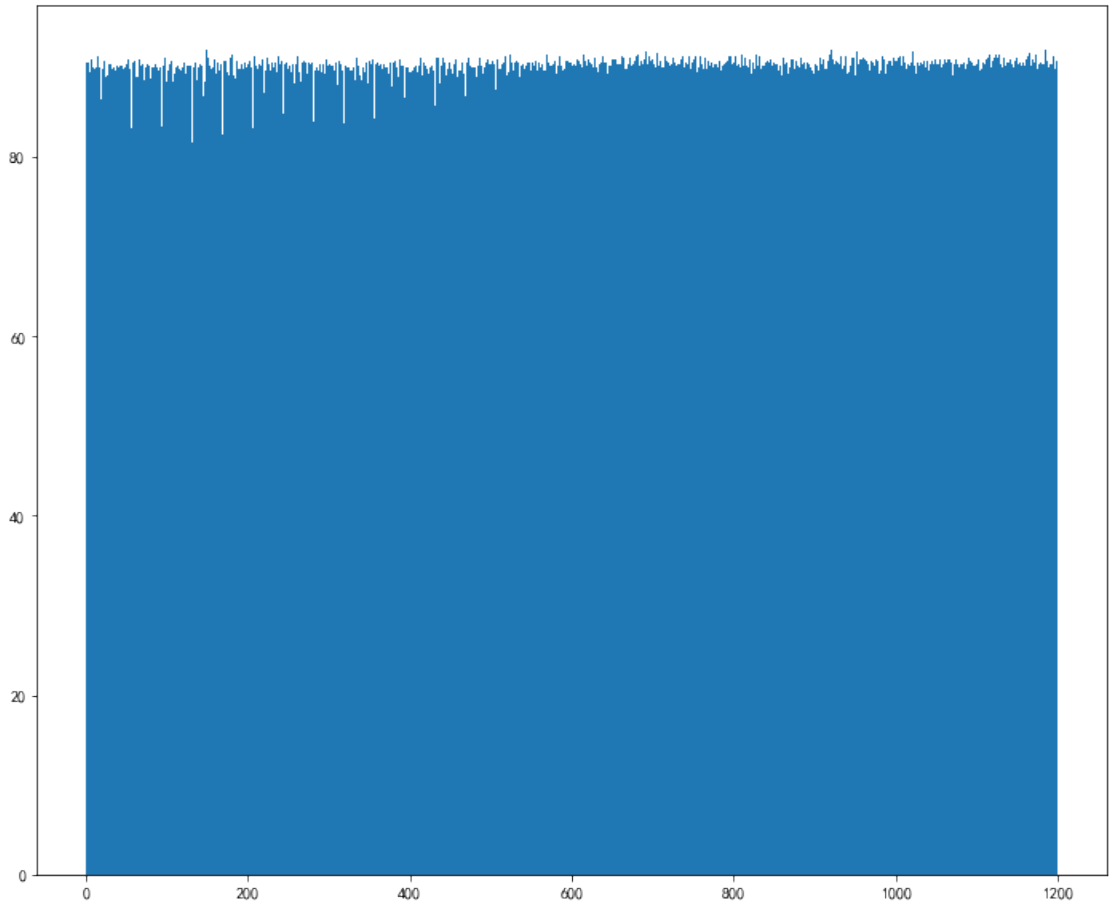
```
Out[99]: <matplotlib.image.AxesImage at 0x21d3ee89630>
```




```
In [100]: d15 = data_extract_by_std[15]

In [128]: np.save?

In [102]: fig, ax = plt.subplots()
          ax.imshow(d15)
          ax.set_title('d15')

Out[102]: Text(0.5, 1.0, 'd15')
```




```
In [103]: d15.std(axis=0).std()

Out[103]: 1.0190820898302566

In [104]: d15_flatten = -1 * d15.flatten()
          d15_flatten.shape

Out[104]: (25600,)

In [105]: plt.subplots()
          x = np.linspace(0,1200,25600)
          plt.bar(x,d15_flatten)

Out[105]: <BarContainer object of 25600 artists>
```




```
In [312]: y_mean_list.index(0)
```

```
Out[312]: 0
```

$$C = \frac{m_f - m_e}{\rho_\theta}$$

$$u_c^2 = u_A^2(C) + c_{mf}^2 u^2(m_f) + c_{me}^2 u^2(m_e) + c_{\rho\theta}^2 u^2(\rho_\theta)$$

```
In [1]: np.array(data) # data shape:2*3*4
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)

<ipython-input-1-5605209407eb> in <module>
----> 1 np.array(data) # data shape:2*3*4


NameError: name 'np' is not defined
```

In [ ]: