In [1]:

```
# -*- coding: utf-8 -*-
import numpy as np
from matplotlib import pyplot as plt

coast_txt = '70.38404228477067   70.7377611871728        71.32808403652885       71.9254268414118
5       72.43099813907867       72.96869540983428       73.3793987506353        73.8119201361614
6       74.32262280726555       74.72937227677205       75.08066243770631       75.1539717700931
6       75.53011168151919       76.35030999779465       76.76246377524616       77.3635566329568
1       77.80697908530104       78.38205850985814       79.09411804409736       79.6973061733228
80.15458096023976       80.74112089061165       81.28327385947648       81.70438530404381
82.27054465330004       82.6466119980671        83.08766874448281       83.58844182040787
83.89039209518833       84.16744909425032       84.37200712647908       84.65991516037518
85.01657363559819       85.29093762529308       85.50255936850775       85.82651139553067
86.32129142369452       86.9178075729872        87.44985514390359       87.68827993977864
87.50944877597189       86.85660089538312       86.02312693533503       85.45620876716893
85.18378096491047       84.76964535297101       84.22365944088735       83.92894360096358
83.66721946723473       83.13907427085121       82.64981371486574       81.9404809877423
80.71861318538976       78.93923643649502       77.39444495574747       75.65531753958001
74.3326764061046        73.41608739730432       72.3571507603265        70.63490444533123
68.46765618906382       67.03766288189601       65.749604737688 62.21453334766436       54.72598
465745872       47.41922489341726       44.657566200963586      43.07888952723648       40.75756
525322306       38.648600402789 36.12681357901546       34.02368728092076       32.5728853226139
56      31.20502950564515       29.716036692713704      28.397742470504355      27.1081132173929
7       25.637146132584554      23.8552988054435        21.664833441206305      19.3045363248047
86      17.3305825899181        15.924253062349187      14.796313905087509      13.7526149970476
05      12.717955969028203      11.763838371520052      10.986199423823084      10.1440514670539
95      9.498045898631984       9.174501353960842       8.876558925266258       8.66802961803697
3       8.430552097442087       7.894684101440229       6.927507951036395       5.92149661744368
3       5.242957105134755       4.470518656244107       3.8938951672206032      3.16965159569514
74      2.3507061646098317      1.547042568272398       1.0160962004570568      0.78813523263249
03      0.6003043482870623      0.3648097250536013      0.17259603625035566     0.04495718065465
587     9.030771920836081e-05   0.029606438138634 48      0.035692173713357686    -0.0118601254308
07352   -0.0549483463342937 54   -0.07376403964806516    -0.05175412733404933    0.11795248314609
516     0.5371045981319293      1.0159611566880025      1.2579563250125223      1.37969534837598
7       1.589623178351121       2.017580203325206       2.405471812761733       2.61176125680998
8       2.8298286321454005      3.0103609663882795      3.1426103397983187      3.24015406065903
47      3.293570218456984       3.301726382034415       3.2865746036953807      3.26485913791940
83      3.249066590415582       3.2459909185143263      3.2543533502634494      3.29341285375709
7       3.3900857132373954      3.49547440703077        3.529746172601694       3.62285484579296
7       3.8618139183563507      3.9536789760133746      4.0313185892806365      4.21257873134108
8       4.310618719719117       4.417329106393094       4.515260422603292       4.61118564585243
4.714054468578936       4.827455810016113       4.946616685911019       5.059863595381696
5.150596601155944       5.210065549855707       5.243150575236788       5.258269755722903
5.265212707232154       5.268204946494954       5.259349613201016       5.229346198756556
5.17118622383863        5.0838557203962935      4.978333936931056       4.8694580917182 4.766211
768662836       4.650711213634316       4.482406571833632       4.273484685330637       4.095167
38673907        3.960168178916943       3.8315637896337558      3.7218010901347625      3.676070
6901077853      3.6960865019605693      3.7577737719179254      3.8665537519211344      4.038078
606864209       4.234657898082548       4.423254326215453       4.793185258302106       5.404401
21573615        6.067153881328951       7.006337661057961       7.833946482364835       8.792783
318341707       9.620243786035882       10.380051184540983      11.33881213503209       11.73913
629401915       11.733141192442499      11.61530040058059       11.619365692453778      11.88305
0476915109      12.484005214765544      13.122592642097201      13.468171604154985      13.77704
3862750505      13.85362199494957       13.644655518304688      16.66707780910795       23.75056
847847022       25.435906828290445      25.998505876031206      26.169781613861122      26.48902
950455903       27.328807694148278      27.805256317100536      27.917058097965203      27.97705
1808755323      28.19705290184548       28.312196128851127      28.513323681574043      28.99311
317304667       29.556693268097902      30.43393459700344       32.31657394133081       34.55704
958412306       37.040746482935035      38.24167950622797       38.42831644460377       38.57211
```

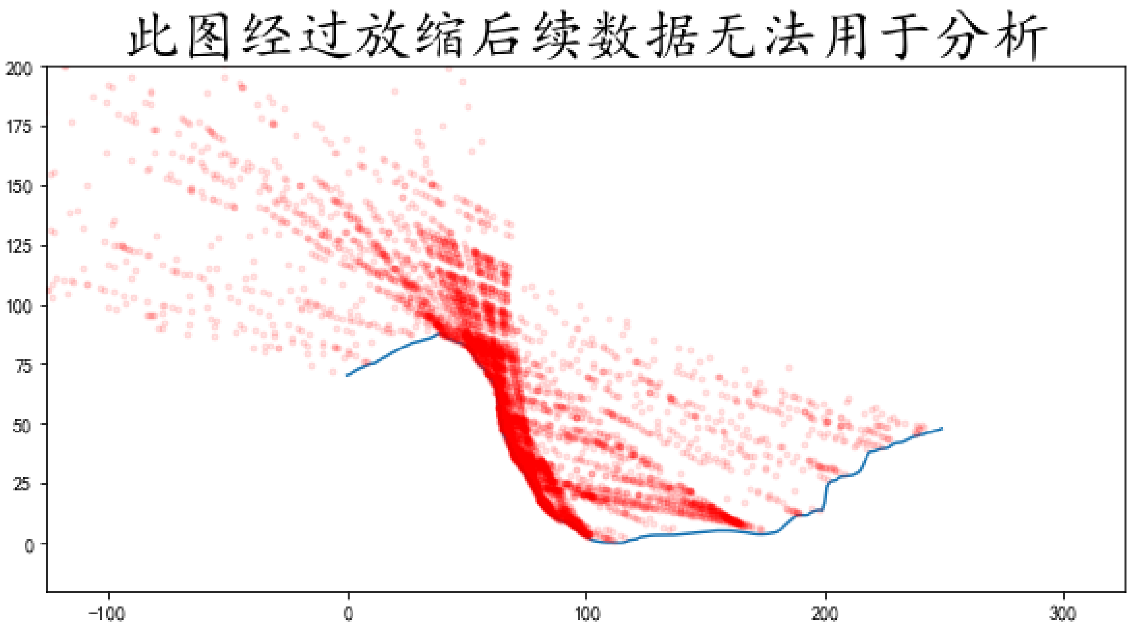
```
0116539896      38.988788846575744      39.41290827018334       39.6413638295544        39.75634
330151359       39.834941340086786      40.30627003193701       41.12662692259052       41.72371
314445744       41.94029601531473       42.034528232968704      42.161225669769514      42.45426
8759386636      42.929278312776006      43.37304606905392       43.859714337274205      44.42338
073358562       44.65987467711081       44.9814993281134        45.38402821385245       45.40830
645267432       45.62169607624446       46.22335016155107       46.29582421752576       46.50514
447377482       46.91765537366781       47.10284902343331       47.394856140890155      47.89489
4568911'
coast = np.array(coast_txt.split(),dtype = np.float).ravel()
x = np.arange(coast.size)
coast_x = x

def get_value(site,site1):
        return np.argmin(np.abs(site-site1))

def norm_v(v1_temp):
        '''单位化'''
        return v1_temp/np.sqrt((v1_temp**2).sum())

def get_euc(x1,x2):
        return np.sqrt(((x1-x2)**2).sum())

def filt_points(points,distance = 'each',f = np.median):
        '''基于3sigma准则进行滤去异常点
        distance :
        euc :   基于欧式距离进行过滤
        each:   基于每一维，有一维不正常即不正常，未编写'''
        if distance  == 'euc':
                center = np.mean(points,axis = 0)
                dis_ls = [get_euc(i,center) for i in points]
                dis_mean = f(dis_ls)
                dis_std = np.std(dis_ls)
                site_anormal = []
                site_abnormal = []
                for i in range(points.shape[0]):
                        if  (dis_ls[i]-dis_mean)<1*dis_std:
                                site_anormal.append(i)
                        else:
                                site_abnormal.append(i)
                return points[site_anormal,:],points[site_abnormal,:]
        elif distance == 'each':
                center = np.mean(points,axis = 0)
                std = np.std(points,axis = 0 )
                site_anormal = []
                site_abnormal = []
                for i in range(points.shape[0]):
                        if ((points[i][0]-center[0])<1*std[0] and (points[i][1]-center[1])<1*st
d[1]):
                                site_anormal.append(i)
                        else:
                                site_abnormal.append(i)
                return points[site_anormal,:],points[site_abnormal,:]

def interp_for_under_coast(x_i,coast_i,x3,y3):
        range_31 = sorted([x_i,x3])
        range_31 = np.arange(np.ceil(range_31[0]),np.floor(range_31[1]))
        x_interp_31 = np.array([x_i,x3])
        y_interp_31 = np.array([coast_i,y3])
        y_interp_31 = y_interp_31[np.argsort(x_interp_31)]
        x_interp_31 = x_interp_31[np.argsort(x_interp_31)]
        interp_31 = np.interp(range_31,x_interp_31,y_interp_31)
        return range_31,interp_31

def cross_esi(direction = '东北风'):
        '''cross east sea island'''
        damp = 1

        tan = (coast[1:]-coast[:-1])/(x[1:]-x[:-1]+1e-16)#海岸线切线正切值（无问题）

        if direction == '东北风':
                vx = -np.ones(x.shape)
                vy = vx
        elif direction == '东南风':
                vx = -np.ones(x.shape)
                vy = np.ones(x.shape)
        elif direction == '西北风':
                vx =  np.ones(x.shape)
                vy = -np.ones(x.shape)
        elif direction == '西南风':
                vx = vy = np.ones(x.shape)
        else:
                print('您输入的风向暂时不支持')
        #截取至方便计算
        vx = vx[:-1]
        vy = vy[:-1]

        v = (vx**2+vy**2)**0.5#算出合速度大小
        theta_w1 = np.arctan(vy/(vx+1e-16))#水与横轴的夹角
        minus_pi_site = np.where((np.array(vx)<0) & (np.array(vy)<0))
        theta_w1[minus_pi_site] -= np.pi
        plus_pi_site = np.where((np.array(vx)<0) & (np.array(vy)>0))
        theta_w1[plus_pi_site] += np.pi

        theta_c = np.arctan(tan)#海岸线与横轴的夹角
        theta_temp = 0.25*np.pi+theta_c-theta_w1#反射角#即使在sin与45°水流例子中依然成立（theta_c-theta_w)<0+0.5pi....result>0

        theta_c = np.arctan(tan)
        v1 = np.array((np.cos(theta_c),np.sin(theta_c))).T
        v2 = np.array((np.cos(theta_c+0.5*np.pi),np.sin(theta_c+0.5*np.pi))).T
        v3 = np.array((vx,vy)).T
        theta_13 = []
        for i in range(v3.shape[0]):
                theta_13_t=np.arccos(np.dot(v1[i],v3[i])/np.sqrt((v1[i]**2).sum()+(v3[i]**2).sum()))
                theta_13.append(theta_13_t)
                v1_temp=-v1[i]#v1反方向向量
                v1_temp=v1_temp/np.sqrt((v1_temp**2).sum()) #v1_temp是v1反向量的单位化
                v2_temp = norm_v(v2[i])#v2单位化
                v3_temp=v3[i]/np.sqrt((v3[i]**2).sum())#v3单位化
                if (v1[i][0]>=0 and v1[i][1]>=0):
                        #if(v3[i][0]<=0 and v3[i][1]>=0):#   =是否能取到，何时取到，先凭感觉的，都取并没有影响
                        if -1<=v3_temp[0]<=v2_temp[0] and v3_temp[1]>=0:
                                theta_w2 = theta_c[i]-theta_13
                        if -1<=v3_temp[0]<=v1_temp[0] and v3_temp[1]<=0:
                                theta_w2 = theta_c[i]-theta_13
                        if v1_temp[0]<=v3_temp[0]<=1 and v3_temp[1]<=0:
                                theta_w2 = theta_c[i]+theta_13
                        if -v1_temp[0]<=v3_temp[0]<=1 and v3_temp[1]>=0:
                                theta_w2 = theta_c[i]+theta_13
                        if v2_temp[0]<=v3_temp[0]<=-v1_temp[0] and v3_temp[1]>=0:
```

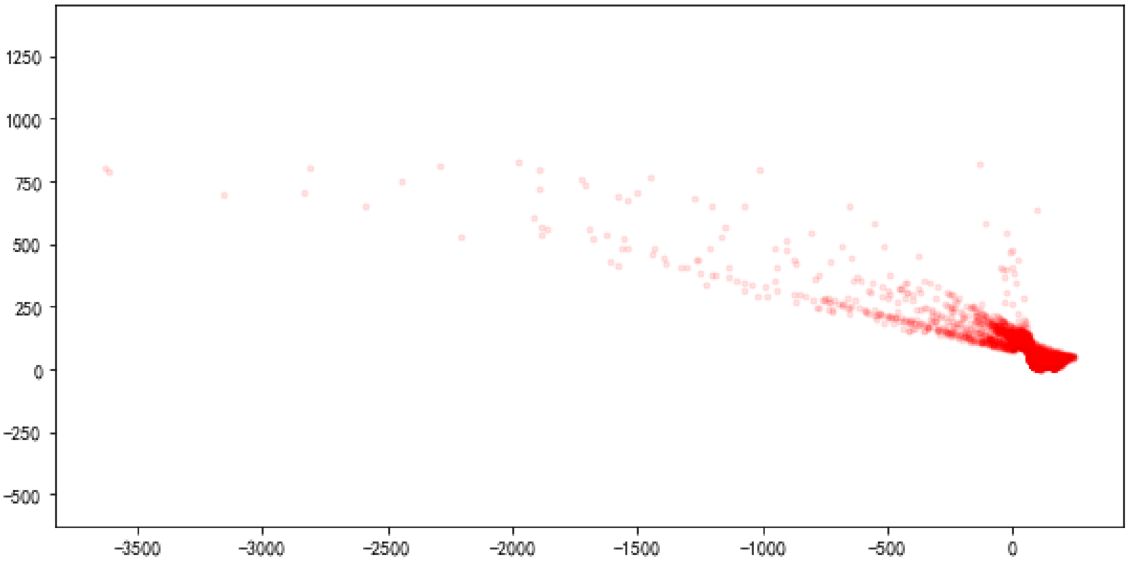
```
                theta_w2 = theta_c[i]-theta_13
        if v1[i][0]>=0 and v1[i][1]<=0:
            if -1<=v3_temp[0]<=-v1_temp[0] and v3_temp[1]<=0:
                theta_w2 = theta_c[i]+theta_13
            if -1<=v3_temp[0]<=v1_temp[0] and v3_temp[1]>=0:
                theta_w2 = theta_c[i]+theta_13
            if v1_temp[0]<=v3_temp[0]<=1 and v3_temp[1]>=0:
                theta_w2 = theta_c[i]-theta_13
            if -v1_temp[0]<=v3_temp[0]<=1 and v3_temp[1]<=0:
                theta_w2 = theta_c[i]-theta_13
    v_2 = v*damp#damp为与河岸交换速度过程中的作用系数

    crossx = np.zeros((theta_w2.size,theta_w2.size))
    crossy = crossx.copy()
    crossx_ls = []
    crossy_ls = []
    trash_m = []
    for i in range(theta_w2.size):
        theta_1 = theta_w2[i]
        for j in range(i+1,theta_w2.size):
            theta_2 = theta_w2[j]
            #x3 = (x[i]*np.tan(theta_1)-x[j]*np.tan(theta_2))/(np.tan(theta_1)-np.tan(theta_2))
            x3 = (coast[i]-coast[j]-np.tan(theta_1)*x[i]+np.tan(theta_2)*x[j])/(np.tan(theta_2)-np.tan(theta_1))
            y3 = coast[j]+(x3-x[j])*np.tan(theta_2)#到这一步得到了交点公式
            crossx[i][j] = x3
            crossy[i][j] = y3
            vec_x3 = np.array((x3,y3))
            vec_x2 = np.array((x[j],coast[j]))
            vec_x32 = vec_x2-vec_x3
            if np.dot(vec_x32,np.array((np.cos(theta_w2[j]),np.sin(theta_w2[j]))))>0:
                continue
            range_31,interp_31 = interp_for_under_coast(x[i],coast[i],x3,y3)
            range_32,interp_32 = interp_for_under_coast(x[j],coast[j],x3,y3)
            '''存在的bug：交点超出海岸线的坐标范围'''
            site_31_temp = np.where((range_31>=x.min())&(range_31<=x.max()))[0]
            site_32_temp = np.where((range_32>=x.min())&(range_32<=x.max()))[0]
            if site_31_temp.size == 0 :
                bool_under_coast_31 = 0
            else:
                interp_31 = interp_31[site_31_temp]
                range_31 = range_31[site_31_temp]
                coast_range_31 = coast[np.where((x>=range_31.min())&(x<=range_31.max()))]
                bool_under_coast_31 = (interp_31<coast_range_31).any()#修正，不能用any()
            if site_32_temp.size == 0 :
                bool_under_coast_32 = 0
            else:
                interp_32 = interp_32[np.where((range_32>=x.min())&(range_32<=x.max()))]
                range_32 = range_32[np.where((range_32>=x.min())&(range_32<=x.max()))]
                coast_range_32 = coast[np.where((x>=range_32.min())&(x<=range_32.max()))]
                bool_under_coast_32 = (interp_32<coast_range_32).any()
            if bool_under_coast_31 or bool_under_coast_32 :
                continue
```

```
                    trash_m.append(v[i]+v[j])#水的输送量，但是水速越快，垃圾越不容易留下来
                    crossx_ls.append(x3)
                    crossy_ls.append(y3)

    return np.array(crossx_ls),np.array(crossy_ls)
```

In [16]:

```
from ssea import *
from matplotlib import pyplot as plt
from sklearn.cluster import AffinityPropagation

crossx_ls,crossy_ls = cross_esi('东北风')
train_set = np.c_[crossx_ls,crossy_ls]
train_set,_ = filt_points(train_set)
af = AffinityPropagation()
#af.fit(train_set)
#train_res = af.predict(train_set)

plt.figure(figsize = (10,5))
plt.rcParams['font.sans-serif'] = ['KaiTi']
plt.rcParams['axes.unicode_minus']=False
plt.plot(coast_x,coast)
#plt.plot(crossx_ls,crossy_ls,'r.')
plt.plot(train_set[:,0],train_set[:,1],'r.',alpha = 0.1)
plt.axis('equal')
plt.xlim(0,200)
plt.ylim(-20,200)
plt.title('此图经过放缩后续数据无法用于分析',fontsize = 30)
plt.figure(figsize = (10,5))
plt.plot(coast_x,coast)
#plt.plot(crossx_ls,crossy_ls,'r.')
plt.plot(train_set[:,0],train_set[:,1],'r.',alpha = 0.1)
plt.axis('equal')
#plt.xlim(0,200)
#plt.ylim(-20,200)
plt.show()
```


此图经过放缩后续数据无法用于分析



项目记录：

9月1日：

1. 完善了4个风向的情况，封装该交点返回功能
2. AP聚类算法复杂度过高，不宜用于此项目，电脑根本无法满足要求
3. 可以采用远离海岸的水流阻尼减少模型减少杂点
4. 可以采用概率分布的形式，找到可能的峰。
5. 离散化，四舍五入取整。（个数为AP聚类的权重）

从结果来看，水速随着远离海岸下降还是必要的...思路回归的原因是无法通过AP聚类完成此类任务，也无法通过3-σ等方式起到滤去远离海岸的点，甚至于，远离海岸是这个模型的主要结果。

采用离散化方式

In [18]:

```
train_set_bk = train_set.copy()
print('离散化前的交点数量为',train_set.shape[0])
```

```
离散化前的交点数量为 7206
```

In [28]:

```
train_set = train_set_bk.copy()
train_set = np.round(train_set)

train_set = train_set.tolist()
train_set_tuple = [tuple(i) for i in train_set]
train_set = set(train_set_tuple)
```

In [29]:

```
print('离散化后的交点数量为',len(train_set))
```

```
离散化后的交点数量为 3344
```

数量减半，但我觉得还是好多

In [30]:

```
train_set = list(train_set)
num_ls = [ ]
for i in train_set:
    num_ls.append(train_set_tuple.count(i))
```

In [36]:

```
num_ls = np.array(num_ls)
print('不是孤立点的交点数量为',num_ls[num_ls>1].size)
```

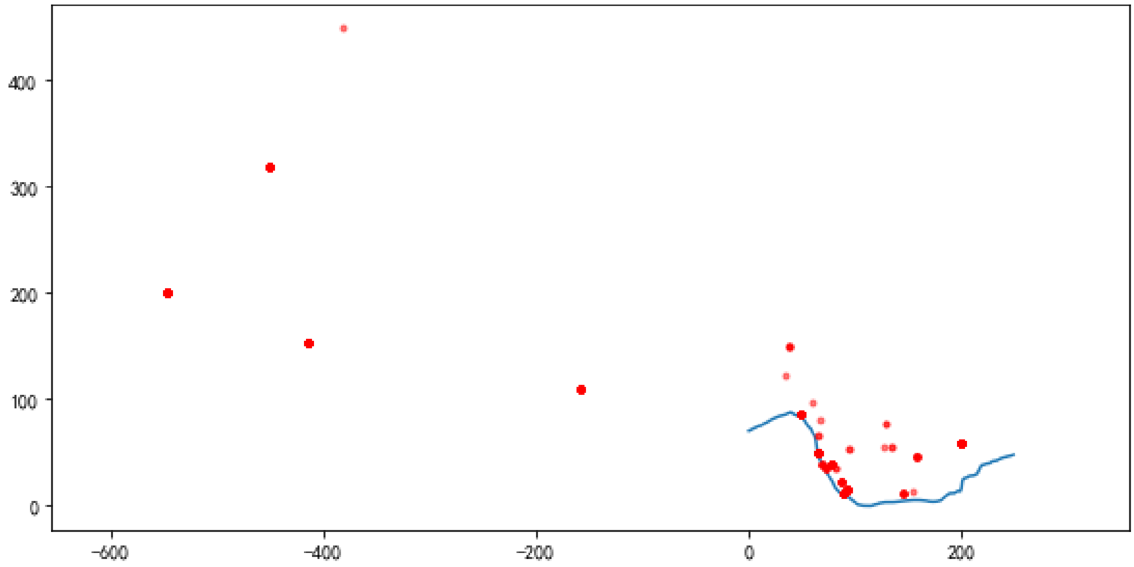
```
不是孤立点的交点数量为 1145
```

最好的应该还是给个随距离的减少，因为有些点离水流的出发点还是太远了

（下图单纯得找出了四舍五入后不是单个点的地方）

In [45]:

```
coor = np.array(train_set)[num_ls[num_ls>1],:]

plt.figure(figsize = (10,5))
plt.plot(coast_x,coast)
#plt.plot(crossx_ls,crossy_ls,'r.')
plt.plot(coor[:,0],coor[:,1],'r.',alpha = 0.5)
plt.axis('equal')
#plt.xlim(0,200)
#plt.ylim(-20,200)
plt.show()
```



AP聚类，结果不可靠，随缘，先放着，缺乏了极其重要的密度信息

In [35]:

```
af.fit(train_set)
train_res = af.predict(train_set)
```

In [ ]:

```
```

In [ ]:

```
```