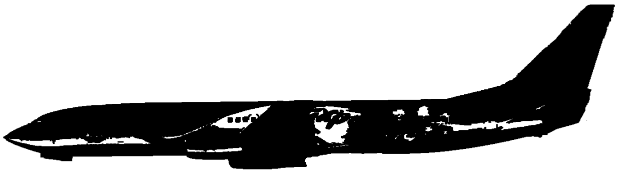
Numerical Analysis_HW1

姓名:徐絡祥 **學號:B092040038** **講師姓名：呂宗澤**

Prototype of Plane

]



Lagrange

Method of selecting points

利用影像處理，找出上緣和下緣的x,y點，並且將y座標轉換，存入up_plane down_plane

uppoints:

x = [0.0, 2.0, 4.0, 6.0, 8.0, 10.0, 12.0, 14.0, 16.0, 18.0, 20.0, 22.0, 24.0, 26.0, 28.0, 30.0, 32.0, 34.0, 36.0, 38.0, 40.0, 42.0, 44.0, 46.0, 48.0, 50.0, 52.0, 54.0, 56.0, 58.0,

60.0, 62.0, 64.0, 66.0, 68.0, 70.0, 72.0, 74.0, 76.0, 78.0, 80.0, 82.0, 84.0, 86.0, 88.0, 90.0, 92.0, 94.0, 96.0, 98.0, 100.0, 102.0, 104.0, 106.0, 108.0, 110.0]

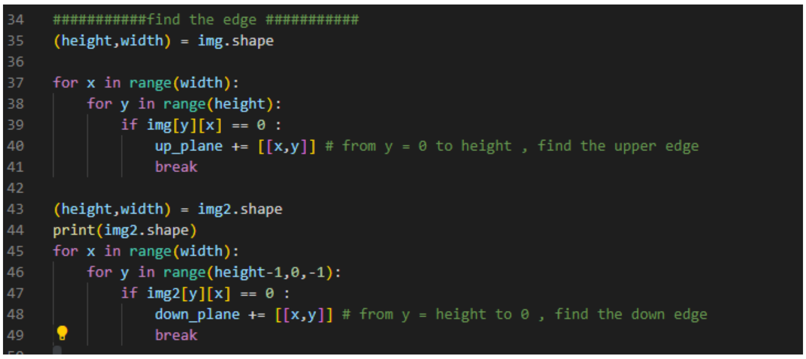
y = [-142, -130, -118, -104, -96, -91, -87, -84, -83, -81, -80, -78, -78, -77, -77, -77, -76, -76, -76, -76, -75, -75, -75, -74, -74, -74, -72, -72, -73, -73, -73, -73, -72, -72, -72, -71, -71, -71, -69, -69, -68, -62, -56, -50, -44, -36, -23, 2, 26, 46, 70, 98, 123, 140, 140, 140]

downpoints:

[4.0, 6.0, 8.0, 10.0, 12.0, 14.0, 16.0, 18.0, 20.0, 22.0, 24.0, 26.0, 28.0, 30.0, 32.0, 34.0, 36.0, 38.0, 40.0, 42.0, 44.0, 46.0, 48.0, 50.0, 52.0, 54.0, 56.0, 58.0, 60.0, 62.0, 64.0, 66.0, 68.0, 70.0, 72.0, 74.0, 76.0, 78.0, 80.0, 82.0, 84.0, 86.0, 88.0, 90.0, 92.0, 94.0, 96.0, 98.0, 100.0, 102.0, 104.0, 106.0, 108.0, 110.0, 112.0, 114.0]

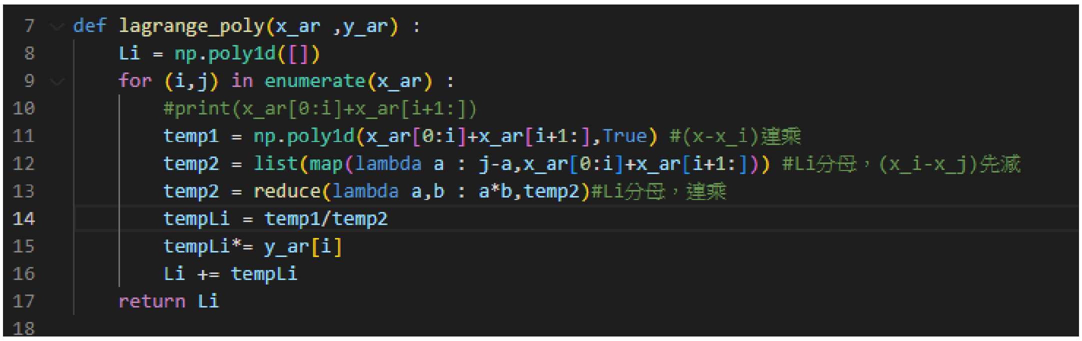
[-16.6, -17.6, -18.4, -20.0, -20.3, -20.6, -19.6, -19.6, -19.6, -19.6, -19.5, -19.5, -19.5, -19.4, -19.4, -19.4, -19.4, -20.2, -20.3, -20.3, -21.6, -22.3, -22.4, -22.4, -22.3,

-22.1, -21.9, -21.8, -19.6, -19.0, -18.9, -18.8, -18.9, -18.9, -19.0, -19.0, -19.0, -18.9, -18.8, -18.6, -18.4, -18.0, -17.7, -17.3, -16.9, -16.4, -15.8, -15.3, -15.2, -14.8, -13.4, -12.7, -11.6, -5.0, 5.6, 12.1]



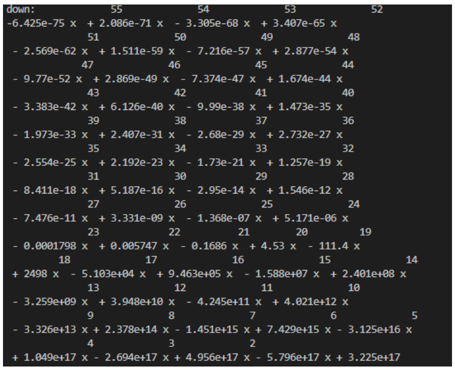
```
###########find the edge ###########
(height,width) = img.shape

for x in range(width):
    for y in range(height):
        if img[y][x] == 0 :
            up_plane += [[x,y]] # from y = 0 to height , find the upper edge
            break

(height,width) = img2.shape
print(img2.shape)
for x in range(width):
    for y in range(height-1,0,-1):
        if img2[y][x] == 0 :
            down_plane += [[x,y]] # from y = height to 0 , find the down edge
            break
```

Function

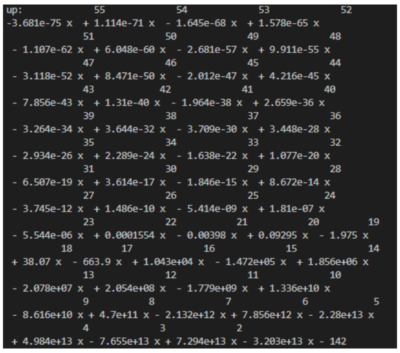


```
def lagrange_poly(x_ar ,y_ar) :
    Li = np.poly1d([])
    for (i,j) in enumerate(x_ar) :
        #print(x_ar[0:i]+x_ar[i+1:])
        temp1 = np.poly1d(x_ar[0:i]+x_ar[i+1:],True) #(x-x_i)連乘
        temp2 = list(map(lambda a : j-a,x_ar[0:i]+x_ar[i+1:])) #Li分母，(x_i-x_j)先減
        temp2 = reduce(lambda a,b : a*b,temp2)#Li分母，連乘
        tempLi = temp1/temp2
        tempLi*= y_ar[i]
        Li += tempLi
    return Li
```

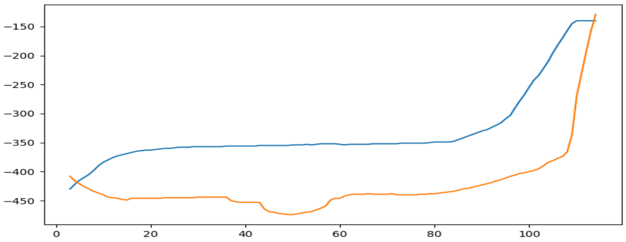
Polynomial and Plotting



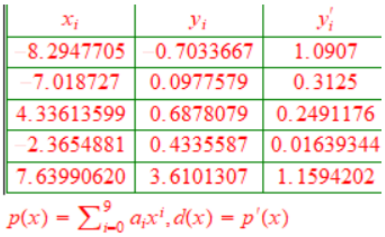
```
down:               55               54               53               52
-6.425e-75 x  + 2.086e-71 x  - 3.305e-68 x  + 3.407e-65 x
              51               50               49               48
 - 2.569e-62 x  + 1.511e-59 x  - 7.216e-57 x  + 2.877e-54 x
             47               46               45               44
 - 9.77e-52 x  + 2.869e-49 x  - 7.374e-47 x  + 1.674e-44 x
              43               42              41               40
 - 3.383e-42 x  + 6.126e-40 x  - 9.99e-38 x  + 1.473e-35 x
              39               38              37               36
 - 1.973e-33 x  + 2.407e-31 x  - 2.68e-29 x  + 2.732e-27 x
              35               34              33               32
 - 2.554e-25 x  + 2.192e-23 x  - 1.73e-21 x  + 1.257e-19 x
              31               30              29               28
 - 8.411e-18 x  + 5.187e-16 x  - 2.95e-14 x  + 1.546e-12 x
              27               26               25               24
 - 7.476e-11 x  + 3.331e-09 x  - 1.368e-07 x  + 5.171e-06 x
              23              22           21         20           19
 - 0.0001798 x  + 0.005747 x  - 0.1686 x  + 4.53 x  - 111.4 x
         18               17               16               15               14
 + 2498 x  - 5.103e+04 x  + 9.463e+05 x  - 1.588e+07 x  + 2.401e+08 x
              13               12               11               10
 - 3.259e+09 x  + 3.948e+10 x  - 4.245e+11 x  + 4.021e+12 x
              9              8              7              6              5
 - 3.326e+13 x + 2.378e+14 x - 1.451e+15 x + 7.429e+15 x - 3.125e+16 x
             4              3              2
 + 1.049e+17 x - 2.694e+17 x + 4.956e+17 x - 5.796e+17 x + 3.225e+17
```



```
up:               55               54               53               52
-3.681e-75 x  + 1.114e-71 x  - 1.645e-68 x  + 1.578e-65 x
              51               50               49               48
 - 1.107e-62 x  + 6.048e-60 x  - 2.681e-57 x  + 9.911e-55 x
              47               46               45               44
 - 3.118e-52 x  + 8.471e-50 x  - 2.012e-47 x  + 4.216e-45 x
              43              42               41               40
 - 7.856e-43 x  + 1.31e-40 x  - 1.964e-38 x  + 2.659e-36 x
              39               38               37               36
 - 3.264e-34 x  + 3.644e-32 x  - 3.709e-30 x  + 3.448e-28 x
              35               34               33               32
 - 2.934e-26 x  + 2.289e-24 x  - 1.638e-22 x  + 1.077e-20 x
              31               30               29               28
 - 6.507e-19 x  + 3.614e-17 x  - 1.846e-15 x  + 8.672e-14 x
              27               26               25              24
 - 3.745e-12 x  + 1.486e-10 x  - 5.414e-09 x  + 1.81e-07 x
              23               22              21              20          19
 - 5.544e-06 x  + 0.0001554 x  - 0.00398 x  + 0.09295 x  - 1.975 x
          18         17               16               15               14
 + 38.07 x  - 663.9 x  + 1.043e+04 x  - 1.472e+05 x  + 1.856e+06 x
              13               12               11               10
 - 2.078e+07 x  + 2.054e+08 x  - 1.779e+09 x  + 1.336e+10 x
              9             8              7              6             5
 - 8.616e+10 x + 4.7e+11 x - 2.132e+12 x + 7.856e+12 x - 2.28e+13 x
              4              3              2
 + 4.984e+13 x - 7.655e+13 x + 7.294e+13 x - 3.203e+13 x - 142
```



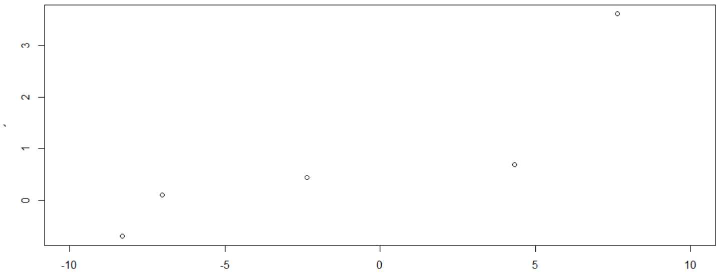
Hermit



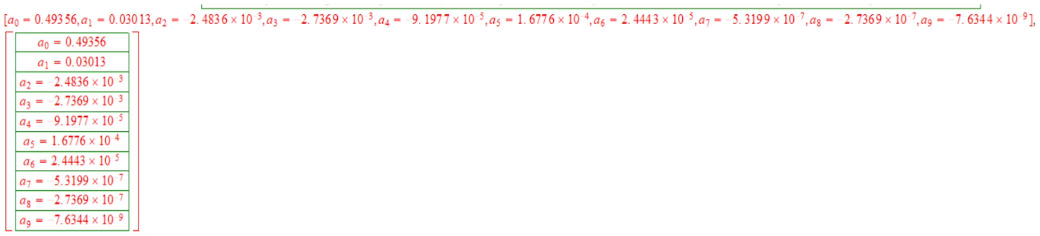
x_i	y_i	y_i'
−8.2947705	−0.7033667	1.0907
−7.018727	0.0977579	0.3125
4.33613599	0.6878079	0.2491176
−2.3654881	0.4335587	0.01639344
7.63990620	3.6101307	1.1594202

$p(x) = \sum_{i=0}^{9} a_i x^i, d(x) = p'(x)$

利用老師的方法並在Geogebra 找出這一些點

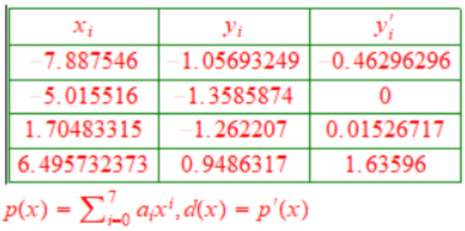


這是飛機上部分我用hermit找的點



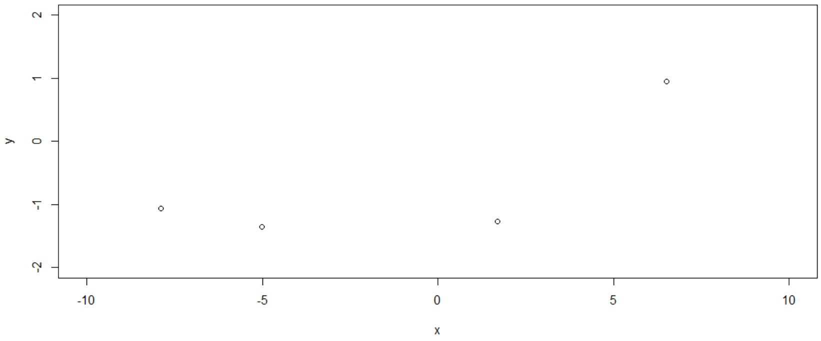
$[a_0 = 0.49356, a_1 = 0.03013, a_2 = -2.4836 \times 10^{-3}, a_3 = -2.7369 \times 10^{-3}, a_4 = -9.1977 \times 10^{-5}, a_5 = 1.6776 \times 10^{-4}, a_6 = 2.4443 \times 10^{-5}, a_7 = -5.3199 \times 10^{-7}, a_8 = -2.7369 \times 10^{-7}, a_9 = -7.6344 \times 10^{-9}],$

$$\begin{bmatrix} a_0 = 0.49356 \\ a_1 = 0.03013 \\ a_2 = -2.4836 \times 10^{-3} \\ a_3 = -2.7369 \times 10^{-3} \\ a_4 = -9.1977 \times 10^{-5} \\ a_5 = 1.6776 \times 10^{-4} \\ a_6 = 2.4443 \times 10^{-5} \\ a_7 = -5.3199 \times 10^{-7} \\ a_8 = -2.7369 \times 10^{-7} \\ a_9 = -7.6344 \times 10^{-9} \end{bmatrix}$$



x_i	y_i	y_i'
−7.887546	−1.05693249	−0.46296296
−5.015516	−1.3585874	0
1.70483315	−1.262207	0.01526717
6.495732373	0.9486317	1.63596

$p(x) = \sum_{i=0}^{7} a_i x^i, d(x) = p'(x)$

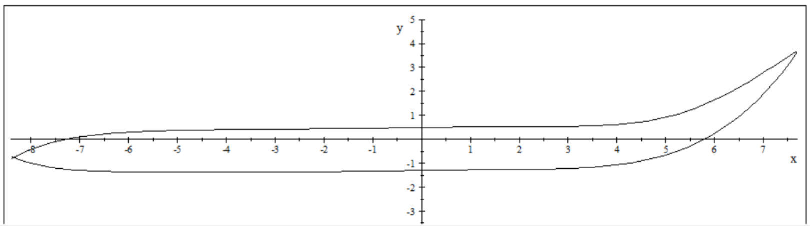


這是飛機下部分我用hermit找的點

$$(1,x,x^2,x^3,x^4,x^5,x^6,x^7)\left[\begin{array}{|c|}\hline a_0 = -1.2978 \\ \hline a_1 = 3.0187\times10^{-2} \\ \hline a_2 = -4.2986\times10^{-3} \\ \hline a_3 = -2.0731\times10^{-3} \\ \hline a_4 = 5.2137\times10^{-4} \\ \hline a_5 = 1.7129\times10^{-4} \\ \hline a_6 = 4.7173\times10^{-6} \\ \hline a_7 = -9.5021\times10^{-7} \\ \hline\end{array}\right] : a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 =$$

$$-9.5021\times10^{-7}x^7 + 4.7173\times10^{-6}x^6 + 1.7129\times10^{-4}x^5 + 5.2137\times10^{-4}x^4 - 2.0731\times10^{-3}x^3 - 4.2986\times10^{-3}x^2 + 3.0187\times10^{-2}x - 1.2978$$

$$-7.6344\times10^{-9}x^9 - 2.7369\times10^{-7}x^8 - 5.3199\times10^{-7}x^7 + 2.4443\times10^{-5}x^6 + 1.6776\times10^{-4}x^5 - 9.1977\times10^{-5}x^4 - 2.7369\times10^{-3}x^3 - 2.4836\times10^{-3}x^2 + 0.03013x + 0.49356$$



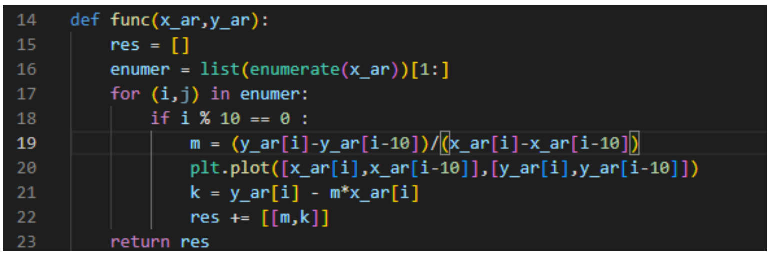
Other Method

利用許多點構成許多一次方程式，

Method of selecting points

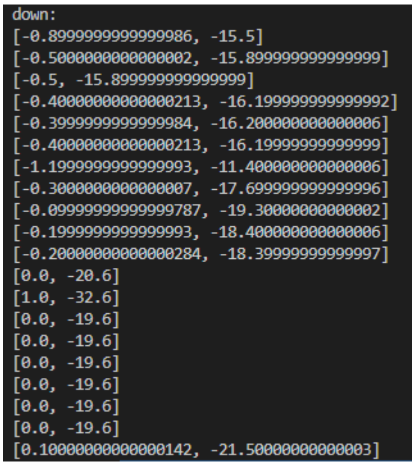
和Lagrange一樣，這裡用每十個像素取一格點

Function

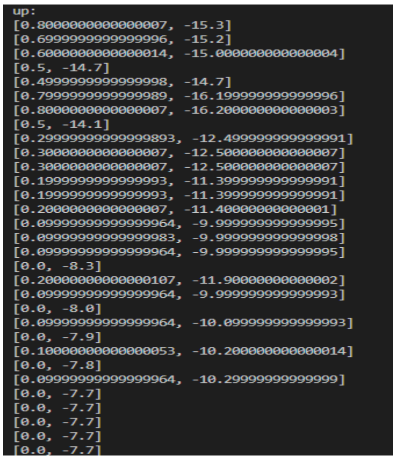


```python
def func(x_ar,y_ar):
    res = []
    enumer = list(enumerate(x_ar))[1:]
    for (i,j) in enumer:
        if i % 10 == 0 :
            m = (y_ar[i]-y_ar[i-10])/(x_ar[i]-x_ar[i-10])
            plt.plot([x_ar[i],x_ar[i-10]],[y_ar[i],y_ar[i-10]])
            k = y_ar[i] - m*x_ar[i]
            res += [[m,k]]
    return res
```

Polynomial and plotting

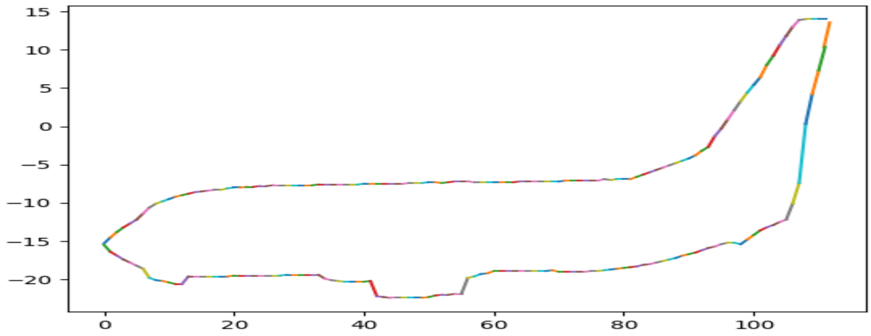


```
down:
[-0.8999999999999986, -15.5]
[-0.5000000000000002, -15.899999999999999]
[-0.5, -15.899999999999999]
[-0.40000000000000213, -16.199999999999992]
[-0.3999999999999984, -16.200000000000006]
[-0.40000000000000213, -16.199999999999999]
[-1.1999999999999993, -11.400000000000006]
[-0.3000000000000007, -17.699999999999996]
[-0.09999999999999787, -19.300000000000002]
[-0.1999999999999993, -18.400000000000006]
[-0.20000000000000284, -18.399999999999997]
[0.0, -20.6]
[1.0, -32.6]
[0.0, -19.6]
[0.0, -19.6]
[0.0, -19.6]
[0.0, -19.6]
[0.0, -19.6]
[0.0, -19.6]
[0.10000000000000142, -21.500000000000003]
```



```
up:
[0.8000000000000007, -15.3]
[0.6999999999999996, -15.2]
[0.6000000000000014, -15.000000000000004]
[0.5, -14.7]
[0.4999999999999998, -14.7]
[0.7999999999999989, -16.199999999999996]
[0.8000000000000007, -16.200000000000003]
[0.5, -14.1]
[0.29999999999999893, -12.499999999999991]
[0.3000000000000007, -12.500000000000007]
[0.3000000000000007, -12.500000000000007]
[0.1999999999999993, -11.399999999999991]
[0.1999999999999993, -11.399999999999991]
[0.2000000000000007, -11.40000000000001]
[0.09999999999999964, -9.999999999999995]
[0.09999999999999983, -9.999999999999998]
[0.09999999999999964, -9.999999999999995]
[0.0, -8.3]
[0.20000000000000107, -11.900000000000002]
[0.09999999999999964, -9.999999999999993]
[0.0, -8.0]
[0.09999999999999964, -10.099999999999993]
[0.0, -7.9]
[0.10000000000000053, -10.200000000000014]
[0.0, -7.8]
[0.09999999999999964, -10.299999999999999]
[0.0, -7.7]
[0.0, -7.7]
[0.0, -7.7]
[0.0, -7.7]
[0.0, -7.7]
```

上面是方程式， y = mx + k 表示成[m , k]



心得

當我在寫Lagrange 時，我一開始把全部的點放進去，然後一直造成overflow，後來我就將資料點減少至大約50幾格點，然後才能做出來。至於hermit method ，一開始我一直糾結想要把機尾那個平平的那一條線用出來，但是一直沒有辦法；那個時候一開始點會一直亂跳，所以我一直調動那些導致變動太大的點，用了很久才找到比較像的點，但是機尾真的找不到，所以我覺得hermit應該要用機尾沒有平才行，所以我最後只好讓機尾直接變尖的。Other method 我是用很多條一條一條的線段組合而成，在這裡因為沒有其他方法計算量那麼大，所以我是每十個點就找一個線段。最後，我覺得Lagrange 是我覺得最好看得圖形，雖然我在other method 畫出來的圖也蠻不錯的，但是Lagrange 是比較平滑且可以微分的，如果將other method 放大來看，他是一段一段的函式組成的飛機。