1_SVG_converter_Copper

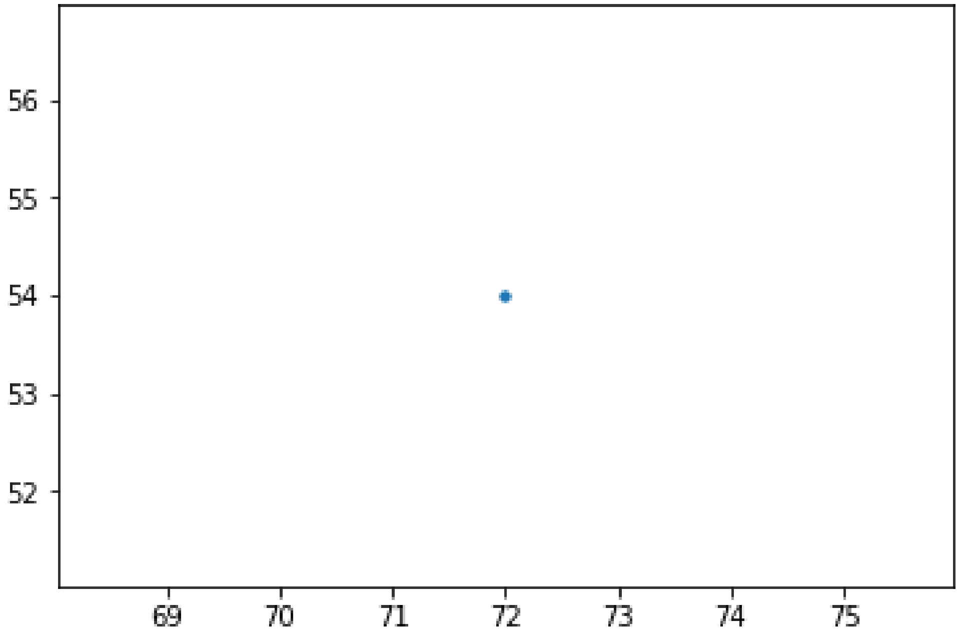
January 16, 2021

```
In [5]: import numpy as np
        import matplotlib.pyplot as plt

        from svg.path import parse_path
        from svg.path.path import Line
        from xml.dom import minidom

        def line_splitter(start, end):
            return (lambda t: (1-t)*start+t*end)

        def cubic_bezier_converter(start, control1, control2, end):
            original_data = np.array([start, control1, control2, end])
            cubic_bezier_matrix = np.array([
                [-1,  3, -3,  1],
                [ 3, -6,  3,  0],
                [-3,  3,  0,  0],
                [ 1,  0,  0,  0]
            ])
            return_data = cubic_bezier_matrix.dot(original_data)

            return (lambda t: np.array([t**3, t**2, t, 1]).dot(return_data))

        # Learned from
        # https://stackoverflow.com/questions/36971363/how-to-interpolate-svg-path-into-a-pixe


        doc = minidom.parse('LaneMap2.svg')
        path_strings = [path.getAttribute('d') for path
                        in doc.getElementsByTagName('path')]
        doc.unlink()

        points_np_all=[]
        points_np_all=np.empty((len(path_strings)),dtype=object)
        print(len(points_np_all))
        #points_np_all[k]=np.array([])

        for k in range(len(path_strings)):
```

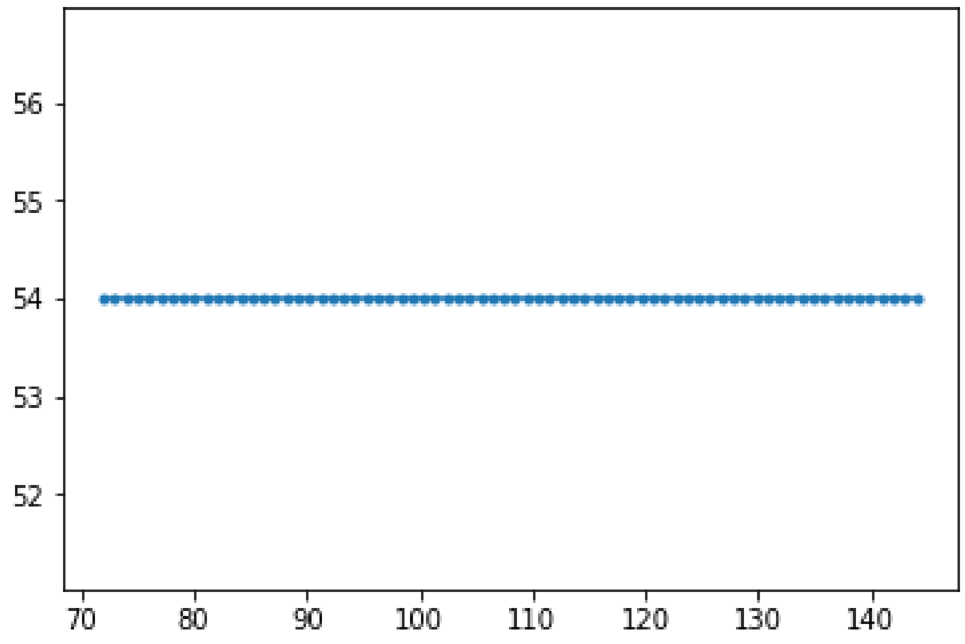
```
#for path_string in path_strings:
    path = parse_path(path_strings[k])
    points_np_merge=np.empty((0,2), float)
    #points_np_merge=np.empty(points_np_merge)
    for dat in path:

#path=parse_path(path_strings[block])

#dat=path[key]

        if type(dat).__name__=='CubicBezier':
            start_np = np.array([dat.start.real, dat.start.imag])
            control1_np = np.array([dat.control1.real, dat.control1.imag])
            control2_np = np.array([dat.control2.real, dat.control2.imag])
            end_np = np.array([dat.end.real, dat.end.imag])
            converted_curve = cubic_bezier_converter(start_np, control1_np, control2_n
            #
            diff_np=start_np-end_np
            n_dots=np.round(np.linalg.norm(diff_np))
            #
            points_np = np.array([converted_curve(t) for t in np.linspace(0, 1, n_dots
        elif type(dat).__name__=='Line':
            start_np = np.array([dat.start.real, dat.start.imag])
            end_np = np.array([dat.end.real, dat.end.imag])
            converted_line = line_splitter(start_np,end_np)
            #
            diff_np=start_np-end_np
            n_dots=np.round(np.linalg.norm(diff_np))
            #
            points_np=np.array([converted_line(t) for t in np.linspace(0, 1, n_dots)])
        elif type(dat).__name__=='Move':
            #
            n_dots=1
            #
            start_np = np.array([dat.start.real, dat.start.imag])
            end_np = np.array([dat.end.real, dat.end.imag])
            points_np = np.array([start_np,end_np])
        else:
            points_np=np.array([])
        #points_np_merge=np.concatenate(points_np_merge,points_np)
        points_np_merge=np.append(points_np_merge, points_np, axis=0)
#         if k==0:
#             points_np_merge=points_np
#         else:
#             points_np_merge=np.append(points_np_merge,points_np,axis=0)
        plt.plot(points_np[:, 0], points_np[:, 1], '.-')
        plt.show()
        print(len(points_np))
        print(len(points_np_merge))
    #points_np_all1=points_np_all1.append(points_np_merge)
    #points_np_all=points_np_merge
    points_np_all[k]= points_np_merge
#     points_np_all=points_np_all.append(points_np_merge)
    print(len(points_np_all))
    plt.plot(points_np_merge[:, 0], points_np_merge[:, 1], '.-')
    plt.show()
```

```
len(points_np_all)
```

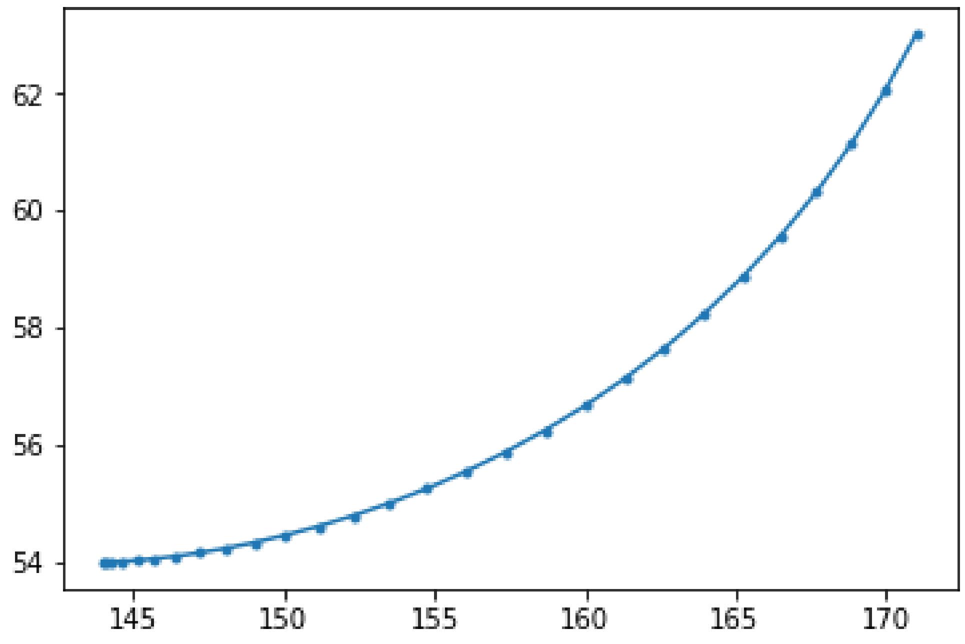
```
4
```



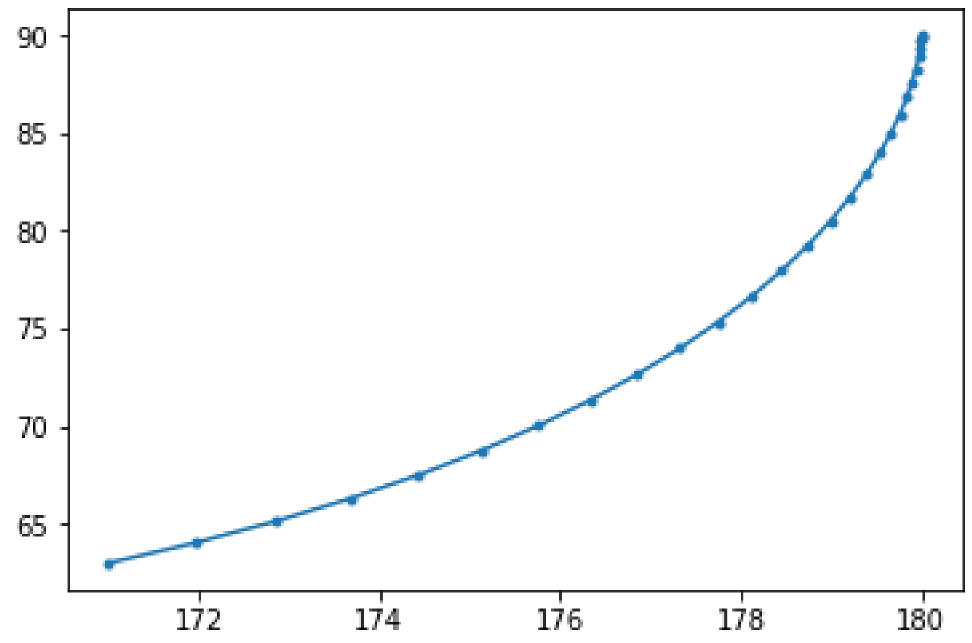
```
2
2
```


56
55
54
53
52
70
80
90
100
110
120
130
140

72
74

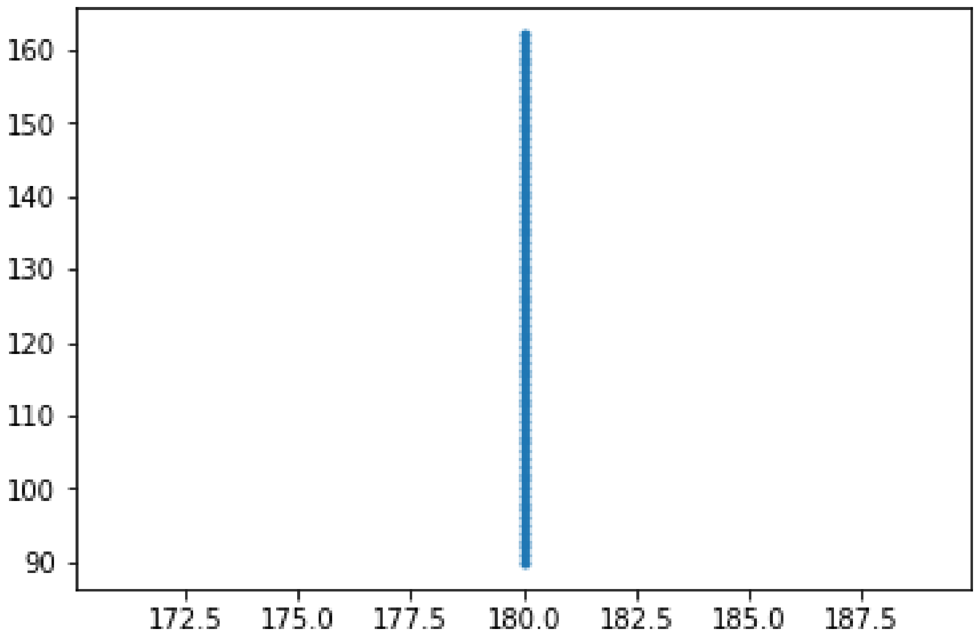

62
60
58
56
54
145
150
155
160
165
170

28
102



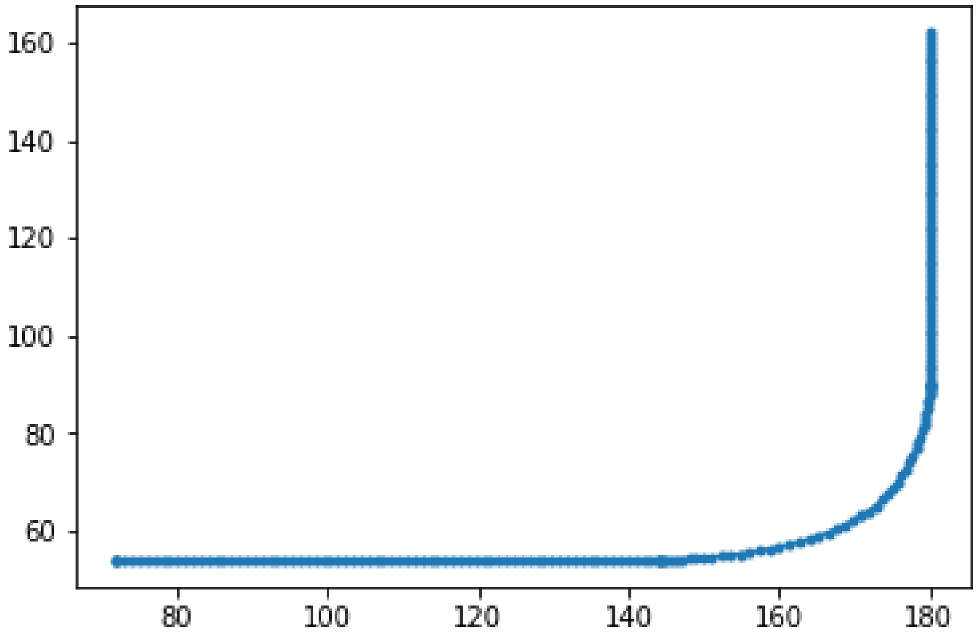
90
85
80
75
70
65
172
174
176
178
180

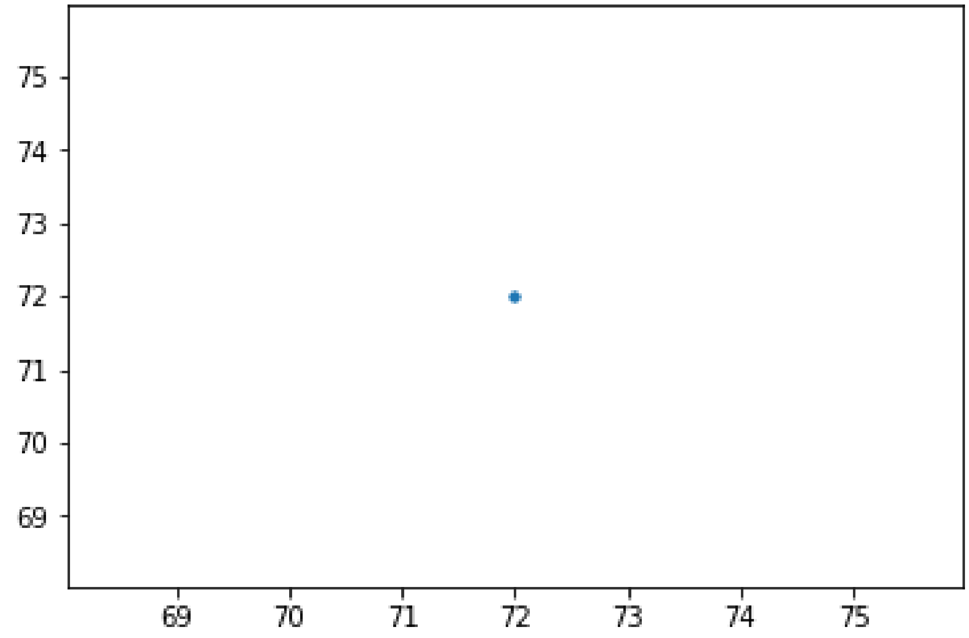
28
130



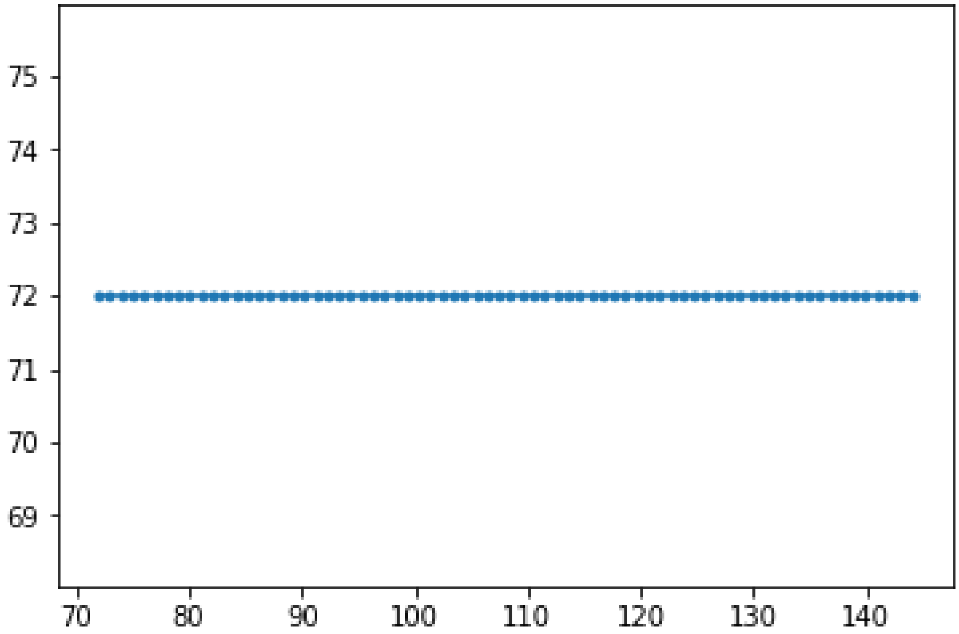
160
150
140
130
120
110
100
90
172.5
175.0
177.5
180.0
182.5
185.0
187.5

```
72
202
4
```

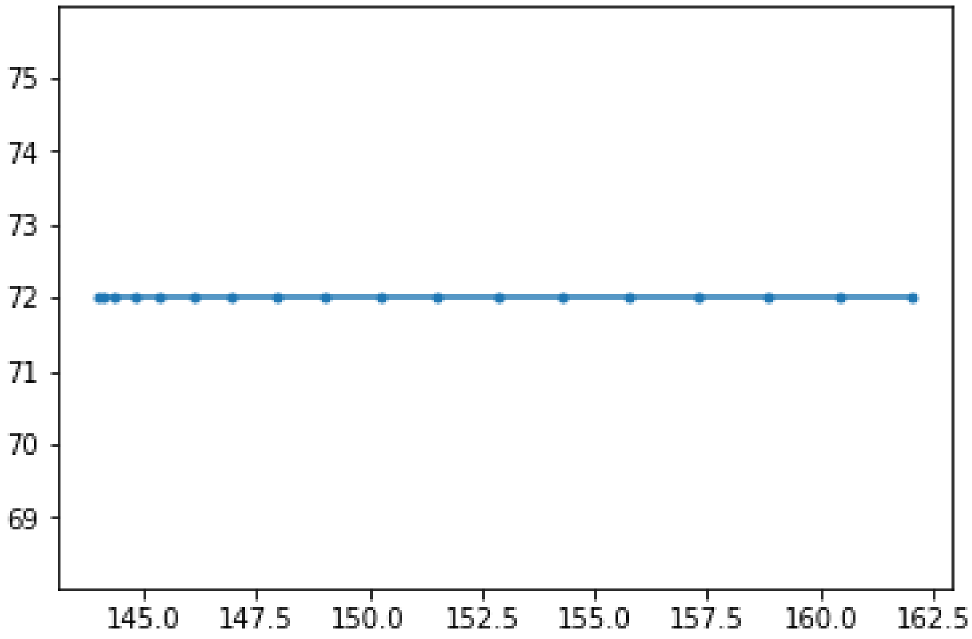

160
140
120
100
80
60
80
100
120
140
160
180


75
74
73
72
71
70
69
69
70
71
72
73
74
75

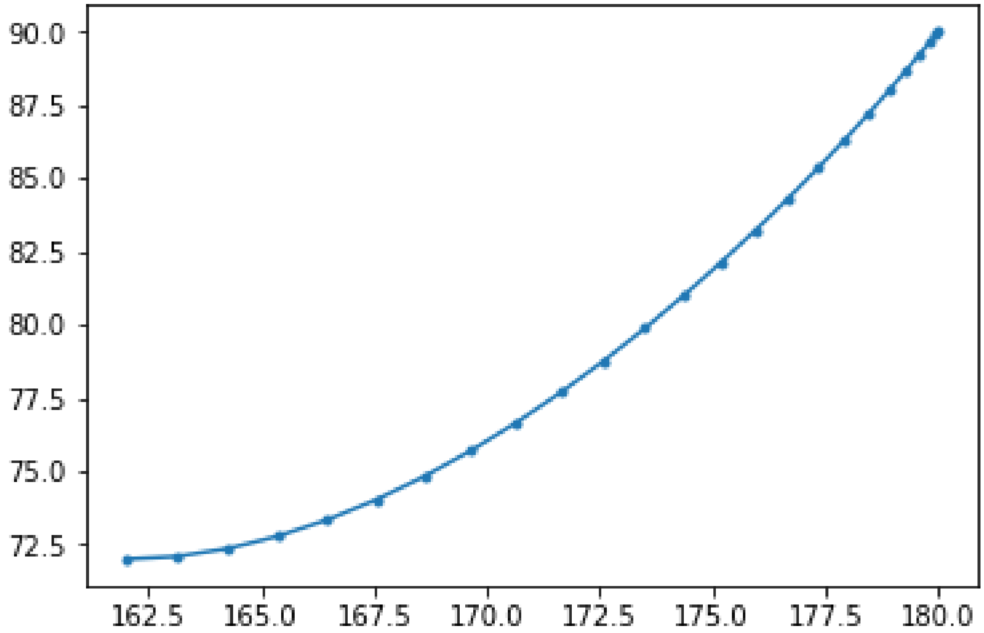
```
2
2
```



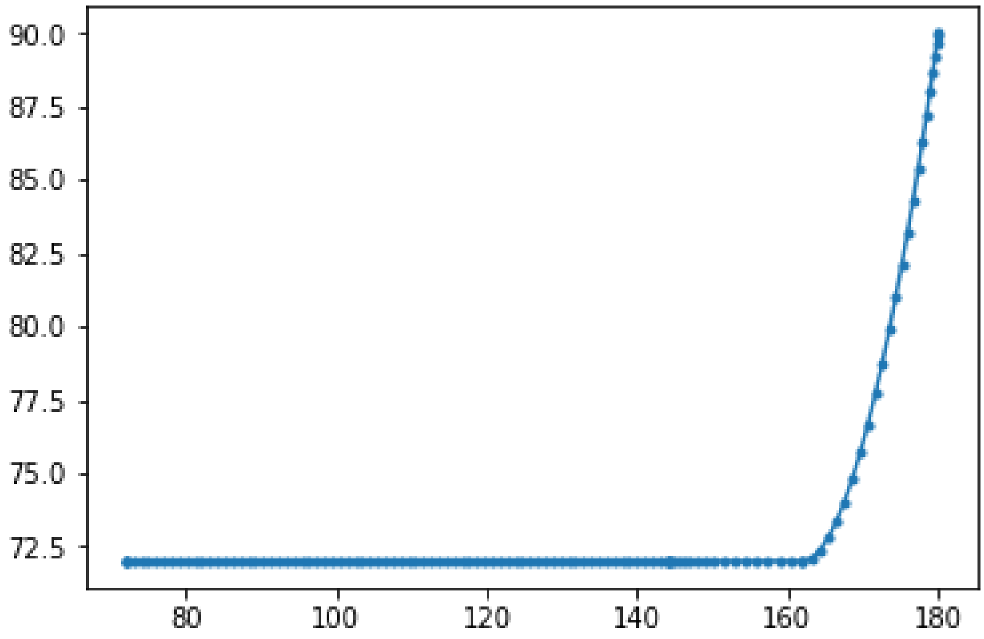
```
72
74
```

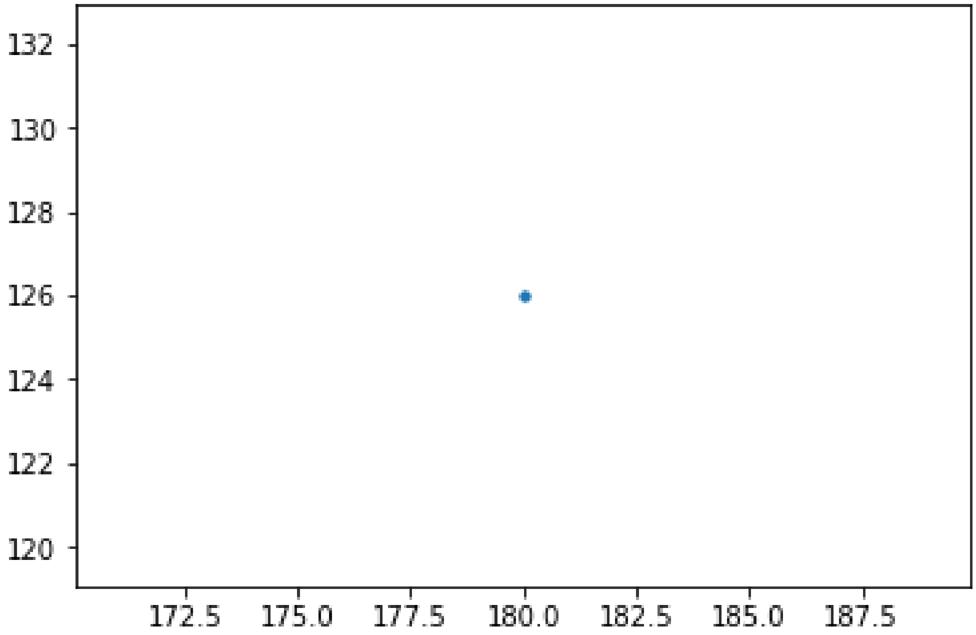

75
74
73
72
71
70
69
145.0
147.5
150.0
152.5
155.0
157.5
160.0
162.5

```
18
92
```

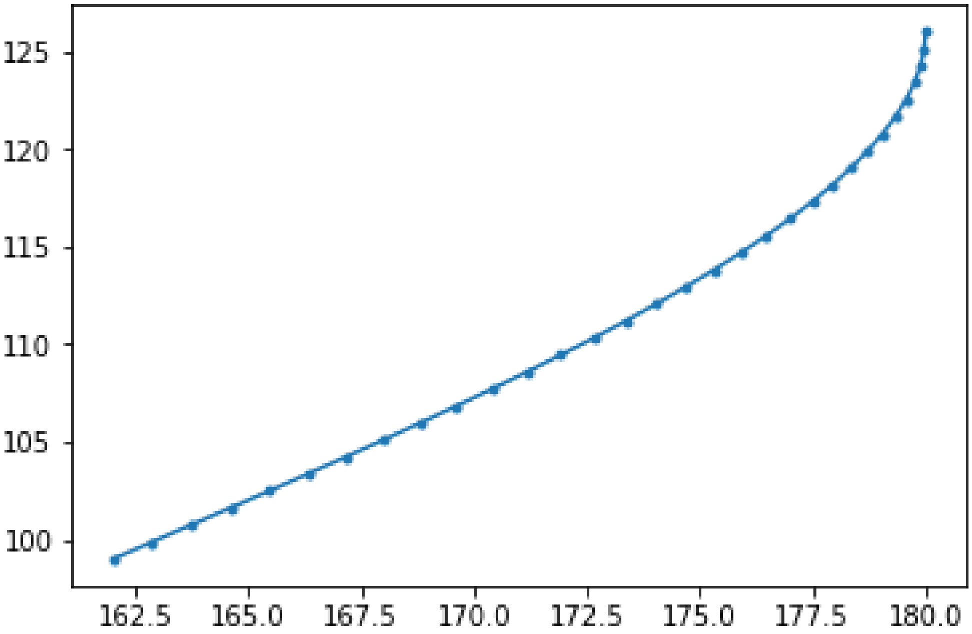

90.0
87.5
85.0
82.5
80.0
77.5
75.0
72.5
162.5
165.0
167.5
170.0
172.5
175.0
177.5
180.0

```
25
117
4
```

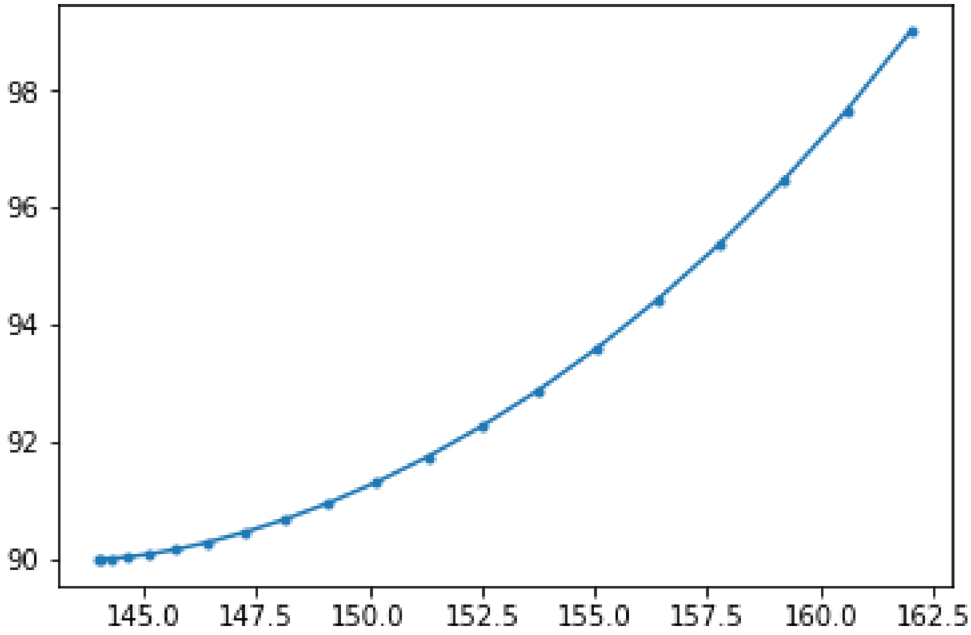

90.0
87.5
85.0
82.5
80.0
77.5
75.0
72.5
80
100
120
140
160
180


132
130
128
126
124
122
120
172.5
175.0
177.5
180.0
182.5
185.0
187.5

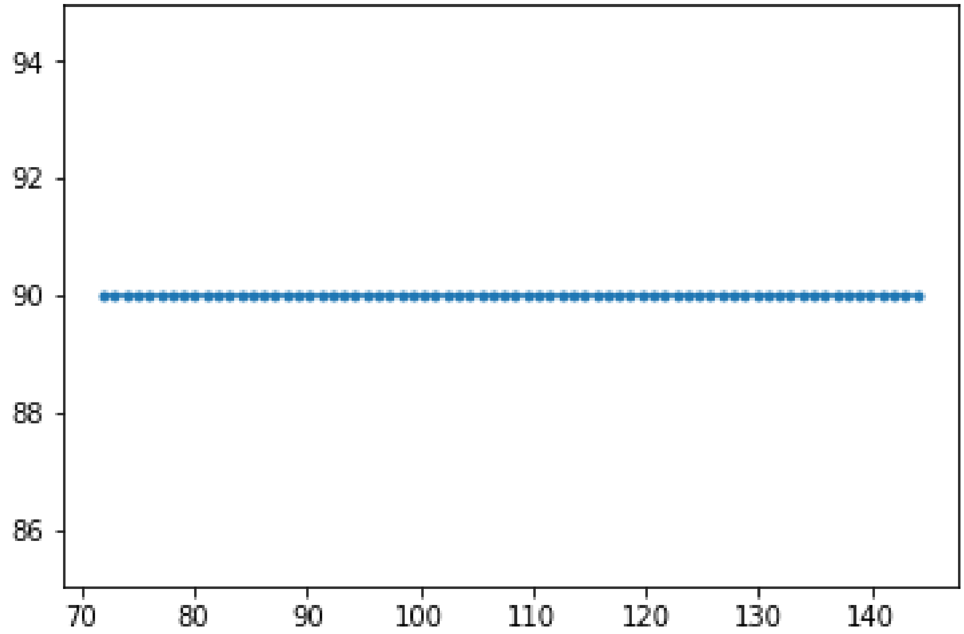
2
2


125
120
115
110
105
100
162.5
165.0
167.5
170.0
172.5
175.0
177.5
180.0

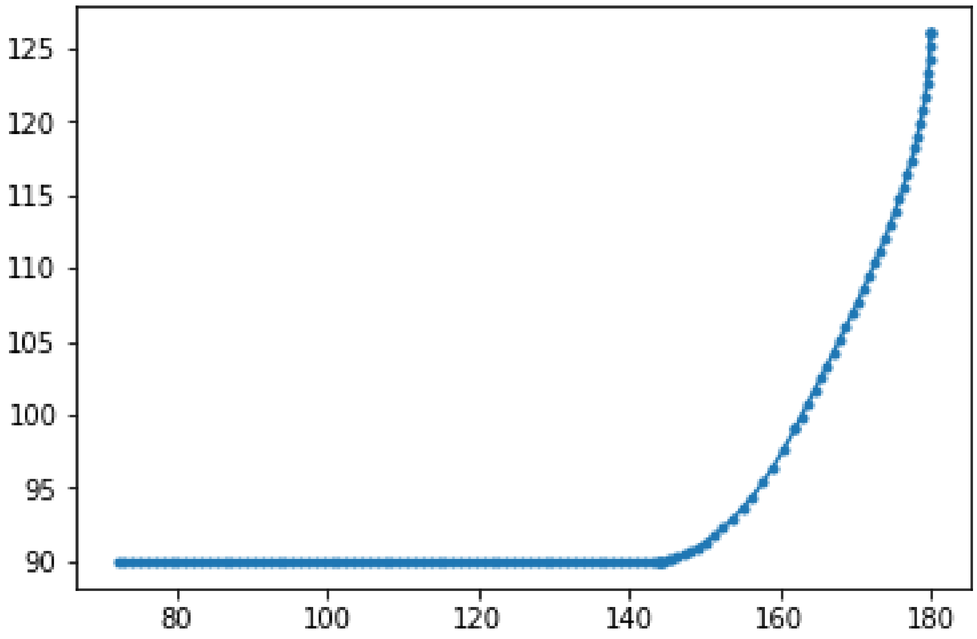
32
34

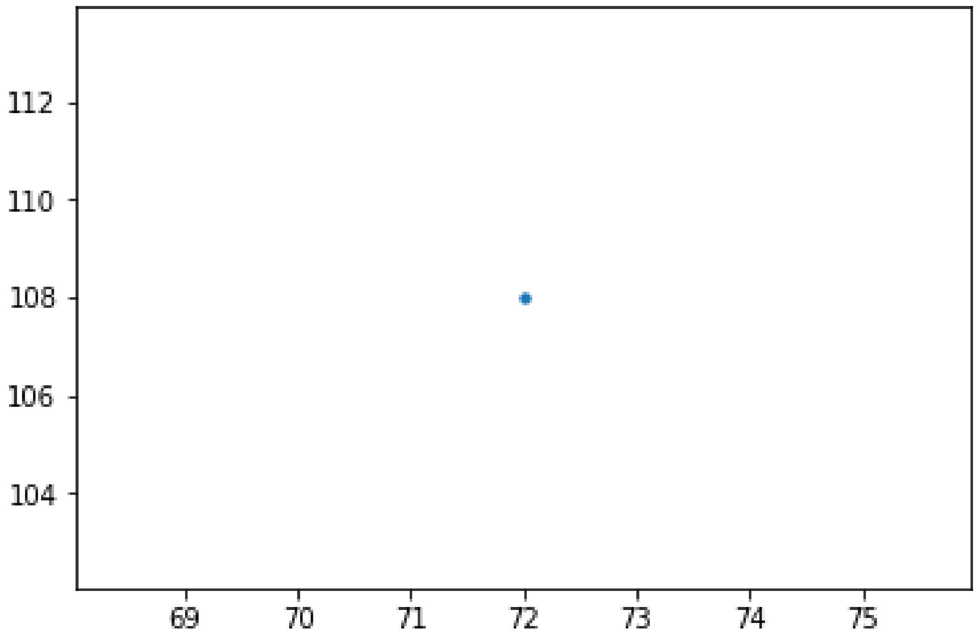

98
96
94
92
90
145.0
147.5
150.0
152.5
155.0
157.5
160.0
162.5

```
20
54
```

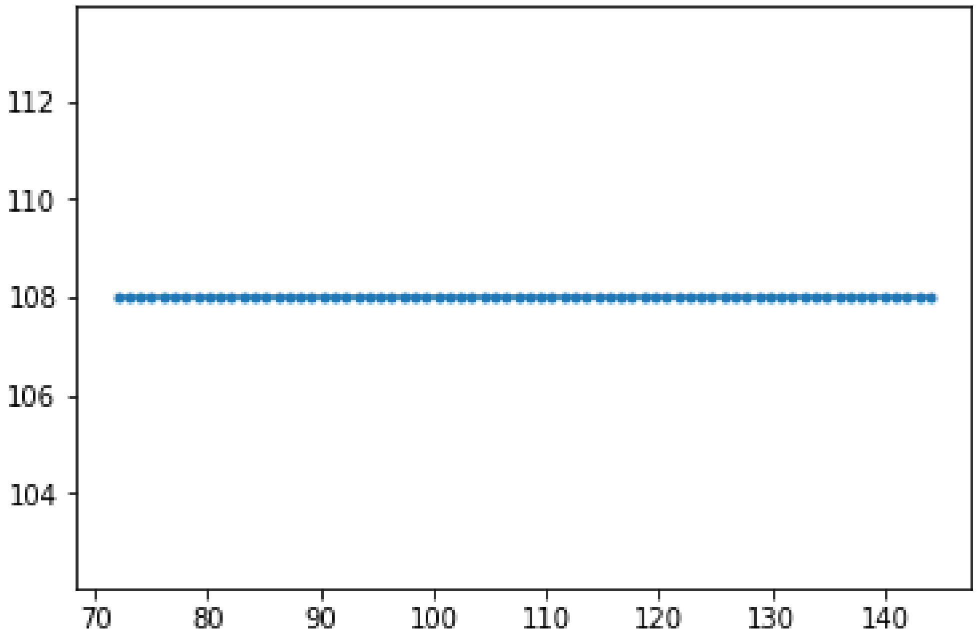


```
72
126
4
```

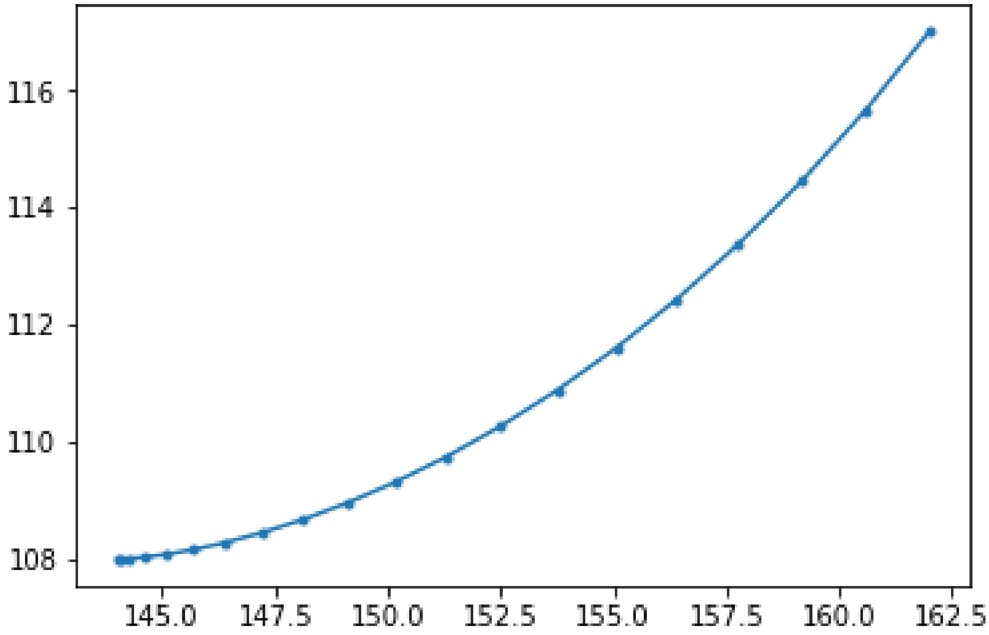

125
120
115
110
105
100
95
90
80
100
120
140
160
180


112
110
108
106
104
69
70
71
72
73
74
75

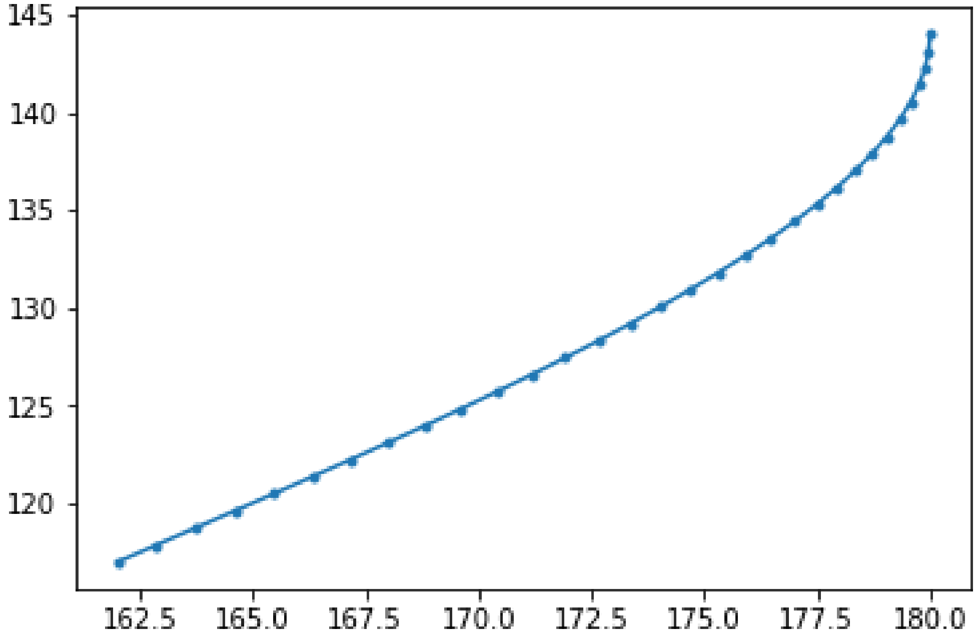
```
2
2
```


112
110
108
106
104
70
80
90
100
110
120
130
140

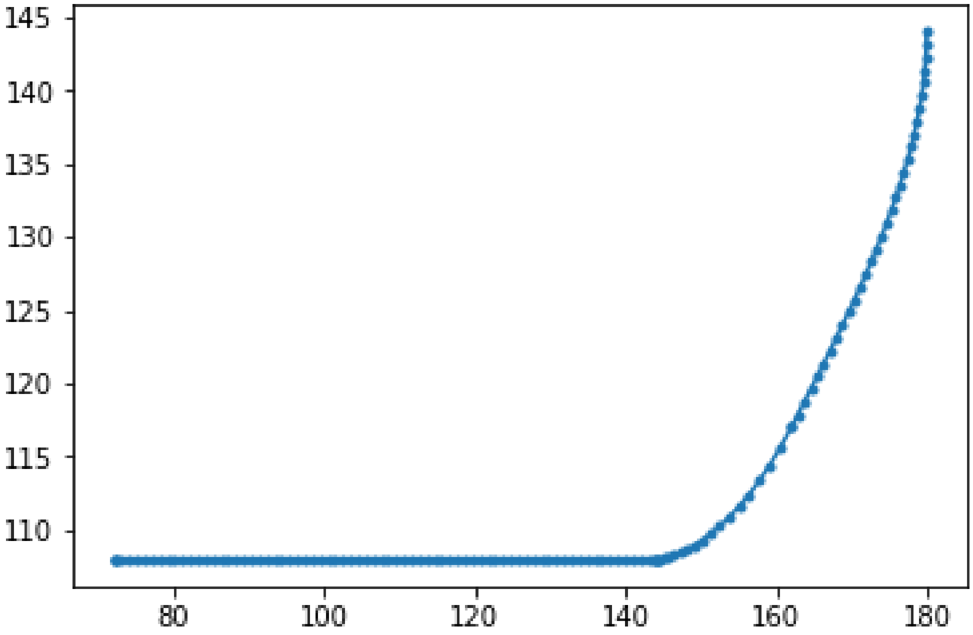
72
74


116
114
112
110
108
145.0
147.5
150.0
152.5
155.0
157.5
160.0
162.5

```
20
94
```

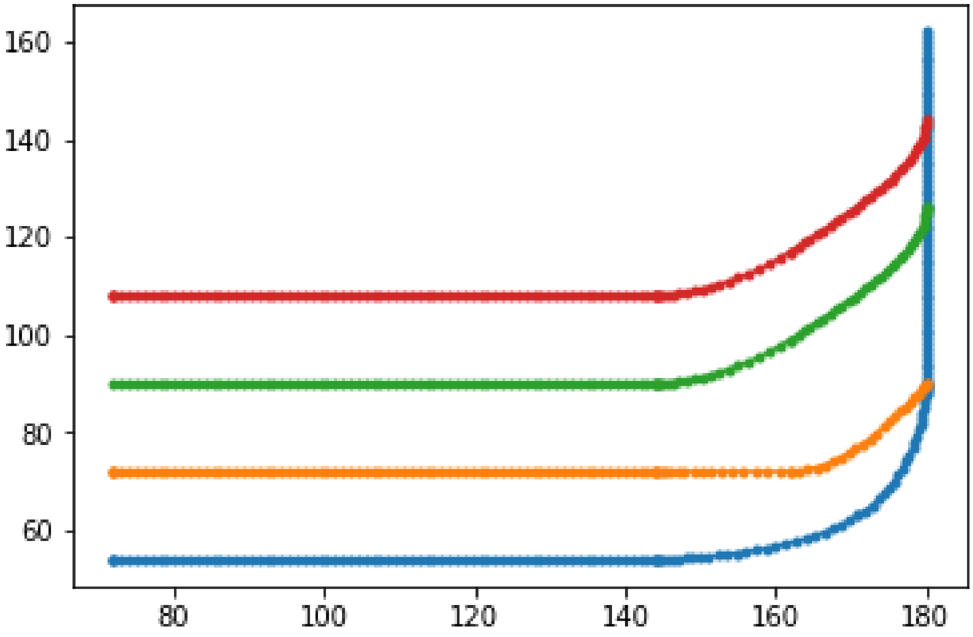

145
140
135
130
125
120
162.5
165.0
167.5
170.0
172.5
175.0
177.5
180.0

```
32
126
4
```



```
Out[5]: 4
```

```
In [8]: len(points_np_all)
        for k in range(len(points_np_all)):
            points_np=points_np_all[k]
            plt.plot(points_np[:, 0], points_np[:, 1], '.-')
        plt.show()
```



```
In [ ]:
```