Assignment08

May 16, 2019

Assignment08
Assignment08: Polynomial fitting
Software Engineering
20154652 Lee Dong Jae

```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
```

Define a polynomial curve

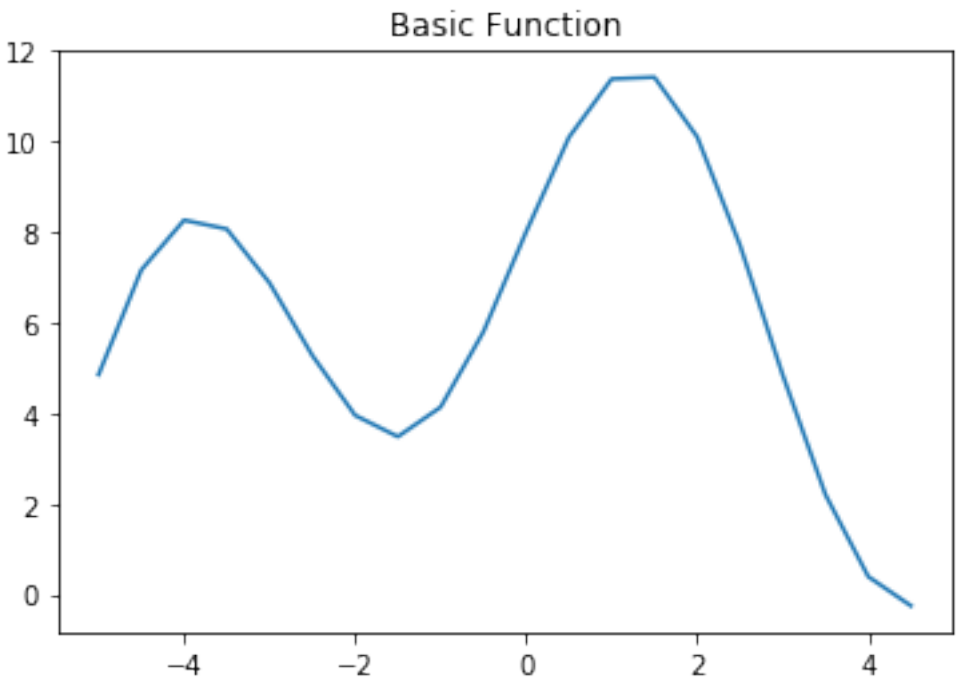
```
In [2]: def basic_f(x):
            f = (8*np.cos((1/4)*x)) - (4*np.sin((24)*x))
            return f
```

Define domain and codomain

```
In [3]: x = np.arange(-5,5, 0.5)
        y = basic_f(x)
```

Show original polynomial curve

```
In [4]: plt.title("Basic Function")
        plt.plot(x, y)
        plt.show()
```



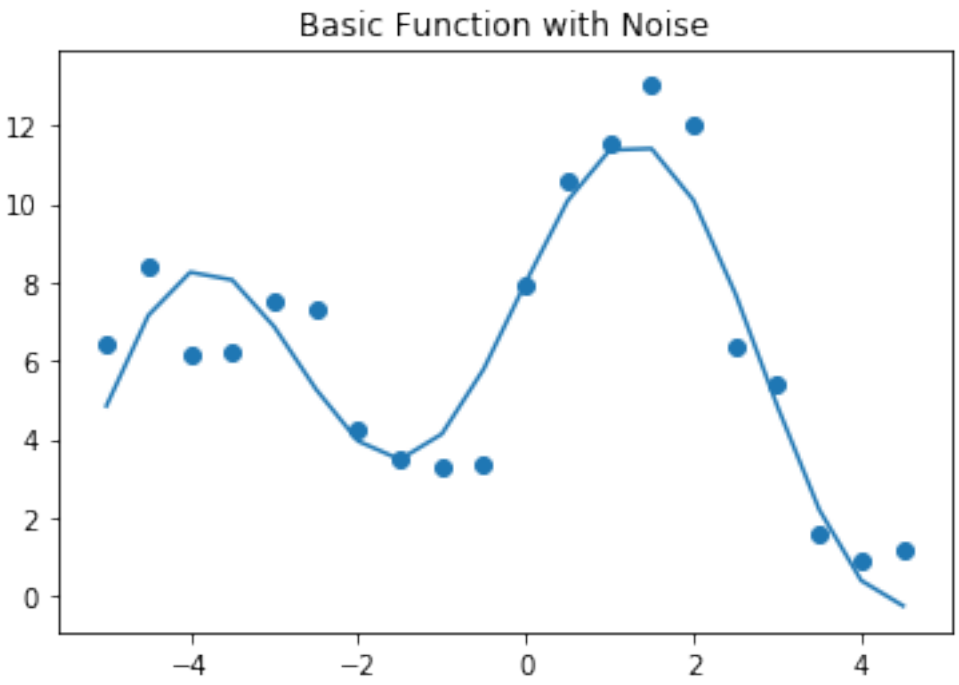
Generate points along the curve with random noise

```
In [5]: def make_noise(f_value):
            noise = np.zeros(20)
            for i in range(20):
                mean = f_value[i]
                std = 1
                temp_noise = np.random.normal(mean, std)
                noise[i] = temp_noise

            return noise
```

```
In [6]: noise = make_noise(y)
```

Plot the generated noisy points along with its original polynomial

```
In [7]: plt.title("Basic Function with Noise")
        plt.plot(x, y)
        plt.scatter(x, noise)
        plt.show()
```



In [8]:
```
info = {'energy_list': []}

#get a Degree - noise number Matrix

def get_matrix(degree, x):
    A = np.zeros((len(x), degree+1))
    for i in range(degree+1):
        for j in range(20):
            if i == 0:
                A[j][i] = 1
            else:
                A[j][i] = (-5+(0.5*j))**i
    return A
```

In [9]:
```
def get_distance(A, z, noise):
    distance = 0
    for i in range(20):
        distance += ( (np.dot(A[i],z)) - noise[i] )**2
    return distance
```

In [10]:
```
def plot_charts(info):
    plt.figure(figsize=(16, 8))
    plt.xticks(np.arange(0,20,1))
    plt.xlabel('Degree')
    plt.ylabel('(Ax-b)^2')
```

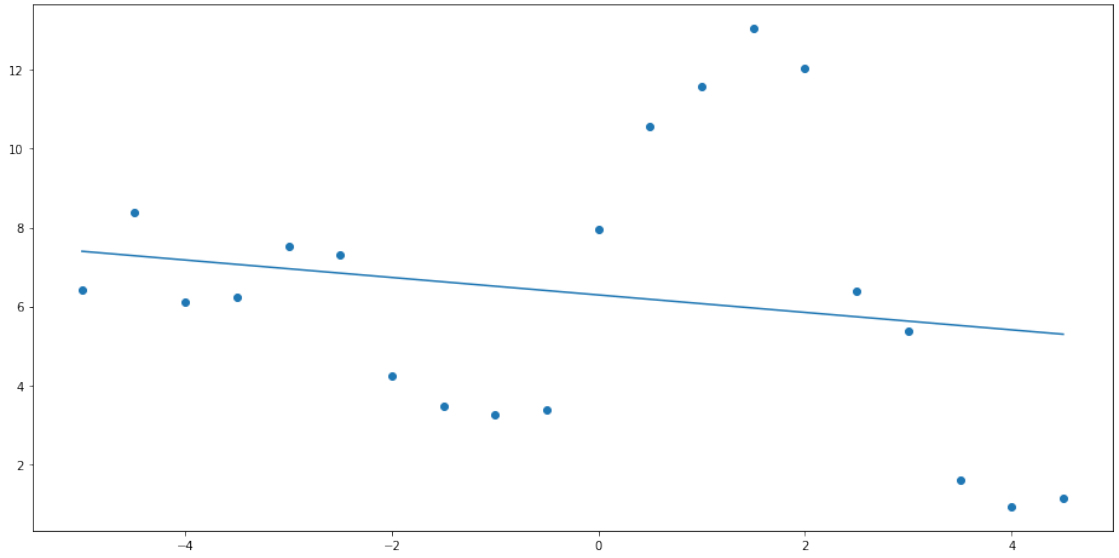
```
            plt.scatter(range(1,len(info['energy_list'])+1), info['energy_list'])
            plt.plot(range(1,len(info['energy_list'])+1), info['energy_list'])
            plt.show()
```

```
In [11]: def plot_curve(degree, distance, z, noise):
            plt.figure(figsize=(16,8))
            print('Degree = ', degree)
            print('Distance = ', distance)
            plt.scatter(x, noise)
            plt.plot(x, np.dot(A,z))
            plt.show()
```

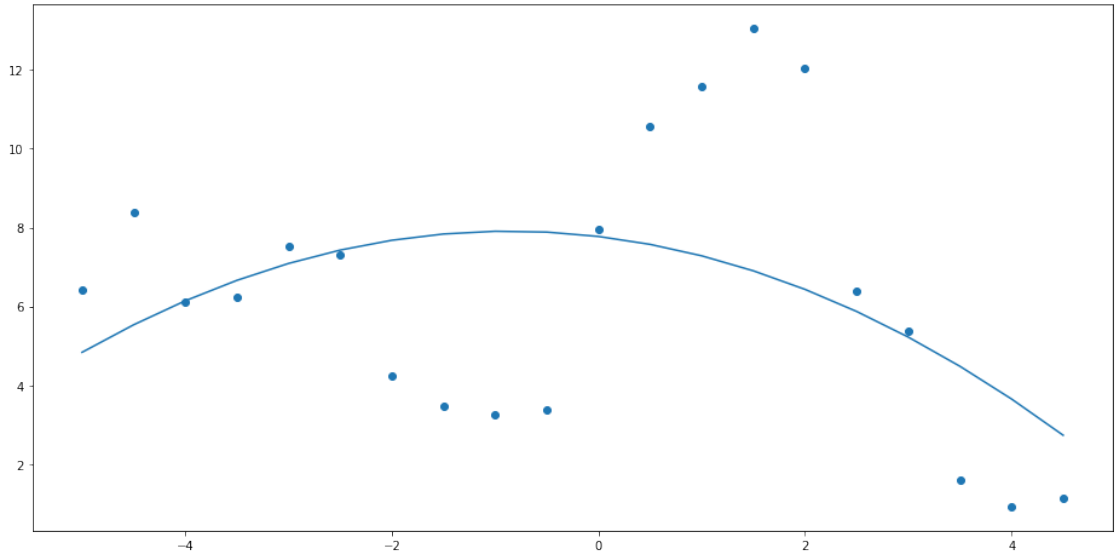
Plot the approximating polynomial curve with varying polynomial degree

```
In [12]: degree = 1
         while(True):
            A = get_matrix(degree, x)

            #do a QR decomposition
            q, r = np.linalg.qr(A)
            r_inverse = np.linalg.inv(r)
            temp_z = np.dot(r_inverse, q.T)

            #obtain co-efficients
            z = np.dot(temp_z, noise)


            dis = get_distance(A, z, noise)
            info['energy_list'].append(dis)
            plot_curve(degree, dis, z, noise)

            #if the number of equation > value num, stop
            if degree == 19:
                print('\n\n\n')
                plot_charts(info)
                break
            degree += 1
```

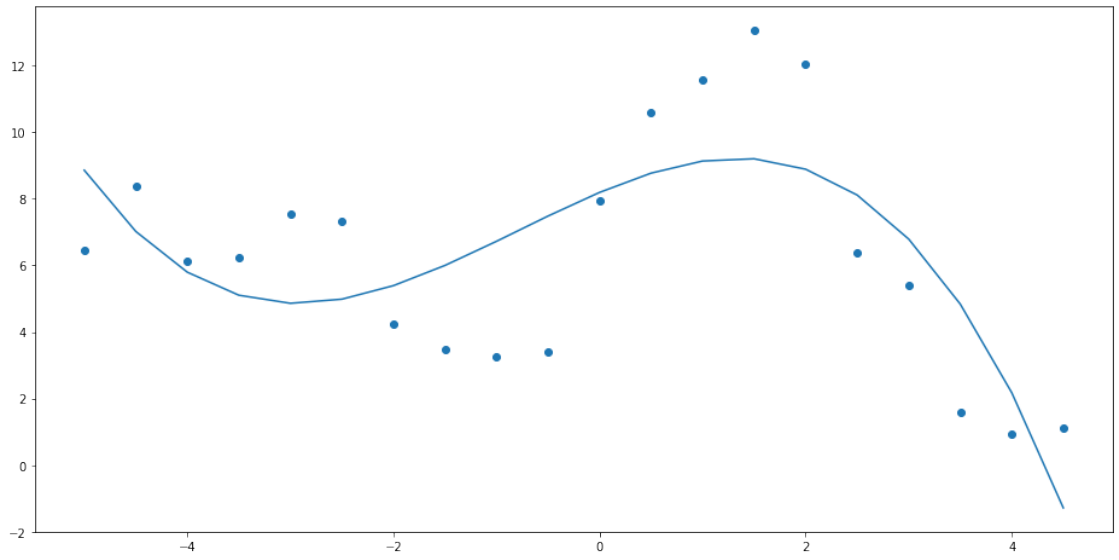
```
Degree =  1
Distance =  234.6093312027995
```



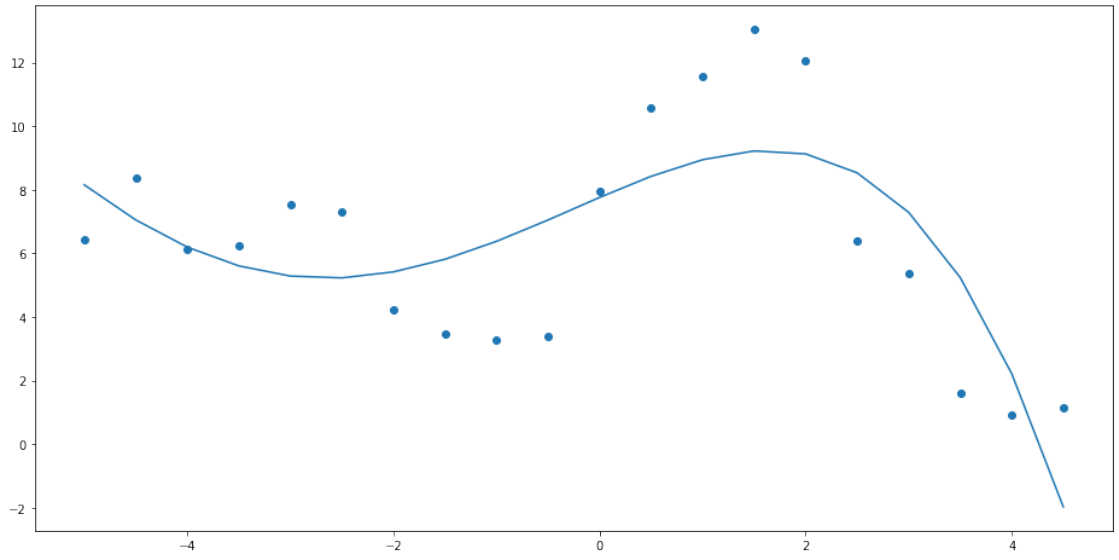

```
Degree =  2
Distance =  199.15580781931413
```



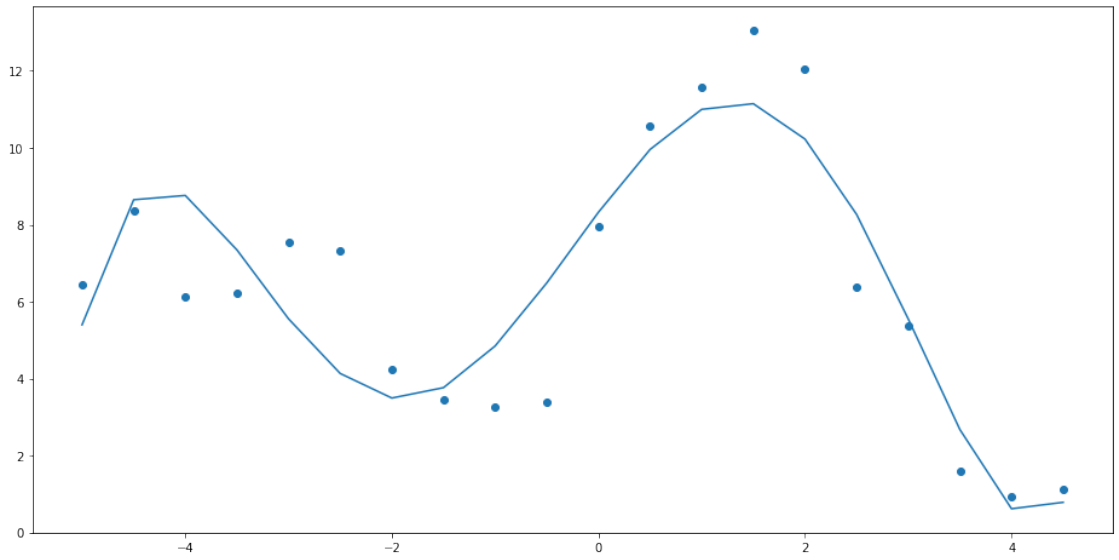

```
Degree =  3
Distance =  115.04169799534866
```



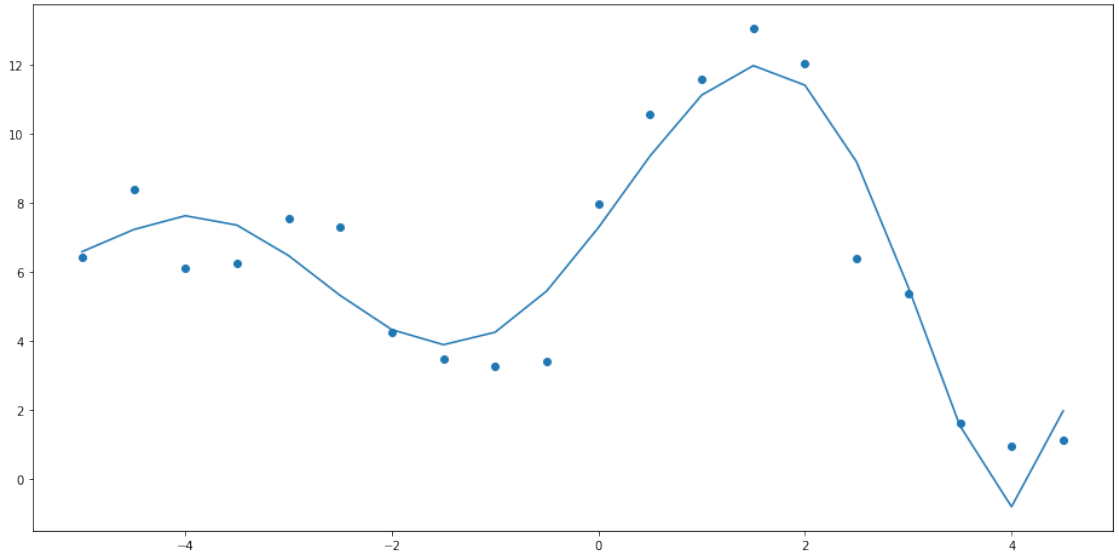

```
Degree =  4
Distance =  112.08735036561448
```



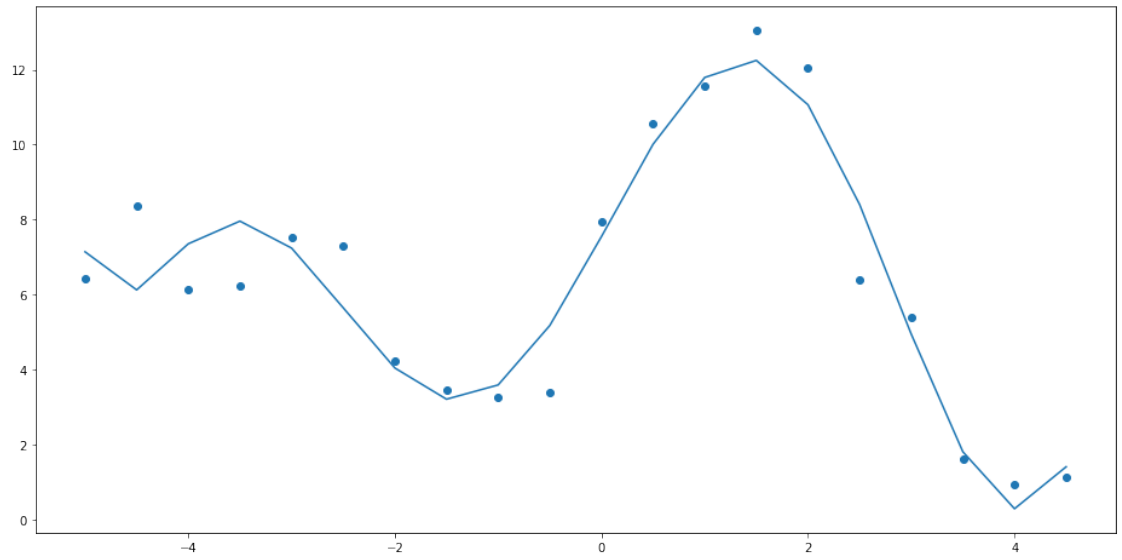

```
Degree =  5
Distance =  48.900053776271804
```



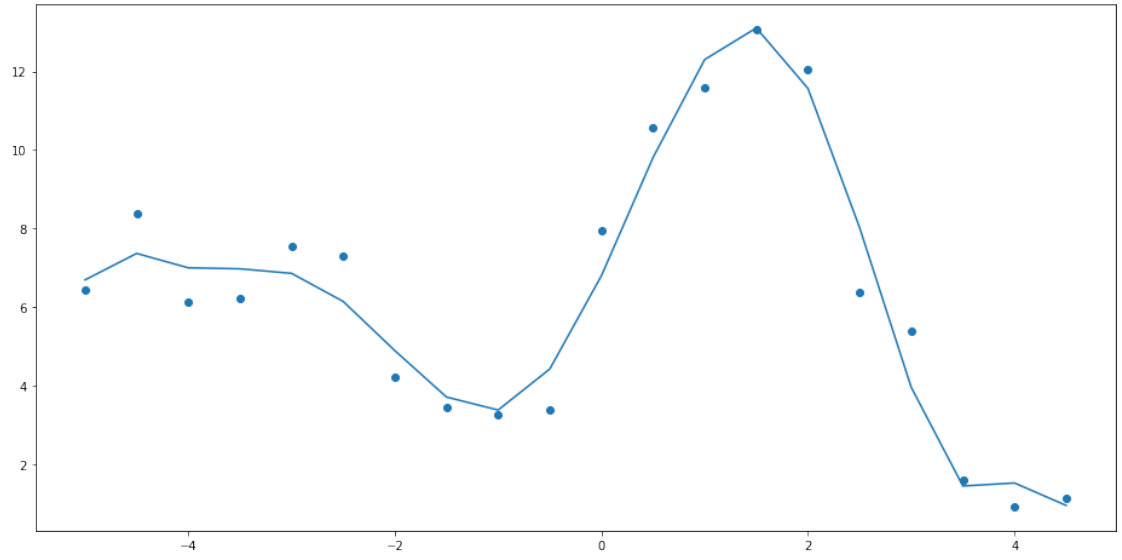

```
Degree =  6
Distance =  30.691346785236483
```



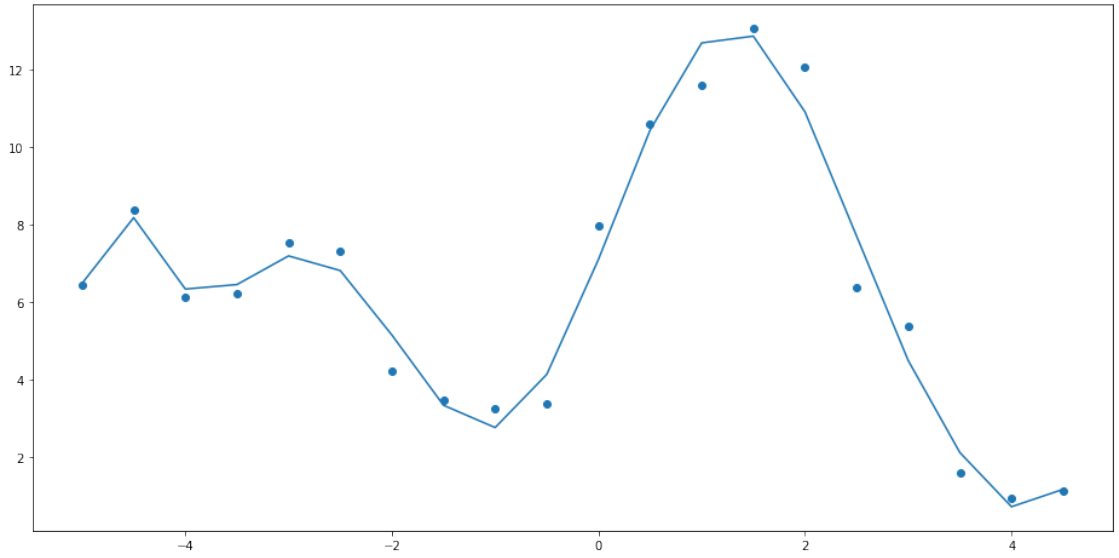

```
Degree =  7
Distance =  23.25826058399508
```



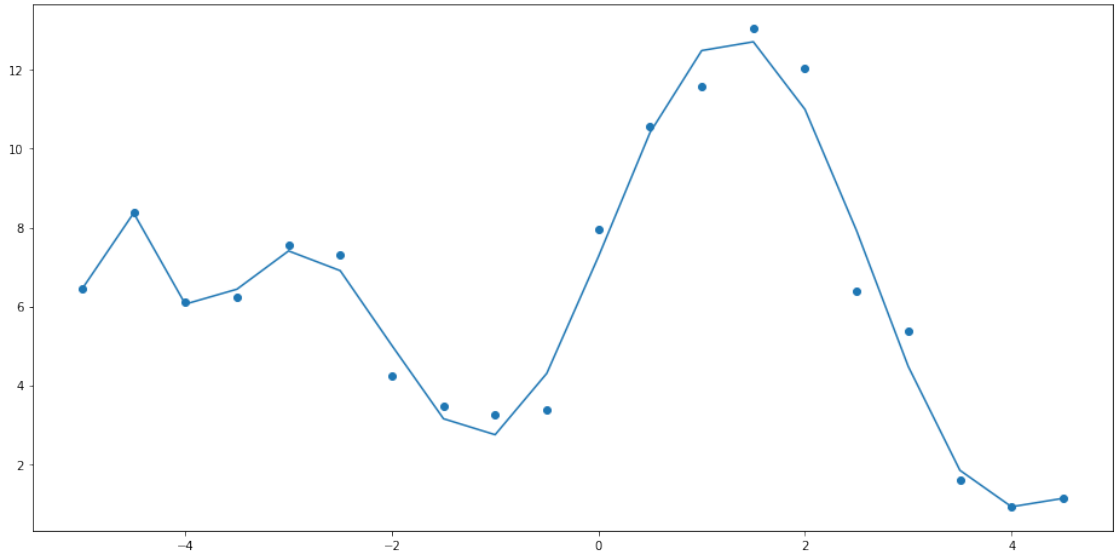

```
Degree =  8
Distance =  13.625430736466608
```



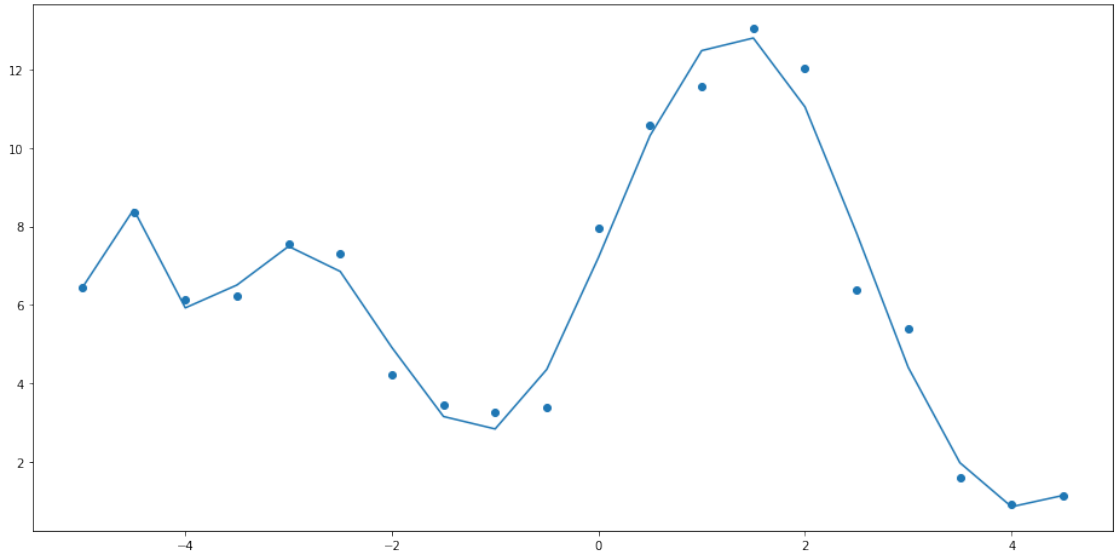

```
Degree =  9
Distance =  8.332798634186096
```



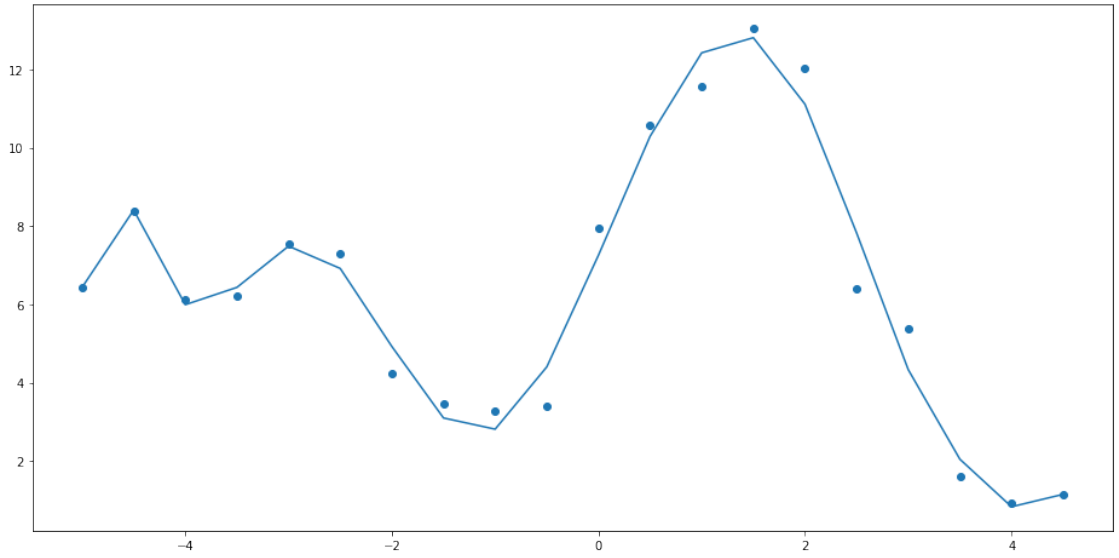

```
Degree =  10
Distance =  7.819371978611616
```



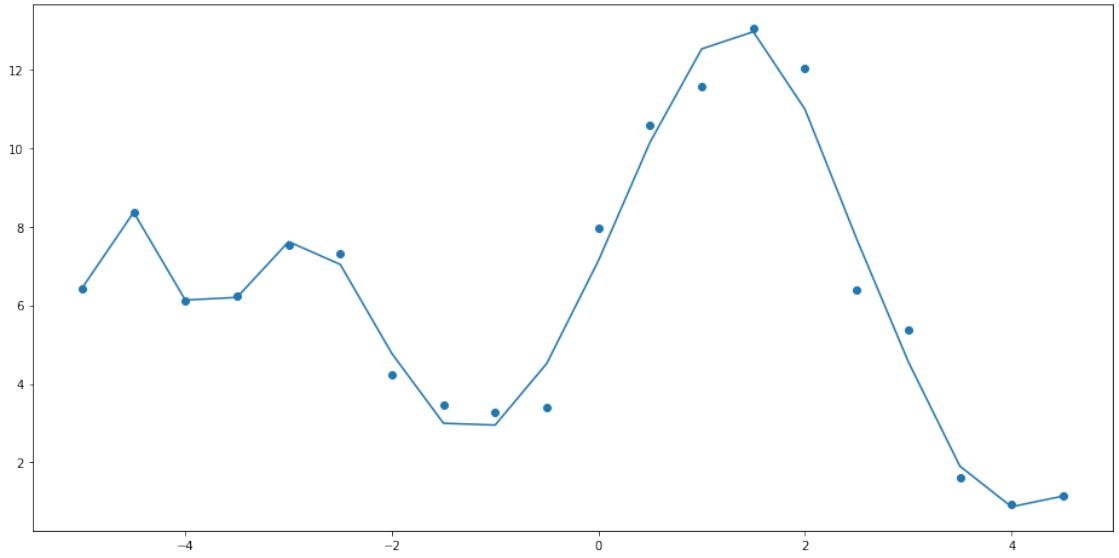

```
Degree =  11
Distance =  7.703779936829851
```



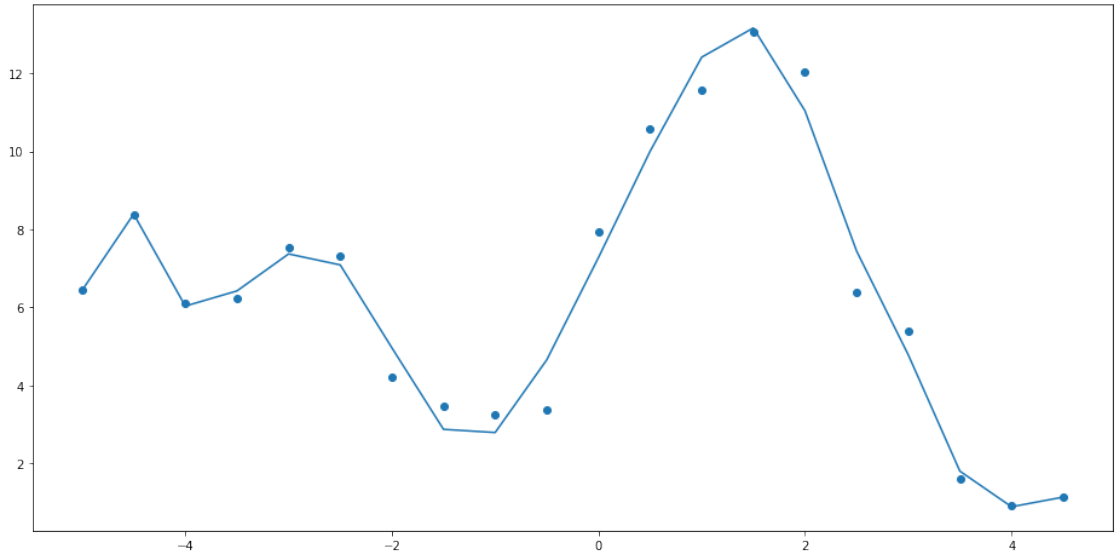

```
Degree =  12
Distance =  7.66207688240814
```



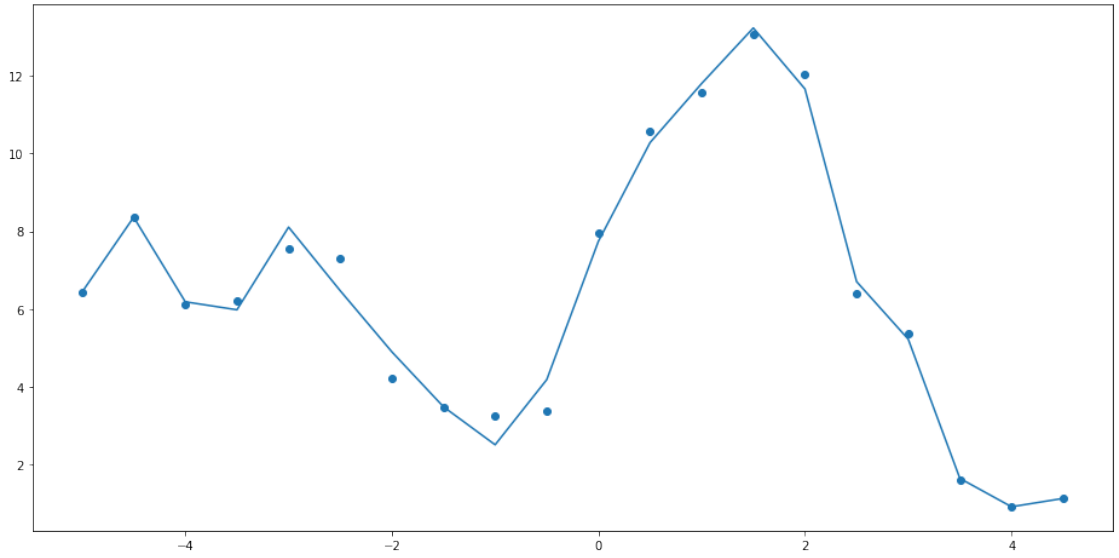

```
Degree =  13
Distance =  7.312203980408039
```



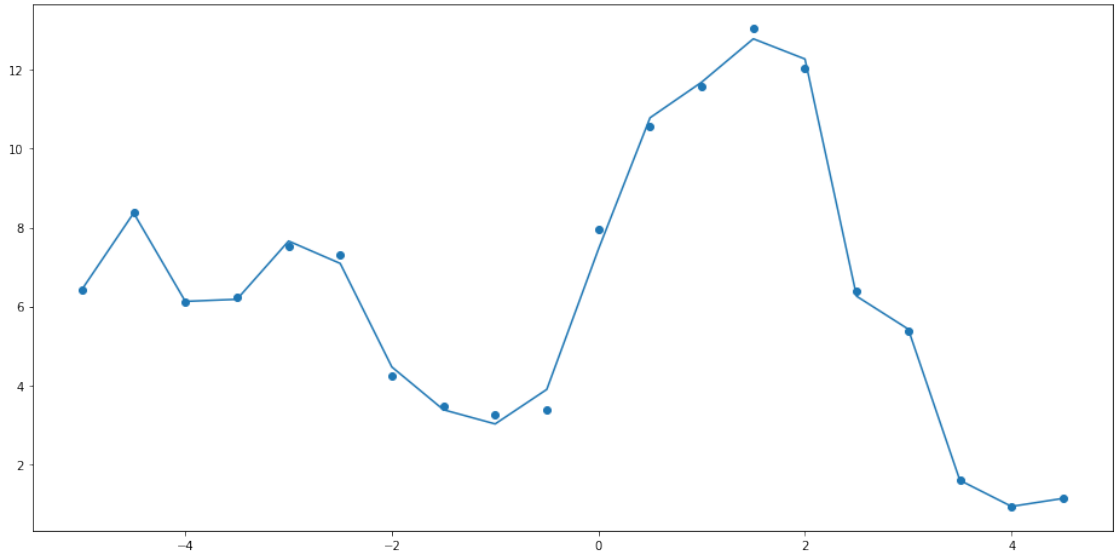

```
Degree =  14
Distance =  6.873693932991438
```



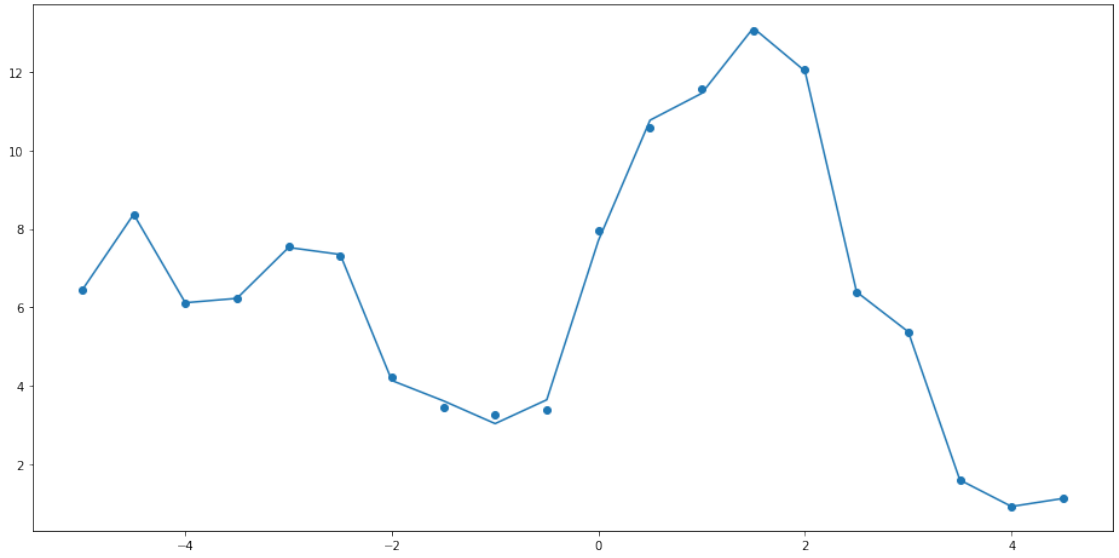

```
Degree =  15
Distance =  3.237918983822286
```



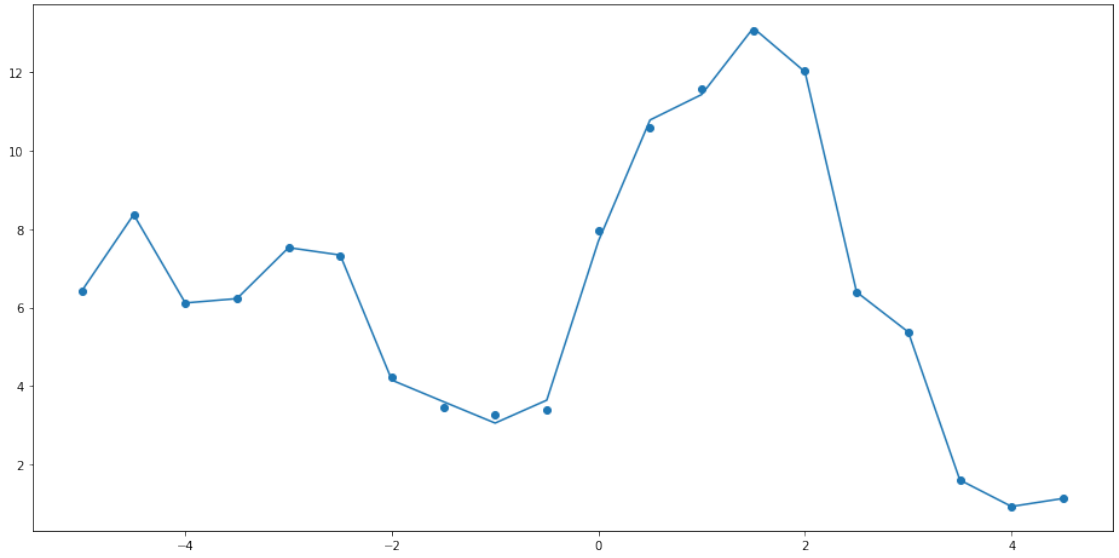

```
Degree =  16
Distance =  0.8966185018622851
```



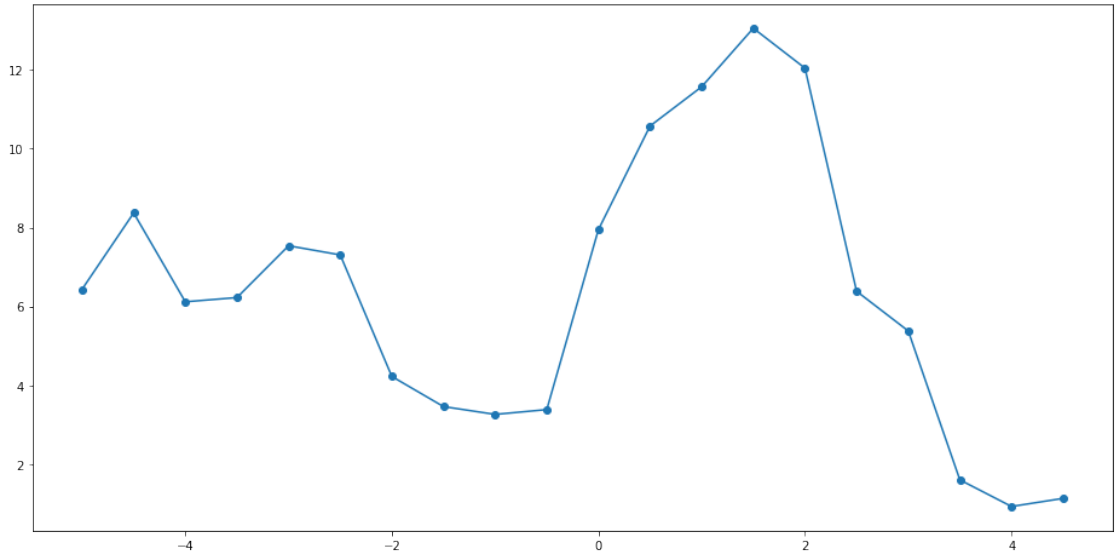

```
Degree =  17
Distance =  0.2695317487629174
```

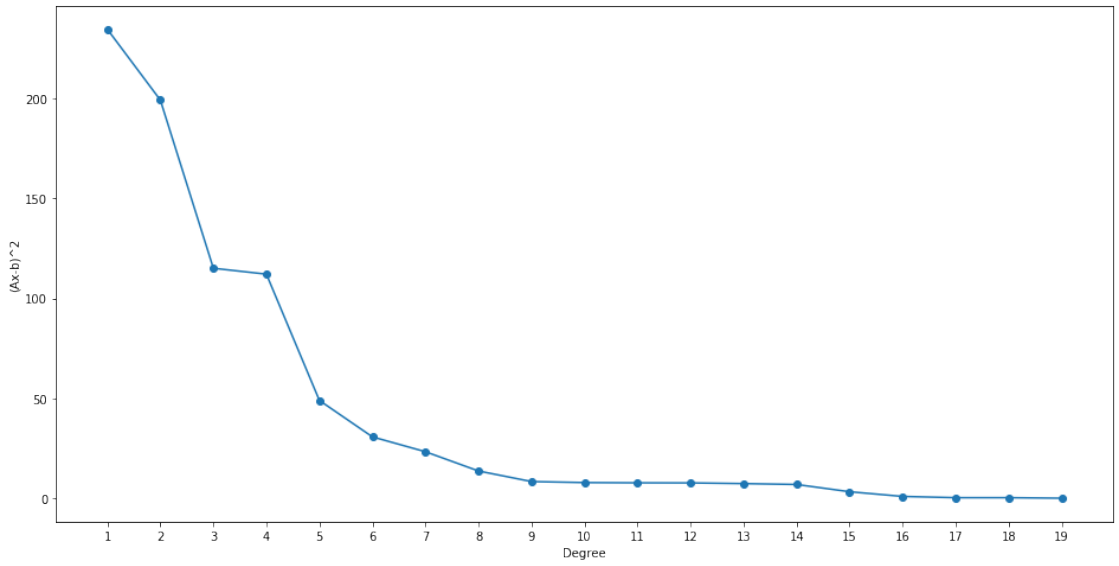



```
Degree =  18
Distance =  0.26774831473700994
```




```
Degree =  19
Distance =  7.455708107684078e-15
```


12
10
8
6
4
2
−4
−2
0
2
4


200
150
100
50
0
(Ax-b)^2
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
Degree

In [ ]: