Assignment08

May 20, 2019

20164245 Hong Jin

[Polynomial fitting]

Solve a least square problem to find an optimal polynomial curve for a given set of two dimensional points.

Demonstrate the effect of the degree of polynomial in fitting a given set of points.

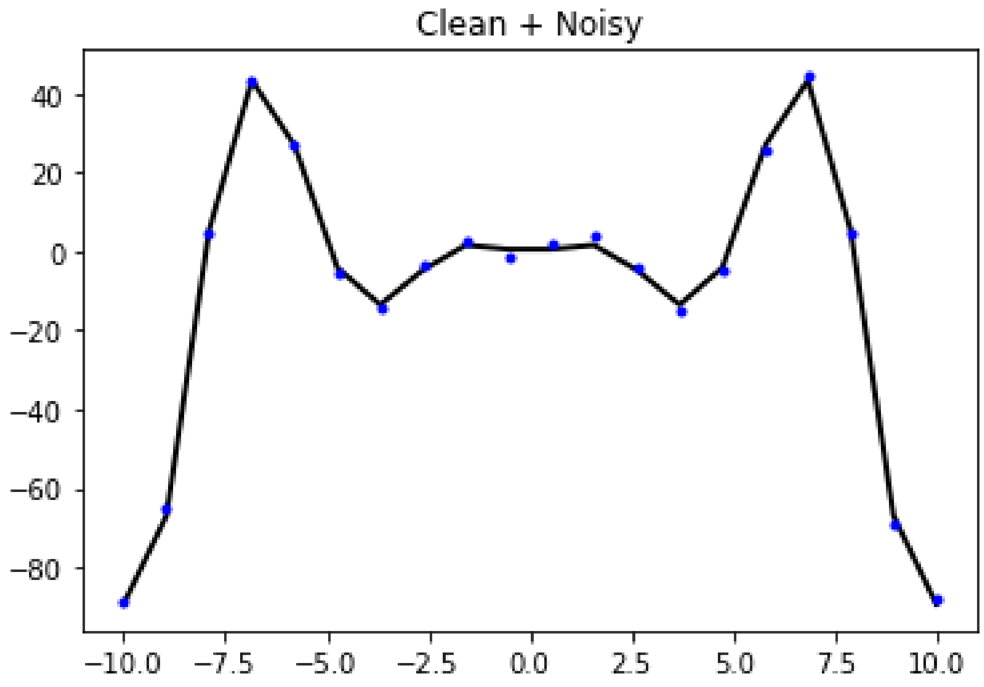
- choose a polynomial curve and generate points along the curve with random noise
- plot the generated noisy points along with its original polynomial without noise
- plot the approximating polynomial curve obtained by solving a least square problem
- plot the approximating polynomial curve with varying polynomial degree

1 Set up

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

        num = 20
        std = 5
        error = []

        def fun(x):
            f = x**2 * np.cos(x) + x * np.sin(x)
            return f
```

2 Clean and Noisy Data

```
In [2]: n = np.random.rand(num)
        n = n - np.mean(n)
        x = np.linspace(-10, 10, num)
        y1 = fun(x)
        y2 = y1 + n * std

        plt.title("Clean + Noisy")
        plt.plot(x, y1,color = 'black', linewidth = 2)
        plt.plot(x, y2, 'b.', linewidth = 2)
        plt.show()
```



3 Define Matrix A

$f_i(x) = x^{i-1}, i = 1, \ldots, p$

$\hat{f}(x) = \theta_1 + \theta_2 x + \cdots + \theta_p x^{p-1}$

$$A = \begin{bmatrix} 1 & x^{(1)} & \cdots & (x^{(1)})^{p-1} \\ 1 & x^{(2)} & \cdots & (x^{(2)})^{p-1} \\ \vdots & \vdots & & \vdots \\ 1 & x^{(N)} & \cdots & (x^{(N)})^{p-1} \end{bmatrix}$$

(x^i means scalar x to ith power; $x^{(i)}$ is ith data point)

$\theta = (A^T A)^{-1} A^T b$

```
In [3]: def defMatrix(p_num):
            model = np.zeros((p_num+1, num))
            for i in range(p_num):
                model[i] = x ** (p_num-i)
            model[p_num] = 1
            return np.matrix(np.transpose(model))
```

4 Compute $\hat{y}$

```
In [4]: def computeYhat(parameter, p_num):
            y_hat = 0
            for j in range(len(parameter)):
                y_hat +=  parameter[j][0] * x ** (p_num - j)

            return y_hat
```

5 Compute Residual with Least squares

$r_i = y^{(i)} - \hat{y}^{(i)}$

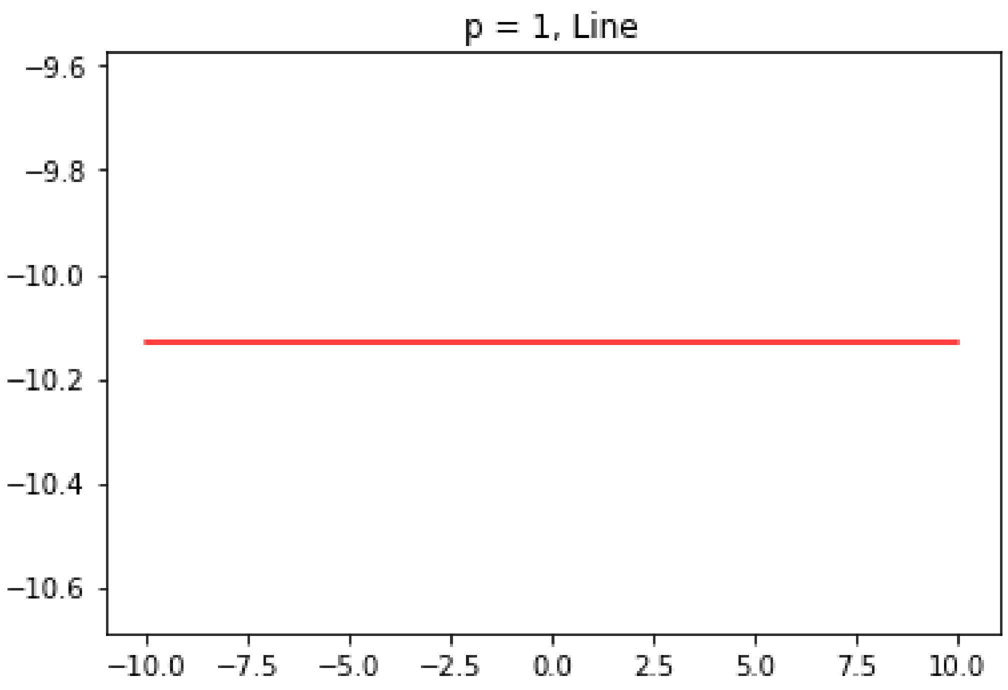
```
In [5]: def computeResidual(f_hat, y):
            return sum((f_hat - y)**2)
```

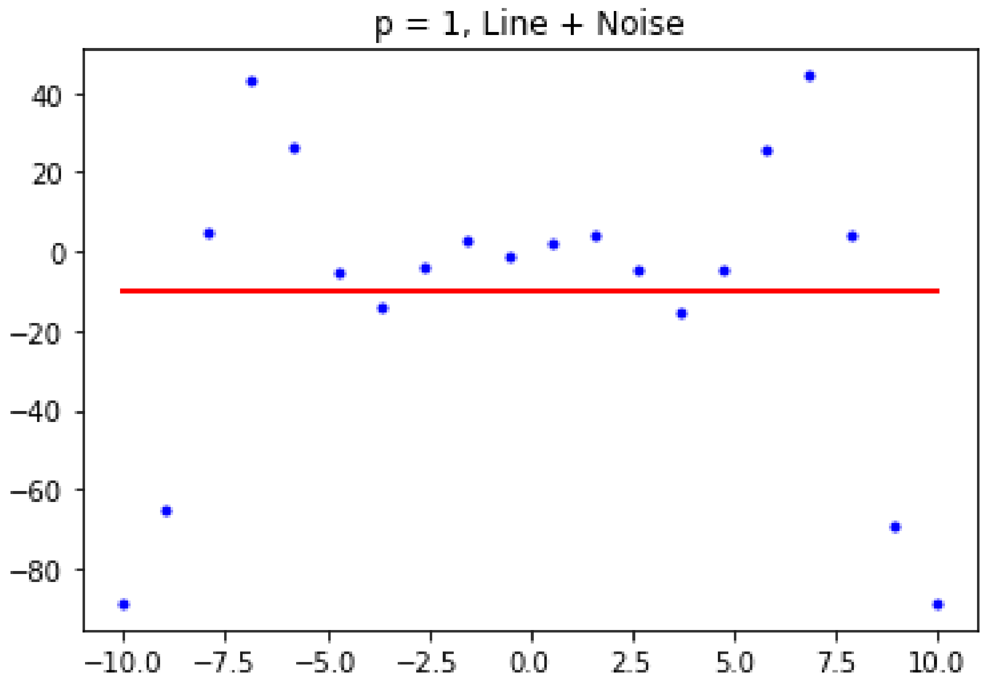
6 Plot graph

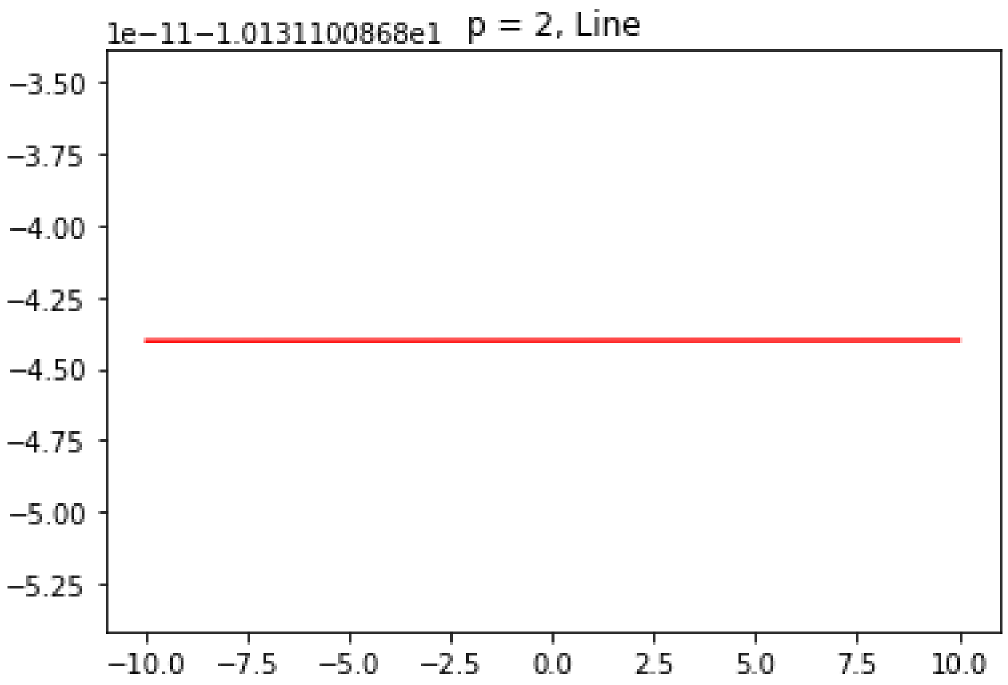
```
In [6]: def plotGraph(n, y_hat):
            plt.title("p = " + str(n+1) + ", Line")
            plt.plot(x, y_hat, 'r')
            plt.show()
            plt.title("p = " + str(n+1) + ", Line + Noise")
            plt.plot(x, y_hat, 'r', x, y2, 'b.', linewidth = 2)
            plt.show()
```

7 Polynomial Fitting

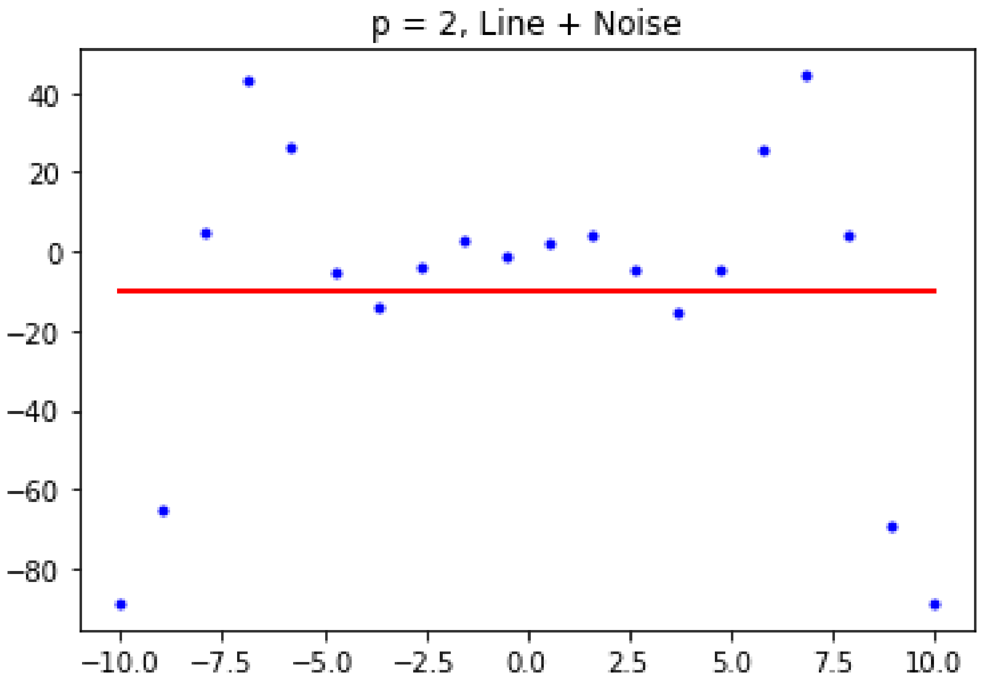
```
In [7]: for i in range(num):
            A = defMatrix(i)
            b = np.matrix(y1)

            theta = (A.T * A).I* A.T*b.T
            theta = np.asarray(theta)

            y_hat = computeYhat(theta, i)

            plotGraph(i, y_hat)

            res = computeResidual(y_hat, y2)
            error.append(res)
```


p = 1, Line
−9.6
−9.8
−10.0
−10.2
−10.4
−10.6
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

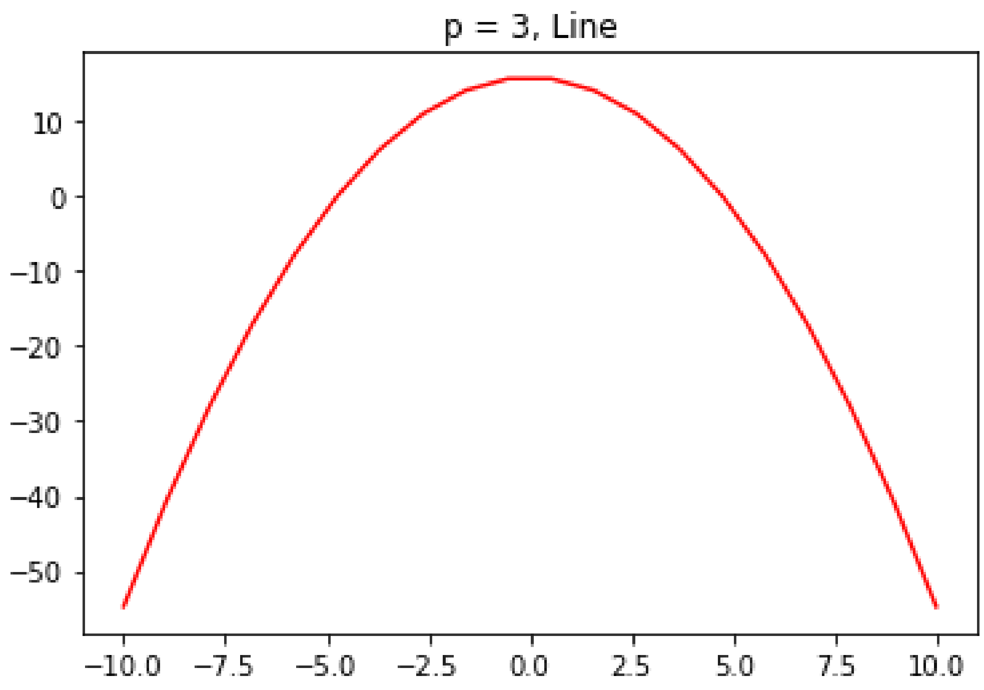

p = 1, Line + Noise
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

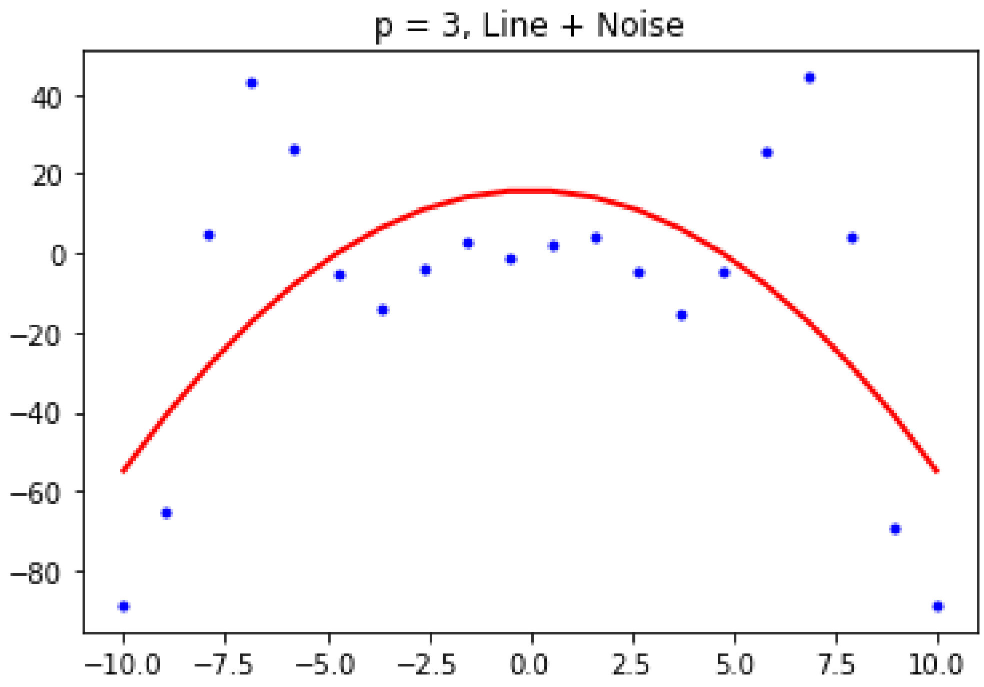


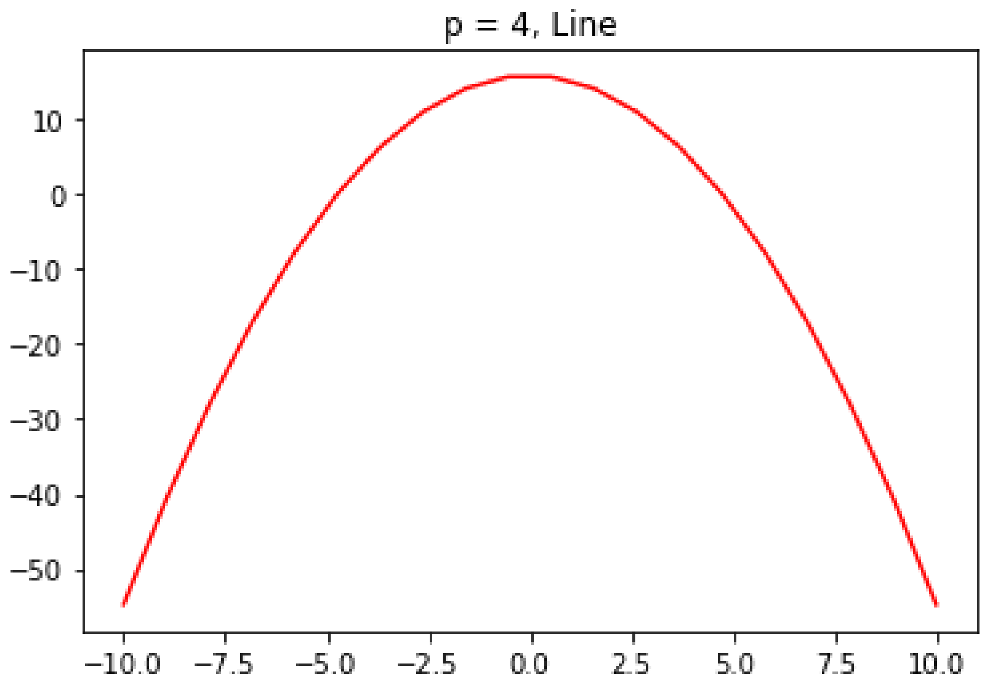
1e−11−1.0131100868e1
p = 2, Line
−3.50
−3.75
−4.00
−4.25
−4.50
−4.75
−5.00
−5.25
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

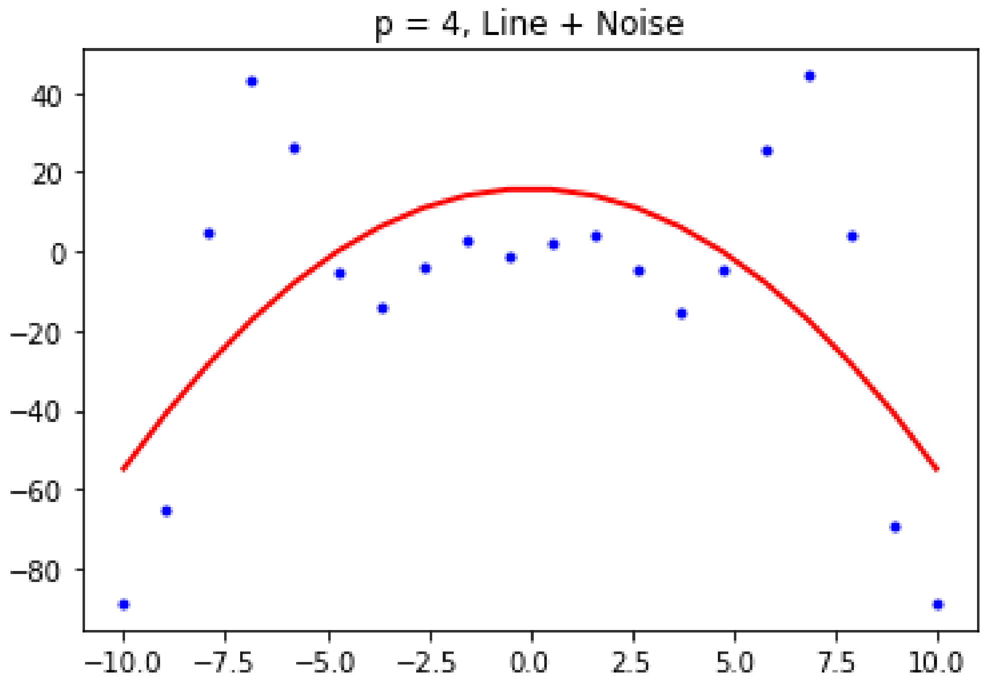


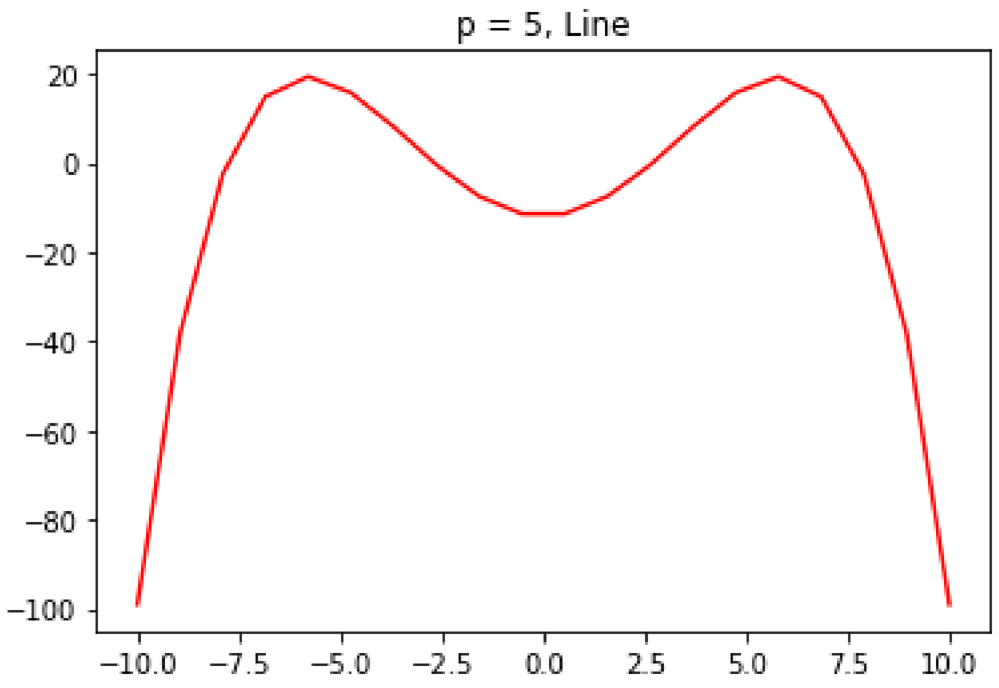
p = 2, Line + Noise
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

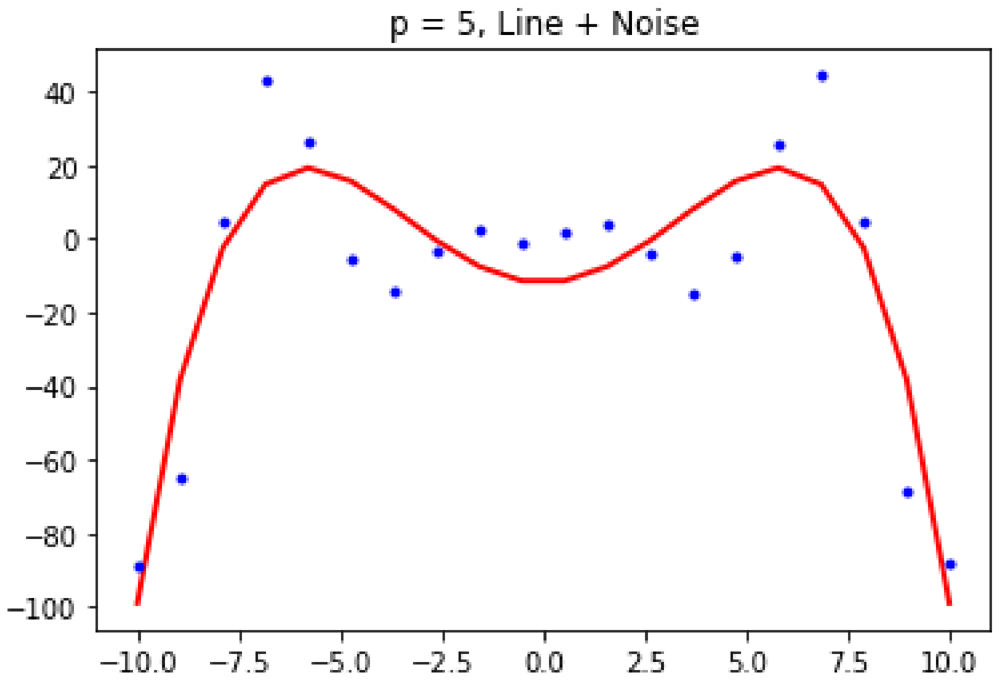

p = 3, Line
10
0
−10
−20
−30
−40
−50
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

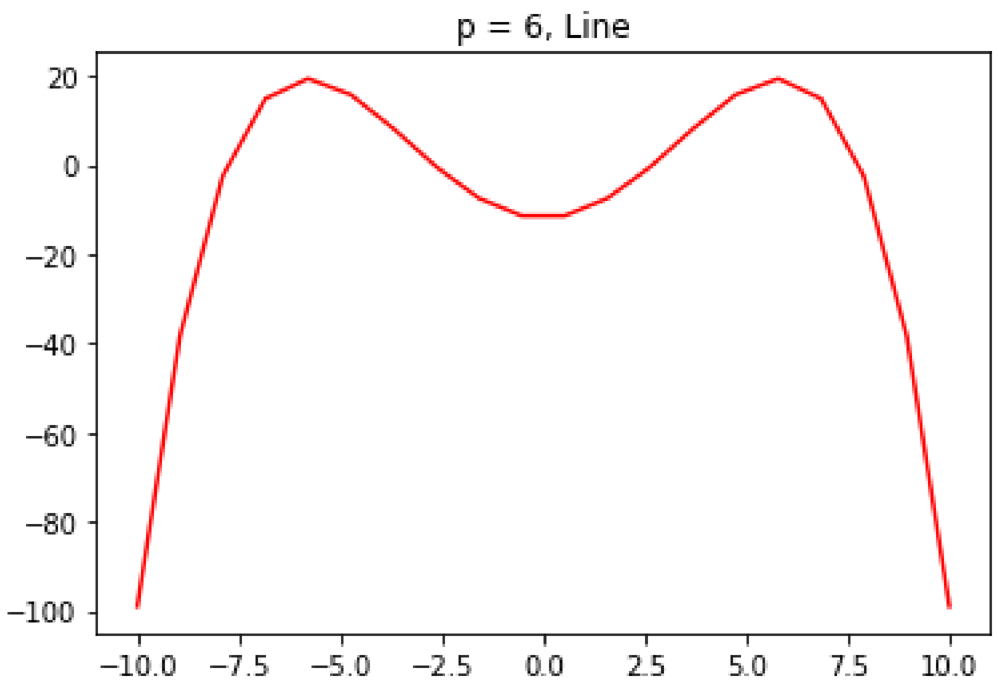

p = 3, Line + Noise
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

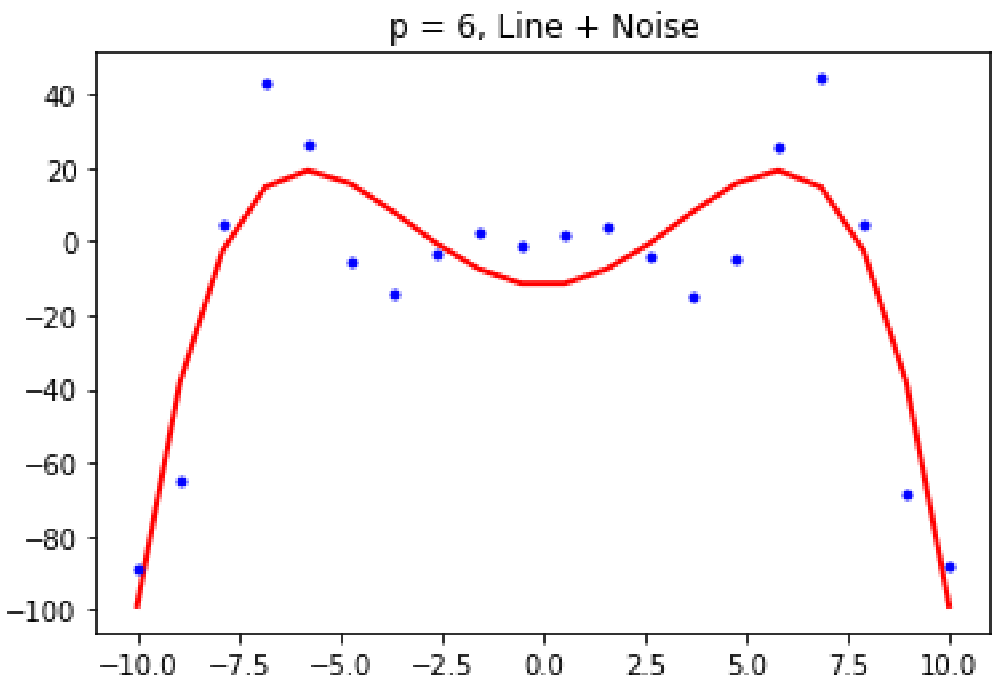

p = 4, Line
10
0
−10
−20
−30
−40
−50
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

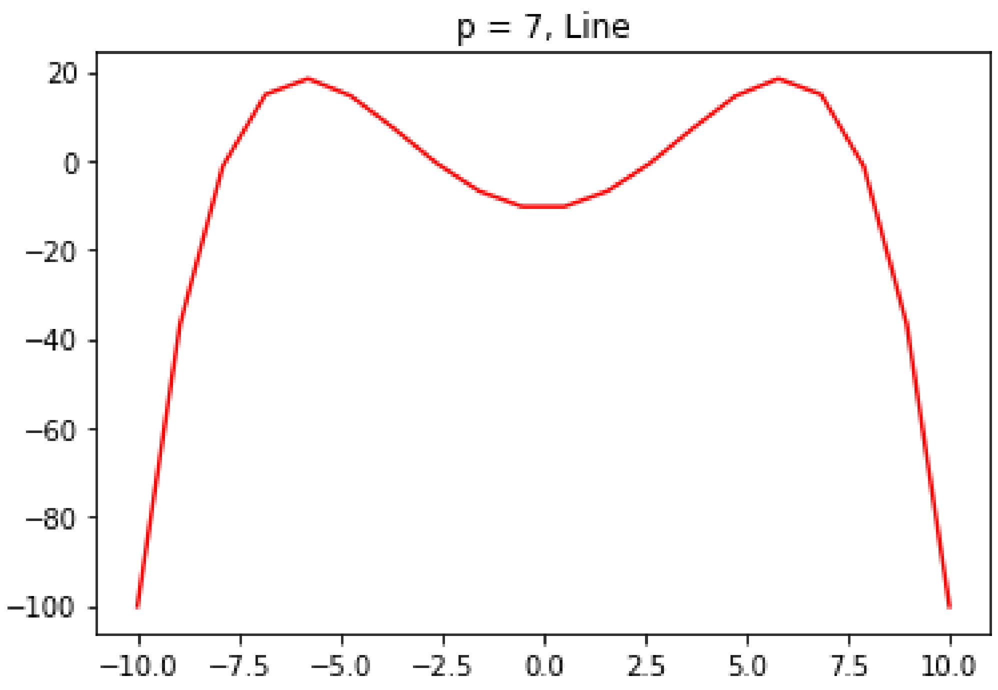

p = 4, Line + Noise
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

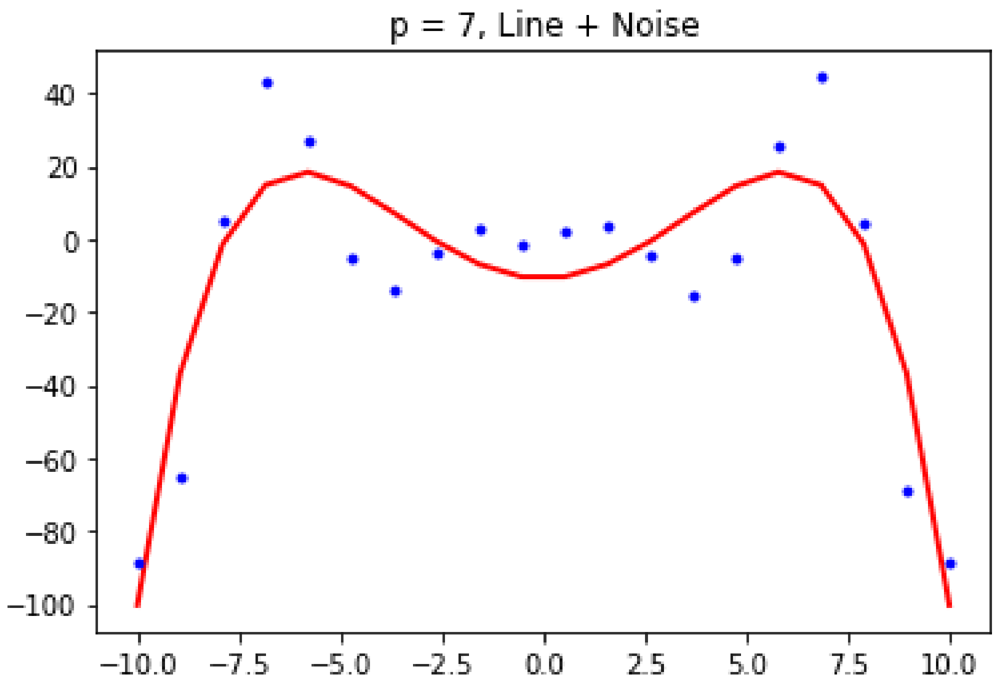

p = 5, Line
20
0
−20
−40
−60
−80
−100
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

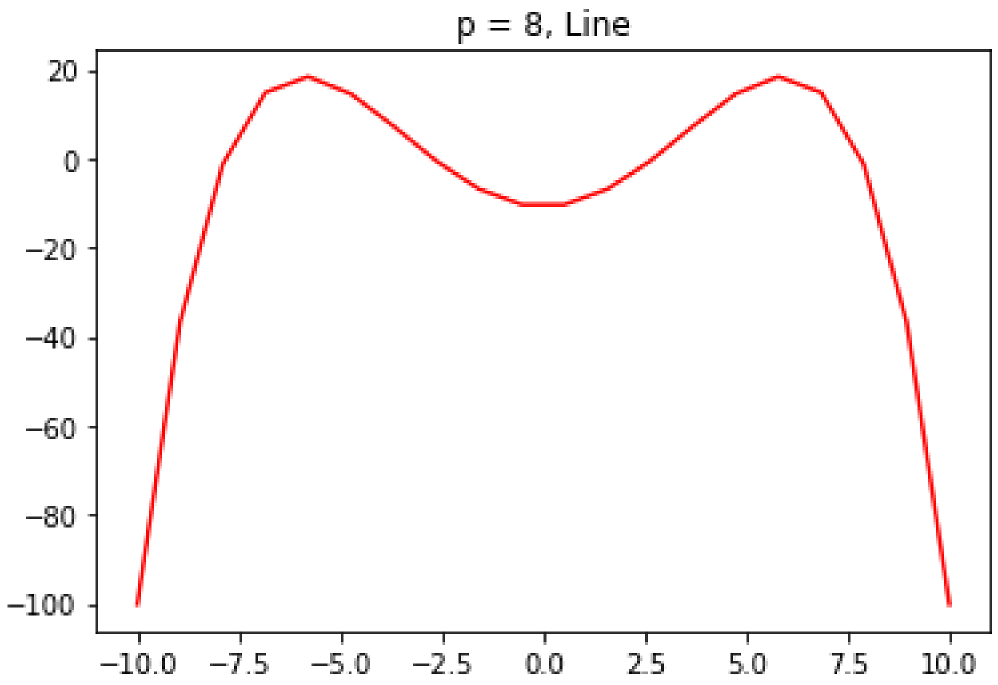

p = 5, Line + Noise
40
20
0
−20
−40
−60
−80
−100
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

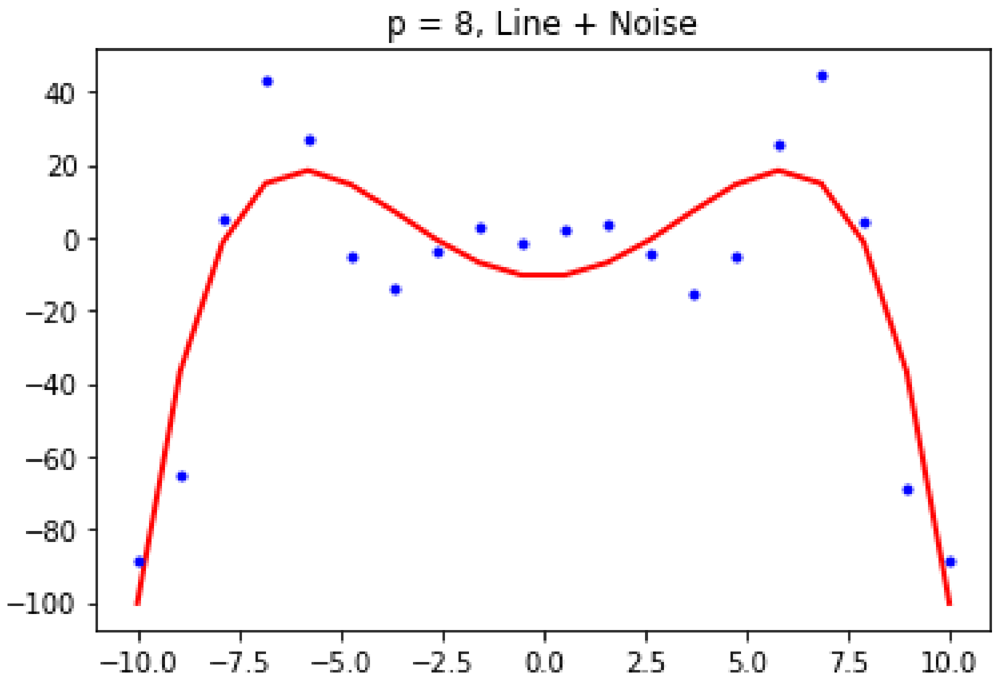

p = 6, Line
20
0
−20
−40
−60
−80
−100
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

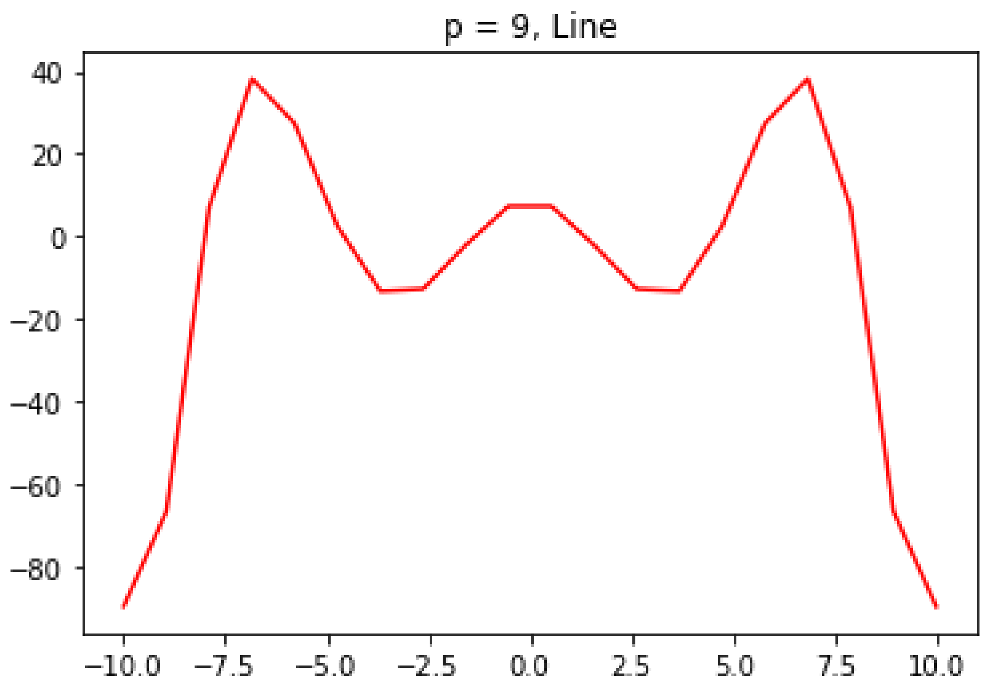

p = 6, Line + Noise
40
20
0
−20
−40
−60
−80
−100
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

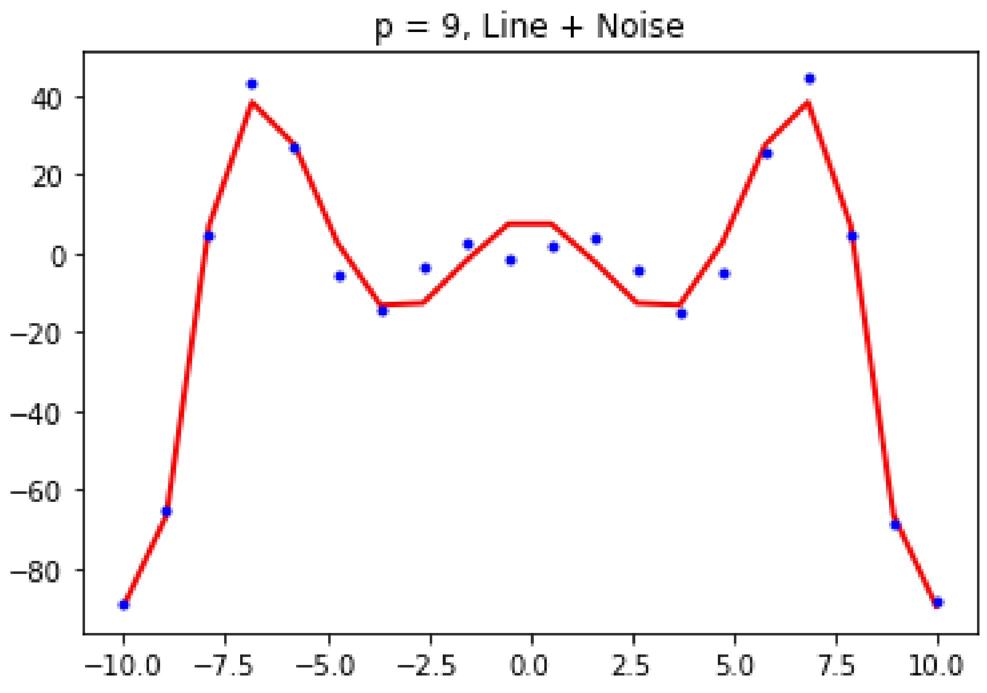

p = 7, Line
20
0
−20
−40
−60
−80
−100
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

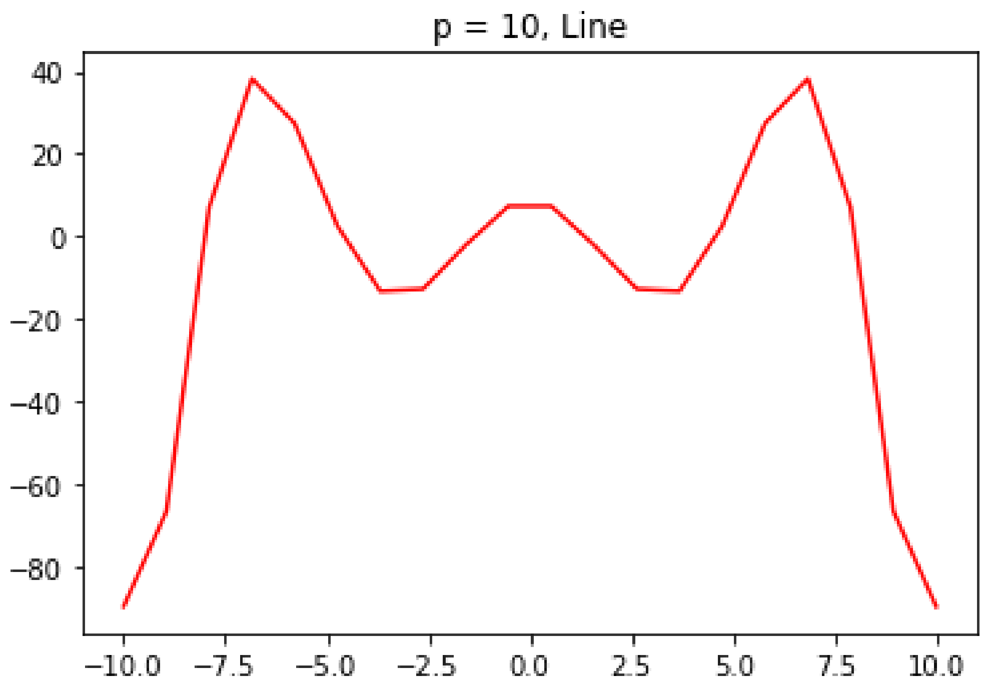

p = 7, Line + Noise
40
20
0
−20
−40
−60
−80
−100
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

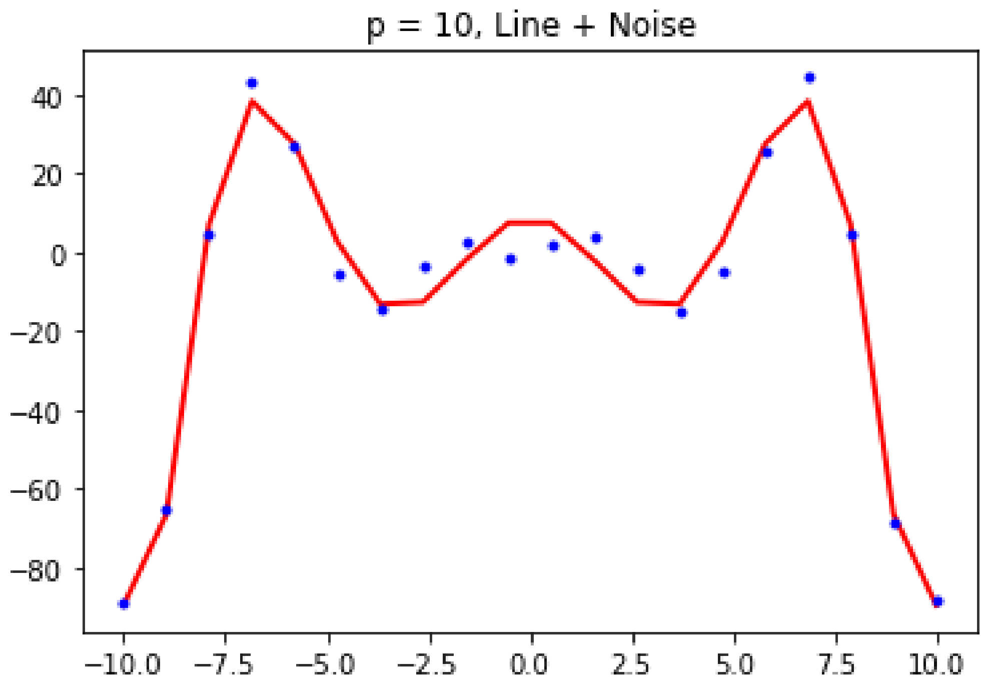

p = 8, Line
20
0
−20
−40
−60
−80
−100
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

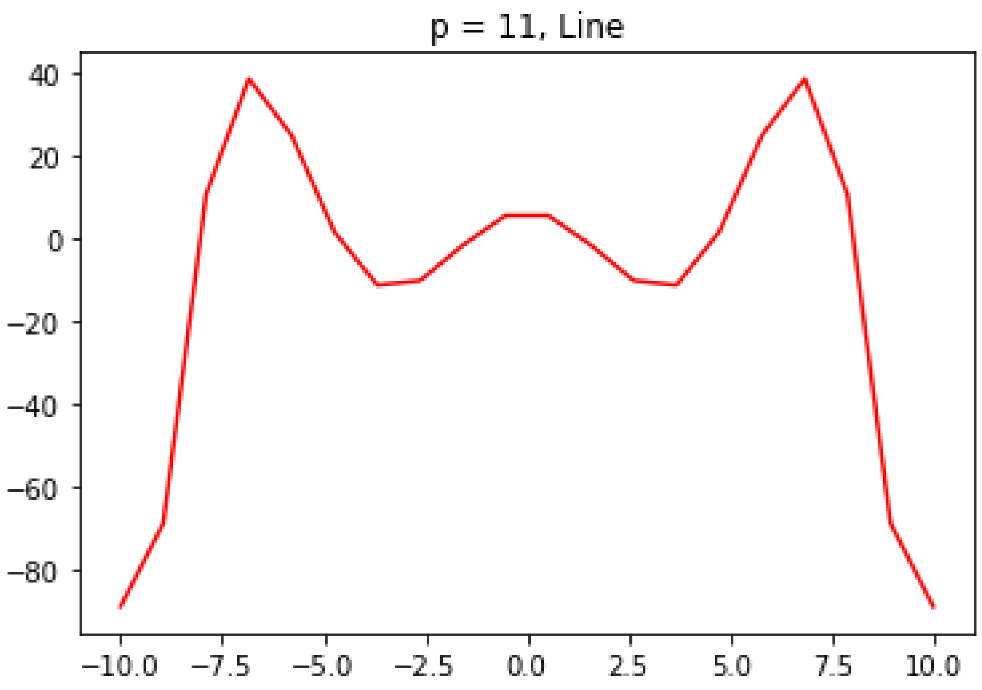

p = 8, Line + Noise
40
20
0
−20
−40
−60
−80
−100
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

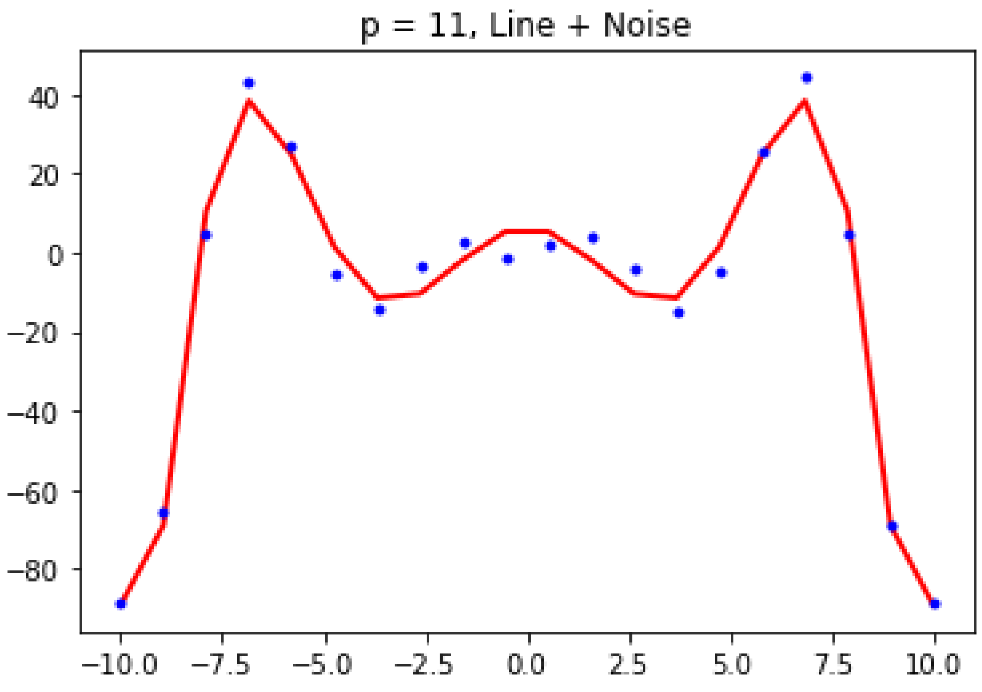

p = 9, Line
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

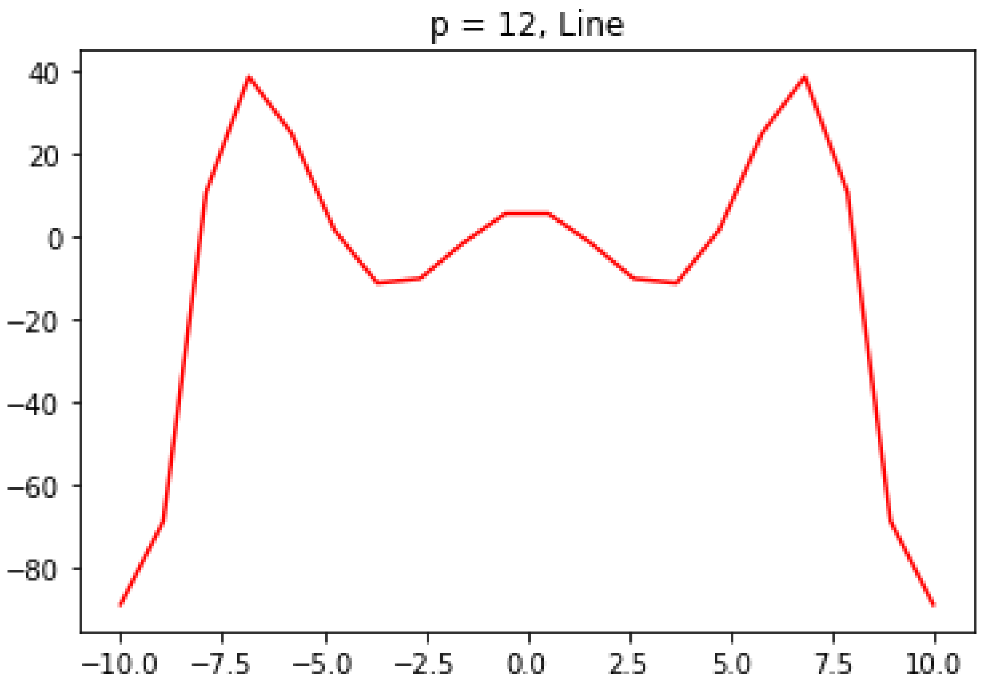

p = 9, Line + Noise
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

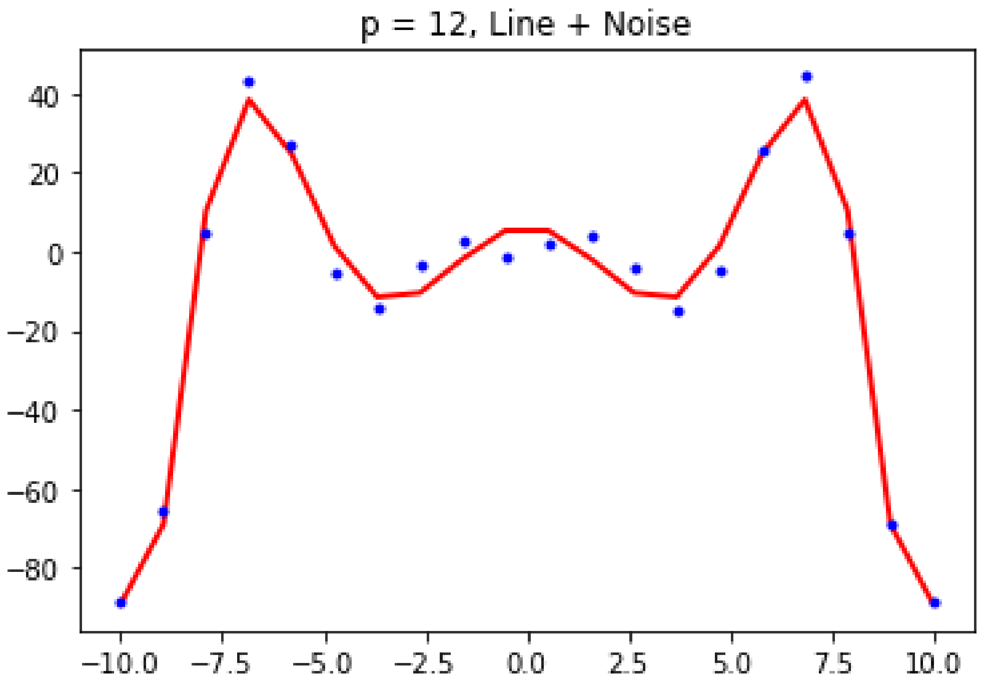

p = 10, Line
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

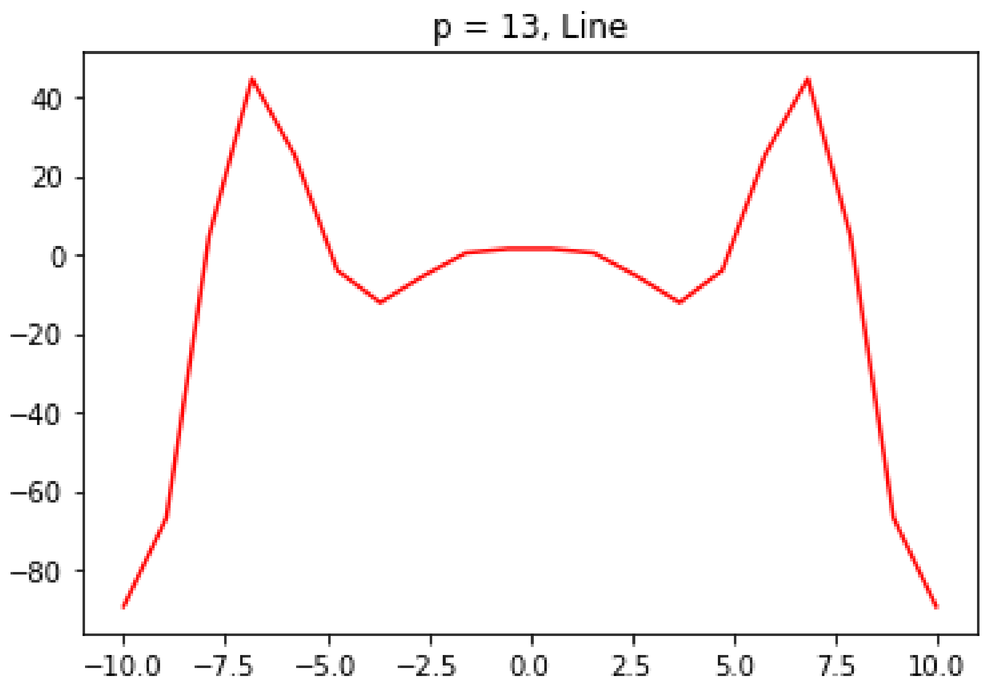

p = 10, Line + Noise
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

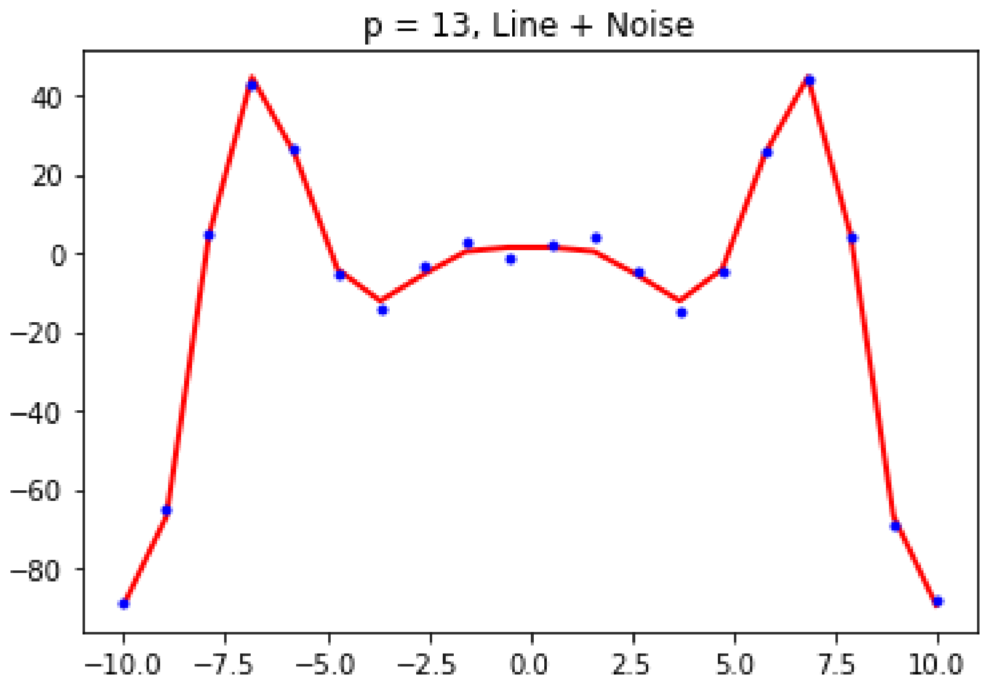

p = 11, Line
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

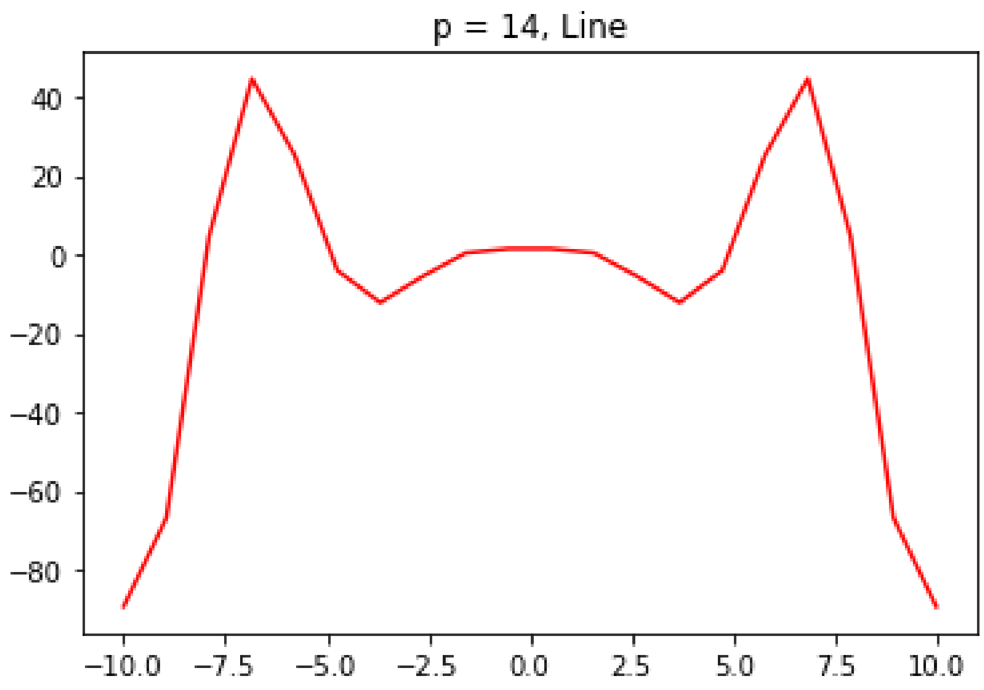

p = 11, Line + Noise
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

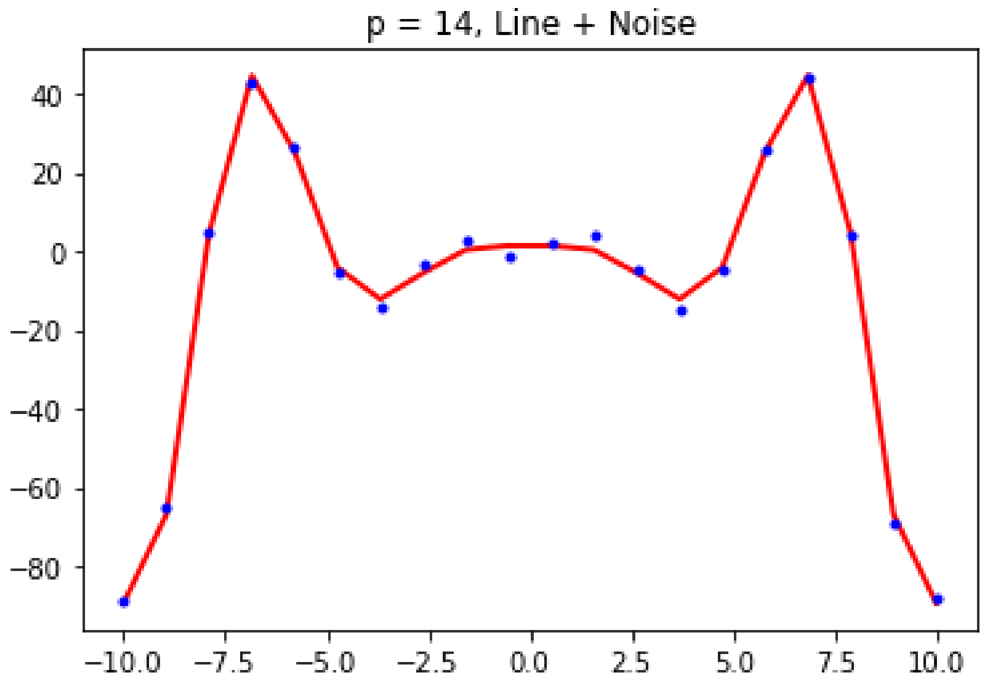

p = 12, Line
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

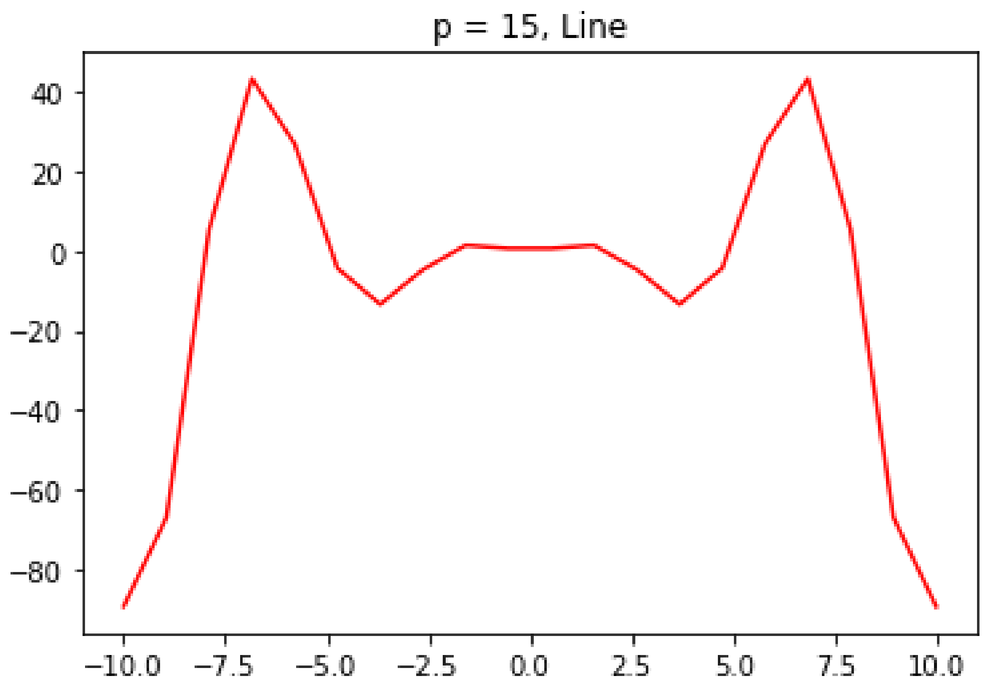

p = 12, Line + Noise
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

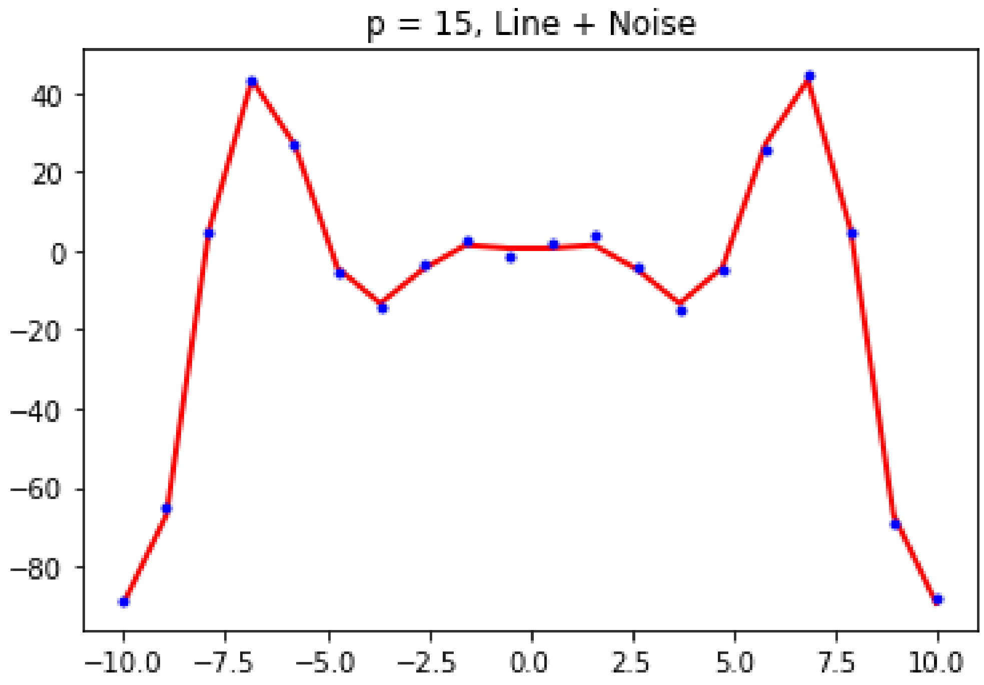

p = 13, Line
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

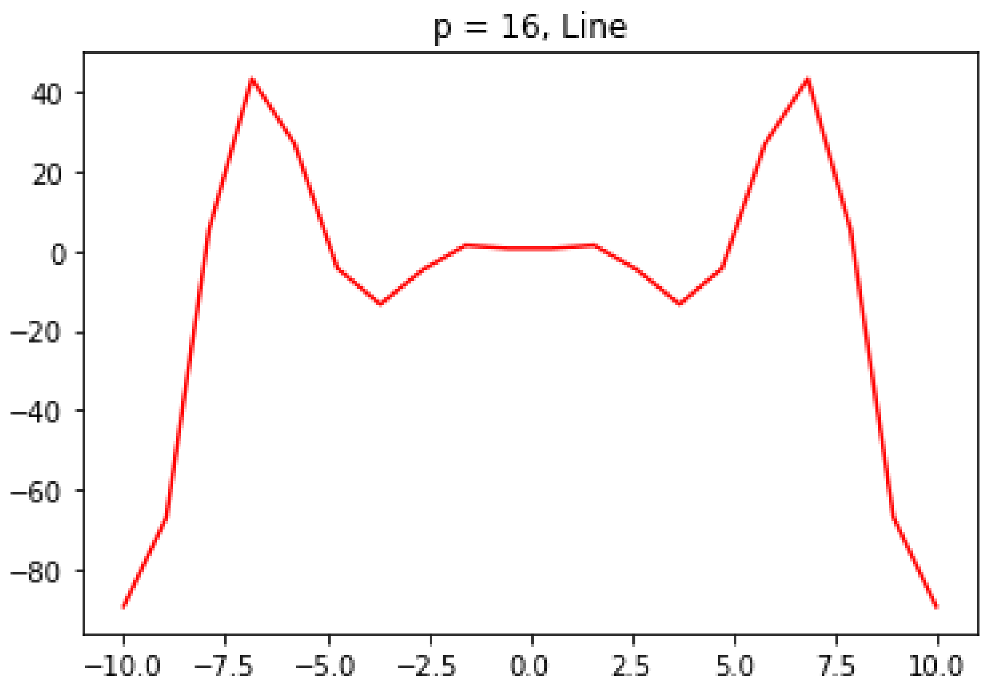

p = 13, Line + Noise
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

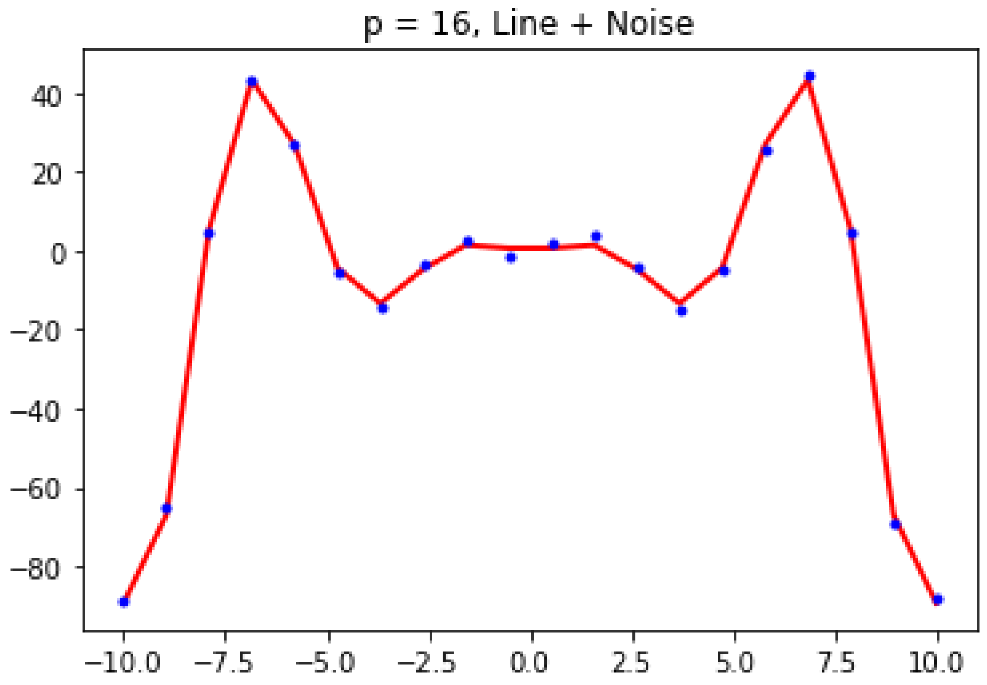

p = 14, Line
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

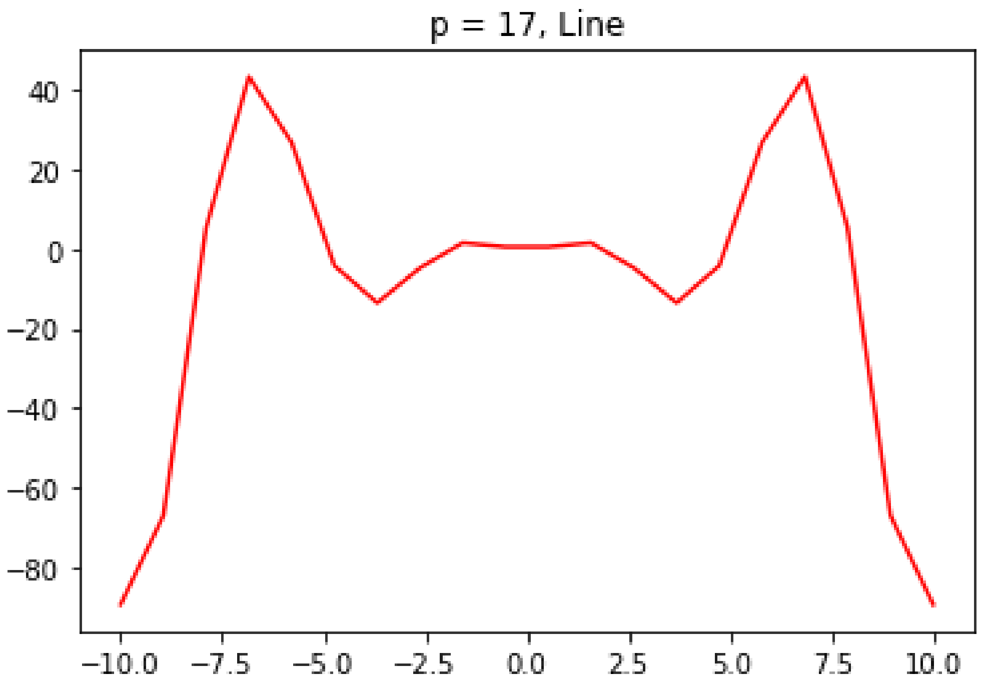

p = 14, Line + Noise
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

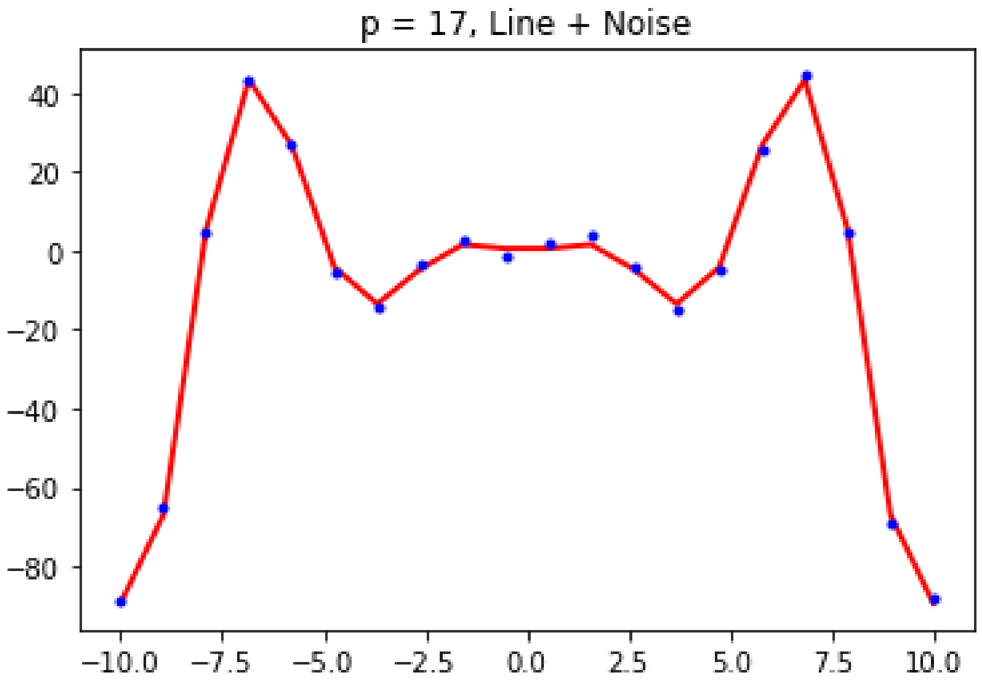

p = 15, Line
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

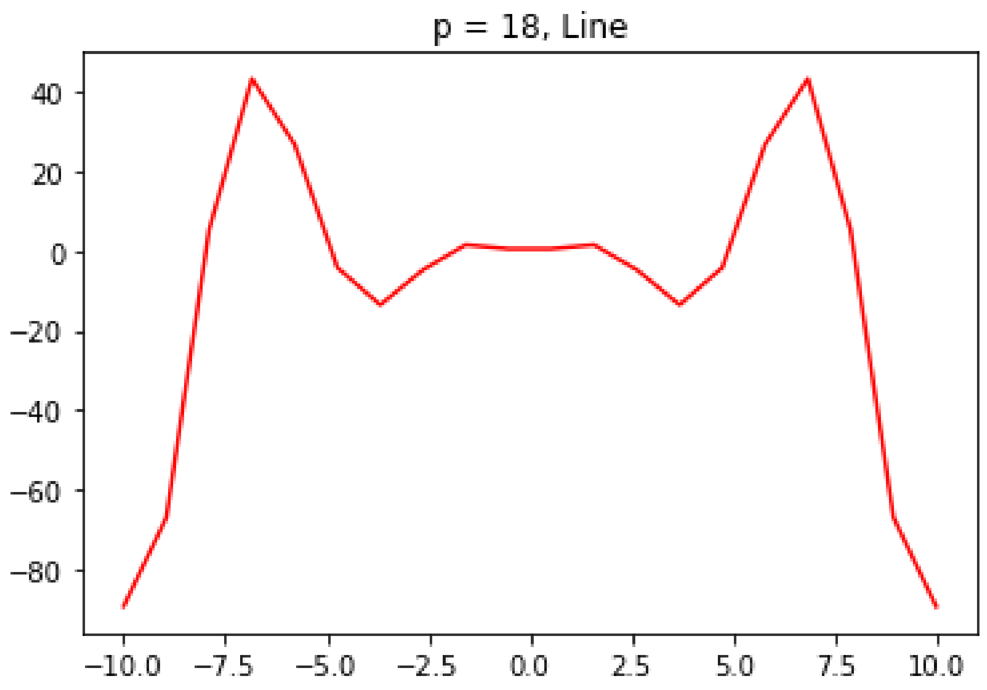

p = 15, Line + Noise
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

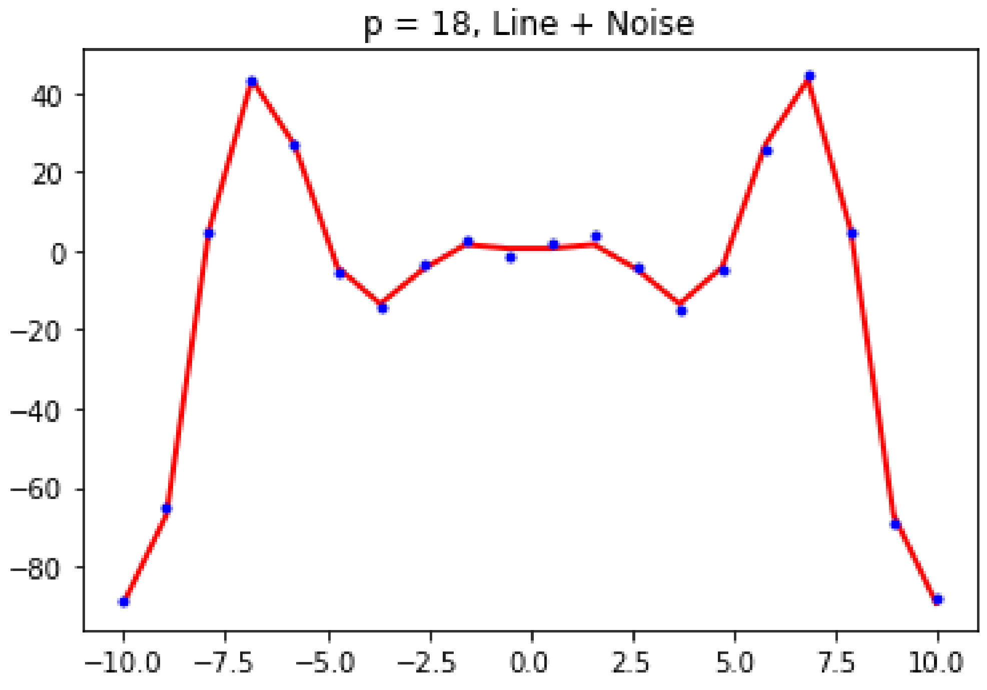

p = 16, Line
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

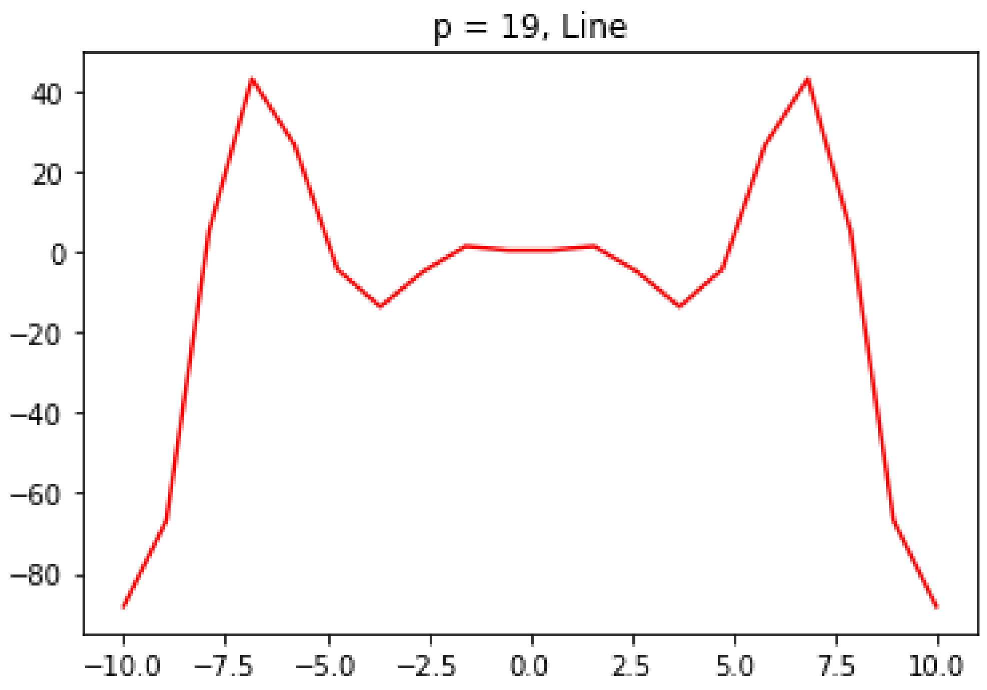

p = 16, Line + Noise
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

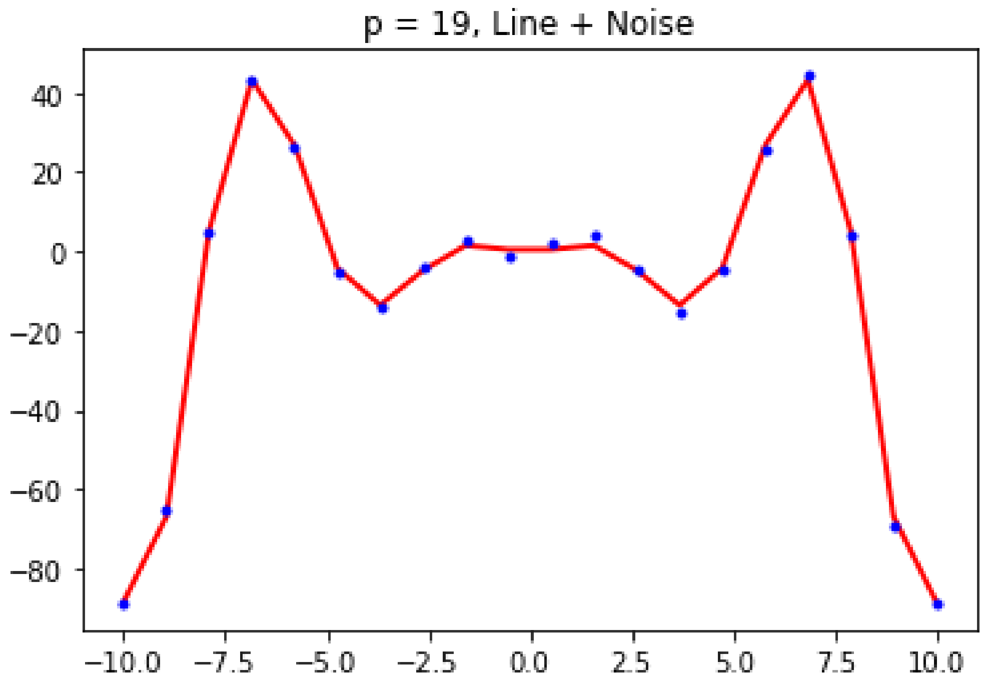

p = 17, Line
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

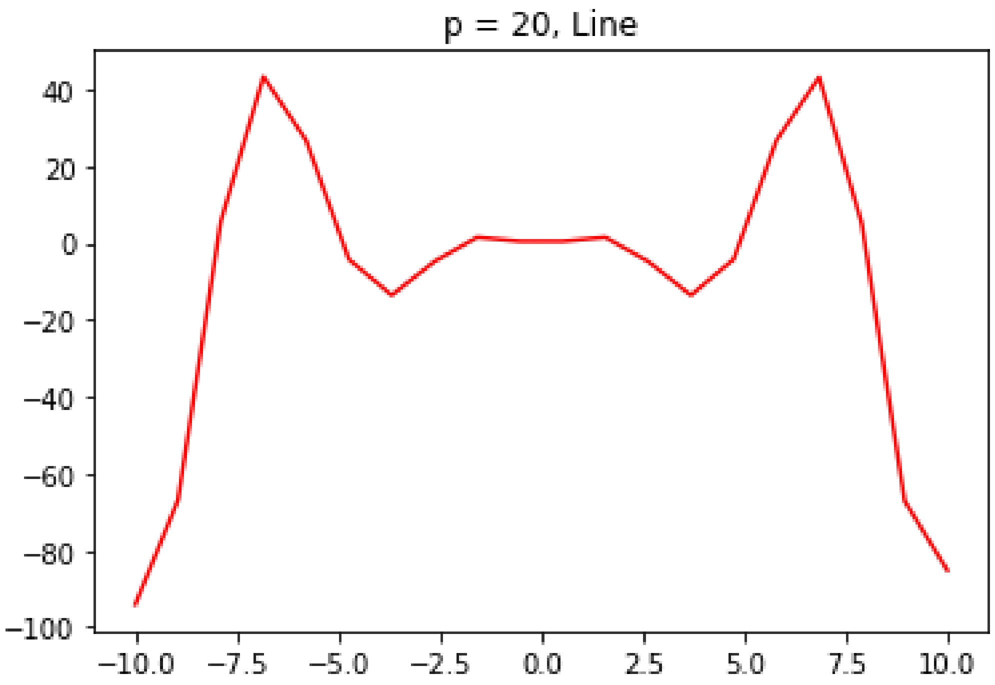

p = 17, Line + Noise
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

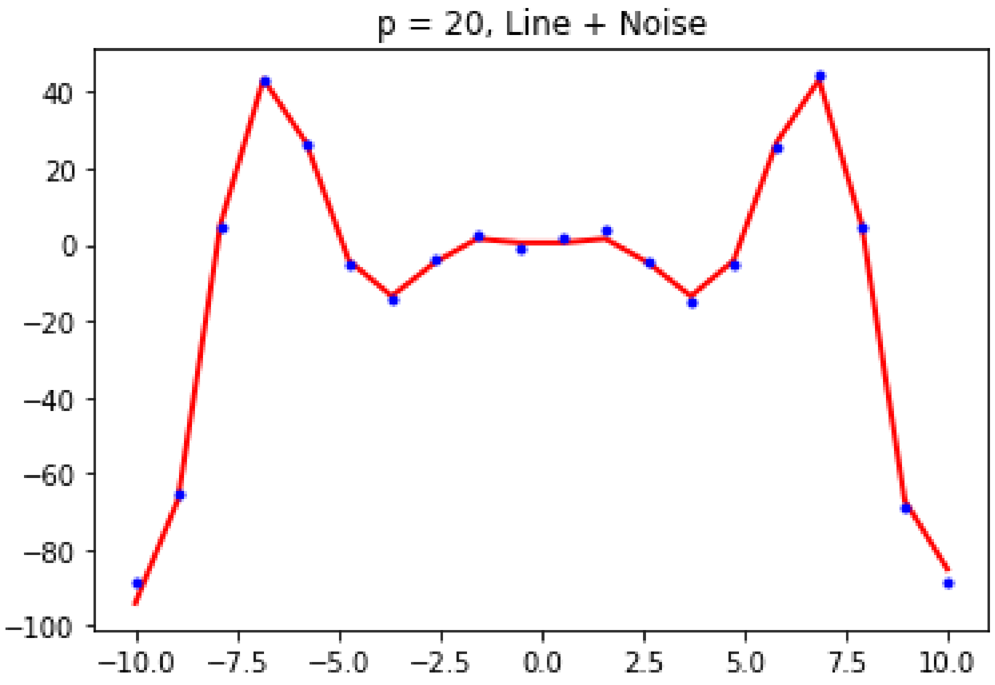

p = 18, Line
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


p = 18, Line + Noise
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


p = 19, Line
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

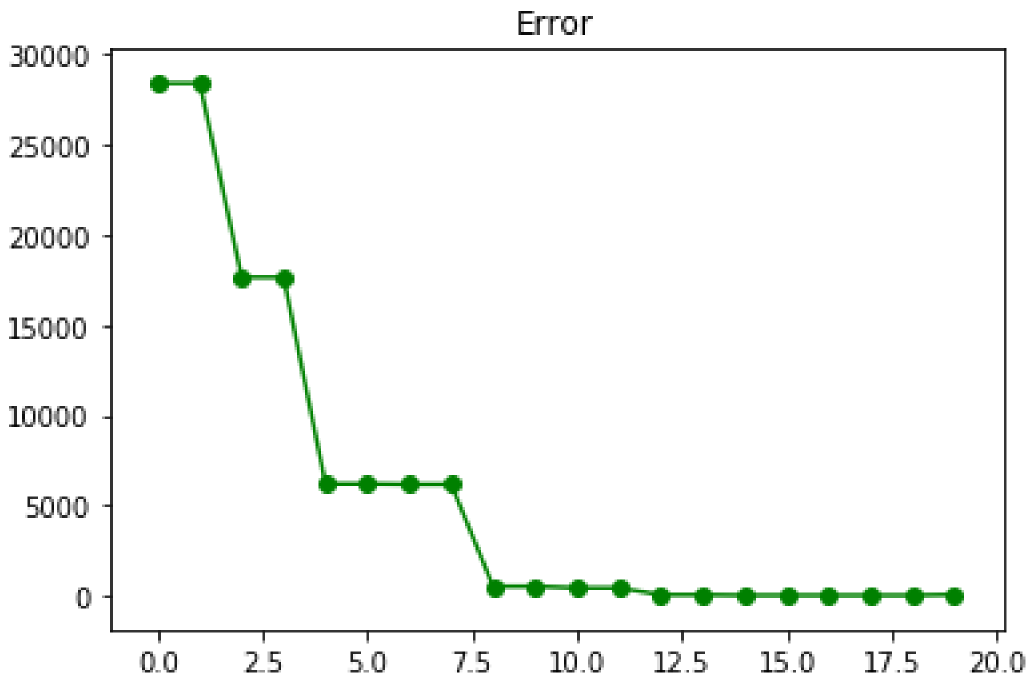

p = 19, Line + Noise
40
20
0
−20
−40
−60
−80
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


p = 20, Line
40
20
0
−20
−40
−60
−80
−100
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


p = 20, Line + Noise
40
20
0
−20
−40
−60
−80
−100
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

8 Draw error graph

```
In [8]: error_x = range(0,num)
        plt.title("Error")
        plt.plot(error_x, error, 'g')
        plt.scatter(error_x, error, color='g')
        plt.show()
```


Error