Assignment 07

November 15, 2018

1 Syed Farhan Alam Zaidi

2 2018210031

3 Assignment 07

3.1 Non-Linear Least Square Fitting

Github Link: https://github.com/farhan-93/assignment07.git

Import required libraries

```
In [80]: import numpy as np
         import matplotlib.pyplot as plt
```

Below function will generate the data.

```
In [81]: def make_data():


             num = 1001
             std = 5

             def fun (x):
                 #f = np.sin (x) * (1 / (1 + np.exp (-x)))
                 f = np.abs (x) * np.sin (x)
                 return f

             n = np.random.rand (num)
             nn = n - np.mean (n)
             x = np.linspace (-10,10, num)
             y1 = fun (x)
             y2 = y1 + nn * std

             plt.figure(0)
             plt.title("Clean Data")
             plt.plot (x, y1, 'b.')
             plt.plot(x,y1,'b')
```

```
    plt.figure(1)
    plt.title("Noisy Data")
    plt.plot (x, y2, 'k.')
    plt.plot(x,y2,'k')

    plt.figure(3)
    plt.title("Clean and Noisy Data")
    plt.plot (x, y2, 'k.' )
    plt.plot(x,y2,'k',label="Noisy")
    plt.plot (x, y1, 'b.')
    plt.plot(x,y1, 'b',label="Clean")
    plt.legend()
    plt.show ()

    return x, y2, y1
```

Below function will calculate the best nonlinear curve line for abouve random generaetd data. And calculate the approximation function

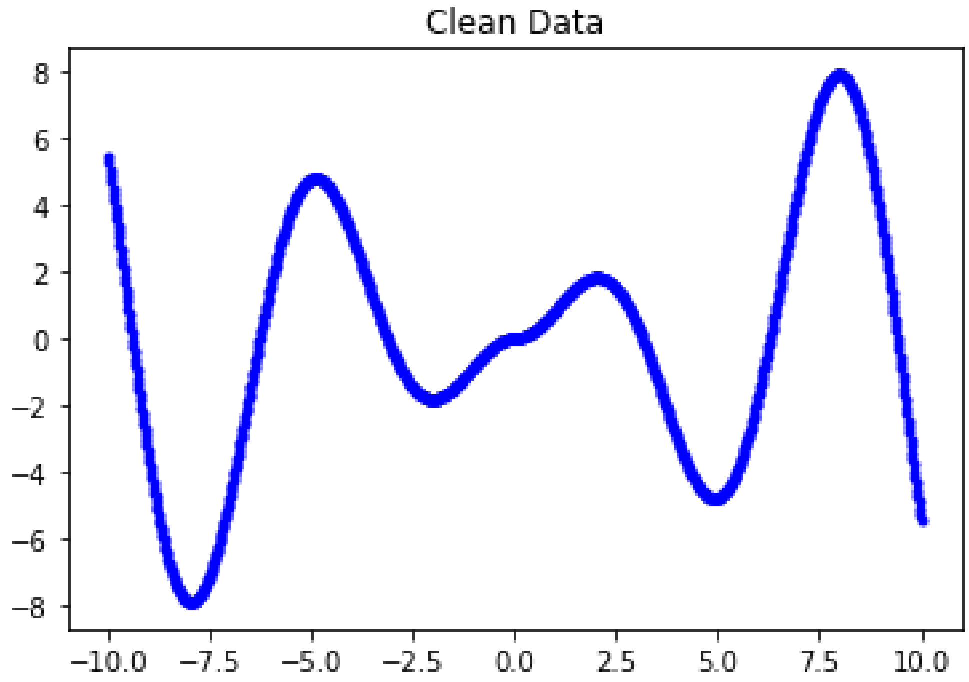
$$f(\hat{x}) = \theta_1 + \theta_2 x + \ldots + \theta_p x^{p-1}$$

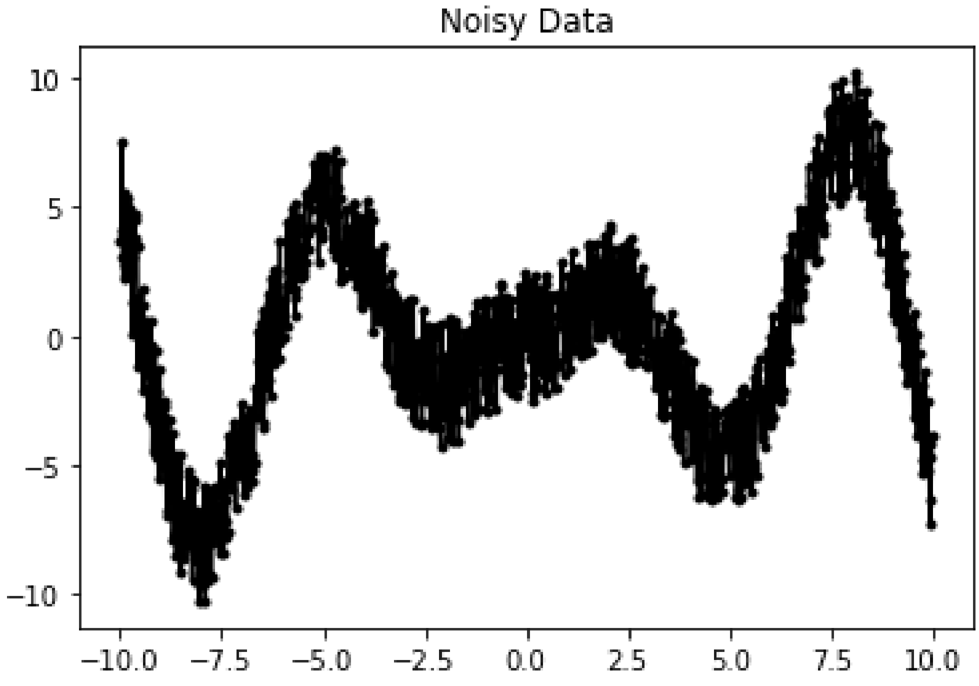
```
In [82]: ############## d=p-1
         def least_squares(x, y,d):

                 n = len(x)
                 ############## Create the matrix A (polynomial matrix)
                 A=np.c_[np.ones(n)]
                 for z in range(1,d):
                     A = np.c_[A,x**z]
                 #print (A.shape)

                 ''''It is the Solution of least square problem. In this expression,
                 its calculate the dot product of the Pseudo_inverse(A) and b'''
                # value of a and b for the linear equation
                 theta=np.empty([0, d])
                 theta=np.linalg.inv(A.T.dot(A)).dot(A.T).dot(y)
                 curve=0.0
                 ######
                 for k in range(0,d):
                     curve= curve +(theta[k]*x**k)

                 return theta,curve
```

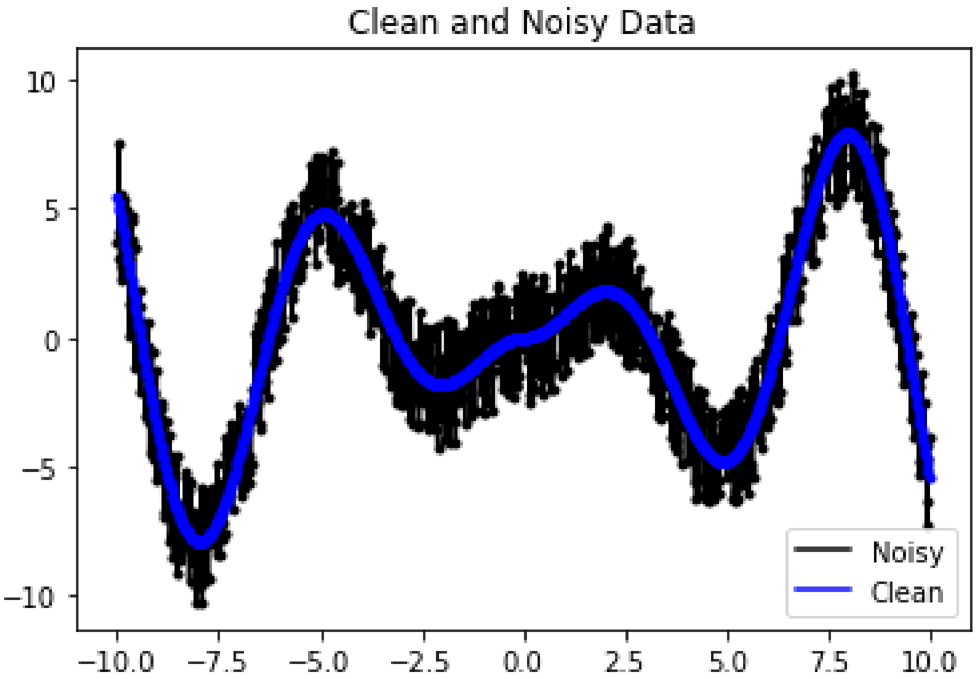
d is the degree of polynomial. We calculate the 15 degree polynomials below. It can be chage and can be any integer.

```
In [85]: # d is the degree of plynomial
```

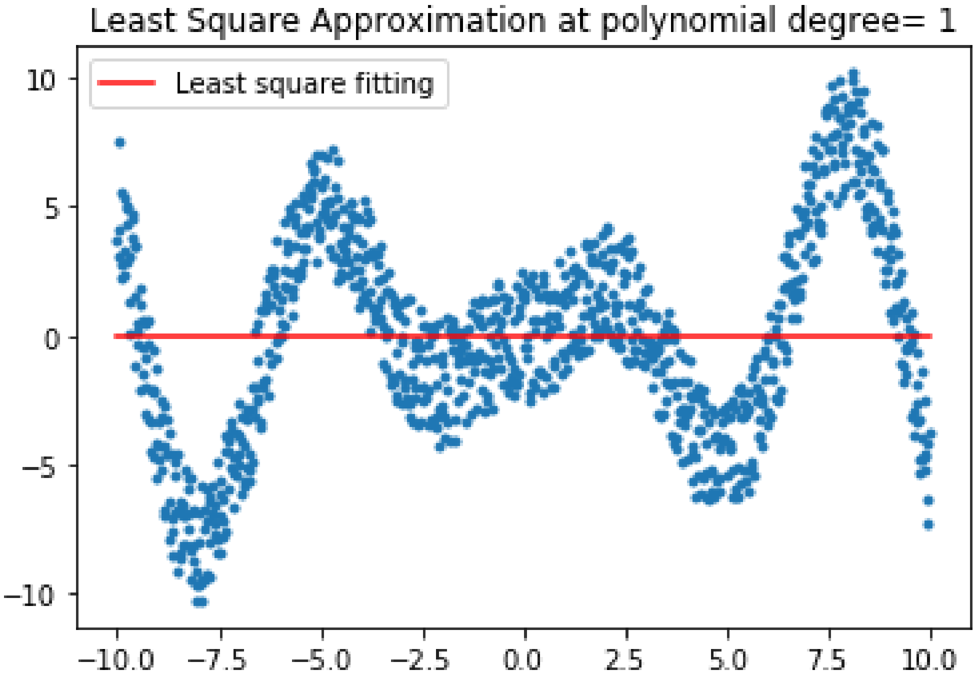
```
d=15

######## Generate data
x, y, y_clean = make_data()
error=[]
print(error)

############ plot the least square approximation
for i in range(1,d+1):
    ###### Calculate the theta values
    theta, curve = least_squares(x, y,i)
    residual =np.linalg.norm(y-curve)
    plt.figure(i)
    plt.title("Least Square Approximation at polynomial degree= "+str(i))
    plt.scatter(x, y, marker=".")
    ##### cakculate the polynomial fit.
    plt.plot(x, curve, 'r', label="Least square fitting")
    plt.legend()
    error=np.append(error,residual)

 ############ plot the Error
plt.figure(100)
plt.title("Graph: Error")
plt.xlabel("Degree of polynomial")
plt.ylabel("Error")
plt.plot(error)


plt.show()
```

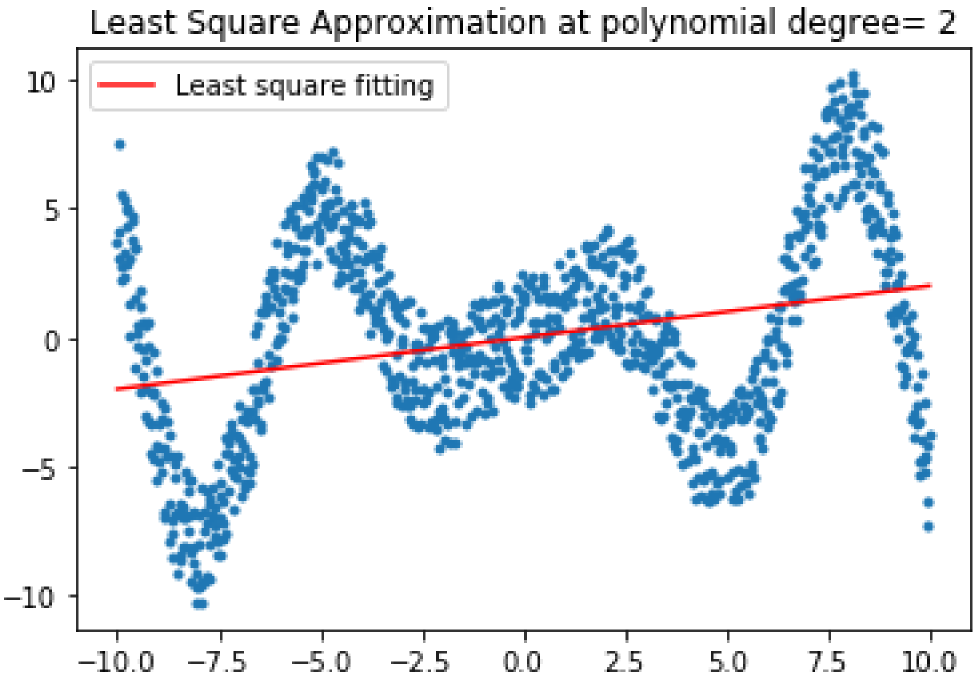

Clean Data
8
6
4
2
0
−2
−4
−6
−8
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

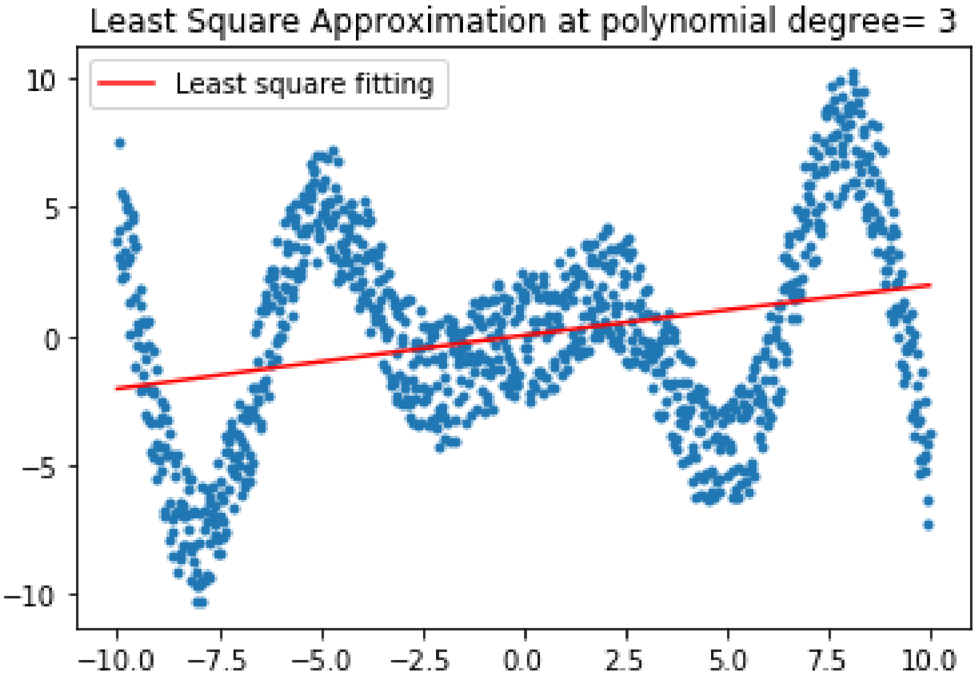

Noisy Data
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

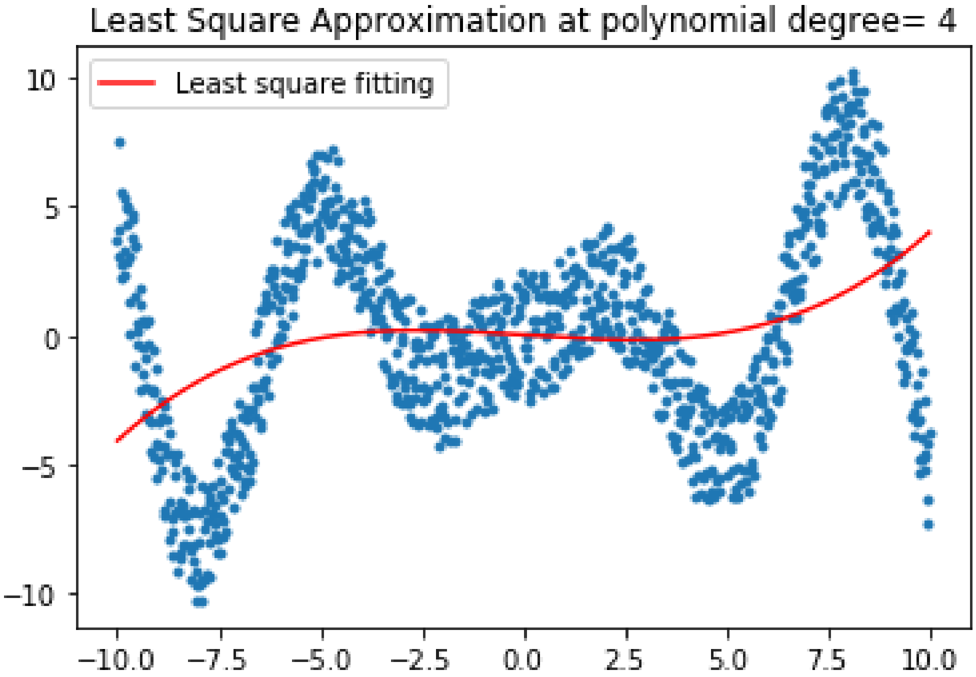

Clean and Noisy Data
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Noisy
Clean

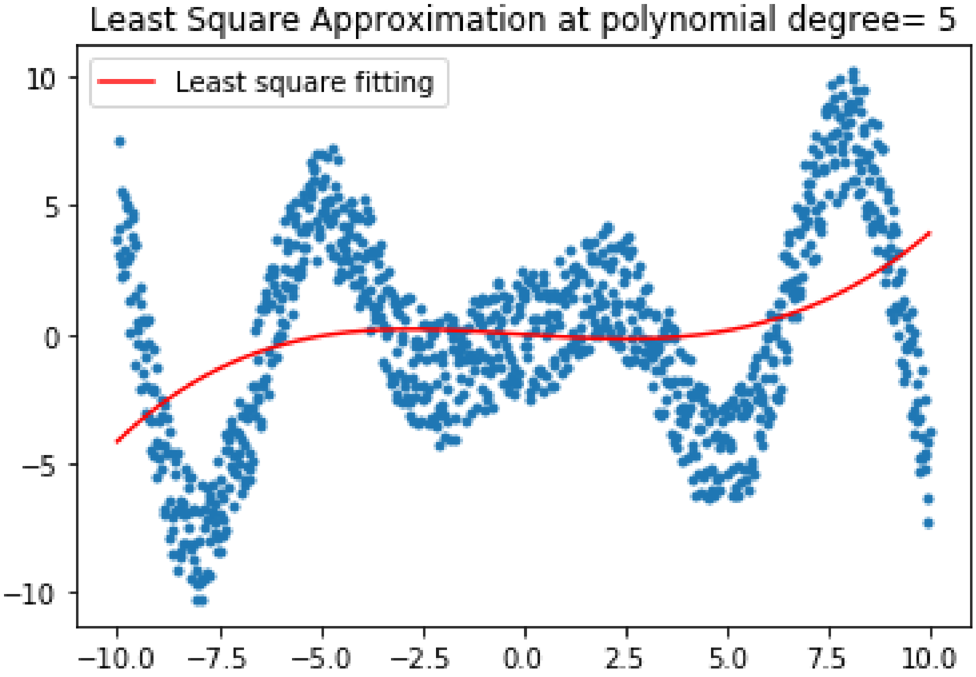
[]

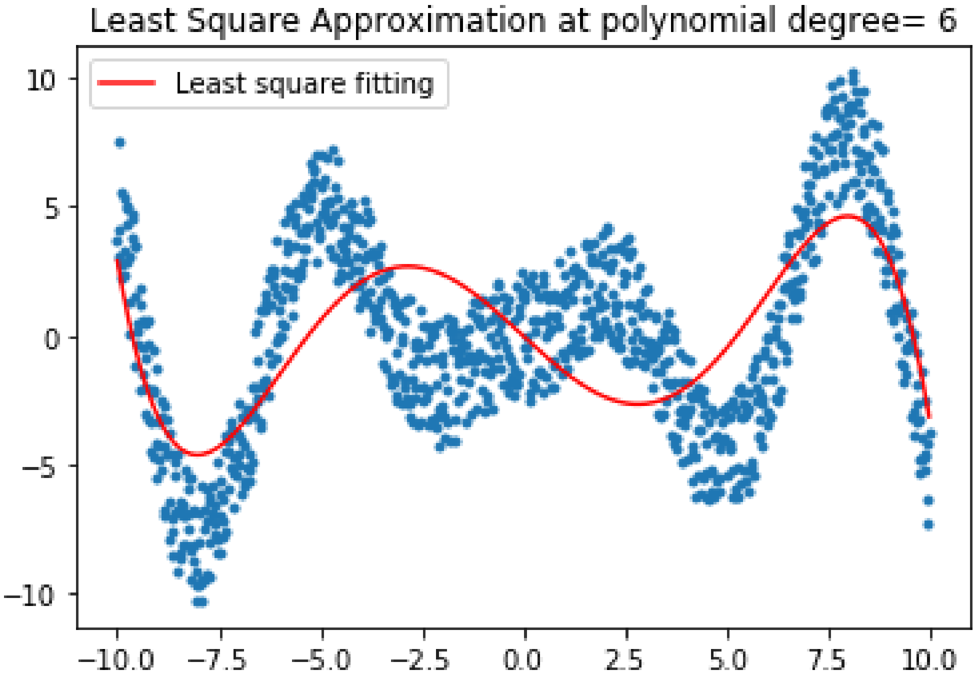

Least Square Approximation at polynomial degree= 1
Least square fitting
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

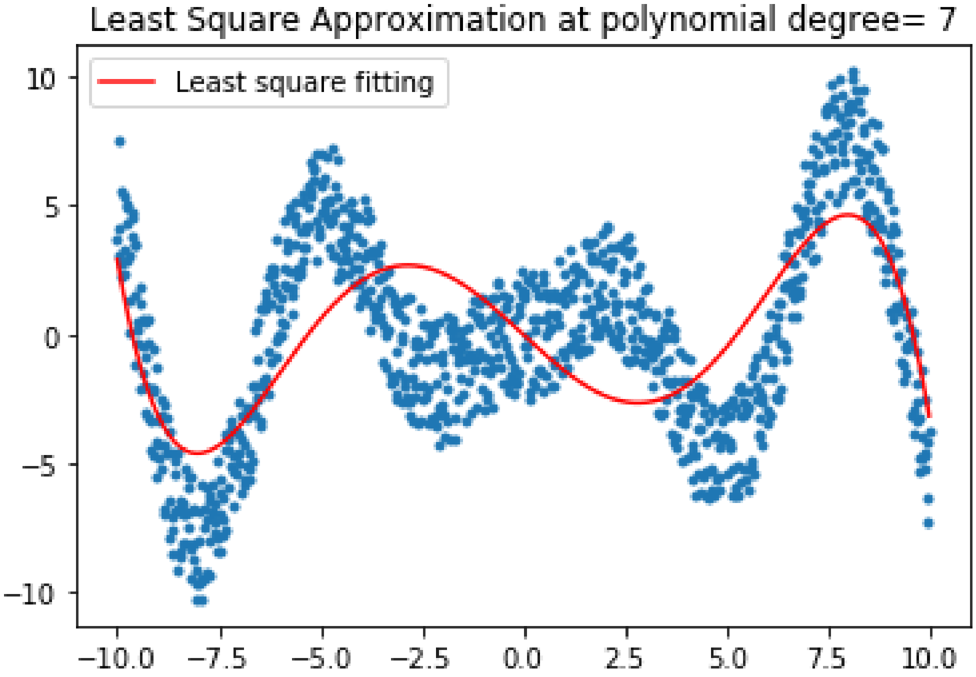

Least Square Approximation at polynomial degree= 2
Least square fitting
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

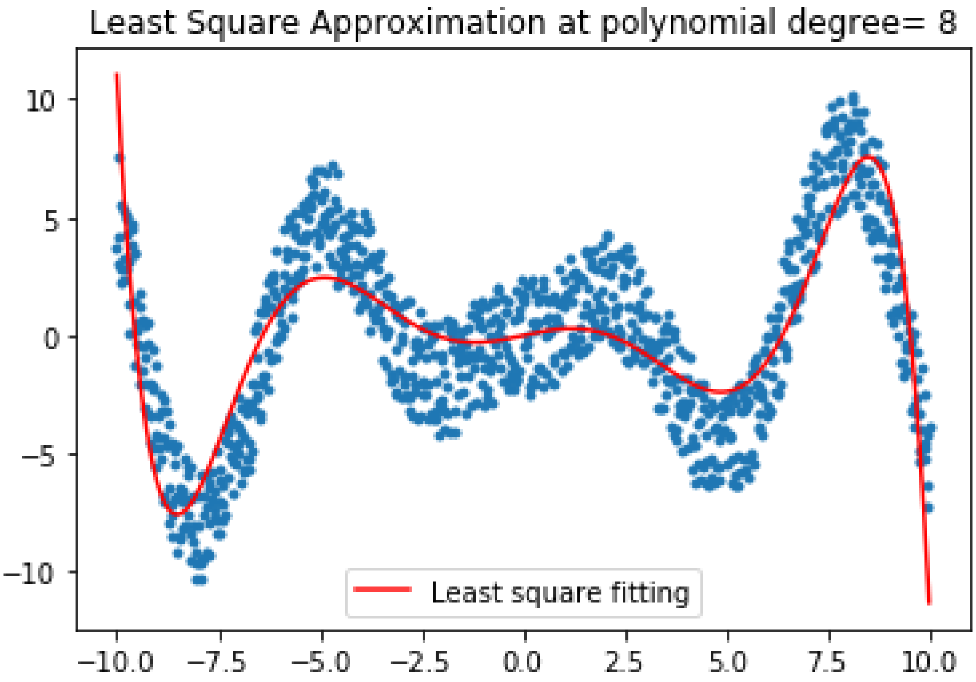

Least Square Approximation at polynomial degree= 3
Least square fitting
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

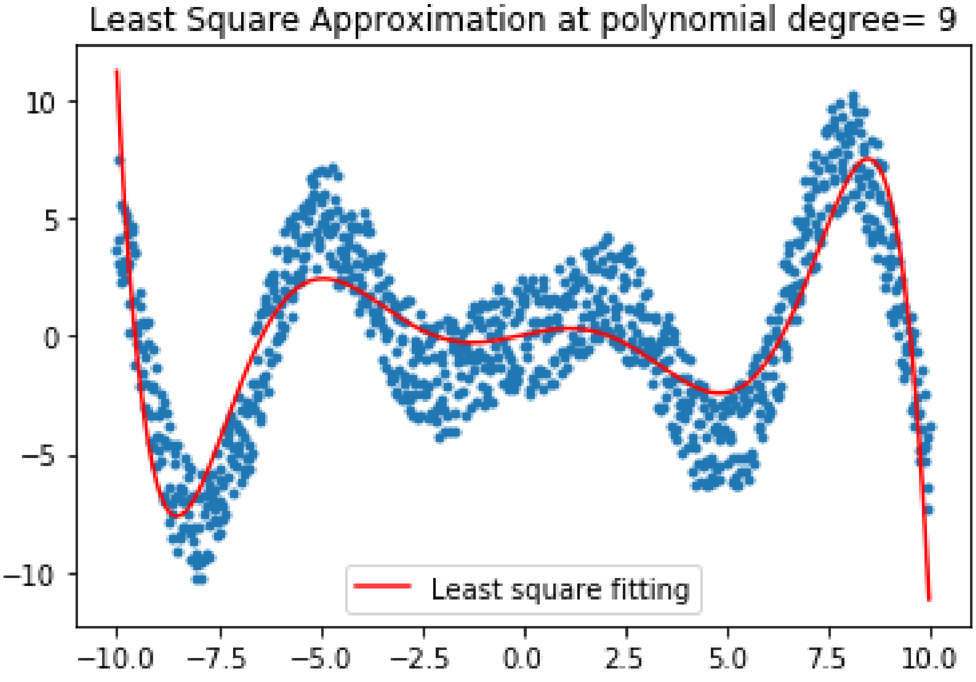

Least Square Approximation at polynomial degree= 4
Least square fitting
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

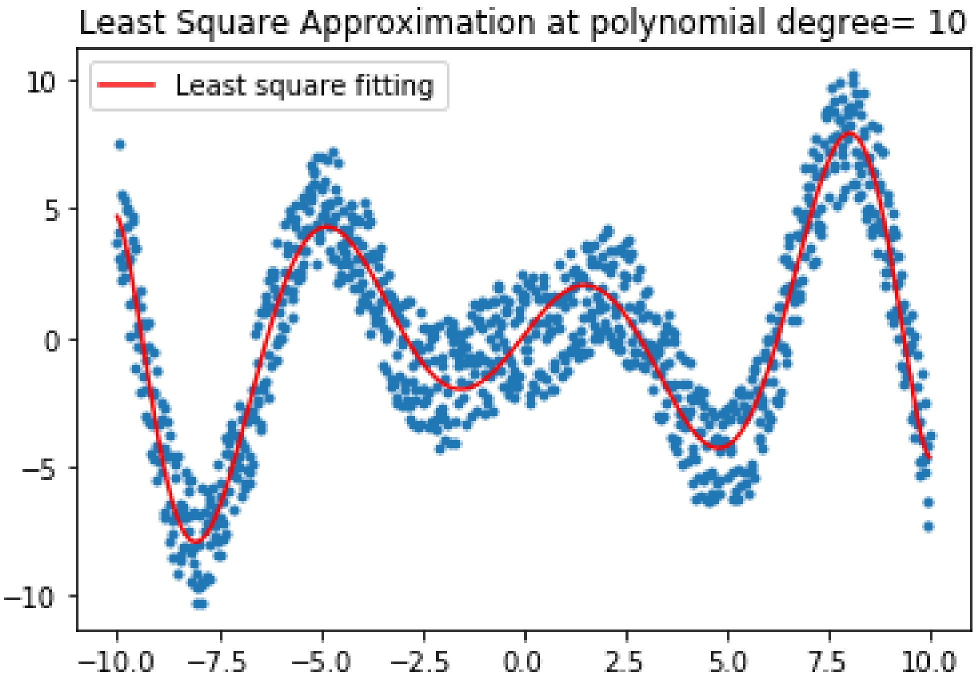

Least Square Approximation at polynomial degree= 5
Least square fitting
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

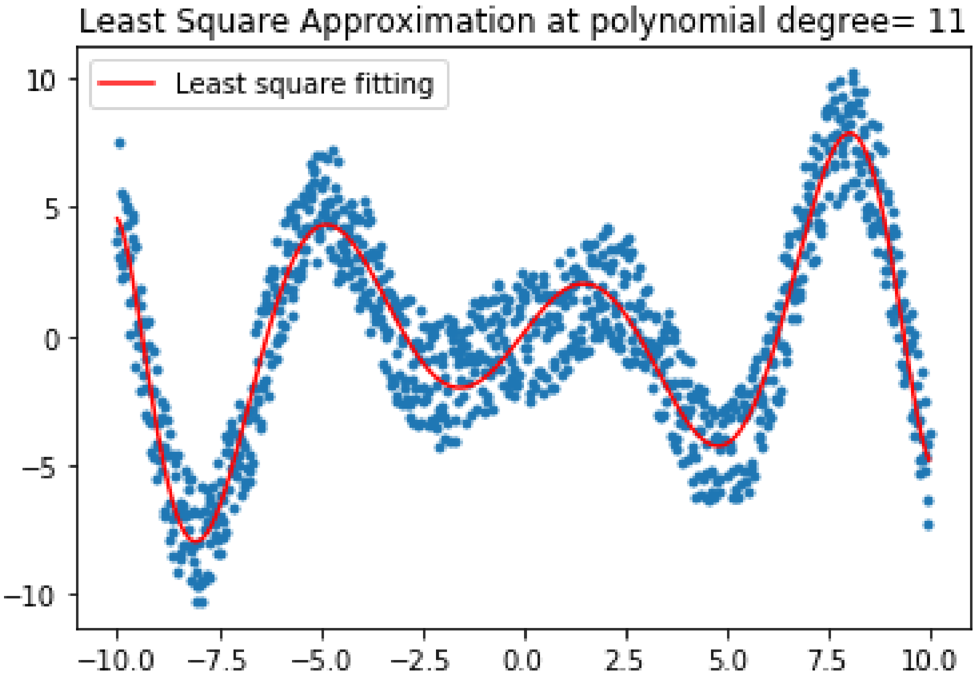

Least Square Approximation at polynomial degree= 6
Least square fitting
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

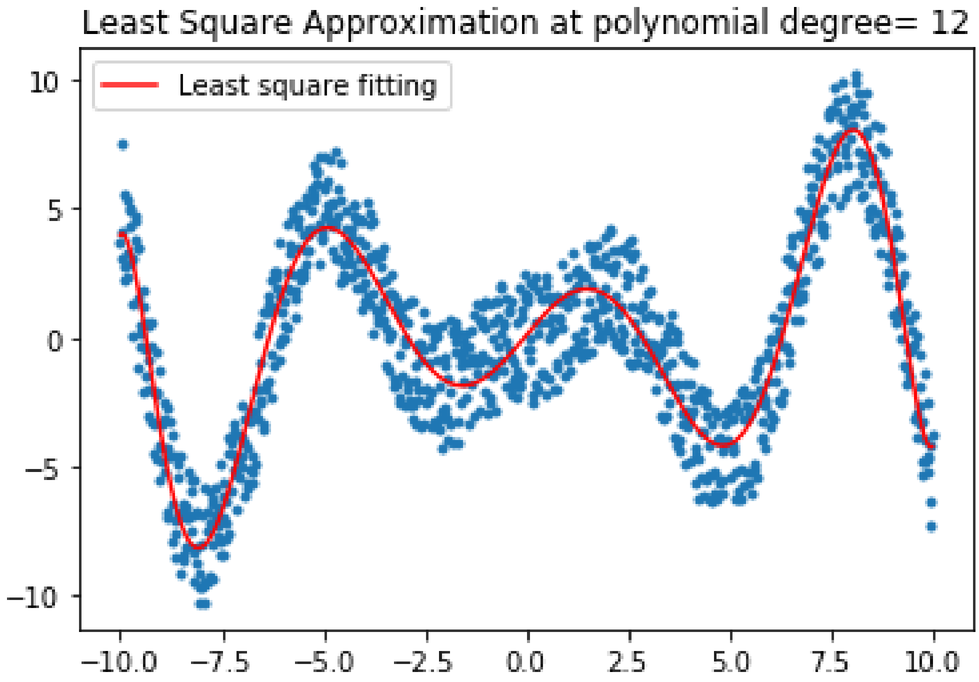

Least Square Approximation at polynomial degree= 7
Least square fitting
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

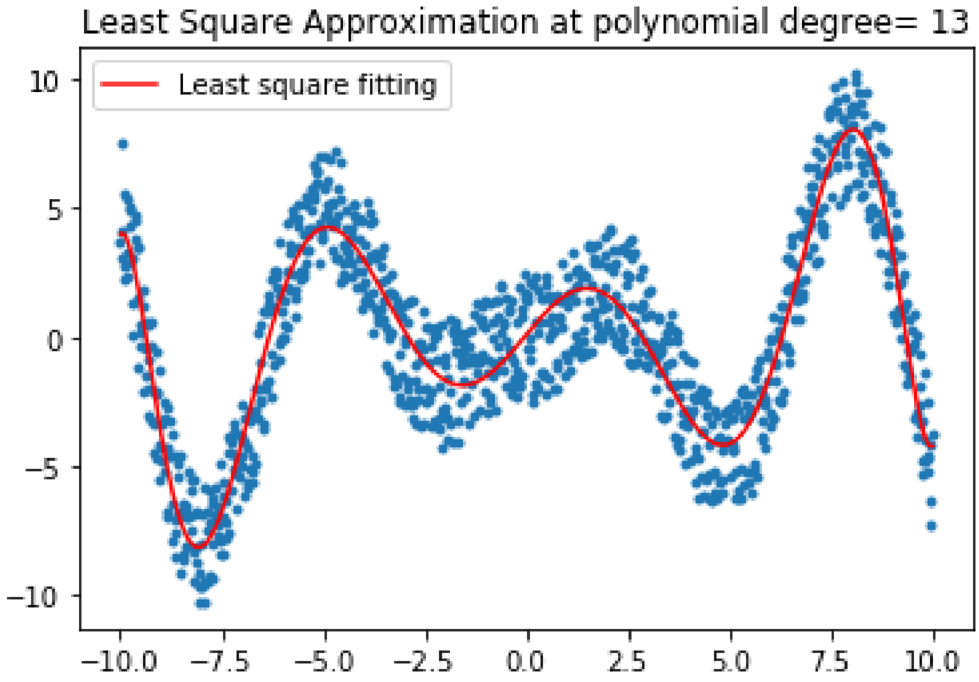

Least Square Approximation at polynomial degree= 8
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Least square fitting

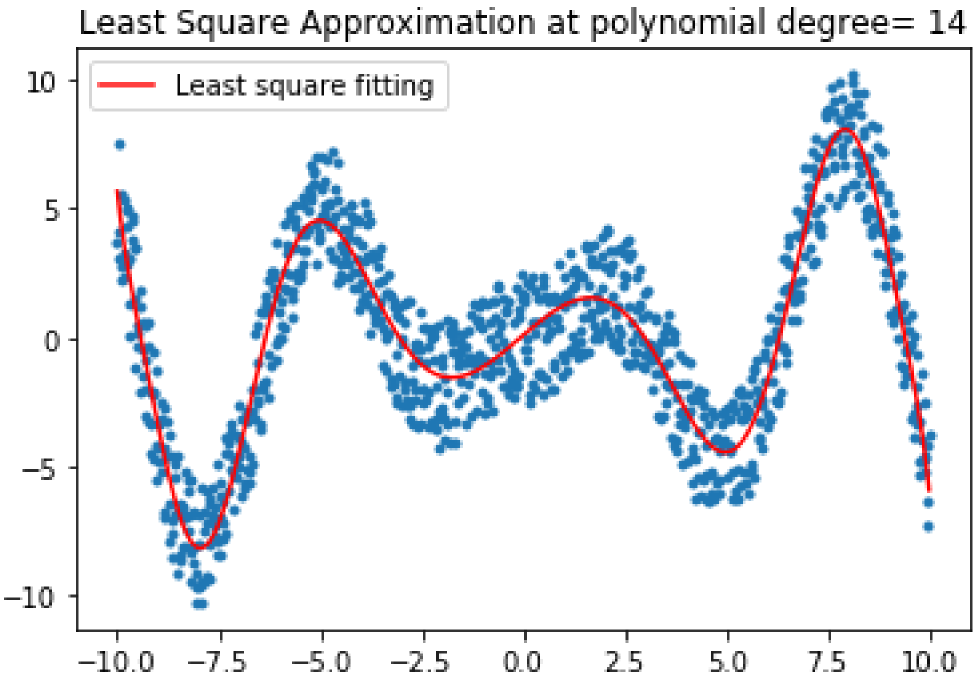

Least Square Approximation at polynomial degree= 9
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Least square fitting

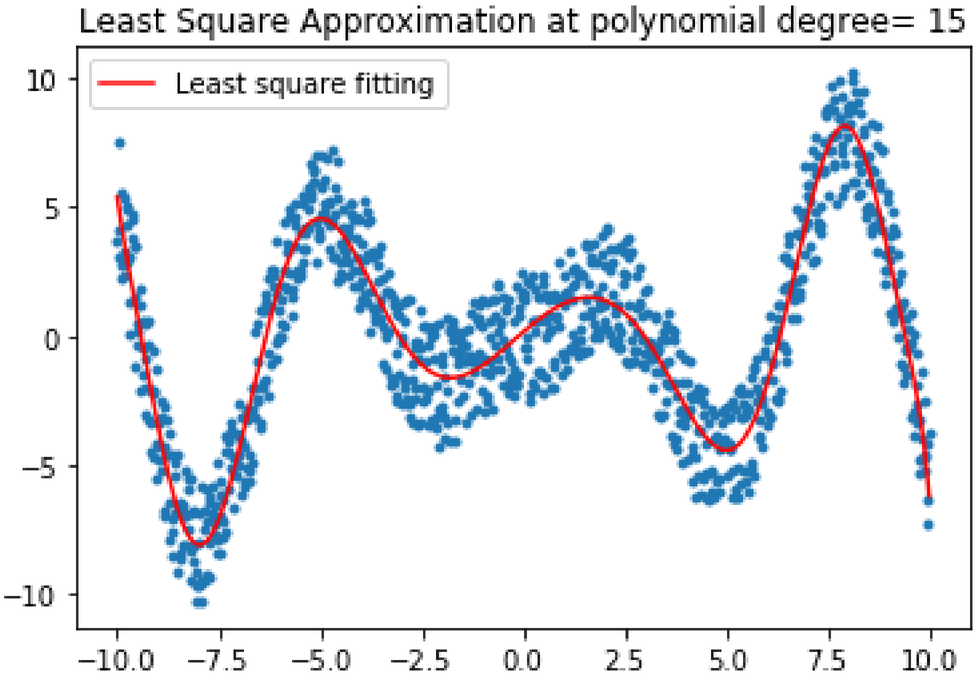

Least Square Approximation at polynomial degree= 10
Least square fitting
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Least Square Approximation at polynomial degree= 11
Least square fitting
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Least Square Approximation at polynomial degree= 12
Least square fitting
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Least Square Approximation at polynomial degree= 13
Least square fitting
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Least Square Approximation at polynomial degree= 14
Least square fitting
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Least Square Approximation at polynomial degree= 15
Least square fitting
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

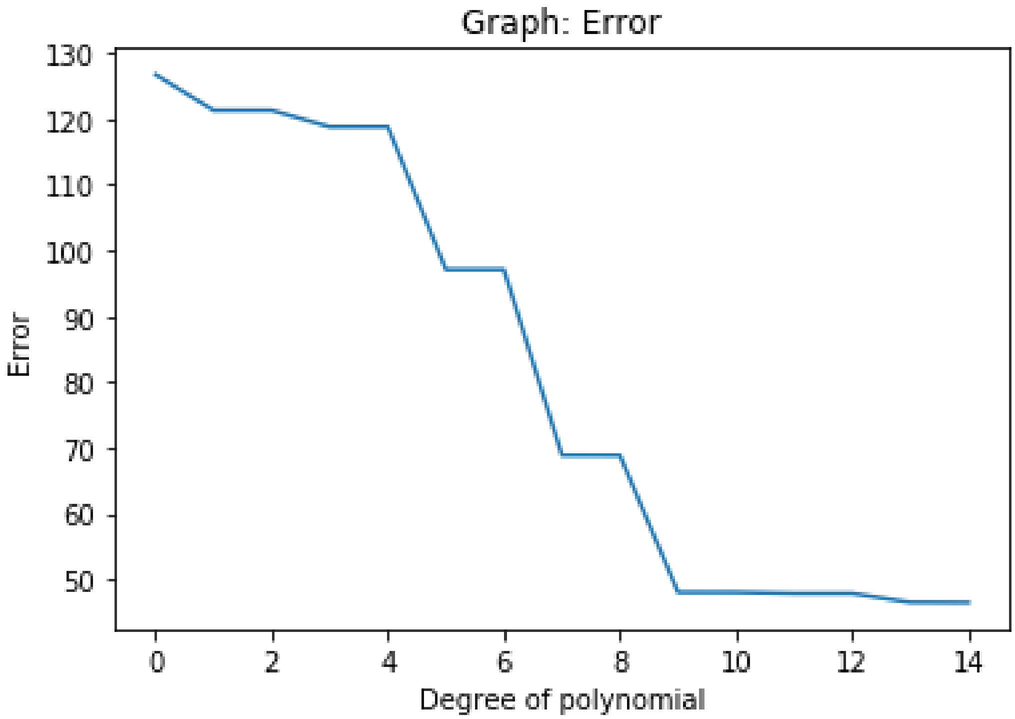


Figure 4 shows the best non linear fit line for the given data. And last picture shows the origional line without noise and best fit line in a same plot.

```
In [86]: print(error)
```

```
[126.7519552  121.33947865 121.33801059 118.83005683 118.82700223
  97.1200496   97.11999725  68.86863829  68.85642857  48.01081047
  47.99720485  47.83122174  47.83120482  46.47843243  46.43994771]
```