

```

1 %matplotlib inline
2 import matplotlib.pyplot as plt
3 import numpy as np
4 from scipy import linalg
5
6 def actual_distribution():
7     x = np.arange(0, 9, 0.5);
8     y = 0.4345*np.power(x,3) - 5.607*np.power(x,2) + 16.78*x - 10.61
9     return x,y
10
11 def add_noise(y_curve):
12     mu = 0
13     sigma=4.0
14     noise = np.random.normal(mu, sigma, len(y_curve))
15     y_noise = y_curve + noise
16     return y_noise
17
18 def numpy_poly_fitting(x,y, M):
19     z = np.polyfit(x, y, M)
20     f = np.polyld(z)
21     return f
22
23 def my_MLE_fitting(x,y,M):
24     N = len(x)
25     mx = np.zeros([N, M+1])
26     vy = np.zeros([N,1])
27
28     for n in range(0, N):
29         for m in range(0,M+1):
30             mx[n][m] = np.power(x[n],m) #polynomial function
31             vy[n][0] = y[n]
32
33     mxx = np.dot(mx.T, mx)
34     imxx = linalg.inv(mxx)
35     #tmp =np.dot(imxx,mx.T)
36     tmp = linalg.pinv(mx) # direct pseudo inverse calculation
37     w = np.dot(tmp, vy)
38     return w
39

```

```

41 def my_MLE_plot(x,w):
42     M = len(w)-1
43     N = len(x)
44
45     mx = np.zeros([N,M+1])
46
47     for n in range(0, N):
48         for m in range(0,M+1):
49             mx[n][m] = np.power(x[n],m)
50
51     y = np.dot(mx,w)
52     return x,y
53
54
55 #=====
56
57
58 print('start...')
59 # generate true data
60 x_true, y_true = actual_distribution()
61
62 # fit on the actual data
63 f = numpy_poly_fitting(x_true, y_true, M=3)
64 print('f:', f)
65 x_curve = np.linspace(x_true[0], x_true[-1], 50)
66 y_curve = f(x_curve)
67
68 # add noise on the true data
69 y_noise = add_noise(y_curve)
70 x_noise = x_curve
71
72 # estimate the curve from the noisy data
73 w = my_MLE_fitting(x_noise, y_noise, M=60)
74 [x_est, y_est] = my_MLE_plot(x_curve, w)
75
76 # show the plot
77 plt.plot(x_true, y_true, 'ro')
78 plt.plot(x_curve, y_curve, 'red')
79 plt.plot(x_noise, y_noise, 'go')
80 plt.plot(x_est, y_est)
81 plt.show()
82

```