Assignment08

May 21, 2019

20132915 Nam, Geun Woo

[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

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

```
In [2]: number_of_points = 40
```

1 Fitting on first-order polynomial

```
In [3]: def order_1_polynomial(t, y, y_noise):

        #    plt.figure(figsize=(12,8))
        #    plt.plot(t, y)
        #    plt.show()



        #    plt.figure(figsize=(12,8))
        #    plt.plot(t, y_noise)
        #    plt.show()

        #    plt.figure(figsize=(12,8))
        #    plt.plot(t, y_noise, label='noise', color='y')
        #    plt.plot(t, y, ls='dashed', lw=3, color='b', label='original')
        #    plt.grid()
```

```
#     plt.legend()
#     plt.show()

# Ax = b

    A = np.hstack((t.reshape(40,1), np.ones((40,1))))
    b = y_noise.copy()
    x = np.matmul(np.linalg.pinv(A), b)

    y_hat = x[0]*t + x[1]

    plt.figure(figsize=(12,8))
    plt.plot(t, y_noise, label='noise', color='y')
    plt.plot(t, y, ls='dashed', lw=3, color='b', label='original')
    plt.plot(t, y_hat, ls='dotted', lw=3, color='r', label='fitted')
#     fp1 = np.polyfit(t, y_noise, 1)
#     f1 = np.poly1d(fp1)
#     plt.plot(t, f1(t), lw=2, color='g', label='polyfit')
    plt.grid()
    plt.legend()
    plt.show()

    return y_hat
```

2 Fitting on second-order polynomial

```
In [4]: def order_2_polynomial(t, y, y_noise):

#     plt.figure(figsize=(12,8))
#     plt.plot(t, y)
#     plt.show()



#     plt.figure(figsize=(12,8))
#     plt.plot(t, y_noise)
#     plt.show()

#     plt.figure(figsize=(12,8))
#     plt.plot(t, y_noise, label='noise', color='y')
#     plt.plot(t, y, ls='dashed', lw=3, color='b', label='original')
#     plt.grid()
#     plt.legend()
#     plt.show()

# Ax = b
```

```
    A = np.hstack((t.reshape(40,1)**2, t.reshape(40,1), np.ones((40,1))))
    b = y_noise.copy()
    x = np.matmul(np.linalg.pinv(A), b)

    y_hat = x[0]*t**2 + x[1]*t + x[2]

    plt.figure(figsize=(12,8))
    plt.plot(t, y_noise, label='noise', color='y')
    plt.plot(t, y, ls='dashed', lw=3, color='b', label='original')
    plt.plot(t, y_hat, ls='dotted', lw=3, color='r', label='fitted')
#     fp1 = np.polyfit(t, y_noise, 2)
#     f1 = np.poly1d(fp1)
#     plt.plot(t, f1(t), lw=2, color='g', label='polyfit')
    plt.grid()
    plt.legend()
    plt.show()

    return y_hat
```

3 Fitting on thrid-order polynomial

```
In [5]: def order_3_polynomial(t, y, y_noise):

#     plt.figure(figsize=(12,8))
#     plt.plot(t, y)
#     plt.show()



#     plt.figure(figsize=(12,8))
#     plt.plot(t, y_noise)
#     plt.show()

#     plt.figure(figsize=(12,8))
#     plt.plot(t, y_noise, label='noise', color='y')
#     plt.plot(t, y, ls='dashed', lw=3, color='b', label='original')
#     plt.grid()
#     plt.legend()
#     plt.show()

# Ax = b

    A = np.hstack((t.reshape(40,1)**3, t.reshape(40,1)**2,\
                   t.reshape(40,1), np.ones((40,1))))
    b = y_noise.copy()
    x = np.matmul(np.linalg.pinv(A), b)
```

```
    y_hat = x[0]*t**3 + x[1]*t**2 + x[2]*t + x[3]

    plt.figure(figsize=(12,8))
    plt.plot(t, y_noise, label='noise', color='y')
    plt.plot(t, y, ls='dashed', lw=3, color='b', label='original')
    plt.plot(t, y_hat, ls='dotted', lw=3, color='r', label='fitted')
#    fp1 = np.polyfit(t, y_noise, 3)
#    f1 = np.poly1d(fp1)
#    plt.plot(t, f1(t), lw=2, color='g', label='polyfit')
    plt.grid()
    plt.legend()
    plt.show()

    return y_hat
```

4 Fitting on fourth-order polynomial

```
In [6]: def order_4_polynomial(t, y, y_noise):

#    plt.figure(figsize=(12,8))
#    plt.plot(t, y)
#    plt.show()



#    plt.figure(figsize=(12,8))
#    plt.plot(t, y_noise)
#    plt.show()

#    plt.figure(figsize=(12,8))
#    plt.plot(t, y_noise, label='noise', color='y')
#    plt.plot(t, y, ls='dashed', lw=3, color='b', label='original')
#    plt.grid()
#    plt.legend()
#    plt.show()

# Ax = b

    A = np.hstack((t.reshape(40,1)**4, t.reshape(40,1)**3,\
                   t.reshape(40,1)**2, t.reshape(40,1), np.ones((40,1))))
    b = y_noise.copy()
    x = np.matmul(np.linalg.pinv(A), b)

    y_hat = x[0]*t**4 + x[1]*t**3 + x[2]*t**2 + x[3]*t + x[4]

    plt.figure(figsize=(12,8))
    plt.plot(t, y_noise, label='noise', color='y')
```

```
    plt.plot(t, y, ls='dashed', lw=3, color='b', label='original')
    plt.plot(t, y_hat, ls='dotted', lw=3, color='r', label='fitted')
#    fp1 = np.polyfit(t, y_noise, 4)
#    f1 = np.poly1d(fp1)
#    plt.plot(t, f1(t), lw=2, color='g', label='polyfit')
    plt.grid()
    plt.legend()
    plt.show()

    return y_hat
```

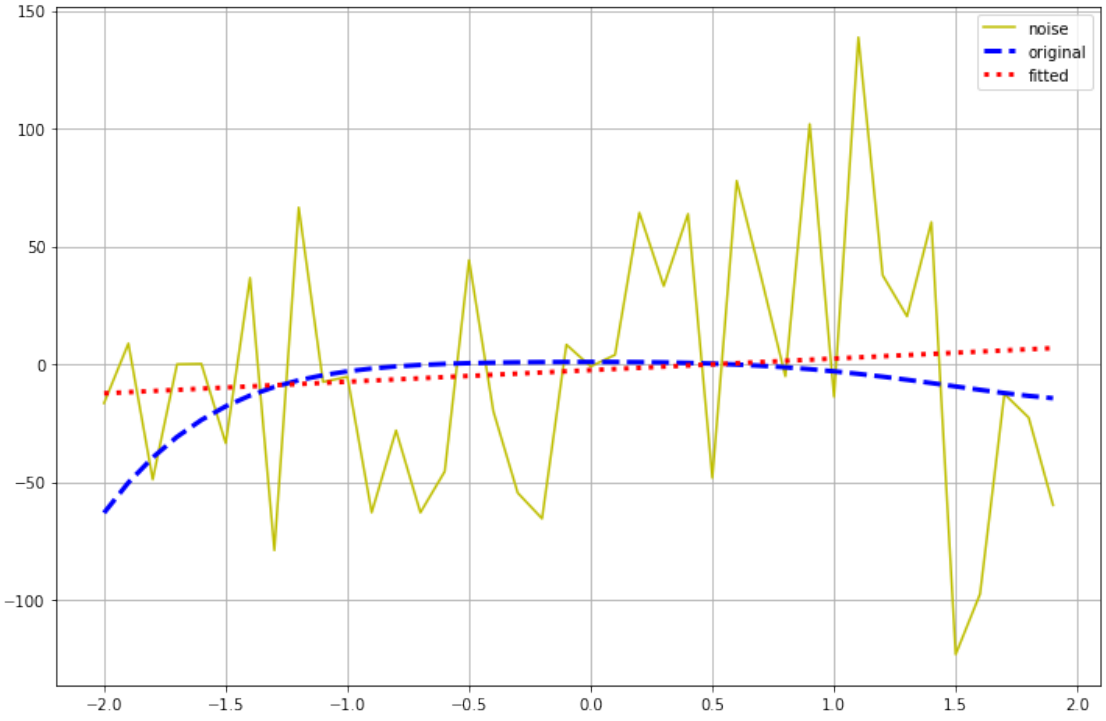
5 Fitting on fifth-order polynomial

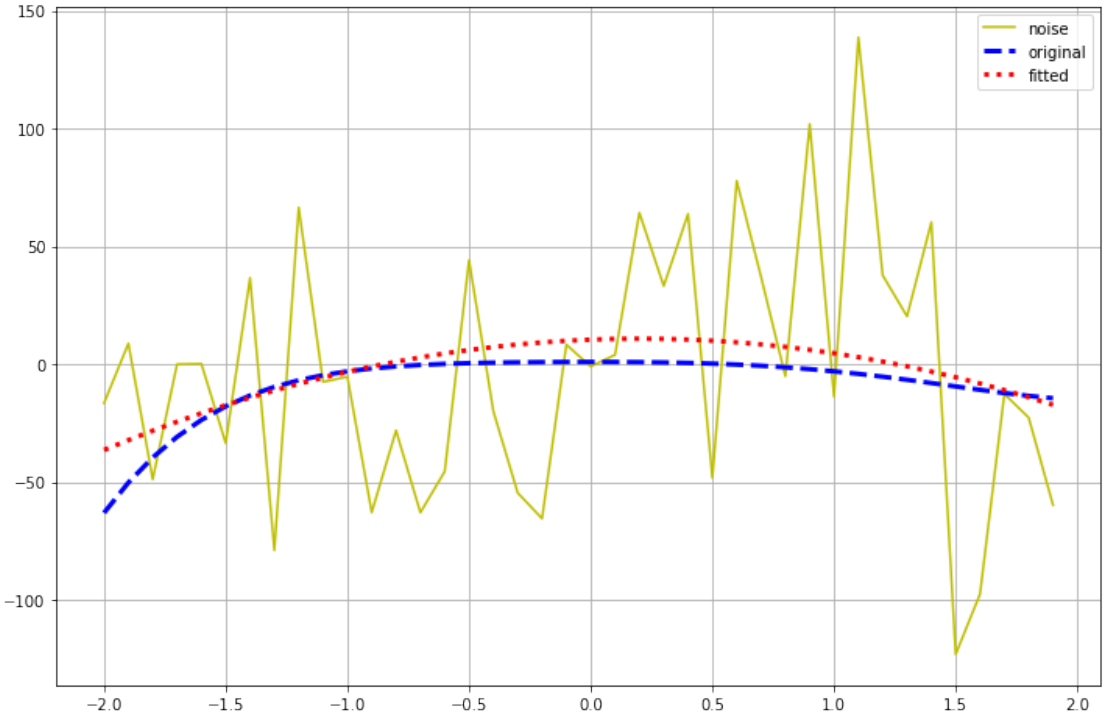
In [7]:
```
def order_5_polynomial(t, y, y_noise):

#    plt.figure(figsize=(12,8))
#    plt.plot(t, y)
#    plt.show()



#    plt.figure(figsize=(12,8))
#    plt.plot(t, y_noise)
#    plt.show()

#    plt.figure(figsize=(12,8))
#    plt.plot(t, y_noise, label='noise', color='y')
#    plt.plot(t, y, ls='dashed', lw=3, color='b', label='original')
#    plt.grid()
#    plt.legend()
#    plt.show()

# Ax = b

    A = np.hstack((t.reshape(40,1)**5, t.reshape(40,1)**4,\
                   t.reshape(40,1)**3, t.reshape(40,1)**2, \
                   t.reshape(40,1), np.ones((40,1))))
    b = y_noise.copy()
    x = np.matmul(np.linalg.pinv(A), b)

    y_hat = x[0]*t**5 + x[1]*t**4 + x[2]*t**3 + x[3]*t**2 + x[4]*t + x[5]

    plt.figure(figsize=(12,8))
    plt.plot(t, y_noise, label='noise', color='y')
    plt.plot(t, y, ls='dashed', lw=3, color='b', label='original')
    plt.plot(t, y_hat, ls='dotted', lw=3, color='r', label='fitted')
#    fp1 = np.polyfit(t, y_noise, 4)
```

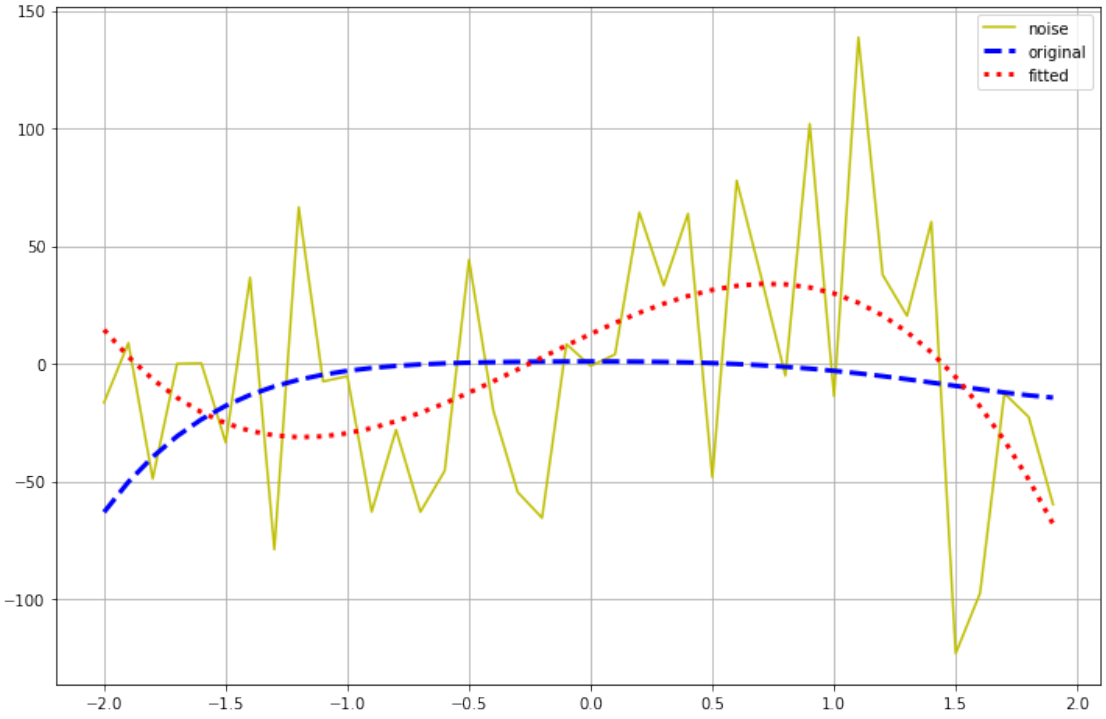
```
    #     f1 = np.poly1d(fp1)
    #     plt.plot(t, f1(t), lw=2, color='g', label='polyfit')
        plt.grid()
        plt.legend()
        plt.show()

        return y_hat
```

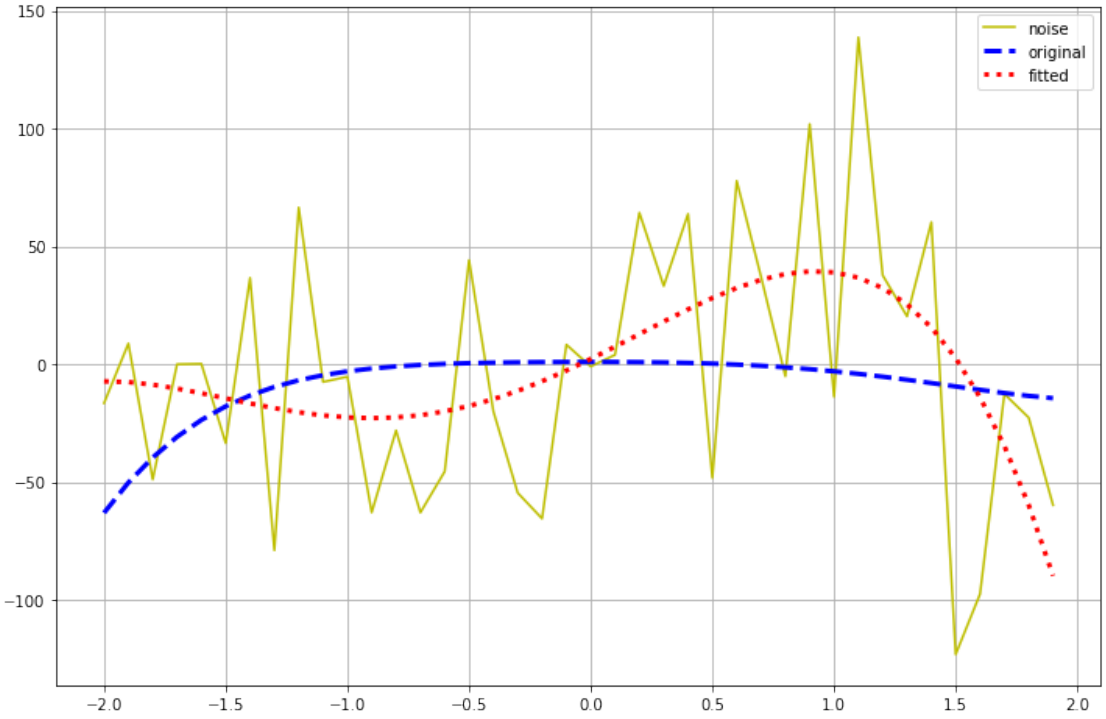
$$y = t^5 - 2t^4 - t^3 - 2t^2 + 1$$

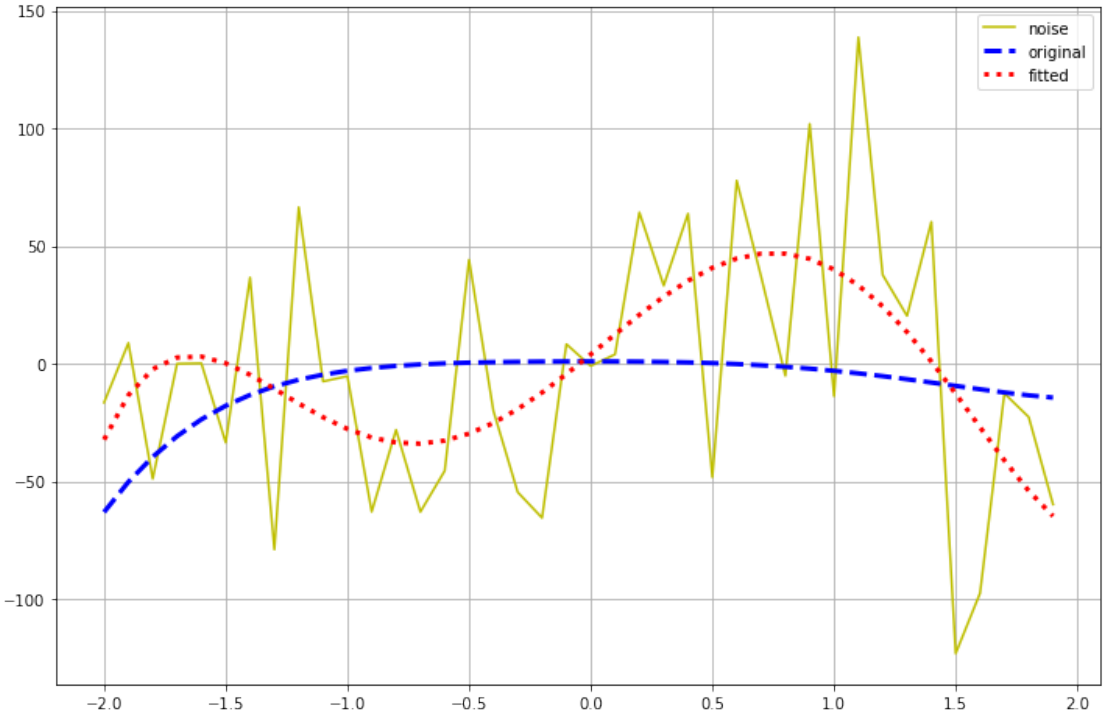
In [8]:
```
t = np.arange(-2, 2, 0.1)
y = t**5 -2*t**4 - t**3 -2*t**2 + 1
y_noise = y + np.random.randn(len(y))*50
order_1_polynomial(t,y,y_noise)
order_2_polynomial(t,y,y_noise)
order_3_polynomial(t,y,y_noise)
order_4_polynomial(t,y,y_noise)
order_5_polynomial(t,y,y_noise)
```




noise
original
fitted
150
100
50
0
−50
−100
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


noise
original
fitted
150
100
50
0
−50
−100
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

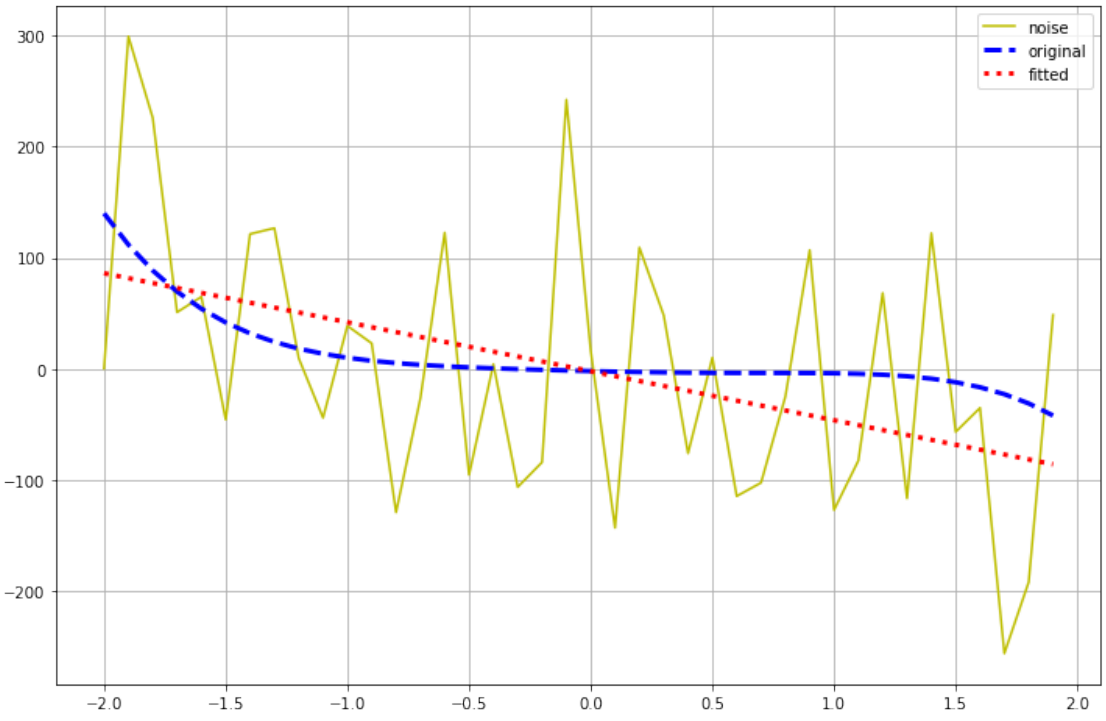

noise
original
fitted
150
100
50
0
−50
−100
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

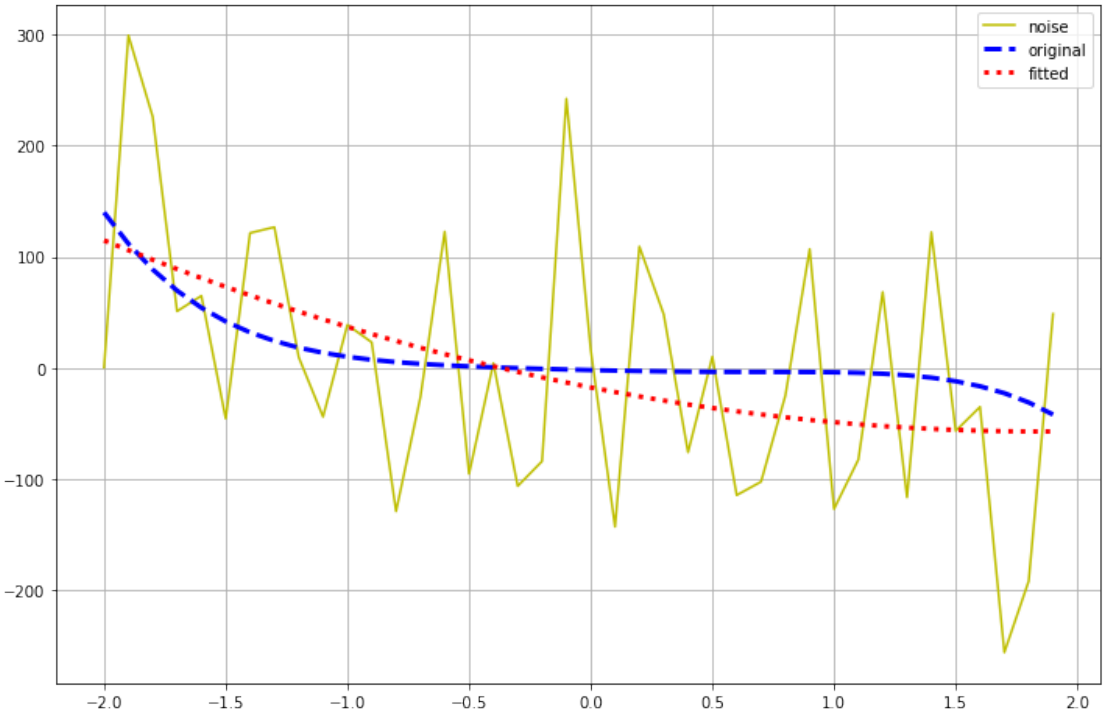

noise
original
fitted
150
100
50
0
−50
−100
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

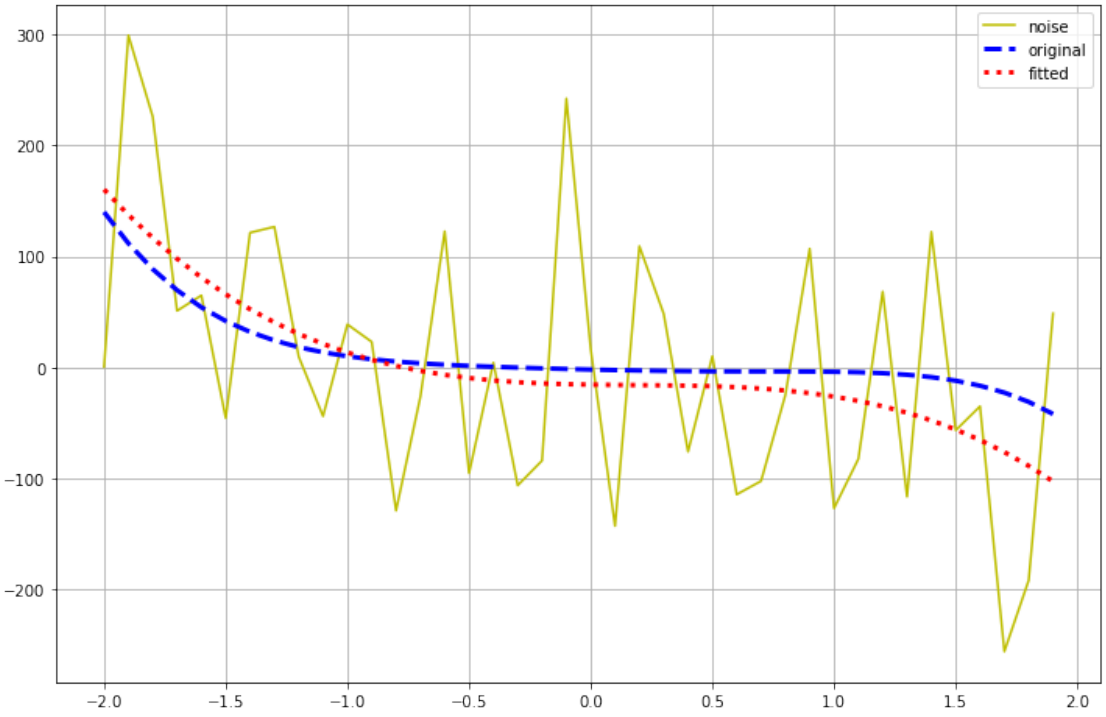
```
Out[8]: array([-32.21874336, -13.32783786,  -2.28698702,   2.65533006,
                 3.04086279,   0.21252153,  -4.67469002, -10.64274234,
               -16.87964771, -22.72852778, -27.67668113, -31.34465086,
               -33.47529216, -33.92283991, -32.64197621, -29.676898  ,
               -25.1503846 , -19.25286534, -12.23148705,  -4.37918174,
                 3.9762659 ,  12.48315089,  20.77678042,  28.4904062 ,
                35.26615696,  40.76597084,  44.68252781,  46.7501821 ,
                46.75589463,  44.55016541,  40.05796601,  33.28967192,
                24.35199504,  13.45891605,   0.94261686, -12.73558696,
               -26.9743138 , -41.02118351, -53.96188506, -64.70924405])
```

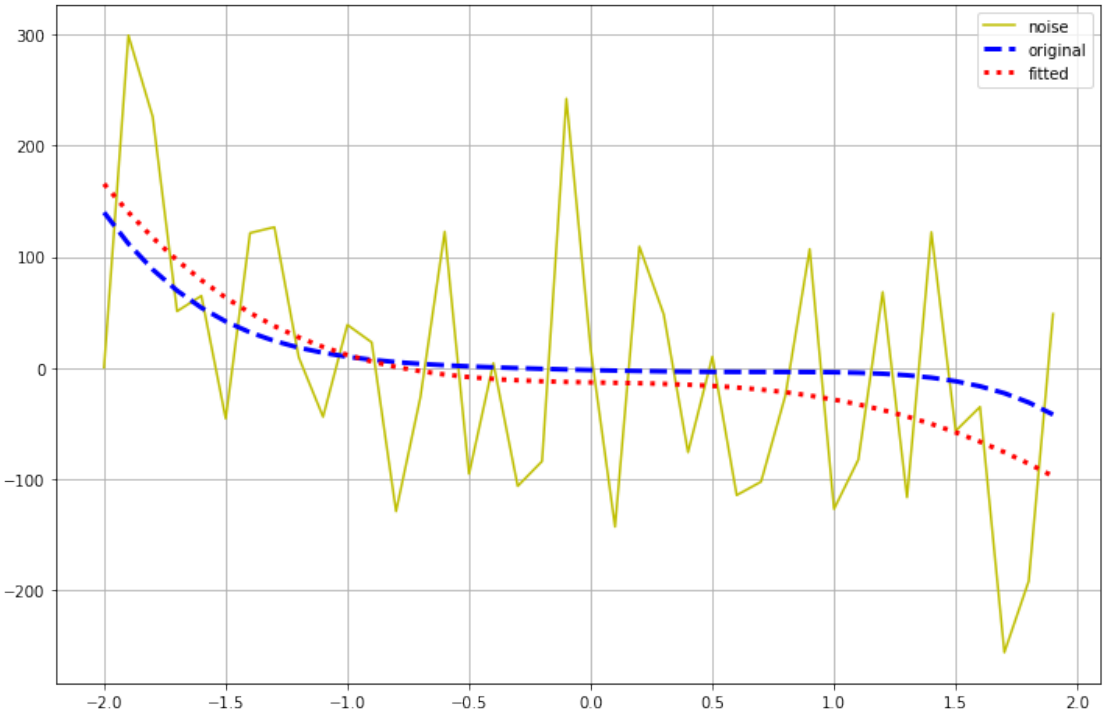
6 $y = -3t^5 + 2t^4 + t^3 + 3t^2 - 5t - 2$

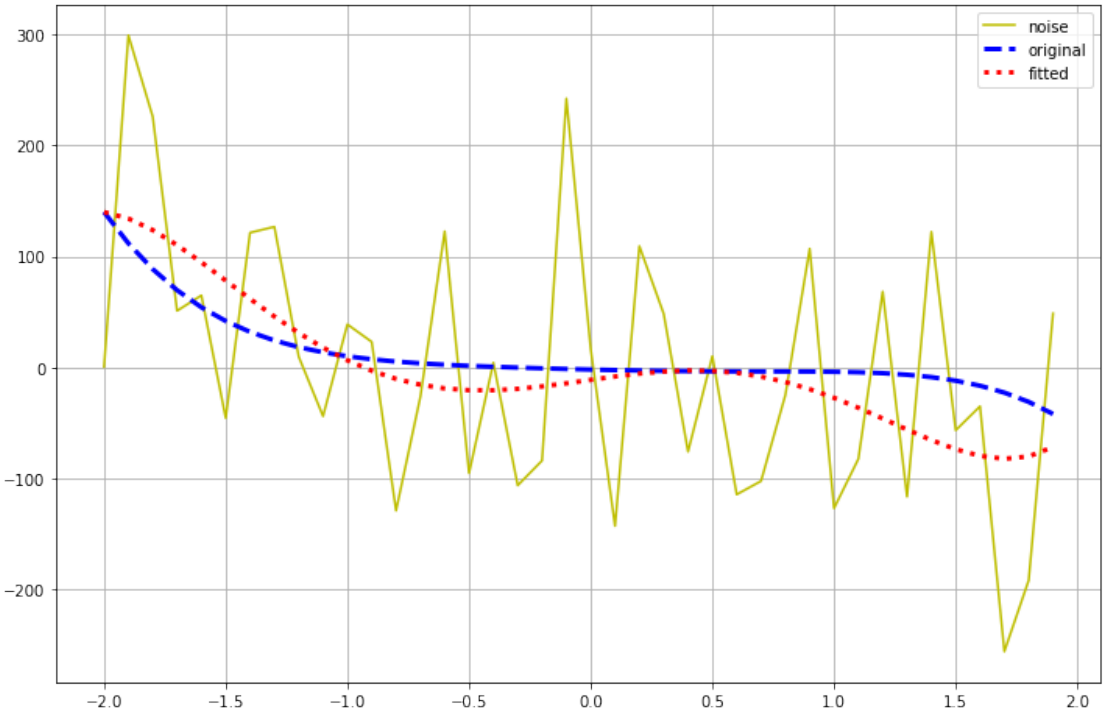
```
In [9]: t = np.arange(-2, 2, 0.1)
        y = -3*t**5 + 2*t**4 + t**3 + 3*t**2 - 5*t -2
        y_noise = y + np.random.randn(len(y))*100
        order_1_polynomial(t,y,y_noise)
        order_2_polynomial(t,y,y_noise)
        order_3_polynomial(t,y,y_noise)
        order_4_polynomial(t,y,y_noise)
        order_5_polynomial(t,y,y_noise)
```




noise
original
fitted
300
200
100
0
−100
−200
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


noise
original
fitted
300
200
100
0
−100
−200
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

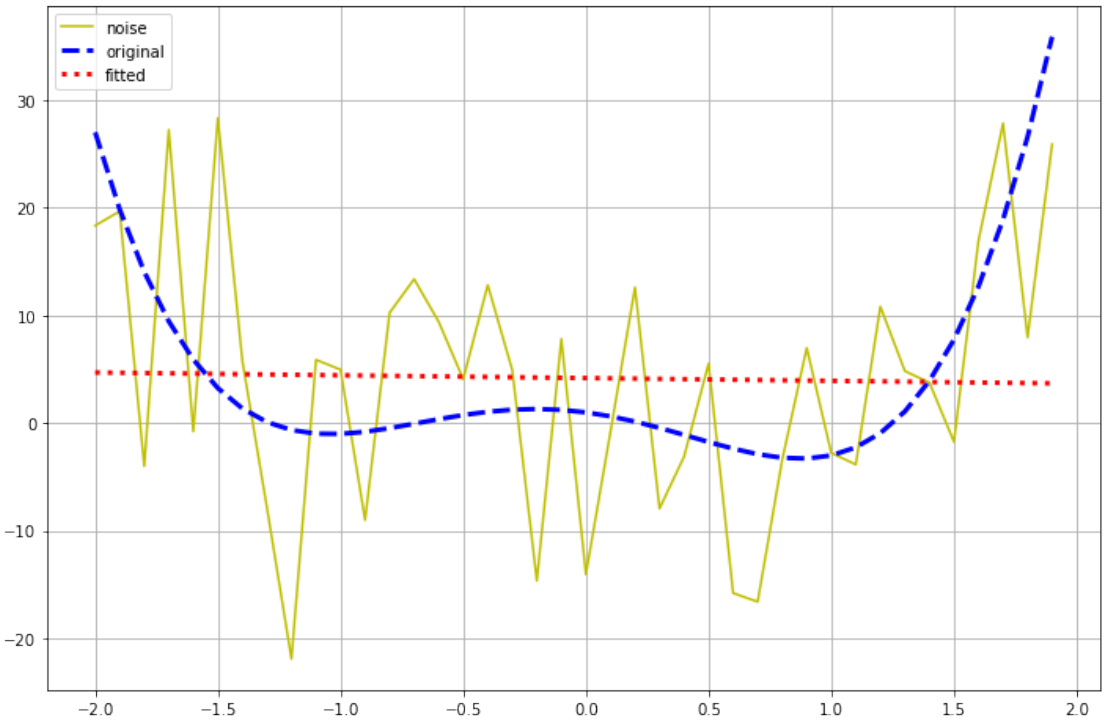

noise
original
fitted
300
200
100
0
−100
−200
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

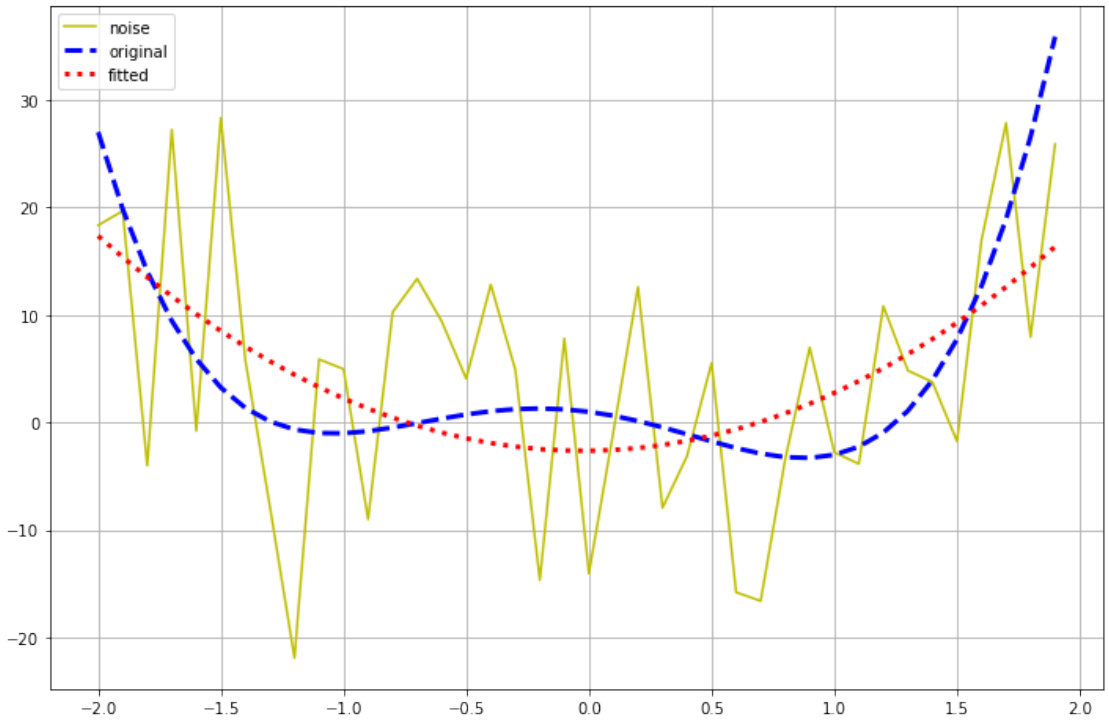

noise
original
fitted
300
200
100
0
−100
−200
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

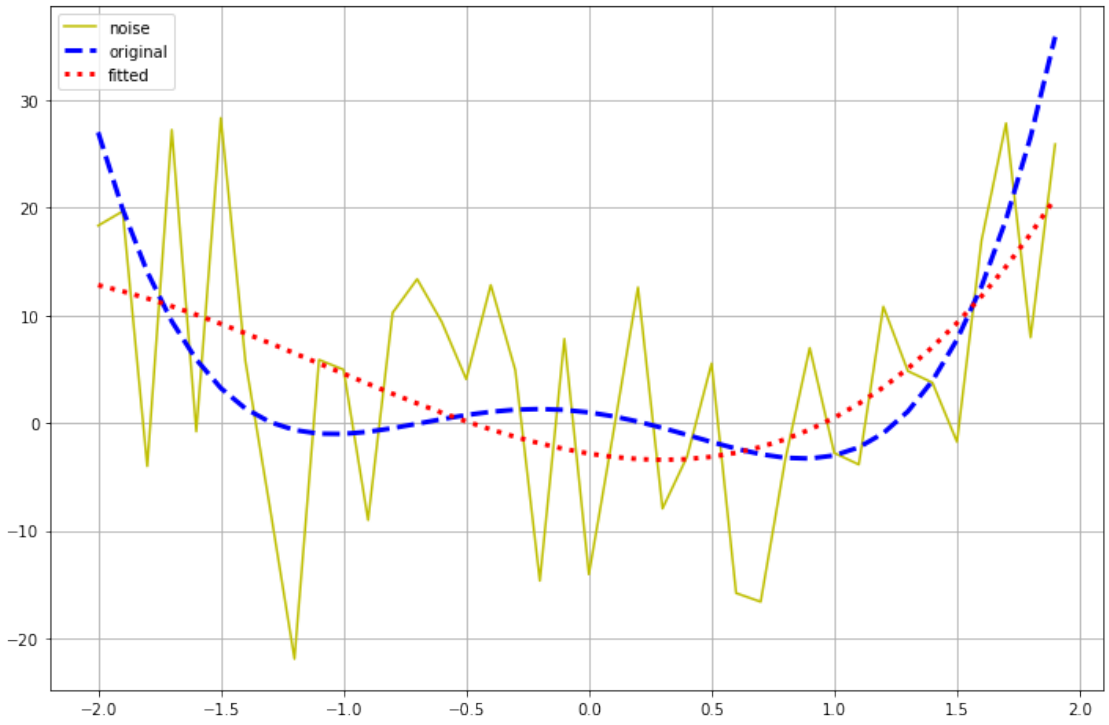
```
Out[9]: array([139.98595402, 134.15556855, 123.77361567, 110.23178663,
                94.72970048,  78.28613734,  61.75027179,  45.81290611,
                31.01770369,  17.77242232,   6.36014753,  -3.04947411,
               -10.38900162, -15.68196632, -19.03163847, -20.60979397,
               -20.64548101, -19.41378677, -17.22460405, -14.41139798,
               -11.31997269,  -8.29723795,  -5.67997589,  -3.78360761,
                -2.89095993,  -3.24103198,  -5.01776195,  -8.3387937 ,
               -13.24424347, -19.68546653, -27.51382387, -36.46944886,
               -46.17001393, -56.09949724, -65.59694934, -73.84525987,
               -79.8599242 , -82.47781013, -80.34592454, -71.91018007])
```

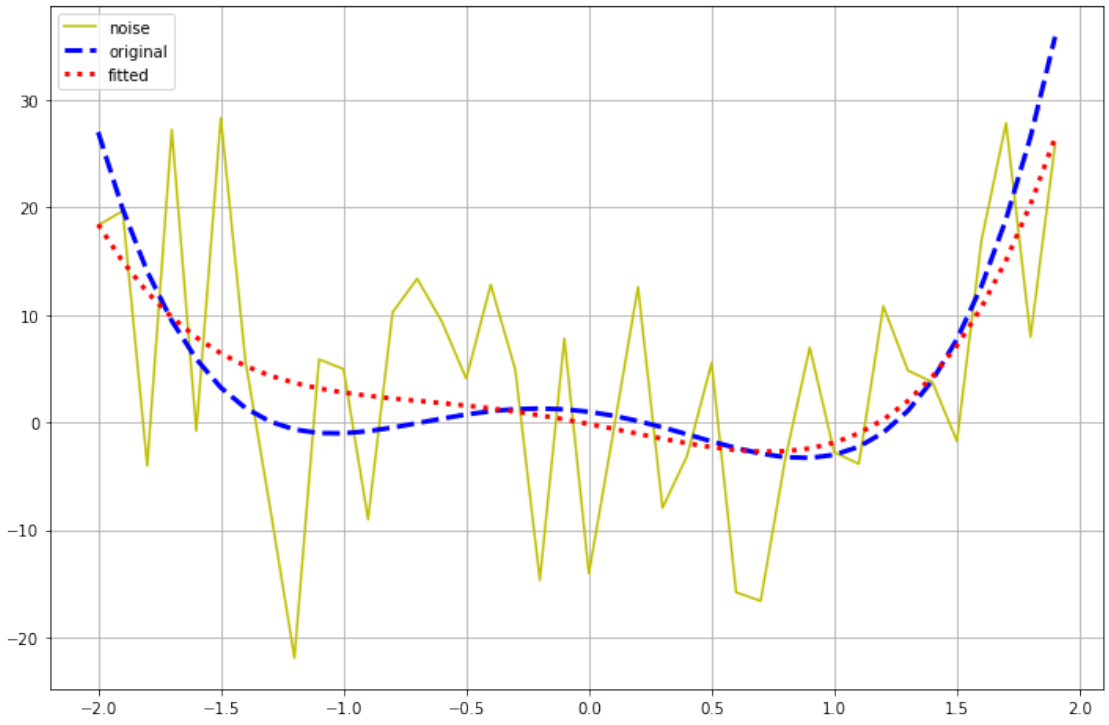
7 $y = 4t^4 + 2t^3 - 7t^2 - 3t + 1$

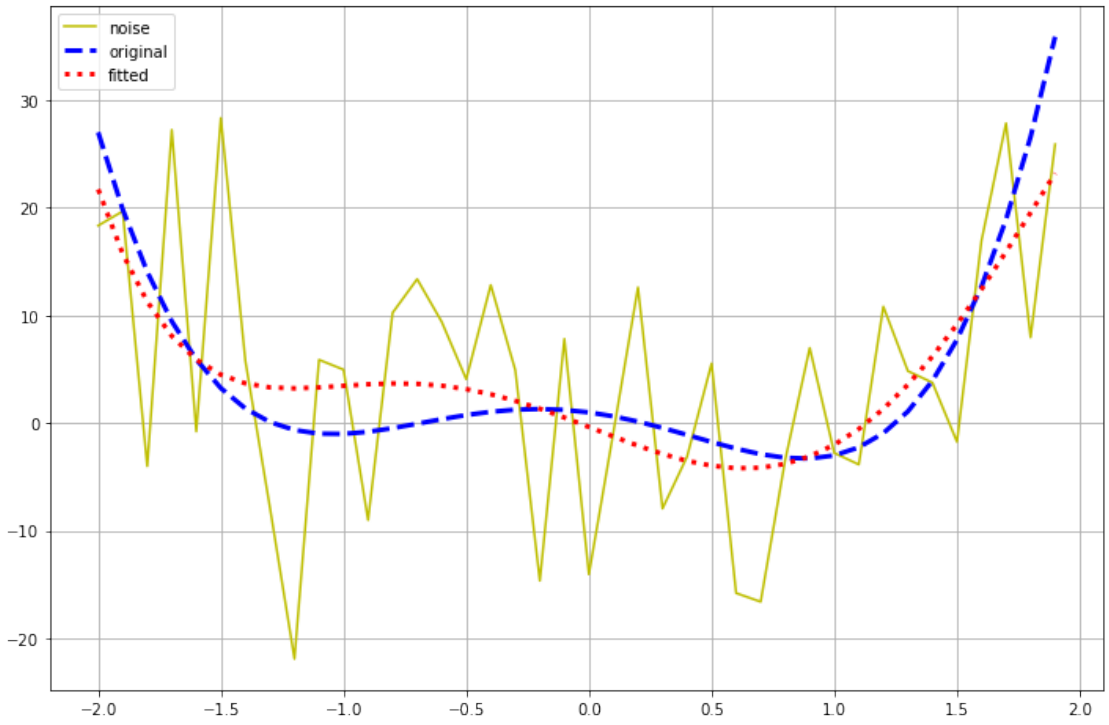
```
In [10]: t = np.arange(-2, 2, 0.1)
         y = 4*t**4 + 2*t**3 - 7*t**2 - 3*t + 1
         y_noise = y + np.random.randn(len(y))*10
         order_1_polynomial(t,y,y_noise)
         order_2_polynomial(t,y,y_noise)
         order_3_polynomial(t,y,y_noise)
         order_4_polynomial(t,y,y_noise)
         order_5_polynomial(t,y,y_noise)
```




noise
original
fitted
30
20
10
0
−10
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


noise
original
fitted
30
20
10
0
−10
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

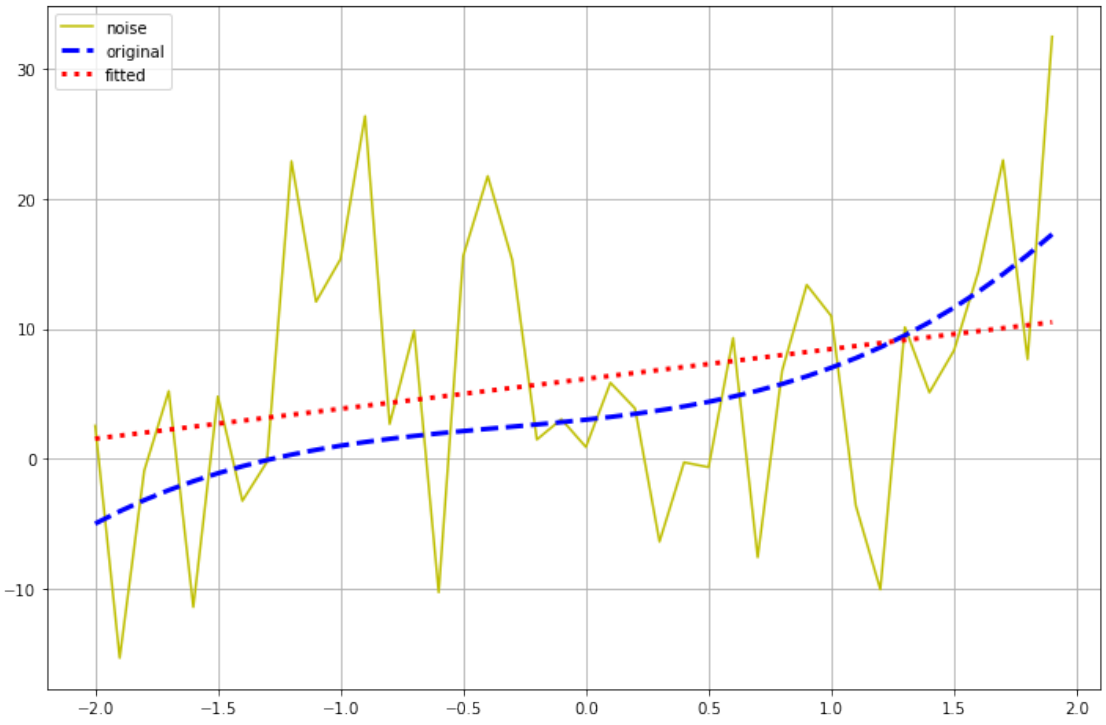

noise
original
fitted
30
20
10
0
−10
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

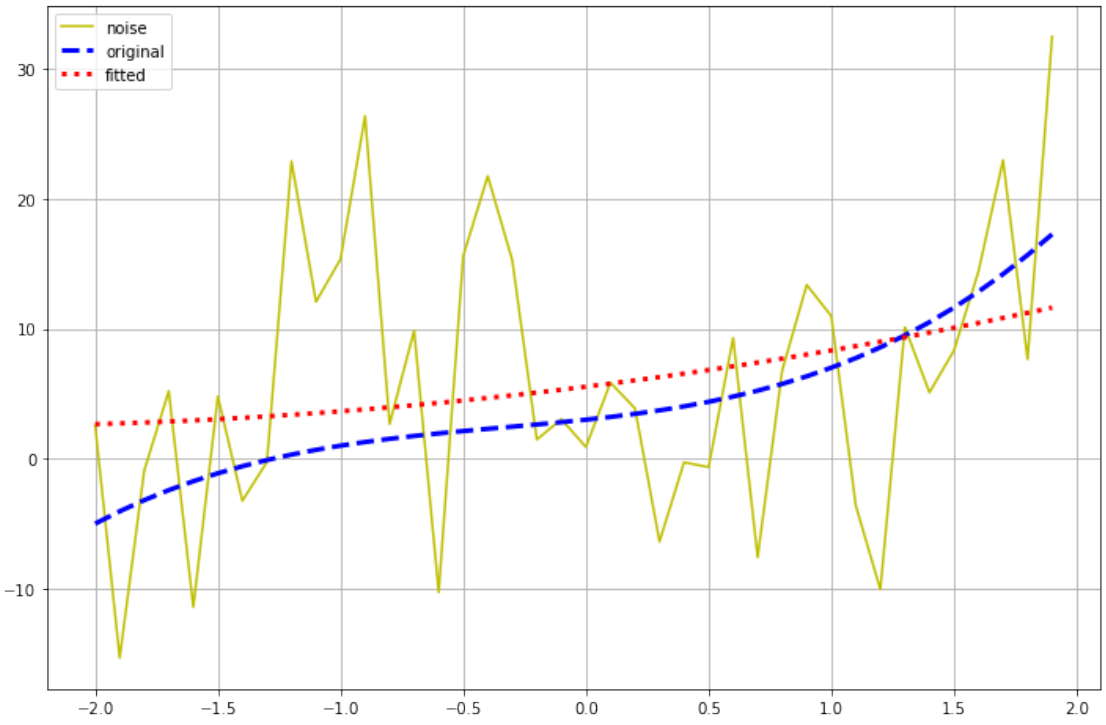

noise
original
fitted
30
20
10
0
−10
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

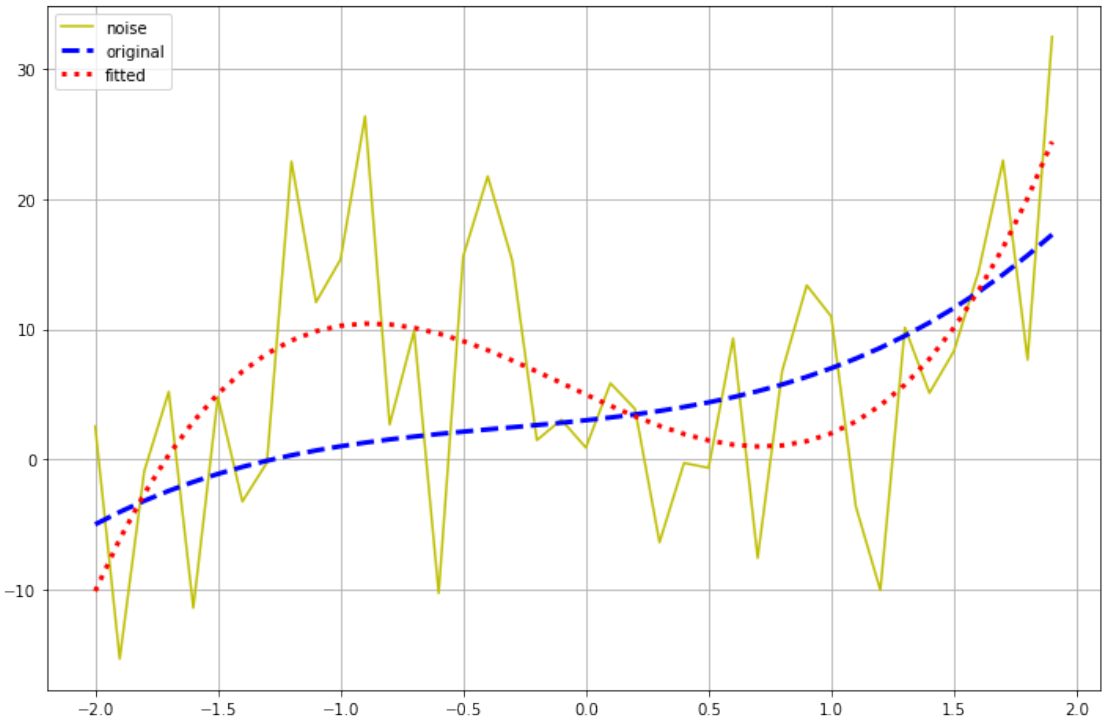
```
Out[10]: array([21.70539474, 15.72481649, 11.27623841,  8.08208972,  5.89477456,
                 4.49523178,  3.69149469,  3.31725088,  3.23040189,  3.31162308,
                 3.46292333,  3.60620484,  3.68182289,  3.6471456 ,  3.47511372,
                 3.15280037,  2.67997083,  2.06764228,  1.33664361,  0.51617516,
                -0.35763151, -1.24315385, -2.09471856, -2.86404179, -3.50166942,
                -3.95841725, -4.18681125, -4.14252781, -3.78583393, -3.08302748,
                -2.00787744, -0.5430641 ,  1.31838066,  3.57163319,  6.19863781,
                 9.16666659, 12.42687908, 15.91288215, 19.53928968, 23.20028239])
```

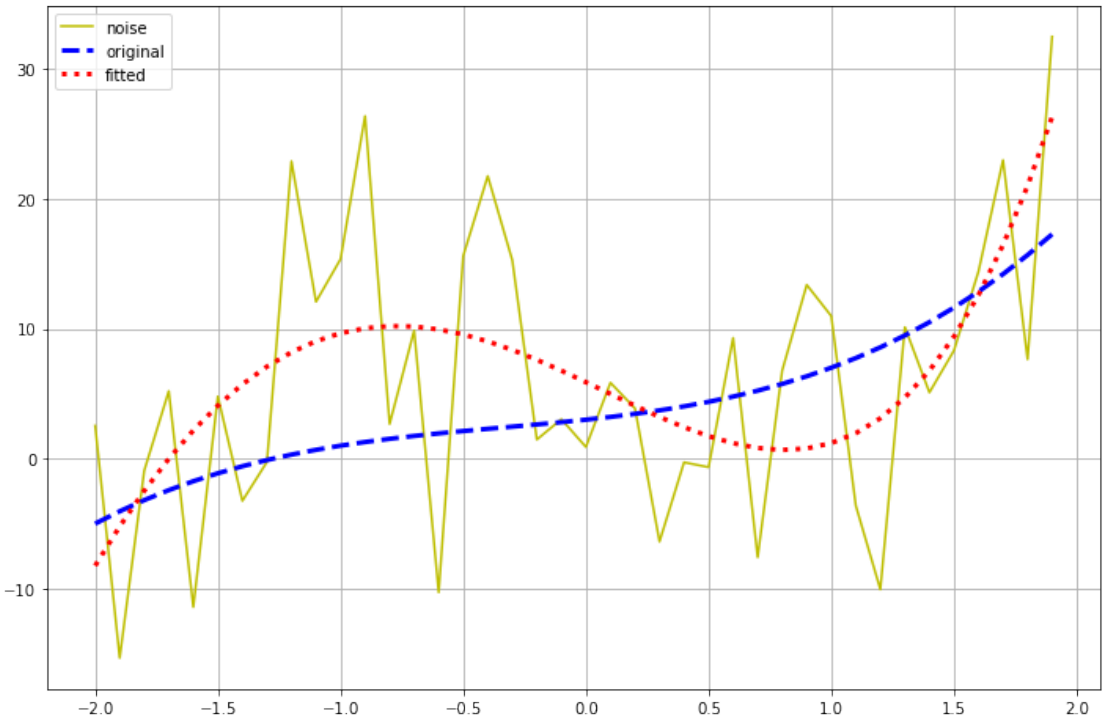
8 $y = t^3 + t^2 + 2t + 3$

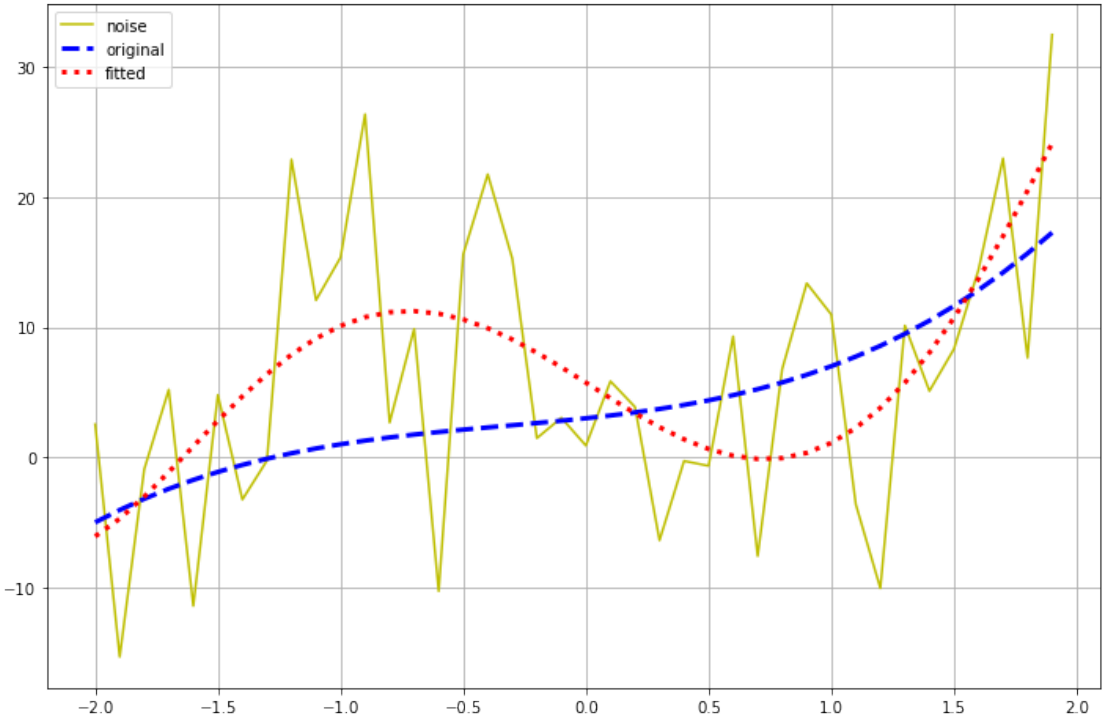
```
In [11]: t = np.arange(-2, 2, 0.1)
         y = t**3 + t**2 + 2*t + 3
         y_noise = y + np.random.randn(len(y))*10
         order_1_polynomial(t,y,y_noise)
         order_2_polynomial(t,y,y_noise)
         order_3_polynomial(t,y,y_noise)
         order_4_polynomial(t,y,y_noise)
         order_5_polynomial(t,y,y_noise)
```




noise
original
fitted
30
20
10
0
−10
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


noise
original
fitted
30
20
10
0
−10
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

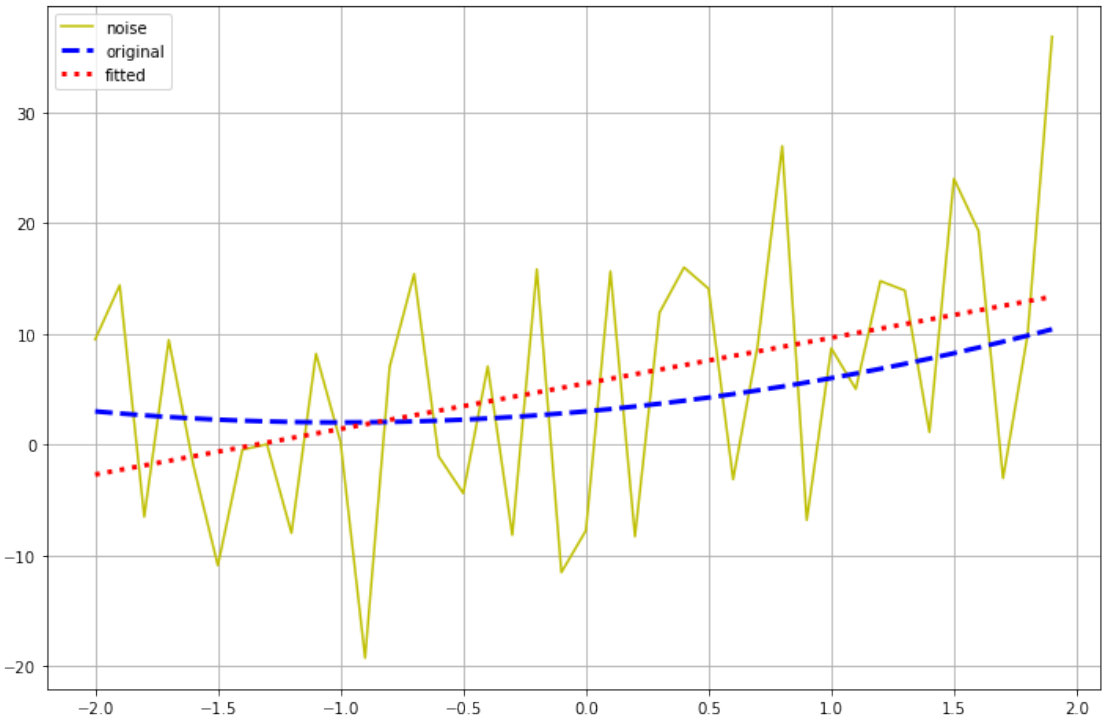

noise
original
fitted
30
20
10
0
−10
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

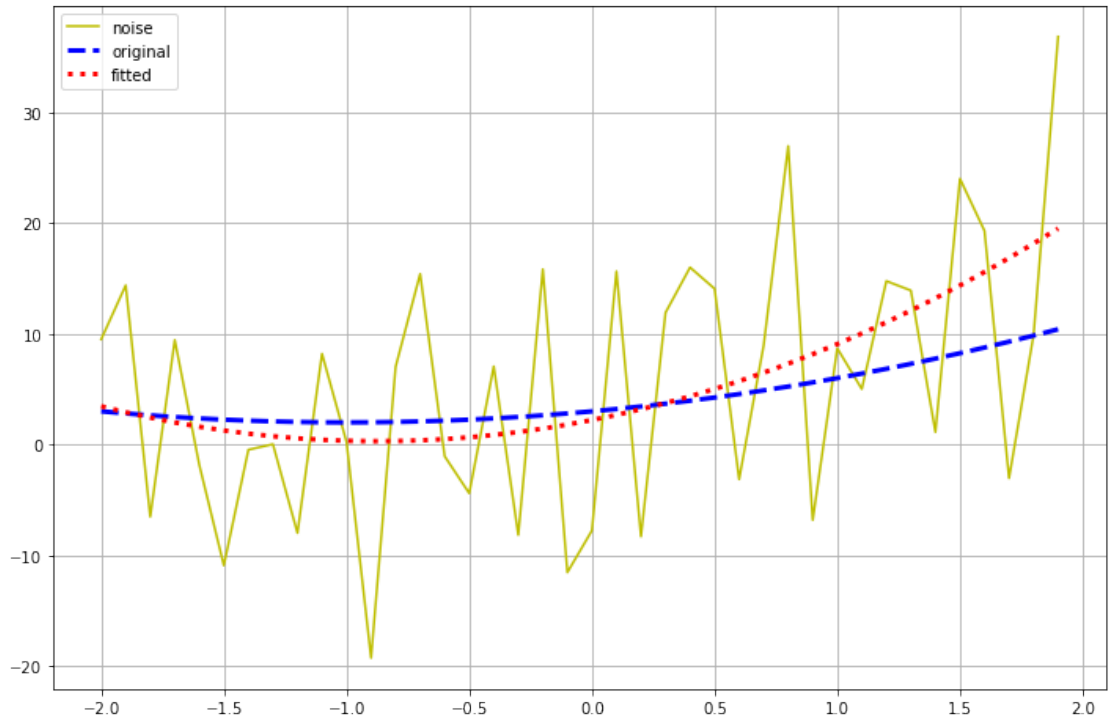

noise
original
fitted
30
20
10
0
−10
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

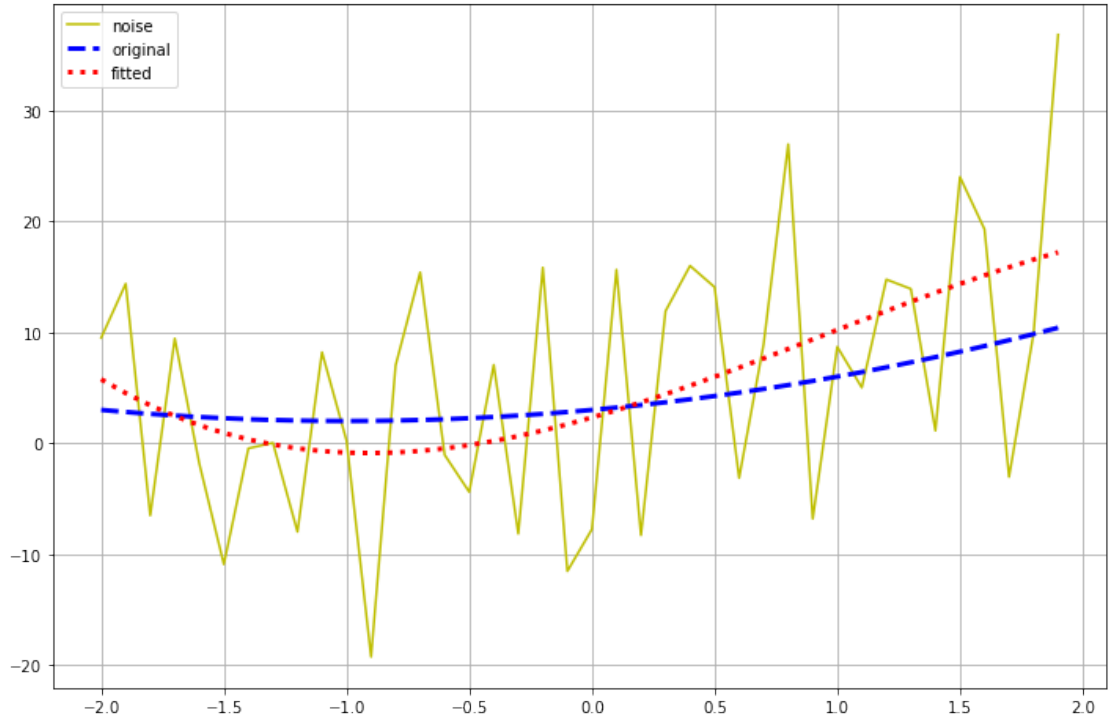
```
Out[11]: array([-6.0751988 , -4.71641042, -3.02355275, -1.13016573,  0.84839042,
                 2.81398823,  4.68478608,  6.39428121,  7.89036283,  9.13436517,
                10.10012051, 10.7730123 , 11.14902818, 11.23381306, 11.04172219,
                10.5948742 ,  9.92220417,  9.05851673,  8.04353906,  6.92097401,
                 5.73755311,  4.54208969,  3.38453189,  2.31501578,  1.38291836,
                 0.63591066,  0.1190108 , -0.12636293, -0.06333907,  0.33946015,
                 1.10697183,  2.25674546,  3.79799602,  5.73065699,  8.04443346,
                10.71785516, 13.71732955, 16.99619486, 20.49377317, 24.13442347])
```

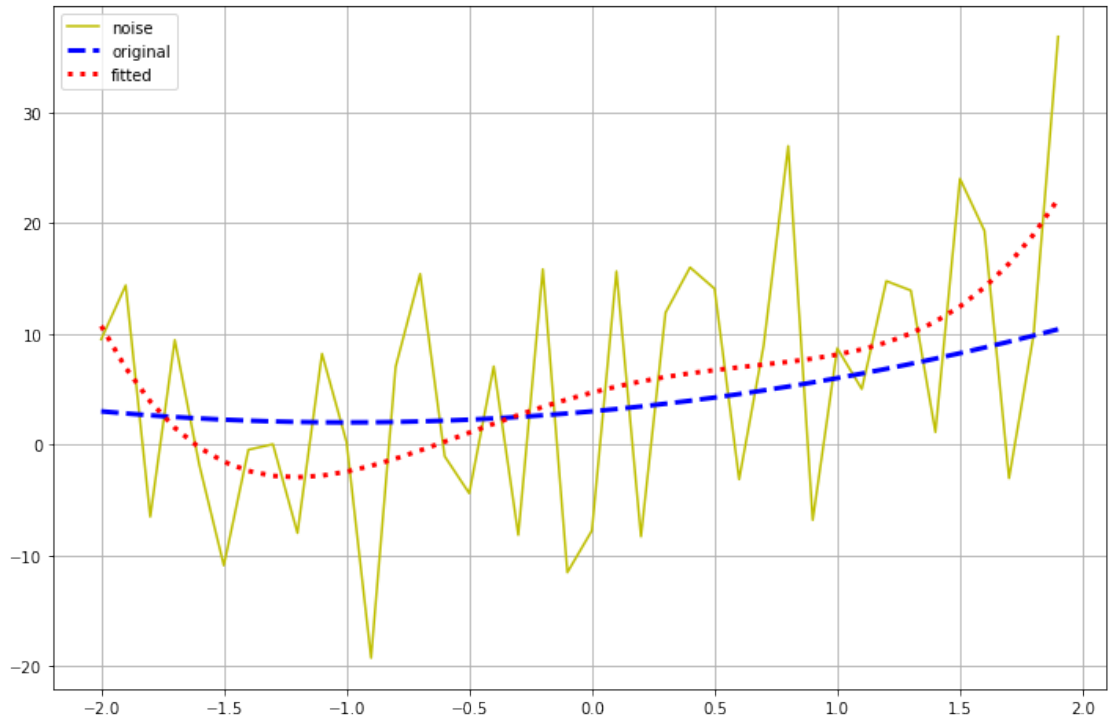
9 $y = t^2 + 2t + 3$

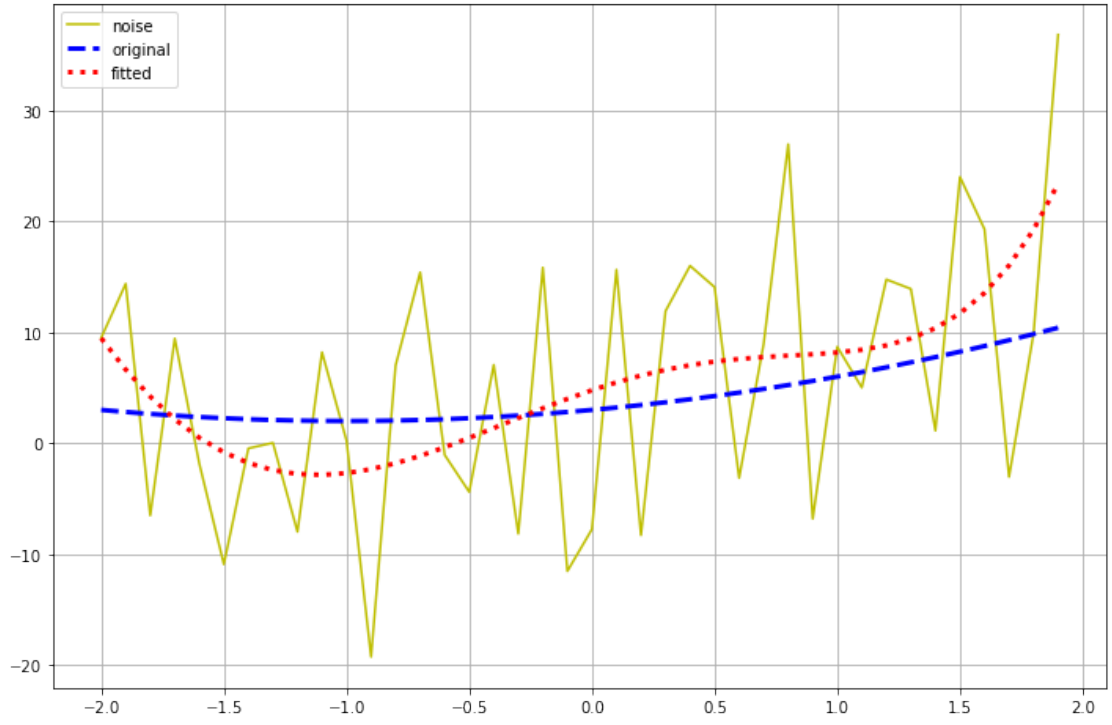
```
In [12]: t = np.arange(-2, 2, 0.1)
         y = t**2+2*t + 3
         y_noise = y + np.random.randn(len(y))*10
         order_1_polynomial(t,y,y_noise)
         order_2_polynomial(t,y,y_noise)
         order_3_polynomial(t,y,y_noise)
         order_4_polynomial(t,y,y_noise)
         order_5_polynomial(t,y,y_noise)
```




noise
original
fitted
30
20
10
0
−10
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


noise
original
fitted
30
20
10
0
−10
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

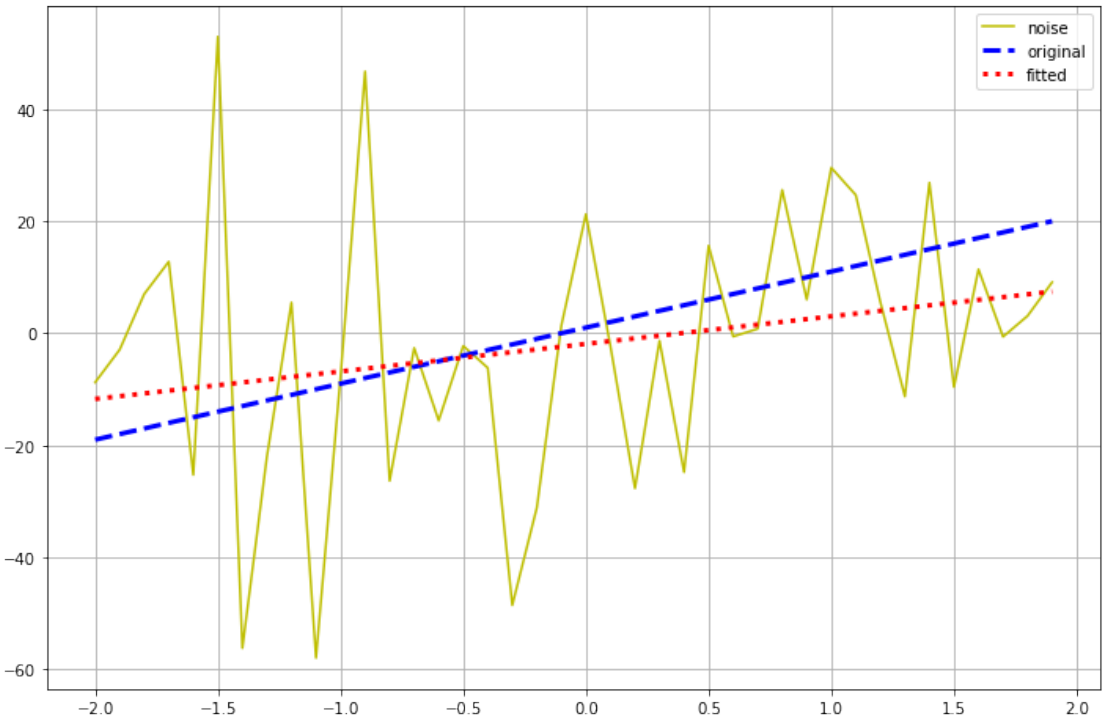

noise
original
fitted
30
20
10
0
−10
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

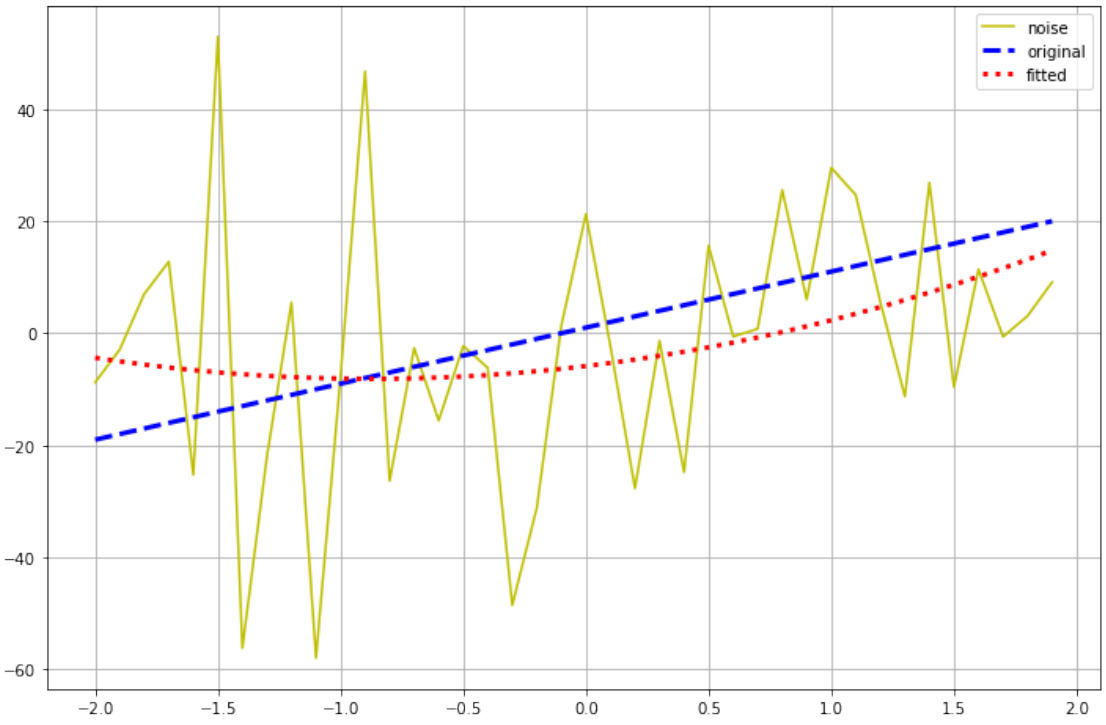

noise
original
fitted
30
20
10
0
−10
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

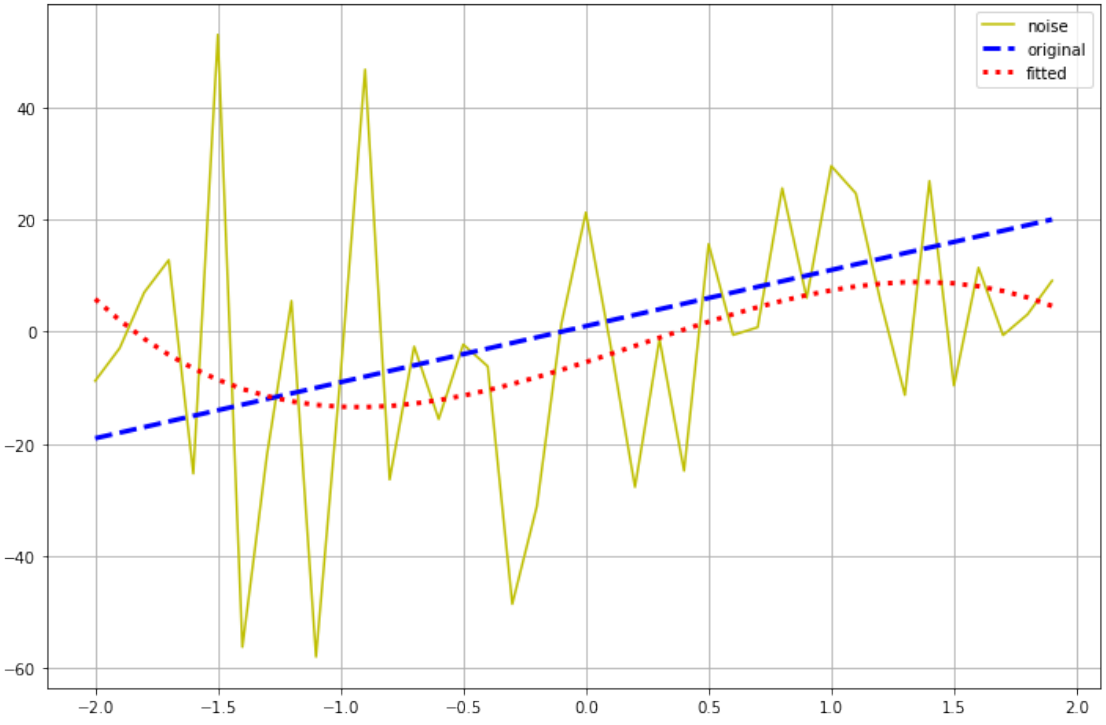
```
Out[12]: array([ 9.46342093,  6.63773103,  4.20354266,  2.15629465,  0.48609815,
                -0.82171801, -1.78606216, -2.42953452, -2.77788195, -2.85945268,
                -2.70465114, -2.34539264, -1.81455818, -1.14544915, -0.37124215,
                 0.4755563 ,  1.36365499,  2.26352325,  3.14793624,  3.99252013,
                 4.77629739,  5.48223202,  6.0977748 ,  6.6154085 ,  7.0331932 ,
                 7.35531146,  7.59261358,  7.76316287,  7.89278089,  8.01559266,
                 8.17457194,  8.42208647,  8.82044319,  9.4424335 , 10.37187853,
                11.70417432, 13.54683713, 16.02004865, 19.25720125, 23.40544321])
```

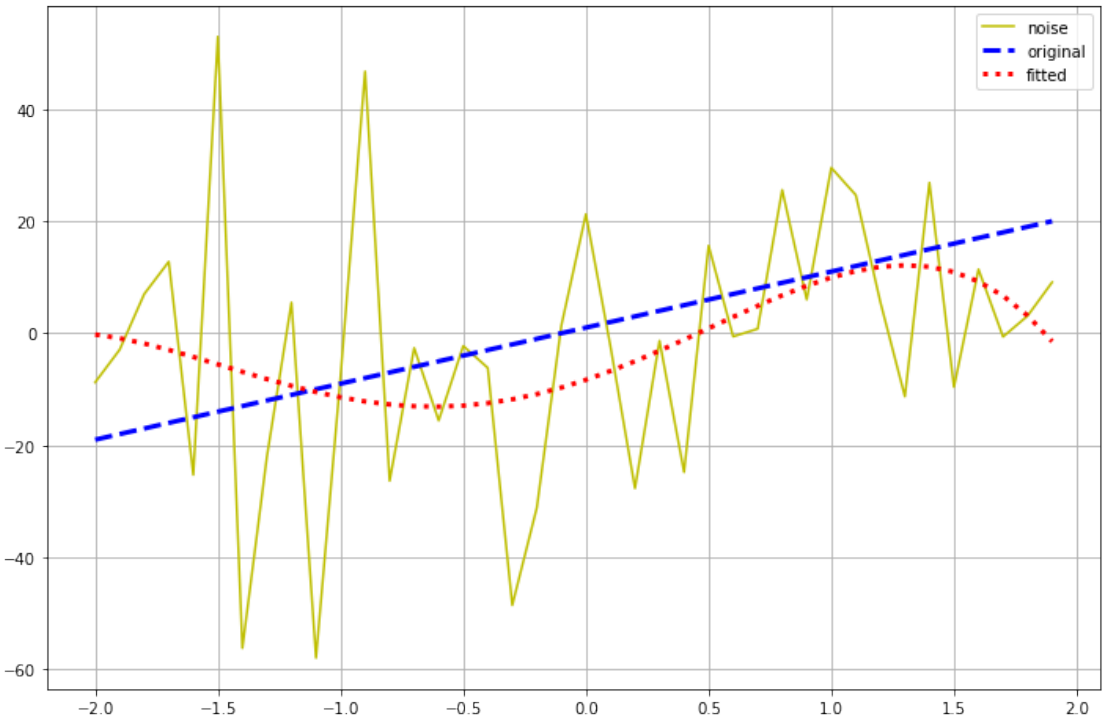
10 $y = 10t + 1$

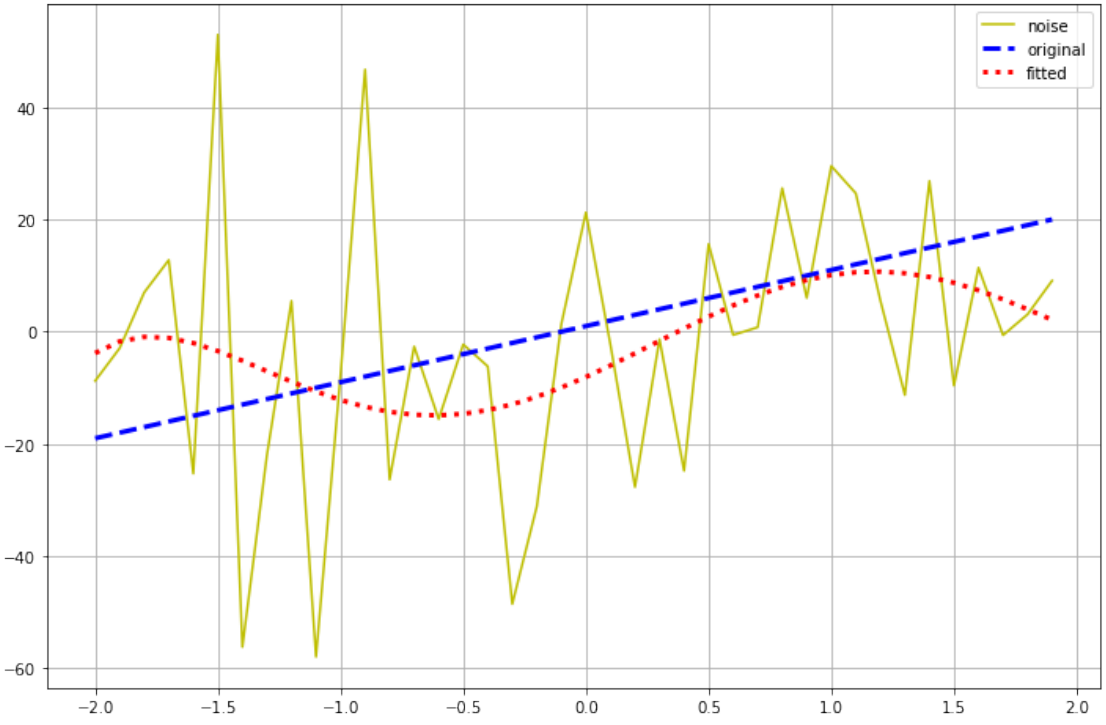
```
In [13]: t = np.arange(-2, 2, 0.1)
         y = 10*t + 1
         y_noise = y + np.random.randn(len(y))*30
         order_1_polynomial(t,y,y_noise)
         order_2_polynomial(t,y,y_noise)
         order_3_polynomial(t,y,y_noise)
         order_4_polynomial(t,y,y_noise)
         order_5_polynomial(t,y,y_noise)
```




noise
original
fitted
40
20
0
−20
−40
−60
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


noise
original
fitted
40
20
0
−20
−40
−60
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

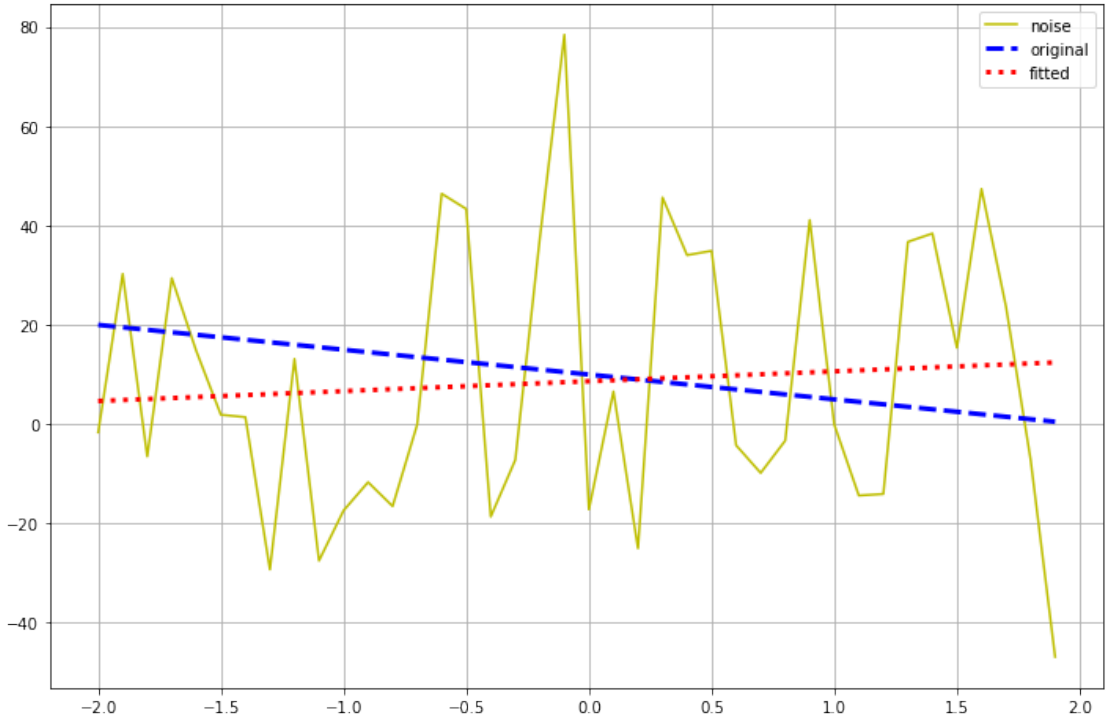

noise
original
fitted
40
20
0
−20
−40
−60
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

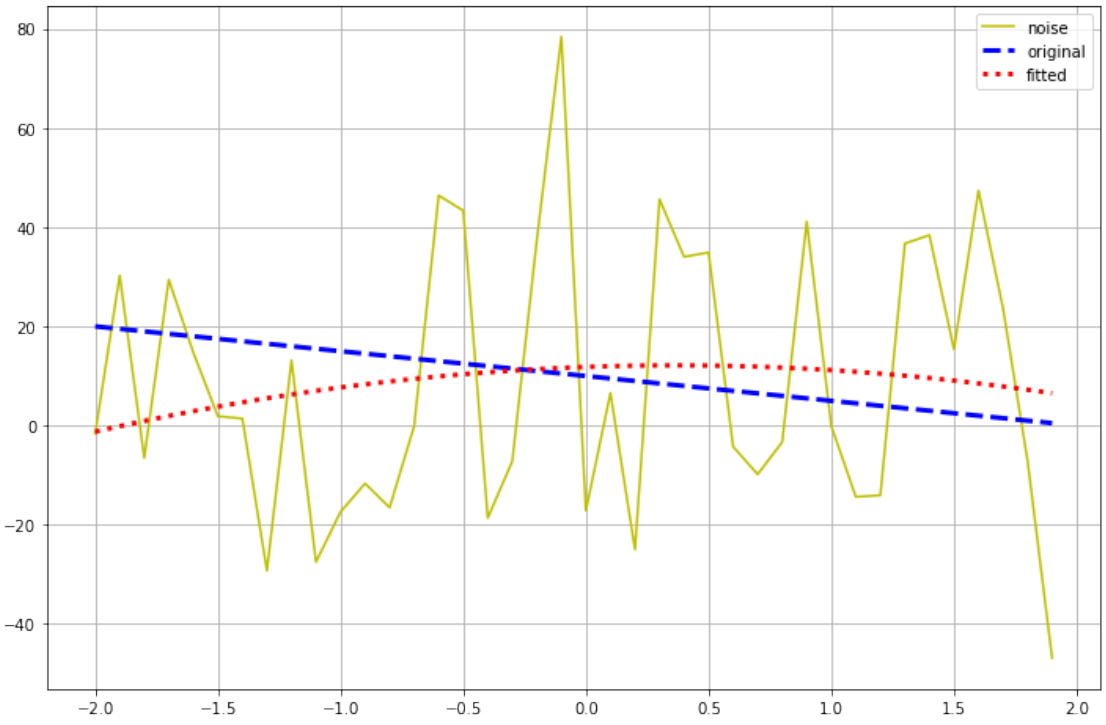

noise
original
fitted
40
20
0
−20
−40
−60
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

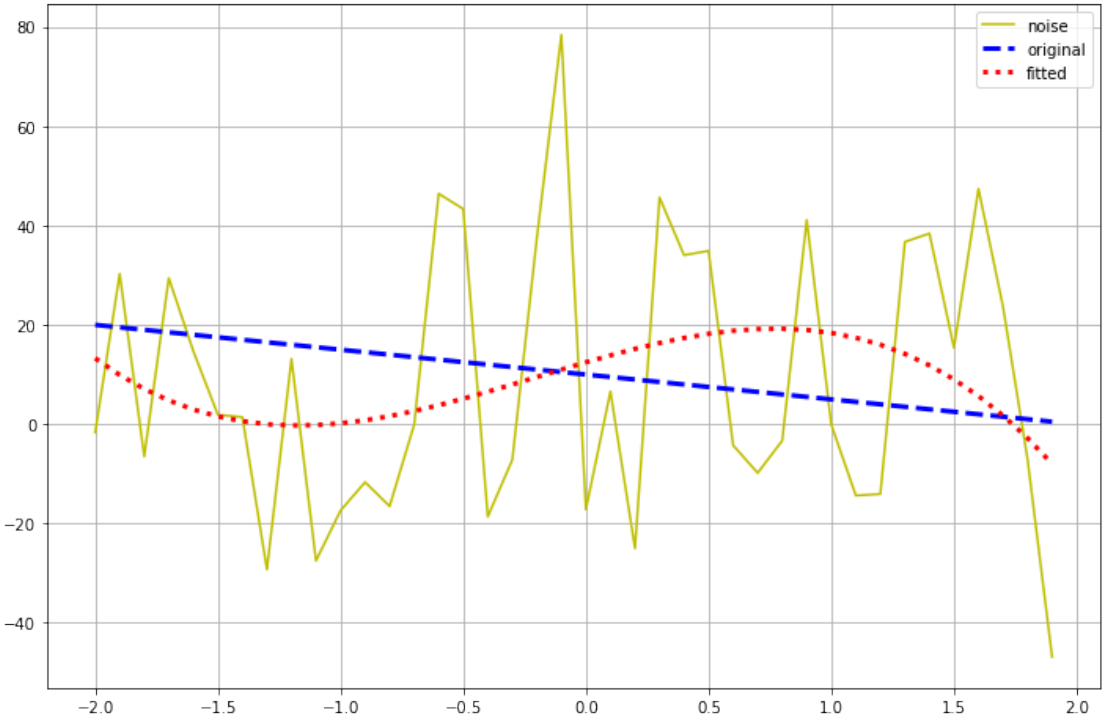
```
Out[13]: array([ -3.81960594,  -1.73334727,  -0.92705205,  -1.11029418,
                 -2.02531014,  -3.44542658,  -5.17348784,  -7.04028344,
                 -8.90297564, -10.64352696, -12.16712771, -13.40062349,
                -14.29094276, -14.80352434, -14.92074494, -14.64034669,
                -13.97386466, -12.94505441, -11.58831948,  -9.94713895,
                 -8.07249496,  -6.02130022,  -3.85482556,  -1.63712742,
                  0.56652456,   2.69122009,   4.67398062,   6.4553319 ,
                  7.98087639,   9.20286574,  10.08177329,  10.58786653,
                 10.70277958,  10.42108562,   9.75186943,   8.72029982,
                  7.36920212,   5.76063064,   3.97744115,   2.12486335])
```

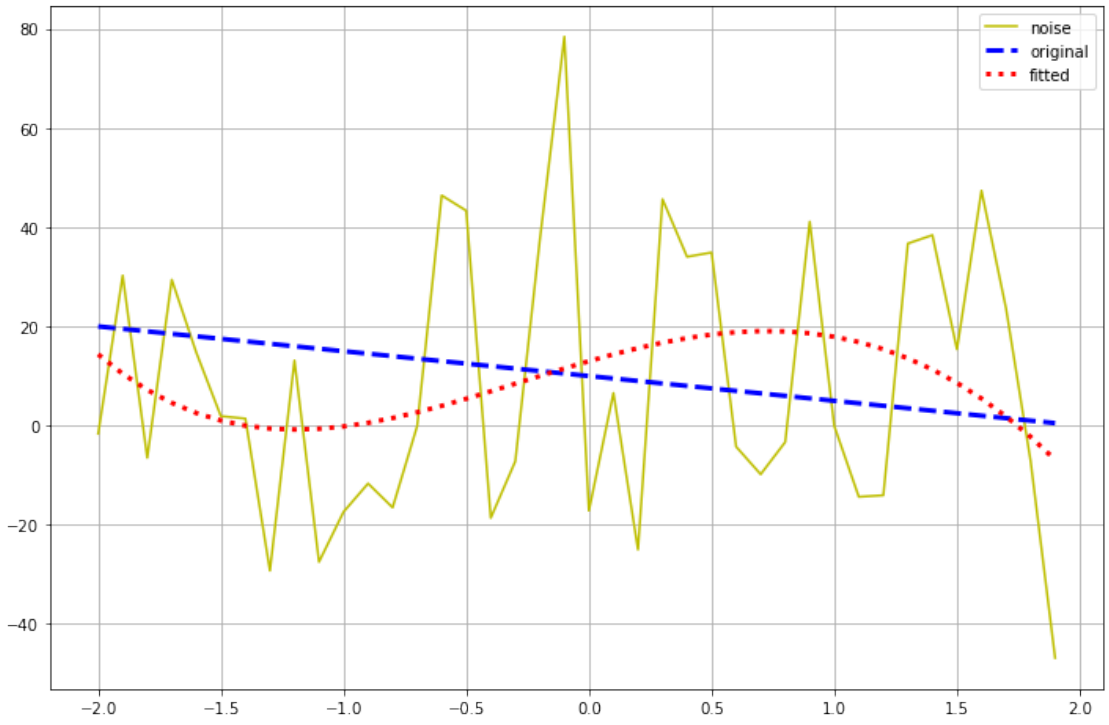
$$y = -5t + 10$$

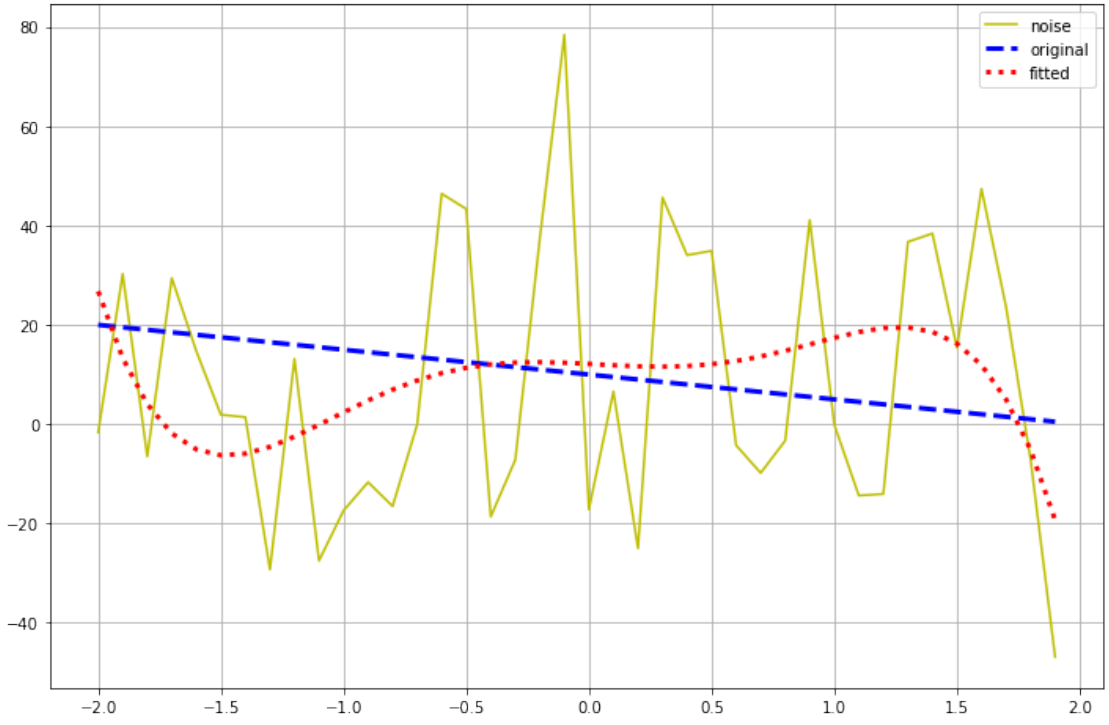
```
In [14]: t = np.arange(-2, 2, 0.1)
         y = -5*t + 10
         y_noise = y + np.random.randn(len(y))*30
         order_1_polynomial(t,y,y_noise)
         order_2_polynomial(t,y,y_noise)
         order_3_polynomial(t,y,y_noise)
         order_4_polynomial(t,y,y_noise)
         order_5_polynomial(t,y,y_noise)
```




noise
original
fitted
80
60
40
20
0
−20
−40
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


noise
original
fitted
80
60
40
20
0
−20
−40
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


noise
original
fitted
80
60
40
20
0
−20
−40
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


noise
original
fitted
80
60
40
20
0
−20
−40
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

```
Out[14]: array([ 26.76439019,  13.30744754,   4.03792435,  -1.8527402 ,
                -5.07716651,  -6.25746365,  -5.93065852,  -4.55412502,
                -2.51101315,  -0.1156782 ,   2.38089015,   4.78563864,
                 6.95802135,   8.80457047,  10.27346726,  11.3491128 ,
                12.04669894,  12.40677908,  12.48983909,  12.37086812,
                12.13392948,  11.86673149,  11.65519833,  11.57804092,
                11.70132773,  12.07305571,  12.71772105,  13.63089014,
                14.77377034,  16.06778088,  17.38912372,  18.56335439,
                19.35995284,  19.48689431,  18.5852202 ,  16.22360889,
                11.89294664,   5.00089841,  -5.13352128, -19.28537747])
```