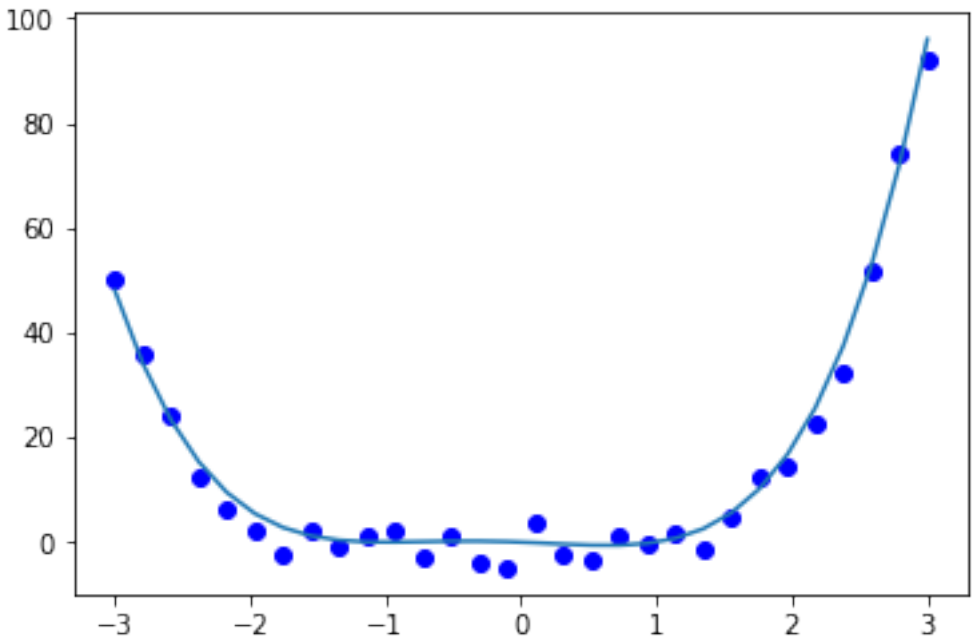
assignment08_20160040

May 21, 2019

In [8]:
```
import numpy as np
import matplotlib.pyplot as plt
import random
from numpy.linalg import *
import copy

#my initial function for generate points
def init_function(x):
    return x**4 + x**3 - x**2 - x

origin_x = np.linspace(-3,3, 30)
origin_y = [init_function(b) for b in origin_x]


# noise_y = origin_y + noise
noise_y = [init_function(c) + random.randrange(-5,5) for c in origin_x]

plt.plot(origin_x, noise_y,'bo')
plt.plot(origin_x,origin_y)
plt.show()
```

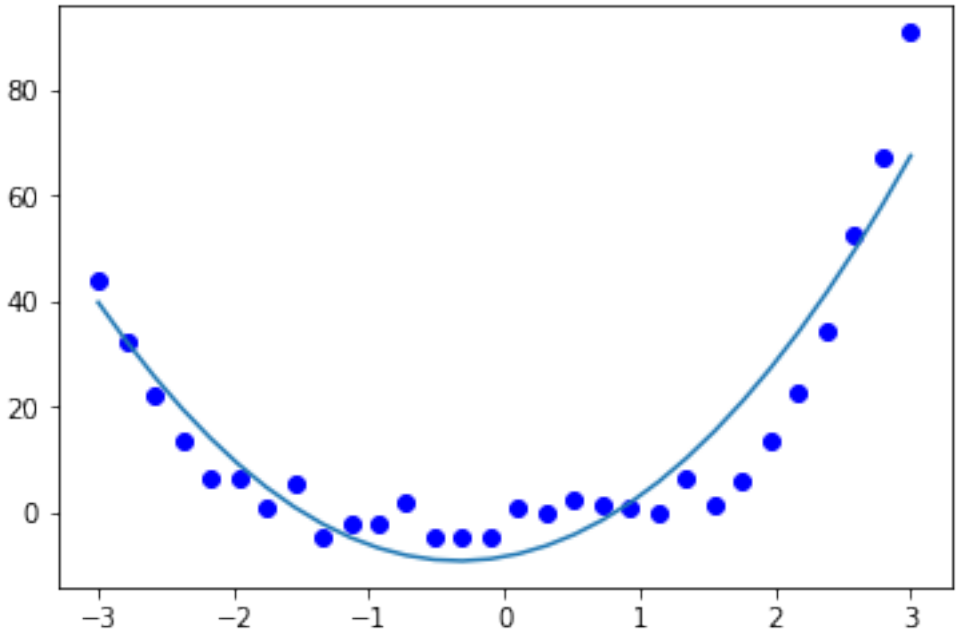



```
In [2]: def residual(result_y, noise_y):
            r = (result_y - noise_y)**2
            s = sum(r)
            return s
```

1 degree = 2

```
In [3]: def f_degree2(x,a,b,c):
            return a*x**2 + b*x + c
        row = 30
        col = 3
        f2 = np.zeros((row, col), dtype = float)
        for i in range(0,row):
            for j in range(0,col):
                f2[i][j] = copy.deepcopy(origin_x[i]**j)

        values = np.zeros((col,1), dtype = float)
        c = np.reshape(np.array(noise_y),(30,1))

        #solving a least square problem
        values = (np.mat(f2.T)*np.mat(f2)).I*np.mat(f2.T)*np.mat(c)
        values_list = np.ravel(values)

        result_y = f_degree2(origin_x,values_list[2], values_list[1], values_list[0])
        plt.plot(origin_x, noise_y,'bo')
```

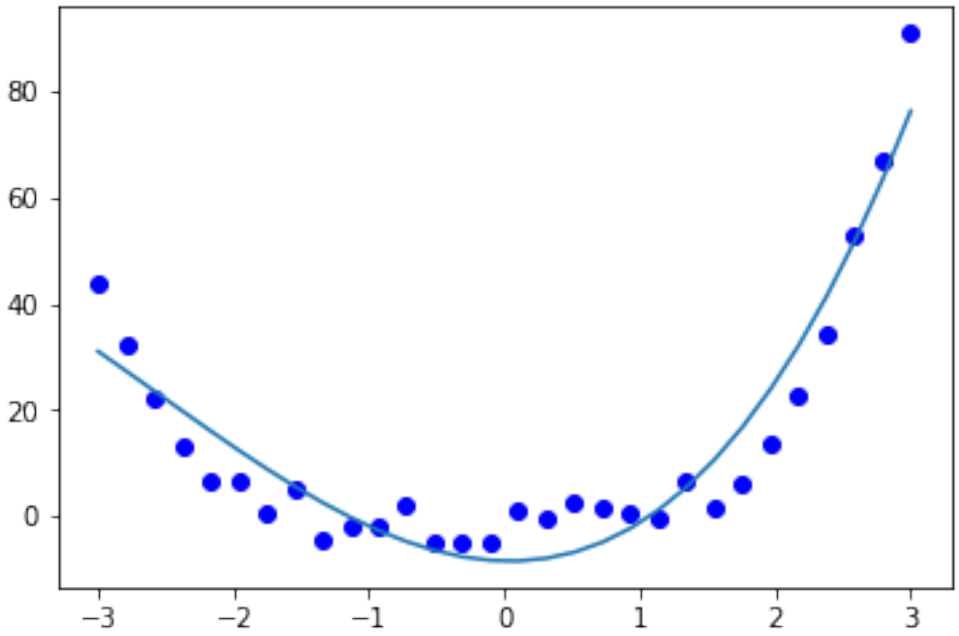
```
plt.plot(origin_x,result_y)
plt.show()

#energy
energy =  np.empty(1, dtype = float)
energy = residual(result_y, noise_y)
```



2 degree = 3

```
In [4]: def f_degree3(x,a,b,c,d):
            return a*x**3 + b*x**2 + c*x + d
        row = 30
        col = 4
        f3 = np.zeros((row, col), dtype = float)
        for i in range(0,row):
            for j in range(0,col):
                f3[i][j] = copy.deepcopy(origin_x[i]**j)
        values = np.zeros((col,1), dtype = float)
        c = np.reshape(np.array(noise_y),(30,1))

        values = (np.mat(f3.T)*np.mat(f3)).I*np.mat(f3.T)*np.mat(c)
        values_list = np.ravel(values)

        result_y = f_degree3(origin_x,values_list[3], values_list[2], values_list[1],
                             values_list[0])
```

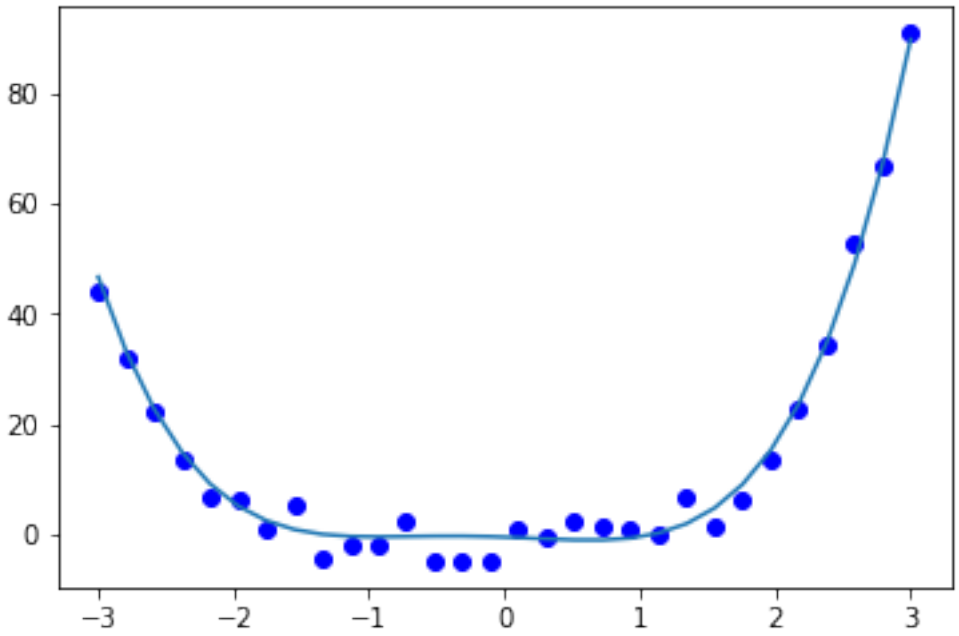
```
plt.plot(origin_x, noise_y,'bo')
plt.plot(origin_x,result_y)
plt.show()

#energy
energy = np.append(energy, residual(result_y, noise_y))
```



3 degree = 4

```
In [5]: def f_degree4(x,a,b,c,d,e):
            return a*x**4 + b*x**3 + c*x**2 + d*x + e
        row = 30
        col = 5
        f4 = np.zeros((row, col), dtype = float)
        for i in range(0,row):
            for j in range(0,col):
                f4[i][j] = copy.deepcopy(origin_x[i]**j)

        values = np.zeros((col,1), dtype = float)
        c = np.reshape(np.array(noise_y),(30,1))

        values = (np.mat(f4.T)*np.mat(f4)).I*np.mat(f4.T)*np.mat(c)
        values_list = np.ravel(values)
```

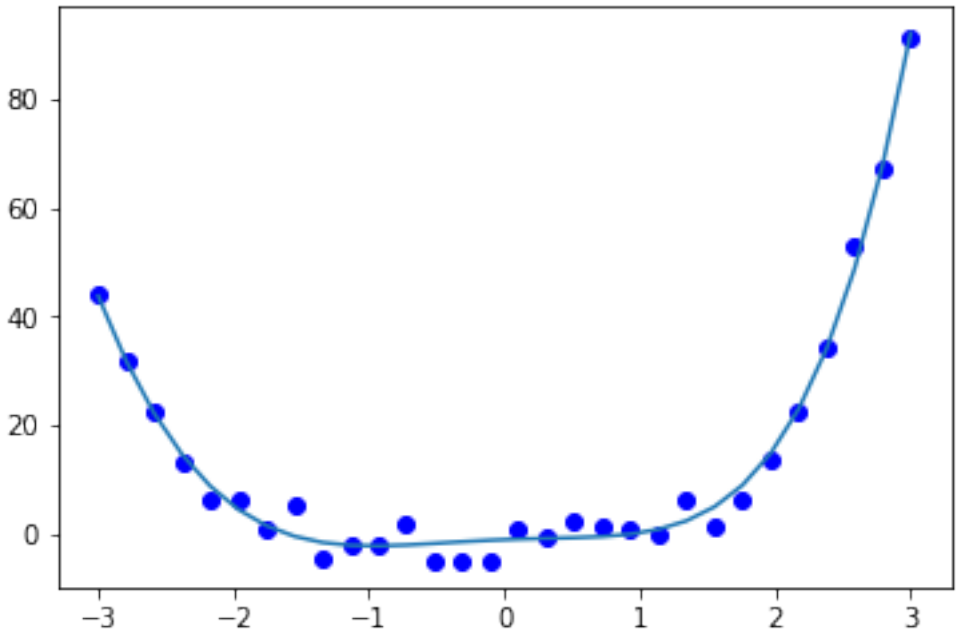
```
result_y = f_degree4(origin_x,values_list[4], values_list[3], values_list[2],
                     values_list[2], values_list[1])
plt.plot(origin_x, noise_y,'bo')
plt.plot(origin_x,result_y)
plt.show()

#energy
energy = np.append(energy, residual(result_y, noise_y))
```



4 degree = 5

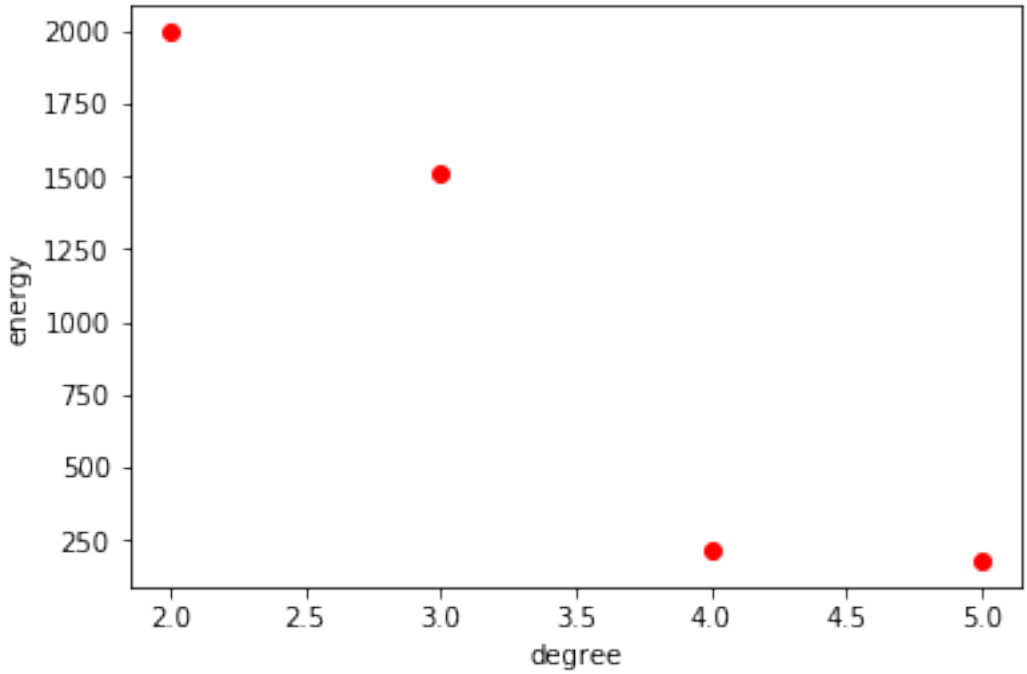
```
In [6]: def f_degree5(x,a,b,c,d,e,f):
            return a*x**5 + b*x**4 + c*x**3 + d*x**2 + e*x + f
        row = 30
        col = 6
        f5 = np.zeros((row, col), dtype = float)
        for i in range(0,row):
            for j in range(0,col):
                f5[i][j] = copy.deepcopy(origin_x[i]**j)

        values = np.zeros((col,1), dtype = float)

        c = np.reshape(np.array(noise_y),(30,1))

        values = (np.mat(f5.T)*np.mat(f5)).I*np.mat(f5.T)*np.mat(c)
```

```
values_list = np.ravel(values)

result_y = f_degree5(origin_x,values_list[5], values_list[4], values_list[3],
                     values_list[2], values_list[1],values_list[0])
plt.plot(origin_x, noise_y,'bo')
plt.plot(origin_x,result_y)
plt.show()

#energy
energy = np.append(energy, residual(result_y, noise_y))
```



5 the effect of the degree of polynomial

```
In [7]: degree = [2,3,4,5]
        plt.plot(degree, energy, 'ro')

        plt.xlabel('degree')
        plt.ylabel('energy')
        plt.show()
```


2000
1750
1500
1250
1000
750
500
250
energy
2.0
2.5
3.0
3.5
4.0
4.5
5.0
degree