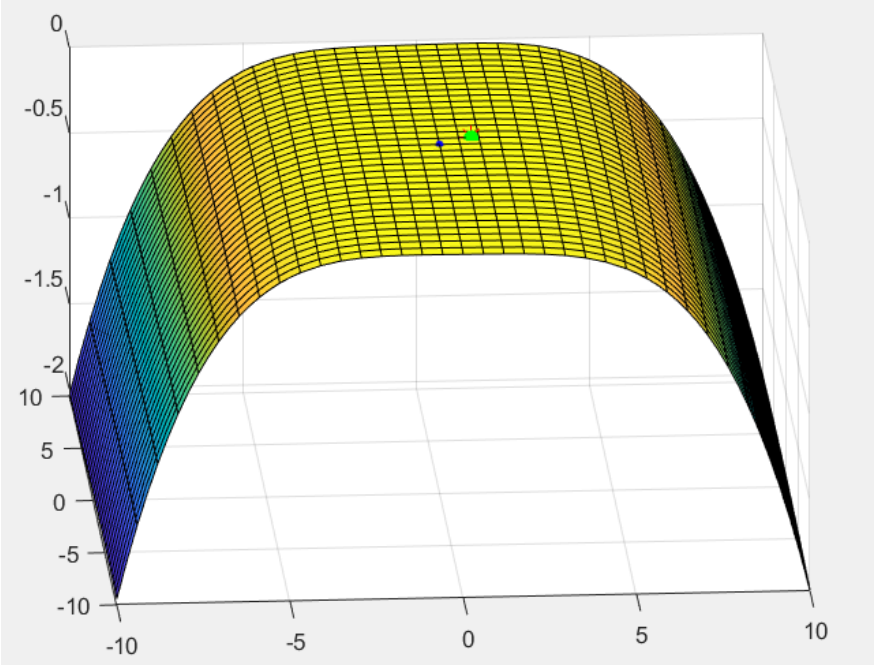
Q1.

In this question, I compute steepest ascent to find maximum value. I also limit the searching range of golden section search between [-10,10]. To run my program, just run the script "Q1_SteepAscent". Please follow my comments if you want to try different f(x,y).

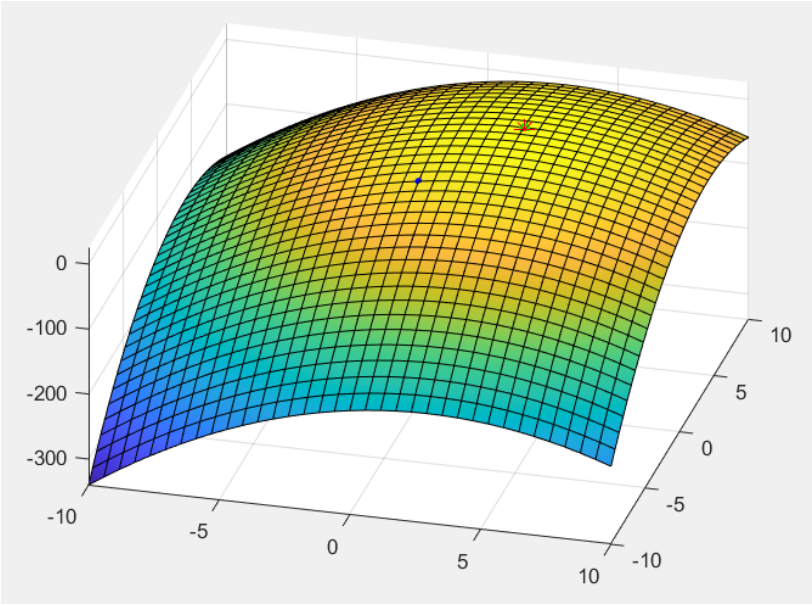
(1) For f(x,y) = $4x^2 + 2y + x^2 + 2xy - 3y^2 - 2x^4$, the result surface plot is



where blue point is start point, green points are intermediate steps, and '*' point is the end point. The optimum point is:

```
>> Q1
Optimum point is at (0.967574, 0.655848)
Maximum value is 4.344006
```

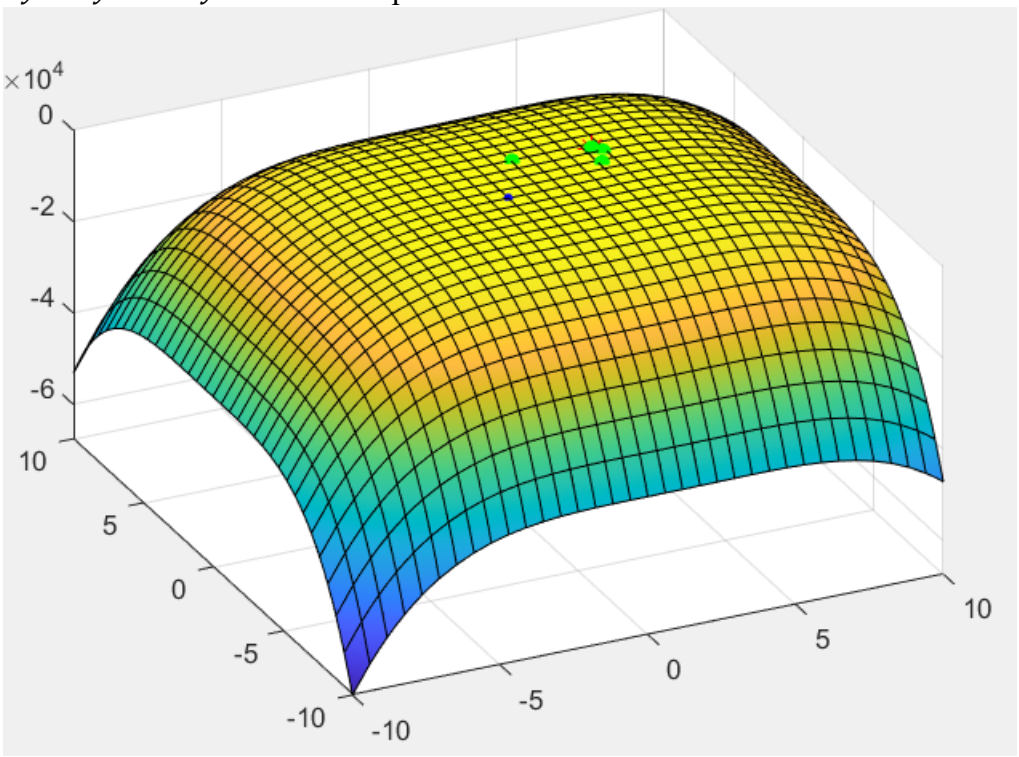
(2) The first function I designed is $f(x,y) = -x^2 + 6x - y^2 + 8y$. The surface plot is



The optimum point is:

```
>> Q1
Optimum point is at (3.000000, 4.000000)
Maximum value is 25.000000
```

(3) The first function I designed is $f(x,y) = -2x^4 + 10x^3 - x^2 + 10x - 3y^4 + 7y^3 - y^2 + 19y$. The surface plot is



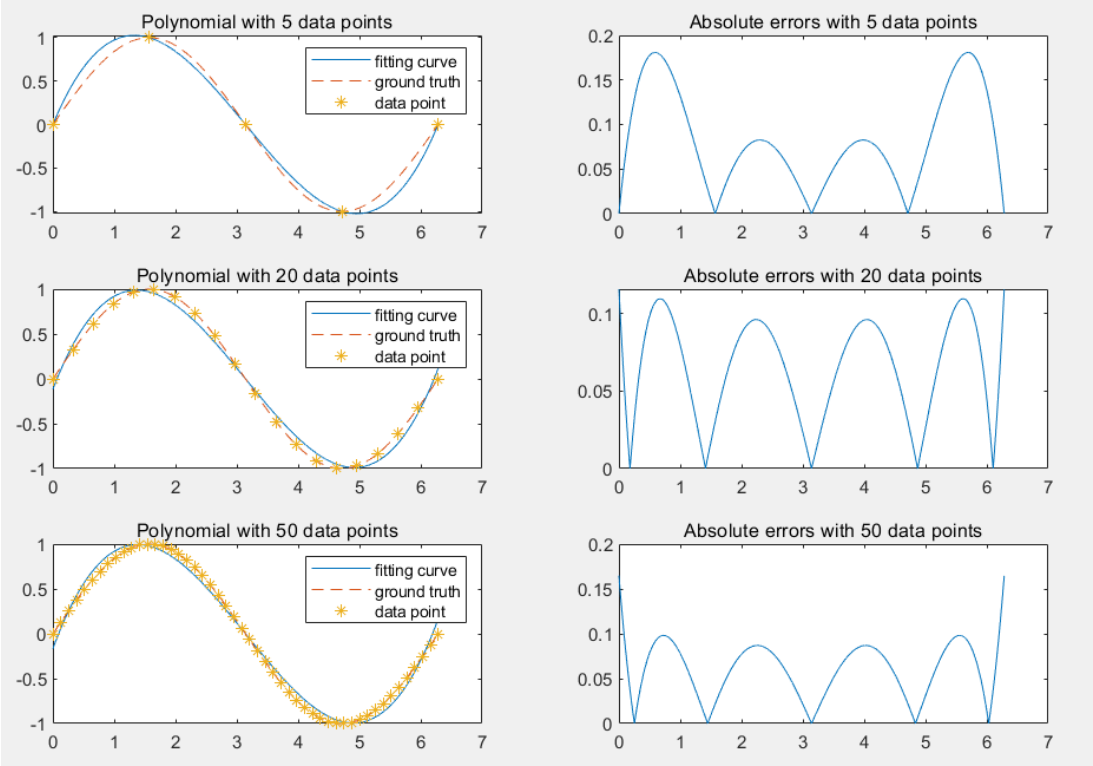
The optimum point is:

```
>> Q1
Optimum point is at (3.771594, 2.046578)
Maximum value is 197.371272
```

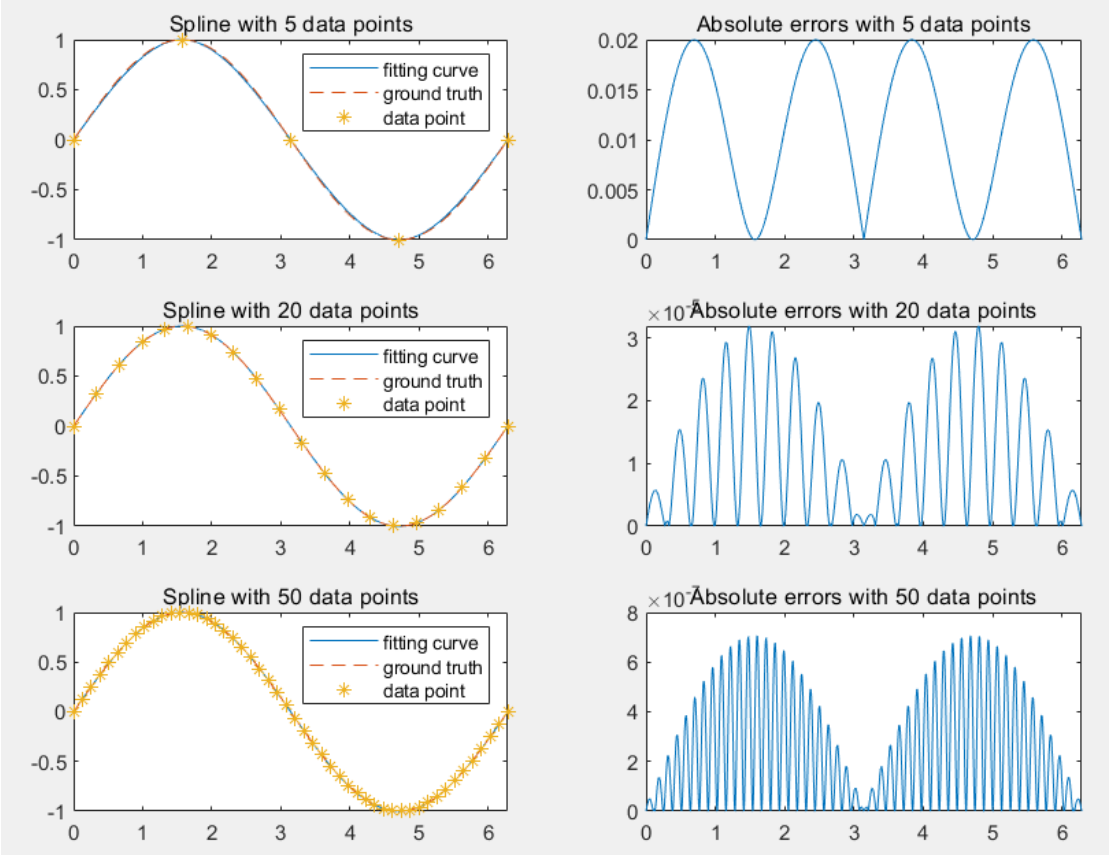
Q2

(1) To run my code for Q2(1), just run the script "Q2_1". It will generate 3 figures directly. The 3 required models are implemented in PolyFit.m, Spline.m, Lagrange.m

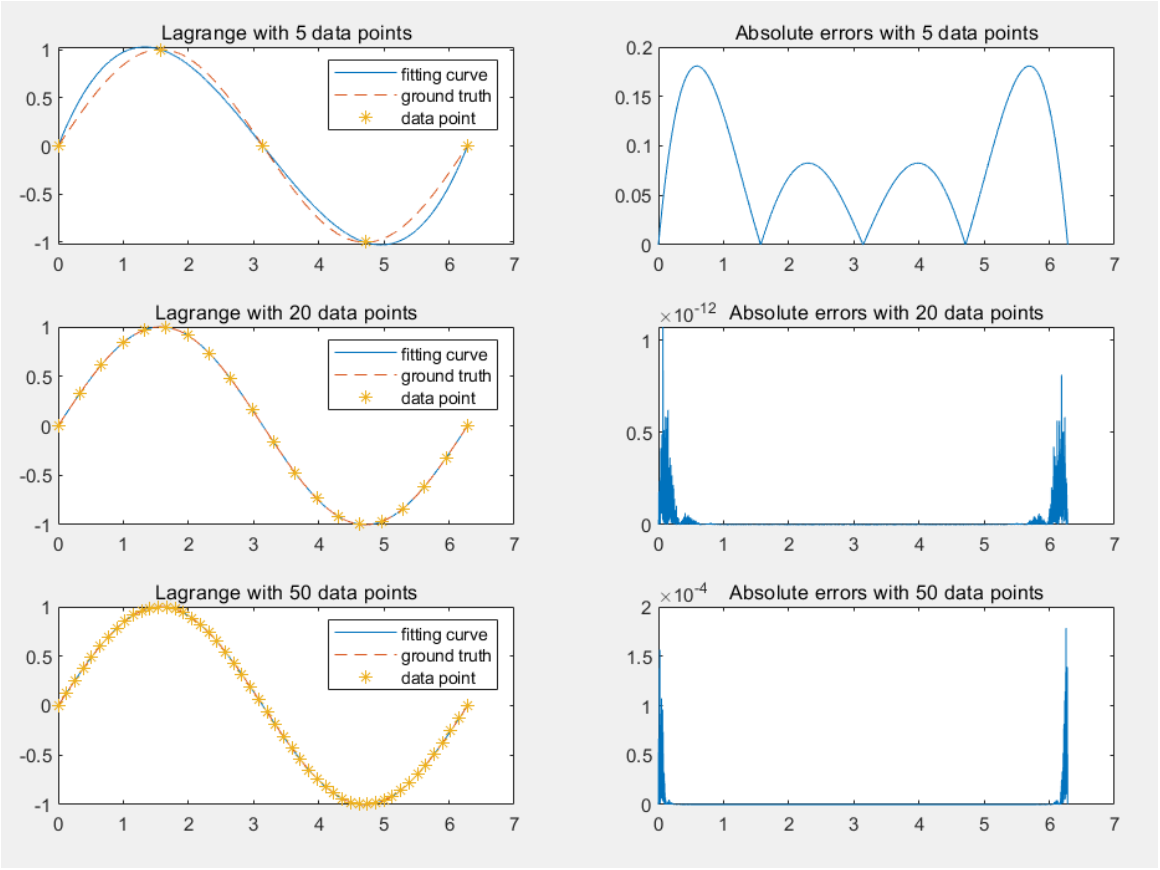
a. The polynomial fitting result is:



b. The result of cubic splines interpolation is:



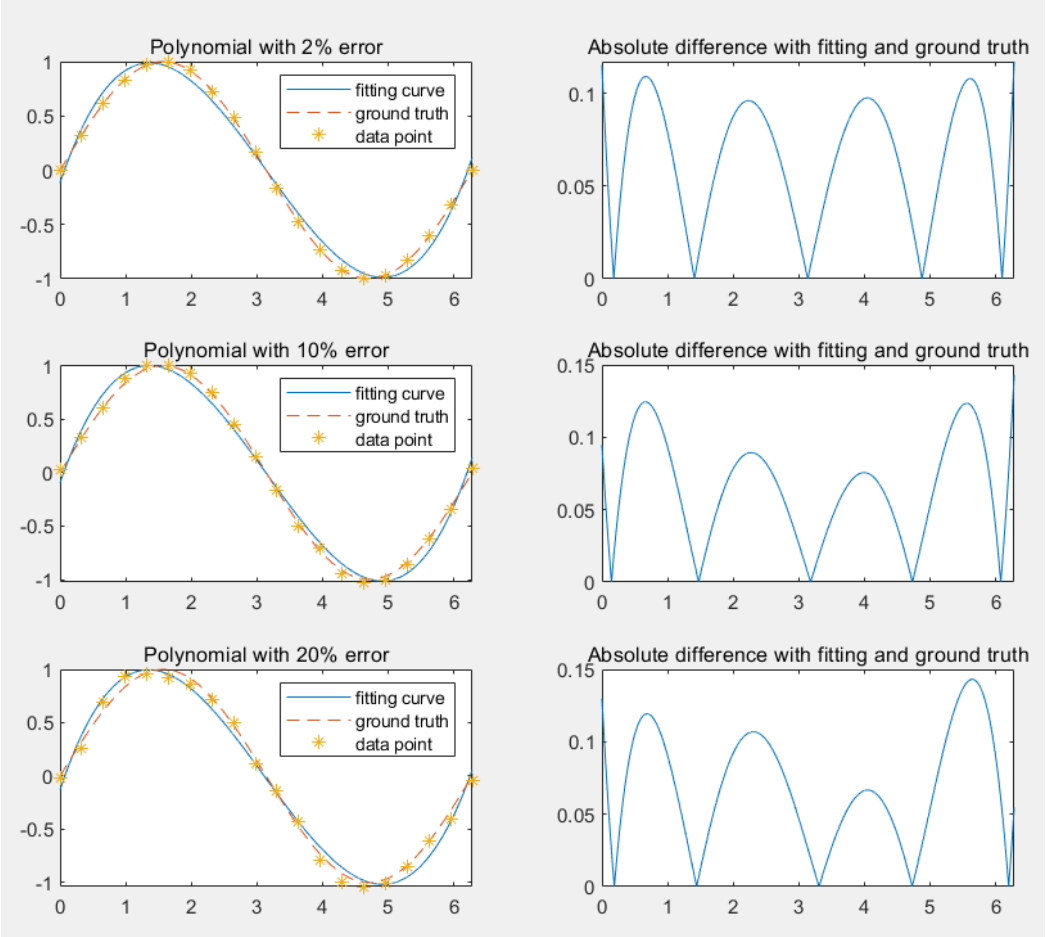
c. The result of Lagrange interpolation is:



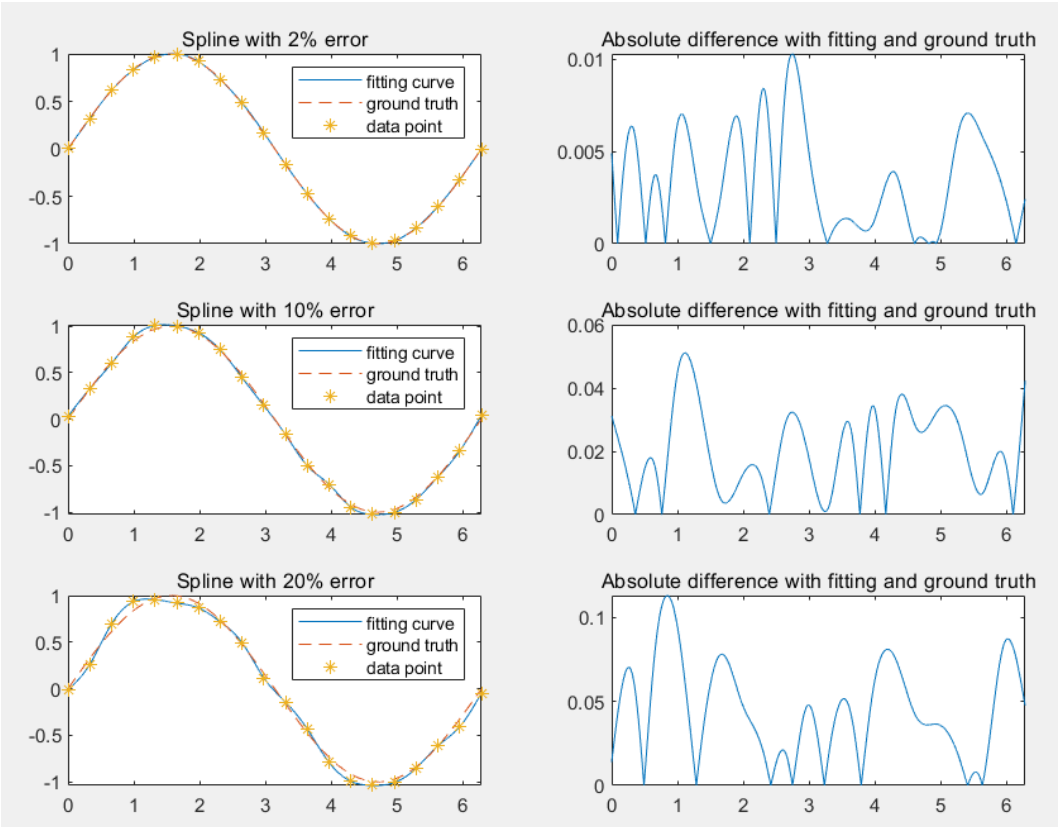
From the above figures, a direct result is that increasing the amount of data points can significantly improve the performance of fitting, no matter which method we choose.
Besides, Lagrange interpolation could be viewed as a general form of polynomial fitting of power 4. When the amount of data points is 5, these two methods are actually equivalent. However, with the increasement on the amount of data points, Lagrange interpolation will have more variables (higher power), thus it has better performance than polynomial fitting with a fixed power of 4 on large dataset. But this improvement comes at a cost. It will become "overfitting" – concretely, if the dataset is perfect, then the fitting result should be perfect. But if the dataset is not so good, then the curve will try to go through every point and miss the real shape. I think in next problem I can check whether my suspension is correct or not.
Cubic spline is much more sensitive to the amount of data. This method uses boundary conditions of segments as constraints to solve unknown variables. Therefore, as you can see, when it approaches to both ends of a segment, the error becomes smaller, and when it comes to the middle of a segment, the error gets larger. However, this method has the best performance when the amount of data is small.

(2) To run my code for Q2(2), just run the script "Q2_2".

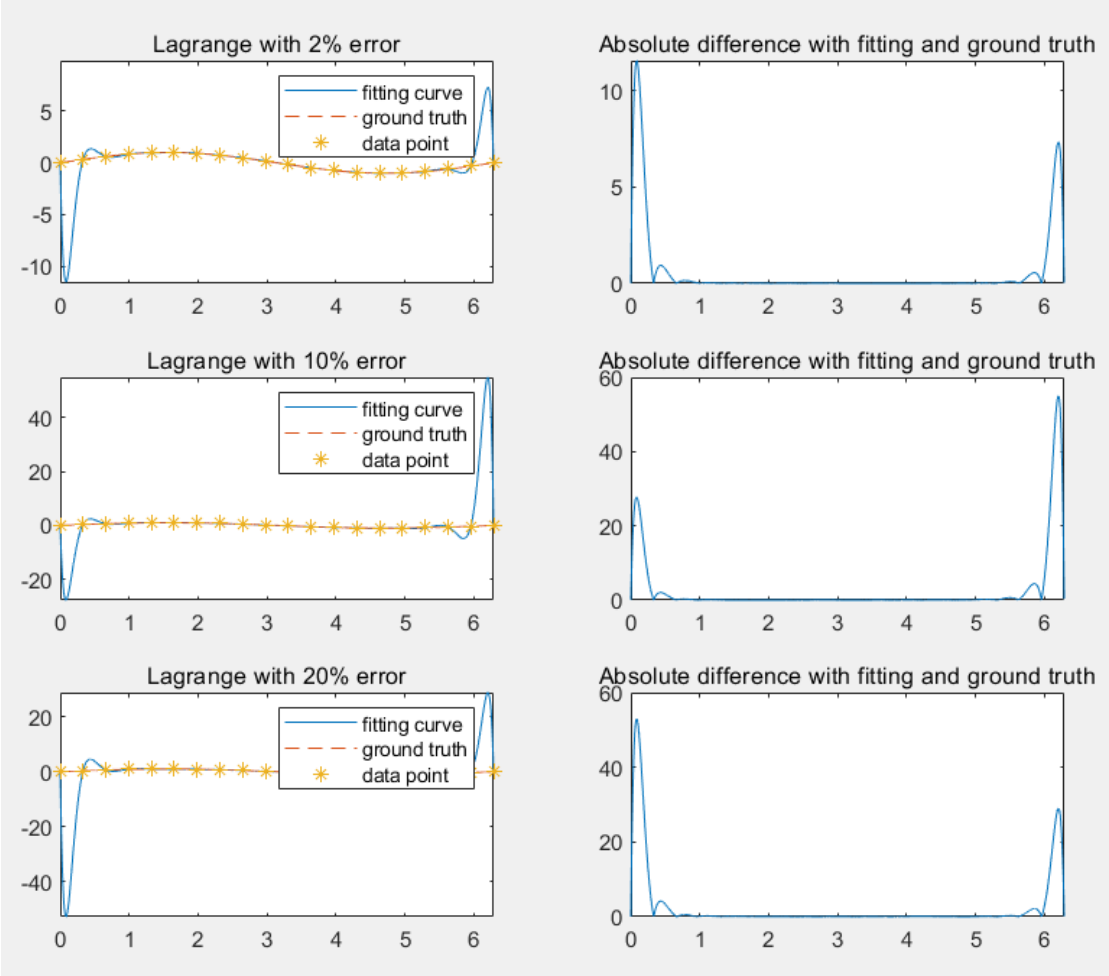
a. For polynomial fitting:



b. For cubic splines:



c. For Lagrange interpolation:



From these figures, we can observe that polynomial fitting is the most stable method. Its difference between ground truth only has very slight changes as errors increasing. And as I proposed in last question, Lagrange interpolation is affected severely by errors. Cubic spline is also affected by errors, but the difference is still acceptable.

Also, combining Q2(1) and Q2(2), we can find out the reason of "overfitting". For Lagrange and cubic spline, there are n variables corresponding to n data points. Since there are a unique solution to these unknown variables, the result curve will go through all data points. But it ignores the actual shape of the ground truth. For polynomial fitting with power of 4, the number of variables is less, which means the model has more data to calibrate its estimation, which can help the fitting curve to find the shape of the original curve.