Assignment 1
Yifei Gu
260906596

Question 1

a)

Taylor expansion:

$$f(x+\delta) = f(x) + \delta f'(x) + \cdots + \frac{\delta^{(n)}}{(n)!} f^{(n)}(x)$$

For the 4 given points, we have:

$$\begin{cases} f(x+\delta) = f(x) + \delta f'(x) + \frac{\delta^2}{2} f''(x) + \frac{\delta^3}{6} f'''(x) + \frac{\delta^4}{24} f''''(x) + \frac{\delta^5}{120} f'''''(x) + \cdots \\ f(x-\delta) = f(x) - \delta f'(x) + \frac{\delta^2}{2} f''(x) - \frac{\delta^3}{6} f'''(x) + \frac{\delta^4}{24} f''''(x) - \frac{\delta^5}{120} f'''''(x) + \cdots \end{cases}$$

$$f(x+\delta) - f(x-\delta) = 2\delta f'(x) + \frac{\delta^3}{3} f'''(x) + \frac{\delta^5}{60} f'''''(x) + \cdots$$

$$\begin{cases} f(x+2\delta) = f(x) + 2\delta f'(x) + 2\delta^2 f''(x) + \frac{4\delta^3}{3} f'''(x) + \frac{2\delta^4}{3} f''''(x) + \frac{4\delta^5}{15} f'''''(x) + \cdots \\ f(x+2\delta) = f(x) - 2\delta f'(x) + 2\delta^2 f''(x) - \frac{4\delta^3}{3} f'''(x) + \frac{2\delta^4}{3} f''''(x) - \frac{4\delta^5}{15} f'''''(x) + \cdots \end{cases}$$

$$f(x+2\delta) - f(x-2\delta) = 4\delta f'(x) + \frac{8\delta^3}{3} f'''(x) + \frac{8\delta^5}{15} f'''''(x) + \cdots$$

$$f'(x) = \frac{8[f(x+\delta) - f(x-\delta)] - [f(x+2\delta) - f(x-2\delta)]}{12\delta} + \frac{\delta^4}{30} f'''''(x) + \cdots$$

b)

In class, we went through the case that given 2 points. Now with 4 points, we can Taylor expend the function till fourth order and cancel them out, leaving the fifth order Taylor expansion as a leading error, which will against with roundoff error (marked with red).

$$\frac{\partial\, Error}{\partial \delta} = \frac{\delta^3 f'''''}{30} - \frac{g\varepsilon f}{\delta^2} = 0$$

$$\delta^5 = \frac{30 g\varepsilon f}{f'''''}$$

$$\delta \sim \sqrt[5]{\frac{g\varepsilon f}{f'''''}}$$

In a context of double precision, the floating-point precision goes as small as $\varepsilon = 10^{-16}$, so:

For **f(x) = exp(x):**

$$\delta \sim \sqrt[5]{10^{-16}} = 10^{-3.2}$$

For **f(x) = exp(0.01x):**

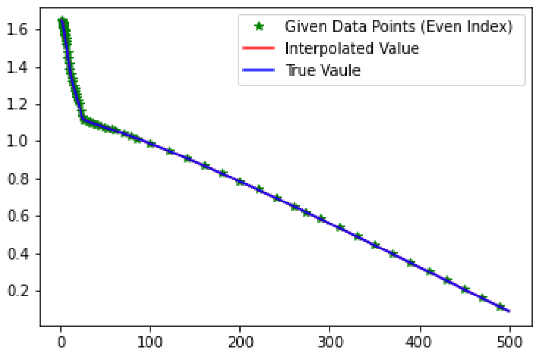
$$\delta \sim \sqrt[5]{10^{-6}} = 10^{-1.2}$$

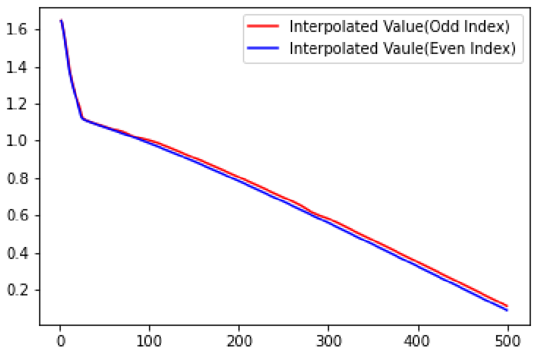
A screenshot of python result:

```
The result for f(x)=exp(x) are :
-16.0 [-1.11022302] [3.82850485]
-15.46551724137931 [3.45904486] [0.74076303]
-14.931034482758621 [2.62060491] [0.09767692]
-14.39655172413793 [2.71136527] [0.00691655]
-13.862068965517242 [2.73416036] [0.01587853]
-13.327586206896552 [2.71924861] [0.00096679]
-12.793103448275861 [2.71788013] [0.0004017]
-12.258620689655173 [2.71831852] [3.668698e-05]
-11.724137931034484 [2.71819742] [8.44042723e-05]
-11.189655172413794 [2.71831792] [3.60892686e-05]
-10.655172413793103 [2.71827322] [8.6134442e-06]
-10.120689655172415 [2.71828085] [9.76026167e-07]
-9.586206896551724 [2.71828122] [6.11224207e-07]
-9.051724137931036 [2.71828156] [2.71443124e-07]
-8.517241379310345 [2.71828183] [2.77682943e-09]
-7.982758620689657 [2.71828183] [1.52608948e-09]
-7.448275862068966 [2.71828182] [5.87954174e-09]
-6.913793103448276 [2.71828183] [8.91235086e-10]
-6.379310344827587 [2.71828183] [2.25102159e-10]
-5.844827586206897 [2.71828183] [4.48534543e-11]
-5.310344827586208 [2.71828183] [3.28612693e-11]
-4.775862068965518 [2.71828183] [2.52553534e-12]
-4.241379310344829 [2.71828183] [1.29318778e-12]
-3.706896551724139 [2.71828183] [7.56728014e-13]
-3.1724137931034484 [2.71828183] [2.13162821e-14]
-2.63793103448276 [2.71828183] [2.59792188e-12]
-2.1034482758620694 [2.71828183] [3.49449802e-10]
-1.5689655172413808 [2.71828178] [4.80118909e-08]
-1.0344827586206904 [2.71827523] [6.60211813e-06]
-0.5 [2.71736488] [0.00091694]
```

```
The result for f(x)=exp(0.01x) are :
-16.0 [0.] [0.0101005]
-15.46551724137931 [0.37833303] [0.36823253]
-14.931034482758621 [0.11050744] [0.10040693]
-14.39655172413793 [0.03227816] [0.02217766]
-13.862068965517242 [0.00673439] [0.00336612]
-13.327586206896552 [0.0121957] [0.0020952]
-12.793103448275861 [0.01045692] [0.00035642]
-12.258620689655173 [0.01030429] [0.00020379]
-11.724137931034484 [0.01004895] [5.15495764e-05]
-11.189655172413794 [0.01008278] [1.77243256e-05]
-10.655172413793103 [0.01010328] [2.7779893e-06]
-10.120689655172415 [0.01010166] [1.15580927e-06]
-9.586206896551724 [0.01010005] [4.50170719e-07]
-9.051724137931036 [0.01010061] [1.10749516e-07]
-8.517241379310345 [0.01010047] [2.73022288e-08]
-7.982758620689657 [0.01010051] [6.55169742e-09]
-7.448275862068966 [0.0101005] [2.08939261e-09]
-6.913793103448276 [0.0101005] [7.32189302e-10]
-6.379310344827587 [0.0101005] [2.01198691e-10]
-5.844827586206897 [0.0101005] [5.64241206e-11]
-5.310344827586208 [0.0101005] [1.15566064e-12]
-4.775862068965518 [0.0101005] [3.63538019e-12]
-4.241379310344829 [0.0101005] [1.00846415e-13]
-3.706896551724139 [0.0101005] [2.46132975e-13]
-3.1724137931034484 [0.0101005] [4.69069228e-15]
-2.63793103448276 [0.0101005] [3.87363752e-15]
-2.1034482758620694 [0.0101005] [8.49841031e-15]
-1.5689655172413808 [0.0101005] [2.55004351e-16]
-1.0344827586206904 [0.0101005] [8.89913143e-16]
-0.5 [0.0101005] [3.36709827e-14]
```

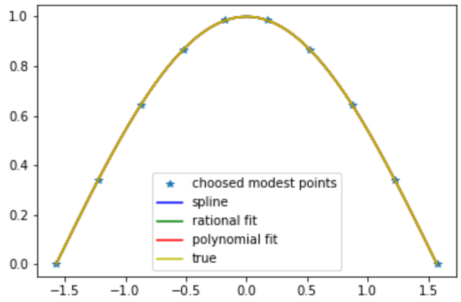
Question 2





For this question, no underlying function is given for the data points. Since we have "enough" many data points to be smooth, so I decide to use spline, which hopefully will not go crazy. What the codes does: 1) Extract the data points with even (or odd) indexes and interpolate all the data points. 2) Calculate the RMS error between interpolated data points and all the true(given) data points. My error of interpolation with **even** data point is **0.00017434301329736406**. My error interpolation with **odd** data point is **0.008358750905287586**.

Question 3

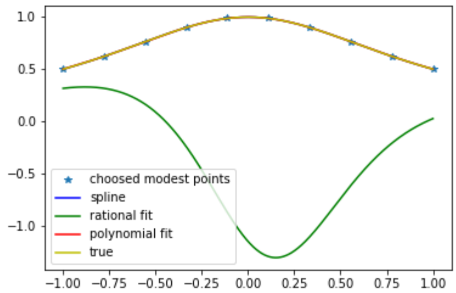


For cos(x) between -pi/2 and pi/2:

The error of **polynomial** fit is 8.827215512648578e-05

The error of **spline fit** is 5.477762871543277e-05

The error of **rational fit** is 1.2194256171766454e-07



For Lorentzian function between -1 and 1:

The error of **polynomial fit** is 0.00021996928998468926

The error of **spline fit** is 0.00010663009713533924

The error of **rational fit** is 0.6945586188799094