Q1

a)

<u>First we look at $x-\delta$ & $x+\delta$</u>

$$f(x+\delta)-f(x-\delta) =$$

$$\left(\cancel{f(x)} + f'(x)\delta + \cancel{\frac{f''(x)\delta^2}{2!}} + \frac{f'''(x)\delta^3}{3!} + \cancel{\frac{f^{(4)}(x)\delta^4}{4!}} + \frac{f^{(5)}(x)\delta^5}{5!} + O(\delta^6)\right)$$

$$-\left(\cancel{f(x)} - f'(x)\delta + \cancel{\frac{f''(x)\delta^2}{2!}} - \frac{f'''(x)\delta^3}{3!} + \cancel{\frac{f^{(4)}(x)\delta^4}{4!}} - \frac{f^{(5)}(x)\delta^5}{5!} + O(\delta^6)\right)$$

$$= 2\delta f'(x) + \frac{f'''(x)\delta^3}{3} + + \frac{2f^{(5)}(x)\delta^5}{5!} + O(\delta^6)$$

Similarly for $\pm 2\delta$

$$f(x+2\delta) + f(x-2\delta) =$$

$$\left(\cancel{f'(x)} + f'(x)2\delta + \cancel{\frac{f''(2\delta)^2}{2!}} + \frac{f'''(x)(2\delta)^3}{3!} + \cancel{\frac{f^{(4)}(x)(2\delta)^4}{4!}} + \frac{f^{(5)}(x)(2\delta)^5}{5!} + O(\delta^6)\right)$$

$$-\left(\cancel{f'(x)} - f'(x)2\delta + \cancel{\frac{f''(2\delta)^2}{2!}} - \frac{f'''(x)(2\delta)^3}{3!} + \cancel{\frac{f^{(4)}(x)(2\delta)^4}{4!}} - \frac{f^{(5)}(x)(2\delta)^5}{5!} + O(\delta^6)\right)$$

$$= 4\delta f'(x) + \frac{8}{3}\delta f'''(x) + \frac{64}{5!}\delta f^{(5)}(x)$$

We want to get rid of the $f^{(3)}$ term to only have a $f'(x)$+ error on the RHS. Thus, multiply the first eqn by 8

$8(*)-(**)$:

$$8\left(f(x+\delta)-f(x-\delta)\right)-\left(f(x+2\delta)+f(x-2\delta)\right)$$

$$= 12\delta f'(x) - \frac{2}{5}\delta^5 f^{(5)}(x) + O(\delta^6)$$

$$\Rightarrow \boxed{f'(x) = \frac{f(x-2\delta) - 8f(x-\delta) + 8f(x+\delta) - f(x+2\delta)}{12\delta} - \frac{1}{30}\delta^4 f^{(5)}(x)}$$

b)

$$|f'(x)| \leq \left|\frac{f(x-2\delta)(1+\epsilon_1) - 8f(x-\delta)(1+\epsilon_2) + 8f(x+\delta)(1+\epsilon_3) - f(x+2\delta)(1+\epsilon_4)}{12\delta}\right| + \left|\frac{\delta^4 f^5(x)}{30}\right|$$

$$\approx \frac{|f(x)(\epsilon_1 - 8\epsilon_2 + 8\epsilon_3 - \epsilon_4)|}{12\delta} + \frac{|\delta^4 f^5(x)|}{30}$$

let $\epsilon = \max(\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4)$

the worst case looks like

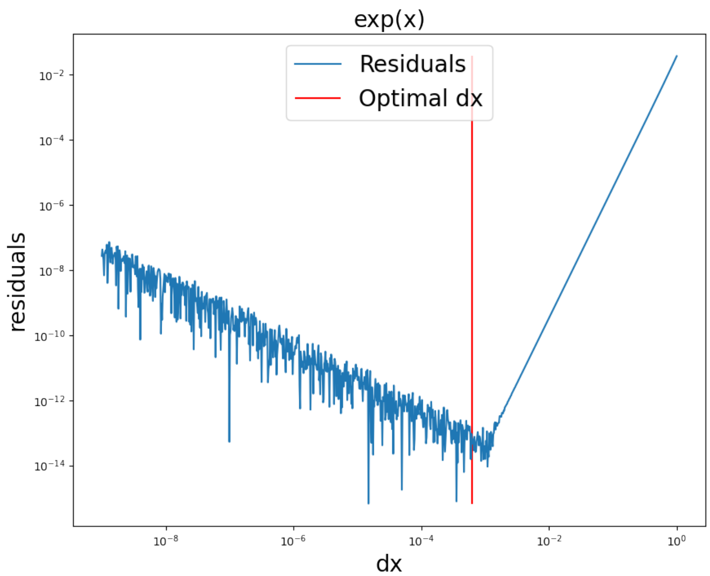
$$\frac{f(x)(\epsilon + 8\epsilon + 8\epsilon + \epsilon)}{12\delta} + \frac{\delta^4 f^{(5)}(x)}{30}$$

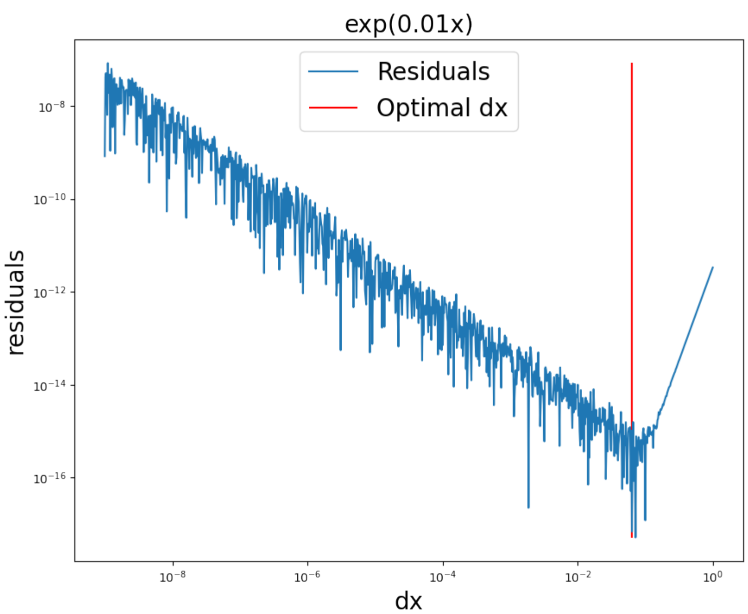
$$= \boxed{\frac{3\epsilon f(x)}{2\delta} + \frac{\delta^4 f^{(5)}(x)}{30}}$$

Optimizing

$$\frac{3\epsilon f(x)}{2\delta^2} + \frac{\delta^3 f^{(5)}(x)}{30} = 0$$

$$\Longrightarrow \quad \boxed{\delta \approx \sqrt[5]{\frac{\epsilon f(x)}{f^{(5)}(x)}}}$$





Question #2

$$f(x+\delta)-f(x-\delta)=$$

$$\left(\cancel{f(x)}+f'(x)\delta+\cancel{\frac{f''(x)\delta^2}{2!}}+\frac{f'''(x)\delta^3}{3!}\right)-\left(\cancel{f(x)}-f'(x)\delta+\cancel{\frac{f''(x)\delta^2}{2!}}-\frac{f'''(x)\delta^3}{3!}\right)$$

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

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

$$|f'(x)|\leq\left|\frac{f(x+\delta)(1+\epsilon_1)-f(x-\delta)(1+\epsilon_2)}{2\delta}\right|-\left|\frac{f'''(x)\delta^2}{3}\right|$$

$$\approx\frac{|f(x)(\epsilon_1-\epsilon_2)|}{2\delta}-\frac{f'''(x)\delta^2}{3}$$

$$\leq\frac{\epsilon f(x)}{\delta}+\frac{f'''\delta^2}{3}$$

Optimizing

$$\frac{\epsilon}{\delta^2}+\frac{f'''\delta}{3}=0$$

$$\boxed{\delta=\sqrt[3]{\frac{\epsilon f(x)}{f'''(x)}}}$$

Now we need to express f''' in terms of f

from # 1 we know that

$$(f(x+h) - f(x-h)) = 2hf'(x) + \frac{f'''(x)h^3}{3}$$

$$(f(x+2h) - f(x-2h)) = 4hf'(x) + \frac{8h^3}{3}f'''(x)$$

$$\Rightarrow f(x+2h) - 2f(x+h) + 2f(x-h) - f(x-2h) = h^3 f'''(x)$$

So subbing our new expression for $f'''(x)$ we get

$$\boxed{\delta = h\left[\frac{\epsilon f(x)}{f(x+2h) - 2f(x+h) + 2f(x-h) - f(x-2h)}\right]^{1/3}}$$

So you can get an expression for delta only with f(x). You end up with +/- h from the third derivative. What I did is that I let h = 0.01 hopping at leas to get the magnitude of the third derivative.

The first picture is the output for my ndiff function for sin(x0) with x0=1

If you set full=False it returns the derivative

If you set full=True it returns the derivative and the estimated error, in that order

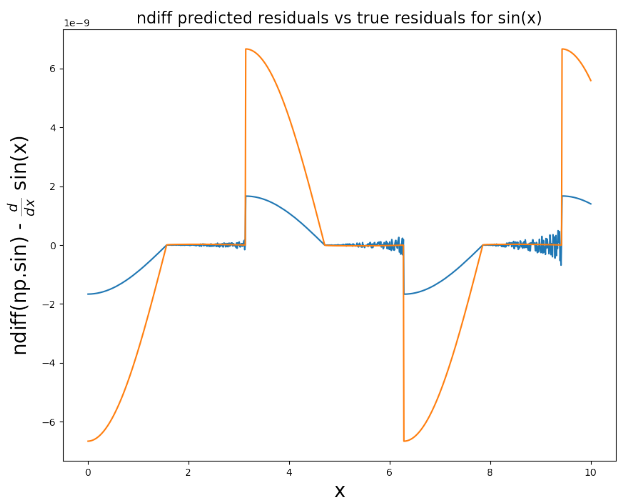
```
ndiff(np.sin, 1, full=False)
```
[44] ✓ 0.3s

```
/tmp/ipykernel_156612/660926180.py:18: RuntimeWarning: invalid value encountered in double_scala
  dx = ((1e-16*y0)/d3y_d3x)**(1/3) #optimal dx

0.5403023049677103
```

```
ndiff(np.sin,1, full=True)
```
[46] ✓ 0.4s

```
/tmp/ipykernel_156612/660926180.py:18: RuntimeWarning: invalid value encountered in double_scala
  dx = ((1e-16*y0)/d3y_d3x)**(1/3) #optimal dx

(0.5403023049677103, -3.601083851834783e-09)
```

To show you how how good the estimation for the error is i plotted the estimated residual beside the true residuals. By "true residuals" I mean that I used the analytic derivative of sin(x) to compute them.



Question 3

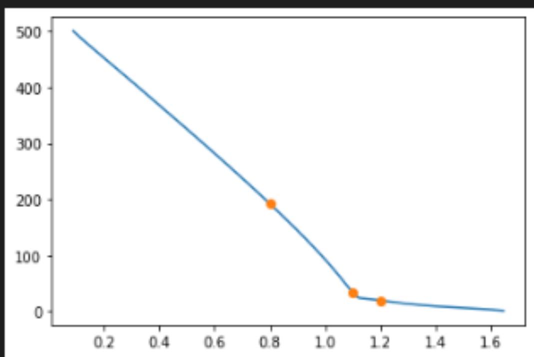
The following show the ouput of my lakeshore function. It can take a number or an array and it interporlates using a cubic spline.

In order to estimate the error, I fitted a line with the two nearest point of the voltage input. I think this a reasonable estimation because the curve is pretty well behave and straight.

```
lake = lakeshore([0.8,1.1,1.2],data)
plt.plot(data[1],data[0])
plt.errorbar( [0.8,1.1,1.2], [lake[0][0], lake[0][1], lake[0][2]] , [lake[1][0], lake[1][1], lake[1][2]], fmt="o")
```

[169] ✓ 0.1s

```
Temperature interpolation =  [192.44416255  33.38617731  19.85525937]
Estimated error = [0.00851531 0.00274331 0.00088792]

<ErrorbarContainer object of 3 artists>
```




```
#Her I am inputing only a number
lakeshore(0.8, data)
```

[178] ✓ 0.3s

```
Temperature interpolation =  [192.44416255]
Estimated error = [0.00851531]

(array([192.44416255]), array([0.00851531]))
```

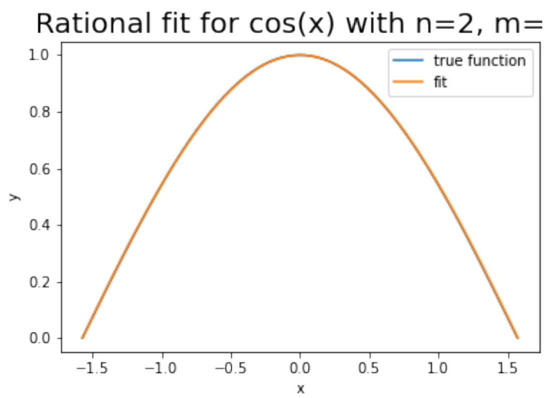
Question 4

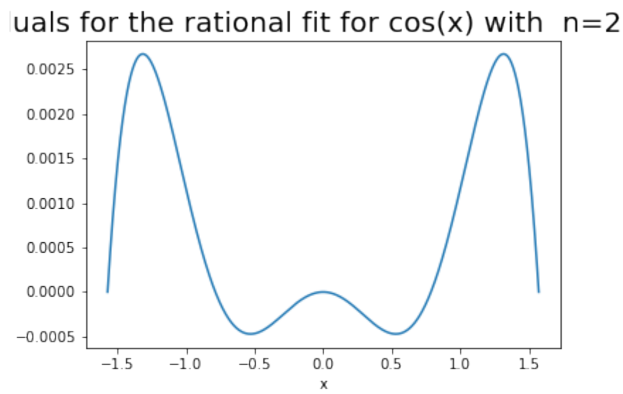
First I am going to show the results for the cos(x), all the plots are label with n = # and m = #.

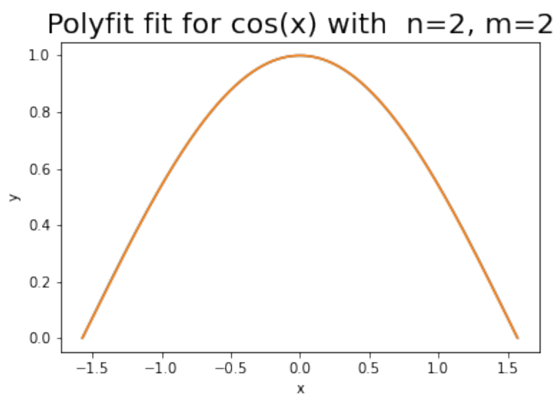
They are the order of the rational fit, but also serve to set the number of points for the others fits. So to make sure we compare apples with apples all the plots have the mention n = # and m=#.

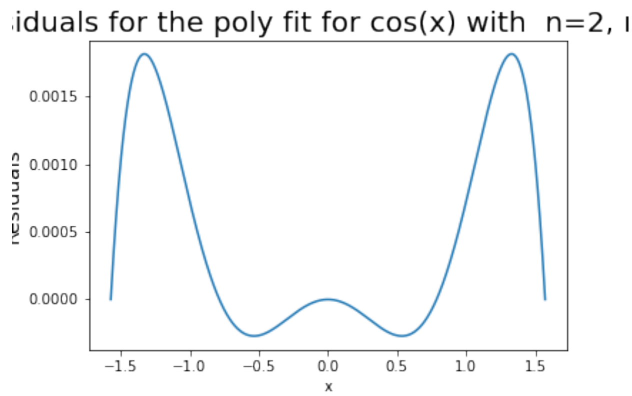
Fits

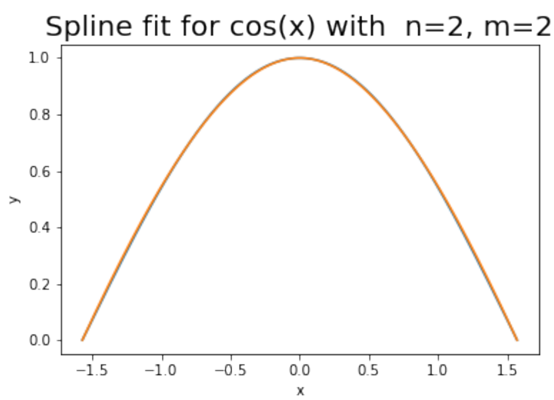
Residuals

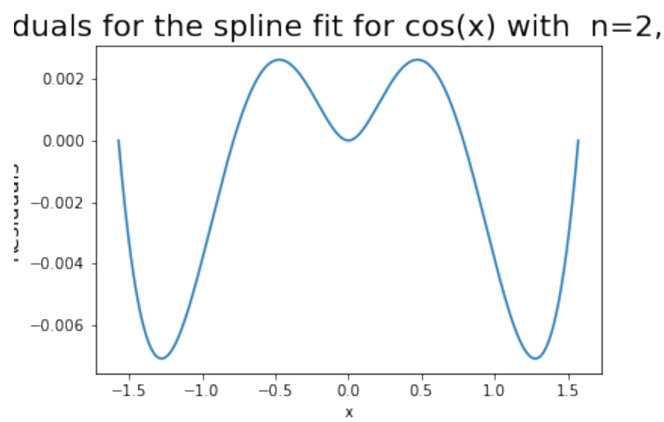


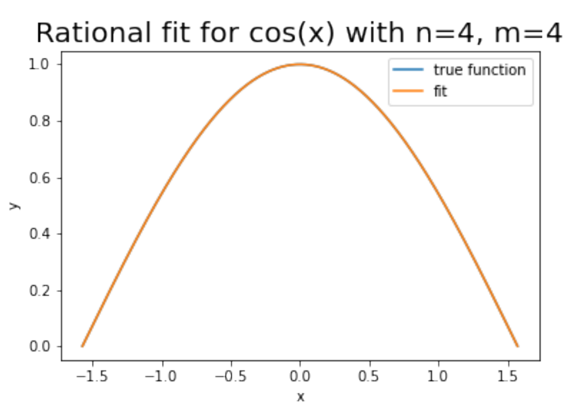


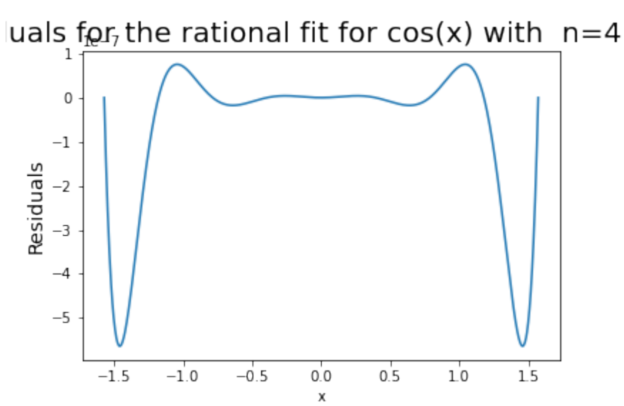


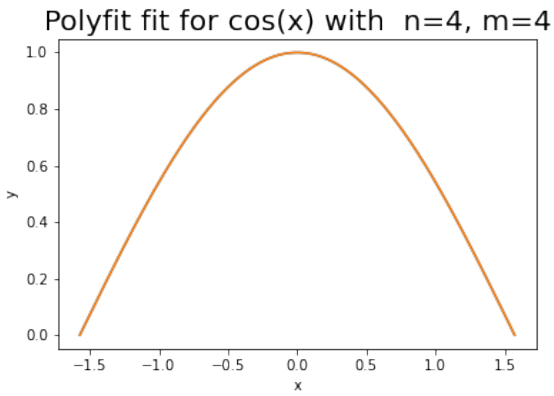


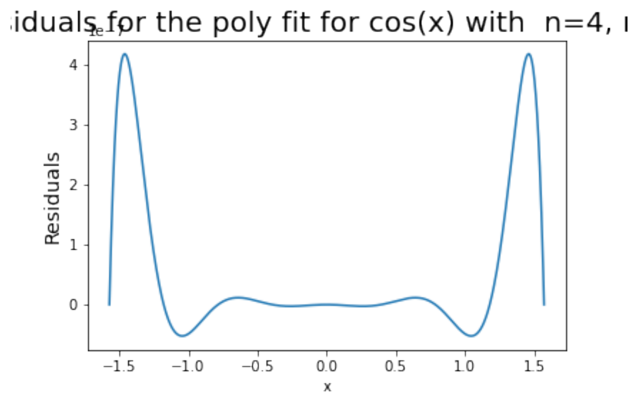


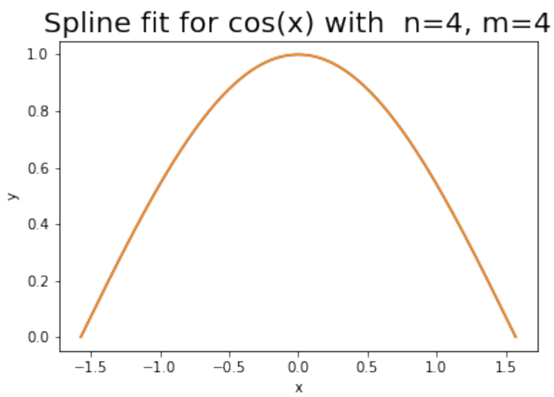


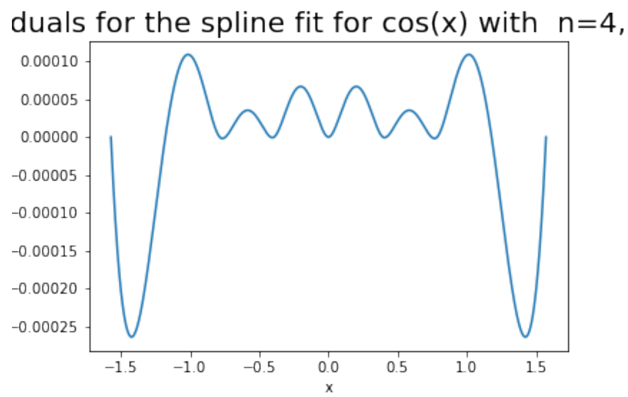




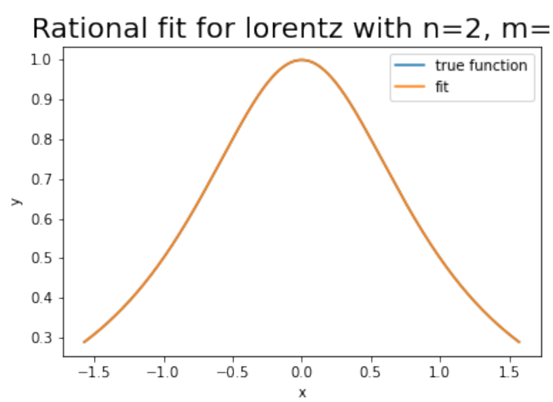


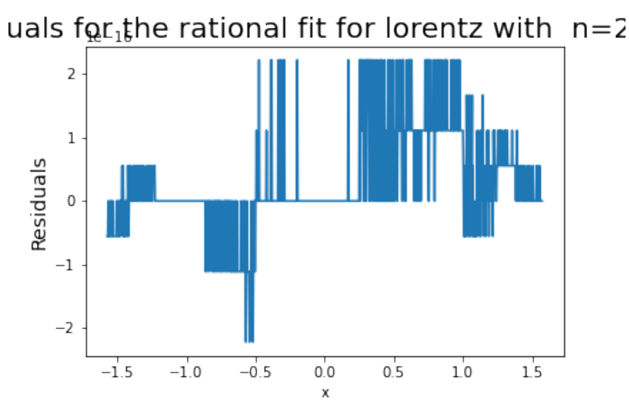


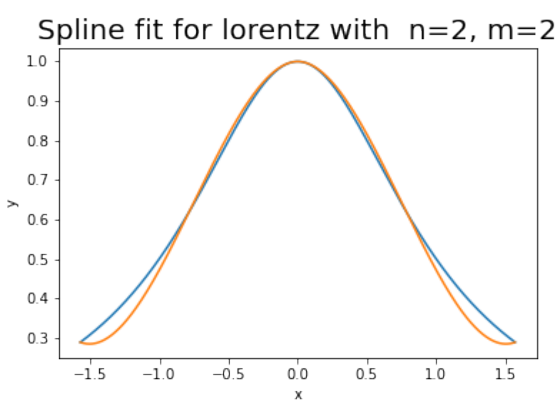


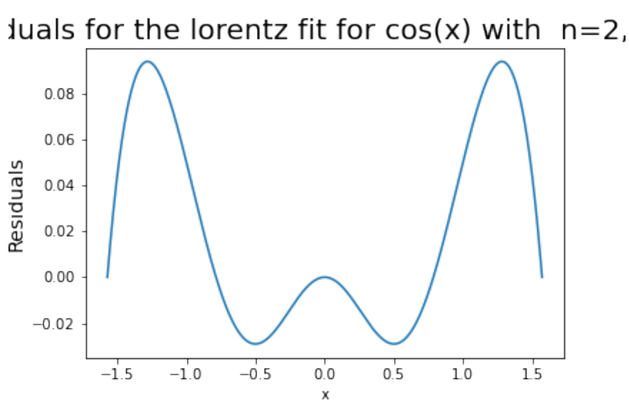


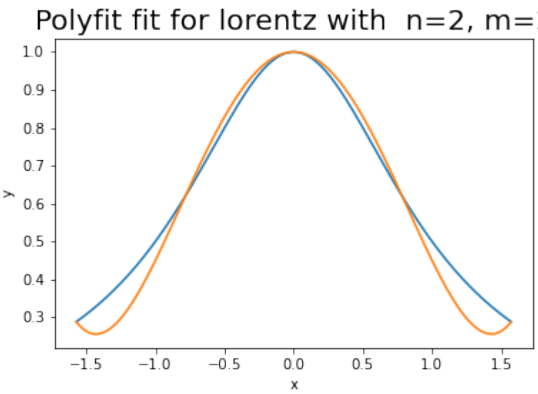
They all do pretty decent. The spline does the worst. Pretty tie between rational and T

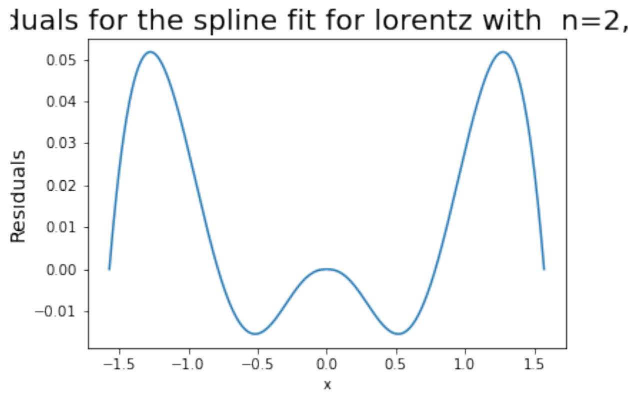


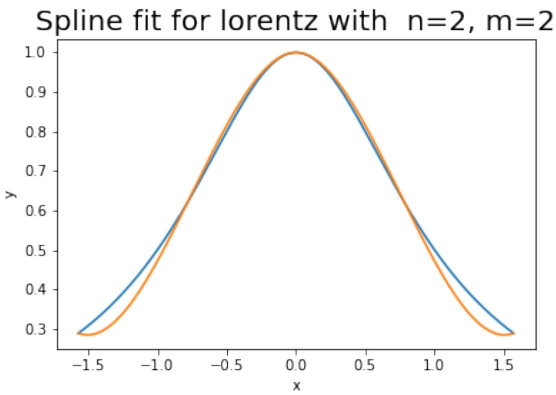


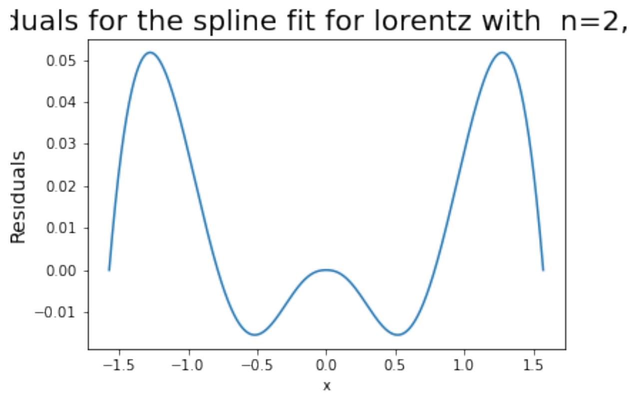


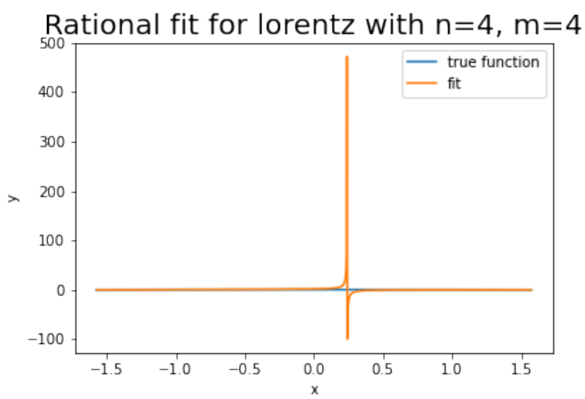


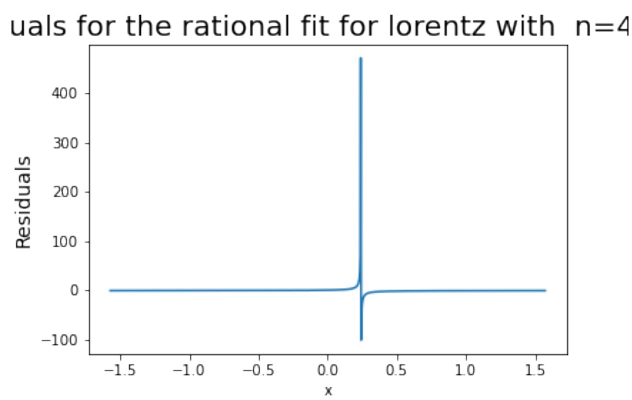

Polyfit fit for lorentz with n=2, m=2
y
x

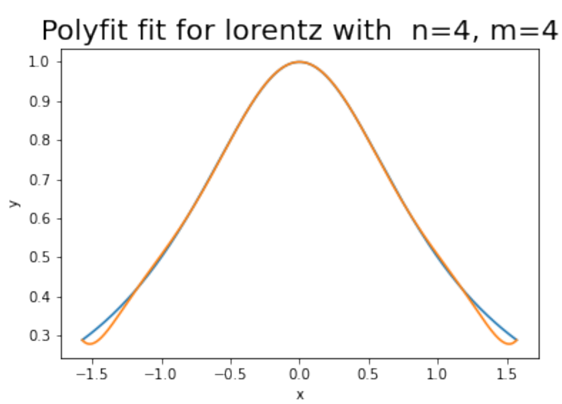

duals for the spline fit for lorentz with n=2,
Residuals
x

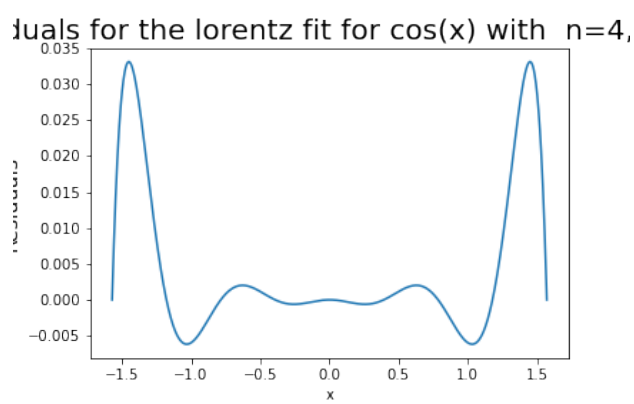

Spline fit for lorentz with n=2, m=2
y
x

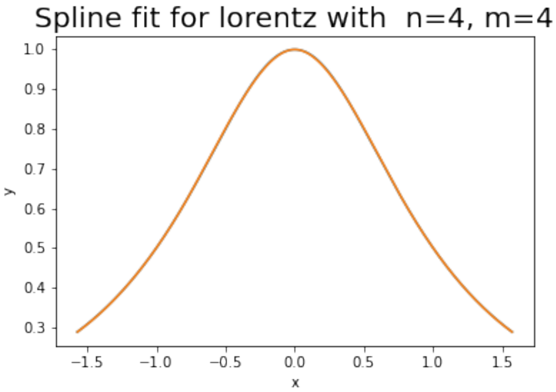

duals for the spline fit for lorentz with n=2,
Residuals
x

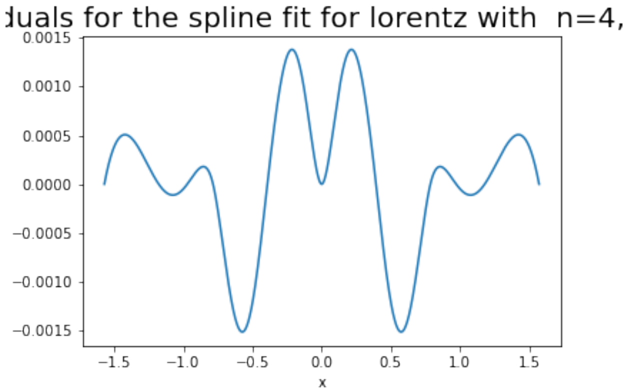

Rational fit for lorentz with n=4, m=4
true function
fit
y
x


uals for the rational fit for lorentz with n=4
Residuals
x


Polyfit fit for lorentz with n=4, m=4
y
x


duals for the lorentz fit for cos(x) with n=4,
x





We see that the rational fit does extremely well with the lorentz function. This was to be expected since the lorentz function is a rational function.

At higher order; however, the rational function fit breaks down. This is because we are trying to fit to many coefficients which lead to near zero eigen values. This cause trouble when we try to invert the matrix.

If you swap linalg.inv to linalg.pinv than the 1/0 from the matrix inversion get sent to 0. This fixes the problem and we get a good fit again.