Phys 512 Problem Set 1

JIAO Hao

PROBLEM 1

(a)

$$f(x+\delta) = f(x) + f'(x)\delta + \frac{1}{2}f''(x)\delta^2 + \frac{1}{6}f'''(x)\delta^3 + \frac{1}{24}f^{(4)}(x)\delta^4 + O(\delta^5)$$
$$f(x-\delta) = f(x) - f'(x)\delta + \frac{1}{2}f''(x)\delta^2 - \frac{1}{6}f'''(x)\delta^3 + \frac{1}{24}f^{(4)}(x)\delta^4 + O(\delta^5)$$
$$f(x+2\delta) = f(x) + 2f'(x)\delta + 2f''(x)\delta^2 + \frac{4}{3}f'''(x)\delta^3 + \frac{2}{3}f^{(4)}(x)\delta^4 + O(\delta^5)$$
$$f(x-2\delta) = f(x) - 2f'(x)\delta + 2f''(x)\delta^2 - \frac{4}{3}f'''(x)\delta^3 + \frac{2}{3}f^{(4)}(x)\delta^4 + O(\delta^5)$$

$$f(x+\delta) - f(x-\delta) = 2f'(x)\delta + \frac{1}{3}f'''(x)\delta^3 + O(\delta^5)$$
$$f(x+2\delta) - f(x-2\delta) = 4f'(x)\delta + \frac{8}{3}f'''(x)\delta^3 + O(\delta^5)$$

$$\Rightarrow \quad 8\times[f(x+\delta) - f(x-\delta)] - [f(x+2\delta) - f(x-2\delta)] = 12f'(x)\delta + O(\delta^5)$$

$$f'(x) \simeq \frac{8[f(x+\delta) - f(x-\delta)] - [f(x+2\delta) - f(x-2\delta)]}{12\delta}$$

(b)

$$f' = \text{result} + Af^{(5)}\delta^5 + \frac{\tilde{g}\epsilon f}{\delta}$$
$$\frac{\partial \epsilon_{error}}{\partial \delta} = 5Af^{(5)}\delta^4 - \frac{\tilde{g}\epsilon f}{\delta^2} = 0$$
$$\Rightarrow \quad \delta_{min} = \left(\frac{\tilde{g}\epsilon f}{5Af^{(5)}}\right)^{1/6} \sim (\epsilon f/f^{(5)})^{1/6}$$

Then, use the code 'JIAO Hao 1-1-b.py', we can compare the error of deviative for different δ.
For $f(x) = \exp(x)$, we have:
- For $x_0 = 0,\ \delta_{min} = 10^{-3}$, error=2.7e-14
- For $x_0 = 1,\ \delta_{min} = 10^{-4}$, error=1.2e-13
- For $x_0 = 2,\ \delta_{min} = 10^{-3}$, error=9.8e-13
- For $x_0 = 3,\ \delta_{min} = 10^{-3}$, error=2.4e-12

For $f(x) = \exp(0.01x)$, we have:
- For $x_0 = 0,\ \delta_{min} = 10^{-1}$, error=2.7e-16
- For $x_0 = 1,\ \delta_{min} = 10^{-1}$, error=1.5e-15
- For $x_0 = 10,\ \delta_{min} = 10^{-1}$, error=2.5e-16
- For $x_0 = 100,\ \delta_{min} = 10^{-2}$, error=1.1e-15
- For $x_0 = 1000,\ \delta_{min} = 10^{-1}$, error=1.1e-10

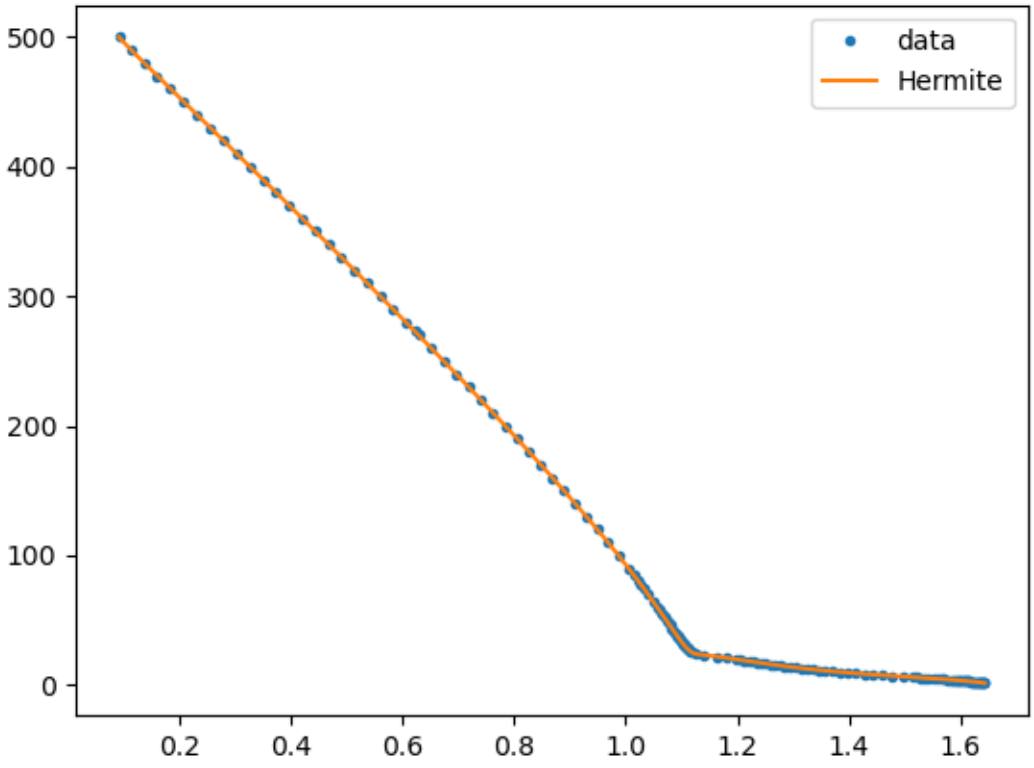

FIG. 1. The temperature-voltage curvature of Lakeshore 670 diodes, the horizontal axis is voltage in V and the vertical axis is temperature in K.

PROBLEM 2

Here we have (x,y,y') for every points, so the best interpolation is Hermite interpolation. Fig.1 shows the result of this integration.

And once we enter the voltage value in the range of $[0.091V, 1.644V]$, this code can show the corresponding temperature.

Since I use Hermite interpolation here, the estimate of error is approximate to

$$R(x) \sim \frac{1}{4!} f^{(4)}(x) \cdot (x - x_1)^2 \cdot (x - x_2)^2,$$

where the x_1 & x_2 are the nearest points around x.

PROBLEM 3

interpolation of cos(x) between $-\pi/2$ and $\pi/2$

First, I compare the result of these three interpolations of cos(x) on five points: $\{(-\pi/2, 0), (-\pi/4, \sqrt{2}), (0, 1), (\pi/4, \sqrt{2}), (\pi/2, 0)\}$ Because the number of points N and the order (n,m) of rational function interpolation have relation $N = m + n - 1$, I can only set $n = m = 3$, since $n = 2, m = 4$ have singular matrices when get parameters.(fig.2) Obviously, the rational function interpolation is most accurate and the linear polynomial is worst.

Then, the interpolation of points leads to very similar conclusion. Here I set $n = 3, m = 4$.(fig.3)

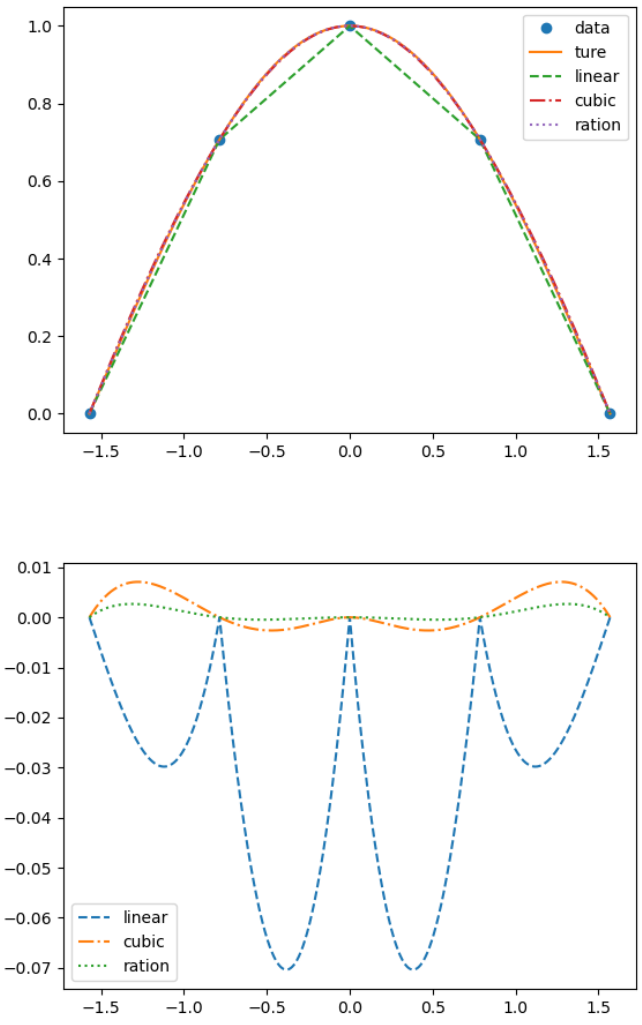

FIG. 2. Compare the accuracy of polynomial, cubic spline, and rational function interpolation of cos(x). Here for fairness each method use the same 5 points: $\{(-\pi/2, 0), (-\pi/4, \sqrt{2}), (0, 1), (\pi/4, \sqrt{2}), (\pi/2, 0)\}$. The top panel compares the results of these interpolation and the real function. The bottom panel shows the error of each interpolation

interpolation of Lorentzian between -1 and 1

Because the Lorentzian is also a rational function with $n = 0, m = 2$, I think the error of rational function interpolation should be very small. But it is imposible for us onlt take one point to do the interpolation. So I also use 5 points first to compare the polynomial, cubic spline, and rational function interpolation. Here also set $n = m = 3$. The result is the same as what I expect.(fig.4)

But things are different when I compare different order rational function interpolation: Here I compare several different choice of n and m: $(n, m) = \{(2, 3), (3, 3), (3, 4), (4, 5)\}$. I found the most accurate interpolation is $(n, m) = (2, 3)$ rather than higher ones. Especially for $(n, m) = (3, 4), (4, 5)$, there are very large errors, which are even larger than the real function. This is really strange.(fig.5)

switch from np.linalg.inv to np.linalg.pinv

Then, I use np.linalg.pinv instead of np.linalg.inv in solving the parameters of rational function and the diversions are successfully removed. And the errors of high order rational function interpolation are very small as well. Result

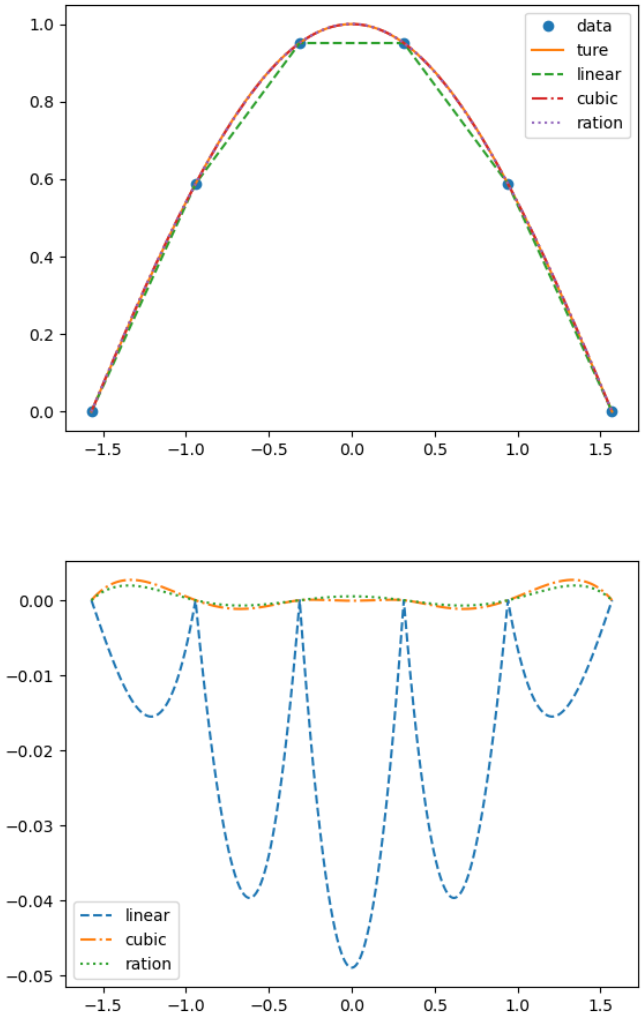

FIG. 3. Compare the accuracy of polynomial, cubic spline, and rational function interpolation of cos(x). Here for fairness each method use 6 points: $\{(-\pi/2, 0), (-3\pi/10, \cos(-3\pi/10)), (-\pi/10, \cos(-\pi/10)), (\pi/10, \cos(\pi/10)), (3\pi/10, \cos(3\pi/10)), (\pi/2, 0)\}$. The top panel compares the results of these interpolation and the real function. The bottom panel shows the error of each interpolation

is show in fig.6

Because np.linalg.pinv tries to deal with singular matrices and it also deal with the large error of high order rational function interpolation, I think these error should be caused by singular matrices.

Let us see p and q for different order rational functions to proof this. Here we only consider the order are the same with the 'pinv' case. The first one $(n, m) = (3, 3)$ can be nomally solved by 'np.linalg.inv', so the p and q of the two cases are same. But others are different: we can see from the cases using 'np.linalg.pinv', all p and q are very similar to constant, which means that the coefficients are very small. I think that is the reason why the interpolations using 'np.linalg.inv' have large error: np.linalg.inv will make mistakes when the results are too small. (fig 7)

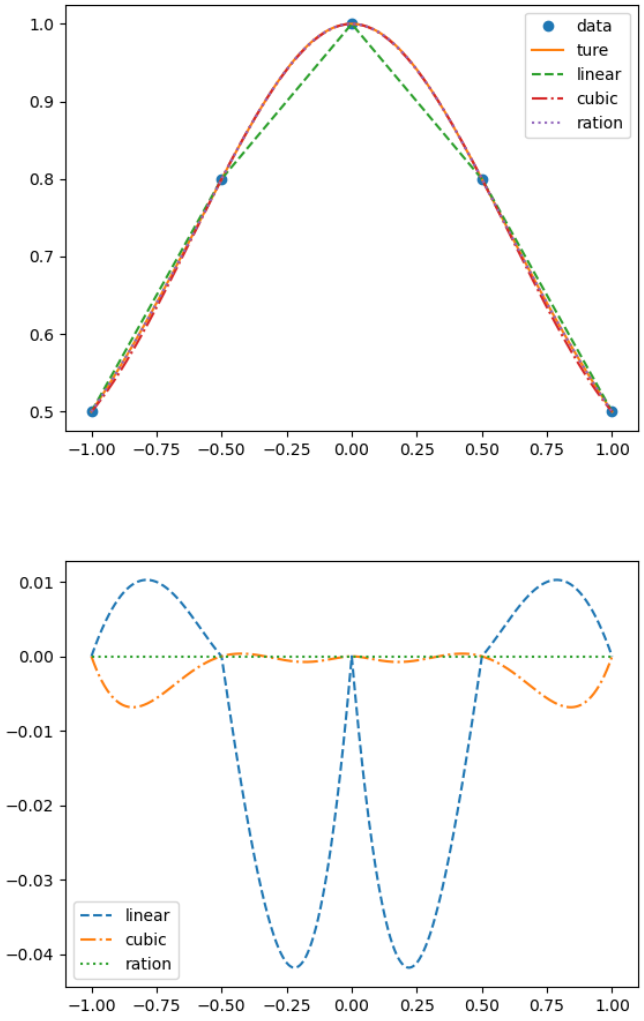

FIG. 4. Compare the accuracy of polynomial, cubic spline, and rational function interpolation of Lorentzian. Here for fairness each method use 5 points: $\{(-1,1/2),(-1/2,4/5),(0,1),(1/2,4/5),(1,1/2)\}$. The top panel compares the results of these interpolations and the real function. The bottom panel shows the error of each interpolation.

PROBLEM 4

The integral is

$$E(z) = \frac{1}{4\pi\epsilon_0}\int \frac{qR^2\sin\theta' d\theta' d\phi'}{(R^2+z^2-2Rz\cos\theta')^{3/2}}$$
$$= \frac{1}{4\pi\epsilon_0}\int \frac{2\pi qR^2\sin\theta' d\theta'}{(R^2+z^2-2Rz\cos\theta')^{3/2}}$$

For simplification, I set $R = q = 1$ and ignore the factor: $\frac{1}{4\pi\epsilon_0}$, And the electric field is show in fig **??**

In thos code, I found that my integrator have problems with the singularity of the integrand eventhough this singularity will not affect the real integral, so I have to set the value at that point equal to 0. On the other hand, quad do not care this singularity.

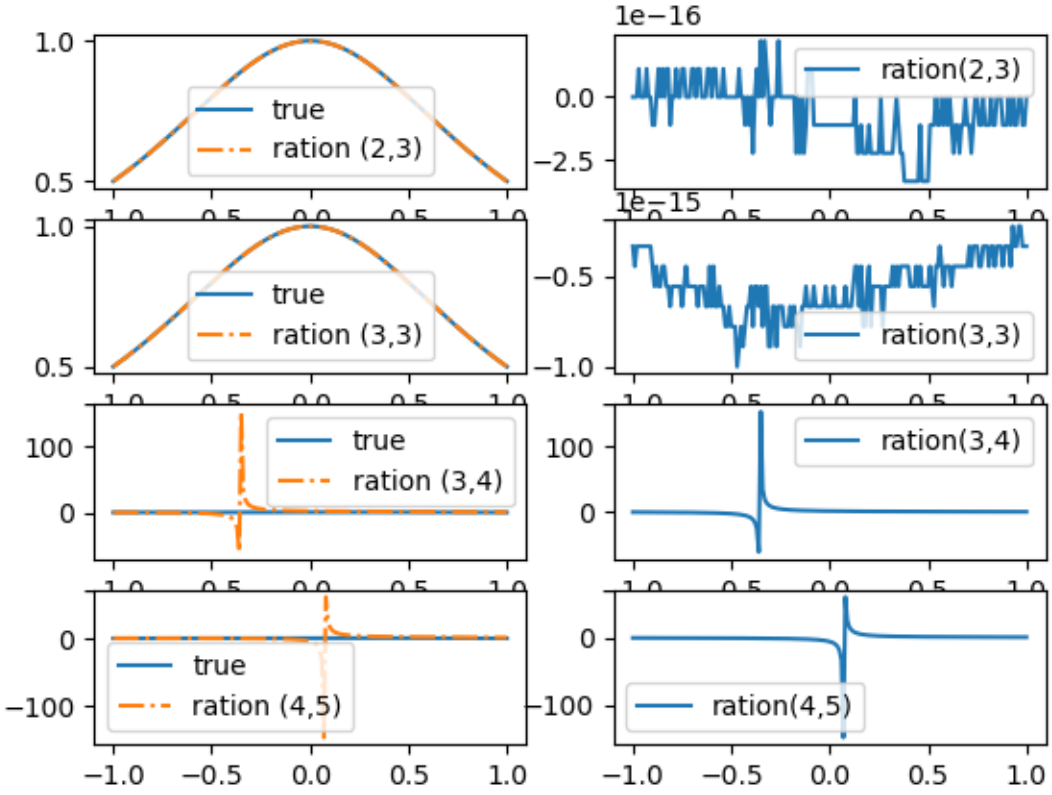

FIG. 5. Compare the result of different order rational function interpolations (left panel) and the error of them (right).

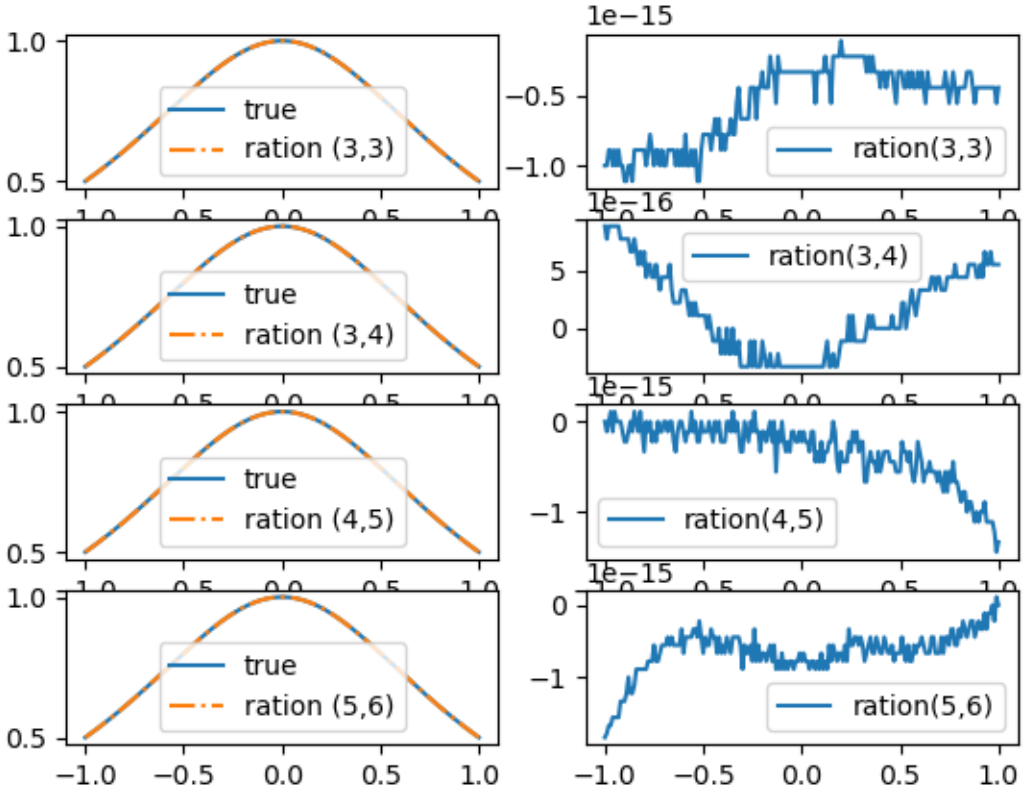

FIG. 6. Compare the result of different order rational function interpolations and the error of them.

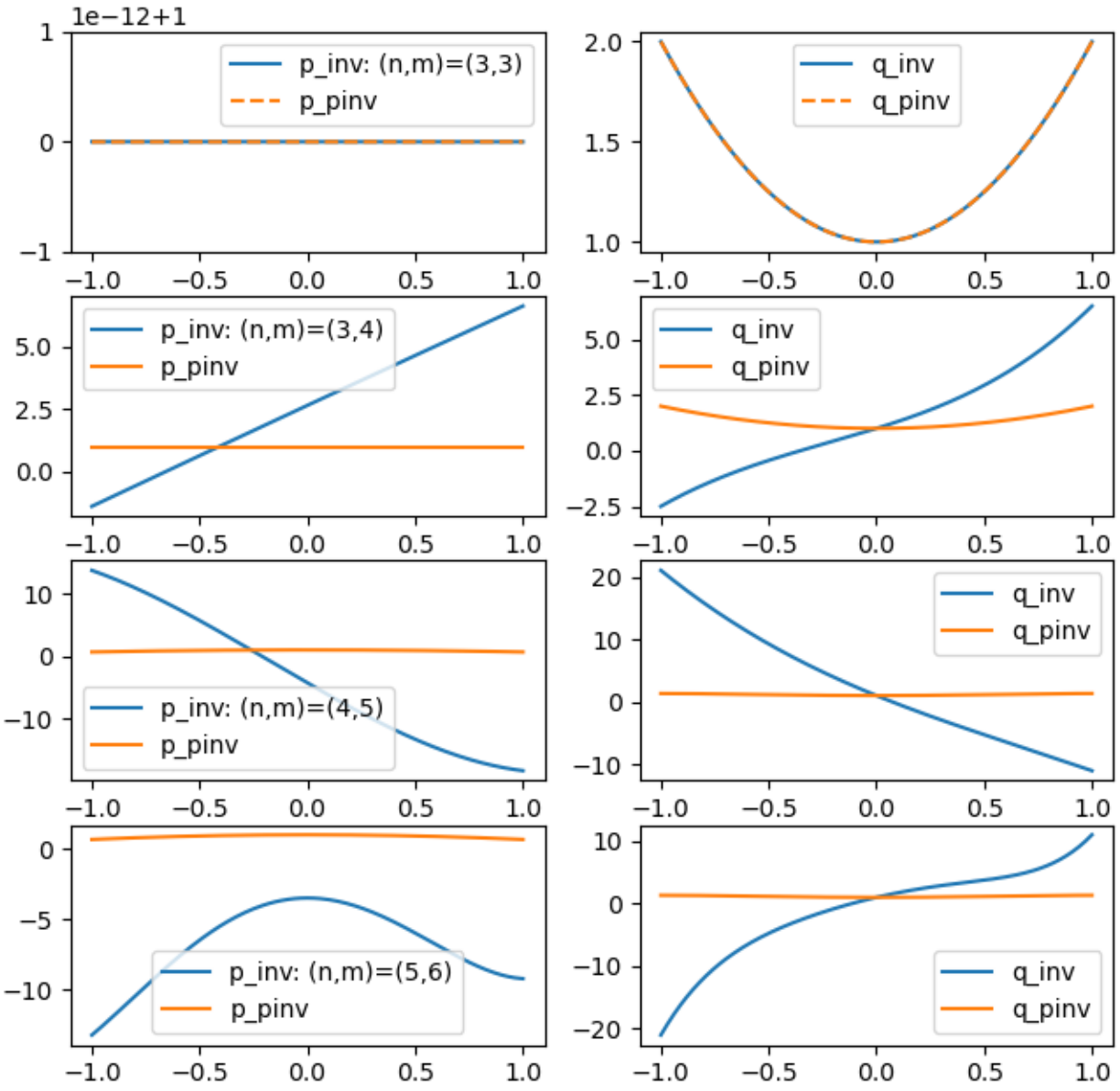

FIG. 7. Compare the result of different order rational function interpolations (left) and the error of them (right).

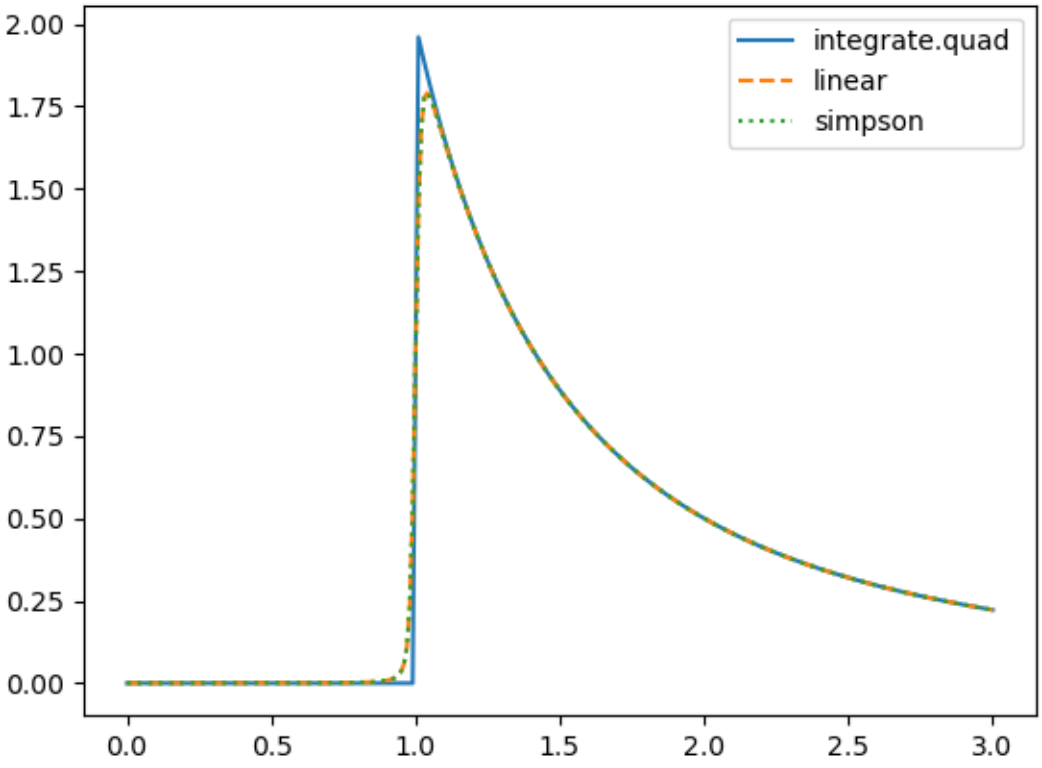

FIG. 8. Compare the result of electric field solved by my integrator and scipy.integrate.quad. Here I set the number of point is 101.