#1.

a)

As seen in lecture, we can take a linear combination of two function calls around x in order to cancel second order terms in the Taylor expansion (in fact, we can cancel all even order terms). We want to cancel the next order term, which is this case is the third order. However, since we have already cancelled the fourth order term, the next significant order is thus of fifth order:

$$D_1 \equiv \frac{f(x+\delta)-f(x-\delta)}{2\delta} = f'(x) + \frac{\delta^2 f'''(x)}{6} + \frac{\delta^4 f^{(5)}(x)}{120} + \frac{\epsilon g_1 f(x)}{2\delta} + \epsilon g_2 f'(x) + \cdots$$

Similarly, we can evaluate at $x \pm 2\delta$ and get:

$$D_2 \equiv \frac{f(x+2\delta)-f(x-2\delta)}{4\delta} = f'(x) + \frac{4\delta^2 f'''(x)}{6} + \frac{16\delta^4 f^{(5)}(x)}{120} + \frac{\epsilon g_3 f(x)}{4\delta} + \epsilon g_4 f'(x) + \cdots$$

We can combine these two expression as follows in order to cancel the third derivatives and have a leading term of f':

$$\frac{4D_1 - D_2}{3} = f'(x) - \frac{\delta^4 f^{(5)}(x)}{30} - \frac{\epsilon g f(x)}{\delta} + \epsilon \bar{g} f'(x) + \cdots$$

Where g and $\bar{g}$ are order unity factors. Thus, our linear combination approximates $f'(x)$ with an error of about:

$$E \approx \left| \frac{\delta^4 f^{(5)}(x)}{30} + \frac{\epsilon g f(x)}{\delta} \right|$$

b)

To minimize this error, we simply set the derivative of E with respect to δ equal to zero:

$$\frac{4\delta^3 f^{(5)}(x)}{30} - \frac{\epsilon g f(x)}{2\delta} = 0$$

$$\Rightarrow \delta_{min} = \left(\frac{15 g \epsilon f(x)}{4 f^{(5)}(x)} \right)^{\frac{1}{5}} \sim \left(\frac{f(x)}{f^{(5)}(x)} \epsilon \right)^{\frac{1}{5}}$$

After coding this up, we can do a quick check by applying this to $f(x) = \exp(x)$. In this case, $f^{(5)} = f$ so, with $\epsilon = 10^{-16}$, we estimate the optimal δ to be around $10^{-3.2}$. As a second check, let's apply this now to $f(x) = \exp(0.01x)$. Now, $f^{(5)} = (0.01)^5 f = 10^{-10} f$ which means the optimal δ is now about $10^{-1.2}$. To verify that this indeed works for these cases, let's plot our error on f' as a function of δ (Figure 1). We indeed see that there are local minima where expected.

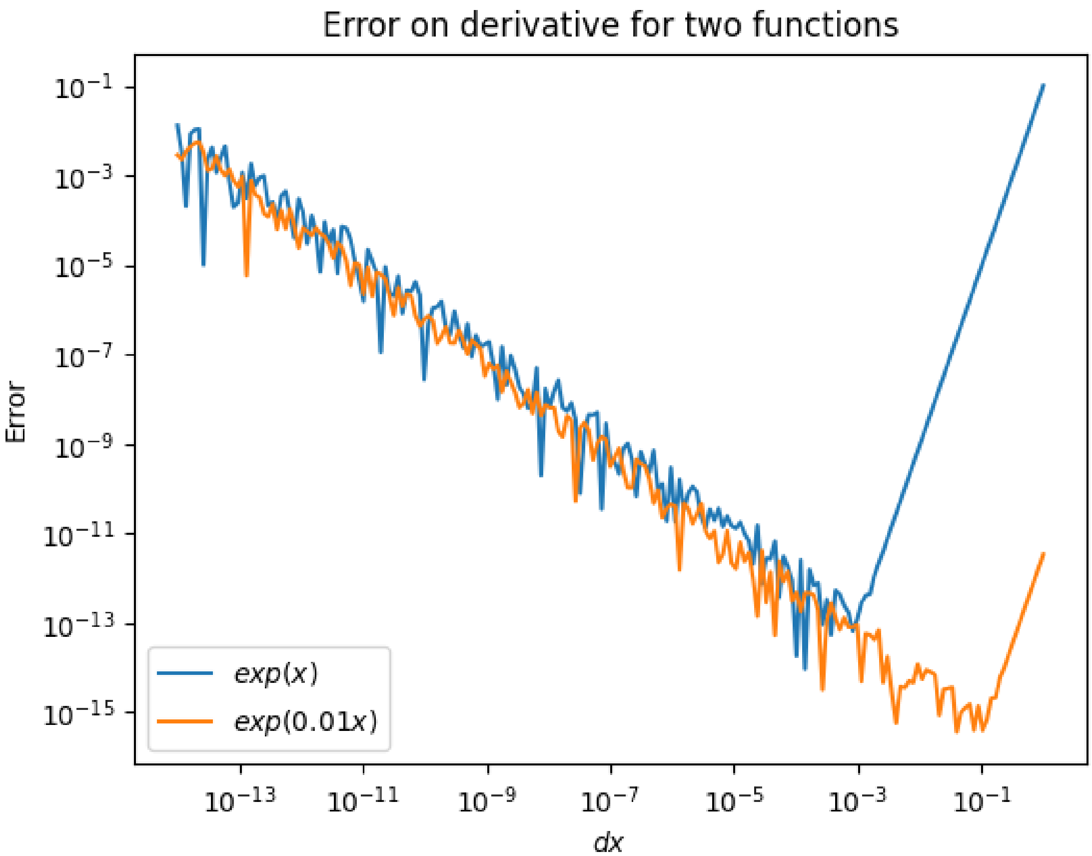


Figure 1: Error on f' for two different functions as a function of δ. We see a minimum around 10^{-3} for $\exp(x)$ and one around 10^{-1} for $\exp(0.01x)$

#2.

As done in lecture, we can use the use the centered derivative in order to get an approximate value for $f(x)$ accurate to about third order. We saw that to minimize the error, we ought to choose the following displacement:

$$\delta_{min} \sim \left(\frac{f(x)}{f'''(x)}\epsilon\right)^{\frac{1}{3}}$$

However, we would ideally need an estimate for f''' in order to properly choose δ. To do so, we will use a similar method as above: expand f around various points close to x and combine them to eliminate lower order terms, so that we are left with f''' and higher order terms.

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

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

And similarly,

$$f(x+2\delta) = f(x) + 2\delta f'(x) + \frac{4\delta^2 f''(x)}{2} + \frac{8\delta^3 f'''(x)}{6} + \frac{16\delta^4 f^{(4)}(x)}{24} + O(\delta^5)$$

$$f(x-2\delta) = f(x) - 2\delta f'(x) + \frac{4\delta^2 f''(x)}{2} - \frac{8\delta^3 f'''(x)}{6} + \frac{16\delta^4 f^{(4)}(x)}{24} + O(\delta^5)$$

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

Combining these two results, we can cancel the f' terms and isolate f''':

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

Thus, we only need to chose an appropriate δ to get $f'''(x)$. After trying different values of δ on different functions and comparing with their analytical third derivatives, I found that $\delta \sim 10^{-2}$ gives close enough results without returning nonsense values (which would happen when δ was too small and the numerator rounded to zero).

We can now evaluate the optimal δ for our first derivative (note that in the case that f''' evaluates to zero, I artificially set it equal to f in order to avoid division by zero errors).

#3.

Let's say we want to interpolate between two points (V_L, T_L) and (V_R, T_R) and we know $T' = \frac{dT}{dV}$ at each of them. This means we have 4 constraints to satisfy if we want the function and its first derivative to match at the boundary. Thus, consider a cubic polynomial

$$P(V) = a + bV + cV^2 + dV^3.$$

Imposing our constraints on P gives us four equations for four unknown coefficients:

$$a + bV_L + cV_L^2 + dV_L^3 = T_L$$

$$a + bV_R + cV_R^2 + dV_R^3 = T_R$$

$$0 + b + 2cV_L + 3dV_L^2 = T_L'$$

$$0 + b + 2cV_R + 3dV_R^2 = T_R'$$

Or equivalently:

$$\begin{pmatrix} 1 & V_L & V_L^2 & V_L^3 \\ 1 & V_R & V_R^2 & V_R^3 \\ 0 & 1 & 2V_L & 3V_L^2 \\ 0 & 1 & 2V_R & 3V_R^2 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} T_L \\ T_R \\ T_L' \\ T_R' \end{pmatrix}$$

Inverting this matrix allows us to find our four coefficients.

What remains is thus to find the correct neighbouring points given an arbitrary V. To do so, we simply look in our data for the two nearest voltages such that V sits in between the two, find the appropriate coefficients and then evaluate $P(V)$. In order to give a *rough* estimate of the error, we can compare with the line that connects the two points since shouldn't ever give crazy values:

$$E \approx \left(T_L + \frac{T_R - T_L}{V_R - V_L} \cdot (V - V_L) \right) - P(V)$$

Of course, in the case that V is one point in our data set, there is no need to interpolate and so, we can simply return the corresponding T and an error of 0.

If we are given an array instead of a single number, we do this procedure for each point in the array. Figure 2 shows the complete interpolation as well as the relative error.

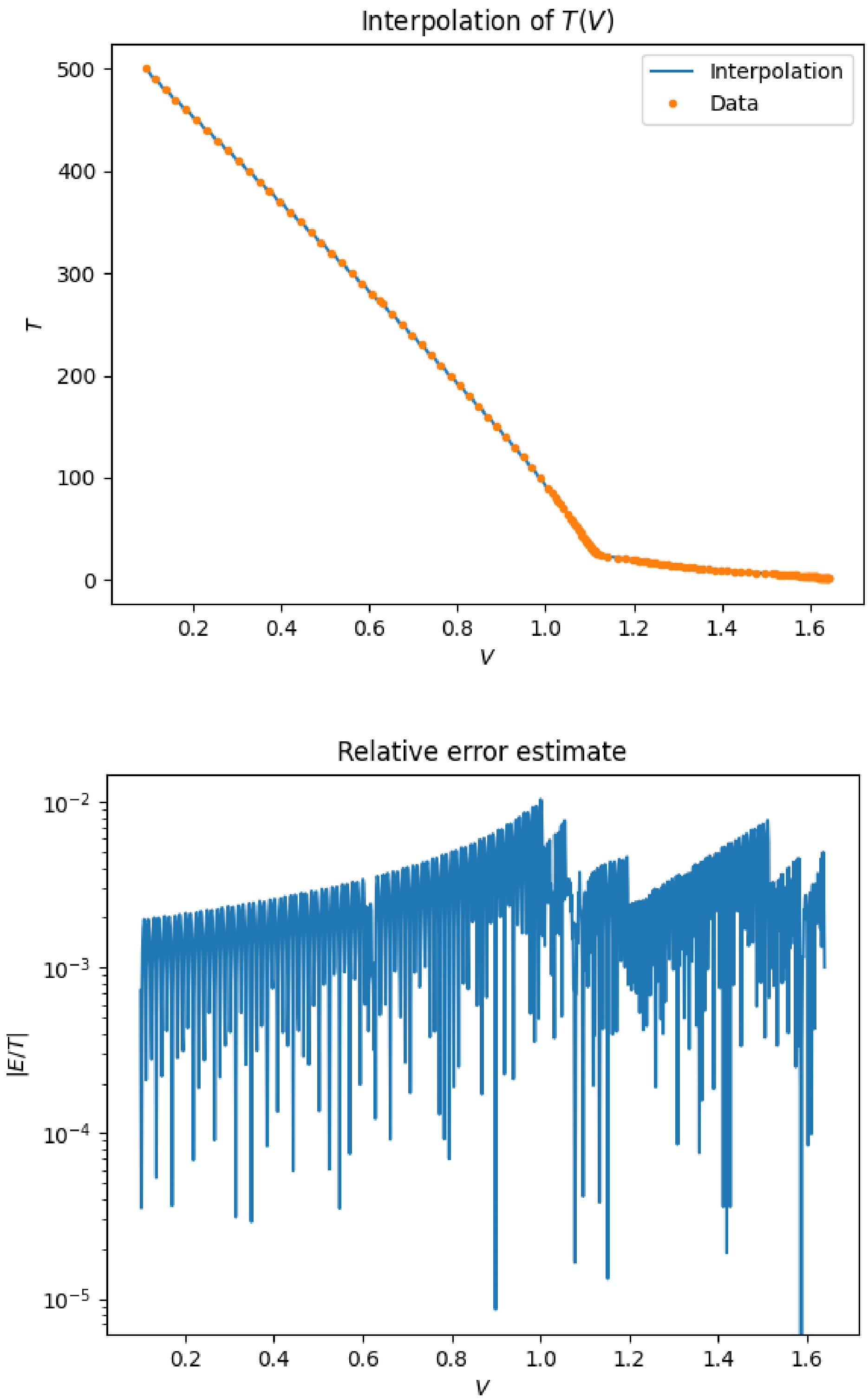
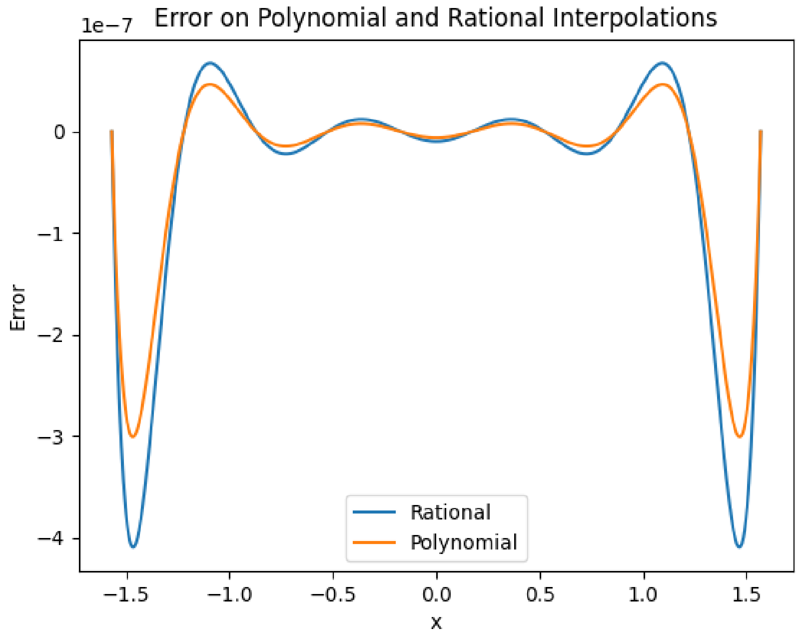


Figure 2: Interpolation on T(V) and its relative error estimate.

#4.

We begin by interpolating $cos(x)$ between $-\pi/2$ and $\pi/2$ using a polynomial fit, a cubic spline as well as a rational function with degrees of $n = 4$ and $m = 5$ (the code used is pretty much the one used in class). We then compare their error in Figure 3:



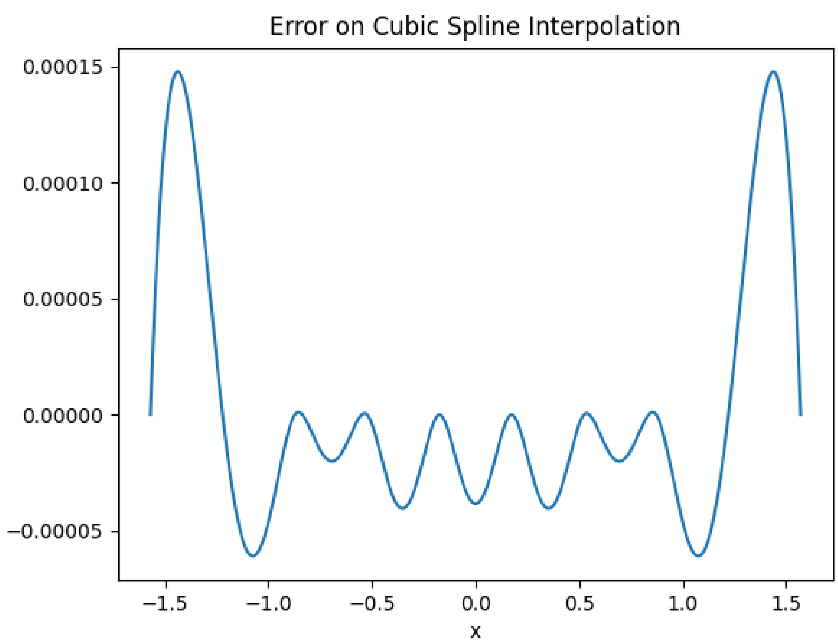
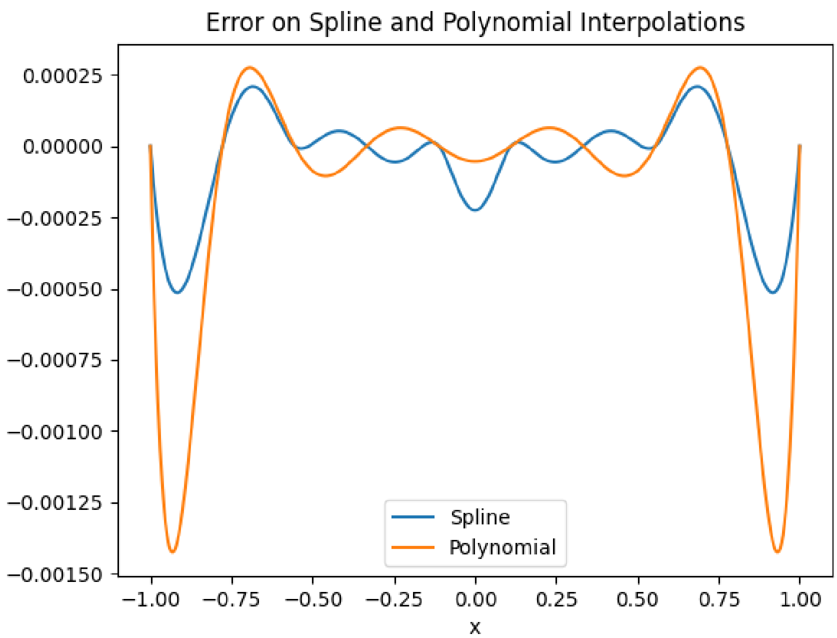


Figure 3: Errors on three different interpolations of $cos(x)$

We can do the same, but on $f(x) = \frac{1}{1+x^2}$ (Figure 4). This time, while the spline and polynomial fits give sensibly the same answer, we see that the rational fit gives nonsense!



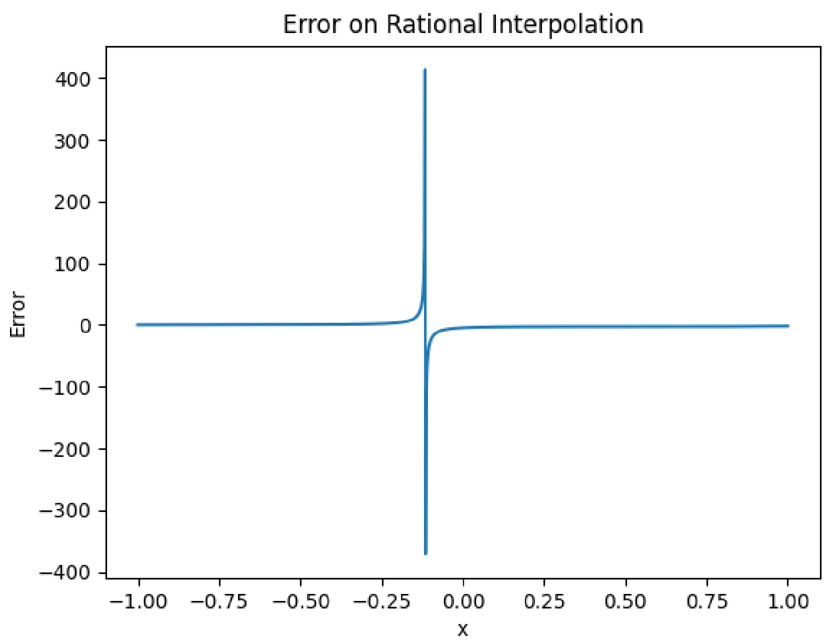


Figure 4: Errors on three different interpolations of a Lorentzian

This seems weird because not only is a Lorentzian already a rational function, but we would also normally expect that higher order should give higher precision, but we see the opposite! Luckily, if we use np.linalg.pinv instead to invert the matrix, we get errors close to machine precision (Figure 5), which now agrees with the fact that a Lorentzian *is* a rational function.

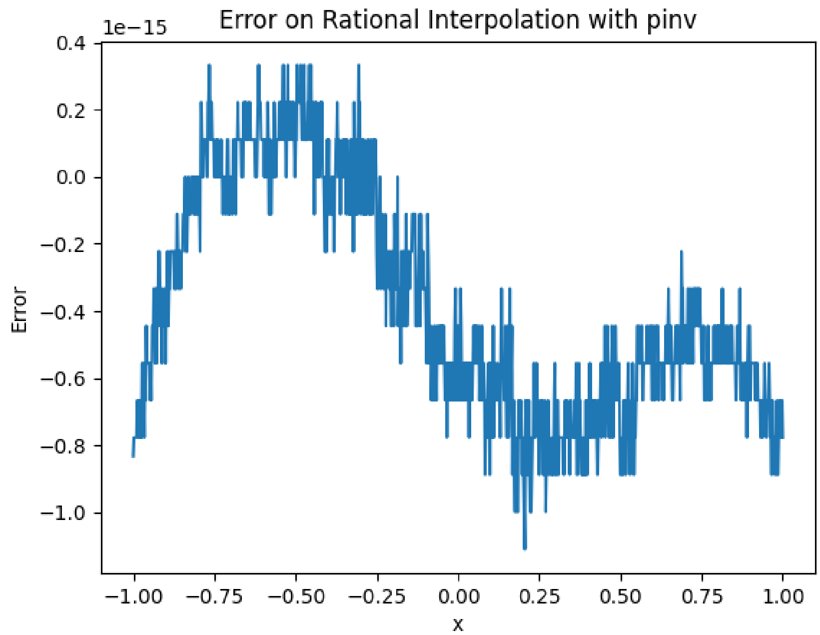


Figure 5: Error on the rational interpolations of a Lorentzian using pinv

If we look at the coefficients that our fit calculated, we would expect something like [1,0,0,0,0] for p and [0,1,0,0,0] for q since these values in theory give us back a Lorentzian exactly. However, we have something around [1,0,-$\frac{1}{3}$,0,0] for p and [0,$\frac{2}{3}$,0,-$\frac{1}{3}$,0] for q. But, this rational functional is in fact identical to a Lorentzian:

$$\frac{1-\frac{1}{3}x^2}{1+\frac{2}{3}x^2-\frac{1}{3}x^4} = \frac{1(1-\frac{1}{3}x^2)}{(1+x^2)(1-\frac{1}{3}x^2)} = \frac{1}{1+x^2}$$

So, since we are fitting a rational function to another, we have a degenerancy in our coefficients since we could multiply the Lorenztian by $P(x)/P(x)$ where $P(x)$ is an arbitrary polynomial of order min(n,$m-2$) and still get the same function, but with different coefficients. This means that our matrix is non invertible and trying to do so will lead to nonsensical answers. However, np.linalg.pinv can deal with these singularities and give a somewhat sensible answer.