Answers

Problem 1.

a)

Begin with the derivative limit expression about point x=x₀ with $\pm\delta$ increment in the centered difference approach:

$$f'(x_0) \sim \frac{f(x_0+\delta)-f(x_0-\delta)}{2\delta}$$

Let Taylor expansion of function f(x) about point x=x₀ equal:

$$f(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \frac{f'''(x_0)}{3!}(x-x_0)^3 + \cdots = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n$$

For x= x₀+δ:

$$f(x_0+\delta) = f(x_0) + f'(x_0)(x_0+\delta-x_0) + \frac{f''(x_0)}{2!}(x_0+\delta-x_0)^2 + \frac{f'''(x_0)}{3!}(x_0+\delta-x_0)^3 + \frac{f^{(4)}(x_0)}{4!}(x_0+\delta-x_0)^4 + \frac{f^{(5)}(x_0)}{5!}(x_0+\delta-x_0)^5 + \cdots = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x_0+\delta-x_0)^n = f(x_0) + f'(x_0)(\delta) + \frac{f''(x_0)}{2!}(\delta)^2 + \frac{f'''(x_0)}{3!}(\delta)^3 + \frac{f^{(4)}(x_0)}{4!}(\delta)^4 + \frac{f^{(5)}(x_0)}{5!}(\delta)^5 + \cdots$$

For x= x₀ – δ:

$$f(x_0-\delta) = f(x_0) + f'(x_0)(x_0-\delta-x_0) + \frac{f''(x_0)}{2!}(x_0-\delta-x_0)^2 + \frac{f'''(x_0)}{3!}(x_0-\delta-x_0)^3 + \frac{f^{(4)}(x_0)}{4!}(x_0-\delta-x_0)^4 + \frac{f^{(5)}(x_0)}{5!}(x_0-\delta-x_0)^5 + \cdots = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x_0-\delta-x_0)^n = f(x_0) + f'(x_0)(-\delta) + \frac{f''(x_0)}{2!}(-\delta)^2 + \frac{f'''(x_0)}{3!}(-\delta)^3 + \frac{f^{(4)}(x_0)}{4!}(-\delta)^4 + \frac{f^{(5)}(x_0)}{5!}(-\delta)^5 + \cdots$$

Substitute the preceding two findings in the derivative expression:

$$f'(x_0) = \frac{f(x_0+\delta)-f(x_0-\delta)}{2\delta}$$

$$f'(x_0) = \frac{f(x_0)+f'(x_0)(\delta)+\frac{f''(x_0)}{2!}(\delta)^2+\frac{f'''(x_0)}{3!}(\delta)^3+\frac{f^{(4)}(x_0)}{4!}(\delta)^4+\frac{f^{(5)}(x_0)}{5!}(\delta)^5+\cdots}{2\delta} - \frac{f(x_0)+f'(x_0)(-\delta)+\frac{f''(x_0)}{2!}(-\delta)^2+\frac{f'''(x_0)}{3!}(-\delta)^3+\frac{f^{(4)}(x_0)}{4!}(-\delta)^4+\frac{f^{(5)}(x_0)}{5!}(-\delta)^5+\cdots}{2\delta}$$

$$f'(x_0) = \frac{f'(x_0)(\delta)+\frac{f'''(x_0)}{3!}(\delta)^3+\frac{f^{(5)}(x_0)}{5!}(\delta)^5+\cdots}{2\delta} - \frac{f'(x_0)(-\delta)+\frac{f'''(x_0)}{3!}(-\delta)^3+\frac{f^{(5)}(x_0)}{5!}(-\delta)^5+\cdots}{2\delta}$$

$$f'(x_0) = \frac{f'(x_0)(\delta)+\frac{f'''(x_0)}{3!}(\delta)^3+\frac{f^{(5)}(x_0)}{5!}(\delta)^5+\cdots}{\delta} + f'(x_0) + \frac{f'''(x_0)}{3!}(\delta)^2 + \frac{f^{(5)}(x_0)}{5!}(\delta)^4 + \cdots \qquad (1)$$

Next, begin with the derivative limit expression about point x=x$_0$ with $\pm 2\delta$ increment in the centered difference approach:

$f'(x_0) \sim \frac{f(x_0+2\delta)-\mathrm{f}(x_0-2\delta)}{4\delta}$

Let Taylor expansion of function f(x) about point x=x$_0$ equal:

$f(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \frac{f'''(x_0)}{3!}(x-x_0)^3 + \cdots = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n$

For x= x$_0$+2δ:

$f(x_0+2\delta) = f(x_0) + f'(x_0)(x_0+2\delta-x_0) + \frac{f''(x_0)}{2!}(x_0+2\delta-x_0)^2 + \frac{f'''(x_0)}{3!}(x_0+2\delta-x_0)^3 + \frac{f^{(4)}(x_0)}{4!}(x_0+2\delta-x_0)^4 + \frac{f^{(5)}(x_0)}{5!}(x_0+2\delta-x_0)^5 + \cdots = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x_0+2\delta-x_0)^n = f(x_0) + f'(x_0)(2\delta) + \frac{f''(x_0)}{2!}(2\delta)^2 + \frac{f'''(x_0)}{3!}(2\delta)^3 + \frac{f^{(4)}(x_0)}{4!}(2\delta)^4 + \frac{f^{(5)}(x_0)}{5!}(2\delta)^5 + \cdots$

For x= x$_0$ – 2δ:

$f(x_0-2\delta) = f(x_0) + f'(x_0)(x_0-2\delta-x_0) + \frac{f''(x_0)}{2!}(x_0-2\delta-x_0)^2 + \frac{f'''(x_0)}{3!}(x_0-2\delta-x_0)^3 + \frac{f^{(4)}(x_0)}{4!}(x_0-2\delta-x_0)^4 + \frac{f^{(5)}(x_0)}{5!}(x_0-2\delta-x_0)^5 + \cdots = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x_0-2\delta-x_0)^n = f(x_0) + f'(x_0)(-2\delta) + \frac{f''(x_0)}{2!}(-2\delta)^2 + \frac{f'''(x_0)}{3!}(-2\delta)^3 + \frac{f^{(4)}(x_0)}{4!}(-2\delta)^4 + \frac{f^{(5)}(x_0)}{5!}(-2\delta)^5 + \cdots$

Substitute the preceding two findings in the derivative expression:

$$f'(x_0) = \frac{f(x_0+2\delta)-\mathrm{f}(x_0-2\delta)}{4\delta}$$

$$f'(x_0) = \frac{f(x_0)+f'(x_0)(2\delta)+\frac{f''(x_0)}{2!}(2\delta)^2+\frac{f'''(x_0)}{3!}(2\delta)^3+\frac{f^{(4)}(x_0)}{4!}(2\delta)^4+\frac{f^{(5)}(x_0)}{5!}(2\delta)^5+\cdots}{4\delta} - \frac{f(x_0)+f'(x_0)(-2\delta)+\frac{f''(x_0)}{2!}(-2\delta)^2+\frac{f'''(x_0)}{3!}(-2\delta)^3+\frac{f^{(4)}(x_0)}{4!}(-2\delta)^4+\frac{5(x_0)}{5!}(-2\delta)^5+\cdots}{2\delta}$$

$$f'(x_0) = \frac{f'(x_0)(2\delta)+\frac{f'''(x_0)}{3!}(2\delta)^3+\frac{f^{(5)}(x_0)}{5!}(2\delta)^5+\cdots}{4\delta} - \frac{f'(x_0)(-2\delta)+\frac{f'''(x_0)}{3!}(-2\delta)^3+\frac{f^{(5)}(x_0)}{5!}(-2\delta)^5+\cdots}{4\delta}$$

(2)

$$f'(x_0) = \frac{f'(x_0)(4\delta)+\frac{f'''(x_0)}{3!}16(\delta)^3+\frac{f^{(5)}(x_0)}{5!}64(\delta)^5+\cdots}{4\delta} = f'(x_0) + \frac{f'''(x_0)}{3!}4\delta^2 + \frac{f^{(5)}(x_0)}{5!}16\delta^4 + \cdots \quad (2)$$

Next combine the resultant derivative expressions (1) and (2) linearly.

$$A[f'(x_0) + \frac{f'''(x_0)}{3!}(\delta)^2 + \frac{f^{(5)}(x_0)}{5!}(\delta)^4 + \cdots] + B[f'(x_0) + \frac{f'''(x_0)}{3!}4\delta^2 + \frac{f^{(5)}(x_0)}{5!}16\delta^4 + \cdots] =$$
$$A[\frac{f(x_0+\delta)-\mathrm{f}((x_0-\delta)}{2\delta}] + B[\frac{f(x_0+2\delta)-\mathrm{f}((x_0-2\delta)}{4\delta}]$$

Select A=4 and B =-1 to eliminate f''' terms:

$$4[f'(x_0) + \frac{f'''(x_0)}{3!}\delta^2 + \frac{f^{(5)}(x_0)}{5!}\delta^4 + \cdots] - 1[f'(x_0) + \frac{f'''(x_0)}{3!}4\delta^2 + \frac{f^{(5)}(x_0)}{5!}16\delta^4 + \cdots] =$$
$$4[\frac{f(x_0+\delta)-\mathrm{f}(x-\delta)}{2\delta}] - 1[\frac{f(x_0+2\delta)-\mathrm{f}(x-2\delta)}{4\delta}]$$

Simplify the expression:

$$3f'(x_0) - 12\frac{f^{(5)}(x_0)}{5!}\delta^4 + \cdots = \frac{8f(x_0+\delta)-8\mathrm{f}(x-\delta)-f(x_0+2\delta)+\mathrm{f}(x-2\delta)}{4\delta}$$

The derivative at x_0 would equal:

$$\boxed{f'(x_0) = \frac{8f(x_0+\delta)-8\mathrm{f}((x_0-\delta)-f(x_0+2\delta)+\mathrm{f}((x_0-2\delta)}{12\delta} + \frac{f^{(5)}(x_0)}{30}\delta^4 + \cdots} \quad (3)$$

b)

Let ε represent the accuracy.

A truncation error ε_t is introduced upon selection of the few first terms in the Taylor series, inherently introducing error based on the approximate nature of the adopted approach. On the other hand, a round-off error ε_r is evident in the representation of real numbers by the computer. Total error ε shall equal $\varepsilon_t + \varepsilon_r$.

In equation (3), the ε_r would manifest in the first term as follows:

$$f(x_0)\frac{8\varepsilon_r+8\varepsilon_r+\varepsilon_r+\varepsilon_r}{12\delta} = f(x_0)\frac{8\varepsilon_r+8\varepsilon_r+\varepsilon_r+\varepsilon_r}{12\delta} = f(x_0)\frac{3\varepsilon_r}{2\delta}$$

The truncation error in (3) is embodied in the leading term:

$$\varepsilon_t = \frac{f^{(5)}(x_0)}{30}\delta^4$$

The total error would then equal:

$$\varepsilon = f(x_0)\frac{3\varepsilon_r}{2\delta} + \frac{f^{(5)}(x_0)}{30}\delta^4 \quad (4)$$

In order to minimize ε, take the first derivative of the expression in (4) with respect to δ and solve for its roots:

$$\frac{d\varepsilon}{d\delta} = -f(x_0)\frac{3\varepsilon_r}{2\delta^2} + \frac{4f^{(5)}(x_0)}{30}\delta^3 = 0$$

$$f(x_0)\frac{3\varepsilon_r}{2\delta^2} = \frac{4f^{(5)}(x_0)}{30}\delta^3$$

$$f(x_0)\frac{3\varepsilon_r}{2\delta^2} = \frac{4f^{(5)}(x_0)}{30}\delta^3$$

$$f(x_0)\frac{90\varepsilon_r}{8} = f^{(5)}(x_0)\delta^5$$

Thus, in terms of machine precision and various properties of the function, δ should be in the order of:

$$\boxed{\delta \sim [\frac{10\varepsilon_r f(x_0)}{f^{(5)}(x_0)}]^{1/5}} \qquad (5)$$

Python adopts $\varepsilon_r = 10^{-16}$ for double precision.

For $f(x)=e^x$, $\delta \sim [\frac{10x10^{-16}exp(x_0)}{exp(x_0)}]^{1/5} \sim 10^{-3}$

For $f(x)=e^{0.01x}$, $\delta \sim [\frac{10x10^{-16}exp(0.01x_0)}{(0.01)^5 exp(0.01x_0)}]^{1/5} \sim 10^{-1}$

At $x_0=1$, the absolute value difference between the calculated f' and the numerical solution for exp(x) and exp(0.01x) are shown in Figures 1 and 2, respectively, to visualize the optimal δ (i.e. $\delta_{optimal}$) in order to minimize error. Corresponding code is provided in Appendix A Problem 1.

As per the location of the minima of residuals in both Figures 1 and 2, the estimated $\delta_{optimal}$ is roughly correct for each f' calculations in Python. Several different x values including 0 and 10 were analyzed. The roughly correct estimation of the optimized value of δ were in close agreement with the observed minima in residuals as the estimated and observed δ values were on the same order of magnitude. For brevity purposes, only the behavior of the function at $x_0=1$ is shown here and in the code.

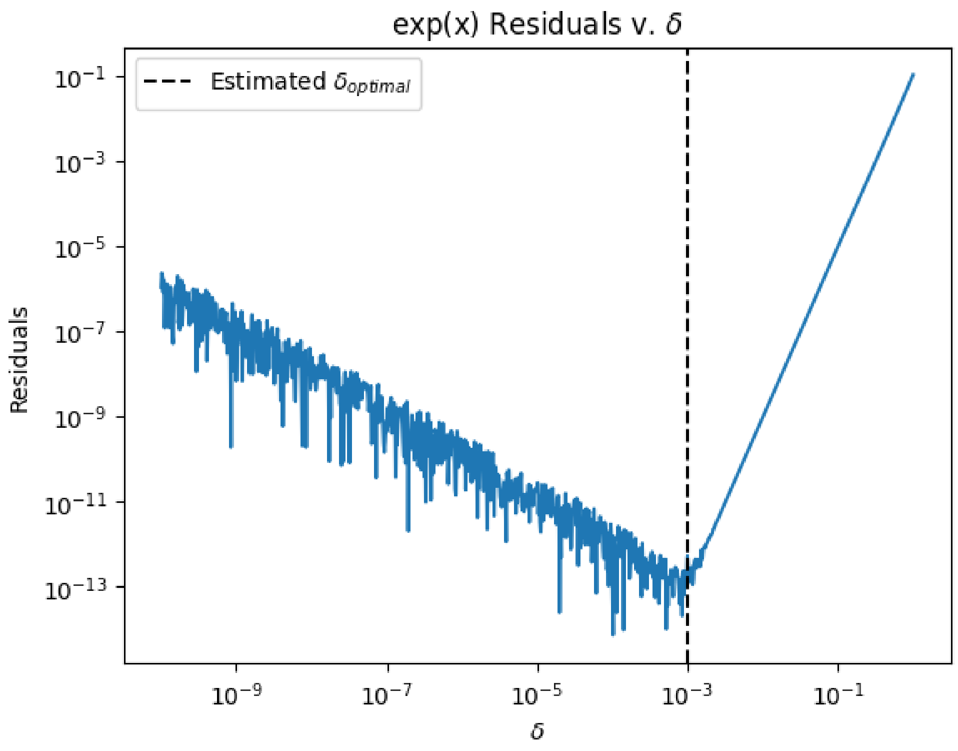

Figure 1. Comparison of estimated $\delta_{optimal}$ to residual trends of exp(x) at $x_0 = 1$

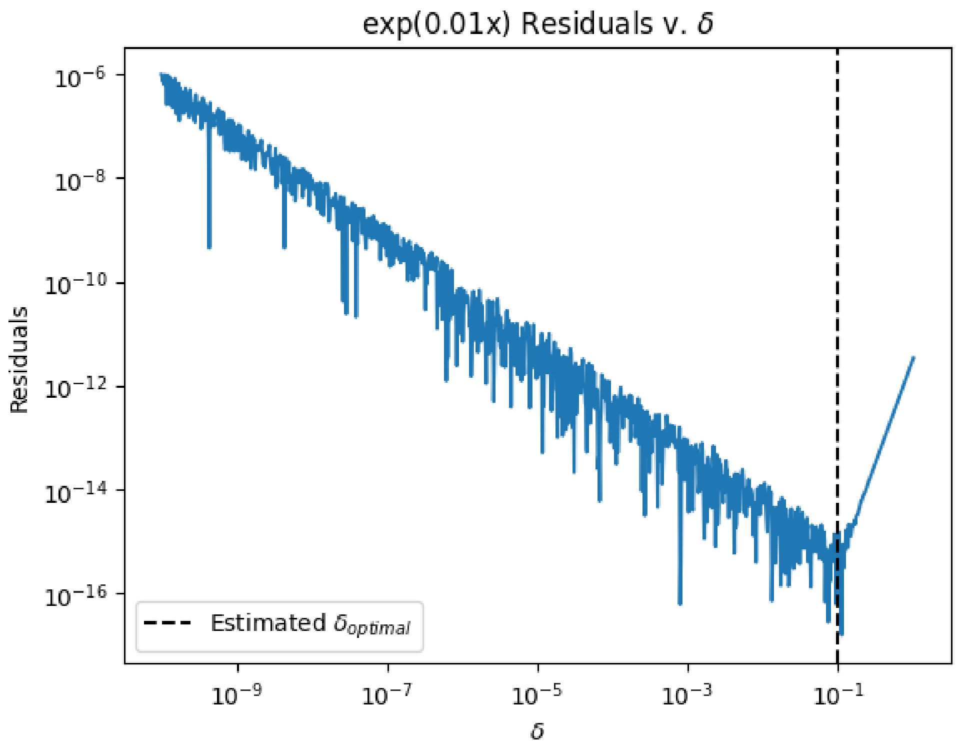

Figure 2. Comparison of estimated $\delta_{optimal}$ to residual trends of exp(0.01x) at $x_0 = 1$

Problem 2

A numerical differentiator with prototype def ndiff(fun,x,full=False) was constructed where fun is a function and x is a value. If full is set to False, ndiff should return the numerical derivative at x. If full is True, it should return the derivative, dx, and an estimate of the error on the derivative.

Evaluate the centered difference for a third-order derivative (Chapra & Canale, 2011):

$$f'''(x_0) = \frac{f(x_0+2\delta)-2f(x_0+\delta)+2f(x_0-\delta)-f(x_0-2\delta)}{2\delta^3}$$

According to the lecture notes, the optimal δ in the centered difference method could be approximated as:

$$\delta \sim [\frac{\varepsilon_r f(x_0)}{f'''(x_0)}]^{1/3}$$

Optimal δ, thus, may be computed via substitution of the third-order derivative in the preceding expression. After the optimal δ is estimated, the numerical differentiator routine shall calculate the derivative.

The centered difference for the first-order derivative is defined as:

$$f'(x_0) \sim \frac{f(x_0+\delta)-f(x_0-\delta)}{2\delta}$$

where the total error according to lecture notes is:

$$\varepsilon \sim \frac{f(x_0)\varepsilon_r}{\delta} + f'''(x_0)\delta^2$$

The function ndiff constructed with an initial $\varepsilon_r^{1/3}=10^{-16/3}$ was tested for f(x)=sin(x) at $x_0=\pi/6$ was evaluated. Results are provided in Appendix A Problem 2. The error was determined to be on the order of 10^{-11}.

Problem 3.

A routine that takes an arbitrary voltage and interpolates the data points to return a temperature is constructed. A rough quantitative estimate of the error in the interpolation is also provided. The prototype def lakeshore(V,data) processes output data from data=np.loadtxt('lakeshore.txt'). The code supports voltage V being either a number or an array. Enclosed code (Appendix A Problem 3) operates on an array V1 input which can be easily set to a voltage value in the provided domain.

Function splrep was utilized for the cubic spline, whose accuracy was evaluated via a bootstrap approach with N_t=1200 resampled sets. Each randomly formed subsample set included 40% of all available data points. This fraction might be increased to 80% or more for further analysis. The uncertainty was expressed in terms of an overall standard deviation of all differences between the initial cubic spline fit and each resampled subset fit.

An arbitrary set of 15 input voltage values in the domain [0.1, 1.6] were estimated according to the cubic spline fit splrep (Figure 3). Visual inspection of the trends revealed a close fit with the measured data. The calculated error for each trial was on the order of 10^{-4} to 10^{-1}, evidencing some uncertainty, which needs further investigated based upon the demands of the measurement technique for the diode experiment.

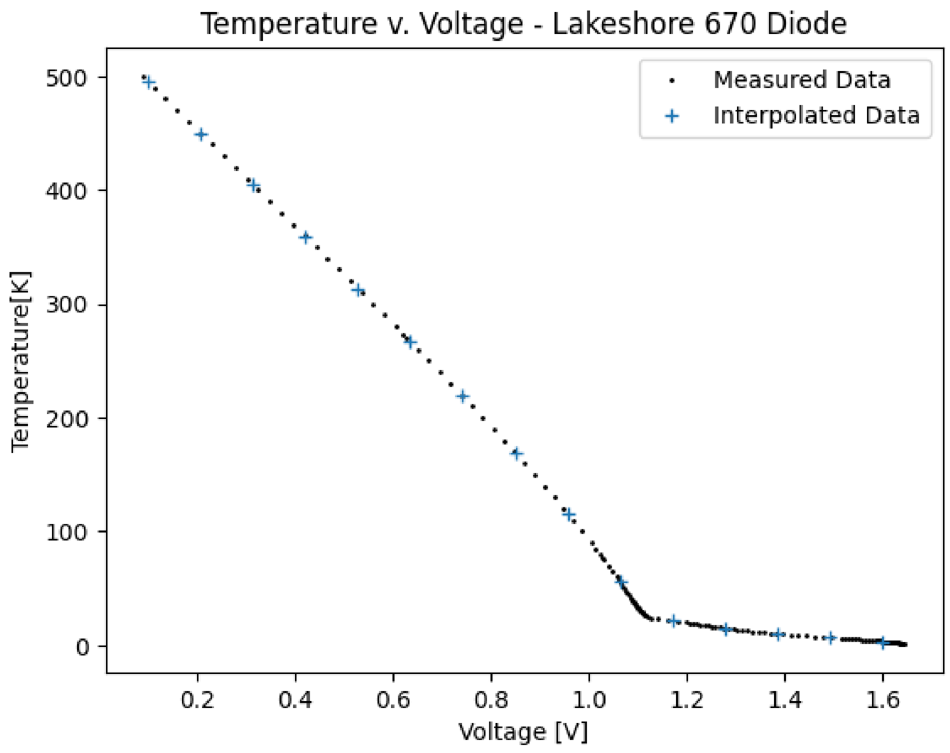

Figure 3. Comparison of experimental data with output of select set of arbitrary input V

Problem 4.

Eight equally spaced-apart points on the cos(x) function within the domain $[-\pi/2, \pi/2]$ were selected. In parallel, eight equally spaced-apart points on the Lorentzian $L(x)=(x^2+1)^{-1}$ function within the domain [-1, 1] were identified. Fits comprising fourth-order polynomial, cubic spline, and rational function (numerator order n=4, denominator order m=5) were implemented first with np.linalg.inv and next with np.linalg.pinv inside the rat_fit function. Whereas the cos(x) function was well-behaved in the rational fit without and with pseudoinverse matrix operation (Figures 4 and 5), the L(x) function was ill-behaved without pseudoinverse operation (Figure 6), performing quite well with pseudoinverse operation(Figure 7). Several other higher order rational functions were tested (not shown), resulting in further ill fits. This discrepancy was not observed at rational fits over fewer points (i.e. four) for lower order rational fits with second-order polynomial, cubic spline, and rational function (numerator order n=2, denominator order m=3) fits for and L(x) (Figures 8 and 9) and cos(x).

Evaluation of the success of each fit was determined via a standard deviation calculation for each fit based on the difference between the predicted abscissa and their corresponding real values. The errors for cos(x) fits were on the order of 10^{-4} for fourth-order polynomial and cubic spline trials, and 10^{-5} for the rational fit, regardless of employment of pseudoinverse operation (Figures 10 and 11). The errors for L(x) fits were on the order of 10^{-3} and 10^{-4} for the fourth-order polynomial and cubic spline trials, in turn, and 10^{1} for the rational fit when no pseudoinverse operation was utilized (Figure 12). The rational fit error was greatly reduced to the order of 10^{-16} upon pseudoinverse incorporation, which did not affect the errors for the remaining L(x) fits in this set (Figure 13).

Further evaluation of the L(x) fits at lower orders provided valuable insight. The errors were on the order of 10^{-2} for the second-order polynomial and cubic spline trials and 10^{-16} for the rational fit when no pseudoinverse operation was utilized (Figure 14). All three errors were not sensitive to the pseudoinverse incorporation in this set (Figure 15).

The results need to be considered in light of the matrix operations that govern the eigenvalue conversions in each fit calculation. The default matrix inv method produces the inverse of a matrix, while the pinv method generates the pseudo-inverse of the matrix, which resolves any problematic computations due to matrix singularity. A detailed analysis of the methodology follows.

Given the original Lorentzian $(x^2+1)^{-1}$ comprises zero and second order polynomials in the numerator and denominator, respectively, representation of a quadratic denominator with a fourth-order polynomial should have zero or near-zero coefficients for all odd terms. Such coefficients become problematic when inv method is adopted. Reciprocals of very small eigenvalues approach infinity. On the other hand, replacement of the very large reciprocals (i.e. infinity) with zeros in pseudoinverse approach remedies the challenging rational fit with higher order functions, maintaining that the zero and/or near-zero coefficients in the polynomial remain negligible.

Resultant p and q coefficients from the rat_fit of L(x) output reveal that the coefficients of x^1 and x^3 terms in the denominator are on the order of 10^{-15} for the np.linalg.pinv trials. These terms attain values in the order of 10^0-10^1 in the np.linalg.inv trial. Since these trivial odd power coefficients will approach infinity during the matrix inversion operation, the pseudo-inversion approach is proper so as to re-generate the negligibly small coefficients of x^1 and x^3 the denominator. Thus, x^1 and x^3 terms shall diminish. In practice, the rational fit dictates that the zeroth term in the denominator shall remain equal to 1. In this Lorentzian example, the approximate form for the rational fit (n=4, m=5) is:

$L(x)=(1-0.33x^2)(1+0.67x^2 -0.33x^4)^{-1}$.

The resultant rational function fit respects the conventional zeroth term unity, producing an acceptable representation with a very good fit with the original function $(1+x^2)^{-1}$, despite the unnecessarily high order of polynomials in the fraction. In brief, pseudoinverse operation serves a much-needed purpose by maintaining the unity of the zeroth order term in the denominator, while the remaining even power terms in the polynomial sufficiently model the original function in the given rational fit.

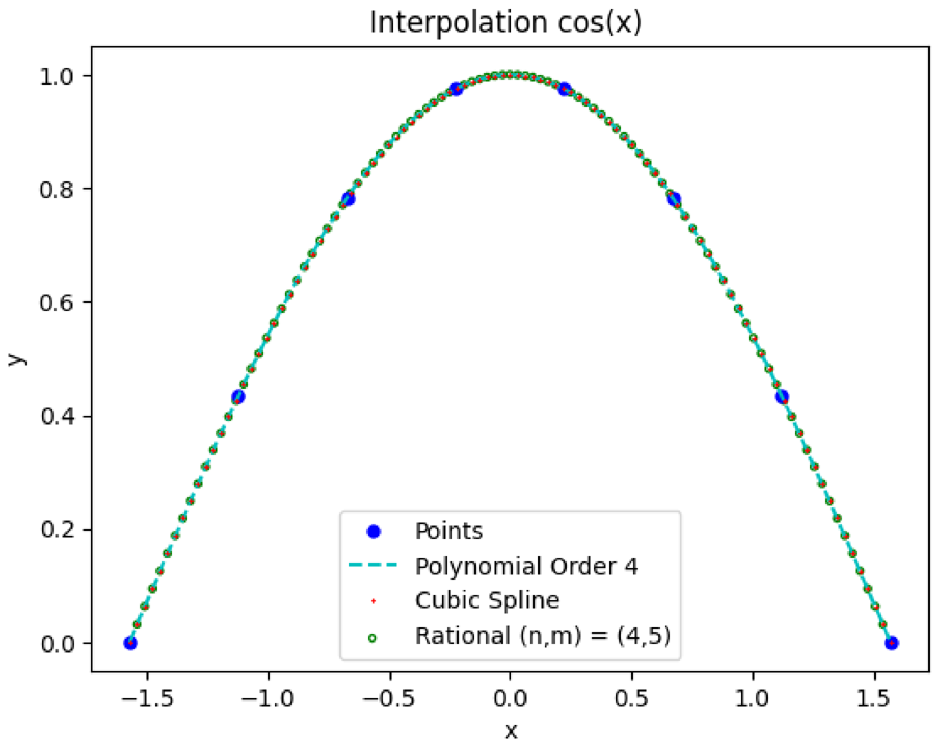

Figure 4. Interpolation of cos(x) without pseudoinverse operation – high order rational

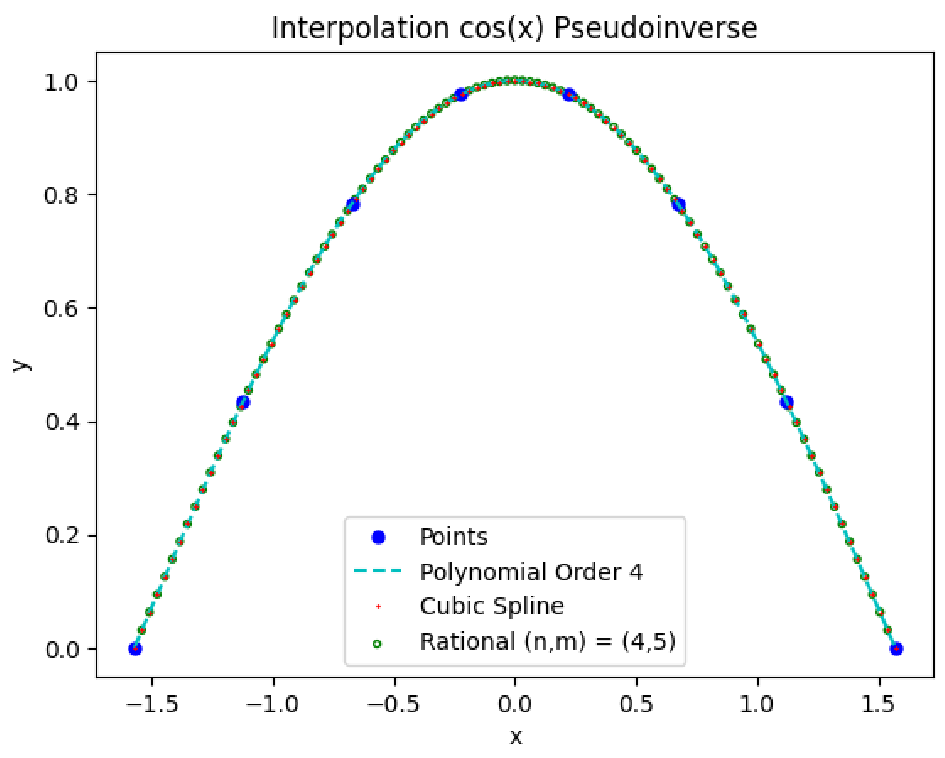

Figure 5. Interpolation of cos(x) with pseudoinverse operation – high order rational

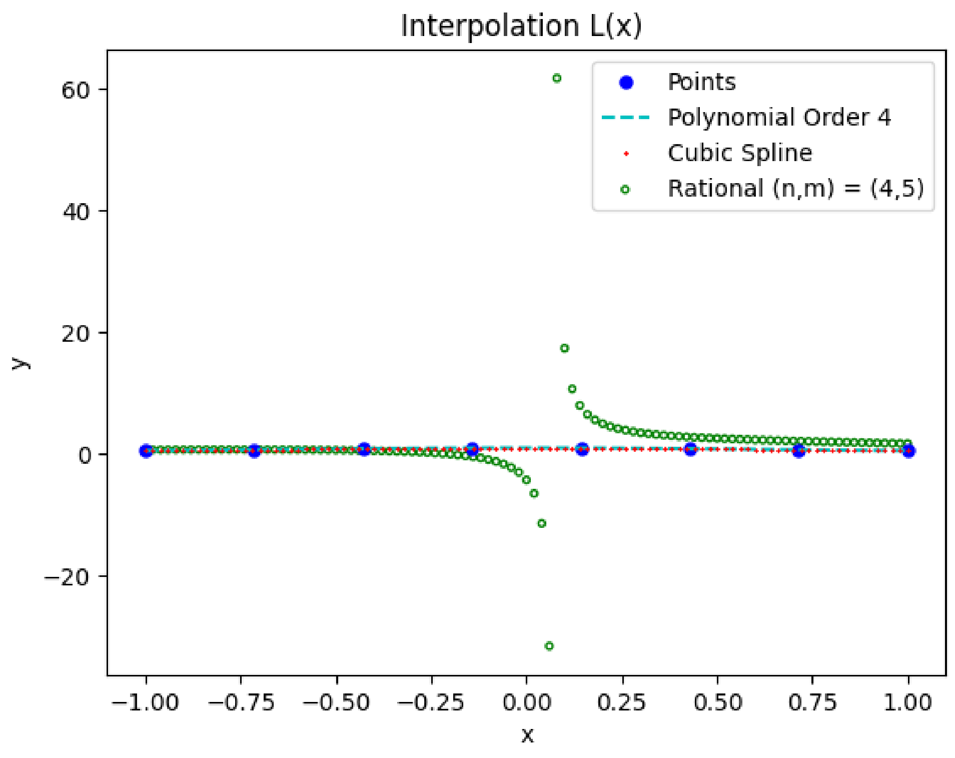

Figure 6. Interpolation of L(x) without pseudoinverse operation – high order rational

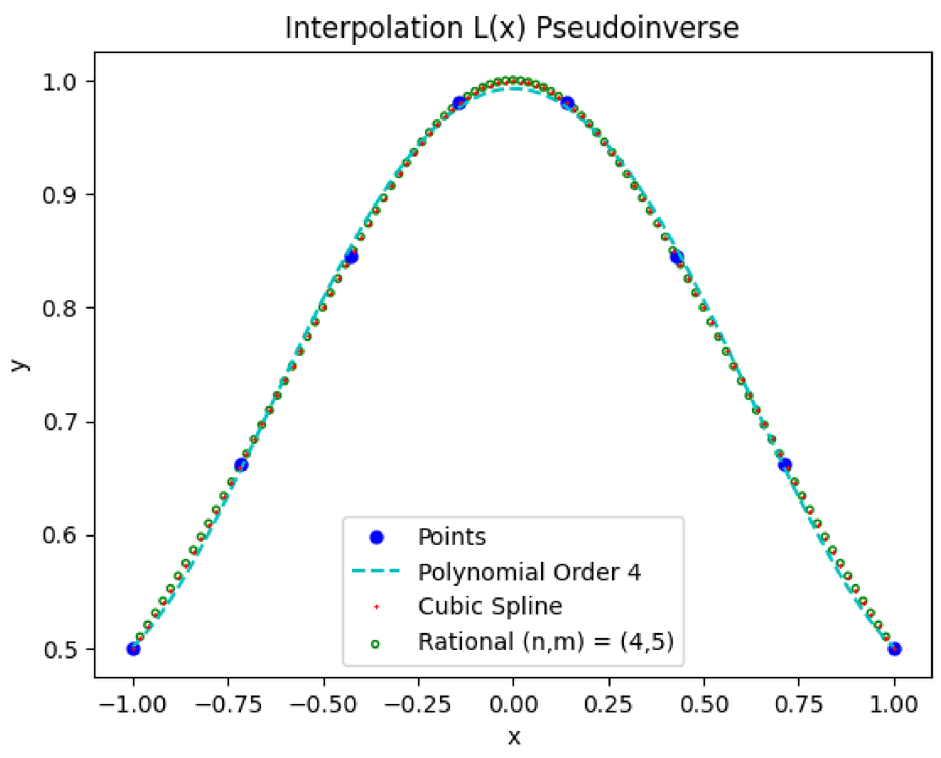

Figure 7. Interpolation of L(x) with pseudoinverse operation – high order rational

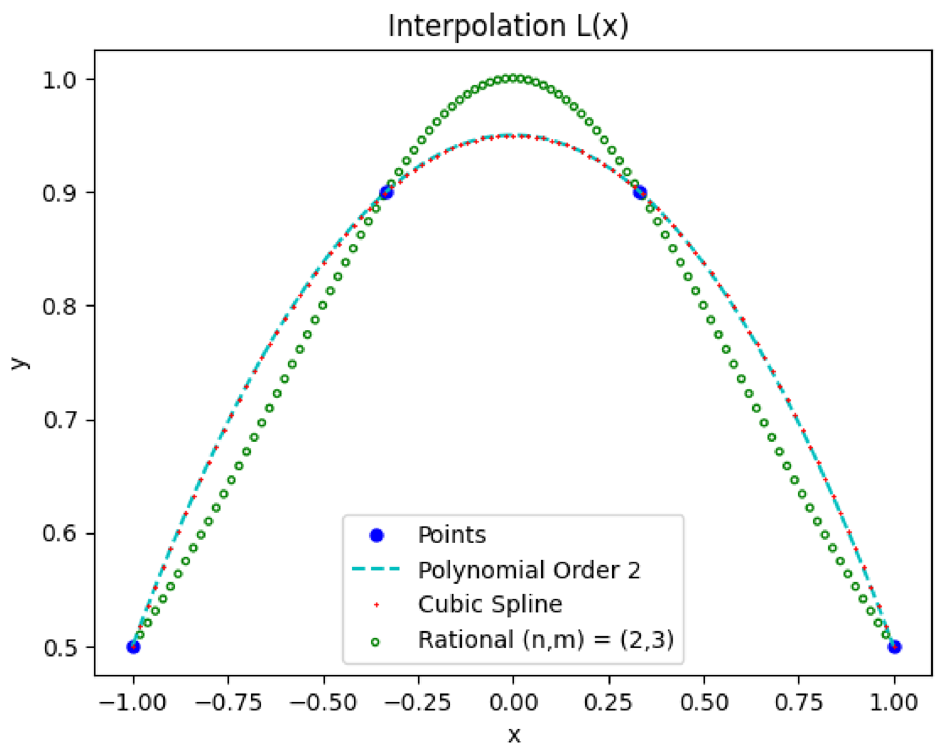

Figure 8. Interpolation of L(x) without pseudoinverse operation – low order rational

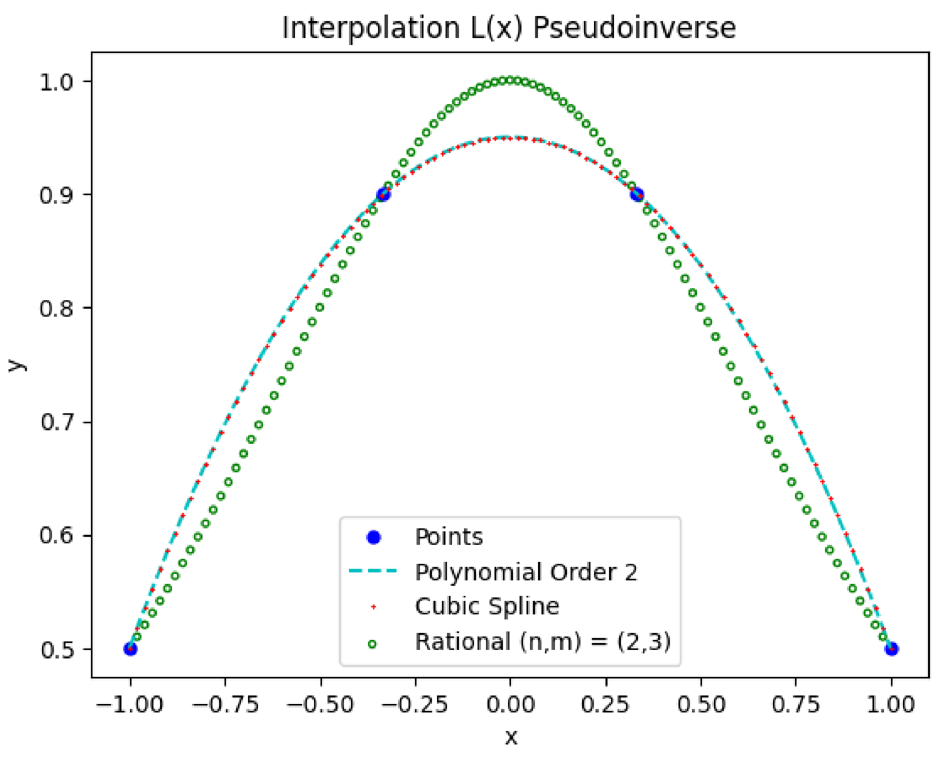

Figure 9. Interpolation of L(x) with pseudoinverse operation – low order rational

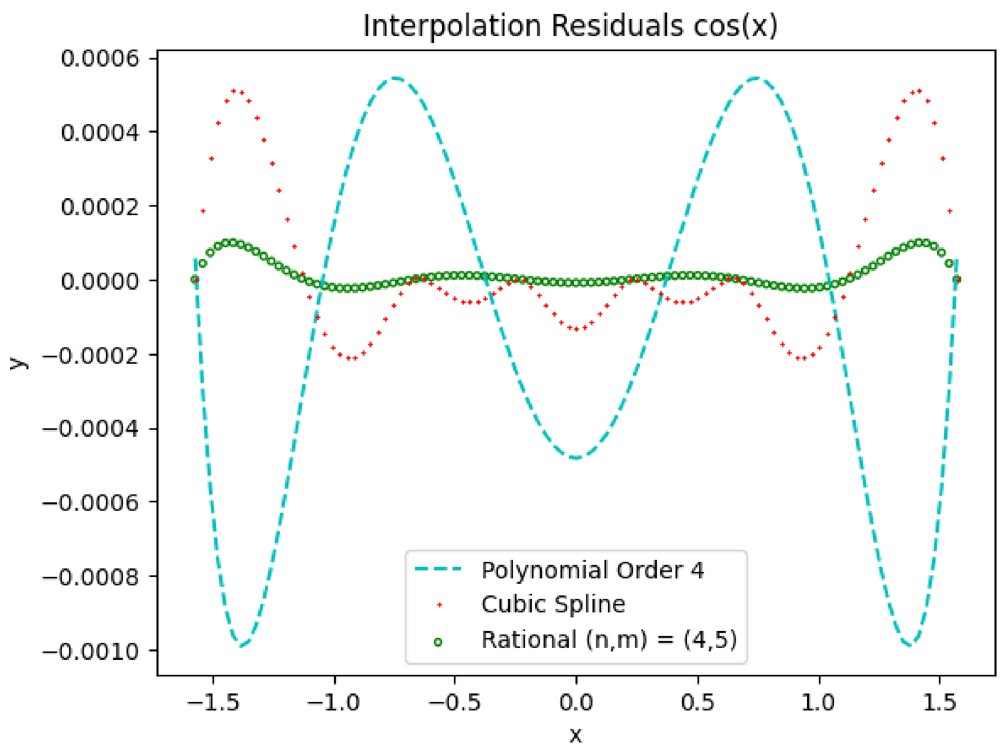

Figure 10. Residuals of cos(x) without pseudoinverse operation – high order rational

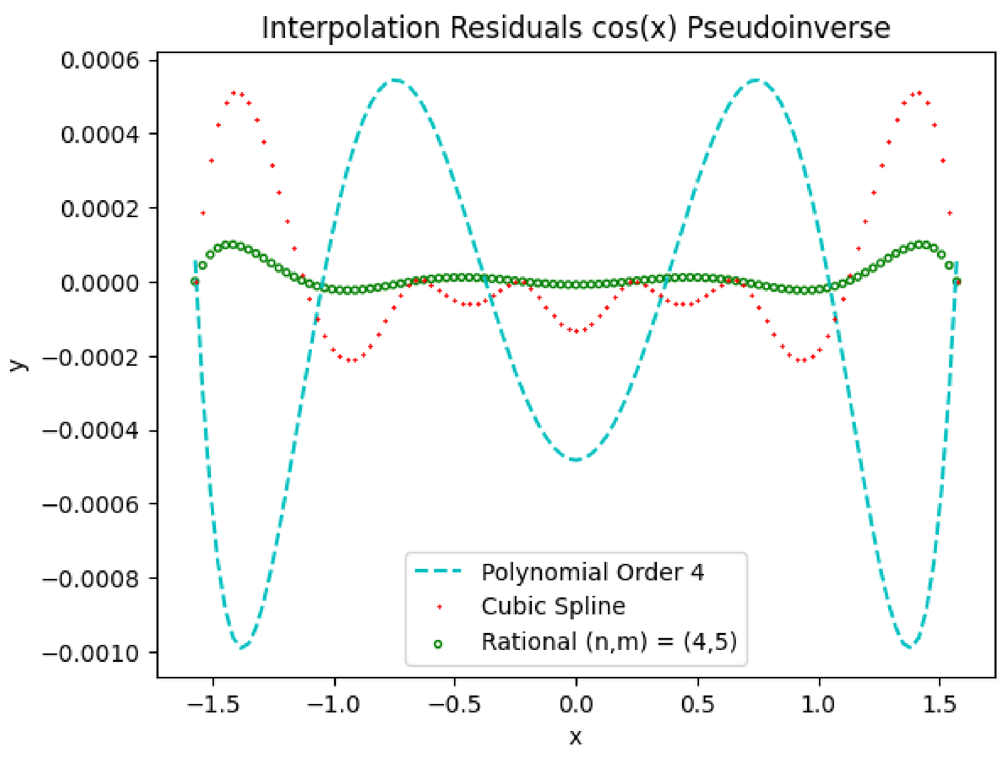

Figure 11. Residuals of cos(x) with pseudoinverse operation – high order rational

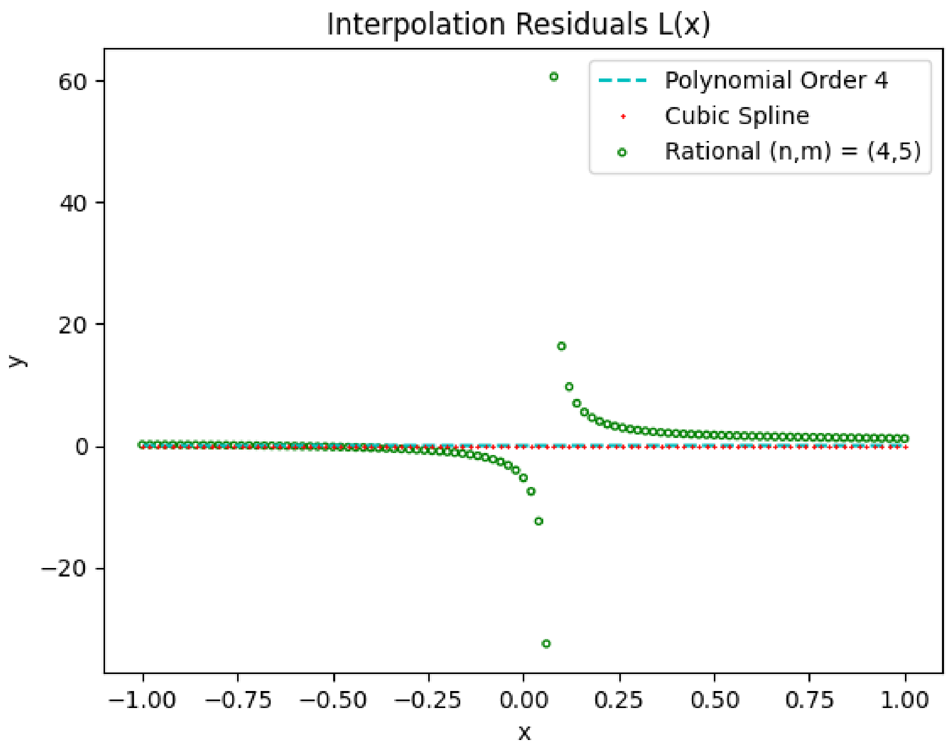

Figure 12. Residuals of L(x) without pseudoinverse operation – high order rational

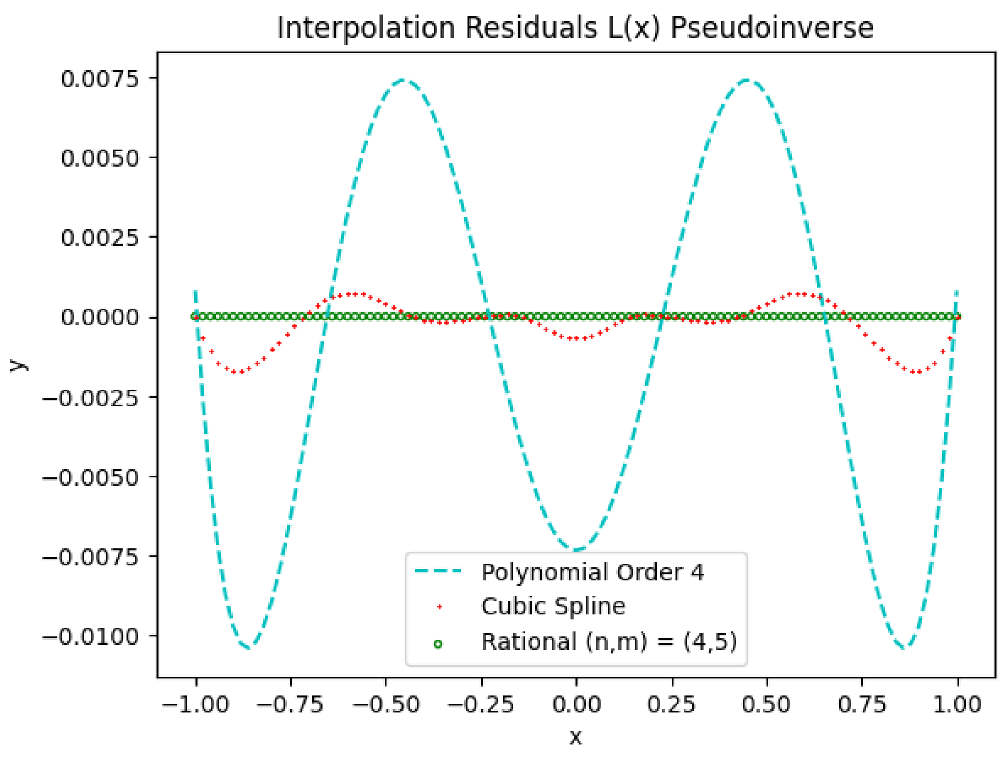

Figure 13. Residuals of L(x) with pseudoinverse operation – high order rational

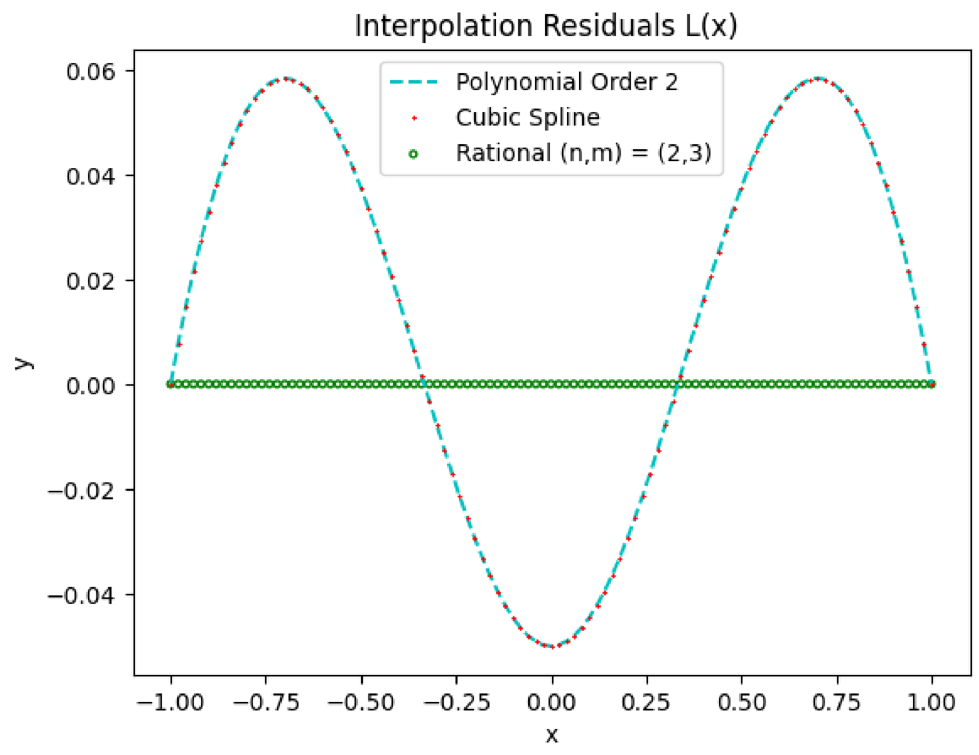

Figure 14. Residuals of L(x) without pseudoinverse operation – low order rational

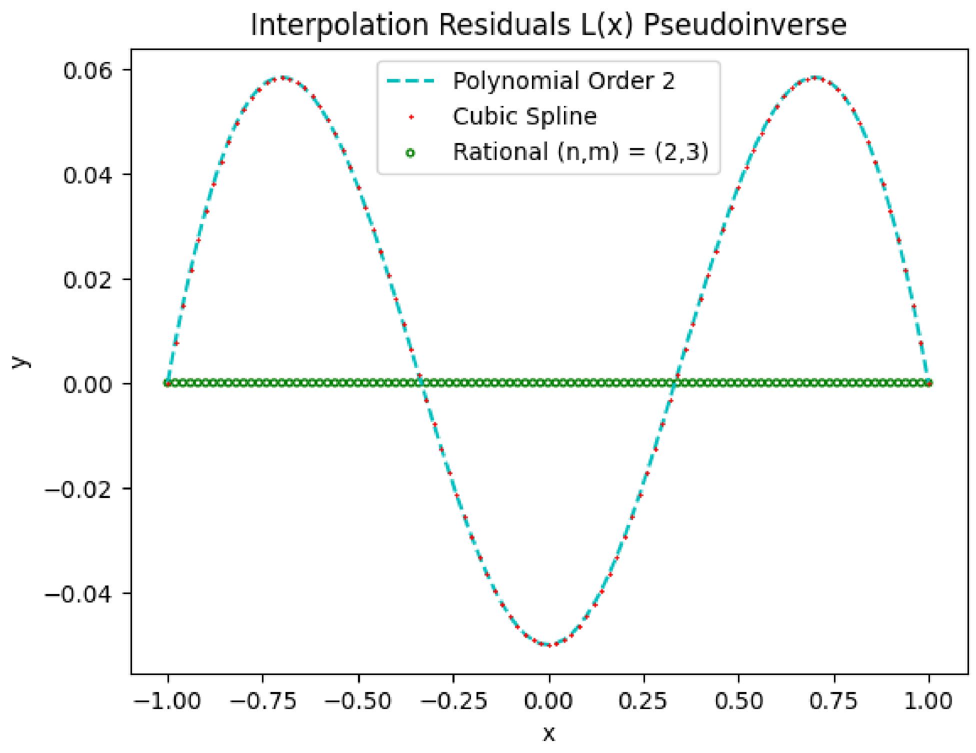

Figure 15. Residuals of L(x) with pseudoinverse operation – low order rational

References

Chapra, S. C., & Canale, R. P. (2011). *Numerical methods for engineers* (Vol. 1221). New York: Mcgraw-Hill.

Appendix A: Python Code

Jupyter notebook with relevant Python code and outputs is available at:

https://github.com/ck22512/comp_phys/blob/main/Assignment1/Assignmentv1v8submit.ipynb