1 Instructions

Perform the numerical and analytical exercises below. Write up your results (preferably using LaTeX or a Jupyter notebook) in a document. Include all relevant figures and explanations. Include your codes at the end of the document.

2 Exercises

1. Typically, interpolation is used when one is given a table of values of a function with spacing between the values small enough that low-order polynomial passing through a few nearby points is sufficient to obtain accurate values of the function between points on the table. Here, you are asked to evaluate the interpolation error of Runge's function
$$f(x) = \frac{1}{1+25x^2}$$
as function of the number of points in the table. That is, evaluate Runge function on the set of points
$$x_j = -1 + \frac{2j}{N-1}, \quad j = 0, 1, \cdots N-1.$$
Then use the appropriate four-point stencils to interpolate Runge's function to a dense set of points in $-1 \le x \le 1$. Find the maximum error in the interpolation as a function of N. Note that for every point that you want to interpolate, you will have to find the four closest points in the table.

2. Consider the Chebyshev T_n expansion of Runge's function
$$f(x) = \frac{1}{1+25x^2} \approx \sum_{n=0}^{N-1} c_n T_n(x)$$
on the interval $-1 \le x \le 1$. Evaluate the approximate L^∞ norm of the error as a function of the maximum order in the Chebyshev expansion. Plot the data on an appropriately scaled graph (i.e. choose one of log-log, log-linear, linear-log, etc). Using the same coefficients found in the first part of this problem, evaluate the derivative of $f(x)$ and the L^∞ norm of the error. Note you are not being ask to expand $f'(x)$ as a series in T_n, but rather to differentiate the previously obtained series for $f(x)$. You may find the following identities useful [1]:
$$T_n(x) = \cos\left(n\cos^{-1} x\right), \tag{1}$$
$$\frac{dTn(x)}{dx} = \frac{n\sin\left(n\cos^{-1} x\right)}{\sqrt{1-x^2}} = nU_{n-1}(x), \tag{2}$$
$$\tag{3}$$
the roots of $T_N(x)$ are given by
$$x_j = \cos\left(\frac{\pi(2j+1)}{2N}\right), \quad j = 0, 1, 2, \cdots, N-1$$
and
$$\sum_{j=0}^{N-1} T_m(x_j)T_n(x_j) = \begin{cases} 0 & i \ne j \\ N/2 & i = j \ne 0 \\ N & i = j = 0 \end{cases}$$

3. Develop and test a code (Python or C/C++) to compute derivatives using equal-spaced centered finite differencing. Compare the minimum error in the derivatives as a function the order of the finite-difference stencil. Try to reach at least eight-order accuracy. You may use Mathematica (or sympy) to generate your stencils. As a test, calculate the derivative of
$$f(x) = \sqrt{1-x^2}$$
at $x = 0.5$.

[1] `https://en.wikipedia.org/wiki/Chebyshev_polynomials`