Apr. 21, 2025 (Due: 08:00 Apr. 27, 2025)

1. Use a linear combination of $f(t)$, $f(t+h)$, $f(t+2h)$ to approximate $f'(t)$ (as accurately as you can). Give an error estimate by taking into account both truncation and rounding errors.

2. Use a linear combination of nine function values $f(x+ih, y+jh)$ (for i, $j \in \{-1,0,1\}$) to approximate

$$\frac{\partial^2}{\partial x^2} f(x,y) + \frac{\partial^2}{\partial y^2} f(x,y)$$

(as accurately as you can). Estimate the truncation error.

3. Use Richardson extrapolation to estimate the derivative of $f(x) = x^3 \mathrm{e}^x$ at $x = 1$. Keep a record for intermediate results. What happens if you iterate for many steps?

4. Use a cubic spline function $s(x)$ to approximate $f(x) = \mathrm{e}^x + \ln x$ over $[1,4]$. Plot the first and second derivatives, as well as the approximation errors. You are encouraged to try different step sizes and observe the behavior of the error with respect to the step size.

(optional) Use contour integration to compute the first and second derivatives.

5. (optional) An ancient way of computing π can be interpreted in modern terms as $\pi \approx n \sin(\pi/n)$, where n is typically chosen as $n = 3 \cdot 2^k$ for $k \in \mathbb{N}$. Unfortunately, the convergence of such an approximation is very slow, and the calculation is expensive—it involves a lot of square roots (because ancient mathematicians had to use geometry instead of Taylor series to compute $\sin(\pi/n)$). Some scholars believe that the Chinese mathematician Zu Chongzhi discovered a way to largely accelerate the calculation using some sort of extrapolation.

Use Richardson extrapolation to calculate π to seven correct digits after the decimal point, based on the asymptotic expansion

$$n \sin \frac{\pi}{n} = \pi - \frac{\pi^3}{3!\, n^2} + \frac{\pi^5}{5!\, n^4} - \cdots .$$

What is the largest value of n in your calculation?

For simplicity, you may call library functions to perform calculations such as `sin(pi/n)` directly in your program.

6. (optional) Read the paper "Inverse Problems Light: Numerical Differentiation" by M. Hanke and O. Scherzer in the reading folder. Implement the numerical differentiation algorithm in this paper.