COL 726 Homework 5

9–16 August, 2020

1. Suppose f is a smooth function and p is its interpolating polynomial on equally spaced points $t_1, t_2, \ldots, t_n$. Denote $h = t_{i+1} - t_i$ and $M = \max_{t \in [t_1, t_n]} |f^{(n)}(t)|$. Using the fact that $f(t) - p(t) = \frac{1}{n!} f^{(n)}(\theta) \cdot (t - t_1)(t - t_2) \cdots (t - t_n)$ for some $\theta \in [t_1, t_n]$, prove that

$$|f(t) - p(t)| \leq \frac{Mh^n}{4\binom{n}{i}} \quad \text{when } t_i < t < t_{i+1}.$$

2. In numerical integration, we usually fix the quadrature nodes and determining the corresponding weights. One can also take the opposite approach: fix the weights and solve for the node locations. For example, we could ask that all nodes have equal weight, so that the quadrature rule becomes simply $Q_n(f) = w \sum_{i=1}^{n} f(x_i)$. Find such a quadrature rule with three nodes on the interval $[-1, 1]$. What is the degree of the resulting rule?

   Hint: If solving the equations is giving you extremely complicated expressions, try solving them in a different order.

3. One way to avoid oscillations in interpolation is to divide the domain into pieces and fit low-order polynomials on them, e.g. a separate cubic polynomial on each interval $[t_i, t_{i+1}]$.

   (a) Suppose you know both the function's values $y_i = f(t_i)$ and derivatives $m_i = f'(t_i)$ at all interpolation points. Give an explicit formula for the desired cubic polynomial on $[t_i, t_{i+1}]$, i.e. which matches the function values and derivatives at t_i and t_{i+1}. Using this formula, implement a function `ynew = piecewiseCubic(t, y, m, tnew)` to perform piecewise cubic interpolation, given the vectors $\mathbf{t} = [t_1, \ldots, t_n]$, $\mathbf{y} = [y_1, \ldots, y_n]$, $\mathbf{m} = [m_1, \ldots, m_n]$ and the evaluation point t_{new}.

   (b) Now suppose you don't know the derivatives m_i. Derive a formula to estimate them from the three points (t_{i-1}, y_{i-1}), (t_i, y_i), (t_{i+1}, y_{i+1}) as accurately as possible. (Don't assume the points are equally spaced.) Also give modified three-point formulas that work for the first and last points. Using these, implement a function `m = estimateDerivatives(t, y)`, which returns a vector of derivatives that can be passed to `piecewiseCubic`.