Problem Sheet 2

discussion: week of Monday, 24.10.2022

2.1. Use quadratic interpolation in the points $(0, 1), (1, e), (2, e^2)$ to approximate the function $f(x) = \exp(x)$ on $[0, 2]$. What is the error at $x = 1.2$? Compute an upper estimate for the pointwise error

$$|f(x) - p(x)|$$

for any fixed point $x \in [0, 2]$. Also compute an upper estimate for the maximal error, i.e.

$$||f - p||_{\infty,[0,2]}.$$

2.2. We aim to approximate a function f on the interval $[a, b]$ by a *piecewise* polynomial of degree n. Proceed as follows: partition $[a, b]$ in N subintervals $[t_j, t_{j+1}]$, $j = 0, \ldots, N-1$, of length $h = (b - a)/N$ with $t_j = a + jh$. On each subinterval $[t_j, t_{j+1}]$ select the interpolation points $x_{i,j} := t_j + \frac{1}{n} ih$, $i = 0, \ldots, n$, and approximate f on $[t_j, t_{j+1}]$ by the polynomial that interpolates in the points $x_{i,j}$, $i = 0, \ldots, n$. In this way, one obtains a function p that is a polyomial of degree n on each subinterval. Show:

$$\|f - p\|_{\infty,[a,b]} \le \frac{1}{(n+1)!} h^{n+1} \|f^{(n+1)}\|_{\infty,[a,b]}.$$

Sketch the function p for $n = 1$.

2.3. **a)** Write a `matlab` or `python` program that realizes the Neville scheme. Input are the vectors **x**, **f** (knots and data values) of length $n+1$. Output is an array N (size $(n+1)\times(n+1)$) that contains the columns of the Neville scheme.

Remark: use the algorithm described in the lecture; you could check, if you wish, your algorithm with your code of Problem 1.3.

b) Let, for a function f and a point x_0,

$$D_{sym}(h) := \frac{f(x_0 + h) - f(x_0 - h)}{2h}$$

be the symmetric difference quotient.

Use your program of a) to generate the first 4 columns of the Neville scheme of the extrapolation of $D_{sym}(0)$ for the function $f(x) = \tan(x)$ and $x_0 = \pi/4$ (*note:* $f'(x) = 1/\cos^2(x)$ so that $f'(\pi/4) = 2$) and $h_i = 2^{-i}$, $i = 0, \ldots, n_{max}$ with $n_{max} = 10$. Plot in a `loglog` plot the error versus h for these 4 columns, i.e., plot for $m \in \{0, 1, 2, 3\}$ the values $\texttt{abs}(N([1 : n_{max} + 1 - m], m) - 2)$ versus $h([1 : n_{max} + 1 - m])$. Include in the plot the auxiliary lines $h \mapsto h^2$, $h \mapsto h^3$, $h \mapsto h^4$. What convergence rates do you observe?

2.4. ("harmonic series") The goal is the efficient evaluation/approximation of

$$S(N) := \sum_{n=1}^{N} \frac{1}{n}$$

for large N. We use the fact that $S(N)$ can be written as

$$S(N) = \ln N + a_0 + \frac{a_1}{N} + \frac{a_2}{N^2} + \cdots \tag{1}$$

Determine the coefficients a_0, a_1, a_2 as follows: 1) Write a routine to evaluate $S(N)$. 2) Set up a linear system of equations for the coefficients a_0, a_1, a_2 that is obtained for $N = 10$, 100, 1000. (The terms $+\cdots$ in (1) are simply ignored). Solve for the coefficients (in `matlab` this is achieved with \, in `python` this can be done with `numpy.linalg.solve`).

What is the error of your approximation for $N = 10^6$ and $N = 10^8$? What is the run time of your approximation for $N = 10^8$ and $N = 10^9$? What is the run time for the evaluation of $S(10^8)$ and $S(10^9)$ on your computer? (Use `tic`, `toc` or `time.time()`)

2.5. The goal is to evaluate numerically the series $\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$. Modify the way you proceeded in Problem 2.4 appropriately. To that end, introduce the function

$$S'(N) := \sum_{n=1}^{N} \frac{1}{n^2}$$

and approximate

$$S'(N) = a_0 + \frac{a_1}{N} + \frac{a_2}{N^2} + \cdots$$

Use $N = 100$, 1000, 10000. What is the error $a_0 - \pi^2/6$?