Problem Sheet 1

discussion: week of Monday, 17.10.2022

1.1. Consider the approximation of the integral $\int_0^1 f(x)$ by the trapezoidal rule

$$\int_0^1 f(x)\,dx \approx T(h) := \sum_{i=0}^{N-1} \frac{h}{2}\left(f(x_i) + f(x_{i+1})\right), \qquad x_i = ih, \qquad h = \frac{1}{N}$$

as well as the box rule

$$\int_0^1 f(x)\,dx \approx R(h) := \sum_{i=0}^{N-1} h f(x_i), \qquad x_i = ih, \qquad h = \frac{1}{N}.$$

a) Write one program that, given N and a function f, returns $T(h)$ and one that returns $R(h)$ with $h = 1/N$.

b) For $f(x) = e^x$ and $N(i) = 2^i$, $i = 1, \ldots, 10$, set $h(i) = 1/N(i)$. For these 10 values of h, compute the actual errors $\texttt{err}(i) = \int_0^1 e^x\,dx - T(h(i)) = e - 1 - T(h(i))$. Plot the (absolute value) of the true error in a "log-log" plot. That is, in `matlab` notation with the vectors h and `err` (both of length 10)

$$\texttt{loglog}(h, \texttt{abs}(\texttt{err}))$$

Does the true error behave like Ch^2?

Do the same thing for the box rule as well. What error behaviour do you see?

Remark: In `python` loglog plotting is realized with `matplotlib.pyplot.loglog` and the array limits have to be adapted since arrays start at 0 in `python`.

1.2. Again, consider the approximation of the integral $\int_0^1 f(x)$ by the trapezoidal rule $T(h)$ from exercise 1.1. The aim is to design an error estimator as it was done for the "box scheme" in the lecture.

a) For the error estimator, make the ansatz

$$\int_0^1 f(x)\,dx - T(h) \stackrel{!}{=} Ch^2$$

and compute C from (calculated) $T(h)$ and $T(h/2)$. The error estimator is then $E(h) = Ch^2$.

b) Modify your program from exercise 1.1(a) such that it also computes the error estimator $E(h)$. For the test-case in exercise 1.1(b) do also compute the ratios $\texttt{r}(i) := \texttt{err}(i)/E(h(i))$ of the true error to estimated error. Plot the ratio `r` in a semilogarithmic plot versus h. That is, in `matlab` notation with the vectors h and `r` (both of length 10)

$$\texttt{semilogx}(h, \texttt{r})$$

1.3. Let the data $(0, 0)$, $(1, 2)$, $(4, 8)$ be given.

a) Write down the (quadratic) interpolating polynomial p in the Lagrange form, i.e., $p(x) = \sum_{i=0}^{2} f_i \ell_i(x)$.

b) The quadratic interpolating polynomial $p(x) = a_0 + a_1 x + a_2 x^2$ can also be determined by solving a linear system of equations for the coefficients a_0, a_1, a_2. Formulate the system and solve for the a_i.

c) Check your approximation $p(2)$ using the `matlab` or `numpy` routines $\texttt{polyfit}(\mathbf{x}, \mathbf{f}, n)$ and `polyval`. Note that `polyfit` also returns the coefficients a_i of part c). Compare.

Note: The use of `polyfit` and `polyval` in the context of polynomial interpolation should rather be viewed as a "quick and dirty" method since `polyfit` is based on polynomial approximation in the monomial basis $\{1, x, x^2, \ldots, x^n\}$ and therefore numerically unsuitable for large/largish n.

1.4. **a)** Write a routine that has as input the vectors $\mathbf{x}$ and $\mathbf{f}$ of knots and data (both vectors have the length $n+1$) and returns the value of the interpolating polynomial at the point 0, i.e., $p(0)$ and the polynomial p of degree n satisfies $p(\mathbf{x}_i) = \mathbf{f}_i$ for all i. To simplify matters, you may use `polyfit` and `polyval`.

b) Define the values $h_i := 2^{-i}$, $i = 0, 1, \ldots$. Based on a), write a function whose input is a function handle f and integers $m, n \in \mathbb{N}_0$. The output is a matrix $N \in \mathbb{R}^{(m+1)\times(n+1)}$ with entries $N_{j,k} := p_{j,k}(0)$, where $j = 0, \ldots, m$, $k = 0, \ldots, n$ and $p_{j,k}$ is the polynomial of degree k that interpolates the data $(h_{j+\ell}, f(h_{i+\ell}))$, $\ell = 0, \ldots, k$.

Remark: This matrix N is the so-called *Neville*-scheme for interpolation although it is usually set up differently. (This will be done later in class.)

c) Apply your routine of b) to the function

$$f(h) = \frac{e^h - 1}{h}$$

and select $m = 10$ and $n = 2$. You expect the columns of N to converge to $\lim_{h\to 0} f(h) = 1$. To check that, plot (in `matlab` notation)

$\texttt{loglog}(h(1:m{+}1), \texttt{abs}(N(:,1){-}1), \quad h(1:m{+}1), \texttt{abs}(N(:,2){-}1), \quad h(1:m{+}1), \texttt{abs}(N(:,3){-}1))$

What convergence behavior do you observe for these 3 graphs?

1.5. A common way to approximate a function by a polynomial is to use Taylor polynomials around a point x_0 given as

$$f(x) \simeq \sum_{j=0}^{n} \frac{f^{(j)}(x_0)}{j!}(x - x_0)^j = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2 + \ldots$$

Compute the Taylor polynomial of degree $n = 2$ for the function $f = \sin(x)$ at $x_0 = 0$ and at $x_0 = \frac{\pi}{2}$. Compare it with the interpolating polynomial of order 2 from the lecture, what do you observe?