Valentin Helml
Dirk Praetorius
Julian Streitberger

Summer term 2022
March 11 2022

Exercise
Scientific programming in mathematics

Series 2

Exercise 2.1. We call a matrix $A \in \mathbb{C}^{n\times n}$ strictly diagonal dominant if

$$\sum_{\substack{k=1\\k\neq j}}^{n} |A_{jk}| < |A_{jj}| \quad \text{for all } j \in \{1,\dots,n\}.$$

Write a MATLAB function `dominant` which returns `true` if A is diagonal dominant, and `false` otherwise. Think about how you can test your code! What are suitable test-examples?

Exercise 2.2. The sieve of Eratosthenes is an algorithm to compute all prime numbers up to any given limit $n \in \mathbb{N}$. The algorithm initializes a vector `primeNumbers = 2:n`. Then, the algorithm starts with deleting every multiple of $p := 2$ from `primeNumbers`. In the successive step, find the smallest number in `primeNumbers` greater than p. If there is no such number, stop the algorithm, otherwise set this new number to p and proceed analogously. Write a MATLAB function `eratosthenes` that takes a given number n as input and returns the vector of all prime numbers smaller than or equal to n.

Exercise 2.3. For given values $x_1 < \dots < x_n$ and function values $y_j \in \mathbb{R}$, $j = 1,\dots,n$, linear algebra asserts the existence of a unique polynomial $p(t) = \sum_{j=1}^{n} a_j t^{j-1}$ of degree $n-1$ with $p(x_j) = y_j$ for all $j = 1,\dots,n$ as shown in the lecture. For fixed $t \in \mathbb{R}$, we aim to evaluate $p(t)$. One can compute $p(t)$ with the Neville scheme without the calculation of the coefficient vector $a \in \mathbb{R}^n$. To this end, define for $j, m \in \mathbb{N}$ with $m \geq 2$ and $j + m \leq n+1$, the values

$$\begin{aligned} p_{j,1} &:= y_j, \\ p_{j,m} &:= \frac{(t - x_j)p_{j+1,m-1} - (t - x_{j+m-1})p_{j,m-1}}{x_{j+m-1} - x_j}. \end{aligned}$$

Then, it holds that $p(t) = p_{1,n}$. Write a MATLAB function `neville` which takes $t \in \mathbb{R}$ and vectors $x, y \in \mathbb{R}^n$ as input and calculates $p(t)$ by means of the Neville algorithm. Consider the following schematic procedure.

$$\begin{array}{ccccccccccccc}
y_1 & = & p_{1,1} & \longrightarrow & p_{1,2} & \longrightarrow & p_{1,3} & \longrightarrow & \cdots & \longrightarrow & p_{1,n} & = & p(t) \\
 & & & \nearrow & & \nearrow & & & & \nearrow & & & \\
y_2 & = & p_{2,1} & \longrightarrow & p_{2,2} & & & & & & & & \\
 & & & \nearrow & & & & \nearrow & & & & & \\
y_3 & = & p_{3,1} & \longrightarrow & \vdots & & & & & & & & \\
\vdots & & \vdots & & \vdots & \nearrow & & & & & & & \\
y_{n-1} & = & p_{n-1,1} & \longrightarrow & p_{n-1,2} & & & & & & & & \\
 & & & \nearrow & & & & & & & & & \\
y_n & = & p_{n,1} & & & & & & & & & &
\end{array} \tag{1}$$

Test your code with a known polynomial p and function values $y_j = p(x_j)$.

Exercise 2.4. On a first glance, the Neville scheme (1) computes a matrix $P \in \mathbb{R}^{n\times n}$ with $P_{1,n} = p(t)$. However, it is possible to avoid the storage of P but to overwrite the vector y instead. Write a MATLAB function `neville2` which avoids auxiliary storage.

Exercise 2.5. Consider a sufficiently smooth function $f \in C^k([x_0 - \delta, x_0 + \delta])$ for $x_0 \in \mathbb{R}$, $\delta > 0$ and $k \in \mathbb{N}$. To approximate the derivative $f'(x_0)$, one can use Richardson extrapolation g_n of the one-sided difference quotient D^1 defined by

$$D^1(f,h) := \frac{f(x_0 + h) - f(x_0)}{h}.$$

Considering $h_k = \rho^k$ and $0 < \rho < 1$, one employs the Neville scheme and defines $g_n := p(0)$, when p is the unique polynomial of degree $n-1$ with $p(h_k) = D^1(f, h_k)$ for $k = 1, \ldots, n$. Implement the Richardson extrapolation and compare for different values $0 < \rho < 1$ the behaviour of the error $|g_n - f'(x_0)|$ for the function $f(x) = \exp(x)$ versus the naive approach with $f'(x_0) \approx D^1(f, h_n)$.

Exercise 2.6. The Newton scheme is an algorithm to compute the zero $x^\star \in \mathbb{R}$ of a function $f : \mathbb{R} \to \mathbb{R}$, i.e., $f(x^\star) = 0$. Starting with an initial guess $x_0 \in \mathbb{R}$, the sequence $(x_k)_{k\in\mathbb{N}}$ is inductively defined via

$$x_{k+1} := x_k - \frac{f(x_k)}{f'(x_k)} \quad \text{for } k \in \mathbb{N}_0.$$

Consider the parameter $\tau_{\text{abs}} > 0$ (absolute tolerance), τ_{rel} (relative tolerance), and $N_{\max}$ (maximal iteration count). To stop the Newton iteration, use the following stopping criterion: Calculate the sequence elements $x_1, \ldots, x_\ell$ until the residual $|f(x_\ell)|$ satisfies that

$$|f(x_\ell)| \le \max\{\tau_{\text{abs}}, \tau_{\text{rel}}|f(x_0)|\}.$$

Then calculate the elements $x_{\ell+1}, \ldots, x_n$ until

$$|f(x_n)| \le \tau_{\text{abs}} \text{ and } |x_n - x_{n-1}| \le \tau_{\text{abs}} \max\{1, |x_n|\}.$$

Then, the Newton iteration converges and x_n is the sought approximation of $x^\star$. If the maximal iteration count $N_{\max}$ is reached before the aforementioned criterion is fulfilled, then the Newton iterion has not converged and should be terminated with an error message (see `help error`). Write a MATLAB function `newton`, which realizes this approach. Test your function with $f(x) = x^2 + \exp(x) - 2$ on the interval $[0, 1]$.

Exercise 2.7. Aitken's Δ^2-method is a method for convergence acceleration of sequences. For an injective sequence $(x_n)_{n\in\mathbb{N}}$ with $x = \lim_{n\to\infty} x_n$ one defines

$$y_n := x_n - \frac{(x_{n+1} - x_n)^2}{x_{n+2} - 2x_{n+1} + x_n}. \tag{2}$$

Under certain assumptions for the sequence $(x_n)_{n\in\mathbb{N}}$, it then holds that

$$\lim_{n\to\infty} \frac{x - y_n}{x - x_n} = 0,$$

i.e., the sequence $(y_n)_{n\in\mathbb{N}}$ converges faster to x than $(x_n)_{n\in\mathbb{N}}$. Write a MATLAB function `aitken` which takes a vector $x \in \mathbb{R}^N$ and returns a vector $y \in \mathbb{R}^{N-2}$. Use suitable loops. Think about how you can test your code! What happens for a geometric sequence $x_n := q^n$ with $0 < q < 1$?

Exercise 2.8. Consider a sufficiently smooth function $f \in C^k([x_0 - \delta, x_0 + \delta])$ for $x_0 \in \mathbb{R}$, $\delta > 0$ and $k \in \mathbb{N}$. From the lecture, you know that the one-sided difference quotient D_h^1 from Exercise 2.5 and the central difference quotient D_h^2 defined for $0 < h < \delta$ by

$$D^2(f, h) := \frac{f(x_0 + h) - f(x_0 - h)}{2h}$$

provide approximations of the derivative $f'(x_0)$. Consider the function f with $f(x) = \exp(x)$ and $x_0 = 0$ as well as $h_j = h_0 \rho^j$ with $h_0 \in \mathbb{R}$ and $0 < \rho < 1$. Compare the convergence rates of $D^1(f, h_j)$, $D^2(f, h_j)$ and the accelerated sequences obtained by the application of Aitken's Δ^2-method to the difference quotients in a log-log plot.