Dirk Praetorius
Winfried Auzinger

Sommersemester 2017
15.03.2017

Übungen zur Vorlesung
Computermathematik

Serie 2

Aufgabe 2.1. Write a function which calculates and returns for a vector $x \in \mathbb{C}^n$ and some $1 \le p < \infty$ the ℓ_p-norm

$$\|x\|_p := \Big(\sum_{j=1}^n |x_j|^p\Big)^{1/p}.$$

The function has to be implemented in two different ways: First, avoid loops and use appropriate vector functions and arithmetic instead; second, use loops and scalar arithmetic.

Aufgabe 2.2. Write a function `tensor` which returns for $n \in \mathbb{N}$ the chessboard-tensor $B \in \mathbb{N}^{n\times n\times n}$ with

$$B_{jk\ell} = \begin{cases} 0 & \text{if } j+k+\ell \text{ even} \\ 1 & \text{if } j+k+\ell \text{ odd} \end{cases}$$

The function has to be implemented in two different ways: First, avoid loops and use appropriate vector functions and arithmetic instead; second, use loops and scalar arithmetic.

Aufgabe 2.3. Let $p(x) = \sum_{j=0}^n a_j x^j$ be a polynomial with coefficient vector $a \in \mathbb{C}^{n+1}$. Write a MATLAB-function which takes a and returns the coefficient vector of the derivative p'.
The function has to be implemented in two different ways: First, avoid loops and use appropriate vector functions and arithmetic instead; second, use loops and scalar arithmetic. Your function should work for column and row vectors a and should always return a column vector; see, e.g., `help reshape` Think about how you can test your code! What are suitable test-examples?

Aufgabe 2.4. Write a MATLAB-function which calculates for given polynomials $p(x)$ and $q(x)$ the result $r(x) = p(x) + q(x)$ and returns the coefficient vector $r \in \mathbb{C}^{n+1}$. $r(x)$ should be a polynomial of minimal degree, i.e., for the leading coefficient there holds $r_{n+1} \neq 0$. The function has to be implemented in two different ways: First, avoid loops and use appropriate vector functions and arithmetic instead; second, use loops and scalar arithmetic. Think about how you can test your code! What are suitable test-examples?

Aufgabe 2.5. Let $p(x) = \sum_{j=0}^n a_j x^j$ be a polynomial with coefficient vector $a \in \mathbb{C}^{n+1}$. Let $x = (x_{jk}) \in \mathbb{C}^{M\times N}$ be a matrix of evaluation points. Write a MATLAB-function which calculates and returns the evaluation matrix $\big(p(x_{jk})\big) \in \mathbb{C}^{M\times N}$. Your function should work for column and row vectors a. The function has to be implemented in two different ways: First, avoid loops and use appropriate vector functions and arithmetic instead; second, use loops and scalar arithmetic. Think about how you can test your code! What are suitable test-examples?
Hint: You can use `reshape` to reduce the case of a matrix x to the case of a vector. Note that the evaluation points can be complex-valued.

Aufgabe 2.6. MATLAB offers a variety of ways to measure the run-time of a function resp. calculation. One easy way is the function `tic-toc`. In this case, `tic` starts the timing and `t=toc` saves the passed time in `t`; see `help tic` resp. `help toc`. Write a MATLAB-function which measures the run-time of at least 2 of the previous exercises. For each exercise, use different input-sizes to compare the run-time of the two different implementations (loops vs. vector arithmetic). Use `fprinft` to display your results.

Aufgabe 2.7. The integral $\int_a^b f\,dx$ of a continuous function $f : [a,b] \to \mathbb{R}$ can be approximated by so called quadrature formulas

$$\int_a^b f\,dx \approx \sum_{j=1}^{n} \omega_j f(x_j),$$

where one fixes some vector $x \in [a,b]^n$ with $x_1 < \cdots < x_n$ and approximates the function f by some polynomial $p(x) = \sum_{j=1}^{n} a_j x^{j-1}$ of degree $\leq n-1$ with $p(x_j) = f(x_j)$ for all $j = 1, \ldots, n$. The weights ω_j can be calculated by the assumption

$$\int_a^b q\,dx = \sum_{j=1}^{n} \omega_j q(x_j) \quad \text{for all polynomials } q \text{ of degree } \leq n-1.$$

This is equivalent to the solution of the linear system

$$\frac{b^{k+1}}{k+1} - \frac{a^{k+1}}{k+1} = \int_a^b x^k\,dx = \sum_{j=1}^{n} \omega_j x_j^k \quad \text{für alle } k = 0, \ldots, n-1.$$

Why is this the case? Write a function `integrate` which takes the (column or row) vector $x \in [a,b]^n$ and the function value vector $f(x)$, and which returns the approximated value of the integral. Therefore, build the linear system as efficiently as possible and solve it with the backslash-operator. With the aid of the resulting vector $\omega \in \mathbb{R}^n$ one obtains the approximated integral as scalar product with the vector $f(x)$. Think about how you can test your code! What are suitable test-examples? Avoid loops and use appropriate vector functions and arithmetic instead.

Aufgabe 2.8. Write a MATLAB function `saveMatrix` which takes a matrix $A \in \mathbb{R}^{M \times N}$ and writes it into an ASCII file `matrix.dat` via `fprintf` (see also `help fopen`). Use `%1.16e` for `fprintf` to write the matrix coefficients! (Why does this make sense?) Optionally, the function takes a string `name` and writes the matrix to the ASCII file `name.dat`. To verify your code, write a MATLAB script which creates a random matrix $A \in \mathbb{R}^{M \times N}$ and writes it to an ASCII file `A.dat`. Load the matrix via `B = load('A.dat')` and check whether A and B coincide.