Homework - Serie 09

Kevin Sturm
Python 3

Test your code with examples! Below the word function refers to python function. Use numpy arrays!

Problem 1.
Write a python script which generates for given $n \in \mathbf{N}$ the following block diagonal matrices $A_1 \in \mathbf{R}^{2n\times 2n}$ and $A_2 \in \mathbf{R}^{3n\times 2n}$:

$$A_1 := \begin{pmatrix} 1 & 1 & & & & & \\ 1 & 1 & & & & & \\ & & 1 & 1 & & & \\ & & 1 & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & 1 \\ & & & & & 1 & 1 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 1 & & & & & \\ 1 & 1 & & & & & \\ 1 & 1 & & & & & \\ & & 1 & 1 & & & \\ & & 1 & 1 & & & \\ & & 1 & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & 1 \\ & & & & & 1 & 1 \\ & & & & & 1 & 1 \end{pmatrix}.$$

Avoid loops!

Problem 2.
Write a python script which generates, for odd $n \geq 5$ three different matrices $A \in \mathbf{R}^{n\times n}$. For $n = 5$ the matrices look like

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & & & & 1 \\ 1 & & & & 1 \\ 1 & & & & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \quad \begin{pmatrix} & & 1 & & \\ & & 1 & & \\ & & 1 & & \\ & & 1 & & \\ & & 1 & & \end{pmatrix}, \quad \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ & & & & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & & & & \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix},$$

where all entries which are not shown have to be initialized with 0. For $n > 5$ the matrices look accordingly. Avoid loops! Instead, use matrix functions and matrix indexing!

Problem 3.

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

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

Avoid loops!

Problem 4.
Let $L = (\ell_{ij})_{i,j=1,\dots,n} \in \mathbf{R}^{n\times n}$ be a regular (i.e. L has full rank) and lower triangular matrix (i.e. $\ell_{ij} = 0$ for $i < j$). We write L in the block form

$$L = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix}$$

with $L_{11} \in \mathbf{R}^{p\times p}$, $L_{21} \in \mathbf{R}^{q\times p}$ and $L_{22} \in \mathbf{R}^{q\times q}$,where $p + q = n$.

(a) Show that $\det(L) = \prod_{j=1}^{n} \ell_{jj}$.

(b) Show that

$$L^{-1} = \begin{pmatrix} L_{11}^{-1} & 0 \\ -L_{22}^{-1} L_{21} L_{11}^{-1} & L_{22}^{-1} \end{pmatrix}.$$

(c) Write a function `invertL`, which L^{-1} recursively calculates the inverse as described.You can test your function with the help of the function `np.linalg.inv`. Avoid loops.

Problem 5. Write a python script which determines the maximum $x_{\max} := \max_{1\le i\le n} x_i$ of a vector $x \in \mathbf{R}^n$. Further, all entries x_i with $|x_i| \ge x_{\max}$ should be replaced by $\operatorname{sign}(x_i)x_{\max}$. Avoid loops and use only appropriate matrix functions and indexing instead.

Problem 6.
Write a python script which displays for a given vector $x \in \mathbf{C}^N$ the trimmed vector $x' \in \mathbf{C}^{N-2}$, where the entries with highest resp. lowest absolute value are cut out of x. The remaining vector should be sorted ascendingly with respect to the absolute value. In case of equality, the elements are sorted ascendingly with respect to the imaginary part. Avoid loops.

Problem 7.
Let $U = (u_{ij}) \in \mathbf{C}^{n\times n}$ be an upper triangular and regular matrix, i.e., $u_{jk} = 0$ for $j > k$, such that $u_{jj} \neq 0$ for all $j = 1, \dots, n$.

(a) Show that for every $b \in \mathbf{C}^n$, there exists a unique solution $x \in \mathbf{C}^n$ of $Ux = b$.

(b) Write a function `solveU`, which, given an upper triangular matrix U as above and a vector $b \in \mathbf{C}^n$, computes the unique solution $x \in \mathbf{C}^n$ of $Ux = b$. Use only loops and arithmetics. You must not use the `scipy.linalg` or `numpy.linalg` module to solve the linear system. However, you can use it to test your implementation.

Problem 8. The integral $\int_a^b f\,dx$ of a continuous function $f : [a, b] \to \mathbf{R}$ 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 $x = (x_1, \dots, x_n) \in [a, b]^n$ with $x_1 < \dots < x_n$ and approximates the function f by some polynomial $p(x) = \sum_{j=1}^{n} a_j x^{j-1}$ of degree $\le n - 1$ with $p(x_j) = f(x_j)$ for all $j = 1, \dots, n$. The weights ω_j are defined as the solution of

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

(a) Show that (1) is equivalent to

$$\int_a^b x^k \, dx = \sum_{j=1}^{n} \omega_j x_j^k \quad \text{for all } k \in \{0, \dots, n-1\}. \tag{2}$$

(b) Write a function `integrate` which takes the vector $x = (x_1, \dots, x_n) \in [a, b]^n$ and the function value vector $(f(x_1), \dots, f(x_n))$, and which returns the approximated value of the integral $\sum_{j=1}^{n} \omega_j f(x_j)$. Avoid loops and use appropriate vector functions and arithmetic instead. Hint: (2) is a linear system in $(\omega_1, \dots, \omega_n) \in \mathbf{R}^n$, which you can solve with scipy.