Homework - Serie 08

Kevin Sturm
Python 3

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

Problem 1. ***2d numpy arrays***

Write a function `frobeniusnorm` which computes and returns the frobeniusnorm of a given matrix $A = (a_{ij}) \in \mathbf{R}^{m\times n}$ defined by

$$\|A\|_F := \Big(\sum_{j=1}^{m}\sum_{k=1}^{n} a_{jk}^2\Big)^{1/2}.$$

Compare your result with `numpy.linalg.norm(A,ord='fro')`. Avoid loops.

Problem 2. ***2d numpy arrays***

Write a function which generates and displays a chessboard-matrix $A \in \mathbf{R}^{n\times n}$ of the form

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & 1 & \cdots \\ 1 & 0 & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Avoid loops!

Problem 3. ***1d complex numpy arrays***

Write a function which calculates and returns for a vector $x \in \mathbf{C}^n$ 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. Test the performance of both methods with `tic` and `toc` from the slides.

Problem 4. ***2d numpy arrays***

Write a script which generates and displays for given dimension n an arrow-matrix $A \in \mathbf{R}^{n\times n}$ of the form

$$A = \begin{pmatrix} 1 & \cdots & 1 & 1 & 1 \\ & & & 1 & 1 \\ & & 1 & & 1 \\ & \iddots & & & \vdots \\ 1 & & & & 1 \end{pmatrix},$$

where all entries which are not shown have to be initialized with 0. Avoid loops!

Problem 5. ***homemade matrix product***

Write two functions `matrix_prod(A,B)` and `matrix_prod2(A,B)`, which compute the matrix product of two matrices $A = (a_{ij}) \in \mathbf{R}^{n,m}$, $B = (b_{ij}) \in \mathbf{R}^{m,l}$, $n, m, l \in \mathbf{N}$ defined by

$$(AB)_{ij} := \sum_{\ell=1}^{m} a_{i\ell} b_{\ell j}, \quad 1 \le i \le n,\ 1 \le j \le l$$

in two ways. The first function uses array slicing and vector operations and the second uses python loops. Test the performance of both methods by measuring the time using `tic` and `toc` from the slides and generate the matrices with the numpy package `random`.

Problem 6. ***tensor product***

Write a class which has the methods `tensor1` and `tensor2`, which compute a product $A \odot B$ of two matrices $A = (a_{ij\ell}) \in \mathbf{R}^{k\times n\times m}$, $B = (b_{ij\ell}) \in \mathbf{R}^{m\times n\times l}$, $n, m, l, k \in \mathbf{N}$. The first method returns $A \odot B \in \mathbf{R}^{k\times l}$

$$(A \odot B)_{ij} := \sum_{r=1}^{n} \sum_{\ell=1}^{m} A_{ir\ell} B_{\ell rj}, \quad 1 \le i \le k,\ 1 \le j \le l,$$

and the second returns $A \odot B \in \mathbf{R}^{k}$ defined by

$$(A \odot B)_{i} := \prod_{j=1}^{l} \sum_{r=1}^{n} \sum_{\ell=1}^{m} A_{ir\ell} B_{\ell rj}, \quad 1 \le i \le k.$$

Try to avoid loops whenever you can!

Problem 7. ***wedge product.***

The wedge product (dt. Dach oder Keilprodukt) of the vectors $w_1, \dots, w_n \in \mathbf{R}^d$ with $n \le d$ is defined by

$$(w_1 \wedge \cdots \wedge w_n)(v_1, \dots, v_n) := \det \begin{pmatrix} \langle w_1, v_1 \rangle & \dots & \langle w_n, v_1 \rangle \\ \vdots & \dots & \vdots \\ \langle w_1, v_n \rangle & \dots & \langle w_n, v_n \rangle \end{pmatrix}, \quad v_1, \dots, v_n \in \mathbf{R}^d,$$

where $\langle \cdot, \cdot \rangle$ denotes the Euclidean scalar product on $\mathbf{R}^d$. Write a function which gets the vectors $w_1 \dots, w_n \in \mathbf{R}^d$ and returns the function $(v_1 \dots, v_n) \mapsto (w_1 \wedge \cdots \wedge w_n)(v_1, \dots, v_n)$. You may assume $d = 3$.

Problem 8. ***A simple polynomial class.***

Write a class `poly` which gets a list `coeff` containing the coefficients of the polynomial $p(x) = a_0 + a_1 x + \cdots + a_n x^n$.

(a) Write a class method `poly_eval` which gets a value $x \in \mathbf{R}$ and return $p(x)$.

(b) Write a second method `poly_der_coef` returns the coefficients of the polynomial $x \mapsto p'(x)$. For example if $p(x) = 1 + x + x^3$, then `poly_der_coef` will return $[1, 0, 3]$. Test your code with the polynomial $p(x) = 1 - 2x + x^4 - 10x^5$.