Dr. Christian Vollmann

Winter Term 2022

[November 4, 2022]

Elements of Mathematics

Sheet 01

Due date: **XXX**

Name:

Matriculation number:

Task:	1	2	3	4	Total	Grade
Points:						
Total:	6	6	12	10	34	–

A matrix as a Linear Function

Let $A \in \mathbb{F}^{m\times n}$ be a matrix. Then consider the mapping $f_A \colon \mathbb{F}^n \to \mathbb{F}^m, x \mapsto Ax$.

1. Show that
$$f_A(\lambda x + y) = \lambda f_A(x) + f_A(y),$$
for all $x, y \in \mathbb{F}^n$ and $\lambda \in \mathbb{F}$.

   Hint: A vector is a matrix with just one column, so you can make use of the computation rules for matrices given in the lecture notes.

   Remark: Functions satisfying this property are called **linear**.

2. Use this fact to show the following equivalence:
$$\ker(A) := \{x \in \mathbb{F}^n \colon Ax = 0\} = \{0\} \quad \Leftrightarrow \quad f_A(x) = f_A(y) \text{ implies } x = y,$$
$$\text{(i.e., } f_A \text{ is an injective mapping).}$$

   Hint: Split up the equality $\Leftrightarrow$ into $\Rightarrow$ and $\Leftarrow$ and prove each of them separately.

The Subspaces Kernel and Image

1. Let $A \in \mathbb{F}^{m \times n}$. Show that $\ker(A)$ and $\mathsf{Im}(A)$ are subspaces of $\mathbb{F}^n$ and $\mathbb{F}^m$, respectively.
2. Construct two example matrices and consider their kernel and image.

Rank/Image and Nullity/Kernel
Consider the matrix

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

the column vector $\mathbf{1} := \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ (i.e., a (3×1) matrix) and the row vector $\tilde{\mathbf{1}} := (1\ 1\ 1)$ (i.e., a (1×3) matrix).

1. Show that $A = \mathbf{1}\tilde{\mathbf{1}}$.
2. Find two distinct nonzero vectors x and y, so that $Ax = 0$ and $Ay = 0$.
3. How does the image Im(A) look like? First draw a picture. Then find a basis of Im(A) to determine the rank of the matrix.
4. How does the kernel ker(A) look like? First draw a picture. Then find a basis of ker(A) to determine the nullity of the matrix.

   Remark: You have to prove that your vectors are a basis.

The Matrix-Vector Product

Implement a function that takes as input a matrix $A \in \mathbb{R}^{m\times n}$ and a vector $x \in \mathbb{R}^n$ and then returns the matrix-vector product Ax.

1. Implement the following four ways:
   a) **Dense:** Input expected as `numpy.ndarray`:
      Assume that the matrix and the vector are delivered to your function as `numpy.ndarray`.
      i. Implement the matrix-vector product "by hand" using for loops, i.e., *without* using numpy functions/methods.
      ii. Implement the matrix-vector product using `A.dot(x)`, `A@x`, `numpy.matmul(A,x)` or `numpy.dot(A,x)`.
   b) **Sparse:** Matrix expected in CSR format:
      Assume that the matrix is delivered to your function as `scipy.sparse.csr_matrix` object. The vector x can either be expected as `numpy.ndarray` or simply as a Python `list`.
      i. Access the three CSR lists via `A.data`, `A.indptr`, `A.indices` and implement the matrix-vector product "by hand" using for loops.
      ii. Implement the matrix-vector product using `A.dot(x)` or `A@x` .
2. **Test** your four different routines from above on the following matrix $A \in \mathbb{R}^{n\times n}$ with constant diagonals given by

$$A = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n\times n}$$

   and the input vector
   $$x = (1, \cdots, 1)^\top \in \mathbb{R}^n \quad \text{(you can use: } \texttt{x = numpy.ones(n)}\text{).}$$
   a) Determine how $b := A \cdot x \in \mathbb{R}^n$ looks like in this example in order to facilitate a test.
   b) Test whether your four routines compute the matrix–vector product correctly by checking $A \cdot x = b$.
   c) Use different values for the dimension n (especially large $n \geq 10^5$ – note that you may exceed your hardware capacities for the dense computations).

   Remark: The matrix has "2" on the main diagonal and "-1" on the first off-diagonals.
3. For all cases:
   a) **Memory:** What is the number of Gbytes needed to store an $m \times n$ array of `floats`? Print the number of Gbytes which are needed to store the matrix in all cases.
      Hint: A number implemented as `float` in Python implements double precision and therefore needs `64` Bits of storage. For a `numpy.ndarray` you can type `A.nbytes` and for the `scipy.sparse.csr_matrix` you can type `A.data.nbytes + A.indptr.nbytes + A.indices.nbytes`.
   b) **Computation times:** Measure the time which is needed in each case to compute the matrix-vector product for a random input vector `x = numpy.random.rand(n)`.
      Hint: In the IPython shell you can simply use the *magic function* `%timeit` to measure the time for a certain operation. For example, you can type `%timeit pythonfunction(x)`. Alternatively you can use the package `timeit`.

Sparse Matrix

10	0	0	0	-2
3	9	0	0	0
0	7	8	7	0
3	0	8	7	5
0	8	0	9	13

data

10	-2	3	9	7	8	7	3	8	7	5	8	9	13

(column) indices

0	4	0	1	1	2	3	0	2	3	4	1	3	4

row pointer (indprt)

0	2	4	7	11	14

Example of a Matrix in CSR Format