Exercise Sheet № 2

2.1 Analytical Problems

Remark on arithmetic operations on sets. Let $A \subseteq \mathbb{R}^p$ and $\boldsymbol{x}_0 \in \mathbb{R}^p$. We say $A \pm \boldsymbol{x}_0 = \{\boldsymbol{a} \pm \boldsymbol{x}_0 | \boldsymbol{a} \in A\}$. Let $A \subseteq \mathbb{R}$ and $x_0 \in \mathbb{R}$, then we additionally define $x_0 A = \{a x_0 | a \in A\}$.

Definition 2.1: Landau-Notation for Functions

Let $I \subseteq \mathbb{R}$ be an open interval, $f, g\colon I \to \mathbb{R}$ and $x_0 \in I$. We say:

$$f(x \to x_0) \in \mathcal{O}(g(x \to x_0)) \Leftrightarrow \exists C, \delta > 0\colon |f|_{\mathcal{B}_\delta(x_0)}(x)| \leq C|g|_{\mathcal{B}_\delta(x_0)}(x)|$$

Furthermore

$$f(x \to x_0) \in o(g(x \to x_0)) \Leftrightarrow \forall \varepsilon > 0\colon \exists \delta > 0\colon f|_{\mathcal{B}_\delta(x_0)}(x) \in \varepsilon g(\mathcal{B}_\delta(x_0))$$

Definition 2.2: More types of Landau-Symbols

Let $(f_n)_{n\in\mathbb{N}}$ and $(g_n)_{n\in\mathbb{N}}$ be sequences, then we define:

$$f \in \mathcal{O}(g) \Leftrightarrow \exists N \in \mathbb{N}, C \in \mathbb{R}\colon n \geq N \Rightarrow |f_n| \leq C|g_n|$$
$$f \in \Omega(g) \Leftrightarrow \exists N \in \mathbb{N}, c \in \mathbb{R}\colon n \geq N \Rightarrow |f_n| \geq c|g_n|$$
$$f \in \Theta(g) \Leftrightarrow (f \in \mathcal{O}(g)) \wedge (f \in \Omega(g)) \Leftrightarrow \exists N \in \mathbb{N}, c, C \in \mathbb{R}\colon c \leq C\colon n \geq N \Rightarrow c|g_n| \leq |f_n| \leq C|g_n|$$

Lemma 2.1: Properties of Landau-Symbols

Let $I \subseteq \mathbb{R}$ be an open interval and $f, f_1, f_2, g, g_1, g_2, h\colon I \to \mathbb{R}$, then the following holds true

- i $f \in \mathcal{O}(f)$
- ii $f \in o(g) \Rightarrow f \in \mathcal{O}(g)$
- iii $f \in \mathcal{O}(g) \Rightarrow \forall K \in \mathbb{R}\colon Kf \in \mathcal{O}(g)$
- iv $f \in \mathcal{O}(g_1) \wedge g_1 \in \mathcal{O}(g_2) \Rightarrow f \in \mathcal{O}(g_2)$
- v $f_1 \in \mathcal{O}(g_1) \wedge f_2 \in \mathcal{O}(g_2) \Rightarrow f_1 f_2 \in \mathcal{O}(g_1 g_2)$
- vi $f \in \mathcal{O}(g + h) \wedge h \in \mathcal{O}(g) \Rightarrow f \in \mathcal{O}(g)$

Task 2.1: Landau-Symbols

- i Use Definition 2.1 to prove Lemma 2.1
- ii
  - a Show that for any polynomial $p \in \mathbb{R}_k[x]$, with $a_k > 0$ and $a_i \geq 0$, it follows $p \in \Theta(n^k)$
  - b Let $n \geq 1$, show or disprove:
    - b.i $2n + 3 \operatorname{ld} n \in \Theta(n)$
    - b.ii $\sum_{i=1}^n i^k \in \Theta(n^{k+1})$
    - b.iii $n! \in \Theta((n+1)!)$
  - c For $a > 1$ and $b > 1$, explain the difference between $\Theta(\log_a n)$ and $\Theta(\log_b n)$ as $n \to \infty$
- iii Show the following
  - a $\frac{\sin(x)}{x} - 1 \in \mathcal{O}(x^2)$ as $x \to 0$
  - b $x^2 + 3x \in \mathcal{O}(x)$ as $x \to 0$
  - c $x^2 - x - 6 \in \mathcal{O}(x - 3)$ as $x \to 3$

Subtask i Let $I \subseteq \mathbb{R}$ be open and $f, f_1, f_2, g, g_1, g_2, h\colon I \to \mathbb{R}$. Property i:

$$C \in \mathbb{R}^+ \Rightarrow \forall x \in \mathbb{R}\colon |f(x)| \leq C|f(x)| \Leftrightarrow f \in \mathcal{O}(f)$$

Property ii

$$\begin{aligned}&f(x_0) \in o(g(x_0)) \Leftrightarrow \forall \varepsilon > 0\colon \exists \delta > 0\colon \forall x \in \mathcal{B}_\delta(x_0)\colon |f(x)| \leq \varepsilon |g(x)| \\ &\Rightarrow \varepsilon = C > 0 \Rightarrow \exists \delta_C > 0\colon \forall x \in \mathcal{B}_{\delta_C}(x_0)\colon |f(x)| \leq C|g(x)| \Leftrightarrow f(x_0) \in \mathcal{O}(g(x_0))\end{aligned}$$

Property iii

$$\begin{aligned}&f(x_0) \in \mathcal{O}(g(x_0)) \Leftrightarrow \exists \delta > 0, C > 0\colon \forall x \in \mathcal{B}_\delta(x_0)\colon |f(x)| \leq C|g(x)| \\ &K \in \mathbb{R} \Rightarrow \forall x \in \mathcal{B}_\delta(x_0)\colon |Kf(x)| = |K| \cdot |f(x)| \leq |K|C|g(x)| = \tilde{C}|g(x)| \\ &\Rightarrow \forall K \in \mathbb{R}\colon \exists \tilde{C} > 0, \delta > 0\colon \forall x \in \mathcal{B}_\delta(x_0)\colon |Kf(x)| \leq \tilde{C}|g(x)| \Leftrightarrow Kf(x_0) \in \mathcal{O}(g(x_0))\end{aligned}$$

Property iv

$$\begin{aligned}&f(x_0) \in \mathcal{O}(g_1(x_0)) \wedge g_1(x_0) \in \mathcal{O}(g_2(x_0)) \\ &\Leftrightarrow \exists \delta_1, \delta_2, C_1, C_2 > 0\colon \forall x \in \mathcal{B}_{\min(\delta_1,\delta_2)}(x_0)\colon |f(x)| \leq C_1|g_1(x)| \leq C_2|g_2(x)| \\ &\Rightarrow \forall x \in \mathcal{B}_{\min(\delta_1,\delta_2)}(x_0)\colon |f(x)| \leq C_2|g_2(x)| \Leftrightarrow f(x_0) \in \mathcal{O}(g_2(x_0))\end{aligned}$$

Property v

$$\begin{aligned}&f_1 \in \mathcal{O}(g_1) \wedge f_2 \in \mathcal{O}(g_2) \\ &\Leftrightarrow \exists \delta_1, \delta_2, C_1, C_2 > 0\colon \forall x \in \mathcal{B}_{\min(\delta_1,\delta_2)}(x_0)\colon |f_1(x)| \leq C_1|g_1(x)| \wedge |f_2(x)| \leq C_2|g_2(x)| \\ &\Rightarrow \forall x \in \mathcal{B}_{\min(\delta_1,\delta_2)}(x_0)\colon |f_1(x)f_2(x)| = |f_1(x)| \cdot |f_2(x)| \leq C_1|g_1(x)| \cdot C_2|g_2(x)| = C_1C_2|g_1(x)g_2(x)| \\ &\Leftrightarrow f_1(x_0)f_2(x_0) \in \mathcal{O}(g_1(x_0)g_2(x_0))\end{aligned}$$

Property vi

$$\begin{aligned}&f(x_0) \in \mathcal{O}(g(x_0) + h(x_0)) \wedge h(x_0) \in \mathcal{O}(g(x_0)) \\ &\Leftrightarrow \exists \delta_1, \delta_2, C_1, C_2 > 0\colon \forall x \in \mathcal{B}_{\min(\delta_1,\delta_2)}(x_0)\colon |f(x)| \leq C_1|g(x) + h(x)| \leq C_1|g(x)| + C_1|h(x)| \leq 2C_1|g(x)| \\ &\Rightarrow f(x_0) \in \mathcal{O}(g(x_0))\end{aligned}$$

Note that similar properties can be shown for sequences and Ω, which we will use in the following sub-task.

Subtask ii: We start with Sub-Subtask a. Using the limit-criterion, we get:

$$\limsup_{n\to\infty} \frac{p(n)}{n^k} = a_k \in (0, \infty) \Rightarrow p(n) \in \Theta(n^k)$$

Sub-Subtask b:

Sub-Sub-Subtask b.i:

We show for $n \in \mathbb{N}\colon \operatorname{ld} n \leq n$ by induction. For $n = 1$ it follows $\operatorname{ld}(n) = 0 \leq 1$. Now, for $n \to n + 1$ we get:

$$\operatorname{ld}(n+1) = \operatorname{ld}\left(n\left(1 + \frac{1}{n}\right)\right) = \operatorname{ld}(n) + \operatorname{ld}\left(1 + \frac{1}{n}\right) \leq \operatorname{ld}(n) \leq n$$

Therefore, by Property vi we get that $2n + 3\operatorname{ld} n \in \Theta(n)$.

Sub-Sub-Subtask b.ii

$$\begin{aligned}&\sum_{i=1}^n i^k \leq \sum_{i=1}^n n^k = nn^k = n^{k+1} \Rightarrow \sum_{i=1}^n i^k \in \mathcal{O}(n^{k+1}) \\ &\sum_{i=1}^n i^k \geq \left\lceil \frac{n+1}{2} \right\rceil + \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} (n-i)^k \geq \sum_{i=1}^{\lceil \frac{n}{2} \rceil} \left\lceil \frac{n+1}{2} \right\rceil^k = \left\lceil \frac{n}{2} \right\rceil \cdot \left\lceil \frac{n+1}{2} \right\rceil^k \geq \frac{n}{2}\left(\frac{n+1}{2}\right)^l \geq \frac{n^{k+1}}{2^{k+1}} = \frac{1}{2^{k+1}} n^{k+1}\end{aligned}$$

Choosing $C = \frac{1}{2^k}$ we see that $\sum_{i=1}^n i^k \geq Cn^{k+1}$, therefore $\sum_{i=1}^n i^k \in \Omega(n^{k+1})$ and thus $\sum_{i=1}^n i^k \in \Theta(n^{k+1})$.

Sub-Sub-Subtask b.iii

$$\limsup_{n\to\infty} \frac{n!}{(n+1)!} = \limsup_{n\to\infty} \frac{1}{n+1} = 0 \Rightarrow n! \in \mathcal{O}((n+1)!)$$

Sub-Subtask c Given that $\forall a > 1$ we know that

$$\log_a(x) = \frac{\ln x}{\ln a}$$

We see that

$$\log_a n = \frac{\ln n}{\ln a} = \frac{\ln b}{\ln a} \log_b n \Rightarrow \Theta(\log_a n) = \Theta(\log_b n)$$

We start with Sub-Sub-Subtask a: Let $1 > \varepsilon > 0$, then it follows:

$$\left|\frac{\sin(x)}{x} - 1\right| = \left|\sum_{k=0}^{\infty}(-1)^k \frac{x^{2k}}{(2k+1)!} - 1\right| = \left|\sum_{k=1}^{\infty}(-1)^k \frac{x^{2k}}{(2k+1)!}\right| \leq \left|-\frac{x^2}{3!}\right| \leq x^2$$

Sub-Sub-Subtask b: Let $\varepsilon \in (0,1)$, then $\forall x \in \mathcal{B}_\varepsilon(0)\colon |x^2| < |x|$, therefore:

$$x \in \mathcal{B}_\varepsilon(0) \Rightarrow |x^2 + 3x| \leq |x^2| + 3|x| \leq 4|x|$$

Thus $x^2 + 3x \in \mathcal{O}(x)$ as $x \to 0$.

Sub-Sub-Subtask c: Let $\varepsilon > 0$ be sufficiently small:

$$\begin{aligned} &\forall x \in \mathcal{B}_\varepsilon(3)\colon |x^2 - x - 6| \leq c|x-3| \\ &\Leftrightarrow |x-3| \cdot |x+2| \leq c|x-3| \Leftrightarrow |x-3| \cdot |x+2| < \varepsilon|x+2| < c\varepsilon \\ &\Leftrightarrow x + 2 < c \Rightarrow c > 5 + \varepsilon \end{aligned}$$

Task 2.2: Complexity Analysis of Algorithms

Determine the time-complexities of the following algorithms

i First Algorithm:

```
for i = 1, ..., 2n do
    for j = 1, ..., n do
        x = x + 1
    end
end
```

ii Second Algorithm:

```
i = 1
while i ≤ 2n do
    x = x + 1
    i = i + 1
end
```

Subtask i: Assuming the increment of x has a time-duration of c, we get the following time-complexity of the algorithm:

$$T(n) = \sum_{i=1}^{2n} \sum_{j=1}^{n} c = cn \sum_{i=1}^{2n} = 2cn^2 \in \Theta(n^2)$$

If $S(n)$ denotes the memory-complexity of the algorithm, we can assume that both indices i and j are auxillary variables, therefore $S(n) = 2 \in \Theta(1)$.

Subtask ii: Assuming the increment of x has a time-duration of c_1, and the increment of i has a time-duration of c_2, we get:

$$T(n) = \sum_{i=1}^{2n} c_1 + c_2 = (c_1 + c_2) \sum_{i=1}^{2n} = 2n(c_1 + c_2) \in \Theta(n)$$

Assuming x is an auxillary variable, we get $S(n) = 2 \in \Theta(1)$.

2.2 Programming Problem

Task 2.3: Matrix Multiplication

Write a python script with a function `matprod(A, B)` that returns the product of the matrices $\mathrm{A}, \mathrm{B} \in \mathbb{C}^{n \times n}$. Implement the following algorithm:

Algorithm 1: Matrix-Product

```
name: matprod
input: n × n matrix A, n × n matrix B
output: n × n matrix C

matprod(A,B):
    C = 0_n
    for i = 1,...,n do
        for j = 1,...,n do
            for k = 1,...,n do
                C_ij = C_ij + A_ik B_kj
            end
        end
    end
    return C
```

Test the script using:

$$\mathrm{A} = \begin{bmatrix} -2 & 5 & 1 \\ 0 & 8 & -7 \\ 9 & -4 & -3 \end{bmatrix} \qquad \mathrm{B} = \begin{bmatrix} 3 & -4 & 6 \\ -5 & 2 & -1 \\ 8 & -9 & 0 \end{bmatrix}$$

Discuss the runtime-complexity of the algorithm. Can it be further improved?
Bonus: Modify the algorithm in such a way, that the product of $\mathrm{A} \in \mathbb{C}^{n \times m}$ and $\mathrm{B} \in \mathbb{C}^n m \times p$ can be computed and $\mathrm{C} = \mathrm{AB} \in \mathbb{C}^{n \times p}$. Provide test-examples.

We begin by analyzing the runtime-complexity of the naive-algorithm for square-matrices A and B. Assuming $\mathrm{C}_{ij} = \mathrm{C}_{ij} + \mathrm{A}_{ik}\mathrm{B}_{kj}$ takes constant time c, we get:

$$T(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} c = n^3 c \in \Theta(n^3)$$

Notice, since $\mathrm{C} \in \mathbb{C}^{n \times n}$, we also get $T(n) = cn^2 \in \Theta(n^2)$, as complex numbers may require more memory than reals. The implementation can be found in the supplied script-file called `square_prod.py`.

Assuming the implementation-architecture allows for vectorized operation, such as independent summation, we can eliminate the last for-loop by computing the dot-product $\langle \boldsymbol{r}_i(\mathrm{A}), \boldsymbol{c}_j(\mathrm{B}) \rangle$, where $\boldsymbol{r}_k$ denotes the k-th row and $\boldsymbol{c}_k$ denotes the k-th column.

Algorithm 2: Matrix-Product with dot-product

```
name: matprod
input: n × n matrix A, n × n matrix B
output: n × n matrix C

matprod(A,B):
    C = 0_n
    for i = 1,...,n do
        for j = 1,...,n do
            C_ij = ⟨r_i(A), c_j(B)⟩
        end
    end
    return C
```

An implementation is also found in `square_prod.py` under the name `matprod_fast()`.

Bonus: We begin by adapting the pseudo-code:

Algorithm 3: Matrix-Product for non-square matrices

```
name: matprod
input: n × m matrix A, m × p matrix B
output: n × p matrix C

```

```
matprod(A,B):
    C = 0_{n×p}
    for i = 1,...n do
        for j = 1,...,p do
            for k = 1,...m do
                C_{ij} = C_{ij} + A_{ik} B_{kj}
            end
        end
    end
    return C
```

Note the runtime complexity is now $T(n) = cnpm \in \Theta(n^3)$, and $S(n) = cnp$. The python-implementation can be found in the supplied script-file called `rect_prod.py`. We can additionally use the same trick like before and implicitly remove one for-loop by setting $\mathrm{C}_{ij} = \langle \boldsymbol{r}_i(\mathrm{A}), \boldsymbol{c}_j(\mathrm{B}) \rangle$. The implementation is found in `rect_prod.py` under the name `matprod_fast()`.