КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
Механіко-математичний факультет
Кафедра геометрії, топології та динамічних систем

Курсова робота
на тему:

Топологічна стійкість усереднень функцій двох змінних

Студентки I курсу магістратури
Напряму підготовки "Математика"
Багрій А. Г.

Науковий керівник:
доктор фізико-математичних наук,
професор Пришляк О. О.

Київ 2022

1 Вступ

Лінійний фільтр – це така динамічна система, яка застосовує певний лінійний оператор до вхідного сигналу для виділення або відкидання певних частот сигналу та інших функцій по обробці вхідного сигналу. Лінійні фільтри широко застосовуються в електроніці, цифровій обробці сигналів і зображень, в оптиці, теорії управління та інших областях.

Нехай вхідний сигнал задано кусково-лінійною фунцією $f : \mathbb{R}^2 \to \mathbb{R}$ із скінченною кількістю локальних екстремумів. Лінійний фільтр визначається імпульсною перехідною функцією $h(x, y)$. Тоді дія лінійного фільтру на вхідний сигнал визначається як згортка $f(x, y) * h(x, y)$. Якщо $\iint_{\mathbb{R}} h(t, s)dtds = 1$, то $h(x, y)$ можна розглядати як щільність певної ймовірнісної міри, а згортку $f(x, y) * h(x, y)$ – як усереднення по цій мірі.

При застосуванні фільтру важливим є питання збереження форми вхідного сигналу. Необхідно визначити, які умови на вхідну функцію f та міру гарантують топологічну еквівалентність між функцією f та її усередненням $f * h$. Зокрема, вимагається, щоб кількість екстремумів усереднення $f * h$ зберігалась.

Нехай μ певна ймовірнісна міра на $X = [0, 1] \times [0, 1]$. Тоді для будь-якої вимірної функції $f : \mathbb{R}^2 \to \mathbb{R}$ та числа α визначається вимірна функція $f_\alpha : \mathbb{R}^2 \to \mathbb{R}$:

$$f_\alpha(x, y) = \int_X f(x - u\alpha, y - v\alpha)d\mu, \tag{1}$$

яка називається *α-усередненням* функції відносно міри μ із заданим параметром α.

Розглянемо випадок, коли μ – дискретна ймовірнісна міра на $X = [0, 1] \times [0, 1]$ зі скінченним носієм. Тобто $\exists t_1, ..., t_k \in X, t_i = (u_i, v_i), i = \overline{1, k}$ – точки з X такі, що для довільної борелівської підмножини $A \in X$:

$$\mu(A) = \sum_{t_i \in A} \mu(t_i).$$

Тоді усереднення функції $f : \mathbb{R}^2 \to \mathbb{R}$ матиме вигляд

$$f_\alpha(x, y) = \sum_{i=1}^{k} f(x - u_i\alpha, y - v_i\alpha)\mu(t_i). \tag{2}$$

Означення 1. *Дві неперервні фунції $f : \mathbb{R}^2 \to \mathbb{R}$ та $g : \mathbb{R}^2 \to \mathbb{R}$ називаються* ***топологічно еквівалентними****, якщо існують гомеоморфізми, що зберігають орієнтацію*

$$\phi : \mathbb{R}^2 \to \mathbb{R}^2, \quad \psi : \mathbb{R} \to \mathbb{R} \tag{3}$$

такі, що $\psi \circ f = g \circ \phi$, тобто наступна діаграма є комутативною

$$\begin{array}{ccc} \mathbb{R}^2 & \xrightarrow{f} & \mathbb{R} \\ {\scriptstyle\phi}\downarrow & & \downarrow{\scriptstyle\psi} \\ \mathbb{R}^2 & \xrightarrow{g} & \mathbb{R} \end{array}$$

Означення 2. *Нехай $f : \mathbb{R}^2 \to \mathbb{R}$ – неперервна функція і μ – ймовірнісна міра на $[-1, 1] \times [-1, 1]$. Функція f називається* ***топологічно стійкою*** *відносно усереднень по мірі μ, якщо $\exists \varepsilon > 0$ таке, що для будь-якого $\alpha \in [0, \varepsilon)$ функції f та f_α є топологічно еквівалентними. Зокрема, якщо $\alpha = 0$, то $f_\alpha = f$.*

2 Деякі результати про топологічну стійкість функцій

Введемо деякі поняття, які необхідні для дослідження усереднень функцій.

Означення 3. *Назвемо* ***опуклим кітковим розбиттям*** *простору* $\mathbb{R}^2$ *сім'ю опуклих многокутників* $C = \{c_i\}_{i \in \mathbb{N}}$*, яке задовольняє наступним умовам:*

1. $\bigcup\limits_i c_i = \mathbb{R}^2$,

2. $int(c_i) \cap int(c_j) = \emptyset, \forall c_i, c_j, i, j \in \mathbb{N}$,

3. $c_i \cap c_j$ *– спільне ребро, спільна вершина або порожня множина.*

Означення 4. ***k-скелет*** *розбиття* C *визначається наступним чином:*

- C^0 *– вершини,*
- C^1 *– внутрішність ребра,*
- C^2 *– внутрішність граней.*

Через $S \wedge Q$ позначимо опукле кліткове розбиття, що складається з усіх непорожніх попарних перетинів клітин, тобто

$$S \wedge Q = \{s_i \cap q_j\}_{i,j \in \mathbb{N}}. \tag{4}$$

Розглянемо розбиття C, що зсунуте на певний вектор $\delta_i = (u_i, v_i), i \leq k, k \in \mathbb{N}$. Позначимо його як $C + \delta_i = \{u_j + \delta_i\}_{i,j \in \mathbb{N}}$. Тоді можна визначити таке опукле кліткове розбиття, що утворене непорожніми перетинами клітин для всіх зсувів δ_i: $C + \delta := \bigwedge_{i,j}(c_j + \delta_i)$.

Означення 5. *Функція* $f : \mathbb{R}^2 \to \mathbb{R}$ *називається* ***кусково-лінійною*** *відносно розбиття* $C(f)$*, якщо на кожній клітині* $c_i \in C(f)$ *задана лінійна функція* f_i*:*

$$f(x,y) = \begin{cases} f_1(x,y), & (x,y) \in c_1, \\ f_2(x,y), & (x,y) \in c_2, \\ \vdots \\ f_n(x,y), & (x,y) \in c_n, \end{cases} \tag{5}$$

$n \in \mathbb{N}, n \geq 3,\; f_i(x,y) = f_j(x,y)\; \forall (x,y) \in c_i \cap c_j$.

Твердження 1. *Нехай* $f : \mathbb{R}^2 \to \mathbb{R}$ *– кусково-лінійна функція,* $C(f)$ *– її опукле кліткове розбиття. Тоді якщо* $(x,y) \in R^2$ *– точка локального екстремуму заданої функції і* $\exists c \in C(f)$*– відкрита клітина:* $x \in c$*, то* $f(x,y) = f(\bar{c})$*, тобто екстремум досягається на замиканні клітини.*

Нехай μ – дискретна ймовірнісна міра на $[-1,1] \times [-1,1]$ зі скінченним носієм $t_1, ..., t_k$, $p_i := \mu(t_i) : \sum_{i=1} p_i = 1$. Усереднення (2) – це лінійна комбінація функцій $f_i(x,y)$, початок координат яких зміщений у точки $\omega_i^\alpha := (u_i\alpha, v_i\alpha)$, із відповідними коефіцієнтами p_i. Позначимо ці функції як $f^i(x,y) := f(x - u_i\alpha, y - v_i\alpha)$. Вони визначені на розбитті $c_i^\alpha = \{c_j + w_i^\alpha\}_{j \in \mathbb{N}}$. Тоді

$$f^i(x,y) = \begin{cases} f_1^i(x,y), & (x,y) \in c_1 + w_i^\alpha, \\ f_2^i(x,y), & (x,y) \in c_2 + w_i^\alpha, \\ \vdots \\ f_n^i(x,y), & (x,y) \in c_n + w_i^\alpha, \end{cases} \tag{6}$$

$i = \overline{1, k}$. Лінійна комбінація кусково-лінійних функцій – це також кусково-лінійна функція. Отже, f_α – кусково-лінійна функція відносно опуклого кліткового розбиття $C(f_\alpha)$.

Функція f_α може набувати свого мінімуму на замиканні обмежених клітин з C^α або на обмежених k-скелетах розбиття.

Означення 6. *Функція f називається **однорідною**, якщо $f(tx, ty) = tf(x, y)$ $\forall t \geq 0$.*

Теорема 1. *Нехай $f_\alpha : \mathbb{R}^2 \to \mathbb{R}$ – кусково-лінійна функція з розбиттям $C(f_\alpha)$, а $f_\beta : \mathbb{R}^2 \to \mathbb{R}$ – кусково-лінійна функція з розбиттям $C(f_\beta)$, $\alpha, \beta > 0$. Тоді $C\big(f_\alpha(x, y)\big) = C\big(\frac{\alpha}{\beta} f_\beta(\frac{\beta x}{\alpha}, \frac{\beta y}{\alpha})\big)$ $\forall \alpha, \beta > 0$.*

Доведення. Введемо функцію $\phi_\alpha : \mathbb{R}^2 \to \mathbb{R}^2 :\ \phi_\alpha(x, y) = (\alpha x, \alpha y)$. Для заданої міри μ та параметру α усереднення має вигляд

$$f_\alpha(x, y) = \sum_{i=1}^{k} f(x - u_i\alpha, y - v_i\alpha)p_i. \tag{7}$$

При $\alpha = 1$ маємо

$$f_1(x, y) = \sum_{i=1}^{k} f(x - u_i, y - v_i)p_i. \tag{8}$$

Для $\forall \alpha > 0$ з однорідності маємо

$$f_1(\alpha x, \alpha y) = \sum_{i=1}^{k} f(\alpha x - u_i, \alpha y - v_i)p_i = \alpha \sum_{i=1}^{k} f(x - \frac{u_i}{\alpha}, y - \frac{v_i}{\alpha})p_i = \alpha f_{\frac{1}{\alpha}}. \tag{9}$$

$$\alpha f_{\frac{1}{\alpha}} = f_1 \circ \phi_\alpha \implies f_\alpha = \alpha f_1 \circ \phi_{\frac{1}{\alpha}}. \tag{10}$$

Тоді $f_1 = \frac{1}{\alpha} f_\alpha \circ \phi_\alpha$. Аналогічно $\forall \beta > 0$ $f_1 = \frac{1}{\beta} f_\beta \circ \phi_\beta$.

$$\frac{1}{\alpha} f_\alpha \circ \phi_\alpha = \frac{1}{\beta} f_\beta \circ \phi_\beta \implies f_\alpha = \frac{\alpha}{\beta} f_\beta \circ \phi_\beta \circ \phi_{\frac{1}{\alpha}} = \frac{\alpha}{\beta} f_\beta(\frac{\beta x}{\alpha}, \frac{\beta y}{\alpha}). \tag{11}$$

Тоді $C\big(f_\alpha(x, y)\big) = C\big(\frac{\alpha}{\beta} f_\beta(\frac{\beta x}{\alpha}, \frac{\beta y}{\alpha})\big)$.

□

Наслідок 1. *Нехай $f : \mathbb{R}^2 \to \mathbb{R}$ – однорідна кусково-лінійна функція на розбитті $C(f)$. Якщо $\exists \alpha_0 > 0$ таке, що f_{α_0} має один екстремум, то $\forall \alpha > 0, \alpha < \alpha_0$ f_α теж має один екстремум.*

Наслідок 2. *Якщо $f : \mathbb{R}^2 \to \mathbb{R}$ – однорідна кусково-лінійна функція і $\exists \alpha_0 > 0$ таке, що $\exists! (x, y) \in \mathbb{R}^2$ – точка локального екстремуму f_α, то f – топологічно стійка відносно усереднення по мірі μ для $\forall \alpha \leq \alpha_0, \alpha > 0$.*

Наслідок 3. *Якщо $f : \mathbb{R}^2 \to \mathbb{R}$ – однорідна кусково-лінійна функція і f не є топологічно стійкою відносно усереднення по мірі μ, то $\forall \alpha > 0$ $\exists k \in \{1, 2\} : \exists c_i \in C^k :$ f_α приймає постійне значення на c_i.*

3 Приклади

Приклад 1

Розглянемо функцію

$$f(x, y) = |x| + |y|, \quad x, y \in \mathbb{R}. \tag{12}$$

Дана функція має глобальний мінімум у точці $(0, 0)$. Зафіксуємо носій ймовірнісної міри у точках $t_1 = (1, 1), t_2 = (1, -1), t_3 = (-1, -1), t_4 = (-1, 1), p_i = 1/4, i = \overline{1, 4}$. Нехай $\alpha = 0.5$. Тоді усереднення заданої функції матиме вигляд

$$f_\alpha(x, y) = \begin{cases} -x - y, \quad x, y \in (-\infty, -\alpha), \\ -x + \alpha, \quad x \in (-\infty, -\alpha), y \in [-\alpha, \alpha], \\ -y + \alpha, \quad x \in [-\alpha, \alpha], y \in (-\infty, -\alpha), \\ 2\alpha, \quad x, y \in [-\alpha, \alpha], \\ -x + y, \quad x \in (-\infty, -\alpha), y \in (\alpha, +\infty), \\ y + \alpha, \quad x \in [-\alpha, \alpha], y \in (\alpha, +\infty), \\ x - y, \quad x \in (\alpha, +\infty), y \in (-\infty, -\alpha), \\ x + \alpha, \quad x \in (\alpha, +\infty), y \in [-\alpha, \alpha], \\ x + y, \quad x, y \in (\alpha, +\infty). \end{cases} \tag{13}$$

Дійсно, усереднення не зберігає мінімум заданої функції з точністю до гомеоморфізму – на прямокутнику $[-\alpha, \alpha] \times [-\alpha, \alpha]$ f_α приймає стале значення 2α.

При візуальному аналізі результатів зручно розглядати лінії рівня функцій та вектори, на яких задаються вони задаються. Тобто множини $L_c(f) = \{(x, y) : f(x, y) = c\}, c \in \mathbb{R}$. Для подальшої демонстрації результатів використовується програма, написана на Python, яка візуалізує описану вище множину C.

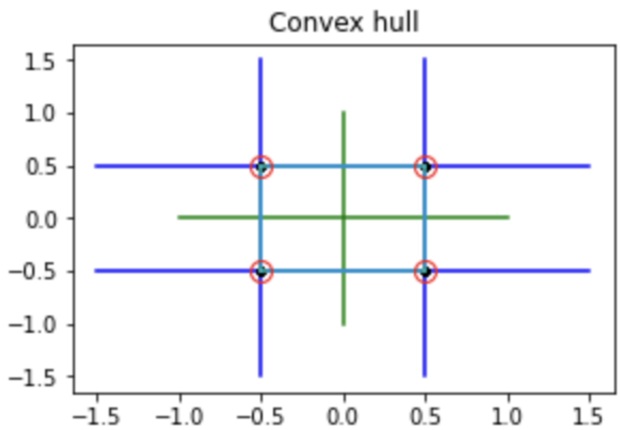

Рис. 1: Зелений графік – вектори початкової функції, сині графіки – вектори функцій f^i, бірюзові лінії формують опуклу обонку

Проаналізувавши отриманий результат та рисунок 1, бачимо, що множина C складається із 4 точок, 4 відрізків та одного прямокутнику. Мінімум f_α досягається на прямокутнику.

Тепер проаналізуємо поведінку усереднення функції неперервно змінюючи p_1, p_2, p_3 при фіксованому $p_4 = 0.25$ врахочуючи, що $p_1 + p_2 + p_3 + p_4 = 1$. Зафіксуємо ітераційний крок $s = 0.005$ і будемо визначати міру як $s \leq p_1 \leq 1 - p_4, s \leq p_2 \leq 1 - p_4 - p_1, p_3 = 1 - p_4 - p_1 - p_2$. Маємо наступний ймовірнісний симплекс:

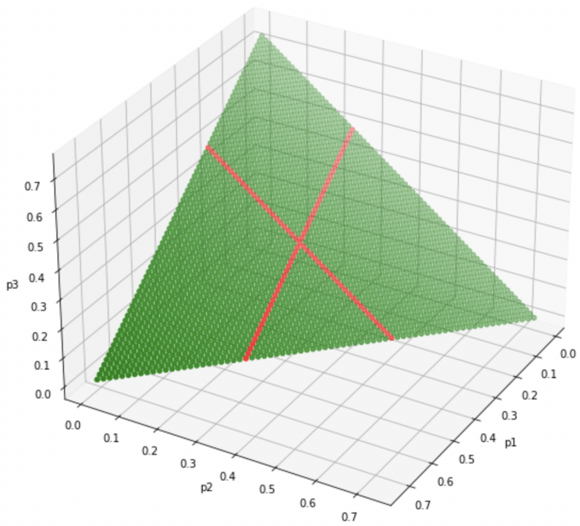

Рис. 2: Зелені точки на симплексі відповідають такій мірі, що усереднення зберігає один мінімум. Червоні точки навпаки – такій, де усереднення має не один мінімум.

Міра (p_1, p_2, p_3) тут визначає дві прямі, що перетинаються. При чому, точка перетину відповідає мірі $p_i = 1/4, i = \overline{1,4}$, де мінімум усереднення – це прямокутник. Відповідно, для інших точок цих прямих мінімум усереднення – це якийсь відрізкок з множини С.

Приклад 2

Розглянемо іншу конфігурацію. Нехай

$$f(x, y) = \begin{cases} 2x + y, & x \geq 0, x \geq -y, \\ -2x + y, & x \leq 0, x \leq y, \\ -y, & x \geq y, x \leq -y. \end{cases} \tag{14}$$

Лінії рівня цієї функції мають наступний вигляд:

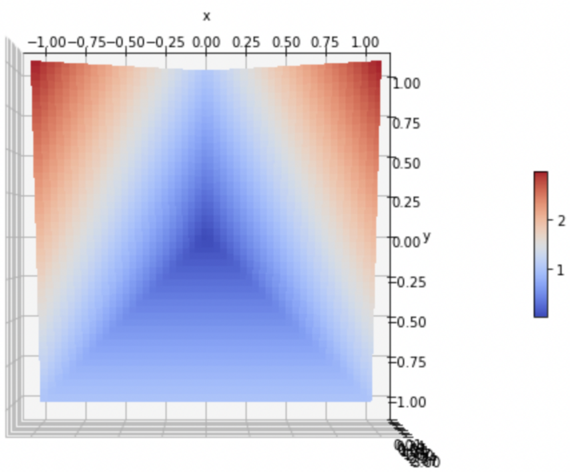

Рис. 3: f(x,y)

Очевидно, що функція має глобальний мінімум у точці $(0,0)$. Зафіксуємо носій ймовірнісної міри у точках $t_1 = (1,1), t_2 = (1,-1), t_3 = (-1,-1), p_1 = 0.3, p_2 = 0.5, p_3 = 0.2$. Нехай $\alpha = 0.5$.

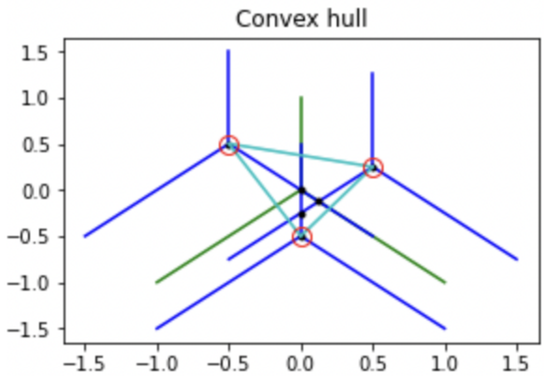

Рис. 4: Зелений графік – вектори початкової функції, сині графіки – вектори функцій f^i, бірюзові лінії формують опуклу обонку

З рисунку 4 видно, що множина C складається з 5 точок, 6 відрізків і одного трикутника. Подальший аналіз проведемо за допомогою іншої програми, що написана на Python.

Розглянемо графік усереднення f_α в $\mathbb{R}^3$. Як бачимо, усереднення не зберігає мінімум початкової функції, натомість, усереднення набуває мінімум на одному із відрізків з C.

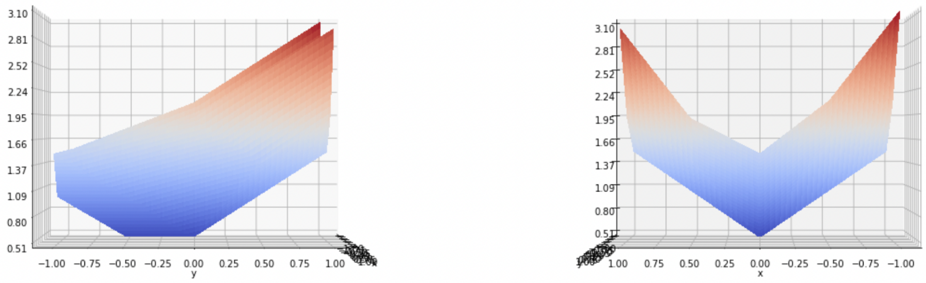

Рис. 5: 1-ий графік – це "погляд"на функцію з площини Oyz, 1-ий графік – це "погляд"на функцію з площини Oyx

Приклад 3

Розглянемо таку функцію

$$f(x,y) = \begin{cases} \frac{3x}{2} + y, & x \geq 0, x \geq -2y, \\ \frac{x}{3} - \frac{4y}{3}, & x \leq 0, x \leq y, \\ -2x + y, & x \geq y, x \leq -2y. \end{cases} \tag{15}$$

Лінії рівня цієї функції мають наступний вигляд:

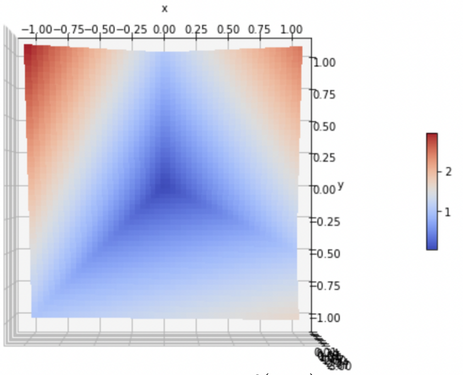

Рис. 6: f(x,y)

Зафіксуємо носій міри аналогічно до попереднього прикладу у точках $t_1 = (1,1), t_2 = (1,-1), t_3 = (-1,-1)$. Проте тепер проаналізуємо поведінку усереднення неперервно змінюючи p_1, p_2, p_3 врахочуючи, що $p_1 + p_2 + p_3 = 1$. Зафіксуємо ітераційний крок $s = 0.005$ і будемо визначати міру як $s \leq p_1 \leq 1, s \leq p_2 \leq 1 - p_1, p_3 = 1 - p_1 - p_2$. Тоді цікавим є наступний результат. Для даної функції в заданих точках міри усереднення буде завжди зберігати мінімум. Це можна побачити подивившись на наступний ймовірнісний симплекс:

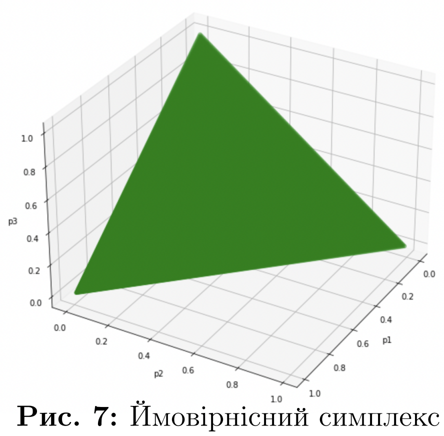

Рис. 7: Ймовірнісний симплекс

Такі результати показують, що існують певні достатні умови для функції та заданої міри, які гарантують топологічну стійкість.

4 Висновки

В даній роботі була сформульована задача усереднення функцій двох змінних відносно дискретної ймовірносної міри із заданим параметром. Була проаналізована поведінка усереднень для різних функцій за допомогою теоретичного підходу, а також із допомогою написаних комп'ютерних програм на мові Python. Отримані загальні результати поведінки усереднень, які надалі допоможуть вивести широки достатні умови для топологічної стійкості функцій.

Хочу висловити подяку члену-кореспонденту НАН України, доктору фізико-математичних наук, професору Максименко С. І. за запропоновану задачу, а також за постійне наставництво та допомогу при роботі з даною темою.

Література

[1] Maksymenko, S. I.; Marunkevich, O. V. Topological stability of averagings of functions. Mat. Zh. 68 (2016), no. 5, 625-633 Ukrainian Math. J. 68 (2016), no. 5, 707-717 pp.

[2] Kolmogorov, A. N.; Fomin, S. V. Элементы теории функций и функционального анализа. Fourth edition, revised. Izdat. "Nauka", Moscow, 1976. 543 pp.

Додаток

Програма 1

```
import numpy as np
import matplotlib.pyplot as plt
import sys
import import_ipynb

from matplotlib import cm
from matplotlib.ticker import LinearLocator

np.set_printoptions(threshold=sys.maxsize)

figure_size = 10
plt.rcParams["figure.figsize"] = (figure_size, figure_size)

x_interval = [-1, 1]
y_interval = [-1, 1]

step_interval = 100

alpha = 0.5
epsilon = 10e-10

# Functions

'''
t - vector of points
t_k = (u_k, v_k)

p - vector of coefficients
sum(p1,...,pn) = 1

alpha > 0
'''
def f_a(x, y, t, p, alpha):
        f_a = np.zeros_like(x)

        for i in range(0, len(t)):
                f_a += f(x[:] - t[i, 0] * alpha,
                        y[:] - t[i, 1] * alpha) * p[i]

        return f_a

'''
Calculates the value of the function
and its averaging

Returns two vectors
'''
def calc_f(t, p, alpha):
        x = np.linspace(x_interval[0], x_interval[1], step_interval)
        y = np.linspace(y_interval[0], y_interval[1], step_interval)
        z = []

        X, Y = np.meshgrid(x, y)

        zs = np.array(f(np.ravel(X), np.ravel(Y)))
```

```
        Z = zs.reshape(X.shape)

        zs_a = np.array(f_a(np.ravel(X), np.ravel(Y), t, p, alpha))
        Z_a = zs_a.reshape(X.shape)

        return Z, Z_a

'''
Function for finding and comparing
minimum value of the input function
and its averaging

Returns bool value whish indicates
whether the averaging saves the extrema
'''
def find_min(Z, Z_alpha, print_result = False):
        elements, count = np.unique(np.floor(Z_alpha/epsilon)
        .astype(int), return_counts = True)
        duplicates = elements[count > 1]

        Z_min = Z.min()
        Z_a_min = Z_alpha.min()

        duplicate_min = np.any(duplicates == np.floor(
        Z_a_min/epsilon).astype(int))

        if print_result:
        print('-------')

        print('Z_min =', Z_min)
        print('Z_a_min =', Z_a_min)

        print('\nPreserves min =', not duplicate_min)

        print()

        return not duplicate_min

'''
Function for plotting the surface of the function

t - vector of points
t_k = (u_k, v_k)

p - vector of coefficients
sum(p1,...,pn) = 1

alpha > 0

is_averaging - indicator if we should plot the input function of its averaging

UTILITIES:
show_from_two_angles - bool variable that indicates if to show plot
from the Ox & Oy
show_colorbar - bool variable that indicates if to show colorbar
xy_angle - observation angle of XY plane
z_angle - observation angle of Z ax
title - title of the plot
'''
def plot_surface(t, p, alpha, is_averaging = False,
```

```
show_from_two_angles = False,
xy_angle = 0, z_angle = 0, title = ''):

show_colorbar = True

if show_from_two_angles:
        fig, ax = plt.subplots(nrows=1, ncols=2,
        figsize=(20,20), subplot_kw={"projection": "3d"})
else:
        fig, ax = plt.subplots(
        subplot_kw={"projection": "3d"})

x = np.linspace(x_interval[0], x_interval[1], step_interval)
y = np.linspace(y_interval[0], y_interval[1], step_interval)
z = []

X, Y = np.meshgrid(x, y)

if is_averaging:
        zs = np.array(f_a(np.ravel(X), np.ravel(Y),
                t, p, alpha))
else:
        zs = np.array(f(np.ravel(X), np.ravel(Y)))

Z = zs.reshape(X.shape)

if show_from_two_angles:
        for i in range(0, 2):
                angle_to_rotate = 90

if i == 1:
        z_angle += angle_to_rotate

        ax[i].view_init(xy_angle, z_angle)

        surf = ax[i].plot_surface(X, Y, Z, cmap=cm.coolwarm,
        linewidth=0, antialiased=False)

        ax[i].zaxis.set_major_locator(LinearLocator(10))
        ax[i].zaxis.set_major_formatter('{x:.02f}')

        ax[i].set_xlabel('x')
        ax[i].set_ylabel('y')
else:
        ax.view_init(xy_angle, z_angle)

        surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
        linewidth=0, antialiased=False)

        ax.zaxis.set_major_locator(LinearLocator(10))
        ax.zaxis.set_major_formatter('{x:.02f}')

        ax.set_xlabel('x')
        ax.set_ylabel('y')

if show_colorbar:
        fig.colorbar(surf, shrink=0.2, aspect=10)

plt.title(title)
plt.show()
```

```
        return Z

'''
Function for finding and plotting the measure
of 4 points (with 1 fixed and 3 continuously changing)
where averaging of the function preserves extrema and where not

t - vector of points
t_k = (u_k, v_k)

alpha > 0
step > 0

UTILITIES:
xy_angle - observation angle of XY plane
z_angle - observation angle of Z ax


Returns two distinct vectors:
with measure that preserves extrema and where not
'''
def test_4_measures(t, alpha, step = 0.05,
        xy_angle = 0, z_angle = 0):
        p1 = p2 = p3 = 0.0
        p4 = 0.25

        p_dif = 1 - p4

        number_of_dec = str(step)[::-1].find('.')

        P = np.empty(shape=(0,4), dtype=float)

        P_good = np.empty(shape=(0,4), dtype=float)
        P_bad = np.empty(shape=(0,4), dtype=float)

        for p1 in np.arange(step, p_dif, step):
                for p2 in np.arange(step, p_dif - p1, step):

                        p3 = p_dif - p2 - p1

                        p1 = float(f'{p1:.{number_of_dec}f}')
                        p2 = float(f'{p2:.{number_of_dec}f}')
                        p3 = float(f'{p3:.{number_of_dec}f}')

                        if p3 < step or p2 < step:
                                continue

                        p = [p1, p2, p3, p4]

                        if any((P[:] == p).all(1)):
                                continue

                        P = np.append(P, [p], axis=0)

                        Z_f, Z_a = calc_f(t, p, alpha)
                        preserves = find_min(Z_f, Z_a)

                        if preserves:
                                P_good = np.append(P_good, [p],
```

```
                                axis=0)
                        else:
                                P_bad = np.append(P_bad, [p],
                                axis=0)

                        if preserves:
                                P_good = np.append(P_good, [p],
                                axis=0)
                        else:
                                P_bad = np.append(P_bad, [p],
                                axis=0)

        col = ['green' if any((P_good[:] == P[i,:]).all(1))
                else 'red' for i in range(0, len(P))]

        fig = plt.figure()

        ax = fig.add_subplot(111, projection='3d')
        ax.view_init(xy_angle, z_angle)

        ax.scatter(P[:,0], P[:,1], P[:,2], c=col, label='sd')

        ax.set_xlabel('p1')
        ax.set_ylabel('p2')
        ax.set_zlabel('p3')

        #print(P)
        return P_good, P_bad

'''
Function for finding and plotting the measure
of 3 points that are continuously changing
where averaging of the function preserves extrema and where not

t - vector of points
t_k = (u_k, v_k)

alpha > 0
step > 0

UTILITIES:
xy_angle - observation angle of XY plane
z_angle - observation angle of Z ax


Returns two distinct vectors:
with measure that preserves extrema and where not
'''
def test_3_measures(t, alpha, step = 0.05,
        xy_angle = 0, z_angle = 0):
        p1 = p2 = p3 = step

        p_dif = 1

        number_of_dec = str(step)[::-1].find('.')

        P = np.empty(shape=(0,3), dtype=float)

        P_good = np.empty(shape=(0,3), dtype=float)
        P_bad = np.empty(shape=(0,3), dtype=float)
```

```
        for p1 in np.arange(step, p_dif, step):
                for p2 in np.arange(step, p_dif - p1, step):

                        p3 = p_dif - p2 - p1

                        p1 = float(f'{p1:.{number_of_dec}f}')
                        p2 = float(f'{p2:.{number_of_dec}f}')
                        p3 = float(f'{p3:.{number_of_dec}f}')

                        if p3 < step or p2 < step:
                                continue

                        p = [p1, p2, p3]

                        if any((P[:] == p).all(1)):
                                continue

                        P = np.append(P, [p], axis=0)

                        Z_f, Z_a = calc_f(t, p, alpha)
                        preserves = find_min(Z_f, Z_a)

                        if preserves:
                                P_good = np.append(P_good, [p],
                                        axis=0)
                        else:
                                P_bad = np.append(P_bad, [p],
                                        axis=0)

        col = ['green' if any((P_good[:] == P[i,:]).all(1))
                 else 'red' for i in range(0, len(P))]

        fig = plt.figure()

        ax = fig.add_subplot(111, projection='3d')
        ax.view_init(xy_angle, z_angle)

        ax.scatter(P[:,0], P[:,1], P[:,2], c=col)

        ax.set_xlabel('p1')
        ax.set_ylabel('p2')
        ax.set_zlabel('p3')

        return P_good, P_bad

'''
Function for demonstrating the result of averaging
analytically and visually
'''
def find_and_plot_averaging(t, p, alpha, show_from_two_angles = True,
        xy_angle = 0, z_angle = 0):
        for i in range(0, len(p)):
                p_set = p[i]

                Z, Z_a = calc_f(t, p_set, alpha)
                preserves = find_min(Z, Z_a, True)

                print('p =', p_set)
```

```
                _ = plot_surface(t, p_set, alpha, True,
                        show_from_two_angles, xy_angle, z_angle)
```

Програма 2

```
import numpy as np
import matplotlib.pyplot as plt
import sys
import math

from scipy.spatial import ConvexHull

np.set_printoptions(threshold=sys.maxsize)

figure_size = 5
plt.rcParams["figure.figsize"] = (figure_size, figure_size)

x_interval = [-1, 1]
y_interval = [-1, 1]

step_interval = 100

alpha = 0.5
epsilon = 10e-5

'''
t - array of measure points
alpha - given constant

Returns an array of averaged zeroes of the function
'''
def average_zero(t, alpha):
        result = np.empty(shape=(0,2), dtype=float)

        for i in range(0, len(t)):

                zero = np.empty(shape=(2), dtype=float)

                zero[0] = t[i, 0] * alpha
                zero[1] = t[i, 1] * alpha

                result = np.append(result, [zero], axis=0)

        return result

'''
vectors - an array of vectors
t - array of measure points
alpha - given constant

Returns an array of averaged vectors on which
the function if given
'''
def average_vectors(vectors, t, alpha):
        vectors_count = len(vectors)

        result = np.empty(shape=(0,vectors_count,2), dtype=float)
```

```
        for i in range(0, len(t)):

                averaged_vectors = np.empty(shape=(0,2),
                        dtype=float)

                for j in range(0, vectors_count):
                        averaged_vector = np.empty(shape=(2),
                                dtype=float)

                        averaged_vector[0] = vectors[j, 0]
                                + t[i, 0] * alpha
                        averaged_vector[1] = vectors[j, 1]
                                + t[i, 1] * alpha

                        averaged_vectors = np.append(
                                averaged_vectors,
                                [averaged_vector], axis=0)

                result = np.append(result, [averaged_vectors],
                        axis=0)

        return result

'''
Returns a list of rays intersections if any

line1 - array with a start and an end of the line
that respresents the first ray
line2 - array with a start and an end of the line
that respresents the second ray
'''
def find_intersection_of_rays(line1, line2):

        if np.array_equal(line1[0], line2[1]):
                return line1[0]

        elif np.array_equal(line2[0], line1[1]):
                return line2[0]

        result = np.empty(shape=(2), dtype=float)

        v1 = [line1[1][0] - line1[0][0], line1[1][1] - line1[0][1]]
        v2 = [line2[1][0] - line2[0][0], line2[1][1] - line2[0][1]]

        l1 = np.sqrt(v1[1]**2 + v1[0]**2)
        l2 = np.sqrt(v2[1]**2 + v2[0]**2)

        n1 = [v1[0] / l1, v1[1] / l1]
        n2 = [v2[0] / l2, v2[1] / l2]

        dx = line2[0][0] - line1[0][0]
        dy = line2[0][1] - line1[0][1]

        det = n2[0] * n1[1] - n1[0] * n2[1]

        if det != 0:
                u = (dy * n2[0] - dx * n2[1]) / det
                v = (dy * n1[0] - dx * n1[1]) / det

                if u >= 0 and v >= 0:
```

```
                if n1[0] == 0 or n2[0] == 0:
                        m0 = n1[1] / (n1[0] + epsilon)
                        m1 = n2[1] / (n2[0] + epsilon)
                else:
                        m0 = n1[1] / n1[0]
                        m1 = n2[1] / n2[0]

        b0 = line1[0][1] - m0 * line1[0][0]
        b1 = line2[0][1] - m1 * line2[0][0]

        x = (b1 - b0) / (m0 - m1)
        y = m0 * x + b0

        result = [x, y]

        return result
        print(result)

# Main

initial_vectors = np.array([[0, 1], [1, -1], [-1, -1]])
t = np.array([[1, 0.5], [-1, 1], [0, -1]])

averaged_vectors = average_vectors(initial_vectors, t, alpha)
averaged_zero = average_zero(t, alpha)

points = np.empty(shape=(0,2), dtype=float)

for l in range(0, len(averaged_zero)):
        start1 = averaged_zero[l]
        end1_list = averaged_vectors[l]

        for i in range(0, len(initial_vectors)):
                end1 = end1_list[i]

                for j in range(l + 1, len(t)):
                        start2 = averaged_zero[j]
                        end2_list = averaged_vectors[j]

                        for k in range(0, len(initial_vectors)):
                                end2 = end2_list[k]

                                point =
                                find_intersection_of_rays(
                                [start1, end1],
                                [start2, end2])

                                if point is not None and not
                                np.isnan(point).any():
                                        points = np.append(
                                        points,
                                        [point],
                                        axis=0)

convexFigure = np.concatenate((points, averaged_zero))

hull = ConvexHull(convexFigure)

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(10, 3))
```

```python
for i in range(0, len(initial_vectors)):
        x_values = [0, initial_vectors[i, 0]]
        y_values = [0, initial_vectors[i, 1]]

plt.plot(x_values, y_values, color='green')

x_values = [averaged_zero[0, 0], averaged_vectors[0, 2, 0]]
y_values = [averaged_zero[0, 1], averaged_vectors[0, 2, 1]]

for i in range(0, len(averaged_vectors)):
        for j in range(0, len(averaged_vectors[i])):
                x_values = [averaged_zero[i, 0],
                        averaged_vectors[i, j, 0]]
                y_values = [averaged_zero[i, 1],
                        averaged_vectors[i, j, 1]]

                plt.plot(x_values, y_values, color='blue')

for ax in (ax1, ax2):
        ax.plot(convexFigure[:, 0], convexFigure[:, 1],
                 '.', color='k')

        if ax == ax1:
                ax.set_title('Given points')
        else:
                ax.set_title('Convex hull')
                for simplex in hull.simplices:
                        ax.plot(convexFigure[simplex, 0],
                        convexFigure[simplex, 1], 'c')

                ax.plot(convexFigure[hull.vertices, 0],
                convexFigure[hull.vertices, 1], 'o', mec='r',
                color='none', lw=1, markersize=10)

plt.show()
```