МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”
ІМ. І. СІКОРСЬКОГО

КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ

КУРСОВА РОБОТА

з дисципліни “Методи оптимізації”

на тему: ПОШУК ПО ДЕФОРМОВАНОМУ МНОГОГРАННИКУ.

МЕТОД НЕЛДЕРА-МІДА

Студента 3-го курсу, групи КМ-42
напряму підготовки 6.040301 –
прикладна математика
Польщака О.С.

Керівник
Ладогубець Т.С.

Національна оцінка: _____

Кількість балів: _____

ECTS ____

КИЇВ-2017

ЗМІСТ

ПОСТАНОВКА ЗАДАЧІ

Дослідити метод Нелдера-Міда для пошуку локального мінімуму корінної функції $f_3(x) = (10(x_1 - x_2)^2 + (x_1 - 1)^2)^{\frac{1}{4}}$, $x^{(0)} = (-1{,}2; 0{,}0)$;

1. Задати розмір початкового симплексу.
2. Задати коефіцієнти деформування многогранника.
3. Використати модифікації методу.
4. Знайти мінімум за допомогою модифікованого симплекс-метода – методом Нелдера-Міда.
    а) Перевірити роботу методу в опуклій області.
    б) Перевірити роботу методу в випуклій області.
5. Застосувати методи одномірного пошуку на межі області.
    а) Визначити межі алгоритмом Свена.
    б) Метод золотого зрізу.
    в) Метод ДСК-Пауела.
    г) Визначити точність МОП.
6. Проаналізувати результати.

Для даного методу нульового порядку підібрати оптимальні вихідні параметри.

Порівняти ефективність методів для розв’язку поставленої задачі.

ТЕОРИТИЧНІ ВІДОМОСТІ

Метод Нелдера-Міда як представник сімейства симплекс-методів корисний в хімії, фізиці, економіці, виробництві та в супротиві матеріалів.

Спосіб описаний для мінімізації функції від n змінних, яка залежить від порівняння значень функції в $(n+1)$ вершинах багатокутника (в загальному випадку не регулярний симплекс), з наступною зміною вершини з найбільшим значенням функції іншою точкою, яка по значенню функції, менша ніж попередня. Багатокутник адаптує свою форму для місцевого ландшафту, а також в області біля локального мінімуму. Метод показав свою ефективність і компактний для реалізації на комп'ютерній програмі.[2;1]

Спочатку, ми повинні розглянути принцип симплекс-методу, запропонованого Спендлі, Хекстом і Хімсвортом. В Еклідовому посторі E^n вершини регулярного симплекса можна представити у вигляді матриці D, стовпці якої є поданням координат вершин симплекса. [3; 164]

У випадку, коли початковий симплекс містить точку на початку координатної системи, $x = (0; 0; \dots 0)$, матриця D буде мати наступну форму:

$$D = \begin{bmatrix} 0 & d_1 & d_2 & \cdots & d_2 \\ 0 & d_2 & d_1 & \cdots & d_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & d_2 & d_2 & \cdots & d_1 \end{bmatrix} \text{ – матриця } n \times (n+1),$$

Якщо точка у початку координат не є обов’язковою для початкового симплекса тоді матриця набуває наступного вигляду:

$$D = \begin{bmatrix} x_0^0 & d_1' & d_2' & \cdots & d_2' \\ x_1^0 & d_2' & d_1' & \cdots & d_2' \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_n^0 & d_2' & d_2' & \cdots & d_1' \end{bmatrix} \text{ – матриця } n \times (n+1), \text{ (*)}$$

де

$$d_1 = \frac{t}{n\sqrt{2}}\left(\sqrt{n+1} + n - 1\right), d_1' = x_i^0 + d_1$$

$$d_1 = \frac{t}{n\sqrt{2}}\left(\sqrt{n+1} - 1\right), d_1' = x_i^0 + d_2$$

$$i = \overrightarrow{1, \dots, n}$$

t – довжина сторони симплекса, n – розмірність функції x_i^0 – координата заданої початкової точки.[3; 164]

Далі наведена ілюстрація кроку симплекс методу:

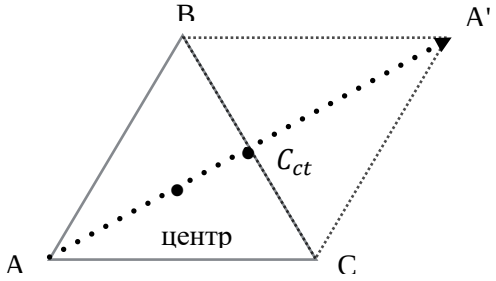


Рис. 1.1. Регулярний сиплекс, у випадку наявності трьох вершин.

На Рис 1.1 регулярний симплекс має три вершини, функція приймає найбільше значення у точці A серед множини вершин симплекса, тому точка A, відображається у точку A', через центроїд – точку яка береться за формулою:

$$C_{ct} = \frac{1}{n}\left[\left(\sum_{i=1}^{n+1} x_i^k\right) - x_h^k\right],$$

де x_h^k – точка в якій функція приймає найбільше значення в множині вершин симплекса. В певному сенсі, центроїд це зміщений центр симплексу, оскільки в розрахунку не задіяна точка що відображається.

Операція за якою відбувається рух симплекса називається відображенням.

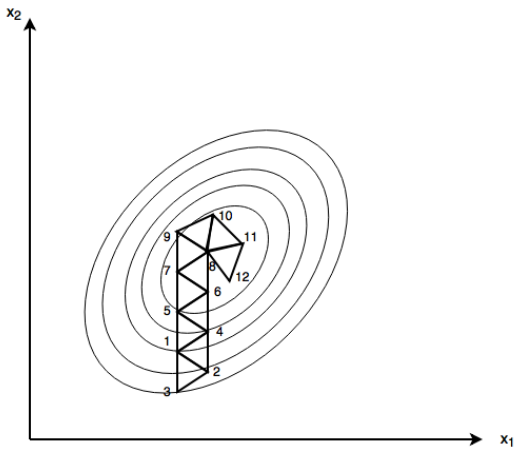


Рис. 1.2. Історія руху регулярного симплекса.

Ітеративний процес який здійснюється симплекс-методом (Рис. 1.2.), підчас якого точка з максимальним значенням функції на множині точок багатокутника відображається на кожному кроці, не потребує взяття похідних и просто реалізується на ЕОМ. Але розмір ребра багатокутника статичний, що може призвести до певних проблем підчас проходження «рівнин» та «гірських хребтів». Тому метод Нелдера-Міда, модифікація симплекс методу, буде корисним.

Метод Нелдера-Міда

Як вже обговорювалося раніше, метод Нелдера-Міда використовує багатокутник (далі полігон), щоб знайти мінімум, але на відміну від симплекс-методу, він деформує полігон. Кожна вершина багатокутника представлена у вигляді вектора x_i^k. Вершина, в якій значення функції максимальне (на множині точок полігону) проектується через центроїд (Рис 1.1.), інші вершини залишаються незмінними (за виключенням гомотетії, або редукції). Нові точки, значення яких менше, виходять в результаті заміни точок з максимальним значенням функції. Цей процес буде продовжуватися до тих пір, поки мінімум функції не буде досягнутий (поки не буде виконуватись критерій зупинки).

Алгоритм[3; 166]:

$$k = 0,$$

k – крок або номер ітерації зміни полігону, збільшується після кожної ітерації.

$$x_i^k = \left[x_{i1}^k, \dots, x_{ij}^k, \dots, x_{in}^k\right]_i^T, i = \overrightarrow{1, \dots, n}, \forall x^k \in E^n,$$

де x_i^k – вершина полігону на k-й ітерації яка задана на Евклідовому просторі E^n, n – розмірність простору.

Нехай x_h^k – точка в якій функція приймає найбільше значення в множині вершин симплекса, x_g^k – точка в якій функція приймає найбільше після $f\left(x_h^k\right)$ значення в множині вершин симплекса, x_l^k – точка в якій функція приймає найменше значення в множині вершин симплекса.

$$x_h^k \in \left\{x_i^k\right\}, f\left(x_h^k\right) = \max\left(\left\{f\left(x_1^k\right), \dots, f\left(x_{n+1}^k\right)\right\}\right),$$

$$x_g^k \in \left\{x_i^k\right\}, f\left(x_g^k\right) = \max\left(\left\{f\left(x_1^k\right), \dots, f\left(x_{n+1}^k\right)\right\}/\left\{x_h^k\right\}\right),$$

$$x_l^k \in \left\{x_i^k\right\}, f\left(x_l^k\right) = \min\left(\left\{f\left(x_1^k\right), \dots, f\left(x_{n+1}^k\right)\right\}\right),$$

Нехай матриця $D_{n(n+1)}$ представляє собою множину координат вершин полігона. Тоді позначимо центроїд як $(n+2)$, приєднаний стовпець матриці. Маємо формулу для знаходження центроїда:

$$x_{n+2}^k = \frac{1}{n}\left[\left(\sum_{i=1}^{n+1} x_i^k\right) - x_h^k\right], (0)$$

Найчастіше початковий полігон будують як симплекс (рівносторонній), але його можна задати довільно, це залежить від певних умов, наприклад від типу функції та розташування початкового полігона.

Розглянемо наступні операції зміни полігона:

1) Віддзеркалення – відображення x_h^k через центроїд x_{n+2}^k, відповідно до формули:

$$x_{n+3}^k = x_{n+2}^k + \alpha\left(x_{n+2}^k - x_h^k\right), \alpha > 0, (1)$$

де α – коефіцієнт віддзеркалення, x_{n+3}^k – точка віддзеркалення, отримана в результаті відповідної операції.

2) Розтягнення – операція яка використовує інформацію про точку x_{n+3}^k з операції (1). Якщо має місце нерівність $f(x_{n+3}^k) \le f(x_l^k)$, тоді вектор $(x_{n+3}^k - x_{n+2}^k)$ проектується через центроїд відповідно до формули:

$$x_{n+4}^k = x_{n+2}^k + \gamma\left(x_{n+3}^k - x_{n+2}^k\right), \quad \gamma > 1, (2)$$

де γ – коефіцієнт розтягнення, x_{n+4}^k – точка розтягнення, отримана в результаті відповідної операції, x_{n+3}^k – точка віддзеркалення (1), x_{n+2}^k – центроїд (0). Якщо має місце нерівність $f(x_{n+4}^k) \le f(x_l^k)$, тоді x_h^k буде змінена на точку x_{n+4}^k і алгоритм повертається до кроку (1), в іншому випадку x_h^k змінюється на x_{n+3}^k– точку віддзеркалення, і алгоритм повертається до кроку (1) теж. В обох випадках, k збільшується на 1: $k = k + 1$.

3) Стиснення – процедура зменшення габаритів полігона, має місце якщо виконується нерівність $\forall i \ne h, f(x_{n+3}^{(k)}) \ge f(x_i^k)$, буває двох типів: перший, внутрішнє стиснення – x_h^k відображається всередині полігона з коефіцієнтом стиснення β; другий, зовнішнє стиснення – x_h^k проектується зовні полігону з коефіцієнтом β. В загальному випадку $\beta \in (0; 1)$. Варто зазначити: у випадку, коли другий тип має дати кращій результат ніж перший, але виконаний саме перший – алгоритм перейде до кроку (1) і виконавши його отримає результат, аналогічний результату зовнішнього стиснення; симетрично виконується для внутрішнього стиснення, таким чином виконується перехід вид одного типу до іншого. У даній роботі використовується внутрішнє стиснення. Тому "стиснення" $(x_h^k - x_{n+2}^k)$ виконуватиметься за формулою:

$$x_{n+5}^k = x_{n+2}^k + \beta(x_h^k - x_{n+2}^k), \quad \beta \in (0; 1)$$

де β – коефіцієнт стиснення, $x_{n+5}^{(k)}$ – точка стиснення, отримана в результаті відповідної операції, $x_{n+2}^{(k)}$ – центроїд. Після завершення операції алгоритм повертається до кроку (1), k збільшується на 1: $k = k + 1$.

4) Редукція – операція гомотетії. Якщо має місце нерівність $f(x_{n+3}^{\mathrm{k}}) > f(x_h^{\mathrm{k}})$ виконується трансформація яка підтягує $\forall i \ne l, \left\{\{x_i^{\mathrm{k}}\}/\{x_l^{\mathrm{k}}\}\right\}$, кожну точку за виключенням x_l^{k} до точки x_l^{k}, в якій функція приймає найменше значення в множині вершин полігона, згідно формули:

$$x_i^{\mathrm{k}} = x_l^{\mathrm{k}} + \sigma(x_i^{\mathrm{k}} - x_l^{\mathrm{k}}), \quad x_i^{\mathrm{k}} \ne x_l^{\mathrm{k}}, \quad \sigma = 0.5,$$

де σ – коефіцієнт гомотетії, $x_l^{(k)}$ точка гомотетії, $x_i^{(k)}$ – точка полігону. Після завершення операції алгоритм повертається до кроку (1), k збільшується на 1: $k = k + 1$.

5) Критерій зупинки алгоритму – перевірка на необхідність припинення обрахунків яка базується на формулі:

$$\sqrt{\frac{1}{n+1}\sum_{i=1}^{n+1}[f(x_i^{\mathrm{k}}) - f(x_{n+2}^{\mathrm{k}})]^2} \leq \varepsilon$$

де $\varepsilon \in \mathbb{R}$ – довільне мале число, $f(x_i^{\mathrm{k}})$ – значення функції в точці полігону, $f(x_{n+2}^{\mathrm{k}})$ – значення функції в центрі ваги полігона, n розмірність функції. Використовується перед кроком (1) після кожної ітерації.

Метод Нелдера-Міда, в порівнянні з симплекс методом: краще адаптується до топографії цільової функції, розтягуючись вздовж довгих нахилених площин, змінюючи напрямок в викривлених впадинах і стискуючись в області мімімуму.

Коефіцієнт віддзеркалення α використовується коли необхідно відобразити точку з найбільшим значенням функції через центроїд – зміщений центр ваги полігона (багатокутника). Коефіцієнт розтягнення γ необхідний для того щоб розтягнути (збільшити) вектор пошуку у випадку якщо значення $f(x)$ менше ніж отримане в результаті операції віддзеркалення. Коефіцієнт стиснення β використовується у випадку коли значення $f(x)$ не менше ніж друге за величиною значення функції (після найбільшого) $f(x_g)$, отриманого до операції (1) – операції відображення. Ці коефіцієнти необхідні для адаптування багатокутника до топології функції.

Отже виникає питання – які параметри необхідно встановити для α, β, γ? Після того як габарити багатокутника потрібним чином встановлені, бажано зберігати топологію незмінною доки в цьому не з’явиться потреба, наприклад зміниться топологія функції. Для збереження сталого розміру багатокутника коефіцієнт α встановлюють 1. До того ж, Нелдер і Мід[2] показали, що при $\alpha = 1$, виконується менше обчислень ніж при $\alpha < 1$. В той же час , α не повинно перевищувати 1, тому що:

1) При $\alpha < 1$ багатокутник краще адаптується до топології функції, особливо коли необхідно змінити напрямок пошуку, зустрівши вигнуту впадину;
2) В області локального мінімуму розміри багатокутника повинні зменшуватись і велике α, в цьому випадку сповільнить пошук мінімуму.

Щоб підібрати доцільні значення β та γ, Нелдер та Мід[2] і Павіані[4] розв’язали багато задач використовуючи різні параметри для β та γ.

Для вирішення задач безумовної оптимізації, Нелдер та Мід рекомендували $\alpha = 1, \beta = 0.5$ і $\gamma = 2$. Розміри багатокутника та його орієнтація опосередковано впливали на швидкість роботи алгоритму, але α, β та γ мали значно більший ефект. Павіані зробив висновок, про те що не можливо остаточно обрати β та γ і зміна β відображається на швидкості підходу до мінімуму більш помітно ніж γ. Павіані рекомендує наступні діапазони вибору параметрів β та γ:

$$0.4 \leq \beta \leq 0.6$$

$$2.8 \leq \gamma \leq 3.0$$

Для $0 < \beta < 0.4$, алгоритм може зупинитися завчасно через сплющення багатокутника. У випадку, коли $\beta > 0.6$, можливо знадобиться надмірна кількість кроків, для досягнення бажаного результату.

В даній роботі ми будемо працювати з наведеним вище діапазоном значень і порівняємо результати, збережені у вигляді таблиць.

Також необхідно зауважити наступне: в програмній реалізації додано перевірку циклічності, і встановлено обмеження на повтор кожної вершини на 3 рази після чого, обирається наступна по вагомості оптимальна точка.

ПРОГРАМНА РЕАЛІЗАЦІЯ

Для програмної реалізації методу Нелдера-Міда розроблено прикладну програму яка закладається з окремих та незалежних модулів (одномірний пошук як окрема частина), за парадигму взяте ООП. Програма написана мовою Python версії 3.5. Графіки побудовані на основі зібраної інформації за допомогою пакету matplotlib. Коефіцієнти α, β, γ та σ, можна змінити або скорегувавши значення в коді, або через консоль під час виконання програми. Для роботи з безумовною або умовної оптимізацією, необхідно перед початком роботи програми змінити стандартні значення в коді, задати нерівність як умову, це можна зробити підчас виконання програми, або через консоль. В Python 3 вбудована функція $eval()$, яка аналізує строку і виконує як фрагмент коду, що є доволі зручним, бо робить код універсальним – користувачу не треба корегувати код щоб змінити параметри для роботи з наведеним методом.

РОЗРАХУНКОВА ЧАСТИНА

Графіки кореневої функції

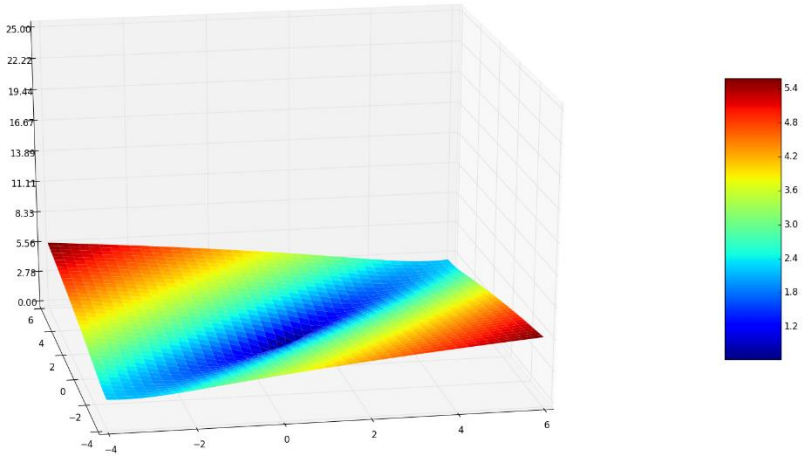


Рис. 2.1 Графік кореневої функції створений пакетом matplotlib

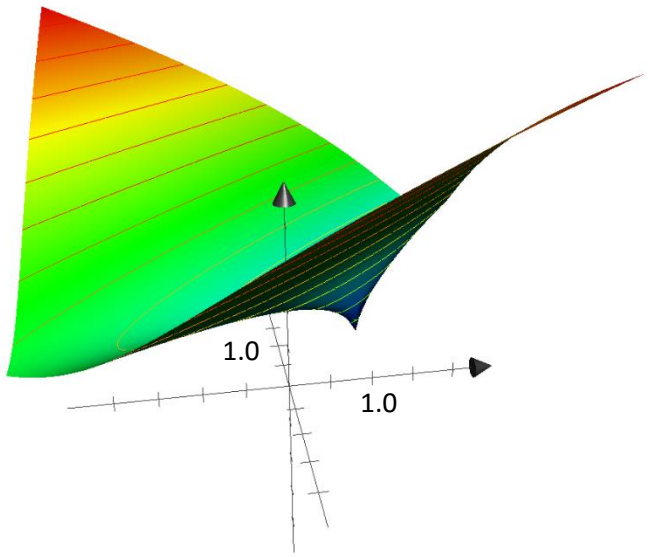


Рис. 2.2 Графік кореневої функції створений програмою Grapher

Знайдемо глобальний мінімум кореневої функції за допомогою WolframAlpha computational knowledge engine[5]:

Global minimum:

$$\min\left\{\left(10\,(x1-x2)^2+(x1-1)^2\right)^{0.25}\right\}=0 \text{ at } (x1, x2)=(1, 1)$$

Мінімум кореневої функції $f_3(x) = (10(x_1 - x_2)^2 + (x_1 - 1)^2)^{\frac{1}{4}}$, знаходиться у точці $x = (1; 1)$, де приймає значення 0.

2.1 Безумовна оптимізація

Розглянемо роботу методу Нелдера-Міда використовуючи наступні параметри.

Початкова точка $x^{(0)} = (-1{,}2; 0{,}0)$; тоді за правилом (*) встановимо полігон

$$x_1^{(0)} = (-1{,}2; 0{,}0),$$

$$x_2^{(0)} = (0.24888873943360235;\ 0.38822856765378105);$$

$$x_3^{(0)} = (-0.8117714323462188;\ 1.4488887394336023);$$

Встановимо коефіцієнти, як запропоновано Нелдером та Мідом:

$$\alpha = 1, \beta = 0.5, \gamma = 2, \sigma = 0.5;$$

В якості довжини ребра встановимо $t = 1.5$;

2.1.1 Результати виконання програми для різних параметрів ε

Для даного набору параметрів виконаємо аналіз оптимального значення точності ε, яке використовується у критерії зупинки (5), керуючись правилом:

$$\varepsilon = 10^{-j}, j = \overrightarrow{1, \dots, 10}$$

Занесемо результати роботи до таблиці:

Точність ε	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
10^{-1}	12	0.24888873943360235, 0.38822856765378105	0.9331763005920716
10^{-2}	37	0.2003601684768797, 0.20711747522458082	0.8943854267462801
10^{-3}	120	0.8093630537473296, 0.690933832536609	0.6482551622590498
10^{-4}	187	1.0040710547030762, 0.8865554178619368	0.6096223188005307
10^{-5}	203	1.0032325114580896, 0.8856881978791556	0.6096899312083286
10^{-6}	261	1.0219280141209233, 0.9047646864688641	0.6092218849382628
10^{-7}	298	1.023444921761996, 0.9063123932454936	0.6092181981001148
10^{-8}	301	1.0237568206219514, 0.9066305988455357	0.6092181406932987
10^{-9}	301	1.0237568206219514, 0.9066305988455357	0.6092181406932987
10^{-10}	2005	1.0954269286317593, 0.9881361579791765	0.593673072683594

Таблиця 2.1.1*

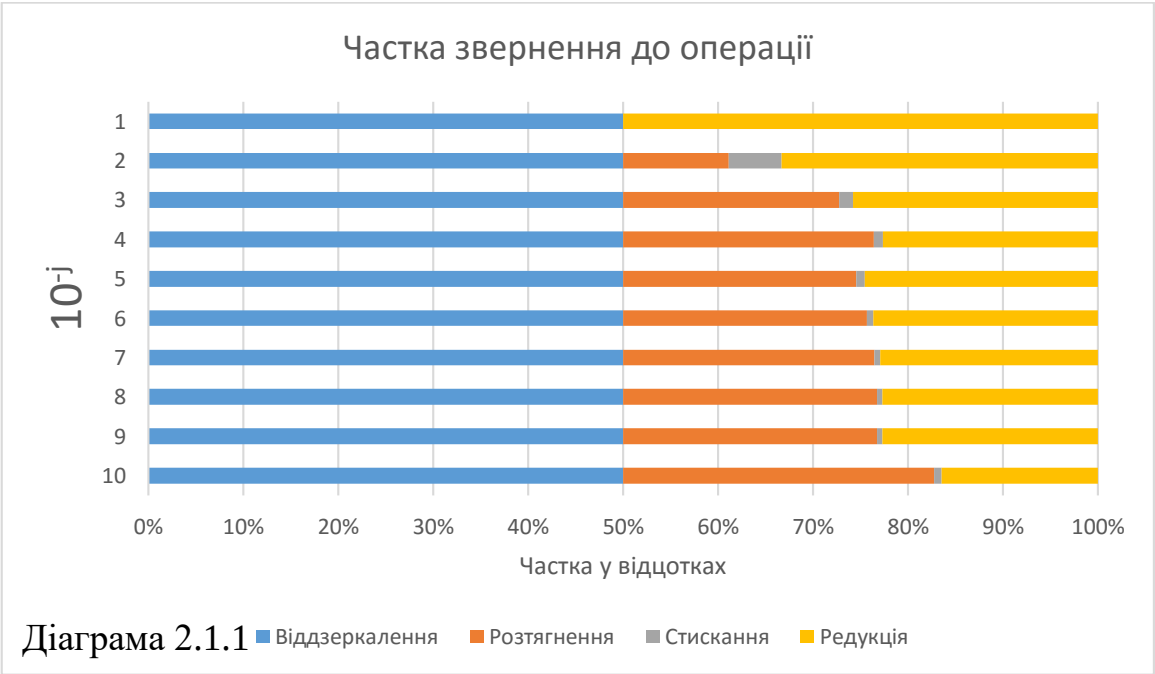
*у 3-4 стовпчиках дані про точку з найменшим значенням $f(x)$, останнього кроку

Необхідно зазначити що перед початком основної роботи алгоритм вираховує $n+1$ значення функції, тобто рахує значення в кожній точці багатокутника. На операцію редукції йде n обчислень функції оскільки ми обраховуємо кожну точку окрім тієї, до якої виконували гомотетію. n – розмірність функії.

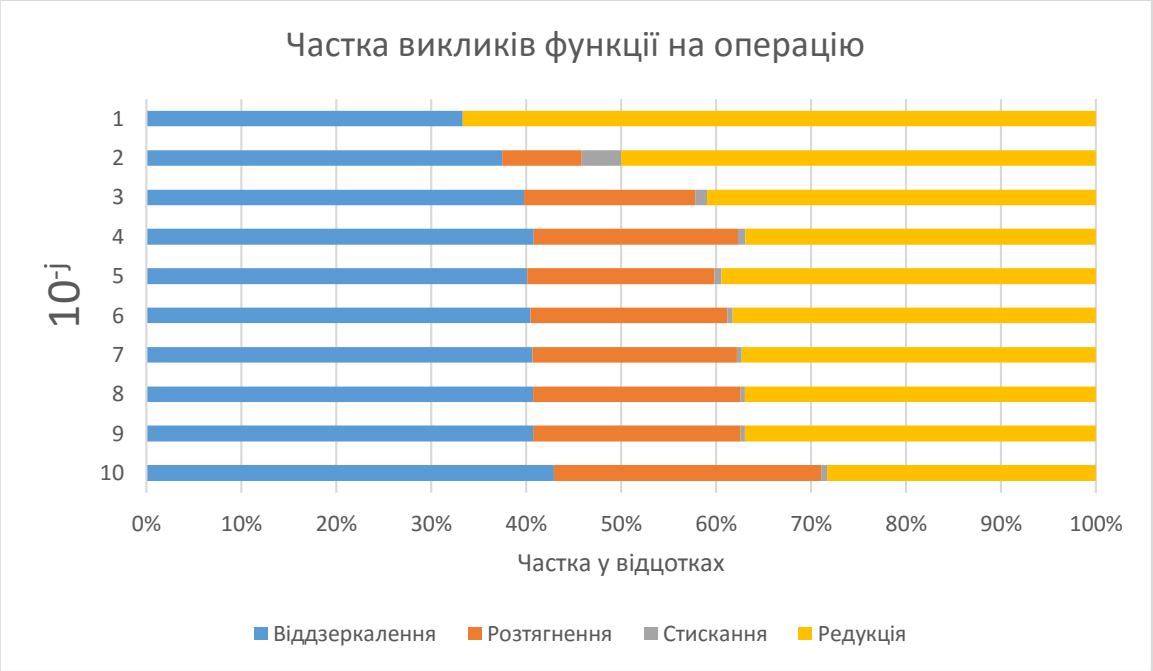
Для того щоб краще зрозуміти причину подібного розподілу кількостей викликів функції погляньмо на статистику виклику кожної операцій:

Точність ε	Віддзеркалення	Розтягнення	Стискання	Редукція	Σ
10^{-1}	2	0	0	2	4
10^{-2}	9	2	1	6	18
10^{-3}	33	15	1	17	66
10^{-4}	53	28	1	24	106
10^{-5}	57	28	1	28	114
10^{-6}	74	38	1	35	148
10^{-7}	85	45	1	39	170
10^{-8}	86	46	1	39	172
10^{-9}	86	46	1	39	172
10^{-10}	601	394	9	198	1202

Таблиця 2.1.2



Діаграма 2.1.1



Діаграма 2.1.2

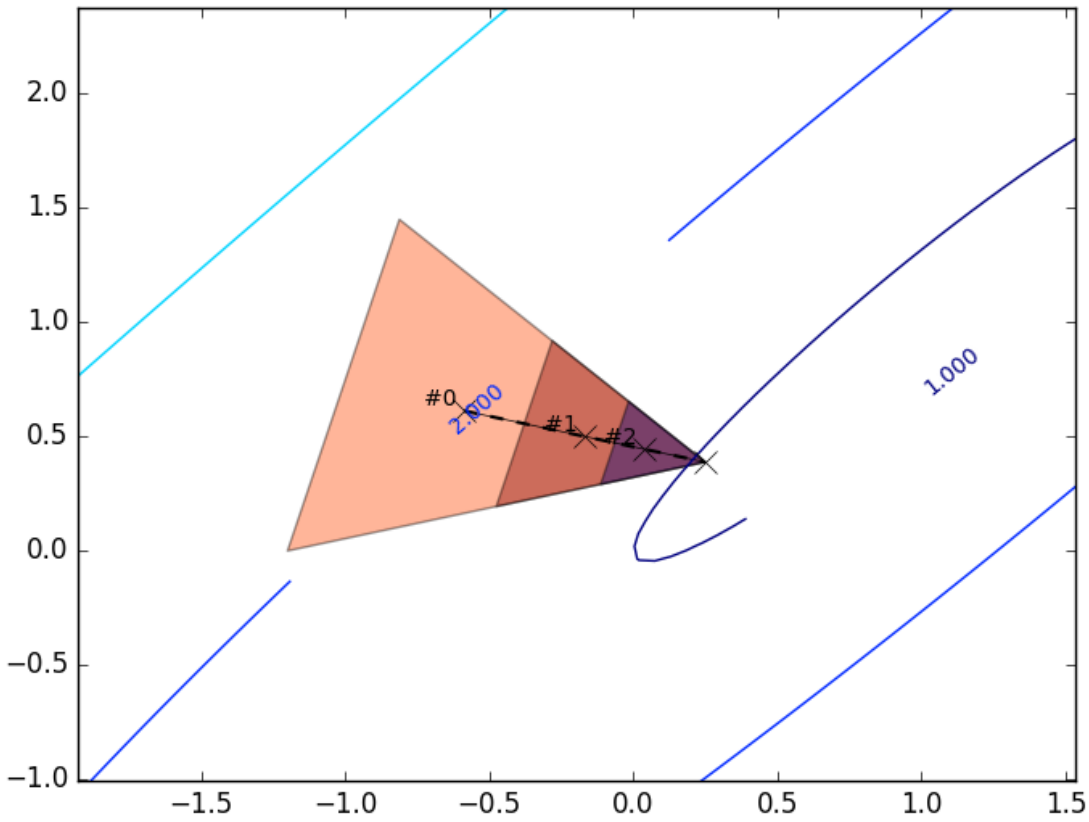


Рис 2.1.3 Графічне зображення руху полігона при $\varepsilon = 10^{-1}$

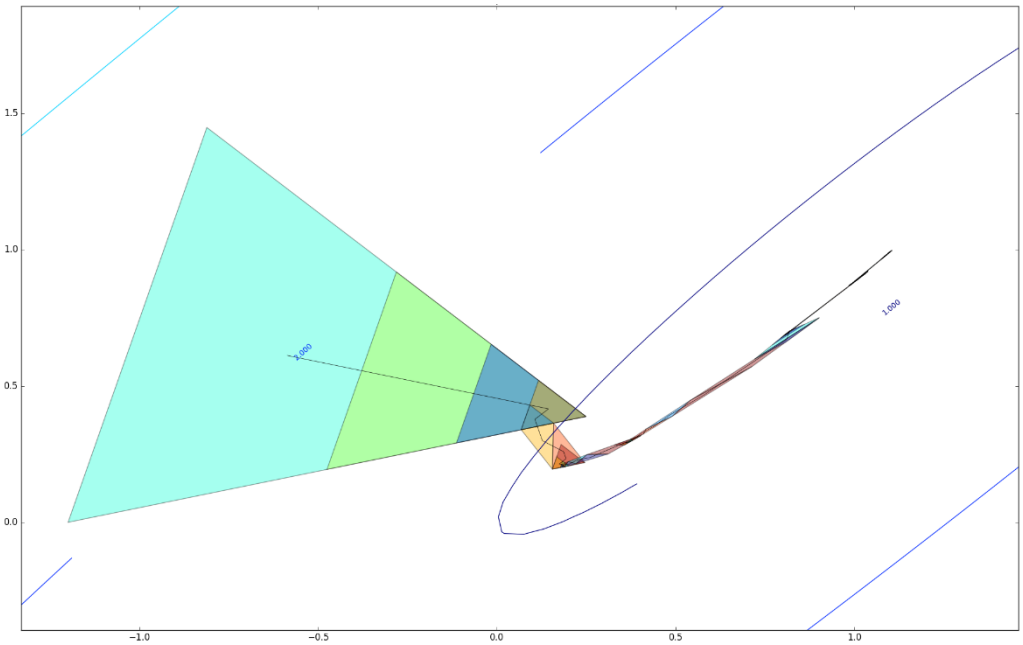


Рис 2.1.4 Графічне зображення руху полігона при $\varepsilon = 10^{-10}$

Варто звернути увагу на те, що кількість звернень до операції не зменшується зі збільшенням точності, оскільки ε задіяна в критерії зупинки і збільшення точності відкриває нову частину шляху, не змінюючи старої. Хоча це може мати місце лиш у даному випадку оскільки програма обирає найбільшу току для відбиття, або найменшу для редукції, якщо, наприклад взяти два числа 0.54 і 0.53: якщо провести округлення до одного знака після коми, отримаємо два однакових числа, підставивши ці числа в деяку функцію по черзі, отримаємо однакові значення, що може призвести до зміни напрямку руху багатокутника, і може пришвидшити момент зупинки алгоритму.

Проведемо модифікації коефіцієнтів для можливої зміни кількості обчислень, та близькості відходження до області мінімуму.

2.1.2 Результати виконання програми для різних коефіцієнтів α

Почнемо з α. Як було наведено в теоретичній частині, α не повинно перевищувати 1, а ось зробити меншим може бути доцільним.

Змінюємо для кожної групи ітерацій α на 0.1 за наступним правилом:

$$\alpha = 0.1 \times j, j = \overrightarrow{j = 9, \dots, 1}$$

Від 9-ти тому що минулий приклад розглядався з $\alpha = 1$. Але включимо результати для $\alpha = 1$ з минулих обчислень для демонстрації змін.

Параметр $\varepsilon = 10^{-5}$.

Параметри $\beta = 0.5$ та $\gamma = 2$, залишимо без змін.

Так само початкова точка і параметри побудови, багатокутника залишаються без змін.

Бажано щоб кількість викликів функції була менша за 600.

Занесемо отримані результати в таблицю, у форматі таблиці 2.1 з наступними стовпчиками: параметр α, кількість обчислень функції, координати точки, значення функції в цій точці, де точка береться з найменшим значенням серед точок багатокутника на останньому кроці.

Значення α	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
1.0	203	1.0032325114580896, 0.8856881978791556	0.6096899312083286
0.9	155	0.41634760170900886, 0.3143987176871879	0.8165615277303465
0.8	131	0.3652035057727145, 0.3692106449782674	0.7968205296868667
0.7	78	0.23077291530094024, 0.24888648470328106	0.8782692103021248
0.6	74	0.21446511435670712, 0.2709347165884527	0.8975390384677683
0.5	74	0.1987577574046839, 0.2907420609875982	0.9232597147508231
0.4	68	0.1518775732251879, 0.362234504011435	1.0382070356526485
0.3	68	0.1518775732251879, 0.362234504011435	1.0382070356526485
0.2	68	0.1518775732251879, 0.362234504011435	1.0382070356526485
0.1	68	0.1518775732251879, 0.362234504011435	1.0382070356526485

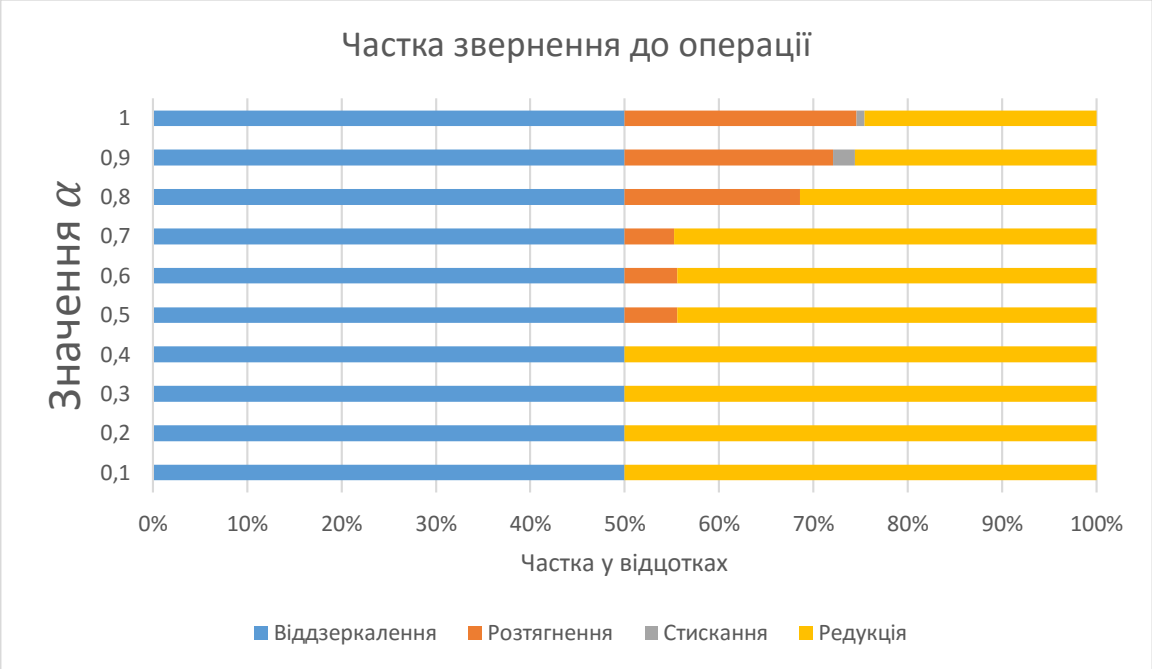
Таблиця 2.1.3

Для того щоб краще зрозуміти причину подібного розподілу кількостей викликів функції погляньмо на статистику виклику кожної операцій:

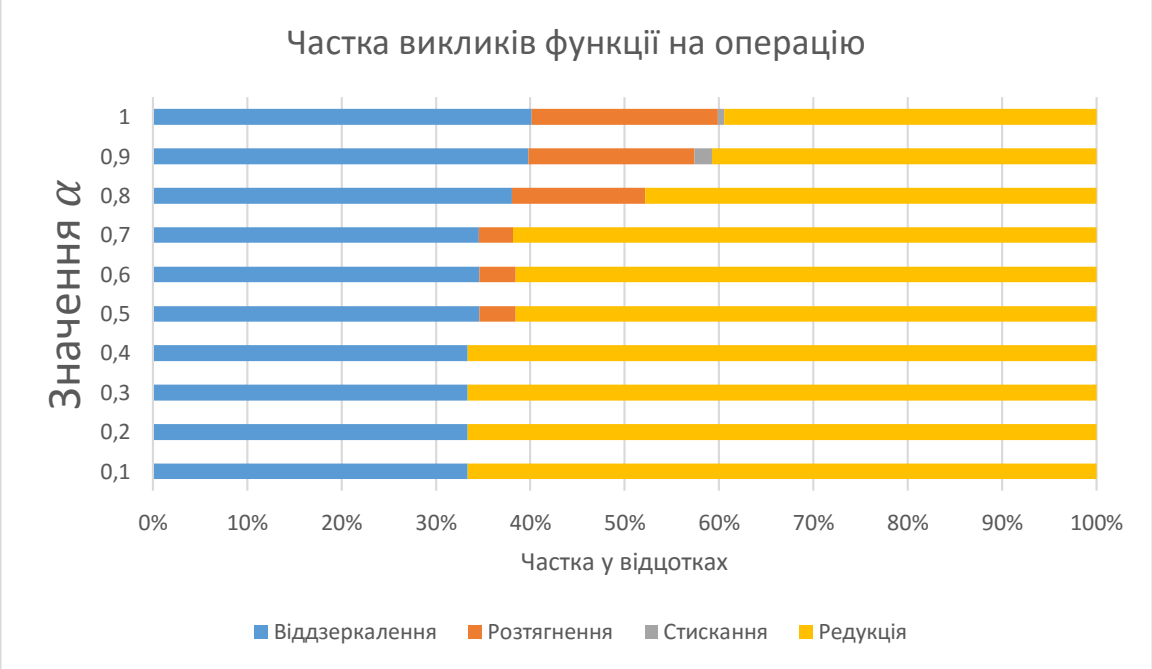
Значення α	Віддзеркалення	Розтягнення	Стискання	Редукція	Σ
1.0	57	28	1	28	114
0.9	43	19	2	22	86
0.8	35	13	0	22	70
0.7	19	2	0	17	38
0.6	18	2	0	16	32
0.5	18	2	0	16	32
0.4	16	0	0	16	32

0.3	16	0	0	16	32
0.2	16	0	0	16	32
0.1	16	0	0	16	32

Таблиця 2.1.4



Діаграма 2.1.3



Діаграма 2.1.4

Коефіцієнт α використовується на кроці (1) – віддзеркалення, він також є пошуковим кроком, і від нього залежать наступні кроки, приймаючи не оптимальне значення він викликає редукцію. В наведених результатах можна побачити як операції віддзеркалення та редукції(4), починаючи з $\alpha = 0{,}4$ ділять між собою, кількість викликів функції. Також спостерігається передчасна

зупинка алгоритму. Дійсно найкращий результат у $\alpha = 1$. Отже коефіцієнт α залишиться без змін.

Необхідно додати, що можна спробувати модифікувати метод і змінювати коефіцієнт віддзеркалення підходячи до області мінімуму і разом з ним і коефіцієнт розтягнення, але в цьому випадку метод втрачає свою простоту.

2.1.3 Результати виконання програми для різних коефіцієнтів β

Як було наведено в теоретичній частині, β не повинно перевищувати 0.6, але модифікації почнемо з 0.9 і завершимо значенням 0.1.

Змінюємо для кожної групи ітерацій β на 0.1 за наступним правилом:

$$\beta = 0.1 \times j, j = \overrightarrow{j = 9, \dots, 1}$$

Від 9-ти тому що, брати $\beta = 1$, не має сенсу, дія повертатиме початкове значення точки не змінюючи її (не враховуючи похибки). Включимо результати для $\beta = 0.5$ з минулих обчислень для демонстрації змін.

Параметр $\varepsilon = 10^{-5}$.

Параметри $\alpha = 1$ та $\gamma = 2$, залишимо без змін.

Так само початкова точка і параметри побудови, багатокутника залишаються без змін.

Бажано щоб кількість викликів функції була менша за 600.

Занесемо отримані результати в таблицю, у форматі таблиці 2.1 з наступними стовпчиками: параметр β, кількість обчислень функції, координати точки, значення функції в цій точці, де точка береться з найменшим значенням серед точок багатокутника на останньому кроці.

Значення β	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
0.9	229	0.268382784838004, 0.20700800486186083	0.8700132022561105
0.8	275	0.9977432289450077, 1.007967189165425	0.18002685619236397
0.7	150	0.20828595982498066, 0.17334222534866106	0.8940852144480415
0.6	120	0.4177984176338255, 0.31305213035751234	0.8184333877618895
0.5	203	1.0032325114580896, 0.8856881978791556	0.6096899312083286
0.4	501	0.9999368089833492, 1.0002278495682133	0.030372980206696973

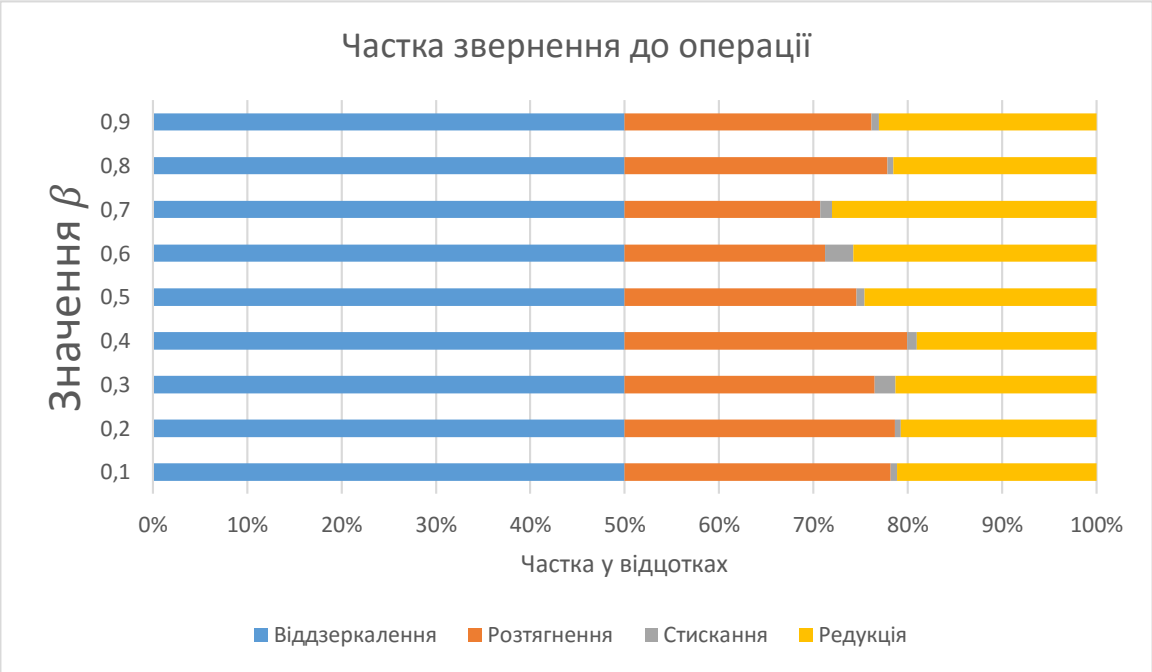
0.3	237	1.0071129539526065, 1.0056160224252202	0.09243446214676719
0.2	284	0.9171090796451777, 0.9098806193159956	0.2932318992408709
0.1	247	0.4791198648885466, 0.5246268477997387	0.7351145132312127

Таблиця 2.1.5

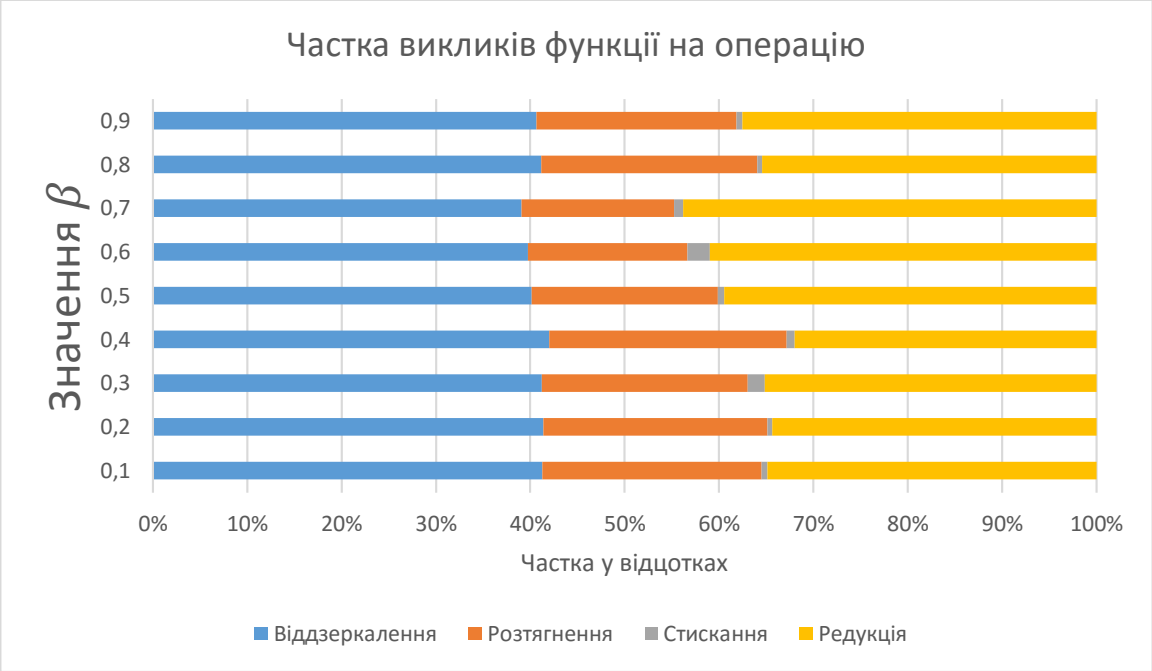
Для того щоб краще зрозуміти причину подібного розподілу кількостей викликів функції погляньмо на статистику виклику кожної операцій:

Значення β	Віддзеркалення	Розтягнення	Стискання	Редукція	Σ
0.9	65	34	1	30	130
0.8	79	44	1	34	158
0.7	41	17	1	23	82
0.6	33	14	2	17	66
0.5	57	28	1	28	114
0.4	147	88	3	56	294
0.3	68	36	3	29	136
0.2	82	47	1	34	164
0.1	71	40	1	30	142

Таблиця 2.1.6



Діаграма 2.1.5



Діаграма 2.1.6

Розглянемо отримані результати. Якщо відкинути $\beta = 0.5$, доволі цікавими будуть $\beta = 0.8$ і $\beta = 0.3$, звісно при $\beta = 0.4$ точність найбільша але 501 обчислення функції все перекреслюють. Тому варто встановити $\beta = 0.3$

2.1.4 Результати виконання програми для різних коефіцієнтів γ

Як було наведено в теоретичній частині, $\gamma \in [2.8; 3.0]$ але модифікації почнемо з 3.0 і завершимо значенням 1.1.

Змінюємо для кожної групи ітерацій β на 0.1 за наступним правилом:

$$\gamma = 0.1 \times j, j = \ j = \overrightarrow{30, \dots, 11}$$

До 11-ти тому що, брати $\gamma = 1$, не має сенсу, дія повертатиме значення аналогічне до результату операції віддзеркалення. Включимо результати для $\gamma = 2$ з минулих обчислень для демонстрації змін.

Параметр $\varepsilon = 10^{-5}$.

Параметри $\alpha = 1$ та $\beta = 0.5$, залишимо без змін.

Так само початкова точка і параметри побудови, багатокутника залишаються без змін.

Бажано щоб кількість викликів функції була менша за 600.

Занесемо отримані результати в таблицю, у форматі таблиці 2.1 з наступними стовпчиками: параметр γ, кількість обчислень функції, координати точки, значення функції в цій точці, де точка береться з найменшим значенням серед точок багатокутника на останньому кроці.

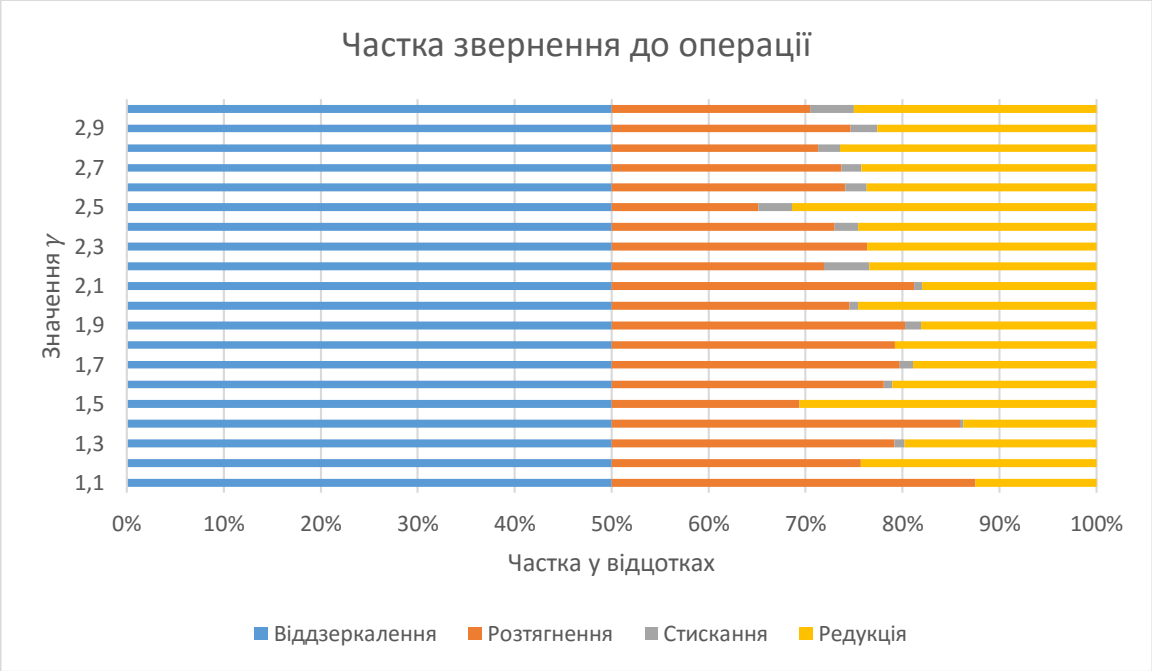
Значення γ	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
3.0	235	0.9999999837685604, 0.9999999805706363	0.0001382894981904044 7
2.9	256	0.999568062451912, 0.9990668494306484	0.040531180645267104
2.8	244	0.9998450963454689, 1.0001009938855199	0.028703863830782383
2.7	342	1.0000000052865923, 1.0000000204263846	0.000219470318953841
2.6	400	0.999999950014371, 0.999999952723	0.00022519812911140312
2.5	160	0.9999999528589394, 0.9999999465387601	0.00022628034688537547
2.4	224	1.0000029457280328, 1.0000026576989962	0.0017559412050990903
2.3	320	1.0485361027662725, 1.0749935015741539	0.3110061579108618
2.2	226	1.0000003887455964, 1.0000003510215132	0.000637681488311221
2.1	424	0.9476375214689348, 0.978616082220347	0.33328514413750165
2.0	203	1.0032325114580896, 0.8856881978791556	0.6096899312083286
1.9	404	0.9996668401244629, 0.9989811989885797	0.04683625406599486
1.8	308	1.115421261218125, 1.155081083276925	0.41284837501970484
1.7	254	0.8554394767324126, 0.7510966370071357	0.6001990061855931
1.6	199	0.697775550840626, 0.5788831295121677	0.6945382277447459
1.5	116	0.3121208105770803, 0.2549018584925592	0.8433736027800514
1.4	492	0.6141684433684478, 0.7430710411417121	0.749180011607416
1.3	167	0.3119906486170284, 0.24170825642154176	0.8503038322169705
1.2	126	0.30477778123 73, 0.2554330815146619	0.8441082399280401
1.1	238	0.25637835502685585, 0.22438730739485999	0.8662975623797903

Таблиця 2.1.7

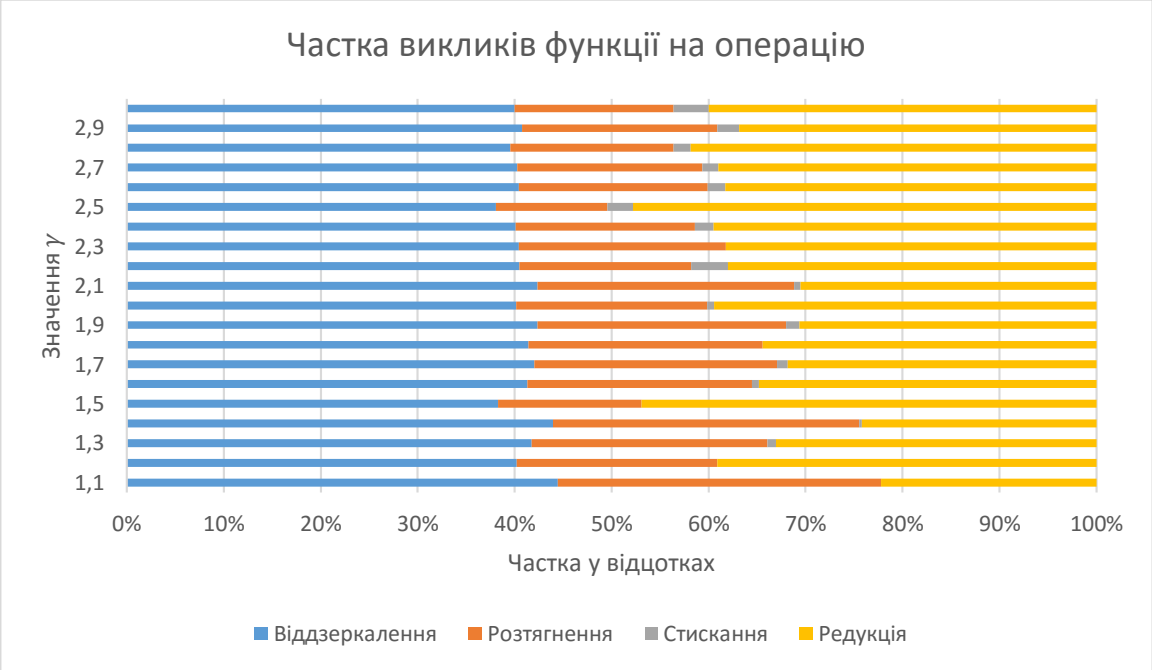
Для того щоб краще зрозуміти причину подібного розподілу кількостей викликів функції погляньмо на статистику виклику кожної операцій:

Значення γ	Віддзеркалення	Розтягнення	Стискання	Редукція	Σ
3.0	66	27	6	33	132
2.9	73	36	4	33	146
2.8	68	29	3	36	136
2.7	97	46	4	47	194
2.6	114	55	5	54	228
2.5	43	13	3	27	86
2.4	63	29	3	31	126
2.3	91	48	0	43	182
2.2	64	28	6	30	128
2.1	125	78	2	45	250
2.0	57	28	1	28	114
1.9	119	72	4	43	238
1.8	89	52	0	37	178
1.7	74	44	2	28	148
1.6	57	32	1	24	114
1.5	31	12	0	19	62
1.4	149	107	1	41	298
1.3	48	28	1	19	96
1.2	35	18	0	17	70
1.1	72	54	0	18	144

Таблиця 2.1.8



Діаграма 2.1.7



Діаграма 2.1.8

Розглянемо отримані результати. Хороший результат отриманий при $\gamma = 3.0$

2.1.5 Результати виконання програми для різних коефіцієнтів σ

Модифікацію σ почнемо зі значення 0.9 і завершимо значенням 0.1.

Змінюємо для кожної групи ітерацій σ на 0.1 за наступним правилом:

$$\sigma = 0.1 \times j, j = \overrightarrow{j = 9, \dots, 1}$$

Від 9-ти тому що, брати $\sigma = 1$, не має сенсу, дія повертатиме початкове значення точок не змінюючи їх (не враховуючи похибки). Включимо результати для $\sigma = 0.5$ з минулих обчислень для демонстрації змін.

Параметр $\varepsilon = 10^{-5}$.

Параметри $\alpha = 1, \beta = 0.5$ та $\gamma = 2$, залишимо без змін.

Так само початкова точка і параметри побудови, багатокутника залишаються без змін.

Бажано щоб кількість викликів функції була менша за 600.

Занесемо отримані результати в таблицю, у форматі таблиці 2.1 з наступними стовпчиками: параметр σ, кількість обчислень функції, координати точки, значення функції в цій точці, де точка береться з найменшим значенням серед точок багатокутника на останньому кроці.

Значення σ	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
0.9	875	1.000000001675245, 1.0000000009489265	5.331844333414005e-05
0.8	356	1.0000001937588383, 1.0000000702215994	0.0006603559094238416
0.7	117	0.27890803663212893, 0.24287090494212193	0.8544245706540695
0.6	520	0.999999998037547, 0.9999999977972994	4.587351037240816e-05
0.5	203	1.0032325114580896, 0.8856881978791556	0.6096899312083286
0.4	349	0.999999984204246, 0.9999999961015082	0.00020199963386688098
0.3	79	0.3746523047129079, 0.40328944578927023	0.7949029584990152
0.2	112	0.49311226677624775, 0.48159696930596	0.7128773052615556
0.1	88	0.33235118711079964, 0.34173820314100556	0.8175013060795896

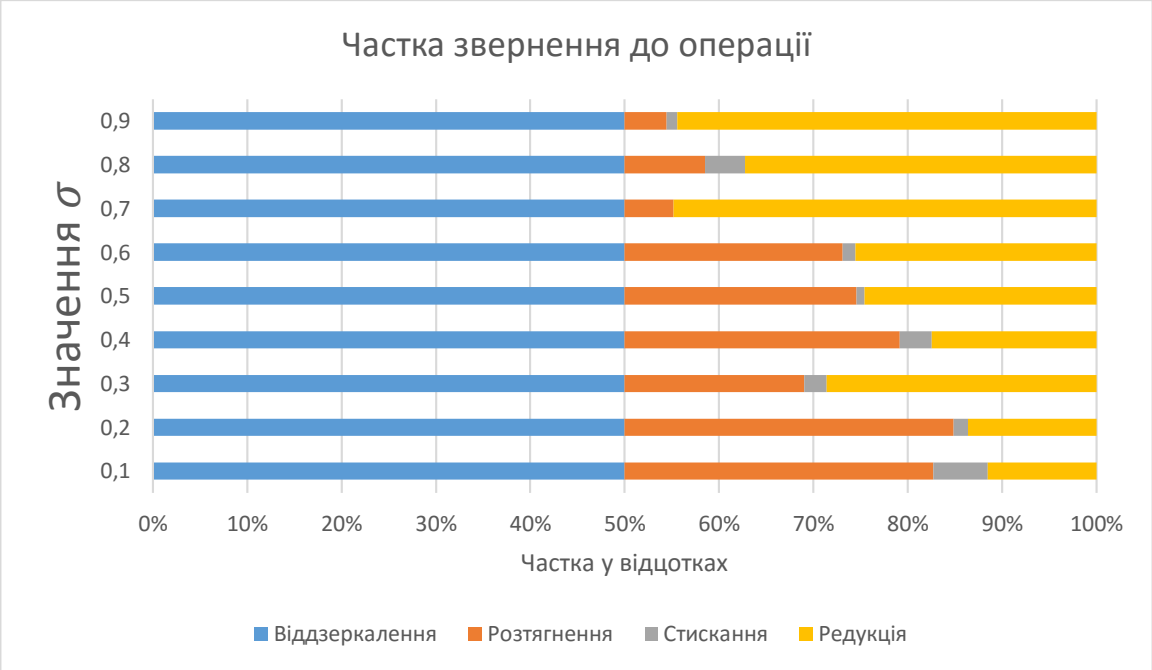
Таблиця 2.1.9

Для того щоб краще зрозуміти причину подібного розподілу кількостей викликів функції погляньмо на статистику виклику кожної операцій:

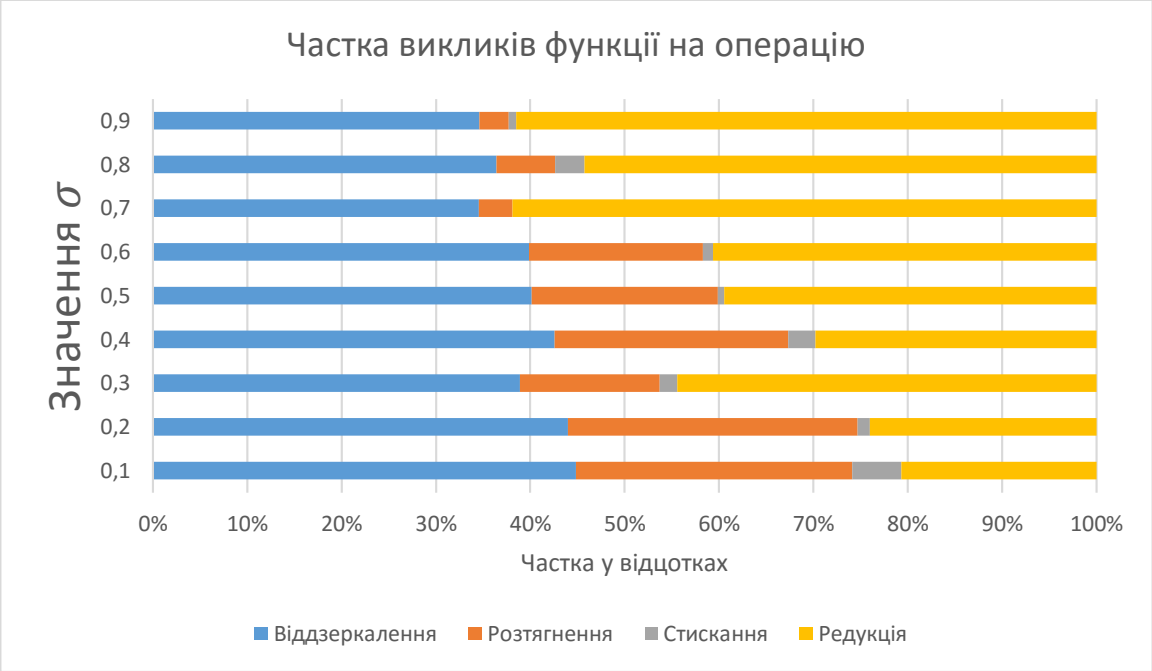
Значення σ	Віддзеркалення	Розтягнення	Стискання	Редукція	Σ
0,9	224	20	5	199	448
0,8	94	16	8	70	188

0,7	29	3	0	26	58
0,6	147	68	4	75	294
0,5	57	28	1	28	114
0,4	103	60	7	36	206
0,3	21	8	1	12	42
0,2	33	23	1	9	66
0,1	26	17	3	6	52

Таблиця 2.1.10



Діаграма 2.1.9



Діаграма 2.1.10

Розглянемо отримані результати. Найкращій результат досягається при $\sigma = 0.4$, інші результати не такі точні, хоч і досягнуті меншою кількістю ітерацій. Тому варто встановити $\sigma = 0.4$

2.1.6 Результати виконання програми для різних значень параметра t

Модифікацію t почнемо зі значення 2.0 і завершимо значенням 1.0.

Змінюємо для кожної групи ітерацій t на 0.1 за наступним правилом:

$$t = 0.1 \times j, j = \ j = \overrightarrow{20, \dots, 10}$$

Параметр $\varepsilon = 10^{-5}$.

Параметри $\alpha = 1, \beta = 0.5, \gamma = 2$ та $\sigma = 0.5$ залишимо без змін.

Так само початкова точка і параметри побудови, багатокутника залишаються без змін.

Бажано щоб кількість викликів функції була менша за 600.

Занесемо отримані результати в таблицю, у форматі таблиці 2.1 з наступними стовпчиками: параметр σ, кількість обчислень функції, координати точки, значення функції в цій точці, де точка береться з найменшим значенням серед точок багатокутника на останньому кроці.

Значення t	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
2	182	0.9999999908336099, 0.9999999886015599	0.00010755957083407969

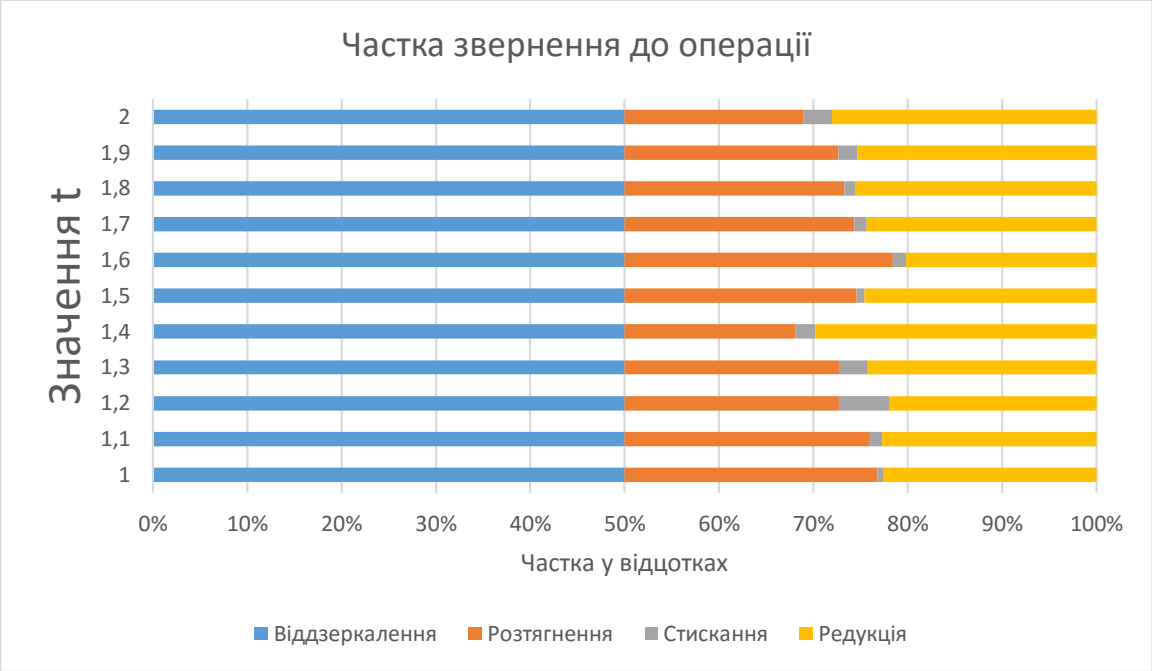
1,9	267	1.000144766068169, 1.0001195300363488	0.012857104439363345
1,8	155	0.5715229938128435, 0.5285032219599276	0.6704885071223441
1,7	283	0.9999999372827717, 0.9999998837038384	0.00042504910535185944
1,6	603	0.9999999898332399, 0.9999999913878325	0.00010626826581436475
1,5	203	1.0032325114580896, 0.8856881978791556	0.6096899312083286
1,4	173	1.0000001132912124, 1.000000124761002	0.0003448995565244207
1,3	234	1.000021225102661, 0.9999328667327478	0.016739734193679117
1,2	231	1.0000001971332284, 1.0000001539162278	0.0004897673444671403
1,1	249	1.0423263377153698, 1.023758880035643	0.269037358951627

Таблиця 2.1.11

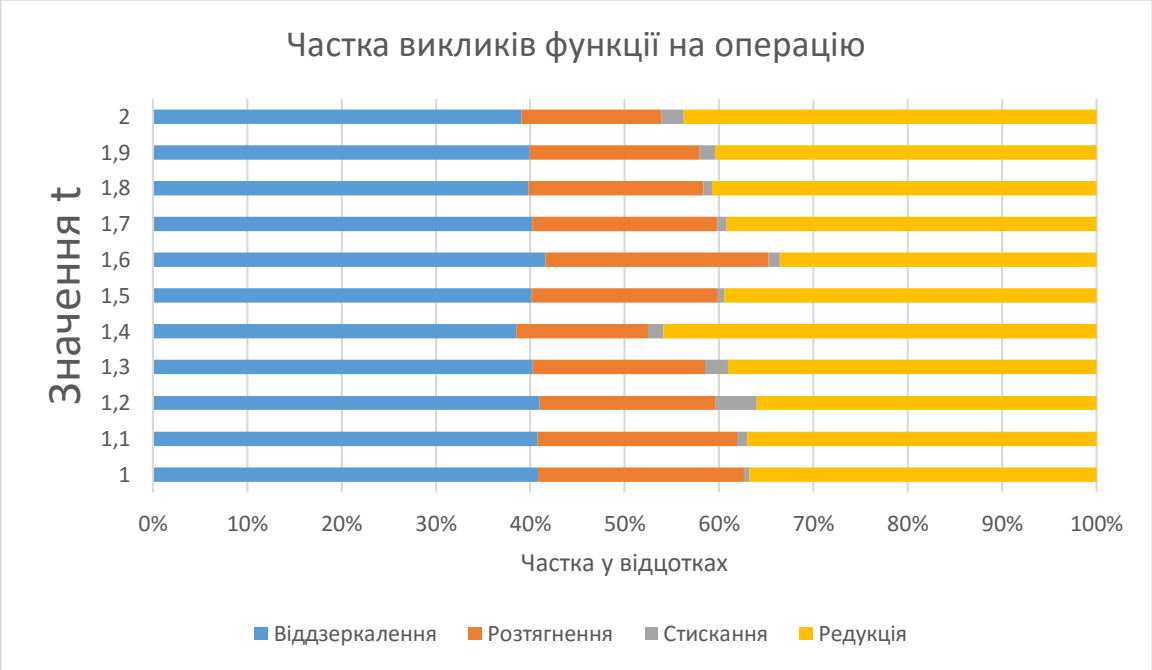
Для того щоб краще зрозуміти причину подібного розподілу кількостей викликів функції погляньмо на статистику виклику кожної операцій:

Значення t	Віддзеркалення	Розтягнення	Стискання	Редукція	Σ
2	50	19	3	28	100
1,9	75	34	3	38	150
1,8	43	20	1	22	86
1,7	80	39	2	39	160
1,6	176	100	5	71	352
1,5	57	28	1	28	114
1,4	47	17	2	28	94
1,3	66	30	4	32	132
1,2	66	30	7	29	132
1,1	77	40	2	35	154

Таблиця 2.1.12



Діаграма 2.1.11



Діаграма 2.1.12

Розглянемо отримані результати. Найкращий результат досягнутий при $t = 2$. Можливо це пов’язано з тим що при збільншенні розмірів багатокутника область мінімуму теж стає ближчою.

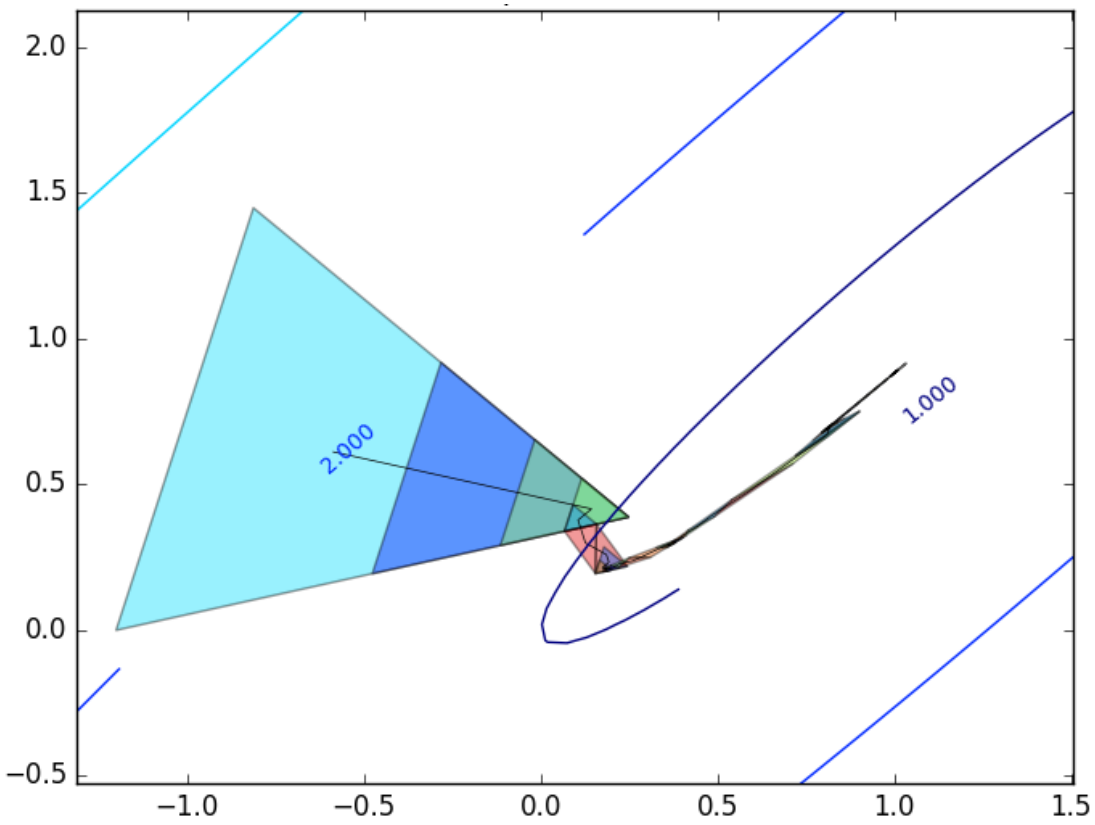


Рис 2.1.5 Графічне зображення руху полігона при $t = 1.5$

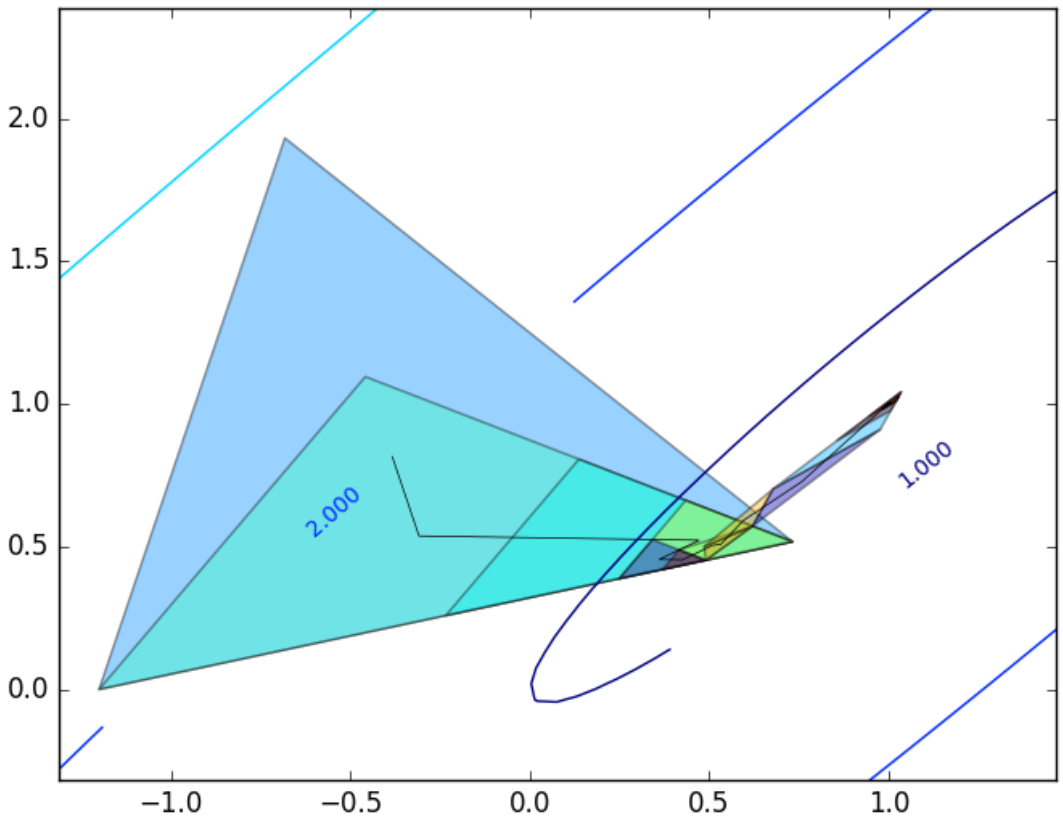


Рис 2.1.6 Графічне зображення руху полігона при $t = 2.0$

2.1.7 Результати виконання програми для нових $\alpha, \beta, \gamma, \sigma, t$

Виходячи з здійсненої роботи задамо нові значення для параметрів $\alpha, \beta, \gamma, \sigma, t$.

$$\alpha = 1,$$

$$\beta = 0.3,$$

$$\gamma = 3,$$

$$\sigma = 0.4,$$

$$t = 2;$$

Для даного набору параметрів виконаємо аналіз оптимального значення точності ε, яке використовується у критерії зупинки (5), керуючись правилом:

$$\varepsilon = 10^{-j}, j = \overrightarrow{1, \dots, 10}$$

Занесемо отримані результати в таблицю, у форматі таблиці 2.1 з наступними стовпчиками: параметр ε, кількість обчислень функції, координати точки, значення функції в цій точці, де точка береться з найменшим значенням серед точок багатокутника на останньому кроці.

Точність ε	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
10^{-1}	15	0.7318516525781364, 0.5176380902050414	0.8535485648322192
10^{-2}	32	0.6371683883556649, 0.5989961302267724	0.618372550947606
10^{-3}	147	1.0000008069115613, 0.99999785324854	0.0030618772777359564
10^{-4}	163	1.0000035519563304, 1.0000037219741784	0.0018953670469108556
10^{-5}	217	0.9999999757697573, 0.9999999641747473	0.00020964105297079734
10^{-6}	276	0.9999999999708289, 0.9999999999651631	5.850982734838354e-06
10^{-7}	294	0.9999999999797383, 0.9999999999754488	4.937933296063816e-06
10^{-8}	323	0.9999999999896241, 0.9999999999838276	4.589446871621433e-06
10^{-9}	344	0.9999999999918832, 0.9999999999858371	4.557511150138357e-06

10^{-10}	428	0.9999999999925961, 0.9999999999866329	4.5009754934048645e-06

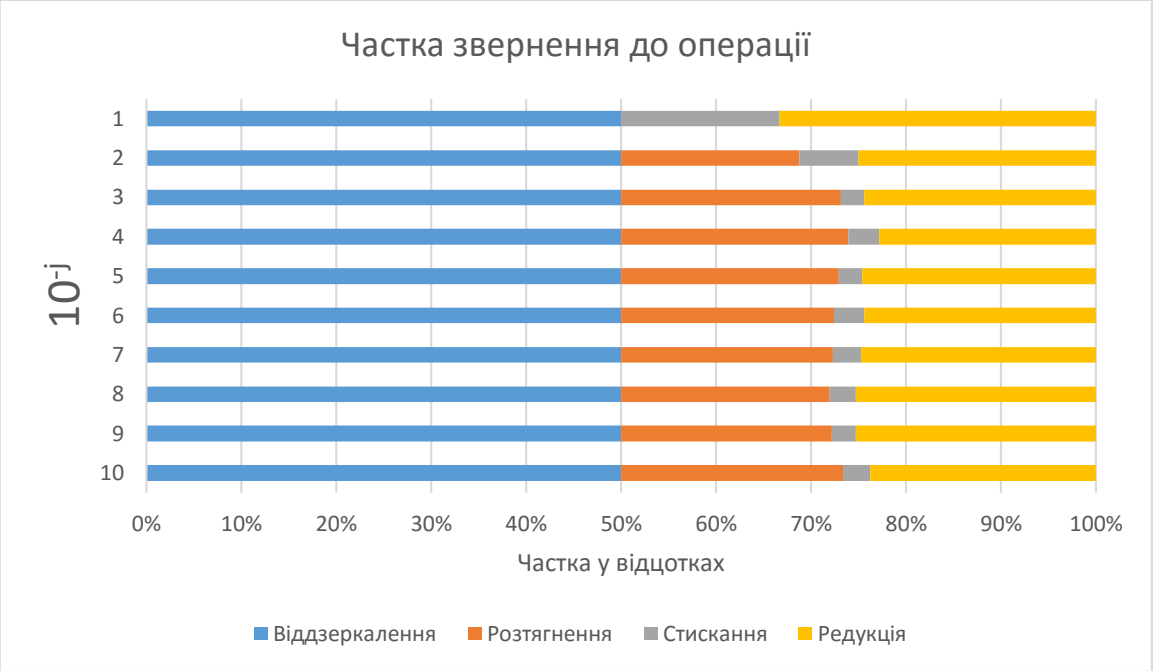
Таблиця 2.1.13

Необхідно зазначити що перед початком основної роботи алгоритм вираховує $n+1$ значення функції, тобто рахує значення в кожній точці багатокутника. На операцію редукції йде n обчислень функції оскільки ми обраховуємо кожну точку окрім тієї, до якої виконували гомотетію. n – розмірність функії.

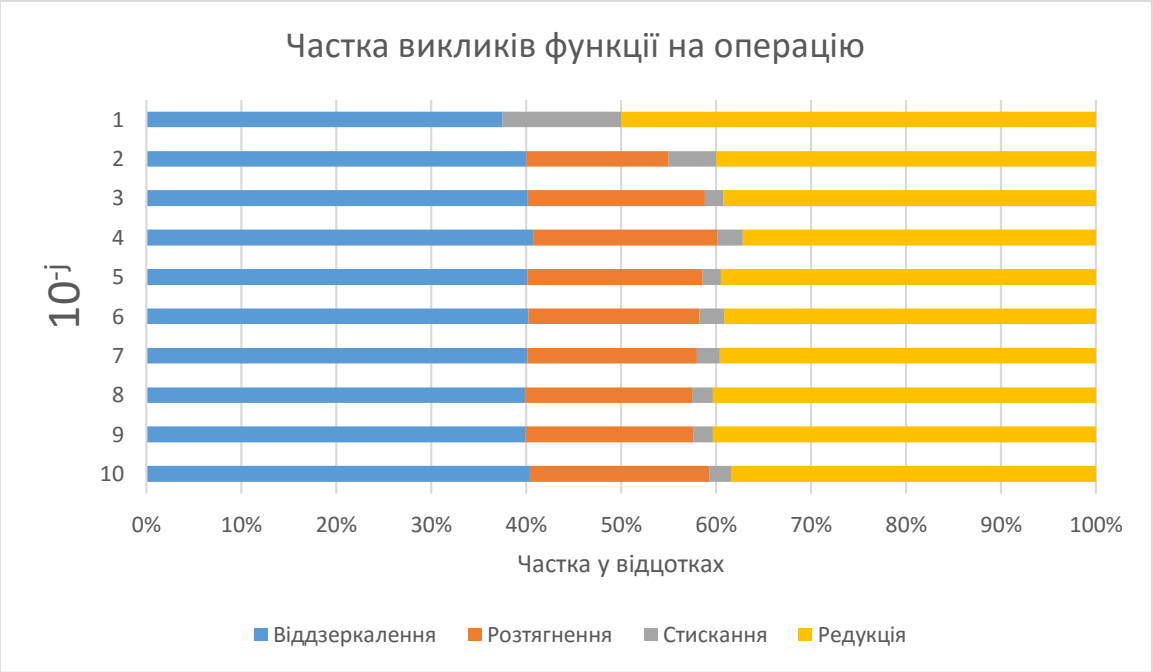
Для того щоб краще зрозуміти причину подібного розподілу кількостей викликів функції погляньмо на статистику виклику кожної операцій:

Точність ε	Віддзеркалення	Розтягнення	Стискання	Редукція	Σ
10^{-1}	3	0	1	2	6
10^{-2}	8	3	1	4	16
10^{-3}	41	19	2	20	82
10^{-4}	46	22	3	21	92
10^{-5}	61	28	3	30	122
10^{-6}	78	35	5	38	156
10^{-7}	83	37	5	41	166
10^{-8}	91	40	5	46	182
10^{-9}	97	43	5	49	194
10^{-10}	122	57	7	58	244

Таблиця 2.1.14



Діаграма 2.1.13



Діаграма 2.1.14

З отриманих результатів можна зробити висновок що зміна коефіцієнтів деформації та розміру сторони багатокутника для початкової точки

$$x^0 = (-1.2; 0.0),$$

виявилося вдалим рішенням, тому параметри

$$\alpha = 1, \beta = 0.3, \gamma = 3, \sigma = 0.4, t = 2, \varepsilon = 10^{-5};$$

будемо використовувати в умовній оптимізації.

2.2 Умовна оптимізація

Наведемо початкові умови для умовної оптимізації.

$$x^0 = (-1.2; 0.0),$$

$$\alpha = 1, \beta = 0.3, \gamma = 3, \sigma = 0.4, t = 2, \varepsilon = 10^{-5};$$

WolframAlpha computational knowledge engine

2.2.1 Результати виконання для випуклої допустимої області

Задамо умову в якості нерівності $(x_1 - a)^2 + (x_2 - b)^2 \leq R$, що має вигляд кола з радіусом $\sqrt{R}$, і центром в точці $(a; b)$. Для даної умови перевірити знаходження вершини багатокутника в допустимій області можна за допомогою нерівності:

$$\sqrt{\left(x_1^k - a\right)^2 + \left(x_2^k - b\right)^2} \leq \sqrt{R}$$

Ліва частина є формулою знаходження відстані між точками. В даному випадку між центром кола $(a; b)$ та деякою вершиною x_1^k, на k-й ітерації.

Розглянемо роботу алгоритму з умовою $(x_1^2 + 1.2) + (x_2^2 - 0) \leq 4$, де центром кола виступає початкова точка $x^0 = (-1.2; 0.0)$, а радіус 2;

Для початку перевіримо умову за допомогою WolframAlpha computational knowledge engine[5]:

$$\min\left\{\left(10\,(x1 - x2)^2 + (x1 - 1)^2\right)^{0.25} \mid (x1 + 1.2)^2 + x2^2 \leq 4\right\} \approx 0.564147 \text{ at } (x1, x2) \approx (0.682839, 0.674474)$$

Порівняємо графіки з глобальним мінімумом та мінімумом в заданій області:

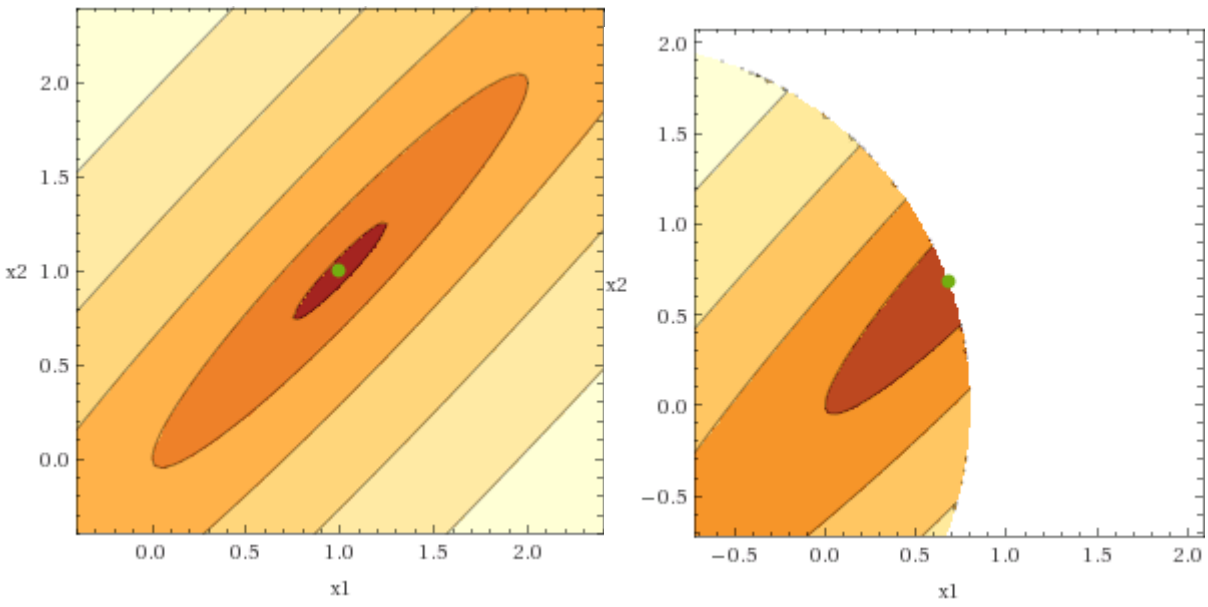


Рис 2.2.1 Порівння на графіках глобального мінімуму та на заданій області

Використовуючи здобуті на етапі безумовної оптимізації параметри, перевіримо роботу алгоритму для заданих умов, і занесем дані у таблицю.

Точність ε	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
10^{-1}	15	0.7318516525781364, 0.5176380902050414	0.8535485648322192
10^{-2}	35	0.6371683883556649, 0.5989961302267724	0.618372550947606
10^{-3}	38	0.654916227013246, 0.6168344785523454	0.6045600269996627
10^{-4}	122	0.6806528419072689, 0.6774471598403463	0.5652503998607546
10^{-5}	173	0.6809891226701507, 0.679518374699039	0.5648404896559822
10^{-6}	221	0.6822051007850544, 0.6753650216898957	0.5643847650964801
10^{-7}	236	0.6822712168517221, 0.6751322338256625	0.564384439186267
10^{-8}	344	0.6820377176965269, 0.6762938479592716	0.5643409111817962
10^{-9}	519	0.6823659350235444, 0.6751443437208322	0.5643172350655798
10^{-10}	823	0.6823065986341069, 0.6759599603541259	0.5642044946656677

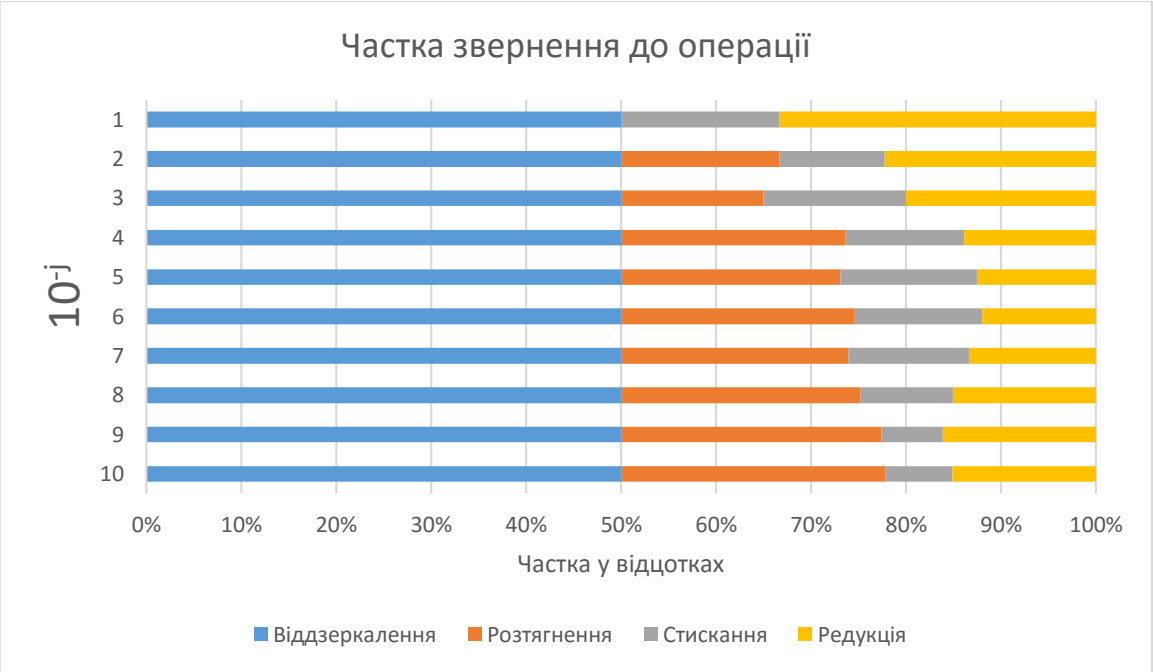
Таблиця 2.2.1

Для того щоб краще зрозуміти причину подібного розподілу кількостей викликів функції погляньмо на статистику виклику кожної операцій:

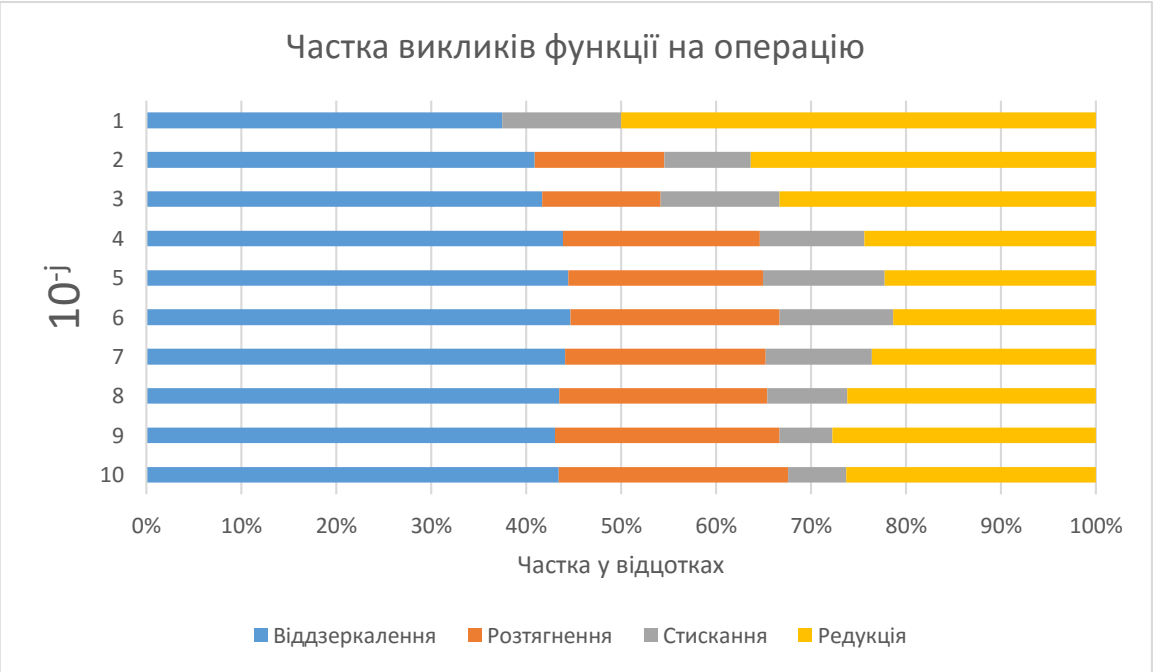
Точність ε	Віддзеркалення	Розтягнення	Стискання	Редукція	Σ
10^{-1}	3	0	1	2	6
10^{-2}	9	3	2	4	18
10^{-3}	10	3	3	4	20
10^{-4}	36	17	9	10	72
10^{-5}	52	24	15	13	104
10^{-6}	67	33	18	16	134
10^{-7}	71	34	18	19	142

10^{-8}	103	52	20	31	206
10^{-9}	155	85	20	50	310
10^{-10}	248	138	35	75	496

Таблиця 2.2.2



Діаграма 2.2.1



Діаграма 2.2.2

Варито звернути увагу на збільшення кількості виконання «стискання» в порівнянні з безумовною оптимізацією при аналогічних параметрах.

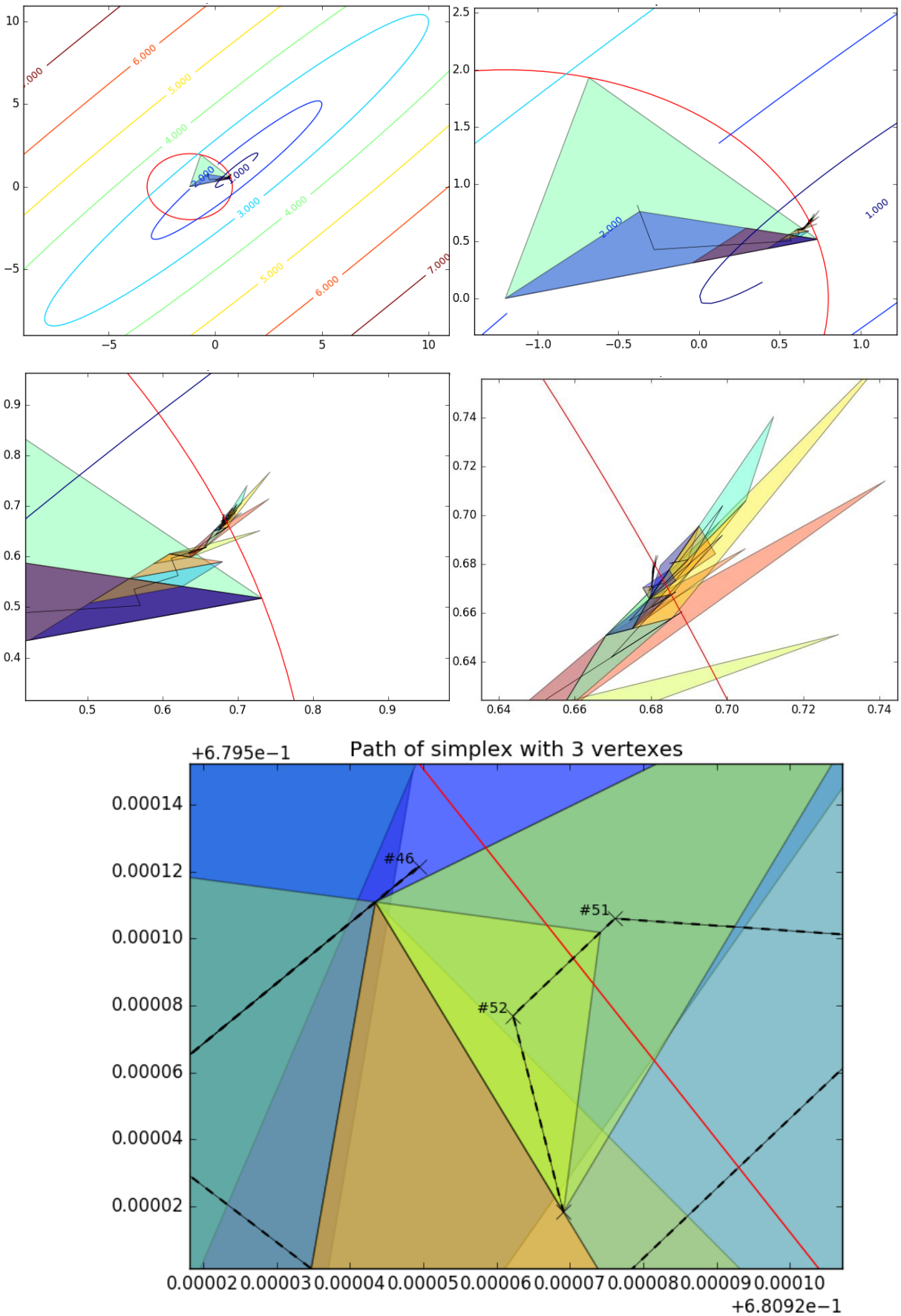


Рис 2.2.2 Графічна історія переміщення полігону при $\varepsilon = 10^{-5}$

На наведених графіках (Рис 2.2.2) можна побачити як багатокутник б'ється об бар'єр і зменшується в розмірах.

Принцип роботи алгоритму при потрапляні «лапи» або вершини багатокутника за допустиму облать простий: зміній значення фунції на вершині встановлюється значення – нескінченість, на мові Python це здійснюжться операцією $float('Inf')$, після чого метод повертається на крок (1), віддзеркалення, описаний в теретичній частині, і обирає нову послідовність дій з урахуванням корректив.

Розглянувши результати можна побачити зміни у розподілені викликів обчислення функції, що пов'язано з бар'єром який виник завдяки встановленю умови.

При $\varepsilon = 10^{-10}$, ми отримаємо найкрашій результат, для точності, але 823 ітерації забагато, варто залишити $\varepsilon = 10^{-5}$.

Перевіримо вплив зміни довжини ребра на кількість обчислень функції.

Занесемо результати в таблицю.

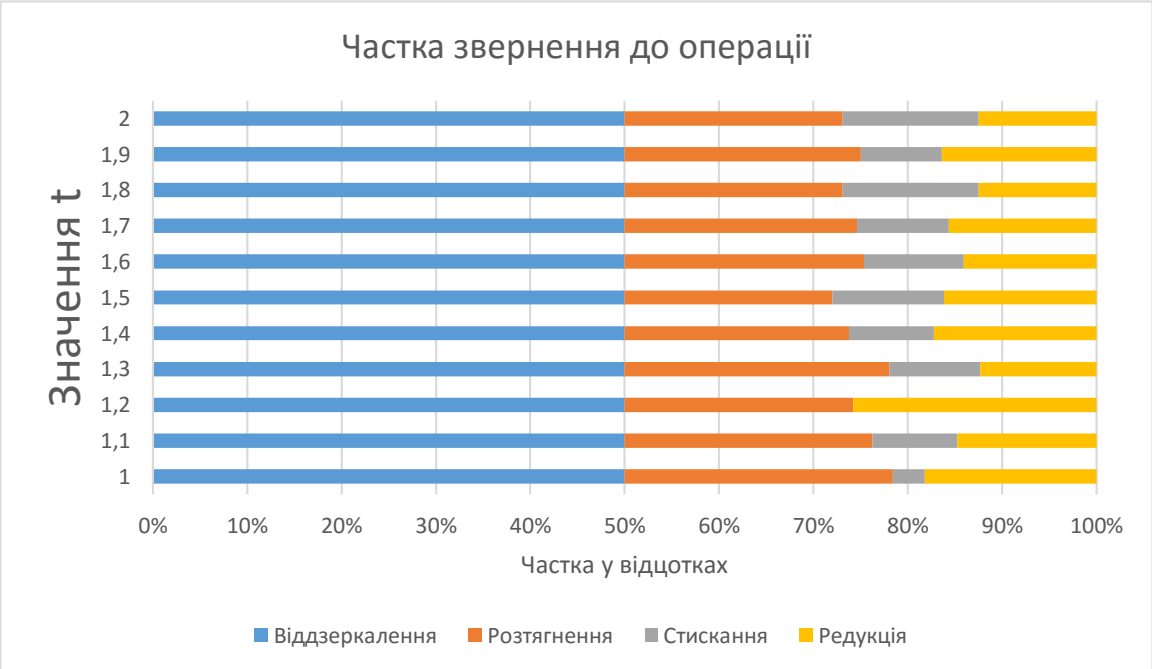
Значення t	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
2	173	0.6809891226701507, 0.679518374699039	0.5648404896559822
1,9	197	0.6821756368638126, 0.676239068344584	0.564250227642806
1,8	173	0.6817262991942342, 0.6775419763391607	0.564401131382724
1,7	226	0.6883047503707721, 0.6589695781096999	0.5702690447336478
1,6	237	0.6832630477481568, 0.6732409565395784	0.5641972905145367
1,5	230	0.690360552388207, 0.6530833023957745	0.5756034783954945
1,4	208	0.6655040818567859, 0.7209904762549704	0.6145920081118773
1,3	189	0.6828110612412578, 0.672619940951698 2	0.5646431731487614
1,2	120	0.24449524792902286, 0.32463798554223167	0.8926807496200319
1,1	261	0.6824785871504635, 0.6754407658763236	0.5641811936422498
1	152	0.507699737968528, 0.4702760206263495	0.7115650988913935

Таблиця 2.1.11

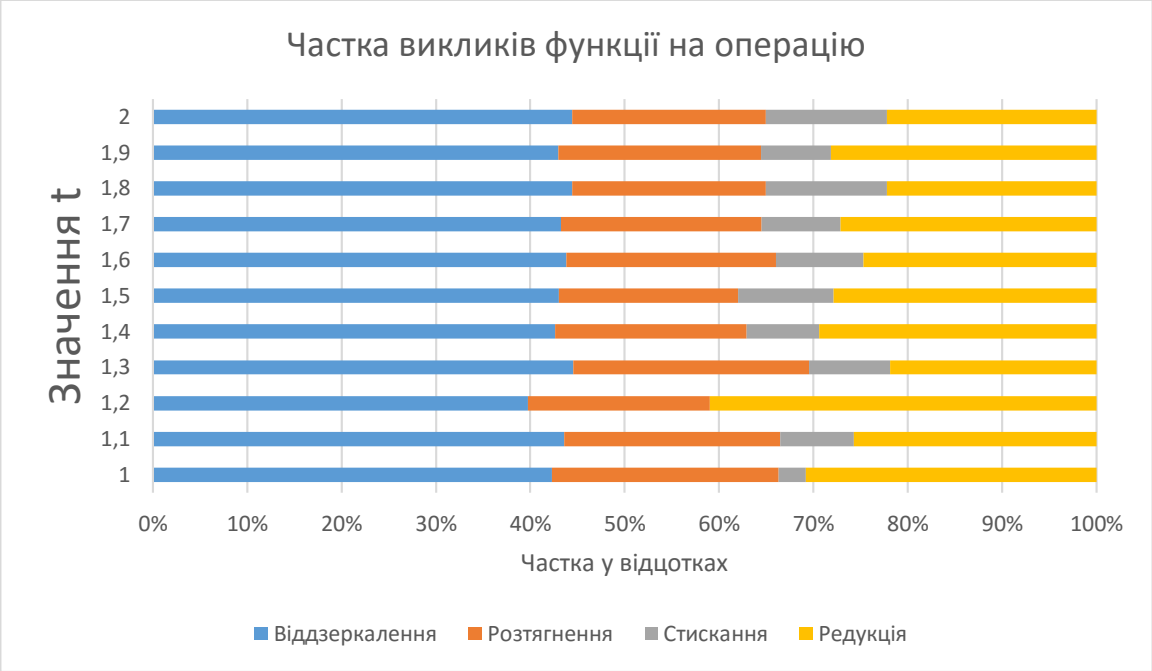
Для того щоб краще зрозуміти причину подібного розподілу кількостей викликів функції погляньмо на статистику виклику кожної операцій:

Значення t	Віддзеркалення	Розтягнення	Стискання	Редукція	Σ
2	52	24	15	13	104
1,9	58	29	10	19	116
1,8	52	24	15	13	104
1,7	67	33	13	21	134
1,6	71	36	15	20	142
1,5	68	30	16	22	136
1,4	61	29	11	21	122
1,3	57	32	11	14	114
1,2	33	16	0	17	66
1,1	78	41	14	23	156
1	44	25	3	16	88

Таблиця 2.2.x



Діаграма 2.2.x



Діаграма 2.2.x

Розглянемо отримані результати. Найкращий результат досягнутий при $t = 1.1$. Можливо це пов'язано з тим що зменшившись достатньо швидко багатокутник ближче підходить до локального мінімуму.

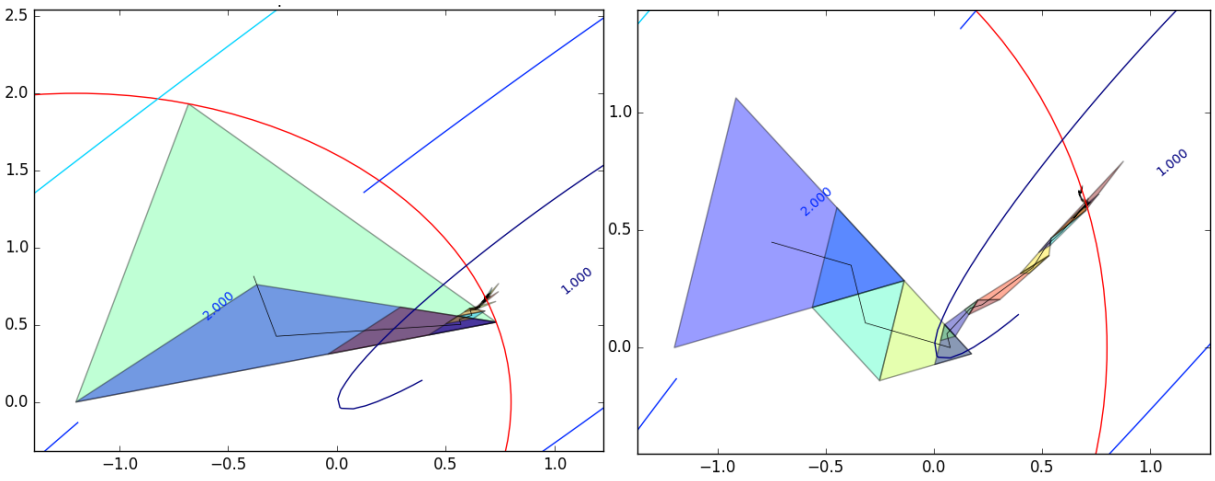


Рис 2.2.x Графічне зображення руху полігона при $t = 2.0$, і $t = 1.1$

2.3.1 Результати виконання для невипуклої допустимої області

Задамо умову в якосі нерівностей

$$\begin{cases}(x_1 - a)^2 + (x_2 - b)^2 \leq R_1, \\ (x_1 - a)^2 + (x_2 - b)^2 \leq R_2\end{cases}$$

що маю вигляд кола з радіусом $\sqrt{R_1}$ і $\sqrt{R_2}$, і центром в точі $(a; b)$. Для даної умови перевірити знаходження вершини багатокутника в допустимій області можна за допомогою нерівності:

$$\begin{cases}\sqrt{\left(x_1^k - a\right)^2 + \left(x_2^k - b\right)^2} \leq \sqrt{R_1} \\ \sqrt{\left(x_1^k - a\right)^2 + \left(x_2^k - b\right)^2} \leq \sqrt{R_2}\end{cases}$$

Ліва частина є формулою знаходження відстані між точками. В даному випадку між центром кола $(a; b)$ та деякою вершиною x_1^k, на k-й ітерації.

Розглянемо роботу алгоритму з умовою

$$\begin{cases}(x_1 + 1.2)^2 + x^2 \leq 2.25, \\ (x_1 + 1.2)^2 + x^2 \geq 1\end{cases}$$

де центром кола виступає початкова точка $x^0 = (-1.2; 0.0)$, а радіус 1.5 та 1

Для початку перевіримо умову за допомогою WolframAlpha computational knowledge engine[5]:

$$\min\left\{\left(10\,(\mathrm{x1} - \mathrm{x2})^2 + (\mathrm{x1} - 1)^2\right)^{0.25} \mid 1. \leq (\mathrm{x1} + 0.1)^2 + (\mathrm{x2} - 2.1)^2 \leq 2.25\right\} \approx 0.50067$$
$$\text{at } (\mathrm{x1}, \mathrm{x2}) \approx (0.976218, 1.05513)$$

Порівняємо графіки з глобальним мінімумом та мінімумом в заданій області:

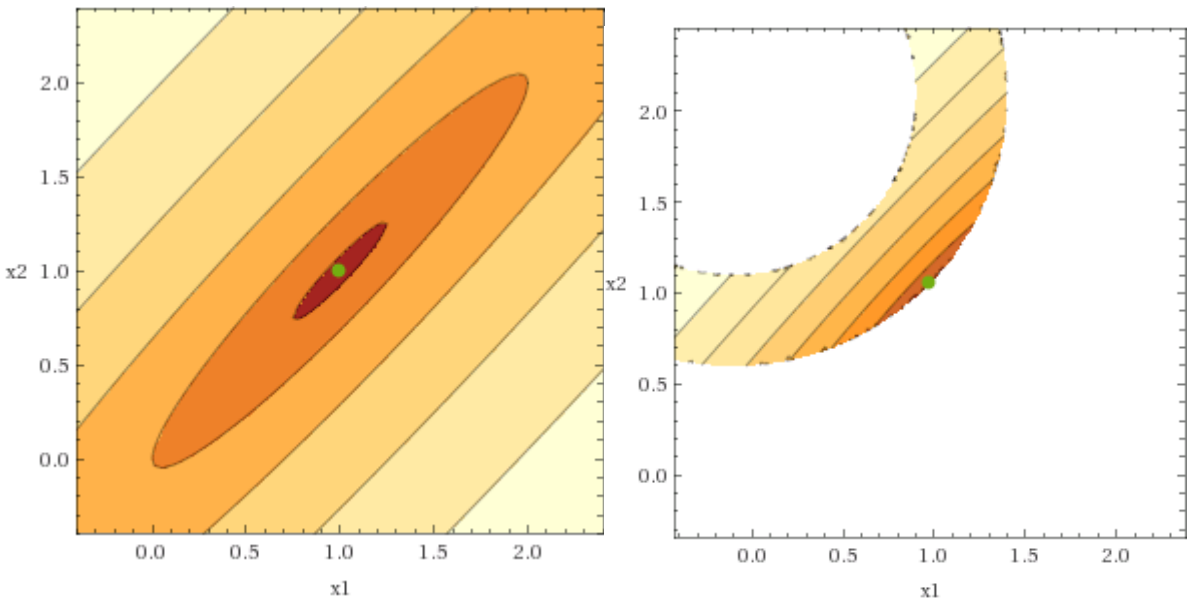


Рис 2.2.1 Порівння на графіках глобального мінімуму та на заданій області

Використовуючи здобуті на етапі безумовної оптимізації параметри, перевіримо роботу алгоритму для заданих умов, і занесем дані у таблицю.

Точність ε	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
10^{-1}	37	-0.271811926719967, 0.8846302955589415	1.9676974158453318
10^{-2}	147	1.0673137661519119, 1.1581831612913827	0.5432616423335246
10^{-3}	147	1.0673137661519119, 1.1581831612913827	0.5432616423335246
10^{-4}	238	0.9529737480669501, 1.0319558470108434	0.5041348026849249
10^{-5}	238	0.9529737480669501, 1.0319558470108434	0.5041348026849249
10^{-6}	417	0.9686853473821786, 1.0474262792124653	0.5009611269987349
10^{-7}	435	0.9686899569441274, 1.0474308869602298	0.5009605472883538
10^{-8}	519	0.9686914275103741, 1.047432363652406	0.5009603833596683
10^{-9}	712	0.9717113256505706, 1.0505073442516695	0.500774945656014
10^{-10}	712	0.9717113256505706, 1.0505073442516695	0.500774945656014

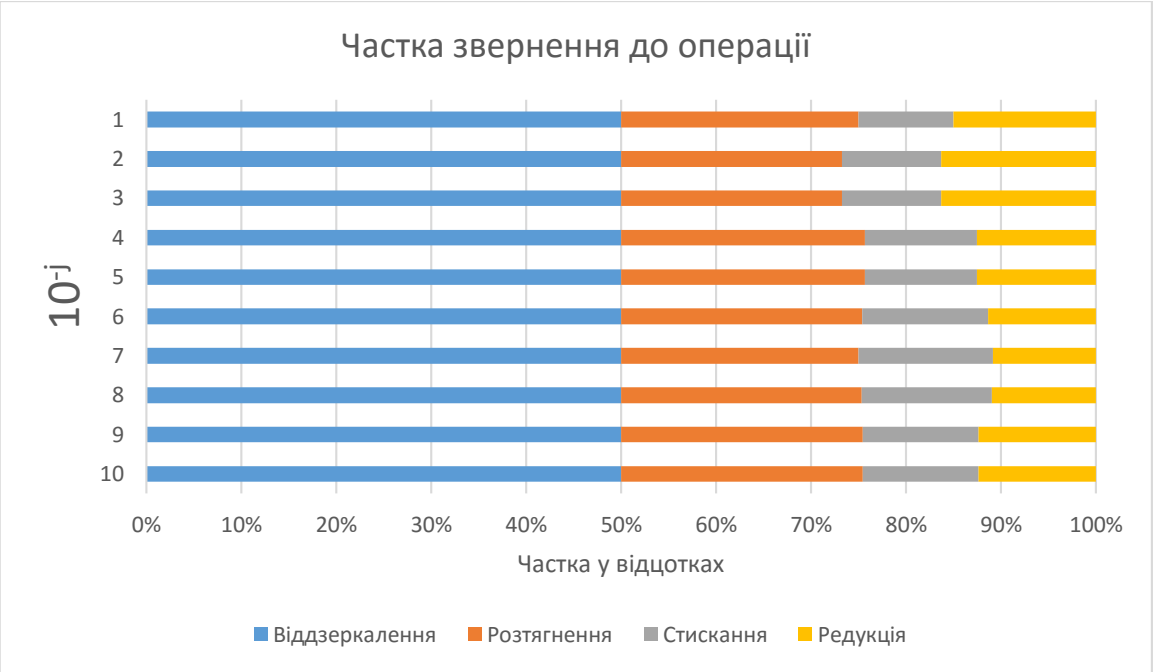
Таблиця 2.2.1

Для того щоб краще зрозуміти причину подібного розподілу кількостей викликів функції погляньмо на статистику виклику кожної операцій:

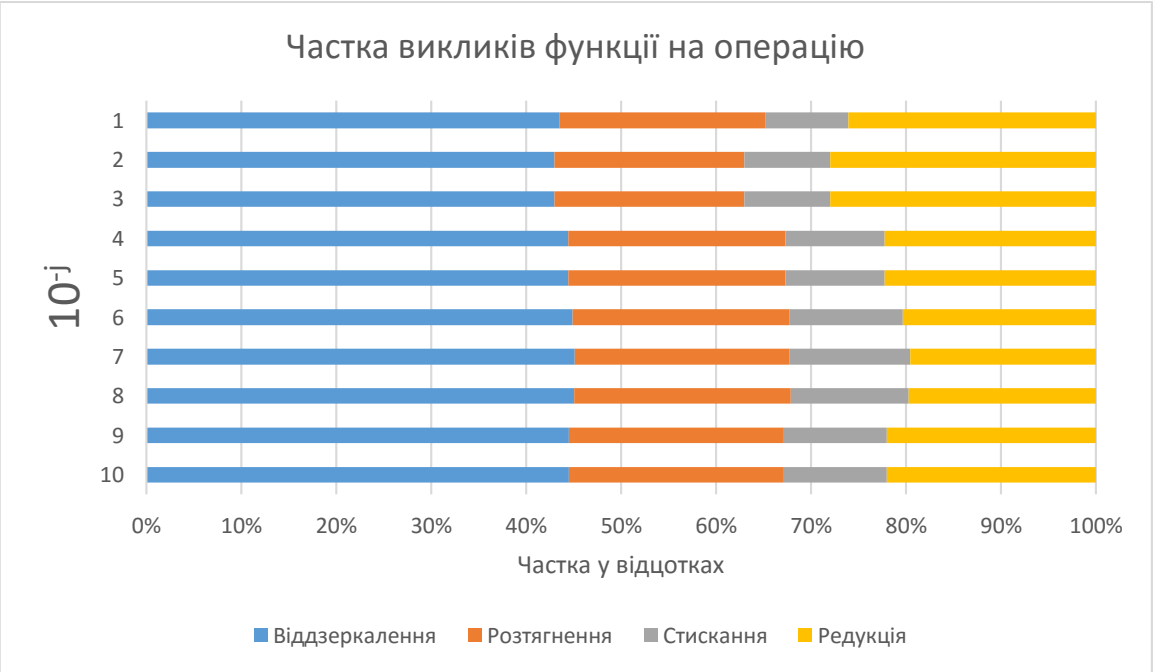
Точність ε	Віддзеркалення	Розтягнення	Стискання	Редукція	Σ
10^{-1}	10	5	2	3	20
10^{-2}	43	20	9	14	86
10^{-3}	43	20	9	14	86
10^{-4}	72	37	17	18	144
10^{-5}	72	37	17	18	144
10^{-6}	128	65	34	29	256
10^{-7}	134	67	38	29	268

10^{-8}	160	81	44	35	320
10^{-9}	218	111	53	54	436
10^{-10}	218	111	53	54	436

Таблиця 2.2.2



Діаграма 2.2.1



Діаграма 2.2.2

Варито звернути увагу на збільшення кількості виконання «стискання» в порівнянні з безумовною оптимізацією при аналогічних параметрах.

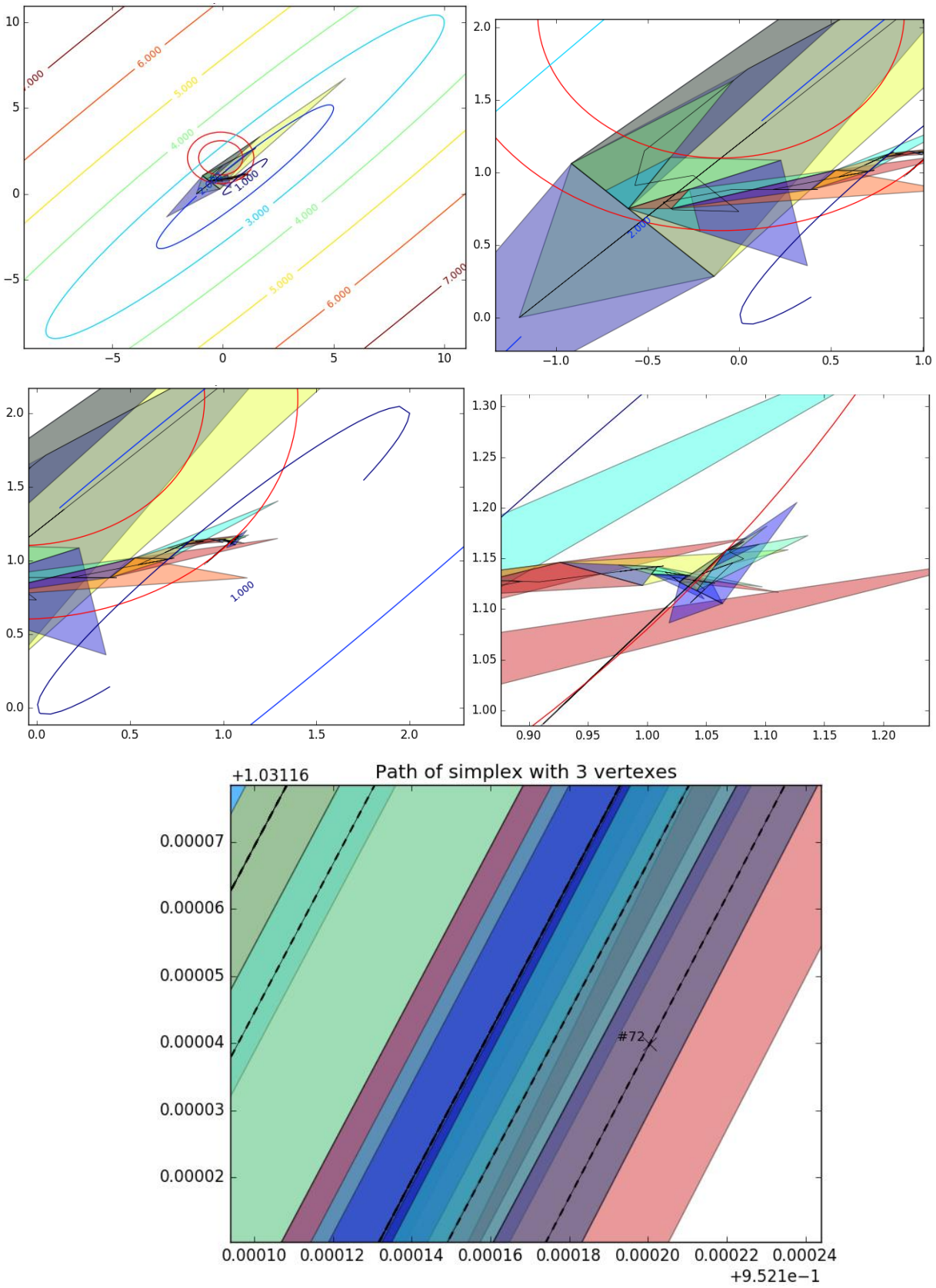


Рис 2.2.2 Графічна історія переміщення полігону при $\varepsilon = 10^{-5}$

На наведених графіках (Рис 2.2.2) можна побачити як багатокутник б’ється об бар’єр і зменшується в розмірах.

Розглянувши результати можна побачити зміни у розподілені викликів обчислення функції, що пов’язано з бар’єром який виник завдяки встановленю умови.

При $\varepsilon = 10^{-10}$, ми отримаємо найкрашій результат, для точності, але 712 ітерації забагато, починаючи з $\varepsilon = 10^{-6}$, метод підходить ближче але це все одно майже в 2 рази більше обрахувань ніж при $\varepsilon = 10^{-5}$, тому варто залишити $\varepsilon = 10^{-5}$.

Перевіримо вплив зміни довжини ребра на кількість обчислень функції.

З отриманих результатів (див додаток Б), допустимий діапазон ребра опинився від 1.4 до 1.

Занесемо результати в таблицю.

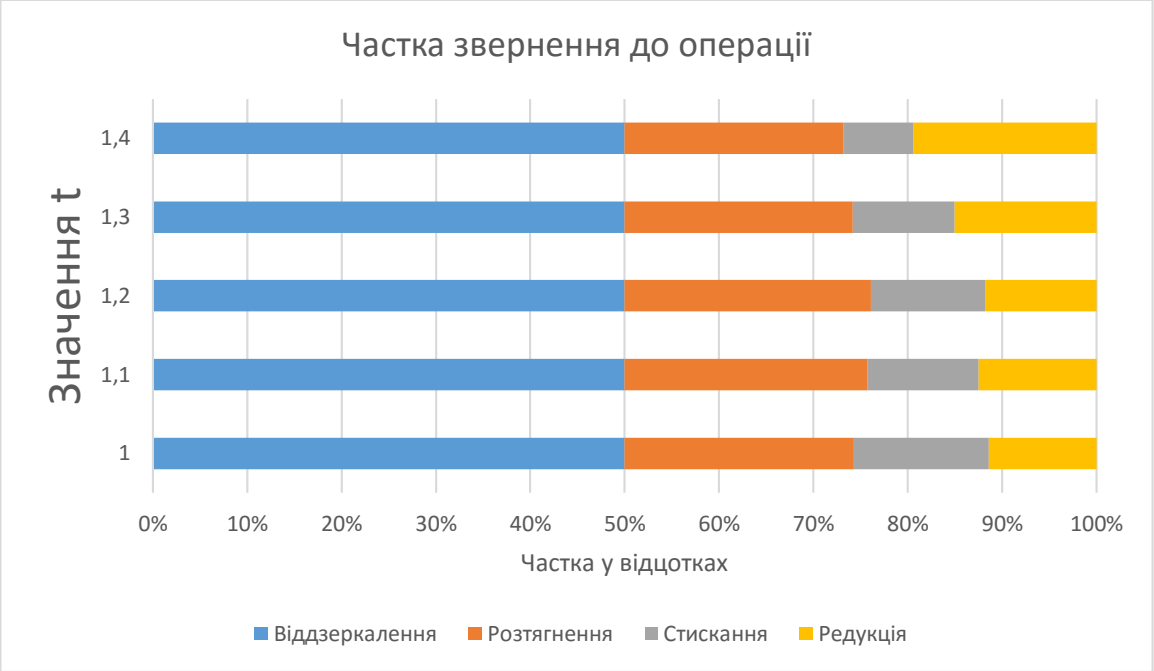
Значення t	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
1,4	231	0.7448670180708767, 0.8605649618516186	0.667863351249428
1,3	202	0.6034170735024573, 0.7751774690537185	0.8200782452405484
1,2	525	0.7234857922331697, 0.8462572545950858	0.6903932729558097
1,1	238	0.9529737480669501, 1.0319558470108434	0.5041348026849249
1	343	1.041589155266959, 1.1269768376038232	0.5226888415974384

Таблиця 2.1.11

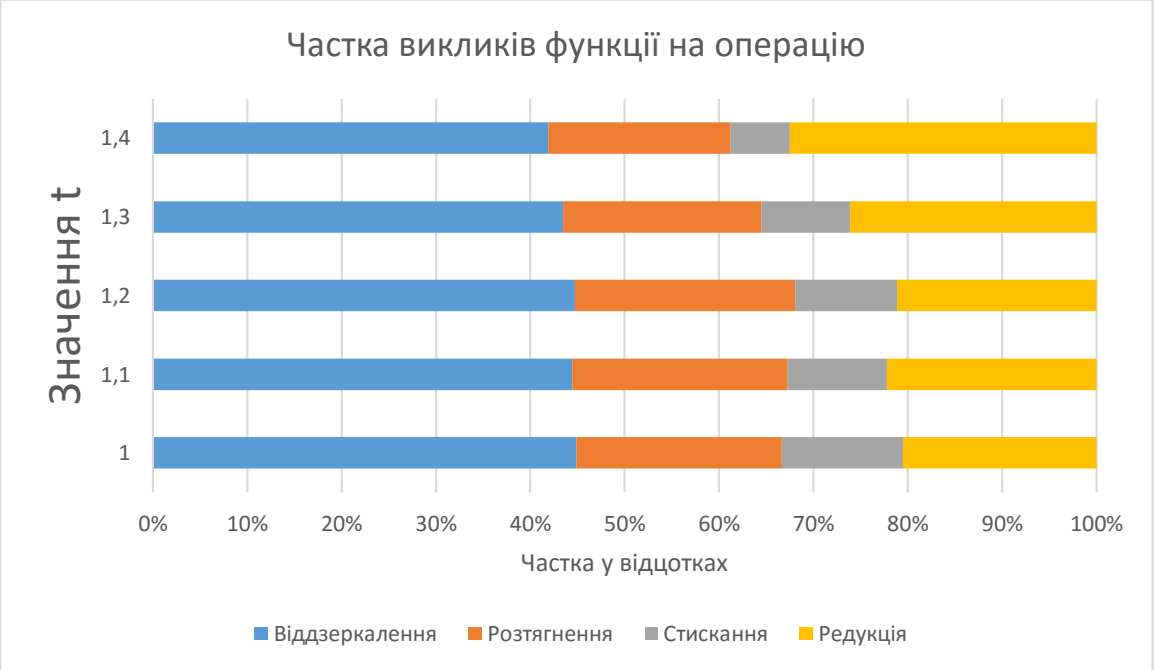
Для того щоб краще зрозуміти причину подібного розподілу кількостей викликів функції погляньмо на статистику виклику кожної операцій:

Значення t	Віддзеркалення	Розтягнення	Стискання	Редукція	Σ
1,4	67	31	10	26	134
1,3	60	29	13	18	120
1,2	161	84	39	38	322
1,1	72	37	17	18	144
1	105	51	30	24	210

Таблиця 2.2.x



Діаграма 2.2.x



Діаграма 2.2.x

Розглянемо отримані результати. Найкращий результат досягнутий при $t = 1.1$. Можливо це пов’язано з вдалими габаритами багатокутника.

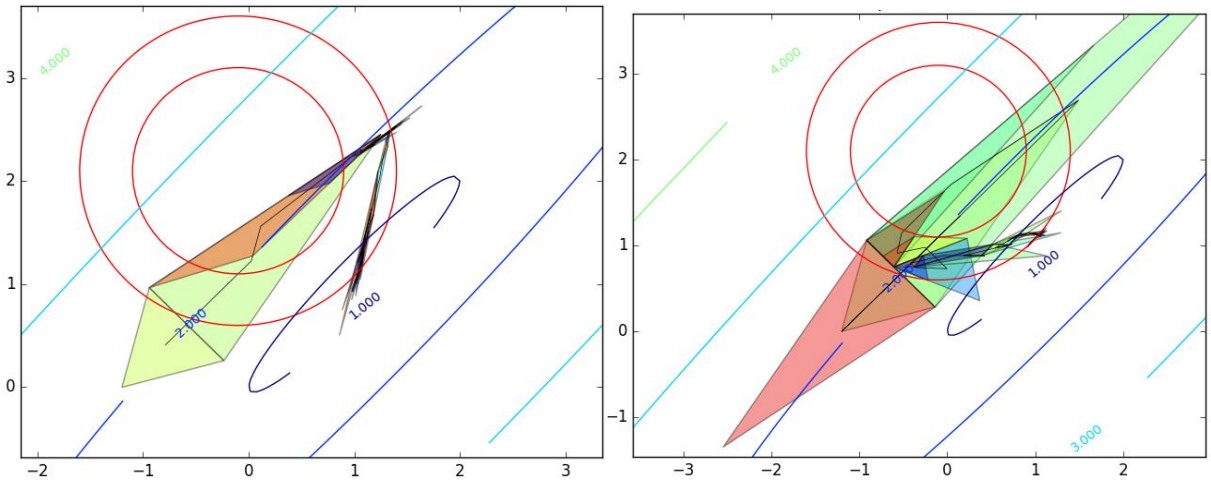


Рис 2.2.x Графічне зображення руху полігона при $t = 1.0$, і $t = 1.1$

2.2.3 Лінійне обмеження

Модифікуємо метод Нелдера-Міда додавши можливість кроку по границі.

Для цього приєднаємо наступні методи:

метод Свена для пошуку інтервалу з мінімумом на прямій,

метод Золотого січення для уточнення інтервалу з мінімумом (в якості мінімуму береться точка з найменшим значенням)

метод ДСК Пауела – знаходить точку мінімуму з певною похибкою, квадратичними апроксимаціями.

Встановимо наступні умови:

$$ax_1 + bx_2 \leq c$$

Правила прості:

Після закінчення крокування методом Нелдера-Міда, естафета передається одномірному пошуку – методу Свена, для пошуку інтервалу.

Після того як Метод свена закінчить свою роботу, результати будуть передані методу заданому користувачем, за бажанням: метод Золотого січення, метод ДСК Пауела.

Встановимо на метод Свена $\Delta\lambda = 0.1$, для методу Золотого січення і $\Delta\lambda = \frac{\|x^k\|}{\|S\|}$, де x^k точка останьої ітерації, найменша за значенням точка в множині вершин багатокутника, ДСК Пауела.

Перелічені методи одномірного пошуку рухаються паралельно лінії границі.

Розглянемо роботу алгоритму з умовою

$$x1 + x2 \leq 0.5$$

Початкова точка $x^0 = (-1.2; 0.0)$

Для початку перевіримо умову за допомогою WolframAlpha computational knowledge engine[5]:

$$\min\left\{(10\,(x1 - x2)^2 + (x1 - 1)^2)^{0.25} \mid x1 + x2 \leq 0.5\right\} \approx 0.860696 \text{ at } (x1, x2) \approx (0.268293, 0.231707)$$

Порівняємо графіки з глобальним мінімумом та мінімумом в заданій області:

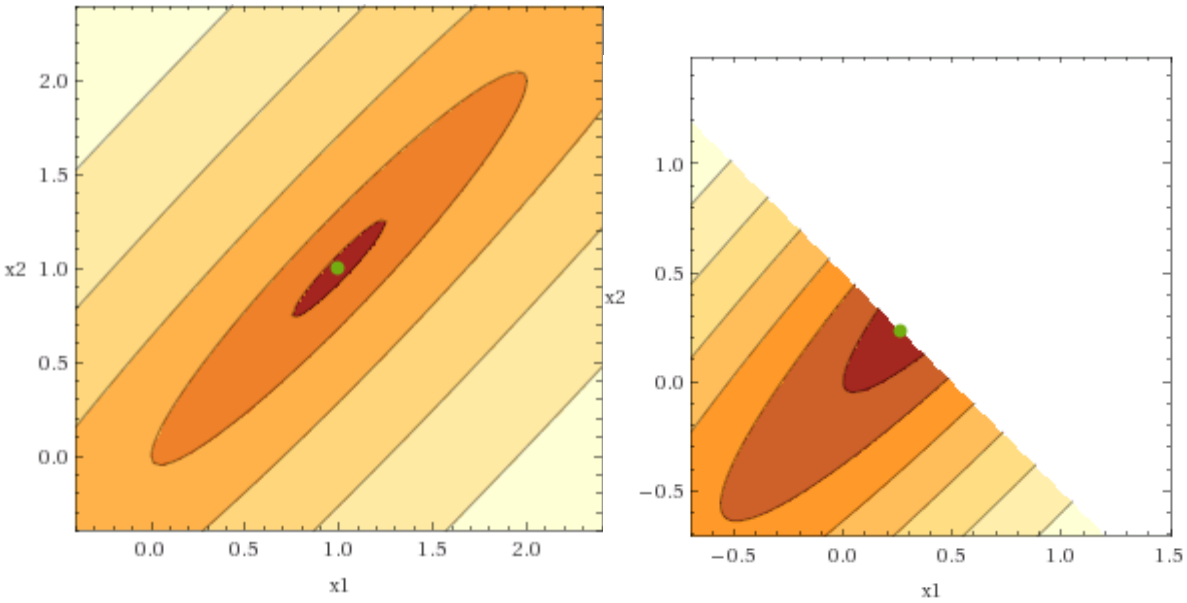


Рис 2.2.1 Порівння на графіках глобального мінімуму та на заданій області

Метод Золотого січення

Встановимо $\Delta\lambda = 0.1$;

Використовуючи здобуті на етапі безумовної оптимізації параметри, перевіримо роботу алгоритму для заданих умов, і занесем дані у таблицю.

Точність ε	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
10^{-1}	24	0.0809402943276, 0.0662790642032	0.959285345159033
10^{-2}	50	0.2107530041407339, 0.17443918188088542	0.8930606750451637
10^{-3}	149	0.26827034683947604, 0.231364695047741	0.8608008527421275
10^{-4}	154	0.26812371935230456, 0.23151132253491247	0.860800464524323

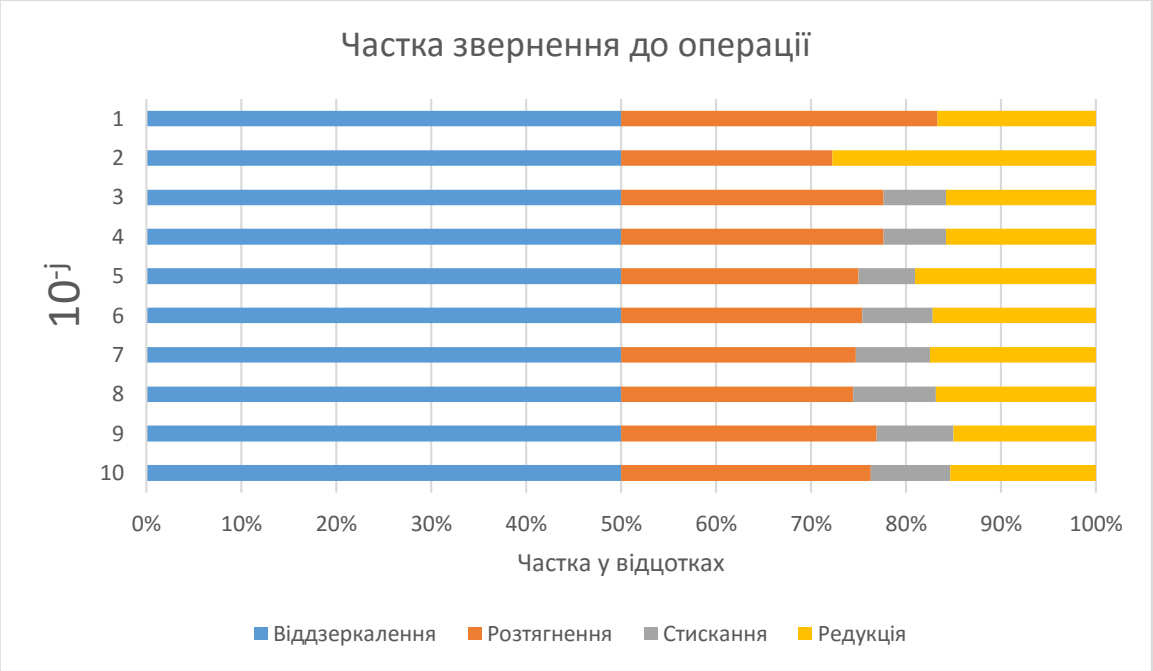
10^{-5}	175	0.26804781526020877, 0.23145162792261292	0.8608393606777439
10^{-6}	242	0.2682913494362415, 0.23170579722595994	0.8606965822230171
10^{-7}	320	0.26829268907688175, 0.23170728754115363	0.8606957703130241
10^{-8}	334	0.26829267197891643, 0.23170730463911893	0.8606957703130191
10^{-9}	481	0.26829134328284665, 0.23170591178609967	0.8606965511212493
10^{-10}	510	0.2682913658319216, 0.231705929900549	0.8606965394549524

Таблиця 2.2.1

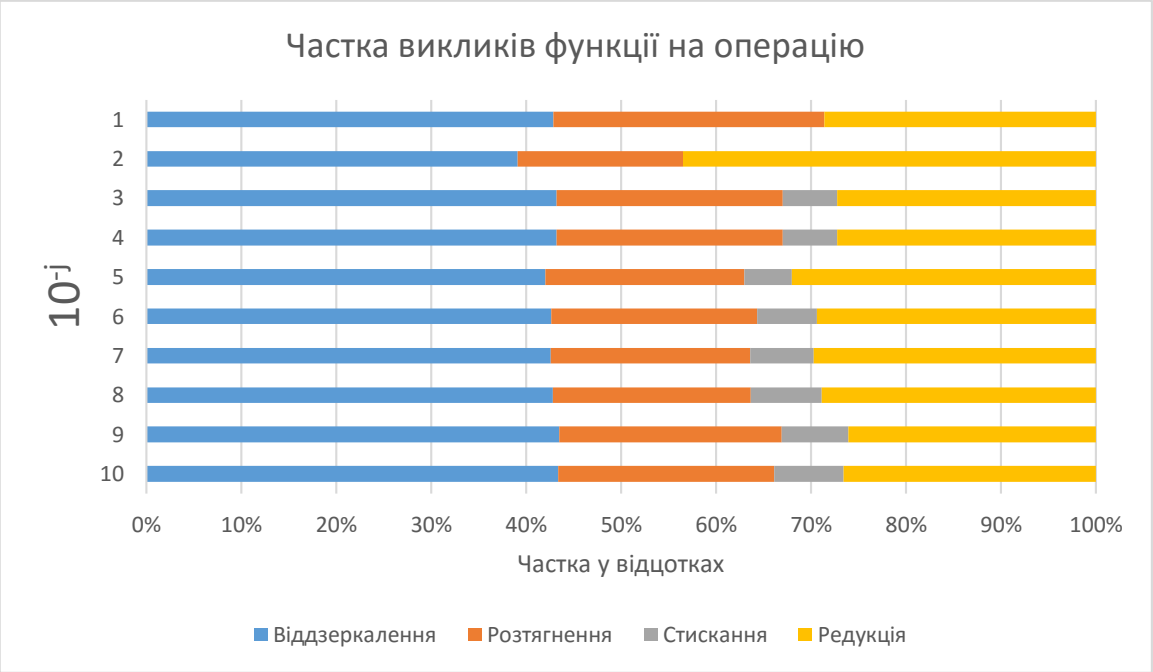
Для того щоб краще зрозуміти причину подібного розподілу кількостей викликів функції погляньмо на статистику виклику кожної операцій:

Точність ε	Віддзеркалення	Розтягнення	Стискання	Редукція	Σ
10^{-1}	3	2	0	1	6
10^{-2}	9	4	0	5	18
10^{-3}	38	21	5	12	76
10^{-4}	38	21	5	12	76
10^{-5}	42	21	5	16	84
10^{-6}	61	31	9	21	122
10^{-7}	83	41	13	29	166
10^{-8}	86	42	15	29	172
10^{-9}	130	70	21	39	260
10^{-10}	137	72	23	42	274

Таблиця 2.2.2



Діаграма 2.2.1



Діаграма 2.2.2

Варито звернути увагу на збільшення кількості виконання «стискання» в порівнянні з безумовною оптимізацією при аналогічних параметрах.

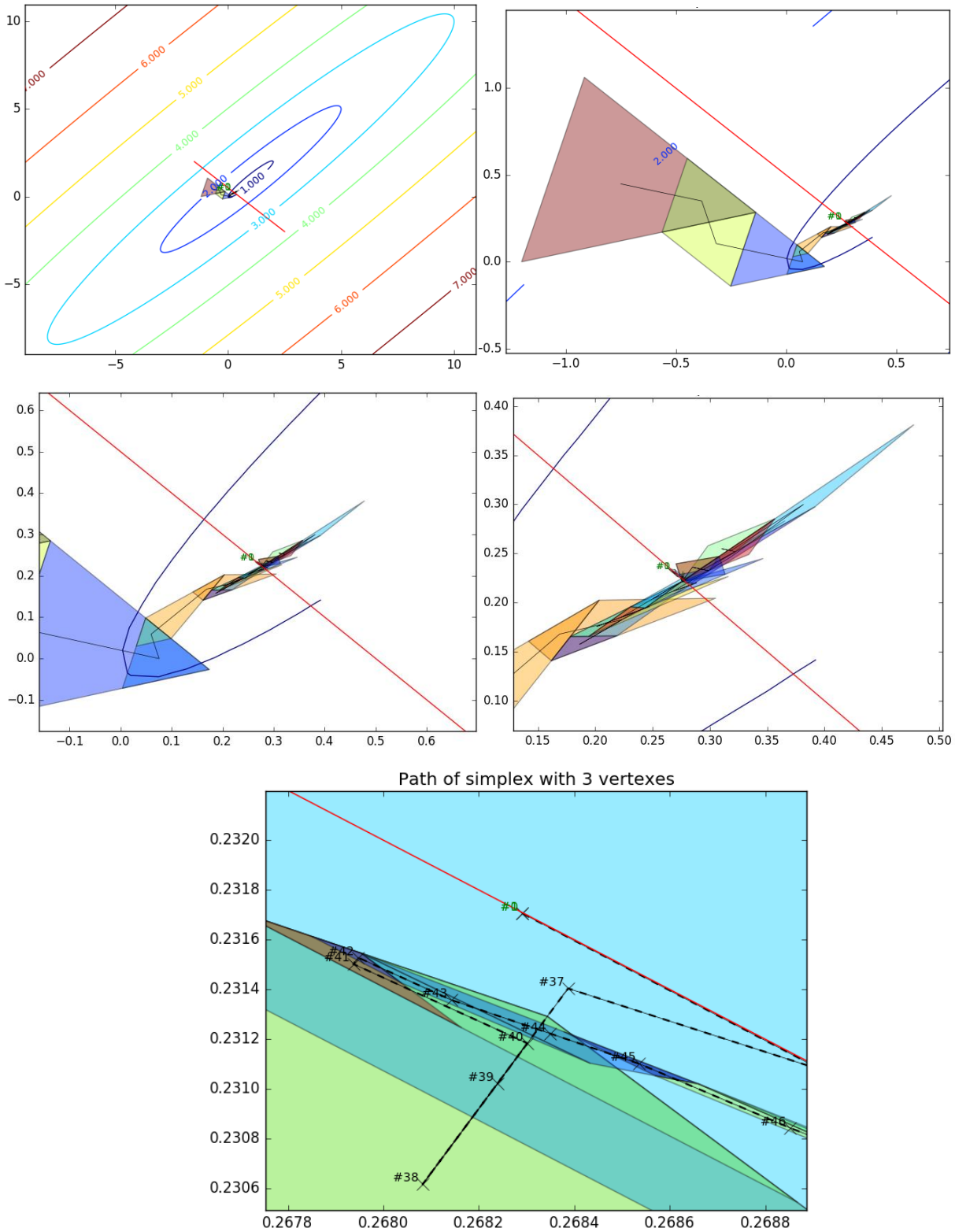


Рис 2.2.2 Графічна історія переміщення полігону при $\varepsilon = 10^{-6}$

На наведених графіках (Рис 2.2.2) можна побачити як багатокутник б’ється об бар’єр і зменшується в розмірах.

Розглянувши результати можна побачити зміни у розподілені викликів обчислення функції, що пов’язано з бар’єром який виник завдяки встановленю умови.

При $\varepsilon = 10^{-10}$, ми отримаємо найкрашій результат, для точності, але 510 ітерації забагато, варто обрати залишити $\varepsilon = 10^{-6}$.

Перевіримо вплив зміни довжини ребра на кількість обчислень функції.

Таблицю побудуємо на отриманих результатах (див додаток Г).

Занесемо результати в таблицю.

Значення t	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
2	287	0.2540882767220329, 0.2167929088598432	0.8690097553523572
1,9	286	0.2682773186422904, 0.2316913896401079	0.8607047411076992
1,8	263	-0.241659775643, -0.303742508261	1.1211978351727365
1,7	494	0.26829005905459946, 0.2317043397167183	0.8606973705880546
1,6	383	0.2682910125992788, 0.23170584747319165	0.8606966644453784
1,5	616	0.2682925187675763, 0.23170689571648687	0.8606959315886827
1,4	261	0.26829210111957624, 0.23170697457974263	0.8606960287855615
1,3	189	0.2682921423488777, 0.23170663147568002	0.8606961153927508
1,2	296	0.26829225997141004, 0.23170712842464872	0.8606959390733054
1,1	242	0.2682913494362415, 0.23170579722595994	0.8606965822230171
1	215	0.2680400731084564, 0.2314418171518793	0.860844395724178

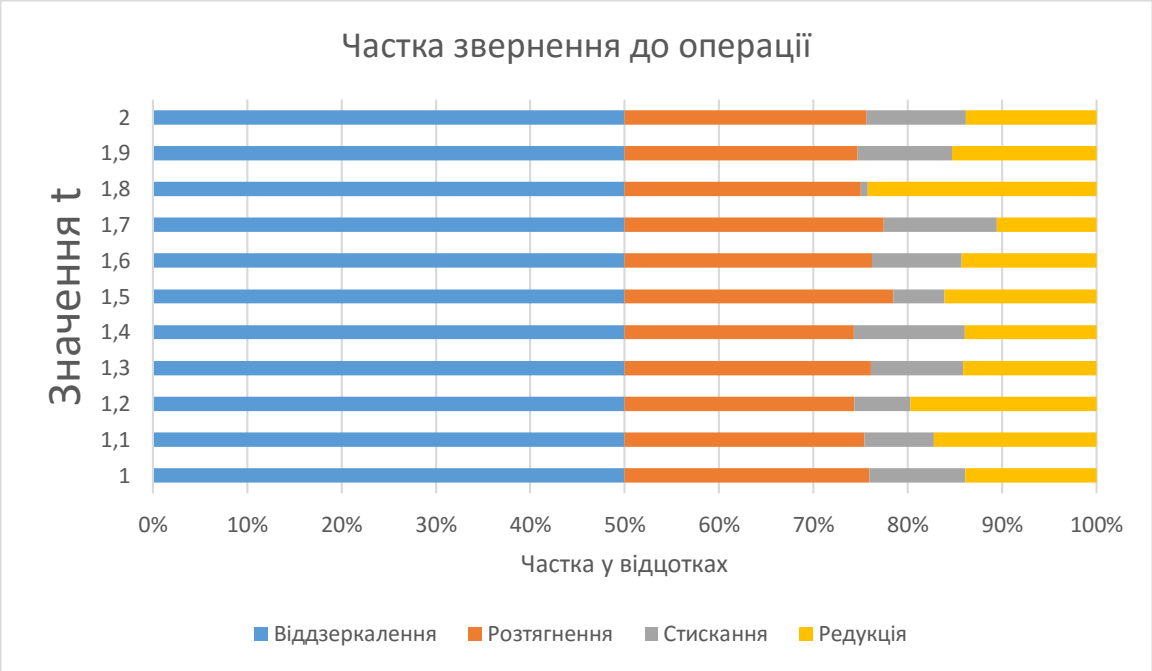
Таблиця 2.1.11

Для того щоб краще зрозуміти причину подібного розподілу кількостей викликів функції погляньмо на статистику виклику кожної операцій:

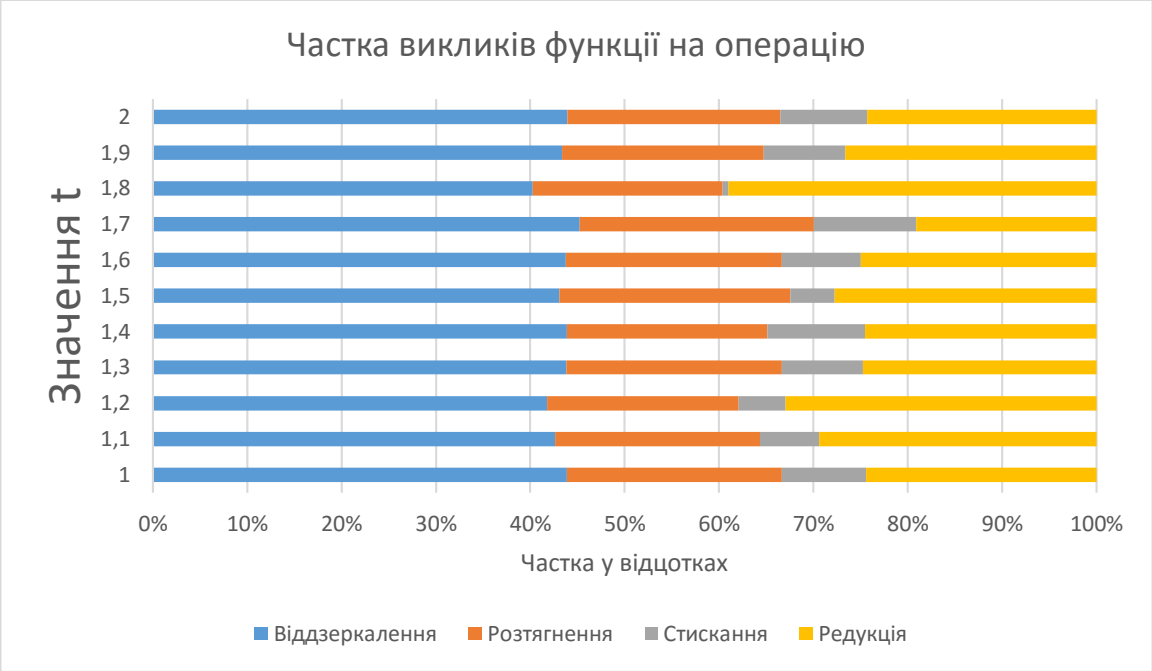
Значення t	Віддзеркалення	Розтягнення	Стискання	Редукція	Σ
2	76	39	16	21	152
1,9	75	37	15	23	150
1,8	64	32	1	31	128

1,7	142	78	34	30	284
1,6	105	55	20	30	210
1,5	174	99	19	56	348
1,4	68	33	16	19	136
1,3	46	24	9	13	92
1,2	76	37	9	30	152
1,1	61	31	9	21	122
1	54	28	11	15	108

Таблиця 2.2.х



Діаграма 2.2.х



Діаграма 2.2.x

Розглянемо отримані результати. Найкращий результат досягнутий при $t = 1.1$. Можливо це пов'язано з вдалими габаритами багатокутника, $t = 1.2$має результат трохи краще але й ітерацій більше

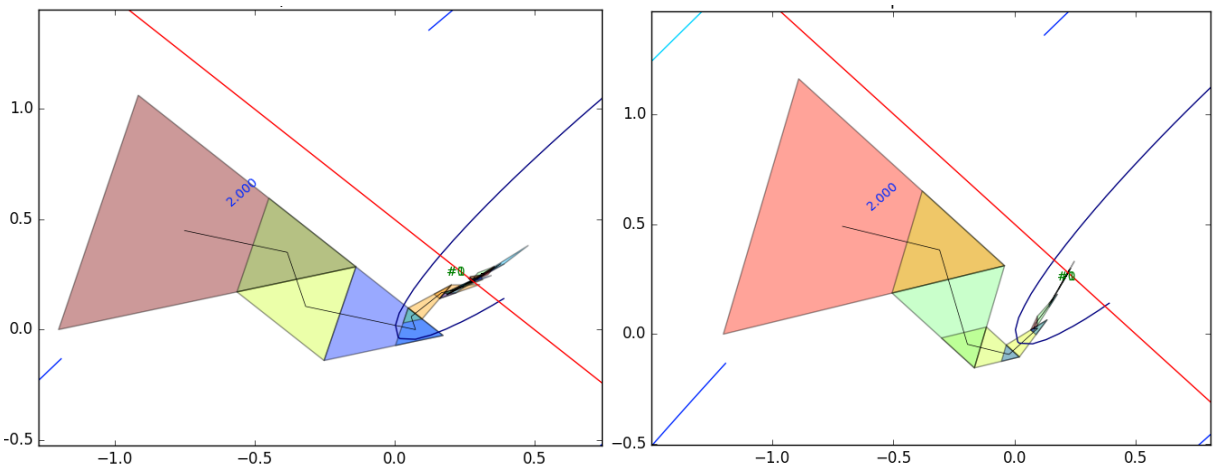


Рис 2.2.x Графічне зображення руху полігона при $t = 1.1$, і $t = 1.2$

Метод Золотого січення $\Delta\lambda$ модифікована

Етап методу Нелдера-Міда ідентичний минулому, тому в цьому розділі будуть лиш результати у вигляді таблиць.

Встановимо $\Delta\lambda = \frac{\|x^k\|}{\|S\|}$, $\|S\| = \sqrt{a^2 + b^2}$, де a і b коефіцієнти при зміних лінійного рівняння, оскільки нам потрібен вектор напрямок, і його норма.

Занесемо результати роботи програми у таблицю мод

Занесемо отримані результати в таблицю, у форматі таблиці 2.1 з наступними стовпчиками: параметр ε, кількість обчислень функції, координати точки, значення функції в цій точці, де точка береться з найменшим значенням серед точок багатокутника на останньому кроці.

Точність ε	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
10^{-1}	25	0.0856867811713, 0.0615325773594	0.9578612551169329
10^{-2}	52	0.211284906457607 8, 0.1739072795640115	0.8930412334454183
10^{-3}	151	0.2679329988603948, 0.2317020430268223	0.860800993494702
10^{-4}	156	0.268126887025008, 0.23150815486220913	0.8608004656084486
10^{-5}	177	0.26804936477118924, 0.23145007841163245	0.8608393606817846
10^{-6}	243	0.26829115126462705, 0.23170599539757436	0.8606965822232765
10^{-7}	322	0.268292690140886 1, 0.23170728647714928	0.8606957703130248
10^{-8}	336	0.268292671569351 9, 0.23170730504868348	0.8606957703130191
10^{-9}	483	0.26829134299263524, 0.23170591207631108	0.8606965511212493
10^{-10}	512	0.26829136581178836, 0.2317059299206823	0.8606965394549523

Таблиця

Теж саме тільки зі стовпцем значень λ замість x

Точність ε	Кількість обчислень $f(x)$	Значення λ	Значення $f(x)$
10^{-1}	25	0.12419955246269115	0.9578612551169329

10^{-2}	52	0.1943917413880448	0.8930412334454183
10^{-3}	151	0.2505035161836073	0.860800993494702
10^{-4}	156	0.2505035161836073	0.8608004656084486
10^{-5}	177	0.2504123202787849	0.8608393606817846
10^{-6}	243	0.2508004902803199	0.8606965822232765
10^{-7}	322	0.2508017021814091	0.8606957703130248
10^{-8}	336	0.2508017021814091	0.8606957703130191
10^{-9}	483	0.2506660365096206	0.8606965511212493
10^{-10}	512	0.25066797020301956	0.8606965394549523

Таблиця

Занесем також набір результатів при різних t.

Значення t	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
2	288	0.25408850390351906, 0.21679268167835702	0.8690097553524078
1,9	287	0.2682775664695181, 0.23169114181288009	0.8607047411078896
1,8	264	-0.24165975031, -0.303742533594	1.1211978351726952
1,7	495	0.2682900692551788, 0.23170432951613892	0.8606973705880918
1,6	384	0.26829131545524276, 0.23170554461722767	0.8606966644455029
1,5	617	0.26829226266614536, 0.2317071518179178	0.860695931588737
1,4	262	0.26829242294765693, 0.23170665275166197	0.8606960287858718
1,3	190	0.2682922402380465, 0.2317065335865114	0.8606961153930859
1,2	298	0.26829250233879537, 0.23170688605726342	0.8606959390732787
1,1	243	0.26829115126462705, 0.23170599539757436	0.8606965822232765

1	216	0.26803985812368125, 0.2314420321366544	0.8608443957240458

Таблиця

Теж саме тільки зі стовпцем значень λ замість x

Значення t	Кількість обчислень $f(x)$	Значення λ	Значення $f(x)$
2	288	0.2540882767220329, 0.2167929088598432	0.8690097553524078
1,9	287	0.2682773186422904, 0.2316913896401079	0.8607047411078896
1,8	264	-0.241659775643, -0.303742508261	1.1211978351726952
1,7	495	0.26829005905459946, 0.2317043397167183	0.8606973705880918
1,6	384	0.2682910125992788, 0.23170584747319165	0.8606966644455029
1,5	617	0.2682925187675763, 0.23170689571648687	0.860695931588737
1,4	262	0.26829210111957624, 0.23170697457974263	0.8606960287858718
1,3	190	0.2682921423488777, 0.23170663147568002	0.8606961153930859
1,2	298	0.26829225997141004, 0.23170712842464872	0.8606959390732787
1,1	243	0.2682913494362415, 0.23170579722595994	0.8606965822232765
1	216	0.2680400731084564, 0.2314418171518793	0.8608443957240458

Метод Метод ДСК Пауела $\Delta\lambda$ модифікована

Етап методу Нелдера-Міда ідентичний минулому, тому в цьому розділі будуть лиш результати у вигляді таблиць.

Встановимо $\Delta\lambda = \frac{\|x^k\|}{\|S\|}$, $\|S\| = \sqrt{a^2 + b^2}$, де a і b коефіцієнти при зміних лінійного рівняння, оскільки нам потрібен вектор напрямок, і його норма.

Занесемо результати роботи програми у таблицю мод

Занесемо отримані результати в таблицю, у форматі таблиці 2.1 з наступними стовпчиками: параметр ε, кількість обчислень функції, координати точки, значення функції в цій точці, де точка береться з найменшим значенням серед точок багатокутника на останньому кроці.

Точність ε	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
10^{-1}	20	0.0582670800581, 0.0889522784727	0.9729948182394906
10^{-2}	42	0.20927999001535402, 0.17591219600626531	0.8931569974802394
10^{-3}	136	0.2679061872856234, 0.2317288546015937	0.8608011615837243
10^{-4}	136	0.2679061872856234, 0.2317288546015937	0.8608011615837243
10^{-5}	151	0.2679932536045411, 0.23150618957828065	0.8608394097274461
10^{-6}	1213	0.2682900222064234, 0.23170712445577799	0.8606965822488442
10^{-7}	1287	0.2682914081456554, 0.23170856847237994	0.8606957703386782
10^{-8}	1296	0.2682914081456554, 0.23170856847237994	0.8606957703386782
10^{-9}	440	0.2682902447610247, 0.23170701030792165	0.860696551140672
10^{-10}	1462	0.26829136375745466, 0.23170593197501596	0.8606965394549523

Таблиця

Теж саме тільки зі стовпцем значень λ замість x

Точність ε	Кількість обчислень $f(x)$	Значення λ	Значення $f(x)$

10^{-1}	20	0.12419955246269115	0.9729948182394906
10^{-2}	42	0.1943917413880448	0.8931569974802394
10^{-3}	136	0.2505035161836073	0.8608011615837243
10^{-4}	136	0.2505035161836073	0.8608011615837243
10^{-5}	151	0.2504123202787849	0.8608394097274461
10^{-6}	1213	0.2508004902803199	0.8606965822488442
10^{-7}	1287	0.2508017021814091	0.8606957703386782
10^{-8}	1296	0.2508017021814091	0.8606957703386782
10^{-9}	440	0.2506660365096206	0.860696551140672
10^{-10}	1462	0.25066797020301956	0.8606965394549523

Таблиця

Для подальшого розрахунку використаємо $\varepsilon = 10^{-5}$

Занесем також набір результатів при різних t.

Значення t	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
2	1251	0.25920060223305064, 0.21176592303507827	0.8693868361306952
1,9	217	0.26782932640460144, 0.23193412110605705	0.8607655735005176
1,8	1157	-0.3238887788983468, -0.5077060781981652	1.2024475787832054
1,7	1343	0.26997590035018987, 0.2247281421680446	0.8625047821919184
1,6	346	0.26746410290357747, 0.23252909175802755	0.8607086643166304
1,5	1506	0.267609828848687, 0.23220142992233797	0.8607555212996978
1,4	233	0.2675380951533741, 0.23246098054594477	0.8607051713475186
1,3	92	0.26806046421801577, 0.23165204425070587	0.8607783765322651
1,2	1267	0.2682914989898282, 0.23170788940623047	0.8606959390856637

1,1	151	0.2679932536045411, 0.23150618957828065	0.8608394097274461
1	187	0.26728146839126005, 0.23220042186907563	0.8608536390647341

Таблиця

Теж саме тільки зі стовпцем значень λ замість x

Значення t	Кількість обчислень $f(x)$	Значення λ	Значення $f(x)$
2	1251	0.23667999664013126	0.8693868361306952
1,9	217	0.250578872270691	0.8607655735005176
1,8	1157	0.45644919504907805	1.2024475787832054
1,7	1343	0.24782571299091277	0.8625047821919184
1,6	346	0.2507335729992535	0.8607086643166304
1,5	1506	0.2499578966875539	0.8607555212996978
1,4	233	0.250730353949826	0.8607051713475186
1,3	92	0.2505138426676632	0.8607783765322651
1,2	1267	0.2517459754772618	0.8606959390856637
1,1	151	0.2504123202787849	0.8608394097274461
1	187	0.25047352077454377	0.8608536390647341

Можна зробити висновок про деяку нестабільність ДСК Пауела.

ВИСНОВКИ

В даній роботі було розглянуто метод Нелдера-Міда або метод деформованого многогранника. Проведено роботу з різними значеннями коефіцієнтів. Прямої залежності кількості обрахувань цільової функції та параметрів немає, корегування проходить експерементально.

Для кореневої функції доволі доцільним вікористовувати наступні парраметри

$$\alpha = 1, \beta = 0.3, \gamma = 3, \sigma = 0.4, t = 2, \varepsilon = 10^{-5};$$

Але якщо знанаходитись біля області мінімуму краще встановити $t = 1.1$

ДОДАТОК А БЕЗУМОВНА ОПТИМІЗАЦІЯ

Даний додаток містить інформацію про останні ітерації для певної модифікації параметра коефіцієнта в наступному форматі:

```
----------------------------------------
Result: візуальне розділення
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5 значення відповідних коефіцієнтів, з якими був проведений ряд ітерацій
Accuracy: 5 точність перевірки критерію зупинки, відємеа степінь 1: 10^-5
Side length: 1.5 довжина сторони багатокутника
itterations: 57 - кількість ітерацій, співпадає з кількістю обчислень функції на етапі перевірки критерію зупинки, а саме з кількістю обчислень функції в центрі ваги багатокутника (усі інші задіяні значення функції вже відомі на момент підрахування критерію)
x: [[1.0032325114580896, 0.8856881978791556], [1.0032868582498309, 0.8857400749332238],
[1.003114106854531, 0.8855668261177909]] x – координати вершин багатокутника на останній ітерації
f(x): [0.6096899312083286, 0.6096967266458907, 0.6096967968329776] значення функції в відповідних точках, порядок співпадає з минулим пунктом
action: reduction остання операція яка була проведена
----------------------------------------
Count of function calling: 203 загальна кількість викликів функції
Кількість звернень до операції, кількість обчислень функції цим типом операції
Count of reflection's: 57 ; function calling: 57 для віддзеркалення
Count of expansion's: 28 ; function calling: 28 для розтягнення
Count of compression's: 1 ; function calling: 1 для стискання
Count of reduction's: 28 ; function calling: 56 для редукції
Sum of operations calling: 114 загальна кількість викликів операцій
Sum of functions calling by operations calling: 142 загальна кількість обчислень функцій, здійснена операціями

----------------------------------------
візуальне розділення
```

Результати виконання програми для різних значень параметра ε і стандартних $\alpha, \beta, \gamma, \sigma, t$

```
----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 10
Side length: 1.5
Itterations: 601
x: [[1.0954269286317593, 0.9881361579791765], [1.0971620527025725, 0.9900038377448878],
[1.095659626787088, 0.9884535697384098]]
f(x): [0.593673072683594, 0.5937326863138356, 0.5935090263220651]
action: expansion
----------------------------------------
Count of function calling: 2005
Count of reflection's: 601 ; function calling: 601
Count of expansion's: 394 ; function calling: 394
Count of compression's: 9 ; function calling: 9
Count of reduction's: 198 ; function calling: 396
Sum of operations calling: 1202
Sum of functions calling by operations calling: 1400

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 9
Side length: 1.5
Itterations: 86
x: [[1.0237568206219514, 0.9066305988455357], [1.023444921761996, 0.9063123932454936],
[1.023939975933578, 0.9068175030986154]]
f(x): [0.6092181406932987, 0.6092181981001148, 0.6092180899161096]
action: reflection
----------------------------------------
```

```
Count of function calling: 301
Count of reflection's: 86 ; function calling: 86
Count of expansion's: 46 ; function calling: 46
Count of compression's: 1 ; function calling: 1
Count of reduction's: 39 ; function calling: 78
Sum of operations calling: 172
Sum of functions calling by operations calling: 211

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 8
Side length: 1.5
Itterations: 86
x: [[1.0237568206219514, 0.9066305988455357], [1.023444921761996, 0.9063123932454936],
[1.023939975933578, 0.9068175030986154]]
f(x): [0.6092181406932987, 0.6092181981001148, 0.6092180899161096]
action: reflection
----------------------------------------
Count of function calling: 301
Count of reflection's: 86 ; function calling: 86
Count of expansion's: 46 ; function calling: 46
Count of compression's: 1 ; function calling: 1
Count of reduction's: 39 ; function calling: 78
Sum of operations calling: 172
Sum of functions calling by operations calling: 211

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 7
Side length: 1.5
Itterations: 85
x: [[1.023444921761996, 0.9063123932454936], [1.0232617664503694, 0.906125488992414],
[1.0237568206219514, 0.9066305988455357]]
f(x): [0.6092181981001148, 0.6092184502150311, 0.6092181406932987]
action: expansion
----------------------------------------
Count of function calling: 298
Count of reflection's: 85 ; function calling: 85
Count of expansion's: 45 ; function calling: 45
Count of compression's: 1 ; function calling: 1
Count of reduction's: 39 ; function calling: 78
Sum of operations calling: 170
Sum of functions calling by operations calling: 209

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 6
Side length: 1.5
Itterations: 74
x: [[1.0219280141209233, 0.9047646864688641], [1.021805896777117, 0.9046399334801265],
[1.0217246444571626, 0.9045571596971194]]
f(x): [0.6092218849382628, 0.6092228086084055, 0.6092228399413834]
action: reduction
----------------------------------------
Count of function calling: 261
Count of reflection's: 74 ; function calling: 74
Count of expansion's: 38 ; function calling: 38
Count of compression's: 1 ; function calling: 1
Count of reduction's: 35 ; function calling: 70
Sum of operations calling: 148
Sum of functions calling by operations calling: 183

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 57
```

```
x: [[1.0032325114580896, 0.8856881978791556], [1.0032868582498309, 0.8857400749332238],
[1.003114106854531, 0.8855668261177909]]
f(x): [0.6096899312083286, 0.6096967266458907, 0.6096967968329776]
action: reduction
---------------------------------------
Count of function calling: 203
Count of reflection's: 57 ; function calling: 57
Count of expansion's: 28 ; function calling: 28
Count of compression's: 1 ; function calling: 1
Count of reduction's: 28 ; function calling: 56
Sum of operations calling: 114
Sum of functions calling by operations calling: 142

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 4
Side length: 1.5
Itterations: 53
x: [[1.0040710547030762, 0.8865554178619368], [1.0049406033709358, 0.8873854507270282],
[1.002176581046138, 0.8846134696801017]]
f(x): [0.6096223188005307, 0.6097334359469495, 0.6097323775885585]
action: reduction
---------------------------------------
Count of function calling: 187
Count of reflection's: 53 ; function calling: 53
Count of expansion's: 28 ; function calling: 28
Count of compression's: 1 ; function calling: 1
Count of reduction's: 24 ; function calling: 48
Sum of operations calling: 106
Sum of functions calling by operations calling: 130

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 3
Side length: 1.5
Itterations: 33
x: [[0.8093630537473296, 0.690933832536609], [0.8106530506891783, 0.6916879882759248],
[0.8078380056324361, 0.6893219928653749]]
f(x): [0.6482551622590498, 0.6489715059332002, 0.6489784179064805]
action: reduction
---------------------------------------
Count of function calling: 120
Count of reflection's: 33 ; function calling: 33
Count of expansion's: 15 ; function calling: 15
Count of compression's: 1 ; function calling: 1
Count of reduction's: 17 ; function calling: 34
Sum of operations calling: 66
Sum of functions calling by operations calling: 83

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 2
Side length: 1.5
Itterations: 9
x: [[0.2003601684768797, 0.20711747522458082], [0.17772128192322967, 0.20105140385499048],
[0.18378735329282, 0.22369029040864052]]
f(x): [0.8943854267462801, 0.9086152960162535, 0.9087953072965621]
action: reduction
---------------------------------------
Count of function calling: 37
Count of reflection's: 9 ; function calling: 9
Count of expansion's: 2 ; function calling: 2
Count of compression's: 1 ; function calling: 1
Count of reduction's: 6 ; function calling: 12
Sum of operations calling: 18
Sum of functions calling by operations calling: 24

---------------------------------------
```

```
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 1
Side length: 1.5
Itterations: 2
x: [[0.24888873943360235, 0.38822856765378105], [-0.11333344542479823, 0.2911714257403358], [-
0.016276303511352952, 0.6533936105987364]]
f(x): [0.9331763005920716, 1.3022306278245854, 1.5326166113437516]
action: reduction
---------------------------------------
Count of function calling: 12
Count of reflection's: 2 ; function calling: 2
Count of expansion's: 0 ; function calling: 0
Count of compression's: 0 ; function calling: 0
Count of reduction's: 2 ; function calling: 4
Sum of operations calling: 4
Sum of functions calling by operations calling: 6
```

Результати виконання програми для різних коефіцієнтів α

```
---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 57
x: [[1.0032325114580896, 0.8856881978791556], [1.0032868582498309, 0.8857400749332238],
[1.003114106854531, 0.8855668261177909]]
f(x): [0.6096899312083286, 0.6096967266458907, 0.6096967968329776]
action: reduction
---------------------------------------
Count of function calling: 203
Count of reflection's: 57 ; function calling: 57
Count of expansion's: 28 ; function calling: 28
Count of compression's: 1 ; function calling: 1
Count of reduction's: 28 ; function calling: 56
Sum of operations calling: 114
Sum of functions calling by operations calling: 142

---------------------------------------
Result:
Coefficients: alpha 0.9 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 43
x: [[0.41634760170900886, 0.3143987176871879], [0.41609186328738423, 0.31428205612155535],
[0.4163407720425606, 0.3143879335814286]]
f(x): [0.8165615277303465, 0.8165685111106679, 0.8165688906559151]
action: reduction
---------------------------------------
Count of function calling: 155
Count of reflection's: 43 ; function calling: 43
Count of expansion's: 19 ; function calling: 19
Count of compression's: 2 ; function calling: 2
Count of reduction's: 22 ; function calling: 44
Sum of operations calling: 86
Sum of functions calling by operations calling: 108

---------------------------------------
Result:
Coefficients: alpha 0.8 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 35
x: [[0.3652035057727145, 0.3692106449782674], [0.3651957476798695, 0.3692022354853194],
[0.36519510005286016, 0.3691951036709505]]
f(x): [0.7968205296868667, 0.7968253710635895, 0.7968255208263363]
action: reduction
---------------------------------------
Count of function calling: 131
Count of reflection's: 35 ; function calling: 35
Count of expansion's: 13 ; function calling: 13
Count of compression's: 0 ; function calling: 0
Count of reduction's: 22 ; function calling: 44
Sum of operations calling: 70
Sum of functions calling by operations calling: 92

---------------------------------------
Result:
Coefficients: alpha 0.7 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 19
x: [[0.23077291530094024, 0.24888648470328106], [0.23075782667952727, 0.24888531465712033],
[0.23075976257923758, 0.24890019806869318]]
f(x): [0.8782692103021248, 0.8782796379142005, 0.878280271673284]
action: reduction
---------------------------------------
```

```
Count of function calling: 78
Count of reflection's: 19 ; function calling: 19
Count of expansion's: 2 ; function calling: 2
Count of compression's: 0 ; function calling: 0
Count of reduction's: 17 ; function calling: 34
Sum of operations calling: 38
Sum of functions calling by operations calling: 55

----------------------------------------
Result:
Coefficients: alpha 0.6 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 18
x: [[0.21446511435670712, 0.2709347165884527], [0.2144390741963892, 0.27093020166660625],
[0.21444397204501042, 0.2709561392220041]]
f(x): [0.8975390384677683, 0.8975615906563199, 0.8975671502352668]
action: reduction
----------------------------------------
Count of function calling: 74
Count of reflection's: 18 ; function calling: 18
Count of expansion's: 2 ; function calling: 2
Count of compression's: 0 ; function calling: 0
Count of reduction's: 16 ; function calling: 32
Sum of operations calling: 36
Sum of functions calling by operations calling: 52

----------------------------------------
Result:
Coefficients: alpha 0.5 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 18
x: [[0.1987577574046839, 0.2907420609875982], [0.19873564911703384, 0.2907361370897763],
[0.19874157301485573, 0.29075824537742634]]
f(x): [0.9232597147508231, 0.9232804275081058, 0.9232868720567502]
action: reduction
----------------------------------------
Count of function calling: 74
Count of reflection's: 18 ; function calling: 18
Count of expansion's: 2 ; function calling: 2
Count of compression's: 0 ; function calling: 0
Count of reduction's: 16 ; function calling: 32
Sum of operations calling: 36
Sum of functions calling by operations calling: 52

----------------------------------------
Result:
Coefficients: alpha 0.4 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 16
x: [[0.1518775732251879, 0.362234504011435], [0.15185546493753788, 0.36222858011361314],
[0.1518613888353597, 0.3622506884012632]]
f(x): [1.0382070356526485, 1.0382306248496576, 1.0382435920662798]
action: reduction
----------------------------------------
Count of function calling: 68
Count of reflection's: 16 ; function calling: 16
Count of expansion's: 0 ; function calling: 0
Count of compression's: 0 ; function calling: 0
Count of reduction's: 16 ; function calling: 32
Sum of operations calling: 32
Sum of functions calling by operations calling: 48

----------------------------------------
Result:
Coefficients: alpha 0.3 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 16
```

```
x: [[0.1518775732251879, 0.362234504011435], [0.15185546493753788, 0.36222858011361314],
[0.1518613888353597, 0.3622506884012632]]
f(x): [1.0382070356526485, 1.0382306248496576, 1.0382435920662798]
action: reduction
---------------------------------------
Count of function calling: 68
Count of reflection's: 16 ; function calling: 16
Count of expansion's: 0 ; function calling: 0
Count of compression's: 0 ; function calling: 0
Count of reduction's: 16 ; function calling: 32
Sum of operations calling: 32
Sum of functions calling by operations calling: 48

---------------------------------------
Result:
Coefficients: alpha 0.2 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 16
x: [[0.1518775732251879, 0.362234504011435], [0.15185546493753788, 0.36222858011361314],
[0.1518613888353597, 0.3622506884012632]]
f(x): [1.0382070356526485, 1.0382306248496576, 1.0382435920662798]
action: reduction
---------------------------------------
Count of function calling: 68
Count of reflection's: 16 ; function calling: 16
Count of expansion's: 0 ; function calling: 0
Count of compression's: 0 ; function calling: 0
Count of reduction's: 16 ; function calling: 32
Sum of operations calling: 32
Sum of functions calling by operations calling: 48

---------------------------------------
Result:
Coefficients: alpha 0.1 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 16
x: [[0.1518775732251879, 0.362234504011435], [0.15185546493753788, 0.36222858011361314],
[0.1518613888353597, 0.3622506884012632]]
f(x): [1.0382070356526485, 1.0382306248496576, 1.0382435920662798]
action: reduction
---------------------------------------
Count of function calling: 68
Count of reflection's: 16 ; function calling: 16
Count of expansion's: 0 ; function calling: 0
Count of compression's: 0 ; function calling: 0
Count of reduction's: 16 ; function calling: 32
Sum of operations calling: 32
Sum of functions calling by operations calling: 48
```

Результати виконання програми для різних коефіцієнтів β

```
----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.9 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 65
x: [[0.268382784838004, 0.20700800486186083], [0.2680707105257737, 0.2070412281096769],
[0.2685402373065092, 0.20694937403066188]]
f(x): [0.8700132022561105, 0.8700261384257882, 0.8700266193214802]
action: reduction
----------------------------------------
Count of function calling: 229
Count of reflection's: 65 ; function calling: 65
Count of expansion's: 34 ; function calling: 34
Count of compression's: 1 ; function calling: 1
Count of reduction's: 30 ; function calling: 60
Sum of operations calling: 130
Sum of functions calling by operations calling: 160

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.8 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 79
x: [[0.9977432289450077, 1.007967189165425], [0.9983502765589316, 1.0085847462887518],
[0.9968069661114418, 1.0070051510622475]]
f(x): [0.18002685619236397, 0.1800173711965216, 0.18001993960155704]
action: reduction
----------------------------------------
Count of function calling: 275
Count of reflection's: 79 ; function calling: 79
Count of expansion's: 44 ; function calling: 44
Count of compression's: 1 ; function calling: 1
Count of reduction's: 34 ; function calling: 68
Sum of operations calling: 158
Sum of functions calling by operations calling: 192

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.7 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 41
x: [[0.20828595982498066, 0.17334222534866106], [0.2082631590053282, 0.1733516970135088],
[0.20823074117007123, 0.17339302044965654]]
f(x): [0.8940852144480415, 0.8940899575073927, 0.8940899224963109]
action: reduction
----------------------------------------
Count of function calling: 150
Count of reflection's: 41 ; function calling: 41
Count of expansion's: 17 ; function calling: 17
Count of compression's: 1 ; function calling: 1
Count of reduction's: 23 ; function calling: 46
Sum of operations calling: 82
Sum of functions calling by operations calling: 105

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.6 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 33
x: [[0.4177984176338255, 0.31305213035751234], [0.4213811028009562, 0.3146167084482533],
[0.41725243733734135, 0.31272033429972196]]
f(x): [0.8184333877618895, 0.8184833899954785, 0.8185190151490759]
action: reduction
----------------------------------------
Count of function calling: 120
```

```
Count of reflection's: 33 ; function calling: 33
Count of expansion's: 14 ; function calling: 14
Count of compression's: 2 ; function calling: 2
Count of reduction's: 17 ; function calling: 34
Sum of operations calling: 66
Sum of functions calling by operations calling: 83

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 57
x: [[1.0032325114580896, 0.8856881978791556], [1.0032868582498309, 0.8857400749332238],
[1.003114106854531, 0.8855668261177909]]
f(x): [0.6096899312083286, 0.6096967266458907, 0.6096967968329776]
action: reduction
----------------------------------------
Count of function calling: 203
Count of reflection's: 57 ; function calling: 57
Count of expansion's: 28 ; function calling: 28
Count of compression's: 1 ; function calling: 1
Count of reduction's: 28 ; function calling: 56
Sum of operations calling: 114
Sum of functions calling by operations calling: 142

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.4 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 147
x: [[0.9999368089833492, 1.0002278495682133], [0.9998252799450988, 1.0001123870732331],
[0.9999760413811682, 1.0002663828909166]]
f(x): [0.030372980206696973, 0.0304067643974368, 0.03030598973189948]
action: expansion
----------------------------------------
Count of function calling: 501
Count of reflection's: 147 ; function calling: 147
Count of expansion's: 88 ; function calling: 88
Count of compression's: 3 ; function calling: 3
Count of reduction's: 56 ; function calling: 112
Sum of operations calling: 294
Sum of functions calling by operations calling: 350

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 68
x: [[1.0071129539526065, 1.0056160224252202], [1.007083046903762, 1.0055686408767681],
[1.0071870247135508, 1.005720265560225]]
f(x): [0.09243446214676719, 0.09246662416073681, 0.092486646447271]
action: reduction
----------------------------------------
Count of function calling: 237
Count of reflection's: 68 ; function calling: 68
Count of expansion's: 36 ; function calling: 36
Count of compression's: 3 ; function calling: 3
Count of reduction's: 29 ; function calling: 58
Sum of operations calling: 136
Sum of functions calling by operations calling: 165

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.2 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 82
```

```
x: [[0.9171090796451777, 0.9098806193159956], [0.9164335633050904, 0.9100538363703066],
[0.9160760065826812, 0.9102142945837878]]
f(x): [0.2932318992408709, 0.29320162741957545, 0.29316665734762254]
action: reduction
---------------------------------------
Count of function calling: 284
Count of reflection's: 82 ; function calling: 82
Count of expansion's: 47 ; function calling: 47
Count of compression's: 1 ; function calling: 1
Count of reduction's: 34 ; function calling: 68
Sum of operations calling: 164
Sum of functions calling by operations calling: 198

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.1 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 71
x: [[0.4791198648885466, 0.5246268477997387], [0.47893664781427436, 0.5242624595768328],
[0.47819377990693523, 0.5226582372125788]]
f(x): [0.7351145132312127, 0.7351310884755327, 0.735131907707002]
action: reduction
---------------------------------------
Count of function calling: 247
Count of reflection's: 71 ; function calling: 71
Count of expansion's: 40 ; function calling: 40
Count of compression's: 1 ; function calling: 1
Count of reduction's: 30 ; function calling: 60
Sum of operations calling: 142
Sum of functions calling by operations calling: 172
```

Результати виконання програми для різних коефіцієнтів γ

```
----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 3.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 66
x: [[0.9999999837685604, 0.9999999805706363], [1.000000018613472, 1.0000000263865605],
[1.0000000134525293, 1.0000000007211847]]
f(x): [0.00013828949819040447, 0.00017559311781385073, 0.00020602938730765273]
action: reduction
----------------------------------------
Count of function calling: 235
Count of reflection's: 66 ; function calling: 66
Count of expansion's: 27 ; function calling: 27
Count of compression's: 6 ; function calling: 6
Count of reduction's: 33 ; function calling: 66
Sum of operations calling: 132
Sum of functions calling by operations calling: 165

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.9 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 73
x: [[0.999568062451912, 0.9990668494306484], [0.9994136634154059, 0.9989272114418415],
[0.9995751599077956, 0.9990746065461317]]
f(x): [0.040531180645267104, 0.040574031735347155, 0.04048345263596706]
action: reflection
----------------------------------------
Count of function calling: 256
Count of reflection's: 73 ; function calling: 73
Count of expansion's: 36 ; function calling: 36
Count of compression's: 4 ; function calling: 4
Count of reduction's: 33 ; function calling: 66
Sum of operations calling: 146
Sum of functions calling by operations calling: 179

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.8 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 68
x: [[0.9998450963454689, 1.0001009938855199], [0.9998432110123547, 1.0000992940123468],
[0.9998420264834622, 1.0000980378346576]]
f(x): [0.028703863830782383, 0.028720099510211226, 0.02872016215182755]
action: reduction
----------------------------------------
Count of function calling: 244
Count of reflection's: 68 ; function calling: 68
Count of expansion's: 29 ; function calling: 29
Count of compression's: 3 ; function calling: 3
Count of reduction's: 36 ; function calling: 72
Sum of operations calling: 136
Sum of functions calling by operations calling: 172

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.7 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 97
x: [[1.0000000052865923, 1.0000000204263846], [1.000000055653298, 1.0000000620459777],
[1.000000021174952, 1.0000000125837063]]
f(x): [0.000219470318953841, 0.00024333328884114857, 0.00018559425208090877]
action: reflection
----------------------------------------
Count of function calling: 342
```

```
Count of reflection's: 97 ; function calling: 97
Count of expansion's: 46 ; function calling: 46
Count of compression's: 4 ; function calling: 4
Count of reduction's: 47 ; function calling: 94
Sum of operations calling: 194
Sum of functions calling by operations calling: 241

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.6 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 114
x: [[0.999999950014371, 0.999999952723], [1.0000000011381294, 0.9999999801801185],
[1.000000069985517, 1.0000000621832876]]
f(x): [0.00022519812911140312, 0.00025745838758641854, 0.0002724101319402264]
action: reduction
----------------------------------------
Count of function calling: 400
Count of reflection's: 114 ; function calling: 114
Count of expansion's: 55 ; function calling: 55
Count of compression's: 5 ; function calling: 5
Count of reduction's: 54 ; function calling: 108
Sum of operations calling: 228
Sum of functions calling by operations calling: 282

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.5 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 43
x: [[0.9999999528589394, 0.9999999465387601], [1.0000000074803424, 1.0000000973432122],
[1.0000000395715452, 0.9999999860992734]]
f(x): [0.00022628034688537547, 0.0003214414395753677, 0.0004167286110352379]
action: reduction
----------------------------------------
Count of function calling: 160
Count of reflection's: 43 ; function calling: 43
Count of expansion's: 13 ; function calling: 13
Count of compression's: 3 ; function calling: 3
Count of reduction's: 27 ; function calling: 54
Sum of operations calling: 86
Sum of functions calling by operations calling: 113

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.4 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 63
x: [[1.0000029457280328, 1.0000026576989962], [0.9999982840424027, 0.9999975338303821],
[0.9999974851272355, 0.9999971111615427]]
f(x): [0.0017559412050990903, 0.0017111153822647487, 0.0016670466444922499]
action: reduction
----------------------------------------
Count of function calling: 224
Count of reflection's: 63 ; function calling: 63
Count of expansion's: 29 ; function calling: 29
Count of compression's: 3 ; function calling: 3
Count of reduction's: 31 ; function calling: 62
Sum of operations calling: 126
Sum of functions calling by operations calling: 157

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.3 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 91
```

```
x: [[1.0485361027662725, 1.0749935015741539], [1.0480786862910538, 1.0746236243596345],
[1.0477893196481798, 1.074387235503687]]
f(x): [0.3110061579108618, 0.31102447834740643, 0.3110279091768485]
action: reduction
---------------------------------------
Count of function calling: 320
Count of reflection's: 91 ; function calling: 91
Count of expansion's: 48 ; function calling: 48
Count of compression's: 0 ; function calling: 0
Count of reduction's: 43 ; function calling: 86
Sum of operations calling: 182
Sum of functions calling by operations calling: 225

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.2 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 64
x: [[1.0000003887455964, 1.0000003510215132], [0.9999995494509769, 0.999999381493822],
[0.9999998896945235, 0.999999552573923]]
f(x): [0.000637681488311221, 0.0008345563926848354, 0.001035258600505466]
action: reduction
---------------------------------------
Count of function calling: 226
Count of reflection's: 64 ; function calling: 64
Count of expansion's: 28 ; function calling: 28
Count of compression's: 6 ; function calling: 6
Count of reduction's: 30 ; function calling: 60
Sum of operations calling: 128
Sum of functions calling by operations calling: 158

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.1 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 125
x: [[0.9476375214689348, 0.978616082220347], [0.9110809703571345, 0.9332834210655014],
[1.0203083806011968, 1.0529410541215738]]
f(x): [0.33328514413750165, 0.33659535680256236, 0.3243037540749686]
action: expansion
---------------------------------------
Count of function calling: 424
Count of reflection's: 125 ; function calling: 125
Count of expansion's: 78 ; function calling: 78
Count of compression's: 2 ; function calling: 2
Count of reduction's: 45 ; function calling: 90
Sum of operations calling: 250
Sum of functions calling by operations calling: 295

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 57
x: [[1.0032325114580896, 0.8856881978791556], [1.0032868582498309, 0.8857400749332238],
[1.003114106854531, 0.8855668261177909]]
f(x): [0.6096899312083286, 0.6096967266458907, 0.6096967968329776]
action: reduction
---------------------------------------
Count of function calling: 203
Count of reflection's: 57 ; function calling: 57
Count of expansion's: 28 ; function calling: 28
Count of compression's: 1 ; function calling: 1
Count of reduction's: 28 ; function calling: 56
Sum of operations calling: 114
Sum of functions calling by operations calling: 142

---------------------------------------
```

```
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 1.9 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 119
x: [[0.9996668401244629, 0.9989811989885797], [0.9994758212934739, 0.9988013581966034],
[0.9997499597783026, 0.9990624566713207]]
f(x): [0.04683625406599486, 0.04686478221855429, 0.046780412899563216]
action: expansion
----------------------------------------
Count of function calling: 404
Count of reflection's: 119 ; function calling: 119
Count of expansion's: 72 ; function calling: 72
Count of compression's: 4 ; function calling: 4
Count of reduction's: 43 ; function calling: 86
Sum of operations calling: 238
Sum of functions calling by operations calling: 281

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 1.8 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 89
x: [[1.115421261218125, 1.155081083276925], [1.115216072784827, 1.1549419744465368],
[1.1158172863661209, 1.1553590228171928]]
f(x): [0.41284837501970484, 0.41286661256598145, 0.4128414491930073]
action: expansion
----------------------------------------
Count of function calling: 308
Count of reflection's: 89 ; function calling: 89
Count of expansion's: 52 ; function calling: 52
Count of compression's: 0 ; function calling: 0
Count of reduction's: 37 ; function calling: 74
Sum of operations calling: 178
Sum of functions calling by operations calling: 215

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 1.7 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 74
x: [[0.8554394767324126, 0.7510966370071357], [0.8593280104482961, 0.7544528722133805],
[0.8530127613628513, 0.7490162217478868]]
f(x): [0.6001990061855931, 0.6002042415721447, 0.60018284561589]
action: reflection
----------------------------------------
Count of function calling: 254
Count of reflection's: 74 ; function calling: 74
Count of expansion's: 44 ; function calling: 44
Count of compression's: 2 ; function calling: 2
Count of reduction's: 28 ; function calling: 56
Sum of operations calling: 148
Sum of functions calling by operations calling: 176

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 1.6 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 57
x: [[0.697775550840626, 0.5788831295121677], [0.6976967486158392, 0.5788159129606678],
[0.6969358725331999, 0.578260319185238]]
f(x): [0.6945382277447459, 0.6945532186590774, 0.6945330332971155]
action: expansion
----------------------------------------
Count of function calling: 199
Count of reflection's: 57 ; function calling: 57
Count of expansion's: 32 ; function calling: 32
Count of compression's: 1 ; function calling: 1
```

```
Count of reduction's: 24 ; function calling: 48
Sum of operations calling: 114
Sum of functions calling by operations calling: 138

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 1.5 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 31
x: [[0.3121208105770803, 0.2549018584925592], [0.3120605052902441, 0.25489284651668437],
[0.3120926796391552, 0.25491290614348244]]
f(x): [0.8433736027800514, 0.8433837282498969, 0.8433710532156541]
action: expansion
----------------------------------------
Count of function calling: 116
Count of reflection's: 31 ; function calling: 31
Count of expansion's: 12 ; function calling: 12
Count of compression's: 0 ; function calling: 0
Count of reduction's: 19 ; function calling: 38
Sum of operations calling: 62
Sum of functions calling by operations calling: 81

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 1.4 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 149
x: [[0.6141684433684478, 0.7430710411417121], [0.6140245571916634, 0.742890324917788],
[0.6138867046580314, 0.7427111346734476]]
f(x): [0.7491800011607416, 0.7491895931266269, 0.7491895403098319]
action: reduction
----------------------------------------
Count of function calling: 492
Count of reflection's: 149 ; function calling: 149
Count of expansion's: 107 ; function calling: 107
Count of compression's: 1 ; function calling: 1
Count of reduction's: 41 ; function calling: 82
Sum of operations calling: 298
Sum of functions calling by operations calling: 339

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 1.3 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 48
x: [[0.3119906486170284, 0.24170825642154176], [0.3117581632346635, 0.2416880107494906],
[0.3121031054577212, 0.24169441424948274]]
f(x): [0.8503038322169705, 0.8503128085840319, 0.8503131692399906]
action: reduction
----------------------------------------
Count of function calling: 167
Count of reflection's: 48 ; function calling: 48
Count of expansion's: 28 ; function calling: 28
Count of compression's: 1 ; function calling: 1
Count of reduction's: 19 ; function calling: 38
Sum of operations calling: 96
Sum of functions calling by operations calling: 115

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 1.2 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 35
x: [[0.3047777812373, 0.2554330815146619], [0.3047400046618437, 0.2553998510554627],
[0.3048436953639759, 0.2554111290967517 3]]
f(x): [0.8441082399280401, 0.8441282084161443, 0.8441062225158847]
action: expansion
```

```
---------------------------------------
Count of function calling: 126
Count of reflection's: 35 ; function calling: 35
Count of expansion's: 18 ; function calling: 18
Count of compression's: 0 ; function calling: 0
Count of reduction's: 17 ; function calling: 34
Sum of operations calling: 70
Sum of functions calling by operations calling: 87

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 1.1 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 72
x: [[0.25637835502685585, 0.22438730739485999], [0.25626829357721737, 0.22447579727299583],
[0.25618981074841773, 0.22458317734184607]]
f(x): [0.8662975623797903, 0.8663118119094876, 0.8663113931824613]
action: reduction
---------------------------------------
Count of function calling: 238
Count of reflection's: 72 ; function calling: 72
Count of expansion's: 54 ; function calling: 54
Count of compression's: 0 ; function calling: 0
Count of reduction's: 18 ; function calling: 36
Sum of operations calling: 144
Sum of functions calling by operations calling: 162
```

Результати виконання програми для різних коефіцієнтів σ

```
----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.9
Accuracy: 5
Side length: 1.5
Itterations: 224
x: [[1.000000001675245, 1.0000000009489265], [0.9999999985154896, 0.9999999977631884],
[1.0000000006067198, 1.0000000013874026]]
f(x): [5.331844333414005e-05, 5.2954368943665056e-05, 5.042020518212302e-05]
action: reduction
----------------------------------------
Count of function calling: 875
Count of reflection's: 224 ; function calling: 224
Count of expansion's: 20 ; function calling: 20
Count of compression's: 5 ; function calling: 5
Count of reduction's: 199 ; function calling: 398
Sum of operations calling: 448
Sum of functions calling by operations calling: 647

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.8
Accuracy: 5
Side length: 1.5
Itterations: 94
x: [[1.0000001937588383, 1.0000000702215994], [1.0000004284681048, 1.0000003267473798],
[1.0000003571728444, 1.0000004736172627]]
f(x): [0.0006603559094238416, 0.0007319672464758365, 0.0007162377788984304]
action: reduction
----------------------------------------
Count of function calling: 356
Count of reflection's: 94 ; function calling: 94
Count of expansion's: 16 ; function calling: 16
Count of compression's: 8 ; function calling: 8
Count of reduction's: 70 ; function calling: 140
Sum of operations calling: 188
Sum of functions calling by operations calling: 258

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.7
Accuracy: 5
Side length: 1.5
Itterations: 29
x: [[0.27890803663212893, 0.24287090494212193], [0.2787853622386841, 0.24264866128360205],
[0.278685792973609, 0.2427482305486771]]
f(x): [0.8544245706540695, 0.8545242690159255, 0.854524310646907]
action: reduction
----------------------------------------
Count of function calling: 117
Count of reflection's: 29 ; function calling: 29
Count of expansion's: 3 ; function calling: 3
Count of compression's: 0 ; function calling: 0
Count of reduction's: 26 ; function calling: 52
Sum of operations calling: 58
Sum of functions calling by operations calling: 84

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.6
Accuracy: 5
Side length: 1.5
Itterations: 147
x: [[0.9999999998037547, 0.9999999977972994], [1.0000000022734175, 1.0000000015083037],
[1.0000000027693547, 1.0000000030367908]]
f(x): [4.587351037240816e-05, 5.761946393958365e-05, 5.381085480592759e-05]
action: reduction
----------------------------------------
```

```
Count of function calling: 520
Count of reflection's: 147 ; function calling: 147
Count of expansion's: 68 ; function calling: 68
Count of compression's: 4 ; function calling: 4
Count of reduction's: 75 ; function calling: 150
Sum of operations calling: 294
Sum of functions calling by operations calling: 369

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 57
x: [[1.0032325114580896, 0.8856881978791556], [1.0032868582498309, 0.8857400749332238],
[1.003114106854531, 0.8855668261177909]]
f(x): [0.6096899312083286, 0.6096967266458907, 0.6096967968329776]
action: reduction
---------------------------------------
Count of function calling: 203
Count of reflection's: 57 ; function calling: 57
Count of expansion's: 28 ; function calling: 28
Count of compression's: 1 ; function calling: 1
Count of reduction's: 28 ; function calling: 56
Sum of operations calling: 114
Sum of functions calling by operations calling: 142

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.4
Accuracy: 5
Side length: 1.5
Itterations: 103
x: [[0.999999984204246, 0.9999999961015082], [0.9999999714476486, 0.9999999743937145],
[1.000000018475948, 1.000000024458222]]
f(x): [0.00020199963386688098, 0.00017330265745087072, 0.00016261329377423276]
action: reduction
---------------------------------------
Count of function calling: 349
Count of reflection's: 103 ; function calling: 103
Count of expansion's: 60 ; function calling: 60
Count of compression's: 7 ; function calling: 7
Count of reduction's: 36 ; function calling: 72
Sum of operations calling: 206
Sum of functions calling by operations calling: 242

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.3
Accuracy: 5
Side length: 1.5
Itterations: 21
x: [[0.3746523047129079, 0.40328944578927023], [0.3746465828878444, 0.40328350708017247],
[0.3746437141133002, 0.4032743739901197]]
f(x): [0.7949029584990152, 0.7949064585814553, 0.7949064588711368]
action: reduction
---------------------------------------
Count of function calling: 79
Count of reflection's: 21 ; function calling: 21
Count of expansion's: 8 ; function calling: 8
Count of compression's: 1 ; function calling: 1
Count of reduction's: 12 ; function calling: 24
Sum of operations calling: 42
Sum of functions calling by operations calling: 54

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.2
Accuracy: 5
Side length: 1.5
Itterations: 33
```

```
x: [[0.49311226677624775, 0.48159696930596], [0.49373335218824993, 0.47973499004814824],
[0.49274817550043926, 0.48311968750522094]]
f(x): [0.7128773052615556, 0.7128802485893005, 0.7128568067087806]
action: expansion
---------------------------------------
Count of function calling: 112
Count of reflection's: 33 ; function calling: 33
Count of expansion's: 23 ; function calling: 23
Count of compression's: 1 ; function calling: 1
Count of reduction's: 9 ; function calling: 18
Sum of operations calling: 66
Sum of functions calling by operations calling: 75

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.1
Accuracy: 5
Side length: 1.5
Itterations: 26
x: [[0.33235118711079964, 0.34173820314100556], [0.33226661261737694, 0.3410523119252293],
[0.3322865284159514, 0.3410250872993167]]
f(x): [0.8175013060795896, 0.8175029825382236, 0.8174870317950531]
action: expansion
---------------------------------------
Count of function calling: 88
Count of reflection's: 26 ; function calling: 26
Count of expansion's: 17 ; function calling: 17
Count of compression's: 3 ; function calling: 3
Count of reduction's: 6 ; function calling: 12
Sum of operations calling: 52
Sum of functions calling by operations calling: 58
```

Результати виконання програми для різних значень параметра t

```
---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 2.0
Itterations: 50
x: [[0.9999999908336099, 0.9999999886015599], [1.0000000315565827, 1.0000000250851526],
[0.9999999942840545, 0.999999996414862]]
f(x): [0.00010755957083407969, 0.00019393640148577152, 9.40001638046011e-05]
action: reflection
---------------------------------------
Count of function calling: 182
Count of reflection's: 50 ; function calling: 50
Count of expansion's: 19 ; function calling: 19
Count of compression's: 3 ; function calling: 3
Count of reduction's: 28 ; function calling: 56
Sum of operations calling: 100
Sum of functions calling by operations calling: 128

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.9
Itterations: 75
x: [[1.000144766068169, 1.0001195300363488], [1.0001448949186797, 1.0001195998215546],
[1.0001449785042653, 1.0001197312922772]]
f(x): [0.012857104439363345, 0.012864998156903729, 0.012865001325353202]
action: reduction
---------------------------------------
Count of function calling: 267
Count of reflection's: 75 ; function calling: 75
Count of expansion's: 34 ; function calling: 34
Count of compression's: 3 ; function calling: 3
Count of reduction's: 38 ; function calling: 76
Sum of operations calling: 150
Sum of functions calling by operations calling: 188

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.8
Itterations: 43
x: [[0.5715229938128435, 0.5285032219599276], [0.5709164453440294, 0.5284781375839047],
[0.5703989630111472, 0.5285256701071441]]
f(x): [0.6704885071223441, 0.6705077849498244, 0.6704812290143509]
action: expansion
---------------------------------------
Count of function calling: 155
Count of reflection's: 43 ; function calling: 43
Count of expansion's: 20 ; function calling: 20
Count of compression's: 1 ; function calling: 1
Count of reduction's: 22 ; function calling: 44
Sum of operations calling: 86
Sum of functions calling by operations calling: 108

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.7
Itterations: 80
x: [[0.9999999372827717, 0.9999998837038384], [1.0000000749170403, 1.000000047888324],
[0.9999999914415546, 1.0000000129144744]]
f(x): [0.00042504910535185944, 0.0003371316109723868, 0.0002616114616540735]
action: reduction
---------------------------------------
Count of function calling: 283
```

```
Count of reflection's: 80 ; function calling: 80
Count of expansion's: 39 ; function calling: 39
Count of compression's: 2 ; function calling: 2
Count of reduction's: 39 ; function calling: 78
Sum of operations calling: 160
Sum of functions calling by operations calling: 199

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.6
Itterations: 176
x: [[0.9999999898332399, 0.9999999913878325], [0.9999999869949714, 0.9999999895376095],
[0.999999983770332, 0.9999999847254668]]
f(x): [0.00010626826581436475, 0.00012365231036599492, 0.00012848474520529076]
action: reduction
---------------------------------------
Count of function calling: 603
Count of reflection's: 176 ; function calling: 176
Count of expansion's: 100 ; function calling: 100
Count of compression's: 5 ; function calling: 5
Count of reduction's: 71 ; function calling: 142
Sum of operations calling: 352
Sum of functions calling by operations calling: 423

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.5
Itterations: 57
x: [[1.0032325114580896, 0.8856881978791556], [1.0032868582498309, 0.8857400749332238],
[1.003114106854531, 0.8855668261177909]]
f(x): [0.6096899312083286, 0.6096967266458907, 0.6096967968329776]
action: reduction
---------------------------------------
Count of function calling: 203
Count of reflection's: 57 ; function calling: 57
Count of expansion's: 28 ; function calling: 28
Count of compression's: 1 ; function calling: 1
Count of reduction's: 28 ; function calling: 56
Sum of operations calling: 114
Sum of functions calling by operations calling: 142

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.4
Itterations: 47
x: [[1.0000001132912124, 1.000000124761002], [1.000000121259462, 1.000000095396981],
[1.0000000699345504, 1.0000000953346142]]
f(x): [0.0003448995565244207, 0.00038244199045463017, 0.0003263449869670708]
action: expansion
---------------------------------------
Count of function calling: 173
Count of reflection's: 47 ; function calling: 47
Count of expansion's: 17 ; function calling: 17
Count of compression's: 2 ; function calling: 2
Count of reduction's: 28 ; function calling: 56
Sum of operations calling: 94
Sum of functions calling by operations calling: 122

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.3
Itterations: 66
```

```
x: [[1.000021225102661, 0.9999328667327478], [1.0000189432148252, 0.9999303859243694],
[1.0000167810029632, 0.9999281810600492]]
f(x): [0.016739734193679117, 0.016753587900750405, 0.016753498509652957]
action: reduction
---------------------------------------
Count of function calling: 234
Count of reflection's: 66 ; function calling: 66
Count of expansion's: 30 ; function calling: 30
Count of compression's: 4 ; function calling: 4
Count of reduction's: 32 ; function calling: 64
Sum of operations calling: 132
Sum of functions calling by operations calling: 164

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.2
Itterations: 66
x: [[1.0000001971332284, 1.0000001539162278], [0.9999999589076074, 0.9999998755979533],
[1.000000131800349, 1.0000001519797852]]
f(x): [0.0004897673444671403, 0.0005163659301341224, 0.00038266924164650546]
action: expansion
---------------------------------------
Count of function calling: 231
Count of reflection's: 66 ; function calling: 66
Count of expansion's: 30 ; function calling: 30
Count of compression's: 7 ; function calling: 7
Count of reduction's: 29 ; function calling: 58
Sum of operations calling: 132
Sum of functions calling by operations calling: 161

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5
Accuracy: 5
Side length: 1.1
Itterations: 77
x: [[0.9999994315740265, 0.9999996545149225], [0.9999997733214567, 0.9999992968975737],
[0.9999984281636174, 0.9999979157626979]]
f(x): [0.0009516368982324674, 0.0012343184486072276, 0.0015024913150998844]
action: reduction
---------------------------------------
Count of function calling: 270
Count of reflection's: 77 ; function calling: 77
Count of expansion's: 40 ; function calling: 40
Count of compression's: 2 ; function calling: 2
Count of reduction's: 35 ; function calling: 70
Sum of operations calling: 154
Sum of functions calling by operations calling: 189
```

Результати виконання програми для різних значень параметра ε і нових $\alpha, \beta, \gamma, \sigma, t$

```
----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 10
Side length: 2.0
Itterations: 122
x: [[0.9999999999925961, 0.9999999999866329], [0.9999999999925892, 0.9999999999866266],
[0.9999999999925787, 0.9999999999866175]]
f(x): [4.5009754934048645e-06, 4.501041781614567e-06, 4.501030630833148e-06]
action: reduction
----------------------------------------
Count of function calling: 428
Count of reflection's: 122 ; function calling: 122
Count of expansion's: 57 ; function calling: 57
Count of compression's: 7 ; function calling: 7
Count of reduction's: 58 ; function calling: 116
Sum of operations calling: 244
Sum of functions calling by operations calling: 302

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 9
Side length: 2.0
Itterations: 97
x: [[0.9999999999918832, 0.9999999999858371], [0.99999999999181, 0.999999999985771],
[0.9999999999917395, 0.9999999999857102]]
f(x): [4.557511150138357e-06, 4.5583939627644635e-06, 4.558378346327993e-06]
action: reduction
----------------------------------------
Count of function calling: 344
Count of reflection's: 97 ; function calling: 97
Count of expansion's: 43 ; function calling: 43
Count of compression's: 5 ; function calling: 5
Count of reduction's: 49 ; function calling: 98
Sum of operations calling: 194
Sum of functions calling by operations calling: 243

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 8
Side length: 2.0
Itterations: 91
x: [[0.9999999999896241, 0.9999999999838276], [0.9999999999893777, 0.9999999999836134],
[0.9999999999892618, 0.9999999999835197]]
f(x): [4.589446871621433e-06, 4.5931946508892704e-06, 4.592990598237077e-06]
action: reduction
----------------------------------------
Count of function calling: 323
Count of reflection's: 91 ; function calling: 91
Count of expansion's: 40 ; function calling: 40
Count of compression's: 5 ; function calling: 5
Count of reduction's: 46 ; function calling: 92
Sum of operations calling: 182
Sum of functions calling by operations calling: 228

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 7
Side length: 2.0
Itterations: 83
x: [[0.9999999999797383, 0.9999999999754488], [0.9999999999791076, 0.9999999999747206],
[0.9999999999820941, 0.9999999999771889]]
f(x): [4.937933296063816e-06, 5.0078894742146785e-06, 4.867267220547887e-06]
action: expansion
```

```
---------------------------------------
Count of function calling: 294
Count of reflection's: 83 ; function calling: 83
Count of expansion's: 37 ; function calling: 37
Count of compression's: 5 ; function calling: 5
Count of reduction's: 41 ; function calling: 82
Sum of operations calling: 166
Sum of functions calling by operations calling: 207

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 2.0
Itterations: 78
x: [[0.9999999999708289, 0.9999999999651631], [0.999999999961554, 0.9999999999578559],
[0.9999999999618443, 0.9999999999598913]]
f(x): [5.850982734838354e-06, 6.33918133791119e-06, 6.217096393751296e-06]
action: reduction
---------------------------------------
Count of function calling: 276
Count of reflection's: 78 ; function calling: 78
Count of expansion's: 35 ; function calling: 35
Count of compression's: 5 ; function calling: 5
Count of reduction's: 38 ; function calling: 76
Sum of operations calling: 156
Sum of functions calling by operations calling: 194

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 2.0
Itterations: 61
x: [[0.9999999757697573, 0.9999999641747473], [0.9999999737082753, 0.999999961168482],
[0.999999970036442, 0.9999999584098948]]
f(x): [0.00020964105297079734, 0.00021812509157984443, 0.00021778371422280 12]
action: reduction
---------------------------------------
Count of function calling: 217
Count of reflection's: 61 ; function calling: 61
Count of expansion's: 28 ; function calling: 28
Count of compression's: 3 ; function calling: 3
Count of reduction's: 30 ; function calling: 60
Sum of operations calling: 122
Sum of functions calling by operations calling: 152

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 4
Side length: 2.0
Itterations: 46
x: [[1.0000035519563304, 1.0000037219741784], [1.000002917806432, 1.0000019413615466],
[1.000002881501992, 1.0000033555557575]]
f(x): [0.0018953670469108556, 0.002061140107744177, 0.001802255849017763]
action: reflection
---------------------------------------
Count of function calling: 163
Count of reflection's: 46 ; function calling: 46
Count of expansion's: 22 ; function calling: 22
Count of compression's: 3 ; function calling: 3
Count of reduction's: 21 ; function calling: 42
Sum of operations calling: 92
Sum of functions calling by operations calling: 113

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 3
Side length: 2.0
```

```
Itterations: 41
x: [[1.0000008069115613, 0.99999785324854], [1.000001395768155, 0.9999994364919855],
[1.0000032530990346, 1.0000006507367942]]
f(x): [0.0030618772777359564, 0.002520128398349644, 0.0029747339501428028]
action: reduction for cycling of min
----------------------------------------
Count of function calling: 147
Count of reflection's: 41 ; function calling: 41
Count of expansion's: 19 ; function calling: 19
Count of compression's: 2 ; function calling: 2
Count of reduction's: 20 ; function calling: 40
Sum of operations calling: 82
Sum of functions calling by operations calling: 102

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 2
Side length: 2.0
Itterations: 8
x: [[0.6371683883556649, 0.5989961302267724], [0.6089978373895395, 0.6052192530975606],
[0.729193493341415, 0.6511969812960962]]
f(x): [0.618372550947606, 0.6254475983385184, 0.6052216287627302]
action: expansion
----------------------------------------
Count of function calling: 32
Count of reflection's: 8 ; function calling: 8
Count of expansion's: 3 ; function calling: 3
Count of compression's: 1 ; function calling: 1
Count of reduction's: 4 ; function calling: 8
Sum of operations calling: 16
Sum of functions calling by operations calling: 20

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 1
Side length: 2.0
Itterations: 3
x: [[0.7318516525781364, 0.5176380902050414], [0.5557857090398522, 0.5565326081474676],
[0.42275538816563457, 0.43481599577223473]]
f(x): [0.8535485648322192, 0.6664987397385063, 0.760593962528077]
action: reduction
----------------------------------------
Count of function calling: 15
Count of reflection's: 3 ; function calling: 3
Count of expansion's: 0 ; function calling: 0
Count of compression's: 1 ; function calling: 1
Count of reduction's: 2 ; function calling: 4
Sum of operations calling: 6
Sum of functions calling by operations calling: 8
```

ДОДАТОК Б УМОВНА ОПТИМІЗАЦІЯ ВИПУКЛА ОБЛАСТЬ

Даний додаток містить інформацію про останні ітерації для певної модифікації параметра коефіцієнта в наступному форматі:

```
----------------------------------------
Result: візуальне розділення
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5 значення відповідних коефіцієнтів, з якими був проведений ряд ітерацій
Accuracy: 5 точність перевірки критерію зупинки, відємеа степінь 1: 10^-5
Side length: 1.5 довжина сторони багатокутника
itterations: 57 - кількість ітерацій, співпадає з кількістю обчислень функції на етапі перевірки критерію зупинки, а саме з кількістю обчислень функції в центрі ваги багатокутника (усі інші задіяні значення функції вже відомі на момент підрахування критерію)
x: [[1.0032325114580896, 0.8856881978791556], [1.0032868582498309, 0.8857400749332238], [1.003114106854531, 0.8855668261177909]] x – координати вершин багатокутника на останній ітерації
f(x): [0.6096899312083286, 0.6096967266458907, 0.6096967968329776] значення функції в відповідних точках, порядок співпадає з минулим пунктом
action: reduction остання операція яка була проведена
----------------------------------------
Count of function calling: 203 загальна кількість викликів функції
Кількість звернень до операції, кількість обчислень функції цим типом операції
Count of reflection's: 57 ; function calling: 57 для віддзеркалення
Count of expansion's: 28 ; function calling: 28 для розтягнення
Count of compression's: 1 ; function calling: 1 для стискання
Count of reduction's: 28 ; function calling: 56 для редукції
Sum of operations calling: 114 загальна кількість викликів операцій
Sum of functions calling by operations calling: 142 загальна кількість обчислень функцій, здійснена операціями

----------------------------------------
візуальне розділення
```

Результати виконання програми для різних значень параметра ε і початкових $\alpha, \beta, \gamma, \sigma, t$

```
----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 10
Side length: 2.0
Itterations: 248
x: [[0.6823065986341069, 0.6759599603541259], [0.6823065989036503, 0.6759599591874117], [0.6823065987873206, 0.6759599603264475]]
f(x): [0.5642044946656677, 0.5642044946810404, 0.5642044945621206]
action: compression
----------------------------------------
Count of function calling: 823
Count of reflection's: 248 ; function calling: 248
Count of expansion's: 138 ; function calling: 138
Count of compression's: 35 ; function calling: 35
Count of reduction's: 75 ; function calling: 150
Sum of operations calling: 496
Sum of functions calling by operations calling: 571

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 9
Side length: 2.0
Itterations: 155
x: [[0.6823659350235444, 0.6751443437208322], [0.6823661046167403, 0.6751437648187485], [0.6823665570643582, 0.675142228135210 9]]
f(x): [0.5643172350655798, 0.5643172355872826, 0.5643172355010246]
action: reduction
----------------------------------------
Count of function calling: 519
Count of reflection's: 155 ; function calling: 155
Count of expansion's: 85 ; function calling: 85
```

```
Count of compression's: 20 ; function calling: 20
Count of reduction's: 50 ; function calling: 100
Sum of operations calling: 310
Sum of functions calling by operations calling: 360

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 8
Side length: 2.0
Itterations: 103
x: [[0.6820377176965269, 0.6762938479592716], [0.6820374861607197, 0.6762948314052003],
[0.6820375719898042, 0.6762945761048011]]
f(x): [0.5643409111817962, 0.5643409218646467, 0.5643409004439435]
action: expansion
----------------------------------------
Count of function calling: 344
Count of reflection's: 103 ; function calling: 103
Count of expansion's: 52 ; function calling: 52
Count of compression's: 20 ; function calling: 20
Count of reduction's: 31 ; function calling: 62
Sum of operations calling: 206
Sum of functions calling by operations calling: 237

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 7
Side length: 2.0
Itterations: 71
x: [[0.6822712168517221, 0.6751322338256625], [0.6822779643253063, 0.6751085351434385],
[0.6822789947952581, 0.6751060618979827]]
f(x): [0.5643844391862 67, 0.5643845346869036, 0.5643843229070507]
action: expansion
----------------------------------------
Count of function calling: 236
Count of reflection's: 71 ; function calling: 71
Count of expansion's: 34 ; function calling: 34
Count of compression's: 18 ; function calling: 18
Count of reduction's: 19 ; function calling: 38
Sum of operations calling: 142
Sum of functions calling by operations calling: 161

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 2.0
Itterations: 67
x: [[0.6822051007850544, 0.6753650216898957], [0.6822348768456379, 0.6752544725400615],
[0.6823105300598072, 0.6749947297801472]]
f(x): [0.5643847650964801, 0.5643854175114396, 0.5643852432819584]
action: reduction
----------------------------------------
Count of function calling: 221
Count of reflection's: 67 ; function calling: 67
Count of expansion's: 33 ; function calling: 33
Count of compression's: 18 ; function calling: 18
Count of reduction's: 16 ; function calling: 32
Sum of operations calling: 134
Sum of functions calling by operations calling: 150

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 2.0
Itterations: 52
x: [[0.6809891226701507, 0.679518374699039], [0.6809634897833953, 0.6796108936216885],
[0.6809941275412342, 0.6796018124186978]]
f(x): [0.5648404896559822, 0.5648585499502821, 0.5648329444304993]
```

```
action: compression
---------------------------------------
Count of function calling: 173
Count of reflection's: 52 ; function calling: 52
Count of expansion's: 24 ; function calling: 24
Count of compression's: 15 ; function calling: 15
Count of reduction's: 13 ; function calling: 26
Sum of operations calling: 104
Sum of functions calling by operations calling: 117

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 4
Side length: 2.0
Itterations: 36
x: [[0.6806528419072689, 0.6774471598403463], [0.680326009444262, 0.67670547779643],
[0.6805022333087231, 0.6783244054163153]]
f(x): [0.5652503998607546, 0.5655784208429993, 0.5653069811302995]
action: compression
---------------------------------------
Count of function calling: 122
Count of reflection's: 36 ; function calling: 36
Count of expansion's: 17 ; function calling: 17
Count of compression's: 9 ; function calling: 9
Count of reduction's: 10 ; function calling: 20
Sum of operations calling: 72
Sum of functions calling by operations calling: 82

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 3
Side length: 2.0
Itterations: 10
x: [[0.654916227013246, 0.6168344785523454], [0.6371683883556649, 0.5989961302267724],
[0.6349289665959806, 0.6071064890019594]]
f(x): [0.6045600269996627, 0.618372550947606, 0.612799573720605]
action: compression
---------------------------------------
Count of function calling: 38
Count of reflection's: 10 ; function calling: 10
Count of expansion's: 3 ; function calling: 3
Count of compression's: 3 ; function calling: 3
Count of reduction's: 4 ; function calling: 8
Sum of operations calling: 20
Sum of functions calling by operations calling: 24

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 2
Side length: 2.0
Itterations: 9
x: [[0.6371683883556649, 0.5989961302267724], [0.6089978373895395, 0.6052192530975606],
[0.654916227013246, 0.6168344785523454]]
f(x): [0.618372550947606, 0.6254475983385184, 0.6045600269996627]
action: compression
---------------------------------------
Count of function calling: 35
Count of reflection's: 9 ; function calling: 9
Count of expansion's: 3 ; function calling: 3
Count of compression's: 2 ; function calling: 2
Count of reduction's: 4 ; function calling: 8
Sum of operations calling: 18
Sum of functions calling by operations calling: 22

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 1
```

```
Side length: 2.0
Itterations: 3
x: [[0.7318516525781364, 0.5176380902050414], [0.5557857090398522, 0.5565326081474676],
[0.42275538816563457, 0.4348159957722347 3]]
f(x): [0.8535485648322192, 0.6664987397385063, 0.760593962528077]
action: reduction
---------------------------------------
Count of function calling: 15
Count of reflection's: 3 ; function calling: 3
Count of expansion's: 0 ; function calling: 0
Count of compression's: 1 ; function calling: 1
Count of reduction's: 2 ; function calling: 4
Sum of operations calling: 6
Sum of functions calling by operations calling: 8
```

Результати виконання програми для різних значень параметра t

```
---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 2.0
Itterations: 52
x: [[0.6809891226701507, 0.679518374699039], [0.6809634897833953, 0.6796108936216885],
[0.6809941275412342, 0.6796018124186978]]
f(x): [0.5648404896559822, 0.5648585499502821, 0.5648329444304993]
action: compression
---------------------------------------
Count of function calling: 173
Count of reflection's: 52 ; function calling: 52
Count of expansion's: 24 ; function calling: 24
Count of compression's: 15 ; function calling: 15
Count of reduction's: 13 ; function calling: 26
Sum of operations calling: 104
Sum of functions calling by operations calling: 117

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.9
Itterations: 58
x: [[0.6821756368638126, 0.676239068344584], [0.6821554496226209, 0.6763547747388667],
[0.682175826076724, 0.6763030330373101]]
f(x): [0.564250227642806, 0.5642458888073653, 0.5642395789121829]
action: reduction
---------------------------------------
Count of function calling: 197
Count of reflection's: 58 ; function calling: 58
Count of expansion's: 29 ; function calling: 29
Count of compression's: 10 ; function calling: 10
Count of reduction's: 19 ; function calling: 38
Sum of operations calling: 116
Sum of functions calling by operations calling: 135

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.8
Itterations: 52
x: [[0.6817262991942342, 0.6775419763391607], [0.6816957850655726, 0.6775094900491785],
[0.6817228627237018, 0.6775613393841046]]
f(x): [0.564401131382724, 0.5644283692302419, 0.5644015272287835]
action: compression
---------------------------------------
Count of function calling: 173
Count of reflection's: 52 ; function calling: 52
Count of expansion's: 24 ; function calling: 24
Count of compression's: 15 ; function calling: 15
Count of reduction's: 13 ; function calling: 26
Sum of operations calling: 104
Sum of functions calling by operations calling: 117

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.7
```

```
Itterations: 67
x: [[0.6883047503707721, 0.6589695781096999], [0.6883038894931838, 0.6589257773358138],
[0.6883168271044424, 0.6589687151121547]]
f(x): [0.5702690447336478, 0.5703037508436043, 0.5702691324691148]
action: compression
---------------------------------------
Count of function calling: 226
Count of reflection's: 67 ; function calling: 67
Count of expansion's: 33 ; function calling: 33
Count of compression's: 13 ; function calling: 13
Count of reduction's: 21 ; function calling: 42
Sum of operations calling: 134
Sum of functions calling by operations calling: 155

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.6
Itterations: 71
x: [[0.6832630477481568, 0.6732409565395784], [0.6833947847280615, 0.6729046039491874],
[0.6833004309261792, 0.6731635411666906]]
f(x): [0.5641972905145367, 0.5642148030732705, 0.5641965420877514]
action: compression
---------------------------------------
Count of function calling: 237
Count of reflection's: 71 ; function calling: 71
Count of expansion's: 36 ; function calling: 36
Count of compression's: 15 ; function calling: 15
Count of reduction's: 20 ; function calling: 40
Sum of operations calling: 142
Sum of functions calling by operations calling: 162

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.5
Itterations: 68
x: [[0.690360552388207, 0.6530833023957745], [0.6903636708886784, 0.6530878592172311],
[0.6903638481837775, 0.6530881817623878]]
f(x): [0.5756034783954945, 0.5755995410333511, 0.5755992551460039]
action: reduction
---------------------------------------
Count of function calling: 230
Count of reflection's: 68 ; function calling: 68
Count of expansion's: 30 ; function calling: 30
Count of compression's: 16 ; function calling: 16
Count of reduction's: 22 ; function calling: 44
Sum of operations calling: 136
Sum of functions calling by operations calling: 158

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.4
Itterations: 61
x: [[0.6655040818567859, 0.7209904762549704], [0.6655102436906817, 0.7210202467591269],
[0.6655121779639993, 0.7210055991744504]]
f(x): [0.6145920081118773, 0.6146157876920363, 0.6145945734876812]
action: compression
---------------------------------------
Count of function calling: 208
```

```
Count of reflection's: 61 ; function calling: 61
Count of expansion's: 29 ; function calling: 29
Count of compression's: 11 ; function calling: 11
Count of reduction's: 21 ; function calling: 42
Sum of operations calling: 122
Sum of functions calling by operations calling: 143

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.3
Itterations: 57
x: [[0.6828110612412578, 0.6726199409516982], [0.6826541100464106, 0.6722381636080619],
[0.6824560039674421, 0.6730014027368585]]
f(x): [0.5646431731487614, 0.5648457090971666, 0.5647551715206517]
action: compression
----------------------------------------
Count of function calling: 189
Count of reflection's: 57 ; function calling: 57
Count of expansion's: 32 ; function calling: 32
Count of compression's: 11 ; function calling: 11
Count of reduction's: 14 ; function calling: 28
Sum of operations calling: 114
Sum of functions calling by operations calling: 128

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.2
Itterations: 33
x: [[0.24449524792902286, 0.32463798554223167], [0.2441969297486661,
0.324075812758437], [0.24371414919244827, 0.3231004550171271]]
f(x): [0.8926807496200319, 0.8926908098443108, 0.8926716574875541]
action: expansion
----------------------------------------
Count of function calling: 120
Count of reflection's: 33 ; function calling: 33
Count of expansion's: 16 ; function calling: 16
Count of compression's: 0 ; function calling: 0
Count of reduction's: 17 ; function calling: 34
Sum of operations calling: 66
Sum of functions calling by operations calling: 83

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.1
Itterations: 78
x: [[0.6824785871504635, 0.6754407658763236], [0.6823588026880134, 0.6758113932607559],
[0.682423048259296, 0.6756229759449266]]
f(x): [0.5641811936422498, 0.5641943615155401, 0.5641844974647803]
action: compression
----------------------------------------
Count of function calling: 261
Count of reflection's: 78 ; function calling: 78
Count of expansion's: 41 ; function calling: 41
Count of compression's: 14 ; function calling: 14
Count of reduction's: 23 ; function calling: 46
Sum of operations calling: 156
Sum of functions calling by operations calling: 179
----------------------------------------
```

```
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.0
Itterations: 44
x: [[0.507699737968528, 0.4702760206263495], [0.5008646436648401, 0.47311113471071575],
[0.5085338587590792, 0.4706634924107443]]
f(x): [0.7115650988913935, 0.7118936282651325, 0.711228819489985]
action: reflection
----------------------------------------
Count of function calling: 152
Count of reflection's: 44 ; function calling: 44
Count of expansion's: 25 ; function calling: 25
Count of compression's: 3 ; function calling: 3
Count of reduction's: 16 ; function calling: 32
Sum of operations calling: 88
Sum of functions calling by operations calling: 104

----------------------------------------
```

ДОДАТОК В УМОВНА ОПТИМІЗАЦІЯ. НЕВИПУКЛА ОБЛАСТЬ

Результати виконання програми для різних значень параметра ε

```
---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 10
Side length: 1.1
Itterations: 218
x:    [[0.9717113256505706,    1.0505073442516695],    [0.9717113255976265,    1.050507344197621],
[0.9717113258567658, 1.0505073444615403]]
f(x): [0.500774945656014, 0.5007749456585122, 0.5007749456443215]
action: compression
---------------------------------------
Count of function calling: 712
Count of reflection's: 218 ; function calling: 218
Count of expansion's: 111 ; function calling: 111
Count of compression's: 53 ; function calling: 53
Count of reduction's: 54 ; function calling: 108
Sum of operations calling: 436
Sum of functions calling by operations calling: 490

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 9
Side length: 1.1
Itterations: 218
x:    [[0.9717113256505706,    1.0505073442516695],    [0.9717113255976265,    1.050507344197621],
[0.9717113258567658, 1.0505073444615403]]
f(x): [0.500774945656014, 0.5007749456585122, 0.5007749456443215]
action: compression
---------------------------------------
Count of function calling: 712
Count of reflection's: 218 ; function calling: 218
Count of expansion's: 111 ; function calling: 111
Count of compression's: 53 ; function calling: 53
Count of reduction's: 54 ; function calling: 108
Sum of operations calling: 436
Sum of functions calling by operations calling: 490

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 8
Side length: 1.1
Itterations: 160
x:    [[0.9686914275103741,    1.047432363652406],    [0.9686911307946411,    1.0474320612774237],
[0.9686913288500295, 1.0474322621274756]]
f(x): [0.5009603833596683, 0.5009604025832592, 0.5009603866738054]
action: compression
---------------------------------------
Count of function calling: 519
Count of reflection's: 160 ; function calling: 160
Count of expansion's: 81 ; function calling: 81
Count of compression's: 44 ; function calling: 44
Count of reduction's: 35 ; function calling: 70
Sum of operations calling: 320
Sum of functions calling by operations calling: 355

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 7
Side length: 1.1
Itterations: 134
x:    [[0.9686899569441274,    1.0474308869602298],    [0.9686900174770491,    1.0474309594857805],
[0.9686912399642589, 1.0474321771058508]]
f(x): [0.5009605472883538, 0.5009605773063036, 0.5009604098422982]
action: compression
```

```
---------------------------------------
Count of function calling: 435
Count of reflection's: 134 ; function calling: 134
Count of expansion's: 67 ; function calling: 67
Count of compression's: 38 ; function calling: 38
Count of reduction's: 29 ; function calling: 58
Sum of operations calling: 268
Sum of functions calling by operations calling: 297

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.1
Itterations: 128
x:    [[0.9686853473821786,    1.0474262792124653],    [0.9686792583017854,    1.047420220789468],
[0.9686900174770491, 1.0474309594857805]]
f(x): [0.5009611269987349, 0.5009619814035088, 0.5009605773063036]
action: compression
---------------------------------------
Count of function calling: 417
Count of reflection's: 128 ; function calling: 128
Count of expansion's: 65 ; function calling: 65
Count of compression's: 34 ; function calling: 34
Count of reduction's: 29 ; function calling: 58
Sum of operations calling: 256
Sum of functions calling by operations calling: 285

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.1
Itterations: 72
x:    [[0.9529737480669501,    1.0319558470108434],    [0.9521369149005962,    1.0310338694780066],
[0.9517906321460908, 1.0306099017280739]]
f(x): [0.5041348026849249, 0.5040274165829385, 0.5038530638897564]
action: reduction for cycling of min
---------------------------------------
Count of function calling: 238
Count of reflection's: 72 ; function calling: 72
Count of expansion's: 37 ; function calling: 37
Count of compression's: 17 ; function calling: 17
Count of reduction's: 18 ; function calling: 36
Sum of operations calling: 144
Sum of functions calling by operations calling: 162

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 4
Side length: 1.1
Itterations: 72
x:    [[0.9529737480669501,    1.0319558470108434],    [0.9521369149005962,    1.0310338694780066],
[0.9517906321460908, 1.0306099017280739]]
f(x): [0.5041348026849249, 0.5040274165829385, 0.5038530638897564]
action: reduction for cycling of min
---------------------------------------
Count of function calling: 238
Count of reflection's: 72 ; function calling: 72
Count of expansion's: 37 ; function calling: 37
Count of compression's: 17 ; function calling: 17
Count of reduction's: 18 ; function calling: 36
Sum of operations calling: 144
Sum of functions calling by operations calling: 162

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 3
Side length: 1.1
```

```
Itterations: 43
x:    [[1.0673137661519119,    1.1581831612913827],    [1.0676632255062408,    1.15842683916539],
[1.0674264840389174, 1.158206836751998]]
f(x): [0.5432616423335246, 0.5430354657678, 0.5430329794114134]
action: reduction
---------------------------------------
Count of function calling: 147
Count of reflection's: 43 ; function calling: 43
Count of expansion's: 20 ; function calling: 20
Count of compression's: 9 ; function calling: 9
Count of reduction's: 14 ; function calling: 28
Sum of operations calling: 86
Sum of functions calling by operations calling: 100

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 2
Side length: 1.1
Itterations: 43
x:    [[1.0673137661519119,    1.1581831612913827],    [1.0676632255062408,    1.15842683916539],
[1.0674264840389174, 1.158206836751998]]
f(x): [0.5432616423335246, 0.5430354657678, 0.5430329794114134]
action: reduction
---------------------------------------
Count of function calling: 147
Count of reflection's: 43 ; function calling: 43
Count of expansion's: 20 ; function calling: 20
Count of compression's: 9 ; function calling: 9
Count of reduction's: 14 ; function calling: 28
Sum of operations calling: 86
Sum of functions calling by operations calling: 100

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 1
Side length: 1.1
Itterations: 10
x:  [[-0.271811926719967,  0.8846302955589415],  [-0.6041720666651462,  0.7513914251958942],  [-
0.37078856820858713, 0.7508074318934483]]
f(x): [1.9676974158453318, 2.1393914879790845, 1.9499942993792472]
action: compression
---------------------------------------
Count of function calling: 37
Count of reflection's: 10 ; function calling: 10
Count of expansion's: 5 ; function calling: 5
Count of compression's: 2 ; function calling: 2
Count of reduction's: 3 ; function calling: 6
Sum of operations calling: 20
Sum of functions calling by operations calling: 23
```

Результати виконання програми для різних значень параметра t

[inf, inf, inf] – **означає, що метод не зміг зайти в допустиму область.**

```
----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 2.0
Itterations: 601
x: [[0.7318516525781364, 0.5176380902050414], [-2.395043642289314e+153, 6.791523347414883e+153],
[-2.6681017786255956e+153, 7.565822686009335e+153]]
f(x): [inf, inf, inf]
action: reflection
----------------------------------------
Count of function calling: 1807
Count of reflection's: 601 ; function calling: 601
Count of expansion's: 601 ; function calling: 601
Count of compression's: 0 ; function calling: 0
Count of reduction's: 0 ; function calling: 0
Sum of operations calling: 1202
Sum of functions calling by operations calling: 1202

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.9
Itterations: 601
x: [[0.6352590699492295, 0.4917561856947893], [-2.275291460174841e+153, 6.451947180044125e+153],
[-2.53469668969431e+153, 7.187531551708855e+153]]
f(x): [inf, inf, inf]
action: reflection
----------------------------------------
Count of function calling: 1807
Count of reflection's: 601 ; function calling: 601
Count of expansion's: 601 ; function calling: 601
Count of compression's: 0 ; function calling: 0
Count of reduction's: 0 ; function calling: 0
Sum of operations calling: 1202
Sum of functions calling by operations calling: 1202

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.8
Itterations: 601
x: [[0.5386664873203229, 0.4658742811845373], [-2.155539278060369e+153, 6.112371012673407e+153],
[-2.4012916007630277e+153, 6.809240417408412e+153]]
f(x): [inf, inf, inf]
action: reflection
----------------------------------------
Count of function calling: 1807
Count of reflection's: 601 ; function calling: 601
Count of expansion's: 601 ; function calling: 601
Count of compression's: 0 ; function calling: 0
Count of reduction's: 0 ; function calling: 0
Sum of operations calling: 1202
Sum of functions calling by operations calling: 1202

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.7
Itterations: 601
x: [[0.44207390469141594, 0.4399923766742852], [-2.0357870959459e+153, 5.772794845302632e+153], [-
2.2678865118317423e+153, 6.4309492831079046e+153]]
f(x): [inf, inf, inf]
action: reflection
----------------------------------------
```

```
Count of function calling: 1807
Count of reflection's: 601 ; function calling: 601
Count of expansion's: 601 ; function calling: 601
Count of compression's: 0 ; function calling: 0
Count of reduction's: 0 ; function calling: 0
Sum of operations calling: 1202
Sum of functions calling by operations calling: 1202

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.6
Itterations: 601
x:         [[0.34548132206250926,         0.4141104721640332],         [-1.9160349138314428e+153,
5.4332186779318916e+153], [-2.134481422900467e+153, 6.0526581488074614e+153]]
f(x): [inf, inf, inf]
action: reflection
----------------------------------------
Count of function calling: 1807
Count of reflection's: 601 ; function calling: 601
Count of expansion's: 601 ; function calling: 601
Count of compression's: 0 ; function calling: 0
Count of reduction's: 0 ; function calling: 0
Sum of operations calling: 1202
Sum of functions calling by operations calling: 1202

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.5
Itterations: 601
x:         [[0.24888873943360235,         0.38822856765378105],         [-1.796282731716984e+153,
5.0936425105611214e+153], [-2.001076333969193e+153, 5.6743670145069684e+153]]
f(x): [inf, inf, inf]
action: reflection
----------------------------------------
Count of function calling: 1807
Count of reflection's: 601 ; function calling: 601
Count of expansion's: 601 ; function calling: 601
Count of compression's: 0 ; function calling: 0
Count of reduction's: 0 ; function calling: 0
Sum of operations calling: 1202
Sum of functions calling by operations calling: 1202

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.4
Itterations: 67
x:    [[0.7448670180708767,    0.8605649618516186],    [0.7448670584745881,    0.8605648223465404],
[0.7448671515448322, 0.8605649273088994]]
f(x): [0.667863351249428, 0.6678629845777881, 0.6678629678161867]
action: reduction
----------------------------------------
Count of function calling: 231
Count of reflection's: 67 ; function calling: 67
Count of expansion's: 31 ; function calling: 31
Count of compression's: 10 ; function calling: 10
Count of reduction's: 26 ; function calling: 52
Sum of operations calling: 134
Sum of functions calling by operations calling: 160

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.3
Itterations: 60
```

```
x:    [[0.6034170735024573,    0.7751774690537185],    [0.6034302891618096,    0.7751757217279431],
[0.6034764363167161, 0.7751931694374083]]
f(x): [0.8200782452405484, 0.820050194000529, 0.819988909583212]
action: reduction
---------------------------------------
Count of function calling: 202
Count of reflection's: 60 ; function calling: 60
Count of expansion's: 29 ; function calling: 29
Count of compression's: 13 ; function calling: 13
Count of reduction's: 18 ; function calling: 36
Sum of operations calling: 120
Sum of functions calling by operations calling: 138

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.2
Itterations: 161
x:    [[0.7234857922331697,    0.8462572545950858],    [0.7235666560694907,    0.8463103785783204],
[0.7236936639077018, 0.8463937696762932]]
f(x): [0.6903932729558097, 0.6903075467628059, 0.6901727925387828]
action: reduction for cycling of min
---------------------------------------
Count of function calling: 525
Count of reflection's: 161 ; function calling: 161
Count of expansion's: 84 ; function calling: 84
Count of compression's: 39 ; function calling: 39
Count of reduction's: 38 ; function calling: 76
Sum of operations calling: 322
Sum of functions calling by operations calling: 360

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.1
Itterations: 72
x:    [[0.9529737480669501,    1.0319558470108434],    [0.9521369149005962,    1.0310338694780066],
[0.9517906321460908, 1.0306099017280739]]
f(x): [0.5041348026849249, 0.5040274165829385, 0.5038530638897564]
action: reduction for cycling of min
---------------------------------------
Count of function calling: 238
Count of reflection's: 72 ; function calling: 72
Count of expansion's: 37 ; function calling: 37
Count of compression's: 17 ; function calling: 17
Count of reduction's: 18 ; function calling: 36
Sum of operations calling: 144
Sum of functions calling by operations calling: 162

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.0
Itterations: 105
x:    [[1.041589155266959,    1.1269768376038232],    [1.0415886796840148,    1.1269762369828846],
[1.0415884224994436, 1.126975941432658]]
f(x): [0.5226888415974384, 0.5226883985103341, 0.5226882463555584]
action: reduction
---------------------------------------
Count of function calling: 343
Count of reflection's: 105 ; function calling: 105
Count of expansion's: 51 ; function calling: 51
Count of compression's: 30 ; function calling: 30
Count of reduction's: 24 ; function calling: 48
Sum of operations calling: 210
Sum of functions calling by operations calling: 234
```

ДОДАТОК Г УМОВНА ОПТИМІЗАЦІЯ. ЛІНІЙНЕ ОБМЕЖЕННЯ

Даний додаток містить інформацію про останні ітерації для певної модифікації параметра коефіцієнта в наступному форматі:

```
---------------------------------------
Result: візуальне розділення
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5 значення відповідних коефіцієнтів, з якими був проведений ряд ітерацій
Accuracy: 5 точність перевірки критерію зупинки, відємеа степінь 1: 10^-5
Side length: 1.5 довжина сторони багатокутника
itterations: 57 - кількість ітерацій, співпадає з кількістю обчислень функції на етапі перевірки критерію зупинки, а саме з кількістю обчислень функції в центрі ваги багатокутника (усі інші задіяні значення функції вже відомі на момент підрахування критерію)
x: [[1.0032325114580896, 0.8856881978791556], [1.0032868582498309, 0.8857400749332238], [1.003114106854531, 0.8855668261177909]] x – координати вершин багатокутника на останній ітерації
f(x): [0.6096899312083286, 0.6096967266458907, 0.6096967968329776] значення функції в відповідних точках, порядок співпадає з минулим пунктом
action: reduction остання операція яка була проведена
---------------------------------------
Count of function calling: 203 загальна кількість викликів функції
Кількість звернень до операції, кількість обчислень функції цим типом операції
Count of reflection's: 57 ; function calling: 57 для віддзеркалення
Count of expansion's: 28 ; function calling: 28 для розтягнення
Count of compression's: 1 ; function calling: 1 для стискання
Count of reduction's: 28 ; function calling: 56 для редукції
Sum of operations calling: 114 загальна кількість викликів операцій
Sum of functions calling by operations calling: 142 загальна кількість обчислень функцій, здійснена операціями
Last point is: [0.2682913658319216, 0.231705929900549] остання точка, досягнута одномірним пошуком
Last value is: 0.8606965394549524 значення в останній точці
Count of function calling by Sven method: 4 кількість кроків методом Свена
Count of function calling by [НАЗВА МЕТОДУ]: 49 кількість кроків методом [НАЗВА МЕТОДУ]


---------------------------------------
візуальне розділення
```

Результати виконання програми для різних значень параметра ε і початкових $\alpha, \beta, \gamma, \sigma, t$

```
---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 10
Side length: 1.1
Itterations: 137
x: [[0.26830795562554094, 0.23168934535233027], [0.26830925515508003, 0.2316880478871847], [0.26830458544823277, 0.23169271028423785]]
f(x): [0.8606965423742026, 0.8606965425016357, 0.8606965422652137]
action: expansion
---------------------------------------
Count of function calling: 510
Count of reflection's: 137 ; function calling: 137
Count of expansion's: 72 ; function calling: 72
Count of compression's: 23 ; function calling: 23
Count of reduction's: 42 ; function calling: 84
Sum of operations calling: 274
Sum of functions calling by operations calling: 316
Last point is: [0.2682913658319216, 0.231705929900549]
Last value is: 0.8606965394549524
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 49

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 9
```

```
Side length: 1.1
Itterations: 130
x: [[0.26833798362547023, 0.23165936430146622], [0.26847524522326915, 0.23152231855848512],
[0.26827834553892366, 0.23171890953002267]]
f(x): [0.860696559381699, 0.86069700534149, 0.8606965538373964]
action: reflection
---------------------------------------
Count of function calling: 481
Count of reflection's: 130 ; function calling: 130
Count of expansion's: 70 ; function calling: 70
Count of compression's: 21 ; function calling: 21
Count of reduction's: 39 ; function calling: 78
Sum of operations calling: 260
Sum of functions calling by operations calling: 299
Last point is: [0.26829134328284665, 0.23170591178609967]
Last value is: 0.8606965511212493
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 44

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 8
Side length: 1.1
Itterations: 86
x: [[0.27003744479888336, 0.22996253181915202], [0.2700382099625712, 0.22996176248909336],
[0.27003775513429634, 0.22996222796171784]]
f(x): [0.8607447048884046, 0.8607447491234596, 0.8607447202605631]
action: compression
---------------------------------------
Count of function calling: 334
Count of reflection's: 86 ; function calling: 86
Count of expansion's: 42 ; function calling: 42
Count of compression's: 15 ; function calling: 15
Count of reduction's: 29 ; function calling: 58
Sum of operations calling: 172
Sum of functions calling by operations calling: 201
Last point is: [0.26829267197891643, 0.23170730463911893]
Last value is: 0.8606957703130191
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 39

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 7
Side length: 1.1
Itterations: 83
x: [[0.2700380949601012, 0.22996184786673327], [0.2700382099625712, 0.22996176248909336],
[0.27003744479888336, 0.22996253181915202]]
f(x): [0.8607447519792932, 0.8607447491234596, 0.8607447048884046]
action: reduction
---------------------------------------
Count of function calling: 320
Count of reflection's: 83 ; function calling: 83
Count of expansion's: 41 ; function calling: 41
Count of compression's: 13 ; function calling: 13
Count of reduction's: 29 ; function calling: 58
Sum of operations calling: 166
Sum of functions calling by operations calling: 195
Last point is: [0.26829268907688175, 0.23170728754115363]
Last value is: 0.8606957703130241
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 34

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.1
Itterations: 61
```

```
x: [[0.27000658524158433, 0.2299834842894312], [0.27002029837840225, 0.22996928170796205],
[0.27003851487949676, 0.22995863178270465]]
f(x): [0.8607460978344255, 0.8607470122154749, 0.8607456542022001]
action: compression
---------------------------------------
Count of function calling: 242
Count of reflection's: 61 ; function calling: 61
Count of expansion's: 31 ; function calling: 31
Count of compression's: 9 ; function calling: 9
Count of reduction's: 21 ; function calling: 42
Sum of operations calling: 122
Sum of functions calling by operations calling: 143
Last point is: [0.2682913494362415, 0.23170579722595994]
Last value is: 0.8606965822230171
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 30

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.1
Itterations: 42
x: [[0.26795430102674833, 0.2315451421560734], [0.2680380643664487, 0.23142563588130222],
[0.26785047181173144, 0.2316128117165849]]
f(x): [0.8608395032739 55, 0.8608496142812698, 0.8608502559153992]
action: reduction
---------------------------------------
Count of function calling: 175
Count of reflection's: 42 ; function calling: 42
Count of expansion's: 21 ; function calling: 21
Count of compression's: 5 ; function calling: 5
Count of reduction's: 16 ; function calling: 32
Sum of operations calling: 84
Sum of functions calling by operations calling: 100
Last point is: [0.26804781526020877, 0.23145162792261292]
Last value is: 0.8608393606777439
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 25

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 4
Side length: 1.1
Itterations: 38
x: [[0.2642878318714637, 0.23393472351163574], [0.2716156660401069, 0.22662316744590597],
[0.26834366058306175, 0.2312913813041 5527]]
f(x): [0.8613635148534549, 0.8614814250710946, 0.8608013059746645]
action: compression
---------------------------------------
Count of function calling: 154
Count of reflection's: 38 ; function calling: 38
Count of expansion's: 21 ; function calling: 21
Count of compression's: 5 ; function calling: 5
Count of reduction's: 12 ; function calling: 24
Sum of operations calling: 76
Sum of functions calling by operations calling: 88
Last point is: [0.26812371935230456, 0.23151132253491247]
Last value is: 0.860800464524323
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 20

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 3
Side length: 1.1
Itterations: 38
x: [[0.2642878318714637, 0.23393472351163574], [0.2716156660401069, 0.22662316744590597],
[0.26834366058306175, 0.2312913813041 5527]]
```

```
f(x): [0.8613635148534549, 0.8614814250710946, 0.8608013059746645]
action: compression
---------------------------------------
Count of function calling: 149
Count of reflection's: 38 ; function calling: 38
Count of expansion's: 21 ; function calling: 21
Count of compression's: 5 ; function calling: 5
Count of reduction's: 12 ; function calling: 24
Sum of operations calling: 76
Sum of functions calling by operations calling: 88
Last point is: [0.26827034683947604, 0.231364695047741]
Last value is: 0.8608008527421275
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 15

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 2
Side length: 1.1
Itterations: 9
x: [[0.16164065637946787, 0.14077837225402226], [0.17835152033485752, 0.1654540869926972],
[0.218956936640103, 0.16623524938151632]]
f(x): [0.9170338550672689, 0.9070061263814998, 0.8936659670972114]
action: reduction for cycling of min
---------------------------------------
Count of function calling: 50
Count of reflection's: 9 ; function calling: 9
Count of expansion's: 4 ; function calling: 4
Count of compression's: 0 ; function calling: 0
Count of reduction's: 5 ; function calling: 10
Sum of operations calling: 18
Sum of functions calling by operations calling: 23
Last point is: [0.2107530041407339, 0.17443918188088542]
Last value is: 0.8930606750451637
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 10

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 1
Side length: 1.1
Itterations: 3
x: [[-0.25136197092713397, -0.14030641395441723], [-0.13748159108202485, 0.2847009496127728],
[0.17364539264005607, -0.026426034109308105]]
f(x): [1.140047243319074, 1.3243575216482033, 1.0201685244112375]
action: reflection
---------------------------------------
Count of function calling: 24
Count of reflection's: 3 ; function calling: 3
Count of expansion's: 2 ; function calling: 2
Count of compression's: 0 ; function calling: 0
Count of reduction's: 1 ; function calling: 2
Sum of operations calling: 6
Sum of functions calling by operations calling: 7
Last point is: [0.080940294327571841, 0.066279064203176127]
Last value is: 0.959285345159
Count of function calling by Sven method: 5
Count of function calling by Golden section method: 5
```

Результати виконання програми для різних значень параметра t

```
----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 2.0
Itterations: 76
x: [[0.2592541235521877, 0.21171240171594127], [0.25960566297302, 0.21151539080308243],
[0.25904960659476706, 0.211831578987109]]
f(x): [0.8693953471115873, 0.8693968348056579, 0.8693939395398077]
action: expansion
----------------------------------------
Count of function calling: 287
Count of reflection's: 76 ; function calling: 76
Count of expansion's: 39 ; function calling: 39
Count of compression's: 16 ; function calling: 16
Count of reduction's: 21 ; function calling: 42
Sum of operations calling: 152
Sum of functions calling by operations calling: 173
Last point is: [0.2540882767220329, 0.2167929088598432]
Last value is: 0.8690097553523572
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 30

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.9
Itterations: 75
x: [[0.2692935570585034, 0.2306751512238949], [0.269294431416972, 0.2306722367370365],
[0.26927514204782055, 0.23068945692680698]]
f(x): [0.8607213390567405, 0.8607219854563173, 0.8607219861677546]
action: reduction
----------------------------------------
Count of function calling: 286
Count of reflection's: 75 ; function calling: 75
Count of expansion's: 37 ; function calling: 37
Count of compression's: 15 ; function calling: 15
Count of reduction's: 23 ; function calling: 46
Sum of operations calling: 150
Sum of functions calling by operations calling: 173
Last point is: [0.2682773186422904, 0.2316913896401079]
Last value is: 0.8607047411076992
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 30

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.8
Itterations: 64
x: [[-0.06375502135439451, -0.4816472625497169], [-0.06372994915165256, -0.4816290560600573
0.06365385784966474, -0.4815723312718705]]
f(x): [1.3024751703109205, 1.3024756276729594, 1.3024756269175364]
action: reduction
----------------------------------------
Count of function calling: 263
Count of reflection's: 64 ; function calling: 64
Count of expansion's: 32 ; function calling: 32
Count of compression's: 1 ; function calling: 1
Count of reduction's: 31 ; function calling: 62
Sum of operations calling: 128
Sum of functions calling by operations calling: 159
Last point is: [-0.24165977564276911, -0.3037425082613423]
Last value is: 1.12119783517
Count of function calling by Sven method: 6
Count of function calling by Golden section method: 30
```

```
---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.7
Itterations: 142
x: [[0.2676956055207413, 0.2322987932505764], [0.2676488460548241, 0.23234166146184945],
[0.26766908171751813, 0.23232677655334144]]
f(x): [0.8607030492432133, 0.8607050552051103, 0.8607031628901853]
action: compression
---------------------------------------
Count of function calling: 494
Count of reflection's: 142 ; function calling: 142
Count of expansion's: 78 ; function calling: 78
Count of compression's: 34 ; function calling: 34
Count of reduction's: 30 ; function calling: 60
Sum of operations calling: 284
Sum of functions calling by operations calling: 314
Last point is: [0.26829005905459946, 0.2317043397167183]
Last value is: 0.8606973705880546
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 30

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.6
Itterations: 105
x: [[0.26920664553460893, 0.2307865491269961], [0.2692075718386384, 0.2307836024747718],
[0.26920902184229206, 0.23078783823017837]]
f(x): [0.8607112419917652, 0.8607118780380189, 0.8607102077969337]
action: compression
---------------------------------------
Count of function calling: 383
Count of reflection's: 105 ; function calling: 105
Count of expansion's: 55 ; function calling: 55
Count of compression's: 20 ; function calling: 20
Count of reduction's: 30 ; function calling: 60
Sum of operations calling: 210
Sum of functions calling by operations calling: 240
Last point is: [0.2682910125992788, 0.23170584747319165]
Last value is: 0.8606966644453784
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 30

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.5
Itterations: 174
x: [[0.2550629132939354, 0.2449328720334352], [0.2550633767799903, 0.2449357815310474],
[0.25506358752756575, 0.24493582695649743]]
f(x): [0.8634961125477423, 0.8634956520171705, 0.8634955431007562]
action: reduction
---------------------------------------
Count of function calling: 616
Count of reflection's: 174 ; function calling: 174
Count of expansion's: 99 ; function calling: 99
Count of compression's: 19 ; function calling: 19
Count of reduction's: 56 ; function calling: 112
Sum of operations calling: 348
Sum of functions calling by operations calling: 404
Last point is: [0.2682925187675763, 0.23170689571648687]
Last value is: 0.8606959315886827
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 30

---------------------------------------
Result:
```

```
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.4
Itterations: 68
x: [[0.26912373750229823, 0.23087463621048673], [0.2691280626793548, 0.23087058469892285],
[0.26912913446751097, 0.2308699412318079]]
f(x): [0.8607073539945619, 0.8607073878694487, 0.8607072881942099]
action: compression
---------------------------------------
Count of function calling: 261
Count of reflection's: 68 ; function calling: 68
Count of expansion's: 33 ; function calling: 33
Count of compression's: 16 ; function calling: 16
Count of reduction's: 19 ; function calling: 38
Sum of operations calling: 136
Sum of functions calling by operations calling: 155
Last point is: [0.26829210111957624, 0.23170697457974263]
Last value is: 0.8606960287855615
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 30

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.3
Itterations: 46
x: [[0.2683373003523267, 0.23165423779730787], [0.26853398569658266, 0.23146455616573736],
[0.2682451638917328, 0.23175360993282498]]
f(x): [0.8606982294916913, 0.8606971235168432, 0.8606961507848512]
action: reduction
---------------------------------------
Count of function calling: 189
Count of reflection's: 46 ; function calling: 46
Count of expansion's: 24 ; function calling: 24
Count of compression's: 9 ; function calling: 9
Count of reduction's: 13 ; function calling: 26
Sum of operations calling: 92
Sum of functions calling by operations calling: 105
Last point is: [0.2682921423488777, 0.23170663147568002]
Last value is: 0.8606961153927508
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 30

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.2
Itterations: 76
x: [[0.22039896270253495, 0.2795997339415809], [0.2203992274077263, 0.27960031456798773],
[0.22039920315666456, 0.27960018523939417]]
f(x): [0.8954126073139651, 0.8954125938470059, 0.895412563689408]
action: reduction
---------------------------------------
Count of function calling: 296
Count of reflection's: 76 ; function calling: 76
Count of expansion's: 37 ; function calling: 37
Count of compression's: 9 ; function calling: 9
Count of reduction's: 30 ; function calling: 60
Sum of operations calling: 152
Sum of functions calling by operations calling: 182
Last point is: [0.26829225997141004, 0.23170712842464872]
Last value is: 0.8606959390733054
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 30

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
```

```
Side length: 1.1
Itterations: 61
x: [[0.27000658524158433, 0.2299834842894312], [0.27002029837840225, 0.22996928170796205],
[0.27003851487949676, 0.22995863178270465]]
f(x): [0.8607460978344255, 0.8607470122154749, 0.8607456542022001]
action: compression
---------------------------------------
Count of function calling: 242
Count of reflection's: 61 ; function calling: 61
Count of expansion's: 31 ; function calling: 31
Count of compression's: 9 ; function calling: 9
Count of reduction's: 21 ; function calling: 42
Sum of operations calling: 122
Sum of functions calling by operations calling: 143
Last point is: [0.2682913494362415, 0.23170579722595994]
Last value is: 0.8606965822230171
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 30

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.0
Itterations: 54
x: [[0.2675602670520142, 0.23156078643319514], [0.26754228451098816, 0.23169968671331106],
[0.26888369662221606, 0.2305981936381196]]
f(x): [0.8609493762452566, 0.860915540787692, 0.8608558342211128]
action: reduction for cycling of min
---------------------------------------
Count of function calling: 215
Count of reflection's: 54 ; function calling: 54
Count of expansion's: 28 ; function calling: 28
Count of compression's: 11 ; function calling: 11
Count of reduction's: 15 ; function calling: 30
Sum of operations calling: 108
Sum of functions calling by operations calling: 123
Last point is: [0.2680400731084564, 0.2314418171518793]
Last value is: 0.860844395724178
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 30
```

ДОДАТОК Д УМОВНА ОПТИМІЗАЦІЯ. ЛІНІЙНЕ ОБМЕЖЕННЯ МОДИФІКАЦІЇ λ

Даний додаток містить інформацію про останні ітерації для певної модифікації параметра коефіцієнта в наступному форматі:

```
---------------------------------------
Result: візуальне розділення
Coefficients: alpha 1.0 beta: 0.5 gamma: 2.0 sigma: 0.5 значення відповідних коефіцієнтів, з якими був проведений ряд ітерацій
Accuracy: 5 точність перевірки критерію зупинки, відємеа степінь 1: 10^-5
Side length: 1.5 довжина сторони багатокутника
itterations: 57 - кількість ітерацій, співпадає з кількістю обчислень функції на етапі перевірки критерію зупинки, а саме з кількістю обчислень функції в центрі ваги багатокутника (усі інші задіяні значення функції вже відомі на момент підрахування критерію)
x: [[1.0032325114580896, 0.8856881978791556], [1.0032868582498309, 0.8857400749332238],
[1.003114106854531, 0.8855668261177909]] x – координати вершин багатокутника на останній ітерації
f(x): [0.6096899312083286, 0.6096967266458907, 0.6096967968329776] значення функції в відповідних точках, порядок співпадає з минулим пунктом
action: reduction остання операція яка була проведена
---------------------------------------
Count of function calling: 203 загальна кількість викликів функції
Кількість звернень до операції, кількість обчислень функції цим типом операції
Count of reflection's: 57 ; function calling: 57 для віддзеркалення
Count of expansion's: 28 ; function calling: 28 для розтягнення
Count of compression's: 1 ; function calling: 1 для стискання
Count of reduction's: 28 ; function calling: 56 для редукції
Sum of operations calling: 114 загальна кількість викликів операцій
Sum of functions calling by operations calling: 142 загальна кількість обчислень функцій, здійснена операціями
Last point is: [0.2682913658319216, 0.231705929900549] остання точка, досягнута одномірним пошуком
Last value is: 0.8606965394549524 значення в останній точці
Count of function calling by Sven method: 4 кількість кроків методом Свена
Count of function calling by [НАЗВА МЕТОДУ]: 49 кількість кроків методом [НАЗВА МЕТОДУ]
Lambda for Sven: 0.25047352077454377 значення числа λ

---------------------------------------
візуальне розділення
```

Метод Золотого січення. Результати виконання програми для різних значень параметра ε і λ

```
---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 10
Side length: 1.1
Itterations: 137
x: [[0.26830795562554094, 0.23168934535233027], [0.26830925515508003, 0.2316880478871847],
[0.26830458544823277, 0.23169271028423785]]
f(x): [0.8606965423742026, 0.8606965425016357, 0.8606965422652137]
action: expansion
---------------------------------------
Count of function calling: 512
Count of reflection's: 137 ; function calling: 137
Count of expansion's: 72 ; function calling: 72
Count of compression's: 23 ; function calling: 23
Count of reduction's: 42 ; function calling: 84
Sum of operations calling: 274
Sum of functions calling by operations calling: 316
Last point is: [0.26829136581178836, 0.2317059299206823]
Last value is: 0.8606965394549523
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 51
Lambda for Sven: 0.25066797020301956

---------------------------------------
```

```
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 9
Side length: 1.1
Itterations: 130
x: [[0.26833798362547023, 0.23165936430146622], [0.26847524522326915, 0.23152231855848512],
[0.26827834553892366, 0.23171890953002267]]
f(x): [0.8606965593816 99, 0.86069700534149, 0.8606965538373964]
action: reflection
---------------------------------------
Count of function calling: 483
Count of reflection's: 130 ; function calling: 130
Count of expansion's: 70 ; function calling: 70
Count of compression's: 21 ; function calling: 21
Count of reduction's: 39 ; function calling: 78
Sum of operations calling: 260
Sum of functions calling by operations calling: 299
Last point is: [0.26829134299263524, 0.23170591207631108]
Last value is: 0.8606965511212493
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 46
Lambda for Sven: 0.2506660365096206

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 8
Side length: 1.1
Itterations: 86
x: [[0.27003744479888336, 0.22996253181915202], [0.2700382099625712, 0.22996176248909336],
[0.27003775513429634, 0.22996222796171784]]
f(x): [0.8607447048884046, 0.8607447491234596, 0.8607447202605631]
action: compression
---------------------------------------
Count of function calling: 336
Count of reflection's: 86 ; function calling: 86
Count of expansion's: 42 ; function calling: 42
Count of compression's: 15 ; function calling: 15
Count of reduction's: 29 ; function calling: 58
Sum of operations calling: 172
Sum of functions calling by operations calling: 201
Last point is: [0.2682926715693519, 0.23170730504868348]
Last value is: 0.8606957703130191
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 41
Lambda for Sven: 0.2508017021814091

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 7
Side length: 1.1
Itterations: 83
x: [[0.2700380949601012, 0.22996184786673327], [0.2700382099625712, 0.22996176248909336],
[0.27003744479888336, 0.22996253181915202]]
f(x): [0.8607447519792932, 0.8607447491234596, 0.8607447048884046]
action: reduction
---------------------------------------
Count of function calling: 322
Count of reflection's: 83 ; function calling: 83
Count of expansion's: 41 ; function calling: 41
Count of compression's: 13 ; function calling: 13
Count of reduction's: 29 ; function calling: 58
Sum of operations calling: 166
Sum of functions calling by operations calling: 195
Last point is: [0.2682926901408861, 0.23170728647714928]
Last value is: 0.8606957703130248
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 36
Lambda for Sven: 0.2508017021814091
```

```
---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.1
Itterations: 61
x: [[0.27000658524158433, 0.2299834842894312], [0.27002029837840225, 0.22996928170796205],
[0.27003851487949676, 0.22995863178270465]]
f(x): [0.8607460978344255, 0.8607470122154749, 0.8607456542022001]
action: compression
---------------------------------------
Count of function calling: 243
Count of reflection's: 61 ; function calling: 61
Count of expansion's: 31 ; function calling: 31
Count of compression's: 9 ; function calling: 9
Count of reduction's: 21 ; function calling: 42
Sum of operations calling: 122
Sum of functions calling by operations calling: 143
Last point is: [0.26829115126462705, 0.23170599539757436]
Last value is: 0.8606965822232765
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 31
Lambda for Sven: 0.2508004902803199

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.1
Itterations: 42
x: [[0.26795430102674833, 0.2315451421560734], [0.2680380643664487, 0.23142563588130222],
[0.26785047181173144, 0.2316128117165849]]
f(x): [0.8608395032739 55, 0.8608496142812698, 0.8608502559153992]
action: reduction
---------------------------------------
Count of function calling: 177
Count of reflection's: 42 ; function calling: 42
Count of expansion's: 21 ; function calling: 21
Count of compression's: 5 ; function calling: 5
Count of reduction's: 16 ; function calling: 32
Sum of operations calling: 84
Sum of functions calling by operations calling: 100
Last point is: [0.26804936477118924, 0.23145007841163245]
Last value is: 0.8608393606817846
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 27
Lambda for Sven: 0.2504123202787849

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 4
Side length: 1.1
Itterations: 38
x: [[0.2642878318714637, 0.23393472351163574], [0.2716156660401069, 0.22662316744590597],
[0.26834366058306175, 0.2312913813041 5527]]
f(x): [0.8613635148534549, 0.8614814250710946, 0.8608013059746645]
action: compression
---------------------------------------
Count of function calling: 156
Count of reflection's: 38 ; function calling: 38
Count of expansion's: 21 ; function calling: 21
Count of compression's: 5 ; function calling: 5
Count of reduction's: 12 ; function calling: 24
Sum of operations calling: 76
Sum of functions calling by operations calling: 88
Last point is: [0.268126887025008, 0.23150815486220913]
Last value is: 0.8608004656084486
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 22
Lambda for Sven: 0.2505035161836073
```

```
----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 3
Side length: 1.1
Itterations: 38
x: [[0.2642878318714637, 0.23393472351163574], [0.2716156660401069, 0.22662316744590597],
[0.26834366058306175, 0.2312913813041552 7]]
f(x): [0.8613635148534549, 0.8614814250710946, 0.8608013059746645]
action: compression
----------------------------------------
Count of function calling: 151
Count of reflection's: 38 ; function calling: 38
Count of expansion's: 21 ; function calling: 21
Count of compression's: 5 ; function calling: 5
Count of reduction's: 12 ; function calling: 24
Sum of operations calling: 76
Sum of functions calling by operations calling: 88
Last point is: [0.2679329988603948, 0.2317020430268223]
Last value is: 0.860800993494702
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 17
Lambda for Sven: 0.2505035161836073

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 2
Side length: 1.1
Itterations: 9
x: [[0.16164065637946787, 0.14077837225402226], [0.17835152033485752, 0.1654540869926972],
[0.218956936640103, 0.16623524938151632]]
f(x): [0.9170338550672689, 0.9070061263814998, 0.8936659670972114]
action: reduction for cycling of min
----------------------------------------
Count of function calling: 52
Count of reflection's: 9 ; function calling: 9
Count of expansion's: 4 ; function calling: 4
Count of compression's: 0 ; function calling: 0
Count of reduction's: 5 ; function calling: 10
Sum of operations calling: 18
Sum of functions calling by operations calling: 23
Last point is: [0.2112849064576078, 0.1739072795640115]
Last value is: 0.8930412334454183
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 12
Lambda for Sven: 0.1943917413880448

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 1
Side length: 1.1
Itterations: 3
x: [[-0.25136197092713397, -0.14030641395441723], [-0.13748159108202485, 0.2847009496127728],
[0.17364539264005607, -0.02642603410930810 5]]
f(x): [1.140047243319074, 1.3243575216482033, 1.0201685244112375]
action: reflection
----------------------------------------
Count of function calling: 25
Count of reflection's: 3 ; function calling: 3
Count of expansion's: 2 ; function calling: 2
Count of compression's: 0 ; function calling: 0
Count of reduction's: 1 ; function calling: 2
Sum of operations calling: 6
Sum of functions calling by operations calling: 7
Last point is: [0.085686781171316506, 0.061532577359431455]
Last value is: 0.957861255117
Count of function calling by Sven method: 5
Count of function calling by Golden section method: 6
```

```
Lambda for Sven: 0.12419955246269115
```

Метод Золотого січення. Результати виконання програми для різних значень параметра t і λ

```
----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 2.0
Itterations: 76
x: [[0.2592541235521877, 0.21171240171594127], [0.25960566297302, 0.21151539080308243],
[0.25904960659476706, 0.211831578987109]]
f(x): [0.8693953471115873, 0.8693968348056579, 0.8693939395398077]
action: expansion
----------------------------------------
Count of function calling: 288
Count of reflection's: 76 ; function calling: 76
Count of expansion's: 39 ; function calling: 39
Count of compression's: 16 ; function calling: 16
Count of reduction's: 21 ; function calling: 42
Sum of operations calling: 152
Sum of functions calling by operations calling: 173
Last point is: [0.25408850390351906, 0.21679268167835702]
Last value is: 0.8690097553524078
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 31
Lambda for Sven: 0.23662133941497687

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.9
Itterations: 75
x: [[0.2692935570585034, 0.2306751512238949], [0.269294431416972, 0.2306722367370365],
[0.26927514204782055, 0.23068945692680698]]
f(x): [0.8607213390567405, 0.8607219854563173, 0.8607219861677546]
action: reduction
----------------------------------------
Count of function calling: 287
Count of reflection's: 75 ; function calling: 75
Count of expansion's: 37 ; function calling: 37
Count of compression's: 15 ; function calling: 15
Count of reduction's: 23 ; function calling: 46
Sum of operations calling: 150
Sum of functions calling by operations calling: 173
Last point is: [0.2682775664695181, 0.23169114181288009]
Last value is: 0.8607047411078896
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 31
Lambda for Sven: 0.2507289824346082

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.8
Itterations: 64
x: [[-0.06375502135439451, -0.4816472625497169], [-0.06372994915165256, -0.48162905606005735], [-
0.06365385784966474, -0.4815723312718705]]
f(x): [1.3024751703109205, 1.3024756276729594, 1.3024756269175364]
action: reduction
----------------------------------------
Count of function calling: 264
Count of reflection's: 64 ; function calling: 64
Count of expansion's: 32 ; function calling: 32
Count of compression's: 1 ; function calling: 1
Count of reduction's: 31 ; function calling: 62
Sum of operations calling: 128
Sum of functions calling by operations calling: 159
Last point is: [-0.24165975031038911, -0.30374253359372222]
Last value is: 1.12119783517
```

```
Count of function calling by Sven method: 5
Count of function calling by Golden section method: 32
Lambda for Sven: 0.3435467859473693

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.7
Itterations: 142
x: [[0.2676956055207413, 0.2322987932505764], [0.2676488460548241, 0.23234166146184945],
[0.26766908171751813, 0.23232677655334144]]
f(x): [0.8607030492432133, 0.8607050552051103, 0.8607031628901853]
action: compression
----------------------------------------
Count of function calling: 495
Count of reflection's: 142 ; function calling: 142
Count of expansion's: 78 ; function calling: 78
Count of compression's: 34 ; function calling: 34
Count of reduction's: 30 ; function calling: 60
Sum of operations calling: 284
Sum of functions calling by operations calling: 314
Last point is: [0.2682900692551788, 0.23170432951613892]
Last value is: 0.8606973705880918
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 31
Lambda for Sven: 0.2506228905754524

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.6
Itterations: 105
x: [[0.26920664553460893, 0.2307865491269961], [0.2692075718386384, 0.2307836024747718],
[0.26920902184229206, 0.23078783823017837]]
f(x): [0.8607112419917652, 0.8607118780380189, 0.8607102077969337]
action: compression
----------------------------------------
Count of function calling: 384
Count of reflection's: 105 ; function calling: 105
Count of expansion's: 55 ; function calling: 55
Count of compression's: 20 ; function calling: 20
Count of reduction's: 30 ; function calling: 60
Sum of operations calling: 210
Sum of functions calling by operations calling: 240
Last point is: [0.26829131545524276, 0.23170554461722767]
Last value is: 0.8606966644455029
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 31
Lambda for Sven: 0.2507354419664706

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.5
Itterations: 174
x: [[0.2550629132939354, 0.2449328720334352], [0.2550633767799903, 0.2449357815310474],
[0.25506358752756575, 0.24493582695649743]]
f(x): [0.8634961125477423, 0.8634956520171705, 0.8634955431007562]
action: reduction
----------------------------------------
Count of function calling: 617
Count of reflection's: 174 ; function calling: 174
Count of expansion's: 99 ; function calling: 99
Count of compression's: 19 ; function calling: 19
Count of reduction's: 56 ; function calling: 112
Sum of operations calling: 348
Sum of functions calling by operations calling: 404
Last point is: [0.26829226266614536, 0.2317071518179178]
```

```
Last value is: 0.860695931588737
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 31
Lambda for Sven: 0.2500509878097819

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.4
Itterations: 68
x: [[0.26912373750229823, 0.23087463621048673], [0.2691280626793548, 0.23087058469892285],
[0.26912913446751097, 0.2308699412318079]]
f(x): [0.8607073539945619, 0.8607073878694487, 0.8607072881942099]
action: compression
----------------------------------------
Count of function calling: 262
Count of reflection's: 68 ; function calling: 68
Count of expansion's: 33 ; function calling: 33
Count of compression's: 16 ; function calling: 16
Count of reduction's: 19 ; function calling: 38
Sum of operations calling: 136
Sum of functions calling by operations calling: 155
Last point is: [0.26829242294765693, 0.23170665275166197]
Last value is: 0.8606960287858718
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 31
Lambda for Sven: 0.250730353949826

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.3
Itterations: 46
x: [[0.2683373003523267, 0.23165423779730787], [0.26853398569658266, 0.23146455616573736],
[0.2682451638917328, 0.23175360993282498]]
f(x): [0.8606982294916913, 0.8606971235168432, 0.8606961507848512]
action: reduction
----------------------------------------
Count of function calling: 190
Count of reflection's: 46 ; function calling: 46
Count of expansion's: 24 ; function calling: 24
Count of compression's: 9 ; function calling: 9
Count of reduction's: 13 ; function calling: 26
Sum of operations calling: 92
Sum of functions calling by operations calling: 105
Last point is: [0.2682922402380465, 0.2317065335865114]
Last value is: 0.8606961153930859
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 31
Lambda for Sven: 0.2506643210233544

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.2
Itterations: 76
x: [[0.22039896270253495, 0.2795997339415809], [0.2203992274077263, 0.27960031456798773],
[0.22039920315666456, 0.2796001852393941 7]]
f(x): [0.8954126073139651, 0.8954125938470059, 0.895412563689408]
action: reduction
----------------------------------------
Count of function calling: 298
Count of reflection's: 76 ; function calling: 76
Count of expansion's: 37 ; function calling: 37
Count of compression's: 9 ; function calling: 9
Count of reduction's: 30 ; function calling: 60
Sum of operations calling: 152
Sum of functions calling by operations calling: 182
```

```
Last point is: [0.26829250233879537, 0.23170688605726342]
Last value is: 0.8606959390732787
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 32
Lambda for Sven: 0.2517459754772618

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.1
Itterations: 61
x: [[0.27000658524158433, 0.2299834842894312], [0.27002029837840225, 0.22996928170796205],
[0.27003851487949676, 0.22995863178270465]]
f(x): [0.8607460978344255, 0.8607470122154749, 0.8607456542022001]
action: compression
----------------------------------------
Count of function calling: 243
Count of reflection's: 61 ; function calling: 61
Count of expansion's: 31 ; function calling: 31
Count of compression's: 9 ; function calling: 9
Count of reduction's: 21 ; function calling: 42
Sum of operations calling: 122
Sum of functions calling by operations calling: 143
Last point is: [0.26829115126462705, 0.23170599539757436]
Last value is: 0.8606965822232765
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 31
Lambda for Sven: 0.2508004902803199

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.0
Itterations: 54
x: [[0.2675602670520142, 0.23156078643319514], [0.26754228451098816, 0.23169968671331106],
[0.26888369662221606, 0.2305981936381196]]
f(x): [0.8609493762452566, 0.860915540787692, 0.8608558342211128]
action: reduction for cycling of min
----------------------------------------
Count of function calling: 216
Count of reflection's: 54 ; function calling: 54
Count of expansion's: 28 ; function calling: 28
Count of compression's: 11 ; function calling: 11
Count of reduction's: 15 ; function calling: 30
Sum of operations calling: 108
Sum of functions calling by operations calling: 123
Last point is: [0.26803985812368125, 0.2314420321366544]
Last value is: 0.8608443957240458
Count of function calling by Sven method: 4
Count of function calling by Golden section method: 31
Lambda for Sven: 0.25047352077454377
```

Метод ДСК Пауела. Результати виконання програми для різних значень параметра ε і λ

```
---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 10
Side length: 1.1
Itterations: 137
x: [[0.26830795562554094, 0.23168934535233027], [0.26830925515508003, 0.2316880478871847],
[0.26830458544823277, 0.23169271028423785]]
f(x): [0.8606965423742026, 0.8606965425016357, 0.8606965422652137]
action: expansion
---------------------------------------
Count of function calling: 1462
Count of reflection's: 137 ; function calling: 137
Count of expansion's: 72 ; function calling: 72
Count of compression's: 23 ; function calling: 23
Count of reduction's: 42 ; function calling: 84
Sum of operations calling: 274
Sum of functions calling by operations calling: 316
Last point is: [0.26829136375745466, 0.23170593197501596]
Last value is: 0.8606965394549523
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 1001
Lambda for Sven: 0.25066797020301956

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 9
Side length: 1.1
Itterations: 130
x: [[0.26833798362547023, 0.23165936430146622], [0.26847524522326915, 0.23152231855848512],
[0.26827834553892366, 0.23171890953002267]]
f(x): [0.860696559381699, 0.86069700534149, 0.8606965538373964]
action: reflection
---------------------------------------
Count of function calling: 440
Count of reflection's: 130 ; function calling: 130
Count of expansion's: 70 ; function calling: 70
Count of compression's: 21 ; function calling: 21
Count of reduction's: 39 ; function calling: 78
Sum of operations calling: 260
Sum of functions calling by operations calling: 299
Last point is: [0.2682902447610247, 0.23170701030792165]
Last value is: 0.860696551140672
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 3
Lambda for Sven: 0.2506660365096206

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 8
Side length: 1.1
Itterations: 86
x: [[0.27003744479888336, 0.22996253181915202], [0.2700382099625712, 0.22996176248909336],
[0.27003775513429634, 0.22996222796171784]]
f(x): [0.8607447048884046, 0.8607447491234596, 0.8607447202605631]
action: compression
---------------------------------------
Count of function calling: 1296
Count of reflection's: 86 ; function calling: 86
Count of expansion's: 42 ; function calling: 42
Count of compression's: 15 ; function calling: 15
Count of reduction's: 29 ; function calling: 58
Sum of operations calling: 172
Sum of functions calling by operations calling: 201
Last point is: [0.2682914081456554, 0.23170856847237994]
Last value is: 0.8606957703386782
```

```
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 1001
Lambda for Sven: 0.2508017021814091

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 7
Side length: 1.1
Itterations: 83
x: [[0.2700380949601012, 0.22996184786673327], [0.2700382099625712, 0.22996176248909336],
[0.27003744479888336, 0.2299625318191520 2]]
f(x): [0.8607447519792932, 0.8607447491234596, 0.8607447048884046]
action: reduction
----------------------------------------
Count of function calling: 1287
Count of reflection's: 83 ; function calling: 83
Count of expansion's: 41 ; function calling: 41
Count of compression's: 13 ; function calling: 13
Count of reduction's: 29 ; function calling: 58
Sum of operations calling: 166
Sum of functions calling by operations calling: 195
Last point is: [0.2682914081456554, 0.23170856847237994]
Last value is: 0.8606957703386782
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 1001
Lambda for Sven: 0.2508017021814091

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 6
Side length: 1.1
Itterations: 61
x: [[0.27000658524158433, 0.2299834842894312], [0.27002029837840225, 0.22996928170796205],
[0.27003851487949676, 0.22995863178270465]]
f(x): [0.8607460978344255, 0.8607470122154749, 0.8607456542022001]
action: compression
----------------------------------------
Count of function calling: 1213
Count of reflection's: 61 ; function calling: 61
Count of expansion's: 31 ; function calling: 31
Count of compression's: 9 ; function calling: 9
Count of reduction's: 21 ; function calling: 42
Sum of operations calling: 122
Sum of functions calling by operations calling: 143
Last point is: [0.2682900222064234, 0.23170712445577799]
Last value is: 0.8606965822488442
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 1001
Lambda for Sven: 0.2508004902803199

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.1
Itterations: 42
x: [[0.26795430102674833, 0.2315451421560734], [0.2680380643664487, 0.23142563588130222],
[0.26785047181173144, 0.2316128117165849]]
f(x): [0.8608395032739 55, 0.8608496142812698, 0.8608502559153992]
action: reduction
----------------------------------------
Count of function calling: 151
Count of reflection's: 42 ; function calling: 42
Count of expansion's: 21 ; function calling: 21
Count of compression's: 5 ; function calling: 5
Count of reduction's: 16 ; function calling: 32
Sum of operations calling: 84
Sum of functions calling by operations calling: 100
Last point is: [0.2679932536045411, 0.23150618957828065]
```

```
Last value is: 0.8608394097274461
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 1
Lambda for Sven: 0.2504123202787849

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 4
Side length: 1.1
Itterations: 38
x: [[0.2642878318714637, 0.23393472351163574], [0.2716156660401069, 0.22662316744590597],
[0.26834366058306175, 0.2312913813041552 7]]
f(x): [0.8613635148534549, 0.8614814250710946, 0.8608013059746645]
action: compression
----------------------------------------
Count of function calling: 136
Count of reflection's: 38 ; function calling: 38
Count of expansion's: 21 ; function calling: 21
Count of compression's: 5 ; function calling: 5
Count of reduction's: 12 ; function calling: 24
Sum of operations calling: 76
Sum of functions calling by operations calling: 88
Last point is: [0.2679061872856234, 0.2317288546015937]
Last value is: 0.8608011615837243
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 2
Lambda for Sven: 0.2505035161836073

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 3
Side length: 1.1
Itterations: 38
x: [[0.2642878318714637, 0.23393472351163574], [0.2716156660401069, 0.22662316744590597],
[0.26834366058306175, 0.2312913813041552 7]]
f(x): [0.8613635148534549, 0.8614814250710946, 0.8608013059746645]
action: compression
----------------------------------------
Count of function calling: 136
Count of reflection's: 38 ; function calling: 38
Count of expansion's: 21 ; function calling: 21
Count of compression's: 5 ; function calling: 5
Count of reduction's: 12 ; function calling: 24
Sum of operations calling: 76
Sum of functions calling by operations calling: 88
Last point is: [0.2679061872856234, 0.2317288546015937]
Last value is: 0.8608011615837243
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 2
Lambda for Sven: 0.2505035161836073

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 2
Side length: 1.1
Itterations: 9
x: [[0.16164065637946787, 0.14077837225402226], [0.17835152033485752, 0.1654540869926972],
[0.218956936640103, 0.16623524938151632]]
f(x): [0.9170338550672689, 0.9070061263814998, 0.8936659670972114]
action: reduction for cycling of min
----------------------------------------
Count of function calling: 42
Count of reflection's: 9 ; function calling: 9
Count of expansion's: 4 ; function calling: 4
Count of compression's: 0 ; function calling: 0
Count of reduction's: 5 ; function calling: 10
Sum of operations calling: 18
Sum of functions calling by operations calling: 23
```

```
Last point is: [0.20927999001535402, 0.17591219600626531]
Last value is: 0.8931569974802394
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 2
Lambda for Sven: 0.1943917413880448

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 1
Side length: 1.1
Itterations: 3
x: [[-0.25136197092713397, -0.14030641395441723], [-0.13748159108202485, 0.2847009496127728],
[0.17364539264005607, -0.026426034109308105]]
f(x): [1.140047243319074, 1.3243575216482033, 1.0201685244112375]
action: reflection
----------------------------------------
Count of function calling: 20
Count of reflection's: 3 ; function calling: 3
Count of expansion's: 2 ; function calling: 2
Count of compression's: 0 ; function calling: 0
Count of reduction's: 1 ; function calling: 2
Sum of operations calling: 6
Sum of functions calling by operations calling: 7
Last point is: [0.058267080058061645, 0.08895227847268633]
Last value is: 0.972994818239
Count of function calling by Sven method: 5
Count of function calling by DSK Paula: 1
Lambda for Sven: 0.12419955246269115
```

Метод ДСК Пауела. Результати виконання програми для різних значень параметра t і λ

```
---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 2.0
Itterations: 74
x: [[0.26001045338475876, 0.2112847358018701], [0.2592541235521877, 0.21171240171594127],
[0.2601329721042685, 0.2112198744337942]]
f(x): [0.8694029094374804, 0.8693953471115873, 0.8694035354619462]
action: reduction
---------------------------------------
Count of function calling: 1251
Count of reflection's: 74 ; function calling: 74
Count of expansion's: 38 ; function calling: 38
Count of compression's: 16 ; function calling: 16
Count of reduction's: 20 ; function calling: 40
Sum of operations calling: 148
Sum of functions calling by operations calling: 168
Last point is: [0.25920060223305064, 0.21176592303507827]
Last value is: 0.8693868361306952
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 1001
Lambda for Sven: 0.23667999664013126

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.9
Itterations: 62
x: [[0.2685604646528471, 0.23120298285781135], [0.2685606039415309, 0.2311865483502641],
[0.2685049249935836, 0.2312396557184944]]
f(x): [0.8607659872250061, 0.8607707625497832, 0.8607708634016598]
action: reduction
---------------------------------------
Count of function calling: 217
Count of reflection's: 62 ; function calling: 62
Count of expansion's: 31 ; function calling: 31
Count of compression's: 10 ; function calling: 10
Count of reduction's: 21 ; function calling: 42
Sum of operations calling: 124
Sum of functions calling by operations calling: 145
Last point is: [0.26782932640460144, 0.23193412110605705]
Last value is: 0.8607655735005176
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 2
Lambda for Sven: 0.250578872270691

---------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.8
Itterations: 42
x: [[-0.22832887668477278, -0.6046699177165862], [-0.2274935091158351, -0.604101347980673], [-
0.22774909525098166, -0.6042793329433848]]
f(x): [1.3077834392976617, 1.307778670739354, 1.3077834861199915]
action: reduction for cycling of min
---------------------------------------
Count of function calling: 1157
Count of reflection's: 42 ; function calling: 42
Count of expansion's: 20 ; function calling: 20
Count of compression's: 0 ; function calling: 0
Count of reduction's: 22 ; function calling: 44
Sum of operations calling: 84
Sum of functions calling by operations calling: 106
Last point is: [-0.3238887788983468, -0.5077060781981652]
Last value is: 1.2024475787832054
```

```
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 1001
Lambda for Sven: 0.45644919504907805

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.7
Itterations: 103
x: [[0.2619234625264355, 0.2328430256018843], [0.25970675124485143, 0.23643190227851635],
[0.2626674252199507, 0.2320366172982837]]
f(x): [0.862428759078716, 0.8625213402590673, 0.8623617543138129]
action: reflection
----------------------------------------
Count of function calling: 1343
Count of reflection's: 103 ; function calling: 103
Count of expansion's: 59 ; function calling: 59
Count of compression's: 19 ; function calling: 19
Count of reduction's: 25 ; function calling: 50
Sum of operations calling: 206
Sum of functions calling by operations calling: 231
Last point is: [0.26997590035018987, 0.2247281421680446]
Last value is: 0.8625047821919184
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 1001
Lambda for Sven: 0.24782571299091277

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.6
Itterations: 102
x: [[0.26920002292488887, 0.230774844581275], [0.2692075718386384, 0.2307836024747718],
[0.26920664553460893, 0.2307865491269961]]
f(x): [0.8607165681795814, 0.8607118780380189, 0.8607112419917652]
action: reduction
----------------------------------------
Count of function calling: 346
Count of reflection's: 102 ; function calling: 102
Count of expansion's: 54 ; function calling: 54
Count of compression's: 18 ; function calling: 18
Count of reduction's: 30 ; function calling: 60
Sum of operations calling: 204
Sum of functions calling by operations calling: 234
Last point is: [0.26746410290357747, 0.23252909175802755]
Last value is: 0.8607086643166304
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 2
Lambda for Sven: 0.2507335729992535

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.5
Itterations: 150
x: [[0.25497898917907347, 0.24461667306933887], [0.25499401320459786, 0.2446394314487624],
[0.2550170399439371, 0.24479421882708785]]
f(x): [0.8635631403156069, 0.8635538277739019, 0.8635299816784262]
action: reduction for cycling of min
----------------------------------------
Count of function calling: 1506
Count of reflection's: 150 ; function calling: 150
Count of expansion's: 87 ; function calling: 87
Count of compression's: 16 ; function calling: 16
Count of reduction's: 47 ; function calling: 94
Sum of operations calling: 300
Sum of functions calling by operations calling: 347
Last point is: [0.267609828848687, 0.23220142992233797]
```

```
Last value is: 0.8607555212996978
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 1001
Lambda for Sven: 0.2499578966875539

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.4
Itterations: 68
x: [[0.26912373750229823, 0.23087463621048673], [0.2691280626793548, 0.23087058469892285],
[0.26912913446751097, 0.2308699412318079]]
f(x): [0.8607073539945619, 0.8607073878694487, 0.8607072881942099]
action: compression
----------------------------------------
Count of function calling: 233
Count of reflection's: 68 ; function calling: 68
Count of expansion's: 33 ; function calling: 33
Count of compression's: 16 ; function calling: 16
Count of reduction's: 19 ; function calling: 38
Sum of operations calling: 136
Sum of functions calling by operations calling: 155
Last point is: [0.2675380951533741, 0.23246098054594477]
Last value is: 0.8607051713475186
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 2
Lambda for Sven: 0.250730353949826

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.3
Itterations: 25
x: [[0.26799564750131305, 0.2317168609674086], [0.26769817338388985, 0.23197367235489078],
[0.2682293372218175, 0.23139033634503364]]
f(x): [0.8607786356771693, 0.8607929384913106, 0.8608051116919566]
action: reduction
----------------------------------------
Count of function calling: 92
Count of reflection's: 25 ; function calling: 25
Count of expansion's: 11 ; function calling: 11
Count of compression's: 6 ; function calling: 6
Count of reduction's: 8 ; function calling: 16
Sum of operations calling: 50
Sum of functions calling by operations calling: 58
Last point is: [0.26806046421801577, 0.23165204425070587]
Last value is: 0.8607783765322651
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 1
Lambda for Sven: 0.2505138426676632

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.2
Itterations: 76
x: [[0.22039896270253495, 0.2795997339415809], [0.2203992274077263, 0.27960031456798773],
[0.22039920315666456, 0.2796001852393941 7]]
f(x): [0.8954126073139651, 0.8954125938470059, 0.895412563689408]
action: reduction
----------------------------------------
Count of function calling: 1267
Count of reflection's: 76 ; function calling: 76
Count of expansion's: 37 ; function calling: 37
Count of compression's: 9 ; function calling: 9
Count of reduction's: 30 ; function calling: 60
Sum of operations calling: 152
Sum of functions calling by operations calling: 182
```

```
Last point is: [0.2682914989898282, 0.23170788940623047]
Last value is: 0.8606959390856637
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 1001
Lambda for Sven: 0.2517459754772618

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.1
Itterations: 42
x: [[0.26795430102674833, 0.2315451421560734], [0.2680380643664487, 0.23142563588130222],
[0.26785047181173144, 0.2316128117165849]]
f(x): [0.860839503273955, 0.8608496142812698, 0.8608502559153992]
action: reduction
----------------------------------------
Count of function calling: 151
Count of reflection's: 42 ; function calling: 42
Count of expansion's: 21 ; function calling: 21
Count of compression's: 5 ; function calling: 5
Count of reduction's: 16 ; function calling: 32
Sum of operations calling: 84
Sum of functions calling by operations calling: 100
Last point is: [0.2679932536045411, 0.23150618957828065]
Last value is: 0.8608394097274461
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 1
Lambda for Sven: 0.2504123202787849

----------------------------------------
Result:
Coefficients: alpha 1.0 beta: 0.3 gamma: 3.0 sigma: 0.4
Accuracy: 5
Side length: 1.0
Itterations: 54
x: [[0.2675602670520142, 0.23156078643319514], [0.26754228451098816, 0.23169968671331106],
[0.26888369662221606, 0.2305981936381196]]
f(x): [0.8609493762452566, 0.860915540787692, 0.8608558342211128]
action: reduction for cycling of min
----------------------------------------
Count of function calling: 187
Count of reflection's: 54 ; function calling: 54
Count of expansion's: 28 ; function calling: 28
Count of compression's: 11 ; function calling: 11
Count of reduction's: 15 ; function calling: 30
Sum of operations calling: 108
Sum of functions calling by operations calling: 123
Last point is: [0.26728146839126005, 0.23220042186907563]
Last value is: 0.8608536390647341
Count of function calling by Sven method: 4
Count of function calling by DSK Paula: 2
Lambda for Sven: 0.25047352077454377
```

ДОДАТОК Е КОД ПРОГРАМИ

```
import math
import matrix
import excel_transfer
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
from matplotlib import mlab

from matplotlib.path import Path
import matplotlib.patches as patches
import matplotlib
from matplotlib.patches import Polygon
from matplotlib.collections import PatchCollection
from mpl_toolkits.mplot3d import axes3d


import openpyxl
from openpyxl.utils import get_column_letter



from resource import expression
import sven_method_lc_cw
import golden_section_search_lc_cw
import dichotomy_method_lc_cw
import dsk_paula_lc_cw
class NMM:
    def __init__(self):
        self.commands = {
            "commands": {
                "none": 0,
                "exit": 1,
                "test": 2,
                "clear": 3,
                "help": 4,
                "new": 5,
                "show slist": 6,
                "show scount": 7,
                "acc": 8,
                "mk": 9,
                "start": 10,
                "show result": 11,
                "image 1": 12,
                "image 2": 13,
                "test 1": 14,
                "start 1": 15,
            },
            "description": {
                "none": "do nothing",
                "exit": "exit from module",
                "test": "do test stuff",
                "clear": "clear something",
                "help": "display helpfull information",
                "new": "enter new raw data",
                "show slist": "show raw data",
                "show acc": "show accuracy",
                "acc": "set accuracy",
                "mk": "set default raw data",
                "start": "start calculation process",
                "show result": "show result",
                "image 1": "show 2D visualization",
                "image 2": "show 3D visualization",
                "start 1": "start with condition"
            }
        }
        self.expression = expression.Expression("No name", "x**2")
        self.condition = expression.Expression("No name", "x < 5")
        self.accuracy = 6
        self.epsilon = [1, 1]
        self.mm = True
        self.msycle = 3
        self.result = {"i": [], "xk": [], "fx": [], "action": [], "f_call": 0,
                       "action_count": {
                           "reflection": [0] * 2,
                           "expansion": [0] * 2,
                           "reduction": [0] * 2,
```

```
                        "compression": [0] * 2
                    }}
        self.pcd = {"i": [], "xk": [], "fxk": [], "action": [], "f_call": 0}
        self.sm = sven_method_lc_cw.SM()
        #self.cof = {"a": 1.0, "g": 2.0, "b": 0.5, "s": 0.5}
        self.cof = {"a": 1.0, "g": 3.0, "b": 0.3, "s": 0.4}
        self.amount_of_results = []
        self.symbol = "s"
        self.simplex_border_length = 1.1
        self.one_dir = "dskp_f_call"
        self.d_for_sven = 0.1
        self.makedefault()



    def showCommands(self):
        print('')
        print("Commands...")
        print("---")
        for item in self.commands["commands"]:
            print(str(item) + ":")
            print("Number: " + str(self.commands["commands"][item]))
            print("Description: " + str(self.commands["description"][item]))
            print("---")

    def enterCommand(self):
        command = "0"
        print('')
        print("Enter command (help for Q&A)")
        while (command not in self.commands):
            command = input("->")
            if (command not in self.commands["commands"]):
                print("There is no such command")
            else:
                return self.commands["commands"][command]

    def showHelp(self):
        print('')
        print("Help v0.002")
        self.showCommands()

    def makedefault(self):
        self.epsilon[0] = 10.0 ** (-self.accuracy)
        self.epsilon[1] = self.epsilon[0]
        self.condition = expression.Expression("Linear Condition", "a*(x1)+b*(x2) >= c or False")
        self.condition.parameters["linear"] = True
        self.condition.parameters["a"] = -1.0
        self.condition.parameters["b"] = -1.0
        self.condition.parameters["c"] = -0.5

        self.start_point = [-1.2, 0.0]

        self.expression.parameters["global_min"] = [1.0, 1.0]

        self.expression.parameters["unimodal"] = True

        self.count_of_vertex = 3

        self.msycle = self.count_of_vertex

        self.result = {"i": [], "xk": [], "fx": [], "action": [], "f_call": 0,
                       "action_count": {
                           "reflection": [0] * 2,
                           "expansion": [0] * 2,
                           "reduction": [0] * 2,
                           "compression": [0] * 2
                       }}

        self.pcd = {"i": [], "xk": [], "fxk": [], "action": [], "f_call": 0}

        self.sm.importparam(self.accuracy, self.expression, self.condition)


    def results_collector(self, result: type({})):
        if self.condition.parameters["linear"] == False:
            self.amount_of_results.append(
                {"i": result["i"].copy(), "xk": self.deepcopy(result["xk"]), "fx": result["fx"].copy(),
"action": result["action"].copy(), "f_call": result["f_call"],
                 "action_count": {
                     "reflection": result["action_count"]["reflection"].copy(),
```

```
                    "expansion": result["action_count"]["expansion"].copy(),
                    "reduction": result["action_count"]["reduction"].copy(),
                    "compression": result["action_count"]["compression"].copy()
                },
                "cof": {"a": self.cof["a"], "g": self.cof["g"], "b": self.cof["b"], "s": self.cof["s"], "l":
self.simplex_border_length, "ac": self.accuracy}
                }
            )
        else:
            self.amount_of_results.append(
                {"i": result["i"].copy(), "xk": self.deepcopy(result["xk"]), "fx": result["fx"].copy(),
                 "action": result["action"].copy(), "f_call": result["f_call"],
                 "action_count": {
                    "reflection": result["action_count"]["reflection"].copy(),
                    "expansion": result["action_count"]["expansion"].copy(),
                    "reduction": result["action_count"]["reduction"].copy(),
                    "compression": result["action_count"]["compression"].copy()
                 },
                 "cof": {"a": self.cof["a"], "g": self.cof["g"], "b": self.cof["b"], "s": self.cof["s"],
                         "l": self.simplex_border_length, "ac": self.accuracy},
                "pcd": {"sm_f_call": self.pcd["sm_f_call"], str(self.one_dir): self.pcd[str(self.one_dir)],
"xk": self.deepcopy(self.pcd["xk"]), "fxk": self.pcd["fxk"].copy(), "d_lambda": self.d_for_sven}
                }
            )

    def write_data(self, result: type([])):
        wb = openpyxl.Workbook()
        dest_filename = 'cw_nm.xlsx'
        ws1 = wb.active
        ws1.title = "range names"

        for row in range(1, 40):
            ws1.append(range(600))

        ws2 = wb.create_sheet(title="Pi")

        ws2['F5'] = 3.14

        ws3 = wb.create_sheet(title="Data 1")
        temp_row = []
        if self.condition.parameters["linear"] == False:
            for row in range(len(result)):
                sum_of_o_call = result[row]["action_count"]["reflection"][0] +
result[row]["action_count"]["expansion"][0] + \
                                result[row]["action_count"]["compression"][0] +
result[row]["action_count"]["reduction"][0]
                sum_of_f_call = result[row]["action_count"]["reflection"][1] +
result[row]["action_count"]["expansion"][1] + \
                                result[row]["action_count"]["compression"][1] +
result[row]["action_count"]["reduction"][1]

                x = result[row]["xk"][-1][0]
                f = result[row]["fx"][-1][0]
                print(x, f)
                print(result[row]["f_call"])
                temp_row = [result[row]["cof"][self.symbol], result[row]["f_call"], str(x[0])+", "+str(x[1]),
f]
                for col in range(4):
                    _ = ws3.cell(column=col + 1, row=row + 1, value="{0}".format(temp_row[col]))
            print(ws3['AA10'].value)

            ws4 = wb.create_sheet(title="Data 2")
            temp_row = []
            for row in range(len(result)):
                sum_of_o_call = result[row]["action_count"]["reflection"][0] +
result[row]["action_count"]["expansion"][0] + \
                                result[row]["action_count"]["compression"][0] +
result[row]["action_count"]["reduction"][0]
                sum_of_f_call = result[row]["action_count"]["reflection"][1] +
result[row]["action_count"]["expansion"][1] + \
                                result[row]["action_count"]["compression"][1] +
result[row]["action_count"]["reduction"][1]

                x = result[row]["xk"][-1][0]
                f = result[row]["fx"][-1][0]
                print(x, f)
                print(result[row]["f_call"])
                temp_row = [result[row]["cof"][self.symbol], result[row]["action_count"]["reflection"][0],
result[row]["action_count"]["expansion"][0], result[row]["action_count"]["compression"][0],
result[row]["action_count"]["reduction"][0], sum_of_o_call]
```

```
                for col in range(len(temp_row)):
                    _ = ws4.cell(column=col + 1, row=row + 1, value="{0}".format(temp_row[col]))
            print(ws4['AA10'].value)
        else:
            for row in range(len(result)):
                sum_of_o_call = result[row]["action_count"]["reflection"][0] +
result[row]["action_count"]["expansion"][
                    0] + \
                                result[row]["action_count"]["compression"][0] + \
                                result[row]["action_count"]["reduction"][0]
                sum_of_f_call = result[row]["action_count"]["reflection"][1] +
result[row]["action_count"]["expansion"][
                    1] + \
                                result[row]["action_count"]["compression"][1] + \
                                result[row]["action_count"]["reduction"][1]
                x = result[row]["pcd"]["xk"]
                f = result[row]["pcd"]["fxk"]
                self.sm.par_sort(x, f)
                print(x, f)
                print(result[row]["f_call"])
                temp_row = [result[row]["cof"][self.symbol], result[row]["f_call"], str(x[0][0]) + ", " +
str(x[0][1]), f[0]]
                for col in range(4):
                    _ = ws3.cell(column=col + 1, row=row + 1, value="{0}".format(temp_row[col]))
            print(ws3['AA10'].value)

            ws4 = wb.create_sheet(title="Data 2")
            temp_row = []
            for row in range(len(result)):
                sum_of_o_call = result[row]["action_count"]["reflection"][0] +
result[row]["action_count"]["expansion"][
                    0] + \
                                result[row]["action_count"]["compression"][0] + \
                                result[row]["action_count"]["reduction"][0]
                sum_of_f_call = result[row]["action_count"]["reflection"][1] +
result[row]["action_count"]["expansion"][
                    1] + \
                                result[row]["action_count"]["compression"][1] + \
                                result[row]["action_count"]["reduction"][1]
                x = result[row]["pcd"]["xk"]
                f = result[row]["pcd"]["fxk"]
                print(x, f)
                print(result[row]["f_call"])
                temp_row = [result[row]["cof"][self.symbol], result[row]["f_call"],
result[row]["pcd"]["d_lambda"], f[0]]
                for col in range(len(temp_row)):
                    _ = ws4.cell(column=col + 1, row=row + 1, value="{0}".format(temp_row[col]))
            print(ws4['AA10'].value)
        wb.save(filename=dest_filenam
    def importparam(self, accuracy):
        # self.accuracy = accuracy
        pass

    def setaccuracy(self):
        task = 0
        print('')
        print("Enter accuracy:")
        while (task != 1):
            self.accuracy = int(input("-> "))
            print("Input is correct? (enter - yes/n - no)")
            command = input("-> ")
            if (command != "n"):
                task = 1
            else:
                if self.accuracy < 0:
                    print("Please enter positive number!")
                    task = 0
        self.epsilon[0] = 10 ** (-self.accuracy)
        self.epsilon[1] = self.epsilon[0]

    def inputnewdata(self):
        self.expression.input_expr()
        self.expression.input_range()
        pass

    def dostaff(self):
        task = 0
        while (task != 1):
            print('')
            print("Nelder–Mead method with fix")
```

```
            print('')
            task = self.enterCommand()
            if task == 2:
                self.print_boundary()
                pass
            elif task == 3:
                pass
            elif task == 4:
                self.showHelp()
            elif task == 5:
                self.inputnewdata()
            elif task == 6:
                self.print_raw_data()
            elif task == 8:
                self.setaccuracy()
            elif task == 9:
                self.makedefault()
            elif task == 10:
                self.resolve()
            elif task == 11:
                self.printresult()

            elif task == 12:
                self.printresult_g()
            elif task == 13:
                self.printresult_3d()
            elif task == 14:
                self.print_boundary_1()
            elif task == 15:
                self.one_dir == "dskp_f_call"
                self.cof = {"a": 1.0, "g": 3.0, "b": 0.3, "s": 0.4}
                # self.cof = {"a": 1.0, "g": 2.0, "b": 0.5, "s": 0.5}
                my_param = 2.0
                # my_param = 10
                self.symbol = "l"
                self.amount_of_results = []
                self.simplex_border_length = 1.1
                self.accuracy = 5
                while my_param > 0.95:
                    self.cof[self.symbol] = round(my_param, 1)
                    #self.cof[self.symbol] = my_param
                    self.simplex_border_length = round(my_param, 1)
                    # self.accuracy = my_param
                    self.resolve()
                    # self.print_boundary_1()
                    self.results_collector(self.result)
                    # my_param -= 1
                    my_param -= 0.1
                self.printresult_m(self.amount_of_results)
                self.write_data(self.amount_of_results)
        pass

    def print_raw_data(self):
        self.expression.show_expr()
        pass

    def resolve(self):
        self.makedefault()
        self.result["f_call"] = self.count_of_vertex
        k = 0
        flag = False
        out_of_condition = False
        exp_r = self.expression
        x_w = self.build_initial_simplex_basic(self.simplex_border_length, len(self.start_point),
self.count_of_vertex, self.start_point)
        center = [0 for _ in range(len(x_w[0]))]
        f_arr = [exp_r.execute_l(x) for x in x_w]
        h_temp = [0 for _ in range(len(x_w[0]))]
        cycling = [0 for _ in range(len(x_w))]
        print(x_w)
        self.par_sort(x_w, f_arr, cycling)
        print(x_w)
        self.collect_data(k, x_w, f_arr, "initial simplex")
        condition = self.check_condition(x_w)
        print("Condition: ", condition)
        self.redirect_way(f_arr, condition)
        self.par_sort(x_w, f_arr, cycling)
        while self.halting_check(f_arr, center) and k <= 600:
            k += 1
            print('')
```

```
print("----------")
print("Index", k)
i = 1
center = x_w[0].copy()
while i < (len(x_w) - 1):
    center = self.sum(center, x_w[i])
    i += 1
#center = self.sum(x_w[0], x_w[1])
center = self.mul(center, 1.0 / float(len(x_w) - 1))
h_temp = self.reflection(center, x_w)
print("center is", center)
print("h_temp is", h_temp)
f_h_temp = exp_r.execute_1(h_temp)
print("fcenter is", f_h_temp)
print("Cycling count", cycling)
if flag:
    self.reduction(x_w, 0)
    i = 0
    while i < len(x_w):
        f_arr[i] = exp_r.execute_1(x_w[i])
        i += 1
    flag = False
    self.collect_data(k, x_w, f_arr, "reduction for cycling of min")
elif f_h_temp <= f_arr[0]:
    h_temp_new = self.expansion(center, h_temp, x_w)
    f_h_temp_new = exp_r.execute_1(h_temp_new)
    if f_h_temp_new < f_arr[0]:
        x_w[-1] = h_temp_new.copy()
        f_arr[-1] = f_h_temp_new
        self.collect_data(k, x_w, f_arr, "expansion")
    else:
        x_w[-1] = h_temp.copy()
        f_arr[-1] = f_h_temp
        self.collect_data(k, x_w, f_arr, "reflection")
elif f_h_temp >= f_arr[-2] and f_h_temp < f_arr[-1]:
    h_temp_new = self.compression(center, h_temp, x_w)
    x_w[-1] = h_temp_new
    f_arr[-1] = exp_r.execute_1(h_temp_new)
    self.collect_data(k, x_w, f_arr, "compression")
else:
    self.reduction(x_w, 0)
    i = 0
    while i < len(x_w):
        f_arr[i] = exp_r.execute_1(x_w[i])
        i += 1
    self.collect_data(k, x_w, f_arr, "reduction")
condition = self.check_condition(x_w)
print("Condition: ", condition)

self.redirect_way(f_arr, condition)

self.par_sort(x_w, f_arr, cycling)
cycling_test = self.find_cycling(x_w, self.result["xk"][-2], cycling, self.msycle)
if cycling_test != None:
    print("--------------------")
    print("Cycling on step", k)
    action_type = self.analyse_cycling(cycling_test, f_arr, self.result["fx"][-2])
    # max_f == max_f_xp
    print("action type", action_type)
    if action_type == 0:
        x_temp = x_w[-1]
        x_w[-1] = x_w[-2]
        x_w[-2] = x_temp
        f_temp = f_arr[-1]
        f_arr[-1] = f_arr[-2]
        f_arr[-2] = f_temp
        cycling[-1] = 0
    elif action_type == 1:
        x_temp = x_w[0]
        x_w[0] = x_w[1]
        x_w[1] = x_temp
        f_temp = f_arr[0]
        f_arr[0] = f_arr[1]
        f_arr[1] = f_temp
        flag = True
        cycling[0] = 0
    elif action_type == 2:
        pass
    else:
        print("WTF")
```

```
            self.par_sort(x_w, f_arr, cycling)
            if self.condition.parameters["linear"]:
                if self.one_dir == "dskp_f_call":
                    self.do_linear_condition_p(x_w[0], f_arr[0])
                    self.printresult()
                    self.dskp.printresult()
                    self.d_for_sven = self.dskp.d_for_sven
                elif self.one_dir == "gss_f_call":
                    self.do_linear_condition_g(x_w[0], f_arr[0])
                    self.printresult()
                    self.gss.printresult()
                    self.d_for_sven = self.gss.d_for_sven

            else:
                self.printresult()

        def do_linear_condition_d(self, x_lin: type([]), f: type([])):
            x = x_lin.copy()
            self.dm = dichotomy_method_lc_cw.DM()
            self.dm.d_for_sven = 0.1
            self.dm.importparam(self.accuracy, self.expression, self.condition, x)
            self.dm.makedefault()
            self.dm.resolve()
            self.pcd["xk"] = [self.dm.result["x1"][-1].copy(), self.dm.result["x2"][-1].copy()]
            self.pcd["fxk"] = self.dm.result["fxk"][-1].copy()

            self.dm.par_sort(self.pcd["xk"], self.pcd["fxk"])

        def do_linear_condition_g(self, x_lin: type([]), f: type([])):
            x = x_lin.copy()
            self.gss = golden_section_search_lc_cw.GSS()
            self.gss.d_for_sven = self.gss.norm(x) / self.gss.norm(
                [self.condition.parameters["a"], self.condition.parameters["b"]])
            self.gss.importparam(self.accuracy, self.expression, self.condition, x)
            self.gss.makedefault()
            self.gss.resolve()
            self.pcd["sm_f_call"] = len(self.gss.sm.result["xk"]) + 1
            self.pcd["gss_f_call"] = len(self.gss.result["i"]) + 1
            self.result["f_call"] += self.pcd["sm_f_call"] + self.pcd["gss_f_call"]
            self.pcd["xk"] = [self.gss.result["a"][-1].copy(), self.gss.result["b"][-1].copy()]
            self.pcd["fxk"] = [self.expression.execute_l(self.pcd["xk"][0]),
    self.expression.execute_l(self.pcd["xk"][1])]
            self.gss.printresult()

            self.gss.par_sort(self.pcd["xk"], self.pcd["fxk"])

        def do_linear_condition_p(self, x_lin: type([]), f: type([])):
            x = x_lin.copy()

            self.dskp = dsk_paula_lc_cw.DSKP()
            self.dskp.d_for_sven = self.dskp.norm(x) / self.dskp.norm(
                [self.condition.parameters["a"], self.condition.parameters["b"]])
            self.dskp.importparam(self.accuracy, self.expression, self.condition, x)
            self.dskp.makedefault()
            self.dskp.resolve()
            self.pcd["sm_f_call"] = len(self.dskp.sm.result["xk"]) + 1
            self.pcd["dskp_f_call"] = len(self.dskp.result["i"])
            self.result["f_call"] += self.pcd["sm_f_call"] + self.pcd["dskp_f_call"]
            self.pcd["xk"] = [self.dskp.result["xst"][-1].copy()]
            self.pcd["fxk"] = [self.dskp.result["fsxt"][-1]]
            self.dskp.printresult()

        def check_condition(self, x_w):
            result = [[True], []]
            i = 0
            while i < len(x_w):
                # x = {"x": x_w[i][0], "y": x_w[i][1]}
                x = {}
                j = 0
                while j < len(x_w[i]):
                    x["x"+str(j+1)] = x_w[i][j]
                    j += 1
                self.condition.parameters.update(x)
                result[1].append(self.condition.execute_d(self.condition.parameters))
                if not result[1][-1]:
                    result[0] = False
                i += 1
            return result

        def redirect_way(self, f, condition):
```

```
    if not condition[0]:
        i = 0
        while i < self.count_of_vertex:
            if not condition[1][i]:
                f[i] = float('Inf')
                self.result["fx"][-1][i] = f[i]
            i += 1

def sven_method(self, x_w, S, flag, part):
    stat = True
    start = 0.0
    interval = []
    self.r_expression = self.expression.copy()
    self.r_expression.rename(self.expression.name)

    self.r_condition = self.condition.copy()
    self.r_condition.rename(self.condition.name)
    # S = matrix.Vector([0.0, 1.0], "Vector S(1)")
    # d_lambda = self.get_d_lambda(x_w, S)
    d_lambda = self.epsilon[0]
    if part < 3:
        # S(flag)
        self.r_expression.replace_arg(self.arguments_list(x_w, flag))
        self.r_condition.replace_arg(self.arguments_list(x_w, flag))
        start = x_w[flag]
    elif part == 3:
        self.r_expression.replace_arg(self.lambda_arguments_list(x_w, S, flag))
        self.r_condition.replace_arg(self.lambda_arguments_list(x_w, S, flag))
        start = x_w[0]
    else:
        print("Wrong Vector S([1, 2])")
        stat = False
    if stat:
        self.sm.makedefault()
        print("In Sven by part #", part)
        print("Flag", flag)
        self.r_expression.show_expr()
        self.sm.expression = self.r_expression.copy()
        self.sm.condition = self.r_condition.copy()

        self.sm.x_start = start
        # d_lambda here
        print("D lambda is", d_lambda)
        self.sm.d = d_lambda
        self.sm.expression.show_expr()
        self.sm.resolve()
        raw_group = self.sm.find_min()
        print("Raw group", raw_group)
        self.sm.printresult()
        self.sm.printresult_graph()
        interval = raw_group["xk"].copy()
        interval.sort()
        interval.pop(1)
        print("Raw group", interval)
        print("Interval is:", interval)
        pass
    else:
        interval = None
    return interval

def get_d_lambda(self, x_w, s):
    try:
        d_lambda = self.norm(x_w) / self.norm(s.vector)
    except ZeroDivisionError:
        d_lambda = float('Inf')
    return d_lambda

@staticmethod
def arguments_list(x_w, flag):
    i = 0
    replace_array = []
    while i < len(x_w):
        if i != flag:
            replace_array.append(x_w[i])
        else:
            replace_array.append(None)
        i += 1
    return replace_array

@staticmethod
```

```
def lambda_arguments_list(x_w, s, flag):
    i = 0
    replace_array = []
    while i < len(x_w):
        replace_array.append("(" + str(x_w[i]) + "+" + str(s.vector[i]) + "*x" + ")")
        i += 1
    return replace_array

@staticmethod
def build_initial_simplex_zero(size_s: float, f_dim: int, count_of_vertex: int) -> list:
    i = 0
    simplex = [[0.0] * f_dim]

    sqrt_two = math.sqrt(2.0)
    try:
        d1 = (size_s / (float(f_dim) * sqrt_two)) * (math.sqrt(float(f_dim) + 1.0) + float(f_dim) - 1.0)
    except ZeroDivisionError:
        d1 = float('Inf')

    try:
        d2 = (size_s / (float(f_dim) * sqrt_two)) * (math.sqrt(float(f_dim) + 1.0) - 1.0)
    except ZeroDivisionError:
        d2 = float('Inf')

    while i < count_of_vertex - 1:
        j = 0
        vertex = []
        while j < f_dim:
            if j == i:
                vertex.append(d1)
            else:
                vertex.append(d2)
            j += 1
        simplex.append(vertex)
        i += 1
    return simplex

@staticmethod
def build_initial_simplex_basic(size_s: float, f_dim: int, count_of_vertex: int, x_start: list) -> list:
    i = 0
    simplex = [x_start.copy()]

    sqrt_two = math.sqrt(2.0)
    try:
        d1 = (size_s / (float(f_dim) * sqrt_two)) * (math.sqrt(float(f_dim) + 1.0) + float(f_dim) - 1.0)
    except ZeroDivisionError:
        d1 = float('Inf')

    try:
        d2 = (size_s / (float(f_dim) * sqrt_two)) * (math.sqrt(float(f_dim) + 1.0) - 1.0)
    except ZeroDivisionError:
        d2 = float('Inf')

    while i < count_of_vertex - 1:
        j = 0
        vertex = []
        while j < f_dim:
            if j == i:
                vertex.append(x_start[j] + d1)
            else:
                vertex.append(x_start[j] + d2)
            j += 1
        simplex.append(vertex)
        i += 1
    return simplex

def reduction(self, x_w, vertex):
    i = 1
    while i < len(x_w):
        x_w[i] = self.dif(x_w[i], x_w[vertex])
        x_w[i] = self.mul(x_w[i], self.cof["s"])
        x_w[i] = self.sum(x_w[i], x_w[vertex])
        self.result["f_call"] += 1
        i += 1
    self.result["action_count"]["reduction"][0] += 1
    self.result["action_count"]["reduction"][1] += i - 1

def expansion(self, center, h_temp, x_w):
    h_temp = self.dif(h_temp, center)
    h_temp = self.mul(h_temp, self.cof["g"])
```

```
        h_temp = self.sum(h_temp, center)
        self.result["f_call"] += 1
        self.result["action_count"]["expansion"][0] += 1
        self.result["action_count"]["expansion"][1] += 1
        return h_temp

    def compression(self, center, h_temp, x_w):
        h_temp = self.dif(x_w[-1], center)
        h_temp = self.mul(h_temp, self.cof["b"])
        h_temp = self.sum(h_temp, center)
        self.result["f_call"] += 1
        self.result["action_count"]["compression"][0] += 1
        self.result["action_count"]["compression"][1] += 1
        return h_temp

    def reflection(self, center, x_w):
        h_temp = self.dif(center, x_w[-1])
        h_temp = self.mul(h_temp, self.cof["a"])
        h_temp = self.sum(h_temp, center)
        self.result["f_call"] += 1
        self.result["action_count"]["reflection"][0] += 1
        self.result["action_count"]["reflection"][1] += 1
        return h_temp

    def par_sort(self, x, f, cycling):
        f_temp = f.copy()
        x_temp = self.deepcopy(x)
        cycling_temp = cycling.copy()
        index = [i for i in range(len(x))]
        f.sort()
        for i in range(len(x)):
            x[i] = x_temp[f_temp.index(f[i])]
            cycling[i] = cycling_temp[f_temp.index(f[i])]
            f_temp[f_temp.index(f[i])] = None

    @staticmethod
    def analyse_cycling(i_cycling, f_x, f_xp):
        answer = None
        if max(f_x) == max(f_xp):
            answer = 0
        elif i_cycling == f_x.index(min(f_x)):
            answer = 1
        else:
            answer = 2
        return answer


    @staticmethod
    def find_cycling(x, xp, cycling, m_cycling):
        answer = None
        if NMM.compare(x, xp):
            i = 0
            while i < len(x):
                if x[i] in xp:
                    cycling[i] += 1
                if cycling[i] >= m_cycling:
                    answer = i
                    cycling[i] = 0
                i += 1
        return answer

    @staticmethod
    def compare0(x1, x2):
        ansver = False
        for i in range(len(x1)):
            for j in range(len(x1[0])):
                if x1[i][j] in x2:
                    ansver = True
        return ansver

    @staticmethod
    def compare(x1, x2):
        ansver = False
        for i in range(len(x1)):
            if x1[i] in x2:
                ansver = True
        return ansver

    def halting_check(self, f_arr, center):
        r = True
```

```
        f_center = self.expression.execute_1(center)
        if math.sqrt(math.pow(sum([item - f_center for item in f_arr]), 2.0) / float(len(f_arr))) <=
self.epsilon[0]:
            r = False
            print("Halting check! - True")
        self.result["f_call"] += 1
        return r

    @staticmethod
    def norm(v):
        return math.sqrt(sum([math.pow(item, 2) for item in v]))

    @staticmethod
    def dif(v1, v2):
        i = 0
        r = []
        while i < len(v1):
            r.append(v1[i] - v2[i])
            i += 1
        return r

    @staticmethod
    def dif_part(v1, v2, part):
        r = v1.copy()
        r[part] -= v2[part]
        return r

    @staticmethod
    def sum(v1, v2):
        i = 0
        r = []
        while i < len(v1):
            r.append(v1[i] + v2[i])
            i += 1
        return r

    @staticmethod
    def sum_part(v1, v2, part):
        r = v1.copy()
        r[part] += v2[part]
        return r

    @staticmethod
    def mul(v1, c):
        r = v1.copy()
        for i in range(len(r)):
            r[i] *= c
        return r

    @staticmethod
    def deepcopy(x):
        xn = [[]for _ in x]
        for i in range(len(x)):
            for j in range(len(x[i])):
                xn[i].append(x[i][j])
        return xn

    def collect_data(self, i, x, fx, action):
        self.result["i"].append(i)
        self.result["xk"].append(self.deepcopy(x))
        self.result["fx"].append(fx.copy())
        self.result["action"].append(action)

    def get_contour_line(self, a, step):
        cl_path_up = []
        cl_path_down = []
        boundary = self.get_boundary_for_cl(a)
        boundary.sort()
        x1 = boundary[0]

        while x1 < boundary[1]:
            pair = self.cl_expression_x2(x1, a)
            cl_path_down.append([x1, pair[0]])
            cl_path_up.append([x1, pair[1]])
            x1 += step

        pair = self.cl_expression_x2(boundary[1], a)
        cl_path_down.append([boundary[1], pair[0]])
        cl_path_up.append([boundary[1], pair[1]])
```

```
    cl_path_down.reverse()

    cl_path_down.pop(-1)
    cl_path_down.pop(0)

    return cl_path_up + cl_path_down

    pass

def cl_expression_x2(self, x1, a):
    part_root = (0.1 *(a**4 - (x1 - 1)**2))**0.5
    x2_up = x1 + part_root
    x2_down = x1 - part_root
    return [x2_down, x2_up]



def get_boundary_for_cl(self, a):
    return [1.0 - math.pow(a, 2.0), 1.0 + math.pow(a, 2.0)]

def printresult_g(self):
    fig, ax = plt.subplots()
    patches = []
    N = 3
    for i in range(len(self.result["i"])):
        for j in range(N):
            polygon = Polygon(np.array(self.result["xk"][i]), True)
        patches.append(polygon)

    p = PatchCollection(patches, cmap=matplotlib.cm.jet, alpha=0.4)

    colors = 100 * np.random.rand(len(patches))
    p.set_array(np.array(colors))

    ax.add_collection(p)
    # Set x ticks
    #plt.xticks(np.linspace(-10, 10, 10, endpoint=True))

    # Set y ticks
    #plt.yticks(np.linspace(-10, 10, 10, endpoint=True))
    plt.show()
    pass

def printresult_3d_0(self):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

    # Grab some test data.
    delta = 0.05
    radius = 10
    sector = self.expression.parameters["global_min"].copy()
    sector = [sector[0] - math.copysign(radius, sector[0]), sector[1] + math.copysign(radius, sector[1])]
    x = np.arange(sector[0], sector[1], delta)
    y = np.arange(sector[0], sector[1], delta)

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

    # Z1 = mlab.bivariate_normal(X, Y, 1.0, 1.0, 0.0, 0.0)
    # Z2 = mlab.bivariate_normal(X, Y, 1.5, 0.5, 1, 1)

    # Z = 10.0 * (Z2 - Z1)
    # "(10*(x1-x2)**2+(x1-1)**2)**0.25"
    # Z = 10.0 * ((X - Y)**2 + (X - 1)**2)**0.25

    Z = self.expression.execute_l([X, Y])

    # X, Y, Z = axes3d.get_test_data(0.05)

    # Plot a basic wireframe.
    ax.plot_wireframe(X, Y, Z, rstride=10, cstride=10)

    plt.show()
    pass

def printresult_3d(self):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

    # Grab some test data.
    delta = 0.025
```

```
        radius = 5
        sector = self.expression.parameters["global_min"].copy()
        sector = [sector[0] - math.copysign(radius, sector[0]), sector[1] + math.copysign(radius, sector[1])]
        x = np.arange(sector[0], sector[1], delta)
        y = np.arange(sector[0], sector[1], delta)

        X, Y = np.meshgrid(x, y)
        Z = self.expression.execute_l([X, Y])
        surf = ax.plot_surface(X, Y, Z, cmap=cm.jet,
                               linewidth=0, antialiased=True)

        ax.set_zlim(0.0, 25.0)
        ax.zaxis.set_major_locator(LinearLocator(10))
        ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))
        fig.colorbar(surf, shrink=0.5, aspect=5)

        plt.show()
        pass

    def print_boundary(self):
        a = 2.0
        cl_x = []
        cl_y = []
        cl = self.get_contour_line(a, 0.1)
        print(cl)

        i = 0
        while i < len(cl):
            cl_x.append(cl[i][0])
            cl_y.append(cl[i][1])
            i += 1
        print(cl_x)
        print(cl_y)
        #cl_x.sort()
        #cl_y.sort()
        #x = np.array(cl_x)
        #y = np.array(cl_y)
        delta = 0.055
        radius = 10
        sector = self.expression.parameters["global_min"].copy()
        sector = [sector[0] - math.copysign(radius, sector[0]), sector[1] + math.copysign(radius, sector[1])]
        x = np.arange(sector[0], sector[1], delta)
        y = np.arange(sector[0], sector[1], delta)

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

        Z = self.expression.execute_l([X, Y])

        print(Z)

        #
        fig, ax = plt.subplots()
        center = []
        m_patches = []
        verts = []
        N = self.count_of_vertex
        for i in range(len(self.result["i"])):
            polygon = Polygon(np.array(self.result["xk"][i]), True)
            # count center...
            j = 1
            center = self.result["xk"][i][0].copy()
            while j < N:
                center = self.sum(center, self.result["xk"][i][j])
                j += 1
            center = self.mul(center, 1.0 / float(N))
            # print("Center is", center)
            verts.append((center[0], center[1]))
            ax.text(center[0], center[1], "#"+str(self.result["i"][i])+" ", color="black", fontsize="10",
verticalalignment='bottom', horizontalalignment='right')
            # ...
            m_patches.append(polygon)

        verts.append((self.result["xk"][-1][0][0], self.result["xk"][-1][0][1]))
        i = 0
        while i < len(self.pcd["xk"]):
            verts.append((self.pcd["xk"][i][0], self.pcd["xk"][i][1]))
            ax.text(self.pcd["xk"][i][0], self.pcd["xk"][i][1], "#" + str(self.result["i"][i]) + " ",
color="green", fontsize="10",
                    verticalalignment='bottom', horizontalalignment='right')
            i += 1
```

```
        path = Path(verts)
        patch = patches.PathPatch(path, facecolor='none', lw=0.5)
        ax.add_patch(patch)
        xs, ys = zip(*verts)
        plt.plot(xs, ys, "x--", lw=1.5, color='black', ms=10)
        p = PatchCollection(m_patches, cmap=matplotlib.cm.jet, alpha=0.4, lw=1.0)

        colors = 100 * np.random.rand(len(m_patches))
        p.set_array(np.array(colors))

        ax.add_collection(p)


        #

        #plt.figure()
        #CS = plt.contour(X, Y, Z,
        #                 colors='k'  # negative contours will be dashed by default
        #                 )
        #plt.clabel(CS, fontsize=9, inline=1)
        #plt.title('Single color - negative contours dashed')

        #plt.figure()
        # circle2 = plt.Circle((self.condition.parameters["a"], self.condition.parameters["b"]),
self.condition.parameters["R"], color='r', fill=False)
        # ax.add_artist(circle2)
        if self.condition.parameters["linear"]:
            NV = [-self.condition.parameters["a"], self.condition.parameters["b"]]
            NV = self.mul(NV, 10)
            BX = [0 - NV[0]*2.0 - self.condition.parameters["c"], -self.condition.parameters["a"] + NV[0] -
self.condition.parameters["c"]]
            BY = [0 - NV[1]*2.0, self.condition.parameters["b"] + NV[1]]
            plt.plot(BX, BY, color='r')
        else:
            circle2 = plt.Circle((self.condition.parameters["a"], self.condition.parameters["b"]),
                                 self.condition.parameters["R1"], color='r', fill=False)
            circle3 = plt.Circle((self.condition.parameters["a"], self.condition.parameters["b"]),
                                 self.condition.parameters["R2"], color='r', fill=False)
            ax.add_artist(circle2)
            ax.add_artist(circle3)

        CS = plt.contour(X, Y, Z)
        plt.clabel(CS, inline=1, fontsize=10)
        plt.title("Path of simplex with "+str(N)+" vertexes")

        plt.show()
        pass

    def print_boundary_1(self):
        a = 2.0
        cl_x = []
        cl_y = []
        cl = self.get_contour_line(a, 0.1)
        print(cl)

        i = 0
        while i < len(cl):
            cl_x.append(cl[i][0])
            cl_y.append(cl[i][1])
            i += 1
        print(cl_x)
        print(cl_y)
        #cl_x.sort()
        #cl_y.sort()
        #x = np.array(cl_x)
        #y = np.array(cl_y)
        delta = 0.055
        radius = 10
        sector = self.expression.parameters["global_min"].copy()
        sector = [sector[0] - math.copysign(radius, sector[0]), sector[1] + math.copysign(radius, sector[1])]
        x = np.arange(sector[0], sector[1], delta)
        y = np.arange(sector[0], sector[1], delta)

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

        #Z1 = mlab.bivariate_normal(X, Y, 1.0, 1.0, 0.0, 0.0)
        #Z2 = mlab.bivariate_normal(X, Y, 1.5, 0.5, 1, 1)

        #Z = 10.0 * (Z2 - Z1)
```

```
        # "(10*(x1-x2)**2+(x1-1)**2)**0.25"
        #Z = 10.0 * ((X - Y)**2 + (X - 1)**2)**0.25

        Z = self.expression.execute_l([X, Y])

        print(Z)

        #
        fig, ax = plt.subplots()
        center = []
        m_patches = []
        verts = []
        N = self.count_of_vertex
        for i in range(len(self.result["i"])):
            polygon = Polygon(np.array(self.result["xk"][i]), True)
            # count center...
            j = 1
            center = self.result["xk"][i][0].copy()
            while j < N:
                center = self.sum(center, self.result["xk"][i][j])
                j += 1
            center = self.mul(center, 1.0 / float(N))
            # print("Center is", center)
            verts.append((center[0], center[1]))
            # ax.text(center[0], center[1], "#"+str(self.result["i"][i])+" ", color="black", fontsize="10",
verticalalignment='bottom', horizontalalignment='right')
            # ...
            m_patches.append(polygon)

        verts.append((self.result["xk"][-1][0][0], self.result["xk"][-1][0][1]))
        i = 0
        while i < len(self.pcd["xk"]):
            verts.append((self.pcd["xk"][i][0], self.pcd["xk"][i][1]))
            ax.text(self.pcd["xk"][i][0], self.pcd["xk"][i][1], "#" + str(self.result["i"][i]) + " ",
color="green",
                    fontsize="10",
                    verticalalignment='bottom', horizontalalignment='right')
            i += 1

        path = Path(verts)
        patch = patches.PathPatch(path, facecolor='none', lw=0.5)
        ax.add_patch(patch)
        xs, ys = zip(*verts)
        # plt.plot(xs, ys, "x--", lw=1.5, color='black', ms=10)
        p = PatchCollection(m_patches, cmap=matplotlib.cm.jet, alpha=0.4, lw=1.0)

        colors = 100 * np.random.rand(len(m_patches))
        p.set_array(np.array(colors))

        ax.add_collection(p)


        #

        #plt.figure()
        #CS = plt.contour(X, Y, Z,
        #                 colors='k'  # negative contours will be dashed by default
        #                 )
        #plt.clabel(CS, fontsize=9, inline=1)
        #plt.title('Single color - negative contours dashed')

        #plt.figure()

        if self.condition.parameters["linear"]:
            NV = [-self.condition.parameters["a"], self.condition.parameters["b"]]
            BX = [0 - NV[0] * 2.0 - self.condition.parameters["c"],
                  -self.condition.parameters["a"] + NV[0] - self.condition.parameters["c"]]
            BY = [0 - NV[1] * 2.0, self.condition.parameters["b"] + NV[1]]

            plt.plot(BX, BY, color='r')
        else:
            circle2 = plt.Circle((self.condition.parameters["a"], self.condition.parameters["b"]),
self.condition.parameters["R1"], color='r', fill=False)
            circle3 = plt.Circle((self.condition.parameters["a"], self.condition.parameters["b"]),
                                 self.condition.parameters["R2"], color='r', fill=False)
            ax.add_artist(circle2)
            ax.add_artist(circle3)

        CS = plt.contour(X, Y, Z)
        plt.clabel(CS, inline=1, fontsize=10)
```

```
        plt.title("Path of simplex with "+str(N)+" vertexes")

        plt.show()
        pass

    def printresult(self):
        print('')
        print("Result:")
        print("Coefficients: alpha", self.cof["a"], "beta:", self.cof["b"], "gamma:",
              self.cof["g"], "sigma:", self.cof["s"])
        print("Accuracy:", self.accuracy)
        print("Side length:", self.simplex_border_length)
        for i in range(len(self.result["i"])):
            print("#" + str(i) + ":")
            print("itteration:", self.result["i"][i])
            print("x:", self.result["xk"][i])
            print("f(x):", self.result["fx"][i])
            print("action:", self.result["action"][i])
            print("----------------------------------------")
        print("Count of function calling:", self.result["f_call"])
        print("Count of reflection\'s:", self.result["action_count"]["reflection"][0], "; function calling:",
self.result["action_count"]["reflection"][1])
        print("Count of expansion\'s:", self.result["action_count"]["expansion"][0], "; function calling:",
self.result["action_count"]["expansion"][1])
        print("Count of compression\'s:", self.result["action_count"]["compression"][0], "; function
calling:", self.result["action_count"]["compression"][1])
        print("Count of reduction\'s:", self.result["action_count"]["reduction"][0], "; function calling:",
self.result["action_count"]["reduction"][1])
        sum_of_o_call = self.result["action_count"]["reflection"][0] +
self.result["action_count"]["expansion"][0] + self.result["action_count"]["compression"][0] +
self.result["action_count"]["reduction"][0]
        sum_of_f_call = self.result["action_count"]["reflection"][1] +
self.result["action_count"]["expansion"][1] + self.result["action_count"]["compression"][1] +
self.result["action_count"]["reduction"][1]
        print("Sum of operations calling:", sum_of_o_call)
        print("Sum of functions calling by operations calling:", sum_of_f_call)
        # self.result["action_count"]["compression"] += 1
        pass

    def printresult_m(self, result: type([])):
        j = 0
        while j < len(result):
            print('')
            print("----------------------------------------")
            print("Result:")
            i =len(result[j]["i"]) - 1
            print("Coefficients: alpha", result[j]["cof"]["a"],"beta:", result[j]["cof"]["b"], "gamma:",
result[j]["cof"]["g"],"sigma:", result[j]["cof"]["s"])
            print("Accuracy:", result[j]["cof"]["ac"])
            print("Side length:", result[j]["cof"]["l"])
            # print("#" + str(i) + ":")
            print("Itterations:", result[j]["i"][i])
            print("x:", result[j]["xk"][i])
            print("f(x):", result[j]["fx"][i])
            print("action:", result[j]["action"][i])
            print("----------------------------------------")
            print("Count of function calling:", result[j]["f_call"])
            print("Count of reflection\'s:", result[j]["action_count"]["reflection"][0], "; function
calling:", result[j]["action_count"]["reflection"][1])
            print("Count of expansion\'s:", result[j]["action_count"]["expansion"][0], "; function calling:",
result[j]["action_count"]["expansion"][1])
            print("Count of compression\'s:", result[j]["action_count"]["compression"][0], "; function
calling:", result[j]["action_count"]["compression"][1])
            print("Count of reduction\'s:", result[j]["action_count"]["reduction"][0], "; function calling:",
result[j]["action_count"]["reduction"][1])
            sum_of_o_call = result[j]["action_count"]["reflection"][0] +
result[j]["action_count"]["expansion"][0] + result[j]["action_count"]["compression"][0] +
result[j]["action_count"]["reduction"][0]
            sum_of_f_call = result[j]["action_count"]["reflection"][1] +
result[j]["action_count"]["expansion"][1] + result[j]["action_count"]["compression"][1] +
result[j]["action_count"]["reduction"][1]
            print("Sum of operations calling:", sum_of_o_call)
            print("Sum of functions calling by operations calling:", sum_of_f_call)
            # result[j]["action_count"]["compression"] += 1
            if self.condition.parameters["linear"]:
                # x = self.deepcopy(self.pcd["xk"])
                x = self.deepcopy(result[j]["pcd"]["xk"])
                costum = ""
                i = 0
                f = []
```

```
        while i < len(x):
            f.append(self.expression.execute_1(x[i]))
            i += 1
        if self.one_dir == "gss_f_call":
            costum = "Golden section method"
            self.gss.par_sort(x, f)
        elif self.one_dir == "dskp_f_call":
            costum = "DSK Paula"
            self.dskp.par_sort(x, f)
        print("Last point is:", x[0])
        print("Last value is:", f[0])
        print("Count of function calling by Sven method:", result[j]["pcd"]["sm_f_call"])
        print("Count of function calling by "+costum+":", result[j]["pcd"][self.one_dir])
        print("Lambda for Sven:", result[j]["pcd"]["d_lambda"])
    j += 1
pass
```

ЛІТЕРАТУРА

1. Spendley W., Hext G. R., Himsworth F. R. Sequential application of simplex designs in optimisation and evolutionary operation //Technometrics. – 1962. – Т. 4. – №. 4. – С. 441-461.
2. J. A. Nelder R. Mead A Simplex Method for Function Minimization – The Comp J.,1965–7:308–313
3. Himmelblau, David Mautner. Applied nonlinear programming. McGraw-Hill Companies, The Univ. of Texas, Austin, Tex., 1972.
4. Paviani D., Ph. D. Dissertation, A New Method for the Solution of the General Nonlinear Programming – The Univ. of Texas, Austin, Tex., 1969
5. WolframAlpha computational knowledge engine – [Електроний ресурс] –Режим доступу: https://www.wolframalpha.com