НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Курсова робота

із дисципліни «Дослідження операцій»

на тему:

«Метод Нелдера-Міда»

Студента групи КМ-82

Біцана І. А.

Керівник:

Старший викладач Ладогубець Т. С

Кількість балів:_____

Оцінка:_____

Київ — 2021

ЗМІСТ

ВСТУП 3

1 ОСНОВНА ЧАСТИНА 4

1.1 Постановка задачі 4

1.2 Теоретична частина 5

1.3 Практична частина 11

ВИСНОВКИ 26

ДОДАТОК А (код програми) 27

ДОДАТОК Б (знімки екрану) 32

СПИСОК ЛІТЕРАТУРИ 33

ВСТУП

Темою даної курсової роботи був обраний метод Нелдера-Міда (деформованого багатогранника) призначений для наближеного рішення безумовної n-мірної задачі нелінійного програмування виду: f (x1, x2, ..., xn). Цей метод відноситься до групи методів 0-го порядку (прямого пошуку), так як в ньому не використовуються похідні. Метод є розвитком симплексного методу Спендлі та інших.

1 ОСНОВНА ЧАСТИНА

1.1 Постановка задачі

Дослідити збіжність методу «Нелдера-Міда» при мінімізації степеневої функції $f_1(x) = (10(x_1 - x_2)^2 + (x_1 - 1)^2)^4$, де початкова точка $x^{(0)} = (-1{,}2; 0{,}0)$ в залежності від:

1. Розміру початкового симплексу.
2. Значень параметрів деформації та редукції багатогранника.
3. Модифікацій метода.

Використати метод Нелдера-Міда в якості метода спуску для умовної оптимізації в залежності від:

1. Розташування локального мінімума (всередині / поза допустимою областю).
2. Виду допустимої області (випукла / невипукла / з лінійними обмеженнями).
3. Виду метода одновимірного пошуку (ДСК-Пауелла або Золотого перетину).
4. Точності метода одновимірного пошуку.
5. Значення параметру в алгоритмі Свена.

Задачею оптимізації називається задача пошуку екстремуму функції, заданої на деякій множині. Як правило, під задачею оптимізації також мається на увазі пошук елементу x, при якому цільова функція $f_1(x)$ досягає екстремуму.

Для того, щоб конкретно поставити задачу оптимізації необхідно знати:

1. Допустиму множину X

2. Цільову функцію $f: \mathrm{X} \to R$
3. Критерій пошуку (максимум чи мінімум)

Тоді вирішити задачу $f(x) \to min$ означає одне з:

1. Показати, що $\mathrm{X} \neq \emptyset$
2. Показати, що цільова функція не обмежена
3. Знайти x
4. Якщо не існує x, то знайти $\inf f(x)$

Якщо допустима множина X = G, то така задача називається задачею безумовної оптимізації, в протилежному випадку – задачею умовної оптимізації.

1.2 Теоретична частина

Метод Нелдера-Міда – метод оптимізації (пошуку мінімуму) функції від декількох змінних. Простий і в той же час ефективний метод, що дозволяє оптимізувати функції без використання градієнтів.

Алгоритм полягає в формуванні симплекса і подальшого його деформування в напрямку мінімуму, за допомогою трьох операцій:

1) Відображення;
2) Розтягування;
3) Стиснення;

Симплекс вдає із себе геометричну фігуру, яка є n - мірним узагальненням трикутника. Для одновимірного простору - це відрізок, для двовимірного - трикутник. Таким чином n - мірний симплекс має n + 1 вершину.

Алгоритм:

1) Нехай $f(x)$ функція, яку необхідно оптимізувати. На першому кроці вибираємо три випадкові точки і формуємо симплекс (трикутник). Обчислюємо значення функції в кожній точці: $f(V1), f(V2), f(V3)$. Сортуємо точки за значеннями функції в цих точках, таким чином отримуємо подвійну нерівність: $f(V2) \leq f(V1) \leq f(V3)$.

Оскільки виконується пошук мінімуму функції, то на даному кроці буде найкращою та точка, в якій значення функції мінімальне. Для спрощення точки позначаються наступним чином:

b = V2, g = V1, w = V3, де b – best (найкраща точка), g – good (добра точка), w – wrong (погана точка) відповідно.

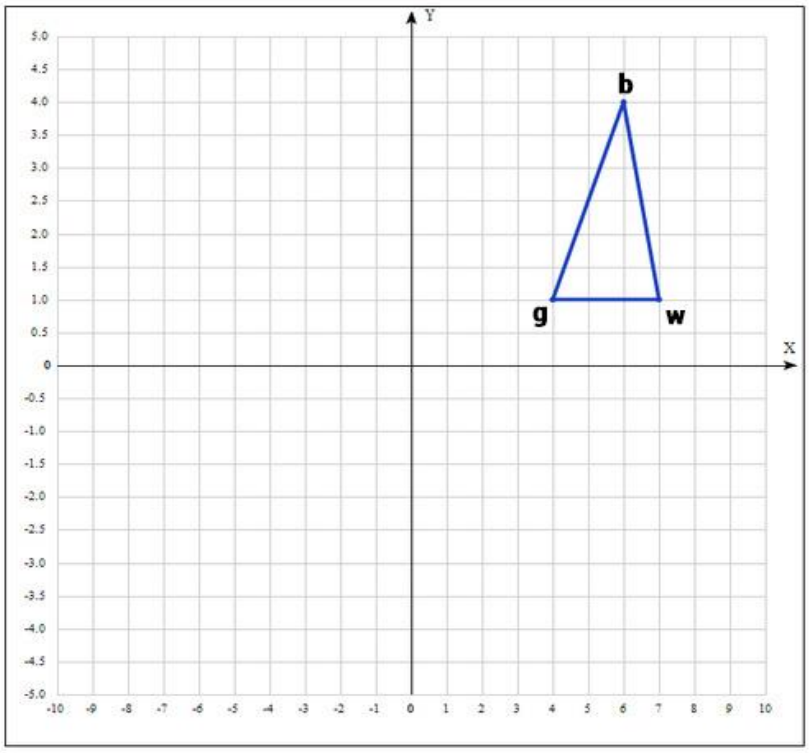


Рис. 1. Початкове положення точок

2) На наступному кроці знаходимо середину відрізка, точками якого є g і b. Оскільки координати середини відрізка рівні напівсумі координат його кінців, отримуємо:

$$mid = \left(\frac{x1 + x2}{2}; \frac{y1 + y2}{2}\right), \text{або}$$

$$mid = \frac{1}{n}\sum_{i=1}^{n} xi$$

3) Застосовуємо операцію відображення:

Знаходимо точку x_r наступним чином:

$$\text{Xr} = mid + \alpha\ (mid - w)$$

Тобто фактично відображуємо точку w відносно mid. Коефіцієнт α зазвичай приймають рівним 1. Після цього перевіряється наша точка:

Якщо $f(\text{Xr}) < f(g)$, то це добра точка.

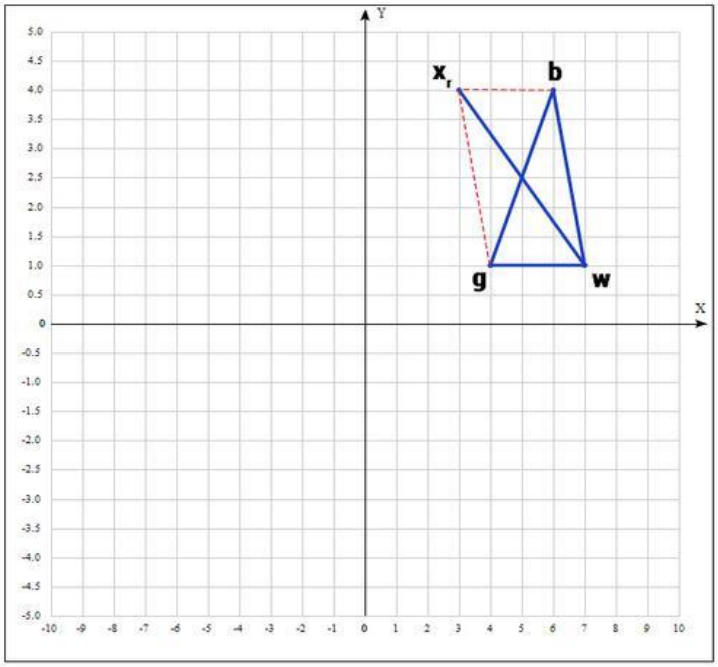


Рис. 2. Нова знайдена точка X_r при відображенні

4) Використовується операція розтягування:

Знаходиться точка x_e наступним чином:

$$\mathrm{Xe} = mid + \Upsilon\ (\mathrm{Xr} - mid)$$

В якості Υ приймаємо $\Upsilon = 2$, тобто відстань збільшуємо у 2 рази. Після цього перевіряємо точку x_e:

Якщо $f(\mathrm{Xe}) < f(b)$, то це означає, що знайдена краща точка, ніж та, яка є на даний момент. Якщо б цього не відбулось, то кращою залишалась би точка x_r. Далі точка w замінюється на x_e и в результаті отримується:

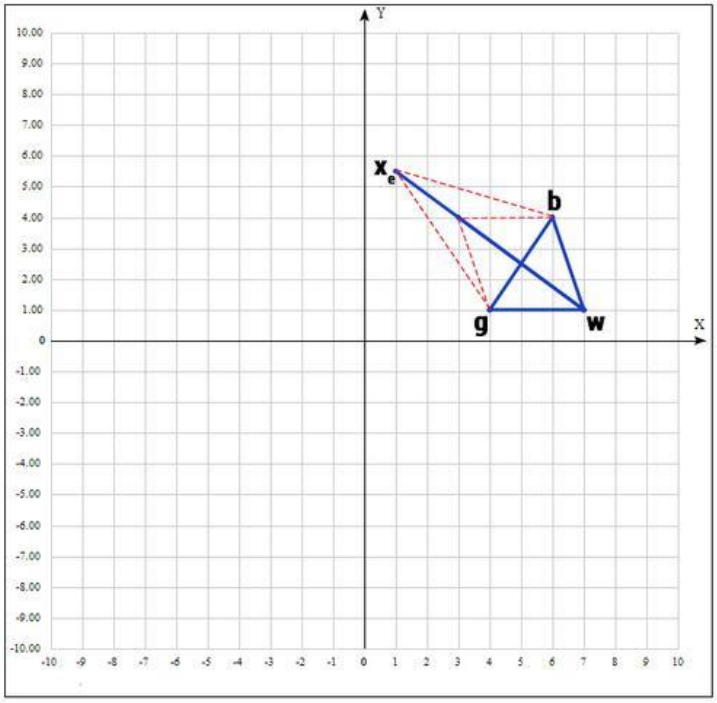


Рис.3. Нова знайдена точка x_e при розтягуванні

5) Якщо впродовж цих кроків не було знайдено хороших точок, то потрібно пробувати операцію стискання. Під час цієї операції зменшується відрізок і точки знаходяться всередині трикутника.

Знаходиться хороша точка x_c:

$$Xc = mid + \beta\ (w - mid)$$

Коефіцієнт β приймається рівним 0.5, тобто точка знаходиться на середині відрізка w… mid:

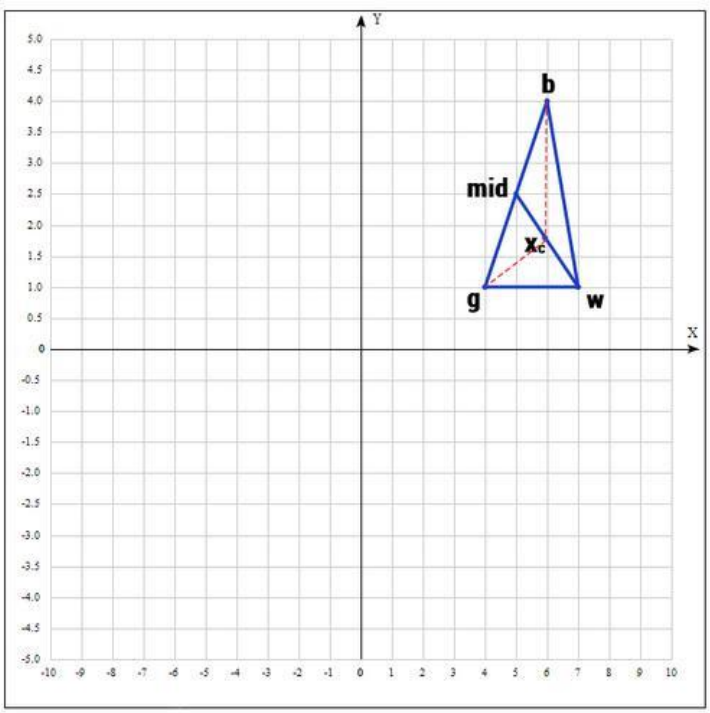


Рис. 4. Нова знайдена точка x_c при стисканні

Існує ще одна операція – скорочення. В даному випадку, ми перевизначаємо весь симплекс. Ми залишаємо тільки «кращу» точку, інші визначаємо наступним чином:

$Xj = b + \delta\ (Xj - b)$, де коефіцієнт δ = 0.5

Загалом, існуючі точки переміщуються по направленню до поточної «кращої» точки. Перетворення виглядає наступним чином:

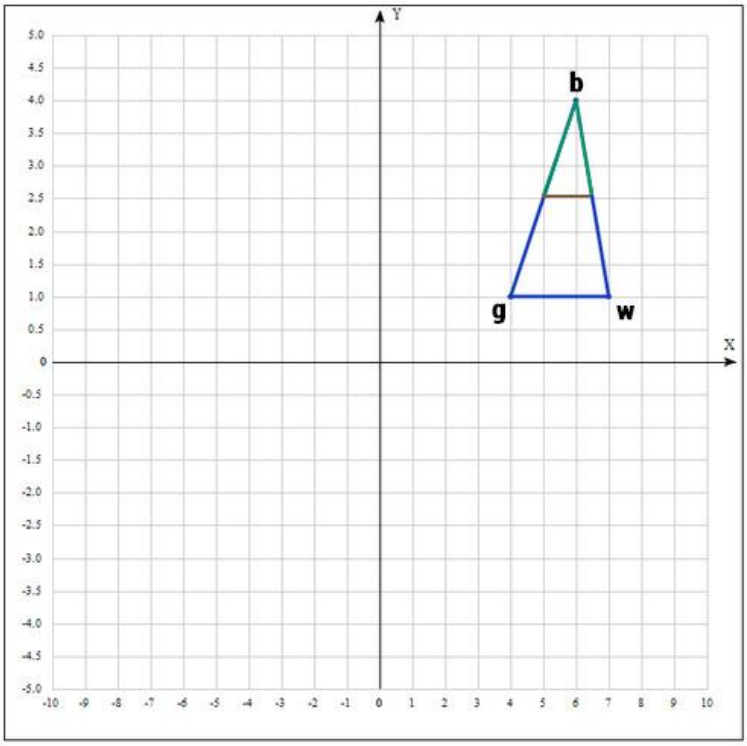


Рис. 5. Зміщення до поточної «кращої» точки

Необхідно відмітити, що дана операція дорого обходиться, оскільки необхідно заміняти точки в симплексі, тому ця операція дуже рідко зустрічається на практиці.

Алгоритм закінчується, коли:

1) Було виконано необхідну кількість ітерацій;
2) Площа симплекса досягла певної величини;
3) Поточне краще рішення досягло необхідної точності.

Перевагами даного методу є те, що він являється дуже ефективним алгоритмом пошуку екстремуму функції багатьох змінних, не накладаючим обмеження на гладкість функції. На кожній ітерації алгоритму проводиться як правило одне-два обчислення значень функції, що надзвичайно ефективно якщо ці обчислення дуже повільні. Крім того даний алгоритм простий в реалізації.

Недоліками даного методу являється неможливість відрізнити локальний максимум від глобального, а також відсутність теорії збіжності і наявність прикладів, коли метод розходиться навіть на гладких функціях.

1.3 Практична частина

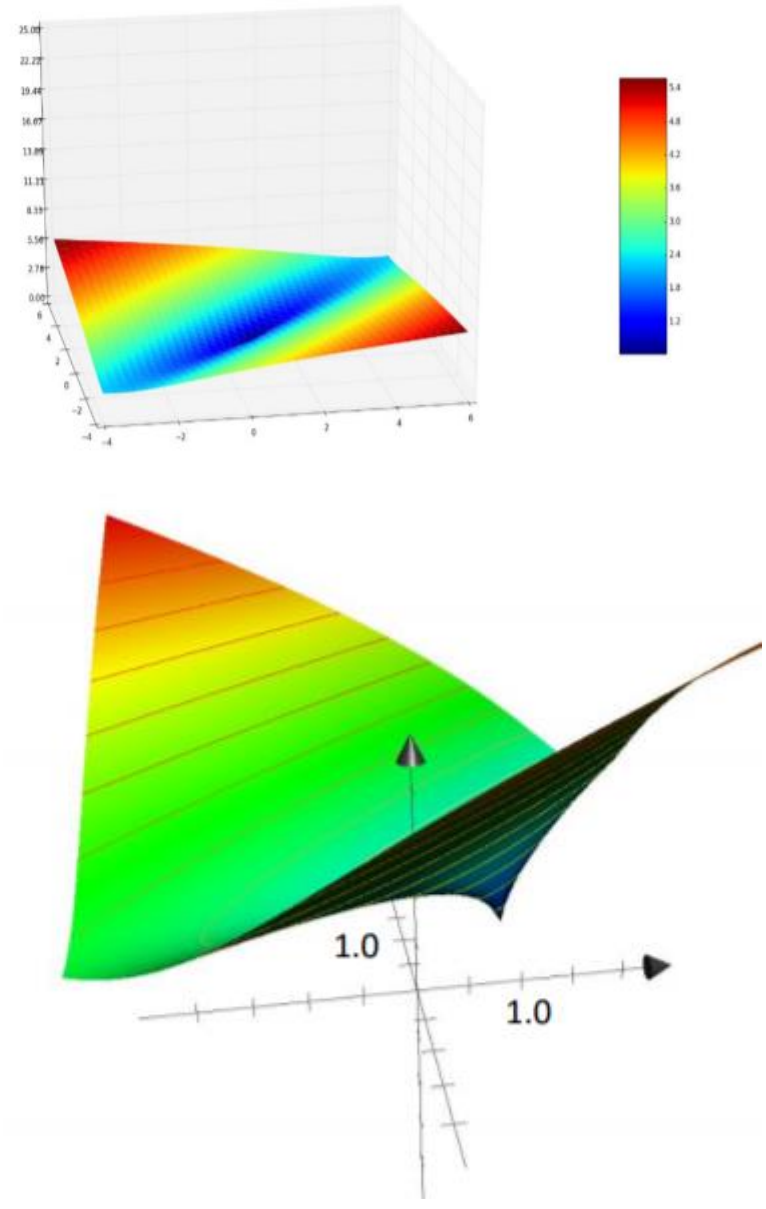


Рис.6. Графіки кореневої ф-ції

Для даної степеневої функції: $f_1(x) = (10(x_1 - x_2)^2 + (x_1 - 1)^2)^4$ було проведено ряд експериментів і організовано можливість заповнення початкових векторів введенням самостійно.

Для демонстрації і аналізу результату було обрано три вектори v1, v2, v3, які мають координати (0, 0), (1, 0) та (0, 1) відповідно. Дійсно, дуже маленький початковий симплекс може привести до локального пошуку. Таким чином, цей симплекс повинен залежати від характеру проблеми.

В результаті експерименту було отримано найкращу точку відносно цих 10 ітерацій та значення вказаної функції в цій точці (Табл. 1).

Таблиця 1. **Результати експерименту**

Wrong	Good	Best
f(0;1) = 14641	f(1;0) = 10000	f(0;0) = 1
f(0.375;0.25) = 0.08944	f(1;0) = 10000	f(0;0) = 1
f(0.390625; 0.09375) = 2.4624548	f(0;0) = 1	f(0.375; 0.25) = 0.08944
f(0.23828; 0.1172) = 0.279116	f(0;0) = 1	f(0.375; 0.25) = 0.08944
f(0.383301; 0.22949) = 0.1448196	f(0.23828; 0.1172) = 0.279116	f(0.375; 0.25) = 0.08944
f(0.660889; 0.484863) = 0.0325781	f(0.383301; 0.22949) = 0.1448196	f(0.375; 0.25) = 0.08944
f(0.787231; 0.643311) = 0.00405859	f(0.375; 0.25) = 0.08944	f(0.660889; 0.484863) = 0.0325781
f(1.0731201; 0.8781738) = 0.0220591	f(0.660889; 0.484863) = 0.0325781	f(0.787231; 0.643311) = 0.00405859
f(1.199463; 1.03662109) = 0.0096491	f(1.0731201; 0.8781738) = 0.0220591	f(0.787231; 0.643311) = 0.00405859

f(0.8338013; 0.763549) = $3.51068*10^{-5}$	f(1.199463; 1.03662109) = 0.0086491	f(0.787231; 0.643311) = 0.00405859
f(0.9077529; 0.7867279) = 0.0005769	f(0.787231; 0.643311) = 0.00405859	f(0.8338013; 0.763549) = $3.51068*10^{-5}$

Як видно з таблиці, найкраща точка має координати (0.8338013; 0.763549) та значення заданої функції в цій точці дорівнює $3.51068*10^{-5}$.

Аналітично, мінімум даної функції досягається в точці з координатами (1;1) і значення функції в цій точці буде дорівнювати 0. Знайдена найкраща точка при 10 ітераціях наближена до точки (1;1), тому можна зробити висновок, що підрахунки були вірними.

На рисунку (Рис. 6) зображено всі точки, які були знайдені впродовж обчислення та обведено найкращу знайдену точку, в якій значення функції є мінімальним:

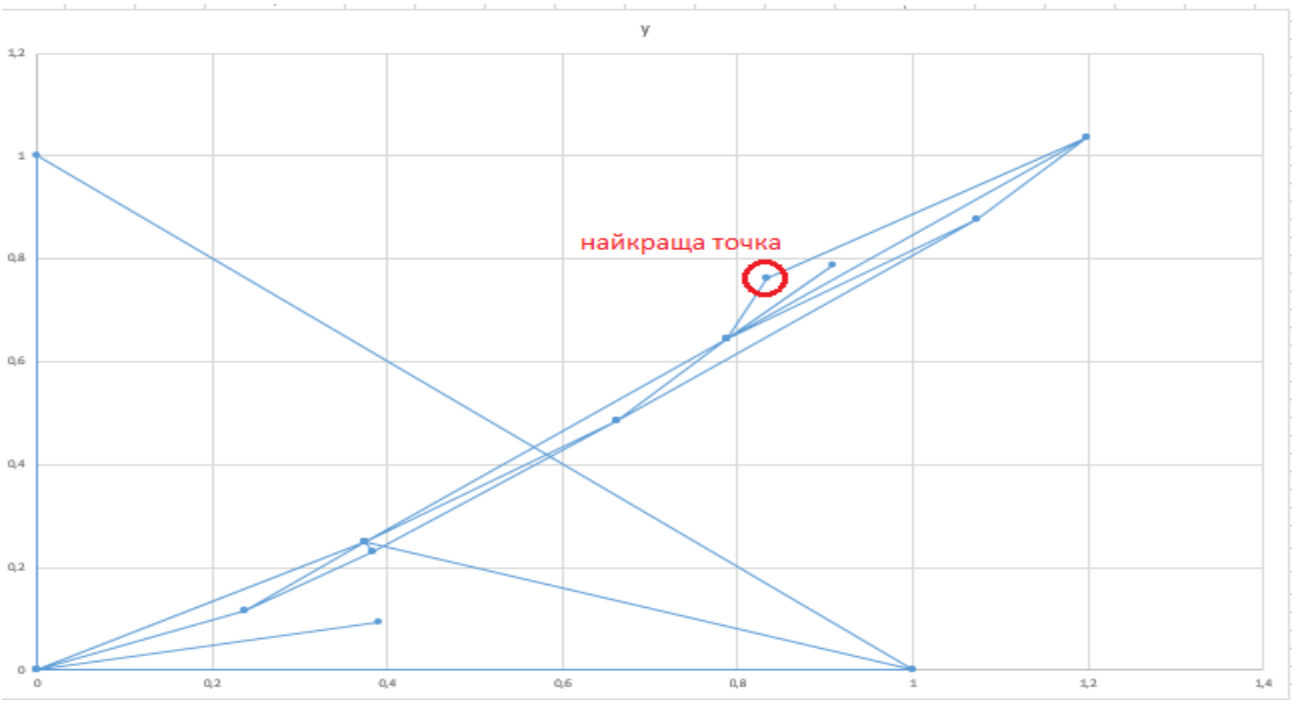


Рис. 7. Знайдені точки під час експерименту

З даних значень можна зробити висновок, що при збільшенні ітерацій даного методу точність знаходження мінімуму буде більшою, оскільки з

кожною наступною ітерацією значення заданої функції буде все більше наближатись до 0, а координати точки, в якій функція набуває мінімального значення, будуть наближатись до координат (1;1).

Результати виконання програми для різних параметрів ε

Для даного набору параметрів виконаємо аналіз оптимального значення точності ε, яке використовується у критерії зупинки. Занесемо результати роботи до таблиці:

Точність ε	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
10^{-1}	12	0.2488873943360235, 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

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 точку 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.

Результати виконання програми для різних коефіцієнтів віддзеркалення α

Почнемо з α, яке не повинно перевищувати 1, а ось зробити меншим може бути доцільним.

Значення α	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
1.0	203	1.0032325114580896, 0.8856881978791556	0.6096899312083286
0.9	155	0.4163476017090088 6, 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

Коефіцієнт α використовується на кроці віддзеркалення, він також є пошуковим кроком, і від нього залежать наступні кроки, приймаючи не оптимальне значення він викликає редукцію. 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.

Результати виконання програми для різних коефіцієнтів стиснення β

β не повинно перевищувати 0.6, але модифікації спробуємо почати з 0.9

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, 0.173342225348661O6	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

Результати виконання програми для різних коефіцієнтів розтягнення γ

Значення γ	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
3.0	235	0.9999999837685604, 0.9999999805706363	0.00013828949819040447
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

Результати виконання програми для різних коефіцієнтів редукції σ

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

Результати виконання програми для різних значень параметра сторони симплекса t

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

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.

Умовна оптимізація

Наведемо початкові умови для умовної оптимізації. $x_0 = (-1.2; 0.0)$, $\alpha = 1$, $\beta = 0.3$, $\gamma = 3$, $\sigma = 0.4$,$t = 2$, $\varepsilon = 10^{-5}$;

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: $\sqrt{((x_1{}^k - a)^2 + (x_2{}^k - b)^2)} \leq \sqrt{R}$ Ліва частина є формулою знаходження відстані між точками. В даному випадку між центром кола $(a; b)$ та деякою вершиною $x_1{}^k$, на k-ій 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:

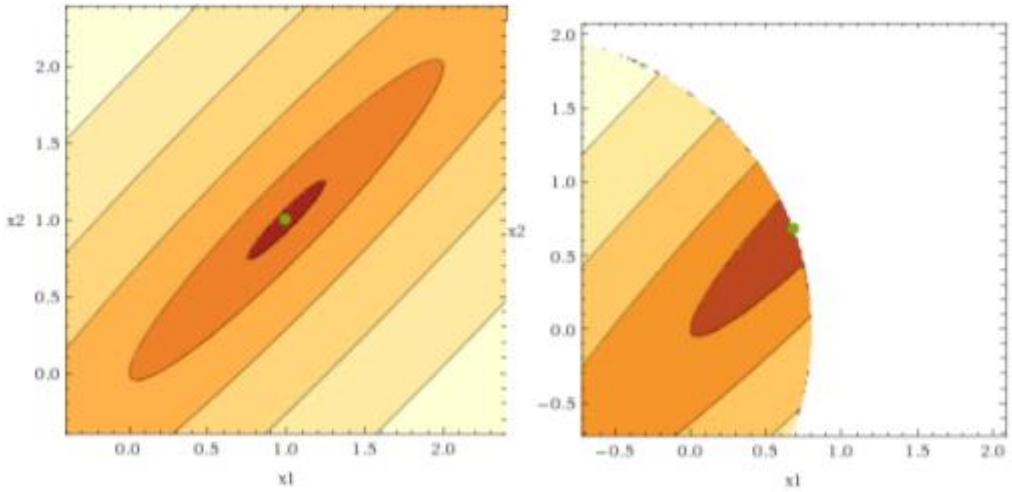
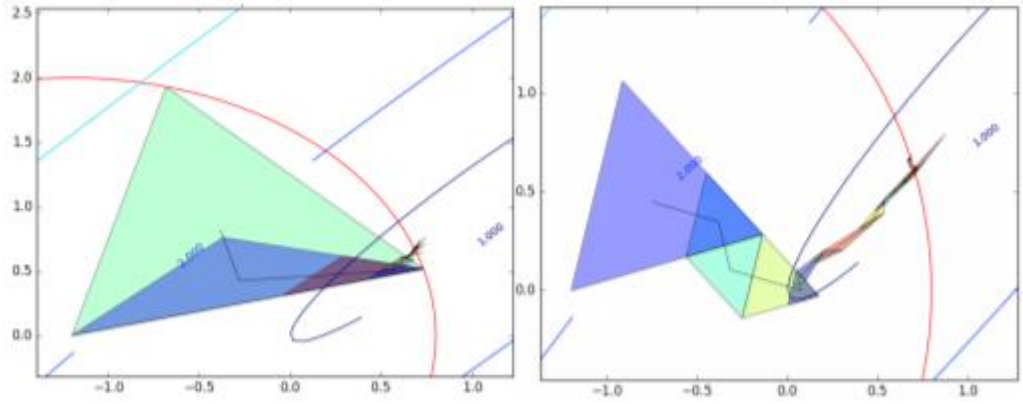


Рис. 8. Порівння на графіках глобального мінімуму та на заданій області

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



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.

Результати виконання для невипуклої допустимої області

Задамо умову в якості нерівностей:

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

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

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

Ліва частина є формулою знаходження відстані між точками. В даному випадку між центром кола $(a;\ b)$ та деякою вершиною $x_1{}^k$, на -й ітерації. Розглянемо роботу алгоритму з умовою

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

де центром кола виступає початкова точка x_0 = (−1.2; 0.0), а радіус 1.5 та 1

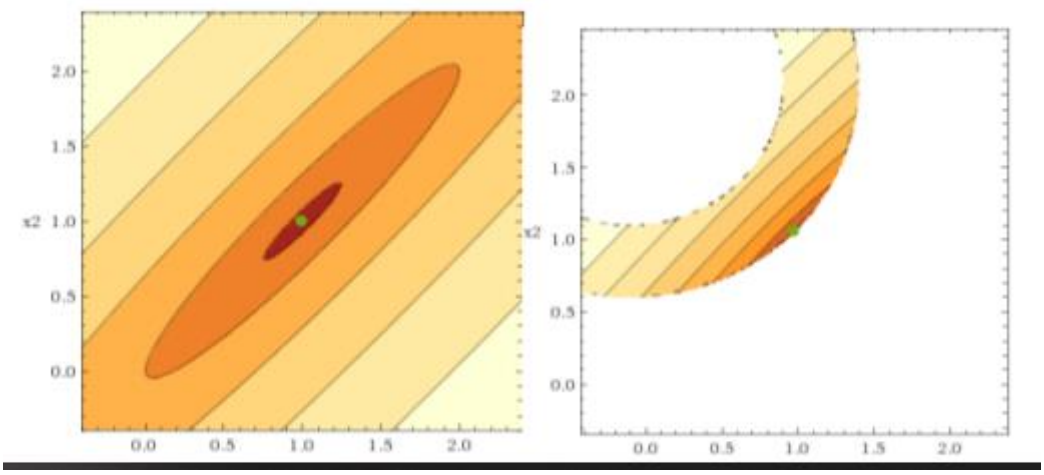


Рис. 9 Порівняння на графіках глобального мінімуму та на заданій області

Використовуючи здобуті на етапі безумовної оптимізації параметри, перевіримо роботу алгоритму для заданих умов, і занесемо 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

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

$\Delta\lambda = \|x^k\|/\|S\|$, 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 $x_1 + x_2 \leq 0.5$ Початкова точка $x\ 0 = (-1.2;\ 0.0)$. Порівняємо графіки з глобальним мінімумом та мінімумом в заданій області:

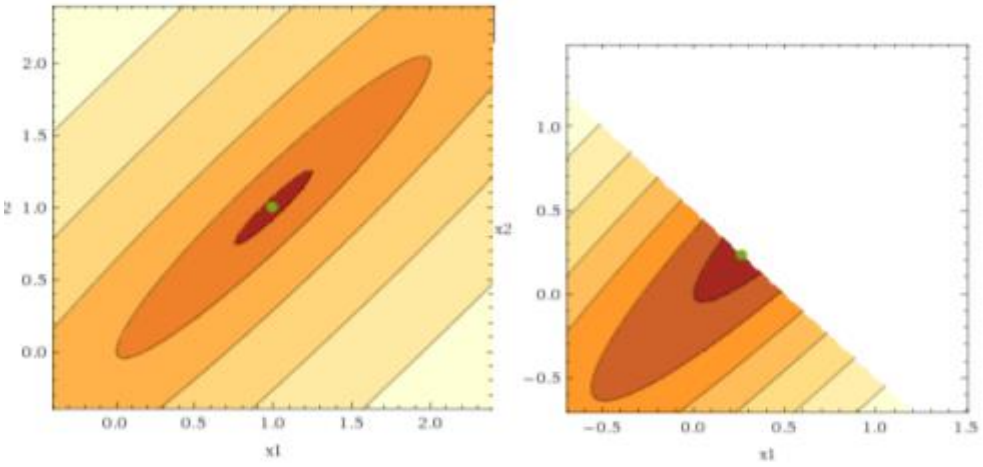


Рис.10 Порівння на графіках глобального мінімуму та на заданій області

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

Метод Золотого січення $\Delta\lambda$ модифікована

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

Точність ε	Кількість обчислень $f(x)$	Значення x	Значення $f(x)$
10^{-1}	25	0.0856867811713, 0.0615325773594	0.9578612551169329
10^{-2}	52	0.2112849064576078, 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	0.2682926901408861, 0.23170728647714928	0.8606957703130248
10^{-8}	336	0.2682926715693519, 0.23170730504868348	0.8606957703130191
10^{-9}	483	0.26829134299263524, 0.23170591207631108	0.8606965511212493
10^{-10}	512	0.26829136581178836, 0.2317059299206823	0.8606965394549523

Метод Метод ДСК Пауела $\Delta\lambda$ модифікована

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

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

ВИСНОВКИ

Під час виконання поставленої задачі було створено програмне забезпечення, яке реалізовує заданий алгоритм відповідно вибраного методу. Мету даної курсової роботи було досягнуто, оскільки програмне забезпечення реалізовує алгоритм та знаходить правильне рішення. Результати обчислень вказані в таблицях. З даних значень можна зробити висновок, що при збільшенні ітерацій даного методу точність знаходження мінімуму буде більшою, оскільки з кожною наступною ітерацією значення заданої функції буде все більше наближатись до 0, а координати точки, в якій функція набуває мінімального значення, будуть наближатись до координат (1;1).

В даній роботі було розглянуто метод Нелдера-Міда або метод деформованого многограника. Проведено роботу з різними значеннями коефіцієнтів. Прямої залежності кількості обрахувань цільової функції та параметрів немає, коригування 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 (код програми)

```
def function (x1, x2):
    function = (10*(x1 - x2)**2 + (x1 - 1)**2)**(1/4)
    return function


def sort(sortf):
    for row in sortf:
        sortf.sort(key=lambda row: row[2])
    return sortf


best = []
good = []
worst = []
sortf = []
i = 0


point1 = [int(input("x1 = ")), int(input("y1 = "))]
point1.append(function(point1[0], point1[1]))
point2 = [int(input("x2 = ")), int(input("y2 = "))]
point2.append(function(point2[0], point2[1]))
point3 = [int(input("x3 = ")), int(input("y3 = "))]
point3.append(function(point3[0], point3[1]))
count_iter_func = 3
```

```
print("Перша точка: [", point1[0], ",", point1[1], "] із значенням в ній", point1[2])
print("Друга точка: [", point2[0], ",", point2[1], "] із значенням в ній", point2[2])
print("Третя точка: [", point3[0], ",", point3[1], "] із значенням в ній", point3[2])

sortf.append(point1)
sortf.append(point2)
sortf.append(point3)

sort(sortf)

best = sortf[0]
good = sortf[1]
worst = sortf[2]

while (i < 40):
    print()
    print("Ітерація №", i + 1)
    print("Наразі:")
    print("best = [", best[0], ",", best[1], "] із значенням", best[2])
    print("good = [", good[0], ",", good[1], "] із значенням", good[2])
    print("worst = [", worst[0], ",", worst[1], "] із значенням", worst[2])
```

```
    point_c = [(good[0] + best[0])/2, (good[1] + best[1])/2]
    print("Середина відрізку навпроти найгіршої точки:", point_c)

    point_r = [2*point_c[0] - worst[0], 2*point_c[1] - worst[1]]
    point_r.append(function(point_r[0], point_r[1]))
    count_iter_func += 1
    print("Віддзеркалене значення відносно середини відрізку: [", point_r[0], ",", point_r[1], "] із значенням в ній", point_r[2])

    if (point_r[2] < best[2]):
        point_e = [2*point_r[0] - point_c[0], 2*point_r[1] - point_c[1]]
        point_e.append(function(point_e[0], point_e[1]))
        count_iter_func += 1
        if (point_e[2] < best[2]):
            worst = point_e
        else:
            worst = point_r
    if (best[2] < point_r[2] < good[2]):
        worst = point_r
    if (worst[2] > point_r[2] > good[2]):
        worst_prom = worst
        worst = point_r
        point_r = worst_prom
```

```
    point_s = [0.5*worst[0] + 0.5*point_c[0], 0.5*worst[1] + 0.5*point_c[1]]
    point_s.append(function(point_s[0], point_s[1]))
    count_iter_func += 1
    print(point_s)
    if (point_s[2] < worst[2]):
        worst = point_s
    if (point_s[2] > worst[2]):
        good = [good[0] + (best[0] - good[0])/2, good[1] + (best[1] - good[1])/2]
        worst = [worst[0] + (best[0] - worst[0])/2, worst[1] + (best[1] - worst[1])/2]
elif (point_r[2] > worst[2]):
    point_s = [0.5 * worst[0] + 0.5 * point_c[0], 0.5 * worst[1] + 0.5 * point_c[1]]
    point_s.append(function(point_s[0], point_s[1]))
    count_iter_func += 1
    print(point_s)
    if (point_s[2] < worst[2]):
        worst = point_s
    if (point_s[2] > worst[2]):
        good = [good[0] + (best[0] - good[0]) / 2, good[1] + (best[1] - good[1]) / 2]
        worst = [worst[0] + (best[0] - worst[0]) / 2, worst[1] + (best[1] - worst[1]) / 2]

sortf = []
sortf.append(best)
```

```
    sortf.append(good)
    sortf.append(worst)


    sort(sortf)
    print(sortf)


    best = sortf[0]
    good = sortf[1]
    worst = sortf[2]
    i = i + 1
    print("Кількість разів обрахування функції для ітерації №", i, "дорівнює",
count_iter_func)
```

ДОДАТОК Б (знімки екрану)

Результати, отримані при обчисленні.

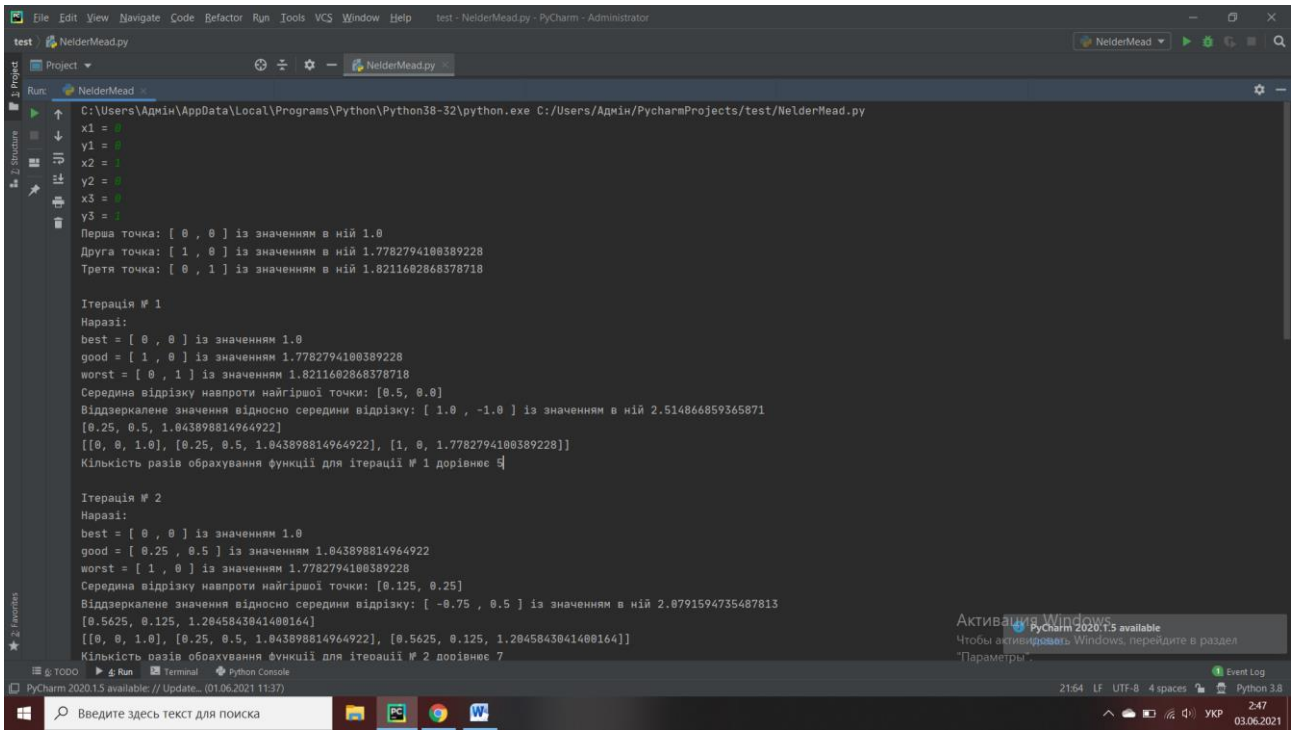


Рисунок Б.1.1 – перший рисунок додатка Б

СПИСОК ЛІТЕРАТУРИ

1. Дослідження операцій. Рекомендації до виконання курсової роботи [Електронний ресурс]: навч. посіб. для студ. спеціальності 113 «Прикладна математика», спеціалізації «Наука про дані та математичне моделювання» / Т. С. Ладогубець, О. Д. Фіногенов; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 180 Кбайт). – Київ: КПІ ім. Ігоря Сікорського, 2019. – 34 с.
2. Bürmen, A., Puhan, J., and Tuma, T. (2006), “Grid Restrained Nelder-Mead Algorithm”, *Comput. Optim. Appl.* **34**, pp. 359–375.
3. А.Г.Трифонов. "Постановка задачи оптимизации и численные методы ее решения" [електронний ресурс] URL: http://matlab.exponenta.ru/optimiz/book_2/2_1.php
4. Химмельблау Д. Прикладное нелинейное программирование. М.: Мир, 1983г.
5. https://habr.com/ru/post/332092/
6. https://buildwiki.ru/wiki/Nelder%E2%80%93Mead_method