НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ

ІНСТИТУТ КІБЕРНЕТИКИ ІМЕНІ В.М. ГЛУШКОВА

Кваліфікаційна наукова
праця на правах рукопису

Стовба Віктор Олександрович

УДК 519.8

ДИСЕРТАЦІЯ

СУБГРАДІЄНТНИЙ МЕТОД З КРОКОМ ПОЛЯКА У ПЕРЕТВОРЕНОМУ ПРОСТОРІ

113 – «Прикладна математика»

Галузь знань 11 – «Математика та статистика»

Подається на здобуття наукового ступеня доктора філософії.

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

В.О. Стовба

Науковий керівник: Стецюк Петро Іванович
доктор фізико-математичних наук,
старший науковий співробітник

Київ – 2020

АНОТАЦІЯ

Стовба В.О. Субградієнтний метод з кроком Поляка у перетвореному просторі. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття ступеня доктора філософії за спеціальністю 113 Прикладна математика. – Інститут кібернетики імені В.М. Глушкова Національної академії наук України, Київ. – 2020.

Зміст дисертації. У **вступі** обґрунтовано актуальність теми, сформульовано мету та задачі досліджень, розкрито наукову новизну та практичну цінність роботи, представлено її загальну характеристику.

У **розділі 1** розглянуто основні етапи розвитку методів негладкої оптимізації, зокрема методи з розтягом простору та методи феєрівського типу, а також їхнє застосування. Дано коротку характеристику сучасного стану розвитку методів оптимізації та наведено низку публікацій авторів, що його описують. Проаналізовано роботи вітчизняних та закордонних вчених: Гаснікова О.В., Гершовича В.І., Гольштейна Є.Г., Даніліна Ю.М., Дем'янова В.Ф., Євтушенка Ю.Г., Єрмольєва Ю.М., Єрьоміна І.І., Жадана В.Г., Журбенка М.Г., Зоркальцева В.І., Неміровського А.С., Нестерова Ю.Є., Нурмінського Є.О., Поляка Б.Т., Пшеничного Б.Н., Ржевського С.В., Скокова В.А., Стеценка С.І., Стецюка П.І., Федорова В.В., Шора Н.З., Щепакіна М.Б., Юдіна Д.Б., Agmon S., Boyd S., Fletcher R., Harchaoui Z., Hestenes M., Karzan F., Kelley J., König H., Motzkin T., Pallaschke D., Polak E., Reeves C., Schoenberg I., Shrader R.. У роботах вищенаведених авторів викладено загальновідомі градієнтні та ε-субградієнтні методи й алгоритми для різних задач оптимізації, аспекти чисельного та експериментального аналізу алгоритмів та їхнє застосування до широкого спектру прикладних задач, а також новітні алгоритми та модифікації, розроблені впродовж останніх років.

Викладено теоретичні основи низки відомих методів негладкої оптимізації, описано їхній розвиток та покращення різними авторами. На основі проведеного огляду зроблено висновки та поставлено задачі для дослідження.

У **розділі 2** розглянуто класичний субградієнтний метод з кроком Поляка та його модифікацію для знаходження точки мінімуму яружних опуклих функцій з відомим оптимальним значенням. Для класичного субградієнтного методу з кроком Поляка наведено обґрунтування монотонності зменшення відстані до точки мінімуму та швидкості збіжності для двох типів функцій: для довільної опуклої функції швидкість рівна $O\left(1/\sqrt{k}\right)$ (тут k – кількість ітерацій), для опуклої функції з гострим мінімумом – швидкості геометричної прогресії. Класичний субградієнтний метод з кроком Поляка удосконалено скалярним параметром $m>1$, який являє собою максимальне зміщення по опуклості функції, яке не відкидає локалізацію точки x^*. Цей параметр дає змогу врахувати деякі спеціальні класи опуклих функцій, наприклад, квадратичні гладкі функції, диференційовні однорідні з показником σ тощо. Для модифікованого параметром $m>1$ методу обґрунтовано монотонність зменшення відстані до точки мінімуму та швидкість збіжності, яка рівна $O\left(1/\sqrt{k}\right)$ при мінімізації довільної опуклої функції та швидкості геометричної прогресії у випадку мінімізації опуклої функції з гострим мінімумом. Проведено низку обчислювальних експериментів з мінімізації гладких та негладких яружних опуклих функцій з використанням Octave-програми *PolyakA*, яка реалізує класичний субградієнтний метод з кроком Поляка та його модифікований параметром $m>1$ варіант. Показано, що класичний субградієнтний метод з кроком Поляка потребує значної кількості ітерацій для знаходження точки мінімуму яружної опуклої функції. Продемонстровано, що модифікований параметром $m>1$ метод працює суттєво швидше, ніж класичний метод, тому може успішно застосовуватись для мінімізації спеціальних класів опуклих функцій.

У **розділі 3** розглянуто субградієнтний метод з кроком Поляка в перетвореному просторі змінних для знаходження точки мінімуму яружних опуклих функцій з відомим оптимальним значенням. Наведено схему модифікації класичного субградієнтного методу з кроком Поляка лінійним перетворенням простору, яке виконується з використанням невиродженої матриці B. Обґрунтовано монотонність зменшення відстані до точки мінімуму та швидкість збіжності модифікованого методу, яка рівна $O\left(1/\sqrt{k}\right)$ при мінімізації довільної опуклої функції та швидкості геометричної прогресії у випадку мінімізації опуклої функції з гострим мінімумом. Модифікований метод удосконалено скалярним параметром $m>1$ – максимальним зміщенням по опуклості, що дає змогу враховувати деякі спеціальні класи опуклих функцій. Обґрунтовано монотонність зменшення відстані до точки мінімуму та швидкість збіжності такого методу в випадку мінімізації довільних опуклих функцій та опуклих функцій з гострим мінімумом. Наведено Octave-реалізацію *PolyakB* субградієнтного методу з кроком Поляка в перетвореному просторі змінних і параметром $m>1$ та її детальний опис. Проведено низку обчислювальних експериментів з мінімізації гладких та негладких яружних опуклих функцій з використанням програми *PolyakB*. Показано, що субградієнтний метод з кроком Поляка, модифікований лінійним перетворенням простору та параметром $m>1$ потребує суттєво меншої кількості ітерацій, ніж класичний метод при мінімізації гладких та негладких яружних опуклих функцій, тому може успішно застосовуватись на практиці. Наведено чисельні результати обчислювальних експериментів, які підтверджують ефективність описаних методів.

У **розділі 4** досліджено низку субградієнтних методів з перетворенням простору та їхнє застосування для знаходження L_p-розв'язків систем лінійних рівнянь різного типу. Сформульовано задачу знаходження L_p-розв'язку системи лінійних рівнянь як задачу мінімізації опуклої функції. Розглянуто узагальнений метод еліпсоїдів та його загальна схема. Наведено опис двох

опуклих задач, які можна розв'язувати з його використанням. Розглянуто задачу визначення параметрів лінійної регресії у формі задачі мінімізації негладкої функції, що являє собою L_p-норму нев'язки системи лінійних рівнянь. Розроблено алгоритм методу еліпсоїдів, який дозволяє розв'язувати цю задачу для великих значень параметра $p \geq 1$. Проведено низку обчислювальних експериментів для трьох задач апроксимації спостережень, що містять аномалії, лінійною та квадратичною функціями за допомогою алгоритма на основі методу еліпсоїдів. Проведено порівняння результатів роботи запропонованого алгоритма з результатами класичних методів, що відповідають значенням параметра $p = 1, 2, \infty$. Розроблено схему зведення задачі апроксимації спостережень квадратичною функцією до задачі визначення параметрів лінійної регресії, що дає змогу розв'язувати задачі апроксимації в постановці останньої. Запропоновано алгоритм на основі методу еліпсоїдів для розв'язання задачі знаходження L_p-розв'язку системи лінійних рівнянь з двосторонніми обмеженнями на змінні: алгоритм Юдіна – Неміровського, який використовує H-форму методу еліпсоїдів. Наведено три функції, до мінімізації яких зводиться задача знаходження розв'язку сумісних систем лінійних рівнянь. Проведено низку обчислювальних експериментів для розв'язання цієї задачі в трьох постановках за допомогою класичного та модифікованого перетворенням простору та параметром $m > 1$ субградієнтного методу з кроком Поляка, а також amsg2p – субградієнтного методу, в якому перетворення простору виконується за допомогою двох останніх субградієнтів та агрегатного вектора. Результати експериментів показують, що запропоновані методи можна успішно застосовувати для розв'язання сумісних систем лінійних рівнянь.

У **розділі 5** розроблено програмні реалізації класичного та модифікованого перетворенням простору та параметром $m \geq 1$ субградієнтного методу з кроком Поляка мовою C++. Реалізовано процедури для обчислення значень функцій та їхніх субградієнтів у заданій точці для прикладів, що розглядались у розділах 2-4. Наведено допоміжні функції для запуску тестових

прикладів з указаних розділів та регулювання вхідних параметрів. Дано детальний опис усіх розроблених програмних функцій та вказівки щодо їхнього запуску.

Ключові слова: субградієнтний метод з кроком Поляка, перетворення простору, метод еліпсоїдів, лінійна регресія, L_p-розв'язок системи лінійних рівнянь.

ABSTRACT

Stovba V.O. Subgradient method with Polyak's step in transformed space. – Qualifying scientific work as a manuscript.

Dissertation for a Doctor of Philosophy Degree by specialty 113 Applied mathematics. – V.M. Glushkov Institute of Cybernetics of the National Academy of Science of Ukraine. – Kyiv, 2020.

The contents of the dissertation. In the introduction the relevance of the research topic is substantiated, the research purpose and tasks are formulated, the research scientific novelty and practical value are explained, and its general description is presented.

In **Chapter 1** the main stages of non-smooth optimization methods development are considered, including methods with space transformation and Fejer-type methods, and their applications. Given are brief description of the state of art of optimization methods development and set of authors' publications describing it. The works of the following domestic and foreign scientists are analyzed: Gasnikov O.V., Gershovich V.I., Holshtein E.G., Danilin Y.M., Demyanov V.F., Evtushenko Y.G., Ermoliev Y.M., Eremin I.I., Zhadan V.G., Zhurbenko M.G., Zorkaltsev V.I., Nemirovskii A.S., Nesterov Y.E., Nurminskii Y.O., Polak E., Polyak B.T., Pshenychny B.N., Rzhevskii S.V., Skokov V.A., Stetsenko S.I., Stetsyuk P.I., Fedorov V.V., Shor N.Z., Schepakin M.B., Yudin D.B., Agmon S., Boyd S., Fletcher R., Harchaoui Z., Hestenes M., Karzan F., Kelley J., König H., Motzkin T., Pallaschke D., Polak E., Reeves C., Schoenberg I., Shrader R.. In publications of the authors above well-known gradient and ε-subgradient methods and algorithms for various optimization problems are presented. Different aspects of numerical and experimental analysis of algorithms, their applications to wide range of applied problems, and modern algorithms and modifications developed within last years are given too.

Theoretical basis of some well-known non-smooth optimization methods is presented, their development and improvement by various authors is described. Based on conducted review conclusions are made and problems are set to be investigated.

In **Chapter 2** classical subgradient method with Polyak's step and its modification were considered for finding minimum point of ravine convex function with minimal value known. For classical subgradient method with Polyak's step monotony of distance decrease to the minimum point and convergence rate for two types of functions are given: for random convex function the rate equals $O\left(1/\sqrt{k}\right)$ (k – the number of iterations) and for convex function with acute minimum the rate equals geometrical progression rate. Classical subgradient method with Polyak's step is improved with scalar parameter $m>1$, which is maximal shift on convexity of the function to be minimized that does not reject localization of the minimum point. This parameter permits to take into account some special classes of convex functions, for example, quadratic smooth functions, differentiable homogeneous of degree σ, etc. For the method modified with parameter $m>1$ monotony of distance decrease to the minimum point and convergence rate are justified, which equals $O\left(1/\sqrt{k}\right)$ in case of random convex function minimization and geometrical progression rate if convex function with acute minimum is minimized. Conducted is set of computational experiments for minimization of smooth and non-smooth ravine convex functions using *PolyakA* Octave program, which implements classical subgradient method with Polyak's step and its modification with parameter $m>1$. It is shown that classical subgradient method with Polyak's step requires a great number of iterations for finding minimum point of ravine convex function. It is demonstrated that the method modified with parameter $m>1$ works significantly faster than classical method, so it can be used for minimization of special classes of convex functions successfully.

In **Chapter 3** subgradient method with Polyak's step in transformed space is considered for finding minimum point of convex function with minimal value known.

Presented is how classical subgradient method with Polyak's step can be improved using linear space transformation, which is implemented with nonsingular matrix B. Justified are monotony of distance decrease to the minimum point and convergence rate of the modified method, which equals $O\left(1/\sqrt{k}\right)$ in case of random convex function minimization and geometrical progression rate if convex function with acute minimum is minimized. Modified method is improved with scalar parameter $m>1$, which is maximal shift on convexity that permits to take into account some special classes of convex functions. Justified are monotony of distance decrease to the minimum point and convergence rate of the method in case of random convex functions and convex functions with acute minimum are minimized. Presented are Octave program *PolyakB* of subgradient method with Polyak's step and space transformation and parameter $m>1$, and its description in detail. Set of computational experiments for minimization of smooth and non-smooth ravine convex functions using *PolyakB* was conducted. It is shown that subgradient method with Polyak's step improved with space transformation and parameter $m>1$ requires significantly fewer number of iterations than classical method in case of smooth and non-smooth ravine convex function minimization, so it can be successfully used in practice. Presented are numerical computational experiments results, which confirm effectiveness of the methods described.

In **Chapter 4** set of subgradient methods with space transformation are investigated, as well as their application to finding L_p-solutions of linear equation systems. Presented is problem of L_p-solution finding of overdetermined linear equation system as convex function minimization problem. Considered is generalized ellipsoid method and its general scheme. Two convex problems are described, which can be solved using this method. Considered is linear regression parameters determination problem as non-smooth function minimization problem. The function is L_p-norm of linear equation system residual. Developed is the ellipsoid method algorithm that permits to solve this problem for big values of parameter $p\geq 1$.

Conducted is set of computational experiments for three problems of approximation of observations containing outliers with linear and quadratic functions using algorithm based on ellipsoid method. Conducted is work comparison of the algorithm presented and classical methods corresponding to parameter values $p = 1, 2, \infty$. Developed is reduction scheme of problem of observation approximation with quadratic function to linear regression parameters determination problem, which permits to solve approximation problems in terms of the last. Presented is the algorithm based on ellipsoid method for solving of the problem of L_p-solution finding of linear equation system with two-sided constraints on variables: Yudin – Nemirovskii algorithm that uses H-form of the ellipsoid method. Shown are three functions to minimization of which problem of finding of consistent linear equation system solution can be reduced. Set of computational experiments was conducted for solving this problem in three forms using classical and modified with space transformation and parameter $m > 1$ subgradient method with Polyak's step, and amsg2p – subgradient method, where space transformation is made using two last subgradients and aggregate vector. Numerical experiment results show that proposed methods can be successfully used for solving consistent linear equation systems.

In **Chapter 5** C++ program implementations of classical and modified with space transformation and parameter $m \geq 1$ subgradient method with Polyak's step were developed. Also procedures for calculation of objective function values and their subgradients at given point for examples from chapters 2, 3, and 4 are implemented. Additional functions for launching of test examples from chapters mentioned above and input parameters regulation are presented as well. Moreover, detailed description of all program functions developed and instructions for their launching are given.

Key words: subgradient method with Polyak's step, space transformation, ellipsoid method, linear regression, L_p-solution of linear equation system.

Список публікацій здобувача

Публікації, в яких опубліковано основні наукові результати дисертації

1. Стецюк, Петр, Виктор Стовба, и Игорь Мартынюк. 2017. «Алгоритм метода эллипсоидов для нахождения Lp-решения системы линейных уравнений». *Теорія оптимальних рішень* 139–46.
2. Стецюк, Петр, Виктор Стовба, и Александр Жмуд. 2018. «Метод эллипсоидов для нахождения решения переопределенной СЛАУ». *Теорія оптимальних рішень* (17):115-23.
3. Stetsyuk, Petro, Viktor Stovba, and Zhanna Chernousova. 2018. “Subgradient Method with Polyak’s Step in Transformed Space”. *Optimization and Applications. OPTIMA 2018. Communications in Computer and Information Science* 974:49-63.
4. Стовба, Віктор, Олександр Жмуд, та Олена Криворучко. 2019. «Експерименти з субградієнтними методами Поляка для розв’язування сумісних СЛАР». *Теорія оптимальних рішень* (18):81-7.
5. Стецюк, Петро, Віктор Стовба, та Олександр Жмуд. 2019. «Про швидкість збіжності субградієнтних методів з кроком Поляка». *Наук. вісник Ужгород. ун-ту. Сер. матем. і інформ* 1(34):94-101.
6. Стецюк, Петро, та Віктор Стовба. 2019. «Субградієнтні методи з кроком Поляка та програма amsg2p». У *Субградієнтні алгоритми та задачі на комбінаторних конфігураціях*. Київ: Унів. вид-во ПУЛЬСАРИ.
7. Стовба, Віктор. 2020. «Метод еліпсоїдів для знаходження параметрів лінійної регресії». *Cybernetics and Computer Technologies* (3):14-24.

Публікації, що засвідчують апробацію матеріалів дисертації

1. Стецюк, Петро, Віктор Стовба, та Ігор Мартинюк. 2016. «Octave-програма dist2p для розділення двох поліедрів». Матеріали XIII Міжнародної

науково-практичної конференції Теоретичні та прикладні аспекти побудови програмних систем (TAAPSD’2016), Київ, Грудень 5-9, 217-20.

2. Стецюк, Петр, Галина Била, и Виктор Стовба. 2017. «Метод эллипсоидов для нахождения L_p-решения системы линейных уравнений». Матеріали VIII Всеукр. наук-практ. конф. за міжнародною участю Інформатика та системні науки (ІСН-2017), Полтава, Березень 16-18, 258-64.
3. Стецюк, Петро, та Віктор Стовба. 2017. Метод еліпсоїдів для лінійної регресії. Матеріали IV Міжнародної науково-практичної конференції Обчислювальний інтелект (результати, проблеми, перспективи) (ComInt-2017), Київ, Травень 16-18, 314-5.
4. Stovba, Viktor, and Oleksandr Zhmud. 2018. “The ellipsoid method for linear regression”. Материалы 6-й международной научной конференции Математическое моделирование, оптимизация и информационные технологии (MMOTI-2018), Кишинэу, Молдова, Март 19-24, 206-8.
5. Стовба, Віктор, та Олександр Жмуд. 2019. «Субградієнтний метод Поляка у перетвореному просторі змінних». Матеріали IX міжнародної школи-семінару «Теорія прийняття рішень» у рамках Міжнародного наукового симпозіуму Інтелектуальні рішення (IntSol-2019), Ужгород, Квітень 15-20, 229-30.

ЗМІСТ

ВСТУП

Мінімізація негладких опуклих функцій та проблеми, що виникають при цьому, є важливою частиною алгоритмічних проблем оптимізації. Численні прикладні задачі потребують мінімізації негладких або близьких до негладких за поведінкою опуклих функцій: задачі планування, керування, проєктування складних технічних об’єктів, технічної та медичної діагностики, аналізу стійкості динамічних систем, пошуку раціональних стратегій в умовах невизначеності або конфлікту тощо. Такі задачі завжди володіють певним набором характеристик та особливостей, які необхідно враховувати при їхньому розв’язанні. Наприклад, в умовах інтенсивного зростання кількості інформації про середовище дедалі частіше задачі, що виникають, є задачами дуже великої розмірності. Ефективний аналіз цієї інформації дає можливість не лише суттєво спростити самі задачі, а й значно прискорити методи їхнього розв’язання.

Однією з важливих проблем оптимізації є мінімізація яружних функцій. Класичний градієнтний метод вкрай повільно збігається для функцій цього типу, а при збільшенні кількості змінних цільової функції така поведінка стає типовою. Для вирішення цієї проблеми побудовано багато методів та їхніх модифікацій. Наприклад, у методі ярів (метод оврагов – Gelfand, and Tsetlin 1962) напрямок наступної точки на k-й ітерації визначається вектором, що з’єднує точки x_k та x_{k+2}, отримані за допомогою градієнтного спуску. В цьому методі рух виконується вздовж яру та закінчується в заглибленні біля точки мінімуму, значення в якій краще уточнювати за допомогою інших методів. Метод ярів дає змогу знайти мінімум доволі складних функцій від 5-10 змінних, однак він потребує індивідуального підбору яружного кроку для кожної функції та необхідності вносити корективи під час роботи методу. Для функцій, близьких до квадратичної, ефективним є метод спряжених градієнтів (Hestenes, and Stiefel 1952), який у випадку квадратичної функції в $\mathbb{R}^n$ знаходить мінімум

не більше ніж за n кроків і збігається значно швидше, ніж градієнтний метод. Для цього методу існує також варіант для неквадратичних цільових функцій (Fletcher, and Reeves 1964).

Одним з ефективних підходів до мінімізації яружних функцій є застосування лінійного перетворення простору. Зокрема, в інституті кібернетики імені В.М. Глушкова НАН України під керівництвом академіка Н.З. Шора розроблено субградієнтні методи з розтягом простору в напрямку субградієнта (відомий метод еліпсоїдів є їхнім частинним випадком) або різниці двох послідовних субградієнтів (r-алгоритми). Операція розтягу простору в застосуванні до певних методів виявилась вкрай ефективною для мінімізації гладких та негладких яружних опуклих функцій, оскільки вона дає змогу вирівняти структуру поверхонь рівня функції. Наприклад, класичний феєрівський метод з перетворенням простору (Стецюк 1996) дозволяє ефективно мінімізувати як гладкі, так і негладкі функції з сильно витягнутими поверхнями рівня. Використання одного перетворення простору дає можливість отримати обчислювальну складність методу не вищу, ніж у класичного субградієнтного спуску. Однак проблема вибору структури матриці перетворення простору залишається відкритою, адже напрям перетворення обирається з врахуванням структури поверхонь рівня функції та напрямку їх витягнутості. Подальше покращення цього методу виконано в роботах Гершовича та Шора (1982) і Стецюка (2011б, 1997аб). Зокрема, в роботі (Стецюк 2011б) запропоновано метод amsg2p, який здійснює перетворення простору за допомогою двох останніх субградієнтів та агрегатного вектора. В роботі (Стецюк 2009) цей метод адаптований для довільного мінімального значення функції й показав суттєве прискорення в порівнянні з класичним феєрівським методом (Стецюк 2009, 2012а).

Наведені результати показують, що методи мінімізації яружних функцій активно розвиваються, що насамперед зумовлено їхніми прикладними застосуваннями. Ще однією важливою проблемою, окрім негладкості, що дедалі частіше виникає на практиці, є велика розмірність задач та її невпинне

зростання. Функцію, що залежить від мільйона змінних, з навіть не сильно витягнутими поверхнями рівня вкрай складно мінімізувати. Навіть найефективніші методи мінімізації яружних функцій у такому випадку працюють вкрай повільно. Також варто зазначити, що метод градієнтного спуску є найпоширенішим серед алгоритмів машинного навчання і використовується майже в кожній моделі навчання. Одна з модифікацій методу активно застосовується для навчання перцептрона та глибоких нейронних мереж і відома під назвою метод зворотного поширення помилки. Отже, розвиток методів градієнтного та субградієнтного типів для мінімізації яружних функцій, що залежать від великої кількості змінних, є **актуальною**.

Актуальність теми. Мінімізація негладких яружних функцій є доволі складною задачею для багатьох наявних методів. Наприклад, класичний субградієнтний метод з кроком Поляка для знаходження наближення до точки мінімуму функції, що залежить від десятків тисяч змінних та не є яружною потребує лише декількох десятків ітерацій. Якщо ж здійснити розтяг поверхонь рівня функції, що залежить від двох-трьох змінних, за декількома напрямками кількість ітерацій методу значно зросте. Отже, актуальною задачею є розробка методів, що використовують перетворення простору для покращення властивостей функції. Прикладами таких методів є r-алгоритми або метод amsg2p. Однак, для них немає ні обґрунтування збіжності, ні оцінок швидкості збіжності. Побудова та теоретичне обґрунтування методів мінімізації негладких яружних функцій є важливим напрямком досліджень сучасної оптимізації, що і визначає **актуальність завдань** дисертаційної роботи.

Мета й завдання дослідження. Метою роботи є розробка методів та алгоритмів негладкої оптимізації з перетворенням простору, обґрунтування їхньої збіжності та оцінок швидкості збіжності.

Для досягнення мети дослідження поставлено такі *завдання*:

- *розробити та обґрунтувати* модифікації субградієнтного методу з кроком Поляка у вихідному та перетвореному просторах змінних для мінімізації яружних опуклих функцій;

- *провести* обчислювальні експерименти з використанням розроблених модифікацій для оцінки ефективності мінімізації гладких та негладких яружних опуклих функцій;
- *розробити* програмні реалізації розроблених модифікацій та *провести* їхнє тестування;
- *дослідити* застосування розроблених модифікацій та алгоритмів для розв'язання прикладних задач.

Об'єкт дослідження – субградієнтні методи з перетворенням простору.

Предмет дослідження – субградієнтний метод з кроком Поляка та метод еліпсоїдів для мінімізації яружних функцій, розв'язання систем лінійних рівнянь та визначення параметрів лінійної регресії.

Методи дослідження. Зазначені задачі розв'язуються за допомогою засобів лінійної алгебри, математичного та функціонального аналізу.

Наукова новизна одержаних результатів. Наукову новизну в цій роботі мають такі теоретичні та практичні результати:

- *вперше* обґрунтовано монотонність зменшення відстані до точки мінімуму субградієнтного методу з кроком Поляка у перетвореному просторі змінних та скалярним параметром $m > 1$;
- *вперше* обґрунтовано швидкість збіжності субградієнтного методу з кроком Поляка у початковому та перетвореному просторах змінних і скалярним параметром $m > 1$ для довільних опуклих функцій;
- *вперше* обґрунтовано швидкість збіжності субградієнтного методу з кроком Поляка у початковому та перетвореному просторах змінних і скалярним параметром $m > 1$ для опуклих функцій з гострим мінімумом;
- *вперше* обґрунтовано швидкість збіжності субградієнтного методу з кроком Поляка у початковому та перетвореному просторах змінних і скалярним параметром $m > 1$ для спеціальних класів опуклих функцій;

- *вперше* розроблено програмну реалізацію субградієнтного методу з кроком Поляка у вихідному та перетвореному просторах змінних і скалярним параметром $m \geq 1$ мовою C++;
- *вперше* побудовано алгоритм методу еліпсоїдів для розв'язання задачі визначення параметрів лінійної регресії при *довільному* значенні параметра $p \geq 1$, який задає L_p-норму нев'язки системи лінійних рівнянь;
- *вперше* побудовано алгоритм на основі метода еліпсоїдів Юдіна-Неміровського для знаходження L_p-розв'язків систем лінійних рівнянь з двосторонніми обмеженнями на змінні.

Практичне значення отриманих результатів. Запропоновані в дисертації субградієнтні методи та алгоритми мають теоретичний та практичний характер для розвитку (суб)градієнтних методів і можуть бути використані для розв'язання задач обробки великих об'ємів даних, розробки методів машинного навчання, технологій штучного інтелекту тощо.

Особистий внесок здобувача. Автором самостійно отримано основні результати дисертаційного дослідження. В опублікованих в співавторстві наукових працях здобувачем здійснено: у публікації (Стецюк, Стовба, та Мартинюк 2016б) – тестові розрахунки з використанням програми dist2p; у статті (Стецюк, Стовба, и Мартынюк 2017а) – одержання формули для обчислення субградієнта цільової функції, проведення обчислювальних експериментів з використанням запропонованих алгоритмів та їхня інтерпретація; у статті (Стецюк, Стовба, и Жмуд 2018б) – опис двох опуклих задач, проведення обчислювальних експериментів та інтерпретація результатів з використанням методу еліпсоїдів; у статті (Стецюк, Стовба, та Жмуд 2019а) – доведення теорем про швидкість збіжності субградієнтного методу з кроком Поляка у перетвореному просторі змінних для довільної опуклої функції та опуклої функції з гострим мінімумом; у публікації (Стецюк, та Стовба 2019б) – проведення обчислювальних експериментів з використанням субградієнтного

методу з кроком Поляка у вихідному та перетвореному просторах змінних і методу amsg2p та інтерпретація результатів; у статті (Стовба, Жмуд, та Криворучко 2019б) – постановка задачі, проведення обчислювальних експериментів з використанням субградієнтного методу з кроком Поляка у вихідному та перетвореному просторах змінних та інтерпретація результатів; у публікації (Stetsyuk, Stovba, and Chernousova 2018) – проведення експериментів з мінімізації яружних функцій з використанням субградієнтного методу з кроком Поляка у вихідному та перетвореному просторах змінних.

Апробація результатів дисертації. Результати дисертації доповідались та обговорювались на:

- XIII Міжнародній науково-практичній конференції «Теоретичні та прикладні аспекти побудови програмних систем (TAAPSD'2016)», 5 – 9 грудня, 2016 року, м. Київ, Україна;
- VIII Всеукраїнській науково-практичній конференції «Інформатика та системні науки (ІСН-2017)», 16 – 18 березня, 2017 року, м. Полтава, Україна;
- IV Міжнародній науково-практичній конференції «Обчислювальний інтелект (результати, проблеми, перспективи) (ComInt-2017)», 16 – 18 травня, 2017 року, м. Київ, Україна;
- 6-й міжнародній науковій конференції «Математическое моделирование, оптимизация и информационные технологии (MMOTI-2018)», 12 – 16 листопада, 2018 року, м. Кишинів, Молдова;
- 9th International Conference "Optimization and Applications (OPTIMA 2018)", 1 – 5 жовтня, 2018 року, м. Петровац, Чорногорія;
- IX міжнародній школі-семінарі «Теорія прийняття рішень» в рамках Міжнародного наукового симпозіуму «Інтелектуальні рішення (IntSol-2019)», 15 – 20 квітня, 2019 року, м. Ужгород, Україна;
- International Conference "Optimization and Equilibrium Problems (ICOEP 2019)", 31 липня – 2 серпня, 2019 року, м. Дрезден, Німеччина;

- Norwegian-Eurasian Workshop “New Resilience Challenges in Ecological-Economic Problems at the Digital Era”, 22 – 27 вересня, 2019 року, м. Київ, Україна;
- засідання відділу методів негладкої оптимізації Інституту кібернетики імені В.М. Глушкова НАН України (керівник засідання – член-кореспондент НАН України О.М. Хіміч), 30 грудня 2020 року.

Публікації. Основні наукові результати дисертаційної роботи у повній мірі викладено в 12 публікаціях, з яких: 2 наукові статті опубліковано в фахових виданнях України; 1 статтю англійською мовою опубліковано в зарубіжному виданні, проіндексованому в наукометричній базі SCOPUS; 1 роботу опубліковано як розділ колективної монографії; 5 тез доповідей опубліковано в збірниках доповідей міжнародних наукових та науково-практичних конференцій і семінарів.

Структура та обсяг дисертації. Дисертаційна робота складається зі вступу, п’яти розділів, загальних висновків, списку використаних літературних джерел, який містить 131 найменування. Загальний обсяг дисертаційних досліджень викладено на 184 сторінках друкованого тексту, де обсяг основного тексту – 127 сторінок. Дисертація включає 8 рисунків, 22 таблиці та 5 додатків на 22 сторінках.

РОЗДІЛ 1. ОГЛЯД ЛІТЕРАТУРИ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1.1 Основні етапи розвитку методів негладкої оптимізації

На практиці дуже часто виникають ситуації, коли доводиться мінімізувати функції, які виявляються негладкими. Такі функції з'являються, наприклад, в задачах найкращого наближення, в статистичних задачах оцінки параметрів методом найменших модулів, в задачах пошуку мінімального дерева Штейнера тощо. Також часто спостерігається ситуація, в якій параметр, що оптимізується, недиференційовно залежить від інших параметрів задачі – така залежність часто буває кусочно-лінійною. Отже, розробка підходів та методів оптимізації недиференційовних функцій була і досі залишається актуальною та важливою проблемою оптимізації.

В загальному випадку мінімізація негладких функцій є вкрай складною задачею. Значення негладкої функції в конкретній точці дає суттєво менше інформації про її поведінку в інших точках як порівняти з диференційовною функцією. Для розв'язання цієї складної задачі в 60-тих роках XX століття був розроблений математичний апарат для роботи з опуклими функціями, які є частинним випадком недиференційовних функцій, – опуклий аналіз. Отже, методи мінімізації негладких функцій суттєво базувались на використанні різних властивостей опуклих функцій.

Основні методи мінімізації диференційовних функцій, а саме градієнтний та Ньютона, були побудовані на лінійній або квадратичній апроксимації функції, що задається першими членами ряду Тейлора. Однак для негладкої функції такий підхід не дає результатів: ця функція не апроксимується достатньо добре ні лінійною, ні квадратичною функціями (Поляк 1983).

Однією з перших спроб побудувати метод мінімізації недиференційовних функцій був *субградієнтний метод*, який є аналогом градієнтного методу, в

якому замість градієнту використовувався субградієнт функції. Субградієнтний метод використовує той же напрямок пошуку, що і градієнтний метод, якщо функція, що мінімізується, є диференційовною, тобто перший метод є певним узагальненням другого. Цей метод запропонував Н.З. Шор у своїй кандидатській дисертації 1964 року, застосувавши його до чисельного розв'язку задачі, двоїстої до транспортної задачі лінійного програмування. Пізніше цей підхід був розширений та обґрунтований Б.Т. Поляком в (1967) та (1969), Ю.М. Єрмольєвим (1966), а також узагальнений Н.З. Шором у монографії (1979) (перекладена англійською (1985)).

Розглянемо субградієнтний метод Шора трохи докладніше. Він являє собою аналог градієнтного методу з заміною градієнта на довільний субградієнт функції:

$$x_{k+1} = x_k - \gamma_k \partial f\left(x_k\right), \tag{1.1}$$

де γ_k – крок, $\partial f\left(x_k\right)$ – довільний субградієнт функції $f\left(x\right)$ в точці x_k. Особливість цього методу полягає в тому, що монотонно спадає не значення функції, а відстань до точки мінімуму. Дуже важливим є підхід до вибору довжини кроку. Вибір кроку γ_k константою може призвести до того, що метод не буде збігатись (наприклад, при $f\left(x\right) = \|x\|$). Якщо ж γ_k обрати таким, як у методі найшвидшого спуску, тобто рівним

$$\gamma_k = \arg\min_{\gamma > 0} f\left(x_k + \gamma s\left(x_k\right)\right), \tag{1.2}$$

де $s = s\left(x\right) \in \mathbb{R}^n$ – нормований вектор напрямку найшвидшого спуску в точці x, функція $f\left(x\right)$ не обов'язково спадає в напрямку антисубградієнту $-\partial f\left(x_k\right)$. Класичний спосіб вибору крокових множників у субградієнтному методі є таким: обирається послідовність кроків, що збігається до нуля. В такому разі метод має вигляд

$$x_{k+1} = x_k - \gamma_k \frac{\partial f(x_k)}{\left\| \partial f(x_k) \right\|},\ \gamma_k \to 0,\ \sum_{k=0}^{\infty} \gamma_k = \infty. \qquad (1.3)$$

Отже, довжина кроку прямує до нуля, тоді як сума довжин кроків є нескінченною. Такими властивостями володіють, наприклад, послідовності $\gamma_k = \frac{\gamma}{k+c}$ та $\gamma_k = \frac{\gamma}{k^{\rho}}$, де $0 < \rho \le 1$. На жаль, оскільки сума кроків рівна нескінченності, довжина кроку спадає дуже повільно, тому такий метод не може збігатись зі швидкістю геометричної прогресії (Поляк 1983).

Іншим способом регулювання кроку в субградієнтному методі (1.1) є використання різниці значення функції в поточній точці $f(x_k)$ та мінімального значення функції f^*, яке не обов'язково має бути відомим. Якщо f^* відоме можна побудувати варіант субградієнтного методу без довільних параметрів (Поляк 1969, 1983):

$$x_{k+1} = x_k - \frac{f(x_k) - f^*}{\left\| \partial f(x_k) \right\|^2} \partial f(x_k). \qquad (1.4)$$

В книзі Поляка (1983) доведено, що такий метод збігається зі швидкістю $O\left(1/\sqrt{k}\right)$ у випадку мінімізації довільної опуклої функції та зі швидкістю геометричної прогресії у випадку функції, що має гострий мінімум. Якщо ж величина f^* невідома можна використовувати її оцінку, яка обчислюється на основі поведінки точок x_k.

В деяких випадках субградієнт доцільно замінити ε-субградієнтом, в яких він обчислюється простіше. Тоді метод (1.3) набуває вигляду

$$x_{k+1} = x_k - \gamma_k \frac{\partial_{\varepsilon_k} f(x_k)}{\left\| \partial_{\varepsilon_k} f(x_k) \right\|},\ \gamma_k \to 0,\ \sum_{k=0}^{\infty} \gamma_k = \infty. \qquad (1.5)$$

Якщо величина ε_k є сталою метод може не збігатись, тому її необхідно змінювати та спрямовувати до нуля при $k \to \infty$ під час роботи методу (Поляк 1983).

Інший напрямок розвитку мінімізації негладких функцій характеризувався використанням інформації, отриманої на попередніх кроках: точок $x_0,...,x_k$ та значень субградієнтів в цих точках $\partial f(x_0),...,\partial f(x_k)$. Користуючись означенням субградієнту функції в точці, можна стверджувати, що точка мінімуму x^* лежить в області, що задається нерівностями

$$Q_k = \left\{ x : \left(\partial f(x_i), x - x_i \right) \le f^* - f(x_i),\ i = \overline{0,k} \right\}$$

якщо значення f^* є відомим та

$$Q_k = \left\{ x : \left(\partial f(x_i), x - x_i \right) \le 0,\ i = \overline{0,k} \right\}$$

в протилежному випадку. Основною ідеєю методів такого типу є вибір точки x_{k+1} так, щоб максимально зменшити область Q_k. Наприклад, в *методі відсікаючої гіперплощини* значення x_{k+1} обирається як точка мінімуму кусочно-лінійної апроксимації функції $f(x)$, що визначається значеннями цієї функції та її субградієнту на попередніх ітераціях. Недоліком методу є необхідність розв'язувати задачу лінійного програмування на кожній ітерації алгоритма, причому кількість обмежень цієї задачі зростає. Інший спосіб вибору точки x_{k+1} реалізується в *методі чебишівських центрів*: такою точкою обирається чебишівський центр многогранника Q_k. Також як точку x_{k+1} можна брати найближчу до x_k точку, що задовольняє нерівності Q_k для довільної підмножини індексів множини $I = \{0,...,k\}$ – така процедура є задачею квадратичного програмування. Суттєвою перевагою методу є те, що для розв'язання цієї задачі на кожній ітерації методу використовуються не всі індекси множини I, отже її розмірність є невеликою. Недоліком методу є необхідність знати оптимальне значення f^* (Поляк 1983).

Ще одним методом мінімізації негладких функцій, що використовує інформацію, отриману на попередніх ітераціях, є *метод центрів ваги*. В ньому точка x_{k+1} обирається як центр ваги многогранника Q_k. Такий вибір зумовлений тим, що відношення об'єму будь-якої з частин, що утворюються проведенням

через центр ваги многогранника Q_k гіперплощини, до об'єму Q_k не більше $1-\frac{1}{e}$, причому ця величина може тільки збільшитись, якщо замість центру ваги взяти будь-яку іншу точку. Отже, на кожному кроці відсікається суттєва частина многогранника Q_k, що і є ключовою особливістю методу центрів ваги. Наприклад, при $n=1$ многогранник Q_k є відрізком, а центром ваги – його середина, тому відсікатись буде половина цього відрізка.

Однією з основних переваг методу центрів ваги є незалежність швидкості його збіжності та характеристик функції, що мінімізується. На швидкість впливає лише розмірність простору n та початкові умови, тому для задач невеликої розмірності метод збігається досить швидко: за ne ітерацій точність розв'язку можна покращити в e разів. До того ж цей метод є оптимальним в тому сенсі, що його швидкість збіжності не може бути перевищена жодним методом оптимізації, що використовує значення $f(x)$ та $\partial f(x)$ (теорема Неміровського – Юдіна (Поляк 1983)). Щоправда, метод володіє одним явним недоліком: для розмірності $n>2$ пошук центру ваги многогранника є надзвичайно складною задачею.

1.2 Методи з розтягом простору

Метод еліпсоїдів. Однією з центральних фігур у створенні та розвитку методів з розтягом простору є Наум Зуселевич Шор, який працював в Інституті кібернетики імені В.М. Глушкова з 1958 до 2006 року. Його робота (Шор 1977) суттєво спрощувала метод центрів ваги, виключаючи необхідність знаходити центр ваги та запам'ятовувати значення субградієнтів $\partial f(x_k)$ попередніх ітерацій, але зберігала високу швидкість збіжності. Описаний в роботі метод відомий під назвою метод еліпсоїдів. Варто зазначити, що в роботі (1976) Д.Б. Юдін та А.С. Неміровський описали метод еліпсоїдів як метод послідовних відсікань, назвавши його модифікованим методом центрованих відсікань

(ММЦВ). Н.З. Шор представив метод еліпсоїдів як частковий випадок субградієнтних методів з розтягом простору в напрямку субградієнта, які були запропоновані ним в 1969-1970 роках. Шор вказав коефіцієнт розтягу простору та параметри регулювання кроку в напрямку нормованого антисубградієнту такими, що субградієнтний метод з розтягом простору збігався з геометричною швидкістю зменшення об'єму еліпсоїда, в якому локалізована точка мінімуму опуклої функції, чим і довів збіжність методу еліпсоїдів.

В методі еліпсоїдів многогранник Q_k вписується в кулю, в результаті чого його центр ваги збігається з центром кулі, який і обирається як точка x_{k+1}. Такий підхід суттєво спрощує процедуру пошуку центра ваги многогранника. Ітерація методу еліпсоїдів полягає в побудові еліпсоїда мінімального об'єму навколо півкулі, що лежить за напрямком субградієнта $\partial f\left(x_{k+1}\right)$, та подальшому перетворенні цього еліпсоїда в кулю за допомогою лінійного перетворення простору. Таким чином, на кожній ітерації достатньо зберігати в пам'яті лише точку x_{k+1} та матрицю перетворення простору H_k – це ще одне спрощення методу як порівняти з методом центрів ваги.

Збіжність методу еліпсоїдів ґрунтується на тому, що з кожною ітерацією об'єм кулі, що локалізує розв'язок, зменшується в $q^{1/n}$ разів, де $q \sim 1 - 1/2n^2$ – знаменник прогресії, що описує швидкість збіжності методу. Як бачимо, щодо питання збіжності метод еліпсоїдів поводить себе так само, як і метод центрів ваги – він збігається зі швидкістю геометричної прогресії зі знаменником, який не залежить від властивостей функції, що мінімізується, а залежить лише від розмірності простору. На жаль, незважаючи на всі переваги методу еліпсоїдів, швидкість його збіжності при великих n суттєво падає і він стає малоефективним.

Поява методу еліпсоїдів дозволила розв'язати набір важливих задач в теорії складності задач математичного програмування. Зокрема, в 1979 році Л.Г. Хачіян побудував перший поліноміальний алгоритм розв'язання задачі лінійного програмування з раціональними коефіцієнтами (1979а,б), чим довів

існування поліноміальних алгоритмів для задач такого типу. На практиці алгоритм Хачіяна демонстрував неоднозначні результати: для деяких задач з всього лише 50 змінними він потребував понад 24 тисячі ітерацій, тоді як симплекс-метод – лише декілька сотень або навіть десятків. Тим не менш, наявні також протилежні результати, коли алгоритм Хачіяна працює в сотні разів швидше симплекс методу. Незважаючи на те, що алгоритм виявився непридатним для практичних обчислень через високу ступінь многочлена, що оцінює час його роботи, результат Хачіяна має велике теоретичне значення – він став причиною інтенсивних досліджень нових практичних алгоритмів для розв'язання задач лінійного програмування.

Того ж 1979 року був побудований поліноміальний алгоритм для розв'язання задач опуклого квадратичного програмування з цілими коефіцієнтами (Козлов, Тарасов и Хачиян 1979), а також поліноміальні алгоритми для розв'язання набору задач аналізу властивостей многогранних опуклих множин, що задаються системою лінійних нерівностей з раціональними коефіцієнтами (Backem and Grötschel 1981), наприклад, задачі виокремлення несуттєвих обмежень, визначення мінімальної твірної системи вершин і твірних многогранної множини тощо. В роботі (Shrader 1980) наведено поліноміальний алгоритм для розв'язання лінійної проблеми додатковості для випадку невід'ємної визначеності матриці M . Постановка цієї проблеми така: для заданої $n \times n$ матриці M та вектора $q \in E_n$ необхідно знайти такий вектор $z \in E_n$, що задовольняє нерівність $Mz + q \geq 0$, $z \geq 0$ та рівність $(z, Mz + q) = 0$. Таку задачу можна розглядати як частковий випадок задачі опуклого квадратичного програмування, для якої існує поліноміальний алгоритм.

Ще одним важливим наслідком появи методу еліпсоїдів вважається результат Грьотшеля, Ловаса та Схрейвера (1981), в якому автори показали, що більшість комбінаторних задач, для яких були відомі поліноміальні алгоритми, можна розв'язати за поліноміальний час, використовуючи метод еліпсоїдів

через перехід до відповідної задачі відокремлення. Прикладами таких задач можуть бути задачі на мінімальний розріз, оптимізація лінійної функції на перетині двох матроїдів, максимальний зважений незалежний набір ребер графа, задача про китайського поштаря, мінімізація субмодулярної функції на структурі підмножин тощо. В роботі (1981) наведено також обґрунтування поліноміальної еквівалентності для заданого класу опуклих тіл задачі (слабкої) оптимізації та (слабкого) відокремлення. Цей результат дав можливість отримати нові результати про NP-повноту деяких задач.

Незважаючи на неспроможність методу еліпсоїдів розв'язувати складні обчислювальні задачі на практиці, його ідея не вичерпала всіх своїх можливостей. Внаслідок цього почали з'являтись численні модифікації цього алгоритма. Наприклад, в роботі (2003а) наведено наближений метод еліпсоїдів, який дає змогу використовувати метод в одновимірному випадку, зберігаючи асимптотичну швидкість збіжності за об'ємом, як і у відомому методі Юдіна – Неміровського – Шора. В роботах (Шор и Гершович 1979) та (König and Pallaschke 1981) були зроблені спроби побудувати модифікації методу еліпсоїдів з більшими коефіцієнтами зменшення об'єму на кожному кроці. Для цього використовувались оптимальні еліпсоїди, описані навколо складних опуклих тіл (сегмент, «кульовий» шар та s-піраміда). Ітерація таких методів полягала в переході в центр мінімального за об'ємом еліпсоїда, описаного навколо одного з цих опуклих тіл. Продовженням робіт (Шор и Гершович 1979) та (König and Pallaschke 1981) є стаття (Стецюк и Буханцов 2002а), в якій прискорення досягається через використання еліпсоїда мінімального об'єму, що містить кульовий шар. Ідея такої модифікації полягає в тому, що на поточній ітерації використовується відтинаюча гіперплощина з попереднього кроку. Якщо вона дає можливість зменшити об'єм апроксимуючого еліпсоїда – зменшення реалізується, інакше використовується звичайний крок методу еліпсоїдів. При мінімізації суттєво-яружних функцій така модифікація демонструє в декілька разів вищу швидкість збіжності, ніж метод еліпсоїдів Юдіна – Неміровського – Шора. Нарешті в роботі (Стецюк, Фесюк и Хомяк

2018a) описано так званий узагальнений метод еліпсоїдів, у якому швидкість збіжності методу визначається величиною знаменника геометричної прогресії $q_n(\alpha)<1$ з параметром α, який задовольняє нерівність $\alpha + 1/\alpha < 2\sqrt[n]{\alpha}$. Залежно від параметра α ми отримуємо коефіцієнт розтягу простору, що відповідає методу Юдіна – Неміровського – Шора, наближеного методу еліпсоїдів (Стецюк 2003a) та інших модифікацій методу.

r-алгоритми. Ще одним важливим напрямом розвитку субградієнтних методів з перетворенням простору були методи, в яких напрямок розтягу визначався різницею двох послідовних субградієнтів. Такі методи були одним з центральних результатів докторської дисертації Н.З. Шора 1970 року і називаються r-алгоритмами (від російського слова «разность»)(Шор 1979, Шор и Журбенко 1971). Їхня прискорена збіжність для яружних функцій пов'язана з вибором кроку з умови точного (наближеного) мінімуму функції у напрямку зміщення, унаслідок чого визначаються ті два послідовні субградієнти, в напрямку різниці яких відбувається покращення властивостей яружної функції в перетвореному просторі змінних. За певного регулювання кроку та коефіцієнтів розтягу простору r-алгоритми є монотонними (або майже монотонними) по функції, що мінімізується (Стецюк 2013).

r-алгоритми є одними з найбільш ефективних та практично перевірених засобів розв'язання великого розмаїття задач негладкої оптимізації. Однак теоретичні обґрунтування збіжності цього класу алгоритмів не є повними. Найбільш загальним результатом про їхню збіжність є теорема, доведена Шором в роботі (Шор 1975) для «ідеалізованого» варіанту r-алгоритма – $r_\mu(\alpha)$-алгоритма. Вона стверджує, що для класу майже диференційовних кусочно-гладких функцій отримані достатні умови, за яких $r_\mu(\alpha)$-алгоритм збігається до локального мінімуму. На жаль, вони є занадто сильними і не виконуються навіть для кусочно-лінійних функцій. Найбільш типовий випадок, коли це відбувається, пов'язаний з порушенням лінійної незалежності множини $G_f(x)$

майже-градієнтів функції $f(x)$ в точці x. За таких умов довести збіжність $r_\mu(\alpha)$-алгоритма для опуклих функцій неможливо (Стецюк 2013).

Тим не менш, на практиці r-алгоритми часто демонструють чудові результати при мінімізації яружних функцій. Одним з таких ефективних алгоритмів є варіант r-алгоритма з адаптивним регулюванням кроку – $r(\alpha)$-алгоритм, в якому величина кроку h_k обирається з умови наближеного пошуку мінімуму за напрямком і так, щоб виконувалась умова $h_k > h_k^*$, де h_k^* відповідає мінімуму функції за напрямком. Такий спосіб регулювання кроку дає змогу зменшити кількості обчислень $f(x)$ та $\partial f(x)$ так, щоб на одну ітерацію $r(\alpha)$-алгоритма припадало не більше за 2-3 обчислення цих значень. Детальний опис такого способу регулювання кроку наведено в роботі (Шор и Стеценко 1989). На його основі, а також на основі його модифікацій, було розроблено низку програмних реалізацій r-алгоритма (Шор та ін 2003), наприклад, octave-програма ralgb5 (Стецюк 2011a).

При мінімізації гладких функцій r-алгоритми за своєю структурою близькі до алгоритмів квазіньютонівського типу зі змінною метрикою – обидва класи алгоритмів використовують схожі «антияружні» прийоми, чим і пояснюється їхня ефективність при мінімізації яружних функцій. Звідси випливають два важливі результати. Перший – граничний варіант r-алгоритма з нескінченним коефіцієнтом розтягу (тобто, $\beta = 0$ і $h_k = h_k^*$) є проєктивним варіантом методу спряжених градієнтів (Шор и Журбенко 1971). Другий результат стверджує, що граничний варіант r-алгоритма з відновленням матриці перетворення простору кожні n ітерацій за звичайних умов гладкості та регулярності функції $f(x)$ має квадратичну швидкість збіжності (Шор и Журбенко 1971).

1.3 Методи феєрівського типу

Ще одним важливим класом методів мінімізації негладких функцій є так звані феєрівські методи. Ці методи беруть свій початок в роботах (Agmon 1954) та (Motzkin and Schoenberg 1954) і є різновидом методу субградієнтного спуску, в яких регулювання кроку в напрямку нормованого субградієнту відбувається деяким спеціальним чином. Для розв'язування задачі безумовної опуклої оптимізації феєрівські методи задаються такою ітеративною процедурою:

$$x_{k+1} = x_k - h_k \frac{\partial f(x_k)}{\|\partial f(x_k)\|} \quad , \; h_k = \gamma \frac{f(x_k) - f^*}{\|\partial f(x_k)\|}, \tag{1.6}$$

де $\partial f(x_k)$ – субградієнт функції $f(x)$ в точці x_k, γ – скаляр, що задовольняє нерівність $0 < \gamma < 2$. Такий вибір параметра γ гарантує монотонне зменшення відстані до мінімуму на кожному кроці процедури (1.6) і дає змогу суттєво спростити процес доведення збіжності цих методів. Крок h_k при $\gamma = 1$ називається класичним феєрівським кроком (або кроком Аґмона – Моцкіна – Шонберга чи кроком Поляка), а відповідний йому метод – класичним феєрівським методом. Цей метод також відомий як субградієнтний метод кроком Поляка (Поляк 1983, 1969). Уперше ams-крок в 1954 році незалежно використали Аґмон (Agmon 1954) та Моцкін і Шонберг (Motzkin and Schoenberg 1954) у релаксаційному методі для знаходження хоча б одного розв'язку сумісної системи лінійних нерівностей. Пізніше було запропоновано узагальнення цього методу для системи опуклих нерівностей (Еремин 1965) та знаходження спільної точки опуклих множин (Брэгман 1967).

Незважаючи на теоретичну прозорість та простоту, класичний феєрівський метод вкрай повільно працює навіть для гладких яружних опуклих функцій, у яких напрямок антисубградієнту в точці до напрямку на мінімум близький до ортогонального. Якщо функція не є яружною метод демонструє задовільні результати для гладких функцій з доволі великою розмірністю задачі ($n \sim 100$).

Однак для яружних функцій – як гладких, так і негладких – метод працює вкрай незадовільно навіть для задач розмірності $n \sim 2-5$ (Стецюк 1997а).

Для прискорення класичного феєрівського методу функціонал в околі мінімуму можна наблизити кусочно-лінійним функціоналом, використовуючи інформацію попередніх ітерацій – такий метод був запропонований в Поляком в (1969):

$$x_{k+1} = P_{Q_k}\left(x_k\right), \quad Q_k = \left\{x:\ f\left(x_i\right) + \left(\partial f\left(x_i\right), x - x_i\right) \le f^*,\ i \in I_k\right\}. \qquad (1.7)$$

Тут I_k – будь-яка множина індексів з $0,1,...,k$, що обов'язково містить k, P_{Q_k} – оператор проєктування на Q_k. Цей метод збігається зі швидкістю геометричної прогресії, а його частинні випадки тісно пов'язані з іншими відомими методами. Наприклад, якщо $I_k = \{0,1,...,k\}$ метод (1.7) дає точний мінімум для кусочно-лінійної функції, причому за своєю ідеєю він є близьким до методу Келлі (Kelley 1960). Основною проблемою методу (1.7) є опис конструктивного способу формування множини I_k для того, щоб обмежити кількість субградієнтів, які необхідно зберігати. Найбільш суттєвою спробою вирішити цю проблему можна вважати метод ортогонального спуску М.Б. Щепакіна (Щепакин 1987). Основною ідеєю методу Щепакіна було послідовне занурення нетупокутного конуса, що містив множину екстремумів задачі, в прямокутний конус та використання набору з не більше ніж $(n+1)$ ортогональних векторів, які є субградієнтами функції в точці або їхніми опуклими комбінаціями. Однак, цей метод не завжди давав точні за функціоналом розв'язки задачі (1.7). Такий висновок випливає з результатів чисельних експериментів з методом ортогонального спуску та його модифікаціями, проведених в роботах (Щепакин и Шубенкова 1993) та (Скоков и Щепакин 1994): для порівняно невеликого розміру задач $(n \sim 10)$ та ступеня яружності, точність розв'язків за функціоналом часто варіювалась у межах $\varepsilon_f \sim 10^{-4} - 10^{-2}$ (Стецюк 1997а).

Ще один метод феєрівського типу, який використовує всього два вектори, наведений в (Camerini, Fratta and Maffioli 1975). Для побудови чергового

напрямку спуску метод використовує лінійну комбінацію двох напрямків: субградієнту функції в точці та напрямку руху на попередньому кроці так, щоб він утворював більш гострий кут у напрямку мінімуму. Як порівняти з розглянутими вище методами, які використовують усього два вектори, що рухаються «стрибками», такий метод забезпечує більш «плавний» рух вздовж яру. На жаль, метод є неефективним для функцій з багатовимірним яром (Стецюк 1997а).

Для отримання більш точного розв'язку задачі безумовної опуклої оптимізації за допомогою класичного феєрівського методу можна скористатись лінійним перетворенням простору для вирівнювання структури поверхонь рівня функції в перетвореному просторі. Нехай $Y = AX$ – простір аргументів, перетворений за допомогою лінійного оператора A, що задається невиродженою матрицею розмірності $n \times n$. Тоді класичний феєрівський метод з перетворенням простору в X має такий вигляд:

$$x_{k+1} = x_k - h_k B \frac{B^T \partial f(x_k)}{\left\| B^T \partial f(x_k) \right\|},\ h_k = \frac{f(x_k) - f^*}{\left\| B^T \partial f(x_k) \right\|}, \qquad (1.8)$$

Цей метод можна записати у двох різних формах. B-форма (1.8) дає змогу дати прозору інтерпретацію процесу в перетвореному просторі. H-форма методу скорочує обсяг пам'яті для зберігання матриці перетворення простору майже вдвічі. В роботі (Стецюк 1996) наведено два субградієнтні методи з класичним феєрівським кроком в перетвореному просторі аргументів. В роботах (Стецюк 1997а,б) наведений детальний опис методів феєрівського типу з перетворенням простору та їхні модифікації. В роботі (Гершович и Шор 1982) запропонована модифікація класичного феєрівського методу на основі використання ε-субградієнтів.

Оскільки класичний феєрівський метод вкрай неефективно працює з яружними функціями, цілком природніми є спроби прискорити його для функцій такого типу. Одна з таких спроб була виконана в роботі (Стецюк 2011б). Вона базується на такій ідеї. Повільна збіжність класичного

феєрівського методу пояснюється тупим кутом між послідовними субградієнтами: чим ближча градусна міра цього кута до 180 градусів, тим повільніше працює метод. Отже, метод можна прискорити за допомогою такого перетворення простору, що призведе до зменшення цього кута. Таке перетворення відбувається за допомогою однорангового еліпсоїдального оператора (Стецюк 1997б). Метод, наведений в (Стецюк 2011б), був названий *amsg2p*, перші три літери якого вказують на спосіб регулювання кроку в напрямку нормованого антисубградієнту, а «g2p» – на спосіб перетворення простору, в якому використовується ams-крок. Таке перетворення здійснюється за допомогою двох останніх субградієнтів (g2) та агрегатного вектора (p) лише на тих ітераціях, коли тупим є хоча б один з кутів: або кут між двома послідовними субградієнтами, або кут між останнім субградієнтом та агрегатним вектором, що є опуклою комбінацією субградієнтів, обчислених на попередніх ітераціях.

Якщо функція не є яружною метод amsg2p рівносильний класичному субградієнтному методу з кроком Поляка (1.6) з однією відмінністю: нерівність

$$\left(x - x^*, \partial f\left(x\right)\right) \geq \gamma\left(f\left(x\right) - f^*\right),\ \gamma \geq 1,$$

що виконується для довільної опуклої функції $f\left(x\right)$ та її субградієнта $\partial f\left(x\right)$, для спеціальних класів функцій реалізує сильніші ams-кроки, ніж це виконувалось для опуклої функції при $\gamma = 1$.

Антияружна техніка методу amsg2p дає можливість суттєво прискорити класичний феєрівський метод при мінімізації яружних опуклих функцій. В роботі (Стецюк 2009) цей метод був адаптований для довільного значення функції $f_{\min}$ і дає можливість або знаходити таку точку, в якій значення опуклої функції $f\left(x\right)$ менше або дорівнює $f_{\min} + \varepsilon$, або гарантувати достатню умову того, що в кулі заданого радіусу не існує точки, в якій значення $f\left(x\right)$ рівне $f_{\min}$. Обчислювальні експерименти, що демонструють ефективність методу

amsg2p та порівняння його з іншими методами з перетворенням простору наведені в роботах (Стецюк 2009, 2012а).

Оцінку розвитку методів з розтягом простору в свій час дали Н.З. Шор та В.І. Гершович в роботі (1982): «Теория всего класса алгоритмов с растяжением пространства далека от совершенства. Нам кажется достаточно реалистичной целью – построение такого алгоритма, который по своей практической эффективности не уступал бы r-алгоритма и был столь же хорошо обоснован как метод эллипсоидов». Одна зі спроб реалізувати такі наміри зроблена в роботі П.І. Стецюка (1996), де для перетворення спеціального еліпсоїда в кулю використовується антияружний прийом, близький до того, що застосовується в r-алгоритмах.

1.4 Сучасний стан розвитку методів оптимізації

На даний момент оптимізація є досить розвиненою областю обчислювальної математики. Теорії та методам розв’язку оптимізаційних задач присвячено багато різних джерел: (Васильев 2011а,б), (Измаилов и Солодов 2003), (Поляк 2014), (Еремин и Астафьев 1976), (Нестеров 2010), (Сухарев, Тимохов и Федоров 2008), (Жадан 2014, 2015, 2017) тощо. В джерелах (Vanderbei 1997) та (Boyd and Vandenberghe 2004) матеріал викладено англійською мовою, причому в останньому представлений досить сучасний погляд на теорію та методи опуклої оптимізації (Жадан 2014). В книгах (Васильев 2011а,б), (Ермольев 1976), (Boyd and Vandenberghe 2004), (Vanderbei 1997), (Аоки 1977), (Базара и Шетти 1982), (Пшеничный и Данилин 1975), (Полак 1974), які мають право вважатись класичними, розглядаються не лише основні визначення, поняття, та методи оптимізації, а й конкретні приклади з різноманітних прикладних областей, які можуть бути сформульовані як оптимізаційні задачі.

Література з чисельних методів розв’язку оптимізаційних задач теж є доволі обширною. Серед них наявні як фундаментальні монографії, що

об’єднують методи розв’язку різних класів задач (наприклад, (Васильев 2011а,б), (Аоки 1977), (Базара и Шетти 1982), (Андреева и Цирулева 2001), (Бирюков 2003), (Галеев 2006), (Гороховик 2007), (Лесин и Лисовец 2011)), так і книги, присвячені методам розв’язання окремих спеціальних класів задач. Наприклад, сучасні методи розв’язання задач безумовної оптимізації викладені в книгах (Васильев 2011а,б), (Гороховик 2007), (Измаилов и Солодов 2003), (Лесин и Лисовец 2011), (Поляк 2014); методи, що базуються на використанні функції Лагранжа та її модифікацій розглядаються в роботах (Бертсекас 1987), (Гольштейн и Третьяков 1989), (Евтушенко 1982), (Измаилов и Солодов 2003), (Карманов 1986); методи розв’язання нелінійних мінімаксних задач детермінованого та ймовірнісного характеру (Нурминский 1979). Фундаментальна монографія (Немировский и Юдин 1979) присвячена дослідженню складності задач і трудоємності методів оптимізації.

Окремо відзначимо ті задачі та проблеми оптимізації, над якими ведеться активна робота впродовж останніх років. Наведемо також авторів, що працюють над ними, та їхній вклад в розвиток сучасної оптимізації.

Активна побудова теорії двоїстості для лексикографічних задач лінійної оптимізації та багатокритеріальної Парето-оптимізації проводилась І.І. Єрьоміним в роботах (2001а, 2003, 2005) та (Eremin 2002a). Розробка широкого класу ітераційних методів феєрівського типу для розв’язання систем лінійних та опуклих нерівностей, і задач математичного програмування також проводилась автором та відображена в роботах (2001б, 2004) та (2002b).

Роботи останніх років Б.Т. Поляка присвячені застосуванню методів оптимізації до розв’язання прикладних задач широкого спектру (Поляк и Шалби 2019б), а також модифікаціям наявних методів оптимізації та характеристиці їхньої поведінки тощо. Наприклад, в роботі (2020) Поляком та Фатхуліним описано застосування проєктивного покоординатного спуску до задачі Фекете (мінімізації енергії системи на сфері). В статті (2017) Поляком та Хлєбніковим запропоновано декілька робастних версій статистичного методу головних компонент та чисельних методів їхньої реалізації, які широко

використовуються для компактного представлення даних в сучасних задачах оптимізації. В роботі (Поляк и Смирнов 2019а) пояснюється немонотонна поведінка прискорених методів безумовної оптимізації.

Стохастичній та опуклій оптимізації присвячені роботи останніх років Ю.Є. Нестерова та А.С. Неміровського. Наприклад, в статті (2018) Гасніковим та Нестеровим був запропонований універсальний метод для задач стохастичної композитної оптимізації. В роботі (Гасников, Двуреченский и Нестеров 2016а) описано сучасний стан методів проекції градієнта для розв'язання задач опуклої стохастичної оптимізації з неточним оракулом. Роботи (Necoara, Nesterov and Glineur 2019) та (Nesterov 2018) присвячені лінійній збіжності методів першого порядку та границям складності для прямодвоїстих методів у опуклій оптимізації. В публікації (2016б) Гасніков, Гаснікова, Нестеров та Чернов запропонували нові чисельні алгоритми розв'язання задачі ентропійно-лінійного програмування та встановили точні оцінки швидкостей збіжності цих методів. В роботі (Воронцова, Гасников и Горбунов 2019) запропоновано прискорений метод спуску за випадковим напрямком з неевклідовою прокс-структурою для розв'язання задач безумовної оптимізації та оцінки швидкості збіжності методу.

Робота над методами розв'язання задач опуклої оптимізації в нескінченновимірних просторах активно проводиться протягом останнім років. Наприклад, в роботі (Ведель та Семенов 2020) запропоновано нові адаптивні двоетапні проксимальні алгоритми для розв'язання задач про рівновагу в метричних просторах Адамара, у вигляді яких можна сформулювати опуклі задачі. Робота (Войтова, Денисов та Семенов 2012) присвячена розв'язанню задачі дворівневої опуклої оптимізації за допомогою альтернуючого проксимального алгоритму. В роботі (Малицкий и Семенов 2014) запропоновано ітераційний алгоритм для розв'язання варіаційної нерівності з монотонним та ліпшицевим оператором, що діє в гільбертовому просторі.

Підхід стохастичної апроксимації до задачі стохастичного програмування був запропонований Неміровським та іншими в роботі (Nemirovski et al. 2009).

Окрім стохастичної оптимізації, роботи Неміровського останніх років зосереджені також на робастній оптимізації (Ben-Tal et at. 2004, Ben-Tal and Nemirovski 2008), алгоритмах першого порядку для розв'язання опуклих оптимізаційних задач великої розмірності (Harchaoui, Juditsky and Nemirovski 2015) та непараметричній статистиці (Juditsky and Nemirovski 2009, Juditsky et al. 2012).

Варто відмітити роботи М.Г. Журбенка та О.П. Лиховида, присвячені різним аспектам субградієнтних алгоритмів оптимізації. Зокрема, в публікації (Журбенко 2012) описується $\alpha(\varepsilon)$-алгоритм – теоретично обгрунтований субградієнтний алгоритм з перетворенням простору. За кількістю ітерацій ефективність цього алгоритму знаходиться на одному рівні з r-алгоритмом, однак трудомісткість однієї ітерації першого суттєво вища. Робота (Журбенко и Лиховид 2019) присвячена $r(\sigma)$-алгоритмам – сімейству модифікацій r-алгоритму з програмним керуванням коефіцієнтами розтягу простору; робота (Журбенко 1999) – квазіньютонівським алгоритмам мінімізації на основі використанні перетворення простору. В публікації (Журбенко 2015) запропоновано алгоритм Поляка з перетворенням простору на основі агрегатних ε-субградієнтів. Різним аспектам модифікацій r-алгоритмів присвячені також роботи (2017) та (Журбенко и Лиховид 2018).

Методи розв'язання задач опуклого програмування тісно пов'язані з розв'язанням задач еліпсоїдального оцінювання. Підхід методу еліпсоїдального оцінювання активно розвивався в роботах М.М. Сальнікова, Б.Т. Поляка, О.Г. Наконечного, В.В. Волосова та інших. Наприклад, в роботі (Сальников 2014) здійснюється оцінювання станів та параметрів динамічної системи за умови відсутності апріорної інформації про ці величини. Стаття (Волосов 1996) присвячена побудові еліпсоїдальних оцінок в задачах нестохастичної ідентифікації параметрів та станів багатовимірних дискретних об'єктів керування.

1.5 Постановка завдання дослідження

Метою роботи є розробка модифікацій субградієнтного методу з кроком Поляка у вихідному та перетвореному просторах змінних для мінімізації яружних функцій, обґрунтування їхньої збіжності та оцінок швидкості збіжності. На сьогодні проблема мінімізації яружних функцій трапляється на практиці дедалі частіше. Для розв’язання задачі технічної та медичної діагностики, проєктування складних технічних об’єктів, аналізу стійкості динамічних систем необхідно мінімізувати функції з суттєво або сильно витягнутими поверхнями рівня. Класичний градієнтний метод вкрай повільно працює для функцій такого типу, тому запропоновано багато різних модифікацій цього методу щоб прискорити його роботу. Застосування лінійного перетворення простору для мінімізації яружних функцій дозволило отримати суттєві результати в цьому напрямі. Це означає, що розробка нових алгоритмів та їхніх модифікацій для вирішення таких задач з урахуванням зростання їхньої розмірності є актуальним напрямом досліджень.

Для виконання цього завдання необхідно вирішити такі задачі:

1. розробити та обґрунтувати модифікації субградієнтного методу з кроком Поляка у вихідному та перетвореному просторах змінних для мінімізації яружних опуклих функцій;
2. навести та обґрунтувати оцінки швидкості збіжності запропонованих модифікацій;
3. провести обчислювальні експерименти з використанням розроблених модифікацій для мінімізації гладких та негладких яружних опуклих функцій;
4. розробити програмні реалізації запропонованих модифікацій та провести їхнє тестування;
5. дослідити застосування розроблених методів для розв’язання прикладних задач.

РОЗДІЛ 2. СУБГРАДІЄНТНИЙ МЕТОД З КРОКОМ ПОЛЯКА ТА ЗМІЩЕННЯ ПО ОПУКЛОСТІ

У другому розділі розглядається субградієнтний метод з кроком Поляка для мінімізації опуклих функцій при відомому оптимальному значенні та його модифікація скалярним параметром $m>1$, який дає змогу врахувати деякі спеціальні класи опуклих функцій. Для обох методів наведено обґрунтування монотонності зменшення відстані до точки мінімуму та швидкості збіжності, яка дорівнює $O\left(1/\sqrt{k}\right)$ (тут k – кількість ітерацій) у випадку довільної опуклої функції та швидкості геометричної прогресії для опуклої функції з гострим мінімумом (підрозділи 2.1 та 2.2). У підрозділі 2.4 наведені результати обчислювальних експериментів обох методів для мінімізації набору гладких та негладких яружних опуклих функцій. Чисельні приклади демонструють ефективність роботи та доцільність використання модифікованого субградієнтного методу з кроком Поляка для мінімізації функцій такого типу.

2.1 Субградієнтний метод з кроком Поляка

Нехай $f(x)$ – опукла функція, $x\in R^n$. Позначимо її мінімальне значення як $f^*=f\left(x^*\right)$ та припустимо, що точка мінімуму x^* єдина. Субградієнт $g_f(x)$ функції $f(x)$ задовольняє таку умову:

$$\left(x-x^*, g_f(x)\right)\geq f(x)-f^*,\ \forall x\in R^n. \qquad (2.1)$$

Тут (x,y) – скалярний добуток векторів $x\in R^n$ та $y\in R^n$.

Якщо $f(x)$ є неперервно-диференційовною в точці $\bar{x}$, то субградієнт $g_f(\bar{x})$ визначається однозначно та збігається з $\nabla f(\bar{x})$ – градієнтом функції

$f(x)$ в точці $\bar{x}$. У точках, де функція $f(x)$ негладка, субградієнт $g_f(\bar{x})$ визначається неоднозначно.

Нерівність (2.1) випливає з визначення субградієнта $g_f(x)$ опуклої функції $f(x)$. Справді, субградієнт $g_f(x)$ в точці x задовольняє нерівність

$$f(y) - f(x) \geq \left(g_f(x), y - x\right), \ \forall y \in R^n, \tag{2.2}$$

яка виконується також для x^* – точки мінімуму. Для точки x^* нерівність (2.2) перетворюється на нерівність $f(x^*) - f(x) \geq \left(g_f(x), x^* - x\right)$, яку, зважаючи на те, що $f(x^*) = f^*$, можна записати як нерівність (2.1).

Якщо f^* відома, то для знаходження наближення до точки x^* можна використати субградієнтний метод з кроком Поляка (1969):

$$x_{k+1} = x_k - h_k \frac{g_f(x_k)}{\left\| g_f(x_k) \right\|}, \quad h_k = \frac{f(x_k) - f^*}{\left\| g_f(x_k) \right\|}, \ k = 0,1,2... \tag{2.3}$$

Крок h_k називається кроком Поляка (або кроком Аґмона – Моцкіна – Шонберга). Вперше цей крок був використаний для мінімізації кусочно-лінійних опуклих функцій. У 1954 році Аґмон (Agmon 1954), Моцкін і Шонберг (Motzkin and Schoenberg 1954) використали цей крок у релаксаційному методі для знаходження розв'язку сумісної системи лінійних нерівностей. У 1965 р. І.І. Єрьомін (Еремин 1965) узагальнив цей релаксаційний метод для системи опуклих функцій.

Геометричний зміст методу (2.3) є таким. Функція $f(x)$ апроксимується лінійною функцією $\tilde{f}(x) = f(x_k) + \left(g_f(x_k), x - x_k\right)$, і крок вибирається так, щоб значення апроксимуючої функції стало рівним f^* (тобто $\tilde{f}(x_{k+1}) = f^*$). Для опуклої функції $f(x)$ крок h_k визначає таку величину максимального зміщення з точки x_k у напрямку нормалізованого антисубградієнта, для якого умова (2.1) гарантує, що кут між антисубградієнтом у точці x_k та напрямком від точки x_{k+1} до точки мінімуму x^* не буде тупим.

Це означає такий факт. Субградієнт у точці x_k визначає гіперплощину, яка локалізує точку x^* у напівпросторі в напрямку антисубградієнта. Якщо її перемістити на величину максимального зміщення, визначену кроком h_k, в напрямку нормованого антисубградієнта, то нова гіперплощина локалізуватиме x^* у такому напівпросторі відносно точки x_{k+1}.

Теорема 2.1 (Поляк 1983). Послідовність $\{x_k\}_{k=0}^{k^*-1}$, породжена методом (2.3), задовольняє такі нерівності:

$$\|x_{k+1}-x^*\|^2 \le \|x_k-x^*\|^2 - \frac{f(x_k)-f^*}{\|g_f(x_k)\|^2},\ k=0,1,2,... \qquad (2.4)$$

Теорема 2.1 гарантує, що в субградієнтному методі з кроком Поляка відстань до точки мінімуму монотонно зменшується. Крім того, задовольняються такі нерівності:

$$\left(x^*-x_{k+1},-g_f(x_k)\right)\ge 0,\ k=0,1,... \qquad (2.5)$$

Дійсно, використовуючи (2.1) та (2.3), маємо:

$$\left(x^*-x_{k+1},-g_f(x_k)\right)=\left(x_{k+1}-x^*,g_f(x_k)\right)=\left(x_k-x^*-h_k\frac{g_f(x_k)}{\|g_f(x_k)\|},g_f(x_k)\right)=$$

$$=\left(x_k-x^*,g_f(x_k)\right)-\left(h_k\frac{g_f(x_k)}{\|g_f(x_k)\|},g_f(x_k)\right)=\left(x_k-x^*,g_f(x_k)\right)-h_k\frac{\|g_f(x_k)\|^2}{\|g_f(x_k)\|}=$$

$$=\left(x_k-x^*,g_f(x_k)\right)-\frac{f(x_k)-f^*}{\|g_f(x_k)\|}\frac{\|g_f(x_k)\|^2}{\|g_f(x_k)\|}=\left(x_k-x^*,g_f(x_k)\right)-\left(f(x_k)-f^*\right)\ge 0.$$

Нерівності (2.5) означають, що для опуклої функції $f(x)$, яка задовольняє умову (2.1), h_k визначає величину максимального зміщення в напрямку нормованого антисубградієнта. Цим гарантується, що кут між антисубградієнтом і напрямком від точки x_{k+1} до точки мінімуму не буде тупим.

Обгрунтування оцінки швидкості збіжності субградієнтного методу з кроком Поляка для довільних опуклих функцій та опуклих функцій з гострим мінімумом була проведена Б.Т. Поляком в (1983). Наведемо ці два результати у вигляді теорем 2.2 та 2.3 без доведення.

Теорема 2.2 (Поляк 1983). Якщо опукла функція $f(x)$ задовольняє нерівність (2.1), тоді справедлива рівність $\lim\limits_{k\to\infty}\sqrt{k}\left(f(x_k)-f^*\right)=0$.

Згідно з теоремою 2.2 субградієнтний метод з кроком Поляка збігається зі швидкістю $O\left(1/\sqrt{k}\right)$ у випадку довільної опуклої функції. Якщо ж функція $f(x)$ має гострий мінімум, то можна досягти геометричної швидкості збіжності методу, що і демонструє теорема 2.3.

Теорема 2.3 (Поляк 1983). Нехай функція $f(x)$ має гострий мінімум, тобто для неї виконується нерівність $f(x)-f^*\geq\alpha\left\|x-x^*\right\|$. Тоді субградієнтний метод з кроком Поляка збігається зі швидкістю геометричної прогресії зі знаменником $q=1-\alpha^2/C^2$, де C – константа, що обмежує норму субградієнта $g_f(x)$.

2.2 Зміщення по опуклості в субградієнтному методі з кроком Поляка

Розглянемо тепер більш широкий клас опуклих функцій $f(x)$, для субградієнтів $g_f(x)$ яких виконується нерівність

$$\left(x-x^*,g_f(x)\right)\geq m\left(f(x)-f^*\right),\ \forall x\in R^n,\ \text{де}\ m\geq 1, \qquad (2.6)$$

яка є узагальненням нерівності 2.1. Тут скалярний параметр m задає величину максимального зміщення по опуклості функції $f(x)$ та вводиться для того, щоб врахувати спеціальні класи опуклих функцій. Наприклад, для кусочно-лінійних негладких функцій $m=1$, для квадратичних гладких $m=2$, для функцій виду

$$f(x)=\sum_{i=1}^{k}\left|\sum_{j=1}^{n}a_{ij}x_j-b_i\right|^p,$$

де $p>1$, значення $m=p$. Параметр m можна активно використовувати для диференційовних однорідних з показником σ опуклих функцій. Для них виконується рівність $\sigma\left(f(x)-f^*\right)=\left(x-x^*,g_f(x)\right)$, отже параметр m можна вибрати $m=\sigma>1$. Якщо ж розглядається довільна опукла функція параметр m рівний одиниці.

Якщо опукла функція $f(x)$ задовольняє умову (2.6) та значення f^* відоме, то для знаходження точки $x_\varepsilon^*\in R^n$ такої, що $f\left(x_\varepsilon^*\right)\le f^*+\varepsilon$, можна використати такий ітеративний метод.

Ініціалізація. Нехай f^* та $m\ge 1$ відомі. Вибираємо початкову точку $x_0\in R^n$, величину $\varepsilon>0$ та переходимо до наступної ітерації з величиною x_0.

Ітеративний процес. Нехай точку $x_k\in R^n$ знайдено на k-й ітерації. Для переходу до $(k+1)$-ї ітерації виконаємо такі дії.

A1. Обчислимо $f(x_k)$ та $g_f(x_k)$. Якщо $f(x_k)-f^*\le\varepsilon$, тоді STOP $\left(k^*=k,\ x_\varepsilon^*=x_k\right)$.

A2. Обчислимо наступну точку

$$x_{k+1}=x_k-h_k\frac{g_f(x_k)}{\left\|g_f(x_k)\right\|},\quad h_k=\frac{m\left(f(x_k)-f^*\right)}{\left\|g_f(x_k)\right\|}. \tag{2.7}$$

A3. Переходимо до $(k+1)$-ї ітерації з точкою x_{k+1}.

Крок h_k тут задає величину максимального зміщення в напрямку нормованого антисубградієнта, при якому для опуклої функції $f(x)$ умова (2.6) гарантує, що кут між антисубградієнтом і напрямком від точки x_{k+1} до точки мінімуму не буде тупим.

Теорема 2.4 є узагальненням теореми 2.1 з введеним параметром m і гарантує, що в методі (2.7) відстань до точки мінімуму монотонно зменшується.

Наведемо цю теорему з доведенням для цілісності викладення матеріалу дисертації.

Теорема 2.4 (Стецюк 2012б). Послідовність $\{x_k\}_{k=0}^{k^*-1}$, породжена методом (2.7), задовольняє такі нерівності:

$$\|x_{k+1}-x^*\|^2 \le \|x_k-x^*\|^2 - \frac{m^2\left(f(x_k)-f^*\right)^2}{\|g_f(x_k)\|^2},\ \ k=0,1,2,... \qquad (2.8)$$

Доведення. З пункту А2 ітеративного процесу для довільного $k\ (0\le k\le k^*-1)$ маємо:

$$\|x_{k+1}-x^*\|^2 = \left\|x_k-x^*-h_k\frac{g_f(x_k)}{\|g_f(x_k)\|}\right\|^2 = \|x_k-x^*\|^2 - 2h_k\left\|\frac{\left(x_k-x^*,g_f(x_k)\right)}{\|g_f(x_k)\|}\right\| +$$

$$+h_k^2\left\|\frac{g_f(x_k)}{\|g_f(x_k)\|}\right\|^2 = \|x_k-x^*\|^2 - 2h_k\frac{\left(x_k-x^*,g_f(x_k)\right)}{\|g_f(x_k)\|}+h_k^2.$$

Зважаючи на те, що з (2.6) випливає

$$\frac{\left(x_k-x^*,g_f(x_k)\right)}{\|g_f(x_k)\|} \ge \frac{m\left(f(x_k)-f^*\right)}{\|g_f(x_k)\|} = h_k,$$

маємо:

$$\|x_{k+1}-x^*\|^2 = \|x_k-x^*\|^2 - 2h_k\frac{\left(x_k-x^*,g_f(x_k)\right)}{\|g_f(x_k)\|}+h_k^2 \le$$

$$\le \|x_k-x^*\|^2 - 2h_k^2+h_k^2 = \|x_k-x^*\|^2 - \left(\frac{m\left(f(x_k)-f^*\right)}{\|g_f(x_k)\|}\right)^2,$$

звідки випливають нерівності (2.8). Теорему доведено.

Крім того, задовольняються нерівності

$$\left(x^*-x_{k+1},-g_f(x_k)\right)\ge 0,\ \ k=0,1,....\qquad (2.9)$$

Дійсно, використовуючи (2.6) та (2.7), маємо:

$$\left(x^* - x_{k+1}, -g_f(x_k)\right) = \left(x_{k+1} - x^*, g_f(x_k)\right) = \left(x_k - x^* - h_k \frac{g_f(x_k)}{\left\|g_f(x_k)\right\|}, g_f(x_k)\right) =$$

$$= \left(x_k - x^*, g_f(x_k)\right) - \left(h_k \frac{g_f(x_k)}{\left\|g_f(x_k)\right\|}, g_f(x_k)\right) = \left(x_k - x^*, g_f(x_k)\right) - h_k \frac{\left\|g_f(x_k)\right\|^2}{\left\|g_f(x_k)\right\|} =$$

$$= \left(x_k - x^*, g_f(x_k)\right) - \frac{m\left(f(x_k) - f^*\right)}{\left\|g_f(x_k)\right\|} \frac{\left\|g_f(x_k)\right\|^2}{\left\|g_f(x_k)\right\|} =$$

$$= \left(x_k - x^*, g_f(x_k)\right) - m\left(f(x_k) - f^*\right) \geq 0.$$

Субградієнтний метод з кроком Поляка та параметром $m > 1$ володіє такою ж швидкістю збіжності, як і класичний субградієнтний метод з кроком Поляка, а саме: збіжність зі швидкістю $O\left(1/\sqrt{k}\right)$ для довільних опуклих функцій та зі швидкістю геометричної прогресії для опуклих функцій з гострим мінімумом. Для доведення цих тверджень знадобиться така лема.

Лема 2.1 (Поляк 1983, с. 121). Субградієнти опуклої функції $f(x)$ є обмеженими на довільній обмеженій множині або множині виду $\left\{x : f(x) \leq \alpha\right\}$.

Теорема 2.5 встановлює швидкість збіжності субградієнтного методу з кроком Поляка та параметром $m > 1$ у випадку довільної опуклої функції, а теорема 2.6 – у випадку опуклої функції з гострим мінімумом.

Теорема 2.5. *Я*кщо опукла функція $f(x)$ задовольняє нерівність (2.6), тоді справедлива рівність $\lim\limits_{k\to\infty} \sqrt{k}\left(f(x_k) - f^*\right) = 0$.

Доведення. З нерівності (2.8) теореми 2.4 випливає, що послідовність $h_k^2 = \dfrac{m^2\left(f(x_k) - f^*\right)^2}{\left\|g_f(x_k)\right\|^2}$ збіжна. Враховуючи обмеженість послідовності x_k ($\left\|x_k - x^*\right\| \leq \left\|x_0 - x^*\right\|$), за лемою 2.1 маємо $\left\|g_f(x_k)\right\| \leq C_1$, де C_1 – константа, що обмежує норму субградієнта $g_f(x)$. Звідси випливає збіжність послідовності

$\left(f\left(x_k\right)-f^*\right)^2$. Припустимо, що $\underline{\lim}_{k\to\infty}\sqrt{k}\left(f\left(x_k\right)-f^*\right)>0$. Тоді $f\left(x_k\right)-f^*>a/\sqrt{k}$ при досить великих k та $a>0$, а це суперечить збіжності послідовності $\left(f\left(x_k\right)-f^*\right)^2$. Звідси випливає, що $\underline{\lim}_{k\to\infty}\sqrt{k}\left(f\left(x_k\right)-f^*\right)=0$. Теорему 2.5 доведено.

Теорема 2.6. Нехай функція $f\left(x\right)$ має гострий мінімум, тобто для неї виконується нерівність $f\left(x\right)-f^*\geq\alpha\left\|x-x^*\right\|$. Тоді метод (2.7) збігається зі швидкістю геометричної прогресії зі знаменником $q_1=1-\left(\frac{m\alpha}{C_1}\right)^2$, де C_1 – константа, що обмежує норму субградієнта $g_f\left(x\right)$.

Доведення. Враховуючи справедливість нерівності (2.8) та нерівності для $f\left(x\right)$ в умові теореми, маємо:

$$\left\|x_{k+1}-x^*\right\|^2\leq\left\|x_k-x^*\right\|^2-\frac{m^2\left(f\left(x_k\right)-f^*\right)^2}{\left\|g_f\left(x_k\right)\right\|^2}\leq\left\|x_k-x^*\right\|^2-\frac{m^2\alpha^2\left\|x_k-x^*\right\|^2}{\left\|g_f\left(x_k\right)\right\|^2}.$$

Використовуючи нерівність $\left\|g_f\left(x_k\right)\right\|\leq C_1$ з теореми 2.5, маємо:

$$\left\|x_{k+1}-x^*\right\|^2\leq\left\|x_k-x^*\right\|^2-\frac{m^2\alpha^2\left\|x_k-x^*\right\|^2}{\left\|g_f\left(x_k\right)\right\|^2}\leq\left\|x_k-x^*\right\|^2-m^2\frac{\alpha^2}{C_1^2}\left\|x_k-x^*\right\|^2=q_1\left\|x_k-x^*\right\|^2,$$

де $q_1=1-\left(\frac{m\alpha}{C_1}\right)^2$, що й означає збіжність методу зі швидкістю геометричної прогресії зі знаменником q_1. Теорему 2.6 доведено.

2.3 Octave-функція PolyakA

Для знаходження наближення x_ε^* точки мінімуму опуклої функції $f\left(x\right)$ від n змінних скористаємось програмою *PolyakA*, яка є реалізацією субградієнтного методу з кроком Поляка з параметром $m\geq1$. Вона написана

некомерційною мовою Octave. Для обчислення значення функції $f(x)$ та її субградієнта $g_f(x)$ у точці x скористаємось octave-функцією **function [f, g] = calcfg(x)**. Назва *calcfg* є параметром, замість якого в процесі виконання програми підставляється назва octave-функції, яка обчислює значення функції та її субградієнта в заданій точці. Назва функції може бути довільною, яку дає змогу синтаксис мови Octave.

Вхідними параметрами програми *PolyakA* є: **calcfg** – посилання на функцію для обчислення $f(x)$ та $g_f(x)$; **x0** – початкова точка; **fstar** – значення функції в точці мінімуму; **m** ($m \geq 1$) – параметр, що визначає величину зміщення в напрямку антисубградієнта, тобто виконання умови (2.6); **epsf** – точність, з якою необхідно знайти наближення; **maxitn –** параметр зупинки (разом з **epsf**).

Вихідними параметрами програми є: **x** – точка мінімуму, знайдена програмою; **f** – значення функції $f(x)$ в точці **x**; **itn** – кількість ітерацій, які виконала програма та **info** – код повернення, значенням якого визначається статус точки **x** на ітерації **itn**=k. Значення **info**=0 означає, що знайдено точку x_k, для якої $f(x_k) - f^* \leq \varepsilon_f$; **info**=4 означає, що **itn** > **maxitn** (ітераційний процес перевищив максимальну кількість ітерацій).

Код програми *PolyakA* подано нижче.

```
function [x,f,itn,info] = PolyakA(calcfg,x0,fstar,m,          #row01
                                  epsf,maxitn,itnp);          #.....
itn=0; x=x0; [f,g] = calcfg(x); dg=norm(g);                   #row02
if(itnp>0)                                                    #row03
   printf("itn %4d f %14.6e \n", itn, f); # xprint = x',      #.....
endif                                                         #.....
for(itn = 1:maxitn)                                           #row04
   if(f-fstar < epsf) info = 0; return; endif                 #row05
   g1=g/dg; hs=m*(f-fstar)/dg;                                #row06
   x -= hs * g1;                                              #row07
   [f,g] = calcfg(x); dg=norm(g);                             #row08
   if(mod(itn,itnp)==0)                                       #row09
```

```
      printf("itn %4d f %14.6e \n",itn,f); # xprint = x',     #.....
   endif                                                      #.....
endfor                                                        #row10
info = 4;                                                     #row11
endfunction                                                   #row12
```

Ітеративний процес виконується в циклі for (рядки 4−10), де на k-й ітерації субградієнт $g_f(x_k)$ зберігається як вектор-стовпчик $\boldsymbol{g}$, а нормований субградієнт $g_f(x_k)$ – як вектор-стовпчик $\boldsymbol{g1}$. У циклі for зупинка відбувається тоді, коли відхилення функції від оптимального значення стає меншим ніж ε_f (рядок 5).

2.4 Обчислювальні експерименти

В цьому підрозділі розглянемо результати застосування класичного субградієнтного методу з кроком Поляка ($m=1$) та його узагальненого варіанту ($m>1$) для мінімізації деяких гладких та негладких опуклих функцій. Оскільки ці два методи відрізняються лише значенням параметра m, надалі називатимемо їх методом А, уточнюючи в дужках яке значення параметра m використовується.

Як критерій зупинки використаємо умову $f(x_k)-f^*\le\varepsilon$; для довільного малого $\varepsilon>0$ це дає змогу нам знайти таку точку $x_\varepsilon^*=x_k$, що $f(x_\varepsilon^*)\le f^*+\varepsilon$. Експерименти були проведені на комп’ютері Intel Core i3-8130U CPU 2.20 GHz 2.21 GHz з операційною системою Windows 10/64x з використанням GNU Octave версії 5.2.0.

Приклад 2.1 (квадратична функція $f_1(x_1,x_2)=x_1^2+tx_2^2$, $t>>1$). Для неї $x^*=(0,0)^T$ та $f^*=0$. Для обчислення $f(x)$ та $g_f(x)$ використаємо таку octave-функцію:

```
function [f,g] = squad(x)
global t;
```

```
f = x(1,1)*x(1,1) + t*x(2,1)*x(2,1);
g(1,1) = 2*x(1,1);
g(2,1) = 2*t*x(2,1);
endfunction
```

Використовуючи значення параметру $t=6$, метод А ($m=2$) знаходить наближення до точки мінімуму з точністю $\varepsilon=10^{-6}$ за 16 ітерацій, породжуючи таку послідовність точок: $x_0=(1.00,1.00)^T$, $x_1=(0.8108,-0.1351)^T$, $x_2=(0.3378,0.3378)^T$, $x_3=(0.2739,-0.0457)^T$, $x_4=(0.1141,0.1141)^T$, $x_5=(0.0925,-0.0154)^T$, …. (див. рис. 2.1).

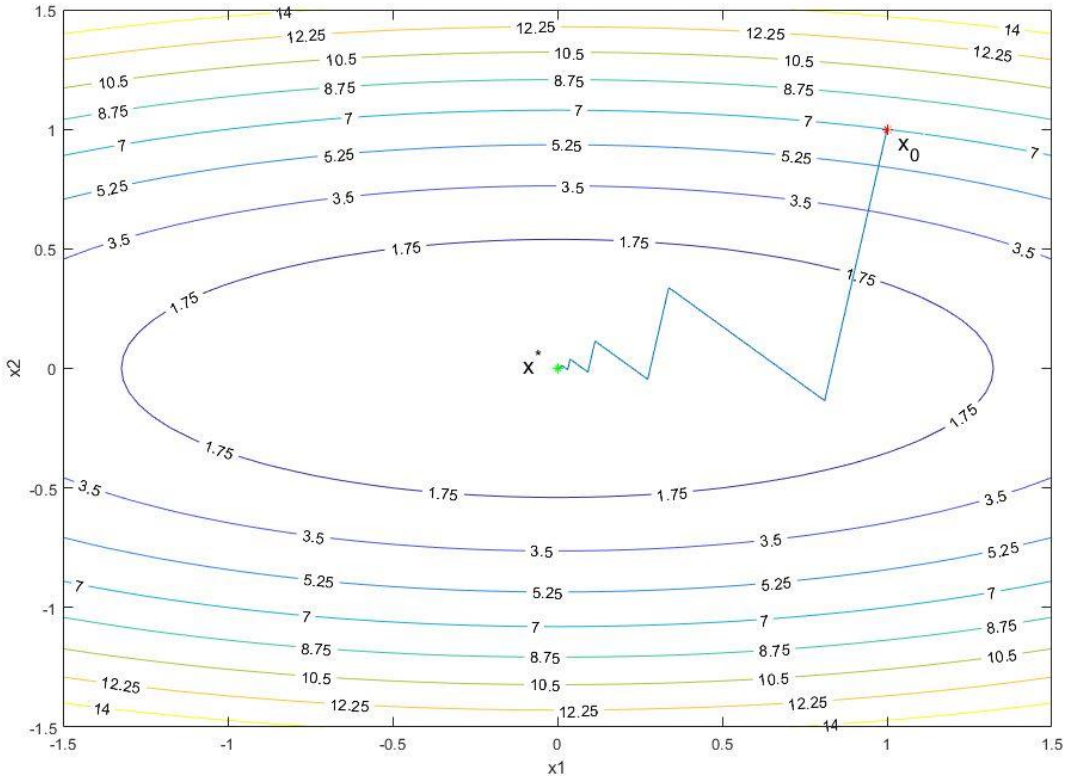


Рис. 2.1. Траєкторія методу А ($m=2$) пошуку наближення до точки мінімуму квадратичної функції $f_1(x_1,x_2)=x_1^2+6x_2^2$: $x_0=(1,1)^T$, $x^*=(0,0)^T$, $f^*=0$.

Однак траєкторія класичного субградієнтного методу з кроком Поляка (метод А при $m=1$) не є зигзагоподібною та прямує до точки мінімуму суттєво повільніше (див. рис. 2.2). Після семи ітерацій методу А ($m=1$) точка x_7

лежить на відстані не більше ніж 0.13 від точки мінімуму, тоді як у методі А ($m=2$) – не більше ніж 0.032, тобто в 4 рази ближче.

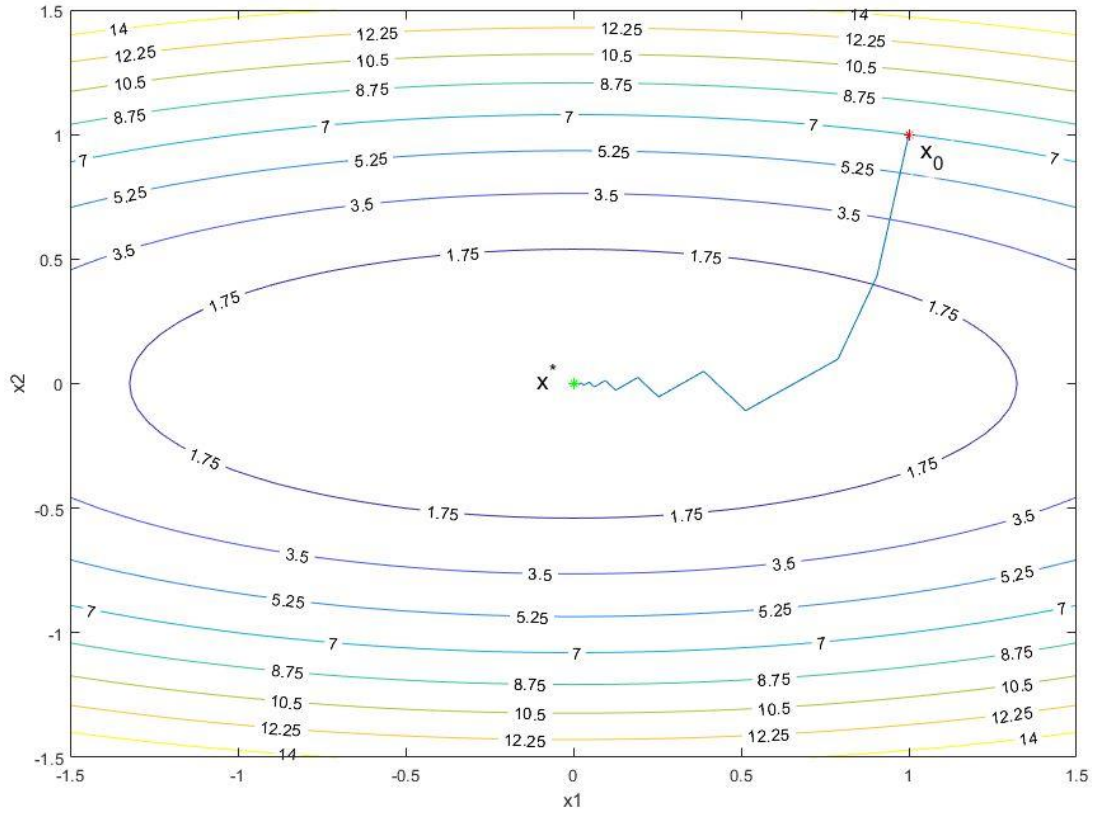


Рис. 2.2. Траєкторія методу А ($m=1$) пошуку наближення до точки мінімуму квадратичної функції $f_1(x_1,x_2)=x_1^2+6x_2^2$: $x_0=(1,1)^T$, $x^*=(0,0)^T$, $f^*=0$.

Для більш детального порівняння методу А при $m=1$ та $m=2$ проаналізуємо кількості ітерацій цих методів у випадку функції $f_1(x_1,x_2)$ для трьох різних значень параметра $t=100,1000,10000$ і точностей $10^{-1}-10^{-10}$, які наведені в таблиці 2.1. Значення в дужках є числом ітерацій методу А при $m=1$, поза дужками – при $m=2$.

Таблиця 2.1

Кількості ітерацій методу А при $m=1$ та $m=2$ для знаходження наближення x_ε^* функції $f_1(x_1,x_2)=x_1^2+tx_2^2:\ x_0=(1,1)^T,\ x^*=(0,0)^T,\ f^*=0$

t \ ε_f	10^{-1}	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}
100	6 (9)	10 (29)	16 (68)	22 (98)	28 (135)	36 (169)
1000	6 (97)	10 (127)	16 (230)	22 (390)	30 (485)	36 (543)
10000	6 (139)	10 (295)	16 (646)	22 (1003)	30 (1332)	36 (1947)

Аналізуючи таблицю 2.1, можна зробити висновок, що кількість ітерацій методу А ($m=1$) суттєво зростає при підвищенні точності ε_f та значення параметра t. В той же час зростання ітерацій методу А ($m=2$) у міру збільшення точності не можна назвати значним: від 6 ітерацій при точності 10^{-1} до 36 при точності 10^{-10}, причому зростання числа ітерацій при збільшенні значення параметра t не спостерігається взагалі. Наприклад, 22 ітерації при точності 10^{-6} зберігаються для всіх трьох значень параметра t. Такі результати демонструють доцільність використання параметра $m>1$ для поліноміальних гладких функцій. Розглянемо, наприклад, квадратичну функцію однієї змінної $f(x)=x^2$. У такому випадку субградієнтний метод з кроком Поляка, фактично, співпадає з методом Ньютона та досягає точки мінімуму за 1 ітерацію, якщо $m=2$. Якщо ж використовувати значення $m=1$ метод збігається зі швидкістю методу дихотомії, тобто геометричної прогресії, отже повільніше. Втім, для степеневих функцій вищих степенів падіння кількості ітерацій може бути ще вагомішим. Розглянемо приклад, що ілюструє таку ситуацію.

Приклад 2.2 (опуклі функції $f_2(x_1,x_2)=(x_1+1.001x_2)^4+(1.001x_1+x_2)^4$ та $f_3(x_1,x_2)=x_1^4+10000x_2^4$). Для них $x_0=(1,1)^T$, $x^*=(0,0)^T$ та $f^*=0$. Будемо знаходити наближення x_ε^* з точністю $\varepsilon_f=10^{-20}$. Для цих функцій варто використовувати значення параметра $m=4$. Кількості ітерацій обох методів для цих функцій наведено в таблиці 2.2.

Таблиця 2.2

Кількості ітерацій методу А при $m=1,2,4$ для знаходження x_ε^* функцій $f_2(x_1,x_2)=(x_1+1.001x_2)^4+(1.001x_1+x_2)^4$ та $f_3(x_1,x_2)=x_1^4+10000x_2^4$

$\varepsilon_f=10^{-20}$	$m=1$	$m=2$	$m=4$
$f_2(x_1,x_2)$	45	19	2
$f_3(x_1,x_2)$	50	36	4

Як бачимо, метод А ($m=1$) потребує 45 та 50 ітерацій для знаходження наближення до точки мінімуму функцій $f_2(x_1,x_2)$ та $f_3(x_1,x_2)$ відповідно (колонка 2). В той же час, використовуючи метод А зі значеннями параметра $m=2$ та $m=4$, ми отримуємо суттєве падіння кількості ітерацій – всього 2 ітерації (проти 45) для функції $f_2(x_1,x_2)$ та 4 (проти 50) для функції $f_3(x_1,x_2)$. Геометрично це означає, що при $m=4$ розмір кроку в 4 рази більший, ніж при $m=1$, тому метод А збігається значно швидше: кількість ітерацій зменшилась у понад 22 рази.

Приклад 2.3 (функція $f_4(x)=\sum_{i=1}^{k}\left|\sum_{j=1}^{n}a_{ij}x_j-b_i\right|^p$, $p\geq 1$). Цю функцію ми вже наводили як приклад на початку підрозділу 2.1, описуючи параметр m: для неї значення $m=p$. Тут $A=\|a_{ij}\|_{i,j=1}^{l,n}$ – довільна $l\times n$-матриця та b – l-вимірний

вектор, для компонентів якого виконується рівність $b_i = \sum_{j=1}^{n} a_{ij},\ i = 1,..,l$. Для функції $f_4^* = 0$ і, якщо матриця A має повний ранг, функція має єдину точку мінімуму $x^* = (1,1,..,1)^T$. Для тестового прикладу візьмемо матрицю A розмірності 500×100, яка має вигляд

$$A = \begin{pmatrix} 100 & 0 & 0... & 0 \\ 0 & 100 & 0... & 0 \\ & A_1 & & \end{pmatrix},$$

де A_1 – 498×100-матриця, яка утворюється з використанням генератора випадкових чисел на інтервалі $[0,3]$.

Функція $f_4(x)$ є негладкою при $p = 1$ та гладкою при $p > 1$. Її субградієнт $g_{f_4}(x)$ обчислюється за формулою

$$g_{f_4}(x) = \sum_{j=1}^{m} \left(p \left| a_j x - b_j \right|^{p-1} sign\left(a_j x - b_j \right) a_j^T \right),$$

де a_j – вектор-рядок матриці A з номером $j = \overline{1,m}$.

При різних значеннях параметрів p та m субградієнтний метод з кроком Поляка демонструє доволі неоднорідну кількість ітерацій. Для кращої ілюстрації цього ефекту ітерації методів наведемо в двох таблицях: для значень параметра p з інтервалу $[1,2]$ (табл. 2.3) та $[3,15]$ (табл. 2.4) для точностей $10^{-1} - 10^{-10}$. Поза дужками наведені ітерації методу А ($m = 1$), а в дужках – ітерації методу А, в якому $m = p$. Оскільки при $m = 1$ метод А збігається з класичним субградієнтним методом з кроком Поляка, в колонці 2 наведено лише один набір ітерацій.

Таблиця 2.3

Кількості ітерацій методу А при $m=1$ та $m=p$ для мінімізації функції $f_4(x)$ для $p\in[1,2]$

ε_f \ p	**1**	**1.2**	**1.4**	**1.6**	**1.8**	**2**
10^{-1}	68	260 (98)	131 (66)	157 (80)	108 (106)	93 (211)
10^{-2}	110	357 (146)	204 (98)	223 (128)	164 (164)	158 (353)
10^{-3}	142	442 (196)	271 (134)	303 (182)	222 (230)	215 (515)
10^{-4}	189	523 (242)	349 (176)	378 (240)	271 (304)	252 (695)
10^{-5}	231	618 (288)	471 (216)	437 (300)	315 (380)	319 (887)
10^{-6}	267	728 (334)	554 (260)	481 (362)	427 (460)	334 (1085)
10^{-7}	299	927 (396)	645 (296)	542 (422)	461 (538)	392 (1287)
10^{-8}	344	1062 (456)	760 (340)	634 (484)	525 (618)	416 (1491)
10^{-9}	380	1189 (485)	852 (382)	698 (546)	568 (698)	476 (1695)
10^{-10}	414	1302 (513)	906 (424)	819 (606)	661 (778)	544 (1901)

Дані табл. 2.3 показують, що при $p\in(1,1.6)$ метод А ($m=p$) працює краще, ніж при $m=1$, однак вже при $p=1.8$ та $p=2$ кількість його ітерацій зростає в 2-3 рази. Таке явище спостерігається майже для всіх точностей ε_f. Наприклад, при $\varepsilon_f=10^{-1}$ та $p=2$ маємо 93 та 211 ітерацій методу при $m=1$ та $m=2$ відповідно; при $\varepsilon_f=10^{-10}$ – 544 та 1901 ітерація відповідно. Однак, в околі значення $p=1.8$ виникають незначні відхилення: 108 та 106 ітерацій методу при $m=1$ та $m=1.8$ відповідно при точності 10^{-1}, однак 661 та 778 ітерацій при точності 10^{-10}, тобто вже класичний метод Поляка ($m=1$) потребує меншої кількості ітерацій. Ймовірно, в околі значення $p=1.8$ лежить точка

«зрівнювання» кількості ітерацій двох методів, після якої доцільно використовувати метод А зі значенням параметру $m = p$.

Таблиця 2.4

Кількості ітерацій методу А при $m = 1$ та $m = p$ для мінімізації функції $f_4(x)$ для $p \in [3,15]$

ε_f \ p	**3**	**4**	**6**	**8**	**10**	**15**
10^{-1}	53 (234)	53 (118)	56 (45)	50 (30)	58 (24)	83 (20)
10^{-2}	105 (376)	75 (188)	91 (71)	68 (45)	72 (34)	85 (26)
10^{-3}	132 (552)	108 (274)	113 (103)	85 (63)	85 (45)	92 (32)
10^{-4}	205 (752)	172 (376)	140 (141)	118 (82)	99 (58)	102 (38)
10^{-5}	271 (970)	205 (492)	147 (185)	145 (103)	119 (72)	121 (46)
10^{-6}	321 (1202)	259 (620)	168 (235)	178 (129)	153 (88)	135 (54)
10^{-7}	364 (1446)	313 (758)	231 (291)	197 (157)	176 (105)	174 (62)
10^{-8}	444 (1698)	353 (900)	256 (351)	218 (189)	223 (125)	222 (72)
10^{-9}	498 (1954)	400 (1048)	313 (417)	266 (225)	248 (146)	256 (80)
10^{-10}	553 (2212)	437 (1200)	353 (487)	302 (263)	291 (169)	277 (92)

Таблиця 2.4 це підтверджує: метод А ($m = 1$) і далі працює ефективніше при $p \in [3,4]$, однак вже при $p = 6$ метод А ($m = 6$) потребує меншої кількості ітерацій, яка і далі спадає зі зростанням значення p. Наприклад, для точності 10^{-3} та $p = 3$ метод А ($m = 1$) потребує 132 ітерації проти 552 в метода А ($m = 3$), тобто в 3 рази менше; однак при $p = 15$ ситуація змінюється на протилежну: 92 ітерації проти 32 у метода А ($m = 15$). В цьому випадку точка «зрівнювання» кількості ітерацій лежить в околі значення $p = 5.5$, після якого метод А ($m = p$) працює швидше.

Отже, можна стверджувати, що для функції $f_4(x)$ класичний субградієнтний метод з кроком Поляка ($m=1$) працює ефективно лише на інтервалі $p\in(1.6,4)$. При інших розглянутих значеннях параметра p субградієнтний метод з кроком Поляка ($m=p$) демонструє меншу кількість ітерацій, тому доцільно використовувати саме його.

Приклад 2.4 (квадратична функція $f_5(x)=(x-e)^T D(x-e)$). Для демонстрації роботи методу А при великій кількості вхідних даних розглянемо мінімізацію функції $f_5(x)$ векторного аргументу $x\in\mathbb{R}^n$, де $n=10\,000\,000$, $e=(1,...,1)^T\in\mathbb{R}^n$. Функція визначається діагональною матрицею D, що містить на діагоналі вектор $d\in\mathbb{R}^n$. Кожен елемент вектора d утворюється за допомогою генератора псевдовипадкових чисел з певного інтервалу, що задається параметром α. Регулювання цього інтервалу дозволяє змінювати $\lambda_{\max}$ та $\lambda_{\min}$ – максимальне та мінімальне власні числа матриці D, а отже її число обумовленості (величину $\lambda=\lambda_{\max}/\lambda_{\min}$). Для функції $f_5(x)$ точка мінімуму $x^*=(1,...,1)^T$ та $f^*=0$. Оскільки функція $f_5(x)$ квадратична для її мінімізації можна використати значення параметра $m=2$. Отримані результати порівняємо з результатами роботи класичного субградієнтного методу з кроком Поляка ($m=1$). Точність знаходження мінімального значення функції $\varepsilon_f=10^{-20}$. Результати роботи методів наведено в таблиці 2.5.

Таблиця 2.5

Кількості ітерацій та значення функції в точках, отриманих за допомогою методу А для мінімізації функції $f_5(x)$ при $m=1,2$ для різних $\lambda=\lambda_{\max}/\lambda_{\min}$

№	$\lambda_{\max}$	$\lambda_{\min}$	$m=1$		$m=2$		q
			itr	$f(x_k^*)$	*itr*	$f(x_k^*)$	
1	3	1	47	6.0030e-21	43	7.2469e-21	1.09

Продовження табл. 2.5

2	2	1	47	3.7858e-21	28	3.6754e-21	1.67
3	1.5	1	47	2.8335e-21	20	8.1206e-22	2.35
4	1.1	1	46	8.6471e-21	11	1.7920e-21	4.18
5	1.01	1	46	8.1212e-21	7	1.7769e-22	6.57

Стовпці 2,3 таблиці 2.5 містять максимальне та мінімальне власні числа матриці D, стопці 4,5 – кількості ітерацій та значення функції в отриманій точці при використанні параметра $m=1$ відповідно, стовпці 6,7 – ті ж самі величини для випадку $m=2$. Останній стовпчик містить значення коефіцієнта q, що є відношенням ітерацій методу А при $m=1$ та $m=2$. Дані таблиці показують, що навіть при значному $n=10\,000\,000$ метод А при $m=1$ знаходить розв'язок з точністю $\varepsilon_f=10^{-20}$ не більше ніж за 47 ітерацій при $\lambda=3$ (випадок 1). Використання значення $m=2$ дозволяє зменшити це число до 43 ітерацій. Також варто відмітити, що класичний субградієнтний метод з кроком Поляка ($m=1$) майже не реагує на зменшення числа обумовленості матриці: 47 ітерацій при $\lambda=3$ (випадок 1) та 46 ітерацій при $\lambda=1.01$ (випадок 5). Однак, використовуючи значення $m=2$, кількість ітерацій вдається скоротити до 7 при $\lambda=1.01$. Значення коефіцієнта q в цьому випадку рівне 6.57, тобто кількість ітерацій зменшилась більше ніж у 6 разів. Octave-код процедури **squad1** для обчислення значення функції $f_5(x)$ та її субградієнта, Octave-код програми запуску прикладу 2.4 та її лістинг наведено в додатку А.

Приклад 2.5 (функція $f_6(x)=x^THx+b^Tx+c$). Розглянемо ще один тестовий приклад з великою кількістю вхідних даних. Мінімізується квадратична функція $f_6(x)$ векторного аргументу $x\in\mathbb{R}^{1000}$, що задається симетричною матрицею H розмірності 1000×1000, вектором $b\in\mathbb{R}^{1000}$ та скаляром $c\in\mathbb{R}$. Елементи матриці H утворюються з використанням генератора псевдовипадкових чисел з певного інтервалу, який задається параметром α, що дозволяє регулювати величини $\lambda_{\max}$ та $\lambda_{\min}$ цієї матриці.

Елементи вектора b обираються так, щоб точка мінімуму $x^* = (1,...,1)^T \in \mathbb{R}^{1000}$ та $f^* = 0$. Для функції $f_6(x)$ можна використати значення параметру $m = 2$, тому проведемо обчислювальні експерименти для значень $m = 1,2$. Точність знаходження мінімального значення функції $\varepsilon_f = 10^{-6}$. Результати експериментів наведено в таблиці 2.6.

Таблиця 2.6

Кількості ітерацій та значення функції в точках, отриманих за допомогою методу А для мінімізації функції $f_6(x)$ при $m = 1,2$ для різних $\lambda = \lambda_{\max} / \lambda_{\min}$

№	λ	$m=1$		$m=2$		q
		itr	$f(x_k^*)$	*itr*	$f(x_k^*)$	
1	1.6307e+06	12	2.6286e-07	2	4.8850e-15	6.0
2	3.3332e+04	12	5.3647e-07	2	2.9286e-11	6.0
3	5.3291e+03	14	5.4279e-07	2	1.8664e-08	7.0
4	4.0023e+03	16	8.9535e-07	2	6.4400e-07	8.0
5	3.7752e+03	70	8.9798e-07	3	2.1280e-07	23.3

Таблиця 2.6 містить кількості ітерацій методу А та значення функції в отриманій точці при використанні параметра $m = 1$ (стовпці 3,4) та $m = 2$ (стовпці 5,6) для різних відношень максимального та мінімального власних чисел матриці H (стовпець 2). Наведено також значення коефіцієнта q, що є відношенням числа ітерацій методу А при $m = 1$ та $m = 2$ (стовпець 7).

Результати експериментів показують, що при зменшенні величини $\lambda = \lambda_{\max} / \lambda_{\min}$ кількість ітерацій класичного субградієнтного методу з кроком Поляка зростає від 12 в першому випадку до 70 в п'ятому випадку. Використання параметра $m = 2$ дозволяє знаходити розв'язок за 2 ітерації в усіх випадках, окрім останнього, де необхідно 3 ітерації. Відмітимо, що окрім суттєвого скорочення кількості ітерацій (більше ніж у 23 рази у випадку 5), використання параметра $m = 2$ дозволяє знаходити розв'язок зі значно більшою

точністю (стовпці 4 та 6). Octave-код процедури **squad2** для обчислення значення функції $f_6(x)$ та її субградієнта, Octave-код програми запуску прикладу 2.5 та її лістинг наведено в додатку Б.

2.5 Висновки до другого розділу

1. Розглянуто класичний субградієнтний метод з кроком Поляка для мінімізації опуклих функцій при відомому оптимальному значенні. Доведено теореми про монотонне зменшення відстані до точки мінімуму та швидкість збіжності методу, яка дорівнює $O\left(1/\sqrt{k}\right)$ у випадку довільної опуклої функції та швидкості геометричної прогресії для опуклої функції з гострим мінімумом.

2. Розглянуто модифікацію класичного субградієнтного методу з кроком Поляка скалярним параметром $m>1$, який задає величину максимального зміщення по опуклості функції $f(x)$ і дає змогу врахувати спеціальні класи опуклих функцій. Доведено теореми про монотонне зменшення відстані до точки мінімуму та швидкість збіжності модифікованого методу, яка дорівнює $O\left(1/\sqrt{k}\right)$ у випадку довільної опуклої функції та швидкості геометричної прогресії для опуклої функції з гострим мінімумом.

3. Розглянуто п’ять тестових прикладів з мінімізації гладких та негладких яружних опуклих функцій з використанням описаних методів. Результати обчислювальних експериментів з цими прикладами показали, що використання скалярного параметра $m>1$ дає змогу суттєво прискорити класичний субградієнтний метод з кроком Поляка при мінімізації функцій з суттєво витягнутими поверхнями рівня.

РОЗДІЛ 3. СУБГРАДІЄНТНИЙ МЕТОД З КРОКОМ ПОЛЯКА ТА ПЕРЕТВОРЕННЯ ПРОСТОРУ

У третьому розділі розглядається субградієнтний метод з кроком Поляка у перетвореному просторі змінних та його модифікація скалярним параметром $m>1$ для знаходження точки мінімуму гладких та негладких яружних опуклих функцій при відомому мінімальному значенні. Для обох методів наведено обґрунтування монотонності зменшення відстані до точки мінімуму та оцінки швидкості збіжності методів: вона дорівнює $O\left(1/\sqrt{k}\right)$ для довільних опуклих функцій та швидкості геометричної прогресії для опуклих функцій з гострим мінімумом (підрозділи 3.1 та 3.2). У підрозділі 3.4 наведено результати обчислювальних експериментів обох методів для набору гладких та негладких яружних опуклих функцій, а також описано складнощі в роботі з ними. Чисельні приклади демонструють доцільність використання операції перетворення простору та параметра $m>1$ як модифікації класичного субградієнтного методу з кроком Поляка, а також демонструють умови їхнього використання для ефективної мінімізації функцій такого типу.

3.1 Субградієнтний метод з кроком Поляка у перетвореному просторі

В основу субградієнтного методу з кроком Поляка з перетворенням простору покладена одна з ідей Н.З. Шора (Сергиенко и Стецюк 2012) для прискорення методів, використовуючи лінійні перетворення простору. Ця ідея стала основою для цілого класу методів, таких як метод еліпсоїдів і *r*-алгоритми (субградієнтні методи з розтягом простору в напрямку послідовних субградієнтів) для опуклої оптимізації (Шор 1979, Shor 1998, Стецюк 2014).

Постановка задачі залишається аналогічною до розділу 2: мінімізувати опуклу функцію $f(x), x \in R^n$ з відомим мінімальним значенням $f^* = f(x^*)$ та єдиною точкою мінімуму x^*.

Зробимо заміну змінних $x = By$, де B – невироджена $n \times n$-матриця (тобто для неї існує обернена матриця $A = B^{-1}$) та введемо функцію $\varphi(y) = f(By)$. Субградієнт $g_\varphi(y)$ опуклої функції $\varphi(y)$ в точці $y = Ax$ задовольняє умову

$$\left(y - y^*, g_\varphi(y)\right) \geq \varphi(y) - \varphi^*, \forall y \in R^n, \tag{3.1}$$

Яка є аналогом умови (2.1) для функції $f(x)$. Тут $g_\varphi(y) = B^T g_f(x)$, $\varphi^* = \varphi(y^*) = f(By^*) = f(x^*) = f^*$. Дійсно, оскільки $A = B^{-1}$ та $x = By$, нерівність (2.1) можна переписати у вигляді

$$\left(BB^{-1}(x - x^*), g_f(x)\right) = \left(A(x - x^*), B^T g_f(x)\right) \geq f(By) - f(By^*), \ \forall By \in R^n,$$

звідки випливає нерівність (3.1).

Субградієнтний метод з кроком Поляка в перетвореному просторі змінних, визначений невиродженою матрицею B (далі метод В), має такий вигляд:

$$x_{k+1} = x_k - h_k B \frac{B^T g_f(x_k)}{\left\| B^T g_f(x_k) \right\|}, \quad h_k = \frac{f(x_k) - f^*}{\left\| B^T g_f(x_k) \right\|}, \ \ k = 0,1,2.... \tag{3.2}$$

Тут h_k є кроком Поляка (Аґмона – Моцкіна – Шонберга) в перетвореному просторі змінних $y = Ax$. Це випливає з того, що в перетвореному просторі змінних метод (3.2) можна записати як субградієнтний процес

$$y_{k+1} = y_k - h_k \frac{g_\varphi(y_k)}{\left\| g_\varphi(y_k) \right\|}, \quad h_k = \frac{\varphi(y_k) - \varphi^*}{\left\| g_\varphi(y_k) \right\|}, \quad k = 0,1,2.... \tag{3.3}$$

Крок Поляка у перетвореному просторі змінних має ті ж властивості, що й крок Поляка в початковому просторі. Вони визначаються мінімальним значенням функції та нерівністю (3.1), пов’язаною з φ^*.

Відзначимо, що якщо покласти матрицю перетворення простору B рівною одиничній матриці відповідної розмірності, метод (3.2) збігається з класичним субградієнтним методом з кроком Поляка (у початковому просторі змінних).

Умови використання субградієнтного методу з кроком Поляка у перетвореному просторі збігаються з умовами для класичного методу: опуклість функції $\varphi(y)$ та виконання умови (3.1). Метод можна використати для знаходження точки $x_\varepsilon^* \in R^n$, для якої $f\left(x_\varepsilon^*\right) \le f^* + \varepsilon$. Він описується такою ітеративною процедурою.

Ініціалізація. Маємо відоме значення f^*. Виберемо початкову точку $x_0 \in R^n$, невироджену $n \times n$-матрицю B та величину $\varepsilon > 0$. Переходимо до наступної ітерації з величиною x_0.

Ітеративний процес. Нехай точку $x_k \in R^n$ знайдено на k-й ітерації. Для переходу до $(k+1)$-ї ітерації виконаємо такі дії.

B1. Обчислимо $f(x_k)$ та $g_f(x_k)$. Якщо $f(x_k) - f^* \le \varepsilon$, тоді STOP ($k^* = k$, $x_\varepsilon^* = x_k$).

B2. Обчислимо наступну точку:

$$x_{k+1} = x_k - h_k B \frac{B^T g_f(x_k)}{\left\| B^T g_f(x_k) \right\|}, \quad h_k = \frac{f(x_k) - f^*}{\left\| B^T g_f(x_k) \right\|}.$$

B3. Переходимо до $(k+1)$-ї ітерації з точкою x_{k+1}.

Теорема 3.1. Послідовність $\{x_k\}_{k=0}^{k^*-1}$, породжена методом (3.2), задовольняє нерівності

$$\left\| A\left(x_{k+1} - x^*\right) \right\|^2 \le \left\| A\left(x_k - x^*\right) \right\|^2 - \frac{\left(f(x_k) - f^*\right)^2}{\left\| B^T g_f(x_k) \right\|^2}, \quad k = 0,1,2,... \tag{3.4}$$

Доведення. Переписавши пункт B2 ітеративного процесу у вигляді

$$Ax_{k+1} = Ax_k - h_k \frac{B^T g_f(x_k)}{\left\| B^T g_f(x_k) \right\|},$$

для довільного k $(0 \le k \le k^* - 1)$ маємо:

$$\left\| A\left(x_{k+1} - x^*\right)\right\|^2 = \left\| A\left(x_k - x^*\right) - h_k \frac{B^T g_f\left(x_k\right)}{\left\| B^T g_f\left(x_k\right)\right\|}\right\|^2 = \left\| A\left(x_k - x^*\right)\right\|^2 -$$

$$-2h_k \left\| \frac{\left(A\left(x_k - x^*\right), B^T g_f\left(x_k\right)\right)}{\left\| B^T g_f\left(x_k\right)\right\|}\right\| + h_k^2 \left\| \frac{B^T g_f\left(x_k\right)}{\left\| B^T g_f\left(x_k\right)\right\|}\right\|^2 = \left\| A\left(x_k - x^*\right)\right\|^2 -$$

$$-2h_k \left\| \frac{\left(x_k - x^*, g_f\left(x_k\right)\right)}{\left\| B^T g_f\left(x_k\right)\right\|}\right\| + h_k^2 = \left\| A\left(x_k - x^*\right)\right\|^2 - 2h_k \frac{\left(x_k - x^*, g_f\left(x_k\right)\right)}{\left\| B^T g_f\left(x_k\right)\right\|} + h_k^2 .$$

Зважаючи на те, що з (2.1) випливає нерівність

$$\frac{\left(x_k - x^*, g_f\left(x_k\right)\right)}{\left\| B^T g_f\left(x_k\right)\right\|} \ge \frac{f\left(x_k\right) - f^*}{\left\| B^T g_f\left(x_k\right)\right\|} = h_k,$$

маємо:

$$\left\| A\left(x_{k+1} - x^*\right)\right\|^2 = \left\| A\left(x_k - x^*\right)\right\|^2 - 2h_k \frac{\left(x_k - x^*, g_f\left(x_k\right)\right)}{\left\| B^T g_f\left(x_k\right)\right\|} + h_k^2 \le$$

$$\le \left\| A\left(x_k - x^*\right)\right\|^2 - 2h_k^2 + h_k^2 = \left\| A\left(x_k - x^*\right)\right\|^2 - \left(\frac{f\left(x_k\right) - f^*}{\left\| B^T g_f\left(x_k\right)\right\|}\right)^2 ,$$

що доводить нерівність (3.4). Теорему доведено.

Теорема 3.1 гарантує, що в субградієнтному методі з кроком Поляка в перетвореному просторі змінних відстань до точки мінімуму зменшується монотонно в перетвореному просторі. Крім того, задовольняються такі нерівності:

$$\left(A\left(x^* - x_{k+1}\right), -B^T g_f\left(x_k\right)\right) \ge 0, \quad k = 0,1,... \tag{3.5}$$

які можна переписати як нерівності

$$\left(y^* - y_{k+1}, -g_\varphi\left(y_k\right)\right) \ge 0, \quad k = 0,1,... \tag{3.6}$$

Для доведення істинності нерівностей (3.5) зазначимо, що

$$\left(A\left(x^* - x_{k+1}\right), -B^T g_f\left(x_k\right)\right) = \left(x^* - x_{k+1}, -g_f\left(x_k\right)\right).$$

Звідси, використовуючи (3.1) і (3.2), маємо:

$$\left(x^* - x_{k+1}, -g_f(x_k)\right) = \left(x_{k+1} - x^*, g_f(x_k)\right) = \left(x_k - x^* - h_k B \frac{B^T g_f(x_k)}{\left\|B^T g_f(x_k)\right\|}, g_f(x_k)\right) =$$

$$= \left(x_k - x^*, g_f(x_k)\right) - \left(h_k B \frac{B^T g_f(x_k)}{\left\|B^T g_f(x_k)\right\|}, g_f(x_k)\right) =$$

$$= \left(x_k - x^*, g_f(x_k)\right) - \frac{h_k}{\left\|B^T g_f(x_k)\right\|} \times \left(BB^T g_f(x_k), g_f(x_k)\right) =$$

$$= \left(x_k - x^*, g_f(x_k)\right) - h_k \frac{\left\|B^T g_f(x_k)\right\|^2}{\left\|B^T g_f(x_k)\right\|} = \left(x_k - x^*, g_f(x_k)\right) -$$

$$- \frac{f(x_k) - f^*}{\left\|B^T g_f(x_k)\right\|} \frac{\left\|B^T g_f(x_k)\right\|^2}{\left\|B^T g_f(x_k)\right\|} = \left(x_k - x^*, g_f(x_k)\right) - \left(f(x_k) - f^*\right) \geq 0.$$

Нерівності (3.6) означають, що для опуклої функції $\varphi(x)$, яка задовольняє умову (3.1), крок h_k визначає величину максимального зміщення в напрямку нормованого антисубградієнта. Цим гарантується, що кут між антисубградієнтом і напрямком від точки y_{k+1} до точки мінімуму y^* не буде тупим у перетвореному просторі змінних.

Для субградієнтного методу з кроком Поляка в перетвореному просторі справедливі дві теореми про швидкість збіжності методу для випадку довільних опуклих функцій та опуклих функцій з гострим мінімумом – аналогічні до теорем 2.2 та 2.3 для субградієнтного методу з кроком Поляка у початковому просторі. Відмінність полягає в наявності матриці перетворення простору B, яка породжує дещо інші формулювання теорем. Наведемо їх з доведенням.

Теорема 3.2. *Я*кщо опукла функція $\varphi(y)$ задовольняє нерівність (3.1), тоді справедлива рівність $\lim\limits_{k\to\infty} \sqrt{k}\left(\varphi(y_k) - \varphi^*\right) = 0$.

Доведення. З нерівності (3.4) теореми 3.1 випливає, що послідовність $h_k^2 = \frac{\left(f(x_k) - f^*\right)^2}{\left\|B^T g_f(x_k)\right\|^2} = \frac{\left(\varphi(y_k) - \varphi^*\right)^2}{\left\|g_\varphi(y_k)\right\|^2}$ збіжна. Зважаючи на те, що послідовність y_k збіжна ($\left\|y_k - y^*\right\| \le \left\|y_0 - y^*\right\|$), з леми 2.1 випливає нерівність $\left\|g_\varphi(y_k)\right\| \le C_2$, де C_2 - константа, що обмежує норму субрадієнта $g_\varphi(y)$. Звідси маємо, що послідовність $\left(\varphi(y_k) - \varphi^*\right)^2$ збіжна. Нехай $\varliminf_{k\to\infty} \sqrt{k}\left(\varphi(y_k) - \varphi^*\right) > 0$. Тоді $\varphi(y_k) - \varphi^* > a/\sqrt{k}$ при досить великих k та $a > 0$, а це суперечить збіжності послідовності $\left(\varphi(y_k) - \varphi^*\right)^2$. Звідси випливає, що $\varliminf_{k\to\infty} \sqrt{k}\left(\varphi(y_k) - \varphi^*\right) = 0$. Теорему 3.2 доведено.

Теорема 3.3. Нехай функція $\varphi(y) = f(By) = f(x)$ має гострий мінімум, тобто для неї виконується нерівність $\varphi(y) - \varphi^* \ge \alpha\left\|y - y^*\right\|$. Тоді метод В збігається зі швидкістю геометричної прогресії зі знаменником $q_2 = 1 - \frac{\alpha^2}{C_2}$, де C_2 - константа, що обмежує норму субградієнта $g_\varphi(y)$.

Доведення. Зважаючи на те, що $\varphi(y) = f(By) = f(x)$ та $g_\varphi(y_k) = B^T g_f(x_k)$, нерівність (3.4) можна переписати так:

$$\left\|y_{k+1} - y^*\right\|^2 \le \left\|y_k - y^*\right\|^2 - \frac{\left(\varphi(y_k) - \varphi^*\right)^2}{\left\|g_\varphi(y_k)\right\|^2}.$$

Звідси, з огляду на нерівність для $\varphi(y)$ в умові теореми, маємо:

$$\left\|y_{k+1} - y^*\right\|^2 \le \left\|y_k - y^*\right\|^2 - \frac{\left(\varphi(y_k) - \varphi^*\right)^2}{\left\|g_\varphi(y_k)\right\|^2} \le \left\|y_k - y^*\right\|^2 - \frac{\alpha^2\left\|y_k - y^*\right\|^2}{\left\|g_\varphi(y_k)\right\|^2}.$$

Враховуючи нерівність $\left\|g_\varphi(y_k)\right\| \le C_2$, маємо:

$$\left\|y_{k+1}-y^*\right\|^2 \le \left\|y_k-y^*\right\|^2 - \frac{\alpha^2\left\|y_k-y^*\right\|^2}{\left\|g_\varphi(y_k)\right\|^2} \le \left\|y_k-y^*\right\|^2 - \frac{\alpha^2}{C_2^2}\left\|y_k-y^*\right\|^2 = q_2\left\|y_k-y^*\right\|^2$$

де $q_2 = 1-\frac{\alpha^2}{C_2}$, що і означає збіжність методу зі швидкістю геометричної прогресії зі знаменником q_2. Теорему 3.3 доведено.

3.2 Метод В та зміщення по опуклості

В розділі 2 розглядався субградієнтний метод з кроком Поляка та параметром $m>1$ – максимальним зміщенням по опуклості функції, що мінімізується, який дозволяв врахувати деякі спеціальні класи функцій – кусочно-лінійні, квадратичні тощо. Цілком природнім кроком було б застосування параметра m в методі В, що забезпечує прискорення методу для додаткових класів опуклих функцій.

Отже, будемо мінімізувати такі опуклі функції $\varphi(y)=f(By)=f(x)$, для субградієнтів $g_\varphi(y)$ яких виконується узагальнена нерівність

$$\left(y-y^*, g_\varphi(y)\right) \ge m\left(\varphi(y)-\varphi^*\right),\ \forall y\in R^n, \qquad (3.7)$$

яка є аналогом нерівності (3.1). Оскільки значення параметра m є інваріантним відносно лінійного перетворення, що задається матрицею B, значення m для функцій $f(x)$ та $\varphi(y)$ співпадають. Субградієнтний метод з кроком Поляка у перетвореному просторі, визначений невиродженою матрицею B, та параметром $m>1$, має такий вигляд:

$$x_{k+1} = x_k - h_k B\frac{B^T g_f(x_k)}{\left\|B^T g_f(x_k)\right\|},\quad h_k = \frac{m\left(f(x_k)-f^*\right)}{\left\|B^T g_f(x_k)\right\|},\quad k=0,1,2.... \qquad (3.8)$$

Тут h_k є кроком Поляка (Агмона – Моцкіна – Шонберга) в перетвореному просторі змінних $y=Ax$, в якому метод (3.8) можна записати у вигляді

$$y_{k+1} = y_k - h_k \frac{g_\varphi(y_k)}{\left\| g_\varphi(y_k) \right\|}, \quad h_k = \frac{m\left(\varphi(y_k) - \varphi^*\right)}{\left\| g_\varphi(y_k) \right\|}, \quad k = 0,1,2.... \tag{3.9}$$

Теорема 3.4. Послідовність $\{x_k\}_{k=0}^{k^*-1}$, породжена методом (3.8), задовольняє нерівності

$$\left\| A\left(x_{k+1} - x^*\right) \right\|^2 \le \left\| A\left(x_k - x^*\right) \right\|^2 - \frac{m^2\left(f(x_k) - f^*\right)^2}{\left\| B^T g_f(x_k) \right\|^2}, \quad k = 0,1,2,... \tag{3.10}$$

Доведення. Переписавши пункт В2 ітеративного процесу у вигляді

$$Ax_{k+1} = Ax_k - h_k \frac{B^T g_f(x_k)}{\left\| B^T g_f(x_k) \right\|},$$

для довільного k $(0 \le k \le k^* - 1)$ маємо:

$$\left\| A\left(x_{k+1} - x^*\right) \right\|^2 = \left\| A\left(x_k - x^*\right) - h_k \frac{B^T g_f(x_k)}{\left\| B^T g_f(x_k) \right\|} \right\|^2 = \left\| A\left(x_k - x^*\right) \right\|^2 -$$

$$-2h_k \left\| \frac{\left(A\left(x_k - x^*\right), B^T g_f(x_k)\right)}{\left\| B^T g_f(x_k) \right\|} \right\| + h_k^2 \left\| \frac{B^T g_f(x_k)}{\left\| B^T g_f(x_k) \right\|} \right\|^2 = \left\| A\left(x_k - x^*\right) \right\|^2 -$$

$$-2h_k \left\| \frac{\left(x_k - x^*, g_f(x_k)\right)}{\left\| B^T g_f(x_k) \right\|} \right\| + h_k^2 = \left\| A\left(x_k - x^*\right) \right\|^2 - 2h_k \frac{\left(x_k - x^*, g_f(x_k)\right)}{\left\| B^T g_f(x_k) \right\|} + h_k^2.$$

Зважаючи на те, що з (2.6) випливає нерівність

$$\frac{\left(x_k - x^*, g_f(x_k)\right)}{\left\| B^T g_f(x_k) \right\|} \ge \frac{m\left(f(x_k) - f^*\right)}{\left\| B^T g_f(x_k) \right\|} = h_k,$$

маємо:

$$\left\| A\left(x_{k+1} - x^*\right) \right\|^2 = \left\| A\left(x_k - x^*\right) \right\|^2 - 2h_k \frac{\left(x_k - x^*, g_f(x_k)\right)}{\left\| B^T g_f(x_k) \right\|} + h_k^2 \le$$

$$\le \left\| A\left(x_k - x^*\right) \right\|^2 - 2h_k^2 + h_k^2 = \left\| A\left(x_k - x^*\right) \right\|^2 - \left(\frac{m\left(f(x_k) - f^*\right)}{\left\| B^T g_f(x_k) \right\|} \right)^2,$$

що доводить нерівність (3.10). Теорему доведено.

Теореми 3.5 та 3.6 дають обґрунтування оцінкам швидкості збіжності субградієнтного методу з кроком Поляка у перетвореному просторі змінних та параметром $m>1$ для довільної опуклої функції та опуклої функції з гострим мінімумом.

Теорема 3.5. Якщо опукла функція $\varphi(y)$ задовольняє нерівність (3.7), тоді справедлива рівність $\lim\limits_{k\to\infty}\sqrt{k}\left(\varphi(y_k)-\varphi^*\right)=0$.

Доведення. З нерівності (3.10) теореми 3.4 випливає, що послідовність $h_k^2=\dfrac{m^2\left(f(x_k)-f^*\right)^2}{\left\|B^T g_f(x_k)\right\|^2}=\dfrac{m^2\left(\varphi(y_k)-\varphi^*\right)^2}{\left\|g_\varphi(y_k)\right\|^2}$ збіжна. Зважаючи на те, що послідовність y_k збіжна ($\left\|y_k-y^*\right\|\le\left\|y_0-y^*\right\|$), з леми 2.1 випливає нерівність $\left\|g_\varphi(y_k)\right\|\le C_3$, де C_3 - константа, що обмежує норму субградієнта $g_\varphi(y)$. Звідси маємо, що послідовність $\left(\varphi(y_k)-\varphi^*\right)^2$ збіжна. Нехай $\lim\limits_{k\to\infty}\sqrt{k}\left(\varphi(y_k)-\varphi^*\right)>0$. Тоді $\varphi(y_k)-\varphi^*>a/\sqrt{k}$ при досить великих k та $a>0$, а це суперечить збіжності послідовності $\left(\varphi(y_k)-\varphi^*\right)^2$. Звідси випливає, що $\lim\limits_{k\to\infty}\sqrt{k}\left(\varphi(y_k)-\varphi^*\right)=0$. Теорему 3.5 доведено.

Теорема 3.6. Нехай функція $\varphi(y)=f(By)=f(x)$ має гострий мінімум, тобто для неї виконується нерівність $\varphi(y)-\varphi^*\ge\alpha\left\|y-y^*\right\|$. Тоді метод (3.9) збігається зі швидкістю геометричної прогресії зі знаменником $q_3=1-\left(\dfrac{m\alpha}{C_3}\right)^2$, де C_3 - константа, що обмежує норму субградієнта $g_\varphi(y)$.

Доведення. Зважаючи на те, що $\varphi(y)=f(By)=f(x)$ та $g_\varphi(y_k)=B^T g_f(x_k)$, нерівність (3.10) можна переписати так:

$$\left\|y_{k+1}-y^*\right\|^2 \le \left\|y_k-y^*\right\|^2 - \frac{m^2\left(\varphi(y_k)-\varphi^*\right)^2}{\left\|g_\varphi(y_k)\right\|^2}.$$

Звідси, з огляду на нерівність для $\varphi(y)$ в умові теореми, маємо:

$$\left\|y_{k+1}-y^*\right\|^2 \le \left\|y_k-y^*\right\|^2 - \frac{m^2\left(\varphi(y_k)-\varphi^*\right)^2}{\left\|g_\varphi(y_k)\right\|^2} \le \left\|y_k-y^*\right\|^2 - \frac{m^2\alpha^2\left\|y_k-y^*\right\|^2}{\left\|g_\varphi(y_k)\right\|^2}.$$

Враховуючи нерівність $\left\|g_\varphi(y_k)\right\| \le C_3$, маємо:

$$\left\|y_{k+1}-y^*\right\|^2 \le \left\|y_k-y^*\right\|^2 - \frac{m^2\alpha^2\left\|y_k-y^*\right\|^2}{\left\|g_\varphi(y_k)\right\|^2} \le \left\|y_k-y^*\right\|^2 - m^2\frac{\alpha^2}{C_3^2}\left\|y_k-y^*\right\|^2 = q_3\left\|y_k-y^*\right\|^2$$

де $q_3 = 1-\left(\frac{m\alpha}{C_3}\right)^2$, що і означає збіжність методу зі швидкістю геометричної прогресії зі знаменником q_3. Теорему 3.6 доведено.

3.3 Octave-функція PolyakB

В цьому підрозділі опишемо та наведемо Octave-код програми *PolyakB*, яка є реалізацією методу В для $m \ge 1$. Будемо знаходити наближення x_ε^* точки мінімуму опуклої функції $f(x)$ від n змінних.

Вхідними параметрами програми *PolyakB* є: **calcfg** – посилання на функцію для обчислення $f(x)$ та $g_f(x)$; **x0** – початкова точка; **fstar** − значення функції в точці мінімуму; **m** ($m \ge 1$) – параметр зміщення по опуклості; **B** – матриця перетворення простору розмірності $n \times n$; точність **epsf** та **maxitn** – параметри зупинки.

Вихідними параметрами програми є: знайдена точка мінімуму **x**, значення функції в знайденій точці мінімуму **f**, кількість ітерацій **itn** та код повернення **info**, значенням якого визначається статус точки **x** на ітерації **itn**=k. Значення **info**=0 означає, що знайдено точку x_k, для якої $f(x_k) - f^* \le \varepsilon_f$; **info**=4 означає,

що **itn** > **maxitn** (ітераційний процес перевищив максимальну кількість ітерацій).

Octave-код функції *PolyakB* подано нижче.

```
function [x,f,itn,info] = PolyakB(calcfg,B,x0,fstar,m,        #row01
                                        epsf,maxitn,itnp);  #.....
   itn=0; x=x0; [f,g] = calcfg(x);                          #row02
   if(itnp>0)                                               #row03
      printf("itn %4d f %14.6e \n", itn, f); # xprint = x',  #.....
   endif                                                    #.....
   for(itn = 1:maxitn)                                      #row04
      if(f-fstar < epsf) info = 0; return; endif            #row05
      g1=B'*g; dg1=norm(g1);                                #row06
      g2=g1/dg1; hs=m*(f-fstar)/dg1;                        #row07
      x -= hs * B * g2;                                     #row08
      [f,g] = calcfg(x); dg=norm(g);                        #row09
      if(mod(itn,itnp)==0)                                  #row10
         printf("itn %4d f %14.6e \n",itn,f); # xprint = x', #.....
      endif                                                 #.....
   endfor                                                   #row11
   info = 4;                                                #row12
endfunction                                                 #row13
```

3.4 Обчислювальні експерименти

Субградієнтний метод з кроком Поляка у перетвореному просторі для різних значень параметра m можна успішно застосовувати для мінімізації так званих яружних функцій. Термін «яружна функція» не є формально визначеним. Він означає, що лінії рівня такої функції наближено мають форму еліпсів з осями, відношення яких є доволі великим. Існує область, при наближенні до якої це відношення зростає – вона називається *яром*. Для таких функцій траєкторія градієнтного методу є зигзагоподібною з малим кроком, внаслідок чого швидкість спуску до точки мінімуму сильно вповільнюється. Це спричиняється значним відхиленням напрямку антиградієнта від напрямку до точки мінімуму, внаслідок чого метод потребує великої кількості ітерацій для

знаходження наближення з заданою точністю. Перетворення простору в субградієнтному методі з кроком Поляка здійснюється так, щоб зменшити витягнутість поверхонь рівня функції, що мінімізується. Саме тому описаний метод є ефективним підходом до мінімізації яружних функцій.

Використовуючи класичний субградієнтний метод з кроком Поляка, можна знайти досить точні наближення до точки мінімуму опуклої функції, поверхні рівня якої не є сильно або суттєво витягнутими. Він збігається до точки мінімуму опуклої функції швидко, якщо кути між послідовними субградієнтами на кожній ітерації близькі до прямого кута (гострими або тупими). Однак недоліком класичного субградієнтного методу з кроком Поляка є його повільна збіжність для яружних функцій. Перетворення простору дає можливість суттєво прискорити його, підібравши матрицю B так, щоб у перетвореному просторі змінних поверхні рівня яружних функцій були менш витягнутими, ніж у вихідному просторі змінних.

Нагадаємо, що класичний субградієнтний метод з кроком Поляка ми називаємо методом А, а його модифікацію з перетворенням простору – методом В, вказуючи в дужках яке саме значення параметра m використовується.

Далі розглянемо та проаналізуємо результати обчислювальних експериментів по застосуванню субградієнтного методу з кроком Поляка в перетвореному просторі при $m \geq 1$ для мінімізації яружних гладких та негладких опуклих функцій, а також порівняємо їх з результатами, одержаними за допомогою класичного субградієнтного методу з кроком Поляка при $m \geq 1$.

Приклад 3.1 (кусочно-лінійна функція $f_1(x_1, x_2) = |x_1| + t|x_2|, \ t > 1$). Для неї $x^* = (0,0)^T$ та $f^* = 0$, а значення параметра $m = 1$. Для обчислення значень $f(x)$ та $g_f(x)$ використаємо таку Octave-функцію:

```
function [f,g] = sabs(x)
global t;
f = abs(x(1,1)) + t*abs(x(2,1));
g(1,1) = sign(x(1,1));
g(2,1) = t*sign(x(2,1));
```

```
endfunction
```

Швидкість збіжності методу А для функції $f_1(x_1, x_2)$ визначається геометричною прогресією зі знаменником

$$q = \sqrt{1 - 1/t^2} \tag{3.11}$$

і буде дуже повільною для великих значень t. Наприклад, для знаходження точки мінімуму з точністю $\varepsilon_f = 10^{-6}$ у випадку злегка яружної функції $f_1(x_1, x_2)$ при значенні $t = 5$ методу А знадобилось 180 ітерацій. Однак, збільшивши ступінь «яружності» цієї ж функції в 5 разів, поклавши $t = 25$, отримуємо вже 4523 ітерацій, а для $t = 50$ маємо аж 18 113 ітерацій, необхідних для досягнення тієї ж точності. Зигзагоподібна траєкторія субградієнтного методу А ($m = 1$) при $t = 5$ показана на рис. 3.1.

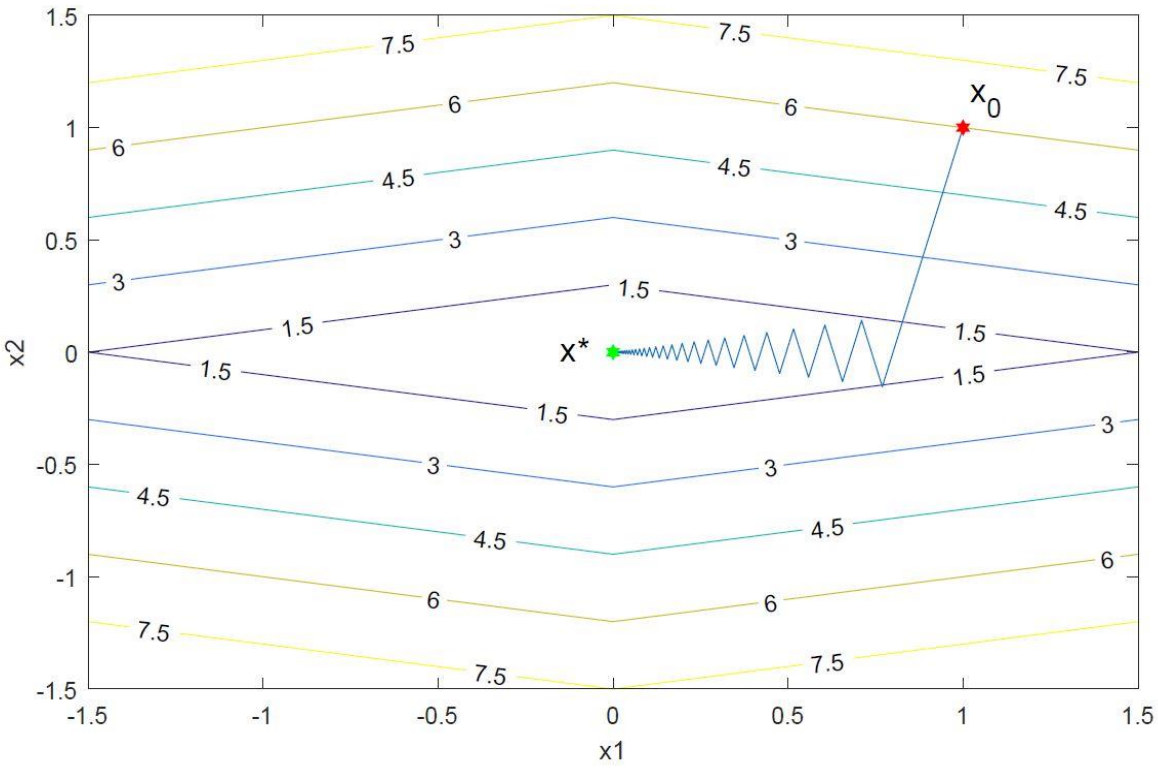


Рис. 3.1. Траєкторія методу А ($m = 1$) для пошуку наближення x_ε^* функції

$$f_1(x_1, x_2) = |x_1| + 5|x_2|: \; x^* = (0,0)^T, \; f^* = 0, \; \varepsilon = 10^{-6}.$$

Якщо ж виконати перетворення простору за допомогою матриці $B=\begin{pmatrix}1 & 0\\ 0 & 1/\alpha\end{pmatrix}$, для якої, наприклад, значення $\alpha=2$, швидкість збіжності субградієнтного методу з кроком Поляка визначатиметься геометричною прогресією зі знаменником $q=\sqrt{1-4/t^2}$, який менший ніж $q=\sqrt{1-1/t^2}$ у методі без перетворення (див. (3.4.1)). При точності $\varepsilon=10^{-6}$ метод В ($m=1$) потребує лише 42 ітерації – його траєкторія при $t=5$ та $\alpha=2$ зображена на рис. 3.2.

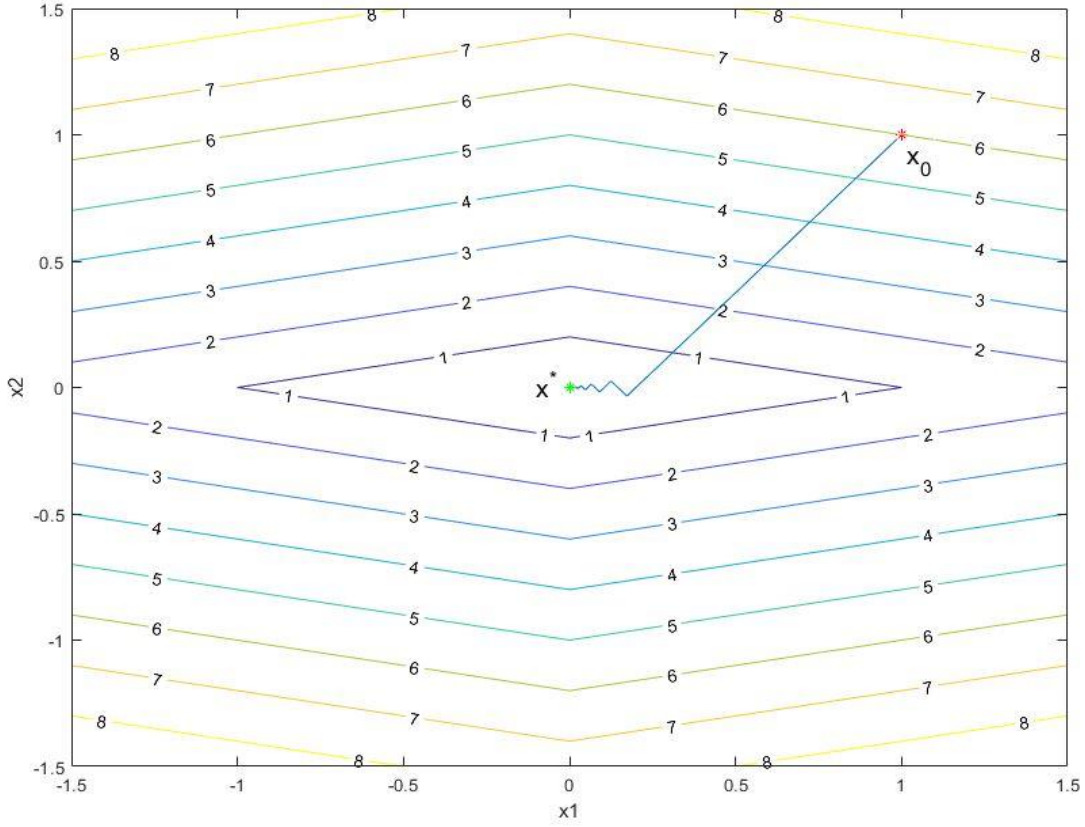


Рис. 3.2. Траєкторія методу В ($m=1$) для пошуку наближення до точки мінімуму функції $f_1(x_1,x_2)=|x_1|+5|x_2|$: $\alpha=2$, $x^*=(0,0)^T$, $f^*=0$, $\varepsilon=10^{-6}$.

Рисунки 3.1 і 3.2 демонструють, що субградієнтний метод з кроком Поляка в перетвореному просторі не лише потребує меншої кількості ітерацій, а і швидше знаходить точніше наближення. Уже після першої ітерації методу В ми

отримуємо розв'язок x_1, який лежить значно ближче до точки мінімуму, ніж той же розв'язок x_1 методу Поляка без перетворення. Збільшимо тепер ступінь яружності функції $f_1(x_1, x_2)$ в 2 рази, поклавши $t = 10$, та проаналізуємо результати роботи методу для точностей $10^{-1} - 10^{-10}$ та шести різних матриць B, яким відповідають колонки 2-7 таблиці 3.1. Матриці B одержано як результат розтягу простору змінних у напрямку x_2 з коефіцієнтами розтягу $\alpha = 1, 1.5, 2, 3, 4, 5$. Зрозуміло, що при $\alpha = 1$ матриця B збігається з одиничною, отже перетворення простору не відбувається і метод В збігається з субградієнтним методом з кроком Поляка без перетворення.

Таблиця 3.1

Кількість ітерацій методу В ($m = 1$) для мінімізації функції

$$f_1(x_1, x_2) = |x_1| + 10|x_2|,\ x_0 = (1,1)^T,\ x^* = (0,0)^T$$

ε_f	$\alpha = 1$	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
10^{-1}	147	63	33	6	10	9
10^{-2}	262	114	62	19	17	13
10^{-3}	377	165	91	31	24	18
10^{-4}	492	216	119	44	31	22
10^{-5}	607	268	148	57	38	27
10^{-6}	722	319	177	70	45	31
10^{-7}	837	370	206	82	53	36
10^{-8}	952	421	234	95	60	40
10^{-9}	1068	472	263	108	67	45
10^{-10}	1183	523	292	121	74	49

З таблиці 3.1 бачимо, що кількість ітерацій субградієнтного методу з кроком Поляка в перетвореному просторі зменшується монотонно, коли зменшується ступінь яружності функції $\varphi_1(y_1, y_2) = |y_1| + \frac{10}{\alpha}|y_2|$ в перетвореному просторі змінних, що відповідає збільшенню коефіцієнта розтягу α. Також значення стовпців 3-7 чудово демонструють, що метод з перетворенням простору працює значно ефективніше, ніж без нього: наприклад, для досягнення точності $\varepsilon_f = 10^{-10}$ при $\alpha = 5$ методу знадобилось лише 49 ітерацій, тоді як при $\alpha = 1$ (класичний метод Поляка) – 1183 ітерації, що у 24 рази більше.

Приклад 3.2 (квадратична функція $f_2(x_1, x_2) = x_1^2 + 6x_2^2$). У розділі 2 ми вже проводили експерименти над цією функцією з методом А для різних значень параметра m і t, який задає ступінь яружності функції. Для узагальнення результатів наведемо кількості ітерацій обох методів А та В при $m = 1, 2$ і $t = 100, 1\,000, 10\,000$ для різних значень точностей – вони зібрані в таблиці 3.2. Як матрицю перетворення простору використаємо

$$B = \begin{pmatrix} 1 & 0 \\ 0 & 1/\alpha \end{pmatrix} \text{ при } \alpha = 10.$$

Таблиця 3.2 складається з чотирьох блоків, кожен з яких відповідає за ітерації певного варіанту методу: без перетворення простору при $m = 1$ (колонки 2-4) та $m = 2$ (колонки 5-7), а також з перетворенням простору при $m = 1$ (колонки 8-10) та $m = 2$ (колонки 11-13). Кожен блок містить по 3 колонки, в яких наведені кількості ітерацій при $t = 100, 1\,000, 10\,000$ відповідно.

Таблиця 3.2

Кількості ітерацій методів A та B при $m=1,2$ і $t=100,1\,000,10\,000$ для мінімізації функції $f_2(x_1,x_2)=x_1^2+tx_2^2$

ε_f	Метод A						Метод B					
	$m=1$			$m=2$			$m=1$			$m=2$		
10^{-1}	9	119	255	6	6	6	6	10	23	2	3	4
10^{-5}	81	311	1167	20	20	20	13	23	71	2	5	6
10^{-10}	173	564	1943	36	36	36	21	42	162	2	8	8
10^{-15}	266	878	2406	52	52	52	30	63	232	2	10	10
10^{-20}	341	1240	3635	68	70	70	38	82	304	2	12	12

Аналізуючи таблицю 3.2, можна зробити декілька висновків. По-перше, зі зростанням параметра t, як правило, збільшується також кількість ітерацій методів, однак при $m=2$ цей ефект не спостерігається. В другому та четвертому блоках ітерації майже скрізь залишаються сталими; винятки з'являються лише при точності 10^{-20}. По-друге, зі зростанням точності кількість ітерацій завжди збільшується, однак і тут з'являється виняток: метод B при $m=2$ і $t=100$ потребує лише 2 ітерацій незалежності від точності. По-третє, застосування параметра $m=2$ та перетворення простору значно прискорює збіжність методу. Наприклад, при точності 10^{-10} метод A ($m=1$) потребує 1943 ітерації; при $m=2$ це вже 36 ітерацій, а, застосувавши метод B ($m=1$), маємо 162 ітерації. Варто зазначити, що перетворення простору працює краще, ніж застосування $m=2$, лише при досить великих значеннях t, наприклад, 304 ітерації методу з перетворенням та 70 з $m=2$ для точності 10^{-20}. Нарешті, по-четверте, найкращі результати демонструє метод B при $m=2$, тобто при комбінуванні обох модифікацій субградієнтного методу з кроком Поляка. Наприклад, він демонструє стабільні 2 ітерації, тоді як метод A ($m=1$) – від 9 до 341 ітерації для різних точностей. У найгіршому випадку (параметр

$t = 10000$ при точності 10^{-20}) метод B ($m = 2$) потребує всього 12 ітерацій проти 3635 в методу A ($m = 1$), що в 300 разів більше.

Приклад 3.3 (функція $f_3(x_1, x_2) = \max\left\{x_1^2 + (2x_2 - 2)^2 - 3,\ x_1^2 + (x_2 + 1)^2\right\}$). Ситуація зі швидкістю збіжності буде складнішою для суттєво-яружної кусочно-квадратичної функції $f_3(x_1, x_2)$, для якої $x^* = (0,0)$ та $f^* = 1$, а значення параметра $m = 1$. Розглянемо кількості ітерацій методу B для різних значень параметра α в матриці перетворення простору $B = \begin{pmatrix} 1 & 0 \\ 0 & 1/\alpha \end{pmatrix}$, які наведені в таблиці 3.3.

Таблиця 3.3

Кількості ітерацій методу B ($m = 1$) для мінімізації функції $f_3(x_1, x_2)$: $x_0 = (1,1)^T$, $x^* = (0,0)$, $f^* = 1$, maxitn = 100 000

ε_f	$\alpha = 1$	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
10^{-1}	16	4	4	5	7	8
10^{-2}	162	37	4	6	7	9
10^{-3}	1604	679	5	6	8	9
10^{-4}	16004	7079	5	6	9	46
10^{-5}	–	71079	6	8	8061	6206
10^{-6}	–	–	6	–	98061	63806
10^{-7}	–	–	6	–	–	–

Як бачимо, зі збільшенням величини α, коефіцієнту розтягу простору в напрямку x_2, монотонного зменшення кількості ітерацій не спостерігається. Найменшого числа ітерацій метод B ($m = 1$) потребує при $\alpha = 2$, а саме: 4

ітерації для досягнення точності $10^{-1}-10^{-2}$, 5 ітерацій для точності $10^{-3}-10^{-4}$ та 6 ітерацій для точності 10^{-5} і вище. Схожий результат спостерігається при $\alpha=3$, однак, починаючи з точності 10^{-6} і вище, кількість ітерацій стрімко збільшується та перевищує *maxitn* (в таблиці позначено прочерком). Для інших значень α метод B демонструє інтенсивне зростання кількості ітерацій при підвищенні точності.

Для наглядної демонстрації ефективності роботи методу B для функції $f_3(x_1,x_2)=\max\left\{x_1^2+(2x_2-2)^2-3,\ x_1^2+(x_2+1)^2\right\}$ розташуємо траєкторії методу при $\alpha=1$ (метод A, без перетворення простору) та $\alpha=2$ (метод B, з перетворенням, $m=1$) на одному рисунку (див. рис. 3.3). Траєкторія методу A зображена суцільною синьою лінією, траєкторія методу B – пунктирною червоною. Також червоним кольором виділено чотири початкові точки, згенеровані методами, причому методу B знадобилось усього 4 ітерації для досягнення точності $\varepsilon_f=10^{-3}$, тоді як методу A – 1604.

Приклад 3.4 (функція $f_4(x)=x^THx+b^Tx+c$). У розділі 2 вже проводились обчислювальні експерименти з функцією $f_4(x)$, в якій матриця H мала розмірність 1000×1000 (приклад 2.5). Результати прикладу 2.5 показали, що використання параметра $m=2$ дозволяє суттєво прискорити роботу класичного субградієнтного методу з кроком Поляка. Розглянемо тепер результати роботи методу B, який використовує перетворення простору, для цієї функції при $m=1,2$ та порівняємо їх з результатами роботи методу A ($m=1,2$). Елементи симетричної матриці H розмірності 1000×1000 генеруються з використанням генератора псевдовипадкових чисел з певного інтервалу, який задається параметром α, що дозволяє регулювати величини $\lambda_{\max}$ та $\lambda_{\min}$ цієї матриці. Елементи вектора $b\in\mathbb{R}^{1000}$ обираються так, щоб точка мінімуму $x^*=(1,...,1)^T\in\mathbb{R}^{1000}$ та $f^*=0$.

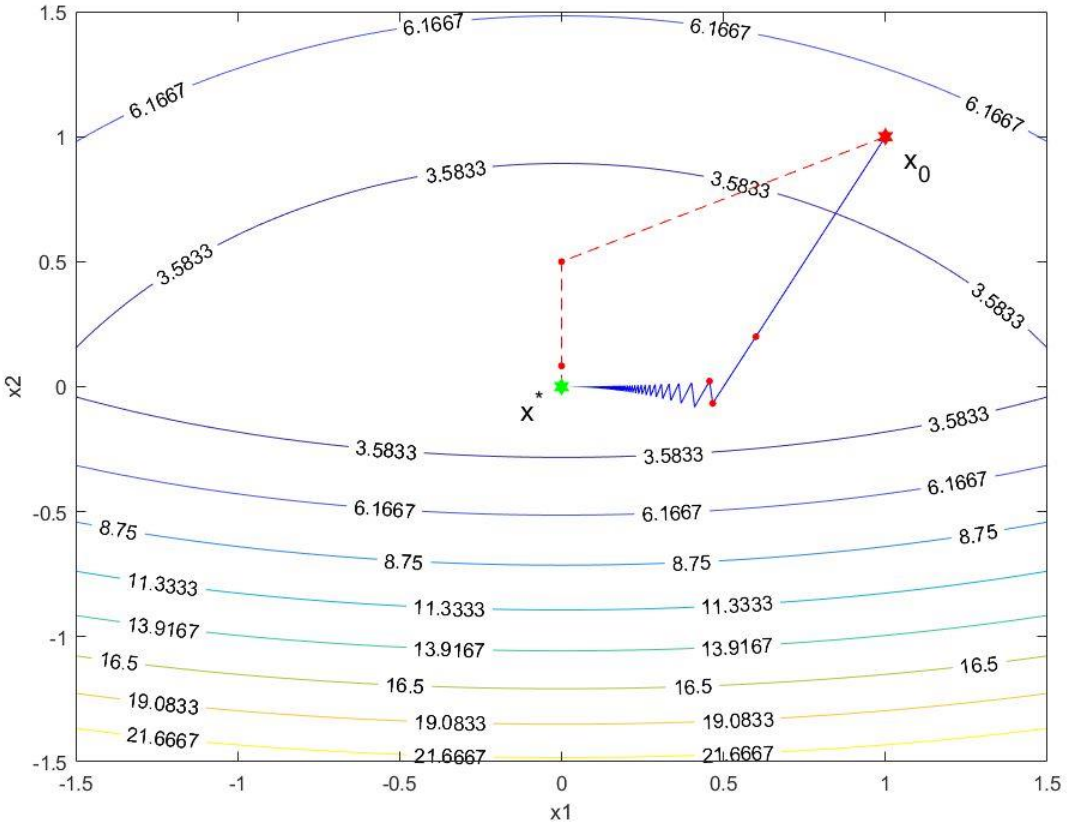


Рис. 3.3. Траєкторія методу А (суцільна лінія) та методу В ($m=1$) (штрихова лінія) для знаходження наближення до точки мінімуму кусочно-квадратичної функції $f_3(x_1,x_2)$: $x_0=(1,1)^T$, $x^*=(0,0)$, $f^*=1$, $B=diag(1,0.5)$.

Перетворення простору в методі В здійснюється в напрямку власного вектора матриці H, що відповідає її максимальному власному числу $\lambda_{\max}$. Будемо знаходити мінімальне значення функції з точністю $\varepsilon_f=10^{-10}$. Результати експериментів наведені в таблиці 3.4.

Таблиця 3.4

Кількості ітерацій методів А та В для мінімізації функції $f_4(x)$ при $m=1,2$ для різних $\lambda=\lambda_{\max}/\lambda_{\min}$, $x^*=(1,...,1)^T$

№	λ	Метод А		Метод В	
		$m=1$	$m=2$	$m=1$	$m=2$
1	1.7865e+06	18	2	18	2

Продовження табл. 3.4

2	3.6610e+04	19	2	19	2
3	5.8537e+03	–	–	21	3
4	4.3965e+03	–	–	23	4
5	4.1623e+03	–	–	25	5

Стовпці 3-6 таблиці 3.4 містять кількості ітерацій методів A та B з використанням параметра $m = 1,2$, при чому прочерк означає, що число ітерацій є більшим ніж 100 і мінімальне значення функції з точністю $\varepsilon_f = 10^{-10}$ не знайдено. Натомість, після виконання 100 ітерацій знайдені значення функції дорівнюють 6.0835e-08, 4.5431e-07 та 4.6922e-06 (використовується $m = 1$) та 1.2108e-09, 4.1774e-08 та 7.6568e-07 (використовується $m = 2$) для випадків 3-5 відповідно. Точність цих розв'язків є досить високою, враховуючи таку незначну кількість ітерацій. Варто також зауважити, що точність розв'язків методу A при $m = 2$ є вищою, ніж при $m = 1$.

Стовпці 5,6 таблиці 3.4 демонструють суттєво менші кількості ітерацій методу B порівняно з методом A, при чому в першому випадку розв'язок знайдено з заданою точністю $\varepsilon_f = 10^{-10}$. Спостерігається незначне зростання кількості ітерацій методу B ($m = 1$) по мірі зменшення величини $\lambda = \lambda_{\max} / \lambda_{\min}$: від 18 до 25 ітерацій. Якщо ж, окрім перетворення простору в напрямку власного вектора, що відповідає максимальному власному числу, використати значення $m = 2$, кількість ітерацій вдається суттєво скоротити: вона коливається в межах від 2 до 5 ітерацій для випадків 1-5. Такі результати дозволяють зробити висновок про доцільність використання модифікацій класичного субградієнтного методу з кроком Поляка на основі перетворення простору та використання параметра m навіть в умовах значного відношення $\lambda = \lambda_{\max} / \lambda_{\min}$ порядку $10^3 - 10^6$.

Розглянемо результати роботи методів A та B для цього ж тесту, в якому точка мінімуму x^* будується з використанням генератора псевдовипадкових

чисел на інтервалі $[0,10]$. Кількості ітерацій методів А та В при $m=1,2$ для різних $\lambda=\lambda_{\max}/\lambda_{\min}$ наведені в таблиці 3.5.

Таблиця 3.5

Кількості ітерацій методів А та В для мінімізації функції $f_4(x)$ при $m=1,2$ для різних $\lambda=\lambda_{\max}/\lambda_{\min}$, $x^*=rand([0,10])$

№	λ	Метод А		Метод В	
		$m=1$	$m=2$	$m=1$	$m=2$
1	1.7865e+06	7399	14	20	5
2	3.6504e+04	1892	18	21	6
3	5.8373e+03	932	22	23	8
4	4.4003e+03	879	26	25	9
5	4.1499e+03	1449	28	27	10

Результати таблиці 3.5 показують, що для отримання розв'язку з заданою точністю найбільшої кількості ітерацій потребує метод А ($m=1$): вона варіюється в межах від 879 до 7399 ітерацій для випадів 1-5. Метод А ($m=2$) та метод В ($m=1$) потребують суттєво меншої та майже однакової кількості ітерацій: наприклад, 28 ітерацій методу А ($m=2$) та 27 ітерацій методу В ($m=1$) для випадку 5. Використання ж обох модифікацій – перетворення простору та параметра m – дозволяє ще більше скоротити кількість ітерацій: для випадку 1 метод В ($m=2$) потребує 5 ітерацій проти 7399 ітерацій методу А ($m=1$), що більше ніж у 1400 разів менше. Octave-код програми запуску прикладу 3.4 та її лістинг наведено в додатку В.

Результати цього тесту для двох різних способів генерування вектора x^*, а також прикладів 2.4 та 2.5 демонструють, що використання запропонованих модифікацій класичного субградієнтного методу з кроком Поляка дозволяють знаходити наближення до точки мінімуму опуклих функцій, що залежать від

великої кількості вхідних даних, не лише за малу кількість ітерацій (порівняно з класичним методом), а ще й з досить великою точністю.

3.5 Висновки до третього розділу

1. Розглянуто субградієнтний метод з кроком Поляка у перетвореному просторі змінних для мінімізації яружних опуклих функцій при відомому оптимальному значенні. Доведено теореми про монотонне зменшення відстані до точки мінімуму та швидкість збіжності методу, яка дорівнює $O\left(1/\sqrt{k}\right)$ у випадку довільної опуклої функції та швидкості геометричної прогресії для опуклої функції з гострим мінімумом.

2. Розглянуто модифікацію субградієнтного методу з кроком Поляка у перетвореному просторі змінних скалярним параметром $m>1$, який задає величину максимального зміщення по опуклості функції $f\left(x\right)$ і дає змогу врахувати спеціальні класи опуклих функцій. Доведено теореми про монотонне зменшення відстані до точки мінімуму та швидкість збіжності модифікованого методу, яка дорівнює $O\left(1/\sqrt{k}\right)$ у випадку довільної опуклої функції та швидкості геометричної прогресії для опуклої функції з гострим мінімумом.

3. Розглянуто чотири тестові приклади з мінімізації гладких та негладких яружних опуклих функцій з використанням описаних методів. Результати обчислювальних експериментів з цими прикладами показали, що підібрана певним чином матриця перетворення простору та використання скалярного параметра $m>1$ дозволяють значно прискорити класичний субградієнтний метод з кроком Поляка при мінімізації функцій з суттєво та сильно витягнутими поверхнями рівня.

РОЗДІЛ 4. СУБГРАДІЄНТНІ МЕТОДИ З ПЕРЕТВОРЕННЯМ ПРОСТОРУ ДЛЯ РОЗВ'ЯЗАННЯ СИСТЕМ ЛІНІЙНИХ РІВНЯНЬ

У четвертому розділі розглядаються три субградієнтні методи з перетворенням простору та їхнє застосування для знаходження L_p-розв'язків систем лінійних рівнянь різного типу. В підрозділі 4.2 наведено опис та загальна схема методу еліпсоїдів, а підрозділи 4.3 та 4.4 присвячені його застосуванню для задач визначення параметрів лінійної регресії та знаходження L_p-розв'язку системи лінійних рівнянь з двосторонніми обмеженнями на змінні відповідно. В підрозділі 4.5 розглядаються субградієнтний метод з кроком Поляка у вихідному та перетвореному просторах змінних та скалярним параметром $m>1$ (розділ 2 і 3) і метод *amsg2p* для знаходження розв'язку сумісних систем лінійних рівнянь. Результати обчислювальних експериментів, наведені в підрозділах 4.3 та 4.5, демонструють ефективність роботи описаних методів та алгоритмів.

4.1 Задача розв'язання системи лінійних рівнянь

Розглянемо систему лінійних алгебраїчних рівнянь (СЛАР)

$$Ax \approx b, \tag{4.1}$$

яку можна переписати у векторній формі

$$\sum_{j=1}^{n} a_{ij} x_j \approx b_i,\ i=\overline{1,m}. \tag{4.2}$$

Тут A – дійсна $m \times n$ матриця, $x \in \mathbb{R}^n$ – вектор змінних, $b \in \mathbb{R}^m$ – вектор правих частин. Необхідно знайти вектор $x_p^* \in \mathbb{R}^n$, який мінімізує L_p-норму вектора $y = Ax - b = \left(y_1, ..., y_m\right)^T \in \mathbb{R}^m$, який є нев'язкою системи (4.1).

Якщо система лінійних рівнянь (4.1) має хоча б один розв'язок її називають *сумісною*. При $m > n$ система є *перевизначеною*. Якщо для неї ранг

матриці A рівний n, то задача система має єдиний розв'язок $x = A^{-1}b$, де A^{-1} – обернена до A матриця. Якщо наявні обмеження $l \le x \le u$, де $l,u \in \mathbb{R}^n$ – n-вимірні вектори, для яких $u_i > l_i$ для всіх $i = \overline{1,n}$, система (4.1) є системою з *двосторонніми обмеженнями на змінні*.

Такі обмеження накладають певні умови на підхід, який використовується для знаходження розв'язку системи (4.1). Однак, запис цієї задачі в термінах задачі опуклого програмування дає можливість використовувати методи мінімізації негладких опуклих функцій. А саме: задача знаходження «найкращої» L_p-норми нев'язки системи (4.1) відповідає задачі мінімізації опуклої функції

$$f_p(x) = \|Ax - b\|_p = \left(\sum_{i=1}^{m} |y_i|^p \right)^{1/p}, \tag{4.3}$$

тобто знаходження

$$x_p^* = \arg\min_{x \in \mathbb{R}^n} f_p(x), \tag{4.4}$$

де $y_i = \sum_{j=1}^{n} a_{ij} x_j - b_i$, $i = \overline{1,m}$, а $p \in \mathbb{R}$ – скалярний параметр. Умова $p \ge 1$ забезпечує опуклість негладкої функції $f_p(x)$. В загальному випадку розв'язок задачі (4.3), (4.4) не обов'язково єдиний. В такому випадку точка x_p^* – одна з точок множини оптимальних розв'язків, якій відповідає мінімальне значення функції f_p^* в задачі (4.3), (4.4).

Найбільш поширеними є три частинні випадки «найкращої» L_p-норми нев'язки системи (4.1), кожному з яких відповідає свій метод розв'язання задачі (4.3), (4.4). L_1-нормі ($p = 1$) $\|y\|_1 = \sum_{i=1}^{m} |y_i|$, яка відома під назвою «манхеттенівська», відповідає метод найменших модулів. Стандартній евклідовій L_2-нормі ($p = 2$) $\|y\|_2 = \sqrt{\sum_{i=1}^{m} y_i^2}$ відповідає метод найменших

квадратів. Чебишевській L_∞-нормі ($p=\infty$) $\|y\|_\infty = \max\limits_{i=\overline{1,m}} |y_i|$ відповідає мінімаксний (чебишевський) метод. Однак форма задачі (4.3), (4.4) дає змогу розв'язувати її для довільних значень $p \geq 1$, використовуючи методи мінімізації негладких опуклих функцій.

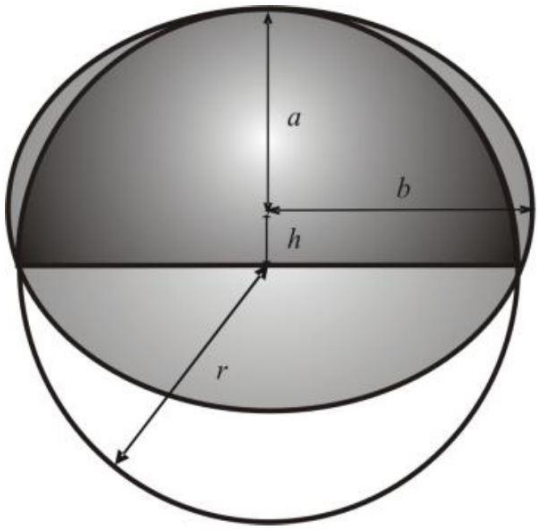
В цьому розділі ми розглянемо розв'язання систем лінійних рівнянь різного типу з використанням трьох субградієнтних методів з перетворенням простору: методу еліпсоїдів, субградієнтного методу з кроком Поляка у вихідному та перетвореному просторах змінних та методу *amsg2p*. Зокрема, в підрозділі 4.2 наведені опис та загальна схема методу еліпсоїдів, а в підрозділах 4.3 та 4.4 – його застосування для визначення параметрів лінійної регресії та знаходження L_p-розв'язку системи лінійних рівнянь з двосторонніми обмеженнями на змінні відповідно.

4.2 Метод еліпсоїдів: опис та загальна схема

Класичний метод еліпсоїдів був уперше запропонований в 1976 році Д.Б. Юдіним та А.С. Немировським (Юдин и Немировский 1976), за основу якого були покладені схеми послідовних відсічень. Він був названий модифікованим методом центрованих відсічень (ММЦВ). У 1977 році незалежно метод еліпсоїдів був відритий Н.З. Шором (1977) і представлений як частинний випадок субградієнтних методів з розтягом простору в напрямку субградієнта, дослідження яких було розпочато в 1969-1970 роках. Н.З. Шор вказав коефіцієнт розтягу простору та параметри регулювання кроку в напрямку нормованого антисубградієнту такими, щоб субградієнтний метод з розтягом простору збігався з геометричною швидкістю спадання об'єму еліпсоїда, в якому локалізована точка мінімуму опуклої функції. Завдяки цьому він отримав цілком прозоре обґрунтування (доведення) збіжності методу еліпсоїдів.

Метод еліпсоїдів базується на використанні в $\mathbb{R}^n$ еліпсоїда мінімального об'єму, який містить кулю, отриману внаслідок перетину n-вимірної кулі та

напівпростору, що проходить через його центр. Цей еліпсоїд сплюснутий в напрямку нормалі до гіперплощини, яка проходить через центр кулі з радіусом r. Він має такі параметри (див рис. 4.1): a – довжина меншої напіввісі в напрямку нормалі, що визначає напівкулю; b – довжина більшої напіввісі (кількість таких напівосей рівна $n-1$); h – відстань від центра кулі до центра еліпсоїда в напрямку меншої напіввісі.



$$a = r\frac{n}{n+1},$$

$$b = r\frac{n}{\sqrt{n^2-1}},$$

$$h = r\frac{n}{n+1}.$$

Рис 4.1. Еліпсоїд мінімального об’єму, що містить напівкулю в $\mathbb{R}^n$.

Ітерація методу еліпсоїдів полягає в переході від поточного еліпсоїда до наступного зі сталим коефіцієнтом зменшення їхніх об’ємів. Цей коефіцієнт визначається відношенням об’єму еліпсоїда з напівосями a та b до об’єму кулі з радіусом r в $\mathbb{R}^n$ і може бути записаний у вигляді

$$q_n = \left(\frac{a}{r}\right)\left(\frac{b}{r}\right)^{n-1} = \frac{n}{n+1}\left(\frac{n}{\sqrt{n^2-1}}\right)^{n-1} < 1. \qquad (4.5)$$

В роботі (Grötschel, Lovasz and Schriyver 1988) показано, що

$$q_n < \exp\left\{-\frac{1}{2n}\right\} < 1, \qquad (4.6)$$

отже при великих n коефіцієнт зменшення об'єму добре апроксимується асимптотичною формулою

$$q_n \approx 1 - \frac{1}{2n}. \tag{4.7}$$

Звідси, для зменшення об'єму еліпсоїда, який локалізує розв'язок задачі, в 10 разів, необхідно виконати K ітерацій, де

$$K = -\frac{\ln 10}{\ln q_n} \approx \left(2\ln 10\right) n \approx 4.6n \tag{4.8}$$

Розглянемо *узагальнений метод еліпсоїдів* (Стецюк, Фесюк и Хомяк 2018а), частинним випадком якого є класичний метод еліпсоїдів Юдіна – Неміровського – Шора. Він являє собою метод з розтягом простору, де коефіцієнт розтягу α задовольняє нерівність

$$\alpha + \frac{1}{\alpha} < 2\sqrt[n]{\alpha}. \tag{4.9}$$

Регулюванням кроку в цьому методі відбувається певним чином, пов'язаним з переходом в центр чергового локалізуючого еліпсоїда, об'єм якого менший, ніж у попереднього еліпсоїда. Узагальнений метод еліпсоїдів призначений для розв'язання такої задачі.

На евклідовому просторі $E^n (n \geq 2)$ розмірності n зі скалярним добутком (x, y) задано векторне поле $g(x) \in E^n$. Необхідно знайти точку x^* таку, щоб для всіх $x \in E^n$ виконувалась нерівність $\left(g(x), x - x^*\right) \geq 0$. Припустимо, що точка x^* існує та $g(x) \neq 0$ для $x \neq x^*$.

Загальну схему методу еліпсоїдів подано нижче.

Ініціалізація. Обираємо точку $x_0 \in \mathbb{R}^n$ та радіус r_0 такими, щоб $\left\| B_0^{-1}\left(x_0 - x^*\right)\right\| \leq r_0$, де B_0 – $n \times n$ матриця. Перейдемо до наступної ітерації зі значеннями x_0, r_0 та B_0.

Ітераційний процес. Нехай на k-й ітерації знайдені $x_k \in \mathbb{R}^n$, r_k та $n \times n$ матриця B_k. Для переходу на $(k+1)$-у ітерацію виконаємо такі дії.

Крок 1. Обчислимо $g_k = g(x_k)$. Якщо $g_k = 0$, тоді STOP ($x^* = x_k$).

Крок 2. Обчислюємо наступну точку

$$x_{k+1} = x_k - h_k B_k \xi_k, \quad \text{де } h_k = \frac{1}{2}\left(1 - \frac{1}{\alpha^2}\right) r_k, \quad \xi_k = \frac{B_k^T g_k}{\left\| B_k^T g_k \right\|}.$$

Крок 3. Перераховуємо матрицю B_{k+1} та радіус r_{k+1}:

$$B_{k+1} = B_k + \left(\frac{1}{\alpha} - 1\right)(B_k \xi_k)\xi_k^T, \qquad r_{k+1} = \frac{1}{2}\left(\alpha + \frac{1}{\alpha}\right) r_k.$$

Крок 4. Переходимо до $(k+1)$-ї ітерації зі значеннями x_{k+1}, r_{k+1} та B_{k+1}.

Теорема 4.1 (Стецюк, Фесюк и Хомяк 2018а). Послідовність точок $\{x_k\}_{k=0}^{\infty}$, що генерується узагальненим методом еліпсоїдів, задовольняє нерівності

$$\left\| A_k (x_k - x^*) \right\| \le r_k, \ k = 0,1,2,\ldots, \tag{4.10}$$

де $A_k = B_k^{-1}$. Якщо α задовольняє нерівність (4.9), тоді відношення об'ємів еліпсоїдів $E_k = \{x : \left\| A_k (x_k - x) \right\| \le r_k\}$ та $E_{k+1} = \{x : \left\| A_{k+1} (x_{k+1} - x) \right\| \le r_{k+1}\}$, які локалізують точку x^*, є величиною сталою і рівною

$$q_n(\alpha) = \frac{vol(E_{k+1})}{vol(E_k)} = \frac{1}{\alpha}\left(\frac{1}{2}\left(\alpha + \frac{1}{\alpha}\right)\right)^n < 1, \ k = 0,1,2,\ldots. \tag{4.11}$$

В теоремі 4.1 співвідношення (4.10) та (4.11) означають, що метод еліпсоїдів збігається (за об'ємом локалізації точки x^*) зі швидкістю геометричної прогресії зі знаменником $q_n(\alpha) < 1$. Величина знаменника залежить від обраного значення α, яке задовольняє нерівність (4.9). Найменший знаменник прогресії реалізується в методі еліпсоїдів Юдіна – Неміровського – Шора. Йому відповідає коефіцієнт розтягу $\alpha_1 = \sqrt{\frac{n+1}{n-1}}$, який досягається в точці мінімуму функції $q_n(\alpha)$ по α. Близький до найменшого знаменник прогресії реалізується в наближеному методі еліпсоїдів (Стецюк 2003а), якому відповідає коефіцієнт

розтягу $\alpha_2 = \sqrt{1+\frac{1}{n^2}} + \frac{1}{n}$. Він досягається в точці мінімуму функції $Q_n(\alpha)$, яка апроксимує зверху функцію $q_n(\alpha)$ згідно з таким співвідношенням:

$$q_n(\alpha) = \frac{1}{\alpha}\left(\frac{1}{2}\left(\alpha+\frac{1}{\alpha}\right)\right)^n = \frac{1}{\alpha}\left(1+\frac{1}{2}\left(\alpha+\frac{1}{\alpha}-2\right)\right)^n \le \frac{1}{\alpha}\exp\left\{\frac{n}{2}\left(\alpha+\frac{1}{\alpha}-2\right)\right\} = Q_n(\alpha).$$

При великих значеннях n знаменники геометричної прогресії в обох методах апроксимуються зверху близькими величинами $q^*(n) = 1-\frac{1}{2n}$ та $Q^*(n) = 1-\frac{1}{2n}+\frac{1}{2n^2}$.

Чудовою властивістю методу еліпсоїдів є те, що його швидкість збіжності залежить лише від розмірності простору змінних n та не залежить від властивостей задачі. Це означає, що він може бути вкрай ефективним при його застосуванні до задач невеликої розмірності. Хоча метод еліпсоїдів запропоновано Юдіним та Неміровським (1976) для розв'язання задач опуклого програмування, його можна успішно застосовувати для розв'язання більш широкого класу задач, таких як задача знаходження сідлових точок опукло-ввігнутих функцій, частинні випадки задач розв'язання варіаційних нерівностей, спеціальні класи задач лінійної та нелінійної доповненості тощо. Наведемо опис двох опуклих задач, для розв'язання яких можна застосовувати метод еліпсоїдів.

Задача безумовної оптимізації опуклої функції. Нехай $f(x), x \in \mathbb{R}^n$ – опукла функція. Позначимо її мінімальне значення як $f^* = f(x^*)$, і, не обмежуючи загальності, припустимо, що точка x^* – єдина точка мінімуму. Нехай є апріорна інформація про те, що точка x^* лежить в кулі $S(x_0, r) = \{x \in \mathbb{R}^n : \|x - x_0\| \le r\}$. Тоді для векторного поля $g(x) = g_f(x)$, де $g_f(x)$ – субградієнт функції $f(x)$ в точці x, виконується нерівність

$$(x - x^*, g(x)) = (x - x^*, g_f(x)) \ge f(x) - f(x^*) = f(x) - f^* \ge 0, \forall x \in \mathbb{R}^n. \quad (4.12)$$

Отже, для знаходження точки x^* можна використати метод еліпсоїдів, встановивши початкову точку x_0, початковий радіус $r_0 = r$ і матрицю $B_0 = I_n$, де I_n – одинична $n \times n$ матриця. Як критерій зупинки можна використовувати умову $r_k \left\| B_k^T g_f(x_k) \right\| \le \varepsilon$, яка при як завгодно малому ε дозволяє знайти точку $x_\varepsilon^* = x_k$, для якої $f\left(x_\varepsilon^*\right) - f^* \le \varepsilon$. Це випливає з того, що нерівність

$$r_k \ge \left\| B_k^{-1}\left(x_k - x^*\right) \right\| \ge \left(B_k^{-1}\left(x_k - x^*\right), \frac{B_k^T g_f(x_k)}{\left\| B_k^T g_f(x_k) \right\|} \right) = \frac{\left(x_k - x^*, g_f(x_k)\right)}{\left\| B_k^T g_f(x_k) \right\|} \ge \frac{f(x_k) - f^*}{\left\| B_k^T g_f(x_k) \right\|},$$

виконується для опуклої функції $f(x)$, враховуючи нерівність (4.12).

Загальна задача опуклого програмування. Необхідно знайти

$$f_0^* = f_0\left(x^*\right) = \min_{x \in \mathbb{R}^n} f_0(x) \tag{4.13}$$

при обмеженнях

$$f_i(x) \le 0,\ i = \overline{1,m}, \tag{4.14}$$

де $f_i(x)$ – опуклі функції, визначені на $\mathbb{R}^n$, $g_i(x)$ – відповідні субградієнти, $i = \overline{0,m}$. Припустимо, що оптимальна точка x^* існує і лежить в кулі $S(x_0, r)$ (формально до системи обмежень (4.14) можна додати обмеження $\left\| x - x_0 \right\| \le r$), і для задачі (4.13), (4.14) виконується умова Слейтера.

Розглянемо векторне поле $g(x)$, яке побудовано наступним чином:

$$g(x) = \begin{cases} g_0(x), & \text{якщо } \max\limits_{1 \le i \le m} f_i(x) \le 0 \\ g_{i^*}(x), & \text{якщо } \max\limits_{1 \le i \le m} f_i(x) = f_{i^*}(x) > 0. \end{cases} \tag{4.15}$$

Покажемо, що $\left(g(x), x - x^*\right) \ge 0$ для $\forall x \in \mathbb{R}^n$. Якщо $\max\limits_{1 \le i \le m} f_i(x) \le 0$, то $g(x) = g_0(x)$ і маємо

$$\left(g(x), x - x^*\right) = \left(g_0(x), x - x^*\right) \ge f_0(x) - f_0\left(x^*\right) \ge 0. \tag{4.16}$$

Якщо $\max\limits_{1 \le i \le m} f_i(x) > 0$, то $g(x) = g_{i^*}(x)$, причому $f_{i^*}(x) > 0$, $f_{i^*}\left(x^*\right) \le 0$, тоді маємо

$$\left(g(x), x - x^*\right) = \left(g_{i^*}(x), x - x^*\right) \geq f_{i^*}(x) - f_{i^*}\left(x^*\right) \geq 0. \qquad (4.17)$$

Отже, з (4.16), (4.17) випливає справедливість нерівності $\left(g(x), x - x^*\right) \geq 0$ для $\forall x \in \mathbb{R}^n$.

Отже, обчислюючи $g(x)$ за формулою (4.15), для локалізації оптимальної точки x^* в задачі (4.13), (4.14) можна використати узагальнений метод еліпсоїдів. Варто відмітити, що цей результат не зміниться, якщо у другій формулі (4.15) замість $g_{i^*}(x)$ взяти $g_{\bar{i}}$, де $\bar{i}$ – довільний індекс, для якого $f_{\bar{i}}(x) > 0$. Зупинка за умовою $r_k \left\| B_k^T g_0(x_k) \right\| \leq \varepsilon$ дозволяє знайти дочку $x_\varepsilon^* = x_k$, для якої $f_0\left(x_\varepsilon^*\right) - f_0^* \leq \varepsilon$, що випливає з істинності нерівності

$$r_k \geq \left\| B_k^{-1}\left(x_k - x^*\right) \right\| \geq \left(B_k^{-1}\left(x_k - x^*\right), \frac{B_k^T g_0(x_k)}{\left\| B_k^T g_0(x_k) \right\|} \right) = \frac{\left(x_k - x^*, g_0(x_k)\right)}{\left\| B_k^T g_0(x_k) \right\|} \geq \frac{f_0(x_k) - f_0^*}{\left\| B_k^T g_0(x_k) \right\|}$$

для опуклої функції $f_0(x)$. Зупинка за умовою $r_k \left\| B_k^T g_0(x_k) \right\| \leq \varepsilon$ використовується в програмній реалізації алгоритма на основі класичного методу еліпсоїдів Юдіна – Неміровського – Шора (Стецюк, Стовба и Мартынюк 2017а).

Однією з задач, для розв'язання якої можна успішно використовувати метод еліпсоїдів, є задача знаходження параметрів лінійної регресії, яку можна звести до задачі мінімізації опуклої функції (4.3). В наступному підрозділі ми розглянемо три приклади задач такого типу та результати обчислювальних експериментів з використанням методу еліпсоїдів для їхнього розв'язання.

4.3 Задача визначення параметрів лінійної регресії

Розглянемо регресійну модель

$$y = f(x, b) + \varepsilon, \quad E(\varepsilon) = 0, \qquad (4.18)$$

де (x_i, y_i), $i = \overline{1,n}$ – вибірка спостережень, причому $x_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}$. Також $b \in \mathbb{R}^k$ – вектор параметрів (коефіцієнтів) регресії, $\varepsilon \in \mathbb{R}^n$ – вектор похибок спостережень. Важливим припущенням є рівність нулю математичного сподівання вектору похибок ε. Якщо функція $f(x,b)$ є лінійною, тобто

$$f(x,b) = b_1 x_1 + ... + b_k x_k, \tag{4.19}$$

модель (4.18) називається *лінійною* регресією. Тут $x_j \in \mathbb{R}^n$ – регресори або фактори моделі. Необхідно знайти такий вектор параметрів b, при якому вектор похибок $\varepsilon = y - f(x,b) = (\varepsilon_1, ..., \varepsilon_n)^T$ буде мінімальним.

Якщо в моделі наявний лише один фактор (без врахування константи), тобто, $k = 2$, говорять про *парну* регресію, причому модель (4.18) набуває вигляду

$$y_i = b_0 + b_1 x_{1i} + \varepsilon_i, \quad i = \overline{1,n}, \tag{4.20}$$

де x_{1i} – значення першого (і єдиного) фактору в i-му спостереженні. Цю модель можна представити також у матричному вигляді:

$$y = Xb + \varepsilon, \tag{4.21}$$

де $y = (y_1, ..., y_n)^T$, $b = (b_0, b_1)^T$, X – $n \times 2$-матриця факторів, яка має вигляд

$$X = \begin{pmatrix} 1 & x_{11} \\ 1 & x_{12} \\ \vdots & \vdots \\ 1 & x_{1n} \end{pmatrix}.$$

Задачу (4.20) можна записати як задачу знаходження «найкращої» L_p-норми нев'язки ε системи (4.21), що відповідає задачі мінімізації опуклої функції

$$f_p(b) = \|Xb - y\|_p = \|\varepsilon\|_p = \left(\sum_{i=1}^{n} |\varepsilon_i|^p \right)^{1/p}, \tag{4.22}$$

тобто знаходження

$$b_p^* = \arg\min_{b \in \mathbb{R}^k} f_p(b). \tag{4.23}$$

Тут $p \in \mathbb{R}$ – скалярний параметр, який при $p \geq 1$ забезпечує опуклість функції $f_p(b)$. Задача (4.22), (4.23) завжди має розв'язок, а якщо $n > k$ та матриця X має повний ранг – цей розв'язок єдиний. В загальному випадку розв'язок не обов'язково єдиний і точка b_p^* належить множині оптимальних розв'язків, яким відповідає мінімальне значення функції f_p^* в задачі (4.22), (4.23).

Використовуючи значення $p = 1, 2, \infty$ задачу (4.22), (4.23) можна розв'язувати за допомогою методу найменших модулів, методу найменших квадратів та мінімаксного методу відповідно. Кожен з них дає свій розв'язок, який може бути більш якісним та стійким залежно від типу розподілу випадкової величини, що характеризує похибку результатів спостережень. В роботі (Стецюк, Стовба, и Жмуд 2018б) задачу (4.22), (4.23) розв'язано з використанням методу еліпсоїдів для значень параметра $1 \leq p \leq 2$. При довільних значеннях параметра p ця задача є загальною задачею безумовної мінімізації опуклої негладкої функції $F_p(b)$, для розв'язання якої можна використовувати методи мінімізації опуклих функцій, наприклад, метод еліпсоїдів або r-алгоритми. Їхнє використання потребує обчислення значення функції $f_p(b)$ та її субградієнта $g_f(b)$ в точці b; перше можна обчислити за формулою (4.22), друге – за формулою

$$g_f(b) = \|Xb - y\|_p^{1-p} \sum_{i=1}^{n} \operatorname{sgn}(x_i b - y_i) \cdot |x_i b - y_i|^{p-1} x_i^T \qquad (4.24)$$

При великих значеннях параметра p розрахунок цих величин є доволі трудомісткою задачею. В такому випадку мінімізація функції (4.22) еквівалентна мінімізації функції $f_p(b) = \max\limits_{i=1,n} |x_i b - y_i|$, що відповідає випадку $p = \infty$. Для спрощення обчислення та його коректності вираз (4.22) для обчислення значення функції пропонується змінити так:

$$f_p(b)=\left(\sum_{i=1}^{n}|\varepsilon_i|^p\right)^{1/p}=\frac{|\varepsilon_{max}|}{|\varepsilon_{max}|}\left(\sum_{i=1}^{n}|\varepsilon_i|^p\right)^{1/p}=\frac{|\varepsilon_{max}|}{\sqrt[p]{|\varepsilon_{max}|^p}}\left(\sum_{i=1}^{n}|\varepsilon_i|^p\right)^{1/p}=$$

$$=|\varepsilon_{max}|\left(\sum_{i=1}^{n}\frac{|\varepsilon_i|^p}{|\varepsilon_{max}|^p}\right)^{1/p}=|\varepsilon_{max}|\left(\sum_{i=1}^{n}\left|\frac{\varepsilon_i}{\varepsilon_{max}}\right|^p\right)^{1/p}=|\varepsilon_{max}|\left(\sum_{i=1}^{n}|z_i|^p\right)^{1/p}=$$

$$=|\varepsilon_{max}|\cdot\|z\|_p,\ \text{де}\ \varepsilon_{max}=\max_{i=\overline{1,n}}|\varepsilon_i|,\ z_i=\frac{\varepsilon_i}{\varepsilon_{max}}\le 1,\ i=\overline{1,n}. \qquad (4.25)$$

Формула (4.24) для обчислення субградієнта функції модифікується так:

$$g_f(b)=\|Xb-y\|_p^{1-p}\sum_{i=1}^{n}\operatorname{sgn}(x_ib-y_i)\cdot|x_ib-y_i|^{p-1}x_i^T=\|\varepsilon\|_p^{1-p}\sum_{i=1}^{n}\operatorname{sgn}(\varepsilon_i)\cdot|\varepsilon_i|^{p-1}x_i^T=$$

$$=\|\varepsilon\|_p^{1-p}\frac{|\varepsilon_{max}|^{p-1}}{|\varepsilon_{max}|^{p-1}}\sum_{i=1}^{n}\operatorname{sgn}(\varepsilon_i)\cdot|\varepsilon_i|^{p-1}x_i^T=\|\varepsilon\|_p^{1-p}|\varepsilon_{max}|^{p-1}\sum_{i=1}^{n}\operatorname{sgn}(\varepsilon_i)\cdot|z_i|^{p-1}x_i^T=$$

$$=\left(\frac{\|\varepsilon\|_p}{|\varepsilon_{max}|}\right)^{1-p}\sum_{i=1}^{n}\operatorname{sgn}(\varepsilon_i)\cdot|z_i|^{p-1}x_i^T=\left(\left(\sum_{i=1}^{n}|\varepsilon_i|^p\right)^{1/p}\Big/|\varepsilon_{max}|\right)^{1-p}\sum_{i=1}^{n}\operatorname{sgn}(\varepsilon_i)\cdot|z_i|^{p-1}x_i^T=$$

$$=\left(\left(\sum_{i=1}^{n}\left|\frac{\varepsilon_i}{\varepsilon_{max}}\right|^p\right)^{1/p}\right)^{1-p}\sum_{i=1}^{n}\operatorname{sgn}(\varepsilon_i)\cdot|z_i|^{p-1}x_i^T=\|z\|_p^{1-p}\sum_{i=1}^{n}\operatorname{sgn}(\varepsilon_i)\cdot|z_i|^{p-1}x_i^T,\quad (4.26)$$

де x_i – вектор-рядок матриці X з номером $i=\overline{1,n}$.

Якщо $\varepsilon_{max}\neq 0$ всі перетворення є коректними. Така умова порушується вкрай рідко і коректно опрацьовується в програмній реалізації методу за допомогою перевірки відповідної умови.

Ідея перетворення виразів (4.22) та (4.24) така. При великих значеннях параметра p обчислення L_p-норми вектора ε може призвести до накопичення обчислювальної похибки при виконанні операцій піднесення до степеню та додавання великих чисел. В модифікованих формулах (4.25) і (4.26) такі операції проводяться над елементами вектору z_i, які або рівні одиниці (їм відповідають максимальні елементи вектора ε_i), або менші одиниці. Якщо

$z_i = 1$ вищенаведені операції не змінюють його; якщо ж $z_i < 1$ при великих значеннях параметра p таке значення прямує до нуля та інтерпретується середовищем розробки як нуль після досягнення певного ліміту. Отже, операції виконуються коректно, а значення функції та її субградієнта обчислюються досить точно навіть при великих значеннях параметра p.

Приклад 4.1 (Кларк 1988, с. 10). Результати спостережень (x_i, y_i), $i = \overline{1,6}$, серед яких наявне одне аномальне, необхідно апроксимувати лінійною функцією $y = cx + d$, де c і d – невідомі параметри. Ця задача є прикладом задачі визначення параметрів лінійної регресії для функції однієї змінної. Результати спостережень (x_i, y_i), $i = \overline{1,6}$ наведені на рисунку 4.2.

x_i	y_i
0	0
1	1
2	2
3	3
4	4
5	0

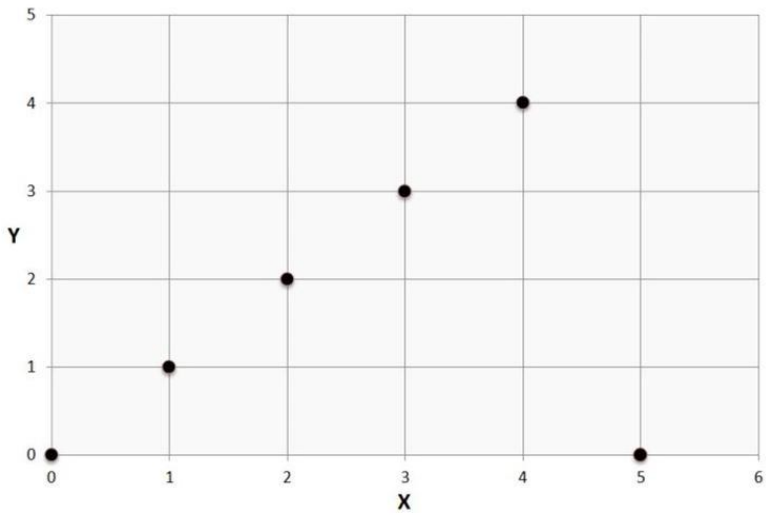


Рис. 4.2. Результати спостережень для функції однієї змінної.

Стецюк та Колесник (2002б) показали, що перевагу варто віддавати методу найменших модулів, адже він відкидає аномальне спостереження $(5,0)$ і точно відновлює лінійну функцію $y = x$. З огляду на спостереження (x_i, y_i) та вигляд лінійної функції $y = cx + d$, вихідну задачу можна записати в такому вигляді: знайти

$$\left(c_p^*, d_p^*\right) = \arg\min_{c,d} f_p\left(c,d\right), \tag{4.27}$$

де

$$f_p\left(c,d\right) = \left(\left|d\right|^p + \left|c+d-1\right|^p + \left|2c+d-2\right|^p + \left|3c+d-3\right|^p + \left|4c+d-4\right|^p + \left|5c+d\right|^p\right)^{\frac{1}{p}}.$$

Для розв'язання задачі (4.27) скористаємось алгоритмом на основі методу еліпсоїдів (Стецюк, Стовба и Мартынюк 2017а) і наведемо результати обчислювальних експериментів при різних значеннях $1 \le p \le 2$ в таблиці 4.1. Нагадаємо, що значенню параметра $p = 1$ відповідає МНМ, а значенню $p = 2$ – МНК.

Таблиця 4.1

Результати розв'язання задачі (4.27) алгоритмом на основі методу еліпсоїдів для значень $1 \le p \le 2$

p	itn	$\overline{c}_p^*$	$\overline{d}_p^*$	$f_p\left(\overline{c}_p^*, \overline{d}_p^*\right)$
1	199	1.00000	3.4424e – 12	5.0000
1.05	174	0.99996	0.00004	4.9999
1.1	136	0.99066	0.00934	4.9966
1.2	119	0.86343	0.13768	4.9047
1.4	101	0.57606	0.47512	4.4615
1.6	104	0.42249	0.70521	4.0324
1.8	111	0.33784	0.85195	3.7011
1.9	104	0.30896	0.90638	3.5672
1.95	109	0.29674	0.93029	3.5068
2	106	0.28571	0.95238	3.4503

Тут параметр p приймає значення з відрізку $[1,2]$ з кроком 0.1 та 0.2, зменшуючи його до 0.05 в околі точок 1 та 2. Друга колонка містить кількості

ітерацій методу еліпсоїдів для знаходження оптимальних параметрів $\bar{c}_p^*$ та $\bar{d}_p^*$ (третя і четверта колонки) з точністю $f_p\left(\bar{c}_p^*, \bar{d}_p^*\right) - f_p\left(c_p^*, d_p^*\right) \le 10^{-12}$. Остання колонка містить значення функції f_p в знайденій точці $\left(\bar{c}_p^*, \bar{d}_p^*\right)$. Використовувались початкова точка $(0,0)$ та радіус локалізації $r = 3$.

З таблиці 4.1 видно, що при $p = 1$ спостереження $(5,0)$ ігнорується і ми отримуємо лінійну функцію $y = x$; при значеннях p дуже близьких до одиниці вплив аномального спостереження теж відчувається мало. Однак при $p \in \{1.4, 1.6, 1.8\}$ вплив стає суттєвим: апроксимуючі прямі мають вже значно більший нахил в напрямку аномального спостереження $(5,0)$ та відхиляються від «правильної» прямої $y = x$. На рисунку 4.3 зображені спостереження (x_i, y_i) червоними точками, а також апроксимуючі прямі при $p = [1,2]$, причому «обмежуючі» прямі при $p = 1$ та $p = 2$ зображені суцільними лініями, а «внутрішні» при $p \in \{1.2, 1.4, 1.6, 1.8\}$ – штриховими лініями з різними розмірами штрихів.

Варто також відмітити зменшення кількості ітерацій (зі 199 до 106) та значення функції f_p по мірі зростання значення параметра p. У випадку $p = 2$, який відповідає МНК, похибка є мінімальною, однак це не гарантує «правильності» отриманого розв'язку. Для заданої вибірки спостережень не можна однозначно стверджувати, що точка $(5,0)$ є саме викидом, а не спостереженням, що вказує на нелінійний характер функції, якою варто апроксимувати спостереження (x_i, y_i). В такому разі МНК при $p = 2$ знаходить більш «правильний» розв'язок.

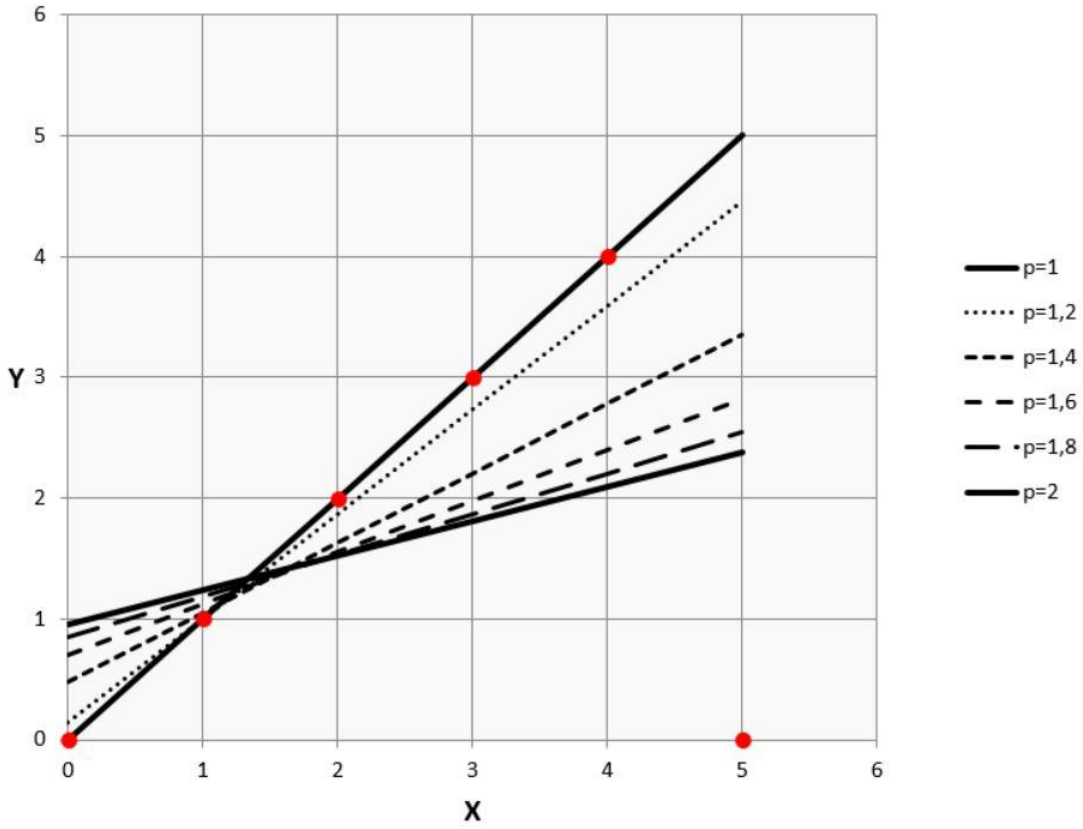


Рис 4.3. Найкращі лінійні функції при $p \in \{1, 1.2, 1.4, 1.6, 1.8, 2\}$.

Для визначення «правильності» розв'язку необхідно аналізувати явище додатковими способами, наприклад, враховувати думку експертів або вплив інших факторів, що можуть впливати на зміст спостережень тощо. Перевага постановки задачі у вигляді знаходження мінімального за L_p-нормою вектора похибок полягає в тому, що в кожному окремому випадку ми маємо можливість підбирати значення параметра p так, щоб відкидати або залишати аномальні спостереження у випадку їхньої істинності.

Розглянемо аналогічний приклад з більшою кількістю спостережень та аномалій для детальнішого аналізу поведінки класичних методів, що відповідають значенням $p = 1, 2, \infty$, а також алгоритма методу еліпсоїдів з великими значеннями параметра p.

Приклад 4.2. Розглянемо набір спостережень (x_i, y_i), де $x_i = y_i = i - 1$, $i = \overline{1,20}$, які необхідно апроксимувати лінійною функцією $y = cx + d$, де c та d

– невідомі. Для імітації 5 % аномальних спостережень змінимо перше спостереження так: $\left(x_1, y_1\right) = \left(0,19\right)$. Для досягнення 10 % аномалій окрім першого спостереження змінимо також друге: $\left(x_2, y_2\right) = \left(1,19\right)$, для 15 % – також третє: $\left(x_3, y_3\right) = \left(2,19\right)$. Отже, утворилась група аномалій ліворуч. Для побудови правосторонньої групи аномалій змінимо такі спостереження: $\left(x_{20}, y_{20}\right) = \left(19,0\right)$, $\left(x_{19}, y_{19}\right) = \left(18,0\right)$, $\left(x_{18}, y_{18}\right) = \left(17,0\right)$. Додаючи по одній аномалії, отримаємо 5 %, 10 % та 15 % викидів відповідно. Для апроксимації збурених спостережень використаємо алгоритм методу еліпсоїдів зі значеннями параметра $p = 1, 2, \infty$. Результати роботи алгоритма для збурених спостережень ліворуч наведено в таблиці 4.2, праворуч – в таблиці 4.3.

Таблиця 4.2

Апроксимація збурених ліворуч спостережень $\left(x_i, y_i\right)$ лінійною функцією за допомогою алгоритма методу еліпсоїдів при різних кількостях аномалій та значеннях параметра p

Відсоток аномалій	p	$\overline{c^*}$	$\overline{d^*}$	$f\left(\overline{c^*}, \overline{d^*}\right)$
5 %	1	1.0000	0.0000	19.000
	2	0.7285	3.5285	17.145
	∞	0.0000	3.0000	16.000
10 %	1	1.0000	0.0000	37.000
	2	0.4985	6.6142	21.196
	∞	0.0000	3.0000	16.000
15 %	1	1.0000	0.0000	54.000
	2	0.3067	9.2857	22.552
	∞	0.0000	3.0000	16.000

Таблиця 4.3

Апроксимація збурених праворуч спостережень $\left(x_i, y_i\right)$ лінійною функцією за допомогою алгоритма методу еліпсоїдів при різних кількостях аномалій та значеннях параметра p

Відсоток аномалій	p	$\overline{c^*}$	$\overline{d^*}$	$f\left(\overline{c^*}, \overline{d^*}\right)$
5 %	1	1.0000	– 7.5265e – 15	19.000
	2	0.72857	1.6286	17.145
	∞	5.8453e – 14	9.0000	9.0000
10 %	1	1.0000	3.3573e – 15	37.000
	2	0.4985	2.9143	21.197
	∞	5.5649e – 14	8.5000	8.500
15 %	1	1.0000	2.6091e – 14	54.000
	2	0.3067	3.8857	22.553
	∞	6.0668e – 14	8.0000	8.0000

У першій колонці таблиць 4.2 і 4.3 наведено процентне відношення аномалій до загального числа спостережень, у колонці 2 – значення параметра p, в колонках 3-5 – знайдені невідомі параметри та значення функції для них. Як бачимо, метод найменших модулів ($p=1$) точно відновлює лінійну функцію при будь-якому відсотку аномалій: параметр $\overline{c^*}$ рівний одиниці, а $\overline{d^*}$ рівний нулю при аномаліях ліворуч та прямує до нуля при аномаліях праворуч, що відповідає функції $y=x$. Метод найменших квадратів ($p=2$) відхиляється від оптимальної лінійної функції в бік групи аномалій, причому при збільшенні відсотку викидів це відхилення росте. Мінімаксний метод ($p=\infty$) апроксимує спостереження майже горизонтальною лінією, y-компонента якої залишається сталою при зростання проценту аномалій ліворуч і змінюється, якщо аномалії розташовані праворуч. Отже, отримані результати демонструють стійкість

методу найменших модулів до появи аномалій різного характеру, яка не спостерігається при використанні МНК або мінімаксного методу.

Оскільки алгоритм методу еліпсоїдів дає змогу розв'язувати вихідну задачу для довільного значення параметра p, розглянемо результати його роботи для цього ж прикладу з правосторонньою групою аномалій об'ємом 15 %. Отримані результати порівняємо з результатами роботи мінімаксного методу ($p=\infty$), в якому необхідно мінімізувати функцію максимуму

$$f_\infty(b)=\max_{i=\overline{1,n}}\left|x_i b-y_i\right|. \qquad (4.28)$$

Результати тестувань наведено в таблиці 4.4.

Таблиця 4.4

Апроксимація збурених праворуч спостережень $\left(x_i, y_i\right)$ лінійною функцією за допомогою алгоритма методу еліпсоїдів при великих значеннях параметра p

p	itn	$\overline{c_p^*}$	$\overline{d_p^*}$	$f_p\left(\overline{c_p^*}, \overline{d_p^*}\right)$
10	112	4.4854e – 02	6.8507	9.3126
10^2	120	4.2751e – 03	7.8780	8.1090
10^3	130	4.2764e – 04	7.9878	8.0109
10^4	141	4.2763e – 05	7.9988	8.0011
10^5	151	4.2766e – 06	7.9999	8.0001
10^6	150	4.2789e – 07	8.0000	8.0000
∞	225	6.0668e – 14	8.0000	8.0000

З таблиці 4.4 видно, що при зростанні p значення параметрів $\overline{c_p^*}$ та $\overline{d_p^*}$ прямують до значень 6.0668e-14 та 8.0000 відповідно. Така поведінка означає, що апроксимуюча пряма все більше наближається до горизонтальної прямої, що відповідає мінімаксному методу Чебишева та випадку $p=\infty$. Отже обчислення за допомогою формул (4.25) та (4.26) виконуються коректно.

Приклад 4.3. Наявні 28 векторних спостережень (u_i, f_i), $u_i \in \mathbb{R}^4$, $f_i \in \mathbb{R}$, взятих з анкети опитування (Стецюк и Колесник 2002б), які необхідно «найкращим чином» наблизити квадратичною функцією $Q(u) = u^T Au + b^T u + c$, яка визначається такими параметрами: симетричною 4×4-матрицею A, вектором $b \in \mathbb{R}^4$ та скаляром $c \in \mathbb{R}$. Як аргумент вона приймає вектор $u \in \mathbb{R}^4$. Аналогічно до прикладу 4.1, цю задачу можна сформулювати як задачу мінімізації опуклої функції, а саме: знайти

$$\left(A_p^*, b_p^*, c_p^*\right) = \arg\min_{A,b,c} F_p\left(A, b, c\right), \tag{4.29}$$

де

$$F_p\left(A, b, c\right) = \left(\sum_{i=1}^{28} \left| f_i - Q\left(u_i\right) \right|^p \right)^{1/p} = \left(\sum_{i=1}^{28} \left| f_i - u_i^T A u_i + b^T u_i + c \right|^p \right)^{1/p}. \tag{4.30}$$

Отже, необхідно визначити трійку параметрів A_p^*, b_p^*, c_p^*, які визначають квадратичну функцію $Q_p^*(u) = u^T A_p^* u + \left(b_p^*\right)^T u + c_p^*$ та мінімізують функцію $F_p\left(A, b, c\right)$. Оскільки матриця A є симетричною, загальна кількість параметрів функції $Q(u)$ є такою:

$$N = \frac{4(4+1)}{2} + 4 + 1 = 15,$$

серед яких 10 параметрів задають компоненти матриці A, 4 параметри – компоненти вектору b та 1 параметр задає скаляр c.

Задача (4.22), (4.23) полягає в знаходженні «найкращої» L_p-норми вектора нев'язок ε для системи (4.21), яка задається матрицею X, що містить першу компоненту спостережень (x_i, y_i), та вектором параметрів b. Якщо 15 невідомих параметрів функції $Q(u)$ розмістити у векторі розмірності 15, а також спеціальним чином побудувати матрицю X, задачу (4.29), (4.30) можна переформулювати в термінах задачі (4.22), (4.23), а саме: знайти

$$\beta_p^* = \arg\min_{\beta \in \mathbb{R}^{15}} \left\| X\beta - f \right\|_p, \tag{4.31}$$

Тут $f \in \mathbb{R}^{28}$ – друга компонента спостережень (u_i, f_i). Вектор параметрів $\beta \in \mathbb{R}^{15}$ має таку структуру:

$$\beta = \left[(a_{11} \dots a_{nn}) \left(a_{ij}\right)_{i<j} (b_1 \dots b_n) c \right],$$

де $a_{11} \dots a_{nn}$ та $a_{ij}, i < j$ – діагональні та наддіагональні елементи матриці A, $b_1 \dots b_n$ – компоненти вектора b, c – невідомий скаляр. Матриця X розмірності 28×15 є блочною матрицею та має такий вигляд:

$$X = \left[U^2 \; W \; U \; e \right].$$

Тут $U = \{u_i\}_{i=1}^{28}$ – матриця розмірності 28×4, що складається з векторів-рядків u_i – першої компоненти спостережень; U^2 – це матриця U, елементи якої піднесені до квадрату. Матриця $W = \left\{ \left(2u_i^k u_i^l \right)_{k<l} \right\}_{i=1}^{28}$ побудована так: її i-й рядок є набором впорядкованих попарних подвійних добутків компонент вектора u_i, причому $k < l$, $1 \le k \le 3$, $2 \le l \le 4$. Нарешті e – одиничний вектор-стовпчик з $\mathbb{R}^{28}$.

Задачу (4.31) будемо розв'язувати за допомогою алгоритма методу еліпсоїдів. Вхідними параметрами є початкова точка $x_0 = (0,\dots,0)^T$, радіус локалізації $r = 1$, точність $\varepsilon_f = 10^{-3}$ та значення параметра $p = 1, 1.3, 1.6, 2, 5$. В таблиці А.1 наведені відхилення $f_i - Q_p^*(u_i)$ для всіх $i = \overline{1,28}$ при різних значень $p \ge 1$, а також всі 28 спостережень (u_i, f_i) з анкети опитування.

Аналізуючи таблицю 4.5 (див. додаток Г), можна зробити декілька висновків.

По-перше, якщо при $p = 1$ не враховувати спостереження з номерами 26-27, то максимальне відхилення серед інших 26-ти значень становить всього 0.005, тоді як при $p = 2$ воно становить 0.022, причому наявні також інші доволі вагомі відхилення: -0.018 (номери 26-27), 0.019 (номер 13), 0.014

(номери 22-23). Загалом серед всіх відхилень при $p=1$ є лише декілька відмінних від нуля (а саме 10), тоді як при $p=2$ таких відхилень 26. Такі висновки вказують на те, що для цього прикладу МНМ ($p=1$) демонструє суттєво кращі результати, ніж МНК ($p=2$), які дуже схожі на результати з прикладу 4.1. МНМ відкидає аномальні спостереження, тоді як МНК враховує їх, зміщуючи розв'язок.

По-друге, у міру того, як значення параметра p прямує від 1 до 2, відхилення зростають. Однак при $p=5$ вони несуттєво зменшуються або залишаються незмінними. Очевидно, залежність значення параметра p від відхилень є нелінійною, тому необхідно провести додаткові дослідження цієї залежності для коректного підбору параметра p.

Однак остаточний вибір значення параметра p визначається конкретним набором спостережень. Наприклад, наявність аномальних спостережень під номерами 26 та 27 може бути спричинена похибкою зчитування даних або помилкою експерта, який визначив значення f_i; в такому випадку повторне зчитування та перегляд значень експертом можуть покращити ситуацію. З іншого боку, якщо випадкова величина, яка характеризує похибку результатів спостережень, має нормальний розподіл ймовірностей, доцільним кроком буде використання МНК. Якщо ж вона має розподіл Пуассона, більш доцільно використовувати МНМ. Можливість використання параметра p дає можливість спростити розв'язання задач, описаних в цьому підрозділі, адже замість двох методів (МНМ та МНК) можна використовувати лише один, а також гнучко підбирати значення параметра p для отримання «найкращих» розв'язків.

4.4 Задача знаходження L_p-розв'язку системи лінійних рівнянь з двосторонніми обмеженнями на змінні

В цьому підрозділі ми розглянемо дві алгоритмічні реалізації методу еліпсоїдів для розв'язання задачі опуклого програмування, пов'язаної зі знаходженням розв'язку системи лінійних рівнянь при двосторонніх обмеженнях на змінні. Перша реалізація використовує метод еліпсоїдів Шора з корекцією несиметричної матриці, друга реалізація – метод еліпсоїдів Юдіна – Неміровського з корекцією симетричної матриці.

Розглянемо систему лінійних рівнянь

$$Ax \approx b, \tag{4.32}$$

за умов

$$l \le x \le u. \tag{4.33}$$

Тут A – $m \times n$-матриця, $b \in \mathbb{R}^m$ – m-вимірний вектор, $l \in \mathbb{R}^n$, $u \in \mathbb{R}^n$ – n-вимірні вектори, для яких $u_i > l_i$ для всіх $i = \overline{1,n}$; $x \in \mathbb{R}^n$ – n-вимірний вектор невідомих параметрів. Необхідно знайти такий вектор $x_p^* \in \mathbb{R}^n$, який є «найкращим» розв'язком системи (4.32), (4.33) в L_p-нормі, тобто коли норма вектору нев'язок $y = Ax - b = (y_1, ..., y_m)^T$ визначена так: $\|y\|_p = \left(\sum_{i=1}^{m} |y_i|^p \right)^{1/p}$, де $p \ge 1$.

Задачі (4.32), (4.33) відповідає задача мінімізації опуклої функції: треба знайти

$$f_p^* = f_p(x_p^*) = \min_{x \in \mathbb{R}^n} f_p(x) = \min_{x \in \mathbb{R}^n} \|Ax - b\|_p \tag{4.34}$$

за обмежень

$$l \le x \le u, \tag{4.35}$$

де $p \in \mathbb{R}$ – скалярний параметр, такий, що $p \ge 1$, що гарантує опуклість функції $f_p(x)$. Як і задача без обмежень на змінні (підрозділ 4.1) задача (4.34), (4.35) завжди має розв'язок. Якщо ранг матриці A рівний n цей розв'язок єдиний; в

загальному випадку існує множина розв'язків, з якої ми будемо обирати той, якому відповідає оптимальне значення функції f_p^*.

Для того, щоб застосувати метод еліпсоїдів для розв'язання задачі (4.34), (4.35), необхідно визначити градієнтне поле $g(x)$ (спосіб побудови в точці $x \in \mathbb{R}^n$ гіперплощини, яка локалізує точку x_p^* в одному з напівпросторів простору $\mathbb{R}^n$) та вибрати початковий радіус області локалізації оптимального розв'язку x_p^*. Для виконання першої вимоги скористаємось такою лемою.

Лема 4.1 (Стецюк, Колесник и Березовский 2003б). Нехай $\partial f_p(x)$ – субградієнт функції $f_p(x)$ в точці x; $t^* = \max\{t_{i^*}, t_{j^*}\}$, $t_{i^*} = \max\limits_{i=\overline{1,n}}\{x_i - u_i\}$, $t_{j^*} = \max\limits_{j=\overline{1,n}}\{l_j - x_j\}$; i^*, j^* – значення i, j $(1 \le i, j \le n)$, на яких досягаються t_{i^*}, t_{j^*}; e_k – k-й орт в E^n, $1 \le k \le n$. Тоді вектор

$$g_p(x) = \begin{cases} \partial f_p(x), & \text{якщо } t^* \le 0 \\ e_{i^*}, & \text{якщо } t^* > 0 \text{ та } t^* > t_{i^*} \\ -e_{j^*}, & \text{якщо } t^* > 0 \text{ та } t^* \le t_{j^*} \end{cases}$$

задовольняє нерівність

$$\left(g_p(x), x - x_p^*\right) \ge 0 \text{ для всіх } x \in \mathbb{R}^n. \qquad (4.36)$$

Пояснимо лему 4.1. Якщо точка x лежить всередині допустимої області, що задається обмеженнями (4.35), то як $g_p(x)$ обирається $\partial f_p(x)$ – субградієнт функції $f_p(x)$ в цій точці, який обчислюється за формулою

$$\partial f_p(x) = \|Ax - b\|_p^{1-p} \sum_{j=1}^{m} \left(sign\left(a_j x - b_j\right) \cdot \left|a_j x - b_j\right|^{p-1} a_j^T \right),$$

де a_j – вектор-рядок матриці A з номером j, $j = \overline{1,m}$. Якщо ж точка лежить поза допустимою областю, то обирається субградієнт до максимально порушеного обмеження у вигляді (4.35). Опуклість функції $f_p(x)$ та обмежень (4.35) для векторного поля $g_p(x)$ гарантує виконання умови (4.36).

Апріорну інформацію про локалізацію точки x_p^* визначимо так: центром кулі буде центр паралелепіпеда, що задається двосторонніми обмеженнями на змінні (4.35), а її радіус покладемо таким, щоб куля містила в собі паралелепіпед і мала мінімальний об'єм. Це забезпечує така лема.

Лема 4.2 (Стецюк, Колесник и Березовский 2003б). Якщо $x_0 = \frac{1}{2}(u+l)$ та $r_0 = \frac{1}{2}\|u-l\|$, то паралелепіпед $P(x) = \{x : l \le x \le u\}$ міститься в n-вимірній кулі $S(x_0, r_0) = \{x : \|x - x_0\| \le r_0\}$.

Враховуючи метод еліпсоїдів Н.З. Шора (1977) та леми 4.1, 4.2, наведемо алгоритм Шора для розв'язанні задачі (4.34), (4.35), який вперше викладено в роботі (Стецюк, Колесник и Березовский 2003б).

Алгоритм Шора для знаходження x_p^*. Згідно з правилом обчислення $g_p(x)$ леми 4.1 побудуємо формулу для обчислення «узагальненого» значення функції в задачі (4.34), (4.35):

$$F_p(x) = \begin{cases} +\infty, & \text{якщо } t^* > 0; \\ f_p(x), & \text{якщо } t^* \le 0. \end{cases}$$

Значення $F_p(x)$ використаємо при побудові критерію зупинки в алгоритмі знаходження x_p^*. Вхідними параметрами алгоритма будуть величина p ($p \ge 1$), за допомогою якої визначена L_p-норма в (4.34), та величина ε_f – точність, з якою необхідно знайти значення $f_p^* = f_p(x_p^*)$.

Ініціалізація. Початкову точку покладемо $x_0 = \frac{1}{2}(u+l)$, початковий радіус – $r_0 = \frac{1}{2}\|u-l\|$. Розглянемо $n \times n$-матрицю B і покладемо $B_0 = I_n$, де I_n – одинична $n \times n$-матриця. Перейдемо до першої ітерації зі значеннями x_0, r_0 та B_0.

Нехай на k-й ітерації ми отримали значення $x_k \in E^n$, r_k та B_k. Для переходу на $(k+1)$-у ітерацію виконаємо таку послідовність кроків.

Крок 1. Обчислимо $F_p(x_k)$. Якщо $F_p(x_k)=0$, тоді STOP: $k^*=k$ та $x_p^*=x_k$. Інакше обчислимо $g_p(x_k)$. Якщо $F_p(x_k)<+\infty$ та $\left\|B_k^T g_p(x_k)\right\| r_k \le \varepsilon_f$, тоді STOP: $k^*=k$ та $x_p^*=x_k$. Інакше переходимо до кроку 2.

Крок 2. Покладемо $\xi_k = \dfrac{B_k^T g_p(x_k)}{\left\|B_k^T g_p(x_k)\right\|}$.

Крок 3. Обчислимо наступну точку $x_{k+1}=x_k - h_k B_k \xi_k$, де $h_k = \dfrac{1}{n+1} r_k$.

Крок 4. Обчислимо $B_{k+1} = B_k + \left(\sqrt{\dfrac{n-1}{n+1}} - 1\right)\left(B_k \xi_k\right)\xi_k^T$ та $r_{k+1} = r_k \dfrac{n}{\sqrt{n^2-1}}$.

Крок 5. Переходимо до $(k+1)$-ї ітерації зі значеннями x_{k+1}, r_{k+1} та B_{k+1}.

Збіжність алгоритма Шора забезпечує така теорема.

Теорема 4.2 (Стецюк, Колесник и Березовский 2003б). Послідовність точок $\{x_k\}_{k=0}^{k^*}$ задовольняє нерівності

$$\left\|B_k^{-1}\left(x_k - x_p^*\right)\right\| \le r_k,\ k=\overline{0,k^*}.$$

На кожній ітерації $k>0$ величина зменшення об’єму еліпсоїда $E_k = \left\{x \in \mathbb{R}^n : \left\|B_k^{-1}(x_k - x)\right\| \le r_k\right\}$, який локалізує x_p^*, є сталою і рівною

$$q = \frac{vol(E_k)}{vol(E_{k-1})} = \frac{n}{n+1}\left(\frac{n}{\sqrt{n^2-1}}\right)^{n-1} < \exp\left\{-\frac{1}{2n}\right\} < 1.$$

З теореми 4.2 випливає, що для зменшення в 10 разів об’єму еліпсоїда, який локалізує x_p^*, необхідно $K = 4.6n$ ітерацій (див. (4.5)-(4.8)). Для задачі (4.34), (4.35) це означає, що для збільшення на один порядок відхилення знайденого рекордного значення функції $f_p(x)$ від її оптимального значення f_p^*, необхідно виконати $4.6n^2$ ітерацій.

Окрім методу еліпсоїдів Шора існує також метод еліпсоїдів Д.Б. Юдіна та А.С. Неміровського (1976), який є варіантом методів послідовних відсічень і працює з симетричною матрицею $H = BB^T$, де B – $n \times n$-матриця в алгоритмі Шора. Нижче запропоновано алгоритм Юдіна-Неміровського для розв'язання задачі (4.34), (4.35).

Алгоритм Юдіна-Неміровського для знаходження x_p^*. Вхідними параметрами алгоритма є величини p ($p \geq 1$) та ε_f – точність, з якою необхідно знайти $f_p\left(x_p^*\right) = f_p^*$.

Ініціалізація. Початкову точку покладемо $x_0 = \frac{1}{2}(u + l)$, початковий радіус – $r_0 = \frac{1}{2}\|u - l\|$. Розглянемо симетричну $n \times n$-матрицю H і покладемо $H_0 = I_n$, де I_n – одинична $n \times n$-матриця. Перейдемо до першої ітерації зі значеннями x_0, r_0 та H_0.

Нехай на k-й ітерації ми отримали значення $x_k \in E^n$, r_k та H_k. Для переходу на $(k+1)$-у ітерацію виконаємо таку послідовність кроків.

Крок 1. Обчислимо $F_p(x_k)$. Якщо $F_p(x_k) = 0$, тоді STOP: $k^* = k$ та $x_p^* = x_k$. Інакше обчислимо $g_p(x_k)$. Якщо $F_p(x_k) < +\infty$ та $r_k \sqrt{g_p^T(x_k) H_k g_p(x_k)} \leq \varepsilon_f$, тоді STOP: $k^* = k$ та $x_p^* = x_k$. Інакше переходимо до кроку 2.

Крок 2. Обчислимо наступну точку

$$x_{k+1} = x_k - h_k \frac{H_k g_p(x_k)}{\sqrt{g_p^T(x_k) H_k g_p(x_k)}}, \quad \text{де } h_k = \frac{1}{n+1} r_k.$$

Крок 3. Обчислимо

$$H_{k+1} = H_k - \frac{2}{n+1} \frac{H_k g_p(x_k) g_p^T(x_k) H_k}{g_p^T(x_k) H_k g_p(x_k)} \quad \text{та} \quad r_{k+1} = r_k \frac{n}{\sqrt{n^2 - 1}}.$$

Крок 4. Переходимо до $(k+1)$-ї ітерації зі значеннями x_{k+1}, r_{k+1} та H_{k+1}.

Збіжність алгоритма Юдіна-Неміровського забезпечує така теорема.

Теорема 4.3. Послідовність точок $\{x_k\}_{k=0}^{k^*}$ задовольняє нерівності

$$\left(x_k - x_p^*\right) H_k^{-1} \left(x_k - x_p^*\right) \le r_k^2,\ k = \overline{0, k^*}.$$

На кожній ітерації $k > 0$ величина зменшення об'єму еліпсоїда $E_k = \left\{x \in \mathbb{R}^n : \left(x_k - x_p^*\right) H_k^{-1} \left(x_k - x_p^*\right) \le r_k^2\right\}$, який локалізує x_p^*, є сталою і рівною

$$q = \frac{vol\left(E_k\right)}{vol\left(E_{k-1}\right)} = \frac{n}{n+1}\left(\frac{n}{\sqrt{n^2-1}}\right)^{n-1} < \exp\left\{-\frac{1}{2n}\right\} < 1.$$

Варто відмітити, що описані алгоритми можна успішно застосовувати для знаходження x_p^* при невеликій кількості змінних. Наприклад, для знаходження точки мінімуму опуклої функції з відносною точністю за значенням функції $\varepsilon_f = 10^{-10}$ при $n = 10$ методу еліпсоїдів необхідно виконати 4600 ітерацій. Для сучасних комп'ютерів таке число ітерацій вимагає не більше секунди процесорного часу. Також наведені алгоритми є стійкими у випадку розв'язання погано-обумовлених систем лінійних рівнянь.

Також підкреслимо, що величина m суттєво не впливає на швидкість збіжності алгоритмів, однак від неї залежить складність обчислення значення функції $f_p(x)$ та її субградієнта $\partial f_p(x)$. При $m \sim 1000$ ці обчислення вноситимуть більш вагомий внесок в трудомісткість обох алгоритмів, ніж алгоритмічні операції – кроки 2-4 в алгоритмі Шора та кроки 2-3 в алгоритмі Юдіна – Неміровського.

4.5 Задача знаходження розв'язку сумісних систем лінійних рівнянь

В цьому підрозділі ми розглянемо задачу розв'язання сумісної системи лінійних рівнянь за допомогою двох субградієнтних методів: субградієнтного методу з кроком Поляка у вихідному та перетвореному просторах змінних (див. розділи 2 і 3 відповідно) та методу *amsg2p*.

Розглянемо сумісну систему лінійних рівнянь

$$Ax = b, \tag{4.37}$$

де A – дійсна $n \times p$ матриця, $b \in \mathbb{R}^p$ – вектор правих частин, $x \in \mathbb{R}^n$ – вектор невідомих змінних, які необхідно знайти.

Множину розв'язків системи (4.37) позначимо X^*. Якщо $n = p$ та A – невироджена $n \times n$ матриця, то система (4.37) має єдиний розв'язок $x = A^{-1}b$, де A^{-1} – обернена до A матриця. Якщо матриця A погано обумовлена, знаходження точки x^*, як правило, є досить складною задачею.

Для зручності систему лінійних рівнянь (4.37) будемо розглядати в тотожному вигляді

$$(a_i, x) = b_i, \quad i = \overline{1, p}, \tag{4.38}$$

де $a_i \in \mathbb{R}^n$ – вектор, отриманий транспонуванням i-го рядка матриці A.

Розглянемо три способи запису задачі розв'язання системи лінійних рівнянь (4.37) як задачі знаходження мінімуму опуклої функції – гладкої або негладкої (Стецюк 2001, с. 263-264):

1) знайти точку мінімуму опуклої функції

$$f_1(x) = \|Ax - b\|^2, \tag{4.39}$$

де $f_1(x)$ – гладка опукла квадратична функція для якої $f_1^* = 0$; її субградієнт $g_{f_1(x)}$ обчислюється за формулою

$$g_{f_1(x)} = 2A^T(Ax - b); \tag{4.40}$$

2) знайти точку мінімуму опуклої функції

$$f_2(x) = \sum_{i=1}^{p} \left| (a_i, x) - b_i \right|, \tag{4.41}$$

де $|\cdot|$ – модуль (абсолютна величина) числа; $f_2(x)$ – негладка опукла функція, для якої $f_2^* = 0$, а її субградієнт $g_{f_2(x)}$ обчислюється за формулою

$$g_{f_2(x)} = \sum_{i=1}^{p} \delta_i a_i, \quad \text{де } \delta_i = \begin{cases} 1, & \text{якщо } (a_i, x) - b_i \geq 0; \\ -1, & \text{якщо } (a_i, x) - b_i < 0; \end{cases} \tag{4.42}$$

3) знайти точку мінімуму опуклої функції

$$f_3(x) = \max_{i=\overline{1,p}} \left| (a_i, x) - b_i \right|, \tag{4.43}$$

де $f_3(x)$ – теж негладка опукла функція, $f_3^* = 0$, а її субградієнт $g_{f_3(x)}$ обчислюється за формулою

$$g_{f_3(x)} = \delta_{i^*} a_{i^*}, \quad \text{де } \delta_{i^*} = \begin{cases} 1, & \text{якщо } (a_i, x) - b_i \geq 0; \\ -1, & \text{якщо } (a_i, x) - b_i < 0. \end{cases} \tag{4.44}$$

Тут i^* – індекс з $i \in \{1,..., p\}$, при якому модуль лінійної функції $(a_i, x) - b_i$ досягає свого максимуму. У разі неоднозначності такого максимуму можна обрати довільний індекс, при якому він досягається.

Знаходження ε-розв'язків опуклих нерівностей (4.39), (4.41), (4.43) рівносильне рівності або достатньої близькості до нуля суми квадратів нев'язок, яку задає функція $f_1(x)$, суми модулів нев'язок, яку задає функція $f_2(x)$, та максимуму модулів нев'язок, що задається функцією $f_3(x)$. Ця близькість визначається параметром ε. Отже, задачу можна розв'язати, знайшовши точки мінімумів функцій $f_i(x), i = \overline{1,3}$ з точністю ε.

Якщо $X^* \neq \varnothing$, то точка мінімуму кожної з функцій $f_i(x), i = \overline{1,3}$ збігається з розв'язком СЛАР (4.41), якщо вона єдина, або з одним із розв'язків, якщо система (4.41) має нескінченну кількість розв'язків. Залежно від вибору функції ми отримаємо свій метод для знаходження розв'язку СЛАР. Назвемо ці методи за типом функції, тобто методу 1 відповідатиме функція $f_1(x)$, методу 2 – $f_2(x)$, і методу 3 – $f_3(x)$.

Для знаходження наближень до точок мінімуму функцій $f_i(x), i = \overline{1,3}$ скористаємось двома субградієнтними методами з кроком Поляка – у вихідному та перетвореному просторах змінних, описаними в розділах 2 і 3, а також ще одним субградієнтним методом *amsg2p*, чий детальний опис наведений у звіті (Стецюк та ін 2016а). Як і субградієнтний метод з кроком Поляка він дає змогу знайти точку мінімуму опуклої функції $f(x)$ при

відомому значенні f^* або його верхній оцінці (Стецюк 2014, с. 386-387). У назві методу «ams» означає, що в ітераційному процесі використовується крок Аґмона – Моцкіна – Шонберга (крок Поляка) в напрямку нормованого антисубградієнту, а «g2p» вказує, що ams-крок використовується в просторі змінних, перетвореному за допомогою двох послідовних субградієнтів (g2) та агрегатного вектора (p). Перетворення простору в методі *amsg2p* проводиться за допомогою однорангового еліпсоїдального оператора (Stetsyuk 2017, Стецюк 2017в), і лише на тих ітераціях методу, коли тупим є хоча б один з кутів – або кут між двома послідовними субградієнтами, або кут між останнім субградієнтом та агрегатним вектором, який є опуклою комбінацією обчислених на попередніх ітераціях субградієнтів.

Подібно до того, як це зроблено в r-алгоритмах Шора та субградієнтному методі з кроком Поляка, перетворення простору в методі *amsg2p* спрямоване на зменшення витягнутості поверхонь рівня опуклих функцій. Для яружних функцій це забезпечує прискорену збіжність методу при довільній початковій точці та досить малих значеннях точності ($\varepsilon_f = 10^{-10} - 10^{-14}$). Варто відмітити, що якщо функція, що мінімізується, не є яружною, метод *amsg2p* переходить у субградієнтний метод з кроком Поляка, тому перший метод є певним узагальненням другого.

Для обчислювальних експериментів ми будемо використовувати програми *PolyakA* та *PolyakB* з розділів 2 і 3 (Octave-реалізації субградієнтного методу з кроком Поляка), а також однойменну програму *amsg2p*, написану мовою Octave (Стецюк 2019в). Вона знаходить точку x_ε^*, де значення опуклої функції $f\left(x_\varepsilon^*\right) \le f^* + \varepsilon_f$ і визначається такими вхідними параметрами: стартовою точкою x_0, радіусом r_0, параметром γ (аналог параметра m у методі Поляка) – зміщенням по опуклості, та параметрами зупинки – ε_g, ε_x та *maxitn*. Як і програми *PolyakA* та *PolyakB* програма *amsg2p* використовує функцію **function**

[f,g] = calcfg(x) для обчислення значення функції та її субградієнта в заданій точці.

Визначимо вхідні дані. В прикладі 2.3 розділу 2 ми вже використовували матрицю, яка утворювалась з використанням генератора випадкових чисел з відрізку [0,3]. В цьому прикладі ми визначимо дві матриці A_1 та A_2 розмірності 500×100: перша заповнюється випадковими числами за допомогою того ж генератора з відрізку [0,3] (далі відрізок 1), друга – відрізку [3,10] (далі відрізок 2). Вектор правих частин b_i визначається так: $b_i=\sum_{j=1}^{p}a_{ij},\ i=\overline{1,n}$. Елементи матриці перетворення простору $B=\left(b_{ij}\right)_{i,j=1}^{p}$ в субградієнтному методі з кроком Поляка в перетвореному просторі покладемо такими: $b_{ii}=0.64,\ i=\overline{1,3}$, $b_{44}=0.67$, $b_{55}=0.63$, $b_{ii}=0.7,\ i=\overline{8,14}$, $b_{66}=b_{20\,20}=b_{15\,15}=0.99$, $b_{77}=b_{ii}=1$, $i=\overline{16,19}$ та $i=\overline{21,100}$. Початкову точку x_0 покладемо рівною $x_0=\left(0,0,...,0\right)^T$, а початковий радіус $r_0=50$. Якщо матриця A має ранг 100, то функція має єдину точку мінімуму $x^*=\left(1,1,...,1\right)^T$, причому $f^*=0$.

Параметр ε визначає точність, з якою буде знаходитись ε-розв'язок по функції, а отже він має різний сенс залежно від вигляду функції. Наприклад, для функції $f_1(x)$ розв'язок являє собою точку, в якій сума квадратів нев'язок менша або рівна ε; для функції $f_2(x)$ – точку, в якій сума модулів нев'язок менша або рівна ε; нарешті для функції $f_3(x)$ – точку, в якій максимальний модуль нев'язки менший або рівний ε.

Скалярний параметр $m\geq1$ в субградієнтному методі з кроком Поляка та його аналог – параметр γ в методі *amsg2p* вводяться для того, щоб взяти до уваги спеціальні класи опуклих функцій. Для набору функцій $f_i(x),\ i=\overline{1,3}$ маємо такі значення цих параметрів: для функції $f_1(x)$ $m=\gamma=2$, для функцій $f_2(x)$ та $f_3(x)$ – $m=\gamma=1$.

Результати обчислювальних експериментів наведені в таблицях 4.6-4.11. Кожна таблиця містить 7 колонок; в 2, 4 та 6 колонках наведені кількості ітерацій, які знадобились методам A, B та *amsg2p* відповідно, щоб знайти наближення x_ε^* до точки мінімуму x^* з точностями, наведеними в колонці 1. Колонки 3, 5 і 7 містять норму різниць знайдених наближень і точки мінімуму. В таблицях 4.6 і 4.7 ітерації та норми відхилень методів при мінімізації функції $f_1(x)$, в таблицях 4.8 і 4.9 – функції $f_2(x)$, а в таблицях 4.10 і 4.11 – функції $f_3(x)$. Нарешті дані в таблицях 4.6, 4.8 та 4.10 відповідають заповненню матриці A за допомогою генератора випадкових чисел з відрізку [0,3], а дані в таблицях 4.6, 4.8 та 4.10 – з відрізку [3,10].

Таблиця 4.6

Кількість ітерацій та відхилення від оптимального розв'язку методів A, B та *amsg2p* при мінімізації функції $f_1(x) = \|Ax - b\|^2$, $A \in [0,3]^{500\times100}$

ε_f	PolyakA		PolyakB		amsg2p	
	itn	$\|x_\varepsilon^* - x^*\|$	*itn*	$\|x_\varepsilon^* - x^*\|$	*itn*	$\|x_\varepsilon^* - x^*\|$
10^{-2}	810	5.633e – 03	118	7.648e – 03	6	2.982e – 03
10^{-4}	1612	6.031e – 04	178	7.203e – 04	9	4.880e – 04
10^{-6}	2510	6.218e – 05	236	7.405e – 05	12	3.705e – 05
10^{-8}	3452	6.303e – 06	294	7.650e – 06	16	2.785e – 06
10^{-10}	4416	6.345e – 07	354	7.343e – 07	19	4.101e – 07

Таблиця 4.7

Кількість ітерацій та відхилення від оптимального розв'язку методів A, B та amsg2p при мінімізації функції $f_1(x) = \|Ax - b\|^2$, $A \in [3,10]^{500\times100}$

ε_f	PolyakA		PolyakB		amsg2p	
	itn	$\|x_\varepsilon^* - x^*\|$	*itn*	$\|x_\varepsilon^* - x^*\|$	*itn*	$\|x_\varepsilon^* - x^*\|$
10^{-2}	3034	2.437e – 03	140	3.309e – 03	6	1.613e – 03
10^{-4}	5834	2.598e – 04	200	3.184e – 04	10	1.179e – 04
10^{-6}	8940	2.671e – 05	258	3.334e – 05	13	2.088e – 05
10^{-8}	12182	2.705e – 06	318	3.244e – 06	16	1.578e – 06
10^{-10}	15494	2.724e – 07	378	3.172e – 07	18	2.210e – 07

Проаналізуємо результати з використанням функції $f_1(x)$. З таблиць 4.6 та 4.7 видно, що найбільшої кількості ітерацій потребує метод A, причому вона суттєво зростає при збільшенні точності: 810 ітерацій при $\varepsilon_f = 10^{-2}$ та 4416 ітерацій при $\varepsilon_f = 10^{-10}$. В той же час методу B знадобилось значно менше ітерацій: всього 118 та 354 ітерації для тих же точностей. Метод *amsg2p* демонструє рекордні 6 та 19 ітерацій відповідно.

При переході з відрізку [0,3] до [3,10] для заповнення матриці A кількість ітерацій методу A стрімко зростає: 3034 та 15494 ітерації для відрізку 2 при точностях 10^{-2} та 10^{-10}. В той же час методи B та *amsg2p* суттєво не реагують на зміну відрізків: максимальний приріст ітерацій для методу B становить 24, у методу *amsg2p* – 1.

Таблиця 4.8

Кількість ітерацій та відхилення від оптимального розв'язку методів A, B та amsg2p при мінімізації функції $f_2(x)=\sum_{i=1}^{p}\left|(a_i,x)-b_i\right|,\ A\in[0,3]^{500\times100}$

ε_f	PolyakA		PolyakB		amsg2p	
	itn	$\left\|x_\varepsilon^*-x^*\right\|$	*itn*	$\left\|x_\varepsilon^*-x^*\right\|$	*itn*	$\left\|x_\varepsilon^*-x^*\right\|$
10^{-2}	51	2.960e – 05	84	3.086e – 05	17	2.400e – 05
10^{-4}	81	4.977e – 07	128	2.708e – 07	24	2.433e – 07
10^{-6}	132	3.586e – 09	180	4.317e – 09	31	1.919e – 09
10^{-8}	160	4.021e – 11	217	3.434e – 11	37	2.369e – 11
10^{-10}	191	2.990e – 13	255	4.599e – 13	43	3.015e – 13

Таблиця 4.9

Кількість ітерацій та відхилення від оптимального розв'язку методів A, B та *amsg2p* при мінімізації функції $f_2(x)=\sum_{i=1}^{p}\left|(a_i,x)-b_i\right|,\ A\in[3,10]^{500\times100}$

ε_f	PolyakA		PolyakB		amsg2p	
	itn	$\left\|x_\varepsilon^*-x^*\right\|$	*itn*	$\left\|x_\varepsilon^*-x^*\right\|$	*itn*	$\left\|x_\varepsilon^*-x^*\right\|$
10^{-2}	62	1.276e – 05	90	1.921e – 05	17	1.267e – 05
10^{-4}	94	1.485e – 07	126	1.890e – 07	24	1.260e – 07
10^{-6}	146	2.110e – 09	168	1.757e – 09	31	5.875e – 10
10^{-8}	248	2.199e – 11	223	2.012e – 11	37	7.946e – 12
10^{-10}	521	1.099e – 13	604	1.962e – 13	43	1.094e – 13

Проаналізуємо тепер дані таблиць 4.8 та 4.9 для випадку функції $f_2(x)$. Ситуація дещо змінилась: метод B демонструє трохи більшу кількість ітерацій, ніж метод A для обох відрізків: 255 ітерацій методу B проти 191 ітерації методу

А при точності 10^{-10} та відрізку 1. Для відрізку 2 і тієї ж точності маємо 604 ітерації методу B проти 521 ітерації методу A. Метод *amsg2p* досі демонструє найменшу з усіх трьох методів кількість ітерацій та не реагує на зміну відрізків.

Таблиця 4.10

Кількість ітерацій та відхилення від оптимального розв'язку методів A, B та *amsg2p* при мінімізації функції $f_3(x) = \max\limits_{i=1,p} \left| (a_i, x) - b_i \right|,\ A \in [0,3]^{500\times100}$

ε_f	PolyakA		PolyakB		amsg2p	
	itn	$\left\| x_\varepsilon^* - x^* \right\|$	*itn*	$\left\| x_\varepsilon^* - x^* \right\|$	*itn*	$\left\| x_\varepsilon^* - x^* \right\|$
10^{-2}	665	5.332e – 03	1103	5.396e –03	178	4.442e – 03
10^{-4}	1430	5.587e – 05	2227	5.964e – 05	315	4.514e – 05
10^{-6}	2262	5.762e – 07	3430	5.951e – 07	455	4.723e – 07
10^{-8}	3120	5.748e – 09	4645	5.956e – 09	595	5.035e – 09
10^{-10}	4003	5.727e – 11	5858	5.902e – 11	752	4.579e – 11

Таблиця 4.11

Кількість ітерацій та відхилення від оптимального розв'язку методів A, B та amsg2p при мінімізації функції $f_3(x) = \max\limits_{i=1,p} \left| (a_i, x) - b_i \right|,\ A \in [3,10]^{500\times100}$

ε_f	PolyakA		PolyakB		amsg2p	
	itn	$\left\| x_\varepsilon^* - x^* \right\|$	*itn*	$\left\| x_\varepsilon^* - x^* \right\|$	*itn*	$\left\| x_\varepsilon^* - x^* \right\|$
10^{-2}	1707	2.288e –03	3815	2.332e – 03	186	2.014e – 03
10^{-4}	3981	2.378e – 05	7050	2.372e – 05	328	2.007e – 05
10^{-6}	6446	2.533e – 07	10332	2.499e – 07	471	1.783e – 07
10^{-8}	9138	2.560e – 09	13711	2.549e – 09	614	2.237e – 09
10^{-10}	11876	2.644e – 11	17109	2.512e – 11	758	1.696e – 11

Результати у випадку функції $f_3(x)$, які наведені в таблицях 4.10 та 4.11, є дещо «погіршеним» варіантом результатів з функцією $f_2(x)$: кількості ітерацій суттєво зросли, однак співвідношення між ними в різних методів збереглись. Наприклад, методи A, B та *amsg2p* при точності 10^{-10} та відрізку 1 потребують відповідно 4003, 5858 та 752 ітерацій; при переході до відрізку 2 ці значення змінюються до 11876, 17109 та 758 ітерацій відповідно. Колонки 7 таблиць 4.10 та 4.11 показують, що метод *amsg2p* залишається стійким до зміни відрізків з максимальним приростом ітерацій рівним 19.

Отже, субградієнтний метод з кроком Поляка з перетворенням простору не завжди потребує меншої кількості ітерацій, ніж той же метод без перетворення. Зростання кількості ітерацій методу B для функцій $f_1(x)$ та $f_2(x)$ пояснюється вибором матриці перетворення простору B: оскільки всі три функції $f_i(x), i=\overline{1,3}$ визначають розв'язок по-різному, один і той же вигляд матриці B може як суттєво прискорювати метод B для однієї функції, так і значно вповільнювати його для іншої.

Для всіх трьох функцій $f_i(x), i=\overline{1,3}$ кількість ітерацій методу *amsg2p* є суттєво меншою, ніж кількості ітерацій методів A та B. Тому його можна успішно використовувати для мінімізації яружних опуклих функцій, а матрично-векторні операції роблять його перспективним у системах паралельних та розподілених обчислень. Однак, метод *amsg2p* має один вагомий недолік: питання про його ефективність та швидкість збіжності досі залишається відкритим. Це означає, що для кожної конкретної функції метод може як працювати дуже ефективно, так і демонструвати протилежний результат.

4.6 Висновки до четвертого розділу

1. Розглянуто узагальнений метод еліпсоїдів та його загальна схема. Наведено опис двох опуклих задач, які можна розв'язувати з його використанням.

2. Розглянуто задачу визначення параметрів лінійної регресії у формі задачі мінімізації негладкої функції, що являє собою L_p-норму нев'язки системи лінійних рівнянь. Наведено загальну схему алгоритма методу еліпсоїдів для мінімізації цієї функції при довільному значенні параметра $p \geq 1$. Проведено порівняння результатів роботи запропонованого алгоритма з результатами класичних методів, які відповідають значенням параметра $p = 1, 2, \infty$, для задачі апроксимації спостережень, що містять аномалії, лінійною функцією. Показано, що перевагу варто віддавати методу найменших модулів, адже він ігнорує групи аномалій та адекватно відновлює лінійну функцію. Продемонстровано, що при використанні великих значень параметра $p \geq 1$ розв'язок наближається до розв'язку, отриманого мінімаксним методом.

3. Розглянуто задачу знаходження L_p-розв'язку системи лінійних рівнянь з двосторонніми обмеженнями на змінні. Запропоновано алгоритм на основі метода еліпсоїдів Юдіна-Неміровського для розв'язання цієї задачі.

4. Розглянуто задачу знаходження розв'язку сумісної системи лінійних рівнянь як задачу мінімізації опуклої функції у трьох різних формах. Проведено обчислювальні експерименти з використанням субградієнтного методу з кроком Поляка у вихідному та перетвореному просторах змінних та скалярним параметром $m > 1$ та методу amsg2p для знаходження точки мінімуму цих функцій. Показано, що найменшої кількості ітерацій потребує метод amsg2p, однак для цього методу немає обґрунтування збіжності та оцінок швидкості збіжності. Продемонстровано, що правильне регулювання параметра $m > 1$ та матриці перетворення простору дає змогу суттєво прискорити субградієнтний метод з кроком Поляка як порівняти з його класичним варіантом.

РОЗДІЛ 5. ПРОГРАМНА РЕАЛІЗАЦІЯ СУБГРАДІЄНТНОГО МЕТОДУ З КРОКОМ ПОЛЯКА МОВОЮ С++

У п’ятому розділі наведені програмні реалізації субградієнтного методу з кроком Поляка у початковому та перетвореному просторах змінних та параметром $m \geq 1$ мовою С++. У підрозділі 5.1 наведено програмний код і детальний опис функцій PolyakA та PolyakB, які реалізують вказаний метод. Процедури для обчислення значень функцій та їхніх субградієнтів наведено в підрозділі 5.2. Детальний опис допоміжних функцій мовою С++, які необхідні для запуску тестових прикладів з розділів 2, 3 та 4, наведено в підрозділі 5.3.

5.1 Функції PolyakA та PolyakB

Ці функції складаються з 24 та 28 рядків відповідно та виконують основні обчислення в циклі for за індексом **itn**. Вхідні та вихідні параметри обох функцій повністю збігаються окрім матриці перетворення простору B, яка є додатковим вхідним параметром функції *polyakB*. Перелік вхідних та вихідних параметрів функцій *PolyakA* та *PolyakB* наведено нижче.

Вхідні параметри: ім’я функції для обчислення значення функції $f(x)$ та її субградієнта $g_f(x)$ в точці x (**calcfg**), початкова точка **x0**, оптимальне значення функції **fstar**, скалярний параметр **m** (зміщення по опуклості), точність **epsf** та максимальна кількість ітерацій **maxitn**. Функція *polyakB* додатково приймає вказівник на матрицю перетворення простору **B**, кількість її рядків (**row**) та стовпців (**col**).

Вихідні параметри: пара векторів. Перший вектор є розв’язком (x_ε^*) – точкою мінімуму функції $f(x)$, знайденою з точністю **epsf**. Другий вектор є набором з трьох чисел, де перше число – значення функції $f(x)$ в точці x_ε^*,

друге – виконана кількість ітерацій, третє – код зупинки (**info**), який набуває значення 0, якщо знайдено розв'язок із заданою точністю та 4, якщо виконана максимальна кількість ітерацій.

Код функції *polyakA*:

```
// функція polyakA
pair<vecdub,vecdub> polyakA(fgpair(*calcfg)(vecdub), vecdub x0, double
fstar, double m, double epsf, int maxitn)
{
      int itn = 0;           // кількість ітерацій
      int info = 4;         // результат роботи програми
      vecdub x = x0;         // вектор-розв'язок x
      auto fg = calcfg(x);  // обчислюємо f (=fg.first) та g (=fg.second)
      double dg = norm(fg.second); // обчислюємо норму субградієнта
      double hs = 0;         // крок Поляка
      for (itn; itn < maxitn; ++itn)
      {
            if (fg.first - fstar < epsf) { info = 0; break; }
            mult(fg.second, 1/dg);              // g1
            hs = m*(fg.first - fstar)/dg;       // hs
            mult(fg.second,hs);
            sub(x,fg.second);                   // перераховуємо x
            fg = calcfg(x);
            dg = norm(fg.second);
      }
      vecdub res(3,0); // містить f, itn, info
      res[0] = fg.first; res[1] = itn; res[2] = info;
      return make_pair(x,res);
}
```

Код функції *polyakB*:

```
// polyakB
pair<vecdub, vecdub> polyakB(fgpair(*calcfg)(vecdub), double* B, int
row, int col, vecdub x0, double fstar, double m, double epsf, int
maxitn)
{
```

```
      int itn = 0;          // кількість ітерацій
      int info = 4;        // результат роботи програми
      vecdub x = x0;        // розв'язок x
      auto fg = calcfg(x); // обчислюємо f (=fg.first) та g (=fg.second)
      double hs = 0, dg = 0;
      vecdub g1, temp;
      double dg1 = 0;
      for (itn; itn < maxitn; ++itn)
      {
            if (fg.first - fstar < epsf) { info = 0; break; }
            g1 = multMT(&B[0],row,col,fg.second);    // g1
            dg1 = norm(g1);           // обчислюємо норму субградієнта g1
            mult(g1, 1/dg1);          // нормуємо субградієнт g1
            hs = m*(fg.first - fstar)/dg1; // обчислюємо крок Поляка
            temp = multM(&B[0],row,col,g1);
            mult(temp,hs);
            sub(x,temp);              // перераховуємо x
            fg = calcfg(x);
            dg = norm(fg.second);
      }
      vecdub res(3, 0);              // містить f, itn, info
      res[0] = fg.first; res[1] = itn; res[2] = info;
      return make_pair(x,res);
}
```

5.2 Процедури для обчислення значення функції та її субградієнта

Одним з вхідних параметрів функцій *polyakA* та *polyakB* є параметр **calcfg**, замість якого необхідно підставляти ім'я функції, яка обчислює значення функції $f(x)$ та її субградієнта $g_f(x)$ в точці x. Наведемо далі програмний код таких функцій, що використовувались в тестових прикладах розділів 2, 3 і 4.

Функція **squad** ($f_1(x_1, x_2) = x_1^2 + tx_2^2$, $t >> 1$, приклади 2.1 та 3.2, розділи 2 і 3):

```
fgpair squad(vecdub x)
```

```
{
      double f = x[0] * x[0] + t*x[1] * x[1];
      vecdub g(2);
      g[0] = 2*x[0];
      g[1] = 2*t*x[1];
      return make_pair(f,g);
}
```

Функція **four1** ($f_2(x_1, x_2) = (x_1 + 1.001x_2)^4 + (1.001x_1 + x_2)^4$, приклад 2.2, розділ 2):

```
fgpair four1(vecdub x)
{
      double f = pow(x[0] + 1.001*x[1], 4) + pow(1.001*x[0] + x[1], 4);
      vecdub g(2);
      g[0] = 4*pow(x[0]+1.001*x[1],3) + 4*1.001*pow(1.001*x[0]+x[1],3);
      g[1] = 4*1.001*pow(x[0]+1.001*x[1],3) + 4*pow(1.001*x[0]+x[1],3);
      return make_pair(f,g);
}
```

Функція **four2** ($f_3(x_1, x_2) = x_1^4 + 10000x_2^4$, приклад 2.2, розділ 2):

```
fgpair four2(vecdub x)
{
      double f = x[0]*x[0]*x[0]*x[0] + 10000*x[1]*x[1]*x[1]*x[1];
      vecdub g(2);
      g[0] = 4*x[0]*x[0]*x[0];
      g[1] = 4*10000*x[1]*x[1]*x[1];
      return make_pair(f,g);
}
```

Функція **psum** ($f_4(x) = \sum_{i=1}^{k} \left| \sum_{j=1}^{n} a_{ij} x_j - b_i \right|^p$, $p \geq 1$, приклад 2.3, розділ 2):

```
fgpair psum(vecdub x)
{
      vecdub y = multM(&A[0][0], M, N, x);        // A*x
      sub(y,b);                                   // A*x - b (= y)
      double f = 0.0;
      vecdub temp;
      for (double el : y)
```

```
      {
            f += exp(log(abs(el))*p);              // обчислюємо f
            temp.push_back(sgn(el)*exp(log(abs(el))*max(1e-10, p-1)));
      }
      mult(temp,p);                                  // temp * p
      vecdub g = multMT(&A[0][0],M,N,temp);    // A' * temp
      return make_pair(f,g);
}
```

Функція **sabs** ($f_5(x_1,x_2)=|x_1|+t|x_2|,\ t>1$, приклад 3.1, розділ 3):

```
fgpair sabs(vecdub x)
{
      double f = abs(x[0]) + t*abs(x[1]);
      vecdub g(2);
      g[0] = sgn(x[0]);
      g[1] = t*sgn(x[1]);
      return make_pair(f,g);
}
```

Функція **maxx** ($f_6(x_1,x_2)=\max\left\{x_1^2+(2x_2-2)^2-3,\ x_1^2+(x_2+1)^2\right\}$, приклад 3.1, розділ 3):

```
fgpair maxx(vecdub x)
{
      double f = x[0] * x[0] + (2*x[1]-2) * (2*x[1]-2) - 3;
      double f2 = x[0] * x[0] + (x[1]+1) * (x[1]+1);
      vecdub g(2);
      g[0] = 2*x[0];
      g[1] = 8*(x[1]-1);
      if (f2 > f) { f = f2; g[1] = 2*(x[1] + 1); }
      return make_pair(f,g);
}
```

Функція **func1** ($f_7(x)=\|Ax-b\|^2$, підрозділ 4.5, розділ 4):

```
fgpair func1(vecdub x)
{
      vecdub y = multM(&A[0][0], M, N, x);     // A*x
      sub(y,b);                                // A*x - b (= y)
      double f = scalprod(y,y);                // (y,y) = ||Ax-b||^2
      mult(y,2);                               // 2*y
```

```
      vecdub g = multMT(&A[0][0], M, N, y);   // 2*A'*y
      return make_pair(f,g);
}
```

Функція **func2** ($f_8(x) = \sum_{i=1}^{p} \left| (a_i, x) - b_i \right|$, підрозділ 4.5, розділ 4):

```
fgpair func2(vecdub x)
{
      vecdub y = multM(&A[0][0], M, N, x);   // A*x
      sub(y,b);                              // A*x - b (= y)
      double f = 0.0;
      vecdub g(N,0);

      // обчислюємо f та g
      for (int i = 0; i < y.size(); ++i)
      {
            if (y[i] >= 0)
            {
                  f += y[i];
                  for (int j = 0; j < N; ++j) g[j] += A[i][j];
            }
            else
            {
                  f -= y[i];
                  for (int j = 0; j < N; ++j) g[j] -= A[i][j];
            }
      }
      return make_pair(f,g);
}
```

Функція **func3** ($f_9(x) = \max_{i=1,p} \left| (a_i, x) - b_i \right|$, підрозділ 4.5, розділ 4):

```
fgpair func3(vecdub x)
{
      vecdub y = multM(&A[0][0], M, N, x);    // A*x
      sub(y, b);                              // A*x - b (= y)
      double f = abs(y[0]);
      vecdub g(N,0);
      int ind = 0;     // індекс максимального елемента
```

```
	// обчислюємо f та знаходимо ind
	for (int i = 1; i < y.size(); ++i)
	{
		if (abs(y[i]) > f)
		{
			f = abs(y[i]);
			ind = i;
		}
	}

	// обчислюємо g
	if (y[ind] >= 0)
		for (int j = 0; j < N; ++j) g[j] = A[ind][j];
	else
		for (int j = 0; j < N; ++j) g[j] = -A[ind][j];

	return make_pair(f,g);
}
```

5.3 Запуск тестових прикладів

Для запуску тестових прикладів з розділів 2, 3 і 4 реалізовано три функції: **run1**, **run2**, **run3**. На початку кожної з них ініціалізуються вхідні параметри, а наприкінці виконується запуск функцій *polyakA* та *polyakB* і виведення результатів їхньої роботи на екран. Кожна з цих функцій здійснює запуск власного набору прикладів. Якщо таких прикладів декілька – вхідному параметра ***calcfg** необхідно присвоїти ім’я функції, яку необхідно мінімізувати.

Функція **run1** демонструє результати роботи програм *polyakA* та *polyakB* для функцій **squad**, **sabs**, **maxx**, **four1** та **four2**. Її програмний код наведено нижче.

```
void run1()
{
```

```
    // squad/sabs/maxx/four1/four2
    fgpair(*calcfg)(vecdub) = squad;
    vecdub x0 = {1,1};
    double fstar = 0.0;
    int m = 1;
    double epsf = 1e-20;

    int maxitn = 100000;
    t = 5;

    double alpha = 1;
    double B[2][2] = {{1,0},{0,1/alpha}};

    // інтерфейс запуску
    cout << "polyakA" << endl;
    auto R = polyakA(calcfg, x0, fstar, m, epsf, maxitn);
    cout << "itn = "<< R.second[1]<<", "<<"f = "<<R.second[0] << endl;
    cout << "x = (" << R.first[0] << ", " << R.first[1] << ")" << endl;
    // cout << "info = " << R.second[2] << endl;
    cout << endl;

    cout << "polyakB" << endl;
    auto S = polyakB(calcfg,&B[0][0],2,2,x0,fstar,m,epsf,maxitn);
    cout << "itn = "<<S.second[1]<<", "<<"f = "<<S.second[0] << endl;
    cout << "x = (" << S.first[0] << ", " << S.first[1] << ")" << endl;
    // cout << "info = " << S.second[2] << endl;
    cout << endl << "MAXITN = " << maxitn << endl;
}
```

Функція **run2** виводить результати роботи програми *polyakA* для функції **psum** прикладу 2.3 розділу 2. Її програмний код наведено нижче.

```
void run2()
{
    // заповнюємо A та b псевдовипадковими числами з інтервалу [0,3]
    fillAb('P', &A[0][0], M, N, b, 0, 3);
```

```
	double m = 1.4;
	p = 1.4;
	double epsf = 1e-4;

	fgpair(*calcfg)(vecdub) = psum;
	vecdub x0(N,0);
	double fstar = 0.0;
	int maxitn = 10000;

	// інтерфейс запуску
	cout << "polyakA" << endl;
	auto R = polyakA(calcfg, x0, fstar, m, epsf, maxitn);
	cout << "itn = "<<R.second[1]<<", "<<"f = "<<R.second[0] << endl;
	// cout << "info = " << R.second[2] << endl;
	cout << endl << "MAXITN = " << maxitn << endl;
}
```

Функція **run3** демонструє результати роботи програм *polyakA* та *polyakB* для набору функцій **func1**, **func2** і **func3** підрозділу 4.5 розділу 4. Її програмний код наведено нижче.

```
void run3()
{
	// заповнюємо A та b псевдовипадковими числами з інтервалу [0,3]
	fillAb('F', &A[0][0], M, N, b, 0, 3);

	// func1/func2/func3
	fgpair(*calcfg)(vecdub) = func2;
	double epsf = 1e-2;
	// double m = 2; // для func1
	double m = 1; // для func2 та func3

	vecdub x0(N,0);
	vecdub xstar(N,1);
	double fstar = 0.0;
	int maxitn = 10000;
```

```
    // підготовка матриці B
    double B[N][N];
    // недіагональні елементи прирівнюємо до нуля
    for (int i = 0; i < N; ++i)
        for (int j = 0; j < N; ++j)
            if (i != j) B[i][j] = 0.0;

    // визначаємо діагональні елементи
    B[0][0] = 0.64; B[1][1] = 0.64; B[2][2] = 0.64; B[3][3] = 0.67;
    B[4][4] = 0.63; B[5][5] = 0.99; B[6][6] = 1.0; B[7][7] = 0.7;
    B[8][8] = 0.7; B[9][9] = 0.7; B[10][10] = 0.7; B[11][11] = 0.7;
    B[12][12] = 0.7; B[13][13] = 0.7; B[14][14] = 0.99;
    B[15][15] = 1.0; B[16][16] = 1.0; B[17][17] = 1.0; B[18][18] = 1.0;
    B[19][19] = 0.99;
    for (int i = 20; i < N; ++i) B[i][i] = 1.0;

    // інтерфейс запуску
    cout << "polyakA" << endl;
    auto R = polyakA(calcfg, x0, fstar, m, epsf, maxitn);
    sub(R.first,xstar);
    cout << "itn = " << R.second[1] << ", ";
    cout << "||x-x*|| = " << norm(R.first) << endl;
    // cout << "info = " << R.second[2] << endl;

    cout << "polyakB" << endl;
    auto S = polyakB(calcfg, &B[0][0], N, N, x0, fstar, m, epsf,
maxitn);
    sub(S.first, xstar);
    cout << "itn = " << S.second[1] << ", ";
    cout << "||x-x*|| = " << norm(S.first) << endl;
    // cout << "info = " << S.second[2] << endl;
    cout << endl << "MAXITN = " << maxitn << endl;
}
```

Варто зазначити, що кількість ітерацій для одного й того ж прикладу з однаковими значеннями вхідних параметрів при запуску програм мовами Octave та C++ можуть трохи відрізнятись. Це можна пояснити так. По-перше,

тип double (числа з плаваючою точкою) мови C++ має обмеження на 15 значущих цифр, тоді як аналогічний тип мови Octave містить понад 15 значущих цифр. Це призводить до накопичення похибки обчислень та зміни поведінки методу. В такому випадку Octave- та C++-програмам необхідно виконати різну кількість ітерацій для знаходження розв'язку з однією й тією ж заданою точністю. По-друге, генерування унікальних псевдовипадкових чисел також призводить до різниці в кількостях ітерацій, незважаючи на те, що в обох програмах використовується вихор Мерсенна. Тим не менш, C++-програма потребує меншої кількості часу, незважаючи на більшу кількість ітерацій. Така поведінка пояснюється кращою оптимізацією середовища Visual Studio, яке використовувалось для розробки та запуску C++-програм, як порівняти з середовищем GNU Octave.

5.4 Допоміжні функції та бібліотеки

Для забезпечення роботи функцій *polyakA* та *polyakB* необхідний певний набір бібліотек, глобальних змінних та допоміжних функцій. Нижче ми наведемо їхній програмний код та опис.

В програмі використовуються такі бібліотеки:

```
#include <iostream>
#include <vector>
#include <random>
#include <algorithm>
```

Для зручності були введені такі імена наявних типів:

```
typedef vector<double> vecdub;
typedef pair<double, vecdub> fgpair;
```

Для доступу до даних з довільної функції було створено декілька глобальних змінних:

```
const int M = 500;
const int N = 100;
double A[M][N];
vecdub b(M,0);
```

```
int t;
double p;
```

Допоміжні функції: **out(x)** – виводить зміст вектора **x**; **sgn(x)** – математична функція $sign(x)$; **norm(x)** – обчислює норму вектора **x**; **mult(x,K)** – домножує вектор **x** на число **K**; **scalprod(x,y)** – обчислює скалярний добуток векторів **x** та **y**; **sub(x,y)** – обчислює різницю векторів **x** та **y**; **multM(M,row,col,x)** – обчислює добуток матриці **M** розмірності **row**×**col** та вектора **x**; **multMT(M,row,col,x)** – обчислює добуток транспонованої матриці **M** розмірності **row**×**col** та вектора **x**.

Розглянемо докладніше функцію **fillAb(mode,A,row,col,b,l1,l2)**, яка призначена для заповнення матриці **A** розмірності **row**×**col** псевдовипадковими числами з відрізку [**l1**,**l2**] та ініціалізації вектора **b** у прикладі 2.3 розділу 2 та прикладі підрозділу 4.5 розділу 4. Параметр **mode** встановлює режим заповнення матриці **A**: ‘F’ – матрицю A заповнити псевдовипадковими числами повністю, ‘P’ – заповнити всі рядки, окрім перших двох. Для генерування псевдовипадкових чисел функція використовує так званий вихор Мерсенна. Цей генератор був розроблений у 1997 році і базується на властивостях простих чисел Мерсенна та забезпечує швидке генерування високоякісних за критерієм випадковості псевдовипадкових чисел.

Код функції **fillAb** та інших допоміжних функцій наведений нижче.

```
void fillAb(char mode,double* A,int row,int col,vecdub &b,int l1,int l2)
{
      random_device rd;       // потрібен для генерації випадкових чисел
      mt19937 gen(rd());      //вихор Мерсенна
      // генерує дійсне псевдовипадкове число з інтервалу [0,1]
      uniform_real_distribution<> dis(0,1);
      int lim = 0;

      if (mode == 'P')
      {
            // заповнюємо нулями перші два рядки
            for (int r = 0; r < 2; ++r)
                  for (int c = 0; c < col; ++c)
```

```
                    A[r*col + c] = 0;

            // два діагональні елементи рівні 100
            A[0] = 100; A[col + 1] = 100;
            b[0] = 100; b[1] = 100;
            lim = 2; // продовжуємо з 3го рядка
      }

      // якщо lim = 0 - заповнюємо матрицю ПВЧ повністю
      double sum = 0.0;
      for (int r = lim; r < row; ++r)
      {
            for (int c = 0; c < col; ++c)
            {
                  // генеруємо псевдовипадкове число з інтервалу [l1,l2]
                  A[r*col + c] = (l2 - l1)*dis(gen) + l1;
                  sum += A[r*col + c];
            }
            b[r] = sum;
            sum = 0;
      }
}

// ДОПОМІЖНІ ФУНКЦІЇ
void out(vector<double> x)
{
      for (double el : x) cout << el << " ";
      cout << endl;
}

int sgn(double x)
{
      if (x > 0) return 1;
      if (x < 0) return -1;
      return 0;
}

double norm(vecdub v)
```

```
{
	double sum = 0;
	for (int i = 0; i < v.size(); ++i)
	{
		sum += v[i] * v[i];
	}
	return sqrt(sum);
}

void mult(vecdub &x, const double K)
{
	for (int i = 0; i < x.size(); ++i) x[i] *= K;
}

double scalprod(vecdub x, vecdub y)
{
	if (x.size() != y.size())
	{
		cout << "Scalar product is incorrect! Different dimensions";
		cout << endl;
		return 0;
	}
	double sum = 0;
	for (int i = 0; i < x.size(); ++i) sum += x[i] * y[i];
	return sum;
}

void sub(vecdub &x, const vecdub y)
{
	for (int i = 0; i < x.size(); ++i) x[i] -= y[i];
}

vecdub multM(double* M, int row, int col, vecdub x)
{
	vecdub res(row,0);
	// перевірка узгодженості розмірностей матриці та вектора
	if (x.size() != col)
	{
```

```
            cerr << "Matrix-vector multiplication is incorrect!;
            cerr << Wrong vector size" << endl;
            return res;
      }
      double sum = 0.0;

      for (int r = 0; r < row; ++r)
      {
            for (int c = 0; c < col; ++c)
            {
                  sum += M[r*col + c] * x[c];
            }
            res[r] = sum;
            sum = 0;
      }
      return res;
}

vecdub multMT(double* M, int row, int col, vecdub x)
{
      vecdub res(col, 0);
      // перевірка узгодженості розмірностей матриці та вектора
      if (x.size() != row)
      {
            cerr << "Matrix-vector multiplication is incorrect!;
            cerr << Wrong vector size" << endl;
            return res;
      }
      double sum = 0.0;

      for (int c = 0; c < col; ++c)
      {
            for (int r = 0; r < row; ++r)
            {
                  sum += M[r*col + c] * x[r];
            }
            res[c] = sum;
            sum = 0;
```

```
    }
    return res;
}
```

5.5 Висновки до п’ятого розділу

1. Наведено вихідний код програмних модулів PolyakA та PolyakB, які реалізують субградієнтний метод з кроком Поляка у вихідному та перетвореному просторах змінних та параметром $m \geq 1$ мовою C++. Описано вхідні та вихідні параметри процедур, а також особливості їхнього запуску.

2. Наведено вихідний код процедур для обчислення значень функцій та їхніх субградієнтів, які використовуються в тестових прикладах у розділах 2-4.

3. Наведено детальний опис та вихідний код усіх допоміжних процедур та бібліотек, необхідних для коректного запуску та роботи тестових прикладів з розділів 2-4.

ЗАГАЛЬНІ ВИСНОВКИ

У дисертаційній роботі проведено дослідження субградієнтного методу з кроком Поляка у вихідному та перетвореному просторах змінних, а також його застосування для мінімізації гладких та негладких яружних опуклих функцій. Отримано наукові результати в розвитку субградієнтних методів, а також їхньому застосуванні до розв'язання прикладних задач.

Основні результати дисертаційної роботи.

1. Обґрунтовано монотонне зменшення відстані до точки мінімуму та швидкість збіжності класичного субградієнтного методу з кроком Поляка та скалярним параметром $m>1$, яка дорівнює $O\left(1/\sqrt{k}\right)$ у випадку довільної опуклої функції та швидкості геометричної прогресії для опуклої функції з гострим мінімумом. Це означає, що складність цього методу не перевищує складності класичного субградієнтного спуску.
2. Обґрунтовано монотонне зменшення відстані до точки мінімуму та швидкість збіжності субградієнтного методу з кроком Поляка у перетвореному просторі змінних та його модифікації скалярним параметром $m>1$, яка дорівнює $O\left(1/\sqrt{k}\right)$ у випадку довільної опуклої функції та швидкості геометричної прогресії для опуклої функції з гострим мінімумом.
3. Проведено обчислювальні експерименти з мінімізації гладких та негладких яружних опуклих функцій з використанням субградієнтного методу з кроком Поляка у вихідному та перетвореному просторах змінних та його модифікацій скалярним параметром $m>1$. Показано, що використання параметра m та операції перетворення простору дає

змогу суттєво прискорити класичний субградієнтний метод з кроком Поляка при мінімізації як довільних опуклих функцій, так і опуклих функцій спеціального типу. Зроблено висновки щодо практичного застосування та подальшого розвитку розглянутих методів.

4. Розроблено програмну реалізацію субградієнтного методу з кроком Поляка у вихідному та перетвореному просторах змінних і скалярним параметром $m \geq 1$ мовою C++. Реалізацію можна використовувати для проведення додаткових досліджень та експериментів, практичного використання або швидкої реалізації будь-якою «C-подібною» мовою програмування.
5. Побудовано алгоритм методу еліпсоїдів для розв'язання задачі визначення параметрів лінійної регресії при довільному значенні параметра $p \geq 1$. Досліджено стійкість алгоритма до появи аномальних спостережень при $p = 1, 2, \infty$ та великих значеннях параметра p. Показано, що перевагу варто віддавати значенню $p = 1$, адже при ньому метод найбільш адекватно відновлює лінійну функцію, відкидаючи аномальні спостереження.
6. Запропоновано алгоритм на основі методу еліпсоїдів Юдіна-Неміровського для знаходження L_p-розв'язку системи лінійних рівнянь при двосторонніх обмеженнях на змінні. Алгоритм є стійким при розв'язанні погано обумовлених систем лінійних рівнянь.
7. Проведено обчислювальні експерименти з використанням субградієнтного методу з кроком Поляка у вихідному та перетвореному просторах змінних та скалярним параметром $m > 1$ і методу amsg2p для розв'язання сумісної системи лінійних рівнянь. Ця задача записується як задача мінімізації опуклої функції у трьох різних формах. Показано, що найменшої кількості ітерацій потребує метод amsg2p, однак для нього немає обґрунтування збіжності та оцінок швидкості збіжності. Продемонстровано, що використання різних значень параметра $m > 1$

та операції перетворення простору дає змогу суттєво прискорити класичний субградієнтний метод з кроком Поляка для мінімізації яружних функцій.

8. Практичне значення роботи полягає в тому, що отримані результати можуть бути ефективно використані для розв’язання задач обробки великих об’ємів даних, розробки методів машинного навчання, технологій штучного інтелекту тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Андреева, Елена, и Валентина Цирулева. 2001. *Вариационное исчисление и методы оптимизации*. Тверь: Тверской государственный университет.

Аоки, Масанао. 1977. *Введение в методы оптимизации*. М.: Наука.

Базара, Мохтар, и К. Шетти. 1982. *Нелинейное программирование. Теория и алгоритмы*. М.: Мир.

Бертсекас, Дмитрий. 1987. *Условная оптимизация и методы множителей Лагранжа*. М.: Радио и связь.

Бирюков, Сергей. 2003. *Оптимизация. Элементы теории и численные методы*. М.: МЗ-Пресс.

Брэгман, Лев. 1967. «Релаксационный метод нахождения общей точки выпуклых множеств и его применение для решения задач выпуклого программирования». *ЖВМ и МФ* 7(3):200-217.

Васильев, Федор. 2011а. *Методы оптимизации. Часть I. Конечномерные задачи оптимизации. Принцип максимума. Динамическое программирование*. М.: Изд-во МЦМНО. 620 с.

Васильев, Федор. 2011б. *Методы оптимизации. Часть II. Оптимизация в функциональных пространствах. Регуляризация. Аппроксимация*. М.: Изд-во МЦМНО. 433 с.

Ведель, Яна, та Володимир Семенов. 2020. «Адаптивні алгоритми для задач про рівновагу в просторах Адамара». *Доповіді Національної академії наук України* 8:26-34.

Войтова, Тетяна, Сергій Денисов, та Володимир Семенов. 2012. «Альтернуючий проксимальний алгоритм для задачі дворівневої опуклої мінімізації». *Доповіді НАН України* 2:56-62.

Волосов, Виктор. 1996. «К построению параметрических семейств эллипсоидальных оценок и их оптимизация в задачах нестохастической идентификации параметров и состояния многомерных дискретных объектов управления». *Проблемы управления и информатики* 4:37-53.

Воронцова, Евгения, Александр Гасников, и Эдуард Горбунов. 2019. «Ускоренные спуски по случайному направлению с неевклидовой прокс-структурой». *Автоматика и Телемеханика* 4:126-143.

Галеев, Эльфат. 2006. *Оптимизация. Теория, примеры, задачи*. М.: URSS.

Гасников, Александр, Павел Двуреченский, и Юрий Нестеров. 2016а. «Стохастические градиентные методы с неточным оракулом». *Труды МФТИ* 8(1):41-91.

Гасников, Александр, Евгения Гасникова, Юрий Нестеров, и Алексей Чернов. 2016б. «Об эффективных численных методах решения задач энтропийно-линейного программирования». *Ж. вычисл. матем. и матем. физ.* 56(4):523-534.

Гасников, Александр, и Юрий Нестеров. 2018. «Универсальный метод для задач стохастической композитной оптимизации». *Ж. вычисл. матем. и матем. физ.* 58(1):52-69.

Гершович, В., и Наум Шор. 1982. «Метод эллипсоидов, его обобщения и приложения». *Кибернетика* (5):61-9.

Гольштейн, Евгений, и Николай Третьяков. 1989. *Модифицированные функции Лагранжа*. М.: Наука.

Гороховик, Валентин. 2007. *Конечномерные задачи оптимизации*. Минск: Издательский центр БГУ.

Евтушенко, Юрий. 1982. *Методы решения экстремальных задач и их применение в системах оптимизации*. М.: Наука.

Еремин, Иван. 1965. «Обобщение релаксационного метода Агмона-Моцкина». *УМН* 20(122):183-7.

Еремин, Иван, и Николай Астафьев. 1976. *Введение в теорию линейного и выпуклого программирования*. М.: Наука.

Еремин, Иван. 2001а. *Двойственность в линейной оптимизации*. Екатеринбург: УрО РАН.

Еремин, Иван. 2001б. «Синтез фейеровских отображений с несовпадающими пространствами их образов». *Докл. РАН* 378(1):11-3.

Еремин, Иван. 2003. *Симметрическая двойственность для лексикографической задачи линейного программирования*. М.: Издательский дом «ИНФРА-М»; Большая Российская энциклопедия.

Еремин, Иван. 2004. «Фейеровские процессы: синтез и рандомизация». *Тр. Ин-та математики и механики УрО РАН* 10(2):58-68.

Еремин, Иван. 2005. *Теория двойственности в линейной оптимизации*. Челябинск: Южно-Уральский госуниверситет.

Ермольев, Юрий. 1966. «Методы решения нелинейных экстремальных задач». *Кибернетика* (4):1-17.

Ермольев, Юрий. 1976. *Методы стохастического программирования*. М.: Наука.

Жадан, Виталий. 2014. *Методы оптимизации. Ч. I. Введение в выпуклый анализ и теорию оптимизации*. М.: МФТИ.

Жадан, Виталий. 2015. *Методы оптимизации. Ч. II. Численные алгоритмы*. М.: МФТИ.

Жадан, Виталий. 2017. *Методы оптимизации. Ч. III. Дополнительные главы: учебное пособие*. М.: МФТИ.

Журбенко, Николай. 1999. «Квазиньютоновские алгоритмы минимизации на основе использования оператора растяжения пространства». *Теория оптимальных решений* 45-50.

Журбенко, Николай. 2012. Субградиентный алгоритм минимизации с преобразованием пространства α(ε)−алгоритм. В *Стохастическое программирование и его приложения*, под редакцией Павла Кнопова и Валерия Зоркальцева, 36-51. Иркутск: Институт систем энергетики им. Л.А. Мелентьева СО РАН.

Журбенко, Николай. 2015. «Алгоритм Поляка на основе агрегатных ε-субградиентов». *Теорія оптимальних рішень* 146-53.

Журбенко, Николай. 2017. «Численная эффективность одной модификации r-алгоритма». *Теорія оптимальних рішень* 33-8.

Журбенко, Николай, и Алексей Лиховид. 2018. «Регуляризация матрицы преобразования r-алгоритма». *Теорія оптимальних рішень* (17):145-51.

Журбенко, Николай, и Алексей Лиховид. 2019. «К численной эффективности одной модификации r-алгоритма». *Компьютерная математика* (1):124-31.

Измаилов, Алексей, и Михаил Солодов. 2003. *Численные методы оптимизации*. М.: Физматлит.

Карманов, Владимир. 1986. *Математическое программирование*. М.: Наука.

Кларк, Френк. 1988. *Оптимизация и негладкий анализ*. М.: Наука.

Козлов, М., Сергей Тарасов, и Леонид Хачиян. 1979. «Полиномиальная разрешимость выпуклого квадратичного программирования». *Докл. АН СССР* 248(5):1049-51.

Лесин, Виктор, и Юрий Лисовец. 2011. *Основы методов оптимизации*. М.: Лань.

Малицкий, Юрий, и Владимир Семенов. 2014. «Вариант экстраградиентного алгоритма для монотонных вариационных неравенств». *Кибернетика и системный анализ* 50(2):125-131.

Немировский, Аркадий, и Давид Юдин. 1979. *Сложность задач и эффективность методов оптимизации*. М.: Наука.

Нестеров, Юрий. 2010. *Введение в выпуклую оптимизацию*. М.: Изд-во МЦНМО.

Нурминский, Евгений. 1979. *Численные методы решения детерминированных и стохастических минимаксных задач*. Киев: Наукова думка.

Полак, Элиях. 1974. *Численные методы оптимизации. Единый подход*. М.: Мир.

Поляк, Борис. 1967. «Один общий метод решения экстремальных задач». *ДАН СССР* 174(1):33-6.

Поляк, Борис. 1969. «Минимизация негладких функционалов». *ЖВМ и МФ* 9(3):509-21.

Поляк, Борис. 1983. *Введение в оптимизацию*. М.: Наука.

Поляк, Борис. 2014. *Введение в оптимизацию*. М.: ЛЕНАНД.

Поляк, Борис, и Михаил Хлебников. 2017. «Метод главных компонент: робастные версии». *Автоматика и телемеханика* (3):130-48.

Поляк, Борис, и Георгий Смирнов. 2019а. «Переходные процессы в матричных дискретных линейных системах». *Автоматика и телемеханика* (9):112-21.

Поляк, Борис, и Лина Шалби. 2019б. «Стабилизация космического аппарата в точках Лагранжа с минимальным расходом топлива». *Автоматика и телемеханика* (12):160-72.

Поляк, Борис, и Ильяс Фатхуллин. 2020. «Применение проективного покоординатного спуска в задаче Фекете». *Ж. вычисл. матем. и матем. физ.* 60(5):815-27.

Пшеничный, Борис, и Юрий Данилин. 1975. *Численные методы в экстремальных задачах*. М.: Наука.

Сальников, Николай. 2014. «Эллипсоидальное оценивание состояний и параметров динамической системы при отсутствии априорной об оцениваемых величинах». *Международный научно-технический журнал «Проблемы управления и информатики»* 2:144-156.

Сергиенко, Иван, и Петр Стецюк. 2012. «О трех научных идеях Н.З. Шора». *Кибернетика и системный анализ* (1):4-22.

Скоков, В., и Михаил Щепакин. 1994. «Численное исследование метода ортогонального спуска». *Кибернетика и системный анализ* (2):139-49.

Стецюк, Петр. 1996. «r-алгоритмы и эллипсоиды». *Кибернетика и системный анализ* (1):113-34.

Стецюк, Петр. 1997а. «Ортогонализующие линейные операторы в выпуклом программировании. I». *Кибернетика и системный анализ* (3):97-119.

Стецюк, Петр. 1997б. «Ортогонализующие линейные операторы в выпуклом программировании. II». *Кибернетика и системный анализ* (5):111-24.

Стецюк, Петр. 2001. «К методам решения плохообусловленных систем линейных уравнений». *Теорія оптимальних рішень* 9-15.

Стецюк, Петр, и Д. Буханцов. 2002а. «К ускорению метода эллипсоидов с помощью использования шарового слоя». *Теория оптимальных решений* 63-70.

Стецюк, Петр, и Юрий Колесник. 2002б. «О робастности метода наименьших модулей». *Компьютерная математика* 114-23.

Стецюк, Петр. 2003а. «Приближенный метод эллипсоидов». *Кибернетика и системный анализ* (3):141-6.

Стецюк, Петр, Юрий Колесник, и Олег Березовский. 2003б. «Об одном методе нахождения L_p-решения системы линейных уравнений». *Теория оптимальных решений* 83-90.

Стецюк, Петр. 2009. «Субградиентные методы переменной метрики, использующие шаг Агмона-Моцкина и одноранговый эллипсоидальный оператор». *Труды АТИК* 1(12):16-25.

Стецюк, Петр. 2011а. «Субградиентные методы с преобразованием пространства для минимизации овражных выпуклых функций». Международная конференция, посвященная 90-летию со дня рождения академика Н.Н. Яненко Современные проблемы прикладной математики и механики: теория, эксперимент и практика, Россия, Новосибирск, май 30 – июнь 4.

Стецюк, Петр. 2011б. «Метод amsg2p для овражных выпуклых функций». *Информационный бюллетень Ассоциации математического программирования* (12):57-8.

Стецюк, Петр. 2012а. Ускоренные модификации субградиентного метода Поляка для овражных выпуклых функций. В *Стохастическое программирование и его приложения*, под редакцией Павла Кнопова и Валерия Зоркальцева, 160-84. Иркутск: Институт систем энергетики им. Л.А. Мелентьева СО РАН.

Стецюк, Петр. 2012б. «Ускорение субградиентного метода Поляка». *Теорія оптимальних рішень* (11):151-60.

Стецюк, Петр. 2013. «r-Алгоритмы: теория и практика». Матеріали XI Міжнародної науково-практичної конференції Математичне та програмне забезпечення інтелектуальних систем (MPZIS-2013), Дніпропетровськ, листопад 20-22.

Стецюк, Петр. 2014. *Методы эллипсоидов и r-алгоритмы*. Кишинэу: Эврика.

Стецюк, Петро, Микола Журбенко, Олег Березовський, та ін. 2016а. *Розробити субградієнтні алгоритми розв'язання задач оптимізації з гарантованою*

точністю (Заключний звіт про науково-дослідну роботу №0114U001055). Київ: Ін-т кібернетики імені В.М. Глушкова НАН України.

Стецюк, Петро, Віктор Стовба, та Ігор Мартинюк. 2016б. «Octave-програма dist2p для розділення двох поліедрів». Матеріали XIII Міжнародної науково-практичної конференції Теоретичні та прикладні аспекти побудови програмних систем (TAAPSD'2016), Київ, Грудень 5-9, 217-20.

Стецюк, Петр, Виктор Стовба, и Игорь Мартынюк. 2017а. «Алгоритм метода эллипсоидов для нахождения Lp-решения системы линейных уравнений». *Теорія оптимальних рішень* 139–46.

Стецюк, Петр, Галина Била, и Виктор Стовба. 2017б. «Метод эллипсоидов для нахождения L_p-решения системы линейных уравнений». Матеріали VIII Всеукр. наук-практ. конф. за міжнародною участю Інформатика та системні науки (ІСН-2017), Полтава, Березень 16-18, 258-64.

Стецюк, Петр. 2017в. «Оптимальный по объему 2d-эллипсоид и его приложения». *Тезисы докладов Международной конференции Конструктивный негладкий анализ и смежные вопросы, посвященной памяти профессора В.Ф. Демьянова, Часть II*. СПб.: Издательство ВВМ, 173-6.

Стецюк, Петро, та Віктор Стовба. 2017г. Метод еліпсоїдів для лінійної регресії. Матеріали IV Міжнародної науково-практичної конференції Обчислювальний інтелект (результати, проблеми, перспективи) (ComInt-2017), Київ, Травень 16-18, 314-5.

Стецюк, Петр, Александр Фесюк, и Ольга Хомяк. 2018а. «Обобщенный метод эллипсоидов». *Кибернетика и системный анализ* 54(4):70-80.

Стецюк, Петр, Виктор Стовба, и Александр Жмуд. 2018б. «Метод эллипсоидов для нахождения решения переопределенной СЛАУ». *Теорія оптимальних рішень* (17):115-23.

Стецюк, Петро, Віктор Стовба, та Олександр Жмуд. 2019а. «Про швидкість збіжності субградієнтних методів з кроком Поляка». *Наук. вісник Ужгород. ун-ту. Сер. матем. і інформ* 1(34):94-101.

Стецюк, Петро, та Віктор Стовба. 2019б. Субградієнтні методи з кроком Поляка та програма amsg2p. У *Субградієнтні алгоритми та задачі на комбінаторних конфігураціях*, під загал. ред. П.І. Стецюка, 34-61. Київ: ПУЛЬСАРИ.

Стецюк, Петро. 2019в. Комп’ютерна програма «Octave-програма amsg2p – субградієнтний метод з кроком Поляка та перетворенням простору змінних». Патент №88538. Державний департамент інтелектуальної власності, Міністерство освіти і науки. Опубліковано травень 11, 2019.

Стовба, Віктор, та Олександр Жмуд. 2019а. «Субградієнтний метод Поляка у перетвореному просторі змінних». Матеріали IX міжнародної школи-семінару «Теорія прийняття рішень», Ужгород, Квітень 15-20, 229-30.

Стовба, Віктор, Олександр Жмуд, та Олена Криворучко. 2019б. «Експерименти з субградієнтними методами Поляка для розв’язування сумісних СЛАР». *Теорія оптимальних рішень* (18):81-7.

Стовба, Віктор. 2020. «Метод еліпсоїдів для знаходження параметрів лінійної регресії». *Cybernetics and Computer Technologies* (3):14-24.

Сухарев, Алексей, Александр Тимохов, и Вячеслав Федоров. 2008. *Курс методов оптимизации*. М.: Физматлит.

Хачиян, Леонид. 1979а. «Полиномиальный алгоритм в линейном программировании». *Вычисл. математика и матем. физика* 20(2):51-68.

Хачиян, Леонид. 1979б. «Полиномиальный алгоритм в линейном программировании». *Докл. АН СССР* 244(5):1093-96.

Шор, Наум, и Николай Журбенко. 1971. «Метод минимизации, использующий операцию растяжения пространства в направлении разности двух последовательных субградиентов». *Кибернетика* (3):51-9.

Шор, Наум. 1975. «Исследование сходимости метода градиентного типа с растяжением пространства в направлении разности двух последовательных субградиентов». *Кибернетика* (4):48-53.

Шор, Наум. 1977. «Метод отсечения с растяжением пространства для решения задач выпуклого программирования». *Кибернетика* (1):94-5.

Шор, Наум, и В. Гершович. 1979. «Об одном семействе алгоритмов для решения задач выпуклого программирования». *Кибернетика* (4):62-7.

Шор, Наум. 1979. *Методы минимизации недифференцируемых функций и их приложения*. Киев: Наукова думка.

Шор, Наум, и Сергей Стеценко. 1989. *Квадратичные экстремальные задачи и недифференцируемая оптимизация*. Киев: Наукова думка.

Шор, Наум, Николай Журбенко, Алексей Лиховид, и Петр Стецюк. 2003. «Развитие алгоритмов недифференцируемой оптимизации и их приложения». *Кибернетика и системный анализ* (4):80-94.

Щепакин, Михаил. 1987. «О методе ортогонального спуска». *Кибернетика* (1):58-62.

Щепакин, Михаил, и Ирина Шубенкова. 1993. «Исследование модифицированного метода ортогонального спуска для поиска нуля выпуклой функции». *Кибернетика и системный анализ* (4):63-72.

Юдин, Давид, и Аркадий Немировский. 1976. «Информационная сложность и эффективные методы решения выпуклых экстремальных задач». *Экономика и матем. методы* 12(1):357-69.

Agmon, Shmuel. 1954. “The relaxation method for linear inequalities”. *Canad. J. Math*. (6):382−92.

Backem, Achim, and Martin Grötschel. 1981. “Characterizations of adjacency of faces of polyhedral”. *Mat. Progr. Study* (14):1-22.

Ben-Tal, Aharon, Alexander Goryashko, Elana Guslitzer, and Arkadi Nemirovski. 2004. “Adjustable robust solutions of uncertain linear programs”. *Mathematical Programming* 99:351-76.

Ben-Tal, Aharon, and Arkadi Nemirovski. 2008. “Selected Topics in Robust Convex Optimization”. *Mathematical Programming* 112(1):125-58.

Boyd, Stephen, and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

Camerini P., L. Fratta L, and F. Maffioli. 1975. “On improving relaxation methods by modified gradient techniques”. *Math. Program* (3):26-34.

Eremin, Ivan. 2002a. *Theory of linear optimization. Ser. Inverse and ill-posed problems*. Utrecht: VSP.

Eremin, Ivan. 2002b. “Fejer processes for infinite systems of convex inequalities”. *Proc. Of the Steklov Inst. of Math. Suppl* (1):S32-S51.

Fletcher, Roger, and C. Reeves. 1964. “Function minimization by conjugate gradients”. *Computer J.* 7(2):149-154.

Gelfand, Israel, and Michael Tsetlin. 1962. “Some methods of control for complex systems”. *Russian Math. Surveys* 17(1):95-117.

Grötschel, Martin, László Lovasz, and Alexander Schriyver. 1981. “The ellipsoid method and its consequences in combinatorial optimization”. *Combinatorica* 1(2):169-97.

Grötschel, Martin, László Lovasz, and Alexander Schriyver. 1988. *Geometric Algorithms and Combinatorial Optimization*. Berlin: Springer-Verlag.

Harchaoui, Zaid, Anatoli Juditsky, and Arkadi Nemirovski. 2015. “Conditional Gradient Algorithms for Norm-Regularized Smooth Convex Optimization”. *Mathematical Programming* 152(1-2):75-112.

Hestenes, Magnus, and Eduard Stiefel. 1952. “Methods of conjugate gradients for solving linear systems”. *Journal of research of the NBS* 49(6):409-436.

Juditsky, Anatoli, and Arkadi Nemirovski. 2009. “Non-parametric estimation via convex programming”. *Annals of Statistics* 37(5A):2278-300.

Juditsky, Arkadi, Fatma Karzan, Arkadi Nemirovski, and Boris Polyak. 2012. “Accuracy Guarantees for \ell_1 recovery of block-sparse signals”. *Annals of Statistics* 40(2):3077-107.

Kelley, J. 1960. “The cutting plane method for solving convex problems”. *J. Soc. For Industr. and Appl. Math*. 8(4):703-12.

König, Hermann, and Diethard Pallaschke. 1981. “On Khachian’s algorithm and minimal ellipsoids”. *Numerische Mathematik* (36):211-23.

Motzkin, Theodore, and Isaac Schoenberg. 1954. “The relaxation method for linear inequalities”. *Canad. J. Math*. (6):393−404.

Necoara, Ion, Yurii Nesterov, and Francois Glineur. 2019. “Linear convergence of first order methods for non-strongly convex optimization”. *Math. Program.* 175:69–107.

Nemirovski, Arkadi, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. 2009. “Stochastic approximation approach to stochastic programming”. *SIAM Journal on Optimization* 19(4):1574-609.

Nesterov, Yurii. 2018. “Complexity bounds for primal-dual methods minimizing the model of objective function”. *Math. Program*. 171:311–30.

Shor, Naum. 1985. *Minimization Methods for Non-Differentiable Functions*. 1st ed. Springer-Verlag Berlin Heidelberg.

Shor, Naum. 1998. *Nondifferentiable optimization and polynomial problems*. London: Kluwer Academic Publishers.

Shrader, R. 1980. *Ellipsoid methods (Rep.)*. Institut für öconometrie und operations research.

Stetsyuk, Petro. 2017. “2d-Ellipsoid of optimal volume and its applications”. Constructive Nonsmooth Analysis and Related Topics (Dedicated to the Memory of V.F. Demyanov) (CNSA).

Stetsyuk, Petro, Viktor Stovba, and Zhanna Chernousova. 2018. “Subgradient method with Polyak’s step in transformed space”. *Optimization and Applications (OPTIMA 2018)* 974:49-63.

Stovba, Viktor, and Olexandr Zhmud. 2018. “The ellipsoid method for linear regression”. 6-я международная научная конференция Математическое моделирование, оптимизация и информационные технологии (MMOTI-2018), Кишинэу, Молдова, Март 19-24, 206-8.

Vanderbei, Robert. 1997. *Linear programming. Foundations and extensions*. Boston, London, Dordrecht: Kluwer Academic Publishers.

ДОДАТОК А

ПРОГРАМНИЙ КОД ТА ЛІСТИНГ ПРИКЛАДУ 2.4 РОЗДІЛУ 2

Octave-код процедури squad1

```
# Code: squad1 function
function [f,g] = squad1(x)
global d;
n = rows(x);
f = d*((x-ones(n,1)).*(x-ones(n,1)));
g = 2.0*d'.*(x-ones(n,1));
return;
endfunction
```

Octave-код програми

```
# run_squad1: running PolyakA for squad1 function
global d;
n = 10000000;
rand("seed",2021);

alpha_mas = [2.0, 1.0, 0.5, 0.1, 0.01];

fstar = 0.d0; epsf = 1.e-20;
maxitn = 50; intp = 1;

for (i = 1:5)
    d = ones(1,n) + alpha_mas(i) * rand(1,n);

    lambda_max = max(d), lambda_min = min(d),
    printf("\n");

     # m = 1
    m1 = 1,
    x0 = zeros(n,1);
    tic(),
    [x1,f1,itn1,info1]=PolyakA(@squad1,x0,fstar,m1,epsf,maxitn,intp);
    toc(),

    # m = 2
    m2 = 2,
    x0 = zeros(n,1);
    tic(),
    [x2,f2,itn2,info2]=PolyakA(@squad1,x0,fstar,m2,epsf,maxitn,intp);
    toc(),

     # print info
    printf("\n");
    q = (itn1*1.0)/(itn2*1.0),
    itn1, itn2,
    f1, f2,
    dn1 = norm(x1-ones(n,1)), dn2 = norm(x2-ones(n,1)),
    printf("\n");
endfor
```

Лістинг програми

```
GNU Octave, version 6.1.0
Copyright (C) 2020 The Octave Project Developers.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.

Octave was configured for "x86_64-w64-mingw32".

Additional information about Octave is available at
https://www.octave.org.

Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.html

Read https://www.octave.org/bugs.html to learn how to submit bug
reports.
For information about changes from previous versions, type 'news'.

>> run_squad1

lambda_max = 3.0000
lambda_min = 1.0000

m1 = 1
Elapsed time is 21.9984 seconds.
m2 = 2
Elapsed time is 21.4586 seconds.

q = 1.0930
itn1 = 47
itn2 = 43
f1 = 6.0030e-21
f2 = 7.2469e-21
dn1 = 7.7015e-11
dn2 = 6.8147e-11

lambda_max = 2.0000
lambda_min = 1.0000

m1 = 1
Elapsed time is 22.3958 seconds.
m2 = 2
Elapsed time is 13.5059 seconds.

q = 1.6786
itn1 = 47
itn2 = 28
f1 = 3.7858e-21
f2 = 3.6754e-21
dn1 = 6.1168e-11
dn2 = 5.2776e-11

lambda_max = 1.5000
lambda_min = 1.0000
```

```
m1 = 1
Elapsed time is 22.8637 seconds.
m2 = 2
Elapsed time is 9.82495 seconds.

q = 2.3500
itn1 = 47
itn2 = 20
f1 = 2.8335e-21
f2 = 8.1206e-22
dn1 = 5.2924e-11
dn2 = 2.6059e-11

lambda_max = 1.1000
lambda_min = 1.0000

m1 = 1
Elapsed time is 21.6163 seconds.
m2 = 2
Elapsed time is 5.20221 seconds.

q = 4.1818
itn1 = 46
itn2 = 11
f1 = 8.6471e-21
f2 = 1.7920e-21
dn1 = 9.2464e-11
dn2 = 4.1348e-11

lambda_max = 1.0100
lambda_min = 1.0000

m1 = 1
Elapsed time is 21.8226 seconds.
m2 = 2
Elapsed time is 3.10046 seconds.

q = 6.5714
itn1 = 46
itn2 = 7
f1 = 8.1212e-21
f2 = 1.7769e-22
dn1 = 8.9926e-11
dn2 = 1.3297e-11
```

ДОДАТОК Б

ПРОГРАМНИЙ КОД ТА ЛІСТИНГ ПРИКЛАДУ 2.5 РОЗДІЛУ 2

Octave-код процедури squad2

```
# Code: squad2 function
function [f,g] = squad2(x)
global H b c
temp = H*x;
f = x'*temp + b'*x + c;
g = 2.0*temp + b;
return;
endfunction
```

Octave-код програми

```
# run_squad2: running PolyakA for squad2 function
global H b c;
n = 1000;
m = 100000;
rand("seed",2021);

alpha_mas = [0.1, 1, 10, 50, 200];

fstar = 0.d0; epsf = 1.d-6;
maxitn = 200; intp = 1;

for (i = 1:5)
    A = ones(m,n) + alpha_mas(i)*rand(m,n);
    A = A/n;

    H = A'*A; rhs = A*ones(n,1); b = -2.0*A'*rhs; c = rhs'*rhs;
    H = 1.0/m*H; b = 1.0/m*b; c = 1.0/m*c;

    t = eig(H);

    lambda_max = max(t), lambda_min = min(t),
    printf("\n");

    # m = 1
    m1 = 1,
    x0 = zeros(n,1);
    tic(),
    [x1,f1,itn1,info1]=PolyakA(@squad2,x0,fstar,m1,epsf,maxitn,intp);
    toc(),

    # m = 1

    m2 = 2,
    x0 = zeros(n,1);
    tic(),
    [x2,f2,itn2,info2]=PolyakA(@squad2,x0,fstar,m2,epsf,maxitn,intp);
    toc(),

    # print info
    printf("\n");
    q = (itn1*1.0)/(itn2*1.0),
    itn1, itn2,
    f1, f2,
    dn1 =  norm(x1-ones(n,1)), dn2 =  norm(x2-ones(n,1)),
```

```
    printf("\n");
endfor
```

Лістинг програми

```
GNU Octave, version 6.1.0
Copyright (C) 2020 The Octave Project Developers.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.

Octave was configured for "x86_64-w64-mingw32".

Additional information about Octave is available at
https://www.octave.org.

Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.html

Read https://www.octave.org/bugs.html to learn how to submit bug
reports.
For information about changes from previous versions, type 'news'.

>> run_squad2

lambda_max = 1.1025e-03
lambda_min = 6.7607e-10

m1 = 1
Elapsed time is 0.0113249 seconds.
m2 = 2
Elapsed time is 0.000977993 seconds.

q = 6
itn1 = 12
itn2 = 2
f1 = 2.6286e-07
f2 = 4.8850e-15
dn1 = 0.015680
dn2 = 2.7274e-03

lambda_max = 2.2501e-03
lambda_min = 6.7504e-08

m1 = 1
Elapsed time is 0.011353 seconds.
m2 = 2
Elapsed time is 0.00407505 seconds.

q = 6
itn1 = 12
itn2 = 2
f1 = 5.3647e-07
f2 = 2.9286e-11
dn1 = 0.024263
dn2 = 0.018719
```

```
lambda_max = 0.036007
lambda_min = 6.7566e-06

m1 = 1
Elapsed time is 0.01618 seconds.
m2 = 2
Elapsed time is 0.00216293 seconds.

q = 7
itn1 = 14
itn2 = 2
f1 = 5.4279e-07
f2 = 1.8664e-08
dn1 = 0.047546
dn2 = 0.047452

lambda_max = 0.6764
lambda_min = 1.6900e-04

m1 = 1
Elapsed time is 0.0155089 seconds.
m2 = 2
Elapsed time is 0.00144506 seconds.

q = 8
itn1 = 16
itn2 = 2
f1 = 8.9535e-07
f2 = 6.4400e-07
dn1 = 0.055398
dn2 = 0.055509

lambda_max = 10.206
lambda_min = 2.7035e-03

m1 = 1
Elapsed time is 0.056674 seconds.
m2 = 2
Elapsed time is 0.00149703 seconds.

q = 23.333
itn1 = 70
itn2 = 3
f1 = 8.9798e-07
f2 = 2.1280e-07
dn1 = 0.015958
dn2 = 5.7170e-03
```

ДОДАТОК В

ПРОГРАМНИЙ КОД ТА ЛІСТИНГ ПРИКЛАДУ 3.4 РОЗДІЛУ 3

Octave-код програми

```
# run_squad2_AB: running PolyakA and PolyakB with m = 1,2 for squad2
function
global H b c;
n = 1000;
m = 50000;
rand("seed",2021);

alpha_mas = [0.1, 1.0, 10.0, 50.0, 200.0];

fstar = 0.d0; epsf = 1.d-10;
maxitn = 10000; intp = 100;

for (i = 1:5)
    A = ones(m,n) + alpha_mas(i)*rand(m,n); A = A/n;

    xs = 10*rand(n,1); # rand xstar - for "run_squad2_AB_rand"
    # xs = ones(n,1); # xstar = (1,...,1)
    H = A'*A; rhs = A*xs; b = -2.0*A'*rhs; c = rhs'*rhs;
    H = 1.0/m*H; b = 1.0/m*b; c = 1.0/m*c;

    [VC VL] = eig(H);
    evl = VL*ones(n,1);
    xi = VC(1:n,n);
    t1 = evl(1,1); # min eigenvalue
    tn = evl(n,1); # max eigenvalue

    lambda_max = max(evl), lambda_min = min(evl),
    beta = sqrt(t1/tn),
    printf("\n");
    B = diag(ones(n,1)) + (beta-1.0)*xi*xi';

    disp("PolyakA (m=1)");
    m1 = 1;
    x0 = zeros(n,1);
    tic(),
    [x1,f1,itn1,info1]=PolyakA(@squad2,x0,fstar,m1,epsf,maxitn,intp);
    toc(),

    disp("PolyakA (m=2)");
    m2 = 2;
    x0 = zeros(n,1);
    tic(),
    [x2,f2,itn2,info2]=PolyakA(@squad2,x0,fstar,m2,epsf,maxitn,intp);
    toc(),

    disp("PolyakB (m=1)");
    m3 = 1;
    x0 = zeros(n,1);
    tic(),
    [x3,f3,itn3,info3]=PolyakB(@squad2,B,x0,fstar,m3,epsf,maxitn,intp);
    toc(),

    disp("PolyakB (m=2)");
    m4 = 2;
    x0 = zeros(n,1);
```

```
    tic(),
    [x4,f4,itn4,info4]=PolyakB(@squad2,B,x0,fstar,m4,epsf,maxitn,intp);
    toc(),

    # print info
    printf("\n");
    itn1, itn2, itn3, itn4,
    f1, f2, f3, f4,
    dn1 = norm(x1-xs), dn2 = norm(x2-xs),
    dn3 = norm(x3-xs), dn4 = norm(x4-xs),
    printf("\n");
endfor
```

Лістинг програми

```
GNU Octave, version 6.1.0
Copyright (C) 2020 The Octave Project Developers.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.

Octave was configured for "x86_64-w64-mingw32".

Additional information about Octave is available at
https://www.octave.org.

Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.html

Read https://www.octave.org/bugs.html to learn how to submit bug
reports.
For information about changes from previous versions, type 'news'.

>> run_squad2_AB

lambda_max = 1.1025e-03
lambda_min = 6.1713e-10
lambda = 1.7865e+06
beta = 7.4817e-04

PolyakA (m=1)
Elapsed time is 0.0141089 seconds.
PolyakA (m=2)
Elapsed time is 0.001472 seconds.
PolyakB (m=1)
Elapsed time is 0.13248 seconds.
PolyakB (m=2)
Elapsed time is 0.0093739 seconds.

itn1 = 18
itn2 = 2
itn3 = 18
itn4 = 2
f1 = 6.4180e-11
f2 = 1.1990e-14
f3 = 6.4174e-11
```

```
f4 = 2.2204e-15
dn1 = 4.0031e-03
dn2 = 3.9959e-03
dn3 = 2.4126e-04
dn4 = 1.5794e-03

lambda_max = 2.2501e-03
lambda_min = 6.1461e-08
lambda = 3.6610e+04
beta = 5.2264e-03

PolyakA (m=1)
Elapsed time is 0.0156779 seconds.
PolyakA (m=2)
Elapsed time is 0.00148606 seconds.
PolyakB (m=1)
Elapsed time is 0.149338 seconds.
PolyakB (m=2)
Elapsed time is 0.00987601 seconds.

itn1 = 19
itn2 = 2
itn3 = 19
itn4 = 2
f1 = 6.0516e-11
f2 = 5.6690e-11
f3 = 3.2743e-11
f4 = 1.0835e-11
dn1 = 0.025964
dn2 = 0.025973
dn3 = 1.2063e-04
dn4 = 0.010752

lambda_max = 0.036011
lambda_min = 6.1519e-06
lambda = 5853.7
beta = 0.013070

PolyakA (m=1)
itn  100 f   6.083472e-08
Elapsed time is 0.271265 seconds.
PolyakA (m=2)
itn  100 f   1.210751e-09
Elapsed time is 1.44648 seconds.
PolyakB (m=1)
Elapsed time is 0.168216 seconds.
PolyakB (m=2)
Elapsed time is 0.019855 seconds.

itn1 = 523
itn2 = 2871
itn3 = 21
itn4 = 3
f1 = 8.1414e-11
f2 = 9.9931e-11
f3 = 3.2756e-11
f4 = 5.8222e-11
```

```
dn1 = 3.2847e-03
dn2 = 2.5923e-03
dn3 = 3.0171e-05
dn4 = 2.5856e-03

lambda_max = 0.6761
lambda_min = 1.5379e-04
lambda = 4396.5
beta = 0.015082

PolyakA (m=1)
itn  100 f   4.543084e-07
Elapsed time is 0.234901 seconds.
PolyakA (m=2)
itn  100 f   4.177377e-08
Elapsed time is 2.93363 seconds.
PolyakB (m=1)
Elapsed time is 0.208213 seconds.
PolyakB (m=2)
Elapsed time is 0.028738 seconds.

itn1 = 442
itn2 = 5541
itn3 = 23
itn4 = 4
f1 = 9.6065e-11
f2 = 9.9476e-11
f3 = 3.8654e-11
f4 = 4.2064e-11
dn1 = 7.1489e-04
dn2 = 5.4792e-04
dn3 = 7.5244e-06
dn4 = 4.4084e-04

lambda_max = 10.202
lambda_min = 2.4511e-03
lambda = 4162.3
beta = 0.015500

PolyakA (m=1)
itn  100 f   4.692160e-06
Elapsed time is 0.287394 seconds.
PolyakA (m=2)
itn  100 f   7.656818e-07
Elapsed time is 5.55822 seconds.
PolyakB (m=1)
Elapsed time is 0.202925 seconds.
PolyakB (m=2)
Elapsed time is 0.0319321 seconds.

itn1 = 386
itn2 = 7397
itn3 = 25
itn4 = 5
f1 = 7.2760e-11
f2 = 9.6406e-11
f3 = 3.8199e-11
```

```
f4 = 2.3647e-11
dn1 = 1.6224e-04
dn2 = 1.7324e-04
dn3 = 2.0261e-06
dn4 = 9.6286e-05

>> run_squad2_AB_rand

lambda_max = 1.1025e-03
lambda_min = 6.1713e-10
lambda = 1.7865e+06
beta = 7.4817e-04

PolyakA (m=1)
Elapsed time is 4.24074 seconds.
PolyakA (m=2)
Elapsed time is 0.00870204 seconds.
PolyakB (m=1)
Elapsed time is 0.149998 seconds.
PolyakB (m=2)
Elapsed time is 0.040179 seconds.

itn1 = 7399
itn2 = 14
itn3 = 20
itn4 = 5
f1 = 8.2714e-11
f2 = 1.9686e-11
f3 = 9.7007e-11
f4 = 8.0433e-12
dn1 = 0.3471
dn2 = 0.1732
dn3 = 2.9807e-04
dn4 = 0.098345

lambda_max = 2.2501e-03
lambda_min = 6.1640e-08
lambda = 3.6504e+04
beta = 5.2340e-03

PolyakA (m=1)
Elapsed time is 1.42519 seconds.
PolyakA (m=2)
Elapsed time is 0.013571 seconds.
PolyakB (m=1)
Elapsed time is 0.173218 seconds.
PolyakB (m=2)
Elapsed time is 0.0452602 seconds.

itn1 = 1892
itn2 = 18
itn3 = 21
itn4 = 6
f1 = 6.9335e-11
f2 = 1.9234e-11
f3 = 4.9866e-11
f4 = 3.0568e-11
```

```
dn1 = 0.023849
dn2 = 0.017192
dn3 = 1.4925e-04
dn4 = 0.018884

lambda_max = 0.036011
lambda_min = 6.1691e-06
lambda = 5837.3
beta = 0.013089

PolyakA (m=1)
Elapsed time is 0.603088 seconds.
PolyakA (m=2)
Elapsed time is 0.0122051 seconds.
PolyakB (m=1)
Elapsed time is 0.195105 seconds.
PolyakB (m=2)
Elapsed time is 0.0569792 seconds.

itn1 = 932
itn2 = 22
itn3 = 23
itn4 = 8
f1 = 9.5952e-11
f2 = 4.4338e-11
f3 = 5.2296e-11
f4 = 9.3223e-12
dn1 = 3.8146e-03
dn2 = 2.6010e-03
dn3 = 3.8107e-05
dn4 = 1.0005e-03

lambda_max = 0.6761
lambda_min = 1.5365e-04
lambda = 4400.3
beta = 0.015075

PolyakA (m=1)
Elapsed time is 0.528141 seconds.
PolyakA (m=2)
Elapsed time is 0.013792 seconds.
PolyakB (m=1)
Elapsed time is 0.195255 seconds.
PolyakB (m=2)
Elapsed time is 0.066335 seconds.

itn1 = 879
itn2 = 26
itn3 = 25
itn4 = 9
f1 = 9.0949e-11
f2 = 4.7294e-11
f3 = 8.7311e-11
f4 = 4.0018e-11
dn1 = 6.6769e-04
dn2 = 3.4942e-04
dn3 = 8.2731e-06
```

```
dn4 = 2.2145e-04

lambda_max = 10.202
lambda_min = 2.4585e-03
lambda = 4149.9
beta = 0.015523

PolyakA (m=1)
Elapsed time is 0.778048 seconds.
PolyakA (m=2)
Elapsed time is 0.0175791 seconds.
PolyakB (m=1)
Elapsed time is 0.213947 seconds.
PolyakB (m=2)
Elapsed time is 0.070801 seconds.

itn1 = 1449
itn2 = 28
itn3 = 27
itn4 = 10
f1 = 5.8208e-11
f2 = 8.7311e-11
f3 = -5.8208e-11
f4 = 2.9104e-11
dn1 = 2.4475e-04
dn2 = 1.6033e-04
dn3 = 2.2742e-06
dn4 = 1.1818e-04
```

ДОДАТОК Г

ТАБЛИЦЯ 4.5 ПІДРОЗДІЛУ 4.3 РОЗДІЛУ 4

Спостереження (u_i, f_i) та відхилення $f_i - Q_p^*(u_i)$ для значень $p = 1, 1.3, 1.6, 2, 5$

i	u_i	f_i	$f_i - Q_p^*(u_i)$				
			$p=1$	$p=1.3$	$p=1.6$	$p=2$	$p=5$
1	(2.0, 8.5, 2.5, 4.7)	0.5	0.000	0.000	0.000	0.000	– 0.001
2	(1.5, 9.0, 2.5, 4.7)	0.5	0.000	– 0.002	– 0.007	– 0.009	– 0.013
3	(1.5, 8.5, 2.5, 5.2)	0.5	– 0.001	0.000	– 0.003	– 0.006	– 0.010
4	(1.5, 8.5, 2.5, 5.2)	0.5	– 0.001	0.000	– 0.003	–0.006	– 0.010
5	(2.0, 8.0, 3.0, 4.7)	0.5	0.000	0.004	0.007	0.009	0.011
6	(2.0, 8.0, 2.5, 5.2)	0.5	0.000	0.008	0.010	0.010	0.011
7	(2.0, 8.5, 2.0, 5.2)	0.5	– 0.001	– 0.002	– 0.003	– 0.005	– 0.007
8	(4.0, 8.8, 5.4, 7.7)	0.9	0.000	– 0.005	– 0.007	– 0.010	– 0.014
9	(3.1, 9.7, 5.4, 7.7)	0.9	0.000	– 0.001	– 0.003	– 0.004	– 0.001
10	(3.1, 8.8, 6.3, 7.7)	0.9	0.000	0.000	0.000	0.000	– 0.006
11	(3.1, 8.8, 5.4, 8.6)	0.9	0.000	0.004	0.004	0.004	– 0.001
12	(4.0, 7.9, 6.3, 7.7)	0.9	0.000	0.000	– 0.002	– 0.004	– 0.011
13	(4.0, 7.9, 5.4, 8.6)	0.9	0.004	0.022	0.020	0.019	0.015
14	(4.0, 8.8, 4.5, 8.6)	0.9	0.000	– 0.002	– 0.005	– 0.006	– 0.007
15	(5.9, 8.7, 4.7, 5.4)	1.0	0.000	– 0.001	– 0.003	– 0.004	– 0.010
16	(4.9, 9.7, 4.7, 5.4)	1.0	0.000	0.010	0.011	0.011	0.013
17	(4.9, 8.7, 5.7, 5.4)	1.0	0.000	0.000	0.011	0.003	0.004
18	(4.9, 8.7, 4.7, 6.4)	1.0	0.000	0.001	0.001	0.001	– 0.002
19	(5.9, 7.7, 5.7, 5.4)	1.0	0.000	– 0.007	– 0.010	– 0.010	– 0.010
20	(5.9, 7.7, 4.7, 6.4)	1.0	0.005	0.017	0.013	0.012	0.012
21	(5.9, 8.7, 3.7, 6.4)	1.0	0.000	– 0.002	– 0.005	– 0.007	– 0.012
22	(2.0, 9.0, 2.2, 7.0)	0.9	0.003	0.010	0.013	0.014	0.015
23	(1.1, 9.9, 2.2, 7.0)	0.9	0.000	– 0.007	– 0.011	– 0.014	– 0.014
24	(1.1, 9.0, 3.1, 7.0)	0.9	0.005	0.021	0.023	0.022	0.017
25	(1.1, 9.0, 2.2, 7.9)	0.9	0.002	0.012	0.013	0.012	0.010

26	(2.0, 8.1, 3.1, 7.9)	0.9	– 0.074	– 0.030	– 0.022	– 0.018	– 0.015
27	(2.0, 8.1, 3.1, 7.9)	0.9	– 0.074	– 0.030	– 0.022	– 0.018	– 0.015
28	(2.0, 9.0, 1.3, 7.9)	0.9	0.000	0.000	0.001	0.004	0.010

ДОДАТОК Д

СПИСОК ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДИСЕРТАЦІЇ ТА ВІДОМОСТІ ПРО АПРОБАЦІЮ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЇ

Публікації, в яких опубліковано
основні наукові результати дисертації:

1. Стецюк, Петр, Виктор Стовба, и Игорь Мартынюк. 2017. «Алгоритм метода эллипсоидов для нахождения Lp-решения системы линейных уравнений». *Теорія оптимальних рішень* 139–46.
2. Стецюк, Петр, Виктор Стовба, и Александр Жмуд. 2018. «Метод эллипсоидов для нахождения решения переопределенной СЛАУ». *Теорія оптимальних рішень* (17):115-23.
3. Stetsyuk, Petro, Viktor Stovba, and Zhanna Chernousova. 2018. "Subgradient Method with Polyak's Step in Transformed Space". *Optimization and Applications. OPTIMA 2018. Communications in Computer and Information Science* 974:49-63.
4. Стовба, Віктор, Олександр Жмуд, та Олена Криворучко. 2019. «Експерименти з субградієнтними методами Поляка для розв'язування сумісних СЛАР». *Теорія оптимальних рішень* (18):81-7.
5. Стецюк, Петро, Віктор Стовба, та Олександр Жмуд. 2019. «Про швидкість збіжності субградієнтних методів з кроком Поляка». *Наук. вісник Ужгород. ун-ту. Сер. матем. і інформ* 1(34):94-101.
6. Стецюк, Петро, та Віктор Стовба. 2019. «Субградієнтні методи з кроком Поляка та програма amsg2p». У *Субградієнтні алгоритми та задачі на комбінаторних конфігураціях*. Київ: Унів. вид-во ПУЛЬСАРИ.
7. Стовба, Віктор. 2020. «Метод еліпсоїдів для знаходження параметрів лінійної регресії». *Cybernetics and Computer Technologies* (3):14-24.

Публікації, що засвідчують
апробацію матеріалів дисертації

1. Стецюк, Петро, Віктор Стовба, та Ігор Мартинюк. 2016. «Octave-програма dist2p для розділення двох поліедрів». Матеріали XIII Міжнародної

науково-практичної конференції Теоретичні та прикладні аспекти побудови програмних систем (TAAPSD’2016), Київ, Грудень 5-9, 217-20.

2. Стецюк, Петр, Галина Била, и Виктор Стовба. 2017. «Метод эллипсоидов для нахождения L_p-решения системы линейных уравнений». Матеріали VIII Всеукр. наук-практ. конф. за міжнародною участю Інформатика та системні науки (ІСН-2017), Полтава, Березень 16-18, 258-64.
3. Стецюк, Петро, та Віктор Стовба. 2017. Метод еліпсоїдів для лінійної регресії. Матеріали IV Міжнародної науково-практичної конференції Обчислювальний інтелект (результати, проблеми, перспективи) (ComInt-2017), Київ, Травень 16-18, 314-5.
4. Stovba, Viktor, and Oleksandr Zhmud. 2018. “The ellipsoid method for linear regression”. Материалы 6-й международной научной конференции Математическое моделирование, оптимизация и информационные технологии (MMOTI-2018), Кишинэу, Молдова, Март 19-24, 206-8.
5. Стовба, Віктор, та Олександр Жмуд. 2019. «Субградієнтний метод Поляка у перетвореному просторі змінних». Матеріали IX міжнародної школи-семінару «Теорія прийняття рішень» у рамках Міжнародного наукового симпозіуму Інтелектуальні рішення (IntSol-2019), Ужгород, Квітень 15-20, 229-30.

Відомості про апробацію результатів дисертації

Результати дисертації доповідались та обговорювались на:

- XIII Міжнародній науково-практичній конференції «Теоретичні та прикладні аспекти побудови програмних систем (TAAPSD’2016)», 5 – 9 грудня, 2016 року, м. Київ, Україна;
- VIII Всеукраїнській науково-практичній конференції «Інформатика та системні науки (ІСН-2017)», 16 – 18 березня, 2017 року, м. Полтава, Україна;

- IV Міжнародній науково-практичній конференції «Обчислювальний інтелект (результати, проблеми, перспективи) (ComInt-2017)», 16 – 18 травня, 2017 року, м. Київ, Україна;
- 6-й міжнародній науковій конференції «Математическое моделирование, оптимизация и информационные технологии (MMOTI-2018)», 12 – 16 листопада, 2018 року, м. Кишинів, Молдова;
- 9th International Conference “Optimization and Applications (OPTIMA 2018)”, 1 – 5 жовтня, 2018 року, м. Петровац, Чорногорія;
- IX міжнародній школі-семінарі «Теорія прийняття рішень» в рамках Міжнародного наукового симпозіуму «Інтелектуальні рішення (IntSol-2019)», 15 – 20 квітня, 2019 року, м. Ужгород, Україна;
- International Conference “Optimization and Equilibrium Problems (ICOEP 2019)”, 31 липня – 2 серпня, 2019 року, м. Дрезден, Німеччина;
- Norwegian-Eurasian Workshop “New Resilience Challenges in Ecological-Economic Problems at the Digital Era”, 22 – 27 вересня, 2019 року, м. Київ, Україна;
- засідання відділу методів негладкої оптимізації Інституту кібернетики імені В.М. Глушкова НАН України (керівник засідання – член-кореспондент НАН України О.М. Хіміч), 30 грудня 2020 року.