НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ КІБЕРНЕТИКИ ІМЕНІ В.М. ГЛУШКОВА

Кваліфікаційна наукова
праця на правах рукопису

Жидков Володимир Олександрович

УДК 519.8

ДИСЕРТАЦІЯ

МЕТОДИ НЕГЛАДКОЇ ОПТИМІЗАЦІЇ ДЛЯ ПОБУДОВИ КРИВИХ У НАТУРАЛЬНІЙ ПАРАМЕТРИЗАЦІЇ ТА ВИЯВЛЕННЯ ДЕФЕКТІВ НА СТРУКТУРОВАНИХ ЗОБРАЖЕННЯХ

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. Жидков

Науковий керівник: **Стецюк Петро Іванович**
д.ф.-м.н., професор,
член-кореспондент НАН України

Київ – 2025

АНОТАЦІЯ

Жидков В.О. Методи негладкої оптимізації для побудови кривих у натуральній параметризації та виявлення дефектів на структурованих зображеннях. 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. 2025.

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 підходи та методи, які використовуються для виявлення дефектів, а також побудови кривих в натуральній параметризації. Проаналізовано роботи вітчизняних та закордонних вчених: Войтюка Д., Пилипаки С., Борисенка В., Устенко С., Устенко І., Лобанова Л., Савицького В., Стецюка П., Хом'як О., Журбенка М., Єрьоміна І., Лаптіна Ю., Пшеничного Б., Шора Н., Роботишина М., Маляра М., Сергієнка І., Литвина О., Ткаченко О., Allen B., Ashrafi S., Bertsekas D., M.A. Rezaei, H. Wu, D. Vucina. У роботах вищенаведених авторів викладено загальновідомі методи негладкої оптимізації та описано їхні властивості, аспекти чисельного й експериментального розв'язання задач виявлення дефектів на регулярних та періодичних структурах, а також побудови кривих в натуральній параметризації, за допомогою різних методів та алгоритмів. Зроблено короткий огляд новітніх підходів до розв'язання описаних проблем. На основі цього огляду зроблено висновки та поставлено задачі для дослідження.

У **розділі 2** наведено опис програми **ralgb5a**, яка реалізує модифікацію *r*-алгоритмів Шора з адаптивним способом регулювання кроку та призначена для мінімізації гладких та негладких опуклих функцій. Програму розроблено некомерційною мовою GNU Octave. Досліджено метод негладких штрафних

функцій для задач опуклого програмування з урахуванням випадку, коли цільова функція визначена не на всьому просторі змінних. Програма **ralgb5a** та методи негладких штрафних функцій використовуються для розв'язання задач побудови кривих у натуральній параметризації та виявлення дефектів на структурованих зображеннях, які представлені в наступних розділах дисертаційної роботи.

У **розділі 3** розроблено математичну модель, алгоритм та програмне забезпечення для задачі побудови S-подібної кривої, яка проходить через дві задані точки із заданими кутами нахилу дотичних у них та забезпечує заданий кут нахилу дотичної в точці із заданою абсцисою. Для керування точкою перегину S-подібної кривої з квадратичним законом розподілу кривини в натуральній параметризації використовується кут нахилу дотичної в точці із заданою абсцисою. Алгоритм базується на модифікації градієнтного методу з розтягом простору в напрямі різниці двох послідовних узагальнених градієнтів. Представлено опис програмної реалізації та обчислювальних експериментів, які показали ефективність розробленого алгоритму для проектування зовнішнього контуру сопла типу Франкля.

У **розділі 4** розроблено математичну модель, алгоритм та програмне забезпечення для задачі побудови кривої, яка проходить через дві задані точки із заданими у них кутами нахилу дотичних та значень кривини. Математична модель представлена оптимізаційною задачею, еквівалентною системі чотирьох нелінійних рівнянь з чотирма невідомими, яка описує задачу побудови необхідної кривої в натуральній параметризації з кубічним розподілом кривини. Алгоритм розв'язання оптимізаційної задачі використовує модифікацію r-алгоритму. Програмне забезпечення мовою Octave реалізує алгоритм розв'язання оптимізаційної задачі та графічну складову для відображення графіку кривої та графіків залежності від довжини кривої трьох її характеристик – кут дотичної, кривина та похідна кривини.

У **розділі 5** сформульовано оптимізаційні задачі для знаходження найкращих за L_p-нормою параметрів регулярних 3D-структур і методи найменших модулів та найменших квадратів для їх розв'язання. Показано, що

при відновленні параметрів 3D-структур з дефектами метод найменших модулів стійкіший, ніж метод найменших квадратів. Наведено результати обчислювальних експериментів для програмних реалізацій методів на основі *r*-алгоритму.

У **розділі 6** розроблено новий алгоритм пошуку дефектів у періодичних структурах за допомогою порівняння експериментальних та ідеалізованих зображень, та представлено загально застосовне алгоритмічне рішення, яке дозволяє аналізувати зображення та виявляти дефекти, порівняно або навіть краще за людину-оператора. Запропонований метод виявляє дефекти шляхом розпізнавання не дефектів, а періодичного фону. Також розроблено математичне формулювання для функції похибок, адаптованої до апроксимації даних та зображень з дефектами. Придатність функції похибок була проаналізована, включаючи обчислювальний експеримент для верифікації адекватності. Запропоновано новий тип функції похибок, спеціально адаптований до апроксимації наборів даних з високою щільністю викидів, з ефективністю, що порівняна, а в деяких випадках навіть перевищує ефективність L_1-норми в цьому аспекті.

Ключові слова: негладка оптимізація, *r*-алгоритм, натуральна параметризація, плоскі криві, аеродинамічні та технічні профілі, регулярні та періодичні зображення, проявлення дефектів.

ABSTRACT

Zhydkov V.O. Nonsmooth optimization methods of construction naturally parametrized curves and revealing defects in structured images. 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. 2025.

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.

Chapter 1 discusses the main approaches and methods used to reveal defects and construct naturally parametrized curves. The works of the following domestic and foreign scientists are analyzed: Voytyuk D., Pylypaka S., Borysenko V., Ustenko S., Ustenko I., Lobanov L., Savytsky V., Stetsyuk P., Khomiak O., Zhurbenko M., Eremin I., Laptin Yu., Pshenychny B., Shor N., Robotyshin M., Malyar M., Sergienko I., Lytvyn O., Tkachenko O., Allen B., Ashrafi S., Bertsekas D., M.A. Rezaei, H. Wu, D. Vucina. The works of the above authors present well-known methods of nonsmooth optimization and describe their properties, aspects of numerical and experimental solving problems of revealing defects on regular and periodic structures, as well as construction naturally parametrized curves using various methods and algorithms. A brief review of the latest approaches to solving the problems described is made. Based on this review, conclusions are made, and research tasks are set to be investigated.

In **Chapter 2**, the **ralgb5a** program is described, which implements a modification of *r*-algorithms with adaptive step adjustment and is designed to minimize smooth and nonsmooth convex functions. The program is implemented using non-commercial GNU Octave language. The nonsmooth penalty function method for convex programming problems is studied, considering the case when the objective

function is not defined on the entire space of variables. The **ralgb5a** program and nonsmooth penalty function methods are used to solve the problems of construction naturally parametrized curves and revealing defects in structured images, which are presented in the following sections of the dissertation.

In **Chapter 3** developed is mathematical model, algorithm, and software for the problem of constructing S-shaped curve that passes through two given points with given tangent angles at them and provides a given tangent angle at a point with a given abscissa. To control the inflection point of a S-shaped curve with a quadratic law of curvature distribution in natural parametrization, the tangent angle at a point with a given abscissa is used. The algorithm is based on a modification of the gradient method with space dilation in the direction of the difference of two consecutive generalized gradients. Description of the software implementation and computational experiments is presented, which showed effectiveness of the developed algorithm for designing the outer contour of a Frankl-type nozzle.

In **Chapter 4** developed are mathematical model, algorithm, and software for the problem of constructing a curve that passes through two given points with given tangent angles and curvature values. The mathematical model is represented by an optimization problem equivalent to a system of four nonlinear equations with four unknown variables, which describes the problem of constructing the required naturally parametrized curve with a cubic curvature distribution. The algorithm for solving the optimization problem uses a modification of the *r*-algorithm. Octave software implements the algorithm for solving the optimization problem and a graphical component for displaying the curve graph and graphs of dependence of the curve length on its three characteristics – tangent angle, curvature, and curvature derivative.

In **Chapter 5**, optimization problems for finding the best L_p-norm parameters of regular 3D structures, as well as the least moduli method and the least squares method for their solving are formulated. It is shown that when restoring the parameters of 3D structures with defects, the least moduli method is more robust than the least squares method. Results of computational experiments for software implementations of methods based on *r*-algorithm are presented.

In **Chapter 6** presented are a new algorithm for finding defects in periodic structures using comparison of experimental and idealized images, and a generally applicable algorithmic solution that allows one to analyze images and reveal defects comparable to or even better than a human operator. The method proposed reveals defects by recognizing not defects, but periodic backgrounds. A mathematical formulation for an error function adapted to approximation of data and images with defects is developed as well. The suitability of the error function is analyzed, including a computational experiment to verify the adequacy. A new type of error function is proposed, specifically adapted to the approximation of datasets with high outlier density, with an efficiency comparable to, and in some cases even exceeding, the L_1-norm efficiency in this aspect.

Keywords: nonsmooth optimization, *r*-algorithm, natural parametrization, flat curve, aerodynamic and technical profiles, regular and periodic images, revealing defects.

Список публікацій здобувача

Публікації, в яких опубліковано основні наукові результати дисертації

1. Жидков, Владимир. 2017. «Определение дефектов в периодических структурах». *Компьютерная математика* 2:21–29.
2. Стецюк, Петро, Ольга Хом'як, та Володимир Жидков. 2023а. «Масштабування даних для задачі побудови кривої в натуральній параметризації з кубічною кривиною». *Фізико-математичне моделювання та інформаційні технології* 37:123–127.
   DOI: 10.15407/fmmit2023.37.123
3. Стецюк, Петро, Ольга Хом'як, та Володимир Жидков. 2023б. Побудова S-подібної параметричної кривої та її застосування для проєктування зовнішнього контура сопла. В *Методи негладкої оптимізації в прикладних задачах*, відповідальні редактори Петро Стецюк та Марія Григорак, 258–292. Київ: ЛАЗУРИТ ПОЛІГРАФ.
4. Стецюк, Петро, Віктор Савицький, та Володимир Жидков. 2023. Пошук дефектів у регулярних 3D-структурах. В *Методи негладкої оптимізації в прикладних задачах*, відповідальні редактори Петро Стецюк та Марія Григорак, 230–257. Київ: ЛАЗУРИТ ПОЛІГРАФ.
5. Жидков, Володимир, Петро Стецюк, та Ольга Хом'як. 2025. Метод найменших квадратів та метод найменших модулів для пошуку дефектів в регулярних зображеннях. *Cybernetics and Computer Technologies* 1:32–42.
   DOI: 10.34229/2707-451X.25.1.3
6. Khomiak, Olga, Petro Stetsyuk, Volodymyr Zhydkov, and Luis Infante. 2023. "Using Optimization to Construct Naturally Parametrized Curve with Cubic Curvature". In *Smart Technologies in Urban Engineering (STUE 2022)*, Romanova T., Sukhonos M., Tsegelnyk Y. (eds). *Lecture Notes in Networks and Systems* 536:14–24.
   DOI: 10.1007/978-3-031-20141-7_2

7. Zhydkov, Volodymyr. 2025. Revealing defects in periodic structures. *Problems of Control and Informatics* 70(1):22–31.
DOI: 10.34229/1028-0979-2025-1-2

Публікації, що засвідчують апробацію матеріалів дисертації

1. Stetsyuk, Petro, Oleksandr Tkachenko, and Volodymyr Zhydkov. 2020. “Using Shor’s r-algorithm for building naturally parametrized curve having cubic curvature”. Proceedings of the 7-th International Conference on Control and Optimization with Industrial Application (COIA 2020), Baku, Azerbaijan, August 26-28. Vol. I. 389–391.
http://www.coia-conf.org/upload/editor/files/COIA2020_V1.pdf
2. Zhydkov, Volodymyr. 2024. “Finding Defects in Periodic Structures”. Матеріали XXIV міжнародної науково-практичної конференції «Інформаційні технології та безпека (ІТБ-2024)», Київ, Україна, Грудень 19. 38–41.
https://drive.google.com/file/d/1wsLVnueNRd62g-k179IHihuJNOYZqR9G/view
3. Zhydkov, Volodymyr, and Olha Khomiak. 2025. “Coordinate scaling for making naturally parametrized curve with cubic curvature”. Proceeding of the International Conference on Management and Control in Solving Engineer Problems (MaCoSEP 2025), Baku, Azerbaijan, March 13-15.
4. Stetsyuk, Petro, Viktor Savitsky, and Volodymyr Zhydkov. 2019. “Optimization Problems for Regular Image Reconstruction”. Book of Absracts of International Conference on Optimization and Equilibrium Problems, Dresden, Germany, July 31 – August 2. 45–46.

ЗМІСТ

ВСТУП

Розробка методів недиференційовної оптимізації є надзвичайно важливою областю досліджень в сучасній прикладній та обчислювальній математиці, оскільки в реальних практичних задачах дедалі частіше доводиться мати справу з функціями, які не є гладкими, градієнти яких мають розриви, або просто не мають похідної в деяких точках. У таких випадках класичні градієнтні методи, які ґрунтуються на існуванні та обчисленні похідних, втрачають свою ефективність або взагалі не застосовні. Такі ситуації часто виникають у машинному навчанні (наприклад, при використанні L_1-регуляризації або порогових функцій втрат), в економіці (де моделі включають негладкі функції витрат або доходів), в обробці та очищенні сигналів від шумів (особливо при розв'язанні задач розрідженої реконструкції), а також у галузях оптимального керування, робототехніки та біоінформатики.

Крім того, у багатьох задачах з прикладних областей цільова функція задається чисельно або експериментально (іноді як «чорна скринька»), тому похідна не просто невідома, а принципово недоступна. У таких умовах важливу роль відіграють спеціалізовані методи – субградієнтні, методи усереднення, проксимальні алгоритми, стохастичні та еволюційні підходи, які дозволяють здійснювати пошук екстремуму без необхідності знати точний градієнт. Їхній розвиток відкриває нові можливості для розв'язання складних задач в умовах обмеженої інформації, невизначеності та великої розмірності, що робить цю сферу досліджень особливо актуальною на тлі зростаючих обчислювальних потреб і міждисциплінарних викликів.

Однією з важливих та актуальних прикладних областей досліджень, де застосування методів недиференційовної оптимізації є затребуваним, є побудова кривих у натуральній параметризації. Ця задача знаходить своє застосування в багатьох галузях, зокрема в комп'ютерній графіці, моделюванні 3D-форм та об'єктів, робототехніці та механіці тощо. Така параметризація є ключовою для моделювання різних аеродинамічних та гідродинамічних поверхонь, оскільки

вона забезпечує рівномірний розподіл точок уздовж кривої відповідно до її довжини, що критично важливо для точного контролю форми, руху або деформації об’єкта. Натуральна параметризація дозволяє уникнути нерівномірностей у дискретизації, які можуть призводити до чисельних нестабільностей, втрати точності або візуальних артефактів, особливо при інтерполяції, аналізі кривини чи оптимізації форм. Завдяки цьому вона забезпечує фізично обґрунтоване та геометрично коректне представлення кривих, що робить її незамінною в точних обчислювальних та інженерних застосуваннях.

Принципово іншою, однак не менш важливою задачею, де методи негладкої оптимізації відіграють одну з ключових ролей, є пошук дефектів у різного роду структурах, зокрема регулярних та періодичних. Такі структури широко використовуються в матеріалознавстві, електроніці, фотоніці, нанотехнологіях і біомедичних системах, де навіть незначні порушення періодичності можуть суттєво впливати на фізичні властивості матеріалів або функціонування пристроїв. Виявлення дефектів дозволяє своєчасно діагностувати пошкодження, контролювати якість виробництва, виявляти приховані неоднорідності чи порушення симетрії, а також оптимізувати структури для заданих функцій. У зв’язку з цим розробка ефективних алгоритмів аналізу, зокрема за допомогою методів обробки сигналів, машинного навчання чи негладкої оптимізації, має велике значення для наукових досліджень та індустріальних застосувань.

Підсумовуючи вищесказане, можна зробити висновок, що розробка нових ефективних методів негладкої оптимізації для описаних задач дозволяє не лише підвищити якість та швидкість їхнього розв’язання, а й запобігти багатьом потенційним проблемам, виникнення яких безпосередньо впливає на якість та надійність інженерних пристроїв та конструкцій, матеріалів, різних аеродинамічних поверхонь та сопел тощо. Це і визначає безумовно високий рівень актуальності описаної проблематики та області загалом.

Актуальність теми. Більшість існуючих методів характеризуються тим, що потребують часто досить серйозного та об’ємного навчання або точної адаптації

під певний тип об’єкта, що вивчається, а також тип дефекту, який треба розпізнати, або клас кривої, яку треба оптимізувати. Також досі залишається багато питань, пов’язаних з надійністю та обґрунтованістю виявлення дефектів. Тому розробка достатньо загальних методів, спроможних виявляти дефекти заздалегідь невідомого типу, а також оптимізувати криві під складні критерії і при цьому вимагати мінімального навчання, ідеально – лише мінімальної інформації про досліджуваний об’єкт, є безумовно **актуальною проблемами**. Розробка саме таких методів і є одним із завдань даного дослідження.

Мета й завдання дослідження. Метою роботи є розробка методів негладкої оптимізації для побудови кривих у натуральній параметризації та виявлення дефектів.

Для досягнення мети дослідження поставлено такі *завдання*:

- розробити метод для побудови S-подібної кривої в натуральній параметризації з квадратичним розподілом кривини на основі методів негладкої оптимізації;
- розробити метод для побудови параметрично заданої кривої з кубічним розподілом кривини з використанням методів негладкої оптимізації;
- розробити метод для пошуку дефектів у регулярних 3D структурах шляхом аналізу зображень з використанням методів негладкої оптимізації;
- розробити метод для пошуку дефектів у періодичних зображеннях з використанням методів негладкої оптимізації;
- дослідити результативність побудови S-подібної параметричної кривої по її застосуванню у практичній задачі проєктування зовнішнього контуру сопла;
- дослідити ефективність побудови параметрично заданої кривої з кубічною кривиною на задачах що можуть реально зустрічатися в інженерній графіці;
- оцінити ефективність застосування побудованих методів для виявлення дефектів по зображеннях використовуючи реальні дані.

Об’єкт дослідження – методи негладкої оптимізації.

Предмет дослідження – методи негладкої оптимізації у задачах побудови параметрично заданих кривих та виявлення особливостей шляхом аналізу зображень.

Методи дослідження. В даній роботі використовуються побудова кривих шляхом застосування методів негладкої оптимізації, з подальшою оцінкою їхньої ефективності шляхом апробації на реальних або наближених до реальних задачах.

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

- *дістала подальшого розвитку* побудова натурально параметризованих кривих за заданими критеріями; досить гнучкі методи такого штибу розроблено *вперше*;
- вперше розроблено загальний метод виявлення дефектів на зображенні шляхом реконструкції фону;
- *вперше* побудовано алгоритмічний засіб, що дозволяє будувати параметрично задану криву з поліноміальним кубічним розподілом кривини за загальними умовами;
- *вперше* побудовано програмний засіб, який дозволяє проявляти дефекти, використовуючи аналіз одного зображення з мінімальною додатковою інформацією про досліджуваний об’єкт.

Практичне значення отриманих результатів. Побудова кривих у натуральній параметризації є давно відомим засобом розробки та моделювання аеродинамічних та гідродинамічних поверхонь через низку корисних властивостей. Незважаючи на це, цей метод дуже рідко застосовується через брак зручного інструментарію для застосування, позаяк зазвичай містить інтегродиференціальні обмеження, що вкрай ускладнює роботу з ним.

Методи та рішення викладені в даній роботі значно загальніші та спрощують роботу з подібними кривими, дозволяючи отримувати більш загальні та точніше оптимізовані до заданих критеріїв рішення. Запропоновані методи та алгоритми

негладкої оптимізації можуть використовуватися для профілювання турбінних лопаток авіаційних двигунів та інших технічних профілів. Ці методи позбавлені ряду недоліків, які характерні для широко відомих методів профілювання турбінних лопаток, як метод Безьє та метод LL-апроксимації. На відміну від останніх, вони забезпечують монотонність кривини отриманого профілю, що унеможливлює різку зміну значень кривини в профілі, яка призводить до відриву потоку, що є неприпустимим. Отримані результати дозволять скоротити час проектування та покращити геометричні та газодинамічні властивості поверхні завдяки тому, що модель поверхні отримується за допомогою профілів з заданими геометричними та газодинамічними властивостями (кривина тощо), зберігає заданий клас гладкості та враховує нормалі та кривину.

Метод виявлення дефектів, викладений в цій роботі, також досить загальний та потребує лише мінімальну інформацію про досліджуваний об’єкт на відміну від багатьох аналогічних за призначенням, що значно спрощує та зменшує ризик помилки при його практичному застосуванні.

Особистий внесок здобувача. Автором самостійно отримано основні результати дисертаційного дослідження. В опублікованих в співавторстві наукових працях здобувачем здійснено: у публікації (Стецюк, Хом’як та Жидков 2023а) – програмна реалізація алгоритму, проведення обчислювального експерименту та інтерпретація результатів; у статті (Khomiak et al. 2023) – програмна реалізація частини алгоритмів та проведення обчислювального експерименту; у розділі монографії (Стецюк, Хом’як та Жидков 2023б) – побудова та проведення обчислювальних експериментів, опис програмного забезпечення; у розділі монографії (Стецюк, Савицький та Жидков 2023) – програмна реалізація частини алгоритмів та їхнє тестування, інтерпретація результатів; у статті (Жидков, Стецюк та Хом’як 2025) – програмна реалізація алгоритму, його тестування та інтерпретація результатів.

Апробація результатів дисертації. Результати дисертації доповідались та обговорювались на:

- 7-th International Conference on Control and Optimization with Industrial Application (COIA 2020), 26–28 серпня 2020 року, Баку, Азербайджан;
- XXIV Міжнародній науково-практичній конференції «Інформаційні технології та безпека (ІТБ-2024)», 19 грудня 2024 року, Київ, Україна;
- International Conference on Management and Control in Solving Engineer Problems (MaCoSEP 2025), 13–15 березня 2025 року, Баку, Азербайджан;
- International Conference “Optimization and Equilibrium Problems (ICOEP 2019)”, 31 липня – 2 серпня 2019 року, Дрезден, Німеччина;
- засіданні відділу методів негладкої оптимізації Інституту кібернетики імені В.М. Глушкова НАН України, 25 квітня 2025 року.

Публікації. Основні наукові результати дисертаційної роботи у повній мірі викладено в 11 роботах, з яких: 4 статті опубліковано у наукових фахових виданнях України; 2 підрозділи опубліковано у колективній монографії; 2 статті опубліковано в матеріалах конференцій, які індексуються в наукометричних базах Web of Science/Scopus; 1 статтю та 2 тез доповідей опубліковано в матеріалах міжнародних наукових та науково-практичних конференцій.

Структура та обсяг дисертації. Дисертаційна робота складається зі вступу, п’яти розділів, загальних висновків, списку використаних літературних джерел, який містить 108 найменувань. Загальний обсяг дисертаційних досліджень викладено на 159 сторінках друкованого тексту, де обсяг основного тексту – 128 сторінок. Дисертація включає 29 рисунків, 13 таблиць та 1 додаток на 3 сторінках.

РОЗДІЛ 1. ОГЛЯД ЛІТЕРАТУРИ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1.1 Підходи та методи виявлення дефектів

Застосування негладких моделей та методів негладкої оптимізації набуває дедалі більшого поширення в сучасних областях науки й технологій. Часто демонструючи навіть кращі результати, як порівняти з відомими та поширеними градієнтними методами, негладкі методи та алгоритми, які на них базуються, дозволяють успішно розв’язувати складні, погано обумовлені оптимізаційні задачі досить значних розмірів, що є особливо важливою та надзвичайно поширеною умовою на практиці. Однак, необхідно умовою для успішного застосування таких методів є їхня адаптація, а також розробка різних модифікацій для конкретного класу задач.

Одним з важливих прикладів таких класів задач є алгоритмічне виявлення дефектів у структурованих зображеннях, що завжди є актуальним питанням для неруйнівного контролю якості без участі людини-оператора. Такі процеси набувають дедалі більшої актуальності у зв’язку з широким застосуванням композитних та стільникових матеріалів.

Позаяк при виготовленні та особливо позаштатному використанні таких матеріалів ймовірність появи дефектів є досить високою, проблема їх виявлення з використанням методів, що не погіршують їхню якість (неруйнівними), є досить актуальною та нагальною. На жаль, виявлення таких дефектів потребує спеціальних методів, оскільки вони найчастіше внутрішні, а також спеціальних методів аналізу через їхню неоднорідність. Дослідженню й розробці таких методів присвячена друга частина даної дисертаційної роботи.

В даній роботі розглядається метод побудови по наданим зображенням об’єктів з дефектами «ідеалізованого» зображення об’єкту яким би він був без дефектів шляхом апроксимації за допомогою методів оптимізації та порівняння з вихідних зображеннях. «Ідеалізоване» зображення будується за припущення, що структура об’єкту відома та у вихідному зображенні є надлишкова інформація

(наприклад, об’єкт має періодичну структуру). Такий метод досить універсальний та має широке застосування.

Щодо вже існуючих алгоритмів для дефектоскопії (розпізнавання дефектів за зображеннями) варто зазначити схожу ситуацію: їх існує досить багато, але жоден не можна назвати широко застосовним та надійним одночасно. Вони достатньо різноманітні, останнім часом для цієї задачі часто використовуються нейронні мережі (Роботишин та Маляр 2022), (Фундитус та Коноваленко 2024), (Perez, Tah, and Mosavi 2019), (Smagulova, Samaitis, and Jasiuniene 2024), (Wu and Zhou 2021).

Майже всі такі рішення потребують довгого навчання та ще й великого масиву експериментальних даних («навчального матеріалу»), який не завжди може бути доступним. Цікаво, що засоби, побудовані на цьому принципі, мають вади при інтерпретації зображень, подібні до тих, які можуть виникати у людей-операторів. Незважаючи на все вищесказане, подібні засоби мають досить високий попит, оскільки можуть автоматизувати процес дефектоскопії та значно зменшити кількість помилок.

Інший клас методів для виявлення дефектів за зображеннями можна умовно назвати «сегментацією»: це поділ на ділянки, які потім більш детально аналізуються (Sivabalan and Ghanadurai 2010), (Ben Gharsallah and Ben Braiek 2015), (Paul et al. 2023), (Meister et al. 2021), в тому числі з використанням нейромереж (Ashrafi et al. 2025). Варто зазначити, що цей клас методів є одним з найстаріших і ще довго не втратить актуальність, оскільки має рекурсивний характер і може легко комбінуватися з іншими методами виявлення дефектів. Він має найвищу ефективність при чітко окреслених границях дефектів. Цей метод потребує невеликих витрат на налагоджування для кожного конкретного типу зображення, і його можна вважати універсальним відносно інших.

Окремої уваги заслуговують алгоритми виявлення дефектів, засновані на аналізі текстури зображення. Вони з’явилися доволі давно, але розвивалися відносно повільно. На сьогодні вже існує достатньо велике розмаїття таких методів. Роботи, що стосуються використання виключно таких методів (Iyer and

Subbaih 2014), (Ghosh and Chakraborty 2017), так і використання допоміжних методів, наприклад, нейромережі (Branca et al. 1995), (Kumar and Shen 2002), (Pierre-Frédéric et. al 2022).

Досить часто для аналізу текстур застосовуються також Фур'є-перетворення: (Si and Kim 2024), (Hu, Wang, and Zhang 2015), (Mirmahdavi et al. 2013). Сегментація теж добре узгоджується з таким класом методів (Ivanchuk and Tumska 2020).

Варто також відмітити методи виявлення дефектів з аналізом зображення методом Фур'є, досить давно використовуються та дають чудові результати на текстурах, де спостерігається періодичність: (Wang and Zuo 2016), (Kunttu et al. 2006).

1.2 Побудова кривих у натуральній параметризації

Іншим важливим та актуальним класом задач є побудова кривих у натуральній параметризації, що є давно відомим і корисним допоміжним засобом розробки та моделювання різноманітного штибу аеродинамічних та гідродинамічних поверхонь через низку корисних властивостей, таких як гладкість, нескінченна диференційовність тощо. Незважаючи на це, цей метод дуже рідко застосований через брак досить зручного інструментарію для застосування, позаяк зазвичай містить інтегродиференціальні обмеження, що вкрай ускладнює роботу з ним. Загальною метою даної роботи є надання достатньо загального, гнучкого та ефективного алгоритмічного модуля, що дозволяє спростити побудову кривої в натуральній параметризації по заданим обмеженням.

Плоскі криві в натуральній параметризації характеризуються єдиним параметром – своєю довжиною (Рашевский 1956). Їхні координати є визначеними інтегралами тригонометричних функцій, аргументами яких є кути нахилу дотичних до кривих, які в свою чергу визначені як інтегральні залежності від функцій розподілу кривини. Якщо функції розподілу кривини є поліномами, то

аргументи тригонометричних функцій також будуть поліномами (їх степінь буде на одиницю вищим, ніж степінь полінома для функцій кривини).

Плоскі криві в натуральній параметризації з поліноміальним (лінійним, квадратичним, кубічним) законом розподілу кривини є хорошою альтернативою кривим Безьє при проектуванні кривих та поверхонь. Криві Безьє представляються кривими поліномами високих степенів, в силу чого їм властиві всі переваги та недоліки поліномів. Для кривих в натуральній параметризації таких недоліків поліномів як «хвилястість» кривини, легко уникнути за рахунок використання лінійного або квадратичного законів розподілу кривини.

Варто зазначити, що проблеми, які виникають при використанні натурально параметризованих кривих, призводять до того, що їх застосування зустрічаються досить рідко, переважно як елементи інженерної графіки, а також в деяких спеціальних застосуваннях, таких як (Rezaei and Zhan 2021), де натуральна параметризація кривої використовується для побудови кривих Шрама-Лоунера, що є досить нішевим та специфічним випадком.

Одним із найстаріших (більше 100 років) і водночас досить універсальних підходів до проєктування є методика NACA (National Advisory Committee for Aeronautics), за якою було створено набір стандартизованих аеродинамічних профілів, які й досі широко використовуються (Allen 2017).

Були проведені численні дослідження, спрямовані на виявлення оптимальних характеристик таких профілів за допомогою моделювання (Ganesh Ram 2014), (Körpe and Güzelbey 2023), оцінки їхньої ефективності для різнопланових застосувань (Izli, Vardar, and Kurtulmus 2007), (Mathew, Thakan, and Jeyan 2020), а також різноманітних методів оптимізації для різних умов (Fakhari and Mrad 2024), (Vucina, Lozina, and Pehnec 2008). Оскільки методика побудови профілів NACA не завжди дає оптимальний результат, були розроблені методи оптимізації, які використовують за основу базовий профіль цього набору та намагаються покращити його, деформуючи в певних межах (Tanabi et al. 2022), (Kallath et al. 2021).

Також застосовуються методи побудови профілів за допомогою інтерполяційних кривих загального призначення, таких як B-сплайни та криві Без'є (Ümütlü and Kiral 2022), (Sun, Li, and Fan 2020), (Rajnarayan, Ning, and Mehr 2018), (Jaiswal 2017).

Щодо причин вибору типу кривих для наближення профілей подібних до вищезгаданих, то параметрично задані криві з поліноміальним розподілом кривини мають ряд зручних властивостей для моделювання різний фізичних процесів та апроксимацію розв’язків інтегродиференціальних рівнянь, що стосуються реальних процесів.

Також, оскільки параметрично задана крива є інтерпретацією точкової частинки, що рухається вздовж кривої зі сталою швидкістю, то природнім застосуванням є, наприклад, відстежування траєкторій потоків та частинок, а також твірних направляючих їх елементів, таких як труби, лопаті турбін і пропелерів, профілі крил тощо. Зокрема, вони гладкі та нескінченно диференційовні, що відповідає властивостям реальних фізичних процесів, які проявляються в маніпуляції неперервними потоками. Природно, що найбільш поширене застосування параметрично заданих кривих нині знаходять у моделюванні аеродинамічних і гідродинамічних поверхонь та потоків.

Однак ті властивості, що роблять ці криві зручними для таких задач, водночас ускладнюють їхнє практичне застосування, оскільки в аналітичному вигляді вони не інтегруються, а точні розв’язки мають трансцендентні значення, що робить їх принципово необчислюваними точно на комп’ютері. Наближений розрахунок є обчислювально трудомістким, оскільки на кожному етапі апроксимаційного наближення необхідно виконувати чисельні обчислення тригонометричних функцій. Для ефективного практичного використання таких кривих необхідне їх багаторазове використання в межах кожної задачі, що робить існуючі методи наразі непрактичними.

Параметричні рівняння кривих у функції натурального параметра широко застосовуються в Національному університеті біоресурсів і природокористування (Київ), наприклад, для опису кінематични характеристик

руху матеріальної частинки по заданій траєкторії (Войтюк та Пилипака 2001). У роботах (Пилипака та Несвідомін, 1996), (Пилипака та Гнітецька, 2002) розроблено підходи до конструювання кривих за їх натуральними рівняннями.

Криві в натуральній параметризації активно використовуються в роботах миколаївської школи (Національний університет кораблебудування імені адмірала Макарова, ДП «Науково-виробничий комплекс газотурбобудування «Зоря − Машпроект») та ДП «Івченко-Прогрес» (Борисенко та ін. 2016), (Устенко 2009), (Устенко 2013). Задачі побудови кривих в натуральній параметризації зводяться до задач знаходження розв'язків систем нелінійних рівнянь, серед яких значна частина рівнянь є інтегральними (залежать від невідомих параметрів підінтегральних функцій та невідомих верхніх границь для визначеного інтегралу). У наведених вище роботах використано такий спосіб їх розв'язання: спочатку частина параметрів виражається через інші, при цьому залишаються лише інтегральні рівняння, що зменшує розмірність системи. Невідомі коефіцієнти знаходяться розв'язанням відповідної системи нелінійних інтегральних рівнянь методами Ньютона, Хука-Дживса та іншими. Недоліком такого підходу є неможливість усунення циклічних розв'язків та складність вибору початкових наближень, оскільки вони приховані за виконаними замінами.

В роботі (Borisenko et al. 2019) розглядається побудова профілю лопатки турбіни з використанням кривих у натуральній параметризації з поліноміальним розподілом кривини (кубічної та квадратичної). В цій роботі застосування натурально параметризованих кривих з поліноміальним розподілом кривини подається як граничний перехід від існуючих інженерних методів, що генерують криву зі шматків дуг з різними радіусами. В такому випадку відбувається згладжування, але контроль заданих характеристик кривої виявляється доволі складним.

Також, найбільш вдалі підходи для моделювання аеродинамічних та гідродинамічних профілів (Jaiswal 2017), (Han, Lee, and Choi 2012), накладають обмеження (Shen et al. 2016) (наприклад, за кривиною), що є природніми або

порівняно легко враховуваними параметрами для натурально параметризованих кривих, з чого випливає, що їх застосування буде логічним для цих задач.

Одним із способів генерування профілів є їхня побудова як похідних від вже існуючих (Chen et al. 2013), згладжуючи їх по кривині. Для такого підходу криві в натуральній параметризації з поліноміальною кривиною є майже ідеальними, за умови наявності достатньо загального та адаптованого оптимізаційного алгоритму, який міг би стабільно підбирати параметри кривої. У згаданій роботі це доводиться робити дуже загальним чином за допомогою генетичного алгоритму, які, в цілому, не відзначаються ані гарантованою надійністю, ані високою швидкістю. Тому питання достатньо ефективного оптимізаційного алгоритму, придатного для розв'язання таких задач, залишається досить актуальним. В цій же роботі показано, що такий підхід призводить до профілів з кращими аеродинамічними характеристиками.

Низка статей та праць конференцій (Fazil and Jayakumar 2011), (Peng et al. 2018), (Ümütlü and Kiral 2022) прямо вказують на те, що контроль кривини вздовж профілей та інші обмеження на кривину є одним з основних факторів, що впливає на аеродинамічну якість профілів.

1.3 Постановка задачі дослідження

Метою роботи є розробка методів негладкої оптимізації для побудови кривих у натуральній параметризації та виявлення дефектів у регулярних структурах. Більшість наявних підходів часто мають багато обмежень та умов, зокрема необхідність знати тип дефекту, який розпізнається, або клас кривої, яка оптимізується. Також ці методи потребують об'ємного та вартісного навчання з точки зору часу та обчислювальних потужностей. Обґрунтування ефективності виявлення тих чи інших дефектів теж залишається відкритим питанням. Тому розробка методів та підходів для виявлення дефектів заздалегідь невідомого типу, адаптації кривих під складні критерії за умови мінімального навчання є актуальною областю досліджень, чому і присвячена ця робота.

Для виконання поставленого завдання необхідно розв’язати такі *задачі*.

1. Розробити метод для побудови S-подібної параметрично заданої кривої з квадратичним законом розподілу кривини з використанням методів негладкої оптимізації.
2. Розробити метод для побудови параметрично заданої кривої з кубічним розподілом кривини з використанням методів негладкої оптимізації.
3. Розробити метод для пошуку дефектів у регулярних 3D структурах шляхом аналізу зображень з використанням методів негладкої оптимізації.
4. Розробити метод для пошуку дефектів у періодичних зображеннях з використанням методів негладкої оптимізації.
5. Дослідити результативність побудови S-подібної параметричної кривої по її застосуванню у практичній задачі проєктування зовнішнього контуру сопла.
6. Дослідити ефективність побудови параметрично заданої кривої з кубічною кривиною на задачах, що зустрічаються в інженерній графіці.
7. Оцінити ефективність застосування побудованих методів для виявлення дефектів по зображеннях, використовуючи реальні дані.

РОЗДІЛ 2. R-АЛГОРИТМИ І МЕТОД НЕГЛАДКИХ ШТРАФНИХ ФУНКЦІЙ

У розділі наведено опис програми **ralgb5a**, яка призначена для мінімізації гладких та негладких опуклих функцій (підрозділ 2.2). Програму розроблено на некомерційній мові GNU Octave. Програма **ralgb5a** реалізує модифікацію *r*-алгоритмів з адаптивним способом регулювання кроку (підрозділ 2.1).

У підрозділі 2.3 описано метод негладких штрафних функцій для задач опуклого програмування з урахуванням випадку, коли цільова функція визначена не на всьому просторі змінних. Метод негладких штрафних функцій можна використовувати в тому числі і для врахування обмежень, заданих у формі рівностей або нерівностей, в задачах математичного програмування.

2.1 Про r(α)-алгоритми та Octave-функцію ralgb5a

Octave функція ralgb5a реалізує подану нижче модифікацію *r*-алгоритмів Шора – субградієнтних методів з розтягом простору в напрямку різниці двох послідовних субградієнтів. Ця модифікація використовує адаптивний спосіб регулювання кроку в напрямі антисубградієнта у перетвореному просторі змінних. Програма розроблена на некомерційній мові GNU Octave.

Для мінімізації гладких та негладких опуклих функцій можна використовувати подану нижче модифікацію *r*-алгоритмів Шора – субградієнтних методів з розтягом простору в напрямку різниці двох послідовних субградієнтів (Шор та Журбенко 1971), (Шор 1979), (Shor 1985), (Shor 1998). Ця модифікація відома як $r(\alpha)$-алгоритми (Журбенко та Марчук 1975), де α – коефіцієнт розтягу простору ($\alpha > 1$), який є одним і тим самим на кожній ітерації *r*-алгоритмів. Опис $r(\alpha)$-алгоритмів наведемо згідно (Стецюк 2017б), (Стецюк 2019).

Що таке $r(\alpha)$-алгоритми? Розглянемо задачу пошуку точки мінімуму опуклої функції $f(x)$, $x \in R^n$. Мінімальне значення функції позначимо

$f^* = f(x^*)$, де $x^* \in X^*$. Будемо вважати, що множина мінімумів X^* є обмеженою, тобто виконується умова $\lim\limits_{\|x\|\to\infty} f(x) = +\infty$, яка забезпечує коректність адаптивного регулювання кроку в r-алгоритмах.

Означення. $r(\alpha)$-Алгоритмом мінімізації функції $f(x)$ називається ітеративна процедура знаходження послідовності n-вимірних векторів $\{x_k\}_{k=0}^{\infty}$ та послідовності $n\times n$-матриць $\{B_k\}_{k=0}^{\infty}$ за таким правилом:

$$x_{k+1} = x_k - h_k B_k \xi_k, \quad B_{k+1} = B_k R_\beta(\eta_k), \quad k = 0,1,2,\ldots, \tag{2.1}$$

де

$$\xi_k = \frac{B_k^T g_f(x_k)}{\left\|B_k^T g_f(x_k)\right\|}, \quad h_k \ge h_k^* = \arg\min_{h\ge 0} f(x_k - hB_k\xi_k), \tag{2.2}$$

$$\eta_k = \frac{B_k^T r_k}{\| B_k^T r_k \|}, \quad r_k = g_f(x_{k+1}) - g_f(x_k). \tag{2.3}$$

Тут x_0 – стартова точка; $B_0 = I_n$ – одинична $n\times n$-матриця; h_k^* – величина кроку до точки мінімуму функції $f(x)$ у напрямку нормованого антисубградієнта в перетвореному просторі змінних; $R_\beta(\eta) = I_n + (\beta - 1)\eta\eta^T$ – оператор стиснення простору субградієнтів у нормованому напрямку η з коефіцієнтом $\beta = 1/\alpha < 1$; $g_f(x_k)$ і $g_f(x_{k+1})$ – субградієнти функції $f(x)$ в точках x_k та x_{k+1}. Якщо на ітерації k процесу (2.1) – (2.3) виконані деякі критерії (умови) зупинки, то вважаємо $k^* = k$, $x_k^* = x_k$ і закінчуємо роботу алгоритму.

Коментар. На кожній ітерації r-алгоритмів реалізується субградієнтний спуск для опуклої функції $\varphi(y) = f(B_k y)$ в перетвореному просторі змінних $y = A_k x$, де $A_k = B_k^{-1}$. Дійсно, якщо обидві частини формули $x_{k+1} = x_k - h_k B_k \xi_k$ домножити зліва на матрицю A_k, то отримаємо

$$y_{k+1} = A_k x_{k+1} = A_k x_k - h_k \xi_k = y_k - h_k \frac{B_k^T g_f(x_k)}{\left\|B_k^T g_f(x_k)\right\|} = y_k - h_k \frac{g_\varphi(y_k)}{\left\|g_\varphi(y_k)\right\|}, \tag{2.4}$$

де вектор $g_\varphi(y_k) = B_k^T g_f(x_k)$ є субградієнтом функції $\varphi(y) = f(B_k y)$ в точці $y_k = A_k x_k$ простору змінних $y = A_k x$. Дійсно, субградієнт $g_f(x_k)$ для опуклої функції $f(x)$ задовольняє нерівності

$$f(x) \geq f(x_k) + (g_f(x_k))^T (x - x_k) \quad \forall x \in E^n,$$

звідки, здійснюючи заміну змінних $x = B_k y$, отримаємо нерівність

$$\varphi(y) \geq \varphi(y_k) + (B_k^T g_f(x_k))^T (y - y_k) = \varphi(y_k) + (g_\varphi(y_k))^T (y - y_k) \quad \forall y \in E^n$$

для субградієнта $g_\varphi(y_k)$ опуклої функції $\varphi(y) = f(B_k y)$.

Сімейство $r(\alpha)$-алгоритмів визначається коефіцієнтом розтягу простору $\alpha > 1$ (допускається $\alpha = \infty$, що відповідає $\beta = 0$) і послідовністю величин кроків $\{h_k\}_{k=0}^{\infty}$, які визначають ті два послідовні субградієнти $g_f(x_k)$ і $g_f(x_{k+1})$, розтягування за різницею яких (див. формулу (2.3)) зменшує степінь витягнутості функції в перетвореному просторі змінних. Вибір коефіцієнта $\alpha > 1$ та величин $\{h_k\}_{k=0}^{\infty}$ у поєднанні з вибором критеріїв зупинки визначають той чи інший варіант $r(\alpha)$-алгоритмів.

Регулювання кроку. Регулювання величини h_k задає певний спосіб реалізації одновимірного спуску в напрямку нормованого антисубградієнта функції $\varphi(y) = f(B_k y)$, який здійснюється в перетвореному просторі змінних $y = A_k x$ (див. формулу (2.4)). В r-алгоритмах використовуються два способи регулювання кроку (Шор та Стеценко 1989), (Стецюк, Бєлих, та Криворучко 2019).

При першому способі величина h_k вибирається з умови $h_k = h_k^*$ ($h_k \approx h_k^*$), що означає точний (чи наближений) одновимірний пошук мінімуму функції, який гарантує, що r-алгоритми є монотонними (майже монотонними) по функції, яка мінімізується. Таке регулювання кроку використовуються в теоретично обґрунтованих модифікаціях $r(\alpha)$-алгоритмів, коли коефіцієнт α можна вибирати досить великим. Це дозволило показати, що граничний варіант r-алгоритмів (при $\beta = 0$) є проективним методом спряжених градієнтів, та

обґрунтувати для граничного варіанту r -алгоритмів з відновленням квадратичну швидкість збіжності до точки мінімуму опуклої двічі неперервної диференційовної функції при деяких умовах гладкості і регулярності. Знайдено також такі умови для кусково-гладких опуклих функцій, при яких $r_\mu(\alpha)$ -алгоритм збігається до точки мінімуму.

Другий спосіб регулювання кроку називається адаптивним і використовується в практичних реалізаціях r -алгоритмів. Він полягає у тому, що величина h_k налаштовується (адаптується) в процесі виконання одновимірного спуску, який завершується, як тільки знайдено субградієнт, що утворює негострий кут з субградієнтом, який визначає напрямок одновимірного спуску (умова завершення спуску за напрямом). Налаштування величини кроку h_k здійснюється за допомогою чотирьох параметрів: $h_0 > 0$ – величина початкового кроку (використовується на першій ітерації, а на кожній наступній – уточнюється); q_1 ($q_1 \le 1$) – коефіцієнт зменшення кроку (якщо умова завершення спуску за напрямком виконується за перший крок); q_2 ($q_2 \ge 1$) – коефіцієнт збільшення кроку. Через кожні n_h кроків одновимірного спуску ($h_h > 1$) величина кроку збільшується в q_2 раз. Оскільки припускається, що $\lim\limits_{\|x\|\to\infty} f(x) = +\infty$, то після скінченної кількості кроків адаптивного спуску в напрямку нормованого антисубградієнта обов’язково виконується умова завершення спуску за напрямом.

Для адаптивного регулювання кроку коефіцієнт α не рекомендується вибирати великим. Він повинен бути узгодженим з коефіцієнтом q_1, тому що вони обидва, хоча і по різному, впливають на зменшення величини кроку h_k. В результаті визначаються ті два послідовні субградієнти, розтягування за різницею яких зменшує степінь витягнутості функції в перетвореному просторі змінних. Цим пояснюється пришвидшена збіжність $r(\alpha)$-алгоритмів для яружних (з витягнутими поверхнями рівня) функцій. Кількість ітерацій

$r(\alpha)$-алгоритму, необхідна для знаходження точки x_{k^*}, для якої $f(x_{k^*}) - f^* \le \varepsilon$, емпірично оцінюється як $k^* = O(n\log(1/\varepsilon))$, де n – кількість змінних.

Octave-функція ralgb5a (Стецюк, Бєлих, та Криворучко 2019)**.** Програма ralgb5a є спрощеною (для зручності використання) версією програми ralgb5 (Стецюк 2014, с. 383–386), в якій використовується метод (2.1) – (2.3). Тут абревіатура «b5» означає, що в основу програми покладено r-алгоритм у B-формі, де коректується $n \times n$-матриця B, а кожна ітерація методу (2.1) – (2.3) вимагає $5n^2$ арифметичних операцій множення, які визначають обчислювальну трудомісткість ітерації (операції додавання враховувати не будемо через їх малий вклад у трудомісткість ітерації). З них $3n^2$ операцій множення потрібно для обчислення векторів $B_k\xi_k$, $B_k^T g_f(x_k)$ і $B_k^T r_k$ (множення матриці на вектор), а $2n^2$ операцій множення вимагає однорангова корекція матриці $B_{k+1} = B_k R_\beta(\eta_k)$. Дійсно, корекція матриці B_{k+1} виконується за формулою

$$B_{k+1} = B_k R_\beta(\eta_k) = B_k(I_n + (\beta - 1)\eta_k\eta_k^T) = B_k + (\beta - 1)(B_k\eta_k)\eta_k^T,$$

звідки легко бачити, що обчислення вектора $\eta = B_k\eta_k$ вимагає n^2 операцій множення, і стільки ж операцій множення вимагає побудова тимчасової однорангової матриці $\eta\eta_k^T$.

В програмі **ralgb5a** зафіксовані два найбільш часто використовувані параметри $q_2 = 1.1$ і $n_h = 3$. При цьому величина початкового кроку для чергової ітерації може максимально збільшуватися в 10^6 раз. У програмі ralgb5a використовується параметр intp (interval for print), який забезпечує друк інформації про хід процесу мінімізації через кожні intp ітерацій. Цей параметр дозволяє скоротити протокол роботи програми при мінімізації функції для сотень і тисяч змінних, коли кількість ітерацій оцінюється тисячами і десятками тисяч. Програма використовує параметри ε_x і ε_g для зупинки ітераційного процесу в точці $x_{k^*} \in [x_k, x_{k+1}]$, де $\|x_{k+1} - x_k\| \le \varepsilon_x$ (зупинка за аргументом), або $\|g_f(x_{k^*})\| \le \varepsilon_g$ (зупинка за нормою градієнта, яка використовується для гладких функцій). Використовуються також стандартна зупинка, якщо перевищено задану

максимальну кількість ітерацій maxitn, та аварійна зупинка, яка сигналізує про те, що або функція $f(x)$ не є обмеженою знизу, або початковий крок h_0 занадто малий, і його треба збільшити.

Якщо ітераційний процес запускається зі стартової точки x_0, то параметри $r(\alpha)$-алгоритму рекомендується вибирати наступними: $\alpha \in [2,4]$, $q_1 = 1.0$ (для негладких функцій), $q_1 = 0.8 \div 0.95$ (для гладких функцій), $h_0 \approx \|x_0 - x^*\|$ – оцінка відстані від стартової точки x_0 до точки мінімуму x^*. Як правило, використовуються такі параметри зупинки: $\varepsilon_x \approx 10^{-6}$, $\varepsilon_g \approx 10^{-12}$, maxitn $\approx 20n$. Тут параметр ε_g використовується для гладких функцій, а параметр ε_x для негладких функцій. Якщо програма ralgb5a завершує роботу за умовою $\|x_{k+1} - x_k\| \le \varepsilon_x = 10^{-8}$, то цього цілком достатньо, щоб на 14-15 порядків зменшити різницю між знайденим рекордним значенням квадратичної функції f_r і її мінімальним значенням f^*.

2.2 Опис програми ralgb5a

Нижче наведемо короткий опис програми ralgb5a, який буде включати короткий код Octave-функції ralgb5a та його застосування до тестового прикладу sabs(100, 1.2), який полягає у мінімізації кусочно-лінійної функції

$$f(x) = \sum_{i=1}^{100} (1.2)^{i-1} |x_i - 1|, \quad f^* = f(x^*) = 0, \quad x^* = (1,1,\ldots,1)^T, \qquad (2.5)$$

де $|a|$ – абсолютна величина числа a. Функція (2.4) є яружною, так як коефіцієнти при $|x_i - 1|$, $i = 1,\ldots,100$ утворюють геометричну прогресію з показником $q = 1.2$, де мінімальний коефіцієнт дорівнює $(1.2)^0 = 1$, а максимальний – $(1.2)^{99} \approx 6.9015\text{e+}07$.

Код програми ralgb5a. Octave-функція ralgb5a знаходить x_r^* – наближення до точки мінімуму опуклої функції $f(x)$ від n змінних. Програма використовує

Octave-функцію `function [f, g] = calcfg (x)`, яка обчислює значення функції $f = f(x)$ і її субградієнта $g = g_f(x)$ в точці x. Ця програма готується користувачем та може мати довільне ім'я, яке підтримує синтаксис Octave. Код програми, що включає і короткі англомовні коментарі для вхідних та вихідних параметрів, наведено нижче.

```
# Input parameters:
#    calcfg - name of the function calcfg(x) for calculation of f and g
#    x - the starting point, x0(1:n) (it is modified in the program)
#    alpha - the value of coefficient of space dilation
#    h0, q1 - parameters of the adaptive step adjustment
#    epsx, epsg, maxitn - stop parameters
#    intp - print information every intp iteration
# Output parameters:
#    xr - a minimum point, which was found by the program, xr(1:n)
#    fr - the value of the function f at the point xr
#    itn - the number of iterations used by the program
#    nfg - the number of function calcfg calls
#    istop - exit code (2 = epsg, 3 = epsx, 4 = maxitn, 5 = error)
function [xr,fr,itn,nfg,istop] = ralgb5a(calcfg,x,alpha,h0,q1,  #row001
                                         epsg,epsx,maxitn,intp);
itn = 0; B = eye(length(x)); hs = h0; lsa = 0; lsm = 0;         #row002
xr = x; [fr,g0] = calcfg(xr); nfg = 1;                          #row003
printf("itn %4d  f%15.6e  fr%15.6e  nfg %4d\n",itn,fr,fr,nfg);  #row004
if(norm(g0) < epsg) istop = 2;  return; endif                   #row005
for (itn = 1:maxitn)                                            #row006
   dx = B * (g1 = B' * g0)/norm(g1);                            #row007
   d = 1; ls = 0; ddx = 0;                                      #row008
   while (d > 0)                                                #row009
      x -= hs * dx; ddx += hs * norm(dx);                       #row010
      [f, g1] = calcfg(x); nfg ++;                              #row011
      if (f < fr)  fr = f; xr = x;  endif                       #row012
      if(norm(g1) < epsg) istop = 2;  return; endif             #row013
      ls ++; (mod(ls,3) == 0) && (hs *= 1.1);                   #row014
      if(ls > 500) istop = 5;  return; endif                    #row015
      d = dx' * g1;                                             #row016
   endwhile                                                     #row017
   (ls == 1) && (hs *= q1);  lsa=lsa+ls; lsm=max(lsm,ls);       #row018
   if(mod(itn,intp)==0)                                         #row019
     printf("itn %4d  f %14.6e  fr %14.6e", itn, f, fr);        #row020
     printf("  nfg %4d  lsa %3d  lsm %3d\n", nfg, lsa, lsm);    #row021
     lsa=0; lsm=0;                                              #row022
   endif                                                        #row023
   if(ddx < epsx) istop = 3;  return; endif                     #row024
   xi = (dg = B' * (g1 - g0) )/norm(dg);                        #row025
   B += (1 / alpha - 1) * B * xi * xi';                         #row026
   g0 = g1;                                                     #row027
endfor                                                          #row028
istop = 4;                                                      #row029
endfunction                                                     #row030
```

При мінімізації негладких функцій рекомендується вибирати: `alpha=2÷3`, `h0=1.0`, `q1=1.0`. При мінімізації гладких функцій рекомендується використовувати `q1=0.8÷0.95`. При правильному підборі цих параметрів можна значно скоротити кількість ітерацій для виконання одних і тих же критеріїв зупинки. Це залежить від конкретного виду функції, що мінімізується, ступеня її яружності і масштабу змінних.

Тестовий приклад. Для тестового прикладу використовується кусково-лінійна функція (2.5). Обчислення значення функції (2.5) та її субградієнта реалізовано такою Octave-функцією:

```
function [f,g] = sabs(x)
global w
temp=x-ones(length(x),1); f=sum(abs(w.*temp)); g=w.*sign(temp);
endfunction
```

для якої значення коефіцієнтів w встановлюються за допомогою операторів

```
global w
n=100; temp=[0:(n-1)]'; w=1.2.**temp;
```

Ітераційний процес запускається зі стартової точки $x_0 = (0,0,\ldots,0)^T$, для якої значення функції $f(x_0) = 4.140899\text{e+}08$. Параметри $r(\alpha)$-алгоритма вибираються такими: $\alpha = 4$ (рекомендується $\alpha \in [2,4]$), $q_1 = 1.0$ (рекомендується для негладких функцій), $h_0 = 10$ (дорівнює $\|x_0 - x^*\|$ – відстані від стартової точки x_0 до точки мінімуму x^*). Використовуються параметри зупинки: $\varepsilon_x = 10^{-8}$, $\varepsilon_g = 10^{-12}$, maxitn=5000. Тут параметр ε_g ролі не відіграє (він використовується для гладких функцій), а параметр ε_x вибраний таким, щоб программа ralgb5a закінчувала роботу за критерієм $\|x_{k+1} - x_k\| \le 10^{-8}$, чого цілком достатньо, щоб на 14-15 порядків зменшити різницю між знайденим рекордним значенням функції f_r та її мінімальним значенням $f^* = 0$.

Для вказаних параметрів головна Octave-програма має такий вигляд:

```
global w
n=100; temp=[0:(n-1)]'; w=1.2.**temp; # w(1,1), w(100,1),
x = zeros(n,1); alpha = 4.0, h0 = 10.0, q1 = 1.0,
epsx = 1.e-8, epsg = 1.e-12, maxitn = 5000, intp=500;
[xr,fr,itn,nfg,istop] =
ralgb5a(@sabs,x,alpha,h0,q1,epsg,epsx,maxitn,intp);
printf("itn %4d  fr %23.15e  istop %d  nfg %4d\n", itn, fr, istop,nfg);
```

```
dx = norm(xr-ones(n,1)),
```

Протокол роботи програми. Обчислення проводились на комп'ютері Pentium 3 GHz в системі Windows 7/32 за допомогою GNU Octave версії 5.1.0. Протокол роботи програми ralgb5a при `intp=500` має вигляд:

```
alpha =  4  h0 =  10  q1 =  1
epsx = 1.0000e-008  epsg = 1.0000e-012  maxitn =  5000
itn    0  f  4.140899e+008  fr  4.140899e+008  nfg    1
itn  500  f  1.718525e+003  fr  1.273433e+003  nfg  532  lsa 531  lsm   4
itn 1000  f  1.409472e+000  fr  1.192802e+000  nfg 1032  lsa 500  lsm   1
itn 1500  f  1.258921e-003  fr  1.258921e-003  nfg 1532  lsa 500  lsm   1
itn 2000  f  1.422859e-006  fr  1.224438e-006  nfg 2032  lsa 500  lsm   1
itn 2046  fr  6.340398755873688e-007  istop 3  nfg 2078
dx = 1.9497e-008
```

Тут вхідні параметри $r(\alpha)$-алгоритма зібрано в два перших рядки. Із протоколу видно, що кількість ітерацій відповідає емпіричній оцінці, тобто на одну ітерацію в середньому затрачено $2078/2046<3$ обчислень значення функції (2.5) та її субградієнта. Якщо вибрати параметр $q_1=0.95$, то кількість ітерацій зменшується до 920 при 1539 викликах функції sabs.

Програма ralgb5a працює під управлінням тих операційних систем, які допускають установку Open Source-пакета для математичних обчислень GNU Octave. Програма використовує версії Octave 3.0.0 і вище. Для неї не потрібно ніяких спеціальних конфігурацій комп’ютера.

2.3 Метод негладких штрафних функцій

Метод штрафних функцій з негладкою функцією штрафу використовується для врахування обмежень, заданих у формі рівностей або нерівностей, в умовних задачах математичного програмування. Він дозволяє звести умовну задачу математичного програмування на мінімум до задачі безумовної мінімізації негладкої функції. Для розв’язання задач опуклого програмування можна використовувати сучасні методи негладкої оптимізації у сполученні з методом негладких штрафних функцій (Еремин 1967), (Шор 1979), (Пшеничный 1983), (Shor 1998), (Polyakova 2000). Розглянемо метод негладких штрафних функцій згідно з монографією (Шор 1979, с. 188–189) та статтею (Bertsekas 1975).

Нехай розглядається задача опуклого програмування

$$\min_{x\in R^n} f_0(x) \quad \text{при} \quad f_i(x)\le 0, \quad i=1,\ldots,m. \tag{2.6}$$

Визначимо функцію

$$S(x)= f_0(x)+\sum_{i=1}^{m} p_i\left[f_i(x)\right], \tag{2.7}$$

де $p_i(t)$, $i=1,\ldots,m$, задовольняють умовам опуклості, $p_i(t)=0$ для $t<0$, $p_i(t)\ge 0$ для $t>0$. Нехай існує x^* – оптимальний розв'язок задачі $\inf\limits_x S(x)$.

Теорема 2.1. Для того, щоб x^* був оптимальним розв'язком задачі (2.6), необхідно, щоб

$$\lim_{t\to+0}\frac{p_i(t)}{t}\ge \overline{y}_i, \quad i=1,\ldots,m,$$

де $\overline{y}=(\overline{y}_1,\ldots,\overline{y}_m)$ – деякі множники Лагранжа задачі (2.6).

Для того, щоб (2.6) і (2.7) мали однакову множину оптимальних розв'язків, достатньо, щоб $\lim\limits_{t\to 0}\frac{p_i(t)}{t}>\overline{y}_i$.

Найпростіший варіант негладкої функції штрафу має вигляд

$$p(t)=\begin{cases}0, & t\le 0,\\ ct, & t>0.\end{cases} \tag{2.8}$$

З теореми 2.1 випливає, що для того, щоб задача мінімізації $S(x)$ була еквівалентна задачі (2.6) при

$$p_i(t)=\begin{cases}0, & t\le 0,\\ c_i t, & t>0.\end{cases} \quad i=1,\ldots,m, \tag{2.9}$$

достатньо вибрати $c_i>\overline{y}_i$.

Великий інтерес представляє штрафна функція

$$T(x)= f_0(x)+p\left[\max_{i\in 1,m} f\left(x_i\right)\right],$$

де $p(t)$ обчислюється за формулою (2.8). Легко бачити, що якщо $c>\sum\limits_{i=1}^{m}\overline{y}_i$, то

$$p\left(\max_{i\in 1,m} f\left(x_i\right)\right) \geq \sum_{i=1}^{m} p_i\left[f_i(x)\right],$$

де p_i обчислюється за формулами (2.9), $c = \sum_{i=1}^{m} c_i$; $c_i > \overline{y_i}$. Таким чином, при $c > \sum_{i=1}^{m} \overline{y_i}$ задача мінімізації функції $T(x)$ еквівалентна задачі (2.6).

При використанні для розв'язання спеціальних умовних задач математичного програмування негладкої штрафної функції у формі функції максимуму можна застосовувати теорему Б.М. Пшеничного про точні штрафні функції (Пшеничный 1983).

Нехай для врахування обмежень у задачі (2.6) використовується негладка штрафна функція у формі функції максимуму:

$$\Phi_N(x) = f_0(x) + N \times \max\{0, f_1(x),\ldots, f_m(x)\}. \qquad (2.10)$$

Розглянемо сімейство параметричних задач

$$V(z) = \inf\left\{ f_0(x) : f_i(x) \leq z_i; i = 1,\ldots,m\right\},$$

залежне від вектора $z \in R^m$. Очевидно, що

$$V(0) = \inf\left\{ f_0(x) : f_i(x) \leq 0; i = 1,\ldots,m\right\}$$

збігається з розв'язком задачі (2.6). Справедлива така теорема (Пшеничный 1983).

Теорема 2.2. Нехай $\inf\limits_{\lambda>0} \dfrac{V(\lambda e) - V(0)}{\lambda} = -L > -\infty$, де e – m-вимірний вектор, усі компоненти якого дорівнюють одиниці. Якщо $N > L$, то тоді точки мінімуму задач $V(0)$ і $\inf\limits_{x\in M} \Phi_N(x)$, де $\Phi_N(x)$ визначається за формулою (2.10), співпадають.

Теорема 2.2 дає інструмент для встановлення точного значення штрафного множника при використанні для розв'язання задачі (2.6) негладкої штрафної функції у формі функції максимуму.

Метод негладких штрафних функцій можна успішно застосовувати для отримання локальних екстремумів в задачах нелінійного програмування загального вигляду:

знайти

$$\min_{x \in R^n} f(x) \tag{2.11}$$

за обмежень виду

$$\varphi_i(x) \le 0, \quad i = 1,\ldots,m, \quad \psi_j(x) = 0, \quad j = 1,\ldots,l, \tag{2.12}$$

шляхом заміни її задачею мінімізації функції

$$S(x) = f(x) + \sum_{i=1}^{m} c_i \max(0, \varphi_i(x)) + \sum_{j=1}^{l} d_j \left|\psi_j(x)\right|. \tag{2.13}$$

При достатньо великих c_i та d_i в умовах існування множників Лагранжа в задачі (2.11) – (2.12) задача мінімізації $S(x)$ за формулою (2.13) еквівалентна задачі (2.11) – (2.12).

Якщо обмеження-рівності відсутні в задачі нелінійного програмування (2.11) – (2.12), то отримаємо її частинний випадок:

знайти

$$\min_{x \in R^n} f(x) \tag{2.14}$$

за обмежень виду

$$\varphi_i(x) \le 0, \quad i = 1,\ldots,m. \tag{2.15}$$

Негладкі штрафні функції для задач (2.14) та (2.15) мають такий вигляд

$$S_1(x) = f(x) + \sum_{i=1}^{m} c_i \max(0, \varphi_i(x)) \text{ та } S_2(x) = f(x) + C \max(0, \max_{i=1,\ldots,m} \varphi_i(x)), \tag{2.16}$$

де $C > 0$ та $c_i > 0,\ i = 1,\ldots,m$ – штрафні коефіцієнти.

При використанні штрафних функцій $S_1(x)$ та $S_2(x)$ за формулами (2.16) проблеми виникають, якщо функція $f(x)$ та функції $\varphi(x)$, $i = 1,\ldots,m$ визначені не на всьому просторі. Тут можна використати продовження функцій з допустимої області задачі на весь простір змінних (Лаптин и Лиховид 2010), (Лаптин 2011), (Лаптин и Бардадым 2011), (Лаптин 2015). Для обґрунтування таких підходів розроблено процедури продовження функцій, які зберігали б опуклість побудованої задачі, якщо початкова задача з обмеженнями була опуклою. Опукла задача безумовної оптимізації, що формується, залежить від

одного числового параметра, значення якого визначається у ході роботи оптимізаційного алгоритму.

Розроблені процедури були використані Лаптіним Ю.П. для розв'язання задач оптимізації конструктивних рішень енергетичних парових котлів та інших складних технічних об'єктів (Лаптин, Медведев, и Волковицкая 1994), (Лаптин и Журбенко 2002), (Лаптин и др. 2003), де істотною проблемою була обмеженість областей визначення використовуваних функцій.

Використання подібних процедур продовження функцій представляє проблеми для неопуклих задач і в тому випадку, якщо область визначення функції $f(x)$ задається неявно. Наприклад, $f\left(x\right)$ визначена при тих значеннях x, які задовольняють систему обмежень (2.15). В цьому випадку можна використати градієнтне поле, побудоване за такою ж схемою, що і для методу еліпсоїдів (Шор 1977), (Nemirovsky and Yudin 1983), (Fischer, Khomiak, and Stetsyuk 2023). Вперше така схема була використана в модифікації *r*-алгоритму для знаходження двоїстих оцінок в квадратичних екстремальних задачах, де функція, що максимізувалася визначена на сімействі невід'ємно визначених симетричних матриць (Шор и Стецюк 1997).

Для задачі опуклого програмування (2.14) – (2.15) суть цієї схеми такий. Позначимо $g_f\left(x\right)$ – субградієнт функції $f\left(x\right)$, $\varphi(x)=\max\limits_{i=1,..,m}\varphi_i(x)$, $g_\varphi\left(x\right)$ – субградієнт функції $\varphi\left(x\right)$. Градієнтне поле $g(x)$ визначимо наступним чином:

$$g\left(x\right)=\begin{cases} g_f\left(x\right), & \text{якщо} \quad \varphi\left(x\right)\le 0, \\ g_\varphi\left(x\right), & \text{якщо} \quad \varphi\left(x\right)>0, \end{cases} \tag{2.17}$$

і будемо вважати, що йому відповідає штрафна функція $S\left(x\right)$:

$$S\left(x\right)=\begin{cases} f\left(x\right), & \text{якщо} \quad \varphi\left(x\right)\le 0, \\ +\infty, & \text{якщо} \quad \varphi\left(x\right)>0. \end{cases} \tag{2.18}$$

Для оракула, який обчислює значення функції та її субградієнта, можна використовувати формули (2.18), (2.17). Для нього субградієнтний процес можна

інтерпретувати як використання дуже великого значення коефіцієнта C для штрафної функції $S_2(x) = f(x) + C\max(0, \max\limits_{i=1,\ldots,m} \varphi_i(x))$.

Використання негладких штрафних функцій приводить до задач мінімізації функцій яружного виду. Для їх розв'язання рекомендується використовувати алгоритми з розтягом простору в напрямку різниці двох послідовних градієнтів, або так звані r-алгоритми. Вище наведений опис варіанта r-алгоритму з адаптивним кроком (програма **ralgb5a**), який найчастіше використовується при розв'язанні подібних задач.

2.4 Висновки до другого розділу

У розділі наведено опис програми **ralgb5a** на некомерційній мові GNU Octave. Вона призначена для мінімізації гладких та негладких опуклих функцій та реалізує модифікацію *r*-алгоритмів з адаптивним способом регулювання кроку. Описано метод негладких штрафних функцій для задач опуклого програмування з урахуванням випадку, коли цільова функція визначена не на всьому просторі змінних. Метод негладких штрафних функцій можна використовувати в тому числі і для врахування обмежень, заданих у формі рівностей або нерівностей, в задачах математичного програмування.

Матеріал розділу використано у звіті (Стецюк, Жидков та Хом’як 2023в).

РОЗДІЛ 3. ПОБУДОВА S-ПОДІБНОЇ ПАРАМЕТРИЧНОЇ КРИВОЇ З КВАДРАТИЧНОЮ КРИВИНОЮ

У розділі описано математичну модель, алгоритм та програмне забезпечення для задачі побудови S-подібної кривої, яка проходить через дві задані точки із заданими кутами нахилу дотичних у них та забезпечує заданий кут нахилу дотичної в точці із заданою абсцисою. Для керування точкою перегину S-подібної кривої з квадратичним законом розподілу кривини в натуральній параметризації використовується кут нахилу дотичної в точці із заданою абсцисою. Алгоритм базується на модифікації градієнтного методу з розтягом простору в напрямі різниці двох послідовних узагальнених градієнтів. Наведено опис програмної реалізації та обчислювальних експериментів, які показали ефективність розробленого алгоритму для проєктування зовнішнього контуру сопла типу Франкля.

3.1 Математична модель задачі та її властивості

Матеріал підрозділу базується на роботі (Стецюк та ін. 2020). Наведемо опис математичної модель, алгоритм та програмне забезпечення для задачі побудови S-подібної кривої, яка проходить через дві задані точки із заданими кутами нахилу дотичних у них та забезпечує заданий кут нахилу дотичної в точці із заданою абсцисою. Для керування точкою перегину S-подібної кривої з квадратичним законом розподілу кривини в натуральній параметризації використовується кут нахилу дотичної в точці із заданою абсцисою. Алгоритм основано на модифікації методу з розтягом простору в напрямі різниці двох послідовних узагальнених градієнтів. Обчислювальні експерименти показали ефективність розробленого алгоритму та його програмної реалізації для проектування фрагментів зовнішнього контуру сопла Франкля (див. рис. 3.1).

Алгоритм використовується для побудови зовнішнього контуру сопла типу Франкля за допомогою двохланкових S-подібних кривих, отриманих з

використанням алгоритмів натуральної параметризації та квадратичного закону розподілу кривини. Зовнішній контур сопла типу Франкля складається з двох S-подібних кривих: перша крива – для дозвукової частини (де сопло звужується) та друга крива – для надзвукової частини (де сопло розширюється). Керування точками перегину цих кривих дає можливість вибирати необхідні форми для фрагментів зовнішнього контуру дозвукової та надзвукової частин сопла.

Вихідними даними для побудови контуру у вигляді плоскої S-подібної кривої є координати точок (x_1, y_1), (x_2, y_2) та кути нахилу дотичних у них (φ_1, φ_2), а також кут φ_p – кут нахилу дотичної в точці (x_p, y_p) та її абсциса x_p, які будуть використовуватися для керування точкою перегину S-подібної кривої. Для надзвукового фрагменту контуру сопла Франкля задано точки та кути нахилу в них наведено на рисунку 3.1.

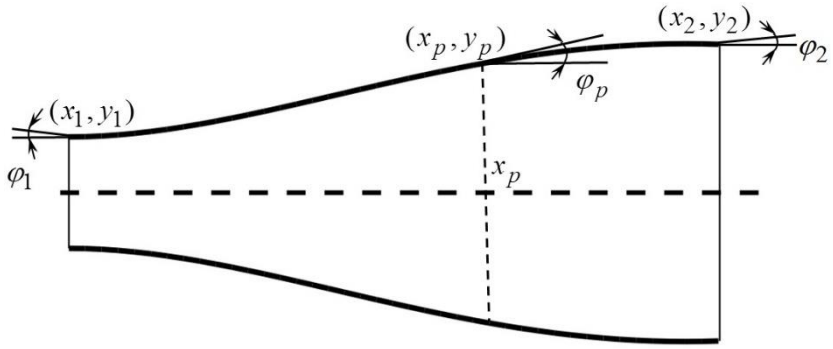


Рисунок 3.1 – Вихідні дані для зовнішнього контуру сопла Франкля у надзвуковій частині

Задача полягає в тому, щоб з'єднати точки (x_1, y_1) та (x_2, y_2) кривою лінією в натуральній параметризації, де кривина $k(s) = as^2 + bs + c$ є квадратичною функцією від s – довжини кривої, щоб забезпечити в точках (x_1, y_1) та (x_2, y_2) задані значення кутів нахилу дотичних φ_1 та φ_2, а в точці з абсцисою x_p, для якої $x_1 < x_p < x_2$, забезпечити кут рівний φ_p. Кути φ_1, φ_2 та φ_p вимірюються в радіанах.

Якщо контур представляється у вигляді S-подібної кривої, то квадратична кривина $k(s)=as^2+bs+c$ забезпечує єдиність розв'язку задачі. Задаючи значення абсциси x_p та кут нахилу дотичної φ_p, отримуємо можливість керувати точкою перегину отриманої S-подібної кривої, що дозволяє автономно моделювати той чи інший вигляд фрагментів зовнішнього контуру сопла Франкля у дозвуковій та надзвуковій його частинах. Для дозвукової частини сопла фрагмент зовнішнього контуру має вигляд, який є «дзеркальним» відображенням фрагменту зовнішнього контуру у надзвуковій частині. Для надзвукової частини сопла абсциса x_p буде розміщуватися ближче до її закінчення, а для дозвукової частини сопла абсциса x_p буде розміщуватися ближче до її початку.

Для кривої $x(s)$, $y(s)$, $s>0$ в натуральній параметризації кут нахилу дотичної $\varphi(s)$ в точці $(x(s),y(s))$ визначається за формулою

$$\varphi(s)=\varphi(0)+\int_0^s k(s)ds=\varphi(0)+\frac{as^3}{3}+\frac{bs^2}{2}+cs, \qquad (3.1)$$

а координати $x(s)$ та $y(s)$ – за формулами

$$x(s)=x(0)+\int_0^s \cos\varphi(s)ds, \quad y(s)=y(0)+\int_0^s \sin\varphi(s)ds. \qquad (3.2)$$

Тут кривина $k(s)=as^2+bs+c$ задається квадратичною функцією від s – довжини кривої, де a, b, c – задані коефіцієнти.

Нехай S – довжина кривої від точки (x_1,y_1) до точки (x_2,y_2), а s_p – довжина кривої від точки (x_1,y_1) до точки (x_p,y_p). Знаходженню параметрів кривини a, b, c та довжин S, s_p відповідає система з п’яти нелінійних рівнянь (Стецюк, Ткаченко, та Грицай 2020):

$$x_2=x_1+\int_0^S \cos\left(\varphi_1+\frac{as^3}{3}+\frac{bs^2}{2}+cs\right)ds, \qquad (3.3)$$

$$y_2 = y_1 + \int_0^S \sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds, \tag{3.4}$$

$$\varphi_2 = \varphi_1 + \frac{aS^3}{3} + \frac{bS^2}{2} + cS, \tag{3.5}$$

$$x_p = x_1 + \int_0^{s_p} \cos\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds, \tag{3.6}$$

$$\varphi_p = \varphi_1 + \frac{as_p^3}{3} + \frac{bs_p^2}{2} + cs_p. \tag{3.7}$$

Система (3.3) – (3.7) має п’ять невідомих: a, b, c – три коефіцієнти квадратичної функції, S – загальна довжина кривої, s_p – довжина ділянки кривої до точки з відомою абсцисою. Система включає п’ять нелінійних рівнянь, серед яких рівняння (3.3), (3.4) та (3.6) є інтегральними та залежать від невідомих параметрів підінтегральних функцій та невідомих верхніх границь для визначеного інтегралу. Інтегральні рівняння (3.3) та (3.4) зв’язують між собою точки (x_1, y_1) та (x_2, y_2) за формулами (3.2). Нелінійне рівняння (3.5) за формулою (3.1) забезпечує потрібний кут φ_2 в точці (x_2, y_2), який визначається за заданим кутом φ_1 в точці (x_1, y_1). Інтегральне рівняння (3.6) зв'язує між собою координати x_1 та x_p за першою формулою із (3.2), а нелінійне рівняння (3.7) забезпечує кут рівний φ_p в точці з абсцисою x_p за формулою (3.1).

Кут нахилу дотичної φ_p в точці з відомою абсцисою x_p будемо використовувати для керування точкою перегину S-подібної кривої, в якій кривина є рівною нулю. Зауважимо, що ордината y_p не використовується як невідома змінна. Її значення обчислюється за формулою $y_p = y_1 + \int_0^{s_p} \sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds$, де s_p – знайдена довжина ділянки кривої до точки з відомою абсцисою.

Виділимо дві властивості системи (3.3) – (3.7), які потрібно враховувати при знаходженні її розв'язків, пов'язаних з проектуванням зовнішнього контуру сопла Франкля у вигляді двохланкових S-подібних кривих.

Перша властивість: у загальному випадку система (3.3) – (3.7) має багато розв'язків. Наприклад, у таблиці 3.1 наведено чотири розв'язки для наступних вихідних даних:

$$x_1 = 0,\ y_1 = 2.46,\ \varphi_1 = 0; \quad x_2 = 1,\ y_2 = 2.75,\ \varphi_2 = 0;$$
$$x_p = 0.7, \varphi_p = 0.209440 = 12^\circ. \tag{3.8}$$

Графіки кривих для цих розв'язків наведені на рисунку 3.2 (криві для першого і другого розв'язків) та на рисунку 3.3 (криві для третього та четвертого розв'язків).

Таблиця 3.1 – Чотири розв'язки системи (3.3) – (3.7)
для вихідних даних (3.8)

	1	2	3	4
a^*	7.92822	-7.03716	8.16618	-6.67708
b^*	-11.4065	16.4890	-28.3207	23.8707
c^*	3.07557	-6.19900	18.3901	-15.9001
S^*	1.05562	2.42487	2.69815	2.89409
s_p^*	0.753992	1.15072	2.39725	2.58685

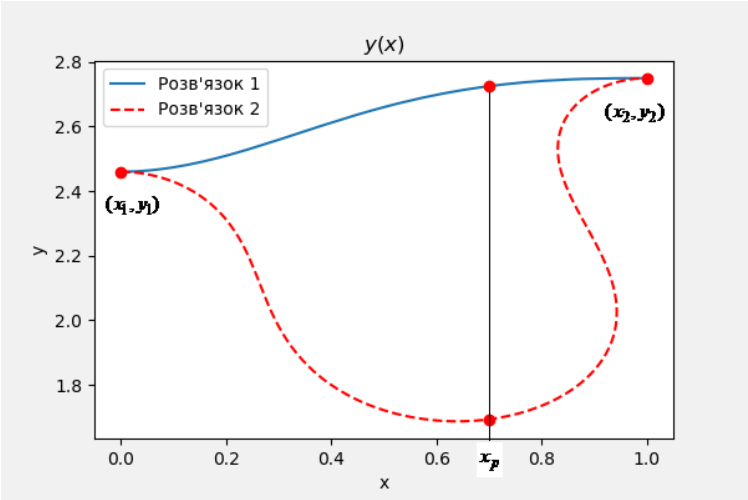


Рисунок 3.2 – Графіки кривих для першого і другого розв’язків з таблиці 3.1

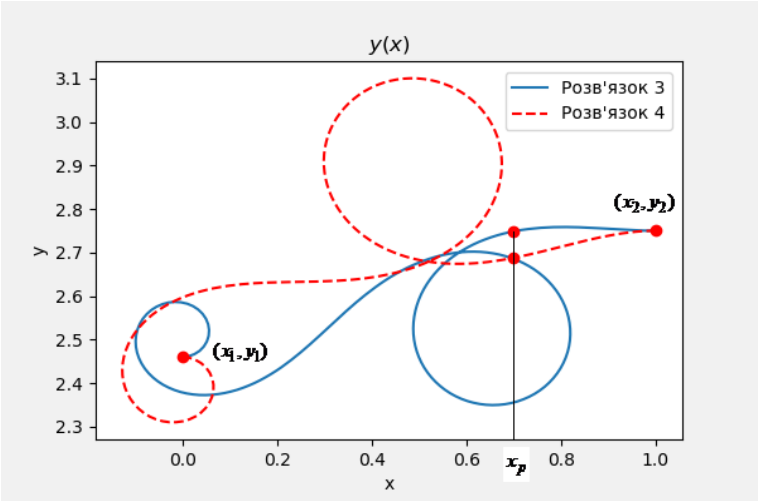


Рисунок 3.3 – Графіки кривих для третього і четвертого розв’язків із таблиці 3.1

Із рисунків 3.2 та 3.3 видно, що тільки для першого розв’язку крива є S -подібною. Вона характеризується наявністю однієї точки перегину, де кривина змінює знак. Її координати $x(s_1)$, $y(s_1)$ визначаються першим розв'язком квадратного рівняння $a^* s^2 + b^* s + c^* = 0$, $s_{1,2} = \dfrac{-b^* \pm \sqrt{(b^*)^2 - 4a^* c^*}}{2a^*}$, $s_1 = 0.35943 \in [0, S^*]$. Для трьох інших розв'язків це не так. Крива для другого

розв'язку включає дві ділянки, кожна із яких є S-подібною кривою, та характеризується довжиною $S^* = 2.42487$, яка є значно більшою за довжину $S^* = 1.05562$ для першого розв'язку. Зауважимо, що для другого розв'язку існує ділянка кривої $y(x)$, де ординати визначені неоднозначно. Для кривих, які відповідають третьому та четвертому розв'язкам, існують по дві ділянки, де ординати кривих $y(x)$ визначаються неоднозначно, причому ділянки кривих, які є ближчими до точки (x_2, y_2), мають циклічний характер.

Тому першою умовою для пошуку розв'язку системи (3.3) – (3.7) є відсікання тих зайвих розв'язків, яким не відповідають *S*-подібні криві. Це легко зробити за допомогою обмеження зверху на довжину кривої. Так, наприклад, якщо до системи рівнянь (3.3) – (3.7) додати нерівність $S \le 2$, то відсікатимуться другий, третій і четвертий розв'язки із таблиці 3.1.

Друга властивість: система (3.3) – (3.7) може бути погано масштабованою (сингулярною). Нехай довжина кривої $S^* = 1000$. Для того, щоб у точці (x_2, y_2) гарантувати близький до нуля кут φ_2, потрібно, щоб для малого ε виконувалася така нерівність

$$-\varepsilon \le \frac{a(1000)^3}{3} + \frac{b(1000)^2}{2} + 1000c \le \varepsilon. \qquad (3.9)$$

Нерівність (3.9) означає, що компоненти розв'язку системи (3.3) – (3.7) будуть величинами дуже різних порядків: так a^* буде мати порядок 10^{-9}, b^* буде мати порядок 10^{-6}, а c^* буде мати порядок 10^{-3}. Це висуває додаткові вимоги для методу розв'язання системи нелінійних рівнянь.

У цьому випадку може допомогти лема (Стецюк, Ткаченко, та Грицай 2020), яка встановлює зв'язок розв'язків вихідної системи (3.3) – (3.7) та масштабованої системи, в якій координати точок домножуються на одну і ту ж величину $\mu > 0$.

Лема 3.1 ($\mu > 0$). *Якщо* a^*, b^*, c^*, S^* та s_p^* є розв'язком системи (3.3) – (3.7), то

$$a^{**} = a^* / \mu^3, b^{**} = b^* / \mu^2, c^{**} = c^* / \mu, S^{**} = \mu S^*, s_p^{**} = \mu s_p^*, \qquad (3.10)$$

є розв’язком такої системи рівнянь:

$$\mu x_2 = \mu x_1 + \int_0^S \cos\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds, \tag{3.11}$$

$$\mu y_2 = \mu y_1 + \int_0^S \sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds, \tag{3.12}$$

$$\varphi_2 = \varphi_1 + \frac{aS^3}{3} + \frac{bS^2}{2} + cS, \tag{3.13}$$

$$\mu x_p = \mu x_1 + \int_0^{s_p} \cos\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds, \tag{3.14}$$

$$\varphi_p = \varphi_1 + \frac{as_p^3}{3} + \frac{bs_p^2}{2} + cs_p. \tag{3.15}$$

За допомогою леми 3.1 можна, використовуючи отриманий розв’язок для добре масштабованої системи (де усі компоненти розв’язку мають один порядок), легко знаходити відповідний йому розв’язок для погано масштабованої (сингулярної) системи. Для цього достатньо знайти розв'язок добре масштабованої системи (3.3) – (3.7) та перерахувати розв'язок для погано масштабованої системи (3.11) – (3.15) за формулою (3.10).

Зауважимо, що в трохи зміненому вигляді система (3.3) – (3.7) розглядається у статті (Борисенко, Устенко, та Устенко 2018), де невідома довжина s_p замінюється її невідомою часткою p у загальній довжині S, тобто $s_p = pS$, $0 < p < 1$. Для знаходження розв'язку системи спочатку з рівнянь (3.5) та (3.7) виражаються невідомі коефіцієнти a і c через невідомий коефіцієнт b та невідомі довжини S, s_p. Підставляючи їх у рівняння (3.3), (3.4) та (3.6) отримуємо систему трьох інтегральних рівнянь з трьома невідомими b, S, s_p. Використання ітераційних методів для пошуку розв'язку системи трьох рівнянь затрудняється складними виразами для підінтегральних функцій.

Нижче розглянемо метод, який звільнений від вказаного недоліку. У його основу покладено модифікацію *r*-алгоритму (Шор та Стецюк 1997) для розв'язання спеціальної задачі мінімізації негладкої функції (сума модулів

нев’язок системи (3.3) – (3.7)) при контролі обмежень на довжини S, s_p, щоб гарантувати їх додатні допустимі значення.

3.2 Оптимізаційна задача та алгоритм її розв’язання

Розглянемо умовну задачу мінімізації суми модулів функцій нев’язок для рівнянь (3.3) – (3.7), яка має вигляд: знайти

$$f^* = f\left(a^*, b^*, c^*, S^*, s_p^*\right) =$$

$$= \min_{a,b,c,S,s_p} \left\{ f\left(a,b,c,S,s_p\right) = \sum_{i=1}^{3} \left| f_i\left(a,b,c,S\right)\right| + \sum_{i=4}^{5} \left| f_i\left(a,b,c,s_p\right)\right| \right\} \tag{3.16}$$

при обмеженнях

$$S_{\min} \le S \le S_{\max}, \tag{3.17}$$

$$\left|x_p - x_1\right| \le s_p \le S, \tag{3.18}$$

$$-\frac{\pi}{2} \le \varphi_1 + a\frac{i^3 S^3}{3N^3} + b\frac{i^2 S^2}{2N^2} + c\frac{iS}{N} \le \frac{\pi}{2}, \quad i = 1,\ldots,N, \tag{3.19}$$

де нев'язки для рівнянь (3.3) – (3.7) задаються такими функціями

$$f_1(a,b,c,S) = x_2 - x_1 - \int_0^S \cos\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds, \tag{3.20}$$

$$f_2(a,b,c,S) = y_2 - y_1 - \int_0^S \sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds, \tag{3.21}$$

$$f_3(a,b,c,S) = \varphi_2 - \varphi_1 - \frac{aS^3}{3} - \frac{bS^2}{2} - cS, \tag{3.22}$$

$$f_4(a,b,c,s_p) = x_p - x_1 - \int_0^{s_p} \cos\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds, \tag{3.23}$$

$$f_5(a,b,c,s_p) = \varphi_p - \varphi_1 - \frac{as_p^3}{3} - \frac{bs_p^2}{2} - cs_p, \tag{3.24}$$

N – кількість точок дискретизації функції $\varphi(s)=\varphi_1+\frac{as^3}{3}+\frac{bs^2}{2}+cs$ на інтервалі $s\in[0,S]$.

Тут цільова функція (3.16) є негладкою та означає мінімізацію суми модулів функцій (3.20) – (3.24) – функцій нев'язок для рівнянь (3.3) – (3.7). Обмеження (3.17) та (3.18) гарантують додатні значення для довжин S та s_p, які є верхніми границями для визначених інтегралів в функціях нев'язок (3.20), (3.21) та (3.23). Тут $S_{\min}=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$ – мінімальна відстань між точками (x_1,y_1) та (x_2,y_2), $S_{\max}>S_{\min}$ – параметри для управління нижньою та верхньою межами на S – загальну довжину кривої. Чим ближчим до $S_{\min}$ є значення верхньої межі $S_{\max}$, тим легше для системи (3.3) – (3.7) уникнути «циклічних» розв'язків (третій та четвертий розв'язки на рис. 3.3), а значить легше знайти розв'язок, який визначає S-подібну криву (перший розв'язок на рис. 3.2).

Обмеження (3.19) забезпечує існування єдиного глобального мінімуму для задачі (3.16) – (3.19). Воно використовує доповнення задачі (3.16) – (3.18) дискретним аналогом неперервного обмеження

$$-\frac{\pi}{2}\le\varphi(s)=\varphi_1+\frac{as^3}{3}+\frac{bs^2}{2}+cs\le\frac{\pi}{2},\quad s\in S. \tag{3.25}$$

Обмеження (3.25) означає, що кути нахилу дотичних в довільній точці кривої не виходять за заданий діапазон $\varphi(s)\in\left[-\pi/2,\pi/2\right]$, тобто функція $y(x)$ на інтервалі $[x_1,x_2]$ є однозначно визначеною. Це відсікає розв'язки системи (3.3) – (3.7), для яких існують такі $\bar{x}\in[x_1,x_2]$, що ордината $y(\bar{x})$ визначається неоднозначно (друга крива із рисунку 3.2, обидві криві з рисунку 3.3).

Якщо в оптимізаційній задачі опустити обмеження (3.19), то отримаємо задачу (3.16) – (3.18), яка розглядалася в статті (Стецюк, Ткаченко, та Грицай 2020). При великих значеннях верхньої межі $S_{\max}$ задача (3.16) – (3.18) є багатоекстремальною задачею нелінійного програмування. Кількість її глобальних мінімумів визначається значенням $S_{\max}$. Наприклад, якщо $S_{\max}=2.0$

, то для вихідних даних (3.8) задача (3.16) – (3.18) має єдину точку глобального мінімуму, дві точки глобального мінімуму – якщо $S_{\max}=2.5$, три – якщо $S_{\max}=2.7$, чотири – якщо $S_{\max}=3.0$. Задача (3.16) – (3.19) має всього одну точку глобального мінімуму, яка співпадає з розв'язком системи (3.3) – (3.7) з найменшою загальною довжиною кривої.

Якщо в результаті пошуку локального мінімуму для задачі (3.16) – (3.19) отримуємо $f^*=0$, то це означає, що знайдена точка глобального мінімуму $\left(a^*,b^*,c^*,S^*,s_p^*\right)^T$, яка є розв'язком системи (3.3) – (3.7). Якщо отримуємо $f^*>0$, то точка $\left(a^*,b^*,c^*,S^*,s_p^*\right)^T$ не є розв'язком системи (3.3) – (3.7). Це може бути як у випадку відсутності розв'язку у системи (3.3) – (3.7) при обмеженнях (3.17), (3.18), так і у випадку, якщо алгоритм зупиниться в «неоптимальній» точці, враховуючи, що для великих значень параметра $S_{\max}$ задача (3.16) – (3.18) є багатоекстремальною.

Задача (3.16) – (3.19) є задачею мінімізації негладкої функції, яка визначена не при всіх значеннях S та s_p, а тільки при тих, які є додатними та дозволяють обчислювати визначені інтеграли для функцій $f_1\left(a,b,c,S\right)$, $f_2\left(a,b,c,S\right)$ та $f_4\left(a,b,c,s_p\right)$. Для знаходження точки локального мінімуму у задачі (3.16) – (3.18) може бути використана модифікація *r*-алгоритму (Шор та Стецюк 1997), (Stetsyuk 2017), яка враховує вказану особливість задачі. У точці, де узагальнений градієнт цільової функції є невизначеним, модифікація *r*-алгоритму використовує узагальнений градієнт до одного із порушених обмежень (3.17), (3.18) і (3.19).

Алгоритм розв'язання задачі (3.16) – (3.19) реалізований за допомогою методу мультистарту та Octave-функції **ralgb5a** (Стецюк 2017б), (Стецюк 2019) та знаходить найкращий локальний мінімум цільової негладкої функції за допомогою запуску модифікації *r*-алгоритму із заданої кількості стартових точок. Алгоритм використовує аналітичний спосіб обчислення узагальнених градієнтів

цільової функції (3.16) та метод трапецій для обчислення інтегралів як у формулах (3.20), (3.21) та (3.23) так і у формулах для узагальнених градієнтів.

Наведемо формули для обчислення узагальненого градієнту цільової функції (3.16). Нехай вектор $g_f = \left(\frac{\partial f}{\partial a}, \frac{\partial f}{\partial b}, \frac{\partial f}{\partial c}, \frac{\partial f}{\partial S}, \frac{\partial f}{\partial s_p} \right)^T$ є узагальненим градієнтом цільової функції $f\left(a,b,c,S,s_p\right) = \sum_{i=1}^{3} \left| f_i\left(a,b,c,S\right) \right| + \sum_{i=4}^{5} \left| f_i\left(a,b,c,s_p\right) \right|$.

Перші три компоненти узагальненого градієнта обчислюються за формулами:

$$\frac{\partial f}{\partial a} = \sum_{i=1}^{5} sign(f_i) \frac{\partial f_i}{\partial a}, \quad \frac{\partial f}{\partial b} = \sum_{i=1}^{5} sign(f_i) \frac{\partial f_i}{\partial b},$$

$$\frac{\partial f}{\partial c} = \sum_{i=1}^{5} sign(f_i) \frac{\partial f_i}{\partial c}, \tag{3.26}$$

де

$$\frac{\partial f_1}{\partial a} = -\int_0^S \frac{s^3}{3} \sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs \right) ds,$$

$$\frac{\partial f_2}{\partial a} = -\int_0^S \frac{s^3}{3} \cos\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs \right) ds, \quad \frac{\partial f_3}{\partial a} = -\frac{S^3}{3}, \tag{3.27}$$

$$\frac{\partial f_4}{\partial a} = -\int_0^{s_p} \frac{s^3}{3} \sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs \right) ds, \quad \frac{\partial f_5}{\partial a} = -\frac{s_p^3}{3},$$

$$\frac{\partial f_1}{\partial b} = -\int_0^S \frac{s^2}{2} \sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs \right) ds,$$

$$\frac{\partial f_2}{\partial b} = -\int_0^S \frac{s^2}{2} \cos\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs \right) ds, \quad \frac{\partial f_3}{\partial b} = -\frac{S^2}{2}, \tag{3.28}$$

$$\frac{\partial f_4}{\partial b} = -\int_0^{s_p} \frac{s^2}{2} \sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs \right) ds, \quad \frac{\partial f_5}{\partial b} = -\frac{s_p^2}{2},$$

$$\frac{\partial f_1}{\partial c} = -\int_0^S s \sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds,$$

$$\frac{\partial f_2}{\partial c} = -\int_0^S s \cos\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds, \quad \frac{\partial f_3}{\partial c} = -S, \tag{3.29}$$

$$\frac{\partial f_4}{\partial c} = -\int_0^{s_p} s \sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds, \quad \frac{\partial f_5}{\partial c} = -s_p.$$

Дві останні компоненти узагальненого градієнта обчислюються за формулами:

$$\frac{\partial f}{\partial S} = \sum_{i=1}^{3} sign(f_i) \frac{\partial f_i}{\partial S}, \quad \frac{\partial f}{\partial s_p} = \sum_{i=4}^{5} sign(f_i) \frac{\partial f_i}{\partial s_p}, \tag{3.30}$$

де

$$\frac{\partial f_1}{\partial S} = -\cos\left(\varphi_1 + \frac{aS^3}{3} + \frac{bS^2}{2} + cS\right),$$

$$\frac{\partial f_2}{\partial S} = -\sin\left(\varphi_1 + \frac{aS^3}{3} + \frac{bS^2}{2} + cS\right). \tag{3.31}$$

$$\frac{\partial f_3}{\partial S} = -aS^2 - bS - c, \quad \frac{\partial f_4}{\partial s_p} = \cos\left(\varphi_1 + \frac{as_p^3}{3} + \frac{bs_p^2}{2} + cs_p\right),$$

$$\frac{\partial f_5}{\partial s_p} = -as_p^2 - bs_p - c. \tag{3.32}$$

У точці, де узагальнений градієнт цільової функції є невизначеним, для модифікації *r*-алгоритму він замінюється на узагальнений градієнт до одного із порушених двосторонніх обмежень (3.17), (3.18) або (3.19). Для цього використовуються шість порушених обмежень $g_i(\circ) \le 0$, $i = \overline{1,6}$, які вибираються у такому порядку: $g_1(S) = S_{\min} - S$, $g_2(S) = S - S_{\max}$ – для двосторонніх обмежень (3.17), $g_3(s_p) = |x_p - x_1| - s_p$, $g_4(S, s_p) = s_p - S$ – для обмежень (3.18),

$$g_5(a,b,c,S) = \max_{i=1}^{N} \left\{ -\frac{\pi}{2} - \varphi_1 - a\frac{i^3 S^3}{3N^3} - b\frac{i^2 S^2}{2N^2} - c\frac{iS}{N} \right\},$$

$g_6(a,b,c,S)=\max\limits_{i=1}^{N}\left\{\varphi_1+a\frac{i^3S^3}{3N^3}+b\frac{i^2S^2}{2N^2}+c\frac{iS}{N}-\frac{\pi}{2}\right\}$ – для двосторонніх обмежень (3.19). Якщо всі шість обмежень виконуються, то тоді використовується узагальнений градієнт цільової функції (3.16), який обчислюється за формулами (3.26) – (3.32).

Описаний алгоритм розв'язання оптимізаційної задачі (3.16) – (3.19) реалізований мовою Octave. За його допомогою можна знаходити розв'язки системи (3.3) – (3.7), яким відповідають *S*-подібні криві. Якщо, у задачі (3.16) – (3.19) опустити обмеження (3.19), а це рівносильно ігноруванню порушених обмежень $g_5(\circ)$ та $g_6(\circ)$, то тоді алгоритм при великих значеннях параметра $S_{\max}$ дозволяє знаходити також і «циклічні» розв'язки. Так, саме за допомогою такого алгоритму були знайдені всі чотири розв'язки, які наведені у таблиці 3.1.

3.3 Обчислювальні експерименти та моделювання сопла Франкля

У підрозділі наведено результати обчислювальних експериментів, які характеризують роботу алгоритму для тестового прикладу (3.8) та його масштабованого варіанту ($\mu=100$)

$$x_1=0,\ y_1=246,\ \varphi_1=0^\circ;\quad x_2=100,\ y_2=275,\ \varphi_2=0^\circ;$$
$$x_p=70,\ \ \varphi_p=0.209440=12^\circ. \tag{3.33}$$

Для вихідних даних (3.8) глобальним мінімумом задачі (3.16) – (3.19) є єдина точка, для якої компоненти a^*,b^*,c^*,S^*,s_p^* співпадають з першим розв'язком системи (3.3) – (3.7) із таблиці 3.1. Для вихідних даних (3.33) глобальним мінімумом є точка з компонентами $\bar{a}^*=a^*\times10^{-6}$, $\bar{b}^*=b^*\times10^{-4}$, $\bar{c}^*=c^*\times10^{-2}$, $\bar{S}^*=S^*\times100$, $\bar{s}_p^*=s_p^*\times100$.

Перший експеримент пов'язаний з дослідженням збіжності алгоритму в залежності від вибору стартової точки та верхньої межі $S_{\max}=qS_{\min}$, де

$q \in \{1.2, 1.5, 2.0, 3.0\}$. Компоненти стартової точки для коефіцієнтів квадратичної функції кривини a_0, b_0, c_0 генерувались за допомогою Octave-датчика випадкових чисел `rand("seed", 2020)`, а для довжин S та s_p вибирались рівними їх нижнім межам $S_0 = S_{\min} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, $(s_p)_0 = |x_p - x_1|$. Критерієм успішного розв'язання задачі (3.16) – (3.19) була вибрана умова, що значення функції, яка мінімізується, є меншим ніж 10^{-6}. Найбільш значимі параметри *r*-алгоритму вибиралися відповідно до рекомендованих у (Крайко 2014) для мінімізації негладких функцій – коефіцієнт розтягу простору $\alpha = 1.5$, величина початкового кроку $h_0 = 1.0$, коефіцієнт зменшення кроку $q_0 = 1.0$, точність зупинки по аргументу $\varepsilon_x = 10^{-6}$, максимальна кількість ітерацій `maxitn=500`.

У таблиці 3.2 наведено результати першого експерименту щодо розв'язання задачі (3.16) – (3.19) для 100 стартових точок, де компоненти a_0, b_0, c_0 належали чотирьом діапазонам $[-5,5]^3$, $[-25,25]^3$, $[-50,50]^3$, $[-100,100]^3$. Розглядалися три різних сценарії. Перший сценарій – задача вирішувалась для вихідних даних (3.8), другий та третій сценарії – задача вирішувалась задача для вихідних даних (3.33). В першому та другому сценаріях стартові точки вибирались одними і тими ж, а для третього сценарію перші три компоненти стартових точок домножувалися на коефіцієнти $\{10^{-6}, 10^{-4}, 10^{-2}\}$, які узгоджуються з коефіцієнтом масштабу $\mu = 100$ для вихідних даних (3.33).

Таблиця 3.2 – Кількість успішних стартових точок для трьох сценаріїв (перший експеримент, $\varphi_1 = \varphi_2 = 0^\circ$, $\varphi_p = 0.209440 = 12^\circ$)

діапазон	$q = 1.2$	$q = 1.5$	$q = 2.0$	$q = 3.0$
$x_1 = 0,\ y_1 = 2.46,\quad x_2 = 1,\ y_2 = 2.75,\quad x_p = 0.7$				
$[-5,5]^3$	100/100	100/100	100/100	99/100

діапазон	$q=1.2$	$q=1.5$	$q=2.0$	$q=3.0$
$x_1=0,\ y_1=2.46,\quad x_2=1,\ y_2=2.75,\quad x_p=0.7$				
$[-25,25]^3$	100/100	100/100	88/100	80/100
$[-50,50]^3$	93/100	84/100	74/100	71/100
$[-100,100]^3$	81/100	70/100	65/100	67/100
$x_1=0,\ y_1=246,\quad x_2=100,\ y_2=275,\quad x_p=70$				
$[-5,5]^3$	95/100	95/100	95/100	95/100
$[-25,25]^3$	87/100	88/100	85/100	81/100
$[-50,50]^3$	86/100	70/100	70/100	73/100
$[-100,100]^3$	78/100	64/100	60/100	62/100
$x_1=0,\ y_1=246,\ x_2=100,\ y_2=275,\ x_p=70,$ $D=\mathrm{diag}\left\{10^{-6},10^{-4},10^{-2}\right\}$				
$[-5,5]^3\times D$	98/100	98/100	98/100	98/100
$[-25,25]^3\times D$	95/100	95/100	95/100	95/100
$[-50,50]^3\times D$	92/100	92/100	92/100	92/100
$[-100,100]^3\times D$	95/100	95/100	95/100	95/100

Із таблиці 3.2 видно, що для добре масштабованих даних із 100 запусків алгоритм завжди збігається до точки глобального мінімуму задачі (3.16) – (3.19), якщо перші три компоненти стартової точки належать діапазонам $[-5,5]^3$ та $[-25,25]^3$. Якщо перші три компоненти належать діапазонам $[-50,50]^3$ та $[-100,100]^3$, то алгоритм не завжди збігається до точки глобального мінімуму. Наприклад, для $S_{\max}=2\times S_{\min}$ та діапазону $[-100,100]^3$ алгоритм знайшов розв’язок в шістдесяти п’яти випадках із ста, а в тридцяти п'яти випадках алгоритм не зміг знайти розв’язку та зупинився в точках, де значення цільової

функції (3.16) є більшим нуля. Це зумовлено тим, що глобальний мінімум у задачі (3.16) – (3.19) єдиний і йому відповідає довжина $S^* = 1.05562$, а стартова точка далека від точки мінімуму, завдяки чому градієнтний процес прямує до іншого розв'язку системи, якому відповідає суттєво більша довжина кривої. Такі розв'язки можна знайти, якщо вибрати значно завищеною верхню межу $S_{\max}$ та розв'язувати задачу (3.16) – (3.18), тобто не враховувати обмеження (3.19). Але цим розв'язкам відповідають «циклічні» криві (рис. 3.3), які не підходять для моделювання зовнішнього контуру сопла Франкля.

Для другого та третього сценаріїв, коли вихідні дані (3.33) погано масштабовані, немає жодного випадку, щоб алгоритму вдалося успішно розв’язати задачу (3.16) – (3.18) у всіх ста випадках. При цьому результати для третього сценарію є кращими за результати для другого, що пояснюється тим, що для третього сценарію стартова точка є ближчою до точки глобального мінімуму, ніж для другого сценарію. З результатів першого експерименту випливає, що краще розв’язувати задачу з добре масштабованими даними, ніж з погано масштабованими даними, а стартову точку вибирати такою, щоб її перших три компоненти були якомога ближчими до нуля. Це підтверджують результати другого та третього експериментів, які наведено у таблицях 3.3 та 3.4.

Таблиця 3.3 – Прискорення *r*-алгоритму (другий експеримент)

ε_x	$q = 1.2$	$q = 1.5$	$q = 2.0$	$q = 3.0$
$\alpha = 1.5, \quad h_0 = 1.0, \quad q_1 = 1.0$				
10^{-4}	133/152/20	142/162/17	142/162/17	142/162/17
10^{-6}	198/243/20	197/232/17	197/232/17	197/232/17
10^{-8}	256/307/20	253/312/17	253/312/17	253/312/17
10^{-10}	302/366/20	303/372/17	314/385/17	289/355/17
10^{-12}	371/457/20	372/465/17	365/450/17	370/457/17
$\alpha = 2.0, \quad h_0 = 1.0, \quad q_1 = 0.95$				

ε_x	$q=1.2$	$q=1.5$	$q=2.0$	$q=3.0$
$\alpha=1.5,\quad h_0=1.0,\quad q_1=1.0$				
10^{-4}	79/130/19	70/116/11	70/116/11	70/116/11
10^{-6}	113/179/19	103/154/11	103/154/11	103/154/11
10^{-8}	133/208/19	127/182/11	127/182/11	127/182/11
10^{-10}	165/258/19	146/205/11	146/205/11	146/205/11
10^{-12}	190/292/19	180/250/11	178/247/11	178/247/11

У таблиці 3.3 наведено витрати *r*-алгоритму (кількість ітерацій / кількість обчислень узагальненого градієнта цільової функції / кількість обчислень узагальненого градієнта до порушених обмежень) залежно від вибору параметрів α, h_0, q_1 для знаходження *S*-подібної кривої для вихідних даних (3.8) з 5-ма різними критеріями зупинки за аргументом $\varepsilon_x=\left\{10^{-4},10^{-6},10^{-8},10^{-10},10^{-12}\right\}$. Алгоритм в обох випадках стартував з однієї і тієї ж точки з компонентами $a_0=b_0=c_0=0$, $S_0=\sqrt{\left(x_2-x_1\right)^2+\left(y_2-y_1\right)^2}$, $\left(s_p\right)_0=\left|x_p-x_1\right|$. З таблиці видно, що завдяки вибору параметрів $\alpha=2.0$, $h_0=1.0$, $q_1=0.95$ можна досягнути прискорення майже в два рази по відношенню до вибору параметрів $\alpha=1.5$, $h_0=1.0$, $q_1=1.0$. Зауважимо, що хоча $q_1=0.95$ не рекомендується вибирати для мінімізації негладких функцій, в даному випадку його можна використовувати, оскільки кількість невідомих є невеликою.

У таблиці 3.4 порівнюються витрати *r*-алгоритму (за кількістю ітерацій – *itn*, за кількістю обчислень узагальненого градієнта цільової функції – *nfg*, за кількістю обчислень узагальненого градієнта до порушених обмежень – *ng*) для знаходження *S*-подібної кривої для добре масштабованих даних (3.8) та погано масштабованих даних (2.33). Витрати (для (3.8) / для (3.33)) наведені для п’ятьох критеріїв зупинки за аргументом $\varepsilon_x=\left\{10^{-5},10^{-6},10^{-7},10^{-8},10^{-9}\right\}$. В таблиці 3.4 також наведено абсолютні відхилення між компонентами розв’язків для добре

масштабованих та погано масштабованих даних. Вони обчислювались за формулами $\Delta a = \left|a^* - \bar{a}^* \times 10^6\right| \Big/ \left|a^*\right|$, $\Delta b = \left|b^* - \bar{b}^* \times 10^6\right| \Big/ \left|b^*\right|$, $\Delta c = \left|c^* - \bar{c}^* \times 10^6\right| \Big/ \left|c^*\right|$, $\Delta S = \left|S^* - \bar{S}^* \times 10^6\right| \Big/ \left|S^*\right|$, $\Delta s_p = \left|s_p^* - \bar{s}_p^* \times 10^6\right| \Big/ \left|s_p^*\right|$. З таблиці 3.4 видно, що при розв'язанні задачі з добре масштабованими даними потрібно майже в два рази менше ітерацій та майже в чотири рази менше обчислень узагальненого градієнта, ніж при розв'язанні задачі з погано масштабованими даними.

Таблиця 3.4 – Витрати *r*-алгоритму на пошук розв'язків добре та погано масштабованих задач (третій експеримент)

	$\varepsilon_x = 10^{-5}$	$\varepsilon_x = 10^{-6}$	$\varepsilon_x = 10^{-7}$	$\varepsilon_x = 10^{-8}$	$\varepsilon_x = 10^{-9}$
itn	84/204	113/219	113/238	133/247	137/258
nfg	139/635	179/653	179/684	208/695	212/718
ng	19/51	19/51	19/51	19/51	19/51
Δa	2.64e-06	7.07e-07	2.87e-08	3.00e-09	1.06e-09
Δb	7.82e-07	5.73e-07	2.19e-08	2.09e-09	8.54e-10
Δc	1.67e-07	4.10e-07	1.58e-08	1.43e-09	6.10e-10
ΔS	2.26e-08	1.07e-08	5.70e-09	1.37e-10	6.23e-11
Δs_p	1.84e-07	2.25e-08	3.34e-09	6.72e-11	1.40e-10

Результати обчислювальних експериментів показують, що стартову точку можна вибирати рівною $\left(0,0,0,S_{\min},\left|x_p - x_1\right|\right)^T$ незалежно від масштабу даних задачі (3.16) – (3.19). Щоб знайти розв'язок із заданою точністю за менший час краще розв'язувати задачу з добре масштабованими даними, ніж з погано масштабованими даними. Масштабування може відігравати суттєву роль для вибору того чи іншого параметру при моделюванні профілів в ітераційному режимі, враховуючи, що масштаб достатньо встановити всього один раз, а задачу потрібно вирішувати багато разів.

Наведемо результати обчислювальних експериментів щодо використання розробленого алгоритму для побудови фрагментів дозвукової та надзвукової частин зовнішнього контуру сопла Франкля. Вони пов'язані з проектуванням сопла з центральним тілом для трьох заданих реперних точок зовнішнього контуру, які визначають дозвукову та надзвукову частини сопла та їх стикування у точці критичного перерізу. В реперних точках похідні (кути нахилу дотичних) дорівнюють нулю. Для даного контуру задачі (3.16) – (3.19) визначаються вихідними даними, для яких виконується умова $|y_2 - y_1|/|x_2 - x_1| \ll 1$, тому відповідні фрагментам контурів *S*-подібні криві є витягнутими відносно горизонтальної осі.

Перший експеримент пов'язаний з дослідженням впливу кута φ_p на форму *S*-подібної кривої для тестового прикладу (3.8), для якого $|y_2 - y_1|/|x_2 - x_1| = 0,29$. В таблиці 3.5 представлені точки глобальних мінімумів трьох задач для різних значень кутів $\varphi_p \in \{16^\circ, 12^\circ, 8^\circ\}$. Використовуючи ці точки за формулами (3.10) легко знайти розв'язки системи (3.3) – (3.7) для масштабованих вихідних даних (3.33), де $\mu = 100$.

Таблиця 3.5 – Глобальні мінімуми трьох задач для вихідних даних (3.8)

	$\varphi_p = 16^\circ$	$\varphi_p = 12^\circ$	$\varphi_p = 8^\circ$
a^*	4.0903	7.9282	11.5304
b^*	-7.3628	-11.4065	-15.2523
c^*	2.3632	3.0756	3.7636
S^*	1.0511	1.0556	1.0630
s_p^*	0.74714	0.75399	0.76259

Розв'язки з таблиці 3.5 проілюстровано на рисунку 3.4, де ліворуч наведено графіки кривих, а праворуч – графіки відповідних кривин. З графіків кривин

видно, що для першої та другої двохланкових кривих знаки кривини змінюються тільки в одній точці (точка перегину), що доводить їх *S*-подібність. Для третьої кривої знак кривини змінюється в двох точках (відмічені на рис. праворуч), тому вона не є *S*-подібною. Зауважимо, що із графіків кривих такий висновок зробити неможливо, оскільки усі три криві виглядають як *S*-подібні.

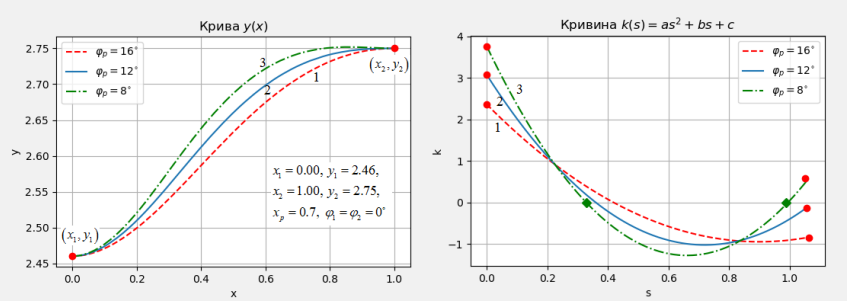


Рисунок 3.4 – Графіки кривих та кривин для трьох розв'язків з таблиці 3.5

Другий експеримент пов'язаний з побудовою двох варіантів кривих для дозвукової та надзвукової частин зовнішнього контуру сопла за трьома заданими реперними точками: в точці $A\left(x_A = 0,\ y_A = 295\right)$ починається дозвукова частина, точка $B\left(x_B = 216,\ y_B = 246\right)$ є точкою критичного перерізу (число Маха $M = 1$), в точці $C\left(x_C = 638,\ y_C = 275\right)$ закінчується надзвукова частина. В таблиці 3.6 наведено точки глобальних мінімумів для задач $(3.16) - (3.19)$ з вихідними даними, де координати точок зменшувалися в $\mu = 100$ раз. Розглядалися два варіанти кутів φ_p при абсцисах x_p, яка для дозвукової частини вибиралась ближчою до абсциси точки A, а для надзвукової частини – ближчою до абсциси точки C.

Таблиця 3.6 – Розв'язки для дозвукової

$\left(x_1 = 0.00,\ y_1 = 2.95,\ x_2 = 2.16,\ y_2 = 2.46,\ x_p = 0.7\right)$ та надзвукової

$\left(x_1 = 2.16,\ y_1 = 2.46,\ x_2 = 6.38,\ y_2 = 2.75,\ x_p = 5.0\right)$ частин сопла

	Дозвукова частина		Надзвукова частина	
	$\varphi_p = -12^\circ$	$\varphi_p = -8^\circ$	$\varphi_p = 4^\circ$	$\varphi_p = 2^\circ$
a^*	0.53981	0.97384	0.021423	0.057183
b^*	-0.66700	-1.65380	-0.13663	-0.28816
c^*	-0.15169	0.22102	0.16123	0.26827
S^*	2.2312	2.2439	4.2327	4.2372
s_p^*	0.70440	0.70109	2.8517	2.8571

Графічна ілюстрація розв'язків наведена на рисунку 3.5, де у верхній частині рисунку представлено графіки кривих для дозвукового та надзвукового фрагментів контуру, а в нижній частині – графіки кривин, які відповідають цим кривим. Перші криві для обох частин контуру є *S*-подібними, для них кривини змінюють знак тільки в одній точці. Другі криві не є *S*-подібними, для них кривина змінює знак в двох точках (відмічені на рисунку 3.5 внизу).

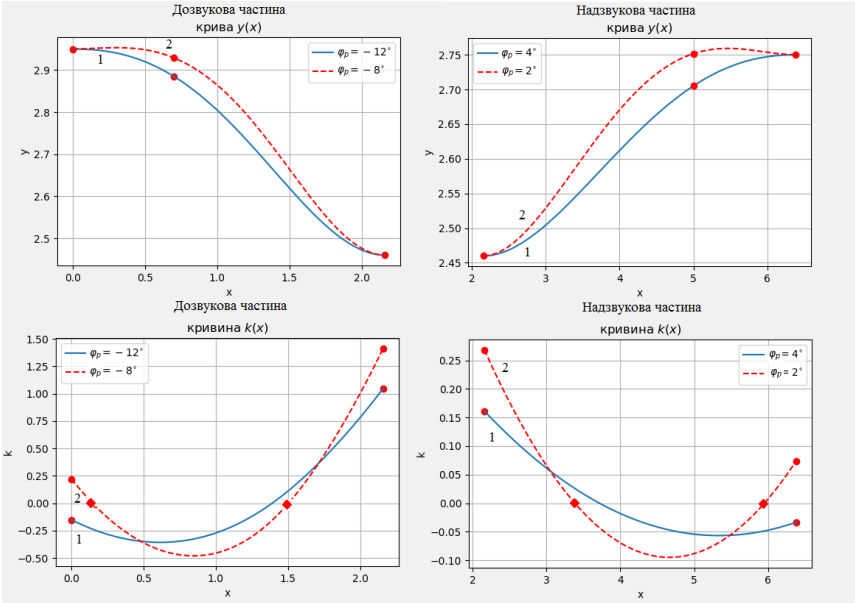


Рисунок 3.5 – Графіки кривих та кривин для розв'язків з таблиці 3.6

Проведені обчислювальні експерименти для проектування фрагментів дозвукової та надзвукової частини зовнішнього контуру сопла Франкля показали

ефективність розробленого алгоритму для побудови двохланкових S-подібних кривих.

3.4 Опис програмного забезпечення

Загальна структура комплексу програм для проектування фрагментів дозвукової та надзвукової частини зовнішнього контуру сопла Франкля має вигляд, як на рисунку 3.6. Комплекс програм включає головну програму **Frankl-quadratic**, яка за допомогою *r*-алгоритму розвязує задачу (3.16) – (3.19), та три підпрограми: **ralgb5a** (реалізує *r*-алгоритм), **fgO1** (обчислює значення функції та узагальненого градієнта) та **plotO1** (забезпечує графічний вигляд знайденої кривої та її характеристик). Нижче наведемо їх короткий опис та коди мовою Octave, за винятком підпрограми ralgb5a. Її опис та Octave код, що включає і короткі англомовні коментарі для вхідних та вихідних параметрів, наведено в (Сергієнко та ін. 2018, с. 26–28).

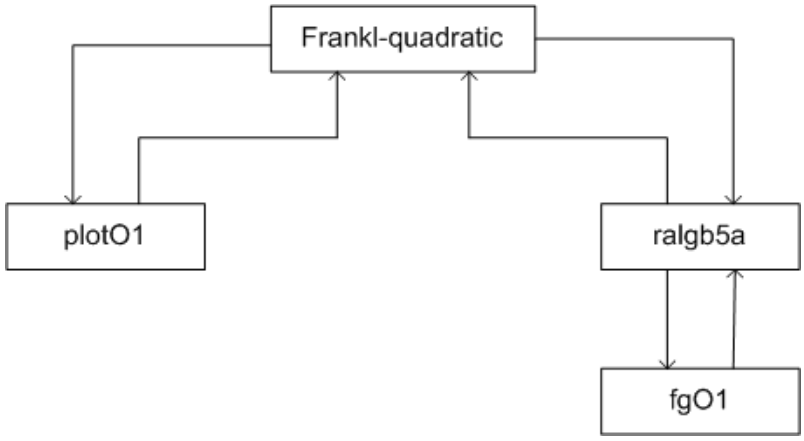


Рисунок 3.6 – Блок-схема програми Frankl-quadratic

Головна програма Frankl-quadratic керує процесом знаходження розв’язку оптимізаційної задачі (3.16) – (3.19). Її код наведено нижче.

```
global x1 y1 phi1 x2 y2 phi2 xp phip
global N Smin Smax spmin
global nfgr
```

```
# test  No 1
#x1 = 0.0;  y1 = 2.46;  phi1 = 0.0;
#x2 = 1.0;  y2 = 2.75;  phi2 = 0.0;
#xp = 0.7;  phip = (9.0/180.0)*pi;

# test  No 1m
#x1 =   0.0;  y1 = 246.0;  phi1 = 0.0;
#x2 = 100.0;  y2 = 275.0;  phi2 = 0.0;
#xp =  70.0;  phip = (9.0/180.0)*pi;

# test  No 2
x1 = 0;   y1 = 2.75; phi1 = 0.0;
x2 = 1.;  y2 = 2.46; phi2 = 0.0;
xp = 0.3; phip = -(12.0/180.0)*pi;

# test  No 2m
#x1 =   0.0;  y1 = 275.0;  phi1 = 0.0;
#x2 = 100.0;  y2 = 246.0;  phi2 = 0.0;
#xp =  30.0;  phip = -(12.0/180.0)*pi;

printf("x1 y1 phi1 %13.5e %13.5e %13.5e = %13.5e\n",
        x1, y1, phi1, 180*(phi1/pi));
printf("x2 y2 phi2 %13.5e %13.5e %13.5e = %13.5e\n",
        x2, y2, phi2, 180*(phi2/pi));
printf("xp phip %13.5e %13.5e = %13.5e\n",
        xp, phip, 180*(phip/pi));

Smin = sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
spmin = xp - x1; Smax = 1.2*Smin;

printf("\n");
alpha = 2.0, h0 = 1.0, q1 = 0.95,
epsx = 1.e-6, epsg = 1.e-8, maxitn = 1000, intp=20;

printf("\n");
N = 200;  eps=1.d-4;

nfgr = 0; x0 = zeros(3,1);
x0(4,1) = Smin; x0(5,1) = spmin;
printf("x0(tr) = %9.5f %9.5f %9.5f %9.5f %9.5f \n",
        x0(1:5));
[xr,fr,itn,nfg,istop] = ralgb5a(@fgO1,x0,alpha,h0,q1,
                        epsg,epsx,maxitn,intp);
printf("..itn %4d  fr %23.15e istop %d  nfg(-) %4d(%3d)\n",
        itn, fr, istop, nfg, nfg-nfgr);
if (fr >= eps)
  printf(" error: fr = %9.2e > eps = %9.2e \n",fr,eps);
endif

a=xr(1,1); b=xr(2,1); c=xr(3,1); S=xr(4,1); sp=xr(5,1);
printf("\n");
printf(" fr %9.2e\n",fr);
printf("a b c S sp  %13.5e %13.5e %13.5e %13.5e %13.5e\n",
```

```
        a,b,c,S,sp);
[dphi2] = pict1(a,b,c,S,sp,200);
print -deps figure-fgO1a.eps
print -dpng figure-fgO1a.png
pause
```

В програмі задаються вхідні дані, встановлюються параметри *r*-алгоритму, запускається процесс оптимізації для розв'язання задачі (3.16) – (3.19), аналізується отриманий розв'язок (перевіряється чи нев'язка більша заданої величини). Компоненти стартової точки для коефіцієнтів квадратичної функції вибираються рівними нулю. Результати розрахунку виводяться у файл протоколу роботи програми та відображаються у вигляді графічного вікна, яке включає графіки кривих та графіки відповідних кривин. Приклади графічного вікна наведено на рисунках 3.7 та 3.8.

Програма Frankl-quadratic використовує підпрограми ralgb5a та plotO1. Підпрограма ralgb5a реалізує роботу *r*-алгоритму для розв'язання оптимізаційної задачі (3.16) – (3.19), використовуючи підпрограму fgO1, код якої наведено нижче.

```
function [f,g] = fgO1(x)
global x1 y1 phi1 x2 y2 phi2 xp phip
global N Smin Smax spmin
global nfgr

a3=x(1,1)/3;  b2=x(2,1)/2;  c=x(3,1);  S=x(4,1); sp=x(5,1);

f = inf; g = zeros(rows(x),1);
if((Smin-S)>0) g(4,1)=-1; return; endif
if((S-Smax)>0) g(4,1)=1; return; endif

if((spmin-sp)>0) g(5,1)=-1; return; endif
if((sp-S)>0) g(4,1)=-1; g(5,1)=1; return; endif

ds = S/N; ss = ds*[1:N];
phi = phi1*ones(1,N) + ((a3*ss+b2).*ss+c).*ss;
[phiA imin] = min(phi); sA=ss(imin);
if((-pi/2-phiA)>0)
   g(1,1)=-sA*sA*sA/3; g(2,1)=-sA*sA/2; g(3,1)=-sA; return;
endif
if((phiA-pi/2)>0)
   g(1,1)=sA*sA*sA/3; g(2,1)=sA*sA/2; g(3,1)=sA; return;
endif
[phiB imax] = max(phi); sB=ss(imax);
if((-pi/2-phiB)>0)
   g(1,1)=-sB*sB*sB/3; g(2,1)=-sB*sB/2; g(3,1)=-sB; return;
```

```
endif
if((phiB-pi/2)>0)
   g(1,1)=sB*sB*sB/3; g(2,1)=sB*sB/2; g(3,1)=sB; return;
endif

# We calculate f = abs(x2-tmpx-x1) + abs(y2-tmpy-y1);

fcos = cos(phi); fsin = sin(phi);
tmpx = sum(fcos) + cos(phi1)/2 - fcos(1,N)/2;
tmpx = ds*tmpx;
tmpy = sum(fsin) + sin(phi1)/2 - fsin(1,N)/2;
tmpy = ds*tmpy;
tmp1 = x2 - tmpx - x1; tmp2 = y2 - tmpy - y1;
f = abs(tmp1) + abs(tmp2);
g(4,1) = - sign(tmp1)*fcos(1,N) - sign(tmp2)*fsin(1,N);

tmpsin = fsin.*ss;  tmpcos = fcos.*ss;
tmpc1 = sum(tmpsin) - tmpsin(1,N)/2;  tmpc1 = ds*tmpc1;
tmpc2 = sum(tmpcos) - tmpcos(1,N)/2;  tmpc2 = ds*tmpc2;
g(3,1) = sign(tmp1)*tmpc1 - sign(tmp2)*tmpc2;

tmpsin = tmpsin.*ss;  tmpcos = tmpcos.*ss;
tmpb1 = sum(tmpsin) - tmpsin(1,N)/2;  tmpb1=ds*tmpb1/2;
tmpb2 = sum(tmpcos) - tmpcos(1,N)/2;  tmpb2=ds*tmpb2/2;
g(2,1) = sign(tmp1)*tmpb1 - sign(tmp2)*tmpb2;

tmpsin = tmpsin.*ss;  tmpcos = tmpcos.*ss;
tmpa1 = sum(tmpsin) - tmpsin(1,N)/2;  tmpa1=ds*tmpa1/3;
tmpa2 = sum(tmpcos) - tmpcos(1,N)/2;  tmpa2=ds*tmpa2/3;
g(1,1) = sign(tmp1)*tmpa1 - sign(tmp2)*tmpa2;

# We calculate f = f + abs(phi2 - phi(S))
tmp3 = phi2 - ((a3*S+b2)*S+c)*S - phi1; f = f + abs(tmp3);
g(1,1) = g(1,1) - sign(tmp3)*S*S*S/3;
g(2,1) = g(2,1) - sign(tmp3)*S*S/2;
g(3,1) = g(3,1) - sign(tmp3)*S;
g(4,1) = g(4,1) - sign(tmp3)*(x(1,1)*S*S +
         x(2,1)*S + x(3,1));

# We calculate f = f + abs(xp-tmpx-x1);
ds = sp/N; ss = ds*[1:N];
phi = phi1*ones(1,N) + ((a3*ss+b2).*ss+c).*ss;

fcos = cos(phi); fsin = sin(phi);
tmpx = sum(fcos) + cos(phi1)/2 - fcos(1,N)/2;
tmpx = ds*tmpx;
tmp4 = xp - tmpx - x1; f = f + abs(tmp4);
g(5,1) = g(5,1) - sign(tmp4)*fcos(1,N);

tmpsin = fsin.*ss;
tmpc1 = sum(tmpsin) - tmpsin(1,N)/2;  tmpc1 = ds*tmpc1;
g(3,1) = g(3,1) + sign(tmp4)*tmpc1;

tmpsin = tmpsin.*ss;
```

```
tmpb1 = sum(tmpsin) - tmpsin(1,N)/2;  tmpb1=ds*tmpb1/2;
g(2,1) = g(2,1) + sign(tmp4)*tmpb1;

tmpsin = tmpsin.*ss;
tmpa1 = sum(tmpsin) - tmpsin(1,N)/2;  tmpa1=ds*tmpa1/3;
g(1,1) = g(1,1) + sign(tmp4)*tmpa1;

# We calculate f = f + abs(phip - phi(sp))
tmp5 = phip - ((a3*sp+b2)*sp+c)*sp - phi1;
f = f + abs(tmp5);
g(1,1) = g(1,1) - sign(tmp5)*sp*sp*sp/3;
g(2,1) = g(2,1) - sign(tmp5)*sp*sp/2;
g(3,1) = g(3,1) - sign(tmp5)*sp;
g(5,1) = g(5,1) -
         sign(tmp5)*(x(1,1)*sp*sp+x(2,1)*sp+x(3,1));

nfgr = nfgr + 1;
endfunction
```

Підпрограма plotO1 відображає вхідні дані задачі в графічне вікно. Її код наведено нижче.

```
function [dphi2] = pict1(a,b,c,S,s1,N)
global x1 y1 phi1 x2 y2 phi2 xp phip
s=0; ds=S/N;
x(1,1)=x1; y(1,1)=y1; ss(1)=s; k(1)=c; dk(1)=b; phi(1) = phi1;
tempx = 0; tempy = 0; t1 = phi1; fx1 = cos(t1); fy1 = sin(t1);
for i=1:N
   s=s+ds; t1 = phi1 + a*s*s*s/3 + b*s*s/2 + c*s;
   fx2 = cos(t1); fy2 = sin(t1);
   tempx = tempx + ds*(fx1+fx2)/2.0; fx1=fx2;
   tempy = tempy + ds*(fy1+fy2)/2.0; fy1=fy2;
   x(i+1,1)= x1 + tempx; y(i+1,1) = y1 + tempy;
   ss(i+1) = s; k(i+1) = a*s*s + b*s + c; dk(i+1) = 2*a*s + b;
   phi(i+1) = phi1 + a*s*s*s/3 + b*s*s/2 + c*s;
endfor
dphi2 = phi(N+1) - phi2;

s=0; ds=s1/N;
xx(1,1)=x1;yy(1,1)=y1; sss(1)=s;phi_1(1)=phi1; kk(1)=c;dkk(1)=b;
tempx = 0; tempy = 0; t1 = phi1; fx1 = cos(t1); fy1 = sin(t1);
for i=1:N
   s=s+ds; t1 = phi1 + a*s*s*s/3 + b*s*s/2 + c*s;
   fx2 = cos(t1); fy2 = sin(t1);
   tempx = tempx + ds*(fx1+fx2)/2.0; fx1=fx2;
   tempy = tempy + ds*(fy1+fy2)/2.0; fy1=fy2;
   xx(i+1,1)= x1 + tempx; yy(i+1,1) = y1 + tempy;
   sss(i+1) = s; kk(i+1) = a*s*s + b*s + c;
   dkk(i+1) = 2*a*s + b;
   phi_1(i+1) = phi1 + a*s*s*s/3 + b*s*s/2 + c*s;
endfor

hfig1=figure; figure(hfig1); #hold on;
```

```
subplot ( 2 , 2 , 1);
plot(x(1:(N+1),1),y(1:(N+1),1),'b','linewidth',2,
     x1,y1,'ro',xx(N+1,1),yy(N+1,1),'ro',x2,y2,'or');
title('y(x)'); grid('on');
axis equal;
subplot ( 2 , 2 , 2 );
plot(ss(1:(N+1)),phi(1:(N+1)),'r','linewidth',2,
     ss(1),phi(1),'b',sss(N+1),phi_1(N+1),'bo',
     ss(N+1),phi(N+1),'bo');
title('\phi(s)'); grid('on');
subplot ( 2 , 2 , 3 );
plot(ss(1:(N+1)),k(1:(N+1)),'g','linewidth',2,
     ss(1),k(1),'ro',ss(N+1),kk(N+1),'ro',
     ss(N+1),k(N+1),'ro');
title('k(s)'); grid('on');
subplot ( 2 , 2 , 4 );
plot(ss(1:(N+1)),dk(1:(N+1)),'g','linewidth',2,
     ss(1),dk(1),'ro',sss(N+1),dkk(N+1),'ro',
     ss(N+1),dk(N+1),'ro');
title('k''(s)'); grid('on');

endfunction
```

За допомогою програми Frankl-quadratic був розраховані фрагменти зовнішнього контуру сопла (рисунки 3.7 та 3.8), який вибраний за стартовий для побудови контуру центрального тіла.

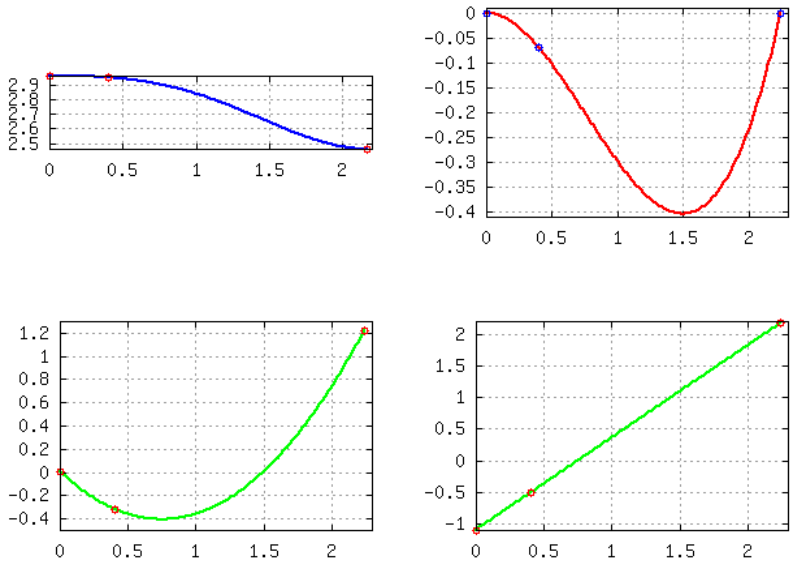


Рисунок 3.7 – Звужуюча частина зовнішнього контуру ($\mu = 100$)

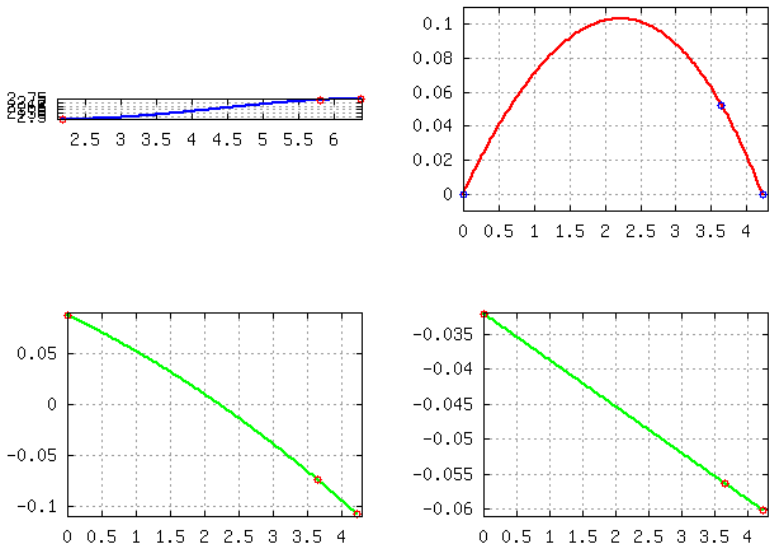


Рисунок 3.8 – Розширююча частина зовнішнього контуру ($\mu = 100$)

Для дозвукової частини зовнішнього контуру використовувались наступні вхідні параметри: $x_1 = 0.0,\ y_1 = 296,\ x_2 = 216.68,\ y_2 = 246.041,\ x_p = 40.0,\ \varphi_1 = -4^\circ$, а для побудови надзвукової частини сопла використовувались такі параметри: $x_1 = 216.68,\ y_1 = 246.041,\ x_2 = 638.0,\ y_2 = 275.0,,\ x_p = 580.0,\ \varphi_2 = 3^\circ$.

3.5 Висновки до третього розділу

Розроблено математичну модель, алгоритм та програмне забезпечення для задачі побудови S-подібної кривої в натуральній параметризації, яка проходить через дві задані точки із заданими кутами нахилу дотичних у них та забезпечує заданий кут нахилу дотичної в точці із заданою абсцисою. Сформульовано систему нелінійних інтегральних рівнянь для квадратичної кривини, досліджено її властивості, описано відповідну задачу мінімізації негладкої функції та алгоритм її розв'язання. Алгоритм базується на модифікації методу з розтягом простору в напрямі різниці двох послідовних узагальнених градієнтів. Проведено обчислювальні експерименти, які показали ефективність

розробленого алгоритму для проектування фрагментів дозвукової та надзвукової частин зовнішнього контуру сопла Франкля.

Розроблений алгоритм побудови зовнішнього контуру сопла за допомогою S-подібних кривих з використанням натуральної параметризації дозволяє управляти формою контуру за допомогою мінімальної кількості параметрів (кути нахилу дотичних в точках з відомими абсцисами у дозвуковій та надзвуковій частинах). При цьому кривина контуру змінюється плавно, забезпечуючи тим самим, зокрема, необхідні геометричні та газодинамічні властивості контуру. Обчислювальні експерименти підтверджують стійку роботу алгоритму для добре масштабованих вихідних даних контуру, який проектується. Використовуючи отриманий розв'язок за формулами (3.10) легко знайти розв'язки з необхідною точністю для погано масштабованих вихідних даних.

Розроблені математична модель, алгоритм та програмне забезпечення можуть бути використані для профілювання фрагментів сопел та перехідних каналів реактивних двигунів (Крайко 2014). Поряд з методом Безьє – Бернштейна розроблений алгоритм можна використовувати для побудови фрагментів профілів перехідних каналів змінного перерізу з необхідними геометричними властивостями. Так, наприклад, за його допомогою можна моделювати фрагменти профілів, які представляються опуклими (угнутими) функціями, як монотонно зростаючими, так і монотонно спадними. Вибір абсциси x_p та кута φ_p дозволяє управляти формою кривої таким чином, щоб в базисних (реперних) точках характеристики кривої відповідали заданим характеристикам профілю, що проектується.

Розроблені алгоритм та програмне забезпечення використовується для побудови зовнішнього контуру сопла з центральним тілом (Сергієнко та ін. 2020).

Матеріал розділу базується на матеріалах розділу з книги (Стецюк, Жидков та Хом'як 2023б) та звіту (Стецюк, Жидков та Хом'як 2023в).

РОЗДІЛ 4. ПОБУДОВА КРИВОЇ В НАТУРАЛЬНІЙ ПАРАМЕТРИЗАЦІЇ З КУБІЧНОЮ КРИВИНОЮ

В цьому розділі описано математичну модель, алгоритм та програмне забезпечення (Octave-програму **NaCCC (Na**tural**C**ubic**C**urvature**C**urve**)**) для задачі побудови кривої, яка проходить через дві задані точки із заданими у них кутами нахилу дотичних та значень кривини.

Математична модель представлена оптимізаційною задачею, еквівалентною системі чотирьох нелінійних рівнянь з чотирма невідомими, яка описує задачу побудови необхідної кривої в натуральній параметризації з кубічним розподілом кривини. Алгоритм розв’язання оптимізаційної задачі використовує модифікацію *r*-алгоритму. Програмне забезпечення мовою Octave реалізує алгоритм розв’язання оптимізаційної задачі та графічну складову для відображення графіку кривої та графіків залежності від довжини кривої трьох її характеристик – кут дотичної, кривина та похідна кривини.

Матеріал викладено в п’яти підрозділах. В підрозділі 4.1 наведено геометричну постановку задачі побудови кривої в натуральній параметризації з кубічною кривиною та розглянуто відповідну систему нелінійних інтегральних рівнянь (СНІР). В підрозділі 4.2 представлені оптимізаційна задача для знаходження розв'язку СНІР та метод її розв'язання з використанням модифікації *r*-алгоритму Шора. В підрозділі 4.3 описано програмне забезпечення для тестового прикладу, наведено коди Octave-програм та протокол роботи тестової програми. В підрозділі 4.4 наведено приклади застосувань. В підрозділі 4.5 досліджено питання масштабування для системи нелінійних рівнянь у задачі побудови кривої у натуральній параметризації з кубічним розподілом кривини, що проходить через дві задані точки та забезпечує в них задані кути нахилу дотичних та задані значення кривин.

4.1 Постановка задачі

Розглянемо таку задачу (див. рис. 4.1): необхідно з'єднати точки $A(x_A, y_A)$ та $B(x_B, y_B)$ двовимірною кривою в натуральній параметризації з кубічним законом розподілу кривини (крива визначається довжиною S, де кривина $k(s) = as^3 + bs^2 + cs + d$ має кубічну залежність від довжини кривої) таким чином, щоб в точках A і B досягалися задані значення кутів нахилу дотичних φ_A, φ_B та задані значення кривин k_A, k_B. Кути φ_A та φ_B вимірюються в радіанах.

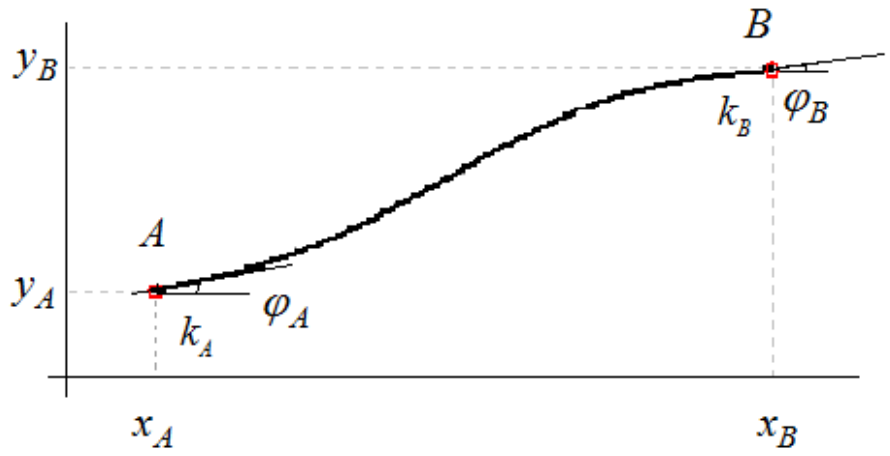


Рисунок 4.1 – Вихідні дані для геометричної моделі задачі

Для кривої $x(s)$, $y(s)$, $s > 0$ в натуральній параметризації кут нахилу дотичної $\varphi(s)$ в точці $(x(s), y(s))$ визначається за формулою

$$\varphi(s) = \varphi(0) + \int_0^s k(s)ds = \varphi(0) + \frac{as^4}{4} + \frac{bs^3}{3} + \frac{cs^2}{2} + d \times s, \qquad (4.1)$$

а координати $x(s)$ та $y(s)$ – за формулами

$$x(s) = x(0) + \int_0^s \cos\varphi(s)ds, \quad y(s) = y(0) + \int_0^s \sin\varphi(s)ds. \qquad (4.2)$$

Тут кривина $k(s) = as^3 + bs^2 + cs + d$ задається кубічною функцією від s – довжини кривої, де a, b, c, d – задані коефіцієнти.

Нехай S – довжина кривої від точки (x_A, y_A) до точки (x_B, y_B).

Знаходженню параметрів кривини a, b, c, d та довжини S відповідає система з чотирьох нелінійних рівнянь та одного лінійного рівняння (Стецюк, Ткаченко, та Ульянова 2019):

$$x_B = x_A + \int_0^S \cos\left(\varphi_A + \frac{as^4}{4} + \frac{bs^3}{3} + \frac{cs^2}{2} + d \times s\right) ds, \tag{4.3}$$

$$y_B = y_A + \int_0^S \sin\left(\varphi_A + \frac{as^4}{4} + \frac{bs^3}{3} + \frac{cs^2}{2} + d \times s\right) ds, \tag{4.4}$$

$$\varphi_B = \varphi_A + \frac{aS^4}{4} + \frac{bS^3}{3} + \frac{cS^2}{2} + d \times S, \tag{4.5}$$

$$k_A = d, \quad k_B = aS^3 + bS^2 + cS + d. \tag{4.6}$$

Система (4.3) – (4.6) має п'ять невідомих: a, b, c, d – чотири коефіцієнти квадратичної функції, S – загальна довжина кривої. Система включає чотири нелінійних рівнянь, серед яких рівняння (4.3) та (4.4) є інтегральними та залежать від невідомих параметрів підінтегральних функцій та невідомої верхньої границі для визначеного інтегралу.

Інтегральні рівняння (4.3) та (4.4) пов’язують координати точок $A(x_A, y_A)$ і $B(x_B, y_B)$ за формулами (4.2). Нелінійне рівняння (4.3) – це рівняння для кута φ_B в точці B, який згідно формули (4.1) визначається заданим кутом φ_A у точці A. Два рівняння (4.6), перше з яких є лінійним, задають умови для кривин $k_A = k(0)$ і $k_B = k(S)$ відповідно в точках A і B.

З лінійного рівняння в (4.6) випливає $d^\star = k_A$, отже, систему (4.3) – (4.6) можна замінити системою з чотирьох нелінійних рівнянь з чотирма невідомими a, b, c, S. Для пошуку розв'язків даної системи розглянемо метод на основі модифікації *r*-алгоритму для розв’язання задачі мінімізації негладкої функції (сума модулів нев’язок рівнянь (4.3) – (4.6)) при контролі обмеження на довжину S, щоб гарантувати її додатне допустиме значення.

4.2 Оптимізаційна задача та алгоритм її розв'язання

Розглянемо умовну задачу мінімізації суми модулів функцій нев'язок для рівнянь (4.3) – (4.6), яка має вигляд (Stetsyuk, Tkachenko, and Zhydkov 2020): знайти

$$f^\star = f(a^\star,b^\star,c^\star,S^\star) = \min_{a,b,c,S}\left\{ f(a,b,c,S) = \sum_{i=1}^{4} \left| f_i(a,b,c,S) \right| \right\} \tag{4.7}$$

при обмеженнях

$$S_{\min} \le S \le S_{\max}, \tag{4.8}$$

$$-\frac{\pi}{2} \le \varphi_A + a\frac{i^4S^4}{4N^4} + b\frac{i^3S^3}{3N^3} + c\frac{i^2S^2}{2N^2} + k_A\frac{iS}{N} \le \frac{\pi}{2}, \quad i = 1,\ldots,N, \tag{4.9}$$

де нев'язки для рівнянь (4.3) – (4.6) задаються такими функціями

$$f_1(a,b,c,S) = x_B - x_A - \int_0^S \cos\left(\varphi_A + \frac{as^4}{4} + \frac{bs^3}{3} + \frac{cs^2}{2} + k_A s \right) ds, \tag{4.10}$$

$$f_2(a,b,c,S) = y_B - y_A - \int_0^S \sin\left(\varphi_A + \frac{as^4}{4} + \frac{bs^3}{3} + \frac{cs^2}{2} + k_A s \right) ds, \tag{4.11}$$

$$f_3(a,b,c,S) = \varphi_B - \varphi_A - \frac{aS^4}{4} - \frac{bS^3}{3} - \frac{cS^2}{2} - k_A S, \tag{4.12}$$

$$f_4(a,b,c,s_p) = k_B - aS^3 - bS^2 - cS - k_A, \tag{4.13}$$

N – кількість рівних підінтервалів на інтервалі $[0,S]$ для дискретизації функції $\varphi(s) = \varphi_A + \frac{as^4}{4} + \frac{bs^3}{3} + \frac{cs^2}{2} + k_A s,\ s \in [0,S]$.

Тут цільова функція (4.7) є негладкою та означає мінімізацію суми модулів функцій (4.10) – (4.13) – функцій нев'язок для рівнянь (4.3) – (4.6). Обмеження (4.8) гарантують додатні значення для довжини S, яка є верхньою границею для визначених інтегралів в функціях нев'язок (4.10) та (4.11). Тут $S_{\min} = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$ – мінімальна відстань між точками (x_A, y_A) та (x_B, y_B), $S_{\max} > S_{\min}$ – параметр для управління верхнею межею на S – загальну довжину кривої.

Обмеження (4.9) забезпечує існування єдиного глобального мінімуму для задачі (4.7) – (4.9). Воно використовує доповнення задачі (4.7) – (4.8) дискретним аналогом неперервного обмеження

$$-\frac{\pi}{2} \le \varphi(s) = \varphi_A + \frac{as^4}{4} + \frac{bs^3}{3} + \frac{cs^2}{2} + k_A s \le \frac{\pi}{2}, \quad s \in [0, S]. \tag{4.14}$$

Обмеження (4.14) означає, що кути нахилу дотичних в довільній точці кривої не виходять за заданий діапазон $\varphi(s) \in \left[-\pi/2, \pi/2\right]$, тобто функція $y(x)$ на інтервалі $\left[x_A, x_B\right]$ є однозначно визначеною. Якщо $S_{\max} \ge S^\star$, то задача (4.7) – (4.9) має єдину точку глобального мінімуму, яка співпадає з розв'язком системи (4.3) – (4.6) з найменшою загальною довжиною кривої.

Якщо в результаті пошуку локального мінімуму для задачі (4.7) – (4.9) отримуємо $f^\star = 0$, то це означає, що знайдена точка глобального мінімуму $(a^\star, b^\star, c^\star, S^\star)^T$, компоненти якої у сукупності з $d^\star = k_A$ є розв'язком системи (4.3) – (4.6). Якщо отримуємо $f^\star > 0$, то точка $(a^\star, b^\star, c^\star, d^\star = k_A, S^\star)^T$ не є розв'язком системи (4.3) – (4.6).

Задача (4.7) – (4.9) є задачею мінімізації негладкої функції, яка визначена не при всіх значеннях S, а тільки при тих, які є додатними та дозволяють обчислювати визначені інтеграли для функцій $f_1(a,b,c,S)$ та $f_2(a,b,c,S)$. Для знаходження точки глобального мінімуму у задачі (4.7) – (4.9) може бути використана модифікація *r*-алгоритму (Шор и Стецюк 1997), яка враховує вказану особливість задачі. У точці, де узагальнений градієнт цільової функції є невизначеним, модифікація *r*-алгоритму використовує узагальнений градієнт до одного із порушених обмежень (4.8) та (4.9).

Наведемо формули для обчислення узагальненого градієнту цільової функції (4.7). Нехай вектор $g_f = \left(\frac{\partial f}{\partial a}, \frac{\partial f}{\partial b}, \frac{\partial f}{\partial c}, \frac{\partial f}{\partial S}\right)^T$ є узагальненим градієнтом цільової функції $f(a,b,c,S) = \sum_{i=1}^{4} \left| f_i(a,b,c,S) \right|$. Компоненти узагальненого

градієнта обчислюються за формулами:

$$\frac{\partial f}{\partial a}=\sum_{i=1}^{4} sign(f_i)\frac{\partial f_i}{\partial a},\quad \frac{\partial f}{\partial b}=\sum_{i=1}^{4} sign(f_i)\frac{\partial f_i}{\partial b},$$
$$\frac{\partial f}{\partial c}=\sum_{i=1}^{4} sign(f_i)\frac{\partial f_i}{\partial c},\quad \frac{\partial f}{\partial S}=\sum_{i=1}^{4} sign(f_i)\frac{\partial f_i}{\partial S}, \tag{4.15}$$

де

$$\frac{\partial f_1}{\partial a}=+\int_0^S \frac{s^4}{4}\sin\left(\varphi_A+\frac{as^4}{4}+\frac{bs^3}{3}+\frac{cs^2}{2}+k_A s\right)ds,\quad \frac{\partial f_3}{\partial a}=-\frac{S^4}{4},$$
$$\frac{\partial f_2}{\partial a}=-\int_0^S \frac{s^4}{4}\cos\left(\varphi_A+\frac{as^4}{4}+\frac{bs^3}{3}+\frac{cs^2}{2}+k_A s\right)ds,\quad \frac{\partial f_4}{\partial a}=-S^3, \tag{4.16}$$

$$\frac{\partial f_1}{\partial b}=+\int_0^S \frac{s^3}{3}\sin\left(\varphi_A+\frac{as^4}{4}+\frac{bs^3}{3}+\frac{cs^2}{2}+k_A s\right)ds,\quad \frac{\partial f_3}{\partial b}=-\frac{S^3}{3},$$
$$\frac{\partial f_2}{\partial b}=-\int_0^S \frac{s^3}{3}\cos\left(\varphi_A+\frac{as^4}{4}+\frac{bs^3}{3}+\frac{cs^2}{2}+k_A s\right)ds,\quad \frac{\partial f_4}{\partial b}=-S^2, \tag{4.17}$$

$$\frac{\partial f_1}{\partial c}=+\int_0^S \frac{s^2}{2}\sin\left(\varphi_A+\frac{as^4}{4}+\frac{bs^3}{3}+\frac{cs^2}{2}+k_A s\right)ds,\quad \frac{\partial f_3}{\partial c}=-\frac{S^2}{2},$$
$$\frac{\partial f_2}{\partial c}=-\int_0^S \frac{s^2}{2}\cos\left(\varphi_A+\frac{as^4}{4}+\frac{bs^3}{3}+\frac{cs^2}{2}+k_A s\right)ds,\quad \frac{\partial f_4}{\partial c}=-S, \tag{4.18}$$

$$\frac{\partial f_1}{\partial S}=-\cos\left(\varphi_A+\frac{aS^4}{4}+\frac{bS^3}{3}+\frac{cS^2}{2}+k_A S\right),\ \frac{\partial f_3}{\partial S}=-aS^3-bS^2-cS-k_A,$$
$$\frac{\partial f_2}{\partial S}=-\sin\left(\varphi_A+\frac{aS^4}{4}+\frac{bS^3}{3}+\frac{cS^2}{2}+k_A S\right),\ \frac{\partial f_4}{\partial a}=-3S^2-2S-c. \tag{4.19}$$

У точці, де узагальнений градієнт цільової функції є невизначеним, для модифікації *r*-алгоритму він замінюється на узагальнений градієнт до одного із порушених двосторонніх обмежень (4.8) або (4.9). Для цього використовуються чотири порушених обмеження $g_i(\circ)\le 0$, $i=\overline{1,4}$, які вибираються у такому порядку: $g_1(S)=S_{\min}-S$, $g_2(S)=S-S_{\max}$ – для двосторонніх обмежень (4.8),

$$g_5(a,b,c,S)=\max_{i=1}^{N}\left\{-\frac{\pi}{2}-\varphi_A-a\frac{i^4S^4}{4N^4}-b\frac{i^3S^3}{3N^3}-c\frac{i^2S^2}{2N^2}-k_A\frac{iS}{N}\right\},$$

$$g_4(a,b,c,S)=\max_{i=1}^{N}\left\{\varphi_A+a\frac{i^4S^4}{4N^4}+b\frac{i^3S^3}{3N^3}+c\frac{i^2S^2}{2N^2}+k_A\frac{iS}{N}-\frac{\pi}{2}\right\}$$

– для двосторонніх обмежень (4.9). Якщо обмеження (4.8), (4.9) виконуються, то тоді використовується узагальнений градієнт цільової функції (4.7), який обчислюється за формулами (4.15) – (4.19).

Алгоритм розвязання задачі (4.7) – (4.9) реалізований мовою Octave за допомогою Octave-функції **ralgb5a**. Він або знаходить глобальний мінімум цільової негладкої функції, або сигналізує, що система обмежень (4.8) – (4.9) є несумісною. Це може бути як у випадку відсутності розв'язку у системи (4.3) – (4.6) при обмеженнях (4.8), (4.9), так і у випадку, якщо алгоритм зупиниться в «неоптимальній» точці, враховуючи, що для великих значень параметра $S_{\max}$ задача (4.7) – (4.8), в якій не враховуються обмеження (4.9), є багатоекстремальною.

Узагальнені градієнти цільової функції (4.7) обчислюються за формулами (4.15) – (4.19), а визначені інтеграли, як у формулах (4.10), (4.11) так і у формулах (4.16), (4.17), (4.18) для компонент узагальнених градієнтів, обчислюються за методом трапецій. Наприклад, у формулі (4.10) визначений інтеграл обчислюється за формулою

$$\int_0^S\cos\left(\varphi(s)\right)ds\approx\left(\frac{\cos\varphi(0)-\cos\varphi(S)}{2}+\sum_{i=1}^{N}\cos\left(\varphi(s_i)\right)\right)h=$$

$$=\left(\frac{\cos\varphi(0)+\cos\varphi(S)}{2}+\sum_{i=1}^{N-1}\cos\left(\varphi(s_i)\right)\right)h,$$

де $h=S/N$, N – кількість підінтервалів на інтервалі $s\in[0,S]$, для дискретизації функції $\cos\left(\varphi(s)\right)=\cos\left(\varphi_A+\frac{as^4}{4}+\frac{bs^3}{3}+\frac{cs^2}{2}+k_As\right)$, а визначений інтеграл із (4.16) обчислюється за формулою

$$\int_0^S \frac{s^4}{4}\sin\left(\varphi(s)\right)ds = \left(-\frac{S^4}{4}\sin\left(\varphi(S)\right) + \sum_{i=1}^{N}\frac{s_i^4}{4}\sin\left(\varphi(s_i)\right)\right)h =$$

$$= \left(\frac{S^4}{4}\sin\left(\varphi(S)\right) + \sum_{i=1}^{N-1}\frac{s_i^4}{4}\sin\left(\varphi(s_i)\right)\right)h.$$

Octave функція **fgK1** реалізує обчислення **[f,g]** – скалярної величини **f** та 4-вимірного вектора **g** в точці **x=**$(a,b,c,S)^T$. Якщо точка **x** задовільняє обмеженням (4.8), (4.9), то **f** – значення цільової функції (4.7), **g** – узагальнений градієнт цільової функції, а якщо точка **x** не задовільняє обмеженням (4.8), (4.9), то **f=**$+\infty$, а **g** – узагальнений градієнт до одного із порушених обмежень, яке вибирається у такому порядку (4.8) та (4.9). Код програми **fgK1** наведено нижче.

```
function [f,g] = fgK1(x)
global xA yA phiA kA xB yB phiB kB
global N Smin Smax
global nfgr

a = x(1,1); b = x(2,1); c = x(3,1);  S = x(4,1);
a4 = a/4; b3 = b/3; c2 = c/2; S2 = S*S;

f = inf; g = zeros(rows(x),1);

# Calculate generalized gradient for constraints (5.8)
# Обчислюємо узагальнені градієнти для обмежень (5.8)
if((Smin-S)>0) g(4,1)=-1; return; endif
if((S-Smax)>0) g(4,1)=1; return; endif

ds = S/N; ss = ds*[1:N];
phi = phiA*ones(1,N) + (((a4*ss+b3).*ss+c2).*ss+kA).*ss;

# Calculate generalized gradient for constraints (5.9)
# Обчислюємо узагальнені градієнти для обмежень (5.9)
[phimin imin] = min(phi); sA=ss(imin); sA2=sA*sA;
if((-pi/2-phimin)>0)
  g(1,1) = -sA2*sA2/4; g(2,1) = -sA2*sA/3; g(3,1) = -sA2/2;
  if(imin==N) g(4,1) = -(a*sA2*sA+b*sA2+c*sA+kA); endif
  return;
endif
if((phimin-pi/2)>0)
  g(1,1) = sA2*sA2/4;  g(2,1) = sA2*sA/3; g(3,1) = sA2/2;
  if(imin==N) g(4,1) = a*sA2*sA+b*sA2+c*sA+kA; endif
  return;
endif
[phimax imax] = max(phi); sB=ss(imax); sB2=sB*sB;
if((-pi/2-phimax)>0)
  g(1,1) = -sB2*sB2/4; g(2,1) = -sB2*sB/3; g(3,1) = -sB2/2;
  if(imax==N) g(4,1) = -(a*sB2*sB+b*sB2+c*sB+kA); endif
  return;
endif
```

```
if((phimax-pi/2)>0)
  g(1,1) = sB2*sB2/4; g(2,1) = sB2*sB/3; g(3,1) = sB2/2;
  if(imax==N) g(4,1) = a*sB2*sB+b*sB2+c*sB+kA; endif
  return;
endif

# Sum of moduli (5.10) and (5.11) and their generalized gradient
# Сума модулів (5.10),(5.11) та їх узагальнений градієнт
fcos = cos(phi); fsin = sin(phi);
tmpx = sum(fcos) + cos(phiA)/2 - fcos(1,N)/2;  tmpx = ds*tmpx;
tmpy = sum(fsin) + sin(phiA)/2 - fsin(1,N)/2;  tmpy = ds*tmpy;
tmp1 = xB - tmpx - xA; tmp2 = yB - tmpy - yA;
f = abs(tmp1) + abs(tmp2);
g(4,1) = -(sign(tmp1)*fcos(1,N) - sign(tmp2)*fsin(1,N));
tmpsin = fsin.*ss.*ss;  tmpcos = fcos.*ss.*ss;
tmpc1 = sum(tmpsin) - tmpsin(1,N)/2;  tmpc1=ds*tmpc1/2;
tmpc2 = sum(tmpcos) - tmpcos(1,N)/2;  tmpc2=ds*tmpc2/2;
g(3,1) = (sign(tmp1)*tmpc1 - sign(tmp2)*tmpc2);
tmpsin = tmpsin.*ss;  tmpcos = tmpcos.*ss;
tmpb1 = sum(tmpsin) - tmpsin(1,N)/2;  tmpb1=ds*tmpb1/3;
tmpb2 = sum(tmpcos) - tmpcos(1,N)/2;  tmpb2=ds*tmpb2/3;
g(2,1) = (sign(tmp1)*tmpb1 - sign(tmp2)*tmpb2);
tmpsin = tmpsin.*ss;  tmpcos = tmpcos.*ss;
tmpa1 = sum(tmpsin) - tmpsin(1,N)/2;  tmpa1=ds*tmpa1/4;
tmpa2 = sum(tmpcos) - tmpcos(1,N)/2;  tmpa2=ds*tmpa2/4;
g(1,1) = (sign(tmp1)*tmpa1 - sign(tmp2)*tmpa2);

# Add modulus of (5.12) and its generalized gradient
# Додати модуль (5.12) та його узагальнений градієнт
tmp3 = phiB - (((a4*S+b3)*S+c2)*S+kA)*S - phiA; f = f+abs(tmp3);
g(1,1) -= sign(tmp3)*S2*S2/4; g(2,1) -= sign(tmp3)*S2*S/3;
g(3,1)-=sign(tmp3)*S2/2;g(4,1)-=sign(tmp3)*(a*S2*S+b*S2+c*S+kA);

# Add modulus of (5.13) and its generalized gradient
# Додати модуль (5.13) та його узагальнений градієнт
tmp4 = kB - a*S2*S - b*S2 - c*S - kA; f = f + abs(tmp4);
g(1,1) -= sign(tmp4)*S2*S;   g(2,1) -= sign(tmp4)*S2;
g(3,1) -= sign(tmp4)*S; g(4,1) -= sign(tmp4)*(3*a*S2+2*b*S+c);

nfgr = nfgr + 1; # цільова функція та її узагальнений градієнт
             # objective function and its generalized gradient
endfunction #fgK1
```

Octave-функція **fgK1** викликається програмою **ralgb5a**, її ім’я передається через перший формальний параметр при виклику програми **ralgb5a** із головної програми **NaCCC** (див. підрозділ 4.1).

4.3 Опис програмного забезпечення

Загальна структура головної програми **NaCCC**, яка розв’язує оптимізаційну задачу (4.7) – (4.9) та візуалізує її розв’язок, показана на рисунку 4.2. Вона

використовує три підпрограми: **ralgb5a** – реалізує *r*-алгоритм; **fgK1** – обчислює значення функції та узагальненого градієнта; **plotK1 –** забезпечує графічний вигляд знайденої кривої та графіки залежності від довжини кривої трьох її характеристик – кут дотичної, кривина та похідна кривини.

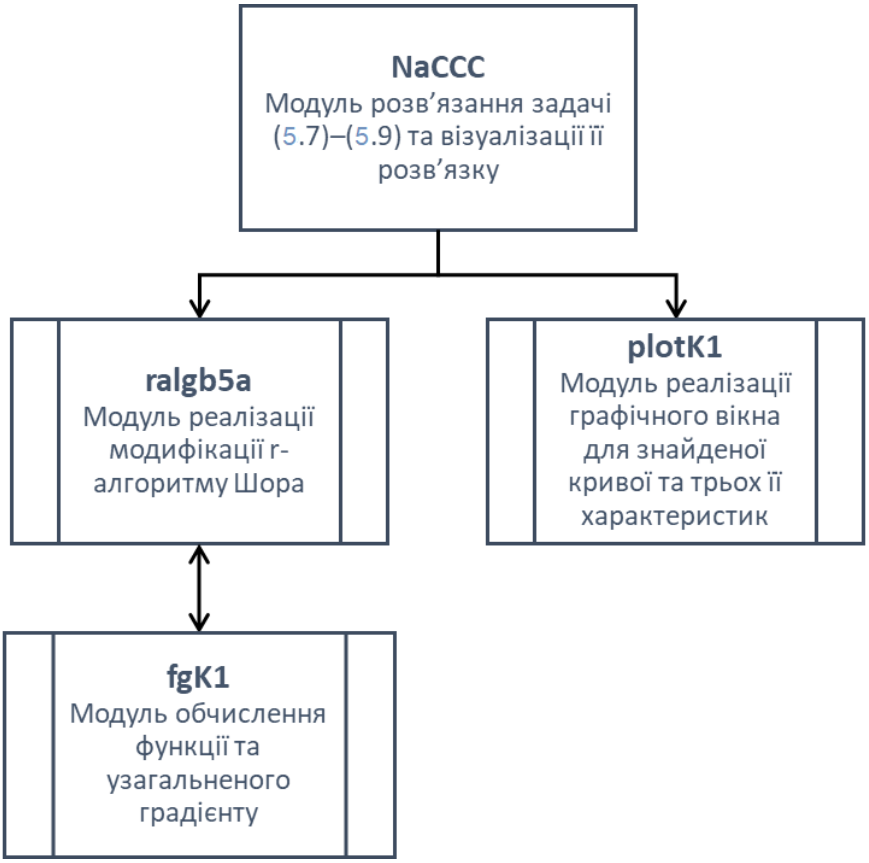


Рисунок 4.2 – Блок-схема головної програми **NaCCC**

Нижче наведемо короткий опис та коди мовою Octave програми **NaCCC** та підпрограми **plotK1.** Опис підпрограми **fgK1**, яка використовується підпрограмою **r**algb5a, та її Octave-код наведено в підрозділі 4.2.

В програмі **NaCCC** задаються вихідні дані та параметри r-алгоритму; компоненти стартової точки для коефіцієнтів квадратичної функції вибираються рівними нулю; обчислюється значення параметрів **Smin** ($S_{\min} = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$) та **Smax** ($S_{\max} = 1.2 \times S_{\min}$) для двостороннього обмеження (4.8); виконується процесс оптимізації для розв'язання задачі (4.7) – (4.9); аналізується отриманий розв'язок (перевіряється чи нев'язка більша

заданої величини). Результати розрахунку виводяться у файл протоколу роботи програми. Графіки кривої, кутів дотичних, кривин та похідних кривини відображаються на екран та зберігаються у файлах форматів ***.eps** та ***.png**.

Octave код програми **NaCCC** з короткими коментарями наведено нижче.

```
#  Code of main program NaCCC
#  код  головної  програми  NaCCC
#
global xA yA phiA kA xB yB phiB kB
global N Smin Smax
global nfgr

# initialize and print input data(test example)
# задаємо та друкуємо вихідні дані (тестовий приклад)
xA = 0.0;  yA = 1.0;  phiA = (10.0/180.0)*pi; kA = 0.0;
xB = 3.0;  yB = 2.0;  phiB = (5.0/180.0)*pi; kB = 0.0;
printf("xA yA phiA kA %13.5e %13.5e %13.5e = %13.5e %13.5e\n",
        xA, yA, phiA, 180*(phiA/pi), kA);
printf("xB yB phiB kB %13.5e %13.5e %13.5e = %13.5e %13.5e\n",
        xB, yB, phiB, 180*(phiB/pi), kB);

# calculate parameters for constaints (5.8)
# обчислюємо параметри для обмежень (5.8)
Smin=sqrt((xB-xA)**2+(yA-yB)**2); Smax=1.01*Smin;

printf("\n"); # задаємо параметри r-алгоритму
              # set parameters for r-algorithm
alpha = 2.0, h0 = 1.0, q1 = 1.00,
epsx = 1.e-8, epsg = 1.e-8, maxitn = 1500, intp=20,

printf("\n");         # задаємо N - дискретизацію інтервалу
                      # set N - interval discretization
N = 200,  eps=1.d-4,  # eps - допуск на сумарну нев'язку
                      # eps - constraint for summary error
nfgr = 0; # кількість обчислень цільової функції та її градієнта
#number of calculations of objective function and its gradient

# set start point and run r-algorithm
# вибираємо стартову точку та запускаємо r-алгоритм
x0 = zeros(3,1); x0(4,1) = Smin;
printf("x0(tr) = %9.5f %9.5f %9.5f %9.5f \n",x0(1:4));
[xr,fr,itn,nfg,istop] = ralgb5a(@fgK1,x0,alpha,h0,q1,
                                epsg,epsx,maxitn,intp);
printf("..itn %4d  fr %23.15e  istop %d  nfg(-) %4d(%3d)\n",
        itn, fr, istop, nfg, nfg-nfgr);

if (fr >= eps) # зупиняємось "не знайдено розв'язок задачі"
               # stop "problem solution not found"
  printf("\n error(1): fr = %9.2e > eps = %9.2e \n",fr,eps);
#  exit;
endif

# save and print solution (parameters of the found curve)
# запам'ятовуємо і друкуємо розв'язок (параметри кривої)
a=xr(1,1); b=xr(2,1); c=xr(3,1); d = kA; S=xr(4,1);
```

```
printf("\n");
printf(" fr %9.2e\n",fr);
printf(" a b c d S  %13.5e %13.5e %13.5e %13.5e %13.5e\n",
        a,b,c,d,S);
plotK1(a,b,c,d,S,200);          # будуємо графіки
                                 # draw plots
print -deps figure_NaCCC.eps     # зберігаємо їх у форматі eps
                                 # save them  in .eps format
print -dpng figure_NaCCC.png     # та у форматі png
                                 # and in .png format
pause  # нажати "Enter" для закінчення екранного режиму
       # press "Enter" to exit screen mode
```

Якщо розв'язок оптимізаційної задачі (4.7) – (4.9) знайдено, то головна програма **NaCCC** візуалізує знайдений розв'язок за допомогою підпрограми **plotK1**. Якщо розв'язок не знайдено, то видається повідомлення «error(1)» про те, що значення отриманої нев'язки (**fr**) є більшим за задане значення допуску на відхилення нев'язки від нуля (**eps**). У цьому випадку потрібно збільшити значення **Smax**, замінивши оператор `"Smax=1.2*Smin"` на оператор `"Smax=1.5*Smin"`.

Octave-функція **plotK1** готує та відображає на екран чотири графіки – графік кривої в натуральній параметризації за квадратичним законом розподілу кривини та графіки трьох характеристик цієї кривої. Вхідними параметрами функції є **a**, **b**, **c** – три коефіцієнти квадратичної функції, **S** – загальна довжина кривої, **sp** – довжина ділянки кривої до точки з відомою абсцисою, **N** – кількість підінтервалів функції $\cos\left(\varphi_A + as^4/4 + bs^3/3 + cs^2/2 + ds\right)$ на інтервалі $s \in [0, S]$.

Код octave–функції **plotK1** наведено нижче.

```
function plotK1(a,b,c,d,S,N)

global xA yA phiA kA xB yB phiB kB

s=0; ds=S/N; # готуємо масиви для графіків
             # prepare arrays for plots
x(1,1)=xA; y(1,1)=yA; ss(1)=s; k(1)=d; dk(1)=c; phi(1) = phiA;
tempx = 0; tempy = 0; t1 = phiA; fx1 = cos(t1); fy1 = sin(t1);
for i=1:N
   s=s+ds; t1 = phiA + a*s*s*s*s/4 + b*s*s*s/3 + c*s*s/2 + d*s;
   fx2 = cos(t1);  tempx = tempx + ds*(fx1+fx2)/2.0;  fx1=fx2;
   fy2 = sin(t1);  tempy = tempy + ds*(fy1+fy2)/2.0;  fy1=fy2;
   ss(i+1) = s; x(i+1,1)= xA + tempx; y(i+1,1) = yA + tempy;
   phi(i+1) = t1; k(i+1) = a*s*s*s + b*s*s + c*s + d;
   dk(i+1) = 3*a*s*s + 2*b*s + c;
endfor
```

```
# draw curve plots and display its three geometric parameters
# будуємо графіки кривої та її 3-х геометричних характеристик
hfig1=figure; figure(hfig1); #hold on;
subplot ( 2 , 2 , 1);
plot(x(1:(N+1),1),y(1:(N+1),1),'b','linewidth',2,
     [xA xB],[yA yB],'ro');
title('y(x)'); grid('on');
axis equal;
subplot ( 2 , 2 , 2 );
plot(ss(1:(N+1)),phi(1:(N+1)),'r','linewidth',2,
     [0 S],[phiA phiB],'bo');
title('\phi(s)'); grid('on');
subplot ( 2 , 2 , 3 );
plot(ss(1:(N+1)),k(1:(N+1)),'g','linewidth',2,
     [0 S],[d a*S*S*S+b*S*S+c*S+d],'ro');
title('k(s)'); grid('on');
subplot ( 2 , 2 , 4 );
plot(ss(1:(N+1)),dk(1:(N+1)),'g','linewidth',2,
     [0 S], [c 3*a*S*S+2*b*S+c],'ro');
title('k''(s)'); grid('on');

endfunction #plotK1
```

Усі чотири графіки функція **plotK1** розміщує в одному графічному 2×2–вікні. Графік кривої – вверху, ліворуч; графік кутів дотичних – вверху, праворуч; графік кривини – внизу, ліворуч, а графік похідних кривини – вверху, праворуч. Приклад графічного 2×2–вікна наведено на рисунку 4.3.

Код програми **NaCCC** орієнтований на такі вихідні дані: $x_1 = 0,\ y_1 = 1,\ k_1 = 0,$ $x_2 = 3,\ y_2 = 2,\ k_2 = 0.$ Протокол роботи програми **NaCCC** наведено нижче.

```
GNU Octave, version 5.1.0
Copyright (C) 2019 John W. Eaton and others.
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'.

xA yA phiA kA 0.00000e+00 1.00000e+00 1.74533e-01 = 1.00000e+01 0.00000e+00
xB yB phiB kB 3.00000e+00 2.00000e+00 8.72665e-02 = 5.00000e+00 0.00000e+00
alpha =  2
h0 =  1
q1 =  1
epsx =  0.000000010000
epsg =  0.000000010000
maxitn =  2500
intp =  20
```

```
N =  200
eps =  0.00010000
x0(tr) =   0.00000   0.00000   0.00000   3.16228
itn    0  f   5.927079e-01  fr   5.927079e-01  nfg    1
itn   20  f   1.897448e-01  fr   1.382702e-01  nfg   37  lsa  36  lsm   4
itn   40  f   3.260452e-03  fr   3.260452e-03  nfg   60  lsa  23  lsm   2
itn   60  f   2.454267e-05  fr   2.454267e-05  nfg   87  lsa  27  lsm   2
itn   80  f   3.894891e-06  fr   3.894891e-06  nfg  116  lsa  29  lsm   3
itn  100  f   9.080045e-08  fr   9.080045e-08  nfg  142  lsa  26  lsm   2
..itn  113  fr   7.532582901176310e-09  istop 3  nfg(-)  161( 20)

 fr  7.53e-09
 a b c d S    2.21464e-01  -1.04308e+00   1.07430e+00   0.00000e+00   3.18855e+00
```

На рисунку 4.3 наведено графіки, які програмою **NaCCC** відображаються на екран та зберігаються у файлах **figure_NaCCC.eps** та **figure_NaCCC.png**.

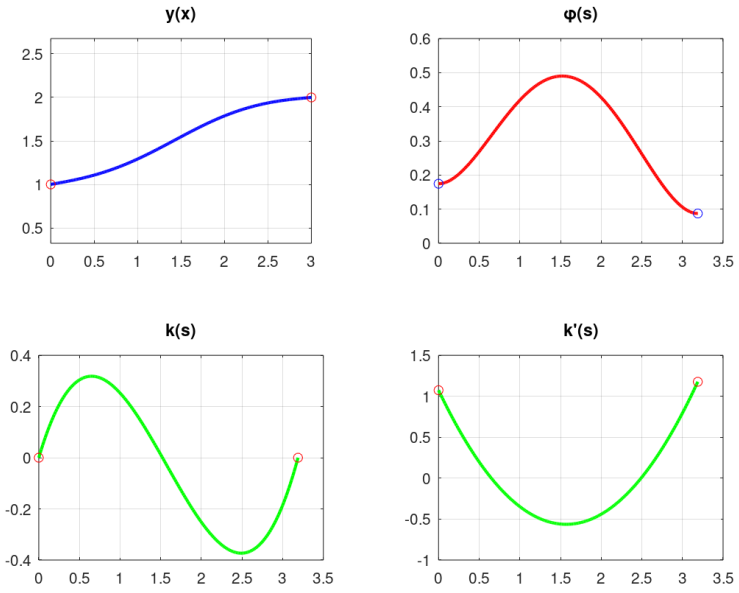


Рисунок 4.3 – Графік побудованої кривої та графіки трьох її характеристик

Якщо у програмі **NaCCC** оператор "Smax=1.2*Smin" замінити на оператор "Smax=1.005*Smin", то протокол роботи програми буде таким:

```
GNU Octave, version 5.1.0
Copyright (C) 2019 John W. Eaton and others.
xA yA phiA kA 0.00000e+00 1.00000e+00 1.74533e-01 = 1.00000e+01 0.00000e+00
xB yB phiB kB 3.00000e+00 2.00000e+00 8.72665e-02 = 5.00000e+00 0.00000e+00

alpha =  2
h0 =  1
q1 =  1
epsx =  0.000000010000
epsg =  0.000000010000
maxitn =  2500
intp =  20
```

```
N =  200
eps =  0.00010000
x0(tr) =   0.00000   0.00000   0.00000   3.16228
itn    0  f   5.927079e-01  fr   5.927079e-01  nfg    1
itn   20  f   4.799532e-01  fr   2.753513e-01  nfg   37  lsa  36  lsm   4
itn   40  f            Inf  fr   1.376621e-02  nfg   68  lsa  31  lsm   3
itn   60  f            Inf  fr   1.140702e-02  nfg   97  lsa  29  lsm   3
itn   80  f   1.126935e-02  fr   1.126935e-02  nfg  131  lsa  34  lsm   5
itn  100  f            Inf  fr   1.125147e-02  nfg  163  lsa  32  lsm   3
..itn  109  fr   1.125113390039975e-02  istop 3  nfg(-)  175( 44)

 error(1): fr =  1.13e-02 > eps =  1.00e-04
```

Це означає, що розв'язок оптимізаційної задачі (4.7) – (4.10) не знайдено, про що сигналізує той факт, що значення отриманої сумарної нев'язки (**fr**) є більшим за задане значення допуску на відхилення нев'язки від нуля (**eps**).

4.4 Приклади застосувань

Алгоритм розв'язання задачі (4.7) – (4.9) перевірений на ряді задач для побудови геометричної моделі сопла Лаваля з центральним тілом. На рисунку 4.4 наведено вигляд фрагменту зовнішнього контура сопла в надзвуковій області для різних значень k_1.

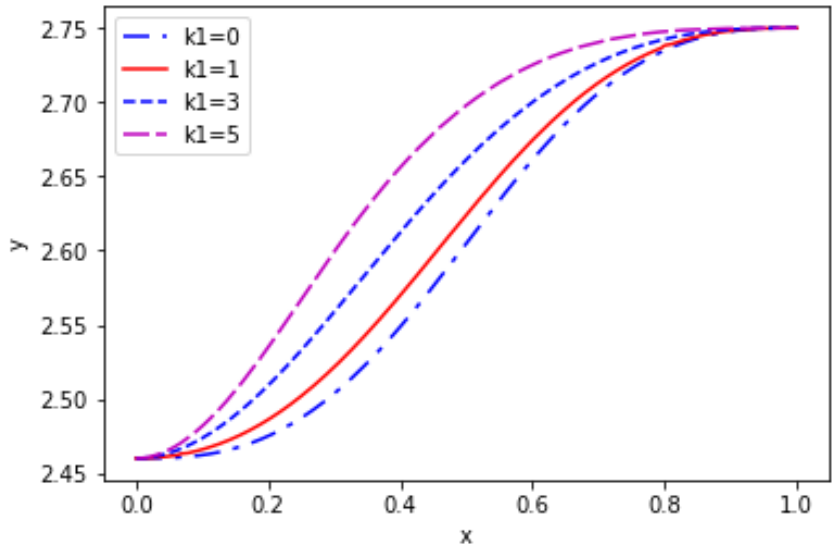


Рисунок 4.4 – Фрагменти зовнішнього контура сопла для різних k_1

У задачі (4.7) – (4.9) за допомогою задання граничних умов на кінцях інтервалу можна будувати різні S-подібні криві (мають одну точку перегину) за заданим графіком кривини. За рахунок різних положень точок перегину

всередині інтервалу можна моделювати різні зовнішні контури аеродинамічних сопел (Лаваля, Франкля, Стентона). Результати розв'язання задачі (4.7) – (4.9) для різних значень кривини $k_1 = \{0,1,3,5\}$ та єдиного значення $k_2 = 0$ наведено в звіті (Стецюк та ін. 2019, с. 76–77). Даний метод можна використовувати для профілювання лопаток компресора шляхом задання скелетної лінії профілю за допомогою S-подібної кривої.

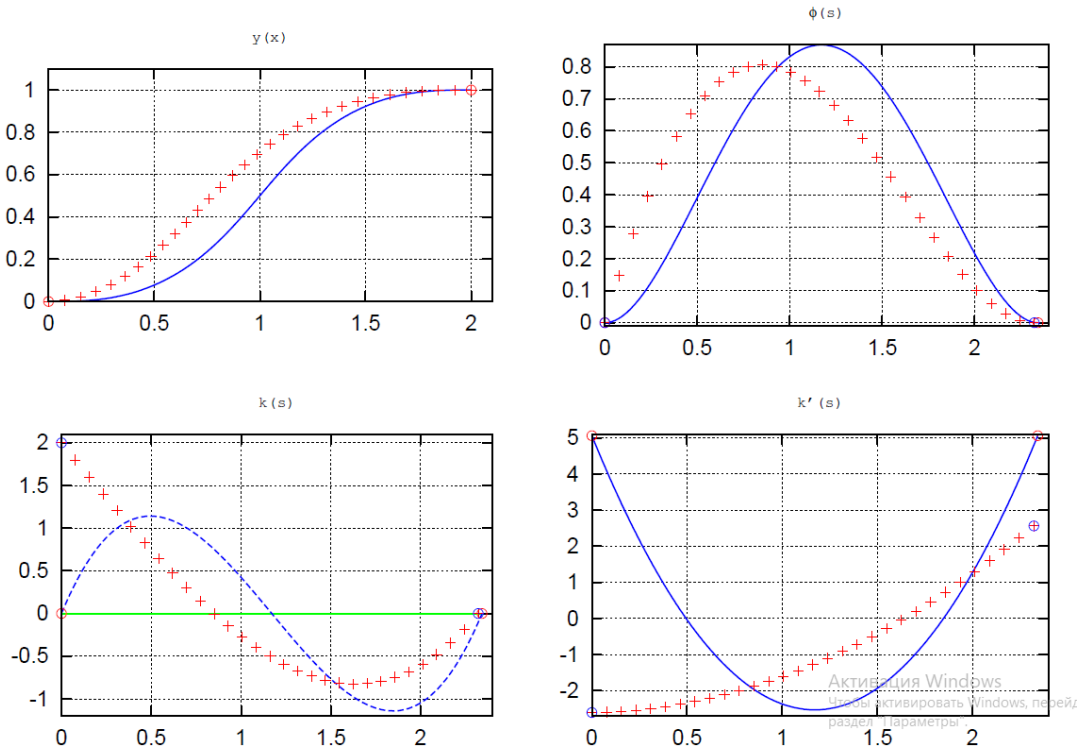


Рисунок 4.5 – Дві S-подібні криві для системи (4.3) – (4.6): $k_1 = 0$, $k_1 = 2$

Порівняння двох розв’язків системи (4.3) – (4.6) для $x_1 = y_1 = 0$, $x_2 = 2$, $y_2 = 1$, $\varphi_1 = \varphi_2 = 0$, $k_2 = 0$ та двох значень $k_1 = 0$ і $k_1 = 2$, що відповідають S-подібним кривим, представлено на рисунку 4.5. На рисунку побудовано суцільні та пунктирні криві (зверху, ліворуч), графіки дотичних кутів (зверху, праворуч), графіки кривини та її похідної (внизу). З графіків кривин видно, що обидві криві є S-подібними, оскільки вони мають єдину точку перегину, що відповідає нульовому значенню кривини.

Алгоритм розв’язання задачі (4.7) – (4.9) використано у методі апроксимації (згладжування) контуру центрального тіла, який полягає у представленні двох

його ланок за допомогою двох S-подібних кривих, де перша крива моделює дозвукову ланку центрального тіла, а друга крива моделює надзвукову ланку центрального тіла. Кожна з цих S-подібних кривих є плоскою кривою з кубічним законом розподілу кривини в натуральній параметризації. Вона отримується або в результаті мінімізації суми квадратів відстаней від кривої до заданого набору точок контуру (метод найменших квадратів), або в результаті мінімізації суми відстаней від кривої до заданого набору точок контуру (метод найменших модулів). На відміну від інших методів згладжування, в тому числі методів згладжування за допомогою сплайн-функцій, тут забезпечується плавна зміна кривини контуру, яка досягається за рахунок натуральної параметризації з кубічним законом розподілу кривини.

Наведемо результати роботи методу апроксимації на прикладі точково-заданого контуру, який включає 14 таких точок:

```
 -216.680000  78.840891
-199.388394  82.662538
-182.333640  92.146215
-170.482179  102.580692
-147.849809  117.679087
-120.476549  134.114910
-114.176480  140.646664
-93.151429   150.726455
-81.003369   156.694929
-64.809795   162.158593
-48.383973   166.193093
-31.324910   168.943913
-14.712846   170.439555
  0.000000   170.830030
```

враховуючи, що похідні в кінцевих точках є нульовими $\varphi_1 = \varphi_2 = 0$.

За методом найменших модулів оптимальними значеннями кривин в крайніх точках фрагменту контуру, які знайдено з точністю до кроку розбиття інтервалів кривин, є кривина $k_1 = 2.42103$ в крайній лівій точці фрагменту та кривина $k_2 = -0.52632$ в крайній правій точці фрагменту. Їм відповідає апроксимуюча плоска крива, яка відображена на рисунку 4.6.

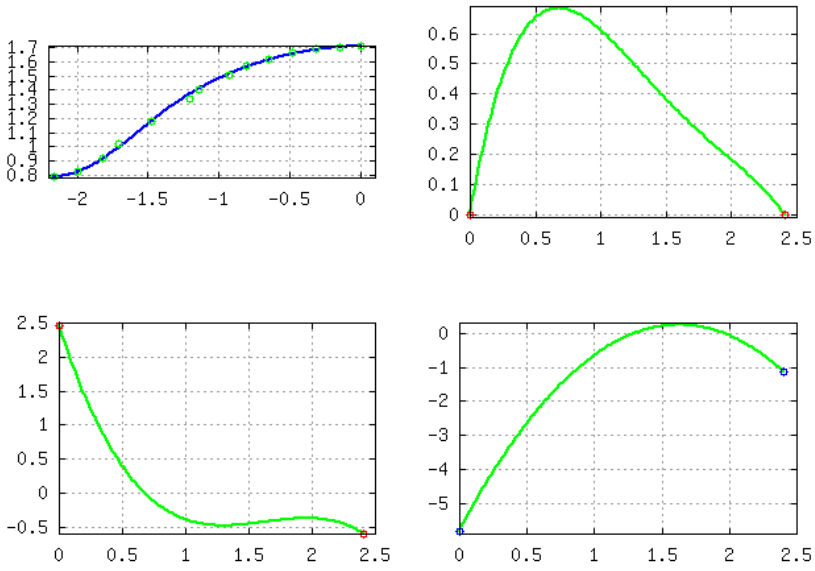


Рисунок 4.6 – Апроксимуюча крива з кубічним законом розподілу кривини

З графіку видно, що усі 14 точок контуру достатньо точно апроксимуються отриманою S-подібною кривою. Тому, близькі значення кривин отримаємо, якщо будемо ці точки апроксимувати плоскою кривою за методом найменших квадратів.

Для візуалізації результатів згладжування використовується графічне зображення поверхонь рівня функції, яка задає суму квадратів відстаней від кривої до заданого набору точок контуру (для методу найменших квадратів), або суму відстаней від кривої до заданого набору точок контуру (для методу найменших модулів). На рисунку 4.7 наведено поверхні рівня функції (сума відстаней від кривої до заданого набору точок) з відміченим оптимальним його значенням, яке досягається на кривинах $k_1 = 2.42103$ та $k_2 = -0.52632$.

Візуалізація значення мінімізуємої функції за допомогою рисунку 4.7 дозволяє зорієнтуватися, яким чином вибрати границі на кривини справа та кривини зліва, щоб побудувати S-подібну апроксимаційну криву. Тому алгоритм дозволяє або за методом найменших модулів або за методом найменших квадратів точково-заданий контур аппроксимувати S-подібною кривою з кубічним законом розподілу кривини в натуральній параметризації.

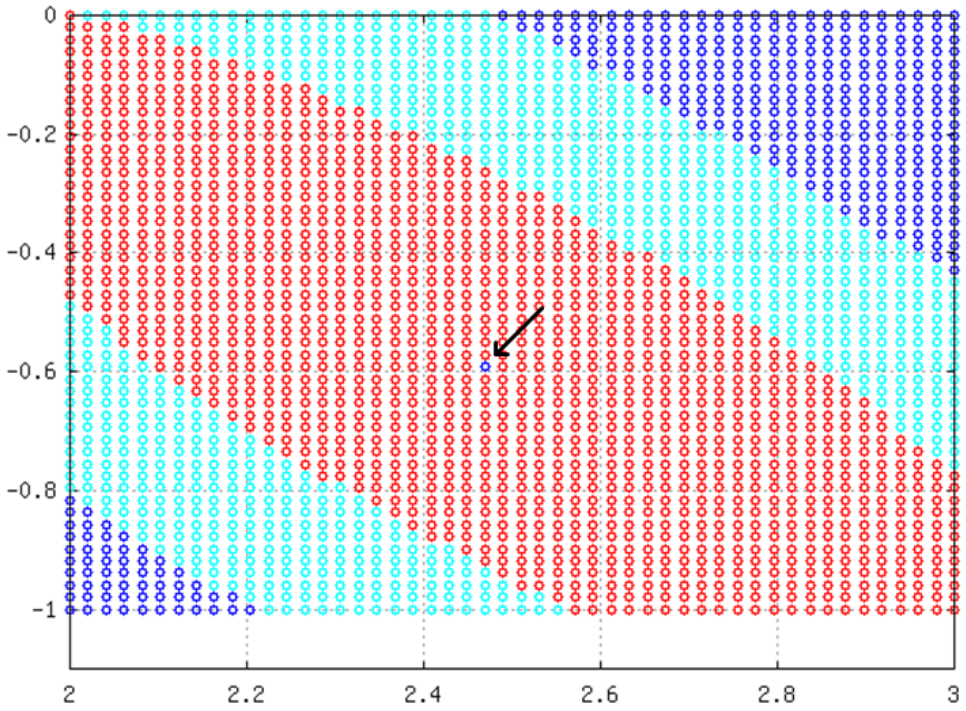


Рисунок 4.7 – Поверхня рівня функції суми відстаней від кривої до заданого набору точок контуру (оптимальне значення позначене синьою точкою)

Розроблений у цьому розділі метод, який включає математичну модель, алгоритм та програмне забезпечення, може бути використано для профілювання перехідних каналів змінного перерізу з необхідними геометричними властивостями (наряду з методом Безьє – Бернштейна). Цей метод можна використовувати для профілювання лопаток компресора шляхом задання скелетної лінії профілю за допомогою S-подібної кривої. За його допомогою можна проектувати S-подібні фрагменти зовнішнього контуру сопла та контуру центрального тіла. Вибір значень кривин k_1 та k_2 надає змогу керувати формою кривої у такий спосіб, щоб у базисних (реперних) точках характеристики кривої відповідали заданим характеристикам профілю, що проектується.

4.5 Масштабування даних

Розглянемо таку задачу: необхідно так з’єднати точки $A(x_A, y_A)$ та $B(x_B, y_B)$

кривою лінією у натуральній параметризації, де кривина $k(s) = as^3 + bs^2 + cs + d$ має кубічну залежність від довжини дуги, щоб забезпечити в точках A та B задані значення кутів нахилу дотичних φ_A та φ_B, та задані значення кривин k_A, k_B. Кути вимірюються в радіанах.

Нехай S – довжина кривої від точки A до точки B. Знаходженню параметрів кривини a, b, c, d та довжини S відповідає система з чотирьох нелінійних рівнянь та одного лінійного рівняння (Сергиенко, Семенов, и Собачкин 2004):

$$x_B = x_A + \int_0^S \cos\left(\varphi_A + \frac{as^4}{4} + \frac{bs^3}{3} + \frac{cs^2}{2} + d \times s\right) ds, \tag{4.20}$$

$$y_B = y_A + \int_0^S \sin\left(\varphi_A + \frac{as^4}{4} + \frac{bs^3}{3} + \frac{cs^2}{2} + d \times s\right) ds, \tag{4.21}$$

$$\varphi_B = \varphi_A + \frac{aS^4}{4} + \frac{bS^3}{3} + \frac{cS^2}{2} + d \times S, \tag{4.22}$$

$$k_A = d, \quad k_B = aS^3 + bS^2 + cS + d. \tag{4.23}$$

З лінійного рівняння в (4.23) отримуємо $d^\star = k_A$. Для пошуку $a^\star$, $b^\star$, $c^\star$, $S^\star$ в (Сергієнко та ін. 2016) використано умовну задачу мінімізації суми модулів функцій нев'язок для рівнянь (4.20) – (4.23), яка має вигляд: знайти

$$\begin{aligned}\left(a^\star, b^\star, c^\star, S^\star\right) = \arg\min_{a,b,c,S} \left\{ \left| x_B - x_A - \int_0^S \cos\left(\varphi_A + \frac{as^4}{4} + \frac{bs^3}{3} + \frac{cs^2}{2} + k_A s\right) ds \right| + \right. \\ + \left| y_B - y_A - \int_0^S \sin\left(\varphi_A + \frac{as^4}{4} + \frac{bs^3}{3} + \frac{cs^2}{2} + k_A s\right) ds \right| + \\ \left. + \left| \varphi_B - \varphi_A - \frac{aS^4}{4} - \frac{bS^3}{3} - \frac{cS^2}{2} - k_A S \right| + \left| k_B - aS^3 - bS^2 - cS - k_A \right| \right\}\end{aligned} \tag{4.24}$$

за обмежень

$$S_{\min} \le S \le S_{\max}, \tag{4.25}$$

$$-\frac{\pi}{2} \le \varphi_A + a\frac{i^4 S^4}{4N^4} + b\frac{i^3 S^3}{3N^3} + c\frac{i^2 S^2}{2N^2} + k_A \frac{iS}{N} \le \frac{\pi}{2}, \quad i = 1, \ldots, N, \tag{4.26}$$

де N – кількість рівних підінтервалів у інтервалі $[0, S]$ для дискретизації функції

$\varphi(s) = \varphi_A + \dfrac{as^4}{4} + \dfrac{bs^3}{3} + \dfrac{cs^2}{2} + k_A s,\ s \in [0, S];\ S_{\min} = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$, $S_{\max}$ – параметр для управління верхнею межею на S – загальну довжину кривої.

Цільова функція (4.24) є кусково-гладкою та забезпечує мінімізацію суми модулів функцій нев'язок для рівнянь (4.20) – (4.23). Обмеження (4.25) гарантують додатні значення довжини S, а обмеження (4.26) забезпечує існування єдиного глобального мінімуму для задачі (4.24) – (4.26) незалежно від вибору величини $S_{\max}$.

Для розв'язання задачі (4.24) – (4.26) на основі модифікації *r*-алгоритму Шора розроблено метод з управлінням параметрами α (коефіцієнт розтягу простору), h_0 (крок на першій ітерації) і q_1 (коефіцієнт зменшення кроку на подальших ітераціях) та критеріями зупинки ε_x (зупинка за аргументом) та *maxitn* (максимальна кількість ітерацій). Стартову точку вибрано рівною $(0, 0, 0, S_{\min})^T$.

Нехай довжина кривої $S^\star = 100$. Щоб у точці (x_B, y_B) гарантувати кут φ_B такий, що $-\dfrac{\pi}{2} \le \varphi_B \le \dfrac{\pi}{2}$, потрібно, щоб виконувалася нерівність

$$-\frac{\pi}{2} \le \frac{100^4}{4} a^\star + \frac{100^3}{3} b^\star + \frac{100^2}{2} c^\star + 100 d^\star \le \frac{\pi}{2}.$$

Якщо $a^\star$, $b^\star$, $c^\star$, $d^\star$ є додатними, то дана нерівність означає, що їх значення будуть величинами різних порядків: так $a^\star$ буде мати порядок 10^{-8}, $b^\star$ – порядок 10^{-6}, $c^\star$ – порядок 10^{-4}, а $d^\star$ – порядок 10^{-2}. Це впливає на швидкість збіжності методу розв'язання задачі (4.24) – (4.26), а при великих значеннях $S^\star$ навіть на неможливість її розв'язання. У цьому випадку може допомогти лема.

Лема ($\mu > 0$). Якщо $a^\star$, $b^\star$, $c^\star$, $d^\star$, $S^\star$ є розв'язком системи (4.20) – (4.23), то $a^{\star\star} = a^\star / \mu^4$, $b^{\star\star} = b^\star / \mu^3$, $c^{\star\star} = c^\star / \mu^2$, $S^{\star\star} = \mu S^\star$ буде розв'язком такої системи рівнянь:

$$\mu x_B = \mu x_A + \int_0^S \cos\left(\varphi_A + \frac{as^4}{4} + \frac{bs^3}{3} + \frac{cs^2}{2} + d \times s\right) ds, \quad (4.27)$$

$$\mu y_B = \mu y_A + \int_0^S \sin\left(\varphi_A + \frac{as^4}{4} + \frac{bs^3}{3} + \frac{cs^2}{2} + d \times s\right) ds, \quad (4.28)$$

$$\varphi_B = \varphi_A + \frac{aS^4}{4} + \frac{bS^3}{3} + \frac{cS^2}{2} + d \times S, \quad (4.29)$$

$$\mu k_A = d, \quad \mu k_B = aS^3 + bS^2 + cS + d. \quad (4.30)$$

Лема встановлює зв'язок розв'язків системи (4.20) – (4.23) та масштабованої системи (4.27) – (4.30), в якій координати точок та значення кривин домножуються на одну і ту ж величину $\mu > 0$.

Для знаходження розв'язків погано масштабованих ($\mu = 10$) та добре масштабованих ($\mu = 1$) задач проведено обчислювальний експеримент для двох наборів даних. Для погано масштабованих задач були вибрані вихідні дані:

$$x_A = 0,\ y_A = 20,\ \varphi_A = \frac{\pi}{18},\ k_A = 0, x_B = 30,\ y_B = 40,\ \varphi_B = \frac{\pi}{36},\ k_B = 0. \quad (4.31)$$

$$x_A = 0,\ y_A = 20,\ \varphi_A = \frac{\pi}{18}, k_A = 0.1, x_B = 30,\ y_B = 40,\ \varphi_B = \frac{\pi}{36},\ k_B = -0.05. \quad (4.32)$$

Для добре масштабованих задач їм відповідають такі вихідні дані:

$$x_A = 0,\ y_A = 2,\ \varphi_A = \frac{\pi}{18},\ k_A = 0, x_B = 3,\ y_B = 4,\ \varphi_B = \frac{\pi}{36},\ k_B = 0. \quad (4.33)$$

$$x_A = 0,\ y_A = 2,\ \varphi_A = \frac{\pi}{18}, k_A = 0.01, x_B = 3,\ y_B = 4,\ \varphi_B = \frac{\pi}{36},\ k_B = -0.005. \quad (4.34)$$

У таблиці 4.1 наведено витрати *r*-алгоритму (за кількістю ітерацій – itn_1 та itn_2, за кількістю обчислень узагальненого градієнта цільової функції – nfg_1 та nfg_2) для знаходження плоских кривих для даних (4.31) та (4.32) та краще масштабованих даних (4.33) та (4.34) для трьох критеріїв зупинки за аргументом $\varepsilon_x = \left\{10^{-6}, 10^{-8}, 10^{-10}\right\}$. При цьому використовувались такі параметри $\alpha = 2.0$, $h_0 = 1.0$, $q_1 = 1.0$, $maxitn = 1500$. В таблиці 4.1 також наведені відносні відхилення між компонентами розв'язків, які визначаються за формулами

$$\Delta a^\star = \left|a_2^\star - a_1^\star / \mu^4\right| / \left|a_2^\star\right|,\ \Delta b^\star = \left|b_2^\star - b_1^\star / \mu^3\right| / \left|b_2^\star\right|,\ \Delta c^\star = \left|c_2^\star - c_1^\star / \mu^2\right| / \left|c_2^\star\right|.$$

Таблиця 4.1 – Витрати *r*-алгоритму на пошук розв'язків задач (4.31), (4.33) та (4.32), (4.34)

Задача ($\mu = 10$)/ Задача ($\mu = 1$)	Задачі (4.31), (4.33)			Задачі (4.32), (4.34)		
	$\varepsilon_x = 10^{-6}$	$\varepsilon_x = 10^{-8}$	$\varepsilon_x = 10^{-10}$	$\varepsilon_x = 10^{-6}$	$\varepsilon_x = 10^{-8}$	$\varepsilon_x = 10^{-10}$
itn_1 / itn_2	163/103	211/133	242/154	254/102	278/131	314/151
nfg_1 / nfg_2	469/132	626/173	696/201	995/146	1039/183	1107/211
$\Delta a^\star$	5.5e-07	3.2e-09	8.9e-12	2.0e-05	1.3e-06	2.0e-09
$\Delta b^\star$	6.5e-07	2.5e-09	3.7e-12	2.8e-06	1.6e-07	2.6e-10
$\Delta c^\star$	6.6e-07	1.8e-09	1.3e-11	4.8e-07	2.4e-08	4.2e-11
$\Delta S^\star$	1.4e-07	1.3e-10	6.0e-12	2.6e-08	7.5e-10	3.1e-12

З таблиці видно, що для розв’язання задач з добре масштабованими даними (задачі (4.33) та (4.34)) потрібно майже в два рази менше ітерацій та майже в чотири рази менше обчислень узагальненого градієнта, ніж для розв'язання задач з погано масштабованими даними (задачі (4.31) та (4.32)).

4.6 Висновки до четвертого розділу

В розділі досліджено геометричну постановку задачі побудови кривої в натуральній параметризації з кубічною кривиною та сформульовано відповідну систему нелінійних інтегральних рівнянь (СНІР). Вперше сформульовано оптимізаційну задачу для знаходження розв’язку СНІР та метод її розв’язання з використанням модифікації *r*-алгоритму Шора. В підрозділі 4.3 описано програмне забезпечення для тестового прикладу, наведено коди відповідних Octave-програм та протокол роботи тестової програми. В підрозділі 4.4 наведено приклади застосувань. В підрозділі 4.5 досліджено питання масштабування для

системи нелінійних рівнянь у задачі побудови кривої у натуральній параметризації з кубічним розподілом кривини, що проходить через дві задані точки та забезпечує в них задані кути нахилу дотичних та задані значення кривин. Показано, що для того, щоб знайти розв’язок із заданою точністю за менший час краще розв'язувати задачу з добре масштабованими даними. Правильне масштабування даних відіграє суттєву роль для вибору того чи іншого параметру при моделюванні фрагментів аеродинамічних та технічних профілів в ітераційному режимі, враховуючи, що масштаб достатньо встановити всього один раз, а задачу (4.24) – (4.26) потрібно вирішувати дуже багато разів.

Матеріал розділу базується на таких роботах: (Khomiak et al. 2023), (Stetsyuk, Tkachenko and Zhydkov 2020), (Стецюк, Хом’як та Жидков 2023а), (Стецюк, Хом’як та Жидков 2023в), (Zhydkov, Khomiak 2025).

РОЗДІЛ 5. ПОШУК ДЕФЕКТІВ У РЕГУЛЯРНИХ 3D-СТРУКТУРАХ

Розглядаються оптимізаційні задачі для знаходження найкращих по L_p-нормі параметрів регулярних 3D-структур і методи найменших модулів та найменших квадратів для їх розв'язання. Показано, що при відновленні параметрів 3D-структур з дефектами метод найменших модулів стійкіший, ніж метод найменших квадратів. Наведено результати обчислювальних експериментів для програмних реалізацій методів на основі *r*-алгоритму Шора.

Регулярні зображення з дефектами (рис. 5.1) характерні при неруйнуючому контролі якості (НКЯ) тонкостінних багатошарових композиційних матеріалів за допомогою методів лазерної інтерферометрії, таких, як метод голографічної інтерферометрії, метод спекл-інтерферометрії та метод ширографії (Lobanov et al. 2005).

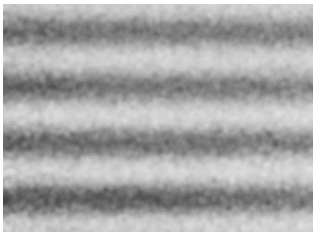
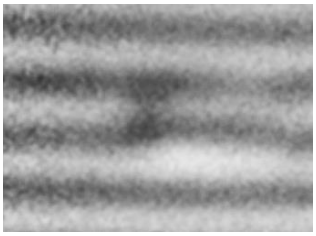
а)  б) 

Рисунок 5.1 – Приклади зображень: а) регулярне; б) регулярне з дефектною областю – порушення регулярності в центрі

Методи лазерної інтерферометрії для НКЯ передбачають вимірювання переміщень або деформацій точок поверхні досліджуваних об'єктів, які виникають у результаті термічного або механічного навантаження. При НКЯ елементів конструкцій, які мають періодичну (регулярну) 3D-структуру, поле вимірюваних методами лазерної інтерферометрії величин (переміщення, деформації) повинне також мати періодичну структуру.

Якщо при виготовленні або експлуатації таких елементів конструкцій виникають дефекти (наприклад, відсутність з'єднання, наявність тріщин, пор,

вм'ятин тощо), то в таких місцях порушується регулярність поля вимірюваних величин, і область дефекту визначається оператором (спостерігачем) як місце, у якому є очевидним це порушення.

Потрібно автоматизувати процес визначення місця розташування дефектів у відповідальних елементах конструкцій, щоб знизити вплив людського фактора при неруйнуючому контролі якості за допомогою зазначених вище методів.

У розділі розглянемо оптимізаційні задачі для регулярних 3D-структур, які введені авторами у роботі (Стецюк та Савицький 2017), (Стецюк и Савицкий 2018) і мають місце при аналізі регулярних зображень із дефектами, і методи їх розв'язання на основі методу Ньютона та алгоритмів негладкої оптимізації.

Матеріал розділу викладений у такому порядку. У підрозділі 5.1 визначено регулярні 3D-структури та пов'язані з нею дефекти. У підрозділі 5.2 сформульовано оптимізаційні задачі для знаходження найкращих за L_p-нормою параметрів регулярних 3D-структур і досліджено їхні властивості. У підрозділі 5.3 описано методи найменших квадратів і найменших модулів для знаходження параметрів регулярних структур.

Показано, що метод найменших квадратів не здатний відновити їхні параметри при наявності хоча б одного дефекту, а метод найменших модулів є стійким до знаходження параметрів регулярних 3D-структур з однією та декількома областями з дефектами. У підрозділі 5.4 описано дві загальні нерівності для суми квадратів двох наборів чисел, перша з яких неявно використовувалася у підрозділі 5.1 при визначенні базисної регулярної 3D-структури.

5.1 Регулярні 3D-структури, їхні параметри та дефекти

3D-структурою будемо називати трійку $\{A,u,v\}$, де A – $m\times n$-матриця, така що $A=\{a_{ij}\}_{i=1,\dots,m}^{j=1,\dots,n}\in R^{m\times n}$, $u\in R^m$ і $v\in R^n$. Параметрами 3D-структури $\{A,u,v\}$

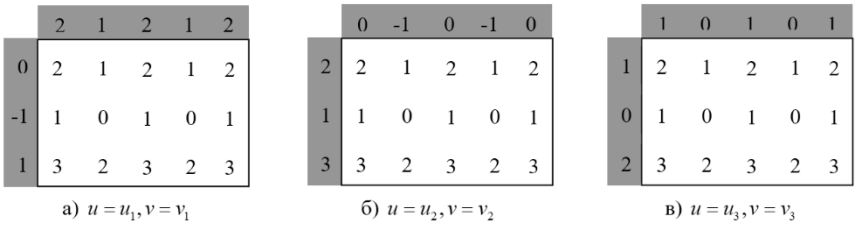
будемо називати m-вимірний вектор u і n-вимірний вектор v, їхні компоненти будуть визначати значення елементів матриці $A \in R^{m \times n}$.

Означення 5.1. 3D-структура $\{A,u,v\}$ називається регулярною, якщо матриця $A \in R^{m \times n}$, а вектори $u \in R^m$ і $v \in R^n$ є такими, що $a_{ij} = u_i + v_j, i = 1,\ldots,m$, $j = 1,\ldots,n$.

Одна із властивостей регулярної 3D-структури визначається наступним твердженням.

Лема 5.1. *Якщо* $\{A,u,v\}$ – регулярна 3D-структура, то регулярною буде й 3D-структура $\{A,\tilde{u},\tilde{v}\}$, де $\tilde{u}_i = u_i + t$, $i = 1,\ldots,m$, $\tilde{v}_j = v_j - t$, $j = 1,\ldots,n$, $\forall t \in R$, $t \neq 0$.

Лема 5.1 означає, що для регулярної 3D-структури параметри u та v визначені неоднозначно. Так, наприклад, на рисунку 5.2 наведено три різні набори параметрів для однієї й тієї ж 3×5-матриці $A = \begin{vmatrix} 2 & 1 & 2 & 1 & 2 \\ 1 & 0 & 1 & 0 & 1 \\ 3 & 2 & 3 & 2 & 3 \end{vmatrix}$, яка визначає регулярну 3D-структуру. Першій регулярній 3D-структурі $\{A,u,v\}$ відповідають параметри $u = u_1, v = v_1$, другій – $u = u_2, v = v_2$, третій – $u = u_3, v = v_3$.



а) $u = u_1, v = v_1$ б) $u = u_2, v = v_2$ в) $u = u_3, v = v_3$

Рисунок 5.2 – Неоднозначні регулярні 3D-структури:

а) $u_1 = (0,-1,1)^T, v_1 = (2,1,2,1,2)^T$; б) $u_2 = (2,1,3)^T, v_2 = (0,-1,0,-1,0)^T$;

в) $u_3 = (1,0,2)^T, v_3 = (1,0,1,0,1)^T$

Щоб параметри регулярної 3D-структури були визначені однозначно, досить зафіксувати величину t з тих або інших міркувань. Наприклад, її значення можна

вибрати таким, щоб вираз $\|u+te_m\|^2+\|v-te_n\|^2$ був мінімальним. Тут e_m і e_n – m -вимірний і n-вимірний вектори, усі компоненти яких дорівнюють одиниці. Це призводить до мінімізації одномірної функції

$$f(t)=\|u+te_m\|^2+\|v-te_n\|^2=$$

$$=\sum_{i=1}^{m}u_i^2+\sum_{j=1}^{n}v_j^2+2\left(\sum_{i=1}^{m}u_i-\sum_{j=1}^{n}v_j\right)t+(m+n)t^2. \qquad (5.1)$$

Вона досягає мінімального значення f^* при $t=t^*$, які визначаються за формулами

$$f^*=f(t^*)=\sum_{i=1}^{m}u_i^2+\sum_{j=1}^{n}v_j^2-\frac{\left(\sum_{i=1}^{m}u_i-\sum_{j=1}^{n}v_j\right)^2}{m+n},\quad t^*=\frac{\sum_{j=1}^{n}v_j-\sum_{i=1}^{m}u_i}{m+n}. \qquad (5.2)$$

За допомогою формул (5.1) і (5.2) можна однозначно визначити параметри регулярної 3D-структури. З виразу для f^* у (5.2) випливає, що зменшити суму квадратів компонент векторів u і v не можна, якщо для них виконана умова $\sum_{i=1}^{m}u_i=\sum_{j=1}^{n}v_j$. Саме цю умову поставимо в основу однозначної (по параметрах) регулярної 3D-структури.

Означення 5.2. Регулярну 3D-структуру $\{A,u^*,v^*\}$ будемо називати базисною, якщо її параметри $u^*\in R^m$ та $v^*\in R^n$ – такі, що $\sum_{j=1}^{m}u_j^*=\sum_{i=1}^{n}v_i^*$.

Справедливе таке твердження.

Лема 5.2. *Я*кщо $\{A,u,v\}$ – регулярна 3D-структура, то регулярна 3D-структура $\{A,u^*,v^*\}$ буде базисною, якщо

$$u^*=u+t^*e_m;\quad v^*=v-t^*e_n,\quad \text{де}\quad t^*=\frac{\sum_{j=1}^{n}v_j-\sum_{i=1}^{m}u_i}{m+n}. \qquad (5.3)$$

Якщо $\{A,u^*,v^*\}$ – базисна регулярна 3D-структура, то 3D-структура $\{A,u,v\}$ буде регулярною, якщо

$$u=u^*+te_m,\quad v=v^*-te_n \tag{5.4}$$

для довільного $t\in R$.

Лема 5.2 дає нам формули (5.3) і (5.4), які зв'язують між собою параметри базисної й звичайної регулярних 3D-структур. Наприклад, базисною регулярною 3D-структурою для 3×5-матриці $A=\begin{vmatrix}2&1&2&1&2\\1&0&1&0&1\\3&2&3&2&3\end{vmatrix}$ є третя регулярна 3D-структура з рисунку 5.2, параметри якої $u=u_3=(1,0,2)^T$ та $v=v_3=(1,0,1,0,1)^T$. Для неї $\sum_{i=1}^{3}u_i=\sum_{j=1}^{5}v_j=3$, отже $u^*=(1,0,2)^T$ і $v^*=(1,0,1,0,1)^T$. Щоб від першої та другої регулярних 3D-структур з рисунку 5.2 перейти до третьої, досить у формулах (5.3) використовувати $u=u_1$, $v=v_1$, $t^*=t_1^*=-1$ або $u=u_2$, $v=v_2$, $t^*=t_2^*=1$. За допомогою формули (5.4) і параметрів базисної регулярної 3D-структури для матриці A розмірності 3×5 можна побудувати не тільки першу, а й другу регулярні 3D-структури (їм у формулі (5.4) відповідають значення $t=t_1=1$ та $t=t_2=-1$), а також низку інших 3D-структур, що задовольняють ті або інші властивості.

Наприклад, при $t=t_4=0.5$ одержуємо регулярну 3D-структуру з параметрами $u=u_4=(0.5,-0.5,1.5)^T$ і $v=v_4=(1.5,0,1.5,0,1.5)^T$, тобто таку, що останні компоненти векторів u і v однакові.

Означення 5.3. Елементарним дефектом у регулярній 3D-структурі $\{A,u,v\}$ будемо називати таку пару індексів i,j, $i\in1,\ldots,m$, $j\in1,\ldots,n$, для яких $a_{ij}\neq u_i+v_j$.

Для регулярних 3D-структур з дефектами будемо дотримуватися таких назв. Якщо регулярна 3D-структура має один елементарний дефект, то її будемо

називати регулярною з одним дефектом. Якщо регулярна 3D-структура має k елементарних дефектів ($k>1$), то її будемо називати регулярною з k дефектами. Приклади регулярних 3D-структур з одним і трьома дефектами 3×5-матриці $\tilde{A}$ наведено на рисунку 5.3, де в «рамки» узяті ті елементи матриці $\tilde{A}$ з індексами i, j, де має місце елементарний дефект $\tilde{a}_{ij} \neq u_i + v_j$. Це зроблене на прикладі тієї ж 3×5-матриці A, що й на рисунку 5.1.

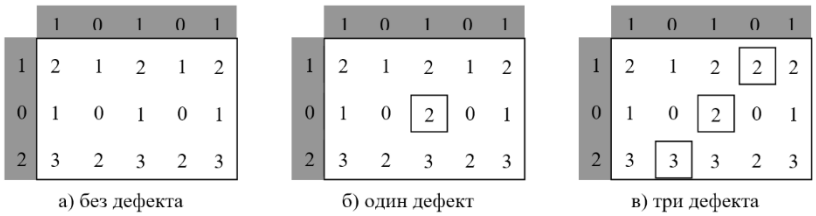


Рисунок 5.3 – Регулярні 3D-структури: а) базисна; б) з одним дефектом $\tilde{a}_{23} = a_{23} + 1$; в) з трьома дефектами $\tilde{a}_{14} = a_{14} + 1,\ \tilde{a}_{23} = a_{23} + 1,\ \tilde{a}_{32} = a_{32} + 1$.

Для нас буде достатньо означення 5.3, але слід зазначити, що для реальних задач поняття «дефекту» може потребувати більш складного визначення. У них для регулярних зображень під «дефектними» розуміються деякі множини пар індексів i, j, у яких порушена умова $a_{ij} \neq u_i + v_j$.

Як правило, ці множини задають зв'язні області, але в принципі вони можуть задавати й незв'язні області.

5.2 Задачі для пошуку найкращих параметрів

Нижче розглянемо формулювання оптимізаційних задач для знаходження найкращих параметрів регулярної 3D-структури й базисної регулярної 3D-структури, де під «найкращими параметрами» будемо розуміти такі значення векторів x^* і y^*, що коефіцієнти $a^*_{ij} = x^*_i + y^*_j$ мінімально (зокрема, в евклідовій і манхетенівській нормах) відхиляються від коефіцієнтів деякої заданої матриці A.

Задача А. Є $m \times n$-матриця A. Для регулярної 3D-структури $\{A, x^*, y^*\}$ потрібно знайти такі вектори $x^* \in R^m$ та $y^* \in R^n$, щоб коефіцієнти $m \times n$-матриці A^* мінімально відхилялися від коефіцієнтів $m \times n$-матриці A.

Пошук найкращих параметрів для регулярної 3D-структури можна забезпечити за допомогою безумовної задачі мінімізації негладкої опуклої функції: знайти

$$f_p^* = f_p(x_p^*, y_p^*) =$$

$$= \min_{x \in R^m,\ y \in R^n} \left\{ f_p(x,y) = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} \left| a_{ij} - x_i - y_j \right|^p \right)^{1/p} \right\}, \qquad (5.5)$$

де $|\cdot|$ – модуль числа, а скалярна величина p така, що $1 \le p \le 2$. Розв'язок задачі (5.5) дає регулярну 3D-структуру $\{A^*, x_p^*, y_p^*\}$, а коефіцієнти матриць A і A^* будуть мінімально різнитися в\по так званій L_p-нормі, тобто, коли норма вектора визначена в такий спосіб: $\|z\|_p = \left(\sum_{i=1}^{N} |z_i|^p \right)^{1/p}$, де $p \ge 1$. Якщо $p = 2$, то коефіцієнти матриць A і A^* будуть мінімально відхилятися в евклідовій нормі, якщо $p = 1$ – то в манхетенівській нормі.

Крім того, можна розглядати й інші значення величини p: $1 < p < 2$. Її вибором визначається той або інший метод для відновлення параметрів регулярних 3D-структур, наприклад, метод найменших квадратів ($p = 2$) або метод найменших модулів ($p = 1$).

Задача (5.5) має багато розв'язків, оскільки значення параметрів регулярної 3D-структури $\{A^*, x_p^*, y_p^*\}$ визначені неоднозначно (див. лему 5.1). Однозначно вони будуть визначені тоді, коли регулярна 3D-структура $\{A^*, x_p^*, y_p^*\}$ буде ще й базисною.

Параметри базисної регулярної 3D-структури можна визначити в такий спосіб: спочатку розв'язати задачу (5.5), а потім для знаходження u^* та v^*

використати формули (5.3), де $u = x_p^*$ і $v = y_p^*$. Але це ж саме можна зробити за допомогою свого формулювання оптимізаційної задачі, яка буде розрахована на пошук найкращих параметрів базисної регулярної 3D-структури.

Задача Б. Є $m \times n$-матриця A. Для базисної регулярної 3D-структури $\{A^{**}, x^{**}, y^{**}\}$ потрібно знайти такі вектори $x^{**} \in R^m$ та $y^{**} \in R^n$, щоб коефіцієнти $m \times n$-матриці A^{**} мінімально відхилялися від коефіцієнтів $m \times n$-матриці A.

Пошук найкращих параметрів для базисної регулярної 3D-структури $\{A^{**}, x^{**}, y^{**}\}$ можна забезпечити, якщо задачу (5.5) доповнити єдиним обмеженням

$$\sum_{i=1}^{m} x_i - \sum_{j=1}^{n} y_j = 0, \tag{5.6}$$

яке задає умови на параметри базисної 3D-структури з леми 5.2. Задача (5.5) – (5.6) є задачею опуклого нелінійного програмування. Їй буде відповідати таке ж, як і для задачі (5.5), оптимальне значення f_p^*, а її оптимальний розв'язок x_p^{**} і y_p^{**} буде визначати найкращі по L_p-нормі параметри базисної регулярної структури $\{A^{**}, x^{**}, y^{**}\}$. Якщо $p = 2$, то для знаходження параметрів базисної регулярної 3D-структури одержимо метод найменших квадратів, а якщо $p = 1$ – метод найменших модулів.

Оптимізаційні задачі (5.5) та (5.5) – (5.6) можна розв'язувати за допомогою стандартного програмного забезпечення для розв'язання задач нелінійного програмування. Однак тут є деякі особливості, які пов'язані з представленням задач (5.5) та (5.5) – (5.6) у формі задач опуклого програмування.

Наприклад, задачу (5.5) можна звести до такої задачі опуклого програмування: знайти

$$\left(f_p^*\right)^p = \left(f_p(z_p^*, x_p^*, y_p^*)\right)^p =$$

$$= \min_{z\in R^{m\times n}, x\in R^m, y\in R^n} \left\{ (f_p(z,x,y))^p = \sum_{i=1}^{m}\sum_{j=1}^{n} z_{ij}^p \right\}, \tag{5.7}$$

за обмежень

$$-z_{ij} - x_i - y_j \le -a_{ij}, \quad i=1,\ldots,m,\ j=1,\ldots,n, \tag{5.8}$$

$$-z_{ij} + x_i + y_j \le a_{ij}, \quad i=1,\ldots,m,\ j=1,\ldots,n, \tag{5.9}$$

яка при $p=1$ перетворюється у задачу лінійного програмування, а при $p=2$ – у задачу квадратичного програмування. Щоб задачу (5.5) – (5.6) звести до задачі опуклого програмування, до задачі (5.7) – (5.9) достатньо додати лінійне обмеження (5.6). Особливістю задач (5.7) – (5.9) та (5.7) – (5.9), (5.6) є те, що цільова функція (5.7) визначена тільки для невід'ємних значень змінних $z_{ij} \ge 0$, які задовольняють обмеження (5.8), (5.9).

Дійсно, якщо для деякої пари індексів i та j скласти нерівності (5.8) і (5.9), то одержимо нерівність $-2z_{ij} \le 0$, яка рівносильна $z_{ij} \ge 0$. Тому при таких значеннях p, коли $1 < p < 2$, для розв'язання задач (5.7) – (5.9) і (5.7) – (5.9), (5.6) застосовні тільки ті ітераційні методи, які працюють із припустимими точками для системи обмежень (5.8) – (5.9). Крім того, навіть при порівняно невеликих значеннях m і n задача (5.7) – (5.9) буде мати велику кількість змінних $N = m\times n + m + n$ і обмежень $M = 2\times m\times n$.

Перспективнішим є розв'язання задачі (5.5) за допомогою субградієнтних методів для мінімізації негладких опуклих функцій (Шор 1979), (Стецюк 2014). При цьому кількість змінних $N = m + n$, а субградієнт опуклої функції $f_p(x,y)$ обчислюється за формулою

$$g_{f_p}(x,y) =$$

$$= \left(f_p(x,y)\right)^{1-p} \begin{cases} -\sum_{j=1}^{n} \operatorname{sgn}\left(a_{ij} - x_i - y_j\right)\left|a_{ij} - x_i - y_j\right|^{p-1}, i=\overline{1,m}, \\ -\sum_{i=1}^{m} \operatorname{sgn}\left(a_{ij} - x_i - y_j\right)\left|a_{ij} - x_i - y_j\right|^{p-1}, j=\overline{1,n}, \end{cases} \tag{5.10}$$

яку легко реалізувати для матлабо-подібних мов програмування. Цей шлях також підходить і для задачі (5.5) – (5.6), бо вона зводиться до такої задачі мінімізації негладкої опуклої функції: знайти

$$F_p^* = F_p\left(x_p^{**}, y_p^{**}\right) = \min_{x \in R^m,\ y \in R^n} \left\{ F_p(x,y) = \right.$$

$$\left. = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} \left| a_{ij} - x_i - y_j \right|^p \right)^{\frac{1}{p}} + M \left| \sum_{i=1}^{m} x_i - \sum_{j=1}^{n} y_j \right|^q \right\}, \qquad (5.11)$$

де скалярні величини p і q такі, що $1 \le p, q \le 2$, а скалярна величина $M > 0$. Слід зазначити, що для розв'язання задачі (5.5) – (5.6) можна використовувати також субградієнтні методи із проекцією на лінійну рівність (5.6). Так, наприклад, при застосуванні *r*-алгоритмів це призводить до певного способу установки початкової матриці B_0 і до наступної її корекції таким чином, щоб для кожної точки ітераційного процесу виконувалася рівність (5.6).

Отже, за допомогою розв'язання задачі (5.5) можна знаходити параметри регулярних 3D-структур, а за допомогою розв'язання задачі (5.5) – (5.6) можна відновлювати параметри базисних регулярних 3D-структур. Справедливе таке твердження.

Теорема 5.1. Якщо 3D-структура $\{A, u, v\}$ – регулярна, то в задачі (5.5) розв'язок x_p^* і y_p^* такий, що $f_p^* = f(x_p^*, y_p^*) = 0$, а 3D-структура $\{A, x_p^*, y_p^*\}$ є регулярною. Якщо регулярна 3D-структура $\{A, u^*, v^*\}$ є базисною, то в задачі (5.5) – (5.6) розв'язок x_p^{**} і y_p^{**} такий, що $f_p^* = f(x_p^{**}, y_p^{**}) = 0$, а $u^* = x_p^{**}$ і $v^* = y_p^{**}$

.

Якщо до регулярної 3D-структури додати один або кілька елементарних дефектів, то $f_p^* \neq 0$ як для задачі (5.5), так і для задачі (5.5) – (5.6). При цьому вибір параметра p суттєво впливає та те, зможемо чи не зможемо ми знайти параметри регулярних 3D-структур. Це ілюструють обчислювальні експерименти по відновленню параметрів $u^* = (1,0,2)^T$ і $v^* = (1,0,1,0,1)^T$ для

базисної регулярної 3D-структури $\left\{A^*, u^*, v^*\right\}$ за 3×5-матрицями $A_1 = \begin{vmatrix} 2 & 1 & 2 & 1 & 2 \\ 1 & 0 & 1+1 & 0 & 1 \\ 3 & 2 & 3 & 2 & 3 \end{vmatrix}$ і $A_2 = \begin{vmatrix} 2 & 1 & 2 & 1+1 & 2 \\ 1 & 0 & 1+1 & 0 & 1 \\ 3 & 2+1 & 3 & 2 & 3 \end{vmatrix}$, які містять один і три елементарні дефекти відповідно (див. рис. 5.3). Їхні результати для десяти значень $p \in [1,2]$ відображено в таблиці 5.1, де наведено значення p (перший стовпчик), оптимальні значення цільових функцій (стовпці $f_p\left(x_p^{**}, y_p^{**}\right)$) і відхилення знайдених параметрів x_p^{**} і y_p^{**} від точних $u^* = (1,0,2)^T$ і $v^* = (1,0,1,0,1)^T$ (стовпчики $\left\|u^* - x_p^{**}\right\|$ і $\left\|v^* - y_p^{**}\right\|$). Тут за допомогою *r*-алгоритму (програма ralgb5 (Стецюк 2017a)) розв'язувалася оптимізаційна задача (5.11) при $M = 1$ і $q = 1$.

Таблиця 5.1 – Відновлення параметрів 3D-структур з дефектами при різних p

p	Один дефект ($A = A_1$)			Три дефекти ($A = A_2$)		
	$f_p(x_p^{**}, y_p^{**})$	$\left\|u^* - x_p^{**}\right\|$	$\left\|v^* - y_p^{**}\right\|$	$f_p(x_p^{**}, y_p^{**})$	$\left\|v^* - y_p^{**}\right\|$	$\left\|v^* - y_p^{**}\right\|$
1.00	1.00000	0.00000	0.00000	3.00000	0.00000	0.00000
1.05	1.00000	0.00000	0.00000	2.84709	0.00000	0.00000
1.06	1.00000	0.00000	0.00001	2.81912	0.00001	0.00001
1.08	0.99999	0.00004	0.00016	2.76550	0.00011	0.00021
1.10	0.99991	0.00021	0.00089	2.71461	0.00063	0.00118
1.20	0.99472	0.00693	0.02762	2.48527	0.01968	0.03653
1.40	0.94620	0.04579	0.13764	2.09220	0.09757	0.18108
1.60	0.87096	0.09568	0.21940	1.79311	0.15558	0.28875
1.80	0.79586	0.14130	0.27027	1.57515	0.19226	0.35681
2.00	0.73030	0.17854	0.30334	1.41421	0.21651	0.40182

З таблиці 5.1 видно, що при $p=1$ (відповідає методу найменших модулів) параметри $u^*=(1,0,2)^T$ та $v^*=(1,0,1,0,1)^T$ відновлюються точно і при одному, і при трьох елементарних дефектах. Про це свідчать оптимальні значення функцій, які для одного та трьох дефектів повинні дорівнювати $\left\|A_1-A^*\right\|=1$ та $\left\|A_2-A^*\right\|=3$ відповідно.

Схожа поведінка спостерігається також для значень p, які близькі до одиниці. Але зовсім інша картина має місце при $p=2$ (відповідає методу найменших квадратів) і значеннях p, які близькі до двійки. Тут говорити про досить точне відновлення параметрів не доводиться, тому що відхилення знайдених параметрів x_2^{**} і y_2^{**} від точних значень параметрів u^* і v^* вже досить великі як при наявності одного, так і при наявності трьох елементарних дефектів.

5.3 Методи найменших квадратів та найменших модулів

Метод найменших квадратів (МНК) і метод найменших модулів (МНМ) є важливими окремими випадками методів розв'язування оптимізаційних задач (5.5) і (5.5) – (5.6). Так, МНК отримуємо, якщо вибрати $p=2$, а МНМ – якщо вибрати $p=1$. Якщо методи пов'язані з розв'язуванням задачі (5.5), то вони дозволяють знаходити найкращі параметри регулярних 3D-структур, а якщо з розв'язуванням задачі (5.5) – (5.6), то найкращі параметри базисних регулярних 3D-структур. Для обох методів виконується теорема 5.1, а значить МНМ і МНК дозволяють відновлювати параметри базисних регулярних 3D-структур.

Це можна зробити двома способами: перший – знайти один з розв'язків задачі (5.5) і застосувати формулу (5.3) з леми 5.2, другий – знайти єдиний розв'язок задачі (5.5) – (5.6).

Для регулярних 3D-структур з одним або декількома дефектами поведінка МНК і МНМ суттєво відрізняється. Зокрема, якщо в МНК немає шансів відновити параметри регулярних 3D-структур з дефектами, то в МНМ такі шанси

дуже великі, що підтверджують результати обчислювальних експериментів з таблиці 5.1.

Відповідні до методів МНК і МНМ оптимізаційні задачі можна спростити. Наприклад, для МНМ задача опуклого програмування (5.7) – (5.9) переходить у таку задачу лінійного програмування:

$$f_1^* = f_1\left(z^*, x^*, y^*\right) = \min_{z \in R^{m \times n}, x \in R^m, y \in R^n} \left\{ f_1(z, x, y) = \sum_{i=1}^{m} \sum_{j=1}^{n} z_{ij} \right\} \qquad (5.12)$$

за обмежень (5.8) – (5.9) (для регулярних структур), або (5.8) – (5.9), (5.6) для (базисних регулярних структур).

Для МНМ задача опуклого програмування (5.7) – (5.9) переходить у задачу квадратичного програмування

$$\left(f_2^*\right)^2 = \left(f_2\left(z^*, x^*, y^*\right)\right)^2 =$$

$$= \min_{z \in R^{m \times n},\ x \in R^m,\ y \in R^n} \left\{ \left(f_2\left(z^*, x^*, y^*\right)\right)^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} z_{ij}^2 \right\}, \qquad (5.13)$$

при обмеженнях (5.8) – (5.9) (для регулярних структур), або (5.8) – (5.9), (5.6) для (базисних регулярних структур).

Для задач лінійного і квадратичного програмування немає ніяких незручностей, пов’язаних з областю визначення цільової функції, які обговорювалися для задачі (5.7) – (5.9). Крім того, їх легко доповнювати лінійними обмеженнями, які можуть задавати додаткові умови на ті параметри регулярних 3D-структур, які потрібно знайти.

Незважаючи на те, що зазначені задачі можуть мати досить великі розміри, для їхнього розв'язування можна успішно застосовувати стандартне програмне забезпечення для задач лінійного і квадратичного програмування. Так, наприклад, задачі лінійного програмування із сотнями тисяч змінних розв’язуються всього за декілька хвилин на сучасних персональних комп’ютерах.

Отже, знаходження параметрів 3D-структур при $m \approx 500$ і $n \approx 500$ виглядає досить реалістичним за допомогою сучасних програм для розв'язування задач

лінійного програмування. Для задачі квадратичного програмування розміри m і n будуть трохи меншими.

Менш чутливими до розмірів 3D-структур будуть методи МНК і МНМ, які використовують оптимізаційні задачі, отримані безпосереднім спрощенням задач (5.5) та (5.5) – (5.6).

МНК для знаходження параметрів регулярної 3D-структури пов'язаний із розв'язуванням такої задачі: знайти

$$\left(f_2^*\right)^2 = \left(f_2\left(x^*, y^*\right)\right)^2 =$$

$$= \min_{x \in R^m,\ y \in R^n} \left\{ \left(f_2(x,y)\right)^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}\left(a_{ij} - x_i - y_j\right)^2 \right\}, \qquad (5.14)$$

а для знаходження параметрів базисної регулярної 3D-структури – з розв’язуванням задачі: знайти

$$\left(f_2^*\right)^2 = \left(f_2\left(x^*, y^*\right)\right)^2 =$$

$$= \min_{x \in R^n,\ y \in R^n} \left\{ \sum_{i=1}^{n}\sum_{j=1}^{m}\left(a_{ij} - x_i - y_j\right)^2 + \left(\sum_{i=1}^{n} x_i - \sum_{j=1}^{m} y_j\right)^2 \right\}. \qquad (5.15)$$

Задачі (5.14) і (5.15) є задачами мінімізації опуклої квадратичної функції із $N = m + n$ змінними, з тією відмінністю, що перша задача має багато розв’язків, а друга – єдиний розв’язок.

Задачі (5.14) і (5.15) відрізняються від тих негладких задач, які безпосередньо випливають із формулювань (5.5) і (5.5) – (5.6). У них з L_2-норми вилучена операція добування квадратного кореня, й зроблене це для того, щоб мати справу з мінімізацією квадратичних функцій.

Крім того, для задачі (5.15) штраф за порушення рівності (5.6) обраний рівним одиниці. Це пояснюється тим, що в цьому випадку знаходження мінімуму квадратичної функції пов’язане з розв’язуванням «блочно-діагональної» системи лінійних рівнянь

$$\begin{vmatrix} nI_m + e_m e_m^T & 0 \\ 0 & mI_n + e_n e_n^T \end{vmatrix} \begin{vmatrix} x^{**} \\ y^{**} \end{vmatrix} = \begin{vmatrix} \sum_{j=1}^{n} a_{ij},\ i = \overline{1,m} \\ \sum_{i=1}^{m} a_{ij},\ j = \overline{1,n} \end{vmatrix}, \tag{5.16}$$

де I_n – одинична $n \times n$-матриця, I_m – одинична $m \times m$-матриця.

З (5.16) одержуємо аналітичний розв'язок задачі (5.15), який має такий вигляд

$$\begin{aligned} x_i^{**} &= \frac{1}{n}\sum_{j=1}^{n} a_{ij} - \frac{1}{n(n+m)}\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij},\ i = \overline{1,m}, \\ y_j^{**} &= \frac{1}{m}\sum_{i=1}^{m} a_{ij} - \frac{1}{m(n+m)}\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij},\ j = \overline{1,n}. \end{aligned} \tag{5.17}$$

Формули (5.17) пояснюють, чому МНК не може відновити регулярну 3D-структуру з одним елементарним дефектом. Дійсно, якщо для якоїсь пари індексів i, j маємо $a_{ij} = u_i^* + v_j^* + \Delta$, де $\Delta \neq 0$, то з формул (5.17) випливає, що $x_i^{**} \neq u_i^*$ для всіх $i = \overline{1,n}$ і $y_j^{**} \neq v_j^*$ для всіх $j = \overline{1,m}$.

Крім того, для єдиного елементарного дефекту $a_{ij} = u_i^* + v_j^* + \Delta$ мають місце формули

$$\begin{aligned} \left\| x^{**} - u^* \right\| &= \sqrt{\frac{n+m-2}{n^2(n+m)} + \frac{m}{n^2(n+m)^2}}\,|\Delta| \neq 0, \\ \left\| y^{**} - v^* \right\| &= \sqrt{\frac{n+m-2}{m^2(n+m)} + \frac{n}{m^2(n+m)^2}}\,|\Delta| \neq 0. \end{aligned} \tag{5.18}$$

Якщо формули (5.18) застосувати для $\Delta = 1$, $m = 3$ і $n = 5$, то одержимо відхилення $\left\| u^* - x_p^{**} \right\| = 0.17854$ і $\left\| v^* - y_p^{**} \right\| = 0.30334$ з останнього рядка таблиці 5.1, які при $p = 2$ були отримані за допомогою *r*-алгоритму для регулярної 3D-структури з одним дефектом $a_{23} = a_{23}^* + 1$.

Задачі (5.14) і (5.15) можуть служити тестовими прикладами для перевірки збіжності алгоритмів мінімізації гладких опуклих функцій. Аналітичний розв'язок (5.18) у комбінації з формулами леми 5.2 дозволяє оцінити відхилення

знайденого розв'язку від точного. При цьому, якщо для задачі (5.15) застосовні алгоритми ньютонівського типу, то для задачі (5.14) вони незастосовні, тому що матриця квадратичної форми вироджена, її ранг рівний $m+n-1$ і на одиницю менший, ніж кількість змінних $N=m+n$.

Але зате задача (5.14) є зручною для градієнтних методів, зокрема для граничних варіантів *r*-алгоритмів, які сходяться не більше, ніж за N ітерацій (див. теорему 1 у роботі (Стецюк 2017б). Для неї, враховуючи що множина розв'язків є неоднозначною, легше знайти одну точку із множини оптимальних точок, ніж єдину оптимальну точку в задачі (5.15).

Тому алгоритми, що розв'язують обидві задачі із близькими витратами для однієї й тієї ж точності, будуть кращими, ніж алгоритми, які добре розв'язують одну задачу і погано – іншу. До того, задачу (5.15) можна ускладнити, помноживши останній член суми на деякий штраф $M>0,$ керування яким дозволяє змінювати ступінь витягнутості квадратичних функцій.

На відміну від МНК, оптимізаційні задачі для МНМ випливають із (5.5) і (5.5) – (5.6) без ускладнень. Зокрема, МНМ для знаходження параметрів регулярної 3D-структури пов'язаний із розв'язуванням такої задачі:

знайти

$$f_1^* = f_1\left(x^*, y^*\right) = \min_{x\in R^m,\ y\in R^n}\left\{ f_1(x,y) = \sum_{i=1}^{m}\sum_{j=1}^{n}\left|a_{ij} - x_i - y_j\right|\right\}, \qquad (5.19)$$

а для знаходження параметрів базисної регулярної 3D-структури – з розв'язуванням задачі:

знайти

$$f_1^* = f_1\left(x^*, y^*\right) = \min_{x\in R^m,\ y\in R^n}\left\{\sum_{i=1}^{m}\sum_{j=1}^{n}\left|a_{ij} - x_i - y_j\right| + \left|\sum_{i=1}^{m}x_i - \sum_{j=1}^{n}y_j\right|\right\}. \qquad (5.20)$$

Задачі (5.19) і (5.20) є задачами мінімізації опуклої кусково-лінійної функції $N=m+n$ змінних, де перша задача має багато розв'язків, а друга – один розв'язок. Для розв'язку негладких задач (5.19) і (5.20) можна використовувати як субградієнтні методи мінімізації негладких опуклих функцій (Шор 1979),

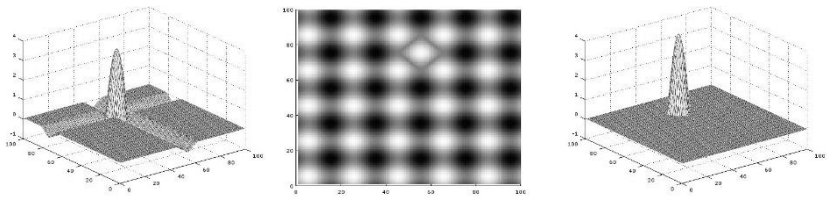
(Стецюк 2014), так і спеціальні ітеративні методи, наприклад, метод варіаційно-зважених квадратичних наближень (Мудров и Кушко 1971).

Робастність МНМ для 3D-структур з дефектами підтверджена обчислювальними експериментами у попередньому розділі. Це означає, що, якщо 3D-структура є регулярною й до неї додано одну або кілька областей з дефектами, то параметри регулярної 3D-структури більш точно визначаються за допомогою МНМ, ніж за допомогою МНК.

Це підтверджується рисунком 5.4, де проілюстровані результати роботи обох методів для автоматичного визначення нерегулярності в 3D-структурах. Це зроблене для двох зображень, які є аналогічними тим, що одержуються методами лазерної інтерферометрії (рисунок 5.4, у центрі).

Верхнє зображення містить одну область із дефектами, а нижнє – дві області з дефектами. Перша дефектна область із центром у точці з координатами $(i, j) = (55, 74)$ зі значним відхиленням від регулярної структури легко ідентифікується оператором неруйнуючого контролю якості. Друга дефектна область із центром у точці з координатами $(i, j) = (37, 37)$ може бути пропущена оператором через низьку контрастність зображення.

На рисунку 5.4 (ліворуч і праворуч) наведені абсолютні різниці між коефіцієнтами матриці для 3D-структур з дефектами й коефіцієнтами матриць регулярних 3D-структур, знайдених за допомогою МНК і МНМ. З цього рисунка випливає, що МНМ однозначно визначає координати дефектів, на відміну від МНК, який помилково виділяє ділянки уздовж стовпців і рядків кожного дефекту.



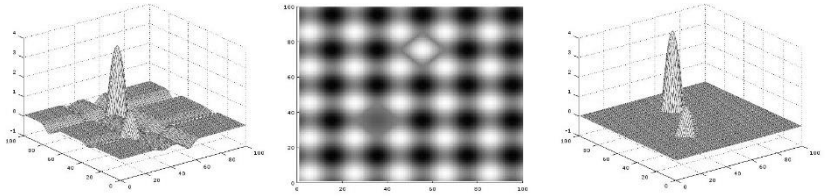


Рисунок 5.4 – Результати МНК (ліворуч) і МНМ (праворуч) для регулярних 3D-структур з дефектами

На основі Octave-Функції ralgb5 (Стецюк 2017а), однієї із сучасних реалізацій *r*-алгоритму Шора, розроблено програми мовою Octave для МНМ, де розв'язується задача (5.14), і для МНК, де розв'язується задача (5.19). Для регулярної 3D-структури, де $n = 600$ і $m = 400$, часові та інші затрати обох програм наведені в таблиці 5.2.

Обчислення проводилися на компютері Pentium 3 Ghz у системі Windows 7/32 використовуючи GNU Octave версії 3.6.4.

Таблиця 5.2 – Затрати програм для МНМ і МНК при знаходженні параметрів регулярної 3D-структури, $m = 400$ і $n = 600$

№	МНК			МНМ		
п/п	itn	Nfg	time	itn	nfg	time
1	457	700	10.58	411	502	11.76
2	494	785	11.59	413	505	11.79
3	426	647	9.77	412	503	11.77
4	453	702	10.50	412	502	11.77
5	488	764	11.34	410	499	11.69

У таблиці 5.2 для п'яти тестових прикладів $a_{ij} = u_i + v_j$, де компоненти векторів $u \in R^{400}$ і $v \in R^{600}$ генерувалися датчиком випадкових чисел у діапазоні від 0 до 10, наведено: itn – кількість ітерацій, nfg – кількість обчислень значення функції і її субградієнта, time – час виконання програми у секундах. З таблиці 5.2 бачимо, що для розв'язання задач за допомогою МНК і МНМ досить близько 10

секунд, що означає, що розроблені програми можна використовувати в діалоговому режимі для знаходження параметрів регулярних зображень.

5.4 МНК та МНМ для пошуку дефектів в регулярних 3D-структурах

Нехай A – задана $m \times n$-матриця, e_m – одиничний m-вектор, e_n – одиничний n-вектор. Якщо $\{A^*, x^*, y^*\}$ – регулярна або базисна регулярна 3D-структура, то $m \times n$-матриця A^* має вигляд $A^* = x^* e_n^T + e_m \left(y^*\right)^T$, де параметри $x^* \in \mathbb{R}^m$, $y^* \in \mathbb{R}^n$. Будемо розглядати оптимізаційні задачі для знаходження параметрів регулярної та базисної регулярної структури, щоб коефіцієнти довільної $m \times n$-матриці A найменше відрізнялися від коефіцієнтів матриці A^* за критерієм найменших квадратів та критерієм найменших модулів. Першому критерію буде відповідати метод найменших квадратів (МНК), а другому – метод найменших модулів (МНМ).

Задача 1 – МНК для регулярної 3D-структури $\{A^*, x^*, y^*\}$. Потрібно знайти такі $x^* \in R^m$ і $y^* \in R^n$, щоб коефіцієнти $m \times n$-матриці $A^* = x^* e_n^T + e_m (y^*)^T$ мінімально відхилялись від коефіцієнтів матриці A за критерієм найменших квадратів.

Задача 1 еквівалентна оптимізаційній задачі: знайти

$$(x^*, y^*) = \arg\min_{x \in R^m,\ y \in R^n} \left\{ f_1(x, y) = \sum_{i=1}^{m} \sum_{j=1}^{n} (a_{ij} - x_i - y_j)^2 \right\}, \qquad (5.21)$$

яка є задачею безумовної мінімізації опуклої квадратичної функції від $m + n$ змінних. Оптимальне значення функції $f_1^* = f_1\left(x^*, y^*\right)$ може досягатися при різних значеннях параметрів $\left(x^*, y^*\right)$, тому що задача (5.21) має безліч розв'язків (Стецюк и Савицкий 2018). Так, якщо $\left(x^*, y^*\right)$ – розв'язок задачі (5.21), то $\left(x^* + te_m, y^* - te_n\right)$ при довільному $t \neq 0$ також буде розв'язком задачі (5.21).

Задача 2 – МНК для базисної регулярної 3D-структури $\{A^{**}, x^{**}, y^{**}\}$**.** Потрібно знайти такі $x^{**} \in R^m$ і $y^{**} \in R^n$, щоб коефіцієнти $m \times n$-матриці $A^{**} = x^{**} e_n^T + e_m (y^{**})^T$ мінімально відхилялись від коефіцієнтів матриці A за критерієм найменших квадратів.

Задача 2 еквівалентна оптимізаційній задачі: знайти

$$(x^{**}, y^{**}) = \arg\min_{x \in R^n, y \in R^n} \left\{ f_2(x,y) = \sum_{i=1}^{n} \sum_{j=1}^{m} (a_{ij} - x_i - y_j)^2 + \left(\sum_{i=1}^{n} x_i - \sum_{j=1}^{m} y_j \right)^2 \right\}, \quad (5.22)$$

яка є задачею безумовної мінімізації опуклої квадратичної функції від $m+n$ змінних. Оптимальне значення функції $f_2^* = f_2(x^{**}, y^{**})$ досягається в єдиній точці $\left(x^{**}, y^{**}\right)$, яку для довільної точки $\left(x^*, y^*\right)$ – розв'язку задачі (5.21), можна розрахувати за формулами

$$x^{**} = x^* - \frac{e_m^T x^* - e_n^T y^*}{n+m} e_m, \quad y^{**} = y^* + \frac{e_m^T x^* - e_n^T y^*}{n+m} e_n.$$

При цьому виконується рівність $f_2^* = f_1^*$.

Задача 3 – МНМ для регулярної 3D-структури $\{A^*, x^*, y^*\}$. Потрібно знайти такі $x^* \in R^m$ і $y^* \in R^n$, щоб коефіцієнти $m \times n$-матриці $A^* = x^* e_n^T + e_m (y^*)^T$ мінімально відхилялись від коефіцієнтів матриці A за критерієм найменших модулів.

Задача 3 еквівалентна оптимізаційній задачі: знайти

$$(x^*, y^*) = \arg\min_{x \in R^m, y \in R^n} \left\{ f_3(x,y) = \sum_{i=1}^{m} \sum_{j=1}^{n} \left| a_{ij} - x_i - y_j \right| \right\}, \quad (5.23)$$

яка є задачею безумовної мінімізації опуклої кусково-лінійної функції від $m+n$ змінних. Оптимальне значення функції $f_3^* = f_3\left(x^*, y^*\right)$ може досягатися при різних значеннях параметрів $\left(x^*, y^*\right)$. Задача (5.23) має безліч розв'язків, як і задача (5.21).

Задача 4 – МНМ для базисної регулярної 3D-структури $\{A^{**}, x^{**}, y^{**}\}$. Потрібно знайти такі $x^{**} \in R^m$ і $y^{**} \in R^n$, щоб коефіцієнти $m \times n$-матриці $A^{**} = x^{**} e_n^T + e_m (y^{**})^T$ мінімально відхилялись від коефіцієнтів матриці A за критерієм найменших модулів.

Задача 4 еквівалентна оптимізаційній задачі: знайти

$$(x^*, y^*) = \arg\min_{x \in R^m, y \in R^n} \left\{ f_4(x,y) = \sum_{i=1}^{m} \sum_{j=1}^{n} \left| a_{ij} - x_i - y_j \right| + \left| \sum_{i=1}^{m} x_i - \sum_{j=1}^{n} y_j \right| \right\}, \qquad (5.24)$$

яка є задачею безумовної мінімізації опуклої кусково-лінійної функції від $m+n$ змінних. Оптимальне значення функції $f_4^* = f_4(x^{**}, y^{**})$ досягається в єдиній точці (x^{**}, y^{**}). При цьому виконується рівність $f_4^* = f_3^*$.

Для розв'язання задач 1 – 4 можна використовувати солвери з NEOS-сервера (Neos Solver 2025). Так, наприклад, для задач 1 та 2 можна використовувати солвери для розв'язання задач нелінійного програмування SNOPT (SNOPT 2025), IPOPT (Ipopt Documentation 2025) та інші. Задачі (5.23) та (5.24) можуть бути зведені до задач лінійного програмування і для їх розв'язання можна використати солвери Gurobi (Gurobi Optimization 2025), CPLEX (CPLEX Optimizer 2025) та інші.

Однак, задачі (5.23) та (5.24) є задачами безумовної мінімізації опуклих кусково-лінійних функцій і для їх розв'язання можна використовувати субградієнтні методи мінімізації негладких функцій. Ці методи також можна використати і для задач (5.21) та (5.22) – задач безумовної мінімізації опуклих квадратичних функцій. Для використання субградієнтних методів потрібні оракули, які обчислюють значення функції та субградієнта в точках ітераційного процесу.

Для обчислення значень функцій $f_1(x,y)$, $f_2(x,y)$ та їх градієнтів реалізовано octave-функції `fg1(x)`, `fg2(x)`, а для обчислення значень функцій $f_3(x,y)$, $f_4(x,y)$ та їх субградієнтів – Octave-функції `fg3(x)`, `fg4(x)`. Обчислення значень функцій та їх (суб)градієнтів проводилися мовою Octave

(GNU Octave 2025) з використанням матрично-векторних операцій. Задана $m \times n$-матриця A передається в Octave-функції через глобальну змінну **A**.

Значення функції $f_1(x,y)$ та її градієнта $g_{f_1}(x,y)$ обчислюються за формулами

$$f_1(x,y) = \sum_{i=1}^{m}\sum_{j=1}^{n}(a_{ij} - x_i - y_j)^2,$$

$$g_{f_1}(x,y) = \begin{cases} -2\sum_{j=1}^{n}(a_{ij} - x_i - y_j), & i = 1,...,m, \\ -2\sum_{i=1}^{m}(a_{ij} - x_i - y_j), & j = 1,...,n. \end{cases} \qquad (5.25)$$

Формули (5.25) поставлені в основу Octave-функції `fg1(x)`, код якої наведено нижче.

```
function [f,g] = fg1(x)
global A  # A - задана m*n матриця
n=columns(A); m=rows(A);
A1 = A - x(1:m,1)*ones(1,n)-ones(m,1)*x((m+1):(m+n),1)';
f=sum(sum(A1.*A1));
g=[-2*sum(A1')'; -2*sum(A1)'];
endfunction
```

Значення функції $f_2(x,y)$ та її градієнта $g_{f_2}(x,y)$ обчислюються за формулами

$$f_2(x,y) = f_1(x,y) + \left(\sum_{i=1}^{n} x_i - \sum_{j=1}^{m} y_j\right)^2,$$

$$g_{f_2}(x,y) = g_{f_1}(x,y) + \begin{cases} 2\left(\sum_{i=1}^{n} x_i - \sum_{j=1}^{m} y_j\right), & i = 1,...,m, \\ -2\left(\sum_{i=1}^{n} x_i - \sum_{j=1}^{m} y_j\right), & j = 1,...,n. \end{cases} \qquad (5.26)$$

Формули (5.26) поставлені в основу Octave-функції `fg2(x)`, код якої наведено нижче.

```
function [f,g] = fg2(x)
global A  # A - задана m*n матриця
n=columns(A); m=rows(A);
A1 = A - x(1:m,1)*ones(1,n)-ones(m,1)*x((m+1):(m+n),1)';
f=sum(sum(A1.*A1));
g=[-2*sum(A1')'; -2*sum(A1)'];
```

```
temp=sum(x(1:m,1))-sum(x((m+1):(m+n),1));
f=f+temp*temp;
g=g+2*temp*[ones(m,1); -ones(n,1)];
endfunction
```

Значення функції $f_3(x,y)$ та її градієнта $g_{f_3}(x,y)$ обчислюються за формулами

$$f_3(x,y)=\sum_{i=1}^{m}\sum_{j=1}^{n}\left|a_{ij}-x_i-y_j\right|,$$

$$g_{f_3}(x,y)=\begin{cases}-\sum\limits_{j=1}^{n} sign\left(a_{ij}-x_i-y_j\right), & i=1,...,m,\\ -\sum\limits_{i=1}^{m} sign\left(a_{ij}-x_i-y_j\right), & j=1,...,n.\end{cases} \qquad (5.27)$$

Формули (5.27) поставлені в основу Octave-функції `fg3(x)`, код якої наведено нижче.

```
function [f,g] = fg3(x)
global A # A - задана m*n матриця
n=columns(A); m=rows(A);
A1 = A - x(1:m,1)*ones(1,n)-ones(m,1)*x((m+1):(m+n),1)';
f=sum(sum(abs(A1)));
g=[-sum(sign(A1'))'; -sum(sign(A1))'];
endfunction
```

Значення функції $f_4(x,y)$ та її градієнта $g_{f_4}(x,y)$ обчислюються за формулами

$$f_4(x,y)=f_3(x,y)+\left|\sum_{i=1}^{m}x_i-\sum_{j=1}^{n}y_j\right|,$$

$$g_{f_4}(x,y)=g_{f_3}(x,y)+\begin{cases}sign\left(\sum\limits_{i=1}^{m}x_i-\sum\limits_{j=1}^{n}y_j\right), & i=1,\ldots,m,\\ -sign\left(\sum\limits_{i=1}^{m}x_i-\sum\limits_{j=1}^{n}y_j\right), & j=1,\ldots,n.\end{cases} \qquad (5.28)$$

Формули (5.28) поставлені в основу Octave-функції `fg4(x)`, код якої наведено нижче.

```
function [f,g] = fg4(x)
global A # A - задана m*n матриця
n=columns(A); m=rows(A);
A1 = A - x(1:m,1)*ones(1,n)-ones(m,1)*x((m+1):(m+n),1)';
f=sum(sum(abs(A1)));
g=[-sum(sign(A1'))'; -sum(sign(A1))'];
temp=sum(x(1:m,1))-sum(x((m+1):(m+n),1));
```

```
f=f+abs(temp);
g=g+sign(temp)*[ones(m,1); -ones(n,1)];
endfunction
```

Нижче наведемо результати обчислювальних експериментів щодо застосування *r*-алгоритму Шора (Шор 1979), (Shor 1998) для знаходження параметрів регулярних зображень невеликих розмірів – 400 пікселів по вертикалі та 600 пікселів по горизонталі, та середніх розмірів – 1000 та 1500 пікселів. При цьому розміри задач (5.21) – (5.24) для невеликих зображень будуть визначатися розмірами $n=600$ і $m=400$, а для середніх зображень – розмірами $n=1500$ і $m=1000$. Обчислення проводились на комп'ютері з процесором Intel Core i5-9400f з 2.9 GHz, використовуючи GNU Octave версії 5.1.0.

Метою обчислювальних експериментів є оцінка часу розв'язання задач (5.21) – (5.24) вказаних розмірів на сучасних ПЕОМ, якщо для реалізації МНК (задачі 1 та 2) та МНМ (задачі 3 та 4) використовується одна і та ж програмна реалізація r-алгоритму, а саме програма **ralgb5a** (Стецюк 2019). При розрахунках вибиралась одна і та ж стартова точка $x_0=\left(0,0,\ldots,0\right)^T$, використовувались такі параметри r-алгоритму: $\alpha=2$, $h_0=1$, $q_1=0.9$ (МНК) і $q_1=0.95$ (МНМ), і такі параметри зупинки: $\varepsilon_x=10^{-6}$, $\varepsilon_g=10^{-12}$, **maxitn=1000**, друк інформації про хід ітераційного процесу здійснювався через кожних **intp = 100** ітерацій. Зауважимо, що програма **ralgb5a** використовує ще два параметри r-алгоритму, значення яких дорівнюють $q_2=1.1$, $n_h=3$ та не підлягають зміні.

5.5 r-Алгоритм для розв'язання тестових задач

Перший експеримент пов'язаний з відновленням за допомогою МНК (задачі 1 та 2) параметрів регулярної та базисної регулярної структур без дефектів для розмірів $n=600$ і $m=400$, $n=1500$ і $m=1000$. Для цього п'ять різних варіантів заданої $m\times n$-матриці A розраховувались за формулою $A=xe_n^T+e_my^T$, де компоненти векторів $x\in\mathbb{R}^m$, $y\in\mathbb{R}^n$ генерувалися датчиком випадкових чисел rand("seed",2025) в діапазоні від 0 до 265. Для обчислення значень функцій

$f_1(x,y)$, $f_2(x,y)$ та їх градієнтів використовуються octave-функції `fg1(x)` та `fg2(x)`.

Перший експеримент реалізує такий Octave-код.

```
clear;              # очищуємо пам'ять для першого тесту
global A            # A - задана m*n матриця
n = 600, m = 400,   # n = 1500, m = 1000,
# Встановлюємо параметри r-алгоритму
alpha = 2, h0 = 1.0,  q1 = 0.9,
epsx = 1.e-6, epsg = 1.e-12, maxitn = 1000, intp=100,
# Запускаємо на розрахунок по п'ять тестових задач (1) та (2)
scale = 256; rand("seed",2025); tab12 = []; ntest = 5;
for ii=1:ntest
   # Генеруємо матрицю A та параметри базисної регулярної 3D-структури
   x = scale*rand(m,1);  y = scale*rand(n,1);
   A = x*ones(1,n)+ones(m,1)*(y)';
   temp = (sum(x)-sum(y))/(n+m);
   xb = x - temp*ones(m,1);  yb = y + temp*ones(n,1);
   # Розв'язуємо задачу (1)
   x0=zeros(m+n,1); tstart=time();
   [xr1,fr1,itn1,nfg1,istop1] =
ralgb5a(@fg1,x0,alpha,h0,q1,epsg,epsx,maxitn,intp);
   temp=(sum(xr1(1:m))-sum(xr1(m+1:m+n)))/(n+m);
   xr1 = xr1 - temp*[ones(m,1);ones(n,1)];
   itn1, nfg1, istop1, fr1, dx1=norm(xr1-[xb; yb]),
   time1=time()-tstart,
   # Розв'язуємо задачу (2)
   x0=zeros(m+n,1); tstart=time();
   [xr2,fr2,itn2,nfg2,istop2] =
ralgb5a(@fg2,x0,alpha,h0,q1,epsg,epsx,maxitn,intp);
   itn2, nfg2, istop2, fr2, dx2=norm(xr2-[xb; yb]),
   time2=time()-tstart,
   tab12 = [tab12; ii itn1 nfg1 time1 dx1 fr1 itn2 nfg2 time2 dx2 fr2];
endfor
tab12,  # друкуємо результати розрахунку для першого тесту
```

Результати першого експерименту для $n=600$ і $m=400$ наведені в табл. 5.3, а для $n=1500$ і $m=1000$ – в табл. 5.4, де itn – кількість ітерацій *r*-алгоритму, nfg – кількість обчислень значення функції та її градієнта за формулами (5.25) та (5.26), time – час виконання програми в секундах, delta – норма відхилення знайденого наближення до точки мінімуму від точки мінімуму, fr – оптимальне значення функції.

Таблиця 5.3 – Затрати *r*-алгоритму для МНК ($m=400$ і $n=600$, $q_1=0.9$)

№	Задача (5.21)					Задача (5.22)				
п/п	itn	nfg	time	delta	fr	itn	nfg	time	delta	fr
1	291	579	4.28	1.5e-06	1.4e-09	295	586	4.54	1.6e-07	1.5e-11

2	306	606	4.73	4.6e-07	9.7e-11	286	573	4.57	7.9e-08	3.5e-12
3	308	612	4.58	2.3e-07	2.9e-11	276	549	4.12	2.1e-07	2.3e-11
4	298	593	4.42	1.6e-07	1.6e-11	291	583	4.32	3.2e-07	4.1e-11
5	299	595	4.43	2.1e-07	1.9e-11	284	562	4.28	1.0e-07	5.5e-12

Таблиця 5.4 – Затрати *r*-алгоритму для МНК ($m=1000$ і $n=1500$, $q_1=0.9$)

№	Задача (5.21)					Задача (5.22)				
п/п	itn	nfg	time	delta	fr	itn	nfg	time	delta	fr
1	298	604	30.12	1.3e-07	1.9e-11	303	615	30.64	3.5e-07	1.8e-10
2	301	615	30.49	3.2e-07	1.4e-10	304	614	30.61	2.5e-07	6.8e-11
3	275	557	27.76	2.2e-07	6.3e-11	294	595	29.70	3.8e-07	1.7e-10
4	306	616	30.60	2.7e-07	8.6e-11	325	657	32.87	2.4e-07	6.5e-11
5	304	612	30.40	1.7e-07	3.3e-11	292	592	29.52	3.2e-07	1.2e-10

З колонки delta табл. 5.3 та 5.4 видно, що знайдені параметри регулярних 3D-структур відхиляються від точних на величину порядку 10^{-7} за евклідовою нормою. Це означає, що наближення до точок мінімуму знайдено досить точно, про що сигналізують і знайдені з точністю $10^{-12} \div 10^{-10}$ мінімальні значення функцій $f_1(x,y)$, $f_2(x,y)$ (див. колонку fr). Для розв'язання задач 1 та 2 за допомогою МНК на базі *r*-алгоритму достатньо менше 5 секунд, якщо $n=600$ і $m=400$, та близько 30 секунд, якщо $n=1500$ і $m=1000$. Це означає, що розроблені програми можна використовувати в діалоговому режимі для знаходження дефектів у регулярних 3D-структурах розмірів $n=600 \div 1500$ і $m=400 \div 1000$, тому що для цього потрібно від 5 до 30 секунд на сучасних персональних ЕОМ з використанням GNU Octave версії 5.1.0.

Другий експеримент пов'язаний з відновленням за допомогою МНМ параметрів базисної регулярної структури з дефектами у невеликій області ($441=21\times 21$ піксель). Він побудований по аналогії з першим експериментом, де коефіцієнти $m\times n$ матриці A для регулярної структури коригувалися за формулою $A=A+0.2\times A(190:210,290:310)$. Для обчислення значень функцій $f_3(x,y)$, $f_4(x,y)$ та їх субградієнтів використовуються Octave-функції `fg3(x)` та `fg4(x)`.

Другий експеримент реалізує такий Octave-код.

```
clear;              # очищуємо пам'ять для другого тесту
global A            # A - задана m*n матриця
n = 600, m = 400,  # n = 1500, m = 1000,
# Встановлюємо параметри r-алгоритму
alpha = 2, h0 = 1.0,  q1 = 0.95,
epsx = 1.e-6, epsg = 1.e-12, maxitn = 1000, intp=100,
# Запускаємо на розрахунок по п'ять тестових задач (3) та (4)
scale = 256; rand("seed",2025); tab34 = []; ntest = 5;
for ii=1:ntest
   # Генеруємо матрицю A та параметри базисної регулярної 3D-структури
   x = scale*rand(m,1);  y = scale*rand(n,1);
   A = x*ones(1,n)+ones(m,1)*(y)';
   temp = (sum(x)-sum(y))/(n+m);
   xb = x - temp*ones(m,1);  yb = y + temp*ones(n,1);
   i1=10; # додаємо 21*21 дефект до матриці A
   A(m/2-i1:m/2+i1,n/2-i1:n/2+i1)=1.2*A(m/2-i1:m/2+i1,n/2-i1:n/2+i1);
   # Розв'язуємо задачу (3)
   x0=zeros(m+n,1); tstart=time();
   [xr3,fr3,itn3,nfg3,istop3] =
ralgb5a(@fg3,x0,alpha,h0,q1,epsg,epsx,maxitn,intp);
   temp=(sum(xr3(1:m))-sum(xr3(m+1:m+n)))/(n+m);
   xr3 = xr3 - temp*[ones(m,1);ones(n,1)];
   itn3, nfg3, istop3, fr3, dx3=norm(xr3-[xb; yb]),
   time3=time()-tstart,
   # Розв'язуємо задачу (4)
   x0=zeros(m+n,1); tstart=time();
   [xr4,fr4,itn4,nfg4,istop4] =
ralgb5a(@fg4,x0,alpha,h0,q1,epsg,epsx,maxitn,intp);
   itn4, nfg4, istop4, fr4, dx4=norm(xr4-[xb; yb]),
   time4=time()-tstart,
   tab34 = [tab34; ii itn3 nfg3 time3 dx3 fr3 itn4 nfg4 time4 dx4 fr4];
endfor
tab34,  # друкуємо результати розрахунку для другого тесту
```

Результати другого експерименту для $n=600$ і $m=400$ наведені в табл. 5.5, а для $n=1500$ і $m=1000$ – в табл. 5.6, де itn – кількість ітерацій *r*-алгоритму, nfg – кількість обчислень значення функції та її субградієнта за формулами (5.27) та (5.28), time, delta та fr – такі ж самі позначення, які були використані для таблиць 5.3 та 5.4.

Таблиця 5.5 – Затрати *r*-алгоритму для МНМ ($m=400$ і $n=600$, $q_1=0.95$)

№	Задача (5.23)					Задача (5.24)				
п/п	itn	nfg	time	delta	fr	itn	nfg	time	delta	fr
1	374	550	5.56	6.8e-07	23377.4	374	550	5.60	7.2e-07	23377.4
2	377	555	5.56	7.5e-07	24339.1	378	556	5.60	7.5e-07	24339.1
3	375	551	5.53	7.5e-07	20420.9	376	552	5.63	7.2e-07	20420.9
4	376	552	5.53	7.7e-07	21921.2	376	552	5.59	7.4e-07	21921.2
5	374	548	5.51	8.2e-07	21439.4	375	549	5.58	7.5e-07	21439.4

Таблиця 5.6 – Затрати *r*-алгоритму для МНМ ($m=1000$ і $n=1500$, $q_1=0.95$)

№	Задача (5.23)					Задача (5.24)				
п/п	itn	nfg	time	delta	fr	itn	nfg	time	delta	fr
1	386	577	38.42	8.3e-07	24611.2	385	576	38.36	8.4e-07	24611.2
2	386	576	38.28	8.2e-07	24757.9	386	576	38.33	8.5e-07	24757.9
3	386	577	38.62	7.9e-07	21368.0	386	577	38.79	8.1e-07	21368.0
4	386	576	38.73	8.1e-07	22587.6	386	576	38.38	8.0e-07	22587.6
5	386	576	38.35	8.1e-07	23040.4	386	576	38.41	8.2e-07	23040.4

Знайдені параметри регулярних 3D-структур відхиляються від точних менше ніж на 10^{-6} за евклідовою нормою (див. колонку delta в табл. 5.5 та 5.6), що означає, що наближення до точок мінімуму знайдено досить точно. Для всіх десяти розрахунків знайдені мінімальні значення функцій $f_3(x,y)$ та $f_4(x,y)$ (див. колонку fr) співпадають з точністю до всіх шести значущих цифр. Для розв'язання задач 1 та 2 за допомогою МНМ на базі *r*-алгоритму достатньо менше 6 секунд, якщо $n=600$ і $m=400$, та менше 40 секунд, якщо $n=1500$ і $m=1000$. Це означає, що на сучасних персональних ЕОМ розроблені програми можна використовувати в діалоговому режимі для аналізу дефектів у регулярних зображеннях невеликих розмірів (6 секунд, GNU Octave версії 5.1.0) та середніх розмірів (40 секунд).

5.6 Висновки до п'ятого розділу

Досліджено регулярні та базисні регулярні 3D-структури, для них сформульовано оптимізаційні задачі знаходження найкращих за L_p-нормою параметрів та методи розв'язування цих задач. Особливу увагу приділено окремим випадкам методів – методу найменших квадратів і методу найменших модулів.

Показано, що при відновленні параметрів 3D-структур з дефектами метод найменших модулів є стійкішим, ніж метод найменших квадратів. Наведено результати обчислювальних експериментів для програмних реалізацій методів на основі *r*-алгоритму, які дозволяють оцінити їхні часові затрати при знаходженні

параметрів регулярних 3D-структур великих розмірів (400 рядків і 600 стовпців та 1000 рядків і 1500 стовпців).

Наведено результати застосування *r*-алгоритму для оцінки часу розв'язання на сучасних ПК тестових задач для регулярних зображень невеликих розмірів – 400 пікселів по вертикалі та 600 пікселів по горизонталі, та середніх розмірів – 1000 та 1500 пікселів. Перший експеримент пов'язаний з відновленням за допомогою МНК параметрів базисної регулярної структури без дефектів, другий експеримент – з відновленням за допомогою МНМ параметрів базисної регулярної структури з дефектами у невеликій області (441 піксель). Розроблені програми можна використовувати в діалоговому режимі для аналізу дефектів у регулярних зображеннях невеликих розмірів (5 секунд) та середніх розмірів (40 секунд).

Регулярні зображення з областями дефектів є характерними при неруйнуючому контролі якості тонкостінних багатошарових композиційних матеріалів за допомогою методів лазерної інтерферометрії, таких, як метод голографічної інтерферометрії, метод спекл-інтерферометрії та метод ширографії (Lobanov et al. 2005).

Описані в розділі методи для пошуку дефектних ділянок дозволяють автоматизувати процес визначення місця розташування дефектів у відповідальних елементах конструкцій і знизити вплив людського фактору при неруйнуючому контролі якості. Оптимізаційні задачі для зображень із 400 пікселями по вертикалі та 600 пікселями по горизонталі можна розв'язувати в діалоговому режимі, тому що для цього потрібно близько 5 секунд на сучасних персональних ЕОМ з використанням GNU Octave версії 5.1.0.

Розроблені Octave-програми легко переписати мовами Фортран та С, використовуючи бібліотеку базових підпрограм лінійної алгебри BLAS (Basic Linear Algebra Subprograms) або бібліотеку математичних прикладних програм Intelr Math Kernel Library (IntelR MKL), які оптимізовані під сучасні обчислювальні машини.

Це може дозволити значно прискорити методи для розв’язування великих задач (з тисячею і більше змінними), наприклад, за рахунок використання обчислювальних потужностей графічного процесора, які в рази перевищують обчислювальні потужності класичних процесорів.

Зокрема, істотне прискорення можна одержати, використовуючи для розрахунків технологію CUDA на графічних прискорювачах. Це підтверджує гібридна реалізація *r*-алгоритму (Стецюк, Хіміч, та Сидорук 2016), де скорочення часу, яке досягається при розв’язувані задач із 1000–8000 змінними, варіюється від 14 до 18 разів.

Матеріал розділу базується на таких роботах: (Стецюк, Савицький та Жидков 2023), (Жидков, Стецюк та Хом’як 2025), (Zhydkov 2024).

РОЗДІЛ 6. ПОШУК ДЕФЕКТІВ У ПЕРІОДИЧНИХ СТРУКТУРАХ

Виявлення дефективних частин та елементів є важливою складовою покращення якості, а також обслуговування та запобігання аваріям. Деякі інструментальні методи побудови зображень досліджуваного об’єкта для тестування та інспекції мають покращені можливості для виявлення дефектів, але наразі мають і недоліки, такі як низька точність класифікації дефектів та потреба в людині-операторі. Таким чином, існує великий попит на рішення, яке є достатньо надійним, може виключити людський фактор та бути математично обґрунтованим, на відміну від ймовірнісних та нейромережевих моделей.

В цьому розділі представлено загально застосовне алгоритмічне рішення для системи, яка може аналізувати зображення та виявляти дефекти, порівняно або навіть краще за людину-оператора. Загальний принцип заснований на тому, що деякі загальні властивості про аналізований об’єкт відомі заздалегідь. Запропонований метод виявляє дефекти шляхом розпізнавання не дефектів, а періодичного фону. Окрім того, розроблено математичне формулювання для функції похибок, адаптованої до апроксимації даних та зображень з дефектами, а її придатність була проаналізована, включаючи обчислювальний експеримент для верифікації адекватності. Використовуючи ab initio міркування, запропоновано новий тип функції похибок, спеціально адаптований до апроксимації наборів даних з високою щільністю викидів, з ефективністю, що порівняна, а в деяких випадках навіть перевищує ефективність L_1-норми в цьому аспекті.

Таким чином, запропоновано алгоритм, при якому зображення аналізованого об’єкту може бути реконструйоване в його «ідеалізованому», бездефектному стані і, далі порівнюючи вихідне зображення з цією реконструкцією, можуть бути виявлені дефекти. На наступній стадії статистична модель, використана у формулюванні запропонованої функції похибок, може бути застосована для оцінки ймовірності дефекту в області, використовуючи аналіз відхилень та похибок вимірювань. Цей підхід продемонстровано на

тестовому прикладі. Описано всі суттєві стадії алгоритму, такі як попередній аналіз та побудова загальної моделі. Алгоритм продемонстрував здатність виявлення дефектів без втручання людини-оператора та машинного навчання, використовуючи лише базову інформацію про досліджуваний об’єкт.

Періодичні зображення з дефектами характерні для результатів інтерферометрії, а такж для зображень регулярних зварних конструкцій, отримані методами спекл-інтерферометрії та шерографії (Lobanov et al. 2005). Методи лазерної інтерферометрії неруйнівного контролю якості (НКЯ) полягають у вимірюванні переміщень (деформацій) поверхонь досліджуваних об'єктів, що виникають внаслідок теплових або механічних навантажень. У випадку НКЯ структурних елементів, що мають періодичну структуру (трансляційну симетрію), результат вимірювання теж повинен мати властивості періодичної структури. Якщо у вимірюваній структурі є відхилення від цієї симетрії, то таке ж відхилення має проявитися і в результатах вимірювань.

У теперішній час завдання пошуку дефектів найчастіше вирішує оператор (спостерігач); з цієї причини виникає потреба в автоматизації цього процесу для зменшення впливу людського фактору при проведенні НКС у методі шерографії. Застосування звичайних підходів, які використовуються для вирішення задач розпізнавання образів, до задачі виявлення дефектів у таких зображеннях є надзвичайно складним через різноманітність типів дефектів. Навіть якщо буде створений великий каталог зображень дефектів, це не виключає можливість помилки через появу раніше невідомого типу дефекту. Примітно, що при пошуку дефекту людиною-оператором виникають подібні проблеми.

Таким чином, більш загальним і надійним методом є реконструкція «ідеального» образу періодичної структури (яким би він був без дефектів) з подальшим визначенням дефекту як відхилення існуючого зображення від «ідеального». Про розвиток такого методу йдеться в цьому розділі.

Матеріал розділу викладено в такому порядку. У підрозділі 6.1 викладено запропоновану статистичну модель зображення конструкції з дефектами та отримані математичні постановки задачі знаходження зображення «ідеальної»

конструкції, яка за своїми властивостями повторює початково задане зображення конструкції, але в якому немає дефектів. У підрозділі 6.2 представлено математичну модель для ідеалізованої структури та формулювання задачі оптимізації. У підрозділі 6.3 розглядаються підходи до розв'язання задачі, наводяться алгоритми та приклад розв'язання задачі.

6.1 Статистична модель задачі

Нехай матриця $M \in \mathbb{R}^{n \times m}$ – деяке експериментально отримане двовимірне зображення розміру $n \times m$, що відповідає досліджуваній структурі, а матриця $L \in \mathbb{R}^{n \times m}$ – двовимірне зображення такого ж розміру, що відповідає ідеальній моделі структури, яка не містить дефектів. Припустимо також, що дефекти займають не більше певної заданої частини зображення. Відношення площі цієї частини до площі всього зображення позначимо як p до 1. Зробимо також таке припущення: зображення отримано експериментальним шляхом, систематичних похибок вимірювань на ньому немає, а шум на зображенні лише білий. Це припущення досить загальне та справедливе для даних переважної більшості вимірювальних установок, зокрема для даних спекл-інтерферометрії.

Типи експериментальних пристроїв, для яких це припущення не виконується, є окремими випадками, і для них статистична модель будується індивідуально. Вони не будуть розглядатися у цій роботі.

Зі зробленого припущення випливає, що $D = M - L$, відхилення зображення правильно побудованого ідеалізованого зображення від виміряного зображення, повинно мати нормальну щільність розподілу значень елементів, відповідну білому шуму, з точністю до p. Це забезпечує критерій для визначення того, чи збігається якась модель ідеалізованого зображення з експериментальними даними.

По суті, дефекти визначаються як систематичні похибки, і тут ховаються два обмеження методу: наявність загальної математичної моделі досліджуваного об'єкту та відсутність систематичних похибок вимірювань. Позаяк ці умови все

одно мають бути виконані практично для будь-якого завдання дефектоскопії, це не можна вважати занадто суворим обмеженнями.

Припустимо, що шум зображення D має приблизно нормальний розподіл зі стандартним відхиленням σ. Тоді ймовірність того, що деяка точка L відповідає точці зображення M, відповідає ймовірності хоча б однієї з двох подій: вона відхиляється з імовірністю, визначеною нормальним розподілом, та/або потрапляє на дефект (систематичну помилку) з імовірністю p.

Таким чином, ймовірність відповідності становить

$$V(i,j)=1-\left(1-e^{-D^2(i,j)/2\sigma^2}\right)\times(1-p), \tag{6.1}$$

де $D(i,j)$ – значення комірки i-го стовпця та i-го рядка матриці. Дотримуючись принципу максимальної правдоподібності (Никулин 1984), на основі цього побудуємо вагову функцію

$$W(M-L)=\sum_{i=1}^{n}\sum_{j=1}^{m}\ln\left(1-\left(1-e^{-\left(M(i,j)-L(i,j)\right)^2/2\sigma^2}\right)\times(1-p)\right) \tag{6.2}$$

як визначення того, наскільки добре збігаються зображення. Для наочності цю вагову функцію для однієї точки, що використовується в роботі, показано на рисунку 6.1. Тут і далі, якщо не вказано інше, вважається, що $p=0.1$, що є реалістичною оцінкою щільності дефектів для більшості застосувань.

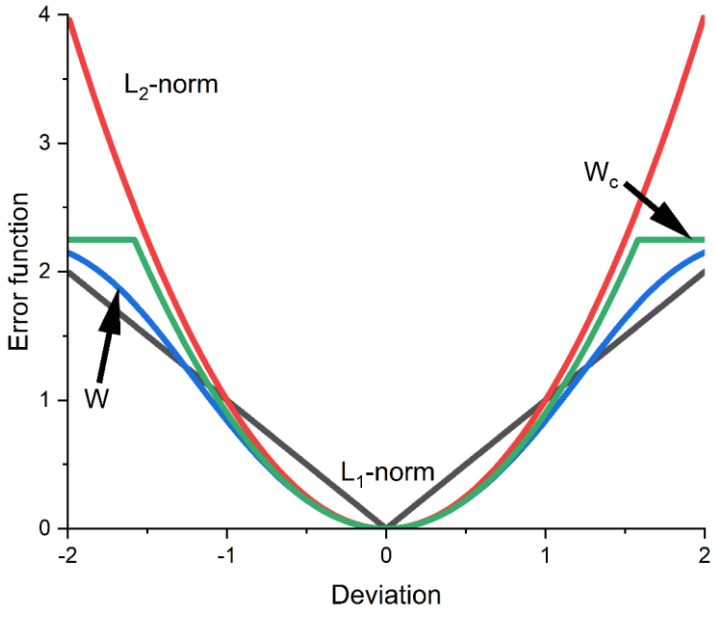


Рисунок 6.1 – Вагові функції для однієї точки

Звернемо увагу, що для малих відхилень ця вагова функція близька до L_2-норми. Якщо врахувати весь діапазон відхилень, то він є найближчим до L_1-норми. Для практичних цілей цю вагову функцію можна замінити «урізаною» – нормою, яку можна обчислити так:

$$W_c(D) = \sum_{i=1}^{n} \sum_{j=1}^{m} \min\left(D^2(i,j), -2\sigma^2 \ln(p)\right), \tag{6.3}$$

і яка дуже близька до (6.2), як видно з рисунку 6.1. Вона розраховується значно швидше за рахунок меншої кількості ресурсомістких операцій порівняно з (6.2). Крім економії обчислювального часу, ця норма має ще й додаткове значення: норма з відкиданням точок, відхилення яких перевищує певне значення.

Якщо для кожного розрахунку норми вибрати порогове значення відхилення таким чином, щоб відкинути деяку постійну p-ту частину точок, то отримаємо норму, яка зберігає всі переваги W_c-норми, маючи переваги перед нею, наприклад: немає необхідності заздалегідь явно знати відхилення гаусового шуму.

Практична реалізація для використання в алгоритмі оптимізації тут виконується так. По-перше, записуються всі відмінності між двома зображеннями. Далі, p частина цих відмінностей відкидається, починаючи з максимальних за модулем. По суті, відмінності піддаються медіанній фільтрації з кінцевим ефектом, звичайним для типу фільтрації: викиди відкидаються, а разом з ними з великою ймовірністю і систематичні помилки; таким чином, під час використання функцій помилок оптимізація ефективно виконується на ідеалізованому зображенні, оскільки викиди, які часто представляють дефекти, відкидаються.

Така реалізація має одну маленьку, але суттєву перевагу: як можна побачити на рисунку 6.1, обмежена L_2-норма по суті не є опуклою, що робить застосування переважної більшості існуючих алгоритмів оптимізації в процесі апроксимації дуже проблематичним. Однак у процесі сортування/відбракування неопукла

частина по суті видаляється, роблячи, по суті, опуклою лише частину, що залишилася, враховану в обчисленні функції помилок.

Для оцінки ефективності цих функцій похибки було проведено простий обчислювальний експеримент: для лінійної регресії при різних p і амплітудах викидів, використовуючи три різні функції похибок для процесу апроксимації.

Для обчислювального експерименту згенеровані набори даних складалися зі 100 точок, рівномірно розподілених від 0 до 1 на осі x, лінійної функції $y=1+2x$, додатково з шумом Гауса та доданими викидами з певною ймовірністю. Експерименти з обчисленнями проводилися для різних величин доданого гаусового шуму σ, амплітуди викидів, моно та уніполярних, щільності викидів (ймовірність того, що точка даних генерується як викид), заданої очікуваної щільності викидів (параметр для функції помилки у (6.2) і (6.3)).

Результуюча загальна похибка у визначенні нахилу (як найбільш типова) для різних типів функції похибки зображена на рисунку 6.2 як графіки залежності від різних параметрів. Використано такі позначення: червоні кола для L_1-норми, чорні квадрати для L_2-норми та сині трикутники для усіченої L_2-норми.

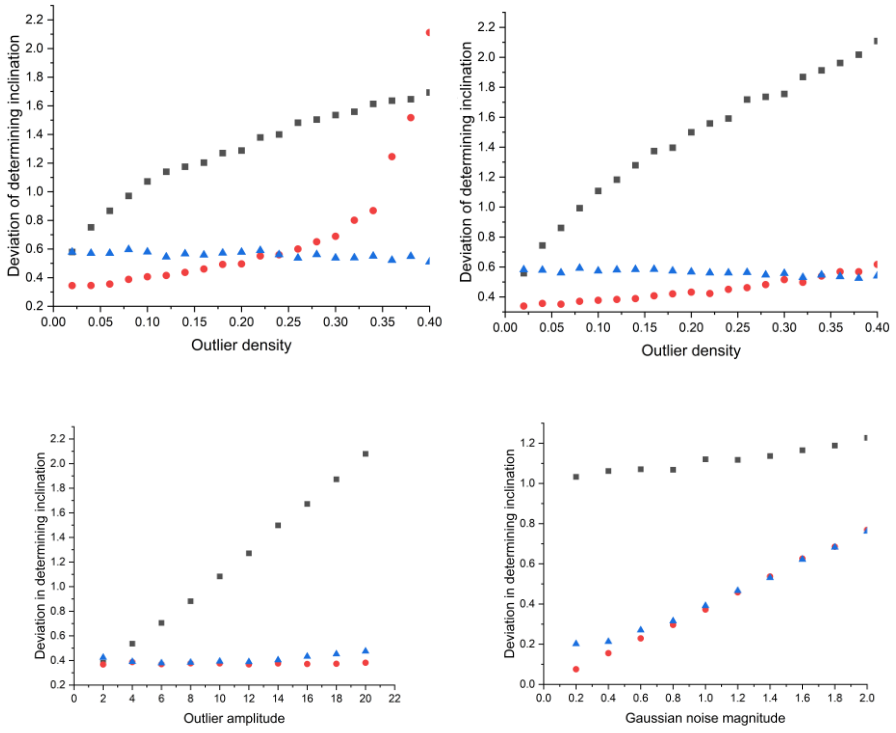


Рисунок 6.2 – Графіки залежності функцій похибок від різних параметрів

Ліворуч угорі на рисунку 6.2 зображено графік поведінки функцій похибок у порівнянні з різними монополярними (відхилення лише в одному напрямку) щільністю викидів, амплітуда викидів дорівнює 10, шум $\sigma = 1$ і $p = 0.5$ в функції похибок. Справа вгорі однакова для випадкових викидів полярності. Тут цікаво відзначити, що усічена L_2 стає справді ефективною лише при високій щільності дефектів.

Також слід зазначити, що апроксимація з використанням L_1-норми поводиться таким чином, що залишає половину точок вище лінійної апроксимації та половину нижче. Це може призвести до певної ситуації «байдужої рівноваги», наприклад кінцева область, де градієнт функції помилки дорівнює нулю. Крім того, через таку саму поведінку дані з монополярними викидами гірше апроксимуються; тут слід зазначити, що дефекти часто є саме такими викидами.

Що стосується інших характерних особливостей поведінки, лівий нижній графік демонструє, що на поведінку помилок функцій L_1 та обмеженої L_2 не впливає величина викиду. Тут для функції помилки $p = 0.25$ та щільність викидів дорівнює 0.1. Правий нижній графік демонструє залежність точності від величини гаусового шуму, при цьому щільність викидів однакова і їх величина дорівнює 10; на цьому графіку легко побачити порівнянну продуктивність – на основі L_1-норми та на основі усіченої L_2-функції похибок, здатні ігнорувати викиди, які за величиною значно перевищують гаусівський шум.

На завершення відмітимо, що L_2-норма не підходить для апроксимації зображень з дефектами та даних із викидами, L_1-норма є хорошою функцією повної помилки для апроксимації та розпізнавання зображень з дефектами, але усічена L_2 може бути більш ефективними в екстремальних випадках.

6.2 Модель ідеалізованої структури та оптимізаційна задача

Побудова моделі зображення ідеалізованої періодичної структури з трансляційною симетрією в одному напрямку може бути виконана, якщо зображення заповнено у всіх клітинках за формулою $L(i, j) = F(ai + bj + c)$, де $F(\cdot)$ – деяка періодична функція, що відповідає очікуваному профілю періодичної структури. Враховуючи те, що ми маємо справу не з самою періодичною структурою, а з її експериментально отриманим зображенням, потрібні певні поправки до формули.

Зокрема, проблема, що розглядається, повинна брати до уваги спотворення результуючого зображення, які можуть бути присутніми у властивих методам лазерної інтерферометрії зображеннях, що використовуються, наприклад, нерівність фону, як представлено в прикладах зображень на рисунку 6.3, особливо лівий; неперпендикулярна орієнтація осі лінзи до досліджуваної поверхні, що призводить до нерівномірної чутливості та трапецієподібної аберації, як на рисунку 6.3, справа; також деталь може мати додаткову кривизну, яка вимагає спеціальної корекції, специфічної для певного типу деталі.

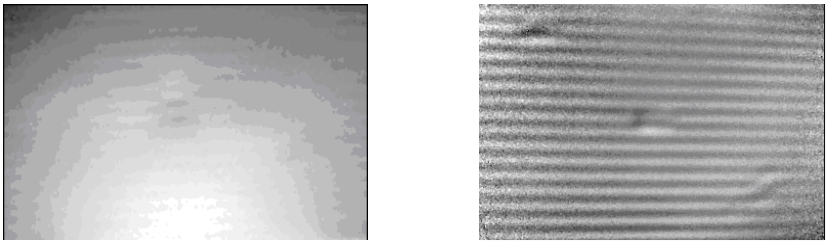


Рисунок 6.3 – Приклади спотворення результуючого зображення

Таким чином, доцільно зробити модель з регульованими параметрами, використовуючи деякі лінійні комбінації:

$$L = Q_1 + F(Q_2) \times Q_3, \tag{6.4}$$

де $\times$ – операція поелементного множення матриці, Q_i – це матриці тієї ж розмірності, що й вихідне зображення, сформовані як лінійна комбінація:

$$Q_i = \sum_{j=1}^{n_i} a_{i,j} H_j . \tag{6.5}$$

Тут усі H_j мають той самий розмір, що й початкове зображення, і є лінійно незалежними, щоб сформувати базис, на якому можна побудувати Q_i відповідно до зображення. Базис побудований таким чином, що її елементи є лінійно незалежними та містять базові лінійні та квадратичні елементи вздовж координат зображення, також можуть містити елементи, які відповідають властивостям об'єкта (характерна кривина тощо) для кращого підгонки.

Аналогічно, для об'єкта з трансляційною симетрією в 2 напрямках можна використовувати

$$L = Q_1 + F\left(Q_2\right) \times F\left(Q_3\right) \times Q_4 , \tag{6.6}$$

де F_1, F_2 – періодичні функції, застосовні як складові зображення.

Модель є досить загальною і дозволяє описати фактично будь-яку періодичну структуру, яка зустрічається на практиці. Щодо вигляду функцій F, то часто достатньо апроксимації за допомогою ряду Фур'є. Використовуючи результати попереднього розділу та формулу (6.4), для зображення з одним напрямком симетрії можна сформулювати таку задачу оптимізації:

$$\Phi^* = \Phi\left(a_{1,1}^*, ..., a_{3,n}^*\right) = \min_{(a_1,...,k_3)\in R^{27}} W\left(M - Q_1 + F\left(Q_2\right) \times Q_3\right), \tag{6.7}$$

де використовується норма W та періодична функція F.

Аналогічно, коли є трасляційна симетрія в 2 напрямках, оптимізаційна задача може бути сформульована таким чином:

$$\Phi^* = \Phi(a_1^*, ...k_4^*) = \min_{(a_1,...k_4)\in \mathbb{R}^{36}} W(M - Q_1 + F_1(Q_2) \times F_2(Q_3) \times Q_4), \tag{6.8}$$

де F_1, F_2 – використані періодичні функції.

6.3 Підходи та приклади розв'язання проблем

Легко побачити, що задача (6.7) у загальному випадку є неопуклою та багатоекстремальною. Безпосереднє розв'язання цих задач, використовуючи

лише вагову функцію, виявляється дуже затратним. Це підтверджено обчислювальними експериментами для пошуку ідеалізованого зображення для задач, як на рисунку 6.3, де розміри зображення були від 600×360 до 2048×2048. Крім того, у цих задачах велике значення має правильний спосіб вибору необхідного екстремуму. Тому доцільно провести попередній аналіз наявних зображень, щоб знайти хороші вихідні точки, в околицях яких буде здійснюватися пошук шуканого екстремуму. Попередній аналіз зображення складається з двох етапів.

По-перше, зображення «вирівнюється» за допомогою першого члена форми (6.5) або (6.6). Це робиться з використанням простої згортки

$$M_e = M - \sum_{i=1}^{n} a_i H_i, \quad \{a\} = \hat{H}^{-1}\langle HM\rangle, \tag{6.9}$$

де $\hat{H}_{i,j} = \langle H_i H_j \rangle$ – це внутрішня згортка базису H, щоб виправити неортогональність базису. Останнє не є необхідним, але значно покращує зручність побудови базису для різних сценаріїв.

На другому етапі застосовується перетворення Фур'є до вирівняного зображення, виявляючи періодичності, які можна представити як набір оцінок моменту кореляції для набору гармонійних функцій. Двовимірне перетворення Фур'є підходить для визначення того, яка з гармонійних функцій найбільше підходить для апроксимації зображення. Приклад результатів для двох етапів попереднього аналізу: вирівняне по Q_1 зображення (справа) представлено на рисунку 6.4 разом з оригінальним зображенням (ліворуч) з видимими місцями помилок, обведеними червоним кольором. Аналіз Фур'є представлено на рисунку 6.5.

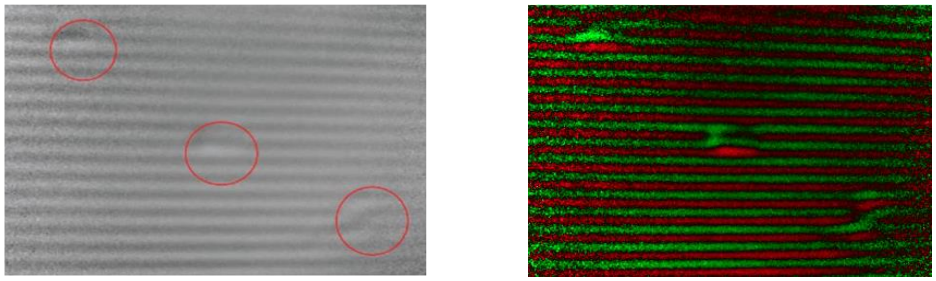


Рисунок 6.4 – Перший етап попереднього аналізу зображення

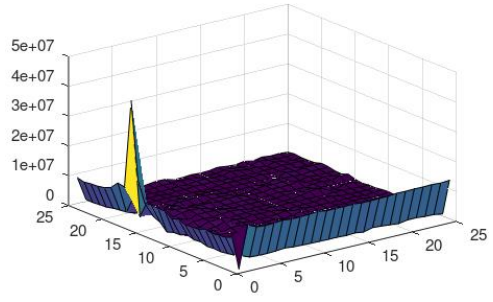


Рисунок 6.5 – Застосування перетворення Фур’є

Слід зазначити, що задачі (6.7), (6.8) можна розв’язувати як програмним забезпеченням для розв’язування задач лінійного програмування, так і алгоритмами мінімізації негладких опуклих функцій (Шор 1979). Подібні задачі розглядалися в (Стецюк и Савицкий 2018), де досліджувалася стійкість методу найменших абсолютних значень для знаходження параметрів регулярних тривимірних структур за наявності одного або кількох дефектів. У (Стецюк и Савицкий 2018) було показано, що оптимізація за L_1-нормою здатна відновити параметри регулярних тривимірних структур з однією або декількома дефектними ділянками, що неможливо з нормою L_2.

Після завершення другого етапу легко визначити, яка з моделей (6.4) або (6.6) краще підходить для апроксимації ідеалізованого образу та початкової відправної точки для процесу оптимізації. Зокрема, параметри основної гармоніки разом із найближчими до неї частотами дозволяють визначити коефіцієнти для Q_3 та Q_2, які використовуються як складові для періодичних функцій у виразах (6.4) та (6.6).

Для визначення типу функції F достатньо проаналізувати основну та вищу гармоніки. Якщо гармоніки, крім основної, менші за рівень шуму, то функція $\sin$ приймається як F; якщо це не так, то приймається ряд Фур’є, параметри якого визначаються за даними попередньо виконаного двовимірного перетворення Фур’є. Початкові коефіцієнти для матриці можуть бути встановлені рівними отриманим при початковій обробці зображення (6.9).

Після оцінки параметрів періодичності зображення за допомогою перетворення Фур’є визначаються початкові наближення для Q_3 та Q_2, такі як

загальна фаза рівня та амплітуда зображення, апроксимовані, як показано на рисунку 6.6 (фаза ліворуч, амплітуда праворуч) і розкладені за базисом H, як у (6.9).

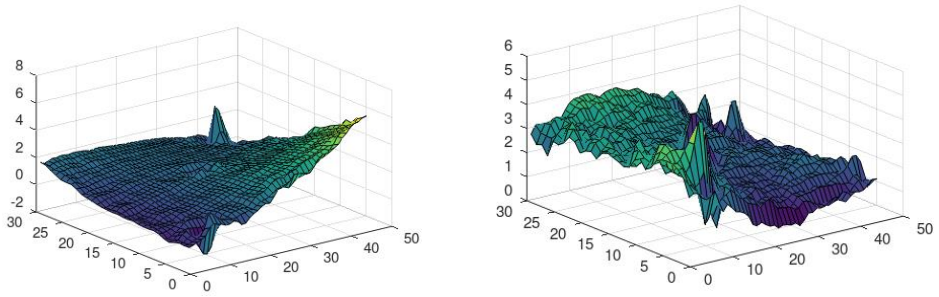


Рисунок 6.6 – Розподіл фаза та амплітуди відносно основної гармоніки

Це хороша початкова точка для початку процесу оптимізації відповідно до формулювання (6.7) або (6.8), що призводить до побудови ідеалізованого зображення, приклад якого, що відповідає вихідному зображенню з рисунку 6.4, показано на рисунку 6.7.

Якщо щільність розподілу на зображенні $D = M - L$ не збігається з нормальним розподілом з відносним відхиленням більше ніж p, то це вказує на помилку у визначенні початкових умов, і описану процедуру слід повторити ще раз. Якщо алгоритм не може визначити нову початкову точку, то пошук зображення виходить за межі можливостей алгоритму в рамках заданих конкретних налаштувань. У цьому випадку алгоритм слід розширити, швидше за все шляхом розширення базису H.

На наступному етапі статистичний аналіз виконується за допомогою аналізу положень точок, де спостерігається найбільше відхилення ідеалізованого зображення від експериментального. Правило аналізу базується на тій самій передумові, що й використана статистична модель, і полягає в наступному.

Відзначимо всі точки, де відхилення є більшою p-ою частиною при сортуванні за зростанням. Якщо щільність їх концентрації більше, то це є сильною ознакою появи дефекту. Щільність концентрації оцінюється в області розміром, що покладаємо приблизно рівною періоду періодичності зображення, і шляхом статистичного аналізу можна оцінити ймовірність дефекту в області.

Приклад результату такого аналізу представлений на рисунку 5.8. Кольори представляють розрахункову ймовірність дефекту: червоний > 99 %, жовтий становить від 90 % до 99 %. Приклад результату такого аналізу наведено на рисунку 6.8.

Алгоритм наразі реалізовано за допомогою GNU Octave 9.1.0, хоча можна використовувати версії до 4.0. Для розв'язання задач оптимізації використовується Octave-програма **ralgb5** (Стецюк 2017a), (Стецюк 2014, с. 384–385), яка реалізує *r*-алгоритм з адаптивним кроковим керуванням і постійним коефіцієнтом розтягу простору. Поточна ітерація алгоритму займає близько 100 секунд для завершення на процесорі, який має рейтинг 2401 у програмному забезпеченні PassMark. Слід також зазначити, що розрахунок функції помилки є дуже паралелізованим і може бути значно прискорений за допомогою належного апаратного забезпечення за потреби.

Рисунок 6.7 – Побудоване ідеалізоване зображення

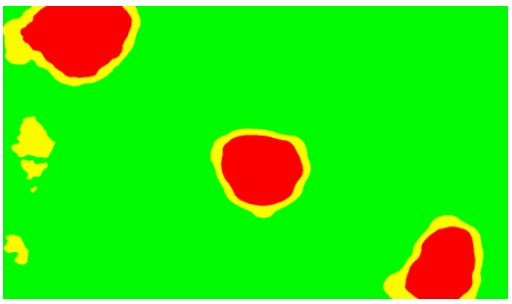


Рисунок 6.8 – Оцінка ймовірності наявності дефекту

Процедура оптимізаційного наближення та аналізу також була апробована на низці зображень у межах проєкту «Ідеатон: Сервісний портал для

автоматизації неруйнівного контролю якості і надання висновків, що засновані на аналітичних алгоритмах для виявлення дефектів у регулярних структурах». Два зображення такого типу наведено на рисунку 6.9.

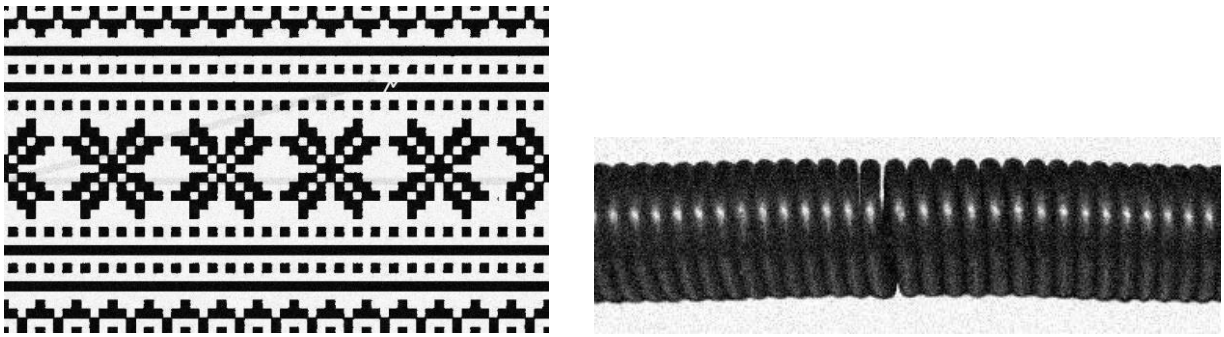


Рисунок 6.9 – Періодичні зображення з дефектами

За результатами роботи запропонованого алгоритму були отримані відновлені зображення, наведені на рисунку 6.10. Результати роботи алгоритму демонструють, що запропонований підхід дозволяє досить ефективно виявляти дефекти на періодичних зображеннях різної (в тому числі високої) складності, причому без використання попередньої підготовки та навчання.

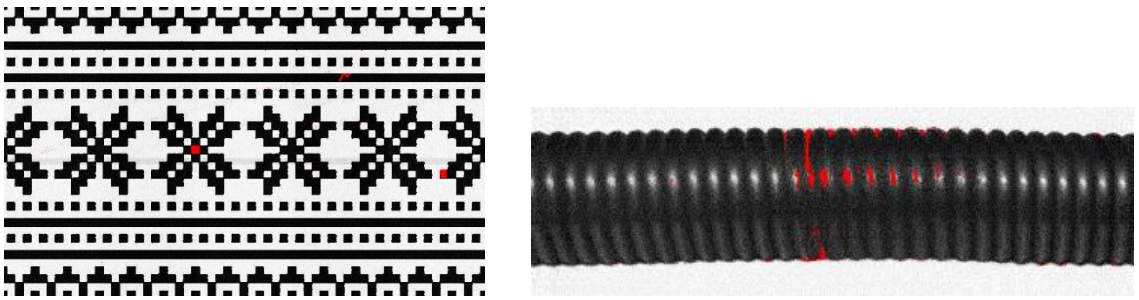


Рисунок 6.10 – Відновлені зображення (дефекти позначено червоним)

6.4 Висновки до шостого розділу

Розроблено новий алгоритм пошуку дефектів у періодичних структурах за допомогою порівняння експериментальних та ідеалізованих зображень. Показано ефективність використання «усіченої» норми для відновлення параметрів ідеалізованої моделі для зображень, отриманих методом шерографії.

Алгоритм досить загальний за своєю природою, легко налаштовується та не потребує навчання. На основі цього алгоритму розроблено програмне забезпечення, яке можна використовувати для пошуку дефектів у періодичних структурах, таких як ті, що властиві стільниковим та композитним панелям тощо, без участі людини-оператора. Ефективність у виявленні дефектів цим засобом не поступається людині-оператору.

Матеріал розділу базується на таких роботах: (Жидков 2017), (Zhydkov 2024), (Zhydkov 2025).

ЗАГАЛЬНІ ВИСНОВКИ

У дисертаційній роботі розроблено та побудовано ряд моделей та методів негладкої оптимізації для побудови кривих у натуральній параметризації та виявлення дефектів різного виду. Ефективність розроблених методів підтверджується результатами обчислювальних експериментів і тестів, проведених в рамках досліджень.

Основі результати дисертаційної роботи наведено нижче.

1. Розроблено Octave-програму **ralgb5a**, яка призначена для мінімізації гладких та негладких опуклих функцій та реалізує модифікацію *r*-алгоритмів з адаптивним способом регулювання кроку. Досліджено метод негладких штрафних функцій для задач опуклого програмування з урахуванням випадку, коли цільова функція визначена не на всьому просторі змінних. Метод негладких штрафних функцій можна використовувати в тому числі і для врахування обмежень, заданих у формі рівностей або нерівностей, в задачах математичного програмування.
2. Розроблено математичну модель, алгоритм та програмне забезпечення для задачі побудови S-подібної кривої в натуральній параметризації, яка проходить через дві задані точки із заданими кутами нахилу дотичних у них та забезпечує заданий кут нахилу дотичної в точці із заданою абсцисою. Сформульовано систему нелінійних інтегральних рівнянь для квадратичної кривини, досліджено її властивості, досліджено відповідну задачу мінімізації негладкої функції та алгоритм її розв’язання. Алгоритм основано на модифікації методу з розтягом простору в напрямі різниці двох послідовних узагальнених градієнтів. Проведено обчислювальні експерименти, які показали ефективність розробленого алгоритму для проектування фрагментів дозвукової та надзвукової частин зовнішнього контуру сопла Франкля.
3. Досліджено геометричну постановку задачі побудови кривої в

натуральній параметризації з кубічною кривиною та сформульовано відповідну систему нелінійних інтегральних рівнянь (CHIP). Вперше сформульовано оптимізаційну задачу для знаходження розв'язку CHIP та метод її розв'язання з використанням модифікації *r*-алгоритму Шора. Досліджено питання масштабування для оптимізаційної задачі побудови кривої у натуральній параметризації з кубічним розподілом кривини, що проходить через дві задані точки та забезпечує в них задані кути нахилу дотичних та задані значення кривин.

4. Для регулярних та базисних регулярних 3D-структур вперше сформульовано оптимізаційні задачі знаходження найкращих за L_p-нормою параметрів та досліджено методи розв'язування розглянутих задач. Показано, що при відновленні параметрів 3D-структур з дефектами метод найменших модулів є стійкішим, ніж метод найменших квадратів. Наведено результати обчислювальних експериментів для програмних реалізацій методів на основі *r*-алгоритму, які дозволяють оцінити їхні часові затрати при знаходженні параметрів регулярних 3D-структур великих розмірів (400 рядків і 600 стовпців та 1000 рядків і 1500 стовпців).
5. Розроблено новий алгоритм пошуку дефектів у періодичних структурах за допомогою порівняння експериментальних та ідеалізованих зображень. Продемонстровано ефективність використання «усіченої» норми для відновлення параметрів ідеалізованої моделі для зображень, отриманих методом шерографії. Алгоритм досить загальний за своєю природою, легко налаштовується і не потребує навчання. Розроблене програмне забезпечення можна використовувати для пошуку дефектів у періодичних структурах, таких як ті, що властиві стільниковим та композитним панелям тощо, без участі людини-оператора.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Войтюк, Дмитро, та Сергій Пилипака. 2001. Конструювання просторової кривої лінії із заданою кривиною, як траєкторії руху матеріальної точки. *Збірник наукових праць НАУ «Механізація сільськогосподарського виробництва»* (10): 74–78.

Борисенко, Валерій, Олексій Агарков, Костянтин Палько, та Максим Палько. 2016. Моделювання плоских кривих у натуральній параметризації. *Геометричне моделювання та інформаційні технології* (1): 21–27.

Борисенко, Валерій, Сергій Устенко, та Ірина Устенко. 2018. «Геометричне моделювання S-подібних скелетних ліній профілів лопаток осьових компресорів». *Вісник двигунобудування* (1):45–52.

Жидков, Владимир. 2017. «Определение дефектов в периодических структурах». *Компьютерная математика* (2):21–29.

Жидков, Володимир, Петро Стецюк, та Ольга Хом’як. 2025. Метод найменших квадратів та метод найменших модулів для пошуку дефектів в регулярних зображеннях. *Cybernetics and Computer Technologies* 1:32–42.

Журбенко, Николай, и Тамара Марчук. 1976. *Алгоритм минимизации негладких функционалов (*$r(\alpha)$*-алгоритм)*. (22). Киев: АН УССР.

Еремин, Иван. 1967. Метод «штрафов» в выпуклом программировании. *Докл. АН СССР* 173(4):748–751.

Крайко, Алла. 2014. «Профилирование сопел и переходных каналов реактивных двигателей». Дис. кандидата фіз.-мат. наук, 01.02.05.

Лаптин, Юрий. 2015. «Методы негладкой оптимизации решения структурированных задач». Дис. доктора фіз.-мат. наук, 01.05.01.

Лаптин, Юрий. 2011. Некоторые вопросы использования негладких штрафных функций. *Теория оптимальных решений* (10):127–135.

Лаптин, Юрий, и Тамара Бардадым. 2011. Некоторые подходы к регуляризации нелинейных задач оптимизации. *Проблемы управления и информатики* (3):57–68.

Лаптин, Юрий, и Николай Журбенко. 2002. Разработка программных средств оптимизации сложных технических объектов. *Теория оптимальных решений* 3–12.

Лаптин, Юрий, Николай Журбенко, Левин Моисей, Волковицкая Полина. 2003. Использование средств оптимизации в системе автоматизированного проектирования энергетических котлоагрегатов КРОКУС. *Энергетика и электрификация* (7):41–51.

Лаптин, Юрий, и Алексей Лиховид. 2010. Использование выпуклых продолжений функций для решения нелинейных задач оптимизации. *Управляющие системы и машины.* (6): 25–31.

Лаптин, Юрий, Медведев Владимир, и Волковицкая Полина. 1994. Система математических моделей расчета и оптимизации конструктивных решений энергетических паровых котлов. *Теория оптимальных решений* 17–22.

Мудров, Владимир, и Валентин Кушко. 1971. *Метод наименьших модулей*. М.: Знание.

Никулин, Михаил. 1984. Отношения правдоподобия критерий. В *Математическая энциклопедия*, Т. 4, Виноградов И.М. (гл. ред.), 151. М.: Советская энциклопедия.

Пилипака, Сергій, та Віктор Несвідомін. 1996. Побудова просторової кривої лінії за заданими натуральними рівняннями. *Прикл. геометрия и инж. графика* (59): 106–107.

Пилипака, Сергій, та Тетяна Гнітецька. 2002. Конструювання просторових кривих, заданих натуральними рівняннями, за допомогою чисельних методів. *Геометричне та комп'ютерне моделювання. Збірник наукових праць, присвячений 75-річчю від дня народження проф. Михайленка В.Є.* (1): 24–26.

Пшеничный, Борис. 1983. *Метод линеаризации*. М.: Наука.

Рашевский, Петр. 1956. *Курс дифференциальной геометрии*. 4-е изд. Москва: ГИТТЛ.

Роботишин, Микола, і Микола Маляр. 2022. «Аналіз деяких методів розв'язання задачі розпізнавання дефектів на зображеннях». *Науковий вісник Ужгородського університету. Серія «Математика і інформатика»* 41(2): 141–150.

Сергиенко, Александр, Василий Семенов, и Александр Собачкин. 2004. *Выбор оптимальных размеров и контура круглого сопла: Учебное пособие*. М.: Изд-во МАИ.

Сергієнко, Іван, Олег Литвин, Олег Литвин, Олександр Ткаченко, та Ольга Грицай. 2016. «Побудова та дослідження операторів ермітової інтерлінації

функцій двох змінних на системі неперетинних ліній із збереженням класу диференційовності». *Проблеми машинобудування* 19(3):60-68.

Сергієнко, Іван, Петро Стецюк, Олег Литвин, Володимир Жидков, та Олег Литвин. 2018. *Розроблення комплексу програм побудови теоретичних контурів зовнішньої і внутрішньої поверхонь сопла з центральним тілом по заданому закону зміни площ* (заключний звіт про науково-дослідну роботу № держ. реєстрації 0118U006687). Київ: Інститут кібернетики імені В.М. Глушкова НАН України.

Сергієнко, Іван, Петро Стецюк, Олег Литвин, Володимир Жидков, та Олег Литвин. 2020. *Розроблення комплексу програм побудови теоретичних контурів зовнішньої і внутрішньої поверхонь сопла з центральним тілом по заданому закону зміни площ (етап2)* (заключний звіт по науково-технічному проєкту). № держ. реєстрації 0119U002303). Київ: Інститут кібернетики імені В.М. Глушкова НАН України.

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

Стецюк, Петр. 2017а. «Субградиентные методы ralgb5 и ralgb4 для минимизации овражных выпуклых функций». *Вычислительные технологии* 22(2):127-149.

Стецюк, Петр. 2017б. «Теория и программные реализации r-алгоритмов Шора». *Кибернетика и системный анализ* (5):43–57.

Стецюк, Петро. 2019. Комп’ютерна програма «Octave-програма ralgb5a: r(α)-алгоритм з адаптивним кроком». Свідоцтво про реєстрацію авторського права на твір № 85010. Україна, Міністерство економічного розвитку і торгівлі, Державний департамент інтелектуальної власності. Дата реєстрації 29.01.2019.

Стецюк, Петро, та Віктор Савицький. 2017. «О робастности метода наименьших модулей для поиска дефектов в регулярных 3D-структурах». Матеріали Дев'ятнадцятого Міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування», присвяченого пам'яті д.ф.-м.н., професора Петренюка Анатолія Яковича, Кропивницький, Квітень 7-8.

Стецюк, Петр, и Виктор Савицкий. 2018. «О поиске дефектов в регулярных 3D-структурах». *Проблемы управления и информатики* (2):33-48.

Стецюк, Петро, Віктор Савицький, та Володимир Жидков. 2023. Пошук дефектів у регулярних 3D-структурах. В *Методи негладкої оптимізації в прикладних задачах*, відповідальні редактори Петро Стецюк та Марія Григорак, 230–257. Київ: ЛАЗУРИТ ПОЛІГРАФ.

Стецюк, Петро, Тамар Бєлих, та Олена Криворучко. 2019. Теорія та програмні реалізації r-алгоритмів Шора. У *Субградієнтні алгоритми та задачі на комбінаторних конфігураціях*, під загал. ред. П.І. Стецюка, 5–33. Київ: Університетське видавництво «Пульсари».

Стецюк, Петро, Володимир Сидорук, Ольга Хом'як та ін. 2019. *Побудова оптимальних профілів аеродинамічних поверхонь з використанням методів негладкої оптимізації* (заключний звіт по науково-технічному проєкту № держ. реєстрації 0119U002305). Київ: Інститут кібернетики імені В.М. Глушкова НАН України.

Стецюк, Петро, Олександр Ткаченко, та Ольга Грицай. 2020. «До побудови зовнішнього контура сопла Франкля за квадратичною кривиною». *Кібернетика та комп'ютерні технології* (1):23-31.

Стецюк, Петро, Олександр Ткаченко, та Катерина Ульянова. 2019. «Використання r-алгоритму для побудови аеродинамічного профілю з кубічною кривиною». Тези доповідей XVII міжнародної науково-практичної конференції «Математичне та програмне забезпечення інтелектуальних систем (MPZIS-2019)», Дніпро, Листопад 20-22.

Стецюк, Петро, Олександр Ткаченко, Ольга Хом'як, та Ольга Грицай. 2020. «Побудова зовнішнього контуру сопла Франкля з використанням S-подібних кривих із квадратичним законом розподілу кривини». *Кібернетика та системний аналіз* (6):120-135.

Стецюк, Петро, Олександр Хіміч, та Володимир Сидорук. 2016. «Реалізація *r*-алгоритму на графічних процесорах». *Комп'ютерна математика* (2):100-109.

Стецюк, Петро, Ольга Хом'як, та Володимир Жидков. 2023а. «Масштабування даних для задачі побудови кривої в натуральній параметризації з кубічною кривиною». *Фізико-математичне моделювання та інформаційні технології* 37:123–127.

Стецюк, Петро, Ольга Хом'як, та Володимир Жидков. 2023б. Побудова S-подібної параметричної кривої та її застосування для проєктування зовнішнього контура сопла. В *Методи негладкої оптимізації в прикладних задачах*, відповідальні редактори Петро Стецюк та Марія Григорак, 258–292. Київ: ЛАЗУРИТ ПОЛІГРАФ.

Стецюк, Петро, Ольга Хом'як, Володимир Жидков та ін. 2023в. *Розробити методи негладкої оптимізації для побудови кривих у натуральній параметризації* (заключний звіт по науково-технічному проєкту № держ.

реєстрації 0121U100459). Київ: Інститут кібернетики імені В.М. Глушкова НАН України.

Устенко, Сергій. 2009. Геометричне моделювання плоских кривих із застосуванням елементів кривини. *Геометричне та комп'ютерне моделювання* (22): 82–87.

Устенко, Сергій. 2013. *Геометрична теорія моделювання криволінійних форм лопаткових апаратів турбомашин з оптимізацією їх параметрів*. Дис. д-ра техн. наук, Київ. 349 с.

Фундитус, Сергій, та Ігор Коноваленко. 2024. «Розпізнавання дефектів на поверхневій структурі покрівлі будівлі за допомогою дронів та згорткових нейронних мереж». Матеріали І Міжнародної науково-технічної конференції «Прикладна механіка», Тернопіль, Україна, Червень 6-7.

Шор, Наум. 1977. Метод отсечения с растяжением пространства для решения задач выпуклого программирования. *Кибернетика* (1):94–95.

Шор, Наум. 1979. *Методы минимизации недифференцируемых функций и их приложения*. Киев: Наукова думка.

Шор, Наум, и Николай Журбенко. 1971. Метод минимизации, использующий операцию растяжения пространства в направлении разности двух последовательных градиентов. *Кибернетика* (3):51–59.

Шор, Наум, и Сергей Стеценко. 1989. *Квадратичные экстремальные задачи и недифференцируемая оптимизаци.* Киев: Наукова думка.

Шор, Наум, и Петр Стецюк. 1997. «Использование модификации r-алгоритма для нахождения глобального минимума полиномиальных функций». *Кибернетика и системный анализ* (4):28–49.

Allen, Bob. 2017. “NACA Airfoils”. *NASA.gov*. NASA. Accessed July 27, https://www.nasa.gov/image-article/naca-airfoils/

Ashrafi, Sara, Sobhan Teymouri, Sepideh Etaati, Javad Khoramdel, Yasamin Borhani, and Esmaeil Najafi. 2025. “Steel Surface Defect Detection and Segmentation Using Deep Neural Networks”. *Results in Engineering* (25): 103972.

Ben Gharsallah, Mohamed, and Ezzedine Ben Braiek. 2015. “Image Segmentation for Defect Detection Based on Level Set Active Contour Combined with Saliency Map”. Proceedings of 16th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), Monastir, Tunisia, December 21-23.

Bertsekas, Dimitri. 1985. Necessary and sufficient conditions for a penalty method to be exact. *Math. Program.* 9(1):87–99.

Borisenko, Valeriy, Serhiy Ustenko, Iryna Ustenko, and Kateryna Kuzma. 2019. “Development of a Method for Geometrical Modeling of the Airfoil Profile of an Axial Turbomachine Blade”. *Eastern-European Journal of Enterprise Technologies* 5(1(101)): 29–38.

Branca, Antonella, W. Delaney, Francesco Lovergine, and Arcangelo Distante. 1995. “Surface Defect Detection by a Texture Analysis with a Neural Network”. Proceedings of 1995 IEEE International Conference on Robotics and Automation, Nagoya, Japan, May 21-27. 1497–1502.

Chen, Jin, Quan Wang, Xiaoping Pang, Songlin Li, and Xiaofeng Guo. 2013. "Improvement of Airfoil Design Using Smooth Curvature Technique". *Renewable Energy* 51: 426–435.

CPLEX Optimizer. High-performance mathematical programming solver for linear programming, mixed-integer programming and quadratic programming. https://www.ibm.com/analytics/cplex-optimizer (accessed: 17.02.2025)

Fakhari, Seyyed Mojtaba, and Hatem Mrad. 2024. "Aerodynamic Shape Optimization of NACA Airfoils Based on a Novel Unconstrained Conjugate Gradient Algorithm". *Journal of Engineering Research*.

Fazil, Jani, and Vijayarangan Jayakumar. 2011. "Investigation of Airfoil Profile Design Using Reverse Engineering Bezier Curve". *ARPN Journal of Engineering and Applied Sciences* 6(7): 43–52.

Fischer, Andreas, Olga Khomyak, and Petro Stetsyuk. 2023. The ellipsoid method and computational aspects. *Commun. Optim. Theory* (21):1–14.

Ganesh Ram, R.K. 2014. "Design Optimization and Analysis of NACA 0012 Airfoil Using Computational Fluid Dynamics and Genetic Algorithm". *Applied Mechanics and Materials* (664): 111–116.

Ghosh, Pallabi, and Rajat Subhra Chakraborty. 2017. "Counterfeit IC Detection By Image Texture Analysis". Proceedings of 2017 Euromicro Conference on Digital System Design (DSD), Vienna, Austria, August 30 – September 1.

GNU Octave https://octave.org/ (accessed: 17.02.2025).

Gurobi Optimization, Inc., Gurobi Optimizer Reference Manual, 2014. http://www.gurobi.com/ (accessed: 17.02.2025).

Han, Soonhung, Yeon-Seung Lee, and Young Bok Choi. 2012. “Hydrodynamic Hull Form Optimization Using Parametric Models”. *Journal of Marine Science and Technology* 17: 1–17.

Hu, Guang-Hua, Qing-Hui Wang, and Guo-Hui Zhang. 2015. “Unsupervised Defect Detection in Textiles Based on Fourier Analysis and Wavelet Shrinkage”. *Applied Optics* (54): 2963–2980.

Ipopt Documentation. https://coin-or.github.io/Ipopt/ (accessed: 17.02.2025).

Ivanchuk, Oleh, and Olha Tumska. 2020. “A Study of Methods for Texture Classification of SEM Images of Micro-surfaces of Objects and Their Segmentation”. *ISTCGCAP* (91): 41–50.

Iyer, Manimozhi, and S. Janakiraman Subbaih. 2014. “Defect Detection in Pattern Texture Analysis”. Proceedings of 2014 International Conference on Communication and Signal Processing, Melmaruvathur, India, April 3-5.

Izli, Nazmi, Ali Vardar, and Ferhat Kurtulmus. 2007. “A Study on Aerodynamic Properties of Some NACA Profiles Used on Wind Turbine Blades”. *Journal of Applied Sciences* 7 (3): 426–433.

Jaiswal, Shikhar A. 2017. “Shape Parameterization of Airfoil Shapes Using Bezier Curves”. In *Innovative Design and Development Practices in Aerospace and Automotive Engineering*, edited by R.P. Bajpai and U. Chandrasekhar, 79–85. Singapore: Springer.

Kallath, Hariharan, Jun Seok Lee, Foster Kwame Kholi, Man Yeong Ha, and June Kee Min. 2021. “A Multi-Objective Airfoil Shape Optimization Study Using Mesh Morphing and Response Surface Method”. *Journal of Mechanical Science and Technology* 35: 1075–1086.

Khomiak, Olga, Petro Stetsyuk, Volodymyr Zhydkov, and Luis Infante. 2023. “Using Optimization to Construct Naturally Parametrized Curve with Cubic Curvature”. In *Smart Technologies in Urban Engineering (STUE 2022)*, Romanova T., Sukhonos M., Tsegelnyk Y. (eds). *Lecture Notes in Networks and Systems* 536:14–24.

Körpe, Durmuş Sinan, and Ibrahim Halil Güzelbey. 2023. “NACA Four-Digit Airfoil Series Optimization: A Comparison between Genetic Algorithm and Sequential Quadratic Programming”. *Journal of Mechanical Science and Technology* (37): 2375–2382.

Kumar, Ajar and Helen C. Shen. 2002. “Texture Inspection for Defects Using Neural Networks and Support Vector Machines”. Proceedings of 2002 International Conference on Image Processing ICIP, Rochester, NY, USA, September 22-25.

Kunttu, Iivari, Leena Lepistö, Juhani Rauhamaa, and Ari Visa. 2006. “Multiscale Fourier Descriptors for Defect Image Retrieval”. *Pattern Recognition Letters* 27(2): 123–132.

Lobanov, Leonid, Vyacheslav Pivtorak, Inna Kyjanets, Viktor Savitsky, and Galina Tkachuk. 2005. “Express control of quality and stressed state of welded structures using method of electron shearography and speckle-interferometry”. *The Paton Welding Journal* :35–40.

Mathew, Bilji C., Amit Thakan, and J. V. Muruga Lal Jeyan. 2020. “A Review on Aerodynamic Performance of NACA Airfoil for Various Reynolds Number”. *Journal of Physics: Conference Series* 1473.

Meister, Sebastian, Mahdieu A. M. Wermes, Jan Stüve, and Roger M. Groves. 2021. "Review of Image Segmentation Techniques for Layup Defect Detection in the Automated Fiber Placement Process". *Journal of Intelligent Manufacturing* (32): 2099–2119.

Mirmahdavi, Seyyed Abdolah, Alireza Ahmadyfard, Abdollah Amirkhani Shahraki, and Parham Khojasteh. 2013. "A Novel Modeling of Random Textures Using Fourier Transform for Defect Detection". Proceedings of 2013 UKSim 15th International Conference on Computer Modelling and Simulation, Cambridge, UK, April 10-12.

Nemirovsky, Arkadi, and David Yudin. 1983. *Problem Complexity and Method Efficiency in Optimization*. John Wiley. New York.

NEOS Solver. https://neos-server.org/ (accessed: 17.02.2025)

Paul, Omari, Sakib Abrar, Richard Mu, Riadul Islam, and Manar D. Samad. 2023. "Deep Image Segmentation for Defect Detection in Photo-Lithography Fabrication". Proceedings of 24th International Symposium on Quality Electronic Design (ISQED), San Francisco, CA, USA, April 5-7.

Peng, Zhijun, Baiman Chen, Frank G.F. Qin, Runhua Jiang, Qin He, Shi Tao, Hanmin Xiao, Ying Chen, and Minlin Yang. 2018. "Numerical Simulation of Aerodynamic Performance of an Airfoil Combined Lift and Drag". Proceedings of the 10th International Conference on Applied Energy (ICAE2018), Hong Kong, China, August 22–25.

Perez, Husein, Joseph H. M. Tah, and Amir Mosavi. 2019. "Deep Learning for Detecting Building Defects Using Convolutional Neural Network." *Sensors* (19): 3556.

Polyakova, Lyudmila. 2000. Nonsmooth Penalty Functions. *IFAC Proceedings Volumes* 33(16):287–291.

Rajnarayan, Dev, Andrew Ning, and Judd Mehr. 2018. “Universal Airfoil Parametrization Using B-Splines”. *Faculty Publications,* Paper 2118.

Rezaei, Mohammad A., and Dapeng Zhan. 2021. *Higher Moments of the Natural Parameterization for SLE Curves*. Michigan State University.

Shen, Xiang, Eldad Avital, Mohammad Amin Rezaienia, Gordon Paul, and Theodosios Korakianitis. 2016. “Computational Methods for Investigation of Surface Curvature Effects on Airfoil Boundary Layer Behavior”. *Journal of Algorithms & Computational Technology* 11(1): 68–82.

Shor, Naum. 1985. *Minimization Methods for Non-Differentiable Functions*. Berlin: Springer-Verlag.

Shor, Naum. 1998. *Nondifferentiable Optimization and Polynomial Problems*. Amsterdam. Kluwer.

Si, Jongwook, and Sungyoung Kim. 2024. “V-DAFT: Visual Technique for Texture Image Defect Recognition with Denoising Autoencoder and Fourier Transform”. *Signal, Image and Video Processing* 18: 7405–7418.

Sivabalan, K.N., and D. Ghanadurai. 2010. “Detection of Defects in Digital Texture Images Using Segmentation”. Inte*rnational Journal of Engineering Science and Technology* 2(10): 5187–5191.

Smagulova, Damira, Vykintas Samaitis, and Elena Jasiuniene. 2024. “Convolutional Neural Network for Interface Defect Detection in Adhesively Bonded Dissimilar Structures”. *Applied Sciences* 14 (22): 10351.

SNOPT (Sparse Nonlinear OPTimizer) https://ccom.ucsd.edu/~optimizers/solvers/snopt/ (accessed: 17.02.2025)

Stetsyuk, Petro. 2017. Shor’s r-Algorithms: Theory and Practice. In *Optimization Methods and Applications: In Honor of the 80th Birthday of Ivan V. Sergienko*, ed. by Butenko S., Pardalos P.M, Shylo V. Springer, 495–520. Springer, Cham.

Stetsyuk, Petro, Viktor Savitsky, and Volodymyr Zhydkov. 2019. “Optimization Problems for Regular Image Reconstruction”. Book of Absracts of International Conference on Optimization and Equilibrium Problems, Dresden, Germany, July 31 – August 2.

Stetsyuk, Petro, Oleksandr Tkachenko, and Volodymyr Zhydkov. 2020. “Using Shor’s r-algorithm for building naturally parametrized curve having cubic curvature”. Proceedings of the 7th International Conference on Control and Optimization with Industrial Applications (COIA 2020), Baku, Azerbaijan, August 26-28.

Sun, Zhaocheng, Zengliang Li, and Menghao Fan. 2020. “Airfoil Shape Optimization Based on Non-Uniform Rational B-Spline and Optimization Algorithm”. *IOP Conference Series: Earth and Environmental Science* 474: 052075.

Tanabi, Naser, Agesinaldo Matos Silva Jr., Marcosiris Amorim Oliveira Pessoa, and Marcos Sales Guerra Tsuzuki. 2023. “Robust Algorithm Software for NACA 4-Digit Airfoil Shape Optimization Using the Adjoint Method”. *Applied Sciences* (13): 4269. https://doi.org/10.3390/app13074269.

Ümütlü, Hatice Cansu Ayaz, and Zeki Kiral. 2022. “Airfoil Shape Optimization Using Bézier Curve and Genetic Algorithm”. *Aviation* 26(1): 32–40.

Villard, Pierre-Frédéric, Maureen Boudart, Ioana Ilea, and Fabien Pierre. 2022. “Anomaly Detection in Textured Images with a Convolutional Neural Network for Quality Control of Micrometric Woven Meshes”. *Fluid Dynamics and Materials Processing* 18(6): 1639–1648.

Vucina, Damir, Zeljan Lozina, and Igor Pehnec. 2008. “A Compact Parameterization for Shape Optimization of Aerofoils”. Proceedings of the World Congress on Engineering, Vol. I, London, UK, July 2–4.

Wang, Fu-liang, and Bo Zuo. 2016. “Detection of Surface Cutting Defect on Magnet Using Fourier Image Reconstruction”. *Journal of Central South University* (23): 1123–1131.

Wu, Hao, and Zhi Zhou. 2021. “Using Convolution Neural Network for Defective Image Classification of Industrial Components”. *Mobile Information Systems*.

Zhydkov, Volodymyr. 2024. “Finding Defects in Periodic Structures”. Матеріали XXIV міжнародної науково-практичної конференції «Інформаційні технології та безпека (ІТБ-2024)», Київ, Україна, Грудень 19.

Zhydkov, Volodymyr. 2025. Revealing defects in periodic structures. *Problems of Control and Informatics* 70(1):22–31.

Zhydkov, Volodymyr, and Olha Khomiak. 2025. “Coordinate scaling for making naturally parametrized curve with cubic curvature”. Proceeding of the International Conference on Management and Control in Solving Engineer Problems (MaCoSEP 2025), Baku, Azerbaijan, March 13-15.

ДОДАДОК А.
СПИСОК ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДИСЕРТАЦІЇ ТА ВІДОМОСТІ ПРО АПРОБАЦІЮ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЇ

Публікації, в яких опубліковано
основні наукові результати дисертації:

1. Жидков, Владимир. 2017. «Определение дефектов в периодических структурах». *Компьютерная математика* 2:21–29.
2. Стецюк, Петро, Ольга Хом'як, та Володимир Жидков. 2023а. «Масштабування даних для задачі побудови кривої в натуральній параметризації з кубічною кривиною». *Фізико-математичне моделювання та інформаційні технології* 37:123–127.
   DOI: 10.15407/fmmit2023.37.123
3. Стецюк, Петро, Ольга Хом'як, та Володимир Жидков. 2023б. Побудова S-подібної параметричної кривої та її застосування для проєктування зовнішнього контура сопла. В *Методи негладкої оптимізації в прикладних задачах*, відповідальні редактори Петро Стецюк та Марія Григорак, 258–292. Київ: ЛАЗУРИТ ПОЛІГРАФ.
4. Стецюк, Петро, Віктор Савицький, та Володимир Жидков. 2023. Пошук дефектів у регулярних 3D-структурах. В *Методи негладкої оптимізації в прикладних задачах*, відповідальні редактори Петро Стецюк та Марія Григорак, 230–257. Київ: ЛАЗУРИТ ПОЛІГРАФ.
5. Жидков, Володимир, Петро Стецюк, та Ольга Хом'як. 2025. Метод найменших квадратів та метод найменших модулів для пошуку дефектів в регулярних зображеннях. *Cybernetics and Computer Technologies* 1:32–42.
   DOI: 10.34229/2707-451X.25.1.3
6. Khomiak, Olga, Petro Stetsyuk, Volodymyr Zhydkov, and Luis Infante. 2023. "Using Optimization to Construct Naturally Parametrized Curve with Cubic Curvature". In *Smart Technologies in Urban Engineering (STUE 2022)*, Romanova T., Sukhonos M., Tsegelnyk Y. (eds). *Lecture Notes in Networks and Systems* 536:14–24.
   DOI: 10.1007/978-3-031-20141-7_2

7. Zhydkov, Volodymyr. 2025. Revealing defects in periodic structures. *Problems of Control and Informatics* 70(1):22–31.
DOI: 10.34229/1028-0979-2025-1-2

Публікації, що засвідчують
апробацію матеріалів дисертації

1. Stetsyuk, Petro, Oleksandr Tkachenko, and Volodymyr Zhydkov. 2020. “Using Shor’s r-algorithm for building naturally parametrized curve having cubic curvature”. Proceedings of the 7-th International Conference on Control and Optimization with Industrial Application (COIA 2020), Baku, Azerbaijan, August 26-28. Vol. I. 389–391.
http://www.coia-conf.org/upload/editor/files/COIA2020_V1.pdf
2. Zhydkov, Volodymyr. 2024. “Finding Defects in Periodic Structures”. Матеріали XXIV міжнародної науково-практичної конференції «Інформаційні технології та безпека (ІТБ-2024)», Київ, Україна, Грудень 19. 38–41.
https://drive.google.com/file/d/1wsLVnueNRd62g-k179IHihuJNOYZqR9G/view
3. Zhydkov, Volodymyr, and Olha Khomiak. 2025. “Coordinate scaling for making naturally parametrized curve with cubic curvature”. Proceeding of the International Conference on Management and Control in Solving Engineer Problems (MaCoSEP 2025), Baku, Azerbaijan, March 13-15.
4. Stetsyuk, Petro, Viktor Savitsky, and Volodymyr Zhydkov. 2019. “Optimization Problems for Regular Image Reconstruction”. Book of Absracts of International Conference on Optimization and Equilibrium Problems, Dresden, Germany, July 31 – August 2. 45–46.

Відомості про апробацію результатів дисертації

Результати доповідались та обговорювались на:

- 7-th International Conference on Control and Optimization with Industrial Application (COIA 2020), 26–28 серпня 2020 року, Баку, Азербайджан;
- XXIV Міжнародній науково-практичній конференції «Інформаційні технології та безпека (ІТБ-2024)», 19 грудня 2024 року, Київ, Україна;
- International Conference on Management and Control in Solving Engineer Problems (MaCoSEP 2025), 13–15 березня 2025 року, Баку, Азербайджан;
- International Conference “Optimization and Equilibrium Problems (ICOEP 2019)”, 31 липня – 2 серпня 2019 року, Дрезден, Німеччина;
- засіданні відділу методів негладкої оптимізації Інституту кібернетики імені В.М. Глушкова НАН України, 25 квітня 2025 року.