НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.42:519.237

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
«___»_____2025 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-науковою програмою «Інженерія програмного забезпечення комп'ютерних та інформаційних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Методологія та програмні засоби підвищення ефективності синтетичного методу побудови багатовимірних поліноміальних регресій»

Виконала:
студентка ІІ курсу, групи ІП-31 мн
Дрозд Валерія Валеріївна _____

Керівник:
заслужений діяч науки та техніки України,
професор, д.т.н., професор
Павлов Олександр Анатолійович _____

Рецензент:
доцент кафедри ІСТ, к.т.н., доцент
Жураковська Оксана Сергіївна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2025 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-наукова програма «Інженерія програмного забезпечення комп'ютерних та інформаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Едуард ЖАРІКОВ

«___»_____2025 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Дрозд Валерії Валеріївні

1. Тема дисертації «Методологія та програмні засоби підвищення ефективності синтетичного методу побудови багатовимірних поліноміальних регресій», науковий керівник дисертації Павлов Олександр Анатолійович, д.т.н., професор, затверджені наказом по університету від «13» березня 2025 р. № 1128-с.

2. Термін подання студентом дисертації «16» травня 2025 р.

3. Об'єкт дослідження – програмне забезпечення та методи побудови багатовимірних поліноміальних регресій.

4. Предмет дослідження – методологія та програмні засоби підвищення ефективності програмного забезпечення синтетичного методу побудови багатовимірних поліноміальних регресій, заданих надлишковим описом.

5. Перелік завдань, які потрібно розробити – дослідити теоретичні властивості декомпозиційного методу, що входить до складу синтетичного методу, та забезпечити можливість підвищення його ефективності; створити підхід, який дозволяє підвищити ефективність модифікованого методу групового урахування аргументів, що входить до складу синтетичного методу; обґрунтувати можливість використання алгоритму статичного управління паралельними асинхронними обчислювальними процесами для покращення швидкодії паралельної реалізації

модифікованого методу групового урахування аргументів; дослідити способи розширення можливостей мов програмування загального призначення та універсальних програмних засобів для автоматизації процесу створення і програмної реалізації індивідуальних алгоритмів побудови багатовимірних поліноміальних регресій; на основі проведеного аналізу розробити методологію розширення можливостей обраної мови програмування загального призначення; модифікувати архітектуру програмної бібліотеки, яка реалізує синтетичний метод побудови багатовимірних поліноміальних регресій; виконати статистичне дослідження ефективності нової версії модифікованого методу групового урахування аргументів; виконати дослідження ефективності в цілому програмної бібліотеки побудови багатовимірних поліноміальних регресій.

6. Орієнтовний перелік публікацій – одна стаття включена до переліку Scopus Q4; шість статей у наукових виданнях категорії Б; шість тез доповідей на Всеукраїнських та Міжнародних наукових конференціях.

7. Дата видачі завдання «5» лютого 2025 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Ознайомлення з предметною областю, вивчення літератури	05.02.2025	
2	Дослідження теоретичних властивостей синтетичного методу	28.02.2025	
3	Розробка теоретичних положень підвищення ефективності синтетичного методу	12.03.2025	
4	Дослідження способів розширення мов програмування та універсальних програмних засобів	19.03.2025	
5	Розробка методології розширення можливостей мови програмування загального призначення	15.04.2025	
6	Проєктування програмного забезпечення	26.04.2025	
7	Розробка програмного забезпечення	01.05.2025	
8	Виконання експериментальних досліджень	05.05.2025	
9	Оформлення пояснювальної записки	06.05.2025	
10	Подання дисертації на попередній захист	10.05.2025	
11	Подання дисертації на захист	16.05.2025	

Студентка Валерія ДРОЗД

Науковий керівник Олександр ПАВЛОВ

РЕФЕРАТ

Розмір пояснювальної записки – 178 аркушів, містить 28 ілюстрацій, 11 таблиць, 3 додатки, 43 посилання на джерела.

Актуальність теми. Регресійний аналіз є одним з найбільш поширених універсальних засобів виявлення прихованих взаємозв'язків у даних, але побудова правильної структури регресійних моделей та реалізація алгоритмів регресійного аналізу в програмному забезпеченні – досить трудомісткі задачі. На сучасному рівні ефективний розв'язок цих задач методологічно реалізується одночасним використанням логічно зв'язаних різнопланових алгоритмів для створення індивідуального алгоритму розв'язання конкретної задачі регресії. Тому використання вищеописаного підходу для підвищення ефективності синтетичного методу побудови багатовимірних поліноміальних регресій є актуальним.

Мета дослідження. Метою дисертаційної роботи є підвищення ефективності програмного забезпечення синтетичного методу побудови багатовимірних поліноміальних регресій, заданих надлишковим описом, шляхом розробки нових компонентів програмного забезпечення, що реалізують теоретично обґрунтовану методологію створення індивідуальних алгоритмів декомпозиційного методу, та підвищення ефективності модифікованого методу групового урахування аргументів, завдяки одночасному використанню низки регулярних критеріїв.

Об'єкт дослідження: програмне забезпечення та методи побудови багатовимірних поліноміальних регресій.

Предмет дослідження: методологія та програмні засоби підвищення ефективності програмного забезпечення синтетичного методу побудови багатовимірних поліноміальних регресій, заданих надлишковим описом.

Для реалізації поставленої мети **сформульовані наступні завдання:**

– дослідити теоретичні властивості декомпозиційного методу, що входить до складу синтетичного методу, та забезпечити можливість підвищення його ефективності, шляхом розробки нової теоретично обґрунтованої методології

створення індивідуальних алгоритмів побудови багатовимірних поліноміальних регресій, заданих надлишковим описом;

– створити підхід, який дозволяє підвищити ефективність модифікованого методу групового урахування аргументів, що входить до складу синтетичного методу, шляхом одночасного використання низки регулярних критеріїв;

– обґрунтувати можливість використання алгоритму статичного управління паралельними асинхронними обчислювальними процесами для покращення швидкодії паралельної реалізації модифікованого методу групового урахування аргументів, шляхом знаходження асимптотичної складності його підалгоритмів;

– дослідити способи розширення можливостей мов програмування загального призначення та універсальних програмних засобів для автоматизації процесу створення і програмної реалізації індивідуальних алгоритмів побудови багатовимірних поліноміальних регресій, заданих надлишковим описом, на основі положень синтетичного методу та запропонованих нових теоретично обґрунтованих методологій;

– на основі проведеного аналізу розробити методологію розширення можливостей обраної мови програмування загального призначення для автоматизації процесу створення та програмної реалізації індивідуальних алгоритмів побудови багатовимірних поліноміальних регресій, заданих надлишковим описом;

– модифікувати архітектуру програмної бібліотеки, яка реалізує синтетичний метод побудови багатовимірних поліноміальних регресій, заданих надлишковим описом, шляхом включення в себе нових складових, що забезпечать підвищення ефективності синтетичного методу;

– виконати статистичне дослідження ефективності нової версії модифікованого методу групового урахування аргументів;

– виконати дослідження ефективності в цілому програмної бібліотеки побудови багатовимірних поліноміальних регресій, заданих надлишковим описом.

Наукова новизна:

– вперше запропонована нова теоретично обґрунтована методологія створення індивідуальних алгоритмів декомпозиційного методу побудови багатовимірних поліноміальних регресій, заданих надлишковим описом;

– вперше розроблено методологію підвищення ефективності модифікованого методу групового урахування аргументів побудови багатовимірних поліноміальних регресій, заданих надлишковим описом, шляхом використання асиметричного та симетричного регулярних критеріїв, що враховують специфіку сформульованої задачі регресії;

– вперше розроблено методологію розширення мови програмування Python, що забезпечує автоматизацію процесу створення та програмної реалізації індивідуальних алгоритмів синтетичного методу побудови багатовимірних поліноміальних регресій, заданих надлишковим описом;

– обґрунтовано модифікацію архітектури програмної бібліотеки реалізації синтетичного методу, яка включає в себе нові складові, що дозволяють підвищити ефективність методу, шляхом автоматизованої інтерактивної реалізації індивідуальних алгоритмів побудови багатовимірних поліноміальних регресій, заданих надлишковим описом.

Практичне значення отриманих результатів полягає в тому, що нова версія програмної бібліотеки побудови багатовимірних поліноміальних регресій, заданих надлишковим описом, суттєво підвищує ефективність її використання для користувачів, що володіють базовими навичками в області програмування та регресійного аналізу, і дозволяє за необхідності вбудовувати її у цільовий застосунок.

Зв’язок з науковими програмами, планами, темами. Робота виконувалась на кафедрі інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

Апробація. Наукові положення дисертації пройшли апробацію на: Восьмій Міжнародній науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології (SoftTech

– 2025)» – м. Київ, Тринадцятій Міжнародній науково-практичній конференції «Комплексне забезпечення якості технологічних процесів та систем» (КЗЯТПС – 2023) – м. Чернігів; П’ятій Міжнародній науково-практичній конференції «Інженерія програмного забезпечення і передові інформаційні технології (SoftTech – 2023)» – м. Київ; Третій Всеукраїнській науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech – 2022 Осінь) – м. Київ; Першій Всеукраїнській науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech – 2021) – м. Київ.

Публікації. Наукові положення дисертації опубліковані в:

1) Pavlov O., Holovchenko M., Mukha I., Lishchuk K., Drozd V. A Modified Method and an Architecture of a Software for a Multivariate Polynomial Regression Building Based on the Results of a Conditional Active Experiment. In: Hu Z., Dychka I., He M. (eds) Advances in Computer Science for Engineering and Education VI. ICCSEEA 2023. Lecture Notes on Data Engineering and Communications Technologies, Vol. 181, pp. 207-222. Springer, Cham (2023) // *The Sixth International Conference on Computer Science, Engineering and Education Applications ICCSEEA 2023, 16 April 2023, Warsaw, Poland*. – 2023, LNDECT Vol. 181, pp. 207-222. DOI: https://doi.org/10.1007/978-3-031-36118-0_19 [Scopus].

2) Павлов О. А., Головченко М. М., Дрозд В. В., Шаргородський В. С. Підвищення ефективності модифікованого методу урахування аргументів для побудови багатовимірних регресій, заданих надлишковим описом // *Міжвідомчий науково-технічний журнал «Адаптивні системи автоматичного управління»*. К.: КПІ ім. Ігоря Сікорського, 2025. Том 1. № 46.

3) Pavlov O. A., Holovchenko M. M., Drozd V. V. Modification of the Decomposition Method of Constructing Multivariate Polynomial Regression which is Linear with Respect to Unknown Coefficients *// Bulletin of National Technical University «KhPI». Series: System analysis, control and information technologies*, – № 2 (12), 2024. – С. 3-10. DOI: https://doi.org/10.20998/2079-0023.2024.02.01.

4) Pavlov O. A., Holovchenko M. M., Drozd V. V. An Adaptive Method for Building a Multivariate Regression // *Bulletin of National Technical University «KhPI». Series: System analysis, control and information technologies*, – № 1 (11), 2024. – С. 3-8. DOI: https://doi.org/10.20998/2079-0023.2024.01.01.

5) Павлов О. А., Халус О. А., Мельников О. В., Дрозд В. В., Кобельський В. В., Медведєв М. Є. Статичні алгоритми управління паралельними асинхронними обчислювальними процесами // *Міжвідомчий науково-технічний журнал «Адаптивні системи автоматичного управління»*. К.: КПІ ім. Ігоря Сікорського, 2024. Том 1. № 44. С. 116-126. DOI: https://doi.org/10.20535/1560-8956.44.2024.302427.

6) Pavlov O. A., Holovchenko M. M., Drozd V. V. Efficiency Substantiation for a Synthetical Method of Constructing a Multivariate Polynomial Regression, Given by a Redundant Representation // *Bulletin of National Technical University «KhPI». Series: System analysis, control and information technologies*, – № 1 (9), 2023. – С. 3-9. DOI: https://doi.org/10.20998/2079-0023.2023.01.01.

7) Pavlov O. A., Holovchenko M. M., Drozd V. V. Construction of a Multivariate Polynomial, Given by a Redundant Description, in Stochastic and Deterministic Formulations Using an Active Experiment // *Bulletin of National Technical University «KhPI». Series: System analysis, control and information technologies*, – № 1 (7), 2022. – С. 3-8. DOI: https://doi.org/10.20998/2079-0023.2022.01.01.

8) Drozd V. V., Holovchenko M. M., Pavlov O. A. Methodology and Software Tools for Enhancing the Efficiency of the Synthetic Method in Multivariate Polynomial Regression Construction // *Восьма Міжнародна науково-практична конференція «Інженерія програмного забезпечення і передові інформаційні технології (SoftTech – 2025)»*, Київ, 13-15 травня 2025 р.: Кафедра ІПІ ФІОТ, КПІ ім. Ігоря Сікорського, 2025 – С. 10-14.

9) Павлов О. А., Головченко М. М., Дрозд В. В. Синтетичний метод побудови багатовимірної поліноміальної регресії // *Комплексне забезпечення якості технологічних процесів та систем (КЗЯТПС – 2023): матеріали тез*

доповідей XIII Міжнарод. наук.-практ. конф. (м. Чернігів, 25-26 травня 2023 р.): у 2 т. Т. 2. – Чернігів: НУ «Чернігівська політехніка», 2023. – С. 272-273.

10) Троцюк П. С., Дрозд В. В., Головченко М. М., Павлов О. А. Програмне забезпечення для статистичного дослідження ефективності методу побудови багатовимірної лінійної регресії, заданої надлишковим описом // *П'ята Міжнародна науково-практична конференція «Інженерія програмного забезпечення і передові інформаційні технології (SoftTech – 2023)»*, Київ, 19-21 грудня 2023 р.: Кафедра ІПІ ФІОТ, КПІ ім. Ігоря Сікорського, 2023. – С. 327-332.

11) Павлов О. А., Головченко М. М., Дрозд В. В., Ревич М. М. Дослідження ефективності методу побудови багатовимірної лінійної регресії, заданої надлишковим описом // *Третя Всеукраїнська науково-практична конференція молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech – 2022 Осінь)*, Київ: Кафедра ІПІ ФІОТ, КПІ ім. Ігоря Сікорського, 2022. – С. 10-13.

12) Дрозд В. В., Головченко М. М. Методи та програмні засоби побудови нелінійних поліноміальних регресій з використанням нормованих ортогональних поліномів Форсайта // *Перша Всеукраїнська науково-практична конференція молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech – 2021)*. Секція кафедри інформатики та програмної інженерії. Матеріали конференції. – Київ. – 2021. 22-26 листопада 2021 р. – С. 82-86.

Ключові слова: БАГАТОВИМІРНА ПОЛІНОМІАЛЬНА РЕГРЕСІЯ, МОВИ ПРОГРАМУВАННЯ, РОЗШИРЕННЯ ДЛЯ МОВИ ПРОГРАМУВАННЯ, ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ, ЛЮДИНО-КОМП'ЮТЕРНА ВЗАЄМОДІЯ, СИНТЕТИЧНИЙ МЕТОД, ДЕКОМПОЗИЦІЙНИЙ МЕТОД, МОДИФІКОВАНИЙ МЕТОД ГРУПОВОГО УРАХУВАННЯ АРГУМЕНТІВ, МЕТОД НАЙМЕНШИХ КВАДРАТІВ.

ABSTRACT

Explanatory note size – 178 pages, contains 28 illustrations, 11 tables, 3 applications, 43 references.

Topicality. Regression analysis is one of the most widely used universal tools for uncovering hidden relationships in data. However, constructing the correct structure of regression models and implementing the regression analysis algorithms in software are quite labor-intensive tasks. At the present level, an effective solution to these tasks is methodologically implemented by simultaneously using logically interconnected algorithms of various kinds to create the individual algorithm for solving a specific regression problem. Therefore, the use of the above-described approach to enhance the efficiency of the synthetic method in multivariate polynomial regression construction is relevant.

The aim of the study. The aim of this dissertation is to enhance the efficiency of the software for the synthetic method in multivariate polynomial regression construction, given by a redundant representation, through new software components development that implements the theoretically substantiated methodology for creating individual algorithms of decomposition method, and to enhance the efficiency of the modified group method of data handling by simultaneously using a series of regular criteria.

The object of research: the software and the methods in multivariate polynomial regression construction.

The subject of research: the methodology and the software tools for enhancing the efficiency of the software for the synthetic method in multivariate polynomial regression construction, given by a redundant representation.

To achieve this goal, the **following tasks** were formulated:

– to research the theoretical properties of the decomposition method, which is a part of the synthetic method, and ensure the possibility of enhancing its efficiency by implementing new theoretically substantiated methodology for creating individual algorithms in multivariate polynomial regression construction, given by a redundant representation;

– to create the approach that allows enhancing the efficiency of the modified group method of data handling, which is a part of the synthetic method, by simultaneously using a series of regular criteria;

– to substantiate the possibility of using the static algorithm for managing parallel asynchronous computational processes to increase the speed of the modified group method of data handling parallel implementation, by calculation the asymptotic computational complexity of its sub-algorithms;

– to research ways to extend the capabilities of general-purpose programming languages and universal software tools for automating the process of creating and programming individual algorithms in multivariate polynomial regression construction, given by a redundant representation, based on the principles of the synthetic method and the proposed new theoretically substantiated methodologies;

– to implement the methodology for extending the chosen general-purpose programming language for automating the process of creating and programming individual algorithms in multivariate polynomial regression construction, given by a redundant representation, based on the performed research;

– to modify the architecture of the software library that implements the synthetic method in multivariate polynomial regression construction, given by a redundant representation, by including new components that will enhance the efficiency of the synthetic method;

– to perform the statistical efficiency research of the new version of the modified group method of data handling;

– to perform the efficiency research of the whole software library in multivariate polynomial regression construction, given by a redundant representation.

The scientific novelty:

– for the first time, a new theoretically substantiated methodology for creating individual algorithms of decomposition method in multivariate polynomial regression construction, given by a redundant representation, has been proposed;

– for the first time, the methodology of enhancing the efficiency of modified group method of data handling in multivariate polynomial regression construction, given by a redundant representation, has been implemented through the use of asymmetric and symmetric regular criteria, considering the specifics of the formulated regression problem;

– for the first time, the methodology for extending the programming language Python to automate the process of creating and programming individual algorithms of the synthetic method in multivariate polynomial regression construction, given by a redundant representation, has been implemented;

– the modification of the software library architecture implementing the synthetic method, which includes new components that allow enhancing the efficiency of the method through the automated interactive implementation of individual algorithms in multivariate polynomial regression construction, given by a redundant representation, has been substantiated.

The practical value of the obtained results is that the newly implemented version of the software library in multivariate polynomial regression construction, given by a redundant representation, significantly enhances the efficiency of its usage for users with basic skills in programming and regression analysis and allows it to be embedded into a target application if necessary.

Relationship with working with scientific programs, plans, topics. Work was performed at the Department of Informatics and Software Engineering of the National Technical University of Ukraine «Kyiv Polytechnic Institute. Igor Sikorsky».

Approbation. The scientific provisions of the dissertation were tested at The Eighth International Scientific and Practical Conference «Software Engineering and Advanced Information Technologies (SoftTech – 2025)» – Kyiv, The Thirteenth International Scientific and Practical Conference «Comprehensive Quality Assurance of Technological Processes and Systems» (KZJTPS – 2023) – Chernihiv; The Fifth International Scientific and Practical Conference «Software Engineering and Advanced Information Technologies (SoftTech – 2023)» – Kyiv; The Third All-Ukrainian Scientific

and Practical Conference of Young Scientists and Students «Software Engineering and Advanced Information Technologies» (SoftTech – 2022 Autumn) – Kyiv; The First All-Ukrainian Scientific and Practical Conference of Young Scientists and Students «Software Engineering and Advanced Information Technologies» (SoftTech – 2021) – Kyiv.

Publications. The scientific provisions of the dissertation were published in:

1) Pavlov O., Holovchenko M., Mukha I., Lishchuk K., Drozd V. A Modified Method and an Architecture of a Software for a Multivariate Polynomial Regression Building Based on the Results of a Conditional Active Experiment. In: Hu Z., Dychka I., He M. (eds) Advances in Computer Science for Engineering and Education VI. ICCSEEA 2023. Lecture Notes on Data Engineering and Communications Technologies, Vol. 181, pp. 207-222. Springer, Cham (2023) // *The Sixth International Conference on Computer Science, Engineering and Education Applications ICCSEEA 2023, 16 April 2023, Warsaw, Poland*. – 2023, LNDECT Vol. 181, pp. 207-222. DOI: https://doi.org/10.1007/978-3-031-36118-0_19 [Scopus].

2) Pavlov O. A., Holovchenko M. M., Drozd V. V., Shargorodskiy V. S. Enhancing the Efficiency of Modified Group Method of Data Handling in Multivariate Polynomial Regression Construction, Given by a Redundant Representation // *Interdepartmental Scientific and Technical Journal «Adaptive Systems of Automatic Control».* K.: Igor Sikorsky Kyiv Polytechnic Institute, 2025. Vol. 1. № 46.

3) Pavlov O. A., Holovchenko M. M., Drozd V. V. Modification of the Decomposition Method of Constructing Multivariate Polynomial Regression which is Linear with Respect to Unknown Coefficients *// Bulletin of National Technical University «KhPI». Series: System analysis, control and information technologies*, – № 2 (12), 2024. – pp. 3-10. DOI: https://doi.org/10.20998/2079-0023.2024.02.01.

4) Pavlov O. A., Holovchenko M. M., Drozd V. V. An Adaptive Method for Building a Multivariate Regression *// Bulletin of National Technical University «KhPI». Series: System analysis, control and information technologies*, – № 1 (11), 2024. – pp. 3-8. DOI: https://doi.org/10.20998/2079-0023.2024.01.01.

5) Pavlov O. A., Khalus O. A., Melnikov O. V., Drozd V. V., Kobelskyi V. V., Medvediev M. Y. Static Algorithms for Managing Parallel Asynchronous Computational Processes // *Interdepartmental Scientific and Technical Journal «Adaptive Systems of Automatic Control»*. K.: Igor Sikorsky Kyiv Polytechnic Institute, 2024. Vol.1. № 44. pp. 116-126. DOI: https://doi.org/10.20535/1560-8956.44.2024. 302427.

6) Pavlov O. A., Holovchenko M. M., Drozd V. V. Efficiency Substantiation for a Synthetical Method of Constructing a Multivariate Polynomial Regression, Given by a Redundant Representation *// Bulletin of National Technical University «KhPI». Series: System analysis, control and information technologies*, – № 1 (9), 2023. – pp. 3-9. DOI: https://doi.org/10.20998/2079-0023.2023.01.01.

7) Pavlov O. A., Holovchenko M. M., Drozd V. V. Construction of a Multivariate Polynomial, Given by a Redundant Description, in Stochastic and Deterministic Formulations Using an Active Experiment *// Bulletin of National Technical University «KhPI». Series: System analysis, control and information technologies*, – № 1 (7), 2022. – pp. 3-8. DOI: https://doi.org/10.20998/2079-0023.2022.01.01.

8) Drozd V. V., Holovchenko M. M., Pavlov O. A. Methodology and Software Tools for Enhancing the Efficiency of the Synthetic Method in Multivariate Polynomial Regression Construction // *The Eighth International Scientific and Practical Conference «Software Engineering and Advanced Information Technologies (SoftTech – 2025)»*, Kyiv, May 13-15, 2025: Informatics and Software Engineering Department, FICT, Igor Sikorsky Kyiv Polytechnic Institute, 2025. – pp. 10-14.

9) Pavlov O. A., Holovchenko M. M., Drozd V. V. Synthetic Method for Constructing Multivariate Polynomial Regression // *Comprehensive Quality Assurance of Technological Processes and Systems (KZJTPS – 2023): Materials of Abstracts of the XIII International Scientific and Practical Conference (Chernihiv, May 25-26, 2023)*: in 2 vols. Vol. 2. – Chernihiv: National University «Chernihiv Polytechnic», 2023. – pp. 272-273.

10) Trotsiuk P. S., Drozd V. V., Holovchenko M. M., Pavlov O. A. Software for Statistical Research of the Efficiency of the Method in Multivariate Linear Regressions

Construction, Given by a Redundant Representation // *The Fifth International Scientific and Practical Conference «Software Engineering and Advanced Information Technologies (SoftTech – 2023)»*, Kyiv, December 19-21, 2023: Informatics and Software Engineering Department, FICT, Igor Sikorsky Kyiv Polytechnic Institute, 2023. – pp. 327-332.

11) Pavlov O. A., Holovchenko M. M., Drozd V. V., Revich M. M. Research of the Efficiency of the Method in Multivariate Linear Regressions Construction, Given by a Redundant Representation // *The Third All-Ukrainian Scientific and Practical Conference of Young Scientists and Students «Software Engineering and Advanced Information Technologies» (SoftTech – 2022 Autumn)*, Kyiv: Informatics and Software Engineering Department, FICT, Igor Sikorsky Kyiv Polytechnic Institute, 2022. – pp. 10-13.

12) Drozd V. V., Holovchenko M. M. Methods and Software Tools in Non-Linear Polynomial Regressions Construction Using Normalized Orthogonal Forsythe Polynomials // *The First All-Ukrainian Scientific and Practical Conference of Young Scientists and Students «Software Engineering and Advanced Information Technologies» (SoftTech – 2021)*. Section of Informatics and Software Engineering Department. Conference Materials. – Kyiv. – November 22-26, 2021. – pp. 82-86.

Keywords: MULTIVARIATE POLYNOMIAL REGRESSION, PROGRAMMING LANGUAGES, PROGRAMMING LANGUAGE EXTENSION, PARALLEL COMPUTING, HUMAN-COMPUTER INTERACTION, SYNTHETIC METHOD, DECOMPOSITION METHOD, MODIFIED GROUP METHOD OF DATA HANDLING, LEAST SQUARES METHOD.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

РА – регресійний аналіз;

БР – багатовимірна регресія;

БПР – багатовимірна поліноміальна регресія;

ПР – поліноміальна регресія;

ОПР – одновимірна поліноміальна регресія;

МНК – метод найменших квадратів;

МГУА – метод групового урахування аргументів;

ММГУА – модифікований метод групового урахування аргументів;

СКВ – середньоквадратичне відхилення;

АО – агрегований оператор;

ПЗ – програмне забезпечення;

ІТ – інформаційні технології.

ВСТУП

Побудова багатовимірної регресії за результатами активного експерименту, як показано, зокрема, в [1–4], є досі актуальною як в теоретичному, так і в практичному аспектах. У [4] був запропонований універсальний синтетичний метод побудови багатовимірної поліноміальної регресії, заданої надлишковим описом. В даній роботі пропонується якісно підвищити ефективність синтетичного методу, шляхом розширення теоретичних і практичних можливостей декомпозиційного методу та модифікованого методу групового урахування аргументів, що входять до його складу.

Регресійний аналіз є одним з найбільш поширених універсальних засобів виявлення прихованих взаємозв’язків у даних, але побудова правильної структури регресійних моделей та реалізація алгоритмів регресійного аналізу в програмному забезпеченні – досить трудомісткі задачі. На сучасному рівні ефективний розв’язок цих задач методологічно реалізується одночасним використанням логічно зв’язаних різнопланових алгоритмів для створення індивідуального алгоритму розв’язання конкретної задачі регресії. Тому використання вищеописаного підходу для підвищення ефективності синтетичного методу побудови багатовимірних поліноміальних регресій є актуальним.

Метою дисертаційної роботи є підвищення ефективності програмного забезпечення синтетичного методу побудови багатовимірних поліноміальних регресій, заданих надлишковим описом, шляхом розробки нових компонентів програмного забезпечення, що реалізують теоретично обґрунтовану методологію створення індивідуальних алгоритмів декомпозиційного методу, та підвищення ефективності модифікованого методу групового урахування аргументів, завдяки одночасному використанню низки регулярних критеріїв.

Для реалізації поставленої мети сформульовані наступні завдання:

– дослідити теоретичні властивості декомпозиційного методу, що входить до складу синтетичного методу, та забезпечити можливість підвищення його

ефективності, шляхом розробки нової теоретично обґрунтованої методології створення індивідуальних алгоритмів побудови багатовимірних поліноміальних регресій, заданих надлишковим описом;

– створити підхід, який дозволяє підвищити ефективність модифікованого методу групового урахування аргументів, що входить до складу синтетичного методу, шляхом одночасного використання низки регулярних критеріїв;

– обґрунтувати можливість використання алгоритму статичного управління паралельними асинхронними обчислювальними процесами для покращення швидкодії паралельної реалізації модифікованого методу групового урахування аргументів, шляхом знаходження асимптотичної складності його підалгоритмів;

– дослідити способи розширення можливостей мов програмування загального призначення та універсальних програмних засобів для автоматизації процесу створення і програмної реалізації індивідуальних алгоритмів побудови багатовимірних поліноміальних регресій, заданих надлишковим описом, на основі положень синтетичного методу та запропонованих нових теоретично обґрунтованих методологій;

– на основі проведеного аналізу розробити методологію розширення можливостей обраної мови програмування загального призначення для автоматизації процесу створення та програмної реалізації індивідуальних алгоритмів побудови багатовимірних поліноміальних регресій, заданих надлишковим описом;

– модифікувати архітектуру програмної бібліотеки, яка реалізує синтетичний метод побудови багатовимірних поліноміальних регресій, заданих надлишковим описом, шляхом включення в себе нових складових, що забезпечать підвищення ефективності синтетичного методу;

– виконати статистичне дослідження ефективності нової версії модифікованого методу групового урахування аргументів;

– виконати дослідження ефективності в цілому програмної бібліотеки побудови багатовимірних поліноміальних регресій, заданих надлишковим описом.

Об’єктом дослідження є програмне забезпечення та методи побудови багатовимірних поліноміальних регресій.

Предметом дослідження є методологія та програмні засоби підвищення ефективності програмного забезпечення синтетичного методу побудови багатовимірних поліноміальних регресій, заданих надлишковим описом.

Робота буде виконана на кафедрі інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Опис предметної області

Регресійний аналіз – це статистичний метод, який дозволяє встановити взаємозв'язок між залежною змінною Y та незалежними змінними X, x_i, $i = \overline{1,m}$, де m – кількість незалежних вхідних змінних [3]. Розв'язання цієї задачі дозволить оцінити і передбачити значення залежної змінної на основі інших факторів [3].

Математична модель залежності відгуку об'єкту Y від деяких вхідних незалежних змінних:

$$Y = F(x_1, x_2, \dots, x_m) + E, \tag{1.1}$$

де Y – відгук об'єкта дослідження;

x_i – незалежні змінні, одна або декілька, що характеризують об'єкт;

E – випадкова величина, що означає похибку.

Чорна скринька – метод, що покладено в основу регресійного аналізу [3]. На вхід деякого об'єкта подаються вектори незалежних змінних X (рисунок 1.1). На виході даного об'єкта знаходиться вектор параметрів Y, що характеризує об'єкт дослідження. При цьому залежну змінну Y називають також функцією відгуку, пояснювальною, вихідною, результуючою, ендогенною змінною, результативною ознакою, а незалежну змінну X – пояснюючою, вхідною, предикторною, екзогенною змінною, регресором, факторною ознакою [3].

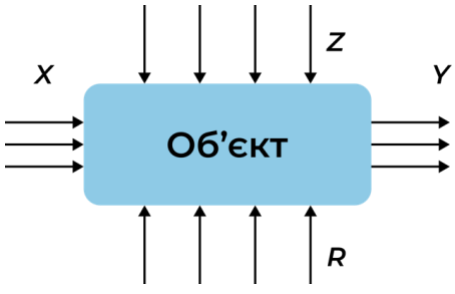


Рисунок 1.1 – Схема «чорної скриньки» регресійного аналізу [3]

Z-фактори – некеровані вхідні змінні, що є незалежними змінними, які можна виміряти. Вектор випадкових факторів R має вплив на об'єкт дослідження, але не піддається вимірюванню [3].

Вхідні незалежні змінні, які подаються на вхід «чорної скриньки» регресійного аналізу [3], мають задовольняти наступним умовам та обмеженням:

– вхідні незалежні змінні X повинні впливати на вихідну змінну Y;

– бажано, щоб вхідні незалежні змінні X були керованими, тобто їх можна задавати по бажанню експериментатора (це означатиме реалізацію активного експерименту на практиці);

– бажано, щоб вхідні незалежні змінні X не були взаємопов'язані між собою, інакше це може призвести до нерозв'язуваності задачі регресійного аналізу;

– точність вимірювання вхідної незалежної змінної X має бути на декілька порядків (знаків після коми) більшою за точність вимірювання вихідної змінної Y.

Активний експеримент – це експеримент, в якому дослідник може задавати довільні значення вхідних незалежних змінних з областей їх означення [4].

Пасивний експеримент – експеримент, при якому експериментатор пасивно фіксує значення вхідних незалежних змінних, що подаються на об'єкт, та відповідні значення вихідної змінної [4].

Надлишковий опис багатовимірної поліноміальної регресії [1, 3, 5] – множина базових поліномів, зважена лінійна згортка невідомої підмножини яких задає багатовимірну поліноміальну регресію.

У термінології регресійного аналізу фактором прийнято називати зважену лінійну згортку відомої підмножини вхідних незалежних змінних [4].

Залежний вектор Y піддається впливу ряду неконтрольованих або неврахованих явищ, тобто деякій похибці. Аналізуючи зв'язок між однією чи декількома незалежними змінними та залежною змінною, буде отримано регресію – одновимірну або багатовимірну [4].

Одновимірна регресія – аналіз зв'язку між однією незалежною вхідною змінною та залежною змінною [4].

Багатовимірна регресія – аналіз зв'язку між декількома незалежними вхідними змінними і залежною змінною [4].

На рисунку 1.2 наведено один із варіантів класифікації регресійних моделей, що найчастіше використовуються на практиці. Далі вони будуть розглянуті більш детально [3].

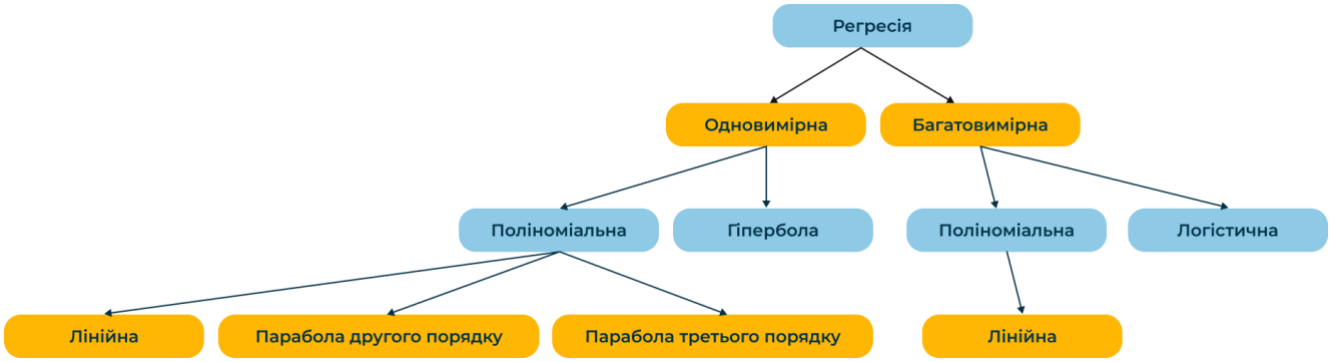


Рисунок 1.2 – Класифікація регресійних моделей

Одновимірні регресійні моделі. Одновимірна лінійна регресія [6]. Графік регресійної моделі лінійної залежності апроксимується звичайною лінією. Можна побудувати одновимірну лінійну регресію, яка має вигляд представлений нижче:

$$Y(x) = b_0 + b_1 x + E, \tag{1.2}$$

де x – вхідна незалежна змінна;

$Y(x)$ – залежна вихідна змінна;

b_0 і b_1 – невідомі значення коефіцієнтів регресійної моделі, які необхідно оцінити;

E – випадкова величина з математичним сподіванням $ME = 0$ та дисперсією $DE = \sigma^2 < \infty$; σ^2 відома або задана її верхня оцінка.

Параболи [7, 8]. У деяких випадках пряма лінія може занадто погано апроксимувати залежності, у такому випадку покращити апроксимацію залежності можна додаванням до моделі регресії ще одного квадратного члену.

$$Y(x) = b_0 + b_1 x + b_2 x^2 + E, \tag{1.3}$$

де x – вхідна незалежна змінна;

$Y(x)$ – залежна вихідна змінна;

b_0, b_1 і b_2 – невідомі значення коефіцієнтів регресійної моделі, які необхідно оцінити;

E – випадкова величина з математичним сподіванням $ME = 0$ та дисперсією $DE = \sigma^2 < \infty$; σ^2 відома або задана її верхня оцінка.

Можна розширити рівняння параболи другого порядку до поліному більш високого порядку, додавши члени x^2, x^3, ... , x^r з відповідними їм коефіцієнтами. Після чого буде отримано поліноміальну регресію [9].

Гіпербола [7]. Рівняння регресійної моделі гіперболи має наступний вигляд:

$$Y(x) = b_0 + b_1 \cdot \frac{1}{x} + E, \tag{1.4}$$

де x – вхідна незалежна змінна;

$Y(x)$ – залежна вихідна змінна;

b_0 і b_1 – невідомі значення коефіцієнтів регресійної моделі, які необхідно оцінити;

E – випадкова величина з математичним сподіванням $ME = 0$ та дисперсією $DE = \sigma^2 < \infty$; σ^2 відома або задана її верхня оцінка.

Одновимірна поліноміальна регресія [9, 10]. Вигляд моделі одновимірної поліноміальної регресії представлено нижче:

$$Y(x) = b_0 + b_1 x + \ldots + b_r x^r + E, \tag{1.5}$$

де x – вхідна незалежна змінна;

$Y(x)$ – залежна вихідна змінна;

$b_0, \dots, b_r$ – невідомі значення коефіцієнтів регресійної моделі, які необхідно оцінити;

E – випадкова величина з математичним сподіванням $ME = 0$ та дисперсією $DE = \sigma^2 < \infty$; σ^2 відома або задана її верхня оцінка.

Багатовимірний (множинний) регресійний аналіз [11, 12, 13]. У багатьох практичних задачах результат залежить не від однієї вхідної незалежної змінної, а від декількох. У таких випадках використовується багатовимірний регресійний аналіз.

Нижче наведено приклад багатовимірної лінійної регресії (БЛР):

$$Y(\bar{x}) = b_0 + b_1 x_1 + b_2 x_2 + \dots + b_m x_m + E, \tag{1.6}$$

де b_0 – вільний коефіцієнт;

$b_1, b_2, \dots, b_m$ – коефіцієнти багатовимірної регресії;

$\bar{x} = (x_1, \dots, x_m)$ – детермінований вектор вхідних змінних;

m – кількість змінних;

E – випадкова величина з математичним сподіванням $ME = 0$ та дисперсією $DE = \sigma^2 < \infty$; σ^2 відома або задана її верхня оцінка.

У багатьох інформаційних експертних системах чи системах класифікації використовуються багатовимірні поліноміальні (степеневі) регресії [4].

Модель багатовимірної поліноміальної регресії (БПР) має наступний вигляд:

$$Y(\bar{x}) = \sum b_{i_1 \dots i_t}^{j_1 \dots j_t} (x_{i_1})^{j_1} \cdot (x_{i_2})^{j_2} \cdot \dots \cdot (x_{i_t})^{j_t} + E$$

$$\forall (i_1 \dots i_t) \in K, \forall (j_1 \dots j_t) \in K(i_1 \dots i_t), \tag{1.7}$$

де $\bar{x} = (x_1, \dots, x_m)$ – детермінований вектор вхідних змінних;

E – випадкова величина з математичним сподіванням $ME = 0$ та дисперсією $DE = \sigma^2 < \infty$; σ^2 відома або задана її верхня оцінка.

У задачах класифікації, де необхідно визначити, до якої категорії відноситься спостереження часто застосовують логістичну регресію (одновимірну чи багатовимірну) [1]. Багатовимірна логістична регресія задається наступним чином:

$$y^*(\bar{x}) = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_m x_m + E,$$

$$Y = \begin{cases} 0, y^* \leq 0 \\ 1, y^* > 0 \end{cases}, \quad (1.8)$$

де b_0 – вільний коефіцієнт;

$b_1, b_2, \ldots, b_m$ – коефіцієнти логістичної регресії;

$\bar{x} = (x_1, \ldots, x_m)$ – детермінований вектор вхідних змінних;

m – кількість змінних;

E – випадкова величина з математичним сподіванням $ME = 0$ та дисперсією $DE = \sigma^2 < \infty$; σ^2 відома або задана її верхня оцінка.

Як видно з наведеного аналізу у підрозділі 1.1, у першому розділі розглядаються регресійні моделі, в аналітичному виразі яких невідомі коефіцієнти, які необхідно оцінити, входять лінійно. Саме такий клас регресійних моделей є предметом дослідження даної роботи.

Під побудовою багатовимірної поліноміальної регресії розуміється знаходження частини її коефіцієнтів з заданою точністю, знаходження істинної структури регресії та знаходження коефіцієнтів БПР евристичним методом для тих коефіцієнтів, які неможливо знайти з заданою точністю.

1.2 Аналіз теоретичних можливостей синтетичного методу та ефективності існуючих програмних засобів його реалізації

Для побудови БПР та інших видів регресії існує достатньо широкий набір класичних універсальних методів статистичного аналізу, до них належать: класичний метод найменших квадратів, метод найбільшої правдоподібності, метод групового урахування аргументів, еволюційні методи (генетичний алгоритм та

інші), методи, що використовують штучні нейронні мережі, факторний аналіз, синтетичний метод [4]. Крім того, кожен з цих методів використовує один або декілька критеріїв для визначення значимості членів, що входять у регресії, сюди можна віднести: суму квадратів залишків, коефіцієнт детермінації R^2, статистику Малоуза, F-статистику, регресійну суму квадратів, залишкову суму квадратів (ЗСК), t-впорядкований пошук, виключення та покрокову регресію.

Як було показано в [4] жоден з наведених вище універсальних методів не гарантує точного розв'язку, а вони лише доповнюють один одного для отримання більш якісного розв'язку. Стосовно критеріїв, що визначають значимість членів БПР та якість оцінки їх коефіцієнтів, то вони зазвичай є досить громіздкими для обчислення та гарантують шуканий рівень достовірності лише при використанні декількох критеріїв одночасно, що в свою чергу лише збільшує використання обчислювальних ресурсів. Однак, з аналізу можна зробити заключення, що синтетичний метод дозволяє оцінити частину коефіцієнтів БПР з заданою точністю, а іншу досить добре оцінити з використанням евристичного методу. Тому далі будуть розглянуті теоретичні властивості складових саме синтетичного методу та шляхи його вдосконалення як з теоретичної, так і з практичної точки зору.

Синтетичний метод [4] полягає в послідовному використанні декомпозиційного методу та модифікованого методу групового урахування аргументів (ММГУА). Спочатку задача розв'язується декомпозиційним методом і знаходяться всі можливі оцінки коефіцієнтів при нелінійних членах БПР з наперед заданою точністю. Пошук істинної (з точки зору методу) структури БПР та оцінки її коефіцієнтів знаходяться за допомогою ММГУА. Якщо виконується побудова БПР на основі пасивного експерименту, то синтетичний метод перетворюється в ММГУА.

Формальний опис декомпозиційного методу [14]

Описується для задачі побудови БПР, яка задана наступним надлишковим описом:

$$Y(\bar{x}) = \sum_{\forall(i_1,\dots,i_t)\in K,\ \forall(j_1,\dots,j_t)\in K(i_1,\dots,i_t)} b_{i_1\dots i_t}^{j_1\dots j_t}\left(x_{i_1}\right)^{j_1}\cdots\left(x_{i_t}\right)^{j_t} + E, \qquad (1.9)$$

де $\bar{x} = (x_1, \dots, x_m)^T$ – детермінований вектор вхідних змінних,

$$x_i \in [c_i,\ d_i],\ c_i > 0,\ i = \overline{1,m}, \qquad (1.10)$$

E – випадкова величина, яка має довільний розподіл; $ME = 0$, $DE = \sigma^2 < \infty$, відома величина дисперсії або її верхня оцінка. Значення коефіцієнтів $b_{i_1\dots i_t}^{j_1\dots j_t}$ невідомі (b_0^0 – константа).

Декомпозиційний метод у загальному випадку передбачає зведення знаходження оцінок при нелінійних членах БПР (1.9) до послідовної побудови одновимірних поліноміальних регресій (ОПР) та розв'язання відповідних систем лінійних рівнянь, розв'язки яких є необхідними для знаходження оцінок при нелінійних членах БПР (1.9).

Загальна алгоритмічна схема отримання оцінок при нелінійних членах БПР (1.9) складається з двох підалгоритмів [14].

Агрегована алгоритмічна схема першого підалгоритму. l-й крок ($l \le L_1$, де L_1 – загальна кількість членів БПР з нелінійною складовою (1.9), кожна з яких містить хоча б одну скалярну змінну у степені, більшому або рівному двом) реалізується для наступного нелінійного члена БПР (1.9), коефіцієнт якого не був оцінений на попередніх кроках першого підалгоритму і який містить скалярну змінну в максимальному степені [14]. Буде позначено її як x_{i_p}. У БПР (1.9) скалярна змінна x_{i_p} замінюється віртуальною скалярною змінною z: $x_{i_p} = a_{i_p}z + b_{i_p}$, де a_{i_p}, b_{i_p} та знаходиться відповідно до методології знаходження оцінок ОПР з

використанням нормованих ортогональних поліномів Форсайта (НОПФ) для $d = d_{i_p}$, $c = c_{i_p}$. Решта змінних приймають фіксовані значення. В цьому випадку БПР (1.9) перетворюється в ОПР, а дані основного віртуального експерименту знаходяться по основному реальному експерименту [14]:

$$\left(x_{i_p,i} \underset{j \neq i_p}{\forall} x_{j,i} = x_j^{\Phi} \to y_i,\ i = \overline{1,n}\right),$$

а саме:

$$\left(z_i \to y_i - \sum_{\forall b_{i_1 \dots i_t}^{j_1 \dots j_t} \in \bigcup_{m=1}^{l-1}\{J_m\}} b_{i_1 \dots i_t}^{j_1 \dots j_t} \left(x_{i_1,i}\right)^{j_1} \cdots \left(x_{i_t,i}\right)^{j_t},\ i = \overline{1,n}\right), \qquad (1.11)$$

де $\bigcup_{m=1}^{l-1}\{J_m\}$ – множина коефіцієнтів, що з необхідною точністю була знайдена на попередніх кроках першого підалгоритму. На першому кроці ця множина порожня.

Якщо є можливість провести повторний віртуальний експеримент для підвищення точності оцінок коефіцієнтів БПР, то це можна зробити відповідно до теорії побудови ОПР. У віртуальному експерименті кількість вихідних даних є y_i, $i = \overline{1, kn}$, а вхідні дані основного експерименту повторюються k разів. Максимальний степінь ОПР дорівнює j_p.

На l-му кроці кількість побудованих ОПР може перевищувати одну, якщо коефіцієнт (коефіцієнти), для якого вона будується, виражається (виражаються) лінійно через інші коефіцієнти при нелінійних членах БПР (1.9), не оцінених на попередніх кроках першого підалгоритму. Це призводить до необхідності розв’язувати систему (системи) лінійних рівнянь, права частина, якої (яких) є оцінками при нелінійних членах ОПР, що необхідно для знаходження оцінок відповідних коефіцієнтів БПР. З точки зору реалізації декомпозиційного методу на практиці це серйозний недолік, оскільки вимагає використання алгоритмів для розв’язання систем лінійних рівнянь та ускладнює саму процедуру побудови ОПР на основі БПР.

Агрегована схема другого підалгоритму. l-й крок ($l \le L_2$, де L_2 – кількість нелінійних складових (1.9) виду $b_{i_1 \dots i_t}^{1 \dots 1} x_{i_1} \cdots x_{i_t}$) реалізується для нелінійного коефіцієнту $b_{i_1 \dots i_{t_l}}^{1 \dots 1}$, який не був оцінений на попередніх кроках першого та другого підалгоритмів і має максимальне значення t_l [14]. Кожна вхідна змінна $x_{i_j}, j = \overline{1, t_l}$, лінійно виражається через віртуальну змінну z: $x_{i_j} = a_{i_j} z + b_{i_j}$ відповідно до теорії побудови ОПР, для $c = c_{i_j}$, $d = d_{i_j}$, $j = \overline{1, t_l}$ [14]. В основному експерименті змінні x_{i_j}, $j = \overline{1, t_l}$, змінюються відповідно до теорії побудови ОПР [14]. Інші скалярні вхідні змінні в кожному спостереженні приймають деякі фіксовані значення. Дані віртуального основного експерименту визначаються даними основного експерименту [14]:

$$\left(x_{i_1,i}, \dots, x_{i_{t_l},i} \forall_j x_{j,i} = x_j^{\Phi}, j \notin \{i_1, \dots, i_{t_t}\} \to y_i,\ i = \overline{1, n} \right),$$

а саме:

$$\left(z_i \to y_i - \sum_{\forall b_{i_1 \dots i_t}^{j_1 \dots j_t} \in \bigcup_{m=1}^{K_1} \{J_m\}} \hat{b}_{i_1 \dots i_t}^{j_1 \dots j_t} \left(x_{i_1,i}\right)^{j_1} \cdots \left(x_{i_t,i}\right)^{j_t} - \right.$$

$$\left. - \sum_{\forall b_{i_1 \dots i_t}^{j_1 \dots j_t} \in \bigcup_{m=1}^{l-1} \{G_m\}} \hat{b}_{i_1 \dots i_t}^{1 \dots 1} \prod_{l=1}^{t_l} x_{i_l,i},\ i = \overline{1, n} \right), \qquad (1.12)$$

де K_1 – кількість членів БПР, які були оцінені під час роботи першого підалгоритму; G_m – множина коефіцієнтів, оцінених з заданою точністю на попередніх кроках другого підалгоритму [14].

Як і у першому підалгоритмі, тут допускається проведення повторного віртуального експерименту для підвищення точності знайдених оцінок БПР. Максимальний степінь ОПР дорівнює t_l.

У [4] було приведено обґрунтування кількості випробувань основного експерименту n та вибору значень z_i, $i = \overline{1, n}$ віртуальної скалярної змінної z.

Там обґрунтовано, що для r_{max} (максимальна степінь ОПР) пропонується наступне компромісне рішення: $n = 10,\ z_1 = -50,\ z_{10} = 50,\ \Delta z = (z_i - z_{i-1}) = const$. У цьому випадку

$$D\hat{\gamma}_2 = 4.26 \cdot 10^{-6}\sigma^2, D\hat{\gamma}_3 = 7.55 \cdot 10^{-9}\sigma^2,$$
$$D\hat{\gamma}_4 = 1.4 \cdot 10^{-12}\sigma^2, D\hat{\gamma}_5 = 1.28 \cdot 10^{-15}\sigma^2, \quad (1.13)$$

це означає що, зі збільшенням j на одиницю $D\hat{\gamma}_j$ зменшується приблизно на три порядки, починаючи з $D\hat{\gamma}_2$.

Внаслідок реалізації першого та другого підалгоритмів декомпозиційного методу отримали множину $\{J\}$ коефіцієнтів при нелінійних членах БПР (1.9), оцінених з заданою точністю.

Всього у декомпозиційному методі виділяють п’ять основних класів регресій, для яких знаходяться оцінки коефіцієнтів при нелінійних членах з заданою точністю; особливий інтерес викликає п’ятий клас, що дозволяє будувати індивідуальний алгоритм знаходження оцінок коефіцієнтів при нелінійних членах БПР та, як наслідок, уникати необхідності розв’язань систем лінійних рівнянь. Крім того, використання індивідуального алгоритму декомпозиційного методу значно підвищує точність оцінок коефіцієнтів при нелінійних членах БПР [14].

Коефіцієнти, оцінки яких з заданою точністю близькі до нуля, виключені з надлишкового опису (1.9) та множини $\{J\}$. На цьому робота декомпозиційного методу закінчується.

Для отримання решти оцінок та остаточної структури БПР використовується ММГУА, викладений в [4] для побудови БПР, заданих надлишковим описом. Дійсно, задача побудови БПР, заданих надлишковим описом (1.9), звелась до наступної:

$$Y(\bar{x}) = \sum_{\forall\left\{b_{i_1\dots i_t}^{j_1\dots j_t}\right\}\setminus\{J\}} b_{i_1\dots i_t}^{j_1\dots j_t}\left(x_{i_1}\right)^{j_1}\cdots\left(x_{i_t}\right)^{j_t} + f(\bar{x}) + E, \quad (1.14)$$

де $f(\bar{x}) = \sum_{\forall b_{i_1 \dots i_t}^{j_1 \dots j_t} \in \{J\}} \hat{b}_{i_1 \dots i_t}^{j_1 \dots j_t} (x_{i_1})^{j_1} \cdots (x_{i_t})^{j_t}$.

Зміст ММГУА полягає у наступних семи кроках. O визначає множину $\forall \left\{ b_{i_1 \dots i_t}^{j_1 \dots j_t} \right\} \setminus \{J\}$, тобто коефіцієнти, що не були знайдені декомпозиційним методом.

Крок 1. Дані експерименту по набору вхідних змінних без урахування членів БПР, що були знайдені з необхідною точністю за допомогою декомпозиційного методу, визначаються як: $D = (x_i \to y_i, i = \overline{1,n})$, розбиваються на дві підмножини $D_1 = \{x_{il} \to y_{il}, l = \overline{1,n_1}\}$, $D_2 = \{x_{jl} \to y_{jl}, l = \overline{1,n_2}\}$, $D_1 \cup D_2 = J$, $D_1 \cap D_2 = \emptyset$.

Крок 2. Методом найменших квадратів з використанням даних лише підмножини D_1 знаходяться оцінки $\hat{b}_j$, $j \in O$. Використовується загальна формула методу найменших квадратів: $\hat{b} = (A^T A)^{-1} A^T y$.

Крок 3. За допомогою алгоритму кластерного аналізу [4], використовуючи оцінки $\hat{b}_j$, $j \in O$, множина коефіцієнтів $\{b_j, j \in O\}$ розбивається на два класи M_1 та M_2, що мають порожній перетин.

Крок 4. Методом найменших квадратів з використанням даних D_1, будується множина часткових описів БПР. У кожний частковий опис входять члени, що відповідають коефіцієнтам з множини M_1, та усі можливі різні комбінації членів множини M_2.

Крок 5. За підмножиною даних D_2 по заданому критерію обирається частковий опис, структура якого приймається такою, що відповідає шуканій БПР.

Крок 6. За множиною даних D методом найменших квадратів знаходяться результуючі оцінки коефіцієнтів БПР за обраним частковим описом.

Крок 7. За допомогою критерія χ^2 перевіряється достовірність отриманих результатів.

У ММГУА, як і у декомпозиційному методі, допускається використання повторів основного експерименту. ММГУА обов'язково знайде структуру БПР та оцінки коефіцієнтів БПР, які не були знайдені декомпозиційним методом, але він

використовує лише один регулярний критерій прийняття структури БПР, що часто призводить до недостовірних результатів, про що повідомить даний метод [4].

Загалом до недоліків декомпозиційного методу можна віднести наступне: формальне використання першого та другого підалгоритмів в більшості випадків приводить до оцінки коефіцієнтів при нелінійних членах БПР значно меншої, ніж при використанні індивідуального алгоритму, що може створити користувач, який володіє теорією регресійного аналізу, проте в роботах [4, 14] відсутні теоретичні основи створення індивідуальних алгоритмів та можливості автоматизації їх програмної реалізації. Також створення індивідуального алгоритму дозволить уникнути необхідності розв'язувати системи лінійних рівнянь при побудові ОПР та знаходженні оцінки чергового члену БПР. При аналізі теоретичних можливостей універсального МГУА [16] виявлено, що використання не одного, а низки критеріїв суттєво підвищують його ефективність, однак в ММГУА використовується лише один регулярний критерій, що суттєво зменшує його ефективність. Можна зробити висновок про доцільність застосування декількох регулярних критеріїв у ММГУА.

Таким чином, виникає наступна проблема – користувач, який володіє теорією регресійного аналізу, аналізує конкретний надлишковий опис БПР і створює індивідуальний алгоритм декомпозиційного методу як складової синтетичного методу. Програмні засоби повинні максимально ефективно автоматизувати процес реалізації цього алгоритму. Крім того, виникає проблема якості алгоритму ММГУА як складової синтетичного методу, яку доцільно вирішувати запровадженням декількох регулярних критеріїв. ММГУА варто використовувати, як заключний етап, який іде після індивідуального алгоритму декомпозиційного методу.

1.3 Аналіз існуючих універсальних методів побудови багатовимірних регресій з метою підвищення алгоритмічної ефективності синтетичного методу

Внаслідок проведеного аналізу існуючих методів побудови БПР, виявлено два підходи, використання яких може суттєво підвищити ефективність синтетичного методу.

Перший з них – це автоматизація процесу створення індивідуальних алгоритмів декомпозиційного методу як складової синтетичного методу детально буде розглянутий у підрозділі 2.1.

Другий – проблеми методу знаходження оцінок коефіцієнтів БПР та структури БПР для тих членів, оцінки коефіцієнтів яких не були знайдені декомпозиційним методом. У [4] для цієї задачі застосовується ММГУА, але, оскільки на відміну від декомпозиційного методу, ММГУА не знаходить оцінки з заданою точністю, варто розглянути варіанти його заміни більш ефективним методом.

Варіантами заміни ММГУА можуть бути метод найменших квадратів (МНК) та класичний універсальний комбінаторний МГУА.

У загальному вигляді задача регресії, що розв’язується, може бути представлена наступним чином [4]:

$$Y = Ab + \bar{E}, \tag{1.15}$$

де $\bar{E} = (E_1, \dots, E_n)^T, Y = (Y_1, \dots, Y_n)^T$, вигляд матриці A залежить від виразу (1.15), вектор невідомих коефіцієнтів БПР, що входить в (1.15), для спрощення позначили b. Тоді $MY = Ab$, вектор оцінок $\hat{b} = \left(\hat{b}_0, \hat{b}_1, \dots, \hat{b}_{\mathrm{m}}\right)^T$, отриманий МНК, має вигляд:

$$\hat{b} = (A^T A)^{-1} A^T Y,\ M\hat{b} = b. \tag{1.16}$$

Даний метод не дозволяє знаходити структуру БПР, але є дуже простим у реалізації і не вимагає великої кількості обчислювальних ресурсів, тому його доцільно застосовувати замість ММГУА у синтетичному методі.

Метод групового урахування аргументів (МГУА) [16] являє собою подальший розвиток регресійного аналізу і є типовим методом індуктивного моделювання, що базується на деяких принципах теорії навчання та самоорганізації, зокрема на принципі «селекції» або спрямованого відбору. Цей метод включає в себе сімейство індуктивних алгоритмів, які забезпечують рекурсивний селективний відбір моделей, на основі яких будуються складніші моделі. Точність моделювання на кожному наступному етапі рекурсії підвищується шляхом ускладнення моделі. Як правило, вихідну вибірку розбивають на навчальну (обсягом $n_{\text{н}}$) та перевірочну (обсягом $n_{\text{пр}}$) вибірки у співвідношенні $\frac{n_{\text{н}}}{n_{\text{пр}}} = \frac{3}{2}$. Перша вибірка використовується для оцінки коефіцієнтів моделі методом найменших квадратів, а друга – для визначення якості моделі за допомогою коефіцієнта детермінації $\hat{R}^2$ або середньоквадратичного відхилення похибки (СКВ) σ^2 (залишкової дисперсії) [16].

З кожним етапом ітерації СКВ зменшується, але після досягнення певного рівня складності, що залежить від характеру і кількості даних, а також загального вигляду моделі, СКВ починає зростати [16].

Далі буде розглянуто процес синтезу моделі оптимальної складності, що використовується на практиці багаторядного поліноміального алгоритму МГУА.

Вибирається загальний вигляд моделей, що перебираються (опорних функцій) з m незалежними змінними, використовуючи узагальнений поліном Колмогорова–Габора:

$$Y = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{j=1}^{m} \sum_{i<j} a_{ij} x_i x_j + \sum_{j=1}^{m} \sum_{i<j} \sum_{k<j} a_{ijk} x_i x_j x_k + \cdots$$

Цей поліном з високою точністю може апроксимувати будь-яку неперервну на кінцевому інтервалі функцію у вигляді полінома певного степеня відповідно до теореми Вейєрштраса. Складність моделі в такому разі визначається кількістю коефіцієнтів a_{ij} $(i, j = 1, \dots, i \leq j)$ [16].

Використовуючи опорні функції, будуються різні варіанти моделей, які включають попарні комбінації вихідних змінних, з яких складаються рівняння вирішальних функцій, зазвичай не вищі другого порядку:

$$y_1 = f(x_1, x_2),\ y_2 = f(x_1, x_3),\ \dots,\ y_s = f(x_i, y_j).$$

Зазвичай як функцію f вибираються прості залежності:

$$y(x_i, x_j) = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j$$

або

$$y(x_i, x_j) = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2.$$

Використовуючи навчальну вибірку, методом найменших квадратів для кожної моделі визначаються коефіцієнти $a_0, a_1, a_2, a_3, a_4, a_5$.

Набір отриманих моделей складає перший ряд селекції. Серед усіх моделей першого ряду селекції вибираються кілька (від 2 до 10) найкращих. Якість моделей визначається коефіцієнтом детермінації або СКВ помилки σ^2 за допомогою перевірочної вибірки [16].

Відібрані часткові моделі формують множину нових змінних, які є вихідними змінними для часткових моделей другого ряду селекції:

$$z_1 = f(y_i, y_j),\ z_2 = f(y_i, y_k),\ \dots.$$

Коефіцієнти нових моделей знаходяться за МНК, використовуючи навчальну вибірку. Якість моделей оцінюється за допомогою перевірочної вибірки, і серед них вибираються найкращі, які використовуються як змінні моделей наступного, третього ряду і т.д. Складність загальної моделі зростає від ряду до ряду. Якщо, як опорні функції використовуються поліноми другого степеня, то на кожному кроці ітерації степінь результуючого полінома подвоюється.

Подальше ускладнення моделі припиняється після досягнення мінімуму СКВ σ^2 (або максимуму коефіцієнта детермінації R^2), визначеного за допомогою перевірочної вибірки, або коли подальше поліпшення критерію селекції не перевищує деякої величини ε (параметр алгоритму).

На заключному етапі, роблячи послідовну заміну змінних, отримують модель, що включає вихідні змінні.

Даний класичний багаторівневий селекційний алгоритм є достатньо громіздким та складним у реалізації, тому буде розглянуто більш зручну його версію, а саме комбінаторний МГУА.

Формальний опис комбінаторного МГУА [16].

На першому ряді перебираються всі моделі-кандидати виду: $x_i = b_0 + b_i x_i$, $i \in [1, n]$.

Методом найменших квадратів розраховується параметр b_i кожної з моделей. Після цього обчислюється заданий зовнішній критерій, за значенням якого моделі сортуються у порядку зростання. Середня величина E_1 зовнішнього критерію перших моделей приймається за ступінь достовірності поточного ряду.

Далі алгоритм переходить на другий ряд і нові моделі-кандидати ускладнюються.

$$y(x_i, x_j) = b_0 + b_i x_i + b_j x_j i, j \in [1, n], j > i.$$

Знову за допомогою МНК обчислюються параметри b_i та b_j і потім для кожної з моделей-кандидатів другого ряду розраховується зовнішній критерій. Після сортування значень критерію визначають ступінь достовірності другого ряду E_2 як середнього результату кращих моделей.

Далі величин достовірностей порівнюється між собою, якщо $E_1 - E_2 \geq limit$, то достовірність моделей поточного ряду значно покращилася і алгоритм продовжить навчання, перейшовши на третій ряд для перебору ще складніших моделей $y\ (x_i, x_j, x_k)$.

Якщо $E_1 - E_2 < limit$, то достовірність моделей покращилася трохи або зовсім погіршилася. У даному випадку процес навчання припиняється і моделлю прийнятної складності стає модель попереднього ряду з найменшим значенням зовнішнього критерію.

Максимально можливе число рядів даного алгоритму дорівнює n. Кількість моделей-кандидатів у кожному ряді дорівнює $M_i = C_n^i$. Загальна максимальна кількість моделей-кандидатів на всіх рядах дорівнює $M = \sum_{i=1}^{n} C_n^i = 2^{n-1}$.

У якості зовнішнього критерія буде використано регулярний критерій квадрату відхилення суми різниць [4].

$$E = \sum_{i=1}^{n} \left[y_i - \widehat{b_0} - \sum_{\forall \hat{b}_l \neq 0} \left(\hat{b}_l \cdot x_{li} \right) \right]^2, \tag{1.17}$$

Далі буде розглянуто методологію порівняння методів побудови БПР для обрання методу, що найкраще розв'яже дану задачу [16].

Вхідні параметри дослідження для кожного з методів:

– i – кількість вхідних незалежних змінних;

– n – кількість спостережень активного експерименту;

– l – кількість повторів активного експерименту;

– p – відсоток перевірочних послідовностей від загального числа повторів активного експерименту;

– нормальний розподіл випадкової величини E з нульовим математичним сподіванням та змінною дисперсією;

– exp_count – кількість незалежних повторів експерименту з вищевказаними параметрами;

– надлишковий опис багатовимірної поліноміальної регресії;

– B – вектор істинних значень коефіцієнтів;

– X – матриця значень вхідних змінних основного експерименту.

Для кожного набору значень параметрів генерується індивідуальний набір реалізацій випадкової величини E з заданою дисперсією. Результати експериментів будуть подані у наступному вигляді:

– відсоток коректних моделей у рамках незалежних повторів експерименту для заданого значення дисперсії випадкової величини E;

– максимальна кількість неправильно визначених нульових коефіцієнтів у моделях в рамках незалежних повторів експерименту для заданого значення дисперсії випадкової величини E;

– середня кількість неправильно визначених нульових коефіцієнтів у моделях в рамках незалежних повторів експерименту для заданого значення дисперсії випадкової величини E;

– середнє значення міри порівнянь для коректних моделей (1.18);

– мінімальне значення міри порівнянь для коректних моделей (1.18);

– максимальне значення міри порівнянь для коректних моделей (1.18);

– середнє значення часу побудови багатовимірної поліноміальної регресії у кожному окремому незалежному повторі експерименту для заданого значення дисперсії випадкової величини E.

У якості міри порівнянь буде обрано наступну $d(\hat{b} - b)$ [18],

де ($\hat{b} = \begin{pmatrix} \hat{b}_0 \\ \vdots \\ \hat{b}_n \end{pmatrix}$, $b = \begin{pmatrix} b_0 \\ \vdots \\ b_n \end{pmatrix}$), а саме значення міри розраховується наступним чином:

$$d(\hat{b} - b) = \left\| \frac{\hat{b}}{\|\hat{b}\|} - \frac{b}{\|b\|} \right\|, \tag{1.18}$$

де $\|x\| = \sqrt{\sum_{i=1}^{n} x_i^2}$. Ця міра порівнянь була вибрана завдяки тому, що вона не залежить від довжини векторів, що порівнюються. Оцінки коефіцієнтів знайдені точно, якщо $d(\hat{b} - b)$ належить відрізку [0; 0,02].

Обрані наступні параметри дослідження [16]:

– надлишковий опис наступного вигляду:

$$Y(\bar{x}) = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 + b_5x_1x_2 + b_6x_1x_4 + b_7x_2^2 + b_8x_4^3 + E;$$

– кількість вхідних змінних – 4;

– кількість нульових коефіцієнтів – 3;

– кількість спостережень активного експерименту – 100 (взято на основі досліджень з [17]);

– кількість повторів активних експериментів – 6;

– кількість повторів перевірочної послідовності – 4;

– діапазон значень коефіцієнтів – [-10; 10];

– діапазон значень вхідних змінних – [1; 40);

– кількість незалежних реалізацій повторного повного експерименту – 1000.

Для відображення результатів дослідження було створено таблицю із генерацією реалізацій випадкових величин з нормальним розподілом та змінним значенням дисперсії. Для кожного з методів проведено незалежні повтори експериментів. Кожна комірка результату буде містити такі значення:

– $correct$ – відсоток правильних структур;

– $mean$ – середнє значення міри порівнянь для правильних структур;

– max – максимальне значення міри порівнянь для правильних структур;

– min – мінімальне значення міри порівнянь для правильних структур;

– $avgzeros$ – середня кількість неправильно визначених нульових коефіцієнті;

– $maxzeros$ – максимальна кількість неправильно визначених нульових коефіцієнті;

– $time$ – середнє значення часу побудови багатовимірної поліноміальної регресії.

Для проведення даного дослідження пропонується інкрементально збільшувати значення середньоквадратичного відхилення при фіксованій кількості

спостережень для кожного з досліджуваних методів та обрати метод, який дає кращі середньозважені показники. Дослідження проілюстровано в таблиці 1.1.

Таблиця 1.1 – Порівняння МНК, ММГУА та комбінаторного МГУА

Метод для порівняння значення середньоквадратичного відхилення	МНК	Комбінаторний МГУА	ММГУА
1	mean: 0,01215 max: 0,07135 min: 0,00029 time: 0,00836	correct: 60,8 mean: 0,01704 max: 0,63019 min: 0,00015 avgzeros: 0,434 maxzeros: 2 time: 0,38259	correct: 95,89 mean: 0,00780 max: 0,12261 min: 0,00003 avgzeros: 0,026 maxzeros: 1 time: 0,08490
2	mean: 0,02598 max: 0,14261 min: 0,00048 time: 0,00793	correct: 62,0 mean: 0,02410 max: 0,72760 min: 0,00024 avgzeros: 0,399 maxzeros: 2 time: 0,35433	correct: 92,85 mean: 0,01999 max: 0,27720 min: 0,00009 avgzeros: 0,029 maxzeros: 1 time: 0,08412
3	mean: 0,03719 max: 0,19223 min: 0,00076 time: 0,00856	correct: 61,8 mean: 0,02726 max: 0,74141 min: 0,00070 avgzeros: 0,405 maxzeros: 2 time: 0,35665	correct: 90,16 mean: 0,03170 max: 0,32922 min: 0,00021 avgzeros: 0,042 maxzeros: 1 time: 0,08412
4	mean: 0,05030 max: 0,37325 min: 0,00149 time: 0,00864	correct: 61,5 mean: 0,03037 max: 0,79902 min: 0,00063 avgzeros: 0,409 maxzeros: 3 time: 0,35534	correct: 87,675 mean: 0,04145 max: 0,36789 min: 0,00026 avgzeros: 0,035 maxzeros: 1 time: 0,08569
5	mean: 0,06292 max: 0,38560 min: 0,00149 time: 0,00805	correct: 64,0 mean: 0,03524 max: 0,70305 min: 0,00070 avgzeros: 0,384 maxzeros: 2 time: 0,35560	correct: 86,64 mean: 0,047954 max: 0,49523 min: 0,00009 avgzeros: 0,036 maxzeros: 1 time: 0,08420
6	mean: 0,07461 max: 0,36994 min: 0,00243 time: 0,00843	correct: 60,09 mean: 0,03270 max: 0,80875 min: 0,00101 avgzeros: 0,416 maxzeros: 2 time: 0,36124	correct: 84,08 mean: 0,06029 max: 0,53873 min: 0,00027 avgzeros: 0,055 maxzeros: 1 time: 0,08261
7	mean: 0,08868 max: 0,57795 min: 0,00245 time: 0,00864	correct: 60,09 mean: 0,03691 max: 0,69989 min: 0,00137 avgzeros: 0,424 maxzeros: 3 time: 0,36592	correct: 84,08 mean: 0,06684 max: 0,59556 min: 0,00018 avgzeros: 0,061 maxzeros: 1 time: 0,08497
8	mean: 0,09930 max: 0,49278 min: 0,00239 time: 0,00822	correct: 62,1 mean: 0,04017 max: 0,62858 min: 0,00065 avgzeros: 0,394 maxzeros: 2 time: 0,34561	correct: 82,525 mean: 0,07861 max: 0,62814 min: 0,00022 avgzeros: 0,079 maxzeros: 1 time: 0,08498

Продовження таблиці 1.1

Метод для порівняння значення середньоквадратичного відхилення	МНК	Комбінаторний МГУА	ММГУА
9	mean: 0,11358 max: 0,65100 min: 0,00334 time: 0,00838	correct: 58,5 mean: 0,04029 max: 0,68433 min: 0,00150 avgzeros: 0,431 maxzeros: 3 time: 0,34937	correct: 81,32 mean: 0,09097 max: 1,01508 min: 0,00024 avgzeros: 0,09 maxzeros: 1 time: 0,08182
10	mean: 0,12257 max: 0,61992 min: 0,00245 time: 0,00763	correct: 61,7 mean: 0,04481 max: 0,64609 min: 0,00132 avgzeros: 0,403 maxzeros: 2 time: 0,37036	correct: 80,05 mean: 0,09115 max: 0,78332 min: 0,00039 avgzeros: 0,112 maxzeros: 1 time: 0,08071

Як видно з таблиці 1.1 кращі середньозважені показники дає ММГУА, відповідно його і буде використано у якості методу знаходження оцінок коефіцієнтів БПР, які не було оцінено під час роботи декомпозиційного методу, але з використанням низки регулярних критеріїв.

1.4 Критичний аналіз підходів до розробки програмних розширень

Для реалізації програмного забезпечення автоматизації створення індивідуального алгоритму декомпозиційного методу, що входить до складу синтетичного методу, будуть розглянуті декілька варіантів програмного забезпечення із різними підходами, які дозволяють автоматизувати процес створення подібних програмних розширень.

1.4.1 Розробка програмних розширень для програмного забезпечення загального призначення

Першим буде розглянуто програмне забезпечення загального призначення Microsoft Office Excel та засіб розширення його можливостей Visual Basic for Applications (VBA) [19].

Visual Basic for Applications [19] – це мова програмування, вбудована в програми Microsoft Office, такі як: Excel, Word, Access, PowerPoint та інші. VBA

дозволяє користувачам автоматизувати рутинні завдання, створювати складні макроси, виконувати обробку даних та створювати користувацькі функції, що виходять за межі стандартних можливостей програмного забезпечення. З його допомогою можна автоматично маніпулювати даними, формувати звіти, взаємодіяти з іншими додатками та виконувати складний аналіз даних, зокрема регресійний аналіз. VBA надає широкий спектр інструментів і функцій для програмування, що робить його потужним засобом для підвищення ефективності роботи та автоматизації бізнес-процесів.

Для використання VBA в Excel необхідно активувати режим розробника. Натиснувши комбінацію клавіш ALT + F11 або на кнопку Visual Basic, відкриється вікно, як на рисунку 1.3, з можливістю написання макросів.

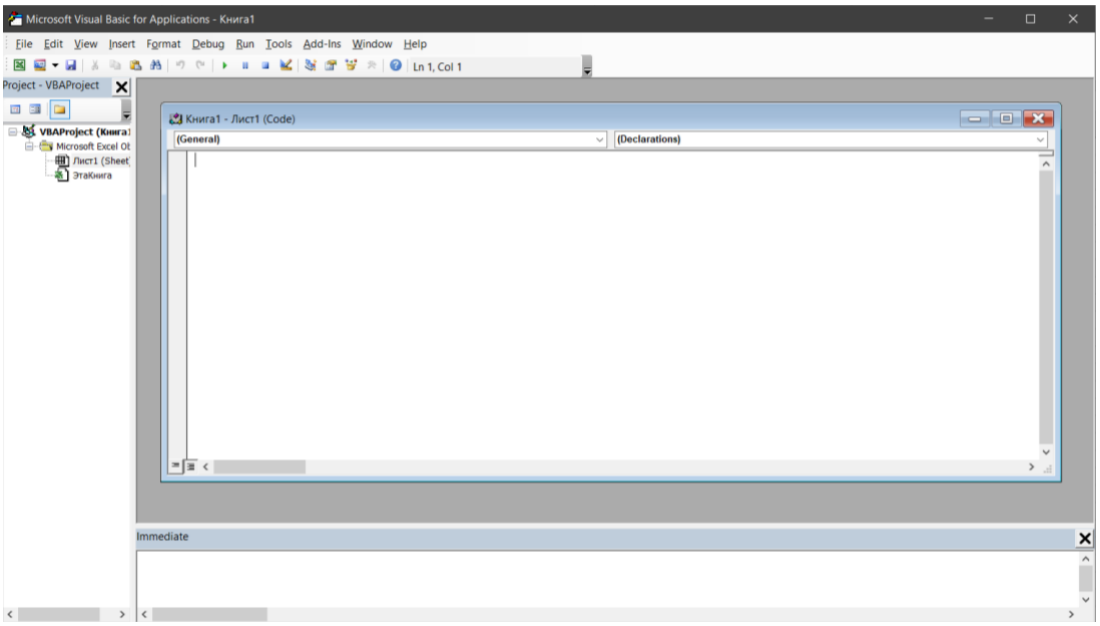


Рисунок 1.3 – Редактор для роботи з VBA в Excel

Синтаксис VBA достатньо простий і схожий на більшість мов програмування, підтримує різні типи даних, такі як Integer, Long, Double, String, Boolean, Variant та інші. На рисунках 1.4-1.6 зображено синтаксис базових конструкцій.

```
Dim i As Integer
Dim s As String
Dim result As Double

i = 5
s = "Hello, VBA!"
result = i * 2.5
```

Рисунок 1.4 – Приклад використання змінних в VBA

```
Dim score As Integer
score = 85

If score >= 90 Then
    MsgBox "Excellent"
ElseIf score >= 75 Then
    MsgBox "Good"
Else
    MsgBox "Needs Improvement"
End If
```

Рисунок 1.5 – Приклад використання умовних операторів в VBA

```
Dim i As Integer
For i = 1 To 10
    Cells(i, 1).Value = i * 10
Next i
```

Рисунок 1.6 – Приклад використання циклів в VBA

Окрім базових конструкцій існують ключові функції. Об’єкт Range є одним з тих, що використовується найчастіше в VBA для роботи з комірками і діапазонами даних в Excel. Він дозволяє читати, записувати, форматувати та маніпулювати даними. Приклад та результат виконання можна побачити на рисунку 1.7.

```
Sub Example()
    Dim rng As Range
    Set rng = Range("A1:A10")

    ' Зміна значення комірок в діапазоні
    rng.Value = "New Value"

    ' Форматування комірок
    rng.Font.Bold = True
    rng.Interior.Color = RGB(255, 255, 0)
End Sub
```

	A
1	New Value
2	New Value
3	New Value
4	New Value
5	New Value
6	New Value
7	New Value
8	New Value
9	New Value
10	New Value

Рисунок 1.7 – Приклад використання Range в VBA

Об’єкт Cells використовується для звернення до конкретних комірок за їхніми номерами рядка і стовпця. Приклад та результат виконання можна побачити на рисунку 1.8.

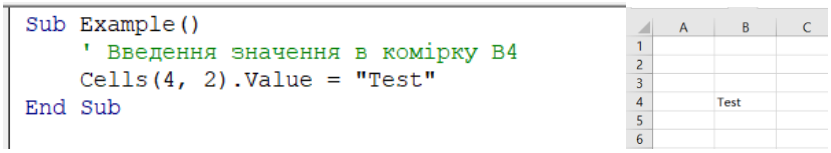


Рисунок 1.8 – Приклад використання Cells в VBA

Також є можливість використання основних математичних функцій: Abs, Sqr, Exp, Log, Round та інші. Для складніших операції при регресійному аналізі доречно використовувати «Analysis ToolPak».

Далі буде розглянуто як, за допомогою VBA, можна створювати індивідуальні алгоритми для MS Excel, на прикладі лінійної регресії.

Найпростішим прикладом регресійного аналізу є побудова лінійної регресії. Нехай задано значення для двох змінних X та Y в діапазонах A2:A11 та B2:B11 відповідно та створено відповідні змінні, які відображають діапазони значень, на рисунку 1.9.

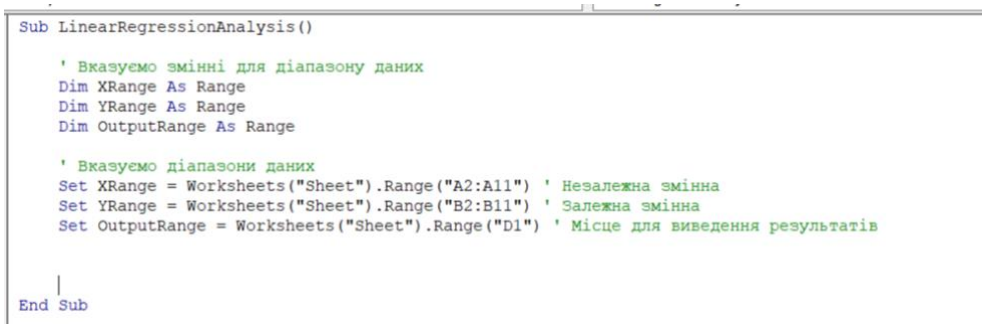


Рисунок 1.9 – Створення змінних та задання діапазону значень

Команда Application.Run у VBA використовується для виклику макросу або функції, яка зберігається в надбудові Excel. У прикладі на рисунку 1.10

Application.Run використовується для запуску функції «Regress» з надбудови «Analysis ToolPak» (ATPVBAEN.XLAM) для проведення регресійного аналізу.

```
Sub LinearRegressionAnalysis()

    ' Вказуємо змінні для діапазону даних
    Dim XRange As Range
    Dim YRange As Range
    Dim OutputRange As Range

    ' Вказуємо діапазони даних
    Set XRange = Worksheets("Sheet").Range("A2:A11") ' Незалежна змінна
    Set YRange = Worksheets("Sheet").Range("B2:B11") ' Залежна змінна
    Set OutputRange = Worksheets("Sheet").Range("D1") ' Місце для виведення результатів


    ' Використовуємо функцію Regression з надбудови Analysis ToolPak
    Application.Run "ATPVBAEN.XLAM!Regress", YRange, XRange, OutputRange, True, True, OutputRange, , , , False


    MsgBox "Регресійний аналіз завершено! Результати знаходяться у діапазоні " & OutputRange.Address

End Sub
```

Рисунок 1.10 – Виклик функції для проведення регресійного аналізу

Значення кожного параметру функції ATPVBAEN.XLAM!Regress:

– $YRange$ – діапазон, що містить дані залежної змінної (значення Y);

– $XRange$ – діапазон, що містить дані незалежної змінної (значення X);

– $OutputRange$ – параметр вказує, де почнеться виведення результатів регресійного аналізу на робочому аркуші;

– $True$ – аргумент вказує, чи включати мітки (заголовки) у вивід;

– $True$ – аргумент визначає, чи показувати додаткові статистичні дані регресії, такі як коефіцієнт детермінації (R-квадрат), стандартну похибку та інші метрики;

– $OutputRange$ (повторно) – місце, куди можуть бути виведені залишкові графіки та інші діаграми, якщо це вказано. У цьому випадку знову встановлено початковий $OutputRange$, що вказує, що жодного окремого місця для цих додаткових графіків не вказано;

– *(пусто)* – параметр залишений порожнім (, ,). Даний параметр призначений для «Нової книги» (New Workbook) для виведення;

– *(пусто)* – порожній параметр, який вказує на відсутність певних налаштувань для довірчих інтервалів для залишків;

– *(пусто)* – параметр також залишений порожнім і використовується для вказівки, чи потрібно зберігати результати на новому аркуші. Порожнє значення означає, що вивід буде зроблено на поточному аркуші;

– $False$ – параметр визначає, чи включати константу в рівняння регресії. Встановлення його на $False$ включає константу (перехоплення) в модель регресії.

У підсумку команда Application.Run налаштована для виконання лінійного регресійного аналізу з використанням зазначених діапазонів для залежних ($YRange$) та незалежних ($XRange$) змінних. Вона розміщує вивід у зазначеному $OutputRange$ на поточному робочому аркуші, включає мітки та додаткові статистичні дані, і уникає створення нових книг або аркушів для виводу. Результати використання функції відображені на рисунку 1.11.

X	Y
2	8
4	7
1	6
8	0
9	3
3	6
8	8
10	9
3	13
0	5

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0,17828
R Square	0,031784
Adjusted R Square	-0,10653
Standard Error	3,864996
Observations	9

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	1	3,432653	3,432653061	0,22979039	0,646293147
Residual	7	104,5673	14,93819242		
Total	8	108			

	Coefficients	*tandard Erro*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	7,240816	2,289893	3,162076139	0,015882265	1,826079573	12,65555308	1,826079573	12,65555
2	-0,17755	0,370388	-0,479364569	0,646293147	-1,053380141	0,6982781	-1,053380141	0,698278

Рисунок 1.11 – Результат виконання функції ATPVBAEN.XLAM!Regress

До програмного забезпечення загального призначення, що дозволяє розв’язувати задачі регресійного аналізу та підтримує розробку зовнішніх програмних розширень, можна також віднести: Google Sheets, LibreOffice, Polaris office, OpenOffice, Zoho Sheet [20]. У вище зазначеному програмному забезпеченні, підхід до розробки програмних розширень аналогічний MS Excel.

Недоліком даного рішення при створенні програмних розширень (у даному випадку індивідуальних алгоритмів декомпозиційного методу побудови

багатовимірних поліноміальних регресій) є те, що їх можна використовувати лише в самому програмному забезпеченні, а вбудова з стороннім програмним забезпеченням буде вкрай складною.

1.4.2 Розробка програмних розширень для програмного забезпечення вузькоцільового призначення

Другим буде розглянуто рішення, яке розроблене для спеціалізованого статистичного програмного пакету загального призначення Statistica [21], а саме одна з його версій Statistica Basic [21], яка надає інструменти, у тому числі, для підключення власноруч розроблених алгоритмів і використання Statistica Basic у якості графічного інтерфейсу користувача.

Statistica – це потужний набір програм для статистичного аналізу даних, який включає в себе широкий спектр інструментів для проведення різних видів аналізу, візуалізації даних та побудови прогнозів [21]. Однією з основних версій цього пакета є Statistica Basic, яка орієнтована на базовий статистичний аналіз і є зручним інструментом для користувачів з різним рівнем знань у галузі статистики [22].

Statistica Basic надає користувачам базовий набір інструментів для проведення статистичного аналізу, що робить його ідеальним вибором для простих і середньоскладних завдань [23]. Цей програмний продукт дозволяє виконувати основні види аналізу.

Описова статистика є фундаментом будь-якого статистичного аналізу, і Statistica Basic пропонує широкий спектр інструментів для її виконання. Користувачі можуть розраховувати основні показники центральної тенденції (середнє значення, медіана, мода) та розсіювання (діапазон, стандартне відхилення, дисперсія). Програмне забезпечення також дозволяє створювати детальні звіти, що містять основні характеристики даних, на основі яких можна робити попередні висновки.

Statistica Basic включає можливості для проведення кореляційного аналізу, який дозволяє визначати ступінь взаємозв'язку між двома змінними. Програмне забезпечення пропонує різні типи кореляційних коефіцієнтів, таких як коефіцієнт Пірсона або Спірмена, що дозволяє досліджувати як лінійні, так і нелінійні зв'язки. Результати кореляційного аналізу можуть бути представлені у вигляді матриць кореляції або графіків розсіювання, що значно полегшує їх інтерпретацію.

Ще однією важливою функцією Statistica Basic є можливість проведення регресійного аналізу, який дозволяє моделювати залежність однієї змінної від інших. Програмне забезпечення підтримує як одновимірну, так і багатовимірну лінійну регресію, що дає змогу будувати моделі, які враховують вплив декількох незалежних змінних. Отримані результати регресійного аналізу можна використовувати для прогнозування та ухвалення рішень.

Окрім числових розрахунків, Statistica Basic пропонує потужні інструменти для візуалізації даних. Користувачі можуть створювати різноманітні типи графіків, включаючи гістограми, діаграми розсіювання та інші. Ці графіки допомагають візуалізувати дані, виявляти тренди та аномалії, а також спрощують представлення результатів аналізу іншим учасникам процесу.

Statistica Basic також забезпечує можливість проведення різних видів тестів для перевірки статистичних гіпотез. Серед них t-тести для порівняння середніх значень, аналіз варіацій для оцінки різниць між групами та інші. Ці інструменти дозволяють користувачам робити обґрунтовані висновки на основі наявних даних.

Однією з ключових характеристик Statistica Basic є його інтуїтивно зрозумілий інтерфейс (рисунок 1.12), який робить програму доступною для широкого кола користувачів, незалежно від рівня їхніх знань у статистиці. Інтерфейс програмного забезпечення ретельно продуманий, щоб забезпечити легкий доступ до основних функцій та інструментів [22].

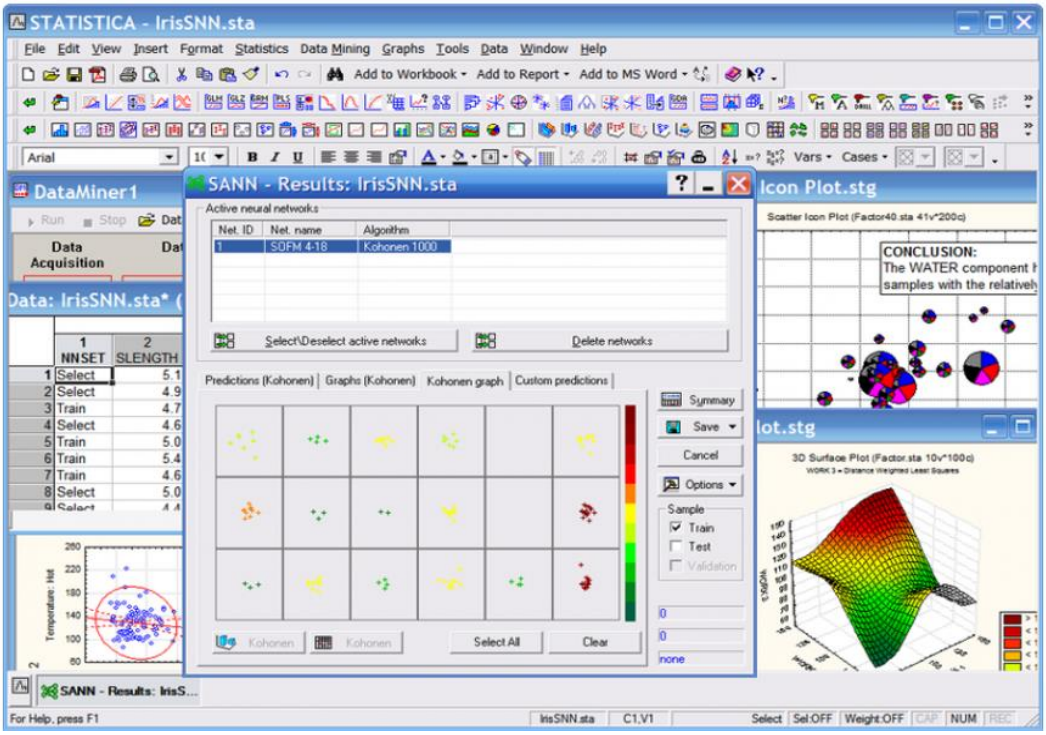


Рисунок 1.12 – Інтерфейс програмного забезпечення Statistica [22]

Інтерфейс Statistica Basic організований таким чином, щоб користувачі могли швидко знаходити необхідні інструменти для виконання статистичного аналізу. Основне робоче вікно поділене на кілька ключових областей.

Головне меню та панель інструментів розташовані у верхній частині вікна і забезпечують швидкий доступ до основних функцій програми, таких як відкриття, збереження файлів, проведення аналізу та побудова графіків.

Область даних, яку користувачі можуть завантажувати та переглядати свої дані у вигляді таблиць. Область даних підтримує різні формати файлів і дозволяє зручно працювати з великими наборами даних.

Панель результатів є результатами аналізу, таких як статистичні звіти та графіки, які відображаються на окремій панелі, що дозволяє користувачам швидко оцінювати отримані дані.

Налаштування та параметри являють собою область, що дозволяє користувачам налаштовувати параметри аналізу, вибирати статистичні методи та формати виведення результатів.

Програма підтримує зручну навігацію через чітко організовані меню та панелі інструментів [3]. Крім того, користувачі можуть налаштовувати інтерфейс відповідно до своїх потреб, що підвищує ефективність роботи.

Також Statistica Basic пропонує потужні можливості вбудови з зовнішніми програмами та скриптами, що робить її гнучким інструментом для проведення складного аналізу даних. Однією з ключових особливостей є можливість підключення до Statistica зовнішніх програм і використання алгоритмів, написаних на різних мовах програмування. Користувачі можуть викликати зовнішні додатки або функції безпосередньо з інтерфейсу Statistica, що особливо корисно для виконання специфічних завдань або обчислень, які виходять за рамки стандартних функцій програми.

Component Object Model (COM) [21] – це стандарт, що дозволяє різним програмам взаємодіяти між собою, обмінюючись даними та викликаючи функції. Statistica Basic підтримує COM-інтерфейс, що дозволяє використовувати зовнішні програми та бібліотеки для виконання додаткових обчислень або обробки даних.

Для підключення зовнішньої програми, спочатку необхідно створити або використовувати готовий алгоритм, написаний на одній із мов програмування, що підтримуються, наприклад, C, C++, Java. Цей алгоритм має бути оформлений у вигляді виконуваного файлу (.exe), бібліотеки, що динамічно підключена (.dll), або скрипта.

Statistica Basic може передавати у зовнішні програми параметри командного рядка, які будуть використані для виконання певних обчислень. Це дозволяє налаштувати специфічні режими роботи зовнішніх програм або обробляти конкретні набори даних. Наприклад, якщо алгоритм потребує обробки файлу даних, його ім’я може бути передано через параметри командного рядка.

Після виконання зовнішньої програми результати можуть бути повернені в Statistica Basic у вигляді вихідних даних, які можуть бути проаналізовані або використані для побудови звітів.

Аналогами програмного забезпечення вузькоцільового призначення, що спеціалізується на розв’язанні прикладних математичних задач (у тому числі задач регресійного аналізу), також є: SPSS Statistics, Mathcad, Matlab [24].

Як і у попередньому випадку, недоліком даного рішення при створенні програмних розширень (у даному випадку індивідуальних алгоритмів декомпозиційного методу побудови багатовимірних поліноміальних регресій) є складність вбудови із сторонніми програмними засобами, фактично розглянуте програмне забезпечення просто дозволяє інтегрувати зовнішні бібліотеки (скрипти) з одним або декількома користувацькими алгоритмами і використовувати власний графічний інтерфейс для візуалізації даних.

1.4.3 Розробка програмних розширень для мов програмування загального призначення

Далі буде розглянуто рішення, яке дозволить вбудовувати власні алгоритми у цільове програмне забезпечення без використання додаткових спеціалізованих програмних засобів. Якщо стоїть задача вбудови будь-якого алгоритму у цільове програмне забезпечення, більш доцільно представити даний алгоритм у складі деякої програмної бібліотеки для обраної мови програмування.

Буде проаналізовано кілька прикладів подібних бібліотек для мови програмування Python [25].

Python [25] – популярна мова програмування, основною перевагою якої є простота. Завдяки ній регресійний аналіз стає все більш доступним і зручним для дослідників та аналітиків. Засоби машинного навчання у Python стрімко розвиваються, тому існує безліч бібліотек, що містять в собі інструменти

регресійного аналізу, які дозволять будувати як прості лінійні моделі, так і складні нелінійні методи.

NumPy

NumPy [26] є базовою бібліотекою для наукових обчислень у Python. Вона забезпечує підтримку багатовимірних масивів та матриць, а також має велику кількість функцій для виконання різноманітних математичних операцій, у тому числі пов'язаних з задачами регресійного аналізу.

Функції NumPy дозволяють створити як лінійні, так і поліноміальні регресії, проте, так як ця бібліотека не створена саме для регресійного аналізу, її інструменти є досить поверхневими.

NumPy використовується в основі багатьох інших бібліотек, таких як Scikit-learn та Pandas.

Нижче наведено приклад, що демонструє, як можна обчислити коефіцієнти лінійної регресії з використанням NumPy (рисунок 1.13).

```
import numpy as np

# Дані
X = np.array([1, 2, 3, 4, 5])
y = np.array([1, 3, 2, 3, 5])

# Обчислення коефіцієнтів лінійної регресії
A = np.vstack([X, np.ones(len(X))]).T
a, b = np.linalg.lstsq(A, y, rcond=None)[0]

# Виведення коефіцієнтів
print(f'Коефіцієнти: a = {a:.1f}, b = {b:.1f}')

# Значення y із знайденими коефіцієнтами
y_pred = a * X + b

print(f'Реальні значення y: {y}')
print(f'Значення y, передбачені із знайденими коефіцієнтами: {y_pred}')
```

```
Коефіцієнти: a = 0.8, b = 0.4
Реальні значення y: [1 3 2 3 5]
Значення y, передбачені із знайденими коефіцієнтами: [1.2 2.  2.8 3.6 4.4]
```

Рисунок 1.13 – Застосування NumPy для знаходження коефіцієнтів лінійної регресії

Pandas

Pandas [27] надає потужні інструменти для роботи з табличними даними, такими як DataFrame. Ця бібліотека дозволяє легко маніпулювати, фільтрувати та аналізувати великі обсяги даних, що є необхідним етапом перед регресійним аналізом.

На рисунку 1.14 показано приклад застосування Pandas для організації та маніпуляції даними перед регресійним аналізом.

```
import pandas as pd

# Створення DataFrame
data = {'Feature1': [1, 2, 3, 4, 5],
        'Feature2': [2, 3, 4, 5, 6],
        'Target': [1, 3, 2, 3, 5]}

df = pd.DataFrame(data)

# Виділення змінних
X = df[['Feature1', 'Feature2']]
y = df['Target']

print(X.head())
print(y.head())
```

```
   Feature1  Feature2
0         1         2
1         2         3
2         3         4
3         4         5
4         5         6
0    1
1    3
2    2
3    3
4    5
Name: Target, dtype: int64
```

Рисунок 1.14 – Застосування Pandas для роботи з DataFrame

SciPy

SciPy [28] розширює функціональність NumPy, надаючи додаткові інструменти для обробки масивів та алгоритми для оптимізації, інтеграції, інтерполяції, алгебраїчних рівнянь, статистики та багатьох інших класів задач. Вона є важливим інструментом для проведення більш складних математичних обчислень, що є необхідними для регресійного аналізу.

Ця бібліотека покриває досить багато потреб поліноміальних регресій. Її функції дозволяють, як створювати степеневі моделі, так і налаштовувати їх, оцінювати їх параметри.

Застосування SciPy для поліноміальної регресії та її візуалізація за допомогою Matplotlib [29] продемонстровані на рисунку 1.15.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# Дані
X = np.array([1, 2, 3, 4, 5])
y = np.array([1, 4, 9, 16, 25])

# Функція для поліноміальної регресії
def poly(x, a, b, c):
    return a * x**2 + b * x + c

# Обчислення коефіцієнтів
popt, _ = curve_fit(poly, X, y)

# Прогнозування
y_pred = poly(X, *popt)

# Візуалізація
plt.scatter(X, y)
plt.plot(X, y_pred, color='red')
plt.show()
```

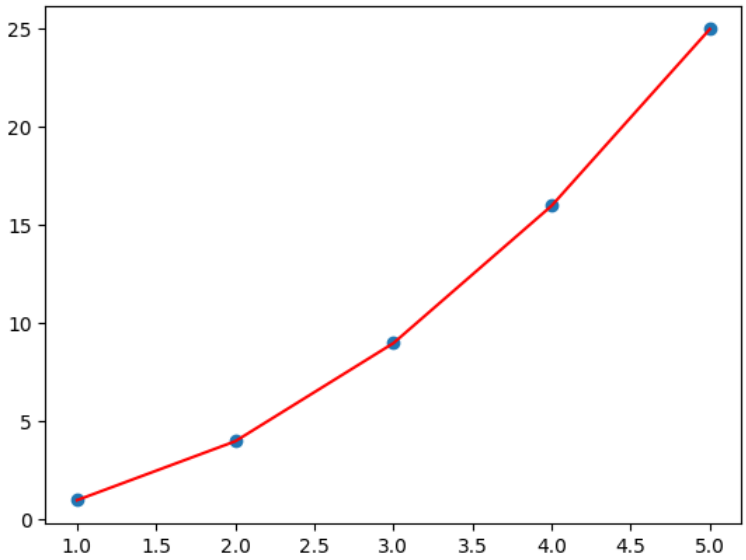


Рисунок 1.15 – Застосування SciPy для поліноміальної регресії

Matplotlib та Seaborn

Matplotlib [29] є базовою бібліотекою для візуалізації даних у Python, тоді як Seaborn [30] побудована на основі Matplotlib і надає простіший інтерфейс для

створення статистичних графіків. За допомогою цих бібліотек можна візуалізувати дані та результати регресійного аналізу, що є важливою частиною процесу дослідження.

Matplotlib не підтримує поліноміальні регресії напряму, проте завдяки ній можна візуалізувати ті поліноміальні моделі, що створені завдяки інших бібліотек. Seaborn, в свою чергу, має інструменти для візуалізації саме поліноміальних регресій, так як можна вказати параметри моделі, що є зручним у роботі з ними.

Використання Matplotlib для візуалізації поліноміальної регресії зображено на рисунку 1.15. Приклад використання Seaborn для побудови графіка регресії наведено на рисунку 1.16.

```
import seaborn as sns
import numpy as np
import pandas as pd

# Дані
data = pd.DataFrame({
    'X': [1, 2, 3, 4, 5],
    'y': [1, 3, 2, 3, 5]
})

# Побудова графіка регресії
sns.lmplot(x='X', y='y', data=data)
plt.show()
```

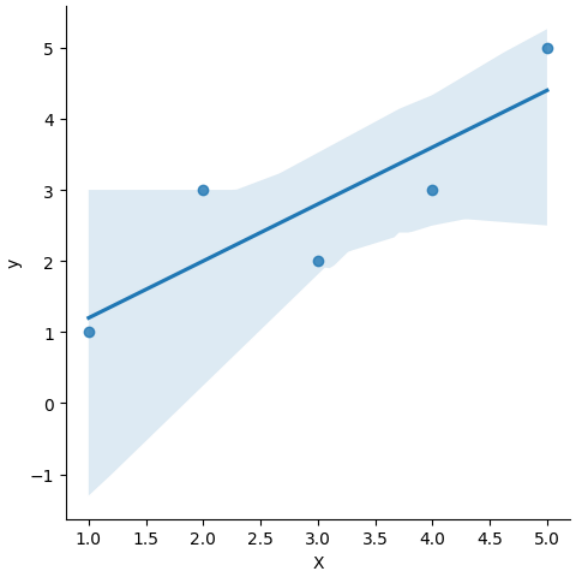


Рисунок 1.16 – Використання Seaborn для візуалізації графіка регресії

Scikit-learn

Scikit-learn [31] – це одна з найпопулярніших бібліотек для машинного навчання в Python. Вона містить прості у використанні інструменти для побудови моделей прогнозування. Серед її інструментів є численні алгоритми для регресійного аналізу, включаючи лінійну, поліноміальну, логістичну регресію та інші методи. Scikit-learn також забезпечує інструменти для попередньої обробки даних, кросвалідації та оптимізації моделей.

Для поліноміальних моделей ця бібліотека має безліч інструментів, що покривають потреби регресійного аналізу. Є функції для створення, тренування, оцінки моделей поліноміальних регресій.

Приклад роботи із багатовимірною лінійною регресією за допомогою Scikit-learn зображено на рисунку 1.17.

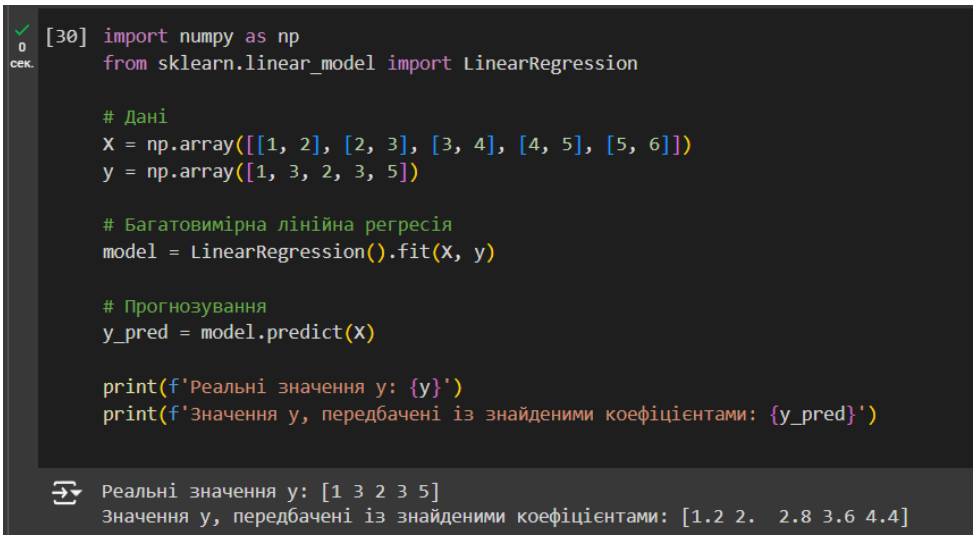


Рисунок 1.17 – Робота з багатовимірною лінійною регресією за допомогою Scikit-learn

Statsmodels

Statsmodels [32] – це бібліотека для класичних статистичних методів у Python. Вона пропонує розширений набір інструментів для регресійного аналізу, таких як лінійна та логістична регресія, а також моделі часових рядів. Statsmodels дозволяє

отримати більш детальну статистичну інформацію про моделі, ніж Scikit-learn, що робить її ідеальною для дослідницьких цілей.

Statsmodels підтримує поліноміальні регресії, надаючи більше можливостей для оцінки моделей. З її функціями можна отримати таку інформацію про модель, як її коефіцієнти, значущість параметрів, коефіцієнт детермінації тощо.

На рисунку 1.18 зображено, як Statsmodels відображає детальну статистичну інформацію про лінійну регерсію.

```python
import numpy as np
import statsmodels.api as sm

# Дані
X = np.array([1, 2, 3, 4, 5])
y = np.array([1, 3, 2, 3, 5])

# Додавання константи (для перехоплення)
X = sm.add_constant(X)

# Лінійна регресія
model = sm.OLS(y, X).fit()

# Результати
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.727
Model:                            OLS   Adj. R-squared:                  0.636
Method:                 Least Squares   F-statistic:                     8.000
Date:                Sat, 17 Aug 2024   Prob (F-statistic):             0.0663
Time:                        13:18:30   Log-Likelihood:                -5.2598
No. Observations:                   5   AIC:                             14.52
Df Residuals:                       3   BIC:                             13.74
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.4000      0.938      0.426      0.699      -2.585       3.385
x1             0.8000      0.283      2.828      0.066      -0.100       1.700
==============================================================================
Omnibus:                        nan   Durbin-Watson:                   2.567
Prob(Omnibus):                  nan   Jarque-Bera (JB):                0.570
Skew:                         0.289   Prob(JB):                        0.752
Kurtosis:                     1.450   Cond. No.                         8.37
==============================================================================
```

Рисунок 1.18 – Оцінка параметрів лінійної регресії за допомогою Statsmodels

Якщо узагальнити, то дані рішення є універсальним підходом для вбудови власних алгоритмів у цільове програмне забезпечення для будь-якої мови програмування. Однак, індивідуальний алгоритм декомпозиційного методу

залежить від умов задачі побудови БПР та не може бути однозначно формалізованим і включеним у деяку програмну бібліотеку. Відповідно, необхідно створити вузькоспеціалізовану мову скриптів та реалізувати програмні засоби автоматизації створення індивідуальних алгоритмів декомпозиційного методу, як програмне розширення для обраної мови програмування, яке буде включено до відповідної програмної бібліотеки. Для спрощення створення індивідуальних алгоритмів окремі обчислювальні процедури, що входять до складу цих алгоритмів, варто представити у вигляді макросів, які використовуються у засобах розширення можливостей програмного забезпечення загального призначення.

1.5 Постановка задачі дослідження

Проведений вище критичний аналіз теоретичних можливостей декомпозиційного методу, що входить до складу синтетичного методу побудови БПР, заданих надлишковим описом, показав, що він гарантує формальну процедуру послідовної оцінки коефіцієнтів при нелінійних членах БПР, при чому оцінки коефіцієнтів на попередніх кроках формального алгоритму використовуються на наступних кроках цього алгоритму, звідки випливає основний недолік теоретично обґрунтованої процедури декомпозиційного методу, а саме, якщо на деяких кроках алгоритму оцінки коефіцієнтів при нелінійних членах БПР не відповідають заданій точності (по величині дисперсії), то автоматично наступні коефіцієнти знаходяться з недопустимою точністю по величині дисперсії. Проведений аналіз розв’язання низки конкретних прикладів показав, що конкретна структура БПР, задана надлишковим описом, дозволяє в багатьох випадках створити індивідуальний алгоритм оцінки невідомих коефіцієнтів при нелінійних членах БПР з заданою точністю, що якісно є більш ефективною, ніж формальна теоретично обґрунтована процедура декомпозиційного методу. Звідки виникає проблема розробки теоретичного обґрунтування можливості створення індивідуального алгоритму

декомпозиційного методу, що випливає з аналізу користувачем структури надлишкового опису БПР, а також розробка компонентів програмного забезпечення, що дозволяє користувачу ефективно у взаємодії з засобами розробки створювати індивідуальний алгоритм і можливості одночасного автоматизованого створення його програмного забезпечення. Так як не всі коефіцієнти БПР з заданою точністю знаходяться індивідуальним алгоритмом декомпозиційного методу, то в синтетичному методі їх оцінки знаходяться реалізацією ММГУА, відповідно, підвищення ефективності ММГУА також підвищує ефективність синтетичного методу в цілому.

Тому *мета дослідження* – підвищення ефективності програмного забезпечення синтетичного методу побудови багатовимірних поліноміальних регресій, заданих надлишковим описом, шляхом розробки нових компонентів програмного забезпечення, що реалізують теоретично обґрунтовану методологію створення індивідуальних алгоритмів декомпозиційного методу, та підвищення ефективності модифікованого методу групового урахування аргументів, завдяки одночасному використанню низки регулярних критеріїв.

Для досягнення вказаної мети потрібно вирішити такі *завдання*:

– дослідити теоретичні властивості декомпозиційного методу, що входить до складу синтетичного методу, та забезпечити можливість підвищення його ефективності, шляхом розробки нової теоретично обґрунтованої методології створення індивідуальних алгоритмів побудови багатовимірних поліноміальних регресій, заданих надлишковим описом;

– створити підхід, який дозволяє підвищити ефективність модифікованого методу групового урахування аргументів, що входить до складу синтетичного методу, шляхом одночасного використання низки регулярних критеріїв;

– обґрунтувати можливість використання алгоритму статичного управління паралельними асинхронними обчислювальними процесами для покращення

швидкодії паралельної реалізації модифікованого методу групового урахування аргументів, шляхом знаходження асимптотичної складності його підалгоритмів;

– дослідити способи розширення можливостей мов програмування загального призначення та універсальних програмних засобів для автоматизації процесу створення і програмної реалізації індивідуальних алгоритмів побудови багатовимірних поліноміальних регресій, заданих надлишковим описом, на основі положень синтетичного методу та запропонованих нових теоретично обґрунтованих методологій;

– на основі проведеного аналізу розробити методологію розширення можливостей обраної мови програмування загального призначення для автоматизації процесу створення та програмної реалізації індивідуальних алгоритмів побудови багатовимірних поліноміальних регресій, заданих надлишковим описом;

– модифікувати архітектуру програмної бібліотеки, яка реалізує синтетичний метод побудови багатовимірних поліноміальних регресій, заданих надлишковим описом, шляхом включення в себе нових складових, що забезпечать підвищення ефективності синтетичного методу;

– виконати статистичне дослідження ефективності нової версії модифікованого методу групового урахування аргументів;

– виконати дослідження ефективності в цілому програмної бібліотеки побудови багатовимірних поліноміальних регресій, заданих надлишковим описом.

Висновки до розділу

В рамках першого розділу було виконано детальний опис предметної області проблематики розв’язання задач побудови багатовимірних поліноміальних регресій, заданих надлишковим описом, на основі активного та пасивного експериментів.

Були розглянуті загальні універсальні методи побудови багатовимірних поліноміальних регресій, у тому числі і синтетичний метод. Синтетичний метод побудови багатовимірних поліноміальних регресій виявився корисним з погляду можливості знаходити оцінки коефіцієнтів з наперед заданою точністю шляхом наявності у його складі декомпозиційного методу. Основним недоліком декомпозиційного методу у складі синтетичного є відсутність формалізованого алгоритму знаходження оцінок коефіцієнтів БПР з заданою точністю, а наведена лише загальна методологія, що не є універсальною. У синтетичному методі описана необхідність створення програмних засобів побудови індивідуальних алгоритмів декомпозиційного методу. Модифікований метод групового урахування аргументів із складу синтетичного також містить недолік пов’язаний з використанням лише одного регулярного критерію при визначенні кінцевої структури БПР.

Критичний аналіз програмних засобів розв’язання задач побудови БПР показав, що жоден з існуючих не може забезпечити реалізацію індивідуального алгоритму декомпозиційного методу, однак можна використати комбінацію підходів розширення можливостей мов програмування загального призначення та програмних засобів загального призначення.

У результаті була поставлена задача розробки методології та програмних засобів підвищення ефективності синтетичного методу побудови багатовимірних поліноміальних регресій.

2 ТЕОРЕТИЧНІ ТА МЕТОДОЛОГІЧНІ ОСНОВИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ СИНТЕТИЧНОГО МЕТОДУ З ВИКОРИСТАННЯМ НОВИХ ОПЕРАТОРІВ, ЩО Є РОЗШИРЕННЯМ УНІВЕРСАЛЬНОЇ МОВИ ПРОГРАМУВАННЯ

2.1 Модифікація декомпозиційного методу побудови БПР, лінійної відносно невідомих коефіцієнтів

Викладається, відповідно до [33].

У підрозділі 1.2 було наведено аналіз теоретичних можливостей синтетичного методу задачі побудови БПР, заданих надлишковим описом (1.9). Зокрема, був розглянутий декомпозиційний метод знаходження оцінок коефіцієнтів БПР з наперед заданою точністю як складова синтетичного методу. Описано перший та другий підалгоритми декомпозиційного методу, а також можливість побудови на їх основі індивідуального алгоритму декомпозиційного методу окремо під кожну задачу. Було поставлено задачу розробити теоретичну основу та програмні засоби автоматизації процесу реалізації індивідуального алгоритму декомопозиційного методу.

2.1.1 Недоліки використання першого та другого підалгоритму декомпозиційного методу

Теоретичне обґрунтування декомпозиційного методу (див. 1.2) гарантує, що послідовне використання першого та другого підалгоритму на попередньому етапі аналізу задачі приводить до гарантованого отримання оцінок всіх коефіцієнтів при нелінійних членах БПР. Але при обмеженій заздалегідь кількості експериментів задана точність (за величиною дисперсії) не гарантується. При знаходженні оцінок наступних коефіцієнтів використовуються оцінки раніше знайдених коефіцієнтів при нелінійних членах БПР. Таким чином, отриманий результат в цілому може бути незадовільним. Тому, при розв’язанні практичних задач, алгоритм

декомпозиційного методу припиняє роботу на першому нелінійному члені БПР, оцінка коефіцієнта якого на етапі попереднього аналізу за дисперсією суттєво відрізняється від заданої.

2.1.2 Теоретичні основи чотирьох узагальнених операторів

У даному пункті викладаються чотири різні теоретично обґрунтовані способи (агреговані оператори) побудови ОПР для оцінки коефіцієнтів при нелінійних членах БПР на етапі попереднього аналізу індивідуального алгоритму декомпозиційного методу для задачі побудови БПР (1.9). Послідовність їх використання задається користувачем внаслідок аналізу ним структури БПР. В результаті реалізації попереднього етапу, індивідуальний алгоритм (другий етап) стає повністю формалізованим, а саме, задається послідовність проведення активних експериментів (з відомими значеннями вхідних змінних), наслідком кожного з яких є оцінка коефіцієнта при найвищому степені спеціальним чином сконструйованої ОПР, що дозволяє оцінити з заданою точністю коефіцієнт при відповідному нелінійному члені БПР.

Перша людино-комп'ютерна алгоритмічна процедура (перший агрегований оператор (АО))

Користувач вибирає нелінійний член БПР з (1.9):

$$b_{i_1\dots i_t}^{j_1\dots j_t}\left(x_{i_1}\right)^{j_1}\cdots\left(x_{i_k}\right)^{j_k}\cdots\left(x_{i_t}\right)^{j_t}, \qquad (2.1)$$

який містить вибрану ним вхідну змінну x_{i_k} в степені $j_k \geq 2$. Алгоритмічно проводиться виконання перевірки наступної умови: не повинно існувати жодного нелінійного члена БПР, що містить $\left(x_{i_k}\right)^{j_l}$ $\forall j_l \geq j_k$. Якщо такі члени присутні, то коефіцієнти при них з заданою точністю будуть знайдені внаслідок реалізації активного експерименту на попередніх кроках другого етапу індивідуального алгоритму. В основному активному експерименті змінна x_{i_k} буде приймати значення $x_{i_k,i} = a_{i_k}z_i + b_{i_k}$, $i = \overline{1,n}$, де коефіцієнти a_{i_k}, b_{i_k} та значення z_i, $i = \overline{1,n}$, вхідної

віртуальної скалярної змінної z спеціальним чином сконструйованої віртуальної ОПР [14] задаються формулами (7), (8) [14]. Інші вхідні змінні x_i, $i = \overline{1,m}$, $i \neq i_k$, в активному експерименті приймають фіксовані значення, причому вхідні змінні, що входять в (2.1), приймають максимально великі по модулю значення (всі інші – мінімально малі по модулю значення). При цьому, відповідно до [14], статистичні дані результатів віртуального активного експерименту задаються в вигляді:

$$\left(z_i \to \bar{y}_i, \bar{y}_i = y_i - \sum b_{l_1 \dots l_{t_l}}^{j_{l_1} \dots j_{l_{t_l}}} \left(x_{l_1,i}\right)^{j_{l_1}} \cdots \left(x_{l_{t_l},i}\right)^{j_{l_{t_l}}}\right), \tag{2.2}$$

де коефіцієнти $b_{l_1 \dots l_{t_l}}^{j_{l_1} \dots j_{l_{t_l}}}$ з заданою точністю були знайдені на попередніх кроках другого етапу індивідуального алгоритму. В цьому випадку, відповідно до [14], дисперсія оцінки коефіцієнта $b_{i_1 \dots i_t}^{j_1 \dots j_t}$ (див. (2.1)) має вигляд:

$$Db_{i_1 \dots i_t}^{j_1 \dots j_t} = \frac{1}{\left(a_{i_k}\right)^{2j_k}} \cdot \frac{1}{\prod_{l=1,\, l \neq i_k}^{t} \left(x_{i_l,\phi}\right)^{2j_l}} \cdot D\hat{\gamma}_{j_k}, \tag{2.3}$$

де, відповідно до [14], $\hat{\gamma}_{j_k}$ є оцінка коефіцієнта при z в максимальному степені j_k віртуальної ОПР (для спрощення, далі через $\hat{\gamma}_{j_k}$ позначається як випадкова величина, так і її реалізація).

Як показано в [14], для віртуальної змінної z, що у віртуальному активному експерименті приймає значення $z_1 < z_2 < \cdots < z_n$, де $\forall i \; z_i - z_{i-1} = \text{const}$, $n = 10$, $z_1 = -50$, $z_{10} = 50$. Дисперсії оцінок коефіцієнтів віртуальної ОПР дорівнюють:

$$D\hat{\gamma}_2 = 1{,}7 \cdot 10^{-6}\sigma^2, D\hat{\gamma}_3 = 4{,}7 \cdot 10^{-9}\sigma^2,$$

$$D\hat{\gamma}_4 = 2{,}6 \cdot 10^{-13}\sigma^2, D\hat{\gamma}_5 = 4{,}5 \cdot 10^{-16}\sigma^2,$$

тобто дисперсія кожного наступного члена зменшується приблизно на три порядки.

Аналіз виразу (2.3) дозволяє на попередньому етапі індивідуального алгоритму задати значення вхідних змінних в активному експерименті (з урахуванням можливих повторів основного активного експерименту [14]), що дозволяє отримати оцінку коефіцієнта (2.1) з заданою точністю чи прийти до

висновку, що коефіцієнт (2.1) з заданою точністю індивідуальним алгоритмом декомпозиційного методу отримати неможливо.

Друга людино-комп'ютерна алгоритмічна процедура (другий АО)

Користувач вибирає нелінійний член БПР з (1.9):

$$b_{i_1 \dots i_t}^{j_1 \dots j_t} \left(x_{i_1}\right)^{j_1} \cdots \left(x_{i_k}\right)^{j_k} \cdots \left(x_{i_t}\right)^{j_t}, \tag{2.4}$$

задає вхідні змінні $x_{i_{t_1}}$, $x_{i_{t_2}}$, …, $x_{i_{t_l}}$, які в основному експерименті будуть приймати значення

$$\forall l \; x_{i_{t_l},i} = a_{i_{t_l}} z_i + b_{i_{t_l}}, i = \overline{1,n}, \tag{2.5}$$

де, як і в першому агрегованому операторі, коефіцієнти $a_{i_{t_l}}$, $b_{i_{t_l}}$ задаються відповідно до формул (7), (8) [14]. Інші вхідні змінні в активному експерименті приймають фіксовані значення, як і в першому агрегованому операторі. Алгоритмічно проводиться перевірка виконання наступної умови: не повинно існувати жодного нелінійного члена БПР, що містить вхідні змінні $x_{i_{t_1}}$, … , $x_{i_{t_l}}$ чи їх підмножину, сумарний степінь яких більше або дорівнює $\sum_{i=1}^{l} j_{t_i}$. Якщо такі члени існують, то їх коефіцієнти повинні бути знайдені з заданою точністю на попередніх кроках другого етапу індивідуального алгоритму декомпозиційного методу.

При виконанні цієї умови опис послідовності дій повністю повторює перший агрегований оператор. В цьому випадку, відповідно до [14], на стадії попереднього аналізу буде отримано дисперсію оцінки коефіцієнта $b_{i_1 \dots i_t}^{j_1 \dots j_t}$:

$$D\hat{b}_{i_1 \dots i_t}^{j_1 \dots j_t} = \frac{1}{\prod_{i=1}^{l} \left(a_{t_i}\right)^{2 j_{t_i}}} \cdot \frac{1}{\prod_{\substack{m=1 \\ i_m \neq t_1, \dots, t_l}}^{t} \left(x_{i_m, \phi}\right)^{2 j_m}} \cdot D\hat{\gamma}_{\sum_{i=1}^{l} j_{t_i}}. \tag{2.6}$$

Як і в першому агрегованому операторі, аналіз виразу (2.6) дозволяє на попередньому етапі індивідуального алгоритму декомпозиційного методу задати вхідні значення активного експерименту (з урахуванням можливих повторів основного експерименту [14]), що дозволяє отримати оцінку (2.4) з заданою

точністю, чи зробити висновок про неможливість реалізації такого активного експерименту.

Третя людино-комп'ютерна алгоритмічна процедура (третій АО)

Користувач вибирає нелінійний член БПР з (1.9):

$$b_{i_1 \dots i_t}^{1 \dots 1} x_{i_1} \cdot x_{i_2} \cdots x_{i_t}. \tag{2.7}$$

В основному активному експерименті вхідні змінні будуть приймати значення $x_{i_l,i} = a_{i_l} z_i + b_{i_l}$, $l = \overline{1,t}$, $i = \overline{1,n}$, a_{i_l}, b_{i_l}, z_i задаються як у першому агрегованому операторі. Інші вхідні змінні в активному експерименті приймають мінімально можливі значення. Алгоритмічно проводиться виконання перевірки наступної умови: не повинно існувати жодного нелінійного члена БПР, що містить вхідні змінні x_{i_l}, $l = \overline{1,t}$, чи їх підмножину, сумарний степінь яких більше або дорівнює t. Якщо такі члени існують, то їх коефіцієнти з заданою точністю повинні бути знайдені на попередніх кроках другого етапу індивідуального алгоритму.

При виконанні цієї умови, аналогічно першому та другому агрегованим операторам, на попередньому етапі індивідуального алгоритму буде отримано дисперсію оцінки коефіцієнта $b_{i_1 \dots i_t}^{1 \dots 1}$ (2.6):

$$D\hat{b}_{i_1 \dots i_t}^{1 \dots 1} = \frac{1}{\prod_{l=1}^{t} \left(a_{i_l}\right)^2} \cdot D\hat{\gamma}_t. \tag{2.8}$$

Реалізація третього агрегованого оператора є реальною, якщо значення усіх додатних коефіцієнтів a_{i_l} [14] близькі чи перевершують одиницю. Наприклад, якщо в [14] $z_{10} - z_1 = 100$, то значення $d_i - c_i$, $i = \overline{1,10}$, повинні бути близькі або більші до 100.

Четверта людино-комп'ютерна алгоритмічна процедура (четвертий АО)

Користувач вибирає нелінійний член БПР з (1.9):

$$b_{i_1 \dots i_t}^{1 \dots 1} x_{i_1} \cdot x_{i_2} \cdots x_{i_t}. \tag{2.9}$$

Підмножина його вхідних змінних $i_{t_1}, \dots, i_{t_m}$, $m < t$, $m \geq 2$, в основному активному експерименті буде приймати значення: $x_{i_{t_l},i} = a_{i_{t_l}} z_i + b_{i_{t_l}}$, $l = \overline{1,m}$, $m < t$, $i = \overline{1,n}$, як і в попередніх агрегованих операторах, відповідно до [14]. В основному активному експерименті інші вхідні члени (2.9) приймають максимально великі значення, а решта – мінімально можливі. Перевіряється виконання умови: не існує нелінійного члена БПР, що містить вхідні змінні $x_{i_{t_1}}, \dots, x_{i_{t_m}}$ чи їх підмножину, сумарний степінь яких більше або дорівнює m. Якщо такі члени є, то на попередніх кроках другого етапу індивідуального алгоритму їх оцінки будуть знайдені з заданою точністю.

При виконанні цієї умови дисперсія оцінки коефіцієнта (2.9) дорівнює:

$$D\hat{b}_{i_1 \dots i_t}^{1 \dots 1} = \frac{1}{\prod_{l=1}^{m} \left(a_{i_{t_l}}\right)^2} \cdot \frac{1}{\prod_{\substack{l=1 \\ \forall j_l \notin \{i_{t_1}, \dots, i_{t_m}\}}}^{t} \left(x_{j_l,\phi}\right)^2} \cdot D\hat{\gamma}_m. \tag{2.10}$$

Відповідно до [14], оцінка коефіцієнта при відповідному члені віртуальної ОПР за результатами основного віртуального активного експерименту задається виразом:

$$\hat{\gamma}_j = \sum_{i=1}^{n} \bar{y}_i Q_j(z_i), \tag{2.11}$$

де $Q_j(z)$ – нормовані ортогональні поліноми Форсайта (НОПФ), які знаходяться за значеннями z_i, $i = \overline{1,n}$, віртуальної вхідної змінної z. Таким чином, оцінки коефіцієнтів, за результатами реальних активних експериментів, знаходяться з використанням лише одного набору НОПФ, коефіцієнти яких заздалегідь знайдені з заданою точністю.

Оцінки коефіцієнтів при нелінійних членах БПР знаходяться за формулами:

– для першої людино-комп'ютерної алгоритмічної процедури (перший агрегований оператор):

$$\hat{b}_{i_1 \dots i_t}^{j_1 \dots j_t} = \frac{1}{\left(a_{i_k}\right)^k \cdot \prod_{l=1,\ l \neq i_k}^{t} \left(x_{i_l,\phi}\right)^{j_l}} \cdot \hat{\gamma}_{j_k}; \tag{2.12}$$

– для другої людино-комп’ютерної алгоритмічної процедури (другий агрегований оператор):

$$\hat{b}_{i_1\dots i_t}^{j_1\dots j_t} = \frac{1}{\prod_{i=1}^{l}\left(a_{t_i}\right)^{j_{t_i}}\cdot\prod^{t}_{\substack{m=1\\ \forall i_m\neq t_1,\dots,t_l}}\left(x_{i_m,\phi}\right)^{j_m}}\cdot\hat{\gamma}_{\sum_{i=1}^{l} j_{t_i}}; \tag{2.13}$$

– для третьої людино-комп’ютерної алгоритмічної процедури (третій агрегований оператор):

$$\hat{b}_{i_1\dots i_t}^{1\dots 1} = \frac{1}{\prod_{l=1}^{t} a_{i_l}}\cdot\hat{\gamma}_t; \tag{2.14}$$

– для четвертої людино-комп’ютерної алгоритмічної процедури (четвертий агрегований оператор):

$$\hat{b}_{i_1\dots i_t}^{1\dots 1} = \frac{1}{\prod_{l=1}^{m} a_{i_{t_l}}\cdot\prod^{t}_{\substack{l=1\\ \forall j_l\notin\left\{i_{t_1},\dots,i_{t_m}\right\}}} x_{j_l,\phi}}\cdot\hat{\gamma}_m. \tag{2.15}$$

Формули (2.12) – (2.15) є наслідком рівняння, що виражають коефіцієнт при максимальному члені віртуальної ОПР через відповідний коефіцієнт $b_{i_1\dots i_t}^{j_1\dots j_t}$. Наслідком цих рівнянь є також формули для дисперсій коефіцієнтів (2.3), (2.6), (2.8), (2.10).

Якщо існують вхідні змінні, область допустимих значень яких містить нуль, то використання можливості в активному експерименті реалізувати фіксоване значення вхідної змінної, що дорівнює нулю, може збільшити кількість коефіцієнтів при нелінійних членах БПР, оцінка яких є розв’язком лінійного рівняння з однією невідомою (див. ілюстративний приклад з підрозділу 2.6).

Надлишковий опис БПР може містити вхідні змінні в степені, більшому, ніж максимальний степінь НОПФ, коефіцієнти яких заздалегідь знайдені з заданою точністю. Тоді в усіх активних експериментах значення таких змінних повинно бути фіксованим.

2.1.3 Методологія використання чотирьох узагальнених операторів для створення користувачем індивідуального алгоритму оцінки коефіцієнтів при нелінійних членах БПР з заданою точністю

Методологія якісного підвищення ефективності декомпозиційного методу полягає в тому, що кваліфікований користувач в області регресійного аналізу, аналізуючи структуру БПР, коефіцієнти при нелінійних членах, якої треба оцінити (за величиною дисперсії) з заданою точністю, використовуючи розширені теоретичні можливості декомпозиційного методу (див. пункт 2.1.2) сам знаходить на етапі попереднього аналізу розв'язку задачі (до проведення реальних експериментів) послідовність кроків алгоритму. Тобто, в якій послідовності буде реалізований активний експеримент для побудови знайденої кількості ОПР для визначеної користувачем послідовності нелінійних членів БПР, що гарантує на кожному кроці отримання оцінок коефіцієнтів з наперед заданою точністю. Проведення, після етапу попереднього аналізу в знайденому порядку, активного експерименту для кожної ОПР використовується для побудови оцінок коефіцієнтів при найвищому степені цієї ОПР. Формальною процедурою дану послідовність кроків можливо отримати лише перебором всіх можливих варіантів, що, очевидно, є неконструктивним.

2.2 Обґрунтування вибору універсальної мови програмування з метою найбільш ефективного розширення її можливостей для автоматизації створення програмної реалізації індивідуального алгоритму знаходження оцінок коефіцієнтів БПР з заданою точністю

Для реалізації функціоналу розширення можливостей у частині автоматизації створення індивідуальних алгоритмів декомпозиційного методу може бути використана будь-яка універсальна мова програмування, наприклад Python, Java, C++, C#, Js [34]. З даної множини мов було обрано Python, тому що відповідна мова

програмування зарекомендувала себе як найбільш підходяща для реалізації Data Science, статистичних методів обробки даних, програмних бібліотек вузького призначення тощо [4]. Зокрема для мови програмування Python існує велика множина високорівневих фреймворків різного призначення та програмних бібліотек, які будуть використовуватись у якості допоміжних засобів при розробці програмного розширення можливостей мови для автоматизації створення та програмної реалізації індивідуальних алгоритмів декомпозиційного методу знаходження деяких оцінок коефіцієнтів БПР з заданою точністю [4].

При розробці програмного розширення автоматизації створення індивідуальних алгоритмів декомпозиційного методу для мови програмування Python необхідно дотримуватись: граматики, лексики, синтаксису та семантики даної мови [25]. Крім того, у підрозділі 1.4 було встановлено, що дане розширення варто розробити у вигляді спеціалізованої бібліотеки, яка буде інтерпретувати скрипт індивідуального алгоритму декомпозиційного методу, при цьому мати можливість вбудови у цільове програмне забезпечення за допомогою ключового слова “import” [25].

Скрипти, які міститимуть представлення індивідуального алгоритму декомпозиційного методу, зберігатимуться у текстових файлах з розширенням .poly, кожен алгоритм у окремому файлі. Спеціалізована бібліотека, у свою чергу, повинна включати функцію-інтерпретатор текстових файлів, що містять скрипти індивідуальних алгоритмів декомпозиційного методу. Дана функція повинна обробляти лексичні, синтаксичні та семантичні помилки у скриптах індивідуальних алгоритмів декомпозиційного методу, в тому числі і ті, що можуть виникнути при некоректному застосуванні декомпозиційного методу. У процесі розробки функції необхідно реалізувати парсер скриптів індивідуальних алгоритмів декомпозиційного методу, а також лексичний аналізатор, синтаксичний аналізатора та семантичний аналізатор.

Парсер перевіряє чи код відповідає граматиці та організовує його у деревовидну структуру [25]. Інтерпретатор – це програма або технічні засоби, що необхідні для виконання інших програм, є різновидом транслятора, який здійснює пооператорну обробку, перетворення у машинний код програми або запиту. Компілятор, на відміну від інтерпретатора, лише перетворює програму в машинний код без її виконання [25].

Далі буде описано дані, які має включати текстовий файл з розширенням .poly, що містить скрипт індивідуального алгоритму декомпозиційного методу.

Ініціалізація 1. Максимальний степінь НОПФ та відповідно ОПР у реалізації декомпозиційного методу: змінна r з додатнім цілим числовим значенням, обов'язково в діапазоні [2..10].

Наприклад, $r = 5$.

Можливі помилки:

– код 0 – дані відсутні (назва вхідної змінної);
– код 1 – невідповідність типу даних (не число);
– код 2 – нецiле число;
– код 3 – недодатнє число;
– код 4 – значення змінної у недопустимому діапазоні ($r < 2$ або $r > 10$).

Ініціалізація 2. Число спостережень у активному експерименті: змінна n з додатнім цілим числовим значенням, обов'язково в діапазоні [3..10000].

Наприклад, $n = 10$.

Можливі помилки:

– код 0 – дані відсутні (назва вхідної змінної);
– код 1 – невідповідність типу даних (не число);
– код 2 – нецiле число;
– код 3 – недодатнє число;

– код 4 – значення змінної у недопустимому діапазоні ($n < 3$ або $n > 10000$);

– код 5 – n повинно бути строго більше r.

Ініціалізація 3. Число вхідних змінних у надлишковому описі: змінна m з додатнім цілим числовим значенням, обов'язково в діапазоні [2..100].

Наприклад, $m = 6$.

Можливі помилки:

– код 0 – дані відсутні (назва вхідної змінної);

– код 1 – невідповідність типу даних (не число);

– код 2 – нецiле число;

– код 3 – недодатнє число;

– код 4 – значення змінної у недопустимому діапазоні ($m < 2$ або $m > 100$).

Ініціалізація 4. Області, в яких можуть приймати значення вхідні змінні: матриця input_vars_ranges_matrix розмірності $m \times 2$, де кожен рядок визначає область зміни значень i-ї змінної, $i = \overline{1,m}$ (нульовий рядок матриці input_vars_ranges_matrix визначає область значень першої вхідної змінної x_1).

Наприклад,

$$\text{input\_vars\_ranges\_matrix} = \begin{matrix}[[1, & 10], \\ [1, & 10], \\ [1, & 10], \\ [1, & 10], \\ [0, & 5], \\ [1, & 5]]\end{matrix}.$$

Можливі помилки:

– код 0 – дані відсутні (назва вхідної змінної);

– код 6 – невідповідність розмірності матриці input_vars_ranges_matrix (кількість рядків $\neq m$, кількість стовпців $\neq 2$);

– код 7 – невідповідність типу даних елементів матриці input_vars_ranges_matrix (не числа);

– код 8 – значення елементів матриці input_vars_ranges_matrix у недопустимому діапазоні (менше -2^{63}, більше $2^{63}-1$).

Ініціалізація 5. Надлишковий опис БПР: матриця redundant_description_matrix розмірності $m \times p$, де кожен стовпець відповідає окремому члену надлишкового опису, а рядок визначає степінь входження кожної вхідної змінної в кожен окремий член надлишкового опису. Кількість стовпців p дорівнює кількості членів надлишкового опису. Кількість рядків m дорівнює кількості вхідних змінних.

Наприклад,

$$\text{redundant\_description\_matrix} = \begin{matrix}[[0, & 2, & 2, & 1, & 1, & 0, & 1, & 2, & 1, & 1], \\ [0, & 1, & 2, & 2, & 2, & 0, & 1, & 0, & 0, & 1], \\ [0, & 3, & 0, & 0, & 0, & 3, & 3, & 0, & 0, & 1], \\ [0, & 0, & 1, & 1, & 2, & 2, & 0, & 3, & 3, & 0],' \\ [0, & 0, & 0, & 0, & 0, & 1, & 0, & 6, & 0, & 1], \\ [0, & 0, & 0, & 0, & 0, & 0, & 1, & 0, & 2, & 1]]\end{matrix}$$

де стовпець з індексом 1 – 2, 1, 3, 0, 0, 0 відповідає члену БПР $b_1 x_1^2 x_3^3 x_2$, і т.д.

Можливі помилки:

– код 0 – дані відсутні (назва вхідної змінної);

– код 6 – невідповідність розмірності матриці redundant_description_matrix (кількість рядків $\neq m$);

– код 7 – невідповідність типу даних елементів матриці redundant_description_matrix (не числа);

– код 8 – значення елементів матриці redundant_description_matrix у недопустимому діапазоні (менше 0, більше 10), степінь вхідної змінної у окремому члені надлишкового опису БПР більше 10 є неконструктивним [4];

– код 14 – дублювання стовпців матриці.

Ініціалізація 6. Області, в яких можуть приймати значення вхідні змінні віртуальної ОПР, верхня та нижня границі: числові змінні z_1, z_n. За замовчуванням $z_1 = -50$, z_n $= 50$.

Наприклад, $z_1 = 1$ z_n $= 11$.

Можливі помилки:

– код 1 – невідповідність типу даних (не число);

– код 4 – значення змінної у недопустимому діапазоні (z_1, z_n менше -2^{63} або z_1, z_n більше $2^{63} - 1$).

Ініціалізація 7. Значення коефіцієнтів НОПФ: матриця q_matrix розмірності $r + 1 \times r + 1$, де кожен стовпець містить коефіцієнти відповідного поліному Форсайта. Може приймати значення default, тоді змінна q_matrix міститиме коефіцієнти НОПФ для віртуальної регресії з вхідними значеннями $r = 5$, $n = 10$, $z_1 = -50$, $z_2 = -38{,}888$, $z_3 = -27{,}778$, $z_4 = -16{,}666$, $z_5 = -5{,}555$, $z_6 = 5{,}555$, $z_7 = 16{,}666$, $z_8 = 27{,}778$, $z_9 = 38{,}888$, $z_{10} = 50$ та точністю коефіцієнтів, що дорівнює 17 знакам після коми. Користувач може розрахувати матрицю коефіцієнтів НОПФ, присвоївши змінній q_matrix значення функції q_matrix_calc з наступними параметрами: r – максимальний степінь поліному Форсайта (ініціалізація 1), n – число спостережень віртуальної ОПР (ініціалізація 2), z_1, z_n – область, в якій може приймати значення вхідна змінна z віртуальної ОПР (інціалізація 6), accur – точність, з якою знаходяться значення коефіцієнтів НОПФ (число знаків після коми, ціле, додатнє).

Наприклад, q_matrix=default.

Можливі помилки:

– код 0 – дані відсутні (назва вхідної змінної);

– код 6 – невідповідність розмірності матриці q_matrix (кількість рядків $\neq r + 1$, кількість стовпців $\neq r + 1$);

– код 7 – невідповідність типу даних елементів матриці q_matrix (не числа);

– код 8 – значення елементів матриці q_matrix у недопустимому діапазоні (менше -2^{63}, більше $2^{63}-1$);

– код 9 – недопустиме значення змінної (не дорівнює default або q_matrix_calc з параметрами).

Ініціалізація 8. Число повторів активного експерименту для ОПР: змінна l з додатнім цілим числовим значенням, обов'язково в діапазоні [1..100]. За замовчуванням $l = 1$.

Наприклад, $l = 2$.

Можливі помилки:

– код 1 – невідповідність типу даних (не число);

– код 2 – нецілe число;

– код 3 – недодатнє число;

– код 4 – значення змінної у недопустимому діапазоні ($l < 1$ або $l > 100$).

Ініціалізація 9. Параметри переходу від реальної до віртуальної ОПР: матриця ab_matrix розмірності $m \times 2$, де кожен рядок визначає параметри a, b [9] переходу від реальної до віртуальної ОПР для i-ї змінної, $i = \overline{1,m}$. Може приймати дійсні числові значення або бути розрахована за допомогою функції ab_calc з наступними параметрами: матриця input_vars_ranges_matrix (ініціалізація 4) – області, в яких можуть приймати значення вхідні змінні, z_1, z_n – верхня та нижня границя віртуальної ОПР (ініціалізація 6).

Наприклад, ab_matrix = ab_calc(input_vars_ranges_matrix, z_1, z_n).

Можливі помилки:

– код 0 – дані відсутні (назва вхідної змінної);

– код 6 – невідповідність розмірності матриці input_vars_ranges_matrix (кількість рядків $\neq m$, кількість стовпців $\neq 2$);

– код 7 – невідповідність типу даних елементів матриці input_vars_ranges_matrix (не числа);

– код 8 – значення елементів матриці input_vars_ranges_matrix у недопустимому діапазоні (менше -2^{63}, більше $2^{63}-1$);

– код 9 – недопустиме значення змінної (не дорівнює ab_calc з параметрами).

Ініціалізація 10. Матриця, що містить значення вхідних змінних, на яких проводиться активний експеримент: input_matrix розмірності $n \times m$, де кожен стовпець даної матриці визначає вхідну змінну в активному експерименті, а рядок відповідно значення цих змінних при кожному спостережені у активному експерименті. Ініціалізується за допомогою функції input_matrix_init, яка приймає на вхід вектор-рядок, значення якого можуть бути задані одним з двох способів. Перший спосіб: значення заданого стовпця матриці input_matrix приймають константні фіксовані числові значення. Другий спосіб: значення заданого стовпця матриці input_matrix приймають значення, які можна розрахувати, викликаши функцію xz_ab, що реалізує формулу (9) [9], з наступними параметрами: ab_matrix[i] – параметри переходу від реальної до віртуальної ОПР для i-ї змінної, $i = \overline{1,m}$ (ініціалізація 9), z_1, z_n – область, в якій може приймати значення вхідна змінна z віртуальної ОПР (ініціалізація 6). z_1, z_n рівномірно розподілені у допустимій області своїх значень. Матриця input_matrix може бути ініціалізована дійсними числами відповідно до синтаксису мови програмування Python. Крім того, матрицю input_matrix можна ініціалізувати за допомогою функції input_matrix_reader з параметром file_path – шлях до текстового файлу зі значеннями input_matrix.

Наприклад,

$$\text{input\_matrix} =$$

$$\text{input\_matrix\_init}([10,\ 5,\ 1,\ \text{xz\_ab}(\text{ab\_matrix}[3], \text{z\_1}, \text{z\_n}),\ 6,\ 3]).$$

Можливі помилки:

– код 0 – дані відсутні (назва вхідної змінної);

– код 6 – невідповідність розмірності матриці input_matrix (кількість рядків $\neq n$, кількість стовпців $\neq m$);

– код 7 – невідповідність типу даних елементів матриці input_matrix (не числа);

– код 8 – значення елементів матриці input_matrix у недопустимому діапазоні (менше -2^{63}, більше $2^{63}-1$, значення вхідних змінних виходять за допустимі діапазони (ініціалізація 4));

– код 9 – недопустиме значення змінної (не дорівнює input_matrix_init або input_matrix_reader з відповідним параметром).

Ініціалізація 11. Точність, з якою необхідно знайти оцінки коефіцієнтів БПР: числова змінна coef_accuracy.

Наприклад, $\text{coef\_accuracy} = 0.01$.

Можливі помилки:

– код 0 – дані відсутні (назва вхідної змінної);

– код 1 – невідповідність типу даних (не число);

– код 4 – значення змінної у недопустимому діапазоні (менше або дорівнює 0, більше $2^{63}-1$).

Ініціалізація 12. Вектор значень активного експерименту для відповідної ОПР: вектор y_act розмірності n. Може приймати значення результатів активного експерименту для обраної конфігурації вхідних змінних (числовий вектор Python) або змінній y_act можна присвоїти значення функції y_act_reader з наступними параметрами: file_path – шлях до текстового файлу з результатами активного експерименту (структура файлу: $x_{1_i}, x_{2_i}, \dots, x_{m_i}, y_i, i = \overline{1,n}$), input_matrix – матриця розміру $n \times m$, що містить значення вхідних змінних, на яких проводиться активний експеримент (ініціалізація 10).

Наприклад, y_act=y_act_reader(file_path.txt, input_matrix).

Можливі помилки:

– код 9 – недопустиме значення змінної (не дорівнює y_act_reader(file_path, input_matrix));

– код 11 – невідповідність розмірності вектора y_act (кількість елементів ≠ n);

– код 12 – невідповідність типу даних елементів вектора y_act (не числа, значення відсутні);

– код 13 – значення елементів вектора y_act у недопустимому діапазоні (менше -2^{63}, більше $2^{63}-1$).

Обчислені значення оцінок коефіцієнтів БПР та їх дисперсії будуть зберігатись у словнику result_dict (оголошений за замовчуванням у функції-інтерпретаторі run_regression_script, яка описана нижче; опис виведення його значення може бути присутній у текстовому файлі, з розширенням .poly) з парами ключів Est_код_члена_БПР (наприклад, Est_0201 означає оцінку коефіцієнта члена БПР: $bx_2^2x_4$ для чотирьох змінних), Var_код_члена_БПР (наприклад, Var_0201 означає дисперсію члена БПР: $bx_2^2x_4$ для чотирьох змінних), значеннями яких є оцінки коефіцієнтів членів БПР та їх дисперсія відповідно (дійсні числа), перед початком обчислення ці значення дорівнюють “None”.

Макрос-оператор 1. Реалізує для заданого члена надлишкового опису БПР агрегований оператор 1 декомпозиційного методу (2.12) і обчислює дисперсію оцінки коефіцієнта даного члена надлишкового опису. Макрос-оператор dec_method_ao1 приймає на вхід наступні параметри: input_matrix – матриця, що містить значення вхідних змінних, на яких проводиться активний експеримент (ініціалізація 10), j-ий стовпець redundant_description_matrix[j], $j=\overline{0,p-1}$ (ініціалізація 5). Макрос-оператор повертає дисперсію заданого члена надлишкового опису БПР, яка записується у словник result_dict за ключем Var_код_члена_БПР, значенням якого і буде відповідна дисперсія (дійсне число).

У випадку неможливості застосування даного макрос-оператора, зміни у словник result_dict не вносяться.

Наприклад,

dec_method_ao1(input_matrix, redundant_description_matrix[1]).

Можливі помилки:

– код 15 – неможливість застосування даного агрегованого оператора до обраного члена надлишкового опису БПР відповідно до теоретичних даних оператора;

– код 16 – значення елемента словника Var_код_члена_БПР у недопустимому діапазоні (менше -2^{63}, більше $2^{63}-1$).

Макрос-оператор 2. Реалізує для заданого члена надлишкового опису БПР агрегований оператор 2 декомпозиційного методу (2.13) і обчислює дисперсію оцінки коефіцієнта даного члена надлишкового опису. Макрос-оператор dec_method_ao2 приймає на вхід наступні параметри: input_matrix – матриця, що містить значення вхідних змінних, на яких проводиться активний експеримент (ініціалізація 10), j-ий стовпець redundant_description_matrix[j], $j=\overline{0,p-1}$ (ініціалізація 5). Макрос-оператор повертає дисперсію заданого члена надлишкового опису БПР, яка записується у словник result_dict за ключем Var_код_члена_БПР, значенням якого і буде відповідна дисперсія (дійсне число). У випадку неможливості застосування даного макрос-оператора, зміни у словник result_dict не вносяться.

Наприклад,

dec_method_ao2(input_matrix, redundant_description_matrix[1]).

Можливі помилки:

– код 15 – неможливість застосування даного агрегованого оператора до обраного члена надлишкового опису БПР відповідно до теоретичних даних оператора;

– код 16 – значення елемента словника Var_код_члена_БПР у недопустимому діапазоні (менше -2^{63}, більше $2^{63}-1$).

Макрос-оператор 3. Реалізує для заданого члена надлишкового опису БПР агрегований оператор 3 декомпозиційного методу (2.14) і обчислює дисперсію оцінки коефіцієнта даного члена надлишкового опису. Макрос-оператор dec_method_ao3 приймає на вхід наступні параметри: input_matrix – матриця, що містить значення вхідних змінних, на яких проводиться активний експеримент (ініціалізація 10), j-ий стовпець redundant_description_matrix[j], $j=\overline{0,p-1}$ (ініціалізація 5). Макрос-оператор повертає дисперсію заданого члена надлишкового опису БПР, яка записується у словник result_dict за ключем Var_код_члена_БПР, значенням якого і буде відповідна дисперсія (дійсне число). У випадку неможливості застосування даного макрос-оператора, зміни у словник result_dict не вносяться.

Наприклад,

dec_method_ao3(input_matrix, redundant_description_matrix[1]).

Можливі помилки:

– код 15 – неможливість застосування даного агрегованого оператора до обраного члена надлишкового опису БПР відповідно до теоретичних даних оператора;

– код 16 – значення елемента словника Var_код_члена_БПР у недопустимому діапазоні (менше -2^{63}, більше $2^{63}-1$).

Макрос-оператор 4. Реалізує для заданого члена надлишкового опису БПР агрегований оператор 4 декомпозиційного методу (2.15) і обчислює дисперсію оцінки коефіцієнта даного члена надлишкового опису. Макрос-оператор dec_method_ao4 приймає на вхід наступні параметри: input_matrix – матриця, що містить значення вхідних змінних, на яких проводиться активний експеримент (ініціалізація 10), j-ий стовпець redundant_description_matrix[j], $j=\overline{0,p-1}$

(ініціалізація 5). Макрос-оператор повертає дисперсію заданого члена надлишкового опису БПР, яка записується у словник result_dict за ключем Var_код_члена_БПР, значенням якого і буде відповідна дисперсія (дійсне число). У випадку неможливості застосування даного макрос-оператора, зміни у словник result_dict не вносяться.

Наприклад,

dec_method_ao4(input_matrix, redundant_description_matrix[1]).

Можливі помилки:

– код 15 – неможливість застосування даного агрегованого оператора до обраного члена надлишкового опису БПР відповідно до теоретичних даних оператора;

– код 16 – значення елемента словника Var_код_члена_БПР у недопустимому діапазоні (менше -2^{63}, більше $2^{63}-1$).

Макрос-оператор 5. Реалізує для заданого члена надлишкового опису знаходження оцінки його коефіцієнта та дисперсії даного коефіцієнта шляхом зведення БПР до віртуальної ОПР з застосуванням першого чи другого підалгоритму декомпозиційного методу (див. підрозділ 1.2). Макрос-оператор dec_method приймає на вхід наступні параметри: input_matrix – матриця, що містить значення вхідних змінних, на яких проводиться активний експеримент (ініціалізація 10), y_act – вектор значень активного експерименту для відповідної ОПР (ініціалізація 11), l – число повторів активного експерименту (ініціалізація 8), надлишковий опис БПР, j-ий стовпець redundant_description_matrix[j], $j = \overline{0, p-1}$ (ініціалізація 5). Макрос-оператор повертає оцінки коефіцієнтів та їх дисперсії, що записуються у словник result_dict за парами ключів Est_код_члена_БПР, Var_код_члена_БПР, значеннями яких є оцінки коефіцієнтів членів БПР та їх дисперсія відповідно (дійсні числа). У випадку неможливості застосування даного макрос-оператора, зміни у словник result_dict не вносяться.

Наприклад,

dec_method(input_matrix, y_act, l, redundant_description_matrix[1]).

Можливі помилки:

– код 16 – значення елементів словника Est_код_члена_БПР, Var_код_члена_БПР у недопустимому діапазоні (менше -2^{63}, більше $2^{63}-1$).

Макрос-оператор 6. Реалізує для заданих членів надлишкового опису знаходження оцінок їх коефіцієнтів та дисперсії даних коефіцієнтів з застосуванням ММГУА (див. підрозділи 1.2, 2.3). Макрос-оператор mgmdh_method приймає на вхід наступні параметри: input_matrix – матриця, що містить значення вхідних змінних, на яких проводиться активний експеримент (ініціалізація 10), y_act – вектор значень активного експерименту для відповідної ОПР (ініціалізація 11), *l* – число повторів активного експерименту (ініціалізація 8), надлишковий опис БПР, множина стовпців redundant_description_matrix, які не були знайдені за допомогою декомпозиційного методу (див. підрозділ 1.2) (ініціалізація 5). Макрос-оператор повертає оцінки коефіцієнтів та їх дисперсії, що записуються у словник result_dict за парами ключів Est_код_члена_БПР, Var_код_члена_БПР, значеннями яких є оцінки коефіцієнтів членів БПР та їх дисперсія відповідно (дійсні числа). У випадку неможливості застосування даного макрос-оператора, зміни у словник result_dict не вносяться.

Наприклад,

mgmdh_method(input_matrix, y_act, l, redundant_description_matrix[0, 1, 2, 3]).

Можливі помилки:

– код 16 – значення елементів словника Est_код_члена_БПР, Var_код_члена_БПР у недопустимому діапазоні (менше -2^{63}, більше $2^{63}-1$).

Варто зазначити, що кожного разу перед безпосереднім викликом будь-якого з макрос-оператора, необхідно ініціалізувати його параметри.

У текстових файлах, з розширенням .poly повинна бути можливість написання коментарів. Для цього буде використано оператор # аналогічно до однорядкових коментарів у мові програмування Python.

При необхідності детального аналізу значень змінних або значень, які розраховують макрос-оператори у текстовому файлі з розширенням .poly, повинна бути можливість виведення цих значень у термінал. Для цього буде використано функцію print аналогічно до вбудованої функції виведення у мові програмування Python.

Для коректної роботи функції-інтерпретатора run_regression_script необхідно реалізувати усі вище описані допоміжні функції, які згадувались у ініціалізаціях, до них відносяться: q_matrix_calc, ab_calc, input_matrix_init, input_matrix_reader, y_act_reader. А також макрос-оператори: dec_method_ao1, dec_method_ao2, dec_method_ao3, dec_method_ao4, dec_method, mgmdh_method. Детальний опис цих функцій буде наведено у підрозділі 3.2.

Приклад .poly файлу:

```
# Максимальний степінь БПР
r=5
# Кількість спостережень
n=10
# Кількість вхідних змінних
m=6
# Ініціалізація матриці, яка містить області, в яких можуть приймати значення вхідні змінні
input_vars_ranges_matrix = [[1, 10],
                            [1, 10],
                            [1, 10],
                            [1, 10],
                            [0,  5],
                            [1,  5]]
# Ініціалізація матриці, яка містить надлишковий опис БПР
```

$$\text{redundant\_description\_matrix} = \begin{matrix}[[0, & 2, & 2, & 1, & 1, & 0, & 1, & 2, & 1, & 1], \\ [0, & 1, & 2, & 2, & 2, & 0, & 1, & 0, & 0, & 1], \\ [0, & 3, & 0, & 0, & 0, & 3, & 3, & 0, & 0, & 1], \\ [0, & 0, & 1, & 1, & 2, & 2, & 0, & 3, & 3, & 0], \\ [0, & 0, & 0, & 0, & 0, & 1, & 0, & 6, & 0, & 1], \\ [0, & 0, & 0, & 0, & 0, & 0, & 1, & 0, & 2, & 1]]\end{matrix}$$

```
# Області, в яких можуть приймати значення вхідні змінні віртуальної ОПР, верхня та нижня границі
z_1=-50 z_n=50
# Матриця із значеннями коефіцієнтів НОПФ
q_matrix=default
# Матриця із значеннями параметрів переходу від реальної до віртуальної ОПР
ab_matrix=ab_calc(input_vars_ranges_matrix,z_1,z_n)
# Ініціалізація матриці, що містить значення вхідних змінних, на яких проводиться активний експеримент
input_matrix=input_matrix_init([10,5,1,xz_ab(ab_matrix[3],z_1,z_n),6,3])
# Точність знаходження оцінок коефіцієнтів БПР
coef_accuracy=0.01
# Застосування агрегованого оператора 1 до члена надлишкового опису БПР з індексом 1
dec_method_ao1(input_matrix, redundant_description_matrix[1])
print(result_dict)
# Вектор значень активного експерименту для відповідної ОПР
y_act=y_act_reader(file_path.txt, input_matrix)
# Обчислення оцінки коефіцієнта члена надлишкового опису БПР з індексом 1
dec_method(input_matrix, y_act, 1, redundant_description_matrix[1])
print(result_dict)
```

Функція-інтерпретатор текстових файлів, що містять скрипти індивідуальних алгоритмів декомпозиційного методу матиме назву run_regression_script.

Функція run_regression_script приймає на вхід:

– шлях до текстового файлу з розширенням .poly, що містить скрипт індивідуальних алгоритмів декомпозиційного методу.

Функція run_regression_script повертає:

– у випадку коректної роботи: словник result_dict з парами ключів Est_код_члена_БПР (наприклад, Est_0201 означає оцінку коефіцієнта члена БПР: $bx_2^2x_4$ для чотирьох змінних), Var_код_члена_БПР (наприклад, Var_0201 означає дисперсію члена БПР: $bx_2^2x_4$ для чотирьох змінних), значеннями яких є оцінки коефіцієнтів членів БПР та їх дисперсія відповідно (дійсні числа), або значення "None", якщо якесь з цих значень не вдалось знайти;

– у випадку некоректної роботи: коди помилок та їх описи або одну з загальних помилок (помилка роботи парсера, невідоме декларування тощо), номер рядка та символу, де виникла помилка.

2.3 Підвищення ефективності ММГУА для побудови багатовимірних регресій, лінійних відносно невідомих коефіцієнтів, заданих надлишковим описом, на основі комплексного використання асиметричного та симетричного регулярних критеріїв

У підрозділі 1.2 було викладено загальну алгоритмічну процедуру ММГУА [4] та обґрунтовано необхідність комплексного застосування низки регулярних критеріїв (замість одного) для підвищення ефективності ММГУА [35]. В синтетичному методі розглядається частковий випадок ММГУА для БПР.

У теорії класичного МГУА використовуються різні критерії регулярності та їх комбінації для покращення шуканого розв'язку. Прикладами таких критеріїв є: асиметричні критерії ЗСК, симетричний критерій $\text{ЗСК}_1 + \text{ЗСК}_2$, що передбачає заміну навчальної та перевірочної послідовності місцями і знаходження суми ЗСК.

Запропоновано підвищити ефективність використання ММГУА у постановці, викладеної в підрозділах 1.1-1.2, з одночасним використанням регулярних критеріїв ЗСК_1 (навчальна-перевірочна послідовності), ЗСК_2 (перевірочна-навчальна послідовності), $\text{ЗСК}_1 + \text{ЗСК}_2$ (навчальна-перевірочна +

перевірочна-навчальна послідовності), а також критерію χ^2 для перевірки гіпотези про розподіл випадкової величини E та вибір кращого претендента на розв'язок.

Перша проблема, яка виникає, полягає у розбитті експериментальних даних на навчальну та перевірочну послідовності. Зазвичай використовують два варіанти: (50%, 50%) та (60%, 40%) [16], кожен з яких має свої переваги та недоліки. У даному дисертаційному дослідженні пропонується використати один будь-який варіант розбиття.

Для підвищення ефективності ММГУА у даному дисертаційному дослідженні будуть використані асиметричні критерії ЗСК_1, ЗСК_2 та симетричний критерій $\text{ЗСК}_1 + \text{ЗСК}_2$. Специфіка ММГУА, зокрема, використання алгоритму розбиття множини коефіцієнтів $\{b_1, \dots, b_r\}$ на два класи M_1, M_2 (підрозділ 1.2) приводить до необхідності створення спеціальної процедури використання симетричного критерію ($\text{ЗСК}_1 + \text{ЗСК}_2$) [35]. Нехай $\left(\bar{x}_{n_1+i} \to y_{n_1+i}, i = \overline{n_1+1, n}\right)$ – перша перевірочна послідовність, яка породжує регулярний критерій ЗСК_1, а $(\bar{x}_i \to y_i, i = \overline{1, n_1})$ – це друга перевірочна послідовність, яка породжує відповідно регулярний критерій ЗСК_2. У першому випадку відбувається розбиття множини коефіцієнтів $\{b_1, \dots, b_r\}$ на два класи M_1^1, M_2^1, а у другому випадку – M_1^2, M_2^2. Тоді для застосування симетричного критерію розбиття множини коефіцієнтів $\{b_1, \dots, b_r\}$ на два класи відбувається наступним чином: $M_1 = M_1^1 \cap M_2^1$, $M_2 = \{b_1, \dots, b_r\}/M_1$. Для симетричного критерію множина всіх часткових описів БР має вигляд:

$$\forall_{M_2^j} \left\{ Y(\bar{x}) = \sum_{\forall b_l \in M_1} b_l \psi_l(\bar{x}) + \sum_{\forall b_l \in M_2^j} b_l \psi_l(\bar{x}) + E \right\}. \qquad (2.16)$$

В результаті послідовного використання критеріїв регулярності ЗСК_1, ЗСК_2, $\text{ЗСК}_1 + \text{ЗСК}_2$ буде отримано три претенденти на розв'язок, які для спрощення буде позначено наступним чином:

$$\left(M_1^1, M_2^{1,j_1}\right), \left(M_1^2, M_2^{2,j_2}\right), \left(M_1, M_2^{j_3}\right). \quad (2.17)$$

Множини (2.17) однозначно задають структуру часткових описів БР, що є претендентами на розв'язок [35].

Нехай χ_1^2, χ_2^2, χ_3^2 є різними реалізаціями критерію χ^2, що перевіряє гіпотезу про розподіл випадкової величини E за множиною знайдених оцінок $\hat{\varepsilon}_i$ для кожного з трьох часткових описів БР, які є претендентами на розв'язок.

Нехай χ_α^2 має критичну область критерію достовірності гіпотези:

$$P(\chi^2 \geq \chi_\alpha^2) = 1 - \alpha. \quad (2.18)$$

Перший критерій розв'язку задачі регресії полягає у тому, що шукана регресія задається частковим описом, на якому виконується:

$$\min(\chi_1^2, \chi_2^2, \chi_3^2) < \chi_\alpha^2. \quad (2.19)$$

Обґрунтування критерію. Відомо, що чим менш достовірна гіпотеза про розподіл випадкової величини, яка перевіряється, тим сильніша тенденція до збільшення значення критерію χ^2 [35].

Другий критерій розв'язку задачі регресії передбачає дві умови.

Умова 1. З трьох претендентів на розв'язок вибирається той, що містить мінімальну кількість членів, якщо виконується наступне:

$$\max(\chi_1^2, \chi_2^2, \chi_3^2) - \min(\chi_1^2, \chi_2^2, \chi_3^2) < \chi_\alpha^2 - \max(\chi_1^2, \chi_2^2, \chi_3^2); \quad (2.20)$$

Умова 2. Якщо лише дві реалізації критерію χ^2 задовольняють умові (2.19), тоді з двох претендентів на розв'язок вибирається той, який містить менше членів, при виконанні умови, яка аналогічна умові (2.20). У протилежному випадку залишається лише один претендент на розв'язок.

Якщо претендентів на розв’язок один чи два (різні регулярні критерії обрали однакові часткові описи БР), то розв’язок задачі регресії обриється відповідно до умови 2.

2.4 Обґрунтування можливості використання оригінального алгоритму статичного управління асинхронними паралельними обчислювальними процесами у ММГУА

У підрозділах 1.2 та 2.3 було описано та запропоновано модифікацію алгоритму ММГУА для задачі побудови БПР. У даному підрозділі буде теоретично обґрунтовано можливість використання алгоритму статичного управління асинхронними паралельними обчислювальними процесами для підвищення швидкодії ММГУА.

2.4.1 Модель статичного управління паралельними асинхронними обчислювальними процесами на ідентичних обчислювальних пристроях

Функцією деякої програми можна логічно розділити на n підфункцій, які починають виконуватися в довільні моменти часу або одночасно $s_j > 0,\ j = \overline{1,m}$ за допомогою паралельних обчислень.

В [2, 15] розглядаються комбінаторні моделі де кількість паралельно обчислювальних пристроїв (логічних процесорів) m $(m < n)$, за допомогою яких виконуються незалежні паралельні обчислення, є заздалегідь відомою і може бути оптимізована з використанням деяких алгоритмів.

В першій моделі продуктивність обчислювальних пристроїв вважаються умовно однаковими. Відомий оціночний час виконання l_i i-ї підпрограми на одному обчислювальному пристрої, $i = \overline{1,n}$. Ставиться задача побудувати розклад σ виконання n підфункцій на m обчислювальних пристроях такий, щоб досягався мінімум функціоналу:

$$\min_{\sigma}\{F(\sigma) = C_{max}(\sigma) - C_{min}(\sigma)\}, \quad (2.21)$$

де $C_{max}(\sigma)$ – момент завершення виконання всіх підфункцій у розкладі σ, тобто, $C_{max}(\sigma) = \max\limits_{i=\overline{1,n}} C_i(\sigma)$, де $C_i(\sigma)$ – момент завершення виконання підфункцій на i-му обчислювальному пристрої; аналогічно, $C_{min}(\sigma) = \min\limits_{i=\overline{1,n}} C_i(\sigma)$.

2.4.2 Алгоритм статичного управління паралельними асинхронними обчислювальними процесами на ідентичних обчислювальних пристроях

В [2, 15] було описано ефективний ПДС-алгоритм для розв'язання задачі, описаної в пункті 2.4.1, даний алгоритм містить дві достатні ознаки оптимальності довільного розкладу σ та включає у себе два етапи: суть першого етапу полягає у побудові початкового розкладу і реалізується за 6 кроків, що описані нижче.

Крок 1. Присвоїти підфункціям номери $i = \overline{1,n}$ впорядковані за спаданням відносно значень l_i.

Крок 2. Присвоїти обчислювальним пристроям номери $j = \overline{1,m}$ впорядковані за спаданням відносно значень s_j.

Крок 3. Ініціалізувати час, коли обчислювальний пристрій стає доступним $C_j = s_j, j = \overline{1,m}$.

Крок 4. Вибрати підфункцію i з максимальним l_i з нерозподілених на обчислювальні пристрої підфункцій та призначити на обчислювальний пристрій j з мінімальним часом завершення обчислень (доступності) C_j.

Крок 5. Ініціалізувати новий час, коли обчислювальний пристрій стає доступним j: $C_j = C_j + l_j$.

Крок 6. Якщо на обчислювальні пристрої розподілені всі підфункції, алгоритм завершує роботу. Інакше, виконати перехід на крок 4.

Суть другого етапу полягає в мінімізації значення функціоналу, що оцінює якість розкладу, шляхом послідовного виконання перестановок типів А і Б [2, 15].

Низка проведених досліджень ПДС-алгоритму показали його ефективність, що дозволяє його використовувати у якості алгоритму статичного управління за першою моделлю [2, 15].

2.4.3 Знаходження аналітично обґрунтованих вхідних даних для використання оригінального алгоритму статичного управління асинхронними паралельними обчислювальними процесами у ММГУА

У підрозділі 1.2 було описано постановку задачі побудови БПР, задану надлишковим описом (1.9). Для розв'язання цієї задачі в [4] було запропоновано ММГУА, методологія реалізації якого описано у підрозділі 1.2, а у підрозділі 2.3 запропоновані модифікації, які покращують якість знайденої з його допомогою структури БПР та точність оцінок коефіцієнтів БПР. Загальна блок-схема ММГУА наведена на рисунку 2.1 [4].

Швидкодія програмної реалізації запропонованого в [4] ММГУА для побудови БПР з використанням обмеженого об'єму даних може бути суттєво збільшена, якщо для знаходження оцінок коефіцієнтів множини часткових описів БПР (див. крок 4 ММГУА, підрозділ 1.2) використовувати асинхронні паралельні обчислення, статичне управління якими реалізується відповідно до першої або другої комбінаторної моделі [15], в залежності від обчислювальних засобів, що використовуються [15]. Параметр управління – це час побудови часткового опису БПР на конкретному обчислювальному пристрої, який залежить від розмірності часткового опису (фактично від кількості спостережень активного чи пасивного експерименту, що входять у підмножину D_1, та потужності множини O (див. підрозділ 1.2)), а також способу реалізації загальної формули МНК при повторних активних експериментах. Результати теоретичних та експериментальних досліджень способу реалізації загальної формули МНК наведені у [36], відповідно до них, для паралельної реалізації матричних операцій необхідно обирати

стрічкової метод – операція множення матриць, LUP метод – операція оберненої матриці.

Фактично для збільшення швидкодії програмної реалізації ММГУА необхідно розподілити задачі знаходження оцінок коефіцієнтів часткових описів, відповідно до їх розмірності, по обчислювальним пристроям, використовуючи алгоритм статичного управління паралельними асинхронними обчислювальними процесами на ідентичних обчислювальних пристроях (див. пункт 2.4.2). При цьому у загальній формулі МНК використання паралельних реалізацій матричних операцій є опціональними.

Для спрощення викладення наступного матеріалу, кількість спостережень активного чи пасивного експерименту, що входять у підмножину D_1, позначено як n, а потужність множини O відповідно m. Тобто у загальній формулі МНК ($(A^TA)^{-1}A^Ty$) для оцінки невідомих коефіцієнтів матриця A матиме розмірність $n \times m$, а вектор y розмірність n. Параметр управління буде визначено як асимптотична функція оцінки часу обрахунку загальної формули МНК відносно її вхідних параметрів (розмірності матриці A та вектора y).

Асимптотична функція оцінки часу обрахунку загальної формули МНК складатиметься із суми часу обрахунку окремих її складових та обрання найбільш трудомісткої складової за критерієм часу обчислення, а саме:

- транспонування матриці A – $n \cdot m$;
- множення A^TA – $n \cdot m^2$;
- обернення $(A^TA)^{-1}$ – m^3;
- множення $(A^TA)^{-1}A^T$ – $n \cdot m^2$;
- множення на вектор y $(A^TA)^{-1}A^Ty$ – $n \cdot m$.

Звідси час обрахунку загальної формули МНК:

$$T_{\text{МНК}}(n,m) = n \cdot m + n \cdot m^2 + m^3 + \ n \cdot m^2 + \ n \cdot m = m^3 + 2 \cdot n \cdot m^2 + 2 \cdot n \cdot m.$$

Час обрахунку загальної формули МНК у асимптотичних оцінках:

$$O\big(T_{\text{МНК}}(n,m)\big) = n \cdot m^2.$$

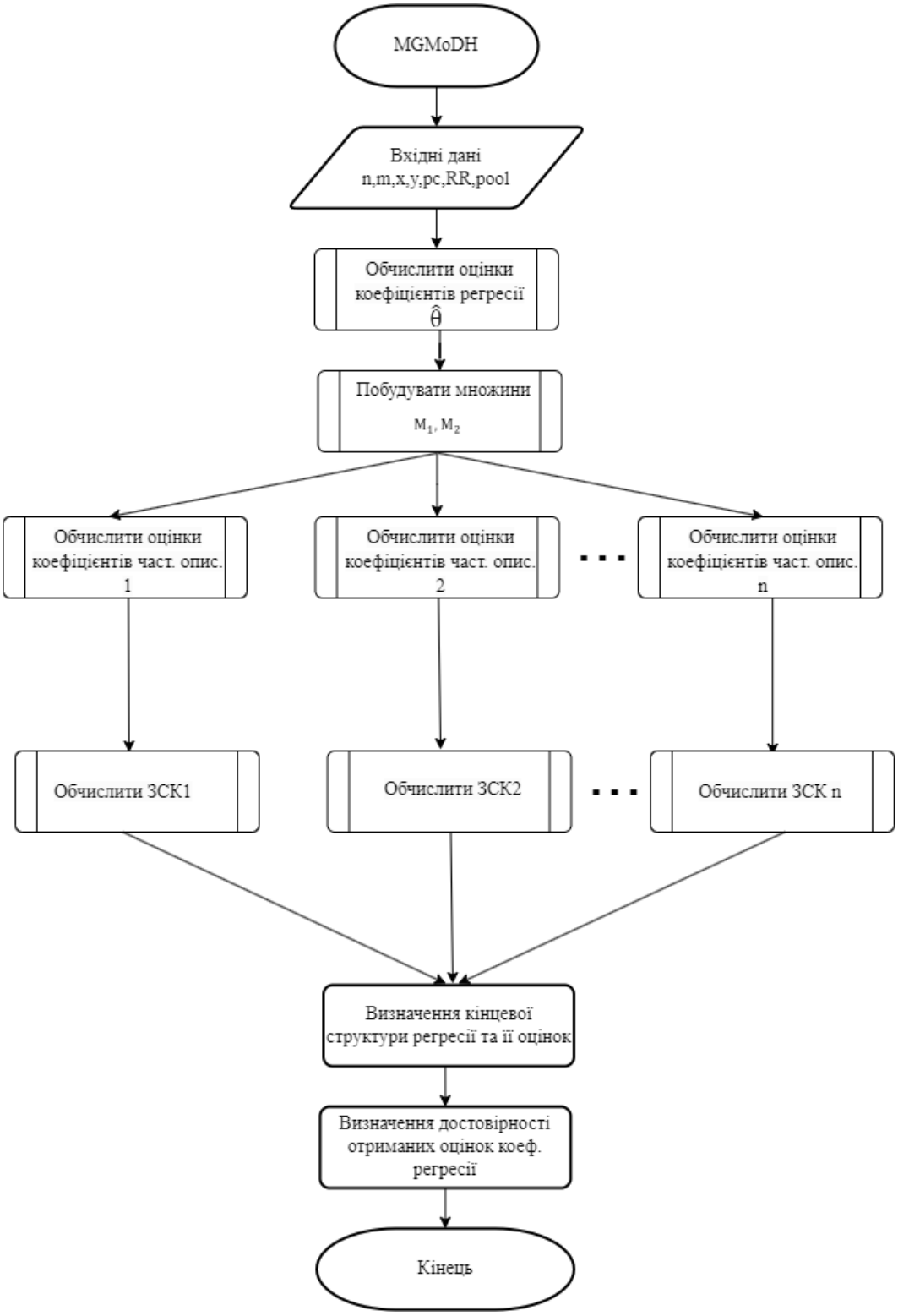


Рисунок 2.1 – Загальна блок-схема ММГУА для БПР [4]

2.5 Методологія створення індивідуальних алгоритмів та розширення мови Python для автоматизації створення та програмної реалізації індивідуальних алгоритмів побудови БПР

Враховуючи вищеописані теоретично обґрунтовані результати, буде наведено загальну методологію створення індивідуальних алгоритмів синтетичного методу.

Крок 1. Користувач, візуально аналізує структуру БПР, заданої надлишковим описом, задає послідовність нелінійних членів БПР, а також приписує кожному нелінійному члену БПР цієї послідовності номер узагальненого оператора, множину фіксованих вхідних змінних та множину вхідних змінних, значення яких в активному експерименті змінюються. Множина всіх вхідних змінних задається нелінійним членам БПР. В разі потреби задаються значення деяких вхідних змінних (наприклад нуль).

Крок 2. Програмне забезпечення, крок за кроком, починаючи з першого нелінійного члена, заданої користувачем послідовності, перевіряє умови виконання узагальненого оператора, знаходить дисперсію оцінки коефіцієнта та необхідну кількість повторних експериментів, якщо вона задовольняє обмеженням, які накладає користувач.

Крок 3. Процес побудови індивідуального алгоритму зупиняється на першому кроці, для якого одна з умов його реалізації не виконується. Користувач отримує відповідну інформацію, аналізує її і вводить корекцію, яка полягає у:

– змінні для даного нелінійного члена БПР множини фіксованих вхідних змінних чи в корекції послідовності нелінійних членів БПР. Користувач може змінювати, доповнювати послідовність нелінійних членів БПР, враховуючи які оцінки коефіцієнтів будуть знайдені з заданою точністю. Таким чином, в результаті роботи користувача з програмним забезпеченням, буде знайдена множина

коефіцієнтів при нелінійних членах БПР, оцінки яких будуть знайдені з заданою точністю.

Крок 4. Програмне забезпечення видає користувачу необхідну інформацію для проведення активного експерименту, з метою знаходження оцінок коефіцієнтів при нелінійних членах БПР з заданою точністю, та дані для проведення активного експерименту для реалізації ММГУА. При реалізації ММГУА можуть бути використанні також, в разі необхідності, результати пасивного експерименту.

Крок 5. Проведення активних експериментів, заданих індивідуальним алгоритмом декомпозиційного методу, отримання оцінок коефіцієнтів при нелінійних членах БПР, заданої надлишковим описом, з використанням цих оцінок реалізація ММГУА та знаходження оцінок всіх коефіцієнтів побудованої БПР.

Методологія розширення мови Python для автоматизації створення та програмної реалізації індивідуальних алгоритмів синтетичного методу побудови БПР буде представлена у наступних п’яти кроках.

Крок 1. Розробка Python-подібних лексики, синтаксису та семантики опису індивідуальних алгоритмів побудови БПР.

Крок 2. Розробка парсеру для зчитування обробки та перетворення у зручну для інтерпретації структуру даних опису індивідуальних алгоритмів побудови БПР.

Крок 3. Розробка інтерпретатора, який буде аналізувати та виконувати індивідуальні алгоритми побудови БПР або повідомляти користувача про неможливість виконання.

Крок 4. Розробка функції-інтерпретатора, яка буде запускати на виконання опис індивідуальних алгоритмів побудови БПР.

Крок 5. Забезпечити можливість вбудови функції-інтерпретатора у сторонні проєкти.

2.6 Ілюстративний приклад використання узагальнених операторів для оцінки коефіцієнтів при нелінійних членах БПР з заданою точністю

Задано надлишковий опис БПР в вигляді:

$$Y(x_1, x_2, x_3, x_4, x_5, x_6) = b_0 + b_1 x_1^2 x_3^3 x_2 + b_2 x_1^2 x_2^2 x_4 + b_3 x_2^2 x_1 x_4 + b_4 x_2^2 x_4^2 x_1 + b_5 x_3^3 x_4^2 x_5 + b_6 x_3^3 x_1 x_2 x_6 + b_7 x_5^6 x_4^3 x_1^2 + b_8 x_4^3 x_6^2 x_1 + b_9 x_1 x_2 x_3 x_4 x_6 + E,$$

де $ME = 0$, $DE = 4$, E розподілена по нормальному закону.

Істинні значення коефіцієнтів наступні:

$$b_0 = 1, b_1 = 2, b_2 = 0, b_3 = 1, b_4 = 0, b_5 = 2, b_6 = 1, b_7 = 2, b_8 = 1, b_9 = 2.$$

Таким чином, в істинному описі БПР відсутні члени з коефіцієнтами b_2, b_4.

Області, в яких можуть приймати значення вхідні змінні є наступними:

$$x_1 \in [1; 10], x_2 \in [1; 10], x_3 \in [1; 10], x_4 \in [1; 10], x_5 \in [0; 5], x_6 \in [1; 5].$$

БПР не містить лінійних складових, тому що за теоретичними положеннями декомпозиційного методу їх оцінка є неефективною і знаходиться з використанням ММГУА (див. підрозділ 2.3), причому присутність лінійної частини в БПР не впливає на точність оцінок коефіцієнтів при нелінійних членах БПР.

Тобто, оцінку коефіцієнта b_0 та коефіцієнтів при можливих лінійних членах БПР знаходить ММГУА, після реалізації індивідуального алгоритму декомпозиційного методу.

Коефіцієнти при нелінійних членах БПР будуть знаходитись з точністю до 2 знаку після коми.

Декомпозиційний метод

Крок 1. Реалізується для нелінійного члену $b_1 x_1^2 x_3^3 x_2$ з використанням другого агрегованого оператора та першого підалгоритму $x_1 = \overline{a_1} z + \overline{b_1}$ та $x_3 = \overline{a_3} z + \overline{b_3}$, в яких $\overline{a_1}$, $\overline{b_1}$, $\overline{a_3}$, $\overline{b_3}$ знаходяться по формулі (7), (8) [14], де $z_1 = -50$, $z_{10} = 50$, $n = 10$, $d_1 = d_3 = 10$, $\mathrm{c}_1 = \mathrm{c}_3 = 1$, $x_i = \overline{a_4} z_\mathrm{i} + \overline{b_4}$, $i = \overline{1, 10}$.

$$z_1 = -50, z_2 = -38{,}888, z_3 = -27{,}778, z_4 = -16{,}666, z_5 = -5{,}555, z_6 = 5{,}555,$$
$$z_7 = 16{,}666, z_8 = 27{,}778, z_9 = 38{,}888, z_{10} = 50,$$

$$\overline{a_1} = \overline{a_3} = \frac{d_1 - c_1}{z_{10} - z_1} = 0{,}09,\ \overline{b_1} = \overline{b_3} = c_1 - \frac{d_1 - c_1}{z_{10} - z_1} z_1 = 5{,}5.$$

У всіх випробуваннях вхідні змінні x_2, x_4, x_5, x_6 приймають наступні значення: $x_2 = 10$, $x_4 = 1, x_5 = 1, x_6 = 1$. Значення x_1, x_3 рівномірно розподілені у діапазоні [1;10] (7), (8) [14]. Тоді у всіх випробуваннях БПР перетворюється в ОПР вигляду:

$$Y(z) = b_0 + 10b_1(0{,}09z + 5{,}5)^2(0{,}09z + 5{,}5)^3 + 100b_2(0{,}09z + 5{,}5)^2 + 100b_3(0{,}09z + 5{,}5) + 100b_4(0{,}09z + 5{,}5) + b_5(0{,}09z + 5{,}5)^3 + 10b_6(0{,}09z + 5{,}5)^3(0{,}09z + 5{,}5) + b_7(0{,}09z + 5{,}5)^2 + b_8(0{,}09z + 5{,}5) + 10b_9(0{,}09z + 5{,}5)(0{,}09z + 5{,}5) + E = b_0 + 10b_1(0{,}09z + 5{,}5)^5 + 100b_2(0{,}09z + 5{,}5)^2 + 100b_3(0{,}09z + 5{,}5) + 100b_4(0{,}09z + 5{,}5) + b_5(0{,}09z + 5{,}5)^3 + 10b_6(0{,}09z + 5{,}5)^4 + b_7(0{,}09z + 5{,}5)^2 + b_8(0{,}09z + 5{,}5) + 10b_9(0{,}09z + 5{,}5)^2 + E = \gamma_0 + \gamma_1 z + \gamma_2 z^2 + \gamma_3 z^3 + \gamma_4 z^4 + \gamma_5 z^5 + E,$$

де $\gamma_5 = 10b_1(0{,}09)^5$.

Нормовані ортогональні поліноми Форсайта побудовані для z_i, $i = \overline{1, 10}$, а їх коефіцієнти знайдені із точністю до 17 перших розрядів після коми (таблиця 2.1).

Таблиця 2.1 – Знайдені коефіцієнти НОПФ

	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5
q_{j0}	0,31622776601683794	0	−0,35903046525330308508	0	0,37601029391503510256	0
q_{j1}		0,0099087567581727680483	0	−0,023723738974404719567	0	0,040733377422616100 65
q_{j2}			0,0003525085350 0849863344	0	−0,0012936966199861974775	0
q_{j3}				0,0000131169489 44919936589	0	−0,0000674335625329765680 96
q_{j4}					5,1116893449436679076· 10^{-7}	0
q_{j5}						2,1143486041566852428· 10^{-8}

Обчислюється оцінка γ_5 по наступному віртуальному активному експерименту ($z_i \to y_i,\ i = \overline{1,10}$), де y_i – результати реального активного експерименту ($x_{1,i}, 10, x_{3,i}, 1, 1, 1 \to y_i, i = \overline{1,10}$).

$x_{1,i} = \overline{a_1} z_i + \overline{b_1}, x_{3,i} = \overline{a_3} z_i + \overline{b_3}, i = \overline{1,10},$

$y_i = b_0 + 10b_1(x_{1,i})^2(x_{3,i})^3 + 100b_2(x_{1,i})^2 + 100b_3(x_{1,i}) + 100b_4(x_{1,i}) + b_5(x_{3,i})^3 + 10b_6(x_{3,i})^3(x_{1,i}) + b_7(x_{1,i})^2 + b_8(x_{1,i}) + 10b_9(x_{1,i})(x_{3,i}) + \delta_i,\ i = \overline{1,10},$

де δ_i – реалізації випадкової величини E в активному експерименті (таблиця 2.2).

А самі значення активного експерименту $y_i,\ i = \overline{1,10}$ прийматимуть такі значення (таблиця 2.3).

Таблиця 2.2 – Реалізації випадкової величини E в активному експерименті 1

δ_i	Значення
δ_1	-1,0659694104455328
δ_2	3,7710681456850645
δ_3	-0,7377368040485024
δ_4	-0,0769108594611502
δ_5	1,659240003527758
δ_6	0,6385687090711256
δ_7	1,393884665512836
δ_8	-0,9162780984774989
δ_9	-4,381571522845034
δ_{10}	-5,414570636039612

Таблиця 2.3 – Активний експеримент 1

y_i	Значення
y_1	154,9340305895544672
y_2	1110,9417201397366571
y_3	6225,0729255879361216
y_4	23926,635118842865047
y_5	70061,057856384178011
y_6	170304,69763225314088
y_7	362608,12276243773514
y_8	699568,70210442562748
y_9	1250680,7199531790753
y_{10}	2105205,5854293639604

Використовуючи НОПФ таблиці 2.1, буде знайдено оцінку $\widehat{\gamma_5}$ віртуальної одновимірної регресії (2.40), (2.41) [4]: $\widehat{\gamma_5} = 0{,}000118172133464$.

В загальному вигляді дисперсія $\widehat{\gamma_5} = \sigma^2 4{,}47047 \cdot 10^{-16}$, $M\widehat{\gamma_5} = \gamma_5$, для $\sigma^2 = 4$, дисперсія $D\widehat{\gamma_5} = 1{,}788188 \cdot 10^{-15}$. Оцінка коефіцієнта $\widehat{b_1}$ знаходиться з рівняння $\widehat{\gamma_5} = 10\widehat{b_1}(0{,}09)^5$, чи $10b_1(0{,}09)^5 = \gamma_5 \pm |\varepsilon_{\widehat{\gamma_5}}|$, $M\varepsilon_{\widehat{\gamma_5}} = 0$, $D\varepsilon_{\widehat{\gamma_5}} = 1{,}788188 \cdot 10^{-15}$.

Якщо вважати $\widehat{b_1}$ випадковою величиною, то

$$\widehat{b_1} = b_1 + \frac{\varepsilon_{\widehat{\gamma_5}}}{10 \cdot (0{,}09)^5},$$

звідки $M\widehat{b_1} = b_1$, $D\widehat{b_1} = \left(\frac{1}{10 \cdot (0{,}09)^5}\right)^2 \cdot 1{,}788188 \cdot 10^{-15} = 5{,}128473 \cdot 10^{-7}$.

Таким чином, $D\widehat{b_1}$ на основі закону трьох сігм, гарантує задану точність оцінки коефіцієнтів b_1.

$$\widehat{b_1} = \frac{\widehat{\gamma_5}}{10 \cdot (0{,}09)^5} = 2{,}001 \approx 2.$$

Решта кроків робиться аналогічно для знаходження оцінок коефіцієнтів БПР. При цьому будуть використані наступні агреговані оператори для відповідних

коефіцієнтів БПР: другий крок для b_2 (другий АО), третій крок для b_4 (другий АО), четвертий крок для b_3 (перший АО), п'ятий крок для b_6 (перший АО), шостий крок для b_5 (перший АО), сьомий крок для b_7 (другий АО), восьмий крок для b_8 (другий АО), дев'ятий крок для b_9 (четвертий АО).

Необхідно знати значення випадкової величини E (таблиця 2.4) та провести ряд активних експериментів (таблиця 2.5). У таблиці 2.5 від значень активного експерименту було віднято члени БПР, оцінки яких було знайдено на попередніх кроках відповідно до (1.12). Результати обчислень решти кроків ілюстративного прикладу, знайдені з необхідною точністю, наведені у таблиці 2.6.

Таблиця 2.4 – Реалізації випадкової величини E, що використовувались відповідно до кроків 2–9 в імітації основного активного експерименту

Крок 2	Крок 3	Крок 4
–2,195369811977885	0,807950141301234	–0,6893135072283657
–0,12063419142686603	2,6127415396270233	–1,1931899114947138
–1,7907751802465064	1,9923563564870557	3,0249102665371534
–2,1026649535817645	–0,005506315074662242	0,5270108534316472
–1,5447658191004847	1,2290579479034371	1,7601025580636376
–0,2002561262551154	0,1419919936945139	2,666542829364775
2,6798399955112573	0,7288220117347707	3,0458456722762604
–2,7977613703718434	–0,5038414467160576	2,130621053328643
2,4754704037663915	–0,24207886186565378	–2,6190016516637336
1,056266088850859	1,4330913734546347	–0,2229956051703742
Крок 5	Крок 6	Крок 7
–0,6893135072283657	1,540866648948249	–1,0904479093952715
–1,1931899114947138	1,869030207418431	0,9699664610876225
3,0249102665371534	–2,4896342020289994	3,2737368583418776
0,5270108534316472	–1,3621035826330083	0,6159984517967693
1,7601025580636376	–1,3059043929109533	–1,4712753850010287
2,666542829364775	2,3330867000197006	1,7289268018974024
3,0458456722762604	0,12292969415601586	–1,2971084475562134
2,130621053328643	1,6969924752545953	–0,7872578872282519
–2,6190016516637336	0,5081198977551179	0,7516094848925081
–0,2229956051703742	1,0298668767026709	0,3787300161358095
Крок 8	Крок 9	
–1,789867368121218	–1,789867368121218	
1,6833116410256515	1,6833116410256515	
–0,8595534529753378	–0,8595534529753378	
–0,7063717798945163	–0,7063717798945163	
–0,5767283796110184	–0,5767283796110184	
–0,8432436883496297	–0,8432436883496297	
–1,2037914210764402	–1,2037914210764402	
1,1739094496264404	1,1739094496264404	
–3,719826795861956	–3,719826795861956	
2,5730133007139497	2,5730133007139497	

Таблиця 2.5 – Значення імітації активного експерименту, що використовувались відповідно до кроків 2–9

Крок 2	Крок 3	Крок 4
3229,804630188022115	253,807950141301234	210510,31068649277163
10365,615699126978254	1871,8317901285396633	211019,85561072850529
21657,941305664153414	6170,8707171572852957	211734,00871030653715
37175,956222754020396	14473,639263268172858	212641,58761121343165
56975,553239608400765	28103,065566252930937	213752,82060280806364
81115,483149376243635	48371,940301948667014	215063,59604307936477
109660,8340080271091	76607,769358722887251	216573,94924603227626
142662,90735953402824	114129,3482006740857	218283,16682109332864
180191,29084506616127	162248,44867929482171	220188,22019898833627
222302,05626608885086	222302,43309137345463	222300,77700439482963
Крок 5	Крок 6	Крок 7
1750,3106864927716343	3,1252022540866648948e7	31252,909552090604729
6250,3668292887612862	3,125904383066860793e7	1,0002259891672675045e6
16753,734912066533153	3,1278057969969397963e7	7,59360014971185824l9e6
36253,027032453539647	3,1315082519139617583e7	3,2002689704319884609e7
67754,685121308126138	3,1376103445133107214e7	9,7661807464882221569e7
114250,91656532930227	3,1467117932131699895e7	2,4299124585327594065e8
178749,57588347216826	3,159413230180529394e7	5,2519873962976431055e8
264255,07062585333264	3,1763166537402075263e7	1,0240170243183424272e9
373738,58108474808027	3,1980162066692697243e7	1,8452059634734396481e9
510250,77700439482963	3,2251202029866876703e7	3,1250102013787300161e9
Крок 8	Крок 9	
29,210132631878782	999,210132631878782	
227,40665185797680313	4003,0033180410256515	
1076,7774913261338523	9000,0204469470246622	
3671,6461235321914843	16000,773631820105484	
9923,7114587954634418	25000,923274120388982	
22812,61421425043517	35999,55675881165037	
46625,709327350683961	48998,95621217892356	
87195,298956262082287	64002,49390984962644	
152100,87059242455957	80995,840179604138044	
251003,57301330071395	100003,57301330071395	

Таблиця 2.6 – Результати обчислення ілюстративного прикладу

Коефіцієнт, що оцінюється b_j	Значення вхідних змінних $x_1 - x_6$	Значення параметрів $\bar{a}, \bar{b}$	Рівняння для оцінки коефіцієнта при нелінійному члені БПР	Знайдена оцінка коефіцієнта $\widehat{b_j}$	Дисперсія оцінки коефіцієнта $D\widehat{b_j}$
b_2	$x_1 = \overline{a_1}z + \overline{b_1}$, $x_2 = \overline{a_2}z + \overline{b_2}$, $x_3 = 1$, $x_4 = 10$, $x_5 = 1$, $x_6 = 1$	$\overline{a_1} = 0{,}09$ $\overline{b_1} = 5{,}5$ $\overline{a_2} = 0{,}09$ $\overline{b_2} = 5{,}5$	$\widehat{b_2} \cdot a_1^2 \cdot a_2^2 \cdot 10 = \widehat{y_4}$	-0,0003	$2{,}428001 \cdot 10^{-6}$

Продовження таблиці 2.6

Коефіцієнт, що оцінюється b_j	Значення вхідних змінних $x_1 - x_6$	Значення параметрів $\bar{a}, \bar{b}$	Рівняння для оцінки коефіцієнта при нелінійному члені БПР	Знайдена оцінка коефіцієнта $\widehat{b_j}$	Дисперсія оцінки коефіцієнта $D\widehat{b_j}$
b_4	$x_1 = 10,$ $x_2 = \overline{a_2}z + \overline{b_2},$ $x_3 = 1,$ $x_4 = \overline{a_4}z + \overline{b_4},$ $x_5 = 1,$ $x_6 = 1$	$\overline{a_2} = 0{,}09$ $\overline{b_2} = 5{,}5$ $\overline{a_4} = 0{,}09$ $\overline{b_4} = 5{,}5$	$\widehat{b_4} \cdot a_2^2 \cdot a_4^2 \cdot 10 = \widehat{\gamma_4}$	-0,0001	$2{,}428001 \cdot 10^{-6}$
b_3	$x_1 = 10,$ $x_2 = \overline{a_2}z + \overline{b_2},$ $x_3 = 1,$ $x_4 = 10,$ $x_5 = 1,$ $x_6 = 1$	$\overline{a_2} = 0{,}09$ $\overline{b_2} = 5{,}5$	$\widehat{b_3} \cdot a_2^2 \cdot 10 \cdot 10 = \widehat{\gamma_2}$	0,9961	$1{,}096121 \cdot 10^{-5}$
b_6	$x_1 = 10,$ $x_2 = 10,$ $x_3 = \overline{a_3}z + \overline{b_3},$ $x_4 = 1,$ $x_5 = 0,$ $x_6 = 5$	$\overline{a_3} = 0{,}09$ $\overline{b_3} = 5{,}5$	$\widehat{b_6} \cdot a_3^3 \cdot 10 \cdot 10 \cdot 5 = \widehat{\gamma_3}$	0,9996	$1{,}420843 \cdot 10^{-7}$
b_5	$x_1 = 1,$ $x_2 = 1,$ $x_3 = \overline{a_3}z + \overline{b_3},$ $x_4 = 10,$ $x_5 = 5,$ $x_6 = 1$	$\overline{a_3} = 0{,}09$ $\overline{b_3} = 5{,}5$	$\widehat{b_5} \cdot a_3^3 \cdot 10^2 \cdot 5 = \widehat{\gamma_3}$	1,9995	$1{,}420843 \cdot 10^{-7}$
b_7	$x_1 = \overline{a_1}z + \overline{b_1},$ $x_2 = 1,$ $x_3 = 1,$ $x_4 = \overline{a_4}z + \overline{b_4},$ $x_5 = 5,$ $x_6 = 1$	$\overline{a_1} = 0{,}09$ $\overline{b_1} = 5{,}5$ $\overline{a_4} = 0{,}09$ $\overline{b_4} = 5{,}5$	$\widehat{b_7} \cdot a_1^2 \cdot a_4^3 \cdot 5^6 = \widehat{\gamma_5}$	2	$2{,}100622 \cdot 10^{-13}$

Продовження таблиці 2.6

Коефіцієнт, що оцінюється b_j	Значення вхідних змінних $x_1 - x_6$	Значення параметрів $\overline{a}, \overline{b}$	Рівняння для оцінки коефіцієнта при нелінійному члені БПР	Знайдена оцінка коефіцієнта $\widehat{b_j}$	Дисперсія оцінки коефіцієнта $D\widehat{b_j}$
b_8	$x_1 = 10,$ $x_2 = 1,$ $x_3 = 1,$ $x_4 = \overline{a_4}z + \overline{b_4},$ $x_5 = 1,$ $x_6 = \overline{a_6}z + \overline{b_6}$	$\overline{a_4} = 0{,}09$ $\overline{b_4} = 5{,}5$ $\overline{a_6} = 0{,}04$ $\overline{b_6} = 3$	$\widehat{b_8} \cdot a_6^2 \cdot a_4^3 \cdot 10 = \widehat{\gamma_5}$	1,0062	$1{,}314371 \cdot 10^{-5}$
b_9	$x_1 = 10,$ $x_2 = \overline{a_2}z + \overline{b_2},$ $x_3 = 10,$ $x_4 = \overline{a_4}z + \overline{b_4},$ $x_5 = 1,$ $x_6 = 5$	$\overline{a_2} = 0{,}09$ $\overline{b_2} = 5{,}5$ $\overline{a_4} = 0{,}09$ $\overline{b_4} = 5{,}5$	$\widehat{b_9} \cdot a_2 \cdot a_4 \cdot 10 \cdot 10 \cdot 5 = \widehat{\gamma_2}$	1,9999	$4{,}384485 \cdot 10^{-7}$

Висновки до розділу

У рамках даного розділу було виконано модифікації декомпозиційного методу, який входить до складу синтетичного методу побудови БПР в частині формалізації загальної процедури створення індивідуальних алгоритмів. Було запропоновано чотири агреговані оператори, які аналізують надлишковий опис БПР та визначають дисперсії оцінок коефіцієнтів її нелінійних членів. Це дозволяє ефективно проводити активний експеримент для вже відомих вхідних даних, а також визначити необхідну кількість повторів даного активного експерименту.

Було обґрунтовано вибір універсальної мови програмування з метою найбільш ефективного розширення її можливостей для автоматизації створення програмної реалізації індивідуальних алгоритмів декомпозиційного методу знаходження оцінок коефіцієнтів БПР з заданою точністю. Було описано ініціалізації та макровизначення основних складових програмного розширення

мови Python для автоматизації процесу створення індивідуальних алгоритмів декомпозиційного методу.

В частині підвищення ефективності ММГУА для побудови БПР, що входить до складу синтетичного методу, було запропоновано використання кількох регулярних критеріїв прийняття структури БПР. Було запропоновано загальний критерій, який обирає одну із структур визначених за допомогою регулярних критеріїв. Крім того, запропоновано паралельну реалізацію ММГУА та алгоритм управління асинхронними обчислювальними процесами для неї.

Розроблено методології створення індивідуальних алгоритмів та розширення мови Python для автоматизації створення та програмної реалізації індивідуальних алгоритмів побудови БПР.

Наведено ілюстративний приклад побудови БПР, заданої надлишковим описом, на основі проведення активного експерименту, який демонструє можливості агрегованих операторів побудови індивідуального алгоритму декомпозиційного методу.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ СИНТЕТИЧНОГО МЕТОДУ

3.1 Технічні рішення та обґрунтування вибору засобів розробки програмного забезпечення підвищення ефективності синтетичного методу

Програмне забезпечення підвищення ефективності синтетичного методу буде представлено у вигляді окремої бібліотеки, яку можна підключати до проєктів, що розробляються за допомогою мови програмування Python. Обґрунтування вибору мови Python наведене у підрозділі 2.2. Бібліотека повинна містити у собі весь необхідний функціонал, що реалізує теоретичні викладки підвищення ефективності синтетичного методу.

Архітектуру бібліотеки буде обрано монолітною [37, 38], даний вибір обумовлений тим, що монолітну архітектуру доцільно використовувати у невеликих бібліотеках вузькоцільового призначення [38], а також через те, що інші архітектури ПЗ (клієнт-серверна, сервісно-орієнтована, мікросервісна, багаторівнева) не використовуються при розробці подібних бібліотек через їх громіздкість та наявність складових не притаманним програмним бібліотекам [37].

Структура бібліотеки буде розроблена з використанням патерну Component-Based Software Engineering (CBSE), оскільки даний патерн рекомендований при розробці ПЗ вузькоцільового призначення та дозволяє перевикористання окремих складових ПЗ при розробці інших складових у рамках даного ПЗ або іншого [39].

Патерн CBSE передбачає те, що внутрішня структура бібліотеки буде поділена на окремі компоненти, кожен з яких розв’язує окрему підзадачу та являє собою набір пов’язаних функцій. Окремі компоненти можуть використовуватись всередині інших компонентів для уникнення дублювання програмного коду та відповідно виникнення архітектурних антипатернів [39]. Компоненти розділені на окремі групи, пов’язані спільною логікою відповідно до структури патерна CBSE. Загальну структуру патерну CBSE можна побачити на рисунку 3.1 [39].

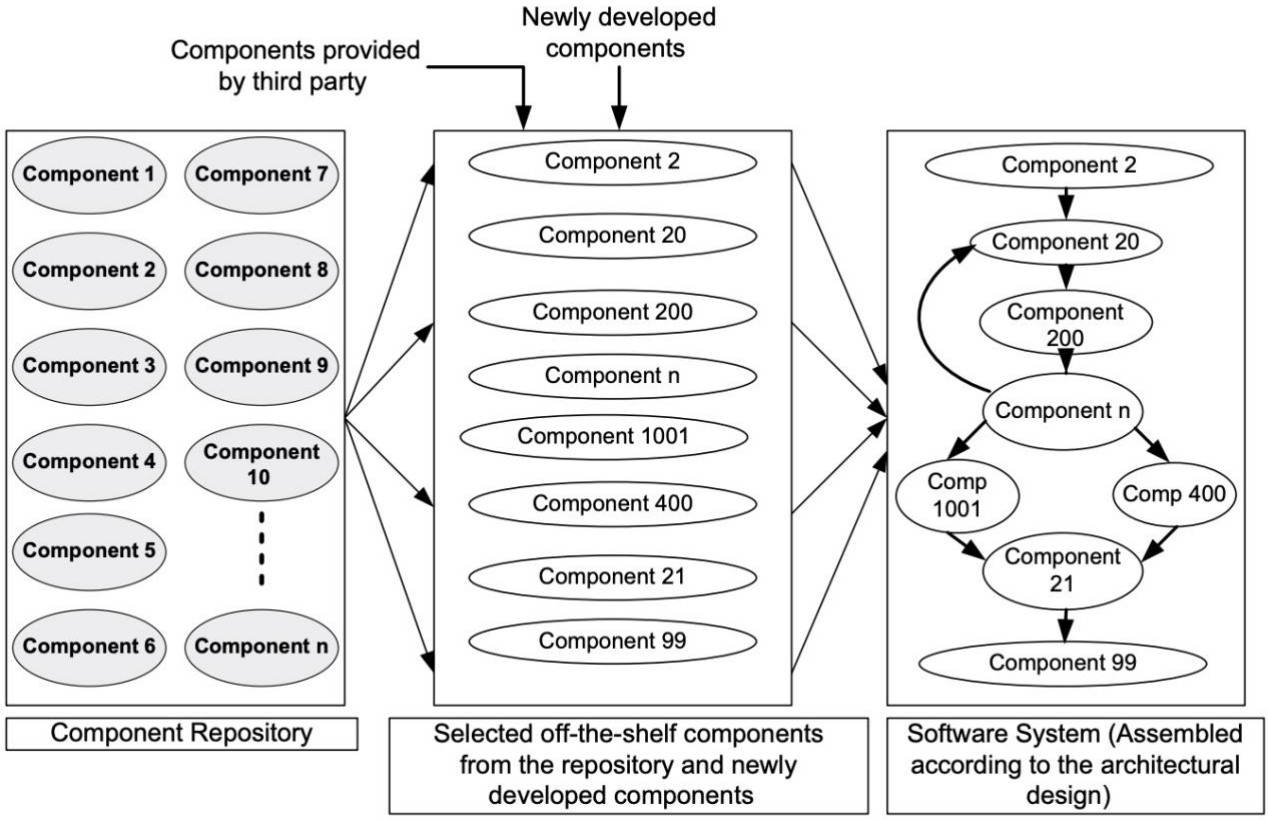


Рисунок 3.1 – Загальна архітектура патерну CBSE [39]

Бібліотека, яка містить реалізацію синтетичного методу та його модифікацій (інструменти створення індивідуальних алгоритмів декомпозиційного методу, підвищення ефективності ММГУА), буде включати набір компонентів, кожен з яких реалізує окремий підалгоритм синтетичного методу. Крім того є необхідність у використанні допоміжних компонентів, що включають власноруч розроблені та зовнішні (обробка помилок, символьні обчислення).

Бібліотека, що реалізує синтетичний метод з усіма модифікаціями, буде представлена компонентом з чистим програмним інтерфейсом з можливістю вбудови у цільове програмне забезпечення. Загальний вигляд архітектури бібліотеки буде представлено у підрозділі 3.3.

Середовищем розробки бібліотеки обрано PyCharm [40] через наявність community версії, а також зручність встановлення допоміжних програмних пакетів, бібліотек та розширень [40].

Одна з модифікацій ММГУА (див. підрозділ 2.4) вимагає використання паралельних обчислень при реалізації окремих підалгоритмів. У [4] було обґрунтовано, що для реалізації паралельних обчислень у алгоритмах подібних до алгоритмів реалізації ММГУА, для мови програмування Python, доцільно використовувати multiprocessing [4].

Multiprocessing [4] – це пакет для мови програмування Python, що дозволяє створювати та керувати процесами, які працюють паралельно з використанням усіх доступних ядер процесора. Найчастіше даний пакет використовується для виконання прив’язаних до процесора задач, швидкість розв’язання яких підвищується у результаті багатоядерної обробки. До таких задач відноситься обчислення часткових описів у ММГУА.

Для розгортання бібліотеки використовуватиметься система управління пакетами pip [41], яка є зручним, універсальним та найбільш відомим інструментом для керування Python-пакетами [42]. Це дозволить користувачам легко завантажувати та встановлювати бібліотеку у цільовому середовищі розробки за допомогою команди pip install synthetic_reg.

3.2 Аналіз вимог до програмного забезпечення підвищення ефективності синтетичного методу

Програмне забезпечення буде представлено у вигляді програмної бібліотеки, яка містить основну функцію-інтерпретатор текстових .poly файлів, що містять скрипти індивідуальних алгоритмів декомпозиційного методу, run_regression_script. Програмна бібліотека повинна містити функцію виклику ММГУА не у складі синтетичного методу. Крім того, повинно забезпечуватись виконання допоміжних функцій, пов’язаних з парсером .poly файлів, їх лексичним,

синтаксичним та семантичним аналізом, а також необхідно передбачити функції виконання підалгоритмів декомпозиційного методу (в тому числі побудови ОПР) та ММГУА.

Нижче буде наведено короткий опис основних функціональних вимог до програмної бібліотеки.

Функція parse_line

Призначення: аналізує окремий рядок скрипту індивідуального алгоритму, усуває зайві символи пробілу в рядку, ігнорує порожні рядки та коментарі, а потім розбиває рядок на окремі команди (присвоєння чи інші операції) для подальшої обробки.

Вхідні параметри:

- line – рядок тексту скрипту .poly файлу.

Повертає:

- не повертає значення; у разі коректної роботи: здійснює послідовний виклик функції parse_assignment для кожної команди;
- у разі помилки: повідомлення про невірний формат рядка з вказанням некоректного фрагмента скрипта.

Функція parse_assignment

Призначення: інтерпретує окреме присвоєння або команду (наприклад, виклик функції print або dec_method) з рядка скрипту.

Вхідні параметри:

- assignment – рядок, що містить команду або присвоєння.

Повертає:

- у разі коректної роботи: оновлює словник vars_dict або виконує відповідні дії (наприклад, вивід результату або виклик підфункцій декомпозиційного методу);
- у разі помилки: повідомлення про помилку (наприклад, повторна ініціалізація змінної, невідомий формат команди) з зазначенням позиції помилки.

Функція validate_value

Призначення: виконує валідацію та перетворення значення, що присвоюється змінним, враховуючи специфіку кожної з них (наприклад, для змінних r, n, m, матриць тощо).

Вхідні параметри:

– key – назва змінної, що перевіряється;

– value – значення, яке потребує валідації.

Повертає:

– у разі коректної роботи: оброблюване значення (число, масив або інший обчислений результат);

– у разі помилки: повідомлення про помилку з детальним описом.

Функція sample_x

Призначення: генерує список числових значень рівномірно розподілених між заданими початковим і кінцевим значеннями.

Вхідні параметри:

– start – початкове значення;

– end – кінцеве значення;

– n – кількість точок розподілу.

Повертає:

– у разі коректної роботи: список з n елементів (дійсних чисел), що рівномірно розподілені між start та end;

– у разі помилки: порожній список або повідомлення про некоректне значення параметра n.

Функція q_matrix_calc

Призначення: обчислює матрицю коефіцієнтів q_matrix, список обчислених поліномів q та дисперсії var_tr для кожного полінома, використовуючи дані зі змінних (наприклад, z_1, z_n, n та ін.) із словника vars_dict.

Вхідні параметри:

- q_matrix – вихідний параметр або початкова матриця, що використовується для обчислень (функція використовує додаткові параметри з vars_dict).

Повертає:

- у разі коректної роботи: кортеж з трьох елементів: q_matrix – матриця коефіцієнтів (масив дійсних чисел), q – список обчислених поліномів (символьних виразів), var_tr – список дисперсій для відповідних поліномів;
- у разі помилки: повідомлення з описом ситуації, що виникла при виконанні обчислень (наприклад, через некоректні значення в vars_dict).

Функція var_theta

Призначення: обчислює дисперсію для конкретного полінома або стовпця з матриці q за допомогою заданого параметра l.

Вхідні параметри:

- q – матриця коефіцієнтів або значень;
- l – число (параметр, що враховується при обчисленні);
- p – порядковий номер (індекс) елемента в матриці q, для якого проводиться обчислення.

Повертає:

- у разі коректної роботи: дійсне число – розраховану дисперсію для p-го елемента;
- у разі помилки: некоректне значення або повідомлення про невідповідність розмірностей.

Функція ab_calc

Призначення: обчислює матрицю коефіцієнтів ab_matrix для кожної вхідної змінної на основі її діапазону значень та заданих параметрів z_1 і z_n.

Вхідні параметри:

– ab_in_str – рядок у форматі ab_calc(input_vars_ranges_matrix, z_1, z_n), що містить параметри для обчислення коефіцієнтів.

Повертає:

– у разі коректної роботи: двовимірний список ab_matrix, де для кожної змінної записані два значення: крок (співвідношення різниці границь до інтервалу) та зміщення;

– у разі помилки: ValueError з зазначенням причини виникнення помилки (наприклад, надлишкова кількість параметрів або відсутність необхідних даних).

Функція input_matrix_val

Призначення: ініціалізує та заповнює матрицю вхідних даних (input_matrix) згідно з параметрами, що передаються в рядку ініціалізації, враховуючи як числові значення, так і функції типу xz_ab.

Вхідні параметри:

– inp – рядок у форматі input_matrix_init([...]), що містить список параметрів для заповнення стовпців матриці.

Повертає:

– у разі коректної роботи: матрицю вхідних даних, де кількість рядків відповідає значенню n із vars_dict;

– у разі помилки: при відсутності ініціалізації змінних (наприклад, n або ab_matrix) або невірному форматі рядка – ValueError із відповідним описом.

Функція split_params

Призначення: розбиває рядок із параметрами на окремі елементи, зберігаючи цілісність вкладених параметрів у круглих дужках як параметр макроса (наприклад, input_matrix_init([...])).

Вхідні параметри:

– s – рядок, що містить параметри, розділені комами.

Повертає:

– у разі коректної роботи: список рядків, де кожен елемент є окремим параметром;

– у разі помилки: список із одного елементу або некоректно розділені параметри, що можуть бути оброблені в подальшому.

Функція y_act_reader

Призначення: зчитує з текстового файлу множину значень y_act, порівнює їх з рядками вхідної матриці та формує список з відповідних числових значень.

Вхідні параметри:

– y_act_str – рядок у форматі y_act_reader(file_path.txt, input_matrix), де вказано шлях до файлу та input_matrix.

Повертає:

– у разі коректної роботи: список значень y_act, довжина якого повинна відповідати кількості рядків input_matrix;

– у разі помилки: ValueError з зазначенням місця помилки, якщо файл не знайдено або значення y_act не співпадають з усіма рядками матриці.

Функція execute_dec_method

Призначення: виконує алгоритм декомпозиційного методу (dec_method) відповідно до команди, що містить ім’я методу (наприклад, _ao1, _ao2, _ao3, _ao4 або стандартний варіант) та параметри для обчислення оцінок коефіцієнтів і їх дисперсії.

Вхідні параметри:

– value – рядок у форматі dec_method(…) або dec_method<name>(…), що містить параметри: вхідну матрицю, вектор y_act (опціонально), l (опціонально), матрицю опису та індекс стовпця.

Повертає:

– у разі коректної роботи: оновлений словник result_dict – зберігає оцінки коефіцієнтів (ключі у форматі d<index>) та (чи) їх дисперсії;

– у разі помилки: ValueError із відповідним описом і вказує позицію помилки при невірному форматі або невідомому методі.

Функція split_params_dec

Призначення: розбиває рядок параметрів для функції execute_dec_method, зберігаючи цілісність параметрів, що містять індекси масивів у квадратних дужках.

Вхідні параметри:

– param_str – рядок з параметрами, розділеними комами.

Повертає:

– у разі коректної роботи: список параметрів, де кожен елемент – окремий параметр з коректно виділеними назвами змінних та індексами;

– у разі помилки: неповний список або некоректно розділені параметри.

Функція handle_print_command

Призначення: обробляє команду виводу даних (print), відображаючи в терміналі значення змінних із vars_dict або результати з result_dict.

Вхідні параметри:

– line – рядок у форматі print(parameter), де parameter – це назви змінних або ключове слово result_dict.

Повертає:

– не повертає значення; у разі коректної роботи: здійснює вивід інформації в термінал;

– у разі помилки: ValueError з зазначенням проблемного рядка при невірному форматі команди.

Функція calculate_div_for_dec

Призначення: обчислює коефіцієнт або дисперсію окремо члена БПР на основі заданих вхідних параметрів.

Вхідні параметри:

– input_matrix – матриця вхідних даних;

– redundant_description_matrix – матриця, що містить надлишковий опис БПР;

– col_id – індекс стовпця (член надлишкового опису), для якого виконується значення оцінки коефіцієнта БПР або його дисперсії.

Повертає:

– у разі коректної роботи: дійсне число – обчислене значення оцінки коефіцієнта БПР або його дисперсії, скориговане за допомогою відповідних параметрів (наприклад, з використанням l та значень з ab_matrix);

– у разі помилки: ValueError з зазначенням місця помилки, при некоректних значеннях у матрицях або помилки, що виникла під час обчислень.

Функція least_sqares

Призначення: обчислює оцінки коефіцієнтів лінійної моделі методом найменших квадратів.

Вхідні параметри:

– matrix – матриця значень вхідних змінних;

– y – вектор спостережень.

Повертає:

– у разі коректної роботи: вектор оцінок коефіцієнтів;

– у разі помилки: повідомлення про неможливість обчислення.

Функція error

Призначення: обчислює сумарну похибку моделі, порівнюючи прогнозовані значення з фактичними для двох наборів даних.

Вхідні параметри:

– matrix – матриця значень вхідних змінних;

– y1, y2 – два набори спостережень;

– teta1, teta2 – відповідні оцінки коефіцієнтів.

Повертає:

– у разі коректної роботи: вектор значень сумарної похибки;

– у разі помилки: повідомлення про помилку через невідповідність розмірностей матриці – matrix та векторів – y1, y2.

Функція com_norm

Призначення: обчислює значення регулярного критерія $ЗСК_1$.

Вхідні параметри:

– matrix – матриця значень вхідних змінних;

– teta1 – відповідні оцінки коефіцієнтів;

– yA1 – набір спостережень (перевірочна послідовність).

Повертає:

– у разі коректної роботи: числове значення критерія;

– у разі помилки: повідомлення про помилку через невідповідність розмірностей матриці – matrix та вектора – yA1.

Функція calc_best_ind

Призначення: для заданого набору членів БПР, що точно містять ненульові коефіцієнти (належать класу M_1), та переліку можливих комбінацій інших членів БПР визначає кращий залишковий опис за критерієм мінімізації похибки.

Вхідні параметри:

– matrix – матриця значень вхідних змінних;

– certanly_non_zero_index – список членів БПР, що належать класу M_1;

– comb – перелік можливих комбінацій членів БПР, що належать класу M_2;

– operations – набір функцій, які обчислюють значення вхідної матриці у відповідності членів БПР;

– yA2 – набір спостережень для знаходження оцінок коефіцієнтів БПР;

– yA1 (опціонально) – додатковий набір спостережень (перевірочна послідовність);

– method – функція для обчислення помилки (регулярний критерій, за замовчуванням використовується функція error, $ЗСК_1$).

Повертає:

– у разі коректної роботи: список членів БПР, що формують кінцеву структуру;

– у разі помилки: повідомлення про помилку через невідповідність розмірностей матриці – matrix та векторів – yA2, yA1, чи некоректні дані.

Функція zsk_mse

Призначення: обчислює оціночні значення випадкової величини похибки вектора y.

Вхідні параметри:

– matrix – матриця значень вхідних змінних;

– y – вектор спостережень;

– teta – оцінки коефіцієнтів результуючої моделі БПР.

Повертає:

– у разі коректної роботи: оціночне числове значення випадкової величини похибки вектора y;

– у разі помилки: повідомлення про помилку при невідповідності розмірностей матриці – matrix та вектора y.

Функція zsk_norm

Призначення: обчислює значення регулярного критерія $ЗСК_2$.

Вхідні параметри:

– matrix – матриця значень вхідних змінних;

– teta2 – відповідні оцінки коефіцієнтів;

– yA2 – набір спостережень (навчальна послідовність).

Повертає:

– у разі коректної роботи: числове значення критерія;

– у разі помилки: повідомлення про помилку через невідповідність розмірностей матриці – matrix та вектора – yA2.

Функція sqrt_of_sum

Призначення: обчислює значення симетричного регулярного критерія $ЗСК_1 + ЗСК_2$.

Вхідні параметри:

– matrix – матриця значень вхідних змінних;

– teta1, teta2 – відповідні оцінки коефіцієнтів;

– yA1, yA2 – набір спостережень (навчальна послідовність).

Повертає:

– у разі коректної роботи: числове значення критерія;

– у разі помилки: повідомлення про помилку через невідповідність розмірностей матриці – matrix та векторів – yA1, yA2.

Функція xi_square

Призначення: обчислює значення критерія χ^2 для значень оцінок випадкової величини похибки вектора y по кожній з трьох моделей БПР (отриманих регулярними критеріями: $ЗСК_1$, $ЗСК_2$, $ЗСК_1 + ЗСК_2$).

Вхідні параметри:

– E1, E2, E3 – вектори значень оцінок випадкової величини похибки вектора y.

Повертає:

– у разі коректної роботи: модель з мінімальним значенням критерія χ^2;

– у разі помилки: повідомлення про потрапляння критерія у критичну область.

Функція run_mgmodh

Призначення: дозволяє користувачу програмної бібліотеки використовувати ММГУА без декомпозиційного методу для побудови БПР в умовах активного та пасивного експериментів.

Вхідні параметри:

– n – число спостережень у активному чи пасивному експерименті;

– m – число вхідних змінних у надлишковому описі;

– input_matrix – матриця, що містить значення вхідних змінних, на яких проводиться активний експеримент чи проведено пасивний експеримент;

– p_c – значення, яке визначає розділення вектору експериментів на навчальну та перевірочну послідовності;

– y – вектор результатів активного чи пасивного експериментів;

– pool_n – трійка, яка визначає кількість потоків, що використовуються при реалізації паралельних обчислень у підалгоритмах ММГУА;

– redundant_description_matrix – надлишковий опис БПР.

Повертає:

– у разі коректної роботи: оцінки коефіцієнтів БПР, які були знайдені кожним з трьох регулярних критеріїв, обраний розв’язок задачі побудови БПР за критерієм χ^2;

– у разі помилки: повідомлення про помилку з її описом.

Нижче будуть сформовані ключові нефункціональні вимоги:

– забезпечити можливість коректної роботи програмної бібліотеки, що включає розширення можливостей мови програмування Python, для автоматизації процесу створення та програмної реалізації індивідуальних алгоритмів побудови багатовимірних поліноміальних регресій, заданих надлишковим описом, на версії Python 3.6 та вище;

– забезпечити цілісність даних у програмній бібліотеці;

– забезпечити надійність роботи програмної бібліотеки та відмовостійкість у випадку коректної обробки задекларованих особливостей роботи підалгоритмів синтетичного методу;

– відповідальність за некоректне використання програмної бібліотеки покладається на користувача;

– забезпечити дотримання лексики, синтаксису та семантики мови програмування Python під час створення індивідуальних алгоритмів побудови багатовимірних поліноміальних регресій, заданих надлишковим описом.

3.3 Проєктування архітектури програмного забезпечення підвищення ефективності синтетичного методу

Архітектура програмної бібліотеки, що реалізує програмне забезпечення підвищення ефективності синтетичного методу побудови БПР на основі повторних активних експериментів, як вже було зазначено (див. підрозділ 3.1) матиме монолітну архітектуру з використанням CBSE патерну реалізації її внутрішньої структури. Програмна бібліотека містить дві основні функції, які доступні користувачу при вбудові бібліотеки у цільове програмне забезпечення. Перша, run_regression_script (див. підрозділ 2.2), дозволяє користувачу виконувати скрипт індивідуального алгоритму побудови БПР, заданих надлишковим описом (використовує як декомпозиційний метод, так і ММГУА). Друга, run_mgmodh (див. підрозділ 3.2), дозволяє користувачу використовувати ММГУА окремо (не у складі синтетичного методу) для ефективного розв’язання задач побудови БПР, заданих надлишковим описом, на основі як активного, так і пасивного експериментів.

У програмній бібліотеці кожен алгоритм, що використовується у синтетичному методі побудови БПР, представлений у вигляді окремого компоненту. Крім того, у бібліотеці присутній ряд допоміжних компонентів: перетворення даних, інтерпретація помилок тощо.

Загальна архітектура програмної бібліотеки наведена на рисунку 3.2.

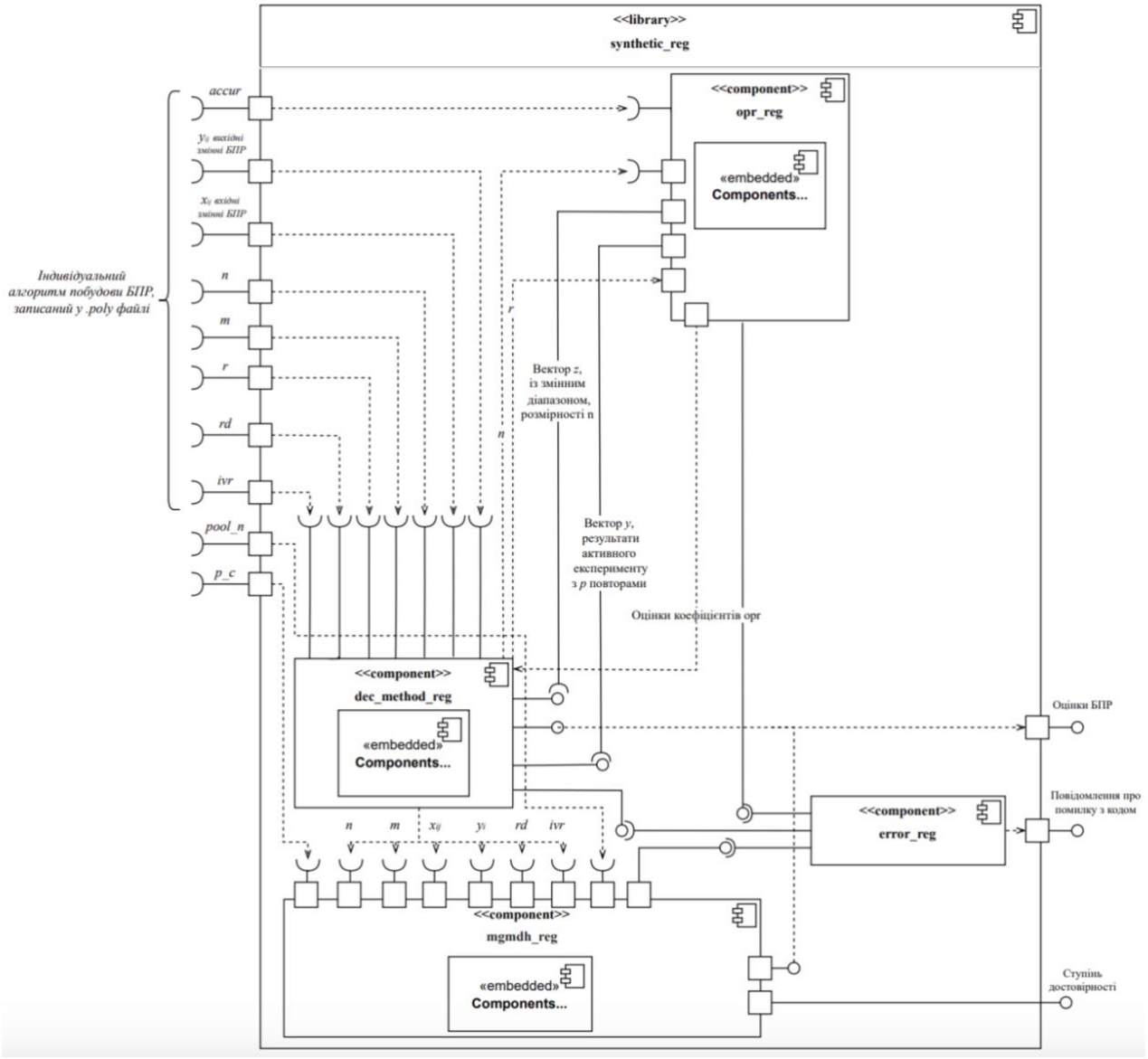


Рисунок 3.2 – Загальна архітектура програмної бібліотеки

Далі буде описано компоненти, що входять до складу програмної бібліотеки реалізації синтетичного методу побудови БПР:

– synthetic_reg – основний компонент, що представляє програмну бібліотеку, реалізує синтетичний метод побудови БПР, дозволяє створювати та виконувати індивідуальні алгоритми побудови БПР, а також окремо використовувати ММГУА;

– dec_method_reg – компонент, що реалізує декомпозиційний метод побудови БПР, представляє перший підалгоритм, який є складовою синтетичного

методу, містить функціонал автоматизації процесу створення та програмної реалізації індивідуальних алгоритмів побудови БПР, використовує компоненти opr_reg та mgmdh_reg як допоміжні;

– mgmdh_reg – компонент, що реалізує ММГУА для побудови БПР, представляє другий підалгоритм, який є складовою синтетичного методу, може використовуватись окремо для побудови БПР на основі активного чи пасивного експерименту;

– opr_reg – компонент для побудови ОПР, що використовується dec_method_reg у якості допоміжного для знаходження оцінок коефіцієнтів ОПР, детальна структура наведена у [4];

– error_reg – компонент, що обробляє усі задекларовані помилки та виводить відповідні повідомлення.

Далі будуть розглянуті оригінальні модифікації архітектури ключових компонентів програмної бібліотеки, до них належать:

– dec_method_reg – компонент, що забезпечує реалізацію декомпозиційного методу, надає можливість створення та виконання скрипту індивідуального алгоритму побудови БПР (рисунок 3.3);

– mgmdh_reg – компонент, що забезпечує реалізацію ММГУА (рисунок 3.4).

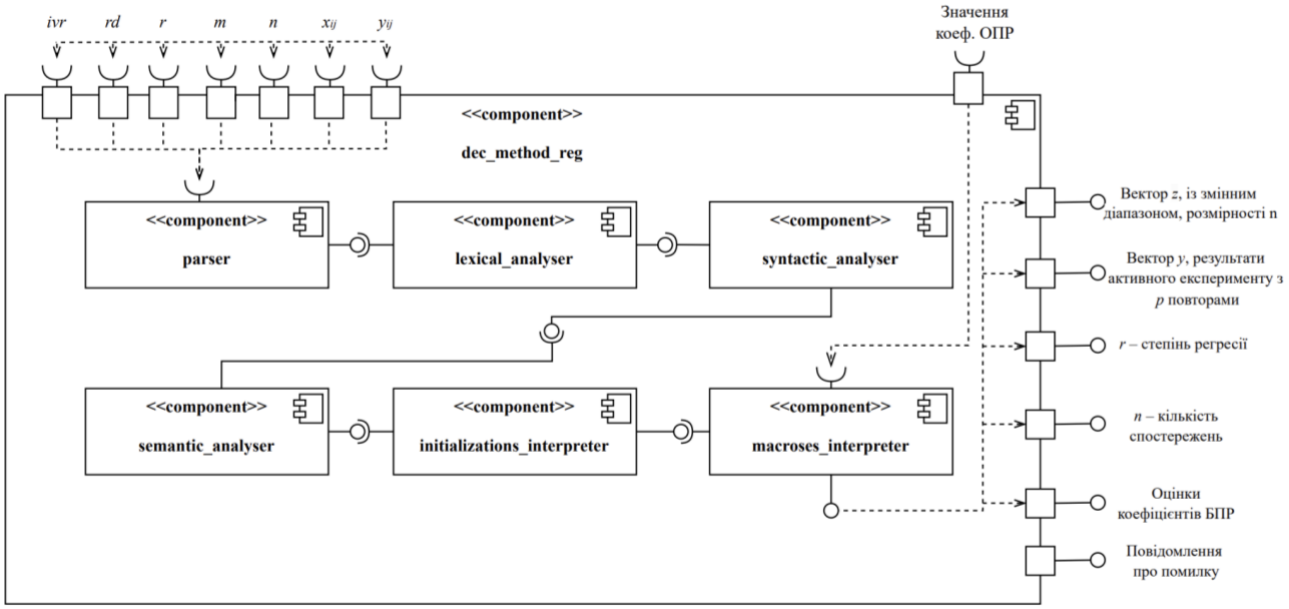


Рисунок 3.3 – Деталізована архітектура компонента dec_method_reg

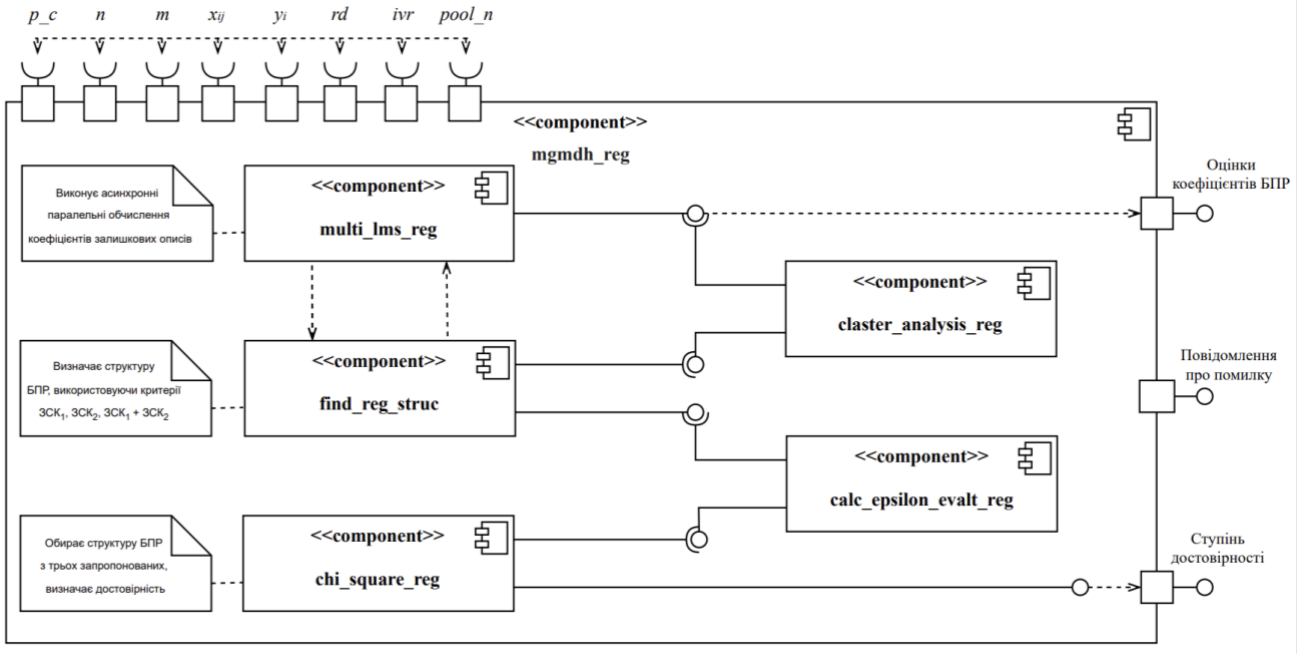


Рисунок 3.4 – Деталізована архітектура компонента mgmdh_reg

Вихідний код реалізації програмної бібліотеки наведено у додатку А.

3.4 Тестування та оцінка якості розробленого програмного забезпечення підвищення ефективності синтетичного методу

Для перевірки коректності роботи програмного забезпечення створення та програмної реалізації індивідуальних алгоритмів синтетичного методу у .poly файл буде записано індивідуальний алгоритм прикладу, що наведений у підрозділі 2.6 та за допомогою функції run_regression_script запущений на виконання. Результат виконання індивідуального алгоритму наведено на рисунку 3.5.

Вміст .poly файлу наведено у додатку Б.

Як видно з рисунку 3.5, оцінки коефіцієнтів БПР та їх дисперсії з точністю до третього знаку після коми співпадають з оцінками та їх дисперсіями знайденими вручну при розв’язанні прикладу з підрозділу 2.6.

```
Regression result:
b1 - Value: 2.0080621159, Dispersion: 6.084E-7
b2 - Value: 0.0002352935, Dispersion: 0.0000018775
b3 - Value: 0.9960767744, Dispersion: 0.0000013702
b4 - Value: 0.0008271898, Dispersion: 0.0000018775
b5 - Value: 1.9990511451, Dispersion: 1.78E-8
b6 - Value: 0.9993935878, Dispersion: 1.78E-8
b7 - Value: 2.0065638541, Dispersion: 0.0
b8 - Value: 1.0101839293, Dispersion: 0.0000155915
b9 - Value: 1.9998388638, Dispersion: 5.48E-8


Process finished with exit code 0
```

Рисунок 3.5 – Виконання індивідуального алгоритму синтетичного методу

Для оцінки якості програмного забезпечення створення та програмної реалізації індивідуальних алгоритмів синтетичного методу побудови БПР необхідно порівняти код програмної реалізації без використання програмного розширення створення та реалізації індивідуальних алгоритмів синтетичного методу та з ним за допомогою визначеного набору метрик якості.

Бібліотеку Radon [43] буде використано для оцінки якості та порівняння між собою коду різних реалізацій. Бібліотека розроблена за допомогою мови програмування Python та працює виключно з файлами *.py та їм подібним. Будуть використані метрики цикломатичної складності, Холстеда, кількості рядків коду та підтримуваності коду. На рисунках 3.6-3.9 наведено результати аналізу програмної реалізації синтетичного методу з використанням розширення та без нього по кожній з метрик.

Результати метрик за двома реалізаціями будуть представлені у таблиці 3.1 у зведеному вигляді.

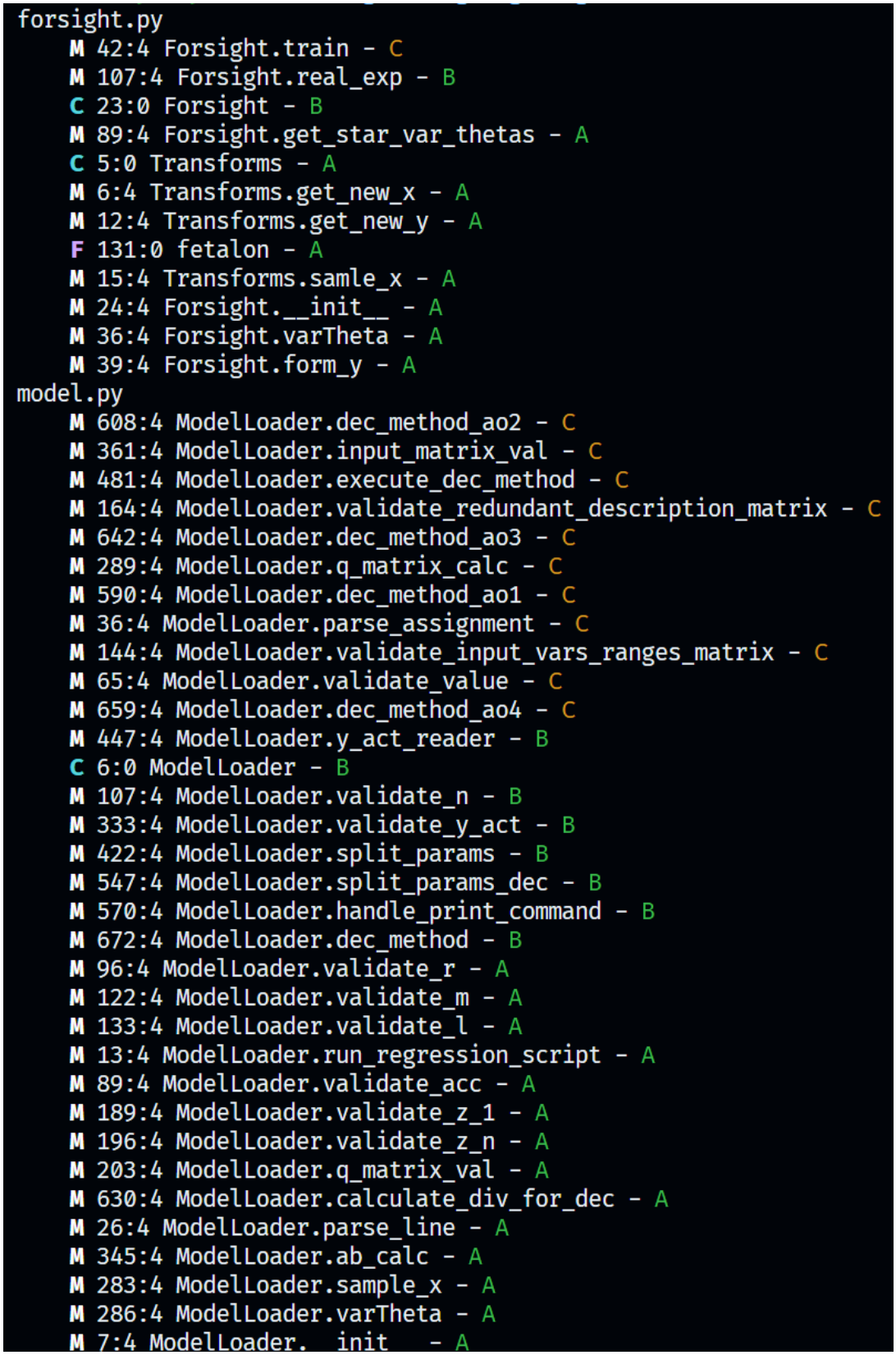


Рисунок 3.6 – Порівняння реалізацій за метрикою цикломатичної складності

```
example.py:
    h1: 1
    h2: 1
    N1: 1
    N2: 1
    vocabulary: 2
    length: 2
    calculated_length: 0.0
    volume: 2.0
    difficulty: 0.5
    effort: 1.0
    time: 0.05555555555555555
    bugs: 0.0006666666666666666
forsight.py:
    h1: 11
    h2: 115
    N1: 81
    N2: 158
    vocabulary: 126
    length: 239
    calculated_length: 825.2851036636135
    volume: 1667.5699017164802
    difficulty: 7.556521739130435
    effort: 12601.028213840185
    time: 700.0571229911214
    bugs: 0.5558566339054934
index.py:
    h1: 1
    h2: 2
    N1: 1
    N2: 2
    vocabulary: 3
    length: 3
    calculated_length: 2.0
    volume: 4.754887502163469
    difficulty: 0.5
    effort: 2.3774437510817346
    time: 0.1320802083934297
    bugs: 0.0015849625007211565
model.py:
    h1: 17
    h2: 416
    N1: 288
    N2: 537
    vocabulary: 433
    length: 825
    calculated_length: 3688.8697910479505
    volume: 7225.5341521495475
    difficulty: 10.97235576923077
    effort: 79281.13134011204
    time: 4404.507296672891
    bugs: 2.408511384049849
```

Рисунок 3.7 – Порівняння реалізацій за метрикою Холстеда

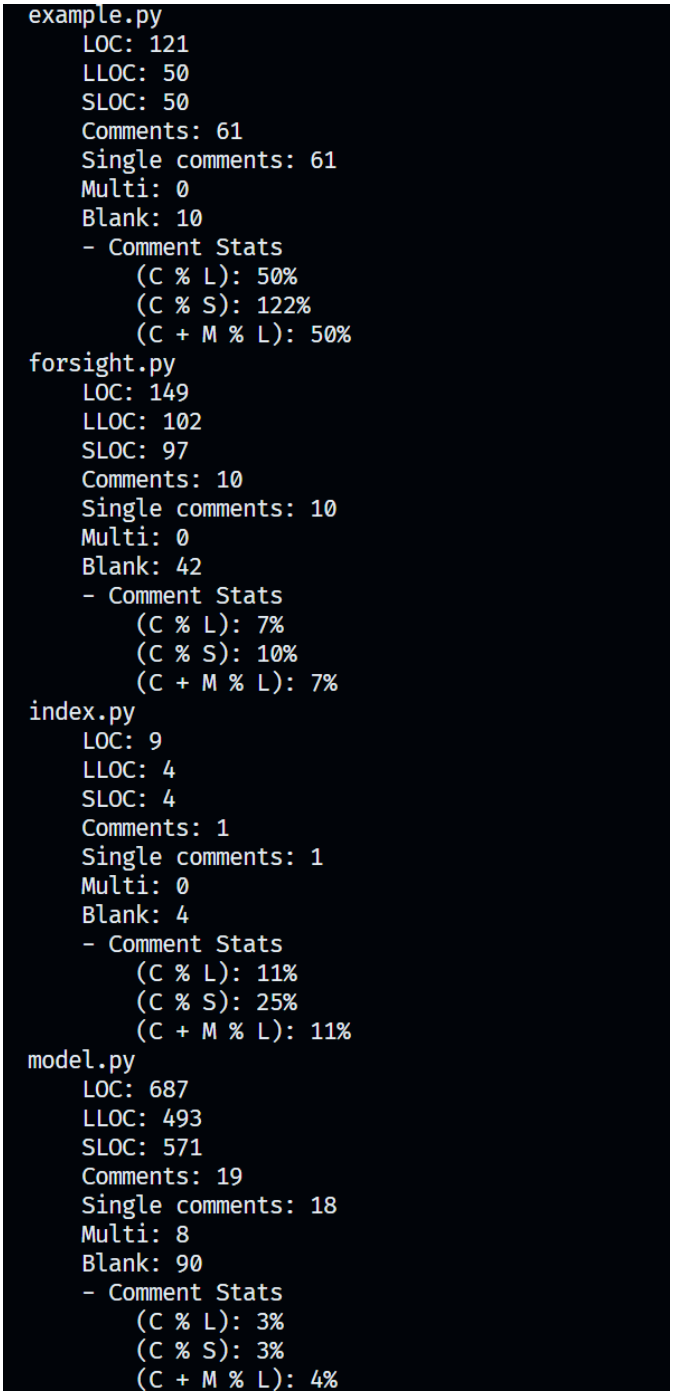


Рисунок 3.8 – Порівняння реалізацій за метрикою кількості рядків

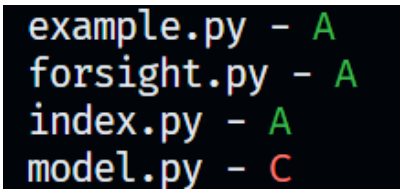


Рисунок 3.9 – Порівняння реалізацій за метрикою підтримуваності коду

Таблиця 3.1 – Порівняльна таблиця двох реалізацій за чотирма метриками

Метрика	Реалізація без використання розширення	Реалізація з розширенням
Цикломатична складність	avg ~ B	A
Холстеда	h1: 17	h1: 1
	h2: 416	h2: 1
	N1: 288	N1: 1
	N2: 537	N2: 1
	Difficulty: 10,97	Difficulty: 0,5
	Effort: 79281	Effort: 1
Кількість рядків	599	50
Підтримуваність коду	avg ~ B	A

Як видно з рисунків 3.6-3.9 та таблиці 3.1, складність реалізації синтетичного методу побудови БПР без використання програмного розширення автоматизації створення та програмної реалізації індивідуальних алгоритмів синтетичного методу на порядок вища, ніж реалізація з використанням розширення.

Крім того, варто зауважити, що для кожної оригінальної задачі побудови БПР, заданої надлишковим описом, необхідно виконувати власну реалізацію синтетичного методу (див. підрозділи 1.2-1.4).

3.5 Статистичне дослідження ефективності одночасного використання асиметричного та симетричного регулярних критеріїв у ММГУА

Для проведення статистичних досліджень ефективності одночасного використання декількох регулярних критеріїв у ММГУА буде використано методологію наведену у підрозділі 1.3, яка використовувалась для порівняння методів побудови БПР.

Обрані наступні параметри дослідження:

– надлишковий опис наступного вигляду:

$$Y(\bar{x}) = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 + b_5x_1x_2 + b_6x_1x_4 + b_7x_2^2 + b_8x_4^3 + E$$

– кількість вхідних змінних – 4;

– кількість нульових коефіцієнтів – 3;

– кількість спостережень активного експерименту – 20, 30, 40;

– кількість повторів активних експериментів – 6;

– кількість повторів перевірочної послідовності – 4;

– діапазон значень коефіцієнтів – [-10; 10];

– діапазон значень вхідних змінних – [1; 40);

– кількість незалежних реалізацій повторного експерименту – 1000.

Результати дослідження будуть представлені у таблицях 3.2-3.4 із генерацією результатів активного експерименту та перевірочної послідовності для заданого числа спостережень. Випадкова величина похибки має нормальний закон розподілу та змінне значення дисперсії від 1 до 10 для кожного з критеріїв. Для кожного з критеріїв буде проведено незалежні повтори експериментів.

Кожна комірка результату буде містити такі значення:

– correct – відсоток правильних структур;

– mean – середнє значення міри порівнянь для правильних структур;

– max – максимальне значення міри порівнянь для правильних структур;

– min – мінімальне значення міри порівнянь для правильних структур;

– avgzeros – середня кількість неправильно визначених нульових коефіцієнтів;

– maxzeros – максимальна кількість неправильно визначених нульових коефіцієнтів;

– correct_hi2 – відсоток правильних структур за критерієм χ^2, коли критерій не потрапляє в критичну область та при цьому визначає правильно структуру;

– correct_h – відсоток правильних структур за критерієм χ^2, коли критерій

потрапляє в критичну область та при цьому визначає правильно структуру;

– time – середнє значення часу побудови БПР.

Таблиця 3.2 – Результати моделювання для 10 повторів імітації основного експерименту, що складається з 20 спостережень

σ^2	Симетричний критерій $ЗСК_1 + ЗСК_2$	Несиметричний критерій $ЗСК_1$	Несиметричний критерій $ЗСК_2$	Структура розв'язку задається $min(\chi_1^2, \chi_2^2, \chi_3^2)$
1	correct: 80,2 mean: 0,002 max: 0,22696 min: 0,0 avgzeros: 0,0212 maxzeros: 2,0 time: 0,00159	correct: 46,8 mean: 0,00269 max: 0,33983 min: 0,0 avgzeros: 0,0658 maxzeros: 3,0 time: 0,00233	correct: 72,2 mean: 0,00268 max: 0,38206 min: 0,0 avgzeros: 0,0314 maxzeros: 3,0 time: 0,00108	correct: 59,2 correct_hi2: 0,364 correct_h: 0,682 mean: 0,00263 max: 0,38206 min: 0,0 avgzeros: 0,049 maxzeros: 3,0 time: 0,00131
2	correct: 80,6 mean: 0,00487 max: 0,27049 min: 0,0 avgzeros: 0,0406 maxzeros: 3,0 time: 0,00331	correct: 46,8 mean: 0,0067 max: 0,72999 min: 0,0 avgzeros: 0,1346 maxzeros: 3,0 time: 0,00485	correct: 72,3 mean: 0,00669 max: 0,72999 min: 0,0 avgzeros: 0,0622 maxzeros: 3,0 time: 0,00225	correct: 61,3 correct_hi2: 0,37 correct_h: 0,7111 mean: 0,00613 max: 0,38919 min: 0,0 avgzeros: 0,093 maxzeros: 3,0 time: 0,00263
3	correct: 78,53333 mean: 0,00855 max: 0,65165 min: 0,0 avgzeros: 0,0646 maxzeros: 3,0 time: 0,00494	correct: 45,8 mean: 0,0111 max: 0,72999 min: 0,0 avgzeros: 0,2062 maxzeros: 3,0 time: 0,00724	correct: 70,8 mean: 0,01113 max: 0,72999 min: 0,0 avgzeros: 0,0948 maxzeros: 3,0 time: 0,00335	correct: 59,6 correct_hi2: 0,36067 correct_h: 0,70394 mean: 0,01057 max: 0,65165 min: 0,0 avgzeros: 0,1462 maxzeros: 3,0 time: 0,00395
4	correct: 77,75 mean: 0,01321 max: 0,65165 min: 0,0 avgzeros: 0,0854 maxzeros: 3,0 time: 0,00657	correct: 45,55 mean: 0,01642 max: 0,72999 min: 0,0 avgzeros: 0,2746 maxzeros: 3,0 time: 0,00961	correct: 70,4 mean: 0,01635 max: 0,72999 min: 0,0 avgzeros: 0,1238 maxzeros: 3,0 time: 0,00446	correct: 59,5 correct_hi2: 0,3685 correct_h: 0,7071 mean: 0,01563 max: 0,65165 min: 0,0 avgzeros: 0,1894 maxzeros: 3,0 time: 0,00527

Продовження таблиці 3.2

σ^2	Симетричний критерій $ЗСК_1 + ЗСК_2$	Несиметричний критерій $ЗСК_1$	Несиметричний критерій $ЗСК_2$	Структура розв'язку задається $min(\chi_1^2, \chi_2^2, \chi_3^2)$
5	correct: 76,96 mean: 0,01785 max: 0,65165 min: 0,0 avgzeros: 0,1068 maxzeros: 3,0 time: 0,00838	correct: 45,48 mean: 0,02205 max: 0,72999 min: 0,0 avgzeros: 0,3434 maxzeros: 3,0 time: 0,01227	correct: 69,72 mean: 0,02143 max: 0,72999 min: 0,0 avgzeros: 0,1554 maxzeros: 3,0 time: 0,00569	correct: 58,72 correct_hi2: 0,364 correct_h: 0,70076 mean: 0,02091 max: 0,65165 min: 0,0 avgzeros: 0,238 maxzeros: 3,0 time: 0,00661
6	correct: 76,3 mean: 0,02265 max: 0,65165 min: 0,0 avgzeros: 0,1322 maxzeros: 3,0 time: 0,01008	correct: 44,2 mean: 0,02888 max: 0,72999 min: 0,0 avgzeros: 0,4196 maxzeros: 3,0 time: 0,01477	correct: 68,6 mean: 0,02768 max: 0,72999 min: 0,0 avgzeros: 0,1944 maxzeros: 3,0 time: 0,00683	correct: 58,13333 correct_hi2: 0,362 correct_h: 0,699839 mean: 0,02688 max: 0,65165 min: 0,0 avgzeros: 0,2882 maxzeros: 3,0 time: 0,00792
7	correct: 75,77143 mean: 0,02933 max: 0,65165 min: 0,0 avgzeros: 0,1558 maxzeros: 3,0 time: 0,01181	correct: 43,51429 mean: 0,03719 max: 0,72999 min: 0,0 avgzeros: 0,4958 maxzeros: 3,0 time: 0,01728	correct: 67,31429 mean: 0,03566 max: 0,72999 min: 0,0 avgzeros: 0,2358 maxzeros: 3,0 time: 0,008	correct: 57,54286 correct_hi2: 0,35829 correct_h: 0,69786 mean: 0,03533 max: 0,69545 min: 0,0 avgzeros: 0,3406 maxzeros: 3,0 time: 0,00925
8	correct: 75,275 mean: 0,036 max: 0,65165 min: 0,0 avgzeros: 0,1778 maxzeros: 3,0 time: 0,01378	correct: 43,225 mean: 0,04588 max: 0,72999 min: 0,0 avgzeros: 0,5702 maxzeros: 3,0 time: 0,02017	correct: 66,65 mean: 0,04371 max: 0,72999 min: 0,0 avgzeros: 0,2694 maxzeros: 3,0 time: 0,00932	correct: 57,375 correct_hi2: 0,36125 correct_h: 0,70034 mean: 0,04321 max: 0,69545 min: 0,0 avgzeros: 0,3894 maxzeros: 3,0 time: 0,01057
9	correct: 74,4 mean: 0,04386 max: 0,66112 min: 0,0 avgzeros: 0,2066 maxzeros: 3,0 time: 0,01555	correct: 42,42222 mean: 0,05568 max: 0,72999 min: 0,0 avgzeros: 0,649 maxzeros: 3,0 time: 0,02275	correct: 65,62222 mean: 0,05344 max: 0,72999 min: 0,0 avgzeros: 0,3108 maxzeros: 3,0 time: 0,01051	correct: 56,55556 correct_hi2: 0,35578 correct_h: 0,6986 mean: 0,05237 max: 0,69545 min: 0,0 avgzeros: 0,4464 maxzeros: 3,0 time: 0,0119
10	correct: 73,64 mean: 0,05123 max: 0,66112 min: 0,00012 avgzeros: 0,2332 maxzeros: 3,0 time: 0,01738	correct: 41,94 mean: 0,06399 max: 0,72999 min: 0,00017 avgzeros: 0,727 maxzeros: 3,0 time: 0,02543	correct: 65,2 mean: 0,06183 max: 0,72999 min: 0,00014 avgzeros: 0,3462 maxzeros: 3,0 time: 0,01173	correct: 55,78 correct_hi2: 0,3504 correct_h: 0,69413 mean: 0,06061 max: 0,69545 min: 0,00012 avgzeros: 0,5036 maxzeros: 3,0 time: 0,01321

Таблиця 3.3 – Результати моделювання для 10 повторів імітації основного експерименту, що складається з 30 спостережень

σ^2	Симетричний критерій $ЗСК_1 + ЗСК_2$	Несиметричний критерій $ЗСК_1$	Несиметричний критерій $ЗСК_2$	Структура розв'язку задається $min(\chi_1^2, \chi_2^2, \chi_3^2)$
1	correct: 87,4 mean: 0,00135 max: 0,12569 min: 0,0 avgzeros: 0,0122 maxzeros: 2,0 time: 0,00198	correct: 65,2 mean: 0,00167 max: 0,33501 min: 0,0 avgzeros: 0,0402 maxzeros: 3,0 time: 0,00335	correct: 81,0 mean: 0,00168 max: 0,33501 min: 0,0 avgzeros: 0,0204 maxzeros: 3,0 time: 0,00141	correct: 74,0 correct_hi2: 0,412 correct_h: 0,80086 mean: 0,00152 max: 0,21569 min: 0,0 avgzeros: 0,0294 maxzeros: 2,0 time: 0,00218
2	correct: 86,7 mean: 0,00336 max: 0,20437 min: 0,0 avgzeros: 0,0244 maxzeros: 2,0 time: 0,00414	correct: 62,1 mean: 0,00398 max: 0,33501 min: 0,0 avgzeros: 0,083 maxzeros: 3,0 time: 0,00689	correct: 81,3 mean: 0,00384 max: 0,33501 min: 0,0 avgzeros: 0,0378 maxzeros: 3,0 time: 0,00293	correct: 72,8 correct_hi2: 0,416 correct_h: 0,787 mean: 0,00379 max: 0,25133 min: 0,0 avgzeros: 0,0598 maxzeros: 3,0 time: 0,00439
3	correct: 86,06667 mean: 0,0059 max: 0,32468 min: 0,0 avgzeros: 0,0358 maxzeros: 2,0 time: 0,00633	correct: 61,2 mean: 0,007 max: 0,33501 min: 0,0 avgzeros: 0,1258 maxzeros: 3,0 time: 0,01048	correct: 80,26667 mean: 0,00694 max: 0,33501 min: 0,0 avgzeros: 0,0568 maxzeros: 3,0 time: 0,00447	correct: 71,73333 correct_hi2: 0,398 correct_h: 0,78255 mean: 0,00683 max: 0,33041 min: 0,0 avgzeros: 0,0896 maxzeros: 3,0 time: 0,0066
4	correct: 84,4 mean: 0,00938 max: 0,37185 min: 0,0 avgzeros: 0,0506 maxzeros: 2,0 time: 0,00829	correct: 60,25 mean: 0,0112 max: 0,58515 min: 0,0 avgzeros: 0,1722 maxzeros: 3,0 time: 0,01376	correct: 78,9 mean: 0,01098 max: 0,58517 min: 0,0 avgzeros: 0,0766 maxzeros: 3,0 time: 0,00586	correct: 70,35 correct_hi2: 0,3895 correct_h: 0,77521 mean: 0,01094 max: 0,58515 min: 0,0 avgzeros: 0,1214 maxzeros: 3,0 time: 0,0087
5	correct: 83,76 mean: 0,01267 max: 0,40834 min: 0,0 avgzeros: 0,0648 maxzeros: 2,0 time: 0,01024	correct: 60,04 mean: 0,01491 max: 0,58515 min: 0,0 avgzeros: 0,2156 maxzeros: 3,0 time: 0,01702	correct: 78,76 mean: 0,01443 max: 0,58517 min: 0,0 avgzeros: 0,095 maxzeros: 3,0 time: 0,00724	correct: 70,76 correct_hi2: 0,4008 correct_h: 0,78483 mean: 0,01435 max: 0,58515 min: 0,0 avgzeros: 0,1466 maxzeros: 3,0 time: 0,01077

Продовження таблиці 3.3

σ^2	Симетричний критерій $ЗСК_1 + ЗСК_2$	Несиметричний критерій $ЗСК_1$	Несиметричний критерій $ЗСК_2$	Структура розв'язку задається $min(\chi_1^2, \chi_2^2, \chi_3^2)$
6	correct: 83,86667 mean: 0,01616 max: 0,43298 min: 0,0 avgzeros: 0,0752 maxzeros: 2,0 time: 0,01227	correct: 59,83333 mean: 0,01973 max: 0,58515 min: 0,0 avgzeros: 0,2624 maxzeros: 3,0 time: 0,02042	correct: 78,63333 mean: 0,01877 max: 0,58517 min: 0,0 avgzeros: 0,1136 maxzeros: 3,0 time: 0,00868	correct: 70,7 correct_hi2: 0,400667 correct_h: 0,78643 mean: 0,01881 max: 0,58515 min: 0,0 avgzeros: 0,1772 maxzeros: 3,0 time: 0,01285
7	correct: 83,65714 mean: 0,02027 max: 0,52842 min: 0,0 avgzeros: 0,0866 maxzeros: 2,0 time: 0,01448	correct: 59,57143 mean: 0,02494 max: 0,58515 min: 0,0 avgzeros: 0,3078 maxzeros: 3,0 time: 0,02416	correct: 78,4 mean: 0,02388 max: 0,58517 min: 0,0 avgzeros: 0,1334 maxzeros: 3,0 time: 0,01025	correct: 70,45714 correct_hi2: 0,3991 correct_h: 0,78535 mean: 0,02369 max: 0,58515 min: 0,0 avgzeros: 0,206 maxzeros: 3,0 time: 0,01494
8	correct: 82,725 mean: 0,0247 max: 0,52842 min: 0,0 avgzeros: 0,1032 maxzeros: 3,0 time: 0,01663	correct: 59,725 mean: 0,02996 max: 0,58515 min: 0,0 avgzeros: 0,347 maxzeros: 3,0 time: 0,02774	correct: 77,775 mean: 0,02915 max: 0,58517 min: 0,0 avgzeros: 0,1558 maxzeros: 3,0 time: 0,01176	correct: 69,925 correct_hi2: 0,39675 correct_h: 0,785 mean: 0,02892 max: 0,58515 min: 0,0 avgzeros: 0,2364 maxzeros: 3,0 time: 0,01703
9	correct: 82,31111 mean: 0,02961 max: 0,53409 min: 0,0 avgzeros: 0,117 maxzeros: 3,0 time: 0,01876	correct: 59,13333 mean: 0,03596 max: 0,58515 min: 0,0 avgzeros: 0,3964 maxzeros: 3,0 time: 0,03131	correct: 77,02222 mean: 0,03545 max: 0,62139 min: 0,0 avgzeros: 0,18 maxzeros: 3,0 time: 0,01328	correct: 69,48889 correct_hi2: 0,39378 correct_h: 0,78429 mean: 0,03494 max: 0,58515 min: 0,0 avgzeros: 0,268 maxzeros: 3,0 time: 0,01911
10	correct: 82,12 mean: 0,03486 max: 0,70514 min: 7e-05 avgzeros: 0,1284 maxzeros: 3,0 time: 0,02098	correct: 58,86 mean: 0,04275 max: 0,73316 min: 7e-05 avgzeros: 0,4398 maxzeros: 3,0 time: 0,03502	correct: 76,76 mean: 0,04123 max: 0,70514 min: 7e-05 avgzeros: 0,199 maxzeros: 3,0 time: 0,01484	correct: 69,38 correct_hi2: 0,3932 correct_h: 0,78484 mean: 0,04082 max: 0,71501 min: 7e-05 avgzeros: 0,2934 maxzeros: 3,0 time: 0,0212

Таблиця 3.4 – Результати моделювання для 10 повторів імітації основного експерименту, що складається з 40 спостережень

σ^2	Симетричний критерій $ЗСК_1 + ЗСК_2$	Несиметричний критерій $ЗСК_1$	Несиметричний критерій $ЗСК_2$	Структура розв'язку задається $min(\chi_1^2, \chi_2^2, \chi_3^2)$
1	correct: 92,2 mean: 0,001 max: 0,17242 min: 0,0 avgzeros: 0,0076 maxzeros: 2,0 time: 0,0025	correct: 74,6 mean: 0,00117 max: 0,17242 min: 0,0 avgzeros: 0,0278 maxzeros: 2,0 time: 0,00449	correct: 89,2 mean: 0,00116 max: 0,22238 min: 0,0 avgzeros: 0,0112 maxzeros: 2,0 time: 0,00181	correct: 83,0 correct_hi2: 0,488 correct_h: 0,87 mean: 0,0012 max: 0,22238 min: 0,0 avgzeros: 0,0182 maxzeros: 2,0 time: 0,00314
2	correct: 93,4 mean: 0,00244 max: 0,17772 min: 0,0 avgzeros: 0,0128 maxzeros: 3,0 time: 0,00482	correct: 75,9 mean: 0,00298 max: 0,31661 min: 0,0 avgzeros: 0,0528 maxzeros: 3,0 time: 0,00863	correct: 88,6 mean: 0,00299 max: 0,31679 min: 0,0 avgzeros: 0,0244 maxzeros: 3,0 time: 0,00348	correct: 84,4 correct_hi2: 0,471 correct_h: 0,87734 mean: 0,00278 max: 0,22238 min: 0,0 avgzeros: 0,033 maxzeros: 2,0 time: 0,00614
3	correct: 92,06667 mean: 0,00454 max: 0,24839 min: 0,0 avgzeros: 0,021 maxzeros: 3,0 time: 0,00711	correct: 75,13333 mean: 0,00528 max: 0,31661 min: 0,0 avgzeros: 0,0772 maxzeros: 3,0 time: 0,01281	correct: 87,66667 mean: 0,00529 max: 0,34136 min: 0,0 avgzeros: 0,0356 maxzeros: 3,0 time: 0,00516	correct: 83,33333 correct_hi2: 0,456 correct_h: 0,87658 mean: 0,00503 max: 0,34136 min: 0,0 avgzeros: 0,049 maxzeros: 2,0 time: 0,00912
4	correct: 91,45 mean: 0,00668 max: 0,24839 min: 0,0 avgzeros: 0,027 maxzeros: 3,0 time: 0,00932	correct: 74,95 mean: 0,00783 max: 0,31661 min: 0,0 avgzeros: 0,1022 maxzeros: 3,0 time: 0,01677	correct: 87,1 mean: 0,00771 max: 0,34136 min: 0,0 avgzeros: 0,0474 maxzeros: 3,0 time: 0,00678	correct: 82,65 correct_hi2: 0,4515 correct_h: 0,8723 mean: 0,00748 max: 0,34136 min: 0,0 avgzeros: 0,0676 maxzeros: 2,0 time: 0,01212
5	correct: 91,08 mean: 0,00933 max: 0,30096 min: 0,0 avgzeros: 0,0334 maxzeros: 3,0 time: 0,01181	correct: 74,52 mean: 0,01124 max: 0,47077 min: 0,0 avgzeros: 0,1274 maxzeros: 3,0 time: 0,02132	correct: 86,28 mean: 0,01112 max: 0,47077 min: 0,0 avgzeros: 0,0604 maxzeros: 3,0 time: 0,0086	correct: 81,96 correct_hi2: 0,4452 correct_h: 0,86859 mean: 0,01059 max: 0,47077 min: 0,0 avgzeros: 0,0858 maxzeros: 3,0 time: 0,01515

Продовження таблиці 3.4

σ^2	Симетричний критерій $ЗСК_1 + ЗСК_2$	Несиметричний критерій $ЗСК_1$	Несиметричний критерій $ЗСК_2$	Структура розв'язку задається $min(\chi_1^2, \chi_2^2, \chi_3^2)$
6	correct: 90,76667 mean: 0,01195 max: 0,30096 min: 0,0 avgzeros: 0,038 maxzeros: 3,0 time: 0,01455	correct: 73,93333 mean: 0,01445 max: 0,47077 min: 0,0 avgzeros: 0,152 maxzeros: 3,0 time: 0,02616	correct: 86,1 mean: 0,01412 max: 0,47077 min: 0,0 avgzeros: 0,0698 maxzeros: 3,0 time: 0,01053	correct: 81,56667 correct_hi2: 0,43835 correct_h: 0,86716 mean: 0,01343 max: 0,47077 min: 0,0 avgzeros: 0,1006 maxzeros: 3,0 time: 0,01833
7	correct: 90,14286 mean: 0,01495 max: 0,31496 min: 0,0 avgzeros: 0,0458 maxzeros: 3,0 time: 0,01719	correct: 73,22857 mean: 0,01837 max: 0,48979 min: 0,0 avgzeros: 0,1832 maxzeros: 3,0 time: 0,0308	correct: 85,51429 mean: 0,01775 max: 0,48979 min: 0,0 avgzeros: 0,0822 maxzeros: 3,0 time: 0,01241	correct: 81,05714 correct_hi2: 0,43743 correct_h: 0,8652 mean: 0,01681 max: 0,48356 min: 0,0 avgzeros: 0,12 maxzeros: 3,0 time: 0,02143
8	correct: 89,4 mean: 0,01831 max: 0,31496 min: 0,0 avgzeros: 0,0538 maxzeros: 3,0 time: 0,01966	correct: 73,2 mean: 0,02228 max: 0,48979 min: 0,0 avgzeros: 0,208 maxzeros: 3,0 time: 0,03529	correct: 84,825 mean: 0,02156 max: 0,48979 min: 0,0 avgzeros: 0,0966 maxzeros: 3,0 time: 0,0142	correct: 80,225 correct_hi2: 0,43225 correct_h: 0,85986 mean: 0,02057 max: 0,48356 min: 0,0 avgzeros: 0,1402 maxzeros: 3,0 time: 0,02438
9	correct: 88,6 mean: 0,02212 max: 0,40559 min: 0,0 avgzeros: 0,0634 maxzeros: 3,0 time: 0,02192	correct: 72,48889 mean: 0,02684 max: 0,48979 min: 0,0 avgzeros: 0,236 maxzeros: 3,0 time: 0,03938	correct: 83,71111 mean: 0,02623 max: 0,51854 min: 0,0 avgzeros: 0,1144 maxzeros: 3,0 time: 0,01587	correct: 79,33333 correct_hi2: 0,42646 correct_h: 0,85673 mean: 0,02509 max: 0,48356 min: 0,0 avgzeros: 0,1608 maxzeros: 3,0 time: 0,02735
10	correct: 88,04 mean: 0,02587 max: 0,4489 min: 4e-05 avgzeros: 0,071 maxzeros: 3,0 time: 0,02455	correct: 72,14 mean: 0,03143 max: 0,48979 min: 4e-05 avgzeros: 0,2622 maxzeros: 3,0 time: 0,04426	correct: 82,96 mean: 0,03088 max: 0,56638 min: 4e-05 avgzeros: 0,1294 maxzeros: 3,0 time: 0,01779	correct: 79,02 correct_hi2: 0,4248 correct_h: 0,85742 mean: 0,02934 max: 0,48505 min: 4e-05 avgzeros: 0,178 maxzeros: 3,0 time: 0,03037

Висновки до розділу

У рамках даного розділу був проведений ґрунтовний аналіз засобів розробки програмного забезпечення автоматизації процесу створення та програмної реалізації індивідуальних алгоритмів побудови БПР, заданих надлишковим описом. Програмне забезпечення буде представлене у вигляді окремої програмної бібліотеки для мови програмування Python.

Під час аналізу вимог до програмної бібліотеки було детально описано повний набір допоміжних функцій, які забезпечать коректну роботу програмного забезпечення автоматизації процесу створення та програмної реалізації індивідуальних алгоритмів побудови БПР. Окремо виділено дві функції, які доступні користувачу бібліотеки, такі як run_regression_script та run_mgmodh. Вони дозволяють виконувати індивідуального алгоритму синтетичного методу і використовувати ММГУА окремо від синтетичного методу відповідно.

Розроблено модифікацію загальної архітектури програмної бібліотеки, а також модифікації її компонентів, що відповідають за декомпозиційний метод та ММГУА.

Виконано тестування програмної бібліотеки та проведено оцінку її якості за допомогою бібліотеки Radon з використанням обґрунтованого набору метрик. В результаті було встановлено працездатність програмної бібліотеки у відповідності до теоретичних викладок. Значення метрик показали, що реалізація синтетичного методу з використанням програмного розширення автоматизації створення та програмної реалізації індивідуальних алгоритмів побудови БПР на порядок простіше, ніж реалізація синтетичного методу без використання даного розширення.

Було проведено статистичне дослідження нової версії ММГУА, яке довело її ефективність при розв’язанні задач побудови БПР, заданих надлишковим описом.

ВИСНОВКИ

В дисертаційній роботі розв’язане актуальне наукове завдання – підвищення ефективності програмного забезпечення синтетичного методу побудови БПР, заданих надлишковим описом, шляхом розробки нових компонентів програмного забезпечення, що реалізують теоретично обґрунтовану методологію створення індивідуальних алгоритмів декомпозиційного методу, та підвищення ефективності ММГУА, завдяки одночасному використанню низки регулярних критеріїв.

При цьому отримано наступні наукові та практичні результати:

– розроблена нова теоретично обґрунтована методологія створення індивідуальних алгоритмів побудови БПР, заданих надлишковим описом;

– створено підхід, який дозволяє підвищити ефективність ММГУА, що входить до складу синтетичного методу, шляхом одночасного використання низки регулярних критеріїв;

– знайдена асимптотична складність загальної формули МНК, що необхідно при використанні алгоритму статичного управління паралельними асинхронними обчислювальними процесами для покращення швидкодії паралельної реалізації ММГУА;

– на основі проведеного аналізу способів розширення можливостей мов програмування загального призначення та універсальних програмних засобів було обґрунтовано і розроблено методологію розширення можливостей обраної мови програмування загального призначення Python для автоматизації процесу створення та програмної реалізації індивідуальних алгоритмів побудови БПР;

– модифіковано архітектуру програмної бібліотеки реалізації синтетичного методу побудови БПР, заданих надлишковим описом, шляхом включення в себе нових складових, що забезпечують підвищення ефективності синтетичного методу, за рахунок програмної реалізації автоматизації процесу створення індивідуальних алгоритмів декомпозиційного методу, ефективність яких ґрунтується на отриманих теоретичних результатах; аналізу результатів статистичних досліджень нової версії

ММГУА, а також, як наслідок, підвищення ефективності в цілому програмної бібліотеки побудови БПР, заданих надлишковим описом.

Таким чином, усі поставлені завдання у даному дисертаційному дослідженні повністю виконані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Ortiz-Herrero L., Maguregui M. I., Bartolomé L. Multivariate (O)PLS regression methods in forensic dating // TrAC Trends in Analytical Chemistry, Vol. 141, 2021, P. 116278. DOI: 10.1016/ j.trac.2021.116278

2) Telenyk S. F, Nowakowski G., Pavlov O. A., Misura E. B., Melnikov O. V., Khalus E. A. Highly efficient scheduling algorithms for identical parallel machines with sufficient conditions for optimality of the solutions obtained // Bulletin of the Polish Academy of Sciences: Technical Sciences, Vol. 72 (1), 2024, P. e148939. DOI: 10.24425/bpasts.2024.148939

3) Згуровський М. З., Павлов О. А. Труднорозв'язувані задачі комбінаторної оптимізації у плануванні і прийнятті рішень. Київ: Наукова думка, 2016, 432 с.

4) Головченко М. М. Методи та програмні засоби багатовимірної поліноміальної регресії за надлишковим описом на основі побудови одновимірної регресії з використанням ортогональних поліномів Форсайта // Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 121 – Інженерія програмного забезпечення з галузі знань 12 – Інформаційні технології. – Національний Технічний Університет України «Київський Політехнічний Інститут імені Ігоря Сікорського», Київ, 2023, 215 с.

5) Zgurovsky M. Z., Pavlov O. A. Combinatorial Optimization Problems in Planning and Decision Making: Theory and Applications. 1st ed. Cham: Springer, 2019. 354 p. DOI: https://doi.org/10.1007/978-3-319-98977-8

6) Sinha P. Multivariate polynomial regression in data mining: Methodology, problems and solutions // International Journal of Scientific & Engineering Research. 2013. Vol. 4, № 12. pp. 962-965. URL: https://www.ijser.org/onlineResearchPaperViewer.aspx?Multivariate-Polynomial-Regression-in-Data-MiningMethodology.pdf (дата звернення: 22.10.2023)

7) Babatunde G., et al. Impact of climatic change on agricultural product yield using k-means and multiple linear regressions // International Journal of Education and

Management Engineering. 2019. Vol. 9, № 3. pp. 16-26. DOI: https://doi.org/10.5815/ijeme.2019.03.02

8) Shahrel M. Z., Mutalib S., Abdul-Rahman S. PriceCop Price monitor and prediction using linear regression and LSVM-ABC methods for e-commerce platform // International Journal of Information Engineering and Electronic Business. 2021. Vol. 13, № 1. pp. 1-14. DOI: https://doi.org/10.5815/ijieeb.2021.01.01

9) Pavlov O. A., Holovchenko M. M., Drozd V. V. Construction of a multivariate polynomial given by a redundant description in stochastic and deterministic formulations using an active experiment // *Bulletin of National Technical University «KhPI». Series: System Analysis, Control and Information Technologies.* 2022. № 1 (7). pp. 3-8. DOI: https://doi.org/10.20998/2079-0023.2022.01.01

10) Hudson D. J. Maximum Likelihood and Least Squares Theory Statistics Lectures // Geneva, CERN. 1964. Vol. 2. DOI: https://doi.org/10.5170/CERN-1964-018

11) Braak C. J. F. ter, Looman C. W. N. Regression // Data analysis in community and landscape ecology / ed. by Jongman R. H. G., ter Braak C. J. F., van Tongeren O. F. R. Cambridge: Cambridge University Press, 1995. pp. 29-77. URL: https://library.wur.nl/WebQuery/wurpubs/436954 (дата звернення: 30.12.2023)

12) Pavlov O., Holovchenko M., Mukha I., et al. Mathematics and software for building nonlinear polynomial regressions using estimates for univariate polynomial regressions coefficients with a given (small) variance // *Advances in Computer Science for Engineering and Education V. ICCSEEA 2022. Lecture Notes on Data Engineering and Communications Technologies.* 2022. Vol. 134. pp. 288-303. DOI: https://doi.org/10.1007/978-3-031-04812-8_25

13) Павлов О. А., Головченко М. М., Ревич М. М. Метод оцінки коефіцієнтів при лінійних членах багатовимірної поліноміальної регресії, заданої надлишковим описом // *Адаптивні системи автоматичного управління: міжвідомчий наук.-техн. збірник. Київ : НТУУ «КПІ»*, 2022. Вип. 1 (40). С. 110-117. DOI: https://doi.org/10.20535/1560-8956.40.2022.261665

14) Pavlov O. A., Holovchenko M. M., Drozd V. V. Efficiency substantiation for a synthetical method of constructing a multivariate polynomial regression given by a redundant representation // *Bulletin of National Technical University «KhPI». Series: System analysis, control and information technologies.* 2023. № 1 (9). pp. 3-9. DOI: https://doi.org/10.20998/2079-0023.2023.01.0

15) Павлов О. А., Халус О. А., Мельников О. В., Дрозд В. В., Кобельський В. В., Медведєв М. Є. Статичні алгоритми управління паралельними асинхронними обчислювальними процесами // *Міжвідомчий науково-технічний журнал «Адаптивні системи автоматичного управління».* К.: КПІ ім. Ігоря Сікорського, 2024. Том 1. № 44. С.116-126. DOI: https://doi.org/10.20535/1560-8956.44.2024.302427

16) Івахненко О. Метод групового урахування аргументів // Математичне моделювання. Київ: Вид. дім «Наукова думка», 2020, 450 с.

17) Троцюк П. С., Дрозд В. В., Головченко М. М., Павлов О. А. Програмне забезпечення для статистичного дослідження ефективності методу побудови багатовимірної лінійної регресії, заданої надлишковим описом // *П'ята Міжнародна науково-практична конференція «Інженерія програмного забезпечення і передові інформаційні технології (SoftTech – 2023)»*, Київ, 19-21 грудня 2023 р.: Кафедра ІПІ ФІОТ, КПІ ім. Ігоря Сікорського, 2024, 399 с. – С. 327-332.

18) Zgurovsky M. Z., Pavlov O. A. Прийняття рішень в мережевих системах з обмеженими ресурсами. Київ: Вид. дім «Наукова думка», 2010, 300 с.

19) Microsoft Support Show the Developer Tab // Офіційний сайт Microsoft Support. URL: https://support.microsoft.com/en-gb/office/show-the-developer-tab-e1192344-5e56-4d45-931b (дата звернення: 25.11.2024)

20) Огляд програмного забезпечення загального призначення, що дозволяє розв'язувати задачі регресійного аналізу // FinAcademy. URL: https://finacademy.net/ua/materials/prohramyanalohy-excel-vid-google-do-apple (дата звернення: 15.10.2024)

21) Офіційна документація Statistica Documentation // Statistica. URL: https://docs.tibco.com/pub/stat/14.2.0/doc/html/UserGuide/Default.html (дата звернення: 16.11.2024)

22) Statistica Basic documentation // TIBCO Support. URL: https://support.tibco.com/s/article/Statistica-Academic-Bundle-Feature-Details (дата звернення: 15.11.2024)

23) Statistica Basic bundle // StatWorks. URL: https://www.statwks.com/product/tibco-statistica-basic-academic/ (дата звернення: 11.11.2024)

24) Marques de Sa E. Applied Statistics Using SPSS, STATISTICA and MATLAB // MathWorks. URL: https://www.mathworks.com/academia/books/applied-statistics-using-spss-statistica-and-matlab-marques-de-sa.html (дата звернення: 13.11.2024)

25) Офіційна документація Python // Python Software Foundation. URL: https://docs.python.org/ (дата звернення: 21.11.2024)

26) Бібліотека NumPy // NumPy. URL: https://numpy.org/devdocs/ (дата звернення: 22.11.2024)

27) Бібліотека Pandas // Pandas Documentation. URL: https://pandas.pydata.org/docs/ (дата звернення: 23.11.2024)

28) Бібліотека SciPy // SciPy Documentation. URL: https://scipy.org/ (дата звернення: 23.11.2024)

29) Бібліотека Matplotlib // Matplotlib. URL: https://matplotlib.org/stable/index.html (дата звернення: 23.11.2024)

30) Бібліотека Seaborn // Seaborn Documentation. URL: https://seaborn.pydata.org/ (дата звернення: 24.11.2024)

31) Бібліотека Sklearn // Scikit-learn Documentation. URL: https://scikit-learn.org/stable/ (дата звернення: 24.11.2024)

32) Бібліотека Statsmodels // Statsmodels Documentation. URL: https://www.statsmodels.org/stable/index.html (дата звернення: 25.11.2024)

33) Pavlov O. A., Holovchenko M. M., Drozd V. V. Modification of the decomposition method of constructing multivariate polynomial regression which is linear with respect to unknown coefficients // *Bulletin of National Technical University «KhPI». Series: System analysis, control and information technologies*, – № 2 (12), 2024. – С. 3–10. DOI: https://doi.org/10.20998/2079-0023.2024.02.01

34) Sebesta R. W. Concepts of Programming Languages // 11th ed. Boston: Addison Wesley, 2015. 800 p.

35) Павлов О. А., Головченко М. М., Дрозд В. В., Шаргородський В. С. Підвищення ефективності модифікованого методу урахування аргументів для побудови багатовимірних регресій, заданих надлишковим описом // *Міжвідомчий науково-технічний журнал «Адаптивні системи автоматичного управління».* К.: КПІ ім. Ігоря Сікорського, 2025. Том 1. № 46

36) Павлов О. А., Головченко М. М., Дрозд В. В., Ревич М. М. Дослідження ефективності методу побудови багатовимірної лінійної регресії, заданої надлишковим описом // *SoftTech – 2022 Осінь. Матеріали III Всеукраїнської наук.-практ. конф. молодих вчених та студентів (Київ, 22-25 жовтня 2022 р.).* Київ: КПІ ім. Ігоря Сікорського, 2023. С. 10-13. (дата звернення: 15.12.2024)

37) Len B., Clements P., Kazman R. Software Architecture in Practice // Addison-Wesley Professional, Boston, 2012

38) Richards M. Software Architecture Patterns // O'Reilly Media, Inc, Sebastopol, 2015

39) Tiwari U., Kumar S. Component-Based Software Engineering // Taylor & Francis, Boca Raton, 2020. DOI: https://doi.org/10.1201/9780429331749

40) PyCharm The Python IDE for data and web professionals // Офіційний сайт JetBrains. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 13.02.2025)

41) PyPI The package installer for Python // Офіційний сайт PyPI. URL: https://pypi.org/project/pip/ (дата звернення: 13.02.2025)

42) Giroux B. A Python package for traveltime computation and raytracing // SoftwareX 16, 2021. DOI: https://doi.org/10.1016/j.softx.2021.100834

43) Опис бібліотеки та програмна документація Radon // Офіційний сайт Radon. URL: https://radon.readthedocs.io/en/latest/ (дата звернення: 25.02.2025)

ДОДАТКИ

ДОДАТОК А

Лістинг коду

Повний код програмного забезпечення можна знайти за наступним посиланням: https://github.com/ValeryDrozd/Synthetic_reg

Файл index.py

```
from model import ModelLoader

if __name__ == '__main__':
    m = ModelLoader()
    data = m.run_regression_script('example.poly')
```

Файл operators.py

```
import re
import numpy as np
import sympy as sp

    def dec_method_ao1(self, input_matrix, redundant_description_matrix, col_id):
        row_idx = next((i for i, row in enumerate(redundant_description_matrix) if
                        row[col_id] > 1 and sum(input_matrix[:, i]) != 0), None)
        if row_idx is None:
            return 'The usage of the method is not allowed'

        self.vars_dict['coef_buf'] = int(
            redundant_description_matrix[row_idx][col_id] if input_matrix[0][row_idx] != input_matrix[1][
                row_idx] else 0)
        usage_flag = any(
            row > 1 and j == row_idx and i != col_id for i, column in enumerate(redundant_description_matrix.T) for
            j, row in enumerate(column) if row > 0 and sum(input_matrix[:, j]) == 0)

        if usage_flag:
            return 'The usage of the method is not allowed'
        return self.calculate_div_for_dec(input_matrix, redundant_description_matrix, col_id) ** 2 * \
```

```
        self.vars_dict['q_matrix'][2][redundant_description_matrix[row_idx][col_id]]

def dec_method_ao2(self, input_matrix, redundant_description_matrix, col_id):
    coefficients = [row[col_id] for i, row in enumerate(redundant_description_matrix) if
                    row[col_id] > 1 and sum(input_matrix[:, i]) != 0]
    row_sum = sum(coefficients)
    col_indices = [i for i, row in enumerate(redundant_description_matrix) if row[col_id] > 1]

    for j in range(len(redundant_description_matrix[0])):
        if j != col_id:
            check_sum = sum(redundant_description_matrix[i][j] for i in col_indices)
            if sum(0 if input_matrix[0, i] ** redundant_description_matrix[i][j] != 0 else 1 for i in
                   range(self.vars_dict['r'])) != 0:
                continue
            if check_sum >= row_sum:
                return 'The usage of the method is not allowed'

    dispersion = self.calculate_div_for_dec(input_matrix, redundant_description_matrix, col_id)
    self.vars_dict['coef_buf'] = [coefficients[i] for i in range(len(col_indices)) if
                                  input_matrix[0][col_indices[i]] != input_matrix[1][col_indices[i]]]
    if not self.vars_dict['coef_buf']:
        return 0
    return dispersion ** 2 * np.prod([self.vars_dict['q_matrix'][2][i] for i in self.vars_dict['coef_buf']])

def calculate_div_for_dec(self, input_matrix, redundant_description_matrix, col_id):
    dispersion = sp.Float(1, precision=self.vars_dict['acc'])
    for i in range(len(redundant_description_matrix)):
        if redundant_description_matrix[i][col_id] >= 1:
            if input_matrix[0][i] == input_matrix[1][i]:
                dispersion /= sp.Float(input_matrix[0][i], precision=self.vars_dict['acc']) ** sp.Float(
                    redundant_description_matrix[i][col_id], precision=self.vars_dict['acc'])
            else:
                dispersion /= sp.Float(self.vars_dict['ab_matrix'][i][0], self.vars_dict['acc']) ** sp.Float(
                    redundant_description_matrix[i][col_id], precision=self.vars_dict['acc'])
    return dispersion / self.vars_dict['l']
```

```
        def dec_method_ao3(self, input_matrix, redundant_description_matrix, col_id):
            if any(input_matrix[0][col_idx] == input_matrix[1][col_idx] and input_matrix[0][col_idx] != 0 for
col_idx in
                   range(len(input_matrix[0]))):
                return 'The usage of the method is not allowed'

            if all(row[col_id] <= 1 if sum(input_matrix[:, i]) != 0 else 0 for i, row in
                   enumerate(redundant_description_matrix)):
                row_idx = [i for i, row in enumerate(redundant_description_matrix) if row[col_id] == 1]
                for i, row_id in enumerate(row_idx):
                    if any(redundant_description_matrix[row_id] > 1 if sum(input_matrix[:, i]) != 0 else 0):
                        return 'The usage of the method is not allowed'
                self.vars_dict['coef_buf'] = len(
                    [i for i in range(len(row_idx)) if input_matrix[0][row_idx[i]] != input_matrix[1][row_idx[i]]])
                return self.calculate_div_for_dec(input_matrix, redundant_description_matrix, col_id) ** 2 * \
                    self.vars_dict['q_matrix'][2][self.vars_dict['coef_buf']]
            return 'The usage of the method is not allowed'

        def dec_method_ao4(self, input_matrix, redundant_description_matrix, col_id):
            if all(row[col_id] <= 1 if sum(input_matrix[:, i]) != 0 else 0 for i, row in
                   enumerate(redundant_description_matrix)):
                row_idx = [i for i, row in enumerate(redundant_description_matrix) if row[col_id] == 1]
                for i, row_id in enumerate(row_idx):
                    if any(redundant_description_matrix[row_id] > 1 if sum(input_matrix[:, i]) != 0 else 0):
                        return 'The usage of the method is not allowed'
                self.vars_dict['coef_buf'] = len(
                    [i for i in range(len(row_idx)) if input_matrix[0][row_idx[i]] != input_matrix[1][row_idx[i]]])
                return self.calculate_div_for_dec(input_matrix, redundant_description_matrix, col_id) ** 2 * \
                    self.vars_dict['q_matrix'][2][self.vars_dict['coef_buf']]
            return 'The usage of the method is not allowed'

        def dec_method(self, input_matrix, y_act, redundant_description_matrix, col_id):
            xv = sp.symbols('xv')
            z_1 = self.vars_dict['z_1']
            z_n = self.vars_dict['z_n']
            n = self.vars_dict['n']
```

```
            diff = (z_n - z_1) / (n - 1)
            w = [sum(y_act[k] * self.vars_dict['q_matrix'][1][p].subs({xv: z_1 + diff * k}) for k in
range(len(y_act))) for
                 p in range(self.vars_dict['r'] + 1)]

            theta = [sum(w[k] * self.vars_dict['q_matrix'][0][i, k] for k in range(self.vars_dict['r'] + 1)) for i in
                     range(self.vars_dict['r'] + 1)]
            idx = self.vars_dict['coef_buf'] if isinstance(self.vars_dict['coef_buf'], int) else sum(
                self.vars_dict['coef_buf'])

            val = theta[idx] * self.calculate_div_for_dec(input_matrix, redundant_description_matrix, col_id)
            return val
```

Файл io.py

```
import re
import numpy as np
import sympy as sp

    def parse_line(self, line):
        line = line.strip()
        if not line:
            return

        assignments = re.split(r'\s+(?=\w+=)', line)

        for assignment in assignments:
            self.parse_assignment(assignment)

    def parse_assignment(self, assignment):
        if '=' in assignment:
            key, value = assignment.split('=', 1)
            key = key.strip()
            value = value.strip()

            if key in self.vars_dict and key != 'l' and key != 'y_act' and key != 'input_matrix':
                raise ValueError(f"Error: Variable {key} is being re-initialized.")
```

```
            if key == 'l':
                self.l = self.validate_l(int(value))
                self.vars_dict[key] = self.l
            else:
                if value.startswith('ab_calc'):
                    self.vars_dict[key] = self.ab_calc(value)
                elif value.startswith('input_matrix_init'):
                    self.vars_dict[key] = self.input_matrix_val(value)
                elif value.startswith('y_act_reader'):
                    self.vars_dict[key] = self.y_act_reader(value)
                else:
                    self.vars_dict[key] = self.validate_value(key, value)
        elif assignment.startswith('print'):
            self.handle_print_command(assignment)
        elif assignment.startswith('dec_method'):
            self.execute_dec_method(assignment)
        else:
            exec(assignment)

    def split_params(self, s):
        params = []
        buffer = []
        depth = 0

        for char in s:
            if char == '(':
                depth += 1
            elif char == ')':
                depth -= 1
            if char == ',' and depth == 0:
                params.append(''.join(buffer).strip())
                buffer = []
            else:
                buffer.append(char)
```

```
    if buffer:
        params.append(''.join(buffer).strip())

    return params

def y_act_reader(self, y_act_str):
    match = re.match(r'y_act_reader\((.*)\)', y_act_str)
    if not match:
        raise ValueError("Invalid input format")

    args = match.group(1).split(',')
    file_path = args[0].strip().strip('"')
    input_matrix = self.vars_dict.get(args[1].strip())

    y_act_values = []

    offset = int(self.org_available_theta_len - len(self.available_theta)) * len(input_matrix[:, 0])
    with open(file_path, 'r') as f:
        for line in f:

            if offset > 0:
                for _ in range(offset - 1):
                    next(f)
                offset = 0
                continue
            left_part, right_part = line.strip().split(' - ')
            row_values = np.array([float(x) for x in left_part.split(',')])

            if any(np.allclose(row_values, input_row, atol=1e-2) for input_row in input_matrix):
                y_act_values.append(float(right_part))

            if len(y_act_values) == len(input_matrix):
                break

    if len(y_act_values) != len(self.vars_dict.get('input_matrix')):
        raise ValueError("Error: y_act values not found for all input_matrix rows")
```

```
    return y_act_values
```

Файл validations.py

```
  def validate_value(self, key, value):
    if key == 'r':
      return self.validate_r(int(value))
    elif key == 'n':
      return self.validate_n(int(value))
    elif key == 'm':
      return self.validate_m(int(value))
    elif key == 'input_vars_ranges_matrix':
      return self.validate_input_vars_ranges_matrix(eval(value))
    elif key == 'redundant_description_matrix':
      return self.validate_redundant_description_matrix(eval(value))
    elif key == 'z_1':
      return self.validate_z_1(eval(value))
    elif key == 'z_n':
      return self.validate_z_n(eval(value))
    elif key == 'q_matrix':
      return self.q_matrix_val(value)
    elif key == 'y_act':
      return self.validate_y_act(eval(value))
    elif key == 'acc':
      return self.validate_acc(eval(value))
    else:
      return eval(value)

  def validate_acc(self, acc):
    if not isinstance(acc, (int, float)):
      raise ValueError("Error code 1: acc is not a number")
    if acc < 0 or acc > 100:
      raise ValueError("Error code 4: acc is out of range")
    return acc

  def validate_r(self, r):
```

```
    if r is None:
        raise ValueError("Error code 0: r is missing")
    if not isinstance(r, int):
        raise ValueError("Error code 1: r is not an integer")
    if r <= 0:
        raise ValueError("Error code 3: r is not a positive number")
    if not 2 <= r <= 10:
        raise ValueError("Error code 4: r is out of range [2, 10]")
    return r

def validate_n(self, n):
    if n is None:
        raise ValueError("Error code 0: n is missing")
    if not isinstance(n, int):
        raise ValueError("Error code 1: n is not an integer")
    if n <= 0:
        raise ValueError("Error code 3: n is not a positive number")
    if not 3 <= n <= 10000:
        raise ValueError("Error code 4: n is out of range [3, 10000]")
    if self.vars_dict.get('r') is None:
        raise ValueError("Error: r is not initialized. Cannot validate n.")
    if n <= self.vars_dict.get('r'):
        raise ValueError("Error code 5: n must be greater than r")
    return n

def validate_m(self, m):
    if m is None:
        raise ValueError("Error code 0: m is missing")
    if not isinstance(m, int):
        raise ValueError("Error code 1: m is not an integer")
    if m <= 0:
        raise ValueError("Error code 3: m is not a positive number")
    if not 2 <= m <= 100:
        raise ValueError("Error code 4: m is out of range [2, 100]")
    return m
```

```
    def validate_l(self, l):
        if l is None:
            raise ValueError("Error code 0: l is missing")
        if not isinstance(l, int):
            raise ValueError("Error code 1: l is not an integer")
        if l <= 0:
            raise ValueError("Error code 3: l is not a positive number")
        if not 1 <= l <= 100:
            raise ValueError("Error code 4: l is out of range [2, 10]")
        return l

    def validate_input_vars_ranges_matrix(self, input_vars_ranges_matrix):
        if input_vars_ranges_matrix is None:
            raise ValueError("Error code 0: input_vars_ranges_matrix is missing")
        if not isinstance(input_vars_ranges_matrix, list):
            raise ValueError("Error code 7: input_vars_ranges_matrix is not a list")
        if self.vars_dict.get('m') is None:
            raise ValueError(
                "Error: m is not initialized. Cannot determine the number of rows for input_vars_ranges_matrix.")
        if len(input_vars_ranges_matrix) != self.vars_dict.get('m'):
            raise ValueError("Error code 6: input_vars_ranges_matrix row count does not match m")
        for row in input_vars_ranges_matrix:
            if not isinstance(row, list) or len(row) != 2:
                raise ValueError("Error code 6: input_vars_ranges_matrix row does not have 2 elements")
            for value in row:
                if not isinstance(value, (int, float)):
                    raise ValueError("Error code 7: input_vars_ranges_matrix contains non-numeric values")
                if value < -2 ** 63 or value > 2 ** 63 - 1:
                    raise ValueError("Error code 8: input_vars_ranges_matrix values out of range")
        return input_vars_ranges_matrix

    def validate_redundant_description_matrix(self, redundant_description_matrix):
        if redundant_description_matrix is None:
            raise ValueError("Error code 0: redundant_description_matrix is missing")
        if not isinstance(redundant_description_matrix, list):
```

```
            raise ValueError("Error code 7: redundant_description_matrix is not a list")
        if self.vars_dict.get('m') is None:
            raise ValueError(
                "Error: m is not initialized. Cannot determine the number of rows for input_vars_ranges_matrix.")
        if len(redundant_description_matrix) != self.vars_dict.get('m'):
            raise ValueError("Error code 6: redundant_description_matrix row count does not match m")
        for row in redundant_description_matrix:
            if not isinstance(row, list):
                raise ValueError("Error code 7: redundant_description_matrix contains non-list rows")
            for value in row:
                if not isinstance(value, int):
                    raise ValueError("Error code 7: redundant_description_matrix contains non-integer values")
                if value < 0 or value > 10:
                    raise ValueError("Error code 8: redundant_description_matrix values out of range")
        if len(redundant_description_matrix) != len(set(map(tuple, redundant_description_matrix))):
            raise ValueError("Error code 14: redundant_description_matrix contains duplicate columns")
        self.available_theta = [i for i in range(len(redundant_description_matrix[0]))]
        self.result_dict = {'b' + str(theta): {} for theta in self.available_theta}
        self.org_available_theta_len = len(self.available_theta)
        return redundant_description_matrix

    def validate_z_1(self, z_1):
        if not isinstance(z_1, (int, float)):
            raise ValueError("Error code 1: z_1 is not a number")
        if z_1 < -2 ** 63 or z_1 > 2 ** 63 - 1:
            raise ValueError("Error code 4: z_1 is out of range")
        return z_1

    def validate_z_n(self, z_n):
        if not isinstance(z_n, (int, float)):
            raise ValueError("Error code 1: z_n is not a number")
        if z_n < -2 ** 63 or z_n > 2 ** 63 - 1:
            raise ValueError("Error code 4: z_n is out of range")
        return z_n
```

Файл dec_method.py

```
import re
import numpy as np
import sympy as sp


    def sample_x(self, start, end, n):
        return [start + (end - start) * i / (n - 1) for i in range(n)]

    def varTheta(self, Q, l, p):
        return sum([Q[p][k] ** 2 for k in range(len(Q))]) / l

    def q_matrix_calc(self, q_matrix):
        x = self.sample_x(self.vars_dict.get('z_1'), self.vars_dict.get('z_n'), self.vars_dict.get('n'))

        xv = sp.symbols('xv')
        n_sp = sp.Float(self.vars_dict.get('n'), self.vars_dict.get('acc'))
        Q = [(1 / sp.sqrt(n_sp)).evalf(self.vars_dict.get('acc'))]

        if self.vars_dict.get('r') >= 2:
            xs = sum(x) / n_sp
            Q1_sum1 = sum([(xi_val - xs) ** 2 for xi_val in x])
            Q.append((xs / sp.sqrt(Q1_sum1)).evalf(self.vars_dict.get('acc')) + (xv / sp.sqrt(Q1_sum1)).evalf(
                self.vars_dict.get('acc')))

        a = []
        b = []
        l = []

        for k in range(0, self.vars_dict.get('r') - 2):
            a_val = sum([xi_val * (Q[k + 1].subs({xv: xi_val})) ** 2 for xi_val in x])
            a.append(a_val.evalf(self.vars_dict.get('acc')))
            if a[-1] <= 1e-96:
                a[-1] = 0
            b_val = sum([xi_val * Q[k + 1].subs({xv: xi_val}) * Q[k].subs({xv: xi_val}) for xi_val in x])
            b.append(b_val.evalf(self.vars_dict.get('acc')))
```

```
                l_val = sp.sqrt(
                    sum([((xi_val - a[k]) * self.Q[k + 1].subs({xv: xi_val}) - b[k] * Q[k].subs({xv: xi_val})) ** 2
for
                         xi_val in x]))
                l.append(l_val.evalf(self.vars_dict.get('acc')))
                Q_next = ((xv * Q[k + 1] - a[k] * Q[k + 1] - b[k] * Q[k]) / l[k]).cancel()
                Q.append(Q_next.evalf(self.vars_dict.get('acc')))

            Q_matrix = [[sp.Float(0, self.vars_dict.get('acc')) for _ in range(len(Q))] for _ in range(len(self.Q))]

            for col, poly in enumerate(Q):
                poly_coeffs = sp.Poly(poly, xv).all_coeffs()
                poly_coeffs = poly_coeffs[::-1]

                for row, coeff in enumerate(poly_coeffs):
                    Q_matrix[row][col] = coeff.evalf(self.vars_dict.get('acc'))

            var_tr = [self.varTheta(Q_matrix, self.vars_dict.get('l'), p) for p in range(len(Q))]

            return Q_matrix, Q, var_tr

        def validate_y_act(self, y_act):
            if not isinstance(y_act, list):
                raise ValueError("Error code 7: y_act is not a list")
            if len(y_act) != self.vars_dict.get('n'):
                raise ValueError("Error code 11: y_act length does not match n")
            for value in y_act:
                if not isinstance(value, (int, float)):
                    raise ValueError("Error code 12: y_act contains non-numeric values")
                if value < -2 ** 63 or value > 2 ** 63 - 1:
                    raise ValueError("Error code 13: y_act values out of range")
            return y_act

        def ab_calc(self, ab_in_str):
            match = re.match(r'ab_calc\((.*)\)', ab_in_str)
            params = match.group(1).split(',')
```

```
            input_vars_ranges_matrix = self.vars_dict.get(params[0].strip())
            z_1 = self.vars_dict.get(params[1].strip())
            z_n = self.vars_dict.get(params[2].strip())
            if len(params) > 3:
                raise ValueError("Error: Too many parameters for ab_calc.")

            ab_matrix = [[(input_vars_ranges_matrix[i][1] - input_vars_ranges_matrix[i][0]) / (z_n - z_1),
                          input_vars_ranges_matrix[i][0] - (
                                  input_vars_ranges_matrix[i][1] - input_vars_ranges_matrix[i][0]) * z_1 / (z_n - z_1)]
                         for i in range(self.vars_dict.get('m'))]

            return ab_matrix

        def input_matrix_val(self, inp):
            if self.vars_dict.get('n') is None:
                raise ValueError("Error: n is not initialized. Cannot determine the number of rows for input_matrix.")

            if self.vars_dict.get('ab_matrix') is None:
                raise ValueError("Error: ab_matrix is not initialized. Cannot calculate xz_ab values.")

            match = re.match(r'input_matrix_init\(\[(.*)\]\)', inp)
            if not match:
                raise ValueError(f"Invalid input_matrix format: {inp}")

            param_str = match.group(1)
            params = self.split_params(param_str)

            input_matrix = [[None for _ in range(len(params))] for _ in range(self.vars_dict.get('n'))]

            for col_idx, param in enumerate(params):
                param = param.strip()
                if param.startswith('xz_ab'):
                    ab_match = re.match(r'xz_ab\((.*)\)', param)
                    if not ab_match:
```

```
                raise ValueError(f"Invalid xz_ab format: {param}")

            ab_params = self.split_params(ab_match.group(1))
            ab_index = int(ab_params[0].split('[')[1].split(']')[0])
            z_1 = self.vars_dict.get(ab_params[1].strip())
            z_n = self.vars_dict.get(ab_params[2].strip())

            ab_value = self.vars_dict['ab_matrix'][ab_index]
            step = (z_n - z_1) / (self.vars_dict.get('n') - 1)

            for row_idx in range(self.vars_dict.get('n')):
                input_matrix[row_idx][col_idx] = round(ab_value[0] * z_1 + ab_value[1], 10)
                z_1 += step
        else:
            try:
                value = eval(param)
                if isinstance(value, (int, float)):
                    for row_idx in range(self.vars_dict.get('n')):
                        input_matrix[row_idx][col_idx] = value
                elif isinstance(value, list):
                    if len(value) != self.vars_dict.get('n'):
                        raise ValueError(f"Array length {len(value)} != n={self.vars_dict.get('n')}")
                    for row_idx in range(self.vars_dict.get('n')):
                        input_matrix[row_idx][col_idx] = value[row_idx]
                else:
                    raise ValueError(f"Unsupported type: {type(value)}")
            except Exception as e:
                raise ValueError(f"Error evaluating {param}: {e}")

    return np.array(input_matrix)
```

Файл model.py

```
class ModelLoader:
    def __init__(self, l=2):
        self.result_dict = {}
```

```
        self.vars_dict = {'l': l, 'removed-cols': []}
        self.available_theta = []
        self.org_available_theta_len = 0

    def run_regression_script(self, path):
        with open(path, encoding='utf-8') as f:
            content = f.read()

        lines = content.splitlines()

        for line in lines:
            line = line.strip()
            if line and not line.startswith('#'):
                self.parse_line(line)

        return self.vars_dict
```

Файл mgmdh.py

```
    def sqrtOfSum(self, b):
        n = len(b)
        sqrtOfSum = 0
        for i in range(n):
            sqrtOfSum += b[i] ** 2
        return np.sqrt(sqrtOfSum)

    def com_norm(self, b_og, b):
        n = len(b)
        sum = 0
        sumofb_og = self.sqrtOfSum(b_og)
        sumofb = self.sqrtOfSum(b)
        for i in range(n):
            sum += (b_og[i] / sumofb_og - b[i] / sumofb) ** 2
        return np.sqrt(sum)

    def least_squares(self, X, Y):
        ans = np.linalg.inv(np.transpose(X) @ X) @ np.transpose(X) @ Y
```

```
    return ans

def con_norm_exp(self, mat, perm_y, half_y, y1, y2, teta1, teta2):
    return self.com_norm(y2, mat[perm_y[half_y:]] @ teta1) + self.com_norm(
        y1, mat[perm_y[:half_y]] @ teta2
    )

def zsk_MSE(self, mat, y, teta):
    t = 0
    for i in range(len(y)):
        sumA = 0
        for j in range(len(mat[0])):
            sumA += mat[i][j] * teta[j]
        t += (y[i] - sumA) ** 2
    return t

def zsk_NORM(self, mat, y, teta):
    return self.com_norm(y, mat @ teta)

def find_val(self, val, arr):
    for i in range(len(arr)):
        if arr[i] == val:
            return i
    return -1

def find_nonzero_zero(self, ans):
    certanlyNonzeroIndex = [np.argmax(np.abs(ans))]
    maybeZeroIndex = [np.argmin(np.abs(ans))]

    rest = [
        ans[i]
        for i in range(0, len(ans))
        if i != certanlyNonzeroIndex[0] and i != maybeZeroIndex[0]
    ]
    for i in range(0, len(rest)):
        tmp = np.max(np.abs(rest))
```

```
        tmp2 = int(np.argmax(np.abs(rest)))
        tmpind = self.find_val(rest[tmp2], ans)

        maxpos = (
            sum([abs(ans[nonzind]) for nonzind in certanlyNonzeroIndex])
            / len(certanlyNonzeroIndex)
            - tmp
        )
        minpos = tmp - abs(ans[maybeZeroIndex[0]])
        if maxpos < minpos:
            certanlyNonzeroIndex.append(tmpind)
        else:
            maybeZeroIndex.append(tmpind)

        rest = [
            ans[i]
            for i in range(0, len(ans))
            if i not in certanlyNonzeroIndex
            and i not in maybeZeroIndex
            and i != tmpind
        ]

    return sorted(certanlyNonzeroIndex), sorted(maybeZeroIndex)

def identify_comb(self, ans1, ans2):
    certanlyNonzeroIndex1, maybeZeroIndex1 = self.find_nonzero_zero(ans1)
    certanlyNonzeroIndex2, maybeZeroIndex2 = self.find_nonzero_zero(ans2)

    if certanlyNonzeroIndex1 != certanlyNonzeroIndex2:
        certanlyNonzeroIndex = list(
            set(certanlyNonzeroIndex1) & set(certanlyNonzeroIndex2)
        )
        maybeZeroIndex = list(set(maybeZeroIndex1) | set(maybeZeroIndex2))
    else:
        certanlyNonzeroIndex = certanlyNonzeroIndex1
        maybeZeroIndex = maybeZeroIndex1
```

```
    if len(certanlyNonzeroIndex) == 0:
        certanlyNonzeroIndex, maybeZeroIndex = self.find_nonzero_zero(ans1)

    comb = []
    for i in range(0, len(maybeZeroIndex) + 1):
        comb.extend(list(combinations(maybeZeroIndex, i)))

    return certanlyNonzeroIndex, comb

def create_A_matrix(self, X, matrix_data, m, p, n_samples):
    return np.exp(np.dot(np.log(np.clip(X, a_min=1e-10, a_max=None)), matrix_data))

def calcBestInd(
    self,
    mat,
    perm_y,
    half_y,
    certanlyNonzeroIndex,
    comb,
    yA2,
    yA1=None,
    method=None,
):
    if method is None:
        method = self.con_norm_exp

    full_con_norm_exp = [0] * len(comb)
    for cmbI in range(len(comb)):
        combinedIndex = certanlyNonzeroIndex + list(comb[cmbI])
        combinedIndex.sort()
        ml = mat[:, combinedIndex].copy()
        if method == self.zsk_MSE:
            teta = self.least_squares(ml[perm_y[half_y:]], yA2)
            full_con_norm_exp[cmbI] = self.zsk_MSE(ml[perm_y[half_y:]], yA2, teta)
        elif method == self.zsk_NORM:
```

```
            tetaml1 = self.least_squares(ml[perm_y[:half_y]], yA1)
            full_con_norm_exp[cmbI] = self.zsk_NORM(
                ml[perm_y[half_y:]], yA2, tetaml1
            )
        elif method == self.con_norm_exp:
            tetaml1 = self.least_squares(ml[perm_y[:half_y]], yA1)
            tetaml2 = self.least_squares(ml[perm_y[half_y:]], yA2)
            full_con_norm_exp[cmbI] = self.con_norm_exp(
                ml, perm_y, half_y, yA1, yA2, tetaml1, tetaml2
            )
        else:
            raise ValueError("Invalid method specified")

    min_zsk = min(full_con_norm_exp)
    czsk = 0
    best_comb = None
    for i in range(len(full_con_norm_exp)):
        if (
            full_con_norm_exp[i] <= 1.03 * min_zsk
            and full_con_norm_exp[i] >= 0.97 * min_zsk
        ):
            if best_comb is None or len(comb[i]) < len(best_comb):
                best_comb = comb[i]
                czsk = full_con_norm_exp[i]
            elif len(comb[i]) == len(best_comb):
                if full_con_norm_exp[i] < czsk:
                    czsk = full_con_norm_exp[i]
                    best_comb = comb[i]
    if certanlyNonzeroIndex is None:
        return list(best_comb)
    if best_comb is None:
        return list(certanlyNonzeroIndex)
    return sorted(list(best_comb) + certanlyNonzeroIndex)

def hi2(self, y1, y2, k=5):
    y_err = y1 - y2
```

```
y_std = np.std(y_err)
y_mean = np.mean(y_err)

y_min = np.min(y_err)
y_max = np.max(y_err)
avgminmax = (np.abs(y_max) + np.abs(y_min)) / k

xi = [np.min(y_err) + i * avgminmax for i in range(k)]
xi_1 = [np.min(y_err) + (i + 1) * avgminmax for i in range(k)]
x_ = [xi, xi_1]
x_star = np.sum([xi, xi_1], axis=0)
x_star = x_star / 2
x_star_avg = np.mean(x_star)

ni = [
   np.sum(
      [
         1 if (element < xi_1[i] and element >= xi[i]) else 0
         for element in y_err
      ]
   )
   for i in range(k)
]

zi = [(xi[i] - x_star_avg) / y_std for i in range(k)]
zi_1 = [(xi_1[i] - x_star_avg) / y_std for i in range(k)]
zi[0] = -np.inf
zi_1[-1] = np.inf
fzi = [norm.cdf(zi[i]) for i in range(k)]
fzi_1 = [norm.cdf(zi_1[i]) for i in range(k)]
p = [fzi_1[i] - fzi[i] for i in range(k)]
npi = [pi * len(y_err) for pi in p]
xi_sq = [(ni[i] - npi[i]) ** 2 / npi[i] for i in range(k)]

return np.sum(xi_sq)
```

```
        def hi2_start(self, ml1, coef_sym, ml2, coef_zsk, ml3, coef_nonsym, y, k=5):
            y_calc1 = np.array([ml1 @ coef_sym.T, ml2 @ coef_zsk.T, ml3 @ coef_nonsym.T])
            best_coef = np.array([self.hi2(y, y_calc1[i], k=k) for i in range(3)])
            ret = int(np.argmin(best_coef))

            return ret, best_coef[ret], [coef_sym, coef_zsk, coef_nonsym]

        def calculate_mse(self, y_true, y_pred):
            return np.mean(np.abs(y_true - y_pred))

        def calculate(self):
            try:
                n = int(self.inputs["n_value"].get())
                m = int(self.inputs["m_value"].get())
                k = int(self.inputs["k_value"].get())
                p = int(self.inputs["p_value"].get())

                try:
                    matrix_data = self.get_matrix_values()
                    matrix_shape = matrix_data.shape

                    if matrix_shape[0] != int(m) or matrix_shape[1] != int(p):
                        self.status_var.set(
                            f"Matrix dimensions ({matrix_shape[0]}x{matrix_shape[1]}) don't match mxp
({int(m)}×{int(p)})"
                        )
                        return


                except Exception as me:
                    self.status_var.set(f"Error processing matrix: {str(me)}")
                    return

                Y = self.loaded_data[:n, -1]
                X = self.loaded_data[:n, :-1]
```

```
np.random.seed(42)
X = (X - X.mean(axis=0)) / X.std(axis=0)

X_train, X_test, Y_train, Y_test = train_test_split(
   X, Y, test_size=k * 0.01, random_state=42
)

perm_y = np.random.permutation(len(Y_train))
half_y = len(Y_train) // 2
Y_1 = Y_train[perm_y[:half_y]]
Y_2 = Y_train[perm_y[half_y:]]

if X.shape[1] != m:
   self.status_var.set(
      "Mismatch: number of features in data does not match m"
   )
   return

A = self.create_A_matrix(X_train, matrix_data, m, p, X_train.shape[0])

A_test = self.create_A_matrix(X_test, matrix_data, m, p, X_test.shape[0])

pred1 = self.least_squares(A[perm_y[:half_y]], Y_1)
pred2 = self.least_squares(A[perm_y[half_y:]], Y_2)



certanlyNonzeroIndex, comb = self.identify_comb(pred1, pred2)

best_comb1 = self.calcBestInd(
   A,
   perm_y,
   half_y,
   certanlyNonzeroIndex,
   comb,
   Y_2,
   Y_1,
```

```
    method=self.con_norm_exp,
)
best_comb2 = self.calcBestInd(
    A, perm_y, half_y, certanlyNonzeroIndex, comb, Y_2, method=self.zsk_MSE
)
best_comb3 = self.calcBestInd(
    A,
    perm_y,
    half_y,
    certanlyNonzeroIndex,
    comb,
    Y_2,
    Y_1,
    method=self.zsk_NORM,
)

A1 = A[:, best_comb1].copy()
A2 = A[:, best_comb2].copy()
A3 = A[:, best_comb3].copy()

pred1 = self.least_squares(A1, Y_train)
pred2 = self.least_squares(A2, Y_train)
pred3 = self.least_squares(A3, Y_train)

retind1 = np.zeros(A.shape[1])
for i, idx in enumerate(best_comb1):
    retind1[idx] = pred1[i]

retind2 = np.zeros(A.shape[1])
for i, idx in enumerate(best_comb2):
    retind2[idx] = pred2[i]

retind3 = np.zeros(A.shape[1])
for i, idx in enumerate(best_comb3):
    retind3[idx] = pred3[i]
```

```
            coef_h_ind, best_coef, coef_h = self.hi2_start(
                A1, pred1, A2, pred2, A3, pred3, Y_train, k=int(n * k / 10)
            )

            A1_test = A_test[:, best_comb1].copy()
            A2_test = A_test[:, best_comb2].copy()
            A3_test = A_test[:, best_comb3].copy()

            Y_pred1 = A1_test @ pred1
            Y_pred2 = A2_test @ pred2
            Y_pred3 = A3_test @ pred3

            mse1 = self.calculate_mse(Y_test, Y_pred1)
            mse2 = self.calculate_mse(Y_test, Y_pred2)
            mse3 = self.calculate_mse(Y_test, Y_pred3)

            self.update_output_field(
                self.prediction_frames[0]["p"], format_coefficient_list(retind1)
            )
            self.update_output_field(
                self.prediction_frames[1]["p"], format_coefficient_list(retind2)
            )
            self.update_output_field(
                self.prediction_frames[2]["p"], format_coefficient_list(retind3)
            )

        except Exception as e:
            self.status_var.set(f"Error during calculation: {str(e)}")
```

ДОДАТОК Б

Приклад .poly файлу

```
# Максимальний степінь БПР
r=5
# Кількість спостережень
n=10
# Кількість вхідних змінних
m=6
# Точність
acc=100
# Повтори активного експерименту
l=1
# Ініціалізація матриці, яка містить області, в яких можуть приймати значення вхідні змінні
input_vars_ranges_matrix=[[1, 10], [1, 10], [1, 10], [1, 10], [0, 5], [1,5]]
# Ініціалізація матриці, яка містить надлишковий опис БПР
redundant_description_matrix=[[0,2,2,1,1,0,1,2,1,1],[0,1,2,2,2,0,1,0,0,1],[0,3,0,0,0,3,3,0,0,1],[0,0,1,1,2,
2,0,3,3,1],[0,0,0,0,0,1,0,6,0,0],[0,0,0,0,0,0,1,0,2,1]]
# Області, в яких можуть приймати значення вхідні змінні віртуальної ОПР, верхня та нижня границі
z_1=-50
z_n=50
# Матриця із значеннями коефіцієнтів НОПФ
q_matrix=default
# Матриця із значеннями параметрів переходу від реальної до віртуальної ОПР
ab_matrix=ab_calc(input_vars_ranges_matrix,z_1,z_n)

# Ініціалізація матриці, що містить значення вхідних змінних, на яких проводиться активний експеримент
input_matrix=input_matrix_init([xz_ab(ab_matrix[0],z_1,z_n),10,xz_ab(ab_matrix[2],z_1,z_n),1,1,1])
# Застосування агрегованого оператора 2 до члена надлишкового опису БПР з індексом 1
dec_method_ao2(input_matrix, redundant_description_matrix[1])
#print(result_dict)
# Вектор значень активного експерименту для відповідної ОПР
y_act=y_act_reader(file_path.txt, input_matrix)
```

```
# Обчислення оцінки коефіцієнта члена надлишкового опису БПР з індексом 1
dec_method(input_matrix, y_act, l, redundant_description_matrix[1])
# print(result_dict)

# Ініціалізація матриці, що містить значення вхідних змінних, на яких проводиться активний експеримент
input_matrix=input_matrix_init([xz_ab(ab_matrix[0],z_1,z_n),xz_ab(ab_matrix[1],z_1,z_n),1,10,1,1])
# Застосування агрегованого оператора 2 до члена надлишкового опису БПР з індексом 2
dec_method_ao2(input_matrix, redundant_description_matrix[2])
#print(result_dict)
# Вектор значень активного експерименту для відповідної ОПР
y_act=y_act_reader(file_path.txt, input_matrix)
# Обчислення оцінки коефіцієнта члена надлишкового опису БПР з індексом 2
dec_method(input_matrix, y_act, l, redundant_description_matrix[2])
#print(result_dict)

# Ініціалізація матриці, що містить значення вхідних змінних, на яких проводиться активний експеримент
input_matrix=input_matrix_init([10,xz_ab(ab_matrix[1],z_1,z_n),1,xz_ab(ab_matrix[2],z_1,z_n),1,1])
# Застосування агрегованого оператора 2 до члена надлишкового опису БПР з індексом 4
dec_method_ao2(input_matrix, redundant_description_matrix[4])
#print(result_dict)
# Вектор значень активного експерименту для відповідної ОПР
y_act=y_act_reader(file_path.txt, input_matrix)
# Обчислення оцінки коефіцієнта члена надлишкового опису БПР з індексом 4
dec_method(input_matrix, y_act, l, redundant_description_matrix[4])
#print(result_dict)

# Ініціалізація матриці, що містить значення вхідних змінних, на яких проводиться активний експеримент
input_matrix=input_matrix_init([10,xz_ab(ab_matrix[1],z_1,z_n),1,10,1,1])
# Застосування агрегованого оператора 1 до члена надлишкового опису БПР з індексом 3
dec_method_ao1(input_matrix, redundant_description_matrix[3])
#print(result_dict)
# Вектор значень активного експерименту для відповідної ОПР
y_act=y_act_reader(file_path.txt, input_matrix)
```

```
# Обчислення оцінки коефіцієнта члена надлишкового опису БПР з індексом 3
dec_method(input_matrix, y_act, l, redundant_description_matrix[3])
#print(result_dict)

# Ініціалізація матриці, що містить значення вхідних змінних, на яких проводиться активний експеримент
input_matrix=input_matrix_init([10,10,xz_ab(ab_matrix[2],z_1,z_n),1,0,5])
# Застосування агрегованого оператора 1 до члена надлишкового опису БПР з індексом 6
dec_method_ao1(input_matrix, redundant_description_matrix[6])
#print(result_dict)
# Вектор значень активного експерименту для відповідної ОПР
y_act=y_act_reader(file_path.txt, input_matrix)
# Обчислення оцінки коефіцієнта члена надлишкового опису БПР з індексом 6
dec_method(input_matrix, y_act, l, redundant_description_matrix[6])
#print(result_dict)

# Ініціалізація матриці, що містить значення вхідних змінних, на яких проводиться активний експеримент
input_matrix=input_matrix_init([1,1,xz_ab(ab_matrix[2],z_1,z_n),10,5,1])
# Застосування агрегованого оператора 1 до члена надлишкового опису БПР з індексом 5
dec_method_ao1(input_matrix, redundant_description_matrix[5])
#print(result_dict)
# Вектор значень активного експерименту для відповідної ОПР
y_act=y_act_reader(file_path.txt, input_matrix)
# Обчислення оцінки коефіцієнта члена надлишкового опису БПР з індексом 5
dec_method(input_matrix, y_act, l, redundant_description_matrix[5])
#print(result_dict)

# Ініціалізація матриці, що містить значення вхідних змінних, на яких проводиться активний експеримент
input_matrix=input_matrix_init([xz_ab(ab_matrix[0],z_1,z_n),1,1,xz_ab(ab_matrix[2],z_1,z_n),5,1])
# Застосування агрегованого оператора 2 до члена надлишкового опису БПР з індексом 7
dec_method_ao2(input_matrix, redundant_description_matrix[7])
#print(result_dict)
# Вектор значень активного експерименту для відповідної ОПР
y_act=y_act_reader(file_path.txt, input_matrix)
```

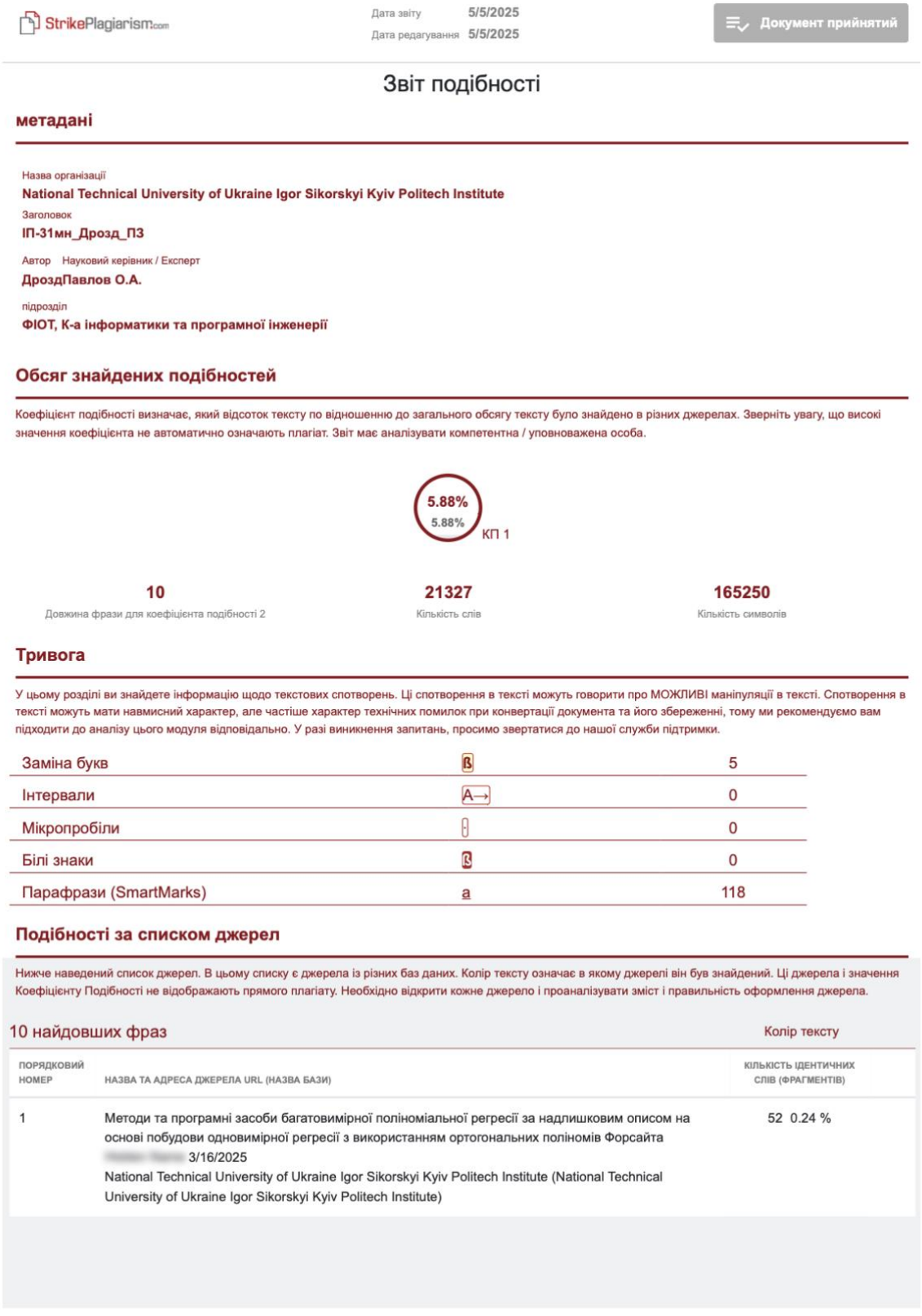
```
# Обчислення оцінки коефіцієнта члена надлишкового опису БПР з індексом 7
dec_method(input_matrix, y_act, l, redundant_description_matrix[7])
#print(result_dict)

# Ініціалізація матриці, що містить значення вхідних змінних, на яких проводиться активний експеримент
input_matrix=input_matrix_init([10,1,1,xz_ab(ab_matrix[3],z_1,z_n),1,xz_ab(ab_matrix[5],z_1,z_n)])
# Застосування агрегованого оператора 2 до члена надлишкового опису БПР з індексом 8
dec_method_ao2(input_matrix, redundant_description_matrix[8])
#print(result_dict)
# Вектор значень активного експерименту для відповідної ОПР
y_act=y_act_reader(file_path.txt, input_matrix)
# Обчислення оцінки коефіцієнта члена надлишкового опису БПР з індексом 8
dec_method(input_matrix, y_act, l, redundant_description_matrix[8])
#print(result_dict)

# Ініціалізація матриці, що містить значення вхідних змінних, на яких проводиться активний експеримент
input_matrix=input_matrix_init([10,xz_ab(ab_matrix[1],z_1,z_n),10,xz_ab(ab_matrix[3],z_1,z_n),1,5])
# Застосування агрегованого оператора 4 до члена надлишкового опису БПР з індексом 9
dec_method_ao4(input_matrix, redundant_description_matrix[9])
#print(result_dict)
# Вектор значень активного експерименту для відповідної ОПР
y_act=y_act_reader(file_path.txt, input_matrix)
# Обчислення оцінки коефіцієнта члена надлишкового опису БПР з індексом 9
dec_method(input_matrix, y_act, l, redundant_description_matrix[9])
print(result_dict)
```

ДОДАТОК В

Результати перевірки роботи на співпадіння



StrikePlagiarism.com

Дата звіту 5/5/2025
Дата редагування 5/5/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації
National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute
Заголовок
ІП-31мн_Дрозд_ПЗ
Автор Науковий керівник / Експерт
ДроздПавлов О.А.
підрозділ
ФІОТ, К-а інформатики та програмної інженерії

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

5.88%
5.88%
КП 1

10	21327	165250
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	5
Інтервали	A→	0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)	a	118

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Методи та програмні засоби багатовимірної поліноміальної регресії за надлишковим описом на основі побудови одновимірної регресії з використанням ортогональних поліномів Форсайта [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	52 0.24 %

2	Методи та програмні засоби багатовимірної поліноміальної регресії за надлишковим описом на основі побудови одновимірної регресії з використанням ортогональних поліномів Форсайта [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	41 0.19 %
3	Методи та програмні засоби багатовимірної поліноміальної регресії за надлишковим описом на основі побудови одновимірної регресії з використанням ортогональних поліномів Форсайта [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	37 0.17 %
4	https://core.ac.uk/download/pdf/33758662.pdf	28 0.13 %
5	Методи та програмні засоби багатовимірної поліноміальної регресії за надлишковим описом на основі побудови одновимірної регресії з використанням ортогональних поліномів Форсайта [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	25 0.12 %
6	Програмне забезпечення для статистичного дослідження ефективності методу побудови багатовимірної лінійної регресії [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	25 0.12 %
7	Методи та програмні засоби багатовимірної поліноміальної регресії за надлишковим описом на основі побудови одновимірної регресії з використанням ортогональних поліномів Форсайта [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	24 0.11 %
8	Методи та програмні засоби багатовимірної поліноміальної регресії за надлишковим описом на основі побудови одновимірної регресії з використанням ортогональних поліномів Форсайта [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	23 0.11 %
9	Методи та програмні засоби багатовимірної поліноміальної регресії за надлишковим описом на основі побудови одновимірної регресії з використанням ортогональних поліномів Форсайта [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	23 0.11 %
10	Методи та програмні засоби багатовимірної поліноміальної регресії за надлишковим описом на основі побудови одновимірної регресії з використанням ортогональних поліномів Форсайта [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	21 0.10 %

з домашньої бази даних (5.55 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Методи та програмні засоби багатовимірної поліноміальної регресії за надлишковим описом на основі побудови одновимірної регресії з використанням ортогональних поліномів Форсайта [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**1012 (80) 4.75 %**
2	Програмне забезпечення для статистичного дослідження ефективності методу побудови багатовимірної лінійної регресії [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**86 (8) 0.40 %**

3	Статичні алгоритми управління паралельними асинхронними обчислювальними процесами [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**72 (6) 0.34 %**
4	Рекомендаційна система на базі алгоритму колаборативної фільтрації [redacted] 3/15/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**13 (1) 0.06 %**

з програми обміну базами даних (0.07 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	КМР_Алєксєєнко_Дмитро_Євгенович [redacted] 11/30/2024 **V. N. Karazin Kharkiv National University (KKNU) (ННІ" Каразінський банківський інститут")**	**14 (1) 0.07 %**

з Інтернету (0.26 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	**https://core.ac.uk/download/pdf/33758662.pdf**	**28 (1) 0.13 %**
2	https://ela.kpi.ua/bitstreams/c92e1d80-faa2-4e94-9d1b-b42210d77767/download	16 (2) 0.08 %
3	**http://ela.kpi.ua/bitstream/123456789/18954/1/Report_2016.pdf**	**12 (1) 0.06 %**

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)