НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова
праця на правах рукопису

ГОЛОВЧЕНКО МАКСИМ МИКОЛАЙОВИЧ

УДК 004.42:519.237

ДИСЕРТАЦІЯ

МЕТОДИ ТА ПРОГРАМНІ ЗАСОБИ БАГАТОВИМІРНОЇ ПОЛІНОМІАЛЬНОЇ РЕГРЕСІЇ ЗА НАДЛИШКОВИМ ОПИСОМ НА ОСНОВІ ПОБУДОВИ ОДНОВИМІРНОЇ РЕГРЕСІЇ З ВИКОРИСТАННЯМ ОРТОГОНАЛЬНИХ ПОЛІНОМІВ ФОРСАЙТА

121 Інженерія програмного забезпечення
12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

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

Науковий керівник: Павлов Олександр Анатолійович, д. т. н., професор

Київ – 2023

АНОТАЦІЯ

Головченко М.М. Методи та програмні засоби багатовимірної поліноміальної регресії за надлишковим описом на основі побудови одновимірної регресії з використанням ортогональних поліномів Форсайта. – Кваліфікаційна наукова праця на правах рукопису.

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.

Дисертаційна робота присвячена розробці універсального синтетичного методу оцінок коефіцієнтів багатовимірної поліноміальної регресії, заданої надлишковим описом та створенням оригінальних програмних засобів, що ефективно реалізують цей метод. У процесі розробки та дослідження ефективності синтетичного методу були отримані такі результати.

Вперше розроблено синтетичний метод побудови багатовимірної поліноміальної регресії, заданої надлишковим описом, що відрізняється від існуючих тим, що органічно поєднує риси класичного методу (теоретично обґрунтовані випадки, в яких оцінка коефіцієнтів при нелінійних членах знаходиться з заданою точністю) з ефективністю евристичних методів (знаходження структури регресії з використанням перевірочної послідовності в модифікованому методі групового урахування аргументів, що входить в склад синтетичного методу), а також включає в себе метод побудови одновимірної поліноміальної регресії на основі довільного повторного активного експерименту з використанням лише одного набору нормованих ортогональних поліномів Форсайта, декомпозиційний метод оцінки коефіцієнтів при нелінійних членах багатовимірної поліноміальної регресії з наперед заданою точністю, що багатовимірну задачу зводить до послідовної побудови відповідних одновимірних поліноміальних регресій.

Вперше обґрунтовано можливість знаходження нормованих ортогональних поліномів Форсайта з наперед заданою точністю, яка досягається за рахунок представлення даних у вигляді раціональних дробів та застосування до них символьних обчислень, що дозволяє отримати оцінки коефіцієнтів при нелінійних членах багатовимірної поліноміальної регресії, значення яких відповідають теоретично виведеним умовам.

Вперше приведено теоретичне обґрунтування зменшення обчислювальної складності програмного забезпечення реалізації методу найменших квадратів на основі повторних експериментів, що полягає в заміні операцій з матрицями повного активного експерименту операціями з матрицями основного експерименту суттєво меншої розмірності.

Вперше запропоновано архітектуру кросплатформної бібліотеки для реалізації синтетичного методу та його складових, яка дозволяє використовувати її компоненти, як окремо, так і в цілому для розв'язання прикладних задач побудови регресійних моделей.

Проведено дослідження ефективності алгоритмів, що реалізують операції з матрицями в методі найменших квадратів, зокрема обґрунтовано можливість використання паралельних обчислень. Розглядаються матричні операції, які входять у формулу для знаходження оцінок коефіцієнтів багатовимірної поліноміальної регресії методом найменших квадратів як складової модифікованого методу групового урахування аргументів та можуть виконуватись ефективніше при застосуванні паралельних обчислень. Було виконано дослідження ефективності алгоритмів множення матриць та обернення матриць, як складових задачі знаходження оцінок коефіцієнтів багатовимірної поліноміальної регресії модифікованим методом групового урахування аргументів. Це дослідження ефективності алгоритмів виконувалось шляхом реалізації алгоритмів на базі 8-ми ядерного мікропроцесора Apple M1 та фіксації часу роботи різних алгоритмів для фіксованої кількості потоків та квадратних матриць заданої розмірності.

Приведено обґрунтування можливості розпаралелювання обчислень в модифікованому методі групового урахування аргументів для знаходження оцінок коефіцієнтів часткових описів та залишкових сум квадратів.

Реалізовано кросплатформну бібліотеку та її програмний інтерфейс для побудови регресійних моделей. Так, для користувачів, які володіють базовими навичками у області програмування і статистичного аналізу та бажають отримати розв'язок задачі побудови багатовимірної поліноміальної регресії достатньо у автоматизованому режимі підключити кросплатформну бібліотеку у власний застосунок та передати у її функції бібліотеки вхідні дані. В цьому випадку функції кросплатформної бібліотеки аналізують надлишковий опис і визначають, що він відноситься до класу, у якому всі лінійні системи мають лише одну змінну, якщо це не так, то задача розв'язується повністю модифікованим методом групового урахування аргументів. У протилежному випадку, програма аналізує можливості декомпозиційного методу та видає вимоги для проведення відповідної кількості повторних активних експериментів для побудови одновимірних поліноміальних регресій та множину коефіцієнтів, які будуть оцінені. Далі формується багатовимірна поліноміальна регресія задана надлишковим описом, яка буде розв'язана модифікованим методом групового урахування аргументів.

Для користувачів, які володіють розширеними навичками у області програмування і статистичного аналізу та бажають отримати розв'язок задачі побудови багатовимірної поліноміальної регресії, повинні ознайомитись з детальною інструкцією по роботі з кросплатформною бібліотекою, у якій описані теоретичні положення та практичні рекомендації з використання синтетичного методу. Далі користувачі за допомогою функцій кросплатформної бібліотеки у частині декомпозиційного методу можуть запрограмувати індивідуальний алгоритм розв'язку задачі на основі теоретичних положень синтетичного методу. У частині модифікованого методу групового урахування аргументів все залишається без змін.

За результатами виконання синтетичного методу користувачу буде виданий кінцевий результат, який містить структуру багатовимірної поліноміальної регресії, знайдені декомпозиційним методом оцінки коефіцієнтів багатовимірної поліноміальної регресії та їх дисперсії та оцінки коефіцієнтів багатовимірної поліноміальної ре-

гресії з оцінками їх дисперсій, знайдені за допомогою модифікованого методу групового урахування аргументів, з оцінкою результатів – має високу ступінь достовірності; задовільну ступінь достовірності; результат недостовірний.

При розробці кросплатформної бібліотеки, що реалізує синтетичний метод, використовувались такі допоміжні засоби.

В якості мови реалізації кросплатформної бібліотеки була обрана Python, оскільки дана мова програмування краще за інші підходить для реалізації data science та статистичних методів обробки даних. Крім того для мови Python існує великий набір високорівневих фреймворків різного призначення та програмних бібліотек, які можна використовувати у якості допоміжних засобів при розробці.

Середовищем розробки кросплатформної бібліотеки було обрано IntelliJ IDEA через наявність безкоштовної експрес-версії та зручність встановлення допоміжних програмних пакетів та бібліотек.

У якості архітектури кросплатформної бібліотеки було використано монолітну архітектуру, оскільки решта архітектур програмного забезпечення – багаторівнева, клієнт-серверна, мікросервісна, сервісно-орієнтована – не рекомендується для використання при розробці такого роду програмного забезпечення. Внутрішня логіка бібліотеки побудована з використанням компонентно-орієнтованого підходу, так як даний підхід добре себе зарекомендував при розробці програмного забезпечення цільового призначення, що використовується при розробці цільового прикладного програмного забезпечення.

В якості пакету для паралельної реалізації деяких підалгоритмів було використано multiprocessing, а для реалізації символьних обчислень було використано бібліотеку SymPy.

Для розгортання кросплатформної бібліотеки було використано систему управління пакетами pip. Це універсальна, зручна і найбільш популярна система управління пакетами, написаними для мови програмування Python. В результаті розгортання користувачі зможуть завантажувати і встановлювати бібліотеку для своїх потреб командою pip install regression_lib_mpr у своєму середовищі розробки.

Дисертаційна робота складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел із 84 найменувань на дев’яти сторінках та чотирьох додатків. Загальний обсяг дисертації становить 185 сторінок, з яких 128 сторінок основного тексту, містить 23 рисунки та 24 таблиці.

Ключові слова: багатовимірна поліноміальна регресія, математичні моделі об’єктів, ідентифікація, нейронні мережі, інтелектуальний аналіз, ефективність алгоритмів, оптимізація, алгоритми моделювання, розробка програмного забезпечення, засоби розробки, програмні бібліотеки, паралельні обчислення, багатопоточність, символьні обчислення.

ABSTRACT

Holovchenko M.M. Methods and software tools for constructing a multivariate polynomial regression from a redundant representation based on the construction of univariate regression using orthogonal polynomials of Forsythe. Qualifying scientific work is presented on the rights of the manuscript.

The philosophy doctor thesis is carried out in specialty 121 – Software Engineering, of the knowledge field 12 – Information Technologies. – National Technical University of Ukraine "Kyiv Polytechnic Institute", Ministry of Education and Science of Ukraine, Kyiv, 2023.

The thesis is devoted to the development of a universal synthetic method of estimating the coefficients of a multivariate polynomial regression given by a redundant representation and to the creation of original software tools that effectively implement this method. In the process of developing and researching the efficiency of the synthetic method, the following results were obtained.

For the first time, a synthetic method of constructing a multivariate polynomial regression given by a redundant representation is developed. The method differs from the existing ones in that it organically combines the features of the classical method (theoretically substantiated cases in which the estimates of the coefficients at nonlinear terms are found with a given accuracy) with the efficiency of heuristic methods (finding the regression structure using a test sequence in the modified Group Method of Data Handling included in the synthetic method). The synthetic method also includes a method of constructing a univariate polynomial regression based on an arbitrary repeated active experiment using only a single set of normalized orthogonal polynomials of Forsythe, a decomposition method for estimating coefficients at nonlinear terms of the multivariate polynomial regression with predetermined accuracy that reduces the multivariate problem to the sequential construction of the corresponding univariate polynomial regressions.

For the first time, the possibility of finding normalized orthogonal polynomials of Forsythe with a predetermined accuracy is substantiated. The accuracy is achieved by presenting data in the form of rational fractions and applying symbolic calculations to them.

This makes it possible to estimate the coefficients at nonlinear terms of a multivariate polynomial regression, the values of which correspond to the theoretically derived conditions.

For the first time, the theoretical substantiation is presented for reducing the computational complexity of the software that implements the least squares method based on repeated experiments. The reduction consists in replacing operations with matrices of the full active experiment by operations with matrices of the main experiment that are significantly smaller in dimensions.

For the first time, the architecture of the cross-platform library for implementation of the synthetic method and its components is proposed. The library allows using its components, both individually and as a whole, to solve applied problems of regression models building.

A study of the efficiency of algorithms that implement operations with matrices in the least squares method was conducted. In particular, the possibility of using parallel calculations was substantiated. Have been considered matrix operations included in the formula for finding estimates of the coefficients of a multivariate polynomial regression using the least squares method as a component of the Modified Group Method of Data Handling and can be performed more efficiently when using parallel calculations. A study of the efficiency of matrix multiplication and matrix inversion algorithms was conducted as they are components of the problem of a multivariate polynomial regression coefficients estimation using the Modified Group Method of Data Handling. The study was performed by implementing the algorithms based on an 8-core Apple M1 microprocessor and fixing the operating time of various algorithms for a fixed number of threads and square matrices of a given dimension.

The substantiation for the possibility of parallelizing calculations in the Modified Group Method of Data Handling is given for the case of estimating the coefficients of partial representations and residual sums of squares.

A cross-platform library and its software interface for constructing regression models have been implemented. Thus, for users who have basic skills in the field of programming and statistical analysis and want to get a solution to the problem of constructing a multivariate polynomial regression, it is enough to automatically connect the cross-platform library to their own application and transfer the input data to the library’s functions. In this case,

the functions of the cross-platform library analyze the redundant representation and determine if it belongs to the class in which all linear systems have only a single variable. If this is not the case, then the problem is solved completely with the Modified Group Method of Data Handling. In the opposite case, the program analyzes the possibilities of the decomposition method and issues requirements for conducting the appropriate number of repeated active experiments to construct univariate polynomial regressions and the set of coefficients to be estimated. Next, a multivariate polynomial regression given by the redundant representation is formed, it will be solved by the Modified Group Method of Data Handling.

For users who have advanced skills in programming and statistical analysis and wish to obtain a solution to the problem of the multivariate polynomial regression construction, they should read the detailed instructions for working with the cross-platform library, which describe the theoretical provisions and practical recommendations for using the synthetic method. Further, with the help of the cross-platform library functions in the decomposition method part, users can program an individual algorithm for the problem solving based on the theoretical provisions of the synthetic method. In the part of the Modified Group Method of Data Handling, everything remains unchanged.

According to the results of the synthetic method, the user will be given the final result containing the structure of the multivariate polynomial regression, the estimates of the coefficients of the multivariate polynomial regression and their variances found by the decomposition method, and the estimates of the coefficients of the multivariate polynomial regression with the estimates of their variances found using the Modified Group Method of Data Handling, with the results evaluation: has a high degree of reliability; has a satisfactory degree of reliability; the result is unreliable.

When developing a cross-platform library that implements the synthetic method, the following tools were used.

Python was chosen as the implementation language of the cross-platform library, because this programming language is better than others for implementing data science and statistical methods of data processing. In addition, for the Python language, there is a large

set of high-level frameworks for various purposes and software libraries that can be used as software development tools.

IntelliJ IDEA was chosen as the cross-platform library development environment due to the availability of a free express version and the convenience of installing pre-intermediate software packages and libraries.

A monolithic architecture was used as the architecture of the cross-platform library, since the rest of the software architectures – multi-level, client-server, microservice, service-oriented – are not recommended for use in the development of this kind of software. The internal logic of the library is built using a component-oriented approach, as this approach has proven itself well in the development of target software, which is used in the development of target application software.

Multiprocessing was used as a package for the parallel computing implementation of some sub-algorithms, and the *SymPy* library was used to implement symbolic computing.

The package installer *pip* was used to deploy the cross-platform library. It is a universal, convenient, and most popular package management system written for the Python programming language. As a result of the deployment, users will be able to download and install the library for their needs with the *pip install regression_lib_mpr* command in their development environment.

The thesis consists of an introduction, four chapters, general conclusions, the reference list with 84 references on nine pages, and four appendices. The total volume of the thesis is 185 pages, of which 128 pages are the main text, contains 23 figures and 24 tables.

Key words: multivariate polynomial regression, mathematical models of objects, identification, neural networks, intelligent analysis, efficiency of algorithms, optimization, modeling algorithms, software development, software tools, software libraries, parallel computing, multithreading, symbolic computing.

Список публікацій здобувача

Наукові праці, в яких опубліковано основні наукові результати дисертації:

1. Павлов О. А., **Головченко М. М.** Побудова одновимірної і багатовимірної поліноміальної регресії за надлишковим описом з використанням активного експерименту // Вісник Національного технічного університету «ХПІ». Серія: Системний аналіз, управління та інформаційні технології. № 1(3), 2020. С. 3–8. doi: 10.20998/2079-0023.2020.01.02.
2. Pavlov A., **Holovchenko M.**, Mukha I., Lishchuk K. 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. Cham: Springer, 2022. Vol. 134. P. 288–303. doi: 10.1007/978-3-031-04812-8_25 (Проіндексовано в **Scopus**)
3. 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. К.: НТУУ «КПІ», 2022. Т. 1. № 40. С. 110–117. doi: 10.20535/1560-8956.40.2022.261665
4. Pavlov A.A. **Holovchenko M.N.**, 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: 10.20998/2079-0023.2022.01.01.
5. Pavlov A. A., **Holovchenko M. N.** Modified method of constructing a multivariate linear regression given by a redundant description // Bulletin of National Technical University “KhPI”. Series: System analysis, control and information technologies. № 2(8), 2022. С. 3–8. doi: 10.20998/2079-0023.2022.02.01.
6. Pavlov, A., **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 / Advances in Computer Sci-

ence for Engineering and Education VI. ICCSEEA 2023 // Lecture Notes on Data Engineering and Communications Technologies. Cham: Springer, 2023. Vol. 181. P. 207–222. doi: 10.1007/978-3-031-36118-0_19 (Проіндексовано в **Scopus**)

7. Pavlov A. A., **Holovchenko M. N.**, 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: 10.20998/2079-0023.2023.01.01.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

8. Дрозд В.В., **Головченко М.М.** Методи та програмні засоби побудови нелінійних поліноміальних регресій з використанням нормованих ортогональних поліномів Форсайта // Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2021) : матеріали тез доповідей І Всеукраїнської наук.-практ. конф. молодих вчених та студентів (м. Київ, 22–26 листопада 2021р.). Секція кафедри інформатики та програмної інженерії. – К. : КПІ ім. Ігоря Сікорського, 2021. – С. 82–86.
9. Павлов О.А., **Головченко М.М.,** Дрозд В.В., Ревич М.М. Дослідження ефективності методу побудови багатовимірної лінійної регресії, заданої надлишковим описом // Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2022 Осінь) : матеріали тез доповідей ІІІ Всеукраїнської наук.-практ. конф. молодих вчених та студентів (м. Київ, 22–25 жовтня 2022 р.). – К. : КПІ ім. Ігоря Сікорського, 2023. – С. 10–13. URL: https://drive.google.com/file/d/1CP9EaBTT_rJAXsINbanSVGnP2jkg9FJ0/view (Дата звернення: 29.08.2023).
10. 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. URL: http://ir.stu.cn.ua/123456789/28221 (Дата звернення: 29.08.2023)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БЛР	–	Багатовимірна лінійна регресія
БПР	–	Багатовимірна поліноміальна регресія
ЗСК	–	Залишкова сума квадратів
МГУА	–	Метод групового урахування аргументів
ММГУА	–	Модифікований метод групового урахування аргументів
ММП	–	Метод максимальної правдоподібності
МНК	–	Метод найменших квадратів
НОПФ	–	Нормовані ортогональні поліноми Форсайта
ОПР	–	Одновимірна поліноміальна регресія
ОС	–	Операційна система
IDE	–	Integrated Development Environment – інтегроване середовище розробки
IT	–	Information Technologies – інформаційні технології

ВСТУП

Актуальність теми. Регресійний аналіз на сьогоднішній день являє собою один з найбільш поширених універсальних засобів виявлення прихованих взаємозв'язків в даних та побудови по результатам статистичних випробувань лінійних та нелінійних одновимірних та багатовимірних регресій, що широко використовується в сучасних інформаційних діагностичних системах.

Однак, кожен з існуючих універсальних методів побудови регресійних моделей має свої достатньо суттєві обмеження при використанні і фактично вони доповнюють один одного. Вибір універсального методу при розв'язанні конкретної задачі фактично не є формалізованим і не гарантує її ефективного розв'язку. Як наслідок, повністю формалізованих універсальних програмних засобів також не існує. Усі існуючі універсальні методи можна розбити на дві категорії: класичні статистичні методи, такі як класичний метод найменших квадратів, метод максимальної правдоподібності, факторний аналіз, множинний дискримінантний аналіз [1, 16, 26, 29, 31, 32, 34, 51, 52, 66, 69, 83] та інші, які певним чином теоретично обґрунтовують отриманий результат, наприклад, з використанням критеріїв – суми квадратів залишків, коефіцієнту детермінації R^2, статистики Малоуза, F-статистика, регресійної суми квадратів, залишкової суми квадратів (ЗСК), t-впорядкованого пошуку, виключення та покрокової регресії тощо [1, 16, 26, 29, 31, 32, 34, 51, 52, 66, 69, 83], та евристичні методи, такі як метод групового урахування аргументів, генетичні алгоритми, штучні нейронні мережі та інші [9, 13, 25, 28, 35, 38, 42, 62, 78], ефективність використання яких підтверджуються розв'язком достатньо великої кількості реальних задач.

Таким чином, на сьогоднішній день визнаного найкращого універсального методу для побудови одновимірних та багатовимірних регресій не існує.

Звідки розробка та дослідження ефективності нових універсальних методів, що органічно поєднують властивості класичних методів та евристичних і на їх основі відповідних програмних засобів є актуальною, як в теоретичному так і в прикладному плані.

Зв'язок роботи з науковими програмами, планами, темами. Тема дисертаційної роботи входить в затверджений план наукової роботи кафедри інформатики та програмної інженерії КПІ ім. Ігоря Сікорського, що враховує постанову Кабінету Міністрів України від 9 травня 2023 р. № 463, яка містить перелік пріоритетних тематичних напрямів наукових досліджень і науково-технічних розробок з урахуванням потреб періоду воєнного стану та відновлення України від наслідків війни, а саме у сфері Інформаційних та комунікаційних технологій напрямку Інтелектуальні інтерактивні інформаційно-аналітичні системи. Запропоновані у дисертації методи створенні в рамках досліджень, що проводяться науковою групою ФІОТ-03 «Нелінійний регресійний аналіз, комбінаторна оптимізація в умовах невизначеності, теорія ПДС-алгоритмів для важкорозв'язуваних задач комбінаторної оптимізації, календарне та оперативне планування виробничих систем з мережевим представленням технологічних процесів» під керівництвом заслуженого діяча науки та техніки України, лауреата державної премії України в галузі науки та техніки професора кафедри інформатики та програмної інженерії О. А. Павлова, яка входить до складу очолюваної ним наукової школи «Математичне моделювання в інтелектуальних інформаційних системах», затвердженої Рішенням Вченої ради КПІ ім. Ігоря Сікорського від 20 лютого 2023 року (протокол №2).

Мета і завдання дослідження. Метою дисертаційної роботи є підвищення точності побудови моделі багатовимірної поліноміальної регресії за рахунок створення кросплатформної спеціалізованої бібліотеки, що реалізує нові теоретичні положення та алгоритми, з використанням сучасних обчислювальних можливостей.

Для досягнення вказаної мети потрібно вирішити такі завдання:

– дослідити сучасні методи побудови одновимірних та багатовимірних регресійних моделей;
– створити програмну реалізацію знаходження нормованих ортогональних поліномів Форсайта з точними значеннями наперед заданої кількості розрядів після коми за рахунок представлення даних у вигляді раціональних дробів та застосування до них символьних обчислень;

- створити метод побудови одновимірної поліноміальної регресії на основі довільного повторного активного експерименту з використанням лише одного набору нормованих ортогональних поліномів Форсайта;
- створити декомпозиційний метод оцінки коефіцієнтів при нелінійних членах багатовимірної поліноміальної регресії з наперед заданою точністю, що багатовимірну задачу декомпозує на послідовність побудови відповідних одновимірних поліноміальних регресій;
- створити, використовуючи постановку задачі побудови багатовимірної поліноміальної регресії, заданої надлишковим описом, модифікований метод групового урахування аргументів;
- створити, на основі отриманих попередніх теоретичних результатів, синтетичний метод побудови багатовимірної поліноміальної регресії, заданої надлишковим описом;
- провести оптимізацію окремих компонент програмного забезпечення, що реалізує синтетичний метод побудови багатовимірної поліноміальної регресії, заданої надлишковим описом;
- розробити архітектуру кросплатформної бібліотеки, яка реалізовуватиме синтетичний метод побудови багатовимірної поліноміальної регресії, заданої надлишковим описом, та матиме зручний програмний інтерфейс для користувачів різної базової підготовки.

Об’єкт дослідження – алгоритмічне та програмне забезпечення знаходження оцінок коефіцієнтів багатовимірної поліноміальної регресії.

Предмет дослідження – процес створення кросплатформної спеціалізованої бібліотеки, що реалізує нові універсальні алгоритми побудови багатовимірної поліноміальної регресії, заданої надлишковим описом.

Методи дослідження – теорія ймовірностей, регресійний аналіз, символьні обчислення, програмні засоби високопродуктивних, зокрема паралельних, обчислень, методи об’єктно-орієнтовано та процедурного програмування.

Наукова новизна отриманих результатів:

– вперше розроблено синтетичний метод побудови багатовимірної поліноміальної регресії, заданої надлишковим описом, що відрізняється від існуючих тим, що органічно поєднує риси класичного методу (теоретично обґрунтовані випадки, в яких оцінка коефіцієнтів при нелінійних членах знаходиться з заданою точністю) з ефективністю евристичних методів (знаходження структури регресії з використанням перевірочної послідовності в модифікованому методі групового урахування аргументів, що входить в склад синтетичного методу), а також включає в себе метод побудови одновимірної поліноміальної регресії на основі довільного повторного активного експерименту з використанням лише одного набору нормованих ортогональних поліномів Форсайта, декомпозиційний метод оцінки коефіцієнтів при нелінійних членах багатовимірної поліноміальної регресії з наперед заданою точністю, що багатовимірну задачу зводить до послідовної побудови відповідних одновимірних поліноміальних регресій;

– вперше обґрунтовано можливість знаходження нормованих ортогональних поліномів Форсайта з наперед заданою точністю, яка досягається за рахунок представлення даних у вигляді раціональних дробів та застосування до них символьних обчислень, що дозволяє отримати оцінки коефіцієнтів при нелінійних членах багатовимірної поліноміальної регресії, значення яких відповідають теоретично виведеним умовам;

– вперше приведено теоретичне обґрунтування зменшення обчислювальної складності програмного забезпечення реалізації методу найменших квадратів на основі повторних експериментів, що полягає в заміні операцій з матрицями повного активного експерименту операціями з матрицями основного експерименту суттєво меншої розмірності;

– вперше запропоновано архітектуру кросплатформної бібліотеки для реалізації синтетичного методу та його складових, яка дозволяє використовувати її

компоненти, як окремо, так і в цілому для розв'язання прикладних задач побудови регресійних моделей.

Практичне значення отриманих результатів. Реалізовано кросплатформну бібліотеку та її програмний інтерфейс для побудови регресійних моделей. Так, для користувачів, які володіють базовими навичками у області програмування і статистичного аналізу та бажають отримати розв’язок задачі побудови багатовимірної поліноміальної регресії достатньо у автоматизованому режимі підключити кросплатформну бібліотеку у власний застосунок та передати у її функції бібліотеки вхідні дані. В цьому випадку функції кросплатформної бібліотеки аналізують надлишковий опис і визначають, що він відноситься до класу, у якому всі лінійні системи мають лише одну змінну, якщо це не так, то задача розв’язується повністю модифікованим методом групового урахування аргументів. У протилежному випадку, програма аналізує можливості декомпозиційного методу та видає вимоги для проведення відповідної кількості повторних активних експериментів для побудови одновимірних поліноміальних регресій та множину коефіцієнтів, які будуть оцінені. Далі формується багатовимірна поліноміальна регресія задана надлишковим описом, яка буде розв’язана модифікованим методом групового урахування аргументів.

Для користувачів, які володіють розширеними навичками у області програмування і статистичного аналізу та бажають отримати розв’язок задачі побудови багатовимірної поліноміальної регресії, повинні ознайомитись з детальною інструкцією по роботі з кросплатформною бібліотекою, у якій описані теоретичні положення та практичні рекомендації з використання синтетичного методу. Далі користувачі за допомогою функцій кросплатформної бібліотеки у частині декомпозиційного методу можуть запрограмувати індивідуальний алгоритм розв’язку задачі на основі теоретичних положень синтетичного методу. У частині модифікованого методу групового урахування аргументів все залишається без змін.

За результатами виконання синтетичного методу користувачу буде виданий кінцевий результат, який містить структуру багатовимірної поліноміальної регресії,

знайдені декомпозиційним методом оцінки коефіцієнтів багатовимірної поліноміальної регресії та їх дисперсії та оцінки коефіцієнтів багатовимірної поліноміальної регресії з оцінками їх дисперсій, знайдені за допомогою модифікованого методу групового урахування аргументів, з оцінкою результатів – має високу ступінь достовірності; задовільну ступінь достовірності; результат недостовірний.

Отримані результати можуть бути використані для підвищення ефективності сучасних інформаційно-діагностичних систем, що використовують регресійні моделі.

Особистий внесок здобувача. Усі результати, що виносяться автором до захисту, отримані особисто у процесі науково-дослідницької роботи. У наукових працях, опублікованих у співавторстві, автору належать:

- у [48] обґрунтування структури алгоритму кластерного аналізу та логіку взаємодії його компонентів;
- у [49] створення критерію вибору шуканої структури багатовимірної поліноміальної регресії на основі використання перевірочної послідовності даних;
- у [50] обґрунтування можливості використання алгоритмів кластерного аналізу у запропонованому методі побудови багатовимірної поліноміальної регресії;
- у [53] результати експериментального дослідження ефективності використання універсальних методів розпаралелювання обчислень в операціях з матрицями у загальній формулі методу найменших квадратів;
- у [14] побудова загальної архітектури програмної бібліотеки, що реалізує метод оцінки коефіцієнтів при нелінійних членах одновимірної поліноміальної регресії з використанням нормованих ортогональних поліномів Форсайта;
- у [47] обґрунтування можливості оцінок коефіцієнтів при нелінійних членах багатовимірної поліноміальної регресії на основі послідовної побудови одновимірних поліноміальних регресій зі спільною областю значень вхідної скалярної змінної та приклад, що ілюструє ефективність запропонованого методу;
- у [54] на основі результатів викладених у [43] обґрунтування можливості оцінок коефіцієнтів при нелінійних членах одновимірної поліноміальної регресії, на основі довільного повторного активного експерименту, з використанням

лише одного набору нормованих ортогональних поліномів Форсайта, що оцінює коефіцієнти віртуальної одновимірної поліноміальної регресії, модифікація загального методу для оцінок коефіцієнтів при нелінійних членах багатовимірної поліноміальної регресії для задачі інтерполяції детермінованого багатовимірного полінома заданого надлишковим описом;

– у [44] створення п’яти часткових випадків надлишкових описів, що приводить до розв’язку систем лінійних рівнянь з однією змінною, участь у виведені верхніх границь оцінок коефіцієнтів при нелінійних членах багатовимірної поліноміальної регресії, а також у дослідженні теоретичних властивостей повторних активних експериментів у загальному методі найменших квадратів;
– у [45] обґрунтування складових підалгоритмів та логіки їх взаємодії в синтетичному методі побудови багатовимірної поліноміальної регресії;
– у [46] розробка та обґрунтування загальної структури програмної бібліотеки, що реалізує синтетичний метод побудови багатовимірної поліноміальної регресії.

Апробація результатів дисертації. Основні результати роботи опубліковано та обговорено на міжнародних та всеукраїнських наукових конференціях, зокрема на: «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2021) (м. Київ, 2021 р.) [14]; The Fifth International Conference on Computer Science, Engineering and Education Applications ICCSEEA 2022 (м. Київ, 2022 р.) [47]; «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2022) (м. Київ, 2022 р.) [53]; The Sixth International Conference on Computer Science, Engineering and Education Applications ICCSEEA 2023 (м. Варшава, 2023 р.) [46]; XIII Міжнародна науково-практична конференція «Комплексне забезпечення якості технологічних процесів та систем» (КЗЯТПС-2023) (м. Чернігів, 2023 р.) [45].

Публікації. За результатами дисертаційних досліджень опубліковано 7 наукових статей, серед яких 2 статті у періодичному науковому виданні, проіндексованому у Scopus, 5 статей у фахових наукових журналах категорії Б.

Структура і обсяг роботи. Дисертаційна робота складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел із 84 найменувань на дев’яти сторінках та чотирьох додатків. Загальний обсяг дисертації становить 185 сторінок, з яких 128 сторінок основного тексту, містить 23 рисунки та 24 таблиці.

1 ПОСТАНОВКА ТА ОБҐРУНТУВАННЯ ПРОБЛЕМИ ДОСЛІДЖЕННЯ

Як показав аналіз існуючої літератури, в сучасних інформаційних діагностичних системах широко використовуються регресійні математичні моделі, так як практично завжди вимірювання супроводжується випадковим спотворенням детермінованої інформації, але ефективність універсальних існуючих методів побудови регресійних моделей не завжди задовольняють практичним вимогам. У даному дисертаційному дослідженні на основі критичного аналізу існуючих методів побудови багатовимірних нелінійних регресій буде сформульовано конкретні вимоги до нового універсального методу побудови регресійних моделей та на основі отриманих теоретичних результатів створення нового програмного забезпечення.

1.1 Види регресійних моделей

Регресійний аналіз – це статистичний метод, який використовується для вивчення взаємозв'язку між залежною змінною (відгуком) та однією або декількома незалежними змінними (пояснювальними факторами) [83]. Задача регресійного аналізу полягає в встановленні взаємозв'язку між залежною та незалежними змінними, щоб оцінити і передбачити значення залежної змінної на основі інших факторів.

Математична модель залежності відгуку об’єкту від деяких факторів:

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

де Y – відгук об’єкта дослідження;

x_i – фактори, незалежні змінні, що характеризують об’єкт;

E – випадкова похибка.

Чорна скринька – метод, що покладено в основу регресійного аналізу. На вхід деякого об’єкта подаються вектори незалежних змінних X (рис. 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, фактором, регресором, факторною ознакою [83].

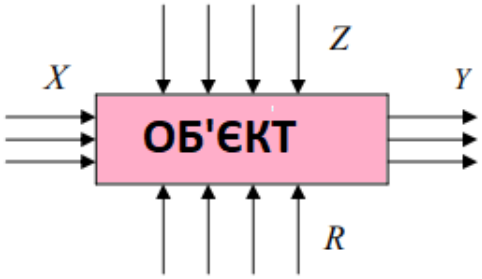

Рисунок 1.1 – Схема «чорної скриньки» регресійного аналізу

Z-фактори – некеровані вхідні змінні, що є незалежними змінними, які можна виміряти, але керувати не можна, а також вектор випадкових факторів R, який впливає на об’єкт дослідження, але його не можливо виміряти.

Фактори, які використовуються в регресійному аналізі, мають задовольняти наступним обмеженням:

– вхідні фактори X повинні впливати на вихідну величину Y;
– бажано, щоб вхідні фактори X були керованими, тобто їх можна задавати по бажанню експериментатора (активний експеримент);
– бажано, щоб фактори X були незалежними між собою;
– точність вимірювання змінної X має бути на декілька порядків більшою за точність вимірювання вихідної величини Y.

Активний експеримент – це експеримент, в якому дослідник може задавати довільні значення вхідних змінних з областей їх означення. Пасивний експеримент – це коли експериментатор пасивно фіксує вхідні значення, що подаються на об’єкт та відповідні значення вихідної змінної.

Надлишковий опис багатовимірної поліноміальної регресії [82, 83, 84] – множина базових поліномів, зважена лінійна згортка невідомої підмножини яких задає багатовимірну поліноміальну регресію.

Залежний вектор Y піддається впливу ряду неконтрольованих або неврахованих факторів, деякій похибці. Аналізуючи зв’язок між однією чи декількома незалежними змінними отримуємо регресію – одновимірну або багатовимірну.

Одновимірна регресія – аналіз зв'язку між однією змінною (регресорами або предикторами) і залежною змінною (відгуком).

Багатовимірна регресія – аналіз зв'язку між декількома незалежними змінними і залежною змінною.

На рис. 1.2 наведено класифікацію регресійних моделей, що найчастіше використовуються на практиці. Далі вони будуть розглянуті більш детально [83].

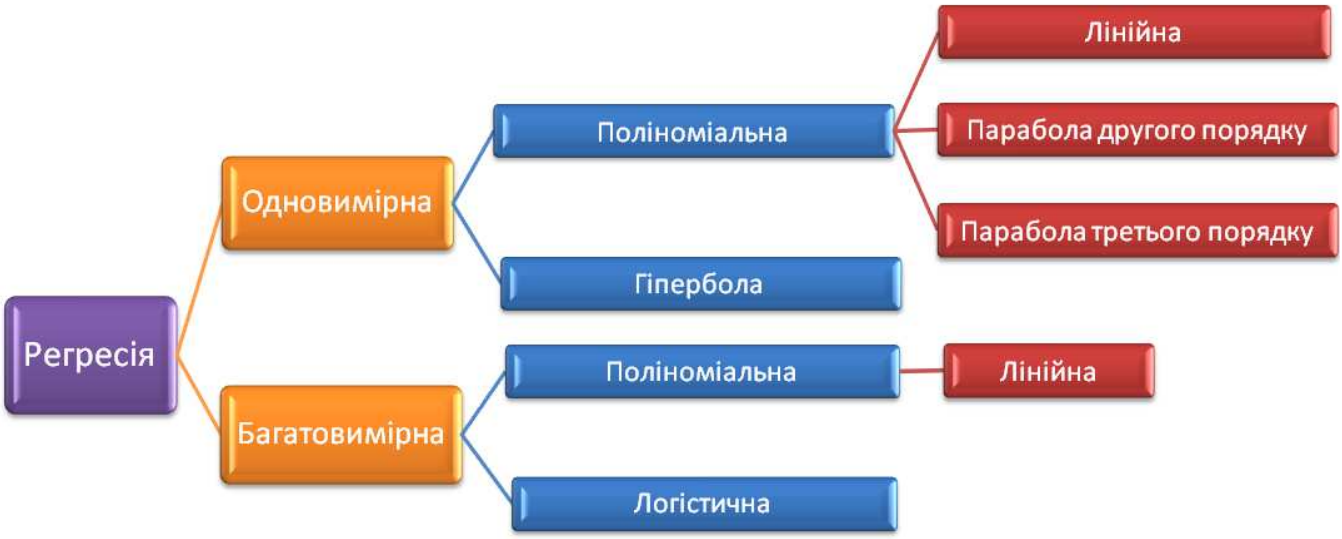

Рисунок 1.2 – Класифікація регресійних моделей

Одновимірні регресійні моделі. Одновимірна лінійна регресія [72]. Графік регресійної моделі при лінійній залежності, апроксимується звичайною лінією. Ми можемо побудувати одновимірну лінійну регресію, яка має вигляд представлений нижче:

$$Y(x) = \theta_0 + \theta_1 x + E \tag{1.2}$$

де x – вхідна скалярна змінна;

$Y(x)$ – вихідні значення;

θ_0 і θ_1 – коефіцієнти, що необхідно оцінити;

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

Параболи [5, 71]. У деяких випадках пряма лінія може занадто погано апроксимувати залежності, в такому випадку потрібно додати до моделі регресії ще один квадратичний член.

$$Y(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + E \tag{1.3}$$

де x – вхідна скалярна змінна;

$Y(x)$ – вихідні значення;

θ_0, θ_1 і θ_2 – коефіцієнти, що необхідно оцінити;

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

В принципі, ми можемо розширити рівняння параболи другого порядку до поліному вищого порядку, додавши члени $x^2, x^3 \ldots x^n$ з їх коефіцієнтами. Після чого отримаємо поліноміальну регресію [54].

Гіпербола [5]. Рівняння гіперболи має наступний вигляд:

$$Y(x) = \theta_0 + \theta_1 \cdot \frac{1}{x} + E \tag{1.4}$$

де x – вхідна скалярна змінна;

$Y(x)$ – вихідні значення;

θ_0 і θ_1 – коефіцієнти, що необхідно оцінити;

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

Одновимірна поліноміальна регресія [22, 54]. Вигляд моделі одновимірної поліноміальної регресії представлено нижче:

$$Y(x) = \theta_0 + \theta_1 x + \ \ldots + \theta_r x^r + E \tag{1.5}$$

де x – вхідна скалярна змінна;

$Y(x)$ – вихідні значення;

$\theta_0 .. \theta_r$ – коефіцієнти, що необхідно оцінити;

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

Багатовимірний (множинний) регресійний аналіз [6, 47, 48]. У багатьох практичних задачах результат залежить не від однієї вхідної змінної, а від декількох. У таких випадках використовується багатовимірний регресійний аналіз.

Нижче наведено приклад багатовимірної лінійної регресії:

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

де b_0 – вільний коефіцієнт;

$b_1, b_2, \ldots, b_n$ – коефіцієнти багатовимірної регресії;

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

m – кількість змінних;

E – випадкова величина, яка має довільний розподіл; $ME = 0$, $DE = \sigma^2 < \infty$, відома величина дисперсії або її верхня оцінка.

В багатьох інформаційних експертних системах чи в інформаційних системах класифікації використовуються багатовимірні поліноміальні (степеневі) регресії.

Модель багатовимірної поліноміальної регресії має наступний вигляд:

$$Y(\bar{x}) = \sum b_{i_1 \ldots i_t}^{j_1 \ldots j_t} (x_{i_1})^{j_1} \cdot (x_{i_2})^{j_2} \cdot \ldots \cdot (x_{i_t})^{j_t} + E$$
$$\forall (i_1 \ldots i_t) \in K, \forall (j_1 \ldots j_t) \in K(i_1 \ldots i_t), \tag{1.7}$$

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

E – випадкова величина, яка має довільний розподіл; $ME = 0$, $DE = \sigma^2 < \infty$, відома величина дисперсії або її верхня оцінка.

У задачах класифікації, де важливо визначити, до якої категорії відноситься спостереження часто застосовують логістичну регресію [2], яка 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;

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

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

m – кількість змінних;

E – випадкова величина, яка має довільний розподіл; $ME = 0$, $DE = \sigma^2 < \infty$, відома величина дисперсії або її верхня оцінка.

Примітка 1.1. Як видно з приведеного аналізу у п. 1.1, у першому розділі розглядаються регресійні моделі, в аналітичний вираз яких невідомі коефіцієнти входять лінійно. Саме такий клас регресійних моделей є предметом дослідження даної дисертаційної роботи.

1.2 Критичний аналіз методів побудови багатовимірних нелінійних регресій

Далі будуть розглянуті класичні універсальні методи статистичного аналізу, за допомогою яких можна будувати одновимірні поліноміальні регресії, багатовимірні лінійні регресії та багатовимірні поліноміальні регресії, а значимість членів, що входять у ці регресії можна оцінювати з використанням наступних критеріїв [1, 16, 26, 29, 31, 32, 34, 51, 52, 66, 69, 83]:

– сума квадратів залишків;
– коефіцієнт детермінації R^2;
– статистика Малоуза;
– F-статистика;
– регресійна сума квадратів;
– залишкова сума квадратів (ЗСК);
– t-впорядкований пошук;
– виключення та покрокова регресія.

Крім того, буде розглянуто ряд евристичних підходів та методів побудови регресійних моделей.

1.2.1 Класичний метод найменших квадратів

Метод найменших квадратів – це статистичний метод, який використовується в регресійному аналізі для побудови моделей, які найкращим чином показують залежність між вхідними незалежними змінними та результатами експерименту (вихідною залежною змінною) [51, 52, 83].

Основний принцип методу найменших квадратів полягає в тому, що він знаходить такі значення параметрів моделі (коефіцієнти регресії), які зменшують розбіжність між фактичними значеннями залежної змінної та прогнозованими значеннями, вираженими у вигляді комбінації незалежних змінних.

Метод найменших квадратів може бути застосований для побудови поліноміальної регресійної моделі (1.5), багатовимірної лінійної регресійної моделі (1.6) та багатовимірної поліноміальної регресійної моделі (1.7). Для побудови БПР за допомогою методу найменших квадратів (МНК), її необхідно звести до послідовності ОПР та розв’язати відповідні системи лінійних рівнянь.

Примітка 1.2. Під побудовою регресії мається на увазі знаходження оцінок невідомих коефіцієнтів, що входять в аналітичний вираз регресійної моделі.

Якісне пояснення суті МНК показано на рис. 1.3.

1.2.2 Метод максимальної правдоподібності

Метод максимальної правдоподібності (ММП) – це статистичний метод, що використовується для оцінювання параметрів моделі шляхом максимізації ймовірності спостережень при заданому наборі параметрів. Цей метод є фундаментальним підходом у математичній статистиці для отримання параметрів моделі на основі вибірки даних [31, 32, 34].

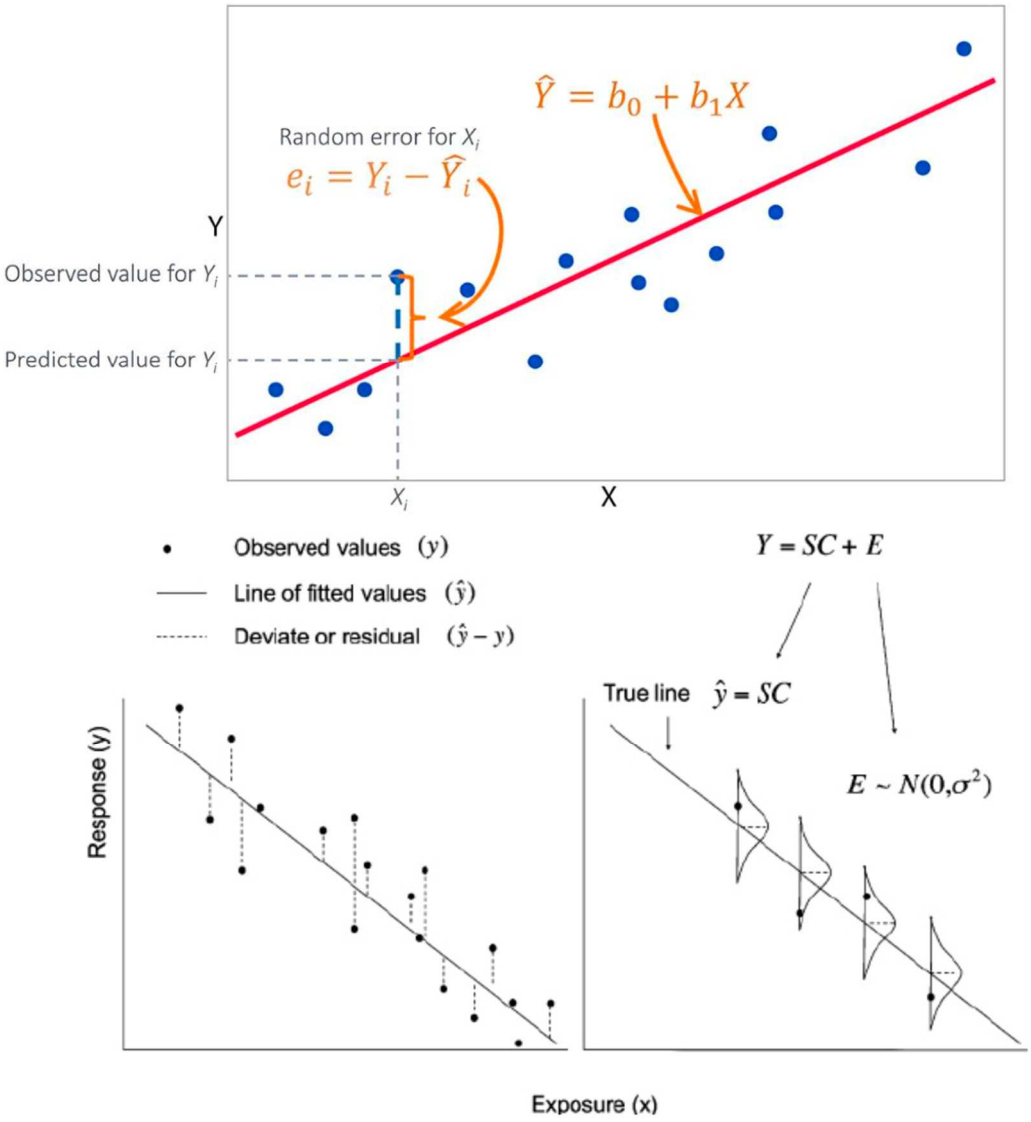

Рисунок 1.3 – Демонстрація МНК [51]

Основна ідея ММП полягає у знаходженні таких значень параметрів, які максимізують ймовірність появи отриманих результатів експерименту. У контексті регресійного аналізу ММП використовується для знаходження оцінок коефіцієнтів регресійної моделі, яка найкраще відображає залежність між вхідними незалежними змінними та вихідною змінною [31, 32, 34]. Наприклад, для простої лінійної регресії, ММП знаходить ті значення нахилу та зміщення лінії, які максимізують ймовірність спостережень відповідно до моделі.

Процес використання ММП включає такі кроки.

Визначення функції правдоподібності: спочатку визначається функція правдоподібності, яка виражає ймовірність спостережень за певних параметрів моделі.

Логарифмування функції правдоподібності: часто для спрощення обчислень функція правдоподібності логарифмується, що дозволяє перейти від множення до додавання.

Максимізація логарифмованої функції правдоподібності: знаходяться значення параметрів, які максимізують логарифмовану функцію правдоподібності.

Оцінка параметрів: знайдені значення параметрів стають оцінками параметрів моделі.

Застосування ММП буде показано на рис. 1.4.

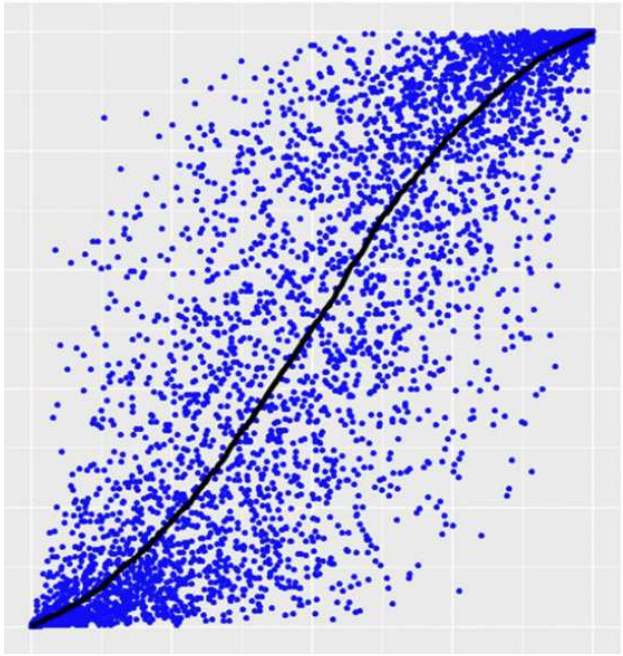

Рисунок 1.4 – Демонстрація ММП для задачі ОПР [76]

1.2.3 Факторний аналіз

Факторний аналіз – це метод статистичного аналізу, який використовується для вивчення взаємозв'язків між змінними і виявлення прихованих структур або факторів. Цей метод широко використовується в психології, соціології, маркетингу, економіці та інших галузях для відображення складних взаємозв'язків між багатьма змінними. Даний метод статистичного аналізу підходить для будь-якої задачі побудови регресії, у тому числі ОПР, БЛР та БПР [1, 29, 69].

Основні концепції факторного аналізу наступні.

Фактори – це приховані змінні, які впливають на змінні, що спостерігаються. Фактори можуть вказувати на системи або поняття, які впливають на дані.

У факторному аналізі зазвичай використовується матриця даних, де кожний рядок відповідає одному спостереженню, а кожний стовпчик – одній змінній.

Факторний аналіз базується на матричних розкладах, таких як метод головних компонент або факторний аналіз за власними числами.

Процес факторного аналізу включає ряд кроків.

Крок 1. Збір та організація даних у вигляді матриці.

Крок 2. Знаходження по матриці експериментальних даних прихованих факторів.

Крок 3. Опис та інтерпретація факторів, що були виявлені. Це може включати призначення імен або понять для факторів, що пояснюють варіабельність у даних.

Крок 4. Використання отриманих факторів для аналізу змінних, побудови моделей або прийняття рішень.

Факторний аналіз допомагає розкрити складні взаємозв'язки між змінними і зрозуміти, які фактори можуть пояснити залежності, що спостерігаються.

1.2.4 Множинний дискримінантний аналіз

Множинний дискримінантний аналіз (МДА) – це метод статистичного аналізу, який використовується для розділення спостережень у дві або більше груп (класів) на основі значень декількох змінних (показників), таким чином, щоб міжгрупова варіація була максимальною, а внутрішньогрупова варіація – мінімальною. МДА є популярним методом у багатьох галузях, включаючи математичну статистику, машинне навчання, біологію, медицину та інші [16, 26, 66].

Основна мета МДА – знайти комбінацію змінних, таку щоб вона найкраще розділяла класи спостережень. Ця комбінація називається дискримінантною функцією. Прикладом може бути розділення пацієнтів на дві групи (наприклад, хворі та здорові) на основі декількох біомедичних показників.

З точки зору регресійного аналізу МДА найкраще підходить для побудови логістичних регресійних моделей, оскільки в них виконується розділення спостережень на деякі групи. Для інших видів регресій МДА застосовувати достатньо складно [26].

Розглянемо модель аналізу даних, яка комбінує особливості множинної регресії та множинного дискримінантного аналізу [66]. Схоже до множинної регресії, цей метод використовує одну або декілька незалежних змінних для прогнозування залежної змінної. Проте, відмінність полягає в тому, що залежна змінна в логістичній регресії має неметричний характер, аналогічно до множинного дискримінантного аналізу.

Ця неметрична природа залежної змінної призводить до відмінностей у методах оцінки та припущеннях щодо типу розподілу, хоча багато інших аспектів схожі на множинну регресію.

Розглянемо невеликий приклад. Фінансові консультанти прагнули створити метод відбору підприємств, які мають потенціал для росту та є привабливими для початкових інвестицій. З метою вирішення цієї задачі, вони проаналізували історичні записи та розподілили фірми на дві категорії: ті, які досягли успіху протягом п'яти років, і ті, які були менш успішними протягом цього періоду. Для кожного підприємства вони мали великий обсяг фінансових та управлінських даних. Далі, вони використали модель логістичної регресії для ідентифікації тих фінансових та управлінських показників, які найкраще відрізняли успішні підприємства від менш успішних, з метою вибору найперспективніших кандидатів для майбутніх інвестицій. Розв’язання цієї задачі продемонстровано на рис. 1.5.

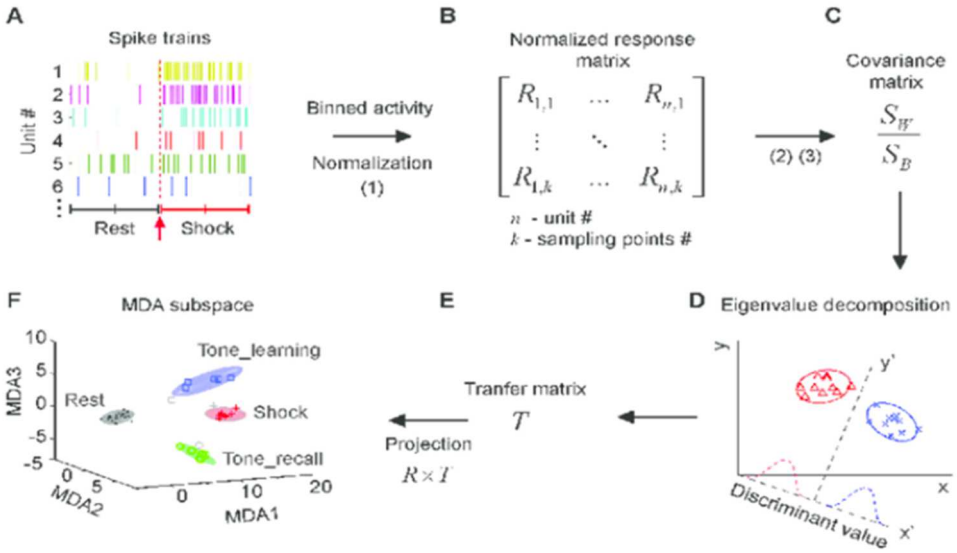

Рисунок 1.5 – Використання МДА та логістичної регресії [66]

1.2.5 Метод групового урахування аргументів

Метод групового урахування аргументів є підходом в аналізі даних та математичній статистиці, який використовується здебільшого в регресійному моделюванні та аналізі. Цей метод включає змінні в модель групами або підгрупами замість окремих змінних, що дозволяє краще ураховувати взаємозв'язки між вхідними змінними та вихідною змінною, і як результат зменшити обчислювальну складність у випадку великої кількості даних [13, 25, 38].

Основна ідея МГУА полягає:

– в генерації з використанням багаторівневого селекційного алгоритму генерації часткових описів шуканої регресії;
– в знаходженні структури шуканої регресії (відповідного часткового опису) з використанням експериментальних даних, що не використовувались для оцінки коефіцієнтів часткових описів.

МГУА складається з таких кроків.

Крок 1. Змінні об'єднуються в підгрупи або групи на підставі їх схожості, тематичного спрямування або іншого обґрунтованого критерію.

Крок 2. Для кожної групи змінних будується окрема регресійна модель, в якій використовуються тільки змінні з даної групи.

Крок 3. За введеними критеріями визначається множина часткових регресійних моделей, що породжують часткові регресійні моделі наступного рівня селекції.

Крок 4. Побудова часткових описів регресії на наступному рівні селекційного алгоритму шляхом заданого правила створення комбінацій часткових описів попереднього рівня селекційного алгоритму.

Крок 5. На кожному рівні селекції, за допомогою перевірочної послідовності, знаходиться частковий опис з мінімальною ЗСК.

Крок 6. Розв'язком є частковий опис регресійної моделі з мінімальною ЗСК, знайденою за перевірочною послідовністю, якщо відповідна ЗСК часткового опису на наступному рівні селекції є більшою, ніж на попередньому рівні.

Цей метод може бути корисним у випадках, коли існують обґрунтовані підстави для групування змінних і припущення, що залежність між змінними відрізняється в різних групах.

Багаторівневий селекційний алгоритм для автоматичної генерації часткових описів імітує процес біологічного відбору, враховуючи попарно послідовні ознаки. Цей підхід вже широко використовується в глибинному навчанні. Для порівняння та вибору оптимальних груп використовуються дві або більше підмножини вибірки даних.

Під час розробки МГУА, головною метою було встановлення органічної аналогії між двома задачами: побудовою моделей для зашумлених даних і передачею сигналу через канал з перешкодами [25]. Цей підхід полегшив створення теоретичної основи для стійкого моделювання. Основний результат, отриманий із цієї теорії, полягає в тому, що складність оптимальної моделі, що прогнозується, залежить від рівня невизначеності в досліджуваних даних: чим вищий цей рівень, тим простіше повинна бути оптимальна модель з меншою кількістю параметрів.

Основний недолік багаторівневого селекційного алгоритму полягає в тому, що множина часткових описів проміжного рівня селекції, яка не використовується для створення часткових описів наступного рівня селекції, може містити складову, без якої згенеровані множини часткових описів наступних рівнів селекційного алгоритму не містять шуканого розв’язку.

1.2.6 Використання штучних нейронних мереж

Застосування штучних нейронних мереж (ШНМ) в регресійному аналізі є популярним і потужним підходом для моделювання залежностей між вхідними і вихідними змінними. ШНМ є складними математичними структурами, і їхні можливості виявляються в здатності навчатися і використовувати не лінійні зв'язки між змінними, а також виявляти внутрішні залежності в даних [42, 62, 78].

Можна виділити такі етапи використання ШНМ для регресійного аналізу.

Етап 1. Вибір архітектури нейронної мережі (наприклад, прямого поширення, рекурентної, зворотного поширення тощо) та кількості шарів і нейронів у кожному шарі.

Етап 2. Визначення складових нейронної мережі, включаючи вхідні, приховані і вихідні шари. Кожен штучний нейрон в прихованому шарі виконує операцію лінійної комбінації вхідних даних з певними вагами, після чого застосовується нелінійна функція активації.

Етап 3. Використання навчальних даних для налаштування ваг і параметрів мережі. Цей процес включає в себе знаходження оптимальних ваг штучних нейронів шляхом мінімізації функції втрат.

Етап 4. Перевірка продуктивності моделі на навчальних, валідаційних і тестових даних для оцінки її точності та здатності до узагальнення.

Переваги використання ШНМ у регресійному аналізі включають здатність моделювати складні та не лінійні залежності між змінними, а також можливість працювати з великими обсягами даних. Однак у використанні ШНМ при побудові різного роду регресій є ряд недоліків, пов’язаних із швидкістю навчання, необхідною великою кількістю даних і відсутністю теоретичного обґрунтування отриманих результатів.

Загалом, ШНМ можуть бути потужним інструментом для вирішення задач побудови регресій, особливо у складних та нелінійних залежностях.

1.2.7 Генетичний алгоритм

Генетичний алгоритм (ГА) може бути застосований для вирішення лінійних та нелінійних задач регресії. Основна ідея генетичного алгоритму полягає в емуляції процесів природного відбору та еволюції для пошуку оптимального рішення. В контексті регресійного аналізу, метою є знаходження оптимальної структури та параметрів моделі, які найкраще апроксимують дані [9, 28, 35].

Основні кроки використання генетичного алгоритму для лінійної та нелінійної регресії.

Крок 1. Потрібно визначити цільову функцію, яка оцінює якість апроксимації моделі до даних. Ця функція повинна враховувати відхилення між спостереженнями і передбаченими значеннями моделі.

Крок 2. Потрібно представити параметри моделі (наприклад, коефіцієнти нелінійної функції) у вигляді генетичних рядків (хромосом), які є потенційними рішеннями.

Крок 3. Створити початкову популяцію генетичних рядків (хромосом) за допомогою випадкового вибору або іншим методом.

Цикл генетичного алгоритму.

Крок 3.1. Відбір. Обираються батьківські особини для подальшого схрещування та створення нащадків на основі їх цільової функції. Особини з кращими результатами мають більше шансів бути обраними.

Крок 3.2. Схрещування (кросинговер). Виконується операція схрещування між батьківськими особинами для створення нових нащадків. Це може бути одноточковий, двоточковий, або інший вид схрещування.

Крок 3.3. Мутація. З малою ймовірністю змінюються деякі гени в генетичних рядках, для введення різноманітності та уникнення локальних розв’язків.

Крок 3.4. Оцінка цільової функції. Розрахувати значення цільової функції для нових нащадків.

Крок 3.5. Оновлення популяції. Обираються нащадки, які входять у популяцію та особини, які видаляються з популяції, враховуючи їхні результати та, можливо, розмір популяції.

Крок 4. Умова завершення. Повторити цикл генетичного алгоритму доки не буде досягнуто заданої кількості ітерацій або поки не буде досягнуто задовільного рівня апроксимації.

Крок 5. Виведення результату. Визначити найкращу знайдену особину (рішення), яка відповідає найкращому значенню цільової функції. Це може бути оптимальними параметрами регресійної моделі.

Важливо налаштувати параметри генетичного алгоритму: розмір популяції, ймовірність схрещування та мутації, локальне покращення, а також критерій зупинки, відповідно до конкретної задачі регресії.

До недоліків ГА можна віднести відсутність теоретичного обґрунтування точності отриманих результатів та, на відміну від МГУА, ГА не використовує перевірочну послідовність, що має суттєве значення при недостатньо великому обсязі експериментальних даних.

1.3 Порівняльний аналіз існуючих пакетів прикладних програм

Розглянемо ряд програмних засобів, які частково чи повністю розв'язують задачу знаходження оцінок коефіцієнтів БПР (програмні засоби можуть окремо будувати ОПР та БЛР). Програмні засоби буде розділено на дві групи: програмні пакети та програмні бібліотеки.

1.3.1 Програмні пакети

STATISTICA [74] – пакет для всестороннього статистичного аналізу, розроблений компанією StatSoft. Програма має різні модулі та процедури реалізовані для аналізу, управління, добування та візуалізації даних. Відповідні модулі називаються data analysis, data management, data mining та data visualization. Остання версія програмного пакету 12.6 для ОС Windows.

У програмному пакеті STATISTICA присутня можливість будувати усі відомі види регресії, у тому числі ОПР, БЛР та БПР. Для прикладу буде розглянуто, як у програмному пакеті STATISTICA виконується багатовимірний поліноміальний регресійний аналіз.

Багатовимірна поліноміальна регресія в програмі STATISTICA здійснюється за допомогою команди Polynomial Regression.

Заповнена таблиця вхідних даних для регресійного аналізу пакету STATISTICA має вигляд, показаний на рис. 1.6, а вікно налаштувань відповідно на рис. 1.7.

	1 y	2 x1	3 x2	4 x3	5 x4
1	-7,0E+12	-50	-50	-50	-50
2	-1,2E+12	-38,888	-38,888	-38,888	-38,888
3	-1,1E+11	-27,778	-27,778	-27,778	-27,778
4	-3,21E+9	-16,666	-16,666	-16,666	-16,666
5	-1,46E+6	-5,555	-5,555	-5,555	-5,555
6	1480189	5,555	5,555	5,555	5,555
7	3,215E+9	16,666	16,666	16,666	16,666
8	1,15E+11	27,778	27,778	27,778	27,778
9	1,21E+12	38,888	38,888	38,888	38,888
10	7,03E+12	50	50	50	50

Рисунок 1.6 – Вхідні дані БПР

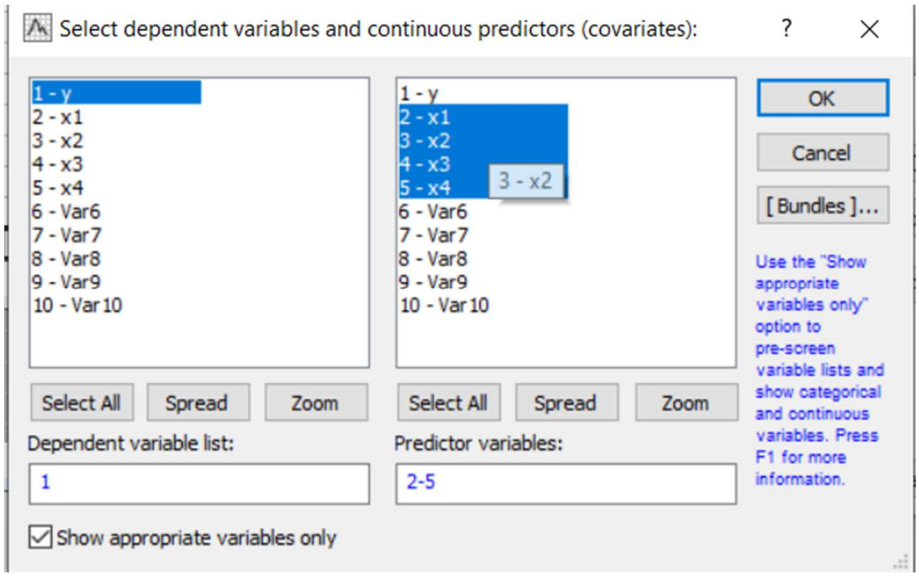

Рисунок 1.7 – Налаштування для побудови БПР у пакеті STATISTICA

Результати виконання багатовимірного поліноміального регресійного аналізу у пакеті STATISTICA буде відображено у таблиці Parameter Estimates (рис. 1.8). Як видно з рис. 1.8, пакет STATISTICA самостійно визначає структуру БПР, однак ми не можемо самостійно задати надлишковий опис, а для визначення значущості коефіцієнтів застосовується ряд критеріїв.

Effect	Parameter Estimates (Spreadsheet39) (*Zeroed predictors failed tolerance check) Sigma-restricted parameterization Comment (B/Z/P)	y Param.	y Std.Err	y t	y p	-95,00% Cnf.Lmt	+95,00% Cnf.Lmt	y Beta (?)	y St.Err.?	-95,00% Cnf.Lmt	+95,00% Cnf.Lmt
Intercept		-5,991884E+06	1,120646E+12	-0,000005	0,999996	-2,649913E+12	2,649901E+12				
x1		7,891579E+10	2,320930E+10	3,400179	0,011440	2,403450E+10	1,337971E+11	0,789221	0,232112	0,240364	1,338077
x1^2		2,048639E+04	8,256815E+08	0,000025	0,999981	-1,952406E+09	1,952447E+09	0,000006	0,232112	-0,548851	0,548862
x2	Zeroed*	0,000000E-01									
x2^2	Zeroed*	0,000000E-01									
x3	Zeroed*	0,000000E-01									
x3^2	Zeroed*	0,000000E-01									
x4	Zeroed*	0,000000E-01									
x4^2	Zeroed*	0,000000E-01									

Рисунок 1.8 – Результати побудови БПР

SPSS Statistics [73] – це аналітичне програмне забезпечення, призначене для статичного аналізу даних. Розширені статистичні процедури та візуалізація, які присутні у даному програмному забезпеченні, використовуються для розуміння даних, вирішення складних задач, проблем, галузевих бізнес-завдань та прийняття якісних рішень. Остання версія програмного пакету 28.0.1.

У програмному пакеті SPSS Statistics присутня можливість будувати усі відомі види регресії, однак пакет SPSS Statistics у першу чергу робить акцент на значущості членів регресійної моделі. Для прикладу буде розглянуто, як у програмному пакеті SPSS Statistics виконується багатовимірний лінійний регресійний аналіз.

Буде проаналізовано залежність здатності вирішувати конфлікти від кількох факторів особи. Для аналізу внесених даних у програму (рис. 1.9) у вкладці «Аналіз» обирається регресія з видом «Лінійна».

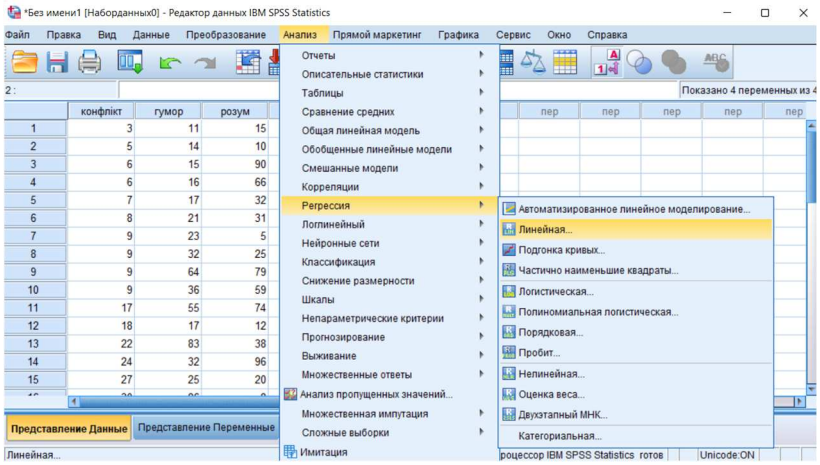

Рисунок 1.9 – Вхідні дані регресійного аналізу у пакеті SPSS Statistics

Предметом аналізу стануть дані: краса, гумор, розум, тобто використовується кілька незалежних змінних. Налаштування параметрів багатовимірної лінійної регресійної моделі показано на рис. 1.10.

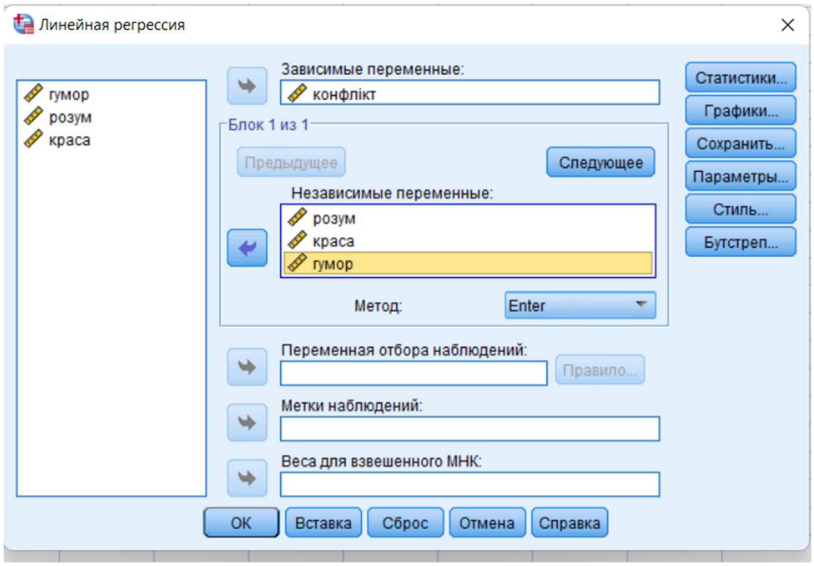

Рисунок 1.10 – Налаштування параметрів БЛР у пакеті SPSS Statistics

Після чого можна виконати аналіз з заданими налаштуванням. На рис. 1.11 представлено результати багатовимірного лінійного регресійного аналізу.

Сводка для модели[b]

Модель	R	R-квадрат	Скорректированный R-квадрат	Стандартная ошибка оценки	Статистика изменений				
					Изменение R квадрат	Изменение F	ст.св.1	ст.св.2	Знач. Изменение F
1	,510[a]	,260	,175	8,292	,260	3,050	3	26	,046

a. Предикторы: (константа), гумор, краса, розум

b. Зависимая переменная: конфлікт

ANOVA[a]

Модель		Сумма квадратов	ст.св.	Средний квадрат	F	Знач.
1	Регрессия	629,159	3	209,720	3,050	,046[b]
	Остаток	1787,541	26	68,752		
	Всего	2416,700	29			

a. Зависимая переменная: конфлікт

b. Предикторы: (константа), гумор, краса, розум

Коэффициенты[a]

Модель		Нестандартизованные коэффициенты		Стандартизованные коэффициенты	т	Знач.	Корреляции		
		B	Стандартная Ошибка	Бета			Нулевого порядка	Частично	Компонент
1	(Константа)	19,920	4,835		4,120	,000			
	розум	-,101	,052	-,356	-1,949	,062	-,164	-,357	-,329
	краса	-,113	,055	-,368	-2,052	,050	-,249	-,373	-,346
	гумор	,146	,069	,367	2,131	,043	,301	,386	,359

a. Зависимая переменная: конфлікт

Рисунок 1.11 – Результати побудови БЛР

В таблиці коефіцієнтів, представленій вище (рис. 1.11) можна побачити коефіцієнти *b*, що вказують, наскільки змінюється залежна змінна при збільшенні на один даної незалежної змінної.

Як і пакет STATISTICA, SPSS Statistics самостійно визначає структуру регресії і керується критеріями для визначення значущості коефіцієнтів.

Deductor [12] – програмний пакет, що надає змогу створювати прикладні аналітичні рішення. Рішення призначені для акумуляції структурованої інформації з різних джерел для поглибленої аналітики з можливістю візуалізації та створення звітів. Є ефективним при використанні професіоналами у сфері аналітики у середніх та великих організаціях, а також для науковців. Остання версія програмного пакету 5.3 (не підтримується з 2015 року) для ОС Windows.

Програмний пакет пропонує велику кількість алгоритмів, серед яких є набір, який називається Data mining, до якого, зокрема, входять методи регресійного аналізу. Усього представлено три види регресії: одновимірна лінійна, багатовимірна лінійна та логістична регресії. Варто зазначити, що ні одновимірна, ні багатовимірна поліноміальна регресія не представлена в цьому пакеті, але їх можна реалізувати, додавши відповідні дані нелінійної складової, і виконувати аналогічні операції, як для побудови лінійної регресії.

Для прикладу буде розглянуто, як у програмному пакеті Deductor виконується багатовимірний лінійний регресійний аналіз.

На рис. 1.12 наведено набір вхідних даних залежності оцінки студента від кількості проведених за підготовкою годин та кількості підготовчих іспитів.

Після внесення вхідних даних потрібно зробити ряд налаштувань, варто зупинитись на налаштуваннях значень стовпців. Тут треба встановити стовпці годин та підготовчих іспитів вхідними, а оцінку – вихідними даними (рис. 1.13).

Після завершення процесу побудови багатовимірної лінійної регресійної моделі результат можна буде побачити на рис. 1.14–1.15.

	hours	prep_exams	score
▶	1	1	76
	2	3	78
	2	3	85
	4	5	88
	2	2	72
	1	2	69
	5	1	94
	4	1	94
	2	0	88
	4	3	92
	4	4	90
	3	3	75
	6	2	96
	5	4	90
	3	4	82
	4	4	85
	6	5	99
	2	1	83
	1	0	62
	2	1	76

Рисунок 1.12 – Вхідні дані регресійного аналізу у пакеті Deductor

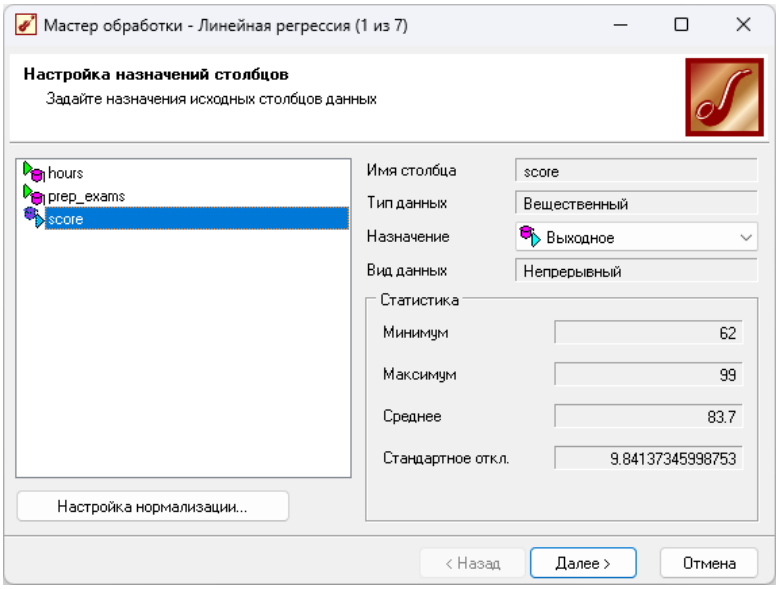

Рисунок 1.13 – Налаштування параметрів БЛР у пакеті Deductor

Атрибут	Коэффициент
9.0 <Константа>	68.40770528
9.0 hours	5.400441043
9.0 prep_exams	-0.5781554028

Рисунок 1.14 – Коефіцієнти регресії

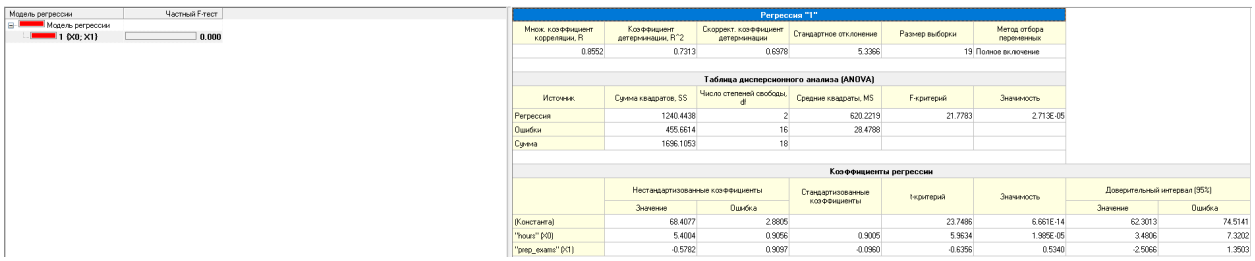

Рисунок 1.15 – Деталізований звіт з регресійного аналізу

1.3.2 Програмні бібліотеки

Scikit-learn [68] – це бібліотека для мови програмування Python, яка надає різноманітні інструменти для машинного навчання, включаючи методи класифікації, регресії, кластеризації, виявлення аномалій, обробки текстових даних, вимірювання якості моделей та багато іншого. Scikit-learn покликана спростити процес розробки і випробування моделей машинного навчання, надаючи зручний інтерфейс для роботи з різними алгоритмами. Scikit-learn надає широкий спектр інструментів для регресійного аналізу (включає ОПР, БЛР, БПР), що дозволяє, як і побудову структури регресійної моделі так і оцінку її коефіцієнтів за допомогою різного роду алгоритмів. Крім того бібліотека містить набір алгоритмів для оцінки якості отриманих результатів. Остання версія 1.3.0.

Розглянемо декілька прикладів роботи бібліотеки scikit-learn для побудови основних видів регресійних моделей.

Одномірний лінійний регресійний аналіз за допомогою бібліотеки scikit-learn може бути виконаний з використанням об’єкта лінійної регресії LinearRegression, в якому використовуються динамічна бібліотека LIBLINEAR та сучасні алгоритми оптимізації, які краще працюють з не стандартизованими змінними. Фрагмент коду, який виконує одномірний лінійний регресійний аналіз з використанням бібліотеки scikit-learn представлено нижче:

```
from sklearn.linear_model import LinearRegression
slr = LinearRegression()
slr.fit(X, y)
```

Також є багато різних моделей регресійного аналізу у бібліотеці scikit-learn. Сюди відноситься модель гребеневої регресії, яку можна ініціалізувати таким чином:

```
from sklearn.linear_model import Ridge
ridge = Ridge(alpha=1.0)
```

Схожим чином можна ініціалізувати lasso-регресор із підмодуля linear_model:

```
from sklearn.linear_model import Lasso
lasso = Lasso(alpha=1.0)
```

Для багатовимірного регресійного аналізу у бібліотеки scikit-learn все майже так само, тільки тепер x – це набір пояснювальних змінних.

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X, y)
```

Тоді коефіцієнтами пояснювальної змінної будуть lr.coef_[i].

Одновимірний поліноміальний регресійний аналіз можна виконати за допомогою бібліотеки scikit-learn, використовуючи наступний код, наведений нижче як приклад.

```
from sklearn.preprocessing import PolynomialFeatures
pr = LinearRegression()
quadratic = PolynomialFeatures(degree=2)
X_quad = quadratic.fit_transform(X)
pr.fit(X_quad, y)
y_quad_fit = pr.predict(X_quad)
```

Тут y_quad_fit – отримана наша поліноміальна регресія 2-го порядку на наборі X_fit. Теж саме справедливо для будь-якого порядку, потрібно тільки замінити параметр degree у PolynomialFeatures на потрібний.

Ще один приклад застосування бібліотеки для поліноміального регресійного аналізу – регресія на основі дерева рішень. Перевага алгоритму дерева рішень полягає в тому, що він не потребує перетворення ознак, у разі якщо ми маємо справу з нелінійними даними.

```
from sklearn.tree import DecisionTreeRegressor
tree = DecisionTreeRegressor(max_depth=3)
tree.fit(X, y)
```

Для багатовимірного поліноміального регресійного аналізу у бібліотеці scikit-learn все майже так само, тільки тепер x – це набір вхідних незалежних змінних [63].

Розглянемо на прикладі PolynomialFeatures з sklearn.preprocessing для поліному третього порядку та двох незалежних змінних.

Оскільки в нас x складається з двох незалежних змінних, то спочатку, побудуємо поліноміальну регресійну модель 3-го порядку, для прикладу:

```
lr = LinearRegression()
poly = PolynomialFeatures(degree)
model = make_pipeline(poly, lr)
model.fit(X, y)
```

Тоді коефіцієнтами отриманого рівняння будуть знаходитись у lr.coef.

Теж саме можна зробити за допомогою коду, наведеного нижче.

```
pr = LinearRegression()
cubic = PolynomialFeatures(degree=3, include_bias=True)
X_cub = cubic.fit_transform(x)
pr.fit(cubic, y)
y_cub_fit = pr.predict(X_cub)
```

Тут X_cub дорівнює:

```
array([[1., 1., 1., 1., 2., 1., 1., 3., 3., 1.]])
```

Даний масив являє собою коефіцієнти для членів, наведених нижче.

$$[1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_2x_1^2, x_2^2x_1, x_2^3]$$

Як було сказано вище, це – поліном третього порядку для двох незалежних змінних – x_1 та x_2.

Аналогічно, багатовимірний поліноміальний регресійний аналіз можна зробити за допомогою DecisionTreeRegressor з модуля sklearn.tree:

```
from sklearn.tree import DecisionTreeRegressor
tree = DecisionTreeRegressor(max_depth=2)
tree.fit(X, y)
```

Тепер виконаємо регресійний аналіз:

```
predict = regressor.predict(np.array([...]).T)
```

Незважаючи на те, що бібліотека scikit-learn дозволяє будувати основний набір регресійних моделей, в аналітичний вираз яких невідомі коефіцієнти входять лінійно, тут відсутня можливість задавати надлишковий опис та гнучко керувати процесом побудови регресій.

NumPy [40] – це основна бібліотека мови програмування Python для обчислень наукового та технічного характеру. Вона надає підтримку для багатовимірних масивів та матриць, а також велику колекцію математичних функцій для виконання операцій на масивах. NumPy є основною складовою для багатьох інших бібліотек у сфері обробки даних, наукових обчислень та машинного навчання. Остання версія 1.25.

Бібліотека NumPy не є прямим інструментом для регресійного аналізу, але вона надає основні інструменти для обробки даних та математичних обчислень, які можуть бути використані для регресійного аналізу. Використовуючи дану бібліотеку користувач може побудувати будь-яку регресійну модель послідовно викликаючи математичні функції необхідні для реалізації того чи іншого методу побудови регресії. Більш з тим використання подібних бібліотек є досить незручним, оскільки виникає необхідність кожного разу, фактично, писати програму з нуля.

R-Stats [79] – статистичний пакет та функціонал мови програмування та середовища аналізу даних R, який використовується для виконання статистичних обчислень, моделювання та аналізу даних. У R-Stats існує велика кількість пакетів, які розширюють можливості R у сфері статистичного аналізу. Остання версія 3.5.0.

У R-Stats входять:

– велика кількість вбудованих статистичних функцій для обчислення базових статистичних метрик, таких як середнє, медіана, дисперсія, кореляція, регресійний аналіз та багато іншого;

– велика кількість статистичних тестів, таких як t-тест, аналіз дисперсії (ANOVA), непараметричні тести та інші, для проведення перевірки статистичних гіпотез;
– пакети візуалізації даних, такі як «ggplot2», який дозволяє створювати стилізовані та інформативні графіки (рис. 1.16) [17];
– спеціалізовані пакети для виконання конкретних статистичних аналізів, таких як «survival» для аналізу ступеню виживання, «tm» для обробки текстових даних, «forecast» для прогнозування часових рядів і т.д.

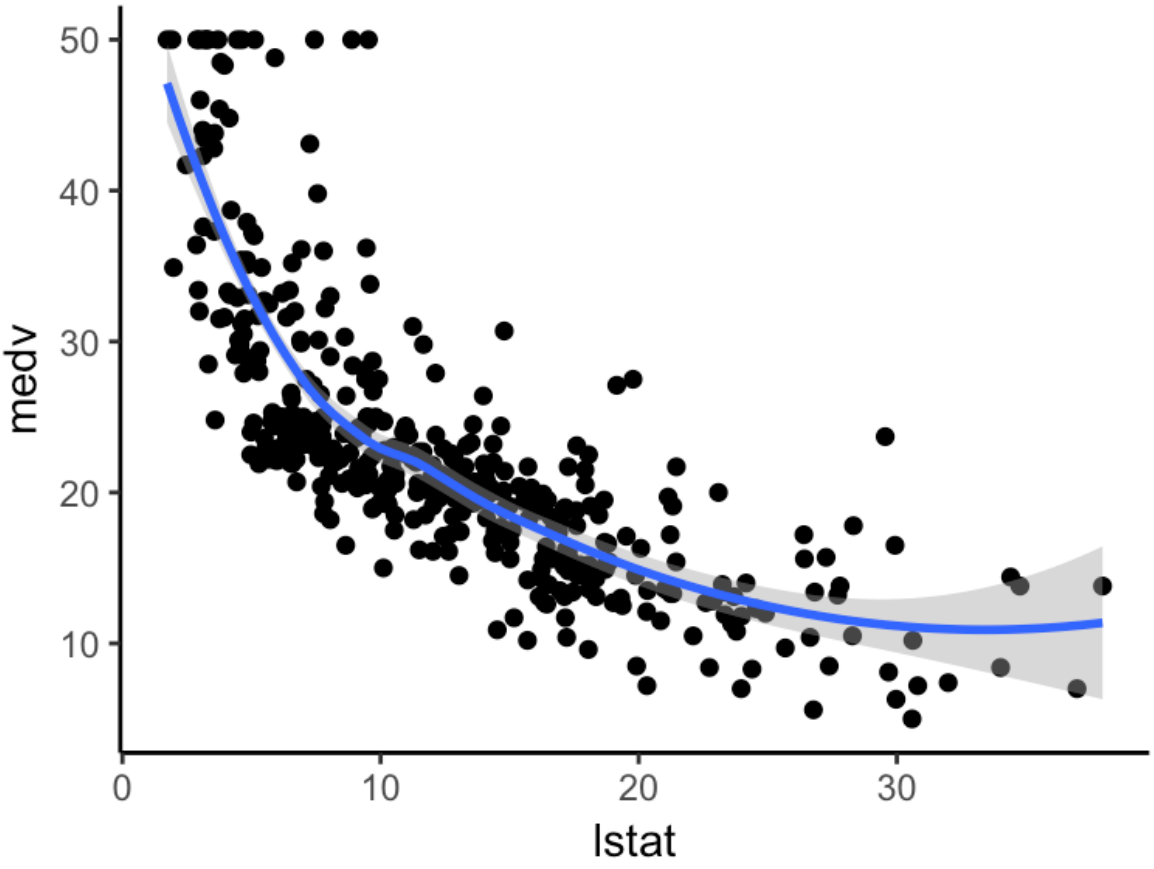

Рисунок 1.16 – Візуалізація ОПР за допомогою ggplot2 [17]

На жаль, у R-Stats відсутні, у явному вигляді, інструменти проведення багатовимірного поліноміального регресійного аналізу та можливість гнучко керувати процесом побудови регресії.

1.4 Формулювання та обґрунтування мети дослідження та переліку задач, що розв'язуються в дисертаційній роботі

Проведений вище критичний аналіз існуючих універсальних методів побудови нелінійних регресій показав:

– хоча класичні методи побудови нелінійних регресій використовують різні теоретично обґрунтовані статистичні критерії, вони не завжди гарантують ефективний розв'язок реальних практичних задач;
– евристичні методи практично завжди ефективно розв'язують реально практичні задачі, але в теоретичному плані нічого не гарантують.

Тому *мета дослідження* – підвищення точності побудови моделі багатовимірної поліноміальної регресії за рахунок створення кросплатформної спеціалізованої бібліотеки, що реалізує нові теоретичні положення та алгоритми, з використанням сучасних обчислювальних можливостей.

З урахуванням проведеного наукового критичного аналізу відомих результатів, для досягнення вказаної мети потрібно вирішити такі *завдання*:

– створити програмну реалізацію знаходження нормованих ортогональних поліномів Форсайта з точними значеннями наперед заданої кількості розрядів після коми за рахунок представлення даних у вигляді раціональних дробів та застосування до них символьних обчислень;
– створити метод побудови одновимірної поліноміальної регресії на основі довільного повторного активного експерименту з використанням лише одного набору нормованих ортогональних поліномів Форсайта;
– створити декомпозиційний метод оцінки коефіцієнтів при нелінійних членах багатовимірної поліноміальної регресії з наперед заданою точністю, що багатовимірну задачу декомпозує на послідовність побудови відповідних одновимірних поліноміальних регресій;

– створити, використовуючи постановку задачі побудови багатовимірної поліноміальної регресії, заданої надлишковим описом, модифікований метод групового урахування аргументів;
– створити, на основі отриманих попередніх теоретичних результатів, синтетичний метод побудови багатовимірної поліноміальної регресії, заданої надлишковим описом;
– провести оптимізацію окремих компонент програмного забезпечення, що реалізує синтетичний метод побудови багатовимірної поліноміальної регресії, заданої надлишковим описом;
– розробити архітектуру кросплатформної бібліотеки, яка реалізовуватиме синтетичний метод побудови багатовимірної поліноміальної регресії, заданої надлишковим описом, та матиме зручний програмний інтерфейс для користувачів різної базової підготовки.

1.5 Висновки до розділу 1

У розділі 1 було наведено основні поняття регресійного аналізу та описані типи регресійних моделей. Після виконання критичного аналізу існуючих класичних та евристичних методів побудови регресій було сформульовано мету досліджень, що полягає у необхідності створення оригінального універсального методу побудови БПР заданою надлишковим описом, що органічно поєднував би у собі характеристику класичних методів (знаходження оцінок коефіцієнтів при нелінійних членах БПР заданих надлишковим описом з заданою точністю) та ефективність евристичних методів.

Проведений критичний аналіз існуючих на сьогодні статистичних програмних пакетів та програмних бібліотек виявив ряд проблем, які існують у програмному забезпеченні, що розв’язує задачу побудови регресійних моделей. Статистичні програмні пакети мають досить складний процес інтеграції із стороннім програмним забезпеченням, їх ліцензії є достатньо багатовартісними, а оновлення версій відбувається

раз на декілька років. З іншого боку, програмні бібліотеки мають або вбудовані методи, що виключають гнучкість при проведенні регресійного аналізу, або просто набір математичних функцій, що призводить до написання програми з нуля кожного разу.

Так як запропонований метод є унікальним, то виникає потреба у створенні спеціалізованого програмного забезпечення, в якому були б враховані викладені вище зауваження до існуючих статистичних програмних засобів.

Сформульовано перелік задач, що необхідно розв’язати для досягнення поставленої мети дослідження.

2 СИНТЕТИЧНИЙ МЕТОД ПОБУДОВИ БАГАТОВИМІРНИХ ПОЛІНОМІАЛЬНИХ РЕГРЕСІЙ ЗАДАНИХ НАДЛИШКОВИМ ОПИСОМ

Задання багатовимірної поліноміальної регресії надлишковим описом дозволило [44, 46, 47, 82] знаходження структури та оцінок коефіцієнтів шуканої регресії звести до послідовної реалізації декомпозиційного методу, який дозволяє при виконанні певних умов знаходити при невеликій кількості експериментів оцінки при нелінійних членах багатовимірної поліноміальної регресії з заданою точністю та модифікувати універсальний евристичний метод групового урахування аргументів, що (як показали експерименти достатньо великої кількості дослідників) статистично значимо знаходить структуру шуканої багатовимірної поліноміальної регресії, з урахуванням коефіцієнтів, знайдених декомпозиційним методом.

Модифікація методу групового урахування аргументів полягає в тому, що завдання багатовимірної поліноміальної регресії, заданої надлишковим описом дозволило замість селекційної багаторівневої процедури створення множин часткових регресій класичним методом групового урахування аргументів використати ефективний алгоритм кластерного аналізу розбиття множини невідомих коефіцієнтів на два класи, при чому в кожну часткову регресію входять всі коефіцієнти першого класу. Таким чином, запропонований синтетичний метод має як ознаки класичних статистичних методів: можливість знаходження оцінок коефіцієнтів з наперед заданою точністю має так і властивості евристичного методу групового урахування аргументів (знаходження структури шуканої регресії з використанням перевірочної послідовності).

2.1 Модифікований метод групового урахування аргументів

Викладається відповідно до [48, 49].

Постановка задачі. Для спрощення викладок ММГУА будемо викладати для побудови БЛР, заданої надлишковим описом.

БЛР, що задана надлишковим описом, має наступний вигляд:

$$Y(\bar{\boldsymbol{x}}) = b_0 + \sum_{j=1}^{m} b_j x_j + E, \tag{2.1}$$

де $\bar{\boldsymbol{x}} = (x_1, \dots, x_m)^T$ – надлишковий вектор детермінованих вхідних змінних (деякі вхідні змінні, можливо, не впливають на вихідну змінну);

E – випадкова величина, у якої $ME = 0$ та має місце практичне обмеження на її дисперсію: $DE < \infty$;

$\bar{x}_i = (x_{1i}, \dots, x_{mi})^T$ – вектор значень вхідних детермінованих змінних в i-му випробуванні, $i = \overline{1,n}$.

Нехай $ME \neq 0$, в цьому випадку внаслідок розв'язку задачі буде оцінюватись не коефіцієнт b_0, а коефіцієнт $b_0 + ME$. Тобто модель регресії буде оцінена з точністю до константи. Той самий висновок стосується всіх регресійних моделей, розглянутих в розділі 2.

Припускається, що може бути реалізований активний експеримент, під яким розуміється, що в експерименті на вхід об'єкту може бути подане будь-яке значення вхідної змінної з наперед заданої області (відрізку). Відрізняється від пасивного експерименту тим, що в пасивному експерименті дослідник не управляє значеннями вхідних даних, а лише їх фіксує.

Результатом активного експерименту є дані $\left(\bar{x}_i \to y_i,\ i = \overline{1,n}\right)$, де

$$y_i = b_0 + \sum_{j=1}^{m} b_j x_{ji} + \varepsilon_i, \tag{2.2}$$

де ε_i – реалізація випадкової величини E в i-му експерименті.

y_i вважається реалізацією віртуальної випадкової величини Y_i,$i = \overline{1,n}$; випадкові величини Y_i, $i = \overline{1,n}$, – незалежні

$$Y_i = b_0 + \sum_{j=1}^{m} b_j x_{ji} + E_i,\ i = \overline{1,n}, \tag{2.3}$$

де випадкові величини E_i, $i = \overline{1,n}$, – віртуальні незалежні копії випадкової величини E (ε_i вважаються реалізаціями випадкової величини E_i, $i = \overline{1,n}$). Тобто, використовується основний прийом математичної статистики, який полягає в тому, що отримані реалізації внаслідок випробувань над фізично існуючою випадковою величиною E

вважаються реалізаціями фізично не існуючих (віртуальних) випадкових величин E_i, які є незалежними і розподіленими так, як випадкова величина E.

Нехай $\boldsymbol{y} = (y_1, \dots, y_n)^T$, $\boldsymbol{Y} = (Y_1, \dots, Y_n)^T$. Значення коефіцієнтів $b_j,\ j = \overline{0,m}$, – невідомі. Надлишковий опис полягає в тому, що деякі вхідні змінні можуть не впливати на значення вихідної змінної. Треба, використовуючи загальну процедуру МНК, знайти справжню структуру БЛР та оцінити значення її коефіцієнтів.

2.1.1 Алгоритмічні процедури методу ММГУА для побудови БЛР, заданої надлишковим описом

Рекомендація до проведення активного експерименту. Проведені експериментальні дослідження, в яких використовувались нульові значення вхідних змінних та експерименти, в яких значення всіх вхідних змінних не дорівнюють нулю, показали, що бажано значення вхідних змінних не брати рівними нулю, тобто $\forall x_{ij} \neq 0$. Якісне пояснення рекомендації випливає з того, що на основі результатів активного експерименту треба виключити всі вхідні змінні, що не впливають на вихідну змінну.

Алгоритм кластерного аналізу. Алгоритм кластерного аналізу складається з наступних процедур.

Знаходження за результатами активного експерименту $\left(\bar{x}_i \to y_i,\ i = \overline{1,n}\right)$ по надлишковому опису (2.1) загальною схемою МНК оцінок коефіцієнтів $b_j,\ j = \overline{0,m}$, та ранжування модулів їх значень:

$$\left|\hat{b}_{j_1}\right| \geq \left|\hat{b}_{j_2}\right| \geq \cdots \geq \left|\hat{b}_{j_{m+1}}\right|.$$

Всі коефіцієнти $b_j,\ j = \overline{0,m}$, послідовною процедурою розбиваються на два класи: M_1 та M_2.

Перший крок: $b_{j_1} \in M_1,\ b_{j_{m+1}} \in M_2$. Якщо $\left|\hat{b}_{j_1}\right| - \left|\hat{b}_{j_2}\right| < \left|\hat{b}_{j_2}\right| - \left|\hat{b}_{j_{m+1}}\right|$, то $b_{j_2} \in M_1$, в протилежному випадку $b_{j_2}, b_{j_3}, \dots, b_{j_{m+1}} \in M_2$. Розбиття завершене.

Другий крок: $b_{j_2} \in M_1$. Якщо $\frac{1}{2}\left(\left|\hat{b}_{j_1}\right| + \left|\hat{b}_{j_2}\right|\right) - \left|\hat{b}_{j_3}\right| < \left|\hat{b}_{j_3}\right| - \left|\hat{b}_{j_{m+1}}\right|$, то $b_{j_3} \in M_1$, в протилежному випадку розбиття завершене, $M_1 = \left\{b_{j_1},\ b_{j_2}\right\}$, $M_2 = \left\{b_{j_3}, \dots, b_{j_{m+1}}\right\}$.

l-тий крок: $b_{j_l} \in M_1$. Якщо $\frac{1}{l}\sum_{i=1}^{l}\left|\hat{b}_{j_i}\right| - \left|\hat{b}_{j_{l+1}}\right| < \left|\hat{b}_{j_{l+1}}\right| - \left|\hat{b}_{j_{m+1}}\right|$, то $b_{j_{l+1}} \in M_1$, в протилежному випадку розбиття завершене, $M_1 = \left\{b_{j_1}, \dots,\ b_{j_l}\right\}$, $M_2 = \left\{b_{j_{l+1}}, \dots, b_{j_{m+1}}\right\}$.

Очевидно, що за обмежену кількість кроків алгоритм остаточно завершує розбиття коефіцієнтів $b_j,\ j = \overline{0,m}$, на два класи M_1 та M_2.

Примітка 2.1. Для спрощення будемо вважати, що БЛР заданим надлишковим описом $b_0 \in M_1$.

В наведеному алгоритмі використовується $\frac{1}{l}\sum_{i=1}^{l}\left|\hat{b}_{j_i}\right|$ як міра середнього відхилення модуля $\left|\hat{b}_{j_{l+1}}\right|$ від $\left|\hat{b}_{j_1}\right|, \left|\hat{b}_{j_2}\right|, \dots, \left|\hat{b}_{j_l}\right|$, тому що ця міра, очевидно, не залежить від того, нормується значення $\left|\hat{b}_j\right|, j = \overline{0,m}$, чи ні.

2.1.2 Агрегований алгоритм знаходження структури БЛР (з оцінкою невідомих коефіцієнтів) заданої надлишковим описом.

Множину M_2 представлено в вигляді:

$$M_2 = \cup_j M_2^j,\ j = \overline{1,\ \sum_{t=1}^{|M_2|} C^t{}_{|M_2|}}, \tag{2.4}$$

де $|M_2|$ – число елементів в множині M_2.

$$\forall_{e_1 \neq e_2} M_2^{e_1} \neq M_2^{e_2}. \tag{2.5}$$

Для кожної j-тої БЛР (кожного j-го часткового опису), $j = \overline{1,\ \sum_{t=1}^{|M_2|} C^t_{|M_2|}}$:

$$Y(\bar{x}) = \widehat{b_0} + \sum_{\forall \hat{b}_e \in M_1}(b_e \cdot x_e) + \sum_{\forall \hat{b}_e \in M_2^j}(b_e \cdot x_e) + E, \tag{2.6}$$

за результатом активного експерименту ($\bar{x}_i \to \bar{y}_i$) загальним методом найменших квадратів отримуємо відповідні нові оцінки $\forall \hat{b}_e \in M_1; \hat{b}_e \in M_2^j$.

Таким чином, розбиття коефіцієнтів БЛР заданої надлишковим описом, на два класи дозволяє за результатами активного експерименту $\left(\bar{x}_i \to y_i,\ i = \overline{1,n}\right)$ знаходити оцінки коефіцієнтів часткових описів шуканої БЛР, що формуються по наступному правилу: в кожну з них входять всі члени, коефіцієнти яких знаходяться в множині M_1, а також всі можливі різні комбінації членів, коефіцієнти яких належать множині M_2.

Примітка 2.2. Якщо в формулі МНК $\hat{b} = (\boldsymbol{A}^T \boldsymbol{A})^{-1}\boldsymbol{y}$, матриця $\boldsymbol{A}$ побудована для оцінок коефіцієнтів БЛР, заданої надлишковим описом за результатами активного експерименту $\left(\bar{x}_i \to y_i,\ i = \overline{1,n}\right)$, то для побудови відповідної матриці для оцінки коефіцієнтів часткового опису БЛР за результатами активного експерименту $\left(\bar{x}_i \to y_i,\ i = \overline{1,n}\right)$ треба в матриці $\boldsymbol{A}$ залишити лише ті стовбці, які відповідають коефіцієнтам, що входять до часткового опису. Вектор $\boldsymbol{y}$ не змінюється.

В загальному випадку при достатньої кількості вхідних змінних m, суттєвому розкиду значень модулів ненульових коефіцієнтів шуканої БЛР, множина M_2 може включати в себе ненульові коефіцієнти, а $|M_2|$ – бути достатньо великим числом, що може привести до побудови недопустимо великої кількості часткових описів БЛР. В цьому випадку пропонується наступна модифікація алгоритму кластерного аналізу: алгоритм доповнюється наступною процедурою.

З множини M_2 виключаються члени $b_{j_l}, b_{j_{l+1}}, \dots, b_{j_{m+1}}$, що задовольняють умовам:

$$\left|\hat{b}_{j_l}\right| - \left|\hat{b}_{j_{m+1}}\right| \le \Delta_{DE,n}, \left|\hat{b}_{j_{l-1}}\right| - \left|\hat{b}_{j_{m+1}}\right| > \Delta_{DE,n}, \tag{2.7}$$

де $\Delta_{DE,n} > 0$ – експертна границя, значення якої знаходиться за результатами експериментів, залежна від значення DE, кількості експериментів n, та статистично значимо гарантує, що $b_{j_k} = 0,\ k = \overline{l, m+1}$. Тобто, $\Delta_{DE,n} > 0$ повинно бути достатньо малим числом, а це означає, що в M_2 можуть залишитись ненульові коефіцієнти, і

необхідність перебору по множині M_2 залишається, але суттєво зменшується кількість часткових описів БЛР, що є залишковими. Шукана регресія знаходиться за допомогою одночасного використання двох критеріїв: мінімум ЗСК та значення χ^2, що перевіряє гіпотезу про належність оцінок реалізацій випадкової величини E заданому розподілу.

2.1.3 Алгоритм побудови ЗСК для часткових описів БЛР

Для знаходження ЗСК необхідно отримати перевірочну послідовність на основі повтору основного активного експерименту ($\bar{x}_i \to y_{n+i},\ i = \overline{1,n}$). В ЗСК для кожного часткового опису задається наступною формулою:

$$\text{ЗСК}\left(M_1, M_2^j\right) = \sum_{i=1}^{n}\left[y_{n+i} - \widehat{b_0} - \sum_{\forall \hat{b}_l \in M_1}\left(\hat{b}_l \cdot x_{li}\right) - \sum_{\forall \hat{b}_l \in M_2^j}\left(\hat{b}_l \cdot x_{li}\right)\right]^2, \qquad (2.8)$$

де $M_2^j \subset M_2$; M_1, M_2^j однозначно задають частковий опис БЛР.

Примітка 2.3. Формула (2.8) реалізує основну ідею МГУА, а саме ЗСК часткових описів знаходиться не по результатам основного експерименту, а по результатам перевірочної послідовності, яка не використовувалась при знаходженні оцінок усіх часткових описів. Ідея полягає в тому, що той часткових опис, структура якого є правильною і має меншу ЗСК на перевірочній послідовності. Таким чином, використання перевірочної послідовності розв'язує задачу ідентифікації моделі шуканої регресії.

2.1.4 Алгоритм знаходження реалізацій випадкової величини *E*

Проранжуємо та пронумеруємо часткові описи БЛР за зростанням значень їх ЗСК, та залишаємо перші t з них, для яких виконується

$$\begin{gathered}\text{ЗСК}(M_1, M_2^1) \leq \text{ЗСК}(M_1, M_2^2) \leq \ldots \leq \text{ЗСК}(M_1, M_2^t), \\ \text{ЗСК}(M_1, M_2^t) - \text{ЗСК}(M_1, M_2^1) < \\ < \text{ЗСК}(M_1, M_2^{t+1}) - \text{ЗСК}(M_1, M_2^t).\end{gathered} \qquad (2.9)$$

Нерівність (2.9) означає, що значення ЗСК у перших t часткових описів практично однакові, а значення ЗСК інших часткових описів суттєво від них відрізняються.

Примітка 2.4. t може дорівнювати одиниці. Практична однаковість значень ЗСК-тів може задаватись в долях від значення мінімальної $ЗСК(M_1, M_2^1)$ та знаходитись експериментально. В середньому це 2÷3% від величини $ЗСК(M_1, M_2^1)$.

Для кожного з часткових описів (M_1, M_2^l), $l = \overline{1,t}$, повторно знайдемо оцінки їх коефіцієнтів з використанням всіх експериментальних даних $(\bar{x}_i \to y_i,\ i = \overline{1,n})$ та $(\bar{x}_i \to y_{n+i},\ i = \overline{1,n})$. В п. 3.4 показано, що в цьому випадку при знаходженні оцінок коефіцієнтів часткового опису БЛР по всьому набору даних використовуються лише обернені матриці, знайдені за результатами активного експерименту $(\bar{x}_i \to y_i,\ i = \overline{1,n})$, а дисперсії оцінок коефіцієнтів БЛР зменшуються в 2 рази (у k разів, якщо повторний експеримент реалізує k повторів основного експерименту).

Оцінки E_i^j реалізацій випадкової величини E для часткового опису БЛР $(M_1, M_2^j),\ j = \overline{1,t}$, знаходимо по формулам:

$$E_i^j = y_i - \widehat{b_0} - \sum_{\forall \hat{b}_l \in M_1}(\hat{b}_l \cdot x_{li}) - \sum_{\forall \hat{b}_l \in M_2^j}(\hat{b}_l \cdot x_{li}),\ i = \overline{1,n},$$
$$E_{n+i}^j = y_{n+i} - \widehat{b_0} - \sum_{\forall \hat{b}_l \in M_1}(\hat{b}_l \cdot x_{li}) - \sum_{\forall \hat{b}_l \in M_2^j}(\hat{b}_l \cdot x_{li}),\ i = \overline{1,n}. \qquad (2.10)$$

Примітка 2.5. Узагальнення. Активний експеримент може складатись з самого початку з k повторів основного експерименту і дані останніх k_1 повторів основного експерименту $k_1 < k$ будуть перевірочною послідовністю, а по даним перших $k - k_1$ повторів по основному експерименту загальною формулою МНК оцінюються коефіцієнти всіх часткових описів, включаючи надлишковий.

2.1.5 Алгоритм побудови лінгвістичної змінної

З множини часткових описів БЛР (M_1, M_2^j), $j = \overline{1,t}$, вибираємо той, що містить мінімальну кількість членів.

Примітка 2.6. В більшості випадків йому відповідає мінімальна ЗСК.

Пропонується вважати цей частковий опис ефективною апроксимацією шуканої БЛР, тому що він містить лише ті вхідні змінні, що суттєво впливають на значення

вихідної змінної. Саме для нього будуємо лінгвістичну змінну. Позначимо цей опис БЛР $\left(M_1, M_2^{j_1}\right)$.

Якщо відома функція щільності випадкової величини E чи відомий її аналітичний вираз з точністю до значень її числових параметрів, за оцінками реалізації випадкової величини E (2.10) для часткового опису БЛР $\left(M_1, M_2^{j_1}\right)$ критерієм χ^2 перевіряємо гіпотезу, що ці оцінки є її реалізаціями. Нехай $\chi^2\left(M_1, M_2^{j_1}\right)$ – це реалізація критерію χ^2 для часткового опису БЛР $\left(M_1, M_2^{j_1}\right)$, що має $r \geq 3$ ступенів свободи. Нехай задано експертні числа q_1, q_2, що задовольняють умовам:

$$P(\chi^2 \geq q_1) = 0.05, P(\chi^2 \geq q_2) = 0.4, \tag{2.11}$$

$r - 2$ – це значення аргументу, при якому функція щільності випадкової величини χ^2 досягає свого єдиного максимуму. Тоді, якщо виконується умова:

$$r - 2 \leq \chi^2\left(M_1, M_2^{j_1}\right) \leq q_2, \tag{2.12}$$

то з високою достовірністю частковий опис БЛР $\left(M_1, M_2^{j_1}\right)$ є шуканою БЛР, що містить всі змінні, кожна з яких суттєво впливає на вихідну змінну. Якщо

$$q_2 \leq \chi^2\left(M_1, M_2^{j_1}\right) < q_1, \tag{2.13}$$

то отриманий результат має достатню ступінь достовірності. Якщо

$$\chi^2\left(M_1, M_2^{j_1}\right) \geq q_1,$$

то отриманий результат є недостовірним.

Примітка 2.7. Якщо є декілька описів з мінімальною кількістю членів $\left(M_1, M_2^{j_1}\right)$, …, $\left(M_1, M_2^{j_p}\right)$, то досліджуються всі, але першим претендентом на розв'язок є той, у якого досягається

$$\min_{i=\overline{1,p}}\left[r - 2 - \chi^2\left(M_1, M_2^{j_i}\right)\right].$$

Примітка 2.8. В якості можливого додаткового аналізу фахівцями з предметної галузі можуть розглядатися інші часткові описи з множини $\left(M_1, M_2^j\right)$, $j = \overline{1,t}$, що пройшли перевірку критерієм χ^2 на належність чисел (2.10) розподілу випадкової величини E.

Узагальнення отриманого результату. Зрозуміло, що тривіальним чином ММГУА розповсюджується на задачу побудови БПР заданої надлишковим описом, в силу того, що всі невідомі коефіцієнти в модель БПР входять лінійно.

2.2 Метод побудови одновимірної поліноміальної регресії, заданої надлишковим описом, з використанням одного набору нормованих ортогональних поліномів Форсайта при довільному повторному активному експерименті

2.2.1 Основні теоретичні положення

2.2.1.1 Оцінка коефіцієнтів одновимірної поліноміальної регресії з використанням нормованих ортогональних поліномів Форсайта

Постановку задачі та аналіз відомих результатів наведемо згідно з [22].

Модель регресії має вигляд

$$Y(x) = \theta_0 + \theta_1 x + \cdots + \theta_r x^r + E, \tag{2.14}$$

де *x* – детермінована змінна, значення якої в експериментах дослідник може задавати довільно; $\theta_i, i = \overline{0,r}$ – невідомі коефіцієнти, *E* – випадкова величина з довільним розподілом, *ME = 0* (*M* – знак математичного очікування); σ_{E}^2(дисперсія) обмежена і її значення невідоме або ж відома верхня оцінка σ^2.

Проведено *n* експериментів, результатом яких є 2 вибірки об'єму *n* (x_i, $i = \overline{1,n}$; $Y(x_i) = y_i, i = \overline{1,n}$).

Згідно з (2.14),

$$y_i = \sum_{j=0}^{r} \theta_j x_i^j + \delta_i, i = \overline{1,n}\,, \tag{2.15}$$

де δ_i – невідома реалізація випадкової величини E в i-тому експерименті. Числа $y_i,\ \delta_i$ можна вважати реалізаціями випадкових величин $Y_i\ i = \overline{1,n};\ \Delta_i\ i = \overline{1,n}$, де Δ_i має розподіл випадкової величини E, а Y_i і Δ_i зв'язані відношенням:

$$Y_i = \sum_{j=0}^{r} \theta_j x_i^j + \Delta_i, \tag{2.16}$$

де $\Delta_i,\ i = \overline{1,n}$ – незалежні випадкові величини розподілені так само, як і випадкова величина E; $Y_i\ \ i = \overline{1,n}$ – незалежні випадкові величини з дисперсією σ_{E}^2.

Оцінками невідомих коефіцієнтів $\theta_i, i = \overline{0,r}$ є їх значення, яким відповідає

$$\min_{\theta_i, i=\overline{0,r}} \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{r} \theta_j x_i^j \right)^2 \tag{2.17}$$

Введемо матричні позначення:

$$\text{A} = \begin{pmatrix} 1 x_1 \dots x_1^r \\ \dots \dots \dots \\ 1 x_1 \dots x_1^r \end{pmatrix}; y = (y_1, \dots y_n)^T,$$

$$Y = (Y_1, \dots Y_n)^T; \theta = (\theta_0, \dots \theta_r)^T,$$

$$\hat{\theta} = \left(\widehat{\theta_0}, \dots \widehat{\theta_r}\right)^T,$$

де $\widehat{\theta_j}$ – оцінки $\theta_i, i = \overline{0,r}$, у відповідності до (2.17). Тоді [22]

$$\hat{\theta} = (A^T A)^{-1} A^T y, \tag{2.18}$$

або $\hat{\theta} = (A^T A)^{-1} A^T Y$, якщо $\theta_i, i = \overline{0,r}$, вважати випадковими величинами. Проблеми, зв'язані з оберненням матриці $(A^T A)^{-1}$ зникають, якщо від моделі (2.14) перейти до моделі регресії, яка задана за допомогою нормованих ортогональних поліномів [22]:

$$Y(x) = w_0 Q_0(x) + w_1 Q_1(x) + \dots + w_r Q_r(x) + E, \tag{2.19}$$

де $Q_j(x),\ j = \overline{0,r}$ – нормовані ортогональні поліноми,

$$Q_j(x) = q_{j_0} + q_{j_1}x + \cdots q_{j_j}x^j \tag{2.20}$$

$$\sum_{i=l}^{n} {Q_j}^2(x_i) = 1, \sum_{i=l}^{n} Q_j(x_i)Q_l(x_i) = 0 \; \forall j \neq l, j, l = \overline{0, r}$$

Дж. Форсайт [22] запропонував рекурентну формулу для знаходження нормованих ортогональних поліномів:

$$\lambda Q_j(x) = xQ_{j-1}(x) - \alpha Q_{j-1}(x) - \beta Q_{j-2}(x) \tag{2.21}$$

$$\alpha = \sum_{i=1}^{n} x_i {Q_{j-1}}^2(x_i); \beta = \sum_{i=1}^{n} x_i Q_{j-1}(x_i)Q_{j-2}(x_i);$$

λ виводиться з умови $\sum_{i=l}^{n} {Q_j}^2(x_i) = 1$.

$$\lambda = \sqrt{\sum_{i=1}^{n} \left(x_i Q_{j-1}(x_i) - \alpha Q_{j-1}(x_i) - \beta Q_{j-2}(x_i)\right)^2}.$$

Для використання формули (2.21) необхідно побудувати нормовані ортогональні поліноми $Q_0(x)$ та $Q_1(x)$. Зрозуміло, що ці поліноми це:

$$Q_0(x) = \frac{1}{\sqrt{n}}; Q_1(x) = -\frac{\bar{x}}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}} + \frac{x}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}},$$

де $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.

Застосування метода найменших квадратів до моделі (2.19) призводить до наступних результатів [22]:

Нехай $w = (w_0, \dots w_r)^T; \widehat{w} = (\widehat{w_0}, \dots \widehat{w_r})^T \widehat{w}_j, j = \overline{0, r}$ оцінки w_j отримані методом найменших квадратів $\widehat{W} = \left(\widehat{W_0}, \dots \widehat{W_r}\right)^T \widehat{W}_j, j = \overline{0, r}$ – випадкові величини, для яких $\widehat{w}_j$ являється відповідною реалізацією. Тоді:

$$\widehat{w}_j = \sum_{j=1}^{n} y_i Q_j(x_i), j = \overline{0,r};\ \widehat{W}_j = \sum_{j=1}^{n} Y_i Q_j(x_i) \tag{2.22}$$

$$M\widehat{W}_j = w_j, j = \overline{0,r}; \operatorname{cov}\left(\widehat{W}_j, \widehat{W}_l\right) = 0, \forall j \neq l.$$

$$D\widehat{W}_j = \sigma_{\text{E}}^2 \tag{2.23}$$

$$M\frac{\sum_{i=l}^{n} Y_i^2 - \sum_{j=1}^{r} \widehat{W}_i^2}{n-(r+1)} = \sigma_{\text{E}}^2 \tag{2.24}$$

Покажемо [83], що $\operatorname{cov}\left(\widehat{W}_j,\ \widehat{W}_l\right) = 0, \forall j \neq l$ $\operatorname{cov}\left(\widehat{W}_j, \widehat{W}_l\right) = 0, l \neq p$, якщо $M\left(\widehat{W}_l \cdot \widehat{W}_p\right) = M\widehat{W}_l \cdot \text{M}\widehat{W}_p$:

$$\widehat{W}_l = \sum_{i=1}^{n} Y_i Q_l(x_i)\,, \widehat{W}_p = \sum_{i=1}^{n} Y_i Q_p(x_i)\,, M\widehat{W}_l = \sum_{i=1}^{n} Q_l(x_i) \cdot MY_i\,, M\widehat{W}_p = \sum_{j=1}^{n} Q_p\left(x_j\right) \cdot MY_j$$

$$M\left(\widehat{W}_l \cdot \widehat{W}_p\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} Q_l(x_i)Q_p\left(x_j\right) \cdot MY_i \cdot MY_j + \sum_{i=1}^{n} M(Y_i)^2 \cdot Q_l(x_i)Q_p(x_i), \tag{2.25}$$

$$M\widehat{W}_l \cdot \text{M}\widehat{W}_p = \sum_{i=1}^{n}\sum_{j=1}^{n} Q_l(x_i)Q_p\left(x_j\right) \cdot MY_i \cdot MY_j + \sum_{i=1}^{n} (MY_i)^2 \cdot Q_l(x_i)Q_p(x_i) \tag{2.26}$$

Розглянемо різницю (2.25) і (2.26):

$$M\left(\widehat{W}_l \cdot \widehat{W}_p\right) - M\widehat{W}_l \cdot \text{M}\widehat{W}_p = \sum_{i=1}^{n} (M(Y_i)^2 - (MY_i)^2) \cdot Q_l(x_i)Q_p(x_i) =$$

$$= \sum_{i=1}^{n} DY_i \cdot Q_l(x_i)Q_p(x_i) = \sigma^2 \sum_{i=1}^{n} Q_l(x_i)Q_p(x_i) = 0,$$

при $l \neq p$, оскільки $Q_l(x)$ і $Q_p(x)$ – це ортогональні поліноми.

Зв’язок моделей (2.14) і (2.19) являється наступною:

$$\theta_j = w_r q_{rj} + w_{r-1} q_{r-1,j} + \cdots + w_j q_{jj} \tag{2.27}$$

і, відповідно,

$$\widehat{\theta_j} = \widehat{w}_r q_{rj} + \cdots + \widehat{w}_j q_{jj}, j = \overline{0,r} \tag{2.28}$$

або

$$\widehat{\theta_j} = \widehat{W}_r q_{rj} + \cdots + \widehat{W}_j q_{jj}, \tag{2.29}$$

якщо $\widehat{\theta_j}$ вважати випадковою величиною.

Враховуючи, що $\mathrm{cov}(\widehat{W_j}, \widehat{W_l}) = 0$, з формул (2.23) і (2.29) отримуємо:

$$D\widehat{\theta_j} = \sigma^2 \sum_{l=r}^{j} q_{lj}^2 \tag{2.30}$$

2.2.1.2 Повторний активний експеримент

В [56] показано: нехай на вхід об'єкта подається послідовність чисел $x_1, \dots, x_n, x_1, \dots, x_n \dots, x_1, \dots, x_n, n > r$, в якій послідовність чисел $x_1, \dots, x_n$ повторюється l разів. Нехай $X = (x_1, \dots, x_n)$, $Y = \left(\frac{\sum_{k=1}^{l} y_{k1}}{l}, \frac{\sum_{k=1}^{l} y_{k2}}{l}, \dots, \frac{\sum_{k=1}^{l} y_{kn}}{l}\right)$, $X' = (x_{11}, x_{21}, \dots, x_{l1}, x_{12}, x_{22}, \dots, x_{l2}, \dots, x_{1n}, x_{2n}, \dots, x_{ln})$, де $x_{ki} = x_i \, \forall k = \overline{1,l}, \forall i = \overline{1,n}$. $Y' = (y_{11}, y_{21}, \dots, y_{l1}, \dots, y_{1n}, y_{2n}, \dots, y_{ln})$, де y_{ij} – значення виходу об'єкта при значенні входу x_{ij}. Нехай $Q_j(x),\ j = \overline{0,r}$, – це НОПФ, побудовані по послідовності чисел X, а $Q'_j(x)$ – НОПФ, побудовані по послідовності чисел X'. Тоді [55] $Q'_j(x) = Q_j(x)/\sqrt{l},\ j = \overline{0,r}$ (для значень $x = x_i = x_{ki},\ k = \overline{1,l}$), звідки випливає [55], що оцінки, отримані по результатам експерименту (X, Y) і (X', Y'), однакові. Іншими словами, оцінки $\theta_j,\ j = \overline{0,r}$, по експерименту (X, Y) відповідають фактично наступній задачі побудови ОПР:

$$Y(x) = \theta_0 + \theta_1 x + \dots + \theta_r x^r + E_1, ME_1 = 0, DE_1 = \frac{\sigma^2}{l}. \tag{2.31}$$

Вибором відповідного значення l можна досягнути заданої точності оцінок $\hat{\theta}_j,\ j = \overline{2,r}$.

2.2.2 Побудова одновимірної поліноміальної регресії з використанням одного набору нормованих ортогональних поліномів Форсайта

Нехай ОПР має вигляд (2.14). В [54] наведено наступний результат. Нехай одновимірна поліноміальна регресія має наступний вигляд:

$$y(x) = \theta_0 + \theta_1 x + \ ... + \theta_r x^r, \tag{2.32}$$

нехай $x = az + b$, де a, b – довільні константи. Тоді між коефіцієнтами γ_0, γ_1, … γ_r полінома

$$\begin{aligned} y(z) = \theta_0 + \theta_1(az + b) + \theta_2(az + b)^2 + \cdots + \\ + \theta_r(az + b)^r = \gamma_0 + \gamma_1 z + \cdots + \gamma_r z^r \end{aligned} \tag{2.33}$$

існує взаємно однозначна відповідність наступного виду

$$\boldsymbol{A_r}\begin{pmatrix} \theta_0 \\ \theta_1 \\ ... \\ \theta_r \end{pmatrix} = \begin{pmatrix} \gamma_0 \\ \gamma_1 \\ ... \\ \gamma_r \end{pmatrix}, \tag{2.34}$$

де матриця $\boldsymbol{A_r}$ має вигляд верхньої трикутної:

$$\boldsymbol{A_r} = \begin{pmatrix} 1 & & & \\ & a & & a_{ij} \\ & & a^2 & \\ & 0 & & \ddots \\ & & & a^r \end{pmatrix}. \tag{2.35}$$

Цей результат дозволяє отримати оцінки коефіцієнтів при нелінійних членах одновимірної поліноміальної регресії (2.14) за результатами обмеженого активного експерименту $x_i \to y_i,\ i = \overline{1,n}\ \forall x_i \in [c,d]$, $c < d$ – довільні дійсні числа, $x_1 = c < x_2 < \cdots < x_n = d,\ x_j - x_{j-1} = \text{const}, j = \overline{2,n}$, з використанням при цьому лише результатів віртуального активного експерименту для задачі віртуальної одновимірної поліноміальної регресії

$$y(z) = \gamma_0 + \gamma_1 z + \cdots + \gamma_r z^r + E \tag{2.36}$$

$(z_i \to y_i, i = \overline{1,n})$, де

$$a = \frac{d-c}{z_n - z_1} > 0,\ b = x_1 - \frac{d-c}{z_n - z_1} \cdot z_1, \tag{2.37}$$

поклавши

$$z_1 < z_2 < \cdots < z_n,$$
$$z_j - z_{j-1} = const, j = \overline{2,n}. \tag{2.38}$$

Значення скалярної вхідної змінної x ОПР (2.14) зв'язані зі значеннями скалярної вхідної змінної z віртуальної ОПР (2.36) виразом

$$x_j = az_j + b, j = \overline{2,n},\ x_1 = c. \tag{2.39}$$

Це дозволяє використовувати лише один набір нормованих ортогональних поліномів Форсайта $Q_j(z), j = \overline{0,r}$, побудованих для значень скалярної змінної z (2.38) для довільних $c < d$ при заданому n. Відповідно до [22], нехай нормовані ортогональні поліноми Форсайта $Q_j(z) = q_{j0} + q_{j1}z + \cdots + q_{jj}z^j,\ j = \overline{0,r}$, де числа $q_{ij},\ j = \overline{0,r}$, було отримано по числам $z_j,\ j = \overline{1,n}$ (див. (2.37)). Тоді оцінки $\hat{\gamma}_j,\ j = \overline{0,r}$, отримані методом найменших квадратів за результатами віртуального експерименту $z_i \to y_i, i = \overline{1,n}$ ($y_i, i = \overline{1,n}$, – це вихідні результати активного експерименту одновимірної поліноміальної регресії (2.14) для $x_i, i = \overline{1,n}$, зв'язаних зі значеннями $z_i, i = \overline{1,n}$, виразами (2.37), (2.39)), в силу (2.27–2.30) мають наступний вигляд:

$$\hat{\gamma}_j = \hat{w}_r q_{rj} + \cdots + \hat{w}_j q_{jj}, j = \overline{0,r}, \tag{2.40}$$
$$\hat{w}_j = \sum_{i=1}^{n} y_i\, Q_j(z_i), D\hat{w}_j = \sigma^2, \tag{2.41}$$
$$M\hat{\gamma}_j = \gamma_j,\ D\hat{\gamma}_j = \sigma^2 \sum_{i=r}^{j} q_{ij}^2. \tag{2.42}$$

Значення $\hat{\gamma}_j,\ j = \overline{0,r}$, в силу методу найменших квадратів є

$$arg \min_{\gamma_0,\gamma_1,\dots,\gamma_r} \left(y_i - \sum_{j=0}^{r} \gamma_j z_i^j\right)^2. \quad (2.43)$$

Нехай $\hat{\theta}_j,\ j = \overline{0,r}$, – розв'язок системи лінійних рівнянь

$$\boldsymbol{A_r} \begin{pmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_r \end{pmatrix} = \begin{pmatrix} \hat{\gamma}_0 \\ \hat{\gamma}_1 \\ \dots \\ \hat{\gamma}_r \end{pmatrix}, \quad (2.44)$$

де змінними є $\theta_j,\ j = \overline{0,r}$. Тоді виконується наступне твердження.

Твердження 2.1. $\hat{\theta}_j,\ j = \overline{0,r}$, є

$$arg \min_{\theta_0,\theta_1,\dots,\theta_r} \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{r} \theta_j x_i^j\right)^2.$$

Доведення проведемо від супротивного. Нехай

$$\sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{r} \hat{\theta}_j x_i^j\right)^2 > \min_{\theta_0,\theta_1,\dots,\theta_r} \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{r} \theta_j x_i^j\right)^2$$

та $\theta_j^*,\ j = \overline{0,r}$, є $\arg \min_{\theta_0,\theta_1,\dots,\theta_r} \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{r} \theta_j x_i^j\right)^2$. Нехай $\gamma_j^*,\ j = \overline{0,r}$, знайдені за системою лінійних рівнянь

$$\begin{pmatrix} \gamma_0^* \\ \gamma_1^* \\ \dots \\ \gamma_r^* \end{pmatrix} = \boldsymbol{A_r} \begin{pmatrix} \theta_0^* \\ \theta_1^* \\ \dots \\ \theta_r^* \end{pmatrix}. \quad (2.45)$$

Тоді в силу взаємної однозначності розв'язків (2.45) та рівнянь $\sum_{j=0}^{r} \theta_j^* x_i^j = \sum_{j=0}^{r} \gamma_j^* z_i^j,\ i = \overline{1,n}$, умова (2.43) не виконується. Твердження 2.1 доведено.

Наслідок. При використанні нормованих ортогональних поліномів Форсайта для $r \le r_{\max}$, при фіксованому $n \ge r$ для довільних $c < d$ (при цьому $[c,d]$ – область можливих значень скалярної детермінованої змінної задачі одновимірної поліноміа-

льної регресії (2.14), для $x_1 = c < x_2 < \cdots < x_n = d,\ x_j - x_{j-1} = \text{const}, j = \overline{2,n}$, оцінка $\hat{\theta}_j, j = \overline{2,r}$, може бути знайдена по одному набору нормованих ортогональних поліномів Форсайта $Q_j(z),\ j = \overline{2,r}$, побудованому для

$$z_1 < z_2 < \cdots < z_n,$$
$$z_j - z_{j-1} = const,\ j = \overline{2,n}, \tag{2.46}$$

з використанням формул (2.37) та (2.39).

Значення коефіцієнтів $q_{ij}, i,j = \overline{0,r}$, повинні бути знайдені попередньо з наперед заданою кількістю розрядів після коми.

Для використання лише одного набору НОПФ (2.46) потрібно знайти значення дисперсій $D\hat{\theta}_j,\ j = \overline{2,r}$, за значеннями дисперсій $D\hat{\gamma}_j,\ j = \overline{2,r}$, що задаються формулою (2.42) [22].

Структура матриці коефіцієнтів $\boldsymbol{A_r}$ системи лінійних рівнянь (2.34) дозволяє отримати розв'язок $\hat{\theta}_j,\ j = \overline{2,r}$ наступного вигляду:

$$\hat{\theta}_r = \frac{1}{a^r}\hat{\gamma}_r;\ \hat{\theta}_{r-1} = \frac{1}{a^{r-1}}\left(\hat{\gamma}_{r-1} - \frac{a_{r-1,r}}{a^r}\hat{\gamma}_r\right)$$

і в загальному вигляді:

$$\hat{\theta}_{r-j} = b_{r-j,1}\hat{\gamma}_{r-j} + b_{r-j,2}\hat{\gamma}_{r-j+1} + \cdots +$$
$$b_{r-j,j+1}\hat{\gamma}_r,\ j = \overline{0,r-2}. \tag{2.47}$$

Примітка 2.9. Якщо $a = \frac{1}{k}$, де $k \geq 1$ – ціле або дійсне число з обмеженою кількістю знаків після коми. Тоді при знаходженні коефіцієнтів $\forall b_{l,m}$ системи рівнянь (2.47) відсутня операція ділення.

В силу (2.40) маємо

$$\hat{\theta}_j = b_{r-j,1}\sum_{l=r}^{j}\widehat{w}_l q_{lj} + b_{r-j,2}\sum_{l=r}^{j+1}\widehat{w}_l q_{lj} + \cdots +$$
$$+ b_{r-j,j+1}\widehat{w}_r q_{rj} = \sum_{l=r}^{j} c_{lj}\widehat{w}_l,\ j = \overline{2,r}. \tag{2.48}$$

В силу (2.41) та наступної властивості $\widehat{w}_j,\ j = \overline{0,r}$ [22]:

$$\forall l \neq m\ cov(\widehat{w}_l, \widehat{w}_m) = 0,$$

якщо $\widehat{w}_l, \widehat{w}_m$ вважати ВВ величинами, то отримаємо:

$$D\hat{\theta}_j = \sigma^2 \sum_{l=r}^{j} {c_{lj}}^2\ ,\ j = \overline{2,r}. \tag{2.49}$$

Це дозволить отримати два можливих алгоритми знаходження оцінок $\hat{\theta}_j, j = \overline{2,r}$, для задачі одновимірної поліноміальної регресії (2.14), $x \in [c,d]$.

Перший алгоритм. Знаходимо такі n, $x_1 = c < x_2 < \cdots < x_n = d$, $x_j - x_{j-1} = \text{const},\ j = \overline{2,n}$, із заданою точністю коефіцієнти нормованих ортогональних поліномів Форсайта

$$Q_j(x),\ j = \overline{0,r} \tag{2.50}$$

за значеннями $x_1, \dots, x_n$; l – кількість повторів підпослідовності $x_1, \dots, x_n$ в активному експерименті, що гарантує виконання обмежень на дисперсії $D\hat{\theta}_j,\ j = \overline{2,r}$ (див. підпункт 2.1.1.2).

Примітка 2.10. В цьому випадку використовується набір НОПФ (2.50), а активний експеримент, що складається з l повторів підпослідовності $x_1, \dots, x_n$, дозволяє зменшити дисперсію $D\hat{\theta}_j,\ j = \overline{2,r}$, в l разів в порівнянні з активним експериментом, що складається з послідовності $x_1, \dots, x_n$.

Другий алгоритм реалізує знаходження оцінок $\hat{\theta}_j, j = \overline{2,r}$, для довільного обмеженого повторного активного експерименту, $x \in [c,d]$, l – кількість повторів підпослідовності $x_1, \dots, x_n$ по результатам наступного віртуального активного експерименту:

$$\left(z_i \to \frac{\sum_{k=1}^{l} y_{ki}}{l}, i = \overline{1,n}\right),$$

з використанням формул 2.47–2.49, в яких σ^2 замінюється на $\frac{\sigma^2}{l}$. Очевидно, що в цьому випадку при довільних значеннях $c < d$ використовується лише один заздалегідь побудований заданою точністю набір НОПФ, побудований для значень $z_i,\ i = \overline{1, \mathrm{n}}$ скалярної змінної z.

2.3 Декомпозиційний метод знаходження оцінок при нелінійних коефіцієнтах багатовимірної поліноміальної регресії, заданої надлишковим описом

Розглядається декомпозиційний метод для оцінювання коефіцієнтів при нелінійних членах багатовимірної поліноміальної регресії, що зводить поставлену задачу до оцінки коефіцієнтів при нелінійних членах одновимірних поліноміальних регресій та розв'язання відповідних невироджених систем лінійних рівнянь, змінними яких і є оцінки коефіцієнтів при нелінійних членах багатовимірної поліноміальної регресії.

Викладається по матеріалам статті [44].

Постановка задачі. БПР задана наступним надлишковим описом:

$$Y(\bar{\boldsymbol{x}}) = \sum_{\forall (i_1,\dots,i_t)\in K,\ \forall (j_1,\dots,j_t)\in K(i_1,\dots,i_t)} b_{i_1\dots i_t}^{j_1\dots j_t} \left(x_{i_1}\right)^{j_1} \cdots \left(x_{i_t}\right)^{j_t} + E, \tag{2.51}$$

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

$$x_i \in [c_i,\ d_i],\ c_i > 0,\ i = \overline{1, m}, \tag{2.52}$$

E – випадкова величина, $ME = 0$, $DE = \sigma^2 < \infty$. Значення коефіцієнтів $b_{i_1\dots i_t}^{j_1\dots j_t}$ невідомі (b_0^0 – константа).

Примітка 2.11. Умова $c_i > 0,\ i = \overline{1, m}$, відповідає найбільш поширеному на практиці випадку. Всі отримані нижче результати тривіально поширюються на довільні дійсні значення чисел $c_i < d_i,\ i = \overline{1, m}$.

Декомпозиційний метод реалізує методологію зведення знаходження оцінок при нелінійних членах БПР (2.51) до послідовної побудови ОПР та розв'язання відповідних систем лінійних рівнянь, змінними яких є оцінки при нелінійних членах БПР (2.51).

Загальна алгоритмічна схема отримання оцінок при нелінійних членах БПР (2.51) складається з двох підалгоритмів.

Агрегована алгоритмічна схема першого підалгоритму. l-й крок ($l \le L_1$, де L_1 – загальна кількість нелінійних складових (2.51), кожна з яких містить хоча б одну скалярну змінну у ступеню, більшому або рівному двом) реалізується для наступного нелінійного члена БПР (2.51), коефіцієнт якого не був оцінений на попередніх кроках першого підалгоритму і який містить скалярну змінну в максимальному ступеню. Позначимо її x_{i_p}. В БПР (2.51) 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} знаходяться відповідно до (2.36–2.39) для $d = d_{i_p}$, $c = c_{i_p}$. В реальному основному експерименті скалярна змінна x_{i_p} приймає значення відповідно до (2.39), а інші скалярні змінні в усіх випробуваннях приймають фіксовані значення. В цьому випадку БПР (2.51) перетворюється в ОПР, а дані основного віртуального експерименту знаходяться по основному реальному експерименту

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

а саме:

$$\left(z_i \to y_i - \sum_{\forall b_{i_1 \dots i_t}^{j_1 \dots j_t} \in \bigcup_{m=1}^{l-1}\{J_m\}} b_{i_1 \dots i_t}^{j_1 \dots j_t} \left(x_{i_1,i}\right)^{j_1} \cdots \left(x_{i_t,i}\right)^{j_t},\ i = \overline{1,n}\right), \tag{2.53}$$

де $\bigcup_{m=1}^{l-1}\{J_m\}$ – множина коефіцієнтів, що з допустимою точністю була оцінена на попередніх кроках першого підалгоритму.

Примітка 2.12. Аналогічно знаходяться дані повторного віртуального експерименту, в якому кількість вихідних даних є $y_i,\ i = \overline{1,kn}$, а вхідні дані основного експерименту повторюються k разів.

Примітка 2.13. Максимальний ступень ОПР дорівнює j_p.

На l-му кроці кількість побудованих ОПР може бути більше однієї, якщо відповідний коефіцієнт (коефіцієнти) першої ОПР виражається (виражаються) лінійно через декілька коефіцієнтів при нелінійних членах БПР (2.51), не оцінених на попередніх кроках першого підалгоритму.

Розв'язком отриманих невироджених систем лінійних рівнянь, праві частини яких є оцінками при нелінійних членах ОПР, є оцінки відповідних коефіцієнтів при нелінійних членах БПР (2.51).

Агрегована схема другого підалгоритму. l-й крок ($l \le L_2$, де L_2 – кількість нелінійних складових (2.51) 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 і має максимальне значення t_l. 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 (2.37) для $c = c_{i_j}$, $d = d_{i_j}$, $j = \overline{1, t_l}$. В основному експерименті змінні x_{i_j}, $j = \overline{1, t_l}$, змінюються відповідно до (2.39). Інші скалярні вхідні змінні в кожному випробуванні приймають фіксовані значення. Дані віртуального основного експерименту визначаються даними основного експерименту

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

а саме:

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

$$\left. - \sum_{\forall b_{i_1 \dots i_t}^{j_1 \dots j_t} \in \bigcup_{m=1}^{l-1} \{G_m\}} \hat{b}_{i_1 \dots i_t}^{1 \dots 1} \prod_{l=1}^{t_l} x_{i_l,i},\ i = \overline{1, n} \right), \qquad (2.54)$$

де K_1 – кількість кроків першого підалгоритму; G_m – множина коефіцієнтів, оцінених з достатньою точністю на попередніх кроках другого підалгоритму.

Примітка 2.14. Аналогічно знаходяться дані для повторного віртуального експерименту.

Примітка 2.15. Максимальний ступень ОПР дорівнює t_l.

На завершення приведемо обґрунтування кількості випробувань основного експерименту n та вибору значень $z_i, i = \overline{1,n}$ віртуальної скалярної змінної z. Як виявилось, при відповідному вибору віртуальної скалярної змінної z суттєво спрощується попередній аналіз і формування вхідних даних основного активного експерименту для отримання з допустимою точністю оцінок коефіцієнтів при нелінійних членах БПР. Це досягається завдяки компромісу між мінімумом кількістю випробувань основного експерименту n, числом $z_n - z_1$ (тим більше a, при заданих d та c і достатньо малою величиною дисперсії оцінок коефіцієнтів при нелінійних членах ОПР. Як наслідок аналізу проведених експериментів, для r_{max} пропонується наступне компромісне рішення: $n = 10$, $z_1 = -50$, $z_{10} = 50$, $\Delta z = (z_i - z_{i-1}) = const$. В цьому випадку

$$D\hat{\gamma}_2 = 4.26 \cdot 10^{-6}\sigma^2, D\hat{\gamma}_3 = 7.55 \cdot 10^{-9}\sigma^2,$$
$$D\hat{\gamma}_4 = 1.4 \cdot 10^{-12}\sigma^2, D\hat{\gamma}_5 = 1.28 \cdot 10^{-15}\sigma^2, \quad (2.55)$$

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

2.3.1 Класи надлишкових описів, для яких повністю чи частково з допустимою точністю оцінюються коефіцієнти при нелінійних членах БПР (2.51)

Клас 1. Надлишковий опис (2.51) задовольняє наступним трьом умовам.

1) $\forall\left(x_{i_l}\right)^{j_l},\ j_l \geq 2$ може входити лише в одну складову (2.51);

2) для будь-яких двох нелінійних складових (2.51) з коефіцієнтами $b_{i_1 \ldots i_{t_i}}^{j_1 \ldots j_{t_i}},\ b_{l_1 \ldots l_{t_l}}^{p_1 \ldots p_{t_l}}$ виконується

$$\{i_1, \ldots, i_{t_i}\} \neq \{l_1, \ldots, l_{t_l}\}; \quad (2.56)$$

3) $$0 \in [c_i,\ d_i], i = \overline{1,m}, \quad (2.57)$$

де m – кількість вхідних змінних.

Тоді l -й крок першого підалгоритму декомпозиційного методу, що реалізується для вхідної скалярної змінної $\left(x_{i_p}\right)^{j_p},\ j_p \geq 2$, нелінійної складової (2.51)

$$b_{i_1\dots i_t}^{j_1\dots j_t}\left(x_{i_1}\right)^{j_1}\cdots\left(x_{i_t}\right)^{j_t}, \tag{2.58}$$

повинен задовольняти наступним додатковим умовам. В усіх випробуваннях активного експерименту в кожній складовій (2.51) $b_{i_1\dots i_t}^{j_1\dots j_t}\left(x_{i_1}\right)^{j_1}\cdots\left(x_{i_t}\right)^{j_t}$, $b_{i_1\dots i_t}^{j_1\dots j_t}\in \bigcup_{m=1}^{l-1}\{J_m\}$ (див. (2.53)), одна вхідна скалярна змінна, індекс якої не входить в множину $\left\{i_1,\dots,i_{t_i}\right\}$ (2.58), дорівнює нулю. В цьому випадку лінійне рівняння для оцінки $b_{i_1\dots i_t}^{j_1\dots j_t}$ (2.58) має вигляд

$$b_{i_1\dots i_t}^{j_1\dots j_t}\left(a_{i_p}\right)^{j_p}\prod_{\substack{m=1\\ m\neq p}}^{t}x_{i_m}^{\Phi}=\hat{\gamma}_{j_p}=\gamma_{j_p}\pm\left|\varepsilon_{\hat{\gamma}_{j_p}}\right|, \tag{2.59}$$

де $\hat{\gamma}_{j_p}$ – оцінка γ_{j_p} – коефіцієнта відповідної віртуальної ОПР (в залежності від змісту є реалізацією чи випадковою величиною), а $\varepsilon_{\hat{\gamma}_{j_p}}$ – реалізація випадкової величини, у якої математичне очікування дорівнює нулю, а дисперсія дорівнює $D\hat{\gamma}_{j_p}$ (якщо $\hat{\gamma}_{j_p}$ вважати випадковою величиною). Нехай $\varepsilon(\sum_{l=1}^{t}j_l)>0$ (це – експертно задана вища границя значення $\left|b_{i_1\dots i_t}^{j_1\dots j_t}\right|$, нижче якої відповідна складова (2.58) як не суттєва виключається з надлишкового опису (2.51)). Тоді значення $x_{i_m}^{\Phi}$ $\forall i_m, i_m\neq p$ кладуть рівними $x_{i_m}^{\Phi}=d_{i_m}>0, m\neq p$, і, таким чином, кількість повторів основного експерименту, яка з використанням закону трьох сигм з відповідною ймовірністю гарантує виконання нерівності

$$\left|\varepsilon_{\hat{\gamma}_{j_p}}\right|\leq 10^{-1}\left(a_{i_p}\right)^{j_p}\varepsilon(\textstyle\sum_{l=1}^{t}j_l)\prod_{\substack{m=1\\ m\neq p}}^{t}d_{i_m}, \tag{2.60}$$

є мінімально можливою.

Примітка 2.16. Для отримання оцінки (2.60) закон трьох сигм застосовується для випадкової величини $\hat{\gamma}_{j_p}$, по реалізації якої знаходять оцінку коефіцієнта $\hat{b}_{i_1\dots i_t}^{j_1\dots j_t}$ (2.58).

Якщо

$$\left|\hat{\gamma}_{j_p}\right| < \left(a_{i_p}\right)^{j_p} \varepsilon(\sum_{l=1}^{t} j_l) \prod_{\substack{m=1 \\ m\neq p}}^{t} d_{i_m}, \tag{2.61}$$

то складова (2.58) виключається з надлишкового опису (2.51). В протилежному випадку знайшли оцінку коефіцієнта складової (2.58) з допустимою точністю (2.60).

l-й крок другого підалгоритму декомпозиційного методу, що реалізується для складової

$$b_{i_1\dots i_t}^{1\dots 1} \prod_{l=1}^{t} x_{i_l},\ t = max, \tag{2.62}$$

повинен задовольняти наступним додатковим умовам. В усіх випробуваннях активного експерименту в кожній складовій (2.51) $b_{i_1\dots i_t}^{j_1\dots j_t}\left(x_{i_1}\right)^{j_1}\cdots\left(x_{i_t}\right)^{j_t}$, $b_{i_1\dots i_t}^{j_1\dots j_t} \in \left\{\bigcup_{m=1}^{K_1}\{J_m\}\right\} \cup \left\{\bigcup_{m=1}^{l-1}\{G_m\}\right\}$ (див. (2.54)), одна вхідна скалярна змінна, індекс якої не належить множині $\{i_1, \dots, i_t\}$ (2.62), дорівнює нулю. Лінійне рівняння для оцінки $b_{i_1\dots i_t}^{1\dots 1}$ (2.62) має вигляд

$$b_{i_1\dots i_t}^{1\dots 1} \prod_{l=1}^{t} a_{i_l} = \hat{\gamma}_t = \gamma_t \pm \left|\varepsilon_{\hat{\gamma}_t}\right|, \tag{2.63}$$

де $\hat{\gamma}_t$ – оцінка коефіцієнта γ_t відповідної віртуальної ОПР. Нехай кількість повторів основного експерименту є допустимою для виконання на основі закону трьох сигм обмеження

$$\left|\varepsilon_{\hat{\gamma}_t}\right| \le 10^{-1}\varepsilon(t) \prod_{l=1}^{t} a_{i_l}. \tag{2.64}$$

Тоді, якщо

$$|\hat{\gamma}_t| < \varepsilon(t) \prod_{l=1}^{t} a_{i_l}, \tag{2.65}$$

то складова (2.62) виключається з надлишкового опису (2.51). В протилежному випадку знайшли оцінку коефіцієнта $b_{i_1\dots i_t}^{1\dots 1}$ (2.62) з допустимою точністю (2.64).

Таким чином, при реалізовуваності всіх повторних експериментів декомпозиційний метод дозволяє з допустимою точністю знаходити всі оцінки при нелінійних членах БПР (2.51).

Наслідок. Якщо для деяких кроків першого чи другого підалгоритмів необхідна кількість повторів основного експерименту є нереалізовуваною, то відповідні коефіцієнти не оцінюються, а умови (2.54), (2.55) дозволяють такі складові надлишкового опису (2.51) виключати на інших кроках першого та другого підалгоритмів.

Примітка 2.17. Кожен крок першого та другого підалгоритмів оцінює лише один коефіцієнт БПР (2.51).

Клас 2. Надлишковий опис (2.51) задається чотирма умовами.

Перша умова: всі вхідні змінні, що входять в складові надлишкового опису (2.51) виду $b_{i_1\dots i_t}^{j_1\dots j_t}\left(x_{i_1}\right)^{j_1}\cdots\left(x_{i_t}\right)^{j_t}$, $\sum_{l=1}^{t} j_l \geq 2$, мають область допустимих значень $\left[1,\ d_{i_l}\right], l=\overline{1,t}$.

Друга умова: якщо довільна вхідна змінна входить в довільну складову (2.51) в ступеню, більшому або рівному двом, вона більше не входить в жодну іншу складову (2.51) $b_{i_1\dots i_t}^{j_1\dots j_t}\left(x_{i_1}\right)^{j_1}\cdots\left(x_{i_t}\right)^{j_t}$, $\sum_{l=1}^{t} j_l \geq t+1$.

Третя умова: всі нелінійні складові виду $b_{i_1\dots i_t}^{1\dots 1}\prod_{l=1}^{t} x_{i_l}$не мають спільних вхідних змінних.

Четверта умова: жодна вхідна змінна, що входить в будь-яку складову (2.51), у якої $\exists j_p \geq 2$, не входить в жодну складову (2.51), коефіцієнт якої має вигляд $b_{i_1\dots i_t}^{1\dots 1}$, $t \geq 2$.

Нехай для довільного кроку першого і другого підалгоритму кількість повторів основного експерименту є реалізовуваною для отримання (з використанням закону трьох сигм) оцінок:

а) для першого підалгоритму:

$$\left|\varepsilon_{\widehat{\gamma}_p}\right| \leq 10^{-1}\left(a_{i_p}\right)^{j_p}\varepsilon\left(\sum_{l=1}^{t} j_l\right); \tag{2.66}$$

б) для другого підалгоритму: оцінки (2.64).

Тоді, якщо при реалізації довільного кроку першого та другого підалгоритмів фіксовані вхідні змінні, що входять в нелінійні члени БПР в виразах (2.53), (2.54) покласти рівними одиниці, то в виразах (2.53), (2.54) від значень y_i будуть відніматися величини $\sum_{\forall(\cdot)} \hat{b}_{i_1 \dots i_t}^{j_1 \dots j_t}$, які по модулю будуть відрізнятися від величин $\sum_{\forall(\cdot)} b_{i_1 \dots i_t}^{j_1 \dots j_t}$, в силу оцінок (2.64), (2.66) та грубості закону трьох сигм, на значення, якими практично можна нехтувати. Тоді з допустимою точністю знаходяться всі оцінки при нелінійних членах БПР (2.51).

Примітка 2.18. Кількість повторів основного експерименту може бути реальною, якщо $a_i > {}^1/_3$, $i = \overline{1,m}$. Це також вірно для другого підалгоритму (перший частковий випадок).

Примітка 2.19. Кількість повторів основного експерименту на деяких кроках першого підалгоритму може бути суттєво меншою, якщо для цього кроку оцінка для $\left|\varepsilon_{\hat{\gamma}_p}\right|$ має вигляд типу (2.60). Тобто, деякі вхідні змінні, що не входять в (2.53), (2.54), можуть приймати максимальні значення.

Примітка 2.20. Якщо виключити третю та четверту умови, що накладаються на надлишковий опис (2.51), то декомпозиційний метод реалізує лише перший підалгоритм.

Примітка 2.21. Кожен крок декомпозиційного методу для надлишкового опису (2.51), що задовольняє чотирьом або першим двом умовам (другий частковий випадок), оцінює лише один коефіцієнт надлишкового опису (2.51).

Узагальнення другого часткового випадку. Надлишковий опис (2.51) задовольняє першій умові (перший частковий випадок). $c_i < d_i$, $c_i > 0$, $i = \overline{1,m}$, – довільні числа. Друга умова задається за результатами реалізації наступної версії декомпозиційного методу. При виконанні l-го ($l \geq 2$) кроку першого підалгоритму для члена $b_{i_1 \dots i_{t_l}}^{j_1 \dots j_{t_l}} \left(x_{i_1}\right)^{j_1} \cdots \left(x_{i_{t_l}}\right)^{j_{t_l}}$ фіксовані значення вхідних змінних в (2.53), що не входять в множину $\left\{x_{i_1}, \dots, x_{i_{t_l}}\right\}$, кладуться по модулю мінімально можливими. При реалізації

довільного кроку другого підалгоритму всі фіксовані змінні, що входять в (2.54), приймають по модулю мінімальні значення.

Нехай для кожного кроку першого та другого підалгоритмів кількість повторів основного експерименту є реалізовуваною для виконання обмежень (2.60), (2.64). Тоді другою умовою, що накладається на надлишковий опис (2.51), є

$$\forall \left|\hat{b}_{i_1\dots i_t}^{j_1\dots j_t}\right| \geq \varepsilon(\textstyle\sum_{l=1}^{t} j_l)\cdot 10^{p},\ p \geq 2,$$

або

$$\left|\hat{b}_{i_1\dots i_t}^{j_1\dots j_t}\right| < \varepsilon(\textstyle\sum_{l=1}^{t} j_l).$$

Очевидно, що для будь-яких реальних значень $c_i < d_i, i = \overline{1,m}, r_{max}$ (максимальна розмірність віртуальної ОПР) існує таке достатньо мале значення натурального p, що заміна в виразах (2.53), (2.54) точних значень $b_{i_1\dots i_t}^{j_1\dots j_t}$ на їх оцінки статистично гарантовано практично не впливає на значення оцінок коефіцієнтів при нелінійних членах БПР (2.51).

Примітка 2.22. На кожному кроці декомпозиційного алгоритму оцінюється лише один коефіцієнт БПР (2.51).

Клас 3. Розмірність систем лінійних рівнянь більша, ніж 1. В цьому випадку дисперсії оцінок коефіцієнтів при нелінійних членах БПР, що дозволяють зробити висновок, задовольняють вони чи ні заданій точності, в загальному вигляді виражаються наступним чином:

$$\boldsymbol{A}\begin{pmatrix}\hat{b}_{j_1}\\ \vdots \\ \hat{b}_{j_k}\end{pmatrix} = \begin{pmatrix}\hat{\gamma}_{l_1}\\ \vdots \\ \hat{\gamma}_{l_k}\end{pmatrix}, \tag{2.67}$$

де $\hat{b}_{j_i}, i = \overline{1,k}$, – оцінки коефіцієнтів БПР, як випадкові величини, $\hat{\gamma}_{l_i}, i = \overline{1,k}$, – незалежні віртуальні копії $\hat{\gamma}_l$ – оцінки l-го коефіцієнта віртуальної ОПР, як випадкової величини. Тоді

$$\hat{b}_{j_l} = {a_{i1}}^{-1}\hat{\gamma}_{l_1} + {a_{i2}}^{-1}\hat{\gamma}_{l_2} + \cdots + {a_{ik}}^{-1}\hat{\gamma}_{l_k}; \quad (2.68)$$

$$D\hat{b}_{j_l} = D\hat{\gamma}_l \sum_{j=1}^{n}\left({a_{ij}}^{-1}\right)^2. \quad (2.69)$$

Клас 4. Надлишковий опис БПР (2.51) є довільним, випадкова величина $E = 0$. Такий випадок зустрічається, коли E – це похибка вимірювання, і нею нехтують при достатній точності вимірювання. В цьому випадку загальна алгоритмічна процедура декомпозиційного методу точно знаходить значення всіх коефіцієнтів при нелінійних членах (2.51) і допускає наступні спрощення:

– загальна кількість випробувань активного експерименту для побудови кожного одновимірного полінома не перевищує $r_{\max}$;
– цілком слушно використовувати лише один набір НОПФ;
– значення коефіцієнтів при лінійних членах багатовимірного полінома, заданого надлишковим описом, знаходиться аналогом алгоритмічної процедури пункту 2.2.1 з використанням НОПФ нульового та першого ступенів.

Точні значення коефіцієнтів при лінійних членах, включаючи константу, знаходяться звичайною інтерполяційною процедурою, що полягає в розв'язку системи лінійних рівнянь з кількістю змінних $m + 1$, де m – кількість вхідних змінних.

Перевага декомпозиційного методу над довільним методом інтерполяції полягає в тому, що коефіцієнти при нелінійних членах надлишкового опису знаходяться з використанням лише одного набору НОПФ, знайдених із заданою точністю, і в необхідності розв'язання невироджених систем лінійних рівнянь, розмірність яких суттєво менше кількості нелінійних складових (2.51), а в більшості випадків їх розмірність дорівнює одиниці.

Клас 5. Надлишковий опис (2.51) довільний. В загальному випадку детальна формалізація першого та другого підалгоритмів для знаходження оцінок з достатньою точністю є неефективною. Візуальний аналіз конкретного надлишкового опису (2.51) з використанням наведених вище теоретичних результатів дозволяє для кожного конкретного випадку створити ефективну послідовність кроків індивідуального

алгоритму декомпозиційного методу (який може бути кроком першого чи другого підалгоритму в довільному порядку), який приводить до оцінки з допустимою точністю максимальної кількості коефіцієнтів при нелінійних членах БПР (2.51).

2.4 Синтетичний метод побудови багатовимірної поліноміальної регресії, заданої надлишковим описом

Синтетичний метод побудови БПР, заданої надлишковим описом, є органічним синтезом декомпозиційного методу та модифікованого методу групового урахування аргументів, які реалізуються у вказаній послідовності. А саме, спочатку задача розв'язується декомпозиційним методом і знаходяться всі можливі оцінки коефіцієнтів при лінійних членах БПР з наперед заданою точністю. Пошук шуканої структури БПР та оцінки її коефіцієнтів виконуються модифікованим методом групового урахування аргументів. Якщо реалізується лише пасивний експеримент, то синтетичний метод перетворюється в модифікований метод групового урахування аргументів.

Формальний опис декомпозиційного методу полягає в наступному.

Внаслідок реалізації декомпозиційного методу отримали множину $\{J\}$ коефіцієнтів при нелінійних членах БПР (2.51), оцінених з допустимою точністю.

Примітка 2.23. Несуттєві коефіцієнти виключені з надлишкового опису (2.51) та множини $\{J\}$.

Для отримання остаточного результату використовується ММГУА, викладений в п. 2.1 для побудови багатовимірної лінійної регресії (БЛР), заданої надлишковим описом. Дійсно, задача побудови БПР, заданої надлишковим описом (2.51), звелась до наступної:

$$Y(\bar{\boldsymbol{x}}) = \sum_{\forall\left\{b_{i_1\dots i_t}^{j_1\dots j_t}\right\}\setminus\{J\}} b_{i_1\dots i_t}^{j_1\dots j_t}\left(x_{i_1}\right)^{j_1}\cdots\left(x_{i_t}\right)^{j_t} + f(\bar{x}) + E, \tag{2.70}$$

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

Задача регресії без жодних змін (з точністю до змісту стовбців матриці $\boldsymbol{A}$ в виразі $(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$ загальної формули методу найменших квадратів) може бути розв'язана ММГУА, так як всі невідомі коефіцієнти в вираз (2.70) входять лінійно.

Примітка 2.24. Як буде показано в п. 3.4, використання k повторів основного експерименту суттєво підвищує обчислювальну ефективність ММГУА, а саме, дисперсії оцінок зменшуються в k разів, а в загальній формулі методу найменших квадратів використовуються лише матриці основного експерименту.

Примітка 2.25. Нехай лінгвістична змінна (див. п. 2.1) вказує, що кінцевий результат є недостовірним. Це означає, що коефіцієнти знайдені декомпозиційним методом оцінені достовірно, статистично значимо структура шуканої БПР знайдена ММГУА правильно, а оцінки коефіцієнтів, знайдені за допомогою ММГУА, є недостовірними (цей результат дотично може бути підтверджений величиною дисперсій знайдених оцінок коефіцієнтів ММГУА). Дійсно, розглянемо загальну формулу МНК

$$\hat{\theta} = (A^{\top}A)^{-1}A^{\top}Y,$$

тоді $\mathrm{D}\hat{\theta} = \sigma^2\bar{a}$, де довільний елемент $\bar{a}$ дорівнює сумі квадратів елементів відповідного рядка матриці $(A^{\top}A)^{-1}A^{\top}$, так як Y_i незалежні ВВ, $\mathrm{D}Y_i = \sigma^2,\ i = \overline{1,n}$. При використанні повторних експериментів дисперсія оцінок коефіцієнтів зменшується у відповідну кількість разів.

Примітка 2.26. Як вже було сказано вище, ММГУА може розв'язувати задачу побудови БПР, заданої надлишковим описом, і за результатами пасивного експерименту. В цьому випадку лише зникають переваги активного експерименту – можливість використання повторів основного експерименту.

Ілюстраційний приклад використання синтетичного методу. Задамо надлишковий опис БПР в вигляді:

$$Y(x_1, x_2, x_3, x_4) = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 + b_5x_3x_4 + b_6x_1^2 +$$
$$+b_7x_1x_2x_3x_4 + b_8x_2^2x_3x_4 + b_9x_1x_2x_4^2 + b_{10}x_1x_2^2x_3^2 + b_{11}x_1x_2x_3x_4^3 + E,$$

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

Істинні значення коефіцієнтів наступні:

$$b_0 = 2, b_1 = 1, b_2 = 0, b_3 = 3, b_4 = 2, b_5 = 0, b_6 = 1{,}5, b_7 = 2{,}3, b_8 = 1{,}5,$$
$$b_9 = 3{,}7, b_{10} = 2{,}8, b_{11} = 1{,}3.$$

Таким чином, в істинному описі БПР відсутні члени з коефіцієнтами b_2, b_5.

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

$$x_1 \in [0; 4], x_2 \in [0; 5], x_3 \in [0; 40], x_4 \in [0; 50].$$

Оцінка коефіцієнтів БПР знаходиться за допомогою синтетичного методу, який полягає в тому, що спочатку знаходиться за допомогою декомпозиційного методу ті коефіцієнти при нелінійних членах БПР, дисперсія оцінок яких гарантує необхідну точність. Всі інші коефіцієнти будуть знаходитись за допомогою ММГУА.

Примітка 2.27. Коефіцієнти при нелінійних членах БПР будуть знаходитись з точністю до 2 знаку після коми.

Використання декомпозиційного методу. Аналіз прикладу показує, що надлишковий опис задовольняє п'ятому частковому випадку декомпозиційного методу, тобто буде реалізований індивідуальний алгоритм на основі першого та другого підалгоритмів.

Крок 1. Реалізується для нелінійного члену $\mathrm{b}_{11}x_1x_2x_3x_4^3$ з використанням першого підалгоритму $x_4 = \overline{a_4}z + \overline{b_4}$, в яких $\overline{a_4}, \overline{b_4}$ знаходяться по формулі (2.37), де $z_1 = -50, z_{10} = 50, n = 10, d = 50, c = 0, x_i = \overline{a_4}z_{\mathrm{i}} + \overline{b_4}, i = \overline{1, 10}$,

$$z_1 = -50, z_2 = -38{,}888, z_3 = -27{,}778, z_4 = -16{,}666, z_5 = -5{,}555,$$
$$z_6 = 5{,}555, z_7 = 16{,}666, z_8 = 27{,}778, z_9 = 38{,}888, z_{10} = 50 \qquad (2.71)$$

У всіх випробуваннях вхідні змінні x_1, x_2, x_3 приймають максимальні значення: $x_1 = 4, x_2 = 5, x_3 = 40$. Значення x_4, рівномірно розподілені у діапазоні $[0; 50]$ (2.37). Тоді у всіх випробуваннях БПР перетворюється в ОПР вигляду:

$$Y(z) = b_0 + 4b_1 + 5b_2 + 40b_3 + b_4(0{,}5z + 25) + 40b_5(0{,}5z + 25) + 16b_6 +$$

$$+800b_7(0{,}5z+25)+1000b_8(0{,}5z+25)+20b_9(0{,}5z+25)^2+160000b_{10}+$$
$$+800b_{11}(0{,}5z+25)^3+E=\ \gamma_0+\gamma_1 z+\gamma_2 z^2+\gamma_3 z^3+E, \qquad (2.72)$$

де γ_3=$800b_{11}(0{,}5)^3$ (випливає з виразу (2.72)).

Нормовані ортогональні поліноми Форсайта побудовані для z_i, $i=\overline{1,10}$, а їх коефіцієнти знайдені із точністю до 17 перших розрядів після коми (табл. 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	0,040733377422616100065
q_{j2}			0,00035250853500849863344	0	−0,001293696619986197477 5	0
q_{j3}				0,000013116948944919936589	0	−0,000067433562532976568096
q_{j4}					5,1116893449436679076· 10^{-7}	0
q_{j5}						2,1143486041566852428· 10^{-8}

Знаходимо оцінку γ_3 по наступному віртуальному активному експерименту ($z_i \to y_i$, $i=\overline{1,10}$), де y_i – результати реального активного експерименту ($4,5,40, x_{4,i} \to y_i, i=\overline{1,10}$).

$x_{4,i}=\overline{a_4}z_i+\overline{b_4}$, $i=\overline{1,10}$, $y_i=b_0+4b_1+5b_2+40b_3+b_4x_{4,\text{i}}+40b_5x_{4,\text{i}}+16b_6+800b_7x_{4,\text{i}}+1000b_8x_{4,\text{i}}+20b_9\left(x_{4,\text{i}}\right)^2+160000b_{10}+800b_{11}\left(x_{4,\text{i}}\right)^3+\delta_i$, $i=\overline{1,10}$, де δ_i – реалізації випадкової величини E в активному експерименті (табл. 2.2).

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

Таблиця 2.2 – Реалізації випадкової величини E в активному експерименті 1

δ_i	Значення
δ_1	3,6813677396866846
δ_2	–0,3690862614955922
δ_3	2,232953958531247
δ_4	–2,4334301259602955
δ_5	0,26216210233904186
δ_6	–2,0906029353464386
δ_7	4,177068185832105
δ_8	0,7285350951729221
δ_9	0,010376071214891655
δ_{10}	0,9989603604415067

Таблиця 2.3 – Активний експеримент 1

y_i	Значення
y_1	448153,68136773968668
y_2	647371,37537837850441
y_3	1920989,8137641985
y_4	5339508,767837394
y_5	11972284,345370851
y_6	22888218,548113316
y_7	39159136,615372665
y_8	61856537,063232854
y_9	92043267,71403943
y_{10}	130800250,99896036

Використовуючи НОПФ табл. 2.1, знаходимо оцінку $\widehat{\gamma_3}$ віртуальної одновимірної регресії (2.40, 2.41): $\widehat{\gamma_3} = 129{,}99988$

В загальному вигляді дисперсія $\widehat{\gamma_3} = \sigma^2 4{,}71934 \cdot 10^{-9}$, $M\widehat{\gamma_3} = \gamma_3$, для $\sigma^2 = 4$, дисперсія $D\widehat{\gamma_3} = 1{,}8877 \cdot 10^{-8}$. Оцінка коефіцієнта $\widehat{b_{11}}$ знаходиться з рівняння $\widehat{\gamma_3}$= $800\widehat{b_{11}}(0{,}5)^3$, чи $800b_{11}(0{,}5)^3 = \gamma_3 \pm \left|\varepsilon_{\widehat{\gamma}_3}\right|$, $M\varepsilon_{\widehat{\gamma}_3} = 0$, $D\varepsilon_{\widehat{\gamma}_3} = 1{,}8877 \cdot 10^{-8}$ (2.59). Якщо вважати $\widehat{b_{11}}$ випадковою величиною, то

$$\widehat{b_{11}} = b_{11} + \frac{\varepsilon_{\widehat{\gamma}_3}}{800 \cdot (0{,}5)^3}$$

Звідки $M\widehat{b_{11}} = b_{11}, D\widehat{b_{11}} = \left(\frac{1}{800 \cdot (0{,}5)^3}\right)^2 \cdot 18{,}87736 \cdot 10^{-9} = 1{,}8877 \cdot 10^{-12}$

Таким чином, $D\widehat{b_{11}}$ на основі закону трьох сігм, гарантує задану точність оцінки коефіцієнтів b_{11}.

$$\widehat{b_{11}} = \frac{\widehat{\gamma_3}}{800 \cdot (0{,}5)^3} = 1{,}2999988 \approx 1{,}3.$$

Крок 2. Реалізується для нелінійного члену $b_9 x_1 x_2 x_4^2$ з використанням першого підалгоритму $x_4 = \overline{a_4} z + \overline{b_4}$, в яких $\overline{a_4}, \overline{b_4}$ знаходяться по формулі (2.37), де $z_1 = -50$, $z_{10} = 50, n = 10, d = 50, c = 0, x_i = \overline{a_4} z_\text{i} + \overline{b_4}, i = \overline{1,10}, z_i, i = \overline{1,10}$ – аналогічні формулі (2.71).

У всіх випробуваннях вхідні змінні x_1, x_2 приймають максимальні значення: $x_1 = 4, x_2 = 5$, а $x_3 = 1$. Тоді у всіх випробуваннях БПР перетворюється в ОПР вигляду: $Y(z) = b_0 + 4b_1 + 5b_2 + 1b_3 + b_4(0{,}5z + 25) + 1b_5(0{,}5z + 25) + 16b_6 + 20b_7(0{,}5z + 25) + 25b_8(0{,}5z + 25) + 20b_9(0{,}5z + 25)^2 + 100b_{10} + E = \gamma_0 + \gamma_1 z + \gamma_2 z^2 + E$, нелінійний член $b_{11} x_1 x_2 x_3 x_4^3$ не входить у віртуальну ОПР у відповідності із загальною теорією (2.53), тобто в активному віртуальному експерименті від значень y_i віднімається $26\left(x_{4,\text{i}}\right)^3$, $i = \overline{1,10}$.

$$\gamma_2 = 20b_9(0{,}5)^2$$

Нормовані ортогональні поліноми Форсайта побудовані для z_i, $i = \overline{1,10}$ (табл. 2.1).

Знаходимо оцінку γ_2 по наступному віртуальному активному експерименту ($z_i \to y_i - 26\left(x_{4,\mathrm{i}}\right)^3$, $i = \overline{1,10}$), де y_i результати реального активного експерименту ($4{,}5{,}1, x_{4,i} \to y_i$, $i = \overline{1,10}$).

$x_{4,i} = \overline{a_4} z_i + \overline{b_4}$, $i = \overline{1,10}$, $y_i = b_0 + 4b_1 + 5b_2 + b_3 + b_4 x_{4,\mathrm{i}} + b_5 x_{4,\mathrm{i}} + 16b_6 + 20b_7 x_{4,\mathrm{i}} + 25b_8 x_{4,\mathrm{i}} + 20b_9\left(x_{4,\mathrm{i}}\right)^2 + 100b_{10} + 20b_{11}\left(x_{4,\mathrm{i}}\right)^3 + \delta_i$, $i = \overline{1,10}$, де δ_i – реалізації випадкової величини E в активному експерименті (табл. 2.4).

Таблиця 2.4 – Реалізації випадкової величини E в активному експерименті 2

δ_i	Значення
δ_1	0,14515038220642715
δ_2	–2,0948381928268938
δ_3	–5,732192147862284
δ_4	–0,7948366145413881
δ_5	–0,22319536014804062
δ_6	–5,399029252319972
δ_7	–0,16405615327362114
δ_8	–6,262947538379162
δ_9	–1,30248045484373
δ_{10}	4,973375837977027

А самі значення активного експерименту $y_i, i = \overline{1,10}$ прийматимуть такі значення (табл. 2.5).

Таблиця 2.5 – Активний експеримент 2

$y_i - 26\left(x_{4,i}\right)^3$	Значення
$y_1 - 26\left(x_{4,1}\right)^3$	313,14515038220642715
$y_2 - 26\left(x_{4,2}\right)^3$	3070,2592258071731062
$y_3 - 26\left(x_{4,3}\right)^3$	10392,878061852137716

Кінець табл. 2.5.

$y_i - 26(x_{4,i})^3$	Значення
$y_4 - 26(x_{4,4})^3$	22293,611449385458614
$y_5 - 26(x_{4,5})^3$	38756,924017139851954
$y_6 - 26(x_{4,6})^3$	59780,200683247680023
$y_7 - 26(x_{4,7})^3$	85383,385229846726367
$y_8 - 26(x_{4,8})^3$	115545,96630646162079
$y_9 - 26(x_{4,9})^3$	150281.57558354515629
$y_{10} - 26(x_{4,10})^3$	189592,97337583797703

Використовуючи НОПФ табл. 2.1, знаходимо оцінку $\widehat{\gamma_2}$ віртуальної одновимірної регресії (2.40, 2.41): $\widehat{\gamma_2} = 18{,}495428053$.

В загальному вигляді дисперсія $\widehat{\gamma_2} = \sigma^2 1{,}797913 \cdot 10^{-6}$, $M\widehat{\gamma_2} = \gamma_2$, для $\sigma^2 = 4$, дисперсія $D\widehat{\gamma_2} = 7{,}192 \cdot 10^{-6}$. Оцінка коефіцієнта $\widehat{b_9}$ знаходиться з рівняння $\widehat{\gamma_2} = 20\widehat{b_9}(0{,}5)^2$, чи $20b_9(0{,}5)^2 = \gamma_2 \pm |\varepsilon_{\widehat{\gamma}_2}|$, $M\varepsilon_{\widehat{\gamma}_2} = 0$, $D\varepsilon_{\widehat{\gamma}_2} = 7{,}19165 \cdot 10^{-6}$(2.59). Якщо вважати $\widehat{b_9}$ випадковою величиною, то

$$\widehat{b_9} = b_9 + \frac{\varepsilon_{\widehat{\gamma}_2}}{20 \cdot (0{,}5)^2},$$

звідки $M\widehat{b_9} = b_9, D\widehat{b_9} = \left(\frac{1}{20\cdot(0{,}5)^2}\right)^2 \cdot 7{,}19165 \cdot 10^{-6} = 2{,}87666 \cdot 10^{-7}$.

Таким чином, $D\widehat{b_9}$ на основі закону трьох сігм, гарантує задану точність оцінки коефіцієнтів b_9.

$$\widehat{b_9} = \frac{\widehat{\gamma_2}}{20\cdot(0{,}5)^2} = 3.699085611 \approx 3{,}7.$$

Крок 3. Реалізується для нелінійного члену $b_{10}x_1x_2^2x_3^2$ з використанням першого підалгоритму $x_3 = \overline{a_3}z + \overline{b_3}$, в яких $\overline{a_3}, \overline{b_3}$ знаходяться по формулі 2.37, де $z_1 = -50$,

$z_{10} = 50, n = 10, d = 40, c = 0, x_i = \overline{a_3} z_{\text{i}} + \overline{b_3}, i = \overline{1,10}, z_i, i = \overline{1,10}$ – аналогічні формулі (2.71).

У всіх випробуваннях вхідні змінні x_1, x_2 приймають максимальні значення: $x_1 = 4, x_2 = 5$, а $x_4 = 1$. Тоді у всіх випробуваннях БПР перетворюється в ОПР вигляду: $Y(z) = b_0 + 4b_1 + 5b_2 + b_3(0{,}4z + 20) + b_4 + b_5(0{,}4z + 20) + 16b_6 + +20b_7(0{,}4z + 20) + 25b_8(0{,}4z + 20) + 100b_{10}(0{,}4z + 20)^2 + E = \gamma_0 + \gamma_1 z + +\gamma_2 z^2 + E$, нелінійні члени $b_9 x_1 x_2 x_4^2$ та $b_{11} x_1 x_2 x_3 x_4^3$ не входять у віртуальну ОПР у відповідності із загальною теорією (2.53), тобто в активному віртуальному експерименті від значень y_{i} віднімається $(74 + 26x_{3,\text{i}})$, $i = \overline{1,10}$.

$$\gamma_2 = 100 b_{10} (0{,}4)^2.$$

Нормовані ортогональні поліноми Форсайта побудовані для z_i, $i = \overline{1,10}$ (табл. 2.1).

Знаходимо оцінку γ_2 по наступному віртуальному активному експерименту $(z_i \to y_i - (74 + 26x_{3,\text{i}}), i = \overline{1,10})$, де y_i результати реального активного експерименту $(4{,}5, x_{3,i}, 1 \to y_i, i = \overline{1,10})$.

$x_{3,i} = \overline{a_3} z_i + \overline{b_3}$, $i = \overline{1,10}$, $y_i = b_0 + 4b_1 + 5b_2 + b_3 x_{3,\text{i}} + b_4 + b_5 x_{3,\text{i}} + 16b_6 + 20b_7 x_{3,\text{i}} + 25b_8 x_{3,\text{i}} + 20b_9 + 100b_{10}\left(x_{3,\text{i}}\right)^2 + 20b_{11} x_{3,\text{i}} + \delta_i$, $i = \overline{1,10}$, де δ_i – реалізації випадкової величини E в активному експерименті (табл. 2.6).

Таблиця 2.6 – Реалізації випадкової величини E в активному експерименті 3

δ_i	Значення
δ_1	1,3329764342402861
δ_2	3,0868907541223654
δ_3	2,439771468810724
δ_4	4,249628942178199
δ_5	–1,6814905564565872
δ_6	–1,3825295533142103
δ_7	–6,977325037535887

Кінець табл. 2.6.

δ_i	Значення
δ_8	–1,6766250667722704
δ_9	1,0831798287055863
δ_{10}	–4,5405701805301035

А самі значення активного експерименту $y_i, i = \overline{1,10}$ прийматимуть такі значення (табл. 2.7).

Таблиця 2.7 – Активний експеримент 3

$y_i - (74 + 26x_{3,\mathrm{i}})$,	Значення
$y_1 - (74 + 26x_{3,1})$,	33,3329764342402861
$y_2 - (74 + 26x_{3,2})$,	5951,3112619541223654
$y_3 - (74 + 26x_{3,3})$,	22926,335294668810724
$y_4 - (74 + 26x_{3,4})$,	50969,374937742178199
$y_5 - (74 + 26x_{3,5})$,	90064,155029443543413
$y_6 - (74 + 26x_{3,6})$,	140221,65999044668579
$y_7 - (74 + 26x_{3,7})$,	201438,79518376246411
$y_8 - (74 + 26x_{3,8})$,	273735,33649813322773
$y_9 - (74 + 26x_{3,9})$,	357076,83715102870559
$y_{10} - (74 + 26x_{3,10})$,	451487,4594298194699

Використовуючи НОПФ табл. 2.1, знаходимо оцінку $\widehat{\gamma_2}$ віртуальної одновимірної регресії (2.40, 2.41): $\widehat{\gamma_2} = 44{,}805731083$.

В загальному вигляді дисперсія $\widehat{\gamma_2} = \sigma^2 1{,}797913 \cdot 10^{-6}$, $M\widehat{\gamma_2} = \gamma_2$, для $\sigma^2 = 4$, дисперсія $D\widehat{\gamma_2} = 7{,}192 \cdot 10^{-6}$.

Оцінка коефіцієнта $\widehat{b_9}$ знаходиться з рівняння $\widehat{\gamma_2} = 100\widehat{b_{10}}(0{,}4)^2$, чи $100b_{10}(0{,}4)^2 = \gamma_2 \pm \left|\varepsilon_{\widehat{\gamma}_2}\right|$, $M\varepsilon_{\widehat{\gamma}_2} = 0$, $D\varepsilon_{\widehat{\gamma}_2} = 7{,}19165 \cdot 10^{-6}$ (2.59). Якщо вважати $\widehat{b_{10}}$ випадковою величиною, то

$$\widehat{b_{10}} = b_{10} + \frac{\varepsilon_{\widehat{\gamma}_2}}{100 \cdot (0{,}4)^2}$$

Звідки $M\widehat{b_{10}} = b_{10}, D\widehat{b_{10}} = \left(\frac{1}{100\cdot(0{,}4)^2}\right)^2 \cdot 7{,}19165 \cdot 10^{-6} = 2{,}809238 \cdot 10^{-8}$.

Таким чином $D\widehat{b_{10}}$ на основі закону трьох сігм, гарантує задану точність оцінки коефіцієнтів b_{10}.

$$\widehat{b_{10}} = \frac{\widehat{\gamma_2}}{100\cdot(0{,}4)^2} = 2{,}8003582 \approx 2{,}8.$$

Крок 4. Реалізується для нелінійного члену $b_8 x_2^2 x_3 x_4$ з використанням першого підалгоритму $x_2 = \overline{a_2} z + \overline{b_2}$, в яких $\overline{a_2}, \overline{b_2}$ знаходяться по формулі 2.37, де $z_1 = -50$, $z_{10} = 50$, $n = 10$, $d = 5$, $c = 0$, $x_i = \overline{a_2} z_{\text{i}} + \overline{b_2}$, $i = \overline{1,10}$, $z_i, i = \overline{1,10}$ – аналогічні формулі (2.71).

У всіх випробуваннях вхідні змінні x_3, x_4 приймають максимальні значення: $x_3 = 40, x_4 = 50$, а $x_1 = 1$. Тоді у всіх випробуваннях БПР перетворюється в ОПР вигляду:

$$\begin{aligned} Y(z) = b_0 + b_1 + b_2(0{,}05z + 2{,}5) + 40b_3 + 50b_4 + 2000b_5 + b_6 \\ + 2000b_7(0{,}05z + 2{,}5) + 2000b_8(0{,}05z + 2{,}5)^2 + E \\ = \gamma_0 + \gamma_1 z + \gamma_2 z^2 + E, \end{aligned}$$

нелінійні члени $b_9 x_1 x_2 x_4^2$, $b_{10} x_1 x_2^2 x_3^2$ та $b_{11} x_1 x_2 x_3 x_4^3$ не входять у віртуальну ОПР у відповідності із загальною теорією (2.53), тобто в активному віртуальному експерименті від значень y_{i} віднімається $(4480\left(x_{2,\text{i}}\right)^2 + (6500000 + 9250)x_{2,\text{i}})$, $i = \overline{1,10}$.

$$\gamma_2 = 2000b_8(0{,}05)^2.$$

Нормовані ортогональні поліноми Форсайта побудовані для $z_i, i = \overline{1,10}$ (табл. 2.1).

Знаходимо оцінку γ_2 по наступному віртуальному активному експерименту ($z_i \to y_i - (4480\left(x_{2,\text{i}}\right)^2 + 6509250x_{2,\text{i}})$, $i = \overline{1,10}$), де y_i результати реального активного експерименту ($1, x_{2,i}, 40{,}50 \to y_i$, $i = \overline{1,10}$).

$x_{2,i} = \overline{a_2} z_i + \overline{b_2}, i = \overline{1,10}, y_i = b_0 + b_1 + b_2 x_{2,\text{i}} + 40b_3 + 50b_4 + 2000b_5 + b_6 + 2000b_7 x_{2,\text{i}} + 2000b_8 \left(x_{2,\text{i}}\right)^2 + 2500b_9 x_{2,\text{i}} + 1600b_{10} \left(x_{2,\text{i}}\right)^2 + 5000000b_{11} x_{2,\text{i}} + \delta_i, i = \overline{1,10}$, де δ_i – реалізації випадкової величини E в активному експерименті (табл. 2.8).

Таблиця 2.8 – Реалізації випадкової величини E в активному експерименті 4

δ_i	Значення
δ_1	–0,30270246370946247
δ_2	2,5611891759651777
δ_3	0,8960803046570015
δ_4	7,752579576700797
δ_5	–2,368178544430582
δ_6	–1,9123191197727147
δ_7	–0,4080675837736328
δ_8	10,182988418917438
δ_9	–4,535346939047777
δ_{10}	3,8212244462421885

А самі значення активного експерименту $y_i, i = \overline{1,10}$ прийматимуть такі значення (табл. 2.9).

Таблиця 2.9 – Активний експеримент 4

$y_i - (4480\left(x_{2,i}\right)^2 + 6509250x_{2,i})$	Значення
$y_1 - (4480\left(x_{2,1}\right)^2 + 6509250x_{2,1})$	224,19729753629053753
$y_2 - (4480\left(x_{2,2}\right)^2 + 6509250x_{2,2})$	3708,895269175965186
$y_3 - (4480\left(x_{2,3}\right)^2 + 6509250x_{2,3})$	9040,0857103046570437
$y_4 - (4480\left(x_{2,4}\right)^2 + 6509250x_{2,4})$	16232,739249576700677
$y_6 - (4480\left(x_{2,6}\right)^2 + 6509250x_{2,6})$	36147,922868380226924
$y_7 - (4480\left(x_{2,7}\right)^2 + 6509250x_{2,7})$	48889,938602416226247

Кінець табл. 2.9.

$y_i - (4480(x_{2,i})^2 + 6509250x_{2,i})$	Значення
$y_8 - (4480(x_{2,8})^2 + 6509250x_{2,8})$	63494,252618418917314
$y_9 - (4480(x_{2,9})^2 + 6509250x_{2,9})$	79922,278733060951898
$y_{10} - (4480(x_{2,10})^2 + 6509250x_{2,10})$	98228,321224446242189

Використовуючи НОПФ табл. 2.1, знаходимо оцінку $\widehat{\gamma_2}$ віртуальної одновимірної регресії (2.40, 2.41): $\widehat{\gamma_2} = 7{,}503340159$.

В загальному вигляді дисперсія $\widehat{\gamma_2} = \sigma^2 1{,}797913 \cdot 10^{-6}$, $M\widehat{\gamma_2} = \gamma_2$, для $\sigma^2 = 4$, дисперсія $D\widehat{\gamma_2} = 7{,}192 \cdot 10^{-6}$. Оцінка коефіцієнта $\widehat{b_8}$ знаходиться з рівняння $\widehat{\gamma_2} = 2000\widehat{b_8}(0{,}05)^2$, чи $2000b_8(0{,}05)^2 = \gamma_2 \pm |\varepsilon_{\widehat{\gamma}_2}|$, $M\varepsilon_{\widehat{\gamma}_2} = 0$, $D\varepsilon_{\widehat{\gamma}_2} = 7{,}19165 \cdot 10^{-6}$ (2.59). Якщо вважати $\widehat{b_8}$ випадковою величиною, то

$$\widehat{b_8} = b_8 + \frac{\varepsilon_{\widehat{\gamma}_2}}{2000 \cdot (0{,}05)^2},$$

звідки $M\widehat{b_8} = b_8, D\widehat{b_8} = \left(\frac{1}{2000 \cdot (0{,}05)^2}\right)^2 \cdot 7{,}19165 \cdot 10^{-6} = 0{,}287666 \cdot 10^{-8}$.

Таким чином $D\widehat{b_8}$ на основі закону трьох сігм, гарантує задану точність оцінки коефіцієнтів b_8.

$$\widehat{b_8} = \frac{\widehat{\gamma_2}}{2000 \cdot (0{,}05)^2} = 1{,}500668031 \approx 1{,}5.$$

Крок 5. Реалізується для нелінійного члену $b_6 x_1^2$, в цьому випадку в силу загальної теорії першого підалгоритму отримуємо рівняння $\left(\frac{1}{25}\right)^2 b_6 = \gamma_2$ чи $\widehat{b_6} = b_6 + 625\varepsilon_{\gamma_2}$, звідки $D\widehat{b_6} = 390625 \cdot 7{,}19165 \cdot 10^{-6} = 2809238{,}28 \cdot 10^{-6} = 2{,}80923828$. Отримане значення $D\widehat{b_6}$ не дозволяє оцінити значення b_6 з точністю до другого знаку після коми, тому згідно з загальною теорією синтетичного методу нелінійний член $b_6 x_1^2$ входить в вираз БПР, що розв'язується ММГУА.

Крок 6. Реалізується для нелінійного члену $b_7x_1x_2x_3x_4$. В силу загальної теорії оцінка b_7 знаходиться другим підалгоритмом: $x_i = \overline{a_i}z + \overline{b_i}, i = \overline{1,4}$. Отримаємо одновимірну регресію четвертого порядку. В силу теорії другого підалгоритму отримуємо рівняння: $0{,}04 \cdot 0{,}05 \cdot 0{,}4 \cdot 0{,}5 \cdot b_7 = \gamma_4$, звідки $\widehat{b_7} = b_7 + 4 \cdot 10^4 \cdot \varepsilon_{\widehat{\gamma_4}}$, де $\mathrm{D}\widehat{b_7} = 2{,}612937 \cdot 10^{-13} \cdot 10^8 = 2{,}612937 \cdot 10^{-5}$ (2.63).

Але якщо для оцінки b_7 реалізувати алгоритм, що є комбінацією першого та другого підалгоритму, можна отримати оцінку коефіцієнта b_7 з дисперсією на два порядки менше. Дійсно, покладемо $x_1 = \overline{a_1}z + \overline{b_1}$, $x_2 = \overline{a_2}z + \overline{b_2}$, $x_3 = 40$, $x_4 = 50$. Тоді у всіх випробуваннях БПР перетворюється в ОПР вигляду:

$$Y(z) = b_0 + b_1\left(\overline{a_1}z + \overline{b_1}\right) + b_2\left(\overline{a_2}z + \overline{b_2}\right) + 40b_3 + 50b_4 + 2000b_5 + b_6\left(\overline{a_1}z + \overline{b_1}\right)^2 + 2000b_7\left(\overline{a_1}z + \overline{b_1}\right) \cdot \left(\overline{a_2}z + \overline{b_2}\right) + E = \gamma_0 + \gamma_1 z + \gamma_2 z^2 + E,$$

нелінійні члени $b_8x_2^2x_3x_4$, $b_9x_1x_2x_4^2$, $b_{10}x_1x_2^2x_3^2$ та $b_{11}x_1x_2x_3x_4^3$ не входять у віртуальну ОПР у відповідності із загальною теорією (2.54), тобто в активному віртуальному експерименті від значень y_i віднімається $(3000\left(x_{2,\mathrm{i}}\right)^2 + 4480x_{1,\mathrm{i}}\left(x_{2,\mathrm{i}}\right)^2 + (6500000 + 9250)x_{1,\mathrm{i}}x_{2,\mathrm{i}})$, $i = \overline{1,10}$.

$$\gamma_2 = 2000 \cdot 0{,}04 \cdot 0{,}05 \cdot b_7 = 4b_7$$

НОПФ побудовані для z_i, $i = \overline{1,10}$ (табл. 2.1).

Знаходимо оцінку γ_2 по наступному віртуальному активному експерименту ($z_i \to y_i - (3000\left(x_{2,\mathrm{i}}\right)^2 + 4480x_{1,\mathrm{i}}\left(x_{2,\mathrm{i}}\right)^2 + 6509250x_{1,\mathrm{i}}x_{2,\mathrm{i}})$, $i = \overline{1,10}$), де y_i результати реального активного експерименту ($x_{1,i}, x_{2,i}, 40{,}50 \to y_i$, $i = \overline{1,10}$).

$$x_{1,i} = \overline{a_1}z_i + \overline{b_1}, x_{2,i} = \overline{a_2}z_i + \overline{b_2}, i = \overline{1,10}, y_i = b_0 + b_1x_{1,\mathrm{i}} + b_2x_{2,\mathrm{i}} + 40b_3 + 50b_4 + 2000b_5 + b_6\left(x_{1,\mathrm{i}}\right)^2 + 2000b_7x_{1,\mathrm{i}}x_{2,\mathrm{i}} + 2000b_8\left(x_{2,\mathrm{i}}\right)^2 + 2500b_9x_{1,\mathrm{i}}x_{2,\mathrm{i}} + 1600b_{10}x_{1,\mathrm{i}}\left(x_{2,\mathrm{i}}\right)^2 + 5000000b_{11}x_{1,\mathrm{i}}x_{2,\mathrm{i}} + \delta_i, i = \overline{1,10},$$

де δ_i – реалізації випадкової величини E в активному експерименті (табл. 2.10).

Таблиця 2.10 – Реалізації випадкової величини E в активному експерименті 5

δ_i	Значення
δ_1	0,8655571424590244
δ_2	–0,5828394206752016
δ_3	0,2976135204638172
δ_4	–6,56941206042814
δ_5	–6,701432143487233
δ_6	3,6459107167709703
δ_7	–2,962221062624017
δ_8	7,865843842074305
δ_9	3,924907526042418
δ_{10}	–3,2415816478487818

А самі значення активного експерименту $y_i, i = \overline{1,10}$ прийматимуть такі значення (табл. 2.11).

Таблиця 2.11 – Активний експеримент 5

$y_i - (3000(x_{2,i})^2 + 4480x_{1,i}(x_{2,i})^2 + 6509250x_{1,i}x_{2,i})$	Значення
$y_1 - (3000(x_{2,1})^2 + 4480x_{1,1}(x_{2,1})^2 + 6509250x_{1,1}x_{2,1})$	222,8655571424590244
$y_2 - (3000(x_{2,2})^2 + 4480x_{1,2}(x_{2,2})^2 + 6509250x_{1,2}x_{2,2})$	1358,1421890849248322
$y_3 - (3000(x_{2,3})^2 + 4480x_{1,3}(x_{2,3})^2 + 6509250x_{1,3}x_{2,3})$	4767,4906678020637832
$y_4 - (3000(x_{2,4})^2 + 4480x_{1,4}(x_{2,4})^2 + 6509250x_{1,4}x_{2,4})$	10442,061836473972045
$y_5 - (3000(x_{2,5})^2 + 4480x_{1,5}(x_{2,5})^2 + 6509250x_{1,5}x_{2,5})$	18395,111057116512662
$y_6 - (3000(x_{2,6})^2 + 4480x_{1,6}(x_{2,6})^2 + 6509250x_{1,6}x_{2,6})$	28629,769199976771139
$y_7 - (3000(x_{2,7})^2 + 4480x_{1,7}(x_{2,7})^2 + 6509250x_{1,7}x_{2,7})$	41120,441987471775609
$y_8 - (3000(x_{2,8})^2 + 4480x_{1,8}(x_{2,8})^2 + 6509250x_{1,8}x_{2,8})$	55902,13457812367403
$y_9 - (3000(x_{2,9})^2 + 4480x_{1,9}(x_{2,9})^2 + 6509250x_{1,9}x_{2,9})$	72938,347216031637826
$y_{10} - (3000(x_{2,10})^2 + 4480x_{1,10}(x_{2,10})^2 + 6509250x_{1,10}x_{2,10})$	92246,758418352147218

Використовуючи НОПФ табл. 2.1, знаходимо оцінку $\widehat{\gamma_2}$ віртуальної одновимірної регресії (2.40, 2.41): $\widehat{\gamma_2} = 9{,}211362377$.

З структури реальної ОПР випливає рівність:

$$40 \cdot 50 \cdot 0{,}04 \cdot 0{,}05 \cdot b_7 = \gamma_2,$$

звідки:

$$\widehat{b_7} = b_7 + 0{,}25\varepsilon_{\widehat{\gamma_2}},$$
$$D\widehat{b_7} = 0{,}0625 \cdot 7{,}19165 \cdot 10^{-6} = 4{,}4947812 \cdot 10^{-7}.$$

Таким чином, індивідуальний алгоритм для нелінійного члену $b_7 x_1 x_2 x_3 x_4$ реалізував оцінку $\widehat{b_7}$ з дисперсією, що на два порядки менше ніж оцінка по другому підалгоритму, а $D\widehat{b_7}$ на основі закону трьох сігм, гарантує задану точність оцінки коефіцієнтів b_7. Остаточний результат:

$$\widehat{b_7} = \frac{\widehat{\gamma_2}}{2000 \cdot 0{,}04 \cdot 0{,}05} = 2{,}302840 \approx 2{,}3.$$

Крок 7. Реалізується для нелінійного члену $b_5 x_3 x_4$. В силу загальної теорії оцінка b_5 знаходиться другим підалгоритмом: $x_i = \overline{a_i} z + \overline{b_i}, i = \overline{3,4}$. Отримаємо одновимірну регресію другого порядку. В силу теорії другого підалгоритму отримуємо рівняння $0{,}4 \cdot 0{,}5 \cdot b_5 = \gamma_2$, звідки $\widehat{b_5} = b_5 + 20 \cdot \varepsilon_{\widehat{\gamma_2}}$, $D\widehat{b_5} = 400 \cdot D\varepsilon_{\widehat{\gamma_2}} = 28{,}7666 \cdot 10^{-4}$ (2.63).

Отримана $D\widehat{b_5}$ гарантує знаходження оцінку коефіцієнта b_5 з заданою точністю.

Покладемо $x_1 = 1, x_2 = 1\ x_3 = \overline{a_3} z + \overline{b_3}, x_4 = \overline{a_4} z + \overline{b_4}$. Тоді у всіх випробуваннях БПР перетворюється в ОПР вигляду:

$$Y(z) = b_0 + b_1 + b_2 + b_3\left(\overline{a_3} z + \overline{b_3}\right) + b_4\left(\overline{a_4} z + \overline{b_4}\right) + b_5\left(\overline{a_3} z + \overline{b_3}\right) \cdot \left(\overline{a_4} z + \overline{b_4}\right) + \\ + b_6 + E = \gamma_0 + \gamma_1 z + \gamma_2 z^2 + E,$$

нелінійні члени $b_7 x_1 x_2 x_3 x_4$, $b_8 x_2^2 x_3 x_4$, $b_9 x_1 x_2 x_4^2$, $b_{10} x_1 x_2^2 x_3^2$ та $b_{11} x_1 x_2 x_3 x_4^3$ не входять у віртуальну ОПР у відповідності із загальною теорією (2.54), тобто в активному

віртуальному експерименті від значень y_{i} віднімається $((2{,}3+1{,}5)x_{3,\text{i}}x_{4,\text{i}}+3{,}7\left(x_{4,\text{i}}\right)^2+2{,}8\left(x_{3,\text{i}}\right)^2+1{,}3x_{3,\text{i}}\left(x_{4,\text{i}}\right)^3$, $i=\overline{1,10}$.

$$\gamma_2=0{,}4\cdot 0{,}5\cdot b_5=0{,}2b_5$$

Нормовані ортогональні поліноми Форсайта побудовані для $z_i, i=\overline{1,10}$ (табл. 2.1).

Знаходимо оцінку γ_2 по наступному віртуальному активному експерименту ($z_i \to y_i-(3{,}8x_{3,\text{i}}x_{4,\text{i}}+3{,}7\left(x_{4,\text{i}}\right)^2+2{,}8\left(x_{3,\text{i}}\right)^2+1{,}3x_{3,\text{i}}\left(x_{4,\text{i}}\right)^3$, $i=\overline{1,10}$), де y_i результати реального активного експерименту ($1{,}1, x_{3,i}, x_{4,i} \to y_i$, $i=\overline{1,10}$).

$x_{3,i}=\overline{a_3}z_i+\overline{b_3}$, $x_{4,i}=\overline{a_4}z_i+\overline{b_4}$, $i=\overline{1,10}$, $y_i=b_0+b_1+b_2+b_3x_{3,\text{i}}+b_4x_{4,\text{i}}+b_5x_{3,\text{i}}x_{4,\text{i}}+b_6+b_7x_{3,\text{i}}x_{4,\text{i}}+b_8x_{3,\text{i}}x_{4,\text{i}}+b_9\left(x_{4,\text{i}}\right)^2+b_{10}\left(x_{3,\text{i}}\right)^2+b_{11}x_{3,\text{i}}\left(x_{4,\text{i}}\right)^3+\delta_i,\ i=\overline{1,10}$, де δ_i – реалізації випадкової величини E в активному експерименті (табл. 2.12).

Таблиця 2.12 – Реалізації випадкової величини E в активному експерименті 6

δ_i	Значення
δ_1	–0,6993492616826527
δ_2	–1,1734430411281076
δ_3	5,002854684430891
δ_4	3,875042246242133
δ_5	0,5486465062226036
δ_6	6,556577165878645
δ_7	–0,5927403212578017
δ_8	2,749241220472613
δ_9	–2,6373245560877985
δ_{10}	–3,9234335924408295

А самі значення активного експерименту $y_i, i=\overline{1,10}$ прийматимуть такі значення (табл. 2.13).

Таблиця 2.13 – Активний експеримент 6

$y_i - (3{,}8x_{3,i}x_{4,i} + 3{,}7(x_{4,i})^2 + 2{,}8(x_{3,i})^2 + 1{,}3x_{3,i}(x_{4,i})^3$	Значення
$y_1 - (3{,}8x_{3,1}x_{4,1} + 3{,}7(x_{4,1})^2 + 2{,}8(x_{3,1})^2 + 1{,}3x_{3,1}(x_{4,1})^3$	3,8006507383173473
$y_2 - (3{,}8x_{3,2}x_{4,2} + 3{,}7(x_{4,2})^2 + 2{,}8(x_{3,2})^2 + 1{,}3x_{3,2}(x_{4,2})^3$	27,77295695887189243
$y_3 - (3{,}8x_{3,3}x_{4,3} + 3{,}7(x_{4,3})^2 + 2{,}8(x_{3,3})^2 + 1{,}3x_{3,3}(x_{4,3})^3$	58,391254684430890646
$y_4 - (3{,}8x_{3,4}x_{4,4} + 3{,}7(x_{4,4})^2 + 2{,}8(x_{3,4})^2 + 1{,}3x_{3,4}(x_{4,4})^3$	81,709842246242133179
$y_5 - (3{,}8x_{3,5}x_{4,5} + 3{,}7(x_{4,5})^2 + 2{,}8(x_{3,5})^2 + 1{,}3x_{3,5}(x_{4,5})^3$	102,82764650622260502
$y_6 - (3{,}8x_{3,6}x_{4,6} + 3{,}7(x_{4,6})^2 + 2{,}8(x_{3,6})^2 + 1{,}3x_{3,6}(x_{4,6})^3$	133,27757716587864308
$y_7 - (3{,}8x_{3,7}x_{4,7} + 3{,}7(x_{4,7})^2 + 2{,}8(x_{3,7})^2 + 1{,}3x_{3,7}(x_{4,7})^3$	150,57245967874220747
$y_8 - (3{,}8x_{3,8}x_{4,8} + 3{,}7(x_{4,8})^2 + 2{,}8(x_{3,8})^2 + 1{,}3x_{3,8}(x_{4,8})^3$	178,36084122047261198
$y_9 - (3{,}8x_{3,9}x_{4,9} + 3{,}7(x_{4,9})^2 + 2{,}8(x_{3,9})^2 + 1{,}3x_{3,9}(x_{4,9})^3$	197,41627544391215168
$y_{10} - (3{,}8x_{3,10}x_{4,10} + 3{,}7(x_{4,10})^2 + 2{,}8(x_{3,10})^2 + 1{,}3x_{3,10}(x_{4,10})^3$	220,5765664075591305

Використовуючи НОПФ табл. 2.1, знаходимо оцінку $\widehat{\gamma_2}$ віртуальної одновимірної регресії. (2.40, 2.41): $\widehat{\gamma_2} = -2{,}656126256 \cdot 10^{-3}$. Остаточний результат:

$$\widehat{b_5} = \frac{\widehat{\gamma_2}}{0{,}4 \cdot 0{,}5} = -0{,}013280631 \approx 0.$$

Реалізація декомпозиційного методу завершена. Остаточний розв'язок знаходимо з використанням ММГУА. БПР, задана надлишковим описом має вигляд:

$$Y(x_1, x_2, x_3, x_4) = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 + b_6x_1^2 + E \qquad (2.73)$$

Примітка 2.28. Активний експеримент для регресії (2.73) має вигляд:

$$(Y(x_{1,\mathrm{i}}, x_{2,\mathrm{i}}, x_{3,\mathrm{i}}, x_{4,\mathrm{i}}) \to y_i - \widehat{b_5}x_{3,\mathrm{i}}x_{4,\mathrm{i}} + \widehat{b_7}x_{1,i}x_{2,i}x_{3,i}x_{4,i} + \widehat{b_8}x_{2,i}^2x_{3,i}x_{4,i} +$$
$$+\widehat{b_9}x_{1,i}x_{2,i}x_{4,i}^2 + \widehat{b_{10}}x_{1,i}x_{2,i}^2x_{3,i}^2 + \widehat{b_{11}}x_{1,i}x_{2,i}x_{3,i}x_{4,i}^3 = \overline{y_i}, i = \overline{1,n})$$

Нехай фізичне обмеження на загальну кількість випробувань дорівнює 180, звідки кількість випробувань для отримання перевірочної послідовності покладемо 20 (якщо взяти менше, то головна ідея МГУА може не спрацювати, якщо взяти більше – можемо отримати надто неточні оцінки коефіцієнтів часткових описів).

Крок 1. Відповідно до загальної теорії задамо вхідні дані основного активного експерименту таким чином, щоб відповідна матриця в загальній формулі МНК була невироджена $\left(\hat{b} = (A^T A)^{-1} A^T\right)$. В результаті проведених експериментів отримано наступні вхідні дані основного активного експерименту (табл. 2.14) (дані табл. 2.14 належать області вхідних змінних).

Для цих даних $\det(A^T A)^{-1} = 11876{,}181485546$.

Крок 2. Для повторного експерименту (16 повторів) знаходимо оцінки коефіцієнтів БПР (2.73) з використанням загальної формули МНК (результати повторного експерименту знаходяться у Додатку А). Результати повторного активного експерименту з використанням теоретичних властивостей повторного експерименту представлені у табл. 2.15.

Таблиця 2.14 – Значення вхідних незалежних змінних основного активного експерименту

x_1	x_2	x_3	x_4
1,50	3,11	1,60	2,84
1,84	2,44	3,51	3,60
1,09	3,7	3,40	2,57
2,94	3,88	3,25	1,98
1,90	3,49	1,07	2,72
3,67	1,98	3,43	1,69
3,89	1,64	3,73	2,26
2,87	1,38	2,06	2,34
2,55	2,62	1,69	3,31
1,52	2,35	1,67	2,77

Таблиця 2.15 – Активний експеримент 7 ММГУА

$\frac{1}{16}\sum_{n=0}^{15}\bar{y}_{n\cdot 10+i}$	Значення
$\frac{1}{16}\sum_{n=0}^{15}\bar{y}_{n\cdot 10+1}$	17,58251501
$\frac{1}{16}\sum_{n=0}^{15}\bar{y}_{n\cdot 10+2}$	26,38535451
$\frac{1}{16}\sum_{n=0}^{15}\bar{y}_{n\cdot 10+3}$	20,20351704
$\frac{1}{16}\sum_{n=0}^{15}\bar{y}_{n\cdot 10+4}$	31,18789814
$\frac{1}{16}\sum_{n=0}^{15}\bar{y}_{n\cdot 10+5}$	17,73209771
$\frac{1}{16}\sum_{n=0}^{15}\bar{y}_{n\cdot 10+6}$	40,24850628
$\frac{1}{16}\sum_{n=0}^{15}\bar{y}_{n\cdot 10+7}$	43,43495478
$\frac{1}{16}\sum_{n=0}^{15}\bar{y}_{n\cdot 10+8}$	28,70925827
$\frac{1}{16}\sum_{n=0}^{15}\bar{y}_{n\cdot 10+9}$	26,81635256
$\frac{1}{16}\sum_{n=0}^{15}\bar{y}_{n\cdot 10+10}$	17,21031246

Примітка 2.29. Відповідні значення вхідних змінних в кожному повторі експерименту приведені в табл. 2.14. Отримані оцінки коефіцієнтів БПР (2.73) мають вигляд:

$$\widehat{b_0} = 2{,}44378642977085$$
$$\widehat{b_1} = 2{,}25186339930398$$
$$\widehat{b_2} = -0{,}24086150052704$$
$$\widehat{b_3} = 2{,}91881336579502$$
$$\widehat{b_4} = 1{,}72336258113898$$
$$\widehat{b_6} = 1{,}21799145995177$$

Крок 3. Реалізуємо алгоритм кластерного аналізу. Отримали: $\mathrm{M}_1 = \{b_0, b_1, b_3, b_4\}, \mathrm{M}_2 = \{b_2, b_6\}$. Отримали наступні часткові описи:

1) $Y(x_1, x_2, x_3, x_4) = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 + b_6x_1^2 + E$

2) $Y(x_1, x_2, x_3, x_4) = b_0 + b_1x_1 + b_3x_3 + b_4x_4 + b_6x_1^2 + E$

3) $Y(x_1, x_2, x_3, x_4) = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 + E$

4) $Y(x_1, x_2, x_3, x_4) = b_0 + b_1x_1 + b_3x_3 + b_4x_4 + E$

Крок 4. Для всіх часткових описів знаходимо ЗСК по перевірочній послідовності (табл. 2.16).

Таблиця 2.16 – Вхідні дані та перевірочна послідовність

x_1	x_2	x_3	x_4	$\bar{y}_{160+i}$
1,50	3,11	1,60	2,84	20,255760516
1,84	2,44	3,51	3,60	20,362339955
1,09	3,7	3,40	2,57	17,90687809
2,94	3,88	3,25	1,98	32,562706081
1,90	3,49	1,07	2,72	19,691441325
3,67	1,98	3,43	1,69	32,109197368
3,89	1,64	3,73	2,26	46,242541902
2,87	1,38	2,06	2,34	32,578457399
2,55	2,62	1,69	3,31	29,896867381
1,52	2,35	1,67	2,77	17,521046519
1,50	3,11	1,60	2,84	19,651629120
1,84	2,44	3,51	3,60	30,848373595
1,09	3,7	3,40	2,57	12,9751057228
2,94	3,88	3,25	1,98	35,6024975480
1,90	3,49	1,07	2,72	16,0423014135
3,67	1,98	3,43	1,69	42,5231865145
3,89	1,64	3,73	2,26	48,5677360937
2,87	1,38	2,06	2,34	27,4611217819
2,55	2,62	1,69	3,31	22,2666647324
1,52	2,35	1,67	2,77	14,45750491664

ЗСК(1) = 291,570642;

ЗСК(2) = 289,475058;

ЗСК(3) = 297,358969;

ЗСК(4) = 303,482894.

Відповідно до загальної теорії, структура шуканої регресії задається другим частковим описом – він має мінімальну ЗСК та мінімальну кількість членів, ЗСК яких практично не відрізняється від мінімальної ЗСК.

Для знайденої структури БПР знаходимо оцінки коефіцієнтів по всім 18 повторам активного експерименту (табл. 2.17)

Таблиця 2.17 – Дані по 18-ти повторах активного експерименту

$\frac{1}{18}\sum_{n=0}^{17}\bar{y}_{n\cdot 10+i}$	Значення
$\frac{1}{18}\sum_{n=0}^{17}\bar{y}_{n\cdot 10+1}$	17,84597944
$\frac{1}{18}\sum_{n=0}^{17}\bar{y}_{n\cdot 10+2}$	26,2986881
$\frac{1}{18}\sum_{n=0}^{17}\bar{y}_{n\cdot 10+3}$	19,67434758
$\frac{1}{18}\sum_{n=0}^{17}\bar{y}_{n\cdot 10+4}$	31,50953188
$\frac{1}{18}\sum_{n=0}^{17}\bar{y}_{n\cdot 10+5}$	17,74707256
$\frac{1}{18}\sum_{n=0}^{17}\bar{y}_{n\cdot 10+6}$	39,92269358
$\frac{1}{18}\sum_{n=0}^{17}\bar{y}_{n\cdot 10+7}$	43,87608636
$\frac{1}{18}\sum_{n=0}^{17}\bar{y}_{n\cdot 10+8}$	28,85487287
$\frac{1}{18}\sum_{n=0}^{17}\bar{y}_{n\cdot 10+9}$	26,73473184
$\frac{1}{18}\sum_{n=0}^{17}\bar{y}_{n\cdot 10+10}$	17,07464171

Остаточний результат побудови БПР (табл. 2.18):

Таблиця 2.18 – Остаточний результат

Ідеальні значення коефіцієнтів БПР, заданої надлишковим описом	Оцінки коефіцієнтів	Дисперсії оцінок
$b_0 = 2$	$\widehat{b_0} = 0{,}9$	$D\widehat{b_0} = 0{,}775374$

Кінець табл. 2.18.

Ідеальні значення коефіцієнтів БПР, заданої надлишковим описом	Оцінки коефіцієнтів	Дисперсії оцінок
$b_1 = 1$	$\widehat{b_1} = 2{,}3$	$D\widehat{b_1} = 0{,}437727$
$b_2 = 0$	$\widehat{b_2} = 0$	$D\widehat{b_2} = 0{,}43621$
$b_3 = 3$	$\widehat{b_3} = 2{,}8$	$D\widehat{b_3} = 0{,}010707$
$b_4 = 2$	$\widehat{b_4} = 2$	$D\widehat{b_4} = 0{,}034809$
$b_5 = 0$	$\widehat{b_5} = 0$	$D\widehat{b_5} = 28{,}7666 \cdot 10^{-4}$
$b_6 = 1{,}5$	$\widehat{b_6} = 1{,}3$	$D\widehat{b_6} = 0{,}019699$
$b_7 = 2{,}3$	$\widehat{b_7} = 2{,}3$	$D\widehat{b_7} = 4{,}4947812 \cdot 10^{-7}$
$b_8 = 1{,}5$	$\widehat{b_8} = 1{,}5$	$D\widehat{b_8} = 0{,}287666 \cdot 10^{-8}$
$b_9 = 3{,}7$	$\widehat{b_9} = 3{,}7$	$D\widehat{b_9} = 2{,}87666 \cdot 10^{-7}$
$b_{10} = 2{,}8$	$\widehat{b_{10}} = 2{,}8$	$D\widehat{b_{10}} = 2{,}809238 \cdot 10^{-8}$
$b_{11} = 1{,}3$	$\widehat{b_{11}} = 1{,}3$	$D\widehat{b_{11}} = 1{,}8877 \cdot 10^{-12}$

Аналіз результатів розв'язання прикладу. Для того, щоб ММГУА максимальна дисперсія для оцінки коефіцієнтів була 0,01, згідно з загальною теорією повторних експериментів, повторів необхідно 140, тобто 1400 випробувань. Щоб зменшити кількість необхідних випробувань, в табл. 2.14 максимальні значення вхідних змінних треба покласти $x_1 = 4, x_2 = 5, x_3 = 40, x_4 = 50$.

Незважаючи на достатньо велику дисперсію оцінки коефіцієнта $\widehat{b_2}$, використання перевірочної послідовності (головна ідея МГУА), лінійний член $b_2 x_2$ був виключений з надлишкового опису БПР.

Як видно з табл. 2.3, 2.5, 2.7, 2.9, 2.11, вихідна змінна приймає астрономічно великі значення (десятки мільйонів), що не може бути в реальному активному експерименті. Річ в тому, що в прикладі всі коефіцієнти БПР одного знаку. Дійсно, в реальному експерименті вихідні значення не можуть астрономічно великими і це пов'язано з тим, що коефіцієнти БПР, що оцінюються, мають різні знаки. Але з зага-

льної теорії випливає, що величина дисперсії оцінок коефіцієнтів при нелінійних членах БПР залежить лише від c_i, d_i, що задають область допустимих значень вхідної змінної x_i, $i = \overline{1,n}$, і не залежить від знаків коефіцієнтів БПР. В прикладі спеціально вибрані однакові знаки коефіцієнтів БПР, щоб показати, що і при астрономічних значення вихідної змінної оцінки коефіцієнтів при нелінійних членах БПР знаходяться точно.

2.5 Висновки до розділу 2

У розділі 2 викладено синтетичний метод побудови багатовимірних поліноміальних регресій заданих надлишкових описів. Метод є органічним поєднанням декомпозиційного методу та ММГУА. Декомпозиційний метод задачу оцінки коефіцієнтів при нелінійних членах БПР зводить до послідовної побудови ОПР та розв'язання відповідних невироджених систем лінійних рівнянь, змінними яких є оцінки коефіцієнтів при нелінійних членах БПР.

Показано, що для побудови ОПР з довільним повторним активним експериментом можна використовувати лише один набір НОПФ з точними значеннями наперед заданої кількості розрядів після коми. Наведено теоретично обґрунтовані часткові випадки, що дозволяють знаходити оцінки коефіцієнтів при нелінійних членах БПР з заданою точністю. Оцінки всіх інших коефіцієнтів БПР знаходяться ММГУА. Модифікація МГУА полягає в тому, що завдання БПР заданою надлишковим описом дозволяє замість багаторівневого селекційного алгоритму побудови множини часткових описів замінити ефективним алгоритмом розбиття множини всіх невідомих коефіцієнтів БПР на два класи, що по-перше суттєво скорочує реалізацію МГУА, а по-друге підвищує ймовірність знаходження правильної структури шуканої БПР.

На завершення наведено приклад, який реалізує практично всі алгоритмічні процедури синтетичного методу. Проведено аналіз отриманого рішення.

3 ДОСЛІДЖЕННЯ ОБЧИСЛЮВАЛЬНИХ АСПЕКТІВ СИНТЕТИЧНОГО МЕТОДУ, ЕЛЕМЕНТИ ОПТИМІЗАЦІЇ ЇХ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В силу логіки структури запропонованого синтетичного методу побудови БПР на основі повторних активних експериментів, проблема створення його ефективного програмного забезпечення може бути декомпозована на дві підпроблеми, перша – це оптимізація програмного забезпечення окремих підагоритмів синтетичного методу та друга – оптимізація логіки зв'язків окремих підпрограм, що складають в цілому програмне забезпечення. У цьому розділі наведено вирішення першої підпроблеми.

3.1 Конструювання програмного забезпечення побудови нормованих ортогональних поліномів Форсайта з наперед заданою точністю

У теоретичних викладках (п. 2.2) було показано, що для знаходження оцінок коефіцієнтів ОПР (необхідних для побудови БПР) з необхідною точністю потрібно побудувати нормовані ортогональні поліноми Форсайта з точними значеннями наперед заданої кількості розрядів після коми. Для цього потрібно розробити підпрограму (рис. 3.1), яка буде будувати НОПФ з нульового по заданий r-тий ступінь і повертати їх коефіцієнти, що будуть зберігатись у матриці Q. Щоб уникнути накопичення помилок при побудові НОПФ, коефіцієнти вже побудованих НОПФ та змінні (2.21), що використовуються для побудови наступних НОПФ зберігаються у вигляді раціональних дробів. Операції з раціональними дробами будуть виконуватись за допомогою символьних обчислень. Для побудови НОПФ будемо використовувати мову програмування Python [70], а для символьних обчислень бібліотеку SymPy [77] для цієї мови програмування. Щоб отримати значення коефіцієнтів НОПФ з точним значенням заданої кількості розрядів після коми, потрібно у заключному раціональному дробі (з матриці Q), що зберігає коефіцієнт НОПФ поділити чисельник на знаменник та округлити до потрібної кількості розрядів після коми.

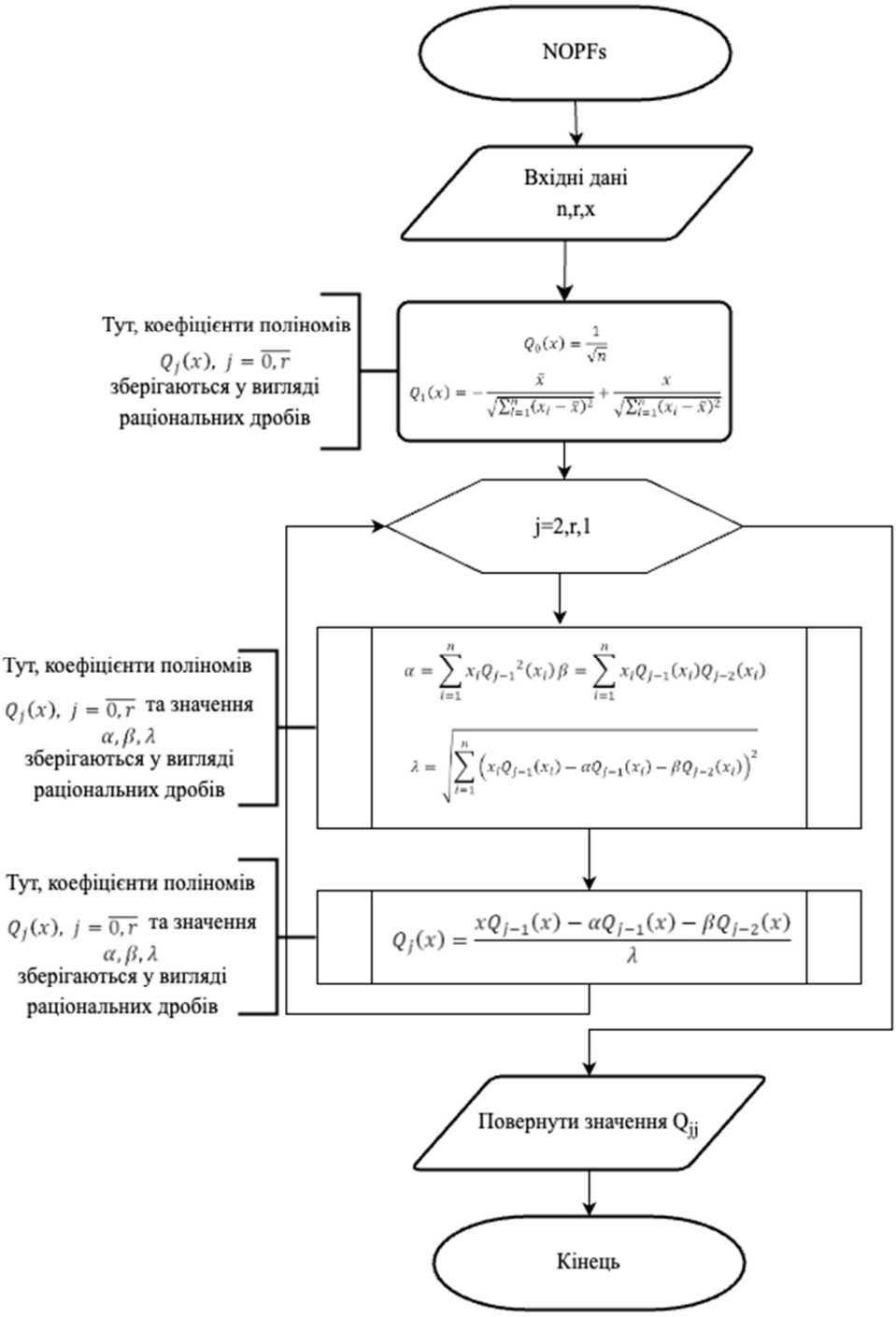

Рисунок 3.1 – Блок-схема алгоритму побудови НОПФ

3.2 Дослідження ефективності алгоритмів, що реалізують операції з матрицями в методі найменших квадратів, зокрема обґрунтування використання паралельних обчислень

Операції над матрицями та векторами, які є складовими формули для знаходження оцінок коефіцієнтів регресії за допомогою МНК як складової ММГУА, можуть

виконуватись ефективніше при застосуванні швидкісних алгоритмів або паралельних обчислень [53]. Для знаходження оцінок коефіцієнтів регресії потрібно виконати чотири різні операції: транспонування матриці, множення матриць, обернення матриці, множення матриці на вектор. Оскільки операції транспонування матриць та множення матриці на вектор є менш трудомісткими, для них не будуть застосовуватись швидкісні алгоритми, а їх паралельні реалізації не дають бажаного приросту у порівнянні з послідовними [53]. Буде досліджено ефективність алгоритмів множення матриць та обернення матриць для задачі знаходження оцінок коефіцієнтів регресій МНК.

Дослідження ефективності алгоритмів множення матриць виконувалось шляхом реалізації алгоритмів за допомогою мови програмування Python [70] на базі 8-ми ядерного мікропроцесора Apple M1, 3.2 GHz, ОЗП 16 ГБайт [4], та фіксації часу роботи різних алгоритмів для фіксованої кількості потоків і квадратних матриць заданої розмірності. Було виконано програмну реалізацію паралельних алгоритмів Фокса [59] та Стрічкового [59], паралельних алгоритмів Штрассена [15] та Винограда-Штрассена [15], класичного послідовного методу множення матриць [11], а також вбудованого у бібліотеку SymPy [77] для мови програмування Python послідовного методу множення матриць. У ході досліджень було виявлено, що кращі результати по часу, паралельні алгоритми показують на 8-ми потоках, для швидкісних алгоритмів багатопоточність неважлива. Розмірність матриць для дослідження ефективності алгоритмів множення була обрана з практичних міркувань для задач побудови регресій, які розв’язуються МНК [83]. Абсолютні значення часу роботи алгоритмів у секундах наведено у табл. 3.1.

Аналізуючи дані табл. 3.1, можна зробити висновок, що швидше за інші працює Стрічковий алгоритм множення матриць. З урахуванням того, що він застосовується без модифікації для неквадратних матриць, на відміну від інших, саме його і будемо використовувати для знаходження оцінок коефіцієнтів регресій МНК у ММГУА. Крім того з табл. 3.1 видно, що розпаралелювання при множенні матриць доцільно використовувати починаючи з розмірності більше 125, для матриць меншої розмірності будемо використовувати звичайний послідовний метод множення матриць.

Таблиця 3.1 – Час роботи алгоритмів множення матриць у секундах

Алгоритм / Розмірність	Стрічковий	Фокса	Штрассена	Винограда-Штрассена	Послідовний класичний	SymPy
4 x 40	0,267	0,284	0,00053	0,00038	0,000426	0,013
8 x 80	0,269	0,29	0,0031	0,0029	0,002897	0,004279
16 x 160	0,301	0,309	0,023	0,022	0,0212	0,0252
32 x 320	0,355	0,358	0,1703	0,1696	0,195	0,197
64 x 640	0,605	0,733	1,345	0,738	1,3567	1,4498
125 x 1250	2,504	3,319	3,942	4,962	10,12	11,74
250 x 2500	18,28	25,05	36,27	40,59	80,98	95,59
500 x 5000	163,79	212,48	353,03	349,81	653,92	773,02
1000 x 10000	1375,02	1797,41	3498,1	2424,09	5261,58	6217,31
2000 x 20000	11378,51	14561,92	31709,9	16818,01	42274,52	49871,43

Дослідження ефективності алгоритмів обернення матриць виконувалось шляхом реалізації алгоритмів на мові програмування Python [70] з використанням 8-ми ядерного мікропроцесора Apple M1, 3.2 GHz, ОЗП 16 ГБайт [4], та фіксації часу роботи різних алгоритмів для фіксованої кількості потоків і матриць заданої розмірності. Для уникнення похибки обчислень використовувались символьні обчислення з використанням бібліотеки SymPy [77] для мови програмування Python, фактично обчислення виконувались у раціональних дробах, що унеможливлює накопичення помилок обчислень при операції ділення. Було виконано програмну реалізацію паралельної та послідовної версії алгоритмів LUP [60] та Жордана-Гауса [60], класичного послідовного методу обернення матриці [11] та вбудованого у бібліотеку SymPy [77] для мови програмування Python послідовного методу обернення матриць. У ході досліджень було виявлено, що кращі результати по часу, паралельні алгоритми показують на 16-ти потоках, а для послідовних, як і у досліджені вище, параметр кількості потоків не має суттєвого значення. Абсолютні значення часу роботи алгоритмів у секундах наведено у табл. 3.2.

Аналізуючи дані табл. 3.2, можна зробити висновок, що швидше за інші працює паралельна реалізація LUP алгоритму обернення матриць. Саме його і будемо використовувати для знаходження оцінок коефіцієнтів МНК у ММГУА. Крім того з табл. 3.2 видно, що розпаралелювання при оберненні матриць доцільно використовувати

починаючи з розмірності більше 64, причому для знаходження оберненої матриці розмірності менше 64 будемо використовувати послідовний LUP алгоритм.

Таблиця 3.2 – Час роботи алгоритмів обернення матриць у секундах

Алгоритм / Розмірність	LUP послідовний	LUP паралельний	Жордана-Гауса послідовний	Жордана-Гауса паралельний	Послідовний класичний	SymPy
4 x 4	0,00047	0,55	0,00054	0,6	0,003486	0,00531
8 x 8	0,0049	0,36	0,0057	0,58	0,08543	0,1241
16 x 16	0,076	0,58	0,0084	0,82	4,641	6,432
32 x 32	1,75	1,2	1,88	2,34	233,084	271,132
64 x 64	45,03	12,01	46,91	29,67	3181,42	3631,29
100 x 100	385,34	96,11	403,11	202,61	14327,8	16279,73
200 x 200	10281,36	2173,02	10415,06	5814,95	131662,3	149090,5
300 x 300	45478,05	9663,84	45922,05	26000,99	464587,4	525545,5

Як показали дослідження, матричні операції для формули МНК у ММГУА більш ефективно можна виконати на багатоядерних процесорах. У програмній реалізації МНК у ММГУА будемо використовувати Стрічковий алгоритм множення матриць для розмірності (висоти чи ширини) від 125 і класичний алгоритм множення матриць для меншої розмірності. Що стосується обернення матриць для формули МНК у ММГУА, то будемо використовувати паралельну реалізацію LUP алгоритму для розмірності від 64 і послідовну реалізацію LUP алгоритму для матриць меншої розмірності. Варто зауважити, що максимальну ефективність паралельні реалізації обраних алгоритмів показали на різній кількості потоків, тому у програмній реалізації МНК у ММГУА для функцій, що виконуватимуть операції множення та обернення матриць доцільно ввести параметр – кількість потоків. Параметр, який регулює кількість потоків дозволить користувачеві більш ефективно користуватись програмною реалізацією ММГУА на своєму пристрої. Крім того, необхідно надати кінцевому користувачу можливість обирати між паралельною та послідовною реалізацією ММГУА, виходячи з конкретних характеристик його технічних засобів.

3.3 Теоретичне обґрунтування використання лише матриць основного експерименту при розробці програмного забезпечення ММГУА, що реалізує повторний експеримент

У загальному вигляді задача регресії, що розв'язується ММГУА, може бути представлена у вигляді:

$$Y = A\theta + \bar{E}, \tag{3.1}$$

де $\bar{E} = (E_1, \dots, E_n)^\top, Y = (Y_1, \dots, Y_n)^\top$, а вигляд матриці A залежить від виразу (3.1), а вектор невідомих коефіцієнтів БПР, що входить в (3.1) для спрощення позначили θ. Тоді $MY = A\theta$, вектор оцінок $\hat{\theta} = \left(\hat{\theta}_0, \hat{\theta}_1, \dots, \hat{\theta}_m\right)^\top$, отриманий МНК, має вигляд

$$\hat{\theta} = (A^\top A)^{-1} A^\top Y,\ M\hat{\theta} = \theta. \tag{3.2}$$

Примітка 3.1. В задачі побудови БПР (1) вектор Y замінюється на вектор

$$Y^* = \left(Y_1 - f(\bar{x}_1), \dots, Y_n - f(\bar{x}_n)\right)^\top, \tag{3.3}$$

де $f(\bar{x}_i)$ – відоме значення нелінійної частини БПР з урахуванням найдених оцінок коефіцієнтів при її нелінійних членах і виключенням нелінійних членів, що вважаються залишковими.

Нехай реалізується повторний активний експеримент $\left(\bar{x}_i \to y_i,\ i = \overline{1,n},\ \bar{x}_i \to y_{n+i},\ i = \overline{1,n},\ \dots,\ \bar{x}_i \to y_{n(p-1)+i},\ i = \overline{1,n}\right)$, тобто проведено np випробувань, в яких послідовність вхідних змінних $\bar{x}_i,\ i = \overline{1,n}$, повторюється p разів. Тоді має місце наступне твердження.

Твердження 3.1. Вектор оцінок $\hat{\theta}$ невідомих коефіцієнтів $\theta_0, \theta_1, \dots, \theta_m$, отриманий МНК, має вигляд

$$\hat{\theta} = (A^\top A)^{-1} A^\top \left(\frac{1}{p}\sum_{j=1}^{p} Y^j\right), \tag{3.4}$$

де випадкові вектори $Y^j,\ j=\overline{2,p}$, є віртуальними незалежними копіями віртуального випадкового вектору Y (3.1), а $Y^1=Y$. Реалізаціями їх компонент є числа $y_{(j-1)n+i}$, $i=\overline{1,n},\ j=\overline{1,p}$.

Оцінка випадкового вектору $\hat{\theta}$ (3.4) є незміщеною, а дисперсії його компонент в p разів менше дисперсій компонент вектору оцінок (3.2).

Доведення. Формула (3.4) випливає з векторної рівності

$$\left((A^\top\cdots A^\top)\begin{pmatrix}A\\ \vdots\\ A\end{pmatrix}\right)^{-1}(A^\top\cdots A^\top)\begin{pmatrix}Y^1\\ \vdots\\ Y^p\end{pmatrix}=\frac{1}{p}(A^\top A)^{-1}A^\top\sum_{j=1}^{p}Y^j. \tag{3.5}$$

Незміщеність випадкового вектору $\hat{\theta}$ (3.4) випливає з рівності

$$M(A^\top A)^{-1}A^\top Y=M(A^\top A)^{-1}A^\top Y^j=\theta, j=\overline{1,p}, \tag{3.6}$$

а властивість дисперсій його компонент – з рівностей

$$D\left[(A^\top A)^{-1}A^\top\frac{1}{p}\sum_{j=1}^{p}Y^j\right]=\frac{1}{p^2}\sum_{j=1}^{p}D\left[(A^\top A)^{-1}A^\top Y^j\right],$$
$$D\left[(A^\top A)^{-1}A^\top Y^j\right]=D[(A^\top A)^{-1}A^\top Y], j=\overline{1,p}. \tag{3.7}$$

Наслідок 1. Знаючи дисперсії компонент вектору оцінок $\hat{\theta}$ (3.2), можна знайти кількість повторів p активного експерименту, щоб найбільша дисперсія оцінок $\hat{\theta}_0,\dots,\hat{\theta}_m$ мала допустиме значення.

Наслідок 2. При довільному p в силу (3.5) не треба в загальній формулі МНК додатково обертати матрицю, використовується лише матриця $(A^\top A)^{-1}A^\top$, де A – матриця основного експерименту, що суттєво спрощує реалізацію програмного забезпечення МНК на повторних експериментах.

3.4 Обґрунтування можливості розпаралелювання обчислень в ММГУА для знаходження оцінок коефіцієнтів часткових описів та ЗСК

Розглянемо блок-схему алгоритму ММГУА у загальному вигляді (рис. 3.2).

Рисунок 3.2 – Блок-схема алгоритму ММГУА

Як показує аналіз приведеної блок-схеми алгоритму ММГУА (рис. 3.2), оцінки коефіцієнтів часткових описів регресій (2.6) за результатами експерименту $(\bar{x}_i \rightarrow y_i,\ i = \overline{1,n})$ не залежать одні від одних і можуть знаходитись за допомогою процедури

організації паралельних обчислень (ітерації відповідного циклу можуть виконуватись паралельно) для кожного окремого часткового опису. Аналогічною процедурою розпаралелювання обчислень можуть знаходитись значення залишкових сум квадратів (2.8) для кожного часткового опису регресій за результатами повторного експерименту $\left(\bar{x}_i \to y_{n+i},\ i = \overline{1,n}\right)$.

3.5 Висновок до розділу 3

У розділі 3 проведено ґрунтовне дослідження окремих аспектів програмного забезпечення реалізації синтетичного методу побудови БПР.

Зокрема, наведено підхід, який дозволяє знаходити значення коефіцієнтів НОПФ з точними значеннями наперед заданої кількості розрядів після коми. Це досягається за рахунок зберігання коефіцієнтів НОПФ та допоміжних змінних, необхідних для їх розрахунку у вигляді раціональних дробів, а також використання механізму символьних обчислень для виконання операцій з раціональними дробами.

Для загальної формули МНК, в результаті досліджень, було обрано паралельні реалізації алгоритмів операцій множення матриць та обертання матриць, що загалом пришвидшить пошук оцінок коефіцієнтів МНК.

Теоретично обґрунтовано та доведено можливість використання лише матриць основного експерименту при розробці програмного забезпечення ММГУА, що реалізує повторний експеримент.

Показано можливість знаходити значення коефіцієнтів часткових описів та ЗСК незалежно один від одного, що дозволяє використовувати багатопоточність для даної задачі.

4 РОЗРОБКА КРОСПЛАТФОРМНОЇ БІБЛІОТЕКИ, ЩО РЕАЛІЗУЄ СИНТЕТИЧНИЙ МЕТОД ПОБУДОВИ БАГАТОВИМІРНОЇ ПОЛІНОМІАЛЬНОЇ РЕГРЕСІЇ

У результаті огляду програмного забезпечення, що реалізує алгоритми методів регресійного аналізу, можна стверджувати, що програмну реалізацію синтетичного методу побудови БПР заданою надлишковим описом варто виконувати у вигляді спеціалізованої кросплатформної бібліотеки. Подібний спосіб реалізації дозволить вирішити проблеми інтеграції у стороннє програмне забезпечення, сумісності з різними платформами та виключить зайві часові затрати на повторну програмну реалізацію синтетичного методу для кожного нового кінцевого програмного забезпечення.

Попередній розділ був присвячений проблемі створення програмного забезпечення, що реалізує запропонований синтетичний метод побудови БПР на основі повторних активних експериментів, у частині оптимізації програмної реалізації окремих підалгоритмів. Даний розділ буде присвячений обґрунтуванню використання допоміжних технічних та програмних засобів, що будуть використовуватись при розробці кросплатформної бібліотеки, розробці програмного інтерфейсу кросплатформної бібліотеки та опису загальної логіки зв’язків окремих компонентів кросплатформної бібліотеки, що реалізує синтетичний метод побудови БПР на основі повторних активних експериментів.

4.1 Обґрунтування використання засобів розробки та розгортання кросплатформної бібліотеки

Для розробки програмного інтерфейсу кросплатформної бібліотеки, що реалізує синтетичний метод побудови БПР на основі повторних активних експериментів та самої кросплатформної бібліотеки може бути використана будь-яка мова загального призначення, такі як C++, C#, Java, Python, Js [7, 19, 20, 33, 75]. З усіх цих варіантів було обрано Python, оскільки дана мова програмування краще за інші підходить для реалізації data science та статистичних методів обробки даних [23, 27, 37, 39, 41].

Крім того для мови Python існує великий набір високорівневих фреймворків різного призначення та програмних бібліотек, які можна використовувати у якості допоміжних засобів при розробці [58].

Середовищем розробки бібліотеки було обрано IntelliJ IDEA [24], через наявність безкоштовної експрес версії та зручність встановлення допоміжних програмних пакетів та бібліотек.

У якості архітектури кросплатформної бібліотеки будемо використовувати монолітну архітектуру [30], оскільки решта архітектур програмного забезпечення: багаторівнева, клієнт-серверна, мікросервісна, сервісно-орієнтована, не рекомендується для використання при розробці такого роду програмного забезпечення [21, 64, 65, 67]. Внутрішня логіка бібліотеки буде побудована з використанням компонентно-орієнтованого підходу, так як даний підхід добре себе зарекомендував при розробці програмного забезпечення цільового призначення, що буде використовуватись при розробці цільового прикладного програмного забезпечення [81].

Компонентно-орієнтований підхід [81] розробки програмного забезпечення (CBSE) є способом розробки програмного забезпечення, що базується на повторному використанні компонентів. Компонентний підхід втілює гібридний підхід між багаторівневою та об’єктною архітектурою. Замість багаторівневого підходу, горизонтальних слоїв виконується розподіл застосунку по вертикалі на модульні компоненти.

Компонент у цьому контексті є групою пов’язаних функцій, які знаходяться за «чистим» інтерфейсом. Процес повторного використання втілює у якості CBSE особливу парадигму розробки програмного забезпечення. Філософія даної парадигми: «Не будуй, а купуй» (“buy, don’t build.”). По своїй суті підхід CBSE передбачає повторне використання попередньо створених та доступних конструкцій програмного забезпечення, замість їх розробки з самого початку (рис. 4.1).

У кросплатформній бібліотеці, що реалізує синтетичний метод побудови БПР на основі повторних активних експериментів, для реалізації кожного окремого підалгоритму буде використовуватись окремий компонент, крім того будуть використову-

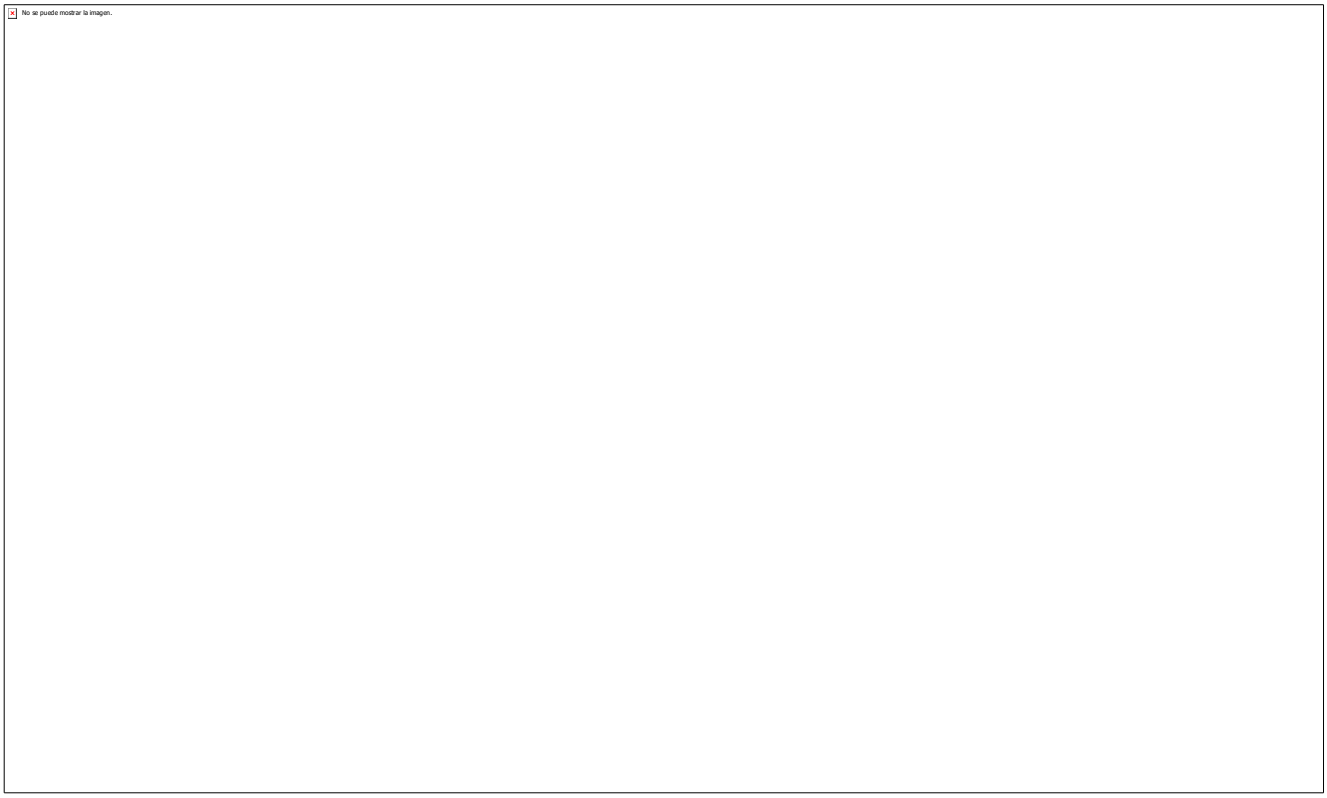

Рисунок 4.1 – Модель CBSE [81]

ватись додаткові, допоміжні компоненти, що включають як власноруч розроблені (реалізація алгоритмів ММГУА, декомпозиційного методу, обробки помилок тощо) так і зовнішні (символьні обчислення).

Крім того, кросплатформна бібліотека буде представляти собою компонент з програмним інтерфейсом, але більш високого рівня з можливістю інтеграції в стороннє програмне забезпечення.

Підхід CBSE при реалізації внутрішньої логіки кросплатформної бібліотеки дозволить працювати з нею різним групам користувачів з точки зору їх базової підготовки у областях програмування та статистичного аналізу (статистика). Користувачі без відповідних знань зможуть у автоматизованому режимі викликати функції кросплатформної бібліотеки, передавати в них дані та отримувати кінцевий результат. При цьому користувачі, які мають достатню ступінь підготовки, використовуючи інструкцію, що містить теоретичні положення синтетичного методу, зможуть в інтерактивному режимі створити унікальний алгоритм декомпозиційного методу, для задачі, що

розв’язується та впливати на створення вхідних даних для повторного експерименту, що реалізує ММГУА.

У розділі 3 було обґрунтовано необхідність використання паралельних обчислень при реалізації деяких підалгоритмів ММГУА. Далі буде розглянуто, як у мові програмування Python можна реалізовувати механізм паралельних обчислень.

Глобальне блокування інтерпретатора Python (GIL) [8] – це м’ютекс, який дозволяє тільки одному потоку утримувати управління інтерпретатором Python.

Це означає, що тільки один потік може перебувати в стані виконання в будь-який момент часу. Вплив GIL не помітний для розробників, які виконують однопотокові програми, але він може бути вузьким місцем в продуктивності прив’язаного до процесора і багатопотокового коду.

Оскільки GIL дозволяє виконувати тільки один потік одночасно навіть в багатопотоковій архітектурі з більш ніж одним ядром процесора, GIL отримав репутацію «сумнозвісної» функції Python [3].

Також, локальні змінні, безумовно, є «виключними для потоку». Жоден інший потік не може отримати до них прямий доступ, і це корисно, але недостатньо для гарантування семантичної безпеки потоку. Локальна змінна в одному потоці не зберігає своє значення в тому ж місці, що і однойменна локальна змінна в іншому потоці. Це в основному є прямим наслідком визначення функцій та локальних змінних. Більшменш кожна структурована мова програмування робить приблизно наступне: кожного разу, коли викликається функція, виділяється новий блок пам’яті, який називається кадр стеку і він стає недійсним або знищується при поверненні функції. У цьому кадрі зберігаються значення локальних змінних. Таким чином, у мові програмування Python можна мати дві або більше копії цієї функції, що виконуються в окремих потоках, і значення *a, b, c* в кожному потоці будуть посилатися на різні об’єкти. Тому, якщо функція повинна викликатися тільки в одному потоці, її потрібно зробити із глобальною видимістю, щоб запобігти конфліктам між потоками [61].

Модулі Threading та Multiprocessing. Threading [80] – це вбудований у мову програмування Python модуль, що має набір методів для роботи з потоками. Він будує інтерфейси потоків вищого рівня над модулем нижчого рівня _thread.

Щоб зрозуміти принцип модулю Threading, нижче наведено приклад із використанням та поясненням коду.

```
# імпорт модулю, конструктора потоку та часу для замірювання
import threading
from threading import Thread
from time import time, sleep

# функція для демонстраційного прикладу
def foo(sec: int, multiply: int) -> None:
    sleep(sec * multiply)
    print(f"foo: {sec=} and {multiply=}")
# для демонстраційного прикладу було створено два потоки
T1 = Thread(target=foo, args=(4, 1,))
T2 = Thread(target=foo, args=(2, 1,))
S = time()
T1.start()
sleep(3)
T2.start()
print(f"{threading.active_count()=}")
T1.join()
print(f"{threading.active_count()=}")
T2.join()
print(f"{threading.active_count()=}")
print(f"Estimated time: {time()-S}")
```

Очікуваний час виконання наведеного вище коду має бути близько 5 секунд, із урахуванням системних процесів, через те, що код виконується в Main Thread, в даному прикладі виконується три потоки одночасно. Перший (T1) виконується 4 секу-

нди, другий (T2) почне виконуватися після того, як в основному потоці пройде 3 секунди, але одночасно з ним вже працює перший потік, який повинен працювати ще 1 секунду разом із другим, тому загальний час становить 4 + 2 – 1 = 5 секунд.

```
threading.active_count()=3
foo: sec=4 and multiply=1
threading.active_count()=2
foo: sec=2 and multiply=1
threading.active_count()=1
Estimated time: 5.012614488601685.
```

Як видно, практичний результат відповідає очікуваному.

Multiprocessing [36] – це пакет, який підтримує породження процесів за допомогою API, подібного до модуля багатопотоковості. Пакет багатопроцесорності пропонує як локальний, так і віддалений паралелізм, ефективно обходячи глобальне блокування інтерпретатора за рахунок використання підпроцесів замість потоків. Завдяки цьому модуль багатопроцесорної обробки дозволяє програмісту повною мірою використовувати декілька процесорів на даному пристрої. Він працює як під Unix, так і під Windows, тобто є кросплатформним. Модуль багатопроцесорності також вводить API, які не мають аналогів у модулі потоків. Яскравим прикладом цього є об’єкт Pool, який пропонує зручний засіб розпаралелювання виконання функції на декілька вхідних значень, розподіляючи вхідні дані між процесами (паралелізм даних). Наступний приклад демонструє поширену практику визначення таких функцій у модулі [36]. Це базовий приклад паралелізму даних з використанням Pool:

```
from multiprocessing import Pool
from multiprocessing import freeze_support
from time import time, sleep
def foo(sec: int) -> None:
    sleep(sec)
    print(f"\nfoo: {sec=}")
if __name__ == '__main__':
    freeze_support()
```

```
S = time()
with Pool(processes=4) as pool:
    print(pool.map(foo, [2, 3, 2, 3]))
print(f"Estimated time: {time()-S}")
```

У даному прикладі приймають участь 4 процеси. Тому очікується, що програма повинна виконуватись за час близький до 3 секунд із урахуванням системних процесів. Вивід буде наступним.

```
foo: sec=2
foo: sec=2
foo: sec=3
foo: sec=3
[None, None, None, None]
Estimated time: 3.6487306354522705
```

Також, не менш важливим є те, що кількість «воркерів» повинна бути меншою або дорівнювати 61, якщо операційною системою є Windows.

Pool.map() та Pool.starmap() в Multiprocessing. Клас multiprocessing.pool.Pool у мові програмування Python надає пул процесів багаторазового використання для виконання спеціальних завдань. Вбудована функція map() дозволяє застосувати функцію до кожного елементу в ітерації. Проблема цієї функції полягає в тому, що вона перетворює наданий ітератор елементів у список і подає всі елементи як задачі в пул процесів, після чого блокує до тих пір, поки всі задачі не будуть виконані. Вона дає один результат, що повертається з заданої цільової функції, викликаної з одним елементом із заданого ітерабельного списку. Зазвичай викликають map() і повторюють результати в циклі for [36].

Пул процесів надає версію функції map(), у якій цільова функція викликається паралельно для кожного елементу з наданого ітератора. Проблема з функцією Pool.map() полягає в тому, що вона приймає тільки один набір елементів, дозволяючи тільки один аргумент для цільової функції завдання. Це відрізняється від вбудованої

функції map(). Функції завдання, які приймають тільки один аргумент, є серйозним обмеженням.

Пул процесів надає версію map(), яка дозволяє передавати декілька аргументів до цільової функції задачі через функцію Pool.starmap(). Вона приймає ім'я функції, яку потрібно застосувати, та ітерабельний список, після чого перетворить наданий ітерабельний список у список і повертає одне завдання для кожного елемента в ітерабельному списку.

Важливо, що кожен елемент в ітерабельному списку, наданий функції starmap(), може сам по собі бути ітерабельним списком, що містить аргументи, які потрібно передати цільовій функції завдання. Це дозволяє цільовій функції завдань отримувати декілька аргументів. Наприклад, ми можемо мати ітерабельний список, в якому кожен елемент ітерабельного списку є ітерабельним списком аргументів для кожного виклику функції.

Як і функція Pool.map(), Pool.starmap() дозволяє видавати задачі в пул процесів частинами. Це може зробити виконання великої кількості завдань у дуже довгому ітерабельному списку більш ефективним, оскільки аргументи та значення, що повертаються з цільової функції завдання, можуть передаватися пакетами з меншими обчислювальними накладними витратами.

Ключова відмінність функції starmap() від функції map() полягає в тому, що starmap() підтримує цільову функцію з більш ніж одним аргументом, тоді як функція map() підтримує цільові функції тільки з одним аргументом.

Модуль concurrent.futures [10] надає високорівневий інтерфейс для асинхронного виконання викликаних функцій.

Асинхронне виконання може здійснюватись потоками, використовуючи ThreadPoolExecutor, або окремими процесами, використовуючи ProcessPoolExecutor. Обидва реалізують один і той же інтерфейс, який визначається абстрактним класом Executor.

Клас concurrent.futures.Executor – абстрактний клас, який надає методи для виконання викликів асинхронно. Його слід використовувати не безпосередньо, а через його конкретні підкласи.

Порівняння модулів Threading та Multiprocessing. Для порівняння модулів Threading та Multiprocessing розробимо невеликий приклад, що містить алгоритм паралельного множення матриць (Додаток Б).

В результаті виконання коду отримаємо наступні результати.

```
mul_n_times_threading time: 4.2416 seconds...
mul_n_times_thread_pool_executor time: 4.6940 seconds...
mul_n_times_multiprocessing time: 2.4654 seconds...
mul_n_times_process_pool_executor time: 2.0815 seconds...
```

Як можна побачити, функції, засновані на багатопоточності, працюють значно повільніше, ніж на багатопроцесорності. Це обумовлено тим, що в Python присутній GIL, таким чином використовується лише один процес, поки інші чекають на його завершення. У той час, як у multiprocessing використовується кілька ядер одночасно.

Для завдань, що виконуються на процесорі (математичні обчислення, обрахунки параметрів тощо), наявність декількох потоків марна. Оскільки в одиницю часу може виконуватися тільки один потік, навіть якщо буде породжено декілька, процесор все одно буде працювати тільки з одним потоком. По суті, вони будуть виконуватися все одно один за іншим. А витрати роботи з кількома потоками сприятимуть зниженню продуктивності, які можна спостерігати під час використання threading у завданнях, що виконуються на процесорі.

Щоб обійти це обмеження, слід використовувати модуль multiprocessing.

Підсумовуючи, threading слід використовувати в таких завданнях, у яких важлива послідовність виконання задачі та детермінізм, також у мережевому програмуванні, під час очікування отримання даних або обробці запитів, доки буде продовжуватися оновлення, відтворення та відображення додатка на стороні клієнта, при цьому мінімізуючи використання процесору.

Якщо необхідно, щоб обчислення виконувались незалежно один від одного, рекомендується використовувати multiprocessing, оскільки цей модуль надає можливість розділити їх між доступними ядрами процесора, тим самим отримати значний приріст у швидкості обробки. З даних міркувань буде використано multiprocessing при розробці деяких підалгоритмів кросплатформної бібліотеки, що реалізує синтетичний метод побудови БПР на основі повторних активних експериментів.

Для розгортання кросплатформної бібліотеки будемо використовувати систему управління пакетами pip [57]. Це універсальна, зручна і найбільш популярна [18] система управління пакетами, написаними для мови програмування Python. В результаті розгортання користувачі зможуть завантажувати і встановлювати бібліотеку для своїх потреб командою pip install regression_lib_mpr у своєму середовищі розробки.

4.2 Програмний інтерфейс кросплатформної бібліотеки

Програмний інтерфейс кросплатформної бібліотеки, що реалізує синтетичний метод побудови БПР на основі повторних активних експериментів включатиме в себе набір функцій, викликавши які користувач зможе у автоматизованому режимі передавати в них дані та отримувати кінцевий результат, або у інтерактивному режимі створити унікальний алгоритм декомпозиційного методу, для знаходження деяких оцінок коефіцієнтів БПР та впливати на створення вхідних даних для повторного експерименту, що реалізує ММГУА.

Для виконання цих задач кросплатформна бібліотека повинна забезпечувати виконання наступних основних функції.

Функція *RA_Forsythe*, що реалізує алгоритм знаходження коефіцієнтів НОПФ з необхідною точністю на інтервалі з рівномірним розподілом вхідної детермінованої змінної.

Функція RA_Forsythe приймає на вхід:

– початкову та кінцеву точки відрізку проведення активного експерименту $c = x_1$ та $d = x_n$ (дійсні числа);
– кількість спостережень n (в діапазоні 2..10000, цілі);
– максимальну степінь НОПФ r (в діапазоні 1..10, цілі).

Функція RA_Forsythe повертає:

– матрицю коефіцієнтів НОПФ Q, $q_{ij}, i,j = \overline{0,r}$;

– код помилки та її сутність при виникненні (виродженість задачі, $n < r$).

Функція *RA_Forsythe2*, що реалізує алгоритм знаходження коефіцієнтів НОПФ з необхідною точністю на інтервалі з довільним розподілом вхідної детермінованої змінної.

Функція RA_Forsythe2 приймає на вхід:

– кількість спостережень n (в діапазоні 2..10000, цілі);

– максимальну степінь НОПФ r (в діапазоні 1..10, цілі);

– вектор значень вхідної детермінованої змінної $x_i, i = \overline{1,n}$, розмірності n (дійсні).

Функція RA_Forsythe2 повертає:

– матрицю коефіцієнтів НОПФ Q, $q_{ij}, i,j = \overline{0,r}$;

– код помилки та її сутність при виникненні (виродженість задачі, $n < r$).

Функція *RA_Forsythe_detx*, що реалізує рівномірний розподіл значень змінної x, для заданих початкової точки та кінцевої.

Функція RA_Forsythe_detx приймає на вхід:

– початкову та кінцеву точки відрізку проведення активного експерименту $c = x_1$ та $d = x_n$ (дійсні числа);

– кількість спостережень n (в діапазоні 2..10000, цілі).

Функція RA_Forsythe_detx повертає:

– вектор значень детермінованої змінної $x_i, i = \overline{1,\mathrm{n}}$, розмірності n (дійсні);

– код помилки та її сутність при виникненні.

Функція *RA_Forsythe_detx_real_to_virtual*, яка реалізує алгоритм знаходження значень змінної x, для заданих початкової точки та кінцевої.

Функція RA_Forsythe_detx_real_to_virtual приймає на вхід:

– початкову та кінцеву точки відрізку проведення активного експерименту $c = x_1$ та $d = x_n$ (дійсні числа);

– кількість спостережень n (в діапазоні 2..10000, цілі);

– початкову та кінцеву точки відрізку проведення віртуального активного експерименту z_1 та z_n (дійсні).

Функція RA_Forsythe_detx_real_to_virtual повертає:

– вектор значень детермінованої змінної $x_{\mathrm{i}}, \mathrm{i} = \overline{1, \mathrm{n}}$, розмірності n (дійсні);
– коефіцієнти лінійного перетворення між реальними та віртуальними значеннями вхідних детермінованих змінних на відрізках проведення активних експериментів a та b (дійсні);
– код помилки та її сутність при виникненні.

Функція *RA_Forsythe_polynomial*, що реалізує МНК з використанням НОПФ для знаходження оцінок коефіцієнтів одновимірної поліноміальної регресії.

Функція RA_Forsythe_polynomial приймає на вхід:

– кількість спостережень n (в діапазоні 2..10000, цілі);
– степінь регресійного поліному r (в діапазоні 2..10, цілі);
– вектор значень детермінованої змінної $x_i, i = \overline{1, n}$, розмірності n (дійсні);
– матрицю коефіцієнтів НОПФ Q, $q_{ij}, i, j = \overline{0, r}$ (дійсні, за замовчуванням для інтервалу (–50, 50) та 10 спостережень);
– вектор вихідних значень результатів активного експерименту $y_i,\ i = \overline{1, n}$, розмірності n (дійсні).

Функція RA_Forsythe_polynomial повертає:

– оцінки коефіцієнтів при нелінійних членах ОПР $\hat{\theta}_j,\ j = \overline{2, r}$, ($\hat{\gamma}_j,\ j = \overline{2, r}$ для ОПР на базі віртуальної ОПР), дисперсії оцінок коефіцієнтів ОПР $D\hat{\theta}_j,\ j = \overline{2, r}$ ($D\hat{\gamma}_j,\ j = \overline{2, r}$);
– код помилки та її сутність при виникненні (виродженість задачі, $n > r$).

Функція *RA_Forsythe_virtual_to_real*, яка реалізує метод знаходження оцінок коефіцієнтів одновимірної поліноміальної регресії $\hat{\theta}_j,\ j = \overline{2, r}$ по значенням оцінок при нелінійних членах віртуальної ОПР $\hat{\gamma}_j,\ j = \overline{2, r}$.

Функція RA_Forsythe_virtual_to_real приймає на вхід:

– степінь регресійного поліному r (в діапазоні 2..10, цілі);

– коефіцієнти лінійного перетворення між реальними та віртуальними значеннями вхідних детермінованих змінних на відрізках проведення активних експериментів *a* та *b* (дійсні);
– вектор значень оцінок для віртуального експерименту $\hat{\gamma}_j,\ j = \overline{2,r}$, розмірності *r* (дійсні);
– дисперсії оцінок коефіцієнтів ОПР $D\hat{\gamma}_j,\ j = \overline{2,r}$;
– матрицю коефіцієнтів НОПФ Q, $q_{ij}, i,j = \overline{0,r}$ (дійсні, за замовчуванням для інтервалу (–50, 50) та 10 спостережень).

Функція RA_Forsythe_virtual_to_real повертає:

– оцінки коефіцієнтів при нелінійних членах ОПР $\hat{\theta}_j,\ j = \overline{2,r}$, дисперсії оцінок коефіцієнтів ОПР $D\hat{\theta}_j,\ j = \overline{2,r}$;
– код помилки та її сутність при виникненні.

Функція *RA_redundant_represent*, що у автоматизованому режимі будує надлишковий опис БПР.

Функція RA_redundant_represent приймає на вхід:

– максимальну степінь надлишкового опису *r* (в діапазоні 2..10, цілі);
– кількість вхідних детермінованих змінних *m* (в діапазоні 2..100, цілі).

Функція RA_redundant_represent повертає:

– кількість членів надлишкового опису *m*;
– матрицю, що містить надлишковий опис $RR_{ij}\ i = \overline{1,m}, j = \overline{1,2}$, розмірності *m* на *2* (перший стовпець містить усі можливі члени надлишкового опису, другий булеве значення, яке показує чи входить член у надлишковий опис);
– код помилки та її сутність при виникненні.

Функція *RA_MGMDH*, що реалізує ММГУА знаходження оцінок БПР.

Функція RA_MGMDH приймає на вхід:

– кількість спостережень *n* (в діапазоні 2..10000, цілі);
– кількість членів надлишкового опису *m* (в діапазоні 2..10000, цілі);

– матрицю значень вхідних детермінованих змінних $X_{ij}, i = \overline{1,n}, j = \overline{1,m}$, розмірності $n \times m$ (дійсні);
– кількість повторів активного експерименту p_c, пара значень яка визначає кількість спостережень основного експерименту та кількість спостережень перевірочної послідовності (в діапазоні 1..100;1..100, цілі, за замовчуванням 1;0);
– матрицю вихідних значень результатів активного експерименту y_{ij}, $i = \overline{1,n}, j = \overline{1,\sum p\_c}$, розмірності n на $\sum p\_c$ (дійсні), кожний стовпець представляє собою новий вихід активного експерименту;
– трійку *pool_n*, яка визначає кількість потоків, що будуть задіяні при використанні паралельних обчислень у підалгоритмах ММГУА (перше значення – кількість потоків для знаходження оцінок коефіцієнтів часткових описів та ЗСК, друге – кількість потоків для паралельного множення матриць у основній формулі МНК, третє – кількість потоків для обернення матриць у основній формулі МНК) (в діапазоні 1..61;1..61;1..61, цілі, за замовчуванням 1;1;1);
– матрицю, що містить надлишковий опис RR_{ij} $i = \overline{1,m}, j = \overline{1,2}$, розмірності m на *2* (перший стовпець містить усі можливі члени надлишкового опису, другий булеве значення, яке показує чи входить член у надлишковий опис).

Функція RA_MGMDH повертає:

– структуру та оцінки коефіцієнтів БПР $\hat{b}_j,\ j = \overline{0,m}$;
– дисперсії оцінок коефіцієнтів БПР $D\hat{b}_j,\ j = \overline{0,m}$;
– ступінь достовірності знайдених оцінок (лінгвістична змінна);
– код помилки та її сутність при виникненні (виродженість задачі).

Функція *RA_to_rational* реалізує перетворення даних довільної структури з дійсного типу у тип раціональних дробів.

Функція RA_to_rational приймає на вхід:

– довільну структуру даних (з тих, що використовуються у бібліотеці) та містять дійсні значення.

Функція RA_to_rational повертає:

– довільну структуру даних (з тих, що використовуються у бібліотеці), яка містить раціональні дроби.

Функція *RA_to_double* що реалізує перетворення даних довільної структури з типу раціональних дробів у дійсний тип з заданою кількістю знаків після коми.

Функція RA_to_double приймає на вхід:

– довільну структуру даних (з тих, що використовуються у бібліотеці) та містять раціональні дроби;

– кількість знаків після коми ε (в діапазоні 0..100, цілі).

Функція RA_to_double повертає:

– довільну структуру даних (з тих, що використовуються у бібліотеці), яка містить дійсні значення.

Функція *RA_decomposition_method*, що реалізує декомпозиційний метод знаходження оцінок БПР.

Функція RA_decomposition_method приймає на вхід:

– кількість спостережень *n* (в діапазоні 2..10000, цілі);

– кількість вхідних детермінованих змінних *m* (в діапазоні 2..100, цілі);

– кількість членів надлишкового опису *m2* (в діапазоні 2..10000, цілі);

– матрицю значень вхідних детермінованих змінних $X_{ij}, i = \overline{1,n}, j = \overline{1,m}$, розмірності $n \times m$ (дійсні);

– кількість повторів активного експерименту *p* (в діапазоні 1..100, цілі, за замовчуванням 1);

– матрицю вихідних значень результатів активного експерименту y_{ij}, $i = \overline{1,n}, j = \overline{1,p}$, розмірності *n* на *p* (дійсні), кожний стовпець представляє собою новий повтор активного експерименту;

– матрицю, що містить надлишковий опис RR_{ij} $i = \overline{1,m}, j = \overline{1,2}$, розмірності *m* на *2* (перший стовпець містить усі можливі члени надлишкового опису, другий булеве значення, яке показує чи входить член у надлишковий опис).

Функція RA_decomposition_method повертає:

– структуру та оцінки коефіцієнтів БПР $\hat{b}_j,\ j = \overline{0,m}$;

– дисперсії оцінок коефіцієнтів БПР $D\hat{b}_j,\ j = \overline{0,m}$;

– код помилки та її сутність при виникненні (виродженість задачі).

Функція *RA_decomposition_active_count*, що визначає необхідну кількість повторів активного експерименту для знаходження оцінок кожного нелінійного члена БПР.

Функція RA_decomposition_active_count приймає на вхід:

– кількість спостережень *n* (в діапазоні 2..10000, цілі);

– кількість вхідних детермінованих змінних *m* (в діапазоні 2..100, цілі);

– кількість членів надлишкового опису *m2* (в діапазоні 2..10000, цілі);

– матрицю значень вхідних детермінованих змінних $X_{ij}, i = \overline{1,n}, j = \overline{1,m}$, розмірності $n \times m$ (дійсні);

– матрицю, що містить надлишковий опис $RR_{ij}\ i = \overline{1,m}, j = \overline{1,2}$, розмірності m на *2* (перший стовпець містить усі можливі члени надлишкового опису, другий булеве значення, яке показує чи входить член у надлишковий опис).

Функція RA_decomposition_active_count повертає:

– кількість повторів активного експерименту необхідна для оцінки нелінійних членів БПР з заданою точністю *p*;

– код помилки та її сутність при виникненні.

Функція *RA_decomposition_method_2*, що надає можливість реалізувати індивідуальний алгоритм декомпозиційного методу знаходження оцінок БПР.

Функція RA_decomposition_method_2 приймає на вхід:

– кількість спостережень *n* (в діапазоні 2..10000, цілі);

– кількість вхідних детермінованих змінних *m* (в діапазоні 2..100, цілі);

– кількість членів надлишкового опису *m2* (в діапазоні 2..10000, цілі);

– матрицю значень вхідних детермінованих змінних $X_{ij}, i = \overline{1,n}, j = \overline{1,m}$, розмірності $n \times m$ (дійсні);

– кількість повторів активного експерименту p (в діапазоні 1..100, цілі, за замовчуванням 1);

– матрицю вихідних значень результатів активного експерименту y_{ij}, $i = \overline{1,n}, j = \overline{1,p}$, розмірності *n* на *p* (дійсні), кожний стовпець представляє собою новий повтор активного експерименту;
– матрицю, що містить надлишковий опис RR_{ij} $i = \overline{1,m}, j = \overline{1,2}$, розмірності *m* на *2* (перший стовпець містить усі можливі члени надлишкового опису, другий булеве значення, яке показує чи входить член у надлишковий опис);
– *k* – член надлишкового опису, для якого буде знайдено оцінку коефіцієнта БПР.

Функція RA_decomposition_method_2 повертає:

– оцінку коефіцієнта БПР $\hat{b}_k$;
– дисперсію оцінки коефіцієнта БПР $D\hat{b}_k$;
– код помилки та її сутність при виникненні (виродженість задачі).

4.3 Проєктування архітектури кросплатформної бібліотеки

Архітектура кросплатформної бібліотеки, що реалізує синтетичний метод побудови БПР на основі повторних активних експериментів, буде мати монолітну архітектуру з використанням компонентно-орієнтованого підходу реалізації її внутрішньої логіки.

У кросплатформній бібліотеці використовується набір окремих незалежних компонентів. Сама бібліотека також реалізована як окремий компонент більш високого рівня з можливістю імпортування в стороннє програмне забезпечення, її загальна архітектура наведена на рис. 4.2.

Далі буде наведено короткий опис компонентів, що входять до складу кросплатформної бібліотеки.

– MPR – основний компонент, фактично сама бібліотека, що реалізує синтетичний метод побудови БПР на основі повторних активних експериментів;
– RA_to_rational – компонент, який перетворює вхідні дані задачі БПР, у раціональні дроби, що є необхідним для забезпечення точності обчислення;

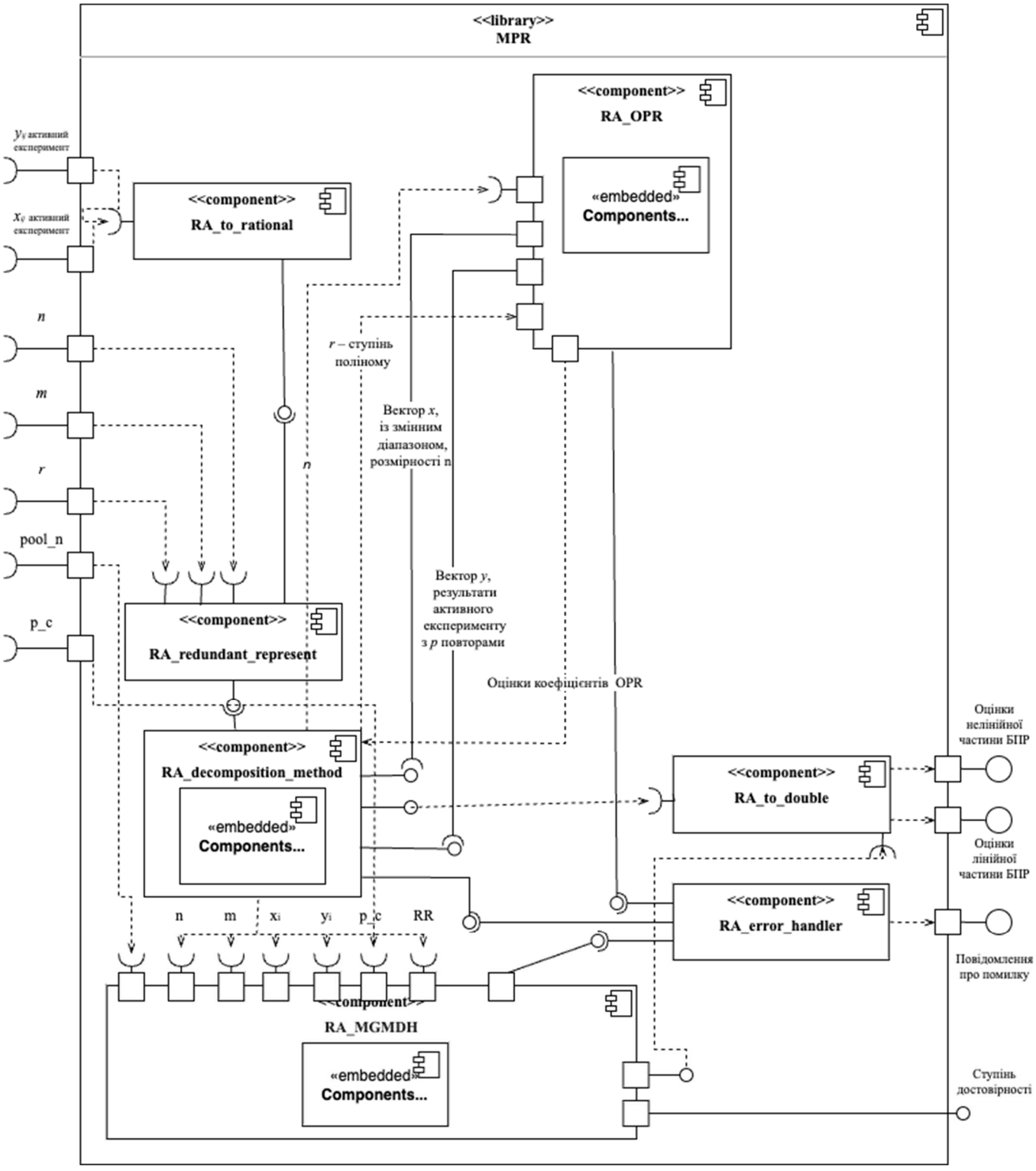

Рисунок 4.2 – Загальна архітектура верхнього рівня кросплатформної бібліотеки

– RA_MGMDH – компонент, який реалізує ММГУА для знаходження оцінок БПР (частина підалгоритмів реалізовується з використанням паралельних обчислень);

– RA_redundant_represent – компонент, який будує надлишковий опис по вхідним даним задачі побудови БПР;
– RA_decomposition_method – компонент, який реалізує декомпозиційний метод знаходження оцінок БПР, працює у зв’язці з RA_OPR;
– RA_OPR – компонент, який знаходить оцінки коефіцієнтів ОПР МНК з використанням лише одного набору НОПФ (необхідний для знаходження оцінок БПР декомпозиційним методом);
– RA_to_double – компонент, який перетворює отримані результати задачі побудови БПР у дійсні числа;
– RA_error_handler – компонент, який обробляє різного роду помилки.

Далі розглянемо деталізовані архітектури окремих компонентів кросплатформної бібліотеки, таких як: RA_OPR (рис. 4.3), RA_decomposition_method (рис. 4.4) та RA_MGMDH (рис. 4.5).

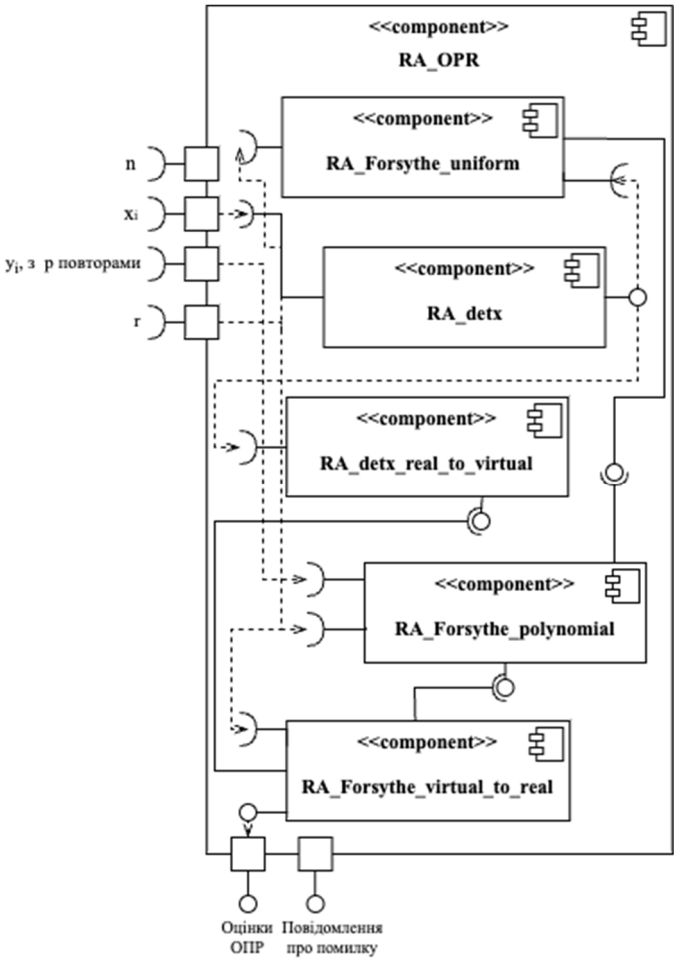

Рисунок 4.3 – Деталізована архітектура компонента побудови ОПР

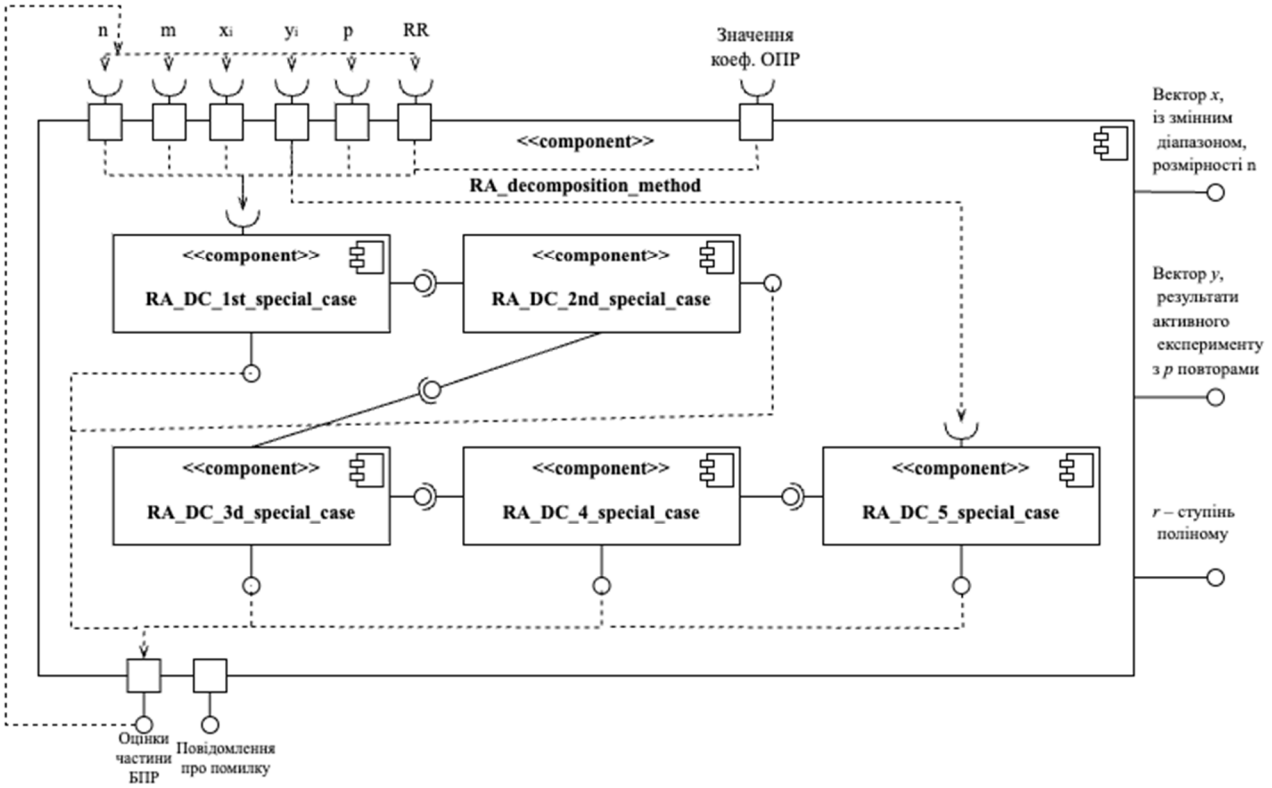

Рисунок 4.4 – Деталізована архітектура компонента, що реалізує декомпозиційний метод побудови БПР

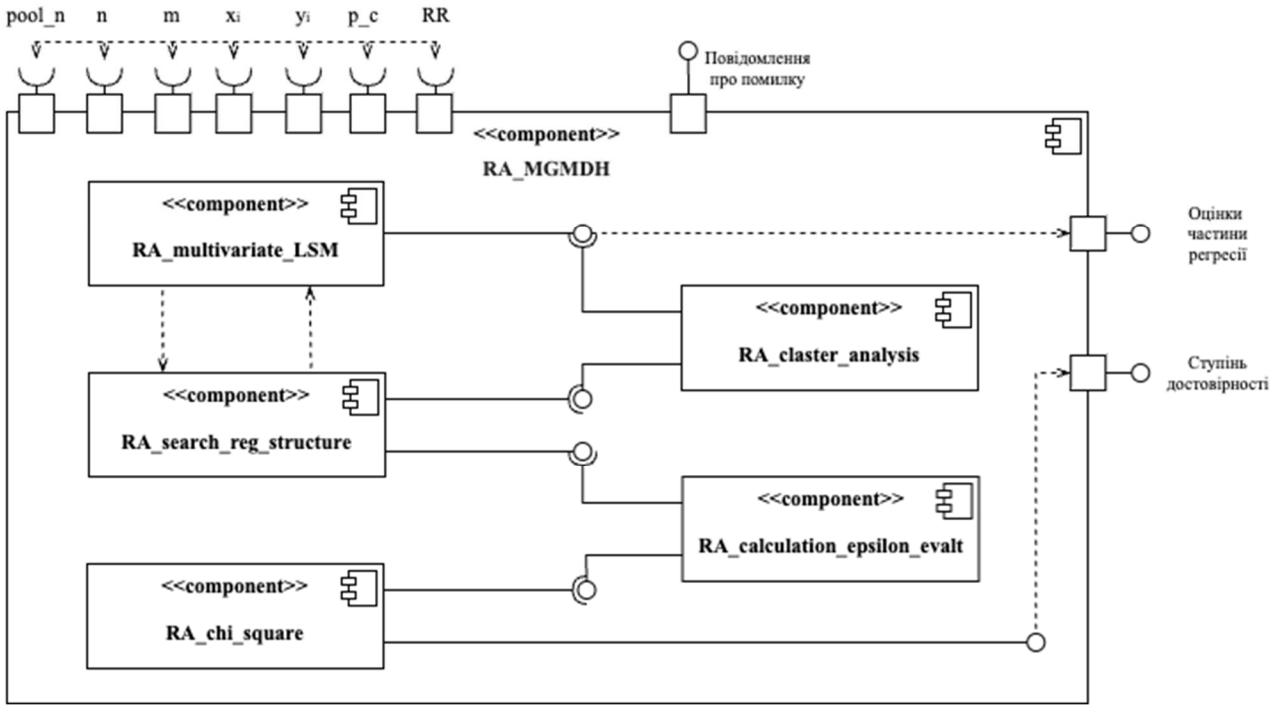

Рисунок 4.5 – Деталізована архітектура компонента, що реалізує ММГУА побудови БПР

Далі буде наведено короткий опис компонентів, що входять до складу компоненту RA_OPR кросплатформної бібліотеки.

– RA_Forsythe_uniform – компонент, який реалізує алгоритм знаходження коефіцієнтів НОПФ заданого ступеня, з заданою точністю на інтервалі з рівномірним розподілом змінної, за замовчуванням містить значення коефіцієнтів НОПФ на інтервалі [–50, 50], n=10;
– RA_detx – компонент, який знаходить значення детермінованої змінної x на заданому інтервалі;
– RA_detx_real_to_virtual – компонент, який знаходить значення детермінованої змінної x на заданому інтервалі проведення активного реального експерименту на базі віртуального експерименту;
– RA_Forsythe_polynomial – компонент, який реалізує МНК з використанням НОПФ для знаходження оцінок коефіцієнтів при нелінійних членах ОПР;
– RA_Forsythe_virtual_to_real – компонент, який реалізує МНК знаходження оцінок коефіцієнтів ОПР з використанням лише одного набору НОПФ.

Далі буде наведено короткий опис компонентів, що входять до складу компоненту RA_decomposition_method кросплатформної бібліотеки.

– RA_DC_1st_special_case – компонент, який реалізує перший частковий випадок декомпозиційного методу знаходження оцінок коефіцієнтів БПР;
– RA_DC_2nd_special_case – компонент, який реалізує другий частковий випадок декомпозиційного методу знаходження оцінок коефіцієнтів БПР;
– RA_DC_3d_special_case – компонент, який реалізує третій частковий випадок декомпозиційного методу знаходження оцінок коефіцієнтів БПР;
– RA_DC_4_special_case – компонент, який реалізує четвертий частковий випадок декомпозиційного методу знаходження оцінок коефіцієнтів БПР;
– RA_DC_5_special_case – компонент, який включає інтерактивний інтерфейс для створення власного алгоритму побудови БПР.

Далі буде наведено короткий опис компонентів, що входять до складу компоненту RA_MGMDH кросплатформної бібліотеки.

– RA_multivariate_LSM – компонент, який реалізує МНК знаходження оцінок БПР; для підвищення швидкодії використовуються паралельні обчислення в операціях множення та обернення матриці в загальній формулі МНК;
– RA_cluster_analysis – компонент, який реалізує суттєве зменшення кількості часткових описів шляхом розбиття всіх коефіцієнтів на два класи. Множина часткових описів гарантовано містить шуканий розв'язок;
– RA_search_reg_structure – компонент, який використовується для знаходження шуканої структури БПР (компонент реалізується у зв'язці з компонентом RA_multivariate_LSM); для підвищення швидкодії використовуються паралельні обчислення у частині знаходженні оцінок коефіцієнтів часткових описів БПР та подальших розрахунків ЗСК;
– RA_calculation_epsilon_evalt – компонент, який використовується для знаходження оцінок реалізацій *E*.
– RA_chi_square – компонент, який містить модифікований критерій χ^2 для визначення ступеня достовірності знайдених оцінок БПР.

Частини коду програмної реалізації кросплатформної бібліотеки знаходяться у Додатку В.

4.4 Дослідження ефективності кросплатформної бібліотеки

У кросплатформній бібліотеці, що реалізує синтетичний метод побудови БПР на основі повторних активних експериментів, оцінки БПР, які було знайдено декомпозиційним методом, теоретично обґрунтовані та не вимагають експериментальних досліджень. Щодо ММГУА, модифікація МГУА полягає в більш ефективному алгоритмі побудови множини допустимих регресій, по яких знаходять розв'язок задачі, при чому суттєво збільшується в порівнянні з класичним МГУА, ймовірність того, що ця множина містить структуру шуканої регресії [46]. Таким чином, ефективність ММГУА за означенням не гірше ефективності класичного МГУА, яка підтверджена статистично значущою кількістю дослідниками. У даній роботі додатково проведено наступні експерименти та зроблено відповідні висновки.

Для невеликої вибірки (до 15 повторів основного експерименту), дисперсії нормального розподілу в межах від 1 до 5, кількістю незалежних змінних від 5 до 40, мінімального модуля значень ненульових еталонних коефіцієнтів БПР від 1 до 10, кількістю проведених експериментів до 500, отримані наступні результати. Відсоток знаходження правильної структури шуканої регресії – 81. У випадках, коли структура регресії знаходилась невірно, кількість нульових коефіцієнтів не перевищувала 3.

Для перевірки коректності та якості ММГУА проведено ряд досліджень з різною значимістю похибки та оцінено отримані результати. За допомогою алгоритму, що викладений у 2.1 знайдено структуру (вилучено змінні з нульовими оцінками коефіцієнтів) та оцінки ненульових коефіцієнтів БЛР заданої надлишковим описом для різних значень дисперсії *DE* від 1% до 30% середнього значення $\left(y_i,\ i = \overline{1,n}\right)$ після чого порівняно знайдені оцінки з еталонними.

Тестова вхідна задача має наступний вигляд:

– розмірність задачі $(n = 2000, m = 4, p = 6)$, де n – кількість спостережень, m – кількість незалежних змінних (факторів), p – кількість повторів активного експерименту. Половина повторів активного експерименту (3 шт.) буде використана для знаходження оцінок, а решта, як перевірочна послідовність;
– істинні значення коефіцієнтів БЛР $(b_0 = 10, b_1 = 0, b_2 = 6, b_3 = 0, b_4 = 3)$.

У табл. 4.1 наведено дослідження для нормального розподілу з нульовим математичним сподіванням та дисперсією похибки від 1 до 45 (приблизно 1–30% від середнього значення $\left(y_i,\ i = \overline{1{,}2000}\right)$).

Із табл. 4.1 можна зробити висновок, що при всіх значеннях структура регресії знаходиться 100%, тобто ММГУА у всіх прикладах виключив усі нульові коефіцієнти, тобто збільшення кількості вхідних даних на порядок приводить до збільшення відсотку отримання правильної структури шуканої регресії від 81% до 100%. а оцінки коефіцієнтів БЛР не відходять більше ніж на 1% від істини на малих значеннях дисперсії (до 10) і не більше ніж 10% на великих (10–45).

Таблиця 4.1 – Дослідження коректності та якості алгоритму при нормальному розподілі похибки

№	Дисперсія	Істинні значення оцінок b_0, b_2, b_4 $(b_1 = 0, b_3 = 0)$	Знайдені значення оцінок $\widehat{b_0}, \widehat{b_2}, \widehat{b_4}$ $(\widehat{b_1} = 0, \widehat{b_3} = 0)$	Модуль різниці	ЗСК	χ^2 (критична обл. 27,6)
1	1	b_0: 10 b_2: 6 b_4: 3	$\widehat{b_0}$: 10,00466 $\widehat{b_2}$: 6,00273 $\widehat{b_4}$: 2,99688	$\lvert\widehat{b_0} - b_0\rvert$: 0,00466 $\lvert\widehat{b_2} - b_2\rvert$: 0,00273 $\lvert\widehat{b_4} - b_4\rvert$: 0,00311	6134,54447	22,96237
2	3	b_0: 10 b_2: 6 b_4: 3	$\widehat{b_0}$: 9,89022 $\widehat{b_2}$: 6,01068 $\widehat{b_4}$: 3,00601	$\lvert\widehat{b_0} - b_0\rvert$: 0,10978 $\lvert\widehat{b_2} - b_2\rvert$: 0,01068 $\lvert\widehat{b_4} - b_4\rvert$: 0,00601	54599,90537	12,94503
3	6	b_0: 10 b_2: 6 b_4: 3	$\widehat{b_0}$: 10,00395 $\widehat{b_2}$: 6,00705 $\widehat{b_4}$: 2,97629	$\lvert\widehat{b_0} - b_0\rvert$: 0,00395 $\lvert\widehat{b_2} - b_2\rvert$: 0,00705 $\lvert\widehat{b_4} - b_4\rvert$: 0,02371	217131,54215	25,26757
4	9	b_0: 10 b_2: 6 b_4: 3	$\widehat{b_0}$: 9,87269 $\widehat{b_2}$: 5,9937 $\widehat{b_4}$: 3,00706	$\lvert\widehat{b_0} - b_0\rvert$: 0,12731 $\lvert\widehat{b_2} - b_2\rvert$: 0,0063 $\lvert\widehat{b_4} - b_4\rvert$: 0,00706	484654,5218	15,07637
5	12	b_0: 10 b_2: 6 b_4: 3	$\widehat{b_0}$: 10,05334 $\widehat{b_2}$: 5,9916 $\widehat{b_4}$: 3,01062	$\lvert\widehat{b_0} - b_0\rvert$: 0,05334 $\lvert\widehat{b_2} - b_2\rvert$: 0,0084 $\lvert\widehat{b_4} - b_4\rvert$: 0,01062	854450,4112	14,5984
6	15	b_0: 10 b_2: 6 b_4: 3	$\widehat{b_0}$: 10,23545 $\widehat{b_2}$: 5,95994 $\widehat{b_4}$: 2,97938	$\lvert\widehat{b_0} - b_0\rvert$: 0,23545 $\lvert\widehat{b_2} - b_2\rvert$: 0,04006 $\lvert\widehat{b_4} - b_4\rvert$: 0,02062	1314934,7194	19,13492
7	18	b_0: 10 b_2: 6 b_4: 3	$\widehat{b_0}$: 9,9047 $\widehat{b_2}$: 5,95343 $\widehat{b_4}$: 3,04148	$\lvert\widehat{b_0} - b_0\rvert$: 0,0953 $\lvert\widehat{b_2} - b_2\rvert$: 0,04657 $\lvert\widehat{b_4} - b_4\rvert$: 0,04148	1914054,09238	15,16067
8	21	b_0: 10 b_2: 6 b_4: 3	$\widehat{b_0}$: 9,94353 $\widehat{b_2}$: 6,05358 $\widehat{b_4}$: 2,98252	$\lvert\widehat{b_0} - b_0\rvert$: 0,05647 $\lvert\widehat{b_2} - b_2\rvert$: 0,05358 $\lvert\widehat{b_4} - b_4\rvert$: 0,01748	2621978,66312	22,99228
9	24	b_0: 10 b_2: 6 b_4: 3	$\widehat{b_0}$: 9,34618 $\widehat{b_2}$: 6,03831 $\widehat{b_4}$: 3,14252	$\lvert\widehat{b_0} - b_0\rvert$: 0,65382 $\lvert\widehat{b_2} - b_2\rvert$: 0,0383 $\lvert\widehat{b_4} - b_4\rvert$: 0,14252	3387699,92796	18,72791
10	27	b_0: 10 b_2: 6 b_4: 3	$\widehat{b_0}$: 10,3758 $\widehat{b_2}$: 5,99068 $\widehat{b_4}$: 2,95809	$\lvert\widehat{b_0} - b_0\rvert$: 0,3758 $\lvert\widehat{b_2} - b_2\rvert$: 0,00932 $\lvert\widehat{b_4} - b_4\rvert$: 0,04191	4417679,75177	18,75059
11	30	b_0: 10 b_2: 6 b_4: 3	$\widehat{b_0}$: 10,60716 $\widehat{b_2}$: 5,83242 $\widehat{b_4}$: 3,10484	$\lvert\widehat{b_0} - b_0\rvert$: 0,60716 $\lvert\widehat{b_2} - b_2\rvert$: 0,16758 $\lvert\widehat{b_4} - b_4\rvert$: 0,10484	5290967,91948	12,435
12	33	b_0: 10 b_2: 6 b_4: 3	$\widehat{b_0}$: 10,7442 $\widehat{b_2}$: 5,85103 $\widehat{b_4}$: 2,9434	$\lvert\widehat{b_0} - b_0\rvert$: 0,7442 $\lvert\widehat{b_2} - b_2\rvert$: 0,14897 $\lvert\widehat{b_4} - b_4\rvert$: 0,0566	6639786,28114	32,80086

Кінець табл. 4.1.

№	Дисперсія	Істинні значення оцінок b_0, b_2, b_4 $(b_1 = 0, b_3 = 0)$	Знайдені значення оцінок $\widehat{b_0}, \widehat{b_2}, \widehat{b_4}$ $(\widehat{b_1} = 0, \widehat{b_3} = 0)$	Модуль різниці	ЗСК	χ^2 (критична обл. 27,6)
13	36	b_0: 10 b_2: 6 b_4: 3	$\widehat{b_0}$: 10,12907 $\widehat{b_2}$: 5,88538 $\widehat{b_4}$: 3,02167	$\lvert\widehat{b_0} - b_0\rvert$: 0,12907 $\lvert\widehat{b_2} - b_2\rvert$: 0,11462 $\lvert\widehat{b_4} - b_4\rvert$: 0,02167	7931400,62274	22,3888
14	39	b_0: 10 b_2: 6 b_4: 3	$\widehat{b_0}$: 10,18048 $\widehat{b_2}$: 6,06844 $\widehat{b_4}$: 2,94954	$\lvert\widehat{b_0} - b_0\rvert$: 0,18048 $\lvert\widehat{b_2} - b_2\rvert$: 0,06844 $\lvert\widehat{b_4} - b_4\rvert$: 0,05046	9419414,41505	21,43799
15	42	b_0: 10 b_2: 6 b_4: 3	$\widehat{b_0}$: 9,80537 $\widehat{b_2}$: 6,00118 $\widehat{b_4}$: 2,90418	$\lvert\widehat{b_0} - b_0\rvert$: 0,19463 $\lvert\widehat{b_2} - b_2\rvert$: 0,00118 $\lvert\widehat{b_4} - b_4\rvert$: 0,09582	10559822,62532	16,48089
16	45	b_0: 10 b_2: 6 b_4: 3	$\widehat{b_0}$: 8,68473 $\widehat{b_2}$: 6,20007 $\widehat{b_4}$: 3,07019	$\lvert\widehat{b_0} - b_0\rvert$: 1,31527 $\lvert\widehat{b_2} - b_2\rvert$: 0,20007 $\lvert\widehat{b_4} - b_4\rvert$: 0,07019	12096673,49536	15,7087

Далі буде досліджено збільшення середнього відносного прискорення реалізації ММГУА з використанням паралельних обчислень в порівняні з його послідовною версією.

Дослідження буде проводитись за допомогою процесора Apple M1, 3.2 GHz, ОЗП 16 ГБайт, на задачах розмірності від 5x500 до 50x5000 з кроком 5x500 (кількість вхідних незалежних змінних х кількість спостережень активного експерименту), при цьому буде обрано 8 потоків для обчислення оцінок коефіцієнтів часткових описів, 8 потоків для паралельної реалізації множення матриць у основній формулі МНК та відповідно 16 потоків для паралельної реалізації обертання матриць у основній формулі МНК. Результати досліджень наведені у табл. 4.2.

Як видно з табл. 4.2, на сучасних процесорах ефективність ММГУА зростає майже у 3 рази. Варто відмітити, що для задач великої розмірності (35 змінних та 3500 спостережень) час побудови БПР ММГУА стає неприйнятно великим. Розв’язати цю проблему у подальшому можна, застосовуючи мультипроцесорні обчислювальні комплекси або спеціалізовані бібліотеки, які дозволяють використовувати в обчисленнях графічний процесор.

Таблиця 4.2 – Дослідження ефективності ММГУА

Розмірність	Послідовна версія, с	Паралельна версія, с	Середнє відносне прискорення по 6 задачах
5x500	0,69	1,43	0,48251748
10x1000	3,36	3,64	0,92307692
15x1500	10,49	7,69	1,36410923
20x2000	71,56	33,53	2,13420817
25x2500	110,48	45,64	2,42068361
30x3000	387,36	124,33	3,11557951
35x3500	11546,88	2664,64	4,33337336
40x4000	25585,89	6016,12	4,25288891
45x4500	48506,74	11370,09	4,26617028
50x5000	77908,51	19124,5	4,07375409
Середнє відносне прискорення:			2,73663616

4.5 Вимоги до користування програмним засобом

4.5.1 Вимоги до користувача з базовими навичками

Користувач, який володіє базовими навичками у області програмування та статистики та бажає отримати розв’язок задачі побудови БПР у автоматизованому режимі повинен підключити кросплатформну бібліотеку у власний застосунок та передати у функції бібліотеки (див. п. 4.2) наступні вхідні дані – ввести (згенерувати) надлишковий опис БПР, ввести $\forall c_i, d_i$, p – максимальна кількість повторів основного експерименту, задання заданої точності оцінок коефіцієнтів при нелінійних членах БПР. Після цього необхідно ввести дані по проведенню повторних активних експериментів та результати активного експерименту.

В цьому випадку функції кросплатформної бібліотеки виконають наступні дії.

Аналізується надлишковий опис і визначається, що він відноситься до класу, в якому всі лінійні системи мають лише одну змінну, якщо це не так, то задача розв’язується повністю ММГУА. В протилежному випадку, програма аналізує можливості де-

композиційного методу та видає вимоги для проведення відповідної кількості повторних активних експериментів для побудови ОПР та множину коефіцієнтів, які будуть оцінені, формування БПР заданої надлишковим описом, що буде розв'язано ММГУА, формування даних для повторного активного експерименту, по результатам якого реалізується ММГУА. Видача користувачу даних та інструкцію по проведенню результуючого активного експерименту. За результатами виконання синтетичного методу користувачу буде виданий кінцевий результат, який містить структуру БПР, знайдені декомпозиційним методом оцінки коефіцієнтів БПР та їх дисперсії (у випадку якщо декомпозиційний метод був задіяний) та оцінки коефіцієнтів БПР з оцінками їх дисперсій, знайдені за допомогою ММГУА, з оцінкою результатів (має високу ступінь достовірності; задовільну ступінь достовірності; результат недостовірний).

4.5.2 Вимоги до користувача з розширеними навичками

Користувач, який володіє розширеними навичками у області програмування та статистики та бажає отримати розв'язок задачі побудови БПР, може ознайомитись з детальною інструкцією по роботі з кросплатформною бібліотекою, у якій описані теоретичні положення та практичні рекомендації з використання синтетичного методу.

Далі повторюються вимоги пункту 4.5.1 з наступною модифікацією. Користувач за допомогою функцій кросплатформної бібліотеки у частині декомпозиційного методу програмує індивідуальний алгоритм розв'язку задачі на основі теоретичних положень синтетичного методу. У частині ММГУА все залишається без змін (див. пункт 4.5.1).

4.6 Висновок до розділу 4

У розділі 4 проведено ґрунтовне дослідження основних аспектів пов'язаних з розробкою кросплатформної бібліотеки, що реалізує синтетичний метод побудови БПР на основі повторних активних експериментів.

Зокрема, було проведено ґрунтовний аналіз засобів розробки програмного забезпечення, технічних засобів та допоміжних програмних засобів, після чого було обрано ті, які було використано при розробці кросплатформної бібліотеки. Особливу

увагу було приділено реалізації внутрішньої логіки кросплатформної бібліотеки, яка дозволяє працювати з бібліотекою різним групам користувачів з точки зори їх підготовки у областях програмування та математичної статистики.

Після цього були детально описані функції кросплатформної бібліотеки, викликаючи які користувач зможе виконати побудову БПР синтетичним методом, при цьому користувач зможе в інтерактивному режимі гнучко керувати складовими синтетичного методу. Також функції кросплатформної бібліотеки, які реалізують синтетичний метод, можна використовувати окремо, наприклад для побудови ОПР.

Використовуючи монолітну архітектуру програмного забезпечення та компонентно-орієнтований підхід, було спроектовано та представлено загальну архітектуру кросплатформної бібліотеки та деталізовано архітектуру деяких її компонентів.

З кросплатформною бібліотекою проведено ряд досліджень, що доводять її якість, коректність та ефективність. Наведені загальні вимоги до користувачі бібліотеки, зокрема наведені інструкції по роботі з кросплатформною бібліотекою для користувачів з базовою підготовкою у області програмування та математичної статистики та користувачів з розширеними навичками.

ВИСНОВКИ

В дисертаційній роботі розв’язане актуальне наукове завдання створення методів та програмних засобів побудови багатовимірної поліноміальної регресії, заданої надлишковим описом, на основі побудови послідовності одновимірних поліноміальних регресій з використанням лише одного набору нормованих ортогональних поліномів Форсайта в умовах повторного активного експерименту.

При цьому отримано наступні наукові та практичні результати:

1. Розроблено синтетичний метод побудови багатовимірної поліноміальної регресії, заданої надлишковим описом, що відрізняється від існуючих тим, що органічно поєднує риси класичного методу з ефективністю евристичних методів.

2. Розроблено модифікований метод групового урахування аргументів, що входить до складу синтетичного методу. Модифікація полягає в тому, що завдання багатовимірної поліноміальної регресії надлишковим описом дозволяє множину часткових описів, один з яких містить шукану структуру регресії, знаходити не з використанням багаторівневого селекційного алгоритму, а на основі ефективного алгоритму розбиття множини коефіцієнтів на два класи, при чому коефіцієнти з першого класу входять в кожний частковий опис, що суттєво зменшує кількість часткових описів і підвищує ймовірність того, що один з них має структуру шуканої регресії.

3. Розроблено метод побудови одновимірної поліноміальної регресії на основі довільного повторного активного експерименту з використанням лише одного НОПФ, що входить до складу синтетичного методу.

4. Розроблено декомпозиційний метод, що входить до складу синтетичного методу та містить теоретично обґрунтовані умови, при виконанні яких оцінки коефіцієнтів при нелінійних членах багатовимірної поліноміальної регресії знаходяться з наперед заданою точністю. Оригінальність методу полягає в тому, що він багатовимірну задачу зводить до послідовної побудови відповідних одновимірних поліноміальних регресій.

5. Обґрунтовано можливість знаходження НОПФ з наперед заданою точністю, як необхідної умови використання декомпозиційного методу, яка досягається за рахунок представлення даних у вигляді раціональних дробів та застосування до них символьних обчислень.

6. Теоретично обґрунтовано зменшення обчислювальної складності програмного забезпечення реалізації методу найменших квадратів на основі повторних експериментів, що полягає в заміні операцій з матрицями повного активного експерименту операціями з матрицями основного експерименту суттєво меншої розмірності.

7. Запропоновано архітектуру кросплатформної бібліотеки для реалізації синтетичного методу та його складових, яка дозволяє використовувати її компоненти, як окремо, так і в цілому для розв'язання прикладних задач побудови регресійних моделей.

8. Обґрунтовано можливість реалізації асинхронних паралельних обчислень при використанні модифікованого методу групового урахування аргументів, що дозволило досягти середнього відносного прискорення обчислень в 2,73 рази, а для задач великої розмірності (35 змінних, 3500 спостережень) – у більш, ніж 4 рази.

9. Реалізовано кросплатформну бібліотеку та її програмний інтерфейс для побудови регресійних моделей, за допомогою якого можна знаходити структуру багатовимірної поліноміальної регресії та оцінки її коефіцієнтів. Програмний інтерфейс дозволяє працювати з бібліотекою в автоматизованому або в інтерактивному режимі.

Таким чином, усі поставлені у даному науковому дослідженні завдання повністю виконані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Abdulrahman, A.T., Alshammari, N.S.: Factor analysis and regression analysis to find out the influencing factors that led to the countries' debt crisis. Advances and Applications in Statistics **78**, 1–16 (2022). https://doi.org/10.17654/0972361722047

2 Agresti A.: Categorical Data Analysis. Wiley-Interscience, New Jork (2002). https://doi.org/10.1002/0471249688

3 Ajitsaria A.: What Is the Python Global Interpreter Lock (GIL)? https://realpython.com/python-gil/ (2018). Accessed 30 Aug 2023

4 Apple M1 microprocessor technical characteristics. https://www.apple.com/ua/business/mac/pdf/Apple-at-Work-M1-Overview.pdf (2021). Accessed 30 Aug 2023

5 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 **9** (3), 16–26 (2019). https://doi.org/10.5815/ijeme.2019.03.02

6 Braak, C.J.F. ter, Looman, C.W.N.: Regression. In: Jongman, R.H.G., ter Braak, C.J.F., van Tongeren, O.F.R. Data analysis in community and landscape ecology, pp. 29–77. Cambridge University Press, Cambridge, 1995. https://library.wur.nl/WebQuery/wurpubs/436954. Accessed 30 Aug 2023

7 Brown, E.: Learning JavaScript: JavaScript Essentials for Modern Application Development. O'Reilly Media, Inc, Sebastopol (2016)

8 Brownlee J.: SuperFastPython – ThreadPoolExecutor vs. the Global Interpreter Lock. https://superfastpython.com/threadpoolexecutor-vs-gil/#ThreadPoolExecutor_vs_the_Global_Interpreter_Lock (2021). Accessed 26 Aug 2023

9 Buckley, J.J., Feuring T.: Linear and non-linear fuzzy regression: Evolutionary algorithm solutions. Fuzzy Sets and Systems **112** (3), 381–394 (2000). https://doi.org/10.1016/s0165-0114(98)00154-7

10 concurrent.futures – Python 3.11.5 documentation. https://docs.python.org/3/library/concurrent.futures.html (2023). Accessed 30 Aug 2023

11 Cormen, T.H., et al.: Introduction to Algorithms, 3rd edn. MIT Press (2009)

12 Deductor Functions. https://basegroup.com/deductor/function (2021). Accessed 20 Feb 2021

13 Draper, N.R., Smith H.: Applied Regression Analysis, 3rd edn. Wiley & Sons, New York (1998). https://doi.org/10.1002/9781118625590

14 Drozd, V., Holovchenko, M.: Metody ta prohramni zasoby pobudovy neliniynykh polinomial'nykh rehresiy z vykorystannyam normovanykh ortohonal'nykh polinomiv Forsayta (Методи та програмні засоби побудови нелінійних поліноміальних регресій з використанням нормованих ортогональних поліномів Форсайта; Methods and software for constructing nonlinear polynomial regressions using normalized orthogonal polynomials of Forsythe). Paper presented at Inzheneriya prohramnoho zabezpechennya i peredovi informatsiyni tekhnolohiyi (SoftTech-2021), Kyiv, 22–26 Nov 2021, pp. 82–86 (2021)

15 Earnest, T.J.: Strassen's Algorithm on the Cell Broadband Engine. Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, pp. 1–7 (2007). https://web.archive.org/web/20100612150812/http://www.mc2.umbc.edu/docs/earnest.pdf. Accessed 30 Aug 2023

16 Flitman, A.M.: Towards analysing student failures: neural networks compared with regression analysis and multiple discriminant analysis. Computers & Operations Research **24** (4), 367–377 (1997). https://doi.org/10.1016/s0305-0548(96)00060-3

17 ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics. https://cran.r-project.org/package=ggplot2 (2023). Accessed 30 Aug 2023

18 Giroux, B.: ttcrpy: A Python package for traveltime computation and raytracing. SoftwareX **16**, 100834 (2021). https://doi.org/10.1016/j.softx.2021.100834

19 Goetz, B., et al.: Java Concurrency in Practice, 1st edn, p. 384. Addison-Wesley Professional, Boston (2006)

20 Hejlsberg, A., et al. The C# Programming Language. Pearson Education, New York (2008)

21 Hofmeister, C., Nord, R., Soni D.: Applied Software Architecture. Addison-Wesley Professional, Boston (2000)

22 Hudson, D.J.: Statistics Lectures, Volume 2: Maximum Likelihood and Least Squares Theory. CERN Reports 64(18). CERN, Geneva (1964). https://doi.org/10.5170/CERN-1964-018

23 Igual, L., et al.: Introduction to Data Science. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-50017-1

24 IntelliJ IDEA – the Leading Java and Kotlin IDE. https://www.jetbrains.com/idea/ (2023). Accessed 30 Aug 2023

25 Ivakhnenko, A.G.: Modelirovanie Slozhnykh Sistem (Complex Systems Modelling). Vyshcha shkola, Kyiv (1987) [in Russian]

26 Johnson, R.A., Wichern D.W.: Applied Multivariate Statistical Analysis, 5th edn. Prentice-Hall, Upper Saddle River, NJ (2002)

27 Kadiyala, A., Kumar, A.: Applications of Python to evaluate environmental data science problems. Environmental Progress & Sustainable Energy **36** (6), 1580–1586 (2017). https://doi.org/10.1002/ep.12786

28 Kapanoglu, M., Koc I.O., Erdogmus S.: Genetic algorithms in parameter estimation for nonlinear regression models: an experimental approach. Journal of Statistical Computation and Simulation **77** (10), 851–867 (2007). https://doi.org/10.1080/10629360600688244

29 Knowles, D., Parts, L., Glass, D., Winn, J.M.: Modeling skin and ageing phenotypes using latent variable models in infer.net. Paper presented at Predictive Models in Personalized Medicine Workshop NIPS 2010, Vancouver, 6-11 Dec 2010. URL: https://www.researchgate.net/publication/241194775. Accessed 30 Aug 2023

30 Len, B., Clements, P., Kazman, R.: Software Architecture in Practice. Addison-Wesley Professional, Boston (2012)

31 Lio, W., Liu, B.: Uncertain maximum likelihood estimation with application to uncertain regression analysis. Soft Computing **24** (13), 9351–9360. https://doi.org/10.1007/s00500-020-04951-3

32 Liu, S.S., Zhu, Y. Simultaneous Maximum Likelihood Estimation for Piecewise Linear Instrumental Variable Models. Entropy **24** (9), 1235 (2022). https://doi.org/10.3390/e24091235

33 Lutz, M.: Learning Python, 3rd edn, p. 1645. O'Reilly Media, Inc, Sebastopol (2013)

34 Mallet, A.A.: A maximum likelihood estimation method for random coefficient regression models. Biometrika **73** (3), 645–656 (1986). https://doi.org/10.1093/biomet/73.3.645

35 Mohan, S.: Parameter estimation of nonlinear Muskingum models using genetic algorithm. Journal of hydraulic engineering **123** (2), 137–142 (1997). https://doi.org/10.1061/(asce)0733-9429(1997)123:2(137)

36 multiprocessing – Process-based parallelism – Python 3.11.5 documentation. https://docs.python.org/3/library/multiprocessing.html#module-multiprocessing (2023). Accessed 30 Aug 2023

37 Nagpal, A., Gabrani, G.: Python for data analytics, scientific and technical applications. 2019 Amity international conference on artificial intelligence (AICAI), IEEE, pp. 140–145, Dubai, 04–06 Feb 2019 (2019). https://doi.org/10.1109/aicai.2019.8701341

38 Nastenko, E., Pavlov, V., Boyko, G., Nosovets, O.: Mnogokriterial'nyy algoritm shagovoy regressii (Multicriteria stepwise regression algorithm). Biomedychna inzheneriya i tekhnologiya **2020** (3), 48–53. https://doi.org/10.20535/2617-8974.2020.3.195661

39 Nelli, F.: Python data analytics: Data analysis and science using PANDAs, Matplotlib and the Python Programming Language. Apress, New York (2015)

40 Numpy documentation. https://numpy.org/doc/stable/ (2022). Accessed 30 Aug 2023

41 Ozgur, C., et al.: MatLab vs. Python vs. R. Journal of Data Science **15** (3), 355–372 (2021). https://doi.org/10.6339/jds.201707_15(3).0001

42 Öztürk, O.B., Başar E.: Multiple linear regression analysis and artificial neural networks based decision support system for energy efficiency in shipping. Ocean Engineering **243**, 110209 (2022). https://doi.org/10.1016/j.oceaneng.2021.110209

43 Pavlov, A. A.: Estimating with a given accuracy of the coefficients at nonlinear terms of univariate polynomial regression using a small number of tests in an arbitrary limited active experiment. Bulletin of National Technical University “KhPI”. Series: System analysis, control and information technologies. **2** (6), 3–7 (2021). https://doi.org/10.20998/2079-0023.2021.02.01

44 Pavlov, A. A., Holovchenko, M. N., 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), 3–9 (2023). https://doi.org/10.20998/2079-0023.2023.01.01

45 Pavlov, A., Holovchenko, M., Drozd, V.: Syntetychnyy metod pobudovy bahatovymirnoyi polinomial'noyi rehresiyi (Синтетичний метод побудови багатовимірної поліноміальної регресії; A synthetic method of a multivariate polynomial regression construction). Paper presented at Kompleksne zabezpechennya yakosti tekhnolohichnykh protsesiv ta system (KZYaTPS-2023), Chernihiv, 25–26 May 2023, Vol. 2, pp. 272–273 (2023). http://ir.stu.cn.ua/123456789/28221 (2023) Accessed 30 Aug 2023

46 Pavlov, A., Holovchenko, M., Mukha, I., et al. 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: Advances in Computer Science for Engineering and Education VI. ICCSEEA 2023. Lecture Notes on Data Engineering and Communications Technologies **181**, 207–222 (2023). https://doi.org/10.1007/978-3-031-36118-0_19

47 Pavlov, A., 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. In: Advances in Computer Science for Engineering and Education V. ICCSEEA 2022. Lecture Notes on Data Engineering and Communications Technologies **134**, 288–303 (2022). https://doi.org/10.1007/978-3-031-04812-8_25

48 Pavlov, A., Holovchenko, M., Revich, M.: Metod otsinky koefitsiyentiv pry liniynykh chlenakh bahatovymirnoyi polinomial'noyi rehresiyi, zadanoyi nadlyshkovym opysom (Метод оцінки коефіцієнтів при лінійних членах багатовимірної поліноміальної регресії, заданої надлишковим описом; A method to estimate coefficients at linear terms of multivariate polynomial regression given by a redundant representation). Adaptyvni systemy avtomatychnoho upravlinnya: mizhvidomchyy nauk.-tekhn. zbirnyk. Kyiv: NTUU «KPI». **1** (40), 110–117 (2022). https://doi.org/10.20535/1560-8956.40.2022.261665 [in Ukrainian]

49 Pavlov, A., Holovchenko, M.: Modified method of constructing a multivariate linear regression given by a redundant description. Bulletin of National Technical University “KhPI”. Series: System analysis, control and information technologies **2** (8), 3–8 (2022). https://doi.org/10.20998/2079-0023.2022.02.01

50 Pavlov, A., Holovchenko, M.: Postroenie odnomernoy i mnogomernoy polinomial'noy regressii po izbytochnomu opisaniyu s ispol'zovaniem aktivnogo eksperimenta

(Univariate and multivariate polynomial regression construction from a redundant representation using an active experiment). Visnyk Natsional'noho tekhnichnoho universytetu «KhPI». Seriya: Systemnyy analiz, upravlinnya ta informatsiyni tekhnolohiyi **1** (3), 3–8 (2020). https://doi.org/10.20998/2079-0023.2020.01.02 [in Russian]

51 Pavlov, A.A., Chekhovskiy, A.V.: Modificirovannyj algoritm postroenija mnogomernoj polinomial'noj regressii po izbytochnomu opisaniju (Modified algorithm for constructing a multivariate polynomial regression by a redundant representation). Visnyk Nats. Techn. Univ. KhPI Series: System Anal., Upravl. Ta Inform. Technol. **29**, 50–54 (2012) [in Russian]

52 Pavlov, A.A., Chekhovskiy, A.V.: Postroenie mnogomernoy polinomial'noy regressii. Aktivnyy eksperiment s ogranicheniyami (Построение многомерной полиномиальной регрессии. Активный эксперимент с ограничениями; Multivariate polynomial regression construction. Active experiment with limitations). Visnyk Nats. Techn. Univ. KhPI Series: System Anal., Upravl. Ta Inform. Technol. **2009** (4), 174–185 (2009) [in Russian]. http://samit.khpi.edu.ua/article/view/116735

53 Pavlov, A.A., Holovchenko, M.N., Drozd, V.V., Revych, M.M.: Doslidzhennya efektyvnosti metodu pobudovy bahatovymirnoyi liniynoyi rehresiyi, zadanoyi nadlyshkovym opysom (Дослідження ефективності методу побудови багатовимірної лінійної регресії, заданої надлишковим описом; Study of the efficiency of a method of building a multivariate linear regression given by a redundant representation). In: Materialy Vseukrayins'koyi naukovo-praktychnoyi konferentsiyi molodykh vchenykh ta studentiv “Inzheneriya prohramnoho zabezpechennya i peredovi informatsiyni tekhnolohiyi” (SoftTech-2022), Kyiv, 22–26 May and 22–25 Oct 2022, pp. 10–13. NTUU “KPI”, Kyiv. https://drive.google.com/file/d/1CP9EaBTT_rJAXsINbanSVGnP2jkg9FJ0/view (2022). Accessed 30 Aug 2023 [in Ukrainian]

54 Pavlov, A.A., Holovchenko, M.N., 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), 3–8 (2022). https://doi.org/10.20998/2079-0023.2022.01.01

55 Pavlov, A.A., Kalashnik, V.V., Kovalenko, D.A.: Postroenie mnogomernoi polinomialnoi regressii. Regressiya s povtoryayuschimisya argumentami vo vhodnyh dannyh (Multivariate polynomial regression construction. Regression with repeated arguments in the input data). Visnyk NTUU KPI Inform., Oper. and Comput. Sci. **62**, 57–61 (2015) [in Russian]

56 Pavlov, A.A., Kovalenko D.A. Nekorektnist' vykorystannya metodiv bahatovymirnoho rehresiynoho analizu dlya vypadku odnovymirnoho polinomial'-noho analizu (Некоректність використання методів багатовимірного регресій-ного аналізу для випадку одновимірного поліноміального аналізу; Incorrectness of using methods of multivariate regression analysis for the case of univariate polyno-mial analysis). Naukovyy ohlyad **3** (46), 94–100 (2018). https://naukajournal.org/index.php/naukajournal/article/view/1502. Accessed 30 Aug 2023 [in Ukrainian]

57 PyPI – the package installer for Python. https://pypi.org/project/pip/ (2023). Accessed 30 Aug 2023

58 Python packages with the Python Package Index. https://pypi.org/ (2023). Accessed 30 Aug 2023

59 Quinn, M.J.: Parallel Programming in C with MPI and OpenMP. McGraw-Hill, New York (2004)

60 Quintana, E.S., et al.: A note on parallel matrix inversion. SIAM Journal of scientific computing **22** (5), 1762–1771 (2001). https://doi.org/10.1137/s1064827598345679

61 Quora – Are local variables in a python function thread safe? https://www.quora.com/Are-local-variables-in-a-python-function-thread-safe (2019). Accessed 30 Aug 2023

62 Rajković, D., et al.: Artificial neural network and random forest regression models for modelling fatty acid and tocopherol content in oil of winter rapeseed. Journal of Food Composition and Analysis **115**, 105020 (2023). https://doi.org/10.1016/j.jfca.2022.105020

63 Raschka, S.: Python Machine Learning. Packt Publishing Ltd., Birmingham (2015). pp. 260–288.

64 Richards, M., Ford N.: Fundamentals of Software Architecture: An Engineering Ap-proach. O'Reilly Media, Inc, Sebastopol (2020).

65 Richards, M.: Software Architecture Patterns. O'Reilly Media, Inc, Sebastopol (2015)

66 Ruff, L., et al.: Deep one-class classification. Paper presented at 35th International Conference on Machine Learning, 80th edn. PMLR, 4393–4402 (2018). http://proceedings.mlr.press/v80/ruff18a/ruff18a.pdf. Accessed 30 Aug 2023

67 Schmidt, D.C., et al.: Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects. Vol. 2. John Wiley & Sons, Hoboken (2013)

68 Scikit-learn Library: Machine learning in Python. https://scikit-learn.org/stable/ (2023). Accessed 30 Aug 2023

69 Scott, J.T.: Factor Analysis and Regression. Econometrica **34** (3), 552–562 (1966). https://doi.org/10.2307/1909769

70 Sebesta, R.W.: Concepts of Programming Languages, 11th edn. Pearson, Addison Wesley (2015)

71 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 **13** (1), 1–14 (2021). https://doi.org/10.5815/ijieeb.2021.01.01

72 Sinha, P.: Multivariate polynomial regression in data mining: Methodology, problems and solutions. International Journal of Scientific & Engineering Research **4** (12), 962–965 (2013). https://www.ijser.org/onlineResearchPaperViewer.aspx?Multivariate-Polynomial-Regression-in-Data-Mining-Methodology.pdf. Accessed 30 Aug 2023

73 SPSS Statistics. https://www.ibm.com/products/spss-statistics (2023). Accessed 30 Aug 2023

74 TIBCO Statistica 14.1.0. https://docs.tibco.com/products/tibco-statistica-14-1-0 (2023). Accessed 30 Aug 2023

75 Sufyan U., ed.: Mastering C++ Programming Language: A Beginner's Guide. CRC Press, Boca Raton (2022)

76 Sur, P., Candès, E. J.: A modern maximum-likelihood theory for high-dimensional logistic regression. Proceedings of the National Academy of Sciences **116** (29), 14516–14525 (2019). https://doi.org/10.1073/pnas.1810420116

77 SymPy, a Python library. https://www.sympy.org/en/ (2023). Accessed 30 Aug 2023

78 Tam, V.W.Y., et al. A prediction model for compressive strength of CO_2 concrete using regression analysis and artificial neural networks. Construction and Building Materials **324**, 126689 (2022). https://doi.org/10.1016/j.conbuildmat.2022.126689

79 The R Manuals. https://cran.r-project.org/manuals.html (2023). Accessed 30 Aug 2023

80 Thread-based parallelism – Python 3.11.5 documentation. https://docs.python.org/3/library/threading.html (2023). Accessed 30 Aug 2023

81 Tiwari U.K., Kumar S.: Component-Based Software Engineering. Taylor & Francis, Boca Raton (2020). https://doi.org/10.1201/9780429331749

82 Zgurovsky, M.Z., Pavlov, A.A.: Prinyatie resheniy v setevykh sistemakh s ogranichennymi resursami (Decision making in network systems with limited resources). Naukova dumka, Kyiv (2010) [in Russian]

83 Zgurovsky, M.Z., Pavlov, A.A.: Trudnoreshaemye zadachi kombinatornoy optimizatsii v planirovanii i prinyatii resheniy (Intractable problems of combinatorial optimization in planning and decision making). Naukova dumka, Kyiv (2016) [in Russian]

84 Zgurovsky, M.Z., Pavlov, A.A.: Combinatorial Optimization Problems in Planning and Decision Making: Theory and Applications, 1st edn. Springer, Cham (2019) https://doi.org/10.1007/978-3-319-98977-8

ДОДАТОК А РЕЗУЛЬТАТИ ПОВТОРНОГО АКТИВНОГО ЕКСПЕРИМЕНТУ ДЛЯ СИНТЕТИЧНОГО МЕТОДУ

Реалізації випадкової величини *E* для повторного активного експерименту наведено в табл. А.1:

Таблиця А.1 – Реалізації випадкової величини *E* для повторного активного експерименту

–3,158352248	–4,929794572	3,977742962	0,192297548
–4,451108744	5,380389356	–1,017437715	–3,078263546
–2,568847743	0,330461557	1,045656851	1,795596311
2,748525408	–5,27131471	3,490245972	1,563272431
3,47845814	–0,368830602	–3,407886017	–0,612604184
–8,482768032	5,217754236	5,980252531	–2,96059646
–2,509666512	2,75907476	–0,692729945	–6,345890921
–4,171060146	–0,019034998	0,488874113	2,329062411
–3,838983583	8,14213737	–5,282134373	4,183212541
2,648117618	–4,767886291	–7,029179334	–0,628234755
–7,443903615	–1,536696693	2,851386365	–0,534322037
3,246004957	–3,113142135	–0,728863494	–0,029833775
2,256003894	–0,144287508	3,113795016	–3,095221572
–1,084198688	3,590914657	–3,686652946	0,488265297
2,522730751	0,541436484	–1,441592882	–2,897990101
2,505582532	3,545931452	5,665455404	0,965502349
–2,598115542	–2,499723893	–2,415777414	–3,282157805
4,724129211	0,490038631	1,229833268	–6,275057926
2,907870424	2,785975544	6,815697464	–0,953396446
5,871480602	–3,791311409	3,15552422	2,399862797
–0,869672262	2,734968826	3,040483615	0,907386744

Кінець табл. А.1.

–1,40581674	–4,67601285	1,230288724	1,494576123
2,509611447	–2,353578094	–4,270445296	1,813619224
2,539387795	–1,81317593	6,091103718	–4,537169058
2,326773317	–5,660200657	–3,137536737	1,620003102
2,157319876	–0,05659154	–5,122015175	–0,027748542
–1,025355211	–3,517292651	–4,650772229	0,87513752
1,215135826	–4,813531231	2,110304595	–3,097235282
–2,475890911	–3,123157214	–1,293934904	4,591414656
–1,301235446	–3,185277296	–3,49329663	–1,272151647
6,645486121	3,776659579	–1,858256557	–0,15517356
–2,529981684	5,07782813	3,278490243	–2,885844642
2,066511755	–5,348832896	–0,634237293	3,346067016
–4,36826283	–4,007328077	1,178661937	–3,762304787
5,164953637	–6,507891865	1,732161549	2,921579348
–3,971991851	–3,123411252	1,912365376	7,077459655
1,705164363	9,873493119	0,682659397	–0,169170502
0,716604891	8,859918568	5,177179611	1,017370852
–0,265496601	1,829645685	0,934096403	–1,795415016
–0,54564116	2,702848088	4,406761292	–0,374981257

Реалізації повторного експерименту наведені в табл. А.2:

Таблиця А.2 – Реалізації повторного експерименту

14,19664775	12,42520543	21,33274296	17,54729755
22,19729126	32,02878936	25,63096228	23,57013645
17,64330226	20,54261156	21,25780685	22,00774631
34,36392541	26,34408529	35,10564597	33,17867243
21,44345814	17,5961694	14,55711398	17,35239582

Продовження табл. А.2.

31,06058197	44,76110424	45,52360253	36,58275354
41,78848349	47,05722476	43,60542006	37,95225908
23,91428985	28,066315	28,57422411	30,41441241
22,15476642	34,13588737	20,71161563	30,17696254
20,18371762	12,76771371	10,50642067	16,90736525
9,911096385	15,81830331	20,20638636	16,82067796
29,89440496	23,53525787	25,91953651	26,61856622
22,46815389	20,06786249	23,32594502	17,11692843
30,53120131	35,20631466	27,92874705	32,1036653
20,48773075	18,50643648	16,52340712	15,0670099
42,04893253	43,08928145	45,2088054	40,50885235
41,70003446	41,79842611	41,88237259	41,0159922
32,80947921	28,57538863	29,31518327	21,81029207
28,90162042	28,77972554	32,80944746	25,04035355
23,4070806	13,74428859	20,69112422	19,9354628
16,48532774	20,08996883	20,39548362	18,26238674
25,24258326	21,97238715	27,87868872	28,14297612
22,72176145	17,85857191	15,9417047	22,02576922
34,15478779	29,80222407	37,70650372	27,07823094
20,29177332	12,30479934	14,82746326	19,5850031
41,70066988	39,48675846	34,42133483	39,51560146
43,27279479	40,78085735	39,64737777	45,17328752
29,30048583	23,27181877	30,1956546	24,98811472
23,51785909	22,87059279	24,6998151	30,58516466
16,23436455	14,3503227	14,04230337	16,26344835
24,00048612	21,13165958	15,49674344	17,19982644
24,11841832	31,72622813	29,92689024	23,76255536

Кінець табл. А.2.

22,27866175	14,8633171	19,57791271	23,55821702
27,24713717	27,60807192	32,79406194	27,85309521
23,12995364	11,45710813	19,69716155	20,88657935
35,57135815	36,41993875	41,45571538	46,62080965
46,00331436	54,17164312	44,9808094	44,1289795
28,80195489	36,94526857	33,26252961	29,10272085
25,7282534	27,82339568	26,9278464	24,19833498
16,98995884	20,23844809	21,94236129	17,16061874

ДОДАТОК Б ПОРІВНЯЛЬНИЙ ПРИКЛАД ДЛЯ МОДУЛІВ THREADING ТА MULTIPROCESSING

```
from threading import Thread
from time import time
import numpy as np
from multiprocessing import Pool
from multiprocessing import freeze_support
from concurrent.futures.thread import ThreadPoolExecutor
from concurrent.futures.process import ProcessPoolExecutor

def gen_sqr_mx(size: int) -> np.ndarray:    # створення матриці size x size та
        заповнення випадковими значеннями
    return np.random.rand(size, size)

def sqr_mx(mx: np.ndarray) -> type(None):   # множення матриць лінійним способом
    [[sum(a * b for a, b in zip(row_a, row_b)) for row_b in zip(*mx)] for row_a in mx]

def mul_n_times_threading(n: int) -> type(None):
    start_time = time()
    threads = []
    [threads.append(Thread(target=sqr_mx, args=(gen_sqr_mx(99),))) for _ in range(n)]
    [t.start() for t in threads]    # запускаємо усі потоки
    [t.join() for t in threads]     # чекаємо на завершення усіх потоків
    print("mul_n_times_threading time: %.4f seconds..." % (time() - start_time))

def mul_n_times_thread_pool_executor(n: int) -> type(None):
    start_time = time()
    with ThreadPoolExecutor(max_workers=n) as tpe:  # запускаємо усі потоки та
        чекаємо на завершення усіх потоків
        [tpe.submit(sqr_mx, gen_sqr_mx(99)) for _ in range(n)]
    print("mul_n_times_thread_pool_executor time: %.4f seconds..." % (time() -
        start_time))

def mul_n_times_multiprocessing(n: int) -> type(None):
    start_time = time()
    with Pool(processes=n) as pool:     # створення пула з n процесами та чекаємо на
        завершення усіх
        pool.map(sqr_mx, [gen_sqr_mx(99) for _ in range(n)])
    print("mul_n_times_multiprocessing time: %.4f seconds..." % (time() - start_time))

def mul_n_times_process_pool_executor(n: int) -> type(None):
    start_time = time()
```

```
    with ProcessPoolExecutor(max_workers=n) as ppe:     # створення пула з n
            процесами та чекаємо на завершення усіх
        [ppe.submit(sqr_mx, gen_sqr_mx(99)) for _ in range(n)]
    print("mul_n_times_process_pool_executor time: %.4f seconds..." % (time() -
            start_time))

def main() -> int:
    mul_n_times_threading(n=4)
    mul_n_times_thread_pool_executor(n=4)
    freeze_support()
    mul_n_times_multiprocessing(n=4)
    mul_n_times_process_pool_executor(n=4)
    return 0

if __name__ == '__main__':
    main()
```

ДОДАТОК В ЧАСТИНА ПРОГРАМНОГО КОДУ

```
from math import floor, sqrt

from sympy import Poly, Rational, Symbol, solve, symbols

from experiments.abstract_experiment import AbstractExperiment
from experiments.forsythe_polynomial_experiment import ForsythePolynomialExperiment



def get_prime_numbers_generator():
    current = 0

    def generate() -> Rational:
        nonlocal current
        temp = current + 1 if current == 2 else current + 2
        while not check_if_prime(temp):
            temp += 2

        current = temp
        return Rational(temp)

    return generate



class MultivariatePolynomialRegressionExperiment(AbstractExperiment):
    def __init__(self, n: int, r: int, m: int, powers: list[list[int]],
                 x: list[list[Rational]]) -> None:
        self.n = n
        self.r = r
```

```
    self.m = m
    self.x = x

    self.powers = powers
    self.prime_generator = get_prime_numbers_generator()
    self.q_matrix = ForsythePolynomialExperiment(n, r, [line[0] for line in
        self.x]).get_q_matrix()

    self.found_thetas: dict[Symbol, Rational] = {}
    self.x_symbols: tuple[Symbol] = symbols('x(1:' + str(m + 1) + ')')
    self.thetas: tuple[Symbol] = symbols('θ(0:' + str(len(powers)) + ')')
    self.temp_x: Symbol = symbols('temp_x')
    self.all_symbols = [*self.x_symbols, *self.thetas, self.temp_x]

    self.polynomial: Poly = Poly(0, *self.all_symbols)

    for i in range(len(powers)):
        curr_part = self.thetas[i]
        for j in range(len(self.powers[i])):
            curr_part *= self.x_symbols[j] ** self.powers[i][j]

        self.polynomial += curr_part

    self.next_experiment = None
    self.temp = 0

def __get_current_polynomial_part(self) -> tuple[list[Rational], int, list[Symbol],
    Symbol] | None:
    self.temp += 1
```

```
    found_num = len(self.found_thetas)

    for i in range(len(self.powers)):
        power = self.powers[i]
        max_power = max(power)
        if max_power >= 2 and self.found_thetas.get(self.thetas[i]) is None:
            # and (i != 9 or found_num > 2):
            return power, max_power, [self.x_symbols[power.index(max_power)]],
        self.thetas[i]

    for i in range(len(self.powers)):
        power = self.powers[i]
        if sum(power) > 1 and self.found_thetas.get(self.thetas[i]) is None:
            non_zero_x = []
            for j in range(len(power)):
                if power[j] != 0:
                    non_zero_x.append(self.x_symbols[j])

            return power, sum(power), non_zero_x, self.thetas[i]

    return None

def __generate_experiment(self, current_x: list[Symbol]):
    eval_config = {}
    for xi in self.x_symbols:
        if xi not in current_x:
            eval_config[xi] = self.prime_generator()

    current_polynomial: Poly = self.polynomial.eval(eval_config)
```

```
    if len(current_x) != 1:
        for xi in current_x:
            current_polynomial = Poly(current_polynomial.subs(xi, self.temp_x))

    expanded_polynomial = self.__expand_polynomial(current_polynomial)
    return expanded_polynomial, eval_config

def get_next_experiment_data(self):
    polynomial_part = self.__get_current_polynomial_part()
    if polynomial_part is None:
        return None
    powers, current_power, current_x, current_theta = polynomial_part
    expanded_polynomial, eval_config = self.__generate_experiment(current_x)

    curr_polynomial = expanded_polynomial[current_power]
    monoms = curr_polynomial.monoms()
    coeffs = curr_polynomial.coeffs()
    if 1 not in monoms[-1]:
        coeffs.pop()

    experiments_number = len(coeffs)
    polynomials = [expanded_polynomial]
    configs = [eval_config]
    if experiments_number > 1:
        exp_pol, config = self.__generate_experiment(current_x)
        polynomials.append(exp_pol)
        configs.append(config)
```

```
    current_x_matrices = [
        [
            [self.x[i][j]
             if eval_config.get(self.x_symbols[j]) is None or eval_config[self.x_symbols[j]]
            == self.temp_x
             else eval_config[self.x_symbols[j]]
             for j in range(self.m)]
            for i in range(self.n)
        ]
        for eval_config in configs
    ]

    return current_x_matrices, experiments_number, polynomials, current_power,
        current_x, current_theta

def __expand_polynomial(self, polynomial: Poly):
    coeffs = polynomial.coeffs()
    monoms = polynomial.monoms()

    result: dict[int, Poly] = {i: Poly(0, self.thetas) for i in range(self.r)}
    for i in range(len(monoms)):
        x_power = monoms[i][0]
        if x_power >= self.r:
            continue

        for j in range(1, len(monoms[i])):
            if monoms[i][j] != 0:
                result[x_power] += polynomial.gens[j] * coeffs[i]
                break
```

```
    for x_power in range(self.r):
        curr_coeffs = result[x_power].coeffs()
        if len(curr_coeffs) == 1 and curr_coeffs[0] == 0:
            result.pop(x_power)
        else:
            result[x_power] = result[x_power].eval(self.found_thetas)

    return result

def process_step_experiment(self, y: list[list[Rational]]):
    if self.next_experiment is None:
        return

    x_matrices, experiments_number, expanded_polynomials, power, x_symbols, theta =
        self.next_experiment

    bs = [
        ForsythePolynomialExperiment(self.n, self.r, [line[0] for line in self.x], y=[[line[i]]
        for line in y])
        .get_thetas(self.q_matrix)
        for i in range(len(x_matrices))
    ]

    if len(bs) == 1:
        b = bs[0]
        polynomial = expanded_polynomials[0]
        for power in polynomial:
            coeffs = polynomial[power].coeffs()
```

```
            monoms = polynomial[power].monoms()
            free_part = Rational(0)
            if 1 not in monoms[-1]:
                free_part = coeffs.pop()
            if len(coeffs) == 1:
                monoms = polynomial[power].monoms()
                theta_index = monoms[0].index(1)
                theta_value = (b[power] - free_part) / coeffs[0]
                self.found_thetas[polynomial[power].gens[theta_index]] = theta_value
        else:
            b_symbols = symbols('b(0:' + str(len(bs)) + ')')
            polynomials = list(map(lambda obj: obj[power], expanded_polynomials))
            for i in range(len(b_symbols)):
                polynomials[i] = Poly(polynomials[i].as_expr() - bs[i][power])

            target_thetas = list(filter(lambda symb: symb in self.thetas, polynomials[0].gens))
            solution = solve(polynomials, target_thetas)
            for theta in solution:
                self.found_thetas[theta] = solution[theta]

    def __evaluate_found_part(self, x_values: list[Rational]) -> Rational:
        result: Rational = Rational(0)
        for i in range(len(self.powers)):
            power = self.powers[i]
            if self.thetas[i] in self.found_thetas:
                power_result: Rational = self.found_thetas[self.thetas[i]]
                for j in range(len(power)):
                    power_result *= x_values[j] ** power[j]
                result += power_result
```

```
        return result

    def convert_data_to_linear_experiment(self, y: list[list[Rational]]):
        new_y = [[yi for yi in line] for line in y]
        for i in range(len(self.x)):
            x_values = self.x[i]
            evaluated_result = self.__evaluate_found_part(x_values)
            new_y[i] = [yi - evaluated_result for yi in new_y[i]]

        return new_y
from functools import partial
import pyperclip as pc
from sympy import Rational, Symbol, symbols
from threading import Thread
from tkinter import Widget, Label, Entry, Button, E, W, EW, Frame
from typing import Literal, Callable

from experiments.constants import MIN_PRECISION
from experiments.main import InvalidPrecisionException, RA_RSS_divide_into_sets,
        RA_RSS_get_regression_structure, \
    RA_RSS_get_start_thetas, RA_RSS_get_thetas_res, \
    RA_convert_experiment_data_to_linear, RA_multivariate_get_next_step_data,
        RA_multivariate_polynomial, \
    RA_prime_generator, RA_to_float, RA_RSS_get_random_from_experiment
from experiments.xi_square import get_lingvo_variable, XiSquareStateLabels
from tabs.abstract_tab import AbstractTab, DEFAULT_FONT, InvalidInputException
from tabs.custom_elements.matrix_frame import MatrixFrame
from utils import InvalidMatrixFileException, float_to_str, read_matrix_from_file, \
```

```
    InvalidMatrixException, parse_str_matrix, load_vector_from_file

class MultivariatePolynomialTab(AbstractTab):
    # tab title
    title = 'Multivariate Polynomial Experiment'

    # saved result
    result = None

    def __init__(self, parent: Frame, reset_tab: Callable):
        super().__init__(parent, reset_tab)
        self.reset_tab = reset_tab
        self.found_non_linear_part = False
        self.found_linear_part = False
        self.found_thetas: dict[Symbol, Rational] = {}
        self.found_var_thetas: dict[Symbol, Rational] = {}
        self.next_experiment = None
        self.started_experiment = False
        self.step_number = 1

        self.prime_generator = RA_prime_generator()

    def __handle_button_click(self):
        # Creates a thread to calculate result
        # This should be as we do not have to freeze view while processing math

        self.elements["calculate_step_button"]["text"] = "Processing..."
        thread = Thread(target=self.__calculate)
        thread.start()
```

```
def __copy_x_matrix(self, index: int, precision: int):
    text = '\n'.join([' '.join(list(map(lambda x: float_to_str(RA_to_float(x, precision)),
        line))) for line in
                  self.next_experiment[0][index]])
    pc.copy(text)

def __handle_next_step(self):
    n, r, m, precision, x, powers = self.get_input_values()
    error_label: Label = self.elements['error_label']
    error_label.config(text="")

    for key in list(self.elements):
        if 'experiment_' in key:
            self.elements[key].grid_remove()
            self.elements.pop(key)

    if self.found_non_linear_part:
        return

    top_bias = 5
    section_height = 3
    self.next_experiment = RA_multivariate_get_next_step_data(n, r, m, powers, x,
                                                              lambda: self.prime_generator(),
                                                              self.found_thetas)
    if self.next_experiment is not None:
        Label(self, text="Step №" + str(self.step_number)).grid(column=0, columnspan=4,
        row=top_bias)
        self.step_number += 1
```

```
x_matrices = self.next_experiment[0]
for i in range(len(x_matrices)):
    x_label = Label(self, text="Experiment " + str(i + 1))
    x_label.grid(column=0, columnspan=2, row=top_bias + section_height * i + 1)

    x_matrix = MatrixFrame(self, matrix_width=len(x_matrices[i][0]),
matrix_height=len(x_matrices[i]),
                           disabled=True)
    x_matrix.set_matrix_inputs(
        [[float_to_str(RA_to_float(xi, precision)) for xi in line] for line in
x_matrices[i]])
    x_matrix.grid(column=0, columnspan=2, row=top_bias + section_height * i + 2,
sticky=EW)

    copy_x_matrix_button = Button(self, text="Copy X matrix",
                                  command=partial(self.__copy_x_matrix, i, precision))
    copy_x_matrix_button.grid(column=0, columnspan=2, row=top_bias +
section_height * i + 3)

    y_label = Label(self, text="Enter Y matrix")
    y_label.grid(column=2, row=top_bias + section_height * i + 1, sticky=E)

    y_load_button = Button(self, text="Load",
                           command=partial(self.set_matrix, "experiment_" + str(i) +
"_y_matrix",
                                           load_vector_from_file))
    y_load_button.grid(column=3, row=top_bias + section_height * i + 1, sticky=W)

    y_matrix = MatrixFrame(self)
```

```
        y_matrix.grid(column=2, columnspan=2, row=top_bias + section_height * i + 2,
    sticky=EW)

        self.elements['experiment_' + str(i) + "_x_matrix"] = x_matrix
        self.elements['experiment_' + str(i) + "_x_label"] = x_label
        self.elements['experiment_' + str(i) + "_copy_x_matrix_button"] =
    copy_x_matrix_button
        self.elements['experiment_' + str(i) + "_y_label"] = y_label
        self.elements['experiment_' + str(i) + "_y_load_button"] = y_load_button
        self.elements['experiment_' + str(i) + "_y_matrix"] = y_matrix

    self.elements['calculate_step_button'].grid(column=0, columnspan=4,
    row=top_bias + section_height * len(
        x_matrices) + section_height + 1)

else:
    self.found_non_linear_part = True

    x_label = Label(self, text="Enter X matrix")
    x_label.grid(column=0, row=top_bias, sticky=E)

    x_load_button = Button(self, text="Load", command=lambda:
    self.set_matrix("experiment_linear_x_matrix"))
    x_load_button.grid(column=1, row=top_bias, sticky=W)

    x_matrix = MatrixFrame(self)
    x_matrix.grid(column=0, columnspan=2, row=top_bias + 1, sticky=EW)

    y_label = Label(self, text="Enter Y matrix")
```

```
        y_label.grid(column=2, row=top_bias, sticky=E)

        y_load_button = Button(self, text="Load", command=lambda:
        self.set_matrix("experiment_linear_y_matrix"))
        y_load_button.grid(column=3, row=top_bias, sticky=W)

        y_matrix = MatrixFrame(self)
        y_matrix.grid(column=2, columnspan=2, row=top_bias + 1, sticky=EW)

        self.elements['experiment_linear_x_matrix'] = x_matrix
        self.elements['experiment_linear_x_label'] = x_label
        self.elements['experiment_linear_x_load_button'] = x_load_button
        self.elements['experiment_linear_y_label'] = y_label
        self.elements['experiment_linear_y_load_button'] = y_load_button
        self.elements['experiment_linear_y_matrix'] = y_matrix

        self.elements['calculate_step_button'].grid(column=0, columnspan=4,
        row=top_bias + 2)

def __get_experiments_data(self):
    ys: list[list[Rational]] = []

    max_index = -1
    for key in self.elements:
        if 'experiment_' in key:
            index = key.split('_')[1]
            if int(index) > max_index:
                max_index = int(index)
```

```
    for i in range(max_index + 1):
        y_matrix = parse_str_matrix(self.elements['experiment_' + str(i) +
        "_y_matrix"].get_matrix_data())
        for i in range(len(y_matrix)):
            while i >= len(ys):
                ys.append([])

            ys[i].append(y_matrix[i][0])

    return ys

def __convert_results_to_str_matrix(self, precision: int) -> list[list[str]]:
    matrix_values: list[list[str]] = []
    thetas_list = list(self.found_thetas.keys())
    var_thetas_list = list(self.found_var_thetas.keys())

    for i in range(len(thetas_list)):
        theta_key = thetas_list[i]
        var_theta_key = None if i >= len(var_thetas_list) else var_thetas_list[i]
        line = [
            str(theta_key) + ": " + float_to_str(RA_to_float(self.found_thetas[theta_key],
        precision)),
            str(var_theta_key) + ": " + float_to_str(
                RA_to_float(self.found_var_thetas[var_theta_key], precision)) if
        var_theta_key is not None else ''
        ]
        matrix_values.append(line)

    return matrix_values
```

```
def __copy_results(self, type: Literal['thetas', 'var_thetas'], precision: int):
    values = list(self.found_thetas.values()) if type == 'thetas' else
        list(self.found_var_thetas.values())
    pc.copy('\n'.join(list(map(lambda value: float_to_str(value), RA_to_float(values,
        precision)))))

def __calculate(self):
    # Calculates the result of experiment and sets it to result matrices
    error_label: Label = self.elements['error_label']
    result_matrix: MatrixFrame = self.elements['result_matrix']
    error_label.config(text="")

    try:
        if not self.started_experiment:
            self.__handle_next_step()
            self.elements['x_matrix'].set_disabled(True)
            self.elements['powers_matrix'].set_disabled(True)
            self.started_experiment = True
            self.elements["calculate_step_button"]["text"] = "Calculate step"
            return

        n, r, m, precision, x, powers = self.get_input_values()
        if not self.found_non_linear_part:
            # get input values
            ys = self.__get_experiments_data()

            self.found_thetas = RA_multivariate_polynomial(n, r, m, powers, x,
        self.found_thetas,
```

```
                                    self.next_experiment, ys)

else:
    y = parse_str_matrix(self.elements['experiment_linear_y_matrix'].get_matrix_data())
    x_matrix = parse_str_matrix(self.elements['experiment_linear_x_matrix'].get_matrix_data())

    y_converted = RA_convert_experiment_data_to_linear(n, r, m, powers, x_matrix,
self.found_thetas, y)

    l = len(y_converted[0])
    thetas = RA_RSS_get_start_thetas(n, m, l, x_matrix, y_converted)
    m1, m2 = RA_RSS_divide_into_sets(thetas, Rational(0.03))

    a_matrix, theta_symbols = RA_RSS_get_regression_structure(n, m, l, x_matrix,
y_converted, (m1, m2),
                                                    Rational(0.02))

    thetas_res, var_thetas_res = RA_RSS_get_thetas_res(l, a_matrix, y_converted,
theta_symbols)
    results = RA_RSS_get_random_from_experiment(l, a_matrix, y_converted,
thetas_res)

    theta_symbols = symbols("θ(0:" + str(len(powers)) + ")")
    var_theta_symbols = symbols("Varθ(0:" + str(len(powers)) + ")")
    new_found_thetas = {}
    new_found_var_thetas = {}
```

```
    for theta_symbol in theta_symbols:
        if theta_symbol in self.found_thetas:
            new_found_thetas[theta_symbol] = self.found_thetas[theta_symbol]
        elif theta_symbol in thetas_res:
            new_found_thetas[theta_symbol] = thetas_res[theta_symbol]
        else:
            new_found_thetas[theta_symbol] = Rational(0)

    for var_theta_symbol in var_theta_symbols:
        if var_theta_symbol in var_thetas_res:
            new_found_var_thetas[var_theta_symbol]                                    =
var_thetas_res[var_theta_symbol]

    self.found_thetas = new_found_thetas
    self.found_var_thetas = new_found_var_thetas
    self.found_linear_part = True

    state, xi = get_lingvo_variable(results)
    self.elements['lingvo_label'].config(
        text='Xi-square: ' + str(RA_to_float(xi, MIN_PRECISION)) + ', results are ' +
str(
            XiSquareStateLabels[state]), )
    self.elements['lingvo_label'].grid(column=0, columnspan=4, row=22)

self.__handle_next_step()

if not self.found_linear_part:
    # fill result matrix
```

```
            result_matrix.change_matrix_bounds(width=1,
        height=len(list(self.found_thetas)))
            result_matrix.set_matrix_inputs(
                [[str(key) + ": " + float_to_str(RA_to_float(self.found_thetas[key], precision))]
        for key in
                 list(self.found_thetas)])

        else:
            matrix_values = self.__convert_results_to_str_matrix(precision)
            result_matrix.change_matrix_bounds(width=2,
        height=len(list(self.found_thetas)))
            result_matrix.set_matrix_inputs(matrix_values)
            self.elements['calculate_step_button'].grid_remove()
            copy_thetas_button: Button = self.elements['copy_thetas_button']
            copy_var_thetas_button: Button = self.elements['copy_var_thetas_button']

            copy_thetas_button.grid(column=0, columnspan=2, row=21)
            copy_thetas_button.configure(command=lambda:      self.__copy_results('thetas',
        precision))

            copy_var_thetas_button.grid(column=2, columnspan=2, row=21)
            copy_var_thetas_button.configure(command=lambda:
        self.__copy_results('var_thetas', precision))

    except (InvalidMatrixFileException,
            InvalidInputException,
            InvalidMatrixException,
            InvalidPrecisionException,
            ) as err:
```

```
            # show error
            error_label.config(text=err)

        # return button text
        self.elements["calculate_step_button"]["text"] = "Calculate step"
        self.update()
        self.update_canvas()

    def copy_result(self):
        # Copies the result into clipboard
        if self.result is not None:
            matrix_str = ""
            for line in self.result:
                matrix_str += float_to_str(line) + '\n'

            pc.copy(matrix_str)

    def get_tab_elements(self) -> dict[str, Widget]:
        # Here all the widgets should be set
        # Returns a dict with all widgets that will be used later
        # The most of them will have a look of two widgets: Label and Entry

        # Enter precision
        precision_label = Label(self, text="Enter precision")
        precision_label.grid(column=0, row=1, sticky=E, padx=5)

        precision_entry = Entry(self, width=20, font=DEFAULT_FONT)
        precision_entry.grid(column=1, row=1, sticky=W)
        precision_entry.insert(0, "50")
```

```
reset_button = Button(self, text="Reset", command=self.reset_tab)
reset_button.grid(column=2, columnspan=2, row=1)

# Enter X-matrix
x_label = Label(self, text="Enter X values")
x_label.grid(column=0, row=3, sticky=E, pady=(15, 5))

x_load_button = Button(self, text="Load",
                       command=lambda: self.set_matrix("x_matrix"))
x_load_button.grid(column=1, row=3, sticky=W, pady=(15, 5))

x_matrix = MatrixFrame(self, matrix_width=4, matrix_height=10, left_aligned=True)
x_matrix.grid(column=0, columnspan=2, row=4, sticky=EW)

powers_label = Label(self, text="Enter powers")
powers_label.grid(column=2, row=3, pady=(15, 5), sticky=E)

powers_load_button = Button(self, text="Load", command=lambda:
    self.set_matrix("powers_matrix"))
powers_load_button.grid(column=3, row=3, sticky=W, pady=(15, 5))

powers_matrix = MatrixFrame(self, matrix_width=1, matrix_height=10)
powers_matrix.grid(column=2, columnspan=2, row=4, sticky=EW)

calculate_step_button = Button(self, text='Calculate step', command=self.__calculate)
calculate_step_button.grid(column=0, columnspan=4, row=18)

# Error label
```

```
error_label = Label(self, text="", fg="red")
error_label.grid(column=0, columnspan=4, row=19)

result_matrix = MatrixFrame(self, disabled=True)
result_matrix.grid(column=0, columnspan=4, row=20, sticky=EW)

# Copy to clipboard button
copy_thetas_button = Button(self, text='Copy θ', command=self.copy_result)
copy_var_thetas_button = Button(self, text='Copy Varθ', command=self.copy_result)

# Error label
lingvo_label = Label(self, text="")

# Set all columns as equal by width (like flex-grow)
for i in range(4):
    self.columnconfigure(i, weight=1)

return {
    'precision_entry': precision_entry,
    'x_matrix': x_matrix,
    'powers_matrix': powers_matrix,
    'calculate_step_button': calculate_step_button,
    'error_label': error_label,
    'result_matrix': result_matrix,
    'copy_thetas_button': copy_thetas_button,
    'copy_var_thetas_button': copy_var_thetas_button,
    'lingvo_label': lingvo_label,
    'histogram': None
}
```

```
def get_input_values(self) -> tuple:
    # Returns all necessary data from inputs
    try:
        precision = int(self.elements['precision_entry'].get())
        x:                                          list[list[Rational]]                                =
        parse_str_matrix(self.elements['x_matrix'].get_matrix_data())
        powers: list[list[Rational]] = [[int(item) for item in line] for line in


        parse_str_matrix(self.elements['powers_matrix'].get_matrix_data())]
        r = max(list(map(lambda power: max(power), powers))) + 1
        return len(x), int(r), len(powers[0]), precision, x, powers
    except ValueError:
        raise InvalidInputException('Invalid input!')

def load_initial_values(self):
    # Process initial loading the data on first form load

    self.set_matrix('x_matrix',
                    lambda:
        read_matrix_from_file('./data/multivariate_polynomial/experiment_1/x-start.txt'))
    self.set_matrix('powers_matrix',
                    lambda:
        read_matrix_from_file('./data/multivariate_polynomial/experiment_1/powers.txt'))
```

ДОДАТОК Г СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці, в яких опубліковано основні наукові результати дисертації:

1. Павлов О. А., **Головченко М. М.** Побудова одновимірної і багатовимірної поліноміальної регресії за надлишковим описом з використанням активного експерименту // Вісник Національного технічного університету «ХПІ». Серія: Системний аналіз, управління та інформаційні технології. № 1(3), 2020. С. 3–8. doi: 10.20998/2079-0023.2020.01.02. *Здобувачу належить обґрунтування можливості використання алгоритмів кластерного аналізу у запропонованому методі побудови багатовимірної поліноміальної регресії.*
2. Pavlov A., **Holovchenko M.**, Mukha I., Lishchuk K. 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. Cham: Springer, 2022. Vol. 134. P. 288–303. doi: 10.1007/978-3-031-04812-8_25 (Проіндексовано в **Scopus**). *Здобувачу належить обґрунтування можливості оцінок коефіцієнтів при нелінійних членах багатовимірної поліноміальної регресії на основі послідовної побудови одновимірних поліноміальних регресій зі спільною областю значень вхідної скалярної змінної та приклад, що ілюструє ефективність запропонованого методу.*
3. 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. К.: НТУУ «КПІ», 2022. Т. 1. № 40. С. 110–117. doi: 10.20535/1560-8956.40.2022.261665. *Здобувачу належить обґрунтування структури алгоритму кластерного аналізу та логіку взаємодії його компонентів.*
4. Pavlov A.A. **Holovchenko M.N.**, 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:

10.20998/2079-0023.2022.01.01. *Здобувачу належить на основі результатів, викладених у [Pavlov A.A. Estimating with a given accuracy of the coefficients at nonlinear terms of univariate polynomial regression using a small number of tests in an arbitrary limited active experiment // Bulletin of National Technical University "KhPI". Series: System analysis, control and information technologies, – № 2 (6), 2021. – С. 3-7. doi: 10.20998/2079-0023.2021.02.01], обґрунтування можливості оцінок коефіцієнтів при нелінійних членах одновимірної поліноміальної регресії, на основі довільного повторного активного експерименту, з використанням лише одного набору нормованих ортогональних поліномів Форсайта, що оцінює коефіцієнти віртуальної одновимірної поліноміальної регресії, модифікація загального методу для оцінок коефіцієнтів при нелінійних членах багатовимірної поліноміальної регресії для задачі інтерполяції детермінованого багатовимірного полінома заданого надлишковим описом.*

5. Pavlov A. A., **Holovchenko M. N.** Modified method of constructing a multivariate linear regression given by a redundant description // Bulletin of National Technical University "KhPI". Series: System analysis, control and information technologies. № 2(8), 2022. С. 3–8. doi: 10.20998/2079-0023.2022.02.01. *Здобувачу належить створення критерію вибору шуканої структури багатовимірної поліноміальної регресії на основі використання перевірочної послідовності даних.*
6. Pavlov, A., **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 / Advances in Computer Science for Engineering and Education VI. ICCSEEA 2023 // Lecture Notes on Data Engineering and Communications Technologies. Cham: Springer, 2023. Vol. 181. P. 207–222. doi: 10.1007/978-3-031-36118-0_19 (Проіндексовано в **Scopus**). *Здобувачу належить розробка та обґрунтування загальної структури програмної бібліотеки, що реалізує синтетичний метод побудови багатовимірної поліноміальної регресії.*
7. Pavlov A. A., **Holovchenko M. N.**, 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: 10.20998/2079-0023.2023.01.01. *Здобувачу належить створення п'яти часткових випадків надлишкових описів, що приводить до розв'язку систем лінійних рівнянь з однією змінною, участь у виведені верхніх границь оцінок коефіцієнтів при нелінійних членах багатовимірної поліноміальної регресії, а також у дослідженні теоретичних властивостей повторних активних експериментів у загальному методі найменших квадратів.*

Наукові праці, які засвідчують апробацію матеріалів дисертації:

8. Дрозд В.В., **Головченко М.М.** Методи та програмні засоби побудови нелінійних поліноміальних регресій з використанням нормованих ортогональних поліномів Форсайта // Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2021) : матеріали тез доповідей І Всеукраїнської наук.-практ. конф. молодих вчених та студентів (м. Київ, 22–26 листопада 2021р.). Секція кафедри інформатики та програмної інженерії. – К. : КПІ ім. Ігоря Сікорського, 2021. – С. 82–86. *Здобувачу належить побудова загальної архітектури програмної бібліотеки, що реалізує метод оцінки коефіцієнтів при нелінійних членах одновимірної поліноміальної регресії з використанням нормованих ортогональних поліномів Форсайта.*
9. Павлов О.А., **Головченко М.М.,** Дрозд В.В., Ревич М.М. Дослідження ефективності методу побудови багатовимірної лінійної регресії, заданої надлишковим описом // Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2022 Осінь) : матеріали тез доповідей ІІІ Всеукраїнської наук.-практ. конф. молодих вчених та студентів (м. Київ, 22–25 жовтня 2022 р.). – К. : КПІ ім. Ігоря Сікорського, 2023. – С. 10–13. URL: https://drive.google.com/file/d/1CP9EaBTT_rJAXsINbanSVGnP2jkg9FJ0/view (Дата звернення: 29.08.2023). *Здобувачу належить результати експериментального дослідження ефективності використання універсальних методів розпаралелювання обчислень в операціях з матрицями у загальній формулі методу найменших квадратів.*
10. Павлов О.А., **Головченко М.М.,** Дрозд В.В. Синтетичний метод побудови багатовимірної поліноміальної регресії // Комплексне забезпечення якості технологі-

чних 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. URL: http://ir.stu.cn.ua/123456789/28221 (Дата звернення: 29.08.2023). *Здобувачу належить обґрунтування складових підалгоритмів та логіки їх взаємодії в синтетичному методі побудови багатовимірної поліноміальної регресії.*