

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова
праця на правах рукопису

ОМЕЛЬЧЕНКО ОЛЕКСАНДР СТАНІСЛАВОВИЧ

681.5:663.5

ДИСЕРТАЦІЯ

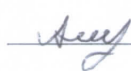
**АВТОМАТИЗОВАНА СИСТЕМА МОНІТОРИНГУ ПРОЦЕСІВ
ВИРОБНИЦТВА СПИРТУ ТА БІОЕТАНОЛУ НА ОСНОВІ
ВІРТУАЛЬНИХ ІНСТРУМЕНТІВ КОНТРОЛЮ**

Спеціальність: 151 «Автоматизація та комп'ютерно-інтегровані
технології»

Галузь знань: 15 «Автоматизація та приладобудування»

Подається на здобуття наукового ступеня доктора філософії

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

 О. С. Омельченко

Науковий керівник:

Луцька Наталія Миколаївна
доктор технічних наук, професор



Київ-2025

АНОТАЦІЯ

Омельченко О.С. – Автоматизована система моніторингу процесів виробництва спирту та біоетанолу на основі віртуальних інструментів контролю – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 151 «Автоматизація та комп'ютерно-інтегровані технології» – Національний університет харчових технологій Міністерства освіти і науки України, Київ, 2025.

Робота присвячена підвищенню ефективності автоматизованого моніторингу та керування технологічними процесами спиртової промисловості, зокрема виробництва спирту та біоетанолу шляхом використання віртуальних інструментів контролю.

В першому розділі розглянуті особливості технологічних процесів виробництва спирту та біоетанолу, ключові проблеми автоматизації та перспективи впровадження інновацій Індустрії 4.0. Проаналізовано сучасні підходи та літературні джерела щодо віртуальних інструментів контролю, зокрема цифрових двійників та віртуальних аналізаторів.

Зроблено висновки про наявність оптимізаційних проблем в спиртовому виробництві пов'язаних зі складним та комплексним характером біологічних та хімічних процесів, які важко чи дорого вирішити класичними засобами автоматизації, проте які можуть бути вирішені впровадженням віртуальних засобів контролю.

Здійснено постановку задач дослідження, що включає розробку математичних моделей для опису основних технологічних процесів, синтез віртуальних сенсорів технологічних змінних та автоматизованої інтелектуальної системи керування процесами спирту та біоетанолу.

В другому розділі проведено системний аналіз технологічних відділень спиртового виробництва, розроблено прикладну онтологію математичних моделей у межах якої досліджено використання віртуальних сенсорів для моніторингу технологічних змінних з урахуванням відповідності існуючих

моделей до поставлених задач дослідження. Згідно результатів було виділено список задач, що можуть бути вирішені з використанням віртуальних інструментів.

Проведено статистичний аналіз даних роботи дистиляційного відділення для виявлення закономірностей, що можуть бути використані в класифікаційних задачах. Розроблено та протестовано на виробничих даних механізм на основі методу Gaussian Mixture Model для оцінки стабільності системи в реальному часі, що дає змогу виявляти ознаки нестабільності на ранніх етапах і мінімізувати пов'язані технологічні витрати.

В третьому розділі здійснено передбачення невимірюваних показників з використанням NARX-моделей, досліджено вплив зміни гіперпараметрів NARX-моделей на їх продуктивність та розглянуто бібліотеки для автоматизованого підбору їх налаштувань. Відповідно до отриманих результатів, NARX-модель змогла пояснити 99% варіацій цільової змінної при прогнозуванні концентрації етанолу.

Проведено порівняння ефективності інтелектуального моніторингу динамічних процесів на основі віртуальних сенсорів у дистиляційному відділенні спиртового виробництва за допомогою методів XGBoost, Decision Tree, NARX, Gradient Boosting та RNN. Моделювання підтвердило можливість компенсації функціоналу сенсорів у разі їхньої відмови або виходу з ладу, що забезпечує безперервність автоматизованого керування процесом. Зокрема:

- моделювання витрати бражки на основі показників інших сенсорів дозволило компенсувати відмову датчика витрати з коефіцієнтом детермінації $R^2=0,96$;
- прогнозування тиску в колоні концентрації за відсутності даних від датчика тиску здатне пояснити 98% цільової змінної, що підтверджує можливість підтримки стабільної роботи системи при відмові відповідного сенсора;
- компенсація відмови датчика температури дозволила пояснити 97,2% варіації цільової змінної, що підтверджує ефективність моделі для інтелектуального моніторингу в режимі реального часу.

Розроблено алгоритми для оптимізації режимів роботи бродильного відділення на основі моделі методами дерев регресії, для пошуку оптимальних налаштувань режимів процесу відповідно до поставленої задачі оптимізації. Запропоновані алгоритми дозволяють збільшити вихід етанолу в середньому на 8.4%, або зменшити час бродіння на 12,2%.

В четвертому розділі розроблено автоматизовану інтелектуальну систему керування з використанням методів Reinforcement learning, яка забезпечує стабільне регулювання в нестационарних процесах спиртового виробництва, де ПІ-регулятор не здатний забезпечити достатньої точності підтримування заданих значень та якісні перехідні процеси, внаслідок змін об'єкта та середовища спричинених типовими виробничими змінами, такими як забруднення, старіння обладнання або коливання температури середовища чи складу сировини. Найкращі результати при змінах параметрів дистиляційної колони продемонстрував Soft Actor-Critic агент з Entropy weight 1 забезпечивши високу точність ($<0,1\%$), мінімальне перевищення (динамічна похибка 21–26% від цільового значення), і найменшу сумарну помилку за інтегрально-квадратичним критерієм.

Розроблено концептуальну структуру автоматизованої системи моніторингу та керування технологічними процесами виробництва спирту та біоетанолу з використанням віртуальних інструментів контролю. Також розроблено архітектуру автоматизованої системи моніторингу виробництвом спирту та біоетанолу з використанням віртуальних сенсорів та запропоновано її реалізацію з використанням програмно-технічних комплексів Unity Pro, Node-RED і хмарних інструментів InfluxDB, Grafana, Grafana Machine Learning. Запропонована архітектура дозволяє впроваджувати віртуальні інструменти в структуру виробничої системи та забезпечує обмін виробничою інформацією.

Наукова новизна роботи полягає в розробці нової концепції автоматизованої системи моніторингу та керування виробничими процесами спирту та біоетанолу на основі віртуальних інструментів контролю. В роботі знайшли подальший розвиток системи моніторингу та автоматизованого керування

технологічними процесами виробництва спирту та біоетанолу, що засновані на використанні віртуальних сенсорів і математичних моделях, які здійснюють аналіз стабільності процесів, прогнозування та ідентифікацію ключових параметрів на основі супутніх змінних, оптимізацію режимів та інтелектуальне управління. Запропонований підхід дозволяє ефективно реагувати на змінну динаміку технологічних процесів і варіативність характеристик вхідної сировини, мінімізувати вплив людського фактора, підвищити точність прийняття рішень, зменшити енерговитрати та покращити якість кінцевої продукції, що забезпечує стійкість і адаптивність виробництва. Вперше сформовано онтологічну структуру математичних моделей технологічних процесів із включенням віртуального сенсора як ключового елемента, яка слугує інструментом підтримки прийняття рішень під час вибору адекватних моделей, забезпечує цілісний підхід до їх класифікації та аналізу, скорочує час пошуку релевантних рішень і полегшує інтеграцію нових знань у наявні системи моделювання. Подальшого розвитку набули методи створення віртуальних сенсорів із використанням машинного навчання для оцінки й прогнозування ключових змінних у динамічних системах, що дозволяє підвищити надійність контролю, виявляти збої у роботі реальних сенсорів, компенсувати втрати даних і тим самим забезпечити вищий рівень інформаційної обізнаності, стабільності та енергоефективності процесів. Крім того, розроблено інтелектуальну систему керування дистиляційною колоною, яка вперше реалізує підхід підкріплювального навчання для стабілізації концентрацій цільових компонентів у продуктах розділення, забезпечуючи адаптивне управління технологічними змінними в реальному часі, швидке реагування на збурення та коливання параметрів, що сприяє підвищенню якості продукції та стабільності функціонування в умовах невизначеності.

Практичне значення отриманих результатів полягає у створенні ефективних інструментів підтримки прийняття рішень і підвищення надійності функціонування технологічних систем виробництва спирту та біоетанолу. Розроблена онтологія математичних моделей технологічних процесів забезпечує

зручну структурування знань, швидкий доступ до релевантних підходів моделювання та полегшує вибір відповідних моделей для конкретних виробничих завдань. Запропонована модель класифікації дозволяє оперативно оцінювати стабільність функціонування виробничих процесів в реальному часі, виявляючи ознаки нестабільності на ранніх етапах і знижуючи пов'язані технологічні втрати. Застосування моделей машинного навчання для передбачення невимірюваних змінних і впровадження інтелектуального моніторингу сприяє підвищенню інформаційної обізнаності про стан об'єкта, надає змогу контролювати справність сенсорів та компенсувати втрати даних у разі їх відмови. Оптимізаційні алгоритми, що базуються на математичних моделях процесів, забезпечують можливість гнучкого налаштування технологічних режимів відповідно до зміни вхідної сировини та досягнення заданих виробничих цілей. Інтелектуальна система керування дистиляційною колоною на основі RL-агентів дає змогу забезпечити високу точність регулювання і стабільність роботи обладнання в умовах варіативності параметрів процесу. Розроблена архітектура автоматизованої системи керування з інтегрованими віртуальними сенсорами підвищує ефективність інформаційної взаємодії та забезпечує гнучкість виробничих структур. Отримані результати знайшли практичне застосування як у вигляді методичних підходів до створення віртуальних сенсорів для харчової промисловості в освітньому процесі кафедри автоматизації та комп'ютерних технологій систем управління ім. проф. А.П. Ладанюка, так і в умовах реального виробництва на «Марилівському спиртовому заводі», що підтверджує їхню прикладну цінність і ефективність.

Ключові слова: автоматизація, інтелектуальні технології, система керування з RL-агентами, технологічні процеси виробництва спирту та біоетанолу, віртуальний сенсор, промисловий IoT, модель машинного навчання, моделювання та оптимізація, моніторинг, нейронні мережі, прогнозування, оцінка ефективності, системний та онтологічний аналіз, хмарні сервіси.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Статті у наукових фахових виданнях України

1. Omelchenko, O., Lutska, N., & Vlasenko, L. (2023). Онтологія математичних моделей технологічних об'єктів промислових підприємств. Ч. 1. формування основних концептів предметної області. Енергетика і автоматика, 1, 23-41.

[https://dx.doi.org/10.31548/energiya1\(65\).2023.023](https://dx.doi.org/10.31548/energiya1(65).2023.023)

2. Omelchenko, O., Lutska, N., & Vlasenko, L. (2023). Онтологія математичних моделей технологічних об'єктів промислових підприємств. Ч. 2. реалізація онтології та її перевірка. Енергетика і автоматика, 2, 39-54.

[https://dx.doi.org/10.31548/energiya2\(66\).2023.039](https://dx.doi.org/10.31548/energiya2(66).2023.039)

3. Омельченко, О., & Луцька, Н. (2024). Хмарна архітектура передачі даних для впровадження моделей машинного навчання в спиртову промисловість. Automation of Technological and Business Processes, 15(4), 38-48.

<https://doi.org/10.15673/atbp.v15i4.2589>

4. Луцька Н. М., & Омельченко О. С. (2024). Особливості застосування віртуальних аналізаторів у цифровій інфраструктурі промислового виробництва. Наукові праці НУХТ, 30(2), 19-30.

<https://doi.org/10.24263/2225-2924-2024-30-2-4>

https://drive.google.com/file/d/1fkOCnkd_LW5HPm9Xu-SwqvBSEGU6eeot/view

5. Омельченко О. С., & Луцька Н. М. (2024). Розробка математичних моделей віртуальних сенсорів для моніторингу динамічних процесів у дистиляційному відділенні спиртового виробництва. Харчова промисловість. 3, 67-81.

<https://doi.org/10.24263/2225-2916-2024-36-9>

Тези доповідей та матеріали конференції

6. Луцька, Н. М., & Омельченко, О. С. (2020). "Використання віртуальних аналізаторів в інтегрованих автоматизованих системах управління виробництвом." У 86 Міжнародна наукова конференція молодих учених, аспірантів і студентів "Наукові здобутки молоді – вирішенню проблем харчування людства у XXI столітті" (р. 373). Національний університет харчових технологій.

<https://dspace.nuft.edu.ua/handle/123456789/31828>

7. Луцька, Н. М., & Омельченко, О. С. (2020). "Віртуальні аналізатори в задачах управління технологічними процесами." У VII Міжнародна науково-технічна Internet-конференція "Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами" (рр. 106–107). Національний університет харчових технологій.

<https://dspace.nuft.edu.ua/handle/123456789/34045>

8. Луцька, Н. М., & Омельченко, О. С. (2021). "Використання моделей регресійних дерев при моделюванні технологічних режимів процесу бродіння." У Наукові праці Третьої міжнародної науково-практичної конференції "Сучасні тенденції розвитку інформаційних систем і телекомунікаційних технологій" (рр. 123–125). Національний університет харчових технологій.

https://drive.google.com/file/d/11bgvp6Ef8BIRBY_6E9c859aip7uoUzmO/view

9. Луцька, Н. М., & Омельченко, О. С. (2021). "Особливості використання регресійних дерев для віртуальних аналізаторів спиртових підприємств." У VIII Міжнародна науково-технічна Internet-конференція "Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами" (рр. 48–49). Національний університет харчових технологій.

<https://dspace.nuft.edu.ua/handle/123456789/36828>

10. Луцька, Н. М., & Омельченко, О. С. (2022). "Перспективи використання методів машинного навчання для побудови промислових віртуальних

аналізаторів." У Матеріали 88 Міжнародної наукової конференції молодих учених, аспірантів і студентів "Наукові здобутки молоді – вирішенню проблем харчування людства у XXI столітті" (Ч. 2, р. 283). Національний університет харчових технологій.

<https://dspace.nuft.edu.ua/handle/123456789/38032>

11. Омельченко, О. С., & Луцька, Н. М. (2022). "Онтологія математичних моделей технологічних об'єктів промислових підприємств." У IX Міжнародна науково-технічна Internet-конференція "Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами" (р. 52). Національний університет харчових технологій.

<https://dspace.nuft.edu.ua/handle/123456789/38984>

12. Омельченко, О. С., & Луцька, Н. М. (2023). "Особливості впровадження хмарного сховища InfluxDB для технологічних даних виробництва." Grail of Science, (30), 160–162.

<https://doi.org/10.36074/grail-of-science.04.08.2023.025>

13. Омельченко, О. С., & Луцька, Н. М. (2023). "Структурний аналіз основних задач бродильного відділення." У Abstracts of XXXIV International Scientific and Practical Conference (pp. 15–19). Florence, Italy.

<https://eu-conf.com/events/current-and-youth-ways-of-solving-the-problems-of-world-science/>

14. Омельченко, О. С., & Луцька, Н. М. (2023). "Автоматизована система керування спиртовим виробництвом з використанням моделей машинного навчання." У X Міжнародна науково-технічна Internet-конференція "Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами" (р. 131). Національний університет харчових технологій.

<https://dspace.nuft.edu.ua/handle/123456789/41684>

15. Омельченко, О. С. (2024). "Проблеми прогнозування якісних показників спиртового виробництва методами машинного навчання." У 90 Міжнародна

наукова конференція молодих учених, аспірантів і студентів "Наукові здобутки молоді – вирішенню проблем харчування людства у XXI столітті" (Ч. 2, р. 369). Національний університет харчових технологій.

<https://dspace.nuft.edu.ua/handle/123456789/45780>

16. Омельченко, О. С., & Луцька, Н. М. (2024). "Формування, реалізація та використання віртуальних аналізаторів у контексті складних технологічних об'єктів." У I Міжнародна науково-практична конференція "Штучний інтелект та інформаційні технології" (pp. 186–187). Національний університет харчових технологій.

https://drive.google.com/file/d/1NcshN3OfNRnNt0ESKm9Rte1wZMUzbU_6/view

17. Омельченко, О. С., & Луцька, Н. М. (2024). "Система моніторингу процесів виробництва спирту та біоетанолу на основі віртуальних інструментів контролю." У XI Міжнародна науково-технічна Internet-конференція "Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами" (pp. 169–170). Національний університет харчових технологій.

<https://dspace.nuft.edu.ua/handle/123456789/46040>

ABSTRACT

O. S. Omelchenko – Automated system for monitoring ethanol and bioethanol production processes based on virtual control tools – Qualification research with manuscript rights.

Dissertation for obtaining the scientific degree of candidate of technical sciences in specialty 151 "Automation and Computer-integrated technologies" – National University of Food Technologies of the Ministry of Education and Science of Ukraine, Kyiv, 2025.

The study is dedicated to enhancing the efficiency of automated monitoring and control of technological processes in the ethanol industry, particularly in the production of ethanol and bioethanol, through the use of virtual control tools.

In the first chapter, the features of technological processes in ethanol and bioethanol production, key automation challenges, and prospects for implementing Industry 4.0 innovations are examined. Modern approaches and literature sources regarding virtual control tools, including digital twins and virtual analyzers, are analyzed. Conclusions are drawn about the presence of optimization challenges in ethanol production related to the complex and intricate nature of biological and chemical processes, which are difficult or costly to address with classical automation methods but can be resolved by implementing virtual control tools. The research objectives are formulated, including the development of mathematical models to describe key processes, analysis of methods for predicting the dynamics of technological parameter changes, and evaluation of the effectiveness of proposed approaches for optimizing the control of ethanol and bioethanol production processes.

In the second chapter, a systemic analysis of technological units in ethanol production is conducted, and an applied ontology of mathematical models is developed, within which the use of virtual sensors for monitoring technological processes is studied, taking into account the suitability of existing models for the research objectives. Based on the results, a list of tasks that can be addressed using virtual tools is identified. A statistical analysis of the distillation unit's operational data was conducted to identify patterns suitable for classification tasks. A Gaussian Mixture Model-based mechanism was developed and validated using production data to monitor system stability in real time, allowing for the early detection of instability indicators and the reduction of related technological expenses.

In the third chapter, prediction of unmeasurable parameters is carried out using NARX models, the impact of NARX model hyperparameter variations on their performance is investigated, and libraries for automated tuning of their settings are explored. According to the obtained results, the NARX model was able to explain 99% of the variations in the target variable when predicting ethanol concentration. A comparison of the effectiveness of intelligent monitoring of dynamic processes in the distillation unit of ethanol production is conducted using XGBoost, Decision Tree, NARX, Gradient Boosting, and RNN methods. Modeling confirmed the possibility of

compensating for sensor functionality in case of their failure or malfunction, ensuring the continuity of automated process control. Specifically:

- modeling the mash flow rate based on the readings of other sensors allowed compensation for the failure of the flow sensor with a coefficient of determination $R^2 = 0.96$;
- forecasting the pressure in the concentration column in the absence of data from the pressure sensor was able to explain 98% of the target variable, confirming the possibility of maintaining stable system operation in case of the respective sensor failure;
- compensation for the failure of the temperature sensor enabled explaining 97.2% of the variation in the target variable, confirming the model's effectiveness for intelligent real-time monitoring.

Algorithms have been developed for optimizing the operating modes of the fermentation section based on regression tree models, aimed at finding optimal process settings according to the optimization task at hand. The proposed algorithms enable an average increase in ethanol yield by 8.4% or a reduction in fermentation time by 12.2%.

Chapter four presents the development of an automated intelligent control system based on Reinforcement Learning methods, which ensures stable regulation in non-stationary processes in ethanol production. In cases where a conventional PI controller fails to maintain setpoints and generate high-quality transient responses due to changes in the process or environment (e.g., contamination, equipment aging, ambient temperature fluctuations, or variations in raw material composition), the proposed system offers a robust alternative. Among various agents, the Soft Actor-Critic (SAC) agent with Entropy weight 1 demonstrated superior performance under varying distillation column parameters, achieving high control accuracy ($<0.1\%$), acceptable overshoot (dynamic error 21–26% of the target value), and the lowest total error according to the integral squared error criterion.

A conceptual structure of an automated system for monitoring and controlling technological processes in the production of ethanol and bioethanol using virtual control instruments has been developed. An architecture for an automated monitoring

system for ethanol and bioethanol production was also developed, integrating virtual sensors and implemented using software and hardware platforms such as Unity Pro, Node-RED, and cloud-based tools (InfluxDB, Grafana, Grafana Machine Learning). The proposed architecture enables the incorporation of virtual instruments into the production system and facilitates real-time exchange of production data.

The scientific novelty of the work lies in the development of a new concept for an automated monitoring and control system for ethanol and bioethanol production processes based on virtual control instruments. The study significantly advances systems for monitoring and automated control of technological processes through the application of virtual sensors and mathematical models that enable process stability analysis, parameter prediction and identification, operational optimization, and intelligent control. This approach allows for effective response to dynamic process variability and raw material fluctuations, minimizes human error, enhances decision-making accuracy, reduces energy consumption, and improves product quality—thereby ensuring production resilience and adaptability. For the first time, an ontological structure of mathematical models for technological processes has been formulated, incorporating the virtual sensor as a core element. This structure serves as a decision support tool in model selection, enabling a comprehensive classification and analysis approach, reducing the time required to find relevant solutions, and facilitating the integration of new knowledge into existing modeling systems. The study further develops methods for creating virtual sensors using machine learning to estimate and predict key variables in dynamic systems. This enhances control reliability, enables detection of real sensor faults, and compensates for data loss—thereby improving situational awareness, process stability, and energy efficiency. Additionally, an intelligent control system for a distillation column has been developed. It is the first to implement a reinforcement learning-based approach for stabilizing the concentrations of target components in separation products. This system provides adaptive real-time control of technological variables, rapid disturbance response, and parameter fluctuation management—contributing to higher product quality and system stability under uncertainty.

The practical significance of the obtained results lies in the creation of effective decision support tools and increased reliability of technological systems in ethanol and bioethanol production. The developed ontology of mathematical models offers structured knowledge representation, quick access to relevant modeling approaches, and facilitates model selection for specific production tasks. The proposed classification model enables real-time assessment of production process stability, early detection of instability symptoms, and reduction of associated technological losses. The application of machine learning models for predicting unmeasured variables and implementing intelligent monitoring enhances awareness of system conditions, allows for sensor integrity checks, and compensates for data loss in case of sensor failure. Optimization algorithms based on mathematical process models enable flexible tuning of technological regimes according to raw material variations and production goals. The reinforcement learning-based intelligent control system for the distillation column ensures high regulation accuracy and equipment stability under varying process parameters. The developed automated control system architecture with integrated virtual sensors improves information interaction efficiency and provides flexibility in production structures. The results have found practical application both in educational settings — as methodological approaches to creating virtual sensors for the food industry at the Department of Automation and Computer-Integrated Control Systems named after Prof. A.P. Ladanyuk — and in real industrial conditions at Marylivsky Distillery, confirming their practical value and effectiveness.

Keywords: automation, intelligent technologies, control system with RL agents, technological processes of ethanol and bioethanol production, virtual sensor, industrial IoT, machine learning model, modeling and optimization, monitoring, neural networks, prediction, performance evaluation, systemic and ontological analysis, cloud services.

ЗМІСТ

ВСТУП	18
РОЗДІЛ 1. СУЧАСНИЙ СТАН ТА ІННОВАЦІЙНІ ТЕНДЕНЦІЇ В АВТОМАТИЗАЦІЇ ВИРОБНИЦТВА СПИРТУ ТА БІОЕТАНОЛУ. ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ	24
1.1 Розвиток спиртової промисловості України: перспективи та виклики..	24
1.2 Особливості та інновації технологічних процесів виробництва спирту та біоетанолу	27
1.3 Перспективи та ключові проблеми автоматизації.....	32
1.4 Огляд сучасних підходів та літературних джерел щодо віртуальних інструментів контролю.....	45
1.4.1 Функції віртуальних інструментів контролю	45
1.4.2 Зміст та особливості функціонування віртуальних інструментів контролю	47
1.4.3 Синтез та розробка віртуальних сенсорів	52
1.4.4 Моделі віртуальних сенсорів.....	56
1.4.5 Застосування віртуальних сенсорів в контексті виробництва етанолу	70
1.5 Цифрові двійники як ключовий інструмент віртуального контролю у виробництві спирту та біоетанолу	71
1.5.1 Передумови застосування цифрових двійників	71
1.5.2 Основи функціонування.....	72
1.5.3 Моделювання DT.....	73
1.5.4 Об'єднання даних в DT	74
Висновки до розділу 1 та постановка задач дослідження.....	75
РОЗДІЛ 2. СИСТЕМНИЙ ПІДХІД ДО АНАЛІЗУ ТА МОДЕЛЮВАННЯ ВИРОБНИЧИХ ПРОЦЕСІВ	79
2.1 Системний аналіз технологічних відділень виробництва	79
2.2 Розробка та використання прикладної онтології віртуальних сенсорів та їх математичних моделей.....	88
2.2.1 Місце математичної моделі в ієрархії моделей	88
2.2.2 Відношення та підкласи класу математичної моделі.....	89
2.2.3 Реалізація онтологічного класифікатора	96
2.2.4 Аналіз побудованої онтології	103
2.2.5 Використання онтології.....	104
2.2.6 Місце віртуальних сенсорів в онтології математичних моделей.....	106
2.2.8 Роль та місце моделі в контексті віртуального сенсора.....	109

2.3 Статистичні та класифікаційні підходи для оцінювання стабільності технологічних процесів	110
2.3.1 Постановка задачі класифікації	110
2.3.2 Статистичний аналіз даних роботи дистиляційного відділення	111
2.3.3 Описова статистика даних	112
2.3.4 Порівняльні гістограми	113
2.3.5 Перевірка нормальності розподілу даних для кожної змінної у двох підмножинах даних за допомогою тесту Шапіро-Вілکا.....	114
2.3.6 Матриці кореляції	115
2.3.7 Boxplots та violin plots для порівняння розподілів між групами.....	119
2.3.8 Дисперсійний аналіз ANOVA	121
2.3.9 Kernel Density Estimate (перевірка на мультимодальність)	123
2.3.10 Класифікація стабільності процесу з використанням Gaussian Mixture Model	124
Висновки до розділу 2	131
РОЗДІЛ 3. МОДЕЛЮВАННЯ ТА ВІРТУАЛІЗАЦІЯ ПРОЦЕСІВ ВИРОБНИЦТВА	132
3.1 Моделювання процесу дистиляції з використанням NARX-моделей... 132	
3.1.1 Аналіз даних з датчиків дистиляційної колони	132
3.1.2 Прогнозування концентрації MoleFractionTX за допомогою NARX-моделі	136
3.1.3 Дослідження впливу змін параметрів на поведінку NARX-моделі та пошук оптимальних налаштувань.....	138
3.1.4 Пошук оптимальних значень з використанням бібліотеки Optuna	143
3.2 Моніторинг динамічних змінних у дистиляційному відділенні спиртового виробництва	149
3.2.1 Дослідження моделей ВС для моніторингу виробничих змінних	149
3.2.2 Моніторинг датчиків, що задіяні в контурах регулювання	162
3.3 Оптимізація режимів роботи бродильного відділення	165
3.3.1 Формулювання задачі оптимізації	165
3.3.2 Навчання дерев регресії.....	167
3.3.3 Оптимізація режимів роботи відділення	170
Висновки до розділу 3	173
РОЗДІЛ 4. ІНТЕЛЕКТУАЛЬНЕ КЕРУВАННЯ ТА АПРОБАЦІЯ ВІРТУАЛЬНИХ ІНСТРУМЕНТІВ У ВИРОБНИЦТВО	175
4.1 Концептуальна структура автоматизованої системи моніторингу та керування технологічними процесами	175

4.2.1 Розробка RL-системи керування	177
4.2.2 Вибір RL-агентів для використання в умовах специфічних для спиртового виробництва	177
4.2.3 Дослідження об'єкту керування.....	180
4.2.4 Моделювання системи з локальними ПІ-регуляторами	184
4.2.5 Моделювання системи з RL-агентом	188
4.3 Розробка та апробація віртуальних інструментів у виробництво	214
4.3.1 Архітектура автоматизованої системи моніторингу.....	214
4.3.2. Обґрунтування вибору програмних засобів та архітектура системи	216
4.3.2 Реалізація віртуального сенсора в умовах спиртового виробництва..	217
4.4 Забезпечення безпеки та стійкості автоматизованої системи управління	225
4.4.1 Забезпечення функціональної стійкості розробленої системи.....	225
4.4.2 Інформаційна безпека системи	227
Висновки до розділу 4	228
ОСНОВНІ ВИСНОВКИ ТА РЕЗУЛЬТАТИ	230
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	234
ДОДАТКИ	248
ДОДАТОК А. Код програми для аналізу даних дистиляції	248
ДОДАТОК Б. Код програми для статистичного аналізу та класифікації.	250
ДОДАТОК В. Код програми для прогнозування концентрації за допомогою NARX-моделей	264
ДОДАТОК Г. Код програми для інтелектуального моніторингу динамічних процесів у дистиляційному відділенні	275
ДОДАТОК Д. Код програми для задачі оптимізації процесу бродіння	302
ДОДАТОК Е. Коди додаткових програм для систем автоматичного керування.....	309
ДОДАТОК Є. Акти впровадження	311

ВСТУП

Актуальність роботи. Покращення ефективності та якості автоматизованого управління процесами у спиртовій промисловості залишається важливою задачею, яка безпосередньо впливає на конкурентоспроможність підприємств галузі та їх здатність відповідати вимогам сучасного ринку. Розробка і впровадження ефективних систем автоматизованого управління дозволяє знизити витрати, підвищити продуктивність та забезпечити стабільність процесів, що в свою чергу сприяє зменшенню енергоспоживання та покращенню якості продукції. Крім того, інтеграція інноваційних технологій, таких як штучний інтелект та системи підтримки прийняття рішень, здатна значно підвищити рівень автоматизації та оптимізації виробничих процесів, створюючи умови для сталого розвитку підприємств.

Спиртове виробництво як об'єкт управління характеризується наявністю інерційних ефектів, нестаціонарних умов та складних нелінійних зв'язків між змінними. Традиційні підходи до автоматизації, засновані на використанні фізичних сенсорів і стандартних алгоритмів управління, часто не забезпечують необхідної точності й адаптивності у таких умовах. Відсутність можливості оперативно отримувати інформацію про ключові параметри процесу, а також обмеження в прогнозуванні поведінки системи з урахуванням її складної динаміки, знижують ефективність управління.

Віртуальні інструменти контролю відкривають нові можливості для моніторингу та управління спиртовим виробництвом, дозволяючи підвищити точність вимірювання важливих показників, компенсувати збої фізичних сенсорів, прогнозувати оптимальні режими роботи та забезпечувати адаптацію до змін виробничих умов.

Зв'язок роботи з науковими програмами, планами, темами. Дисертація виконана в рамках науково-дослідної роботи кафедри автоматизації та комп'ютерних технологій систем управління ім. проф. А.П. Ладанюка «Науково-технічні основи розробки систем автоматизації складних об'єктів з

використанням інтелектуальних методів та технологій Індустрії 4.0» (номер державної реєстрації №0121U113308).

Мета і завдання дослідження. Метою дисертаційної роботи є підвищення ефективності виробництва спирту та біоетанолу шляхом вдосконалення автоматизованої системи моніторингу та керування, яка базується на використанні віртуальних інструментів контролю.

Для досягнення поставленої мети необхідно було вирішити наступні **завдання**:

- провести аналіз сучасного стану розвитку віртуальних інструментів контролю для технологічних комплексів спиртового та біоетанольного виробництва, визначити основні напрями наукових досліджень та їх актуальність, сформулювати основні проблеми у цій галузі та обґрунтувати методи для їх вирішення;
- провести аналіз сучасних математичних моделей технологічних об'єктів з метою формування онтологічної моделі для використання віртуальних сенсорів у моніторингу технологічних процесів;
- виконати синтез та провести моделювання віртуальних сенсорів для визначення та прогнозування ключових технологічних показників спиртового та біоетанольного виробництва;
- здійснити розробку концептуальної структури автоматизованої системи моніторингу та керування технологічними процесами з використанням віртуальних інструментів контролю;
- синтезувати автоматизовану систему керування технологічними процесами з використанням інтелектуальних механізмів, провести її моделювання;
- розробити хмарну архітектуру для системи моніторингу технологічних процесів виробництва спирту та біоетанолу, яка забезпечить високу надійність, гнучкість та масштабованість. Реалізувати запропоновану архітектуру з використанням сучасних методів обробки даних і віртуальних інструментів контролю, а також провести її апробацію.

Об'єктом дослідження є технологічні процеси та комплекси виробництва спирту та біоетанолу, що функціонують в умовах невизначеності та характеризуються нелінійністю та багатозв'язністю.

Предметом дослідження є автоматизовані системи моніторингу та керування технологічними процесами виробництва спирту та біоетанолу, які базуються на віртуальних інструментах, математичних моделях, інтелектуальних механізмах.

Методи дослідження. Методи дослідження включають комплексний підхід, що поєднує теоретичне моделювання, математичне моделювання, алгоритмізацію та експериментальну апробацію результатів. Для дослідження процесів виробництва спирту та біоетанолу використовувались методи системного аналізу та математичного моделювання технологічних процесів з використанням віртуальних сенсорів. Моделювання стабільності системи здійснювалось шляхом побудови статистичних моделей для аналізу динаміки процесів на основі реальних експериментальних даних. Для розробки інтелектуальних систем керування застосовувались методи підкріплювального навчання (Reinforcement Learning, RL), що дозволяє адаптивно регулювати технологічні змінні в реальному часі, враховуючи зміни характеристик сировини та збурення. Створення віртуальних сенсорів базувалось на методах машинного навчання, зокрема на використанні алгоритмів для прогнозування невимірюваних змінних, а також для діагностики і компенсування втрат даних через відмову сенсорів. Крім того, для оптимізації технологічних режимів були застосовані методи математичної оптимізації, включаючи методи ітеративного пошуку.

Наукова новизна одержаних результатів:

- знайшли подальшого розвитку системи моніторингу та керування технологічними процесами виробництва спирту та біоетанолу, що базуються на використанні віртуальних сенсорів та моделях процесів, які виконують аналіз стабільності, визначення та прогнозування ключових показників на основі інших технологічних змінних, оптимізацію технологічних режимів та інтелектуальне

керування. Запропонований підхід дозволяє забезпечити ефективне управління в умовах складної динаміки технологічних процесів та зміни характеристик вхідної сировини, знижуючи людський фактор та підвищуючи точність прийняття рішень. Це сприяє зниженню енерговитрат і покращенню якості продукції, що робить виробничі процеси більш стійкими та адаптивними до змінних умов;

- вперше розроблено онтологію математичних моделей технологічних процесів, в структуру якої включено віртуальний сенсор як ключовий елемент. Онтологія слугує системою підтримки прийняття рішень при вивченні та виборі моделей технологічних процесів. Вона забезпечує структурований підхід до класифікації та аналізу моделей, дозволяючи користувачам швидко ідентифікувати відповідні типи моделей для конкретних завдань моделювання. Це сприяє підвищенню ефективності наукових досліджень і проєктування, зменшує час на пошук релевантних підходів та полегшує інтеграцію нових знань у вже існуючі системи аналізу та моделювання.

- знайшли подальшого розвитку підходи до розробки віртуальних сенсорів на основі моделей машинного навчання для оцінювання та прогнозування ключових змінних у динамічних системах. Запропонований підхід забезпечує підвищену надійність контролю технологічних процесів, дозволяючи здійснювати оцінку стану сенсорів, а також прогнозувати та компенсувати втрати даних у разі їх поломки або недоступності вимірювань. Це дозволяє підвищити інформаційну обізнаність про функціонування технологічних процесів, що, в свою чергу, сприяє підвищенню стабільності, якості продукції та енергоефективності виробництва.

- розроблено інтелектуальну систему керування колоною дистиляції на основі RL-агентів, яка вперше використовує підхід підкріплювального навчання для стабілізації концентрацій цільових компонентів у продуктах розділення. Запропонована система забезпечує адаптивне регулювання технологічних змінних у реальному часі, що дозволяє ефективно реагувати на збурення та коливання параметрів сировини. Це дозволяє мінімізувати відхилення

концентрацій від заданих значень, підвищити якість кінцевого продукту та забезпечити стабільність роботи дистиляційної колони в умовах невизначеності.

Практичне значення отриманих результатів:

- розроблена онтологія математичних моделей слугує системою підтримки прийняття рішень при вивченні та виборі моделей технологічних процесів забезпечуючи їх зручну класифікацію та швидкий пошук відповідних підходів;
- розроблена модель класифікації на базі методу Gaussian Mixture Model для оцінки стабільності системи в реальному часі дає змогу виявляти ознаки нестабільності на ранніх етапах та зменшити пов'язані технологічні витрати;
- передбачення невимірюваних змінних за допомогою моделей машинного навчання та інтелектуальний моніторинг динамічних процесів підвищують надійність й інформаційну забезпеченість системи, дозволяючи здійснювати додатковий контроль над показниками датчиків, їхнім станом, а також надаючи можливість прогнозувати й замінювати необхідні дані в разі поломки датчика чи неможливості вимірювання;
- алгоритми оптимізації на основі моделей процесів дають змогу знаходити оптимальні налаштування технологічних режимів для поставлених виробничих цілей в умовах зміни характеристик сировини;
- розроблена інтелектуальна система керування з використанням RL-агентів для колони дистиляції дозволяє підтримувати високу стабільність та точність регулювання складних процесів спиртового виробництва в умовах зміни параметрів об'єкта.
- розроблена архітектура автоматизованої системи моніторингу виробництвом спирту та біоетанолом з використанням віртуальних сенсорів дозволяє інтегрувати віртуальні інструменти у виробничу систему, забезпечуючи ефективну інформаційну взаємодію та структурну гнучкість;
- результати дослідження, представлені у вигляді підходів до створення віртуальних сенсорів для харчової промисловості, були впроваджені в навчальний процес кафедри автоматизації та комп'ютерних технологій систем

управління ім. проф. А.П. Ладанюка, а також пройшли апробацію на «Марилівському спиртовому заводі».

Особистий внесок у розробку наукових результатів. У дисертаційній роботі представлено наукові результати, отримані безпосередньо автором. Самостійно були сформульовані дослідницькі завдання, обрані методи їх вирішення, а також проведено аналіз та узагальнення експериментальних даних. У випадках використання матеріалів із колективних публікацій, до дисертації включено лише ті концепції та висновки, що розроблені особисто автором.

Апробація роботи. Основні положення дисертації та її результати доповідалися та обговорювалися на наукових конференціях: 86-й, 88-й та 90-й Міжнародних наукових конференціях молодих учених, аспірантів і студентів «Наукові здобутки молоді – вирішенню проблем харчування людства у XXI столітті» (Київ, НУХТ, 2020, 2022, 2024); VII–XI Міжнародних науково-технічних Internet-конференціях «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами» (Київ, НУХТ, 2020–2024); Третій міжнародній науково-практичній конференції «Сучасні тенденції розвитку інформаційних систем і телекомунікаційних технологій» (Київ, НУХТ, 2021); I Міжнародній науково-практичній конференції «Штучний інтелект та інформаційні технології» (Київ, НУХТ, 2024); XXXIV International Scientific and Practical Conference (Флоренція, Італія, 2023).

Публікації. Основні теоретичні та експериментальні результати наукових досліджень висвітлено у 18 наукових працях, з яких 5 статей у наукових фахових виданнях України, у тому числі включених до міжнародних наукометричних баз даних, та 13 тез наукових доповідей.

Структура та об'єм роботи. Дисертаційна робота складається з вступу, чотирьох розділів, списку літератури з 156 найменувань і додатків. Повний обсяг дисертації 313 стор., з яких зміст викладено на 216 стор. друкованого тексту, містить 187 рисунки, 31 таблицю та 7 додатків.

РОЗДІЛ 1. СУЧАСНИЙ СТАН ТА ІННОВАЦІЙНІ ТЕНДЕНЦІЇ В АВТОМАТИЗАЦІЇ ВИРОБНИЦТВА СПИРТУ ТА БІОЕТАНОЛУ. ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Розвиток спиртової промисловості України: перспективи та виклики

Спиртова галузь є однією з найприбутковіших та затребуваних галузей України. Вона відіграє важливу роль у функціонуванні інших галузей, таких як харчова, енергетична, фармацевтична та інші.

Станом на 2014 рік спиртова промисловість в Україні була представлена 84 спиртовими заводами, що сумарно мали загальну потужність понад 60 млн. дал. спирту, у тому числі зернового – 35,6 млн. дал., мелясного – 24,8 млн. дал.

Значна доля виробленого спирту, а саме 20-22 млн. дал. спрямована на задоволення потреб внутрішнього споживчого ринку харчового спирту, потенційний же об'єм для експорту становить близько 10 млн. дал.

Спиртова промисловість повністю задовольняє внутрішні потреби держави у спирті для виготовлення лікєро-горілочаних виробів, використовуючи лише 30-35% своїх загальних потужностей, що є підставою для пошуку нових ринкових та стратегічних можливостей реалізації спиртової продукції [1].

Провідними виробниками спирту є Черкаська, Вінницька, Київська, Чернігівська, Житомирська обл. Враховуючи те, що потужність спиртових заводів України завантажена лише на третину, одним із головних завдань спиртової галузі є розширення асортименту товарної продукції відповідно до потреб ринку [2].

В той час як аналіз світового ринку споживачів спиртової промисловості показує що у розвинених країнах світу 70% виробленого спирту використовують для виробництва палива, 15% - для приготування спиртових напоїв, 15% - в хімічній та лакофарбовій промисловості, в Україні спостерігається інша тенденція: близько 90% виробленого складає харчовий спирт в обсягах виробництва підприємств спиртової галузі який йде на забезпечення

виробництва спиртових напоїв та окремих галузей промисловості таких як фармацевтична. Лише 10% від загальних обсягів складають технічні спирти, які використовуються в хімічній промисловості та для виробництва палива [1].

Відповідно до світових тенденцій щодо екологічності та використання відновлювальних ресурсів, етиловий спирт має потенціал для використання в якості біопалива. Україна належить до енергодефіцитних країн та її забезпеченість власними енергоресурсами не перевищує 30% [3].

Можливість збільшення виробництва спирту, переобладнання заводів для виробництва технічного спирту та реорганізація енергетично-паливної системи України для його використання в якості паливних ресурсів має значні перспективи у зменшенні енергозалежності країни. Тому одним із викликів які стоять перед спиртовою галуззю та промисловістю в цілому є паливна реформа в рамках якої виробництву етанолу відіграє ключову роль.

Перехід на етилове пальне дозволить одночасно розв'язати проблеми забруднення екології та відсутності в Україні власних сировино-паливних ресурсів достатніх для забезпечення у власному пальному.

Іншим завданням яке стоїть перед спиртовою галуззю є покращення ефективності виробництва та впровадження інновацій в цій галузі.

Протягом останніх років динаміка виробництва спирту має тенденцію до зниження, що вказує на поступовий занепад даної галузі (Рис. 1.1, [2]).



Рис. 1.1 Динаміка виробництва спирту в Україні в 2007-6м15 рр., млн. дал

Однією з причин даної тенденції є втрата зовнішніх ринків внаслідок неконкурентоспроможності вітчизняного спирту на міжнародному ринку. Через

низьку енергоефективність виробництва на фоні багаторічної відсутності модернізації, собівартість українського спирту на 58% вища, ніж середньоєвропейська [2].

Виробництво біоетанолу в Україні. Використання біоетанолу, зневодненого етилового спирту, є ефективним способом скорочення споживання сирої нафти та зменшення забруднення довкілля. Ці чинники сприяли значному зростанню виробництва біоетанолу в усьому світі. В Україні розвиток цього напрямку обумовлений необхідністю подолання енергетичного дефіциту, зменшення залежності від імпорту нафти та використанням значного природно-ресурсного потенціалу країни [4]. Зокрема, ухвалено норми, які вимагають, щоб усі сумішеві бензини, що реалізуються на території України, містили не менше 5% біоетанолу. Розроблено і введено в дію галузевий національний стандарт ГСТУ 320.00149943.015-2000 «Бензини моторні сумішеві неетильовані. Технічні умови» та державний національний стандарт ДСТУ 7166:2010 «Біоетанол. Технічні умови», які визначають стандарти якості палива з використанням біоетанолу [5].

Україна має значні перспективи для розвитку виробництва біоетанолу завдяки надлишку сировини та обсягу спиртового виробництва, що перевищує потреби харчової та хімічної промисловості [4]. Екологічні та економічні переваги біоетанолу стимулюють його використання, що призводить до зростання попиту [6, 7]. Уже з 2014 року вітчизняне виробництво не задовольняє потреби у біоетанолі для сумішевих бензинів, що підкреслює необхідність розвитку нових виробничих потужностей [6]. Забезпечення постійно зростаючих потреб у біоетанолі, його якості та продуктивності виробництва є критично важливим для національної енергетичної безпеки.

Для відновлення експортних та промислових перспектив вітчизняна спиртова галузь потребує низки реформ, серед яких розширення асортименту, який би включав інтеграцію в енергетично-паливну систему країни, оновлення обладнання та впровадження новітніх, зокрема інтелектуальних засобів

автоматизації, з метою оптимізації виробництва, яка б сприяла зменшенню собівартості, збільшенню виходу на одиницю ресурсу, та якості продукту.

1.2 Особливості та інновації технологічних процесів виробництва спирту та біоетанолу

Сучасні промислові підприємства та технологічні комплекси оперують складними процесами, які складають основу виробництва готового продукту. Для таких комплексів характерне виникнення труднощів в управлінні, які пов'язані з наявністю великої кількості альтернатив та багатокритеріальності рішень, невизначеності вихідної інформації. З цієї точки зору спиртові заводи являються складними об'єктами управління, які характеризуються багатомірністю, багатозв'язністю, нестаціонарністю, та розглядаються як об'єкти з послідовно-паралельною структурою. В критичних ситуаціях людина не здатна справитись з аналізом та впливом інформації та часто помиляється в ідентифікації виникаючих ситуацій. Розпізнавання ситуації на локальному рівні засобами автоматизованої, та навіть мікропроцесорної техніки неможливе, оскільки потрібен аналіз цілого чи окремих його функціональних частин. Це знижує швидкість та якість прийнятих рішень, роблячи їх значною мірою залежними від досвіду та особистих характеристик людини-оператора [8]. Крім того, неможливість виміру комплексних показників з високою інерційністю під час виконання процесу, та довгий час лабораторних аналізів таких даних ускладнюють забезпечення ефективного управління комплексом виробництва, що зменшує якість продукту та призводить до втрати коштів.

У спиртовому виробництві, особливо в процесах ректифікації, наявні безліч технологічних параметрів що не піддаються прямому вимірюванню та мають непряму залежність від контрольованих параметрів. Так, під час перегонки дозрілої бражки разом з виділенням летких сполук відбувається утворення низки речовин, які ускладнюють подальше очищення спирту та знижують його якість. Новоутворення естерів, альдегідів і ацеталей органічних кислот відбувається завдяки взаємодії спиртів, кислот і альдегідів, продуктів розпаду амінокислот,

сірчистих з'єднань та інших сполук бражки. Ці сполуки потрапляють в продукт разом з сировиною, водою і парою, або утворюються під час перебігу хімічних і біохімічних реакцій у процесах бродіння, перегонки бражки, очищення спирту, а також під час його зберігання в металевих резервуарах [9].

Зазвичай етанол виробляють з глюкози методом мікробної ферментації. Для виробництва спирту можуть використовуватися багато видів сировини, проте незалежно від сировини, процес виробництва етанолу проходить через стандартні стадії (Рис. 1.2) [10]. Незалежно від технології в процесі виробництва етанолу відбуваються складні хімічні та біологічні перетворення, утворюється безліч проміжних речовин, що водночас має велику інерційність та відсутність методів прямого оперативного вимірювання. Ці фактори мають значний вплив на якість кінцевого продукту.

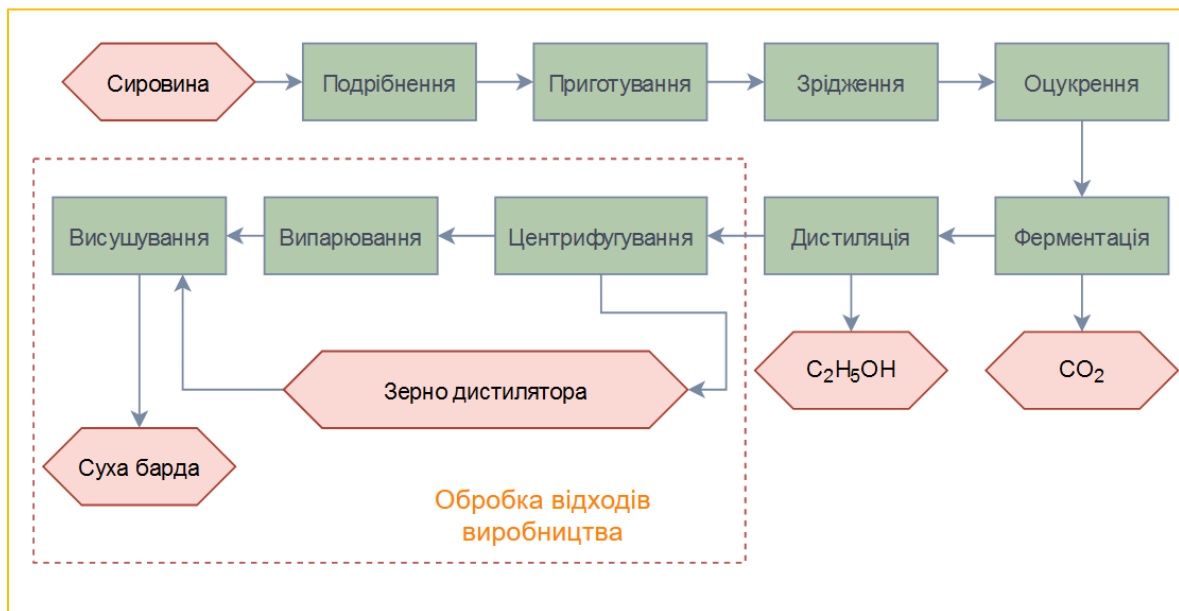


Рис. 1.2 Типовий процес виробництва спирту

У профільній літературі [12, 13] наявні приклади значної зміни якості продукту при виробництві етилового спирту шляхом зміни тих чи інших технологічних параметрів, застосування інших алгоритмів та налаштувань виробництва.

Наприклад в [12] запропонована авторами зміна в технології ректифікації в режимі контрольованих циклів затримки рідини дозволила підвищити ступінь вилучення та кратність концентрації проміжних та кінцевих домішок спирту на

30%, разом з цим скоротивши витрати гріючої пари на 35% в порівнянні з типовою технологією.

В роботі [13] запропоновано зміну в технології ректифікованого етилового спирту шляхом використання кубової рідини спиртової колони для проведення гідроселекції, подача спиртовмісних фракцій, збагачених органічними домішками на тарілку живлення, що призводить до зменшення витрат на нагрівання технологічної води до необхідної температури на 80% за рахунок рециркуляції кубової рідини спиртової, розгінної колони та колони екстрактивної ректифікації.

Часто якість продукту забезпечується шляхом ускладнення структури устаткування, яке керує перебігом процесу. Це підвищує окремі показники та керованість якістю продукту, проте в свою чергу створює ще більше перехресних впливів параметрів між собою, ускладнюючи управління технологічним процесом та потребуючи значної кількості часу на знаходження оптимальних керуючих інструкцій, які б дозволили максимально розкрити ефективність впроваджених технологічних змін.

Особливості виробництва біоетанолу. Вимоги до якості спиртової та біоетанолової продукції суттєво відрізняються, що визначає специфіку технологічних процесів їх виробництва. При виробництві харчового спирту основна увага приділяється забезпеченню високої якості сировини, суворому контролю параметрів ферментації, таких як стерильність, температура та якість дріжджів, а також ретельній ректифікації. Ці заходи спрямовані на мінімізацію вмісту небажаних домішок, таких як сивушні масла, метанол, ацетон та інші.

Натомість виробництво біоетанолу має інші пріоритети. Якість сировини та наявність домішок не є критично важливими, але вирішальне значення має ступінь очищення спирту від води. Якщо для харчового спирту допускається залишковий вміст води на рівні 4–5%, то для біоетанолу такий показник є незадовільним, оскільки він не відповідає вимогам до використання у паливних системах. Щоб забезпечити відповідність цим вимогам, концентрація біоетанолу має перевищувати 99,5% [14].

Для виробництва високоякісного безводного етанолу, що містить не менше 99,5% етанолу за об'ємом, використовуються різноманітні методи дегідратації. Серед найпопулярніших з них можна виділити наступні [15]:

1. Адсорбційна дистиляція.

Цей метод передбачає використання адсорбційних матеріалів, таких як молекулярні сита, що дозволяють відокремлювати воду за розміром молекул. Молекулярні сита мають пори розміром приблизно 3 Å, які адсорбують менші молекули води (діаметром 2,5 Å), тоді як більші молекули етанолу (діаметром 4 Å) залишаються поза ситами. У типовому процесі використовують два адсорбційні шари, які працюють по чергові: один шар адсорбує воду, тоді як інший регенерується при високій температурі (288 °C). Прикладом матеріалу для використання в якості відновлювального адсорбційного матеріалу є цеоліт. Використання органічних адсорбентів, таких як солома чи деревна стружка, має екологічні переваги, але вони не є регенеративними.

2. Азеотропна дистиляція.

Цей метод базується на додаванні до етанол-водяної суміші третього компонента, який змінює відносну леткість компонентів. Такий третій компонент (азеотропоутворювач), наприклад бензол або циклогексан, дозволяє розділяти суміш у колоні, утворюючи трикомпонентний азеотроп. Однак цей метод вимагає значних енергетичних витрат, оскільки азеотропоутворювач необхідно постійно циркулювати, а використання токсичних речовин, як-от бензол, несе ризики для здоров'я людей і довкілля.

3. Хімічна дегідратація.

У цьому методі використовуються гігроскопічні хімічні речовини, такі як оксид кальцію (вапно), які хімічно зв'язують воду. Реакція утворює нерозчинний гідроксид кальцію, який потім видаляється фільтрацією. Хоча метод забезпечує високу чистоту етанолу (понад 97%), він споживає багато енергії і є складним для масштабного виробництва через періодичність процесу.

4. Дифузійна дистиляція.

Цей метод полягає у випаровуванні суміші до температур нижче точки кипіння та розділенні компонентів за рахунок різної дифузійної здатності в інертному газі. Дифузійна дистиляція показала відповідність рівнянням рівноваги "пара-рідина", однак вимагає подальших досліджень для промислового впровадження.

5. Екстрактивна дистиляція.

Цей метод передбачає введення в суміш спеціального розчинника, який збільшує різницю в леткості компонентів, що полегшує їх розділення. Використання розчинників, таких як полімери, дозволяє ефективно розділяти етанол і воду.

6. Мембранні процеси.

Технологія базується на використанні напівпроникних мембран, які пропускають тільки молекули води або етанолу. Мембранні процеси є енергоефективними, але поки що їх масштабованість та вартість обмежують широке впровадження.

7. Вакуумна дистиляція.

Зниження тиску в дистиляційній колоні дозволяє зменшити температуру кипіння суміші, що знижує енергетичні витрати. Метод застосовується для уникнення теплової деградації етанолу, але потребує складного обладнання.

Кожен із цих методів має свої переваги й недоліки, залежно від економічних, екологічних та технічних вимог процесу. Сучасні дослідження спрямовані на зниження енерговитрат і підвищення екологічної безпеки виробництва біоетанолу.

Оптимізація виробництва спирту зіштовхується з певними труднощами через наявність різних технологічних особливостей, характерних для цього процесу. Якість та склад сировини може сильно відрізнятися залежно від регіону, пори року та клімату. Ці зміни в свою чергу можуть вплинути на ефективність та узгодженість виробничого процесу та потребують ретельного контролю та коригування. Крім того, на якість етанолу впливає безліч взаємопов'язаних процесів, таких як подрібнення, ферментація та дистиляція. Умови роботи

кожного з цих процесів можуть відрізнятися і будь-яке відхилення може вплинути на якість кінцевого продукту.

Виробництво спирту є енергомістким процесом, що потребує великої кількості тепла та електроенергії. Досягнення високої енергоефективності при збереженні виробничих потужностей та якості є складним завданням, що потребує передових методів управління та оптимізації, в додаток до цього виробництво етанолу регулюється різними нормами та стандартами, такими як екологічні норми, норми безпеки та стандарти якості. Забезпечення дотримання цих правил може ускладнити виробничий процес та вимагати додаткового моніторингу та контролю.

Для подолання цих труднощів може бути дуже корисним використання передових методів управління процесами та оптимізації, таких як віртуальні аналізатори, програмні датчики та машинне навчання. Ці методи можуть забезпечити моніторинг та контроль виробничого процесу в режимі реального часу, виявляти мінливість та відхилення процесу та оптимізувати процес для досягнення максимальної ефективності та якості.

1.3 Перспективи та ключові проблеми автоматизації

Розпочавшись у кінці першого десятиріччя XXI століття Четверта промислова революція принесла сфері автоматизації як нові перспективи, так і нові виклики.

Основним пріоритетом на якому ґрунтується Індустрія 4.0 є повна автоматизація виробництва, аналіз та інтеграція великих даних, оптимізація наявних систем. Результатом даного впровадження є чергове підвищення ефективності виробництва що ставить неоптимізоване виробництво у вразливе, неконкурентоспроможне становище, змушуючи промисловість слідкувати за технологічними інноваціями [17].

Індустрію 4.0 можна розглядати як новий рівень організації та контролю над усім ланцюжком створення вартості життєвого циклу продуктів. Основою для Четвертої промислової революції є доступність усієї необхідної інформації в

режимі реального часу шляхом підключення всіх інстанцій, залучених до ланцюжка створення вартості. По суті Індустрія 4.0 є переглянутим підходом до виробництва який використовує останні технологічні винаходи та інновації для підвищення рівня автоматизації та цифровізації виробництва, виробничих та промислових процесів та досягнення найвищої якості продукту шляхом впровадження цих технологій [18].

Перехід та впровадження основ Індустрії 4.0 в промисловості відбувається поступово, крок за кроком підвищуючи рівень автоматизації виробництва. Можна виділити шість послідовних кроків розвитку підприємства які описують відповідну зрілість підприємства відповідно вимог Індустрії 4.0 (Рис. 1.3) [19]:

1. Комп'ютеризація

Комп'ютеризація охоплює ізольоване використання інформаційних технологій на підприємстві. Це дозволяє пришвидшити та покращити якість повторювальних дій. Крім того, комп'ютеризація може спростити збирання та зберігання даних, які можна використовувати для аналізу та прийняття рішень, це приводить до скорочення часу та зусиль, необхідних для виконання завдань. Комп'ютеризація може допомогти знизити витрати на робочу силу та підвищити продуктивність, що призведе до економії коштів для підприємств та організацій. Перевагами впровадження цього кроку є підвищення ефективності виробництва та зменшення ризику виникнення помилок.

2. Об'єднання

Наступний крок, Об'єднання, являє собою об'єднання інформаційних технологій на підприємстві. Технології які використовуються можуть взаємодіяти одна з одною, таким чином реалізуючи API-інтерфейси від операційних технологій до бізнес рівня. Таким чином відбувається приблизне відображення основних бізнес-процесів.

Обидва кроки, Комп'ютеризація та Об'єднання описуються терміном Цифровізація та формують основи для впровадження Індустрії 4.0 при цьому не являючись її частиною.

3. Видимість

Наступний крок, Видимість, являє собою відправну точку для Індустрії 4.0. На цьому етапі актуальна цифрова модель підприємства доступна в режимі реального часу, надаючи інформацію про поточний стан виробничої системи. За допомогою цього стає можливою інтеграція стратегічних систем, таких як «Керування життєвим циклом продукту» (Product Lifecycle Management).

4. Прозорість

На етапі Прозорості вже наявні виробничі дані агрегуються та піддаються детальному аналізу для ідентифікації взаємозв'язків елементів в системі. Такі методи аналізу як штучний інтелект мають змогу провести таку обробку виробничих даних. Це дозволяє знаходити оптимуми та помилки в роботі системи, а також своєчасно про них попереджувати. Відповідно правильна оцінка поведінки системи дозволяє ціленаправлено впливати на неї з точки зору моніторингу стану.

5. Прогностична здатність

Прогностична здатність дозволяє проводити побудову можливих сценаріїв майбутньої поведінки системи на основі проаналізованих даних. Таким чином стає можливим передбачення майбутніх ситуацій, що дає змогу своєчасно приймати відповідні рішення та превентивні міри.

6. Адаптивність

Реалізація Адаптивності полягає в можливості реакції автономної системи на зміну поточного стану виробництва. Іншими словами виробнича система являється самооптимізуючою та здатна самостійно приймати відповідні рішення. Таким чином рівень автоматизації визначається в залежності від складності рішення та відношенню витрат до користі. Ціль останнього етапу досягається якщо проаналізовані дані можуть використовуватися для автономного керування в режимі реального часу, що забезпечує підприємство якісно новими перевагами.

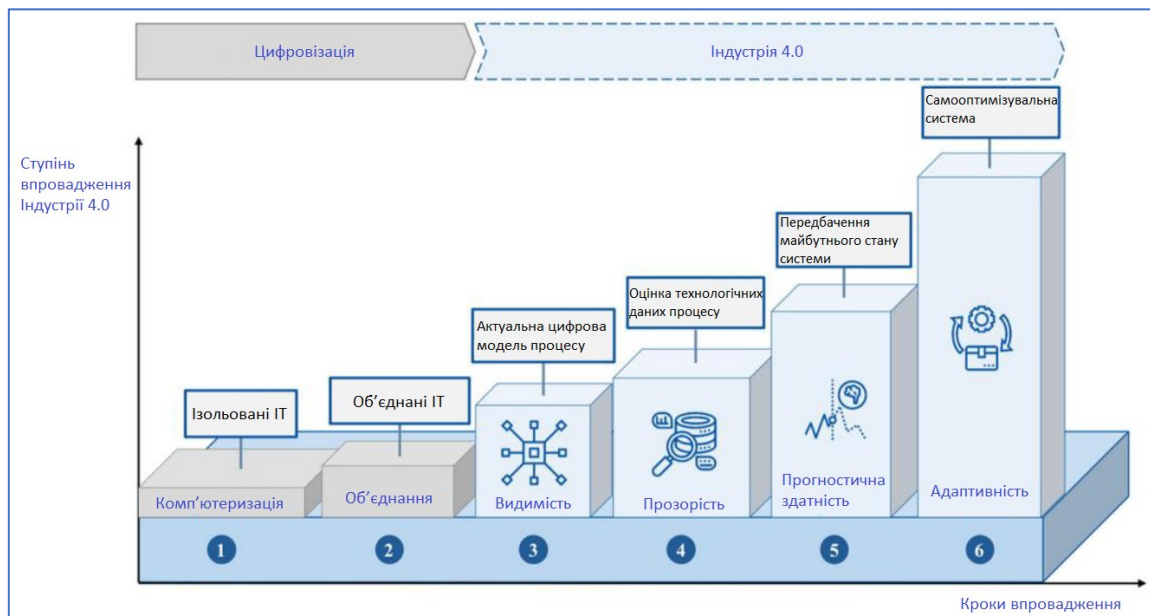


Рис. 1.3 Процес розвитку Індустрії 4.0

Можна виділити дев'ять технологій на яких ґрунтується подальша трансформація індустріального виробництва це [20]:

1. Big data та аналітика

У цифровому світі значно зросли обсяги даних, що генеруються та зберігаються за короткий проміжок часу. Цей швидкий темп зростання кількості складних, різнорідних даних із різних джерел включаючи машини та датчики створив багато проблем з їх зберіганням та обробкою. Наявні інструменти обробки не можуть давати повні результати обробки таких даних протягом розумного періоду часу. Це сприяло розвитку та впровадженню нових технологій обробки Big data. Трьома основними мотивами для впровадження технології Big data є [21]:

- Мінімізація витрат на апаратне забезпечення;
- Перевірка цінності великих масивів даних перед залученням значних обчислювальних ресурсів;
- Зменшення витрат на обробку.

Big data це термін що описує масивні набори даних, які мають велику, більш різноманітну та складну структуру з труднощами зберігання, аналізу та візуалізації для подальших процесів та або результатів. Процес дослідження

об'ємних масивів даних для виявлення прихованих закономірностей та кореляцій називається аналітикою великих даних [22].

За допомогою Big data підприємства можуть аналізувати величезні обсяги інформації, щоб виявляти закономірності, тенденції та ідеї, які можуть допомогти прийняти більш ефективні рішення. Аналізуючи великі обсяги даних, організації можуть виявити недоліки та області для покращення своєї діяльності, отримати краще уявлення про потенційні ризики, такі як фінансові або операційні ризики, що дозволить завчасно вжити запобіжних заходів для управління цими ризиками та їх зниження. Це може призвести до економії коштів та підвищення ефективності.

Серед методів обробки Big data можна виділити такі як фільтр Блума, хешування, індексування та паралельні обчислення. Вони мають як свої переваги, що дає можливість їх використання для обробки великої кількості даних, так і свої недоліки, що накладають певні обмеження на застосування цих методів.

Хешування використовує хеш-функцію для розміщення збережених даних в порядку значення функції вхідних даних, що дозволяє пришвидшувати навігацію по ним. Проте хешування недоцільне, якщо дані організовані в певному порядку, а також для пошуку діапазонів даних. Найкраще хешування працює з даними які є випадковими та дискретними.

Індексація дозволяє сформувати показник для пошуку по відповідному полю в збережених даних, таким чином оптимізуючи структуру під пошук. Недоліком індексації є висока ресурсна ціна пошуку, тому нині розробка нових алгоритмів індексації та їх дослідження є дуже популярною темою дослідження.

Фільтр Блума дозволяє зберігати набір даних з ефективним використанням місця за рахунок ймовірності помилкового спрацювання на основі запитів. Ефективність збереження досягається шляхом допустимої помилки при збереженні хибнопозитивних результатів. Це накладає обмеження на використання даного методу в для тих задач які не допускають таких допущень при збереженні даних.

Паралельні обчислення дозволяють використовувати декілька ресурсів одночасно для виконання задачі. Паралелізм завдань допомагає досягти високої продуктивності великомасштабних наборів даних. Це дозволяє отримати такі переваги як швидка обробка, розділення складних завдань та зменшення енергоспоживання, проте має свої недоліки, такі як частотне масштабування [21].

В контексті Індустрії 4.0 засоби Big data використовуються для збору та оцінки даних з різних джерел, таких як виробничі обладнання та системи та системи управління підприємством. Це дозволяє економити енергію, оптимізує якість продукції та обслуговування обладнання, використовувати ці дані для систем прийняття рішень в реальному часі.

2. Автономні роботи

Роботи відіграють важливу роль в сучасному виробництві. Суттєвим аспектом Індустрії 4.0 являються автономні методи виробництва, основані на використанні роботів, які можуть розумно виконувати задачі приділяючи особливу увагу безпеці, гнучкості та універсальності.

В багатьох областях виробництва вже давно використовують роботів для вирішення складних задач, але роботи продовжують розвиватися стають більш автономними та гнучкими з значно більшим діапазоном можливостей. Завдяки розвитку технологій область використання роботів розширюється та включає в себе різноманітні функції, такі як виробництво, логістика та навіть підтримка документації. Завдяки революціям в області машинного зору тепер роботи можуть виконувати значно більше завдань.

Автономні роботи продукують ряд переваг у виробництві, зокрема:

- Підвищення ефективності та продуктивності. За рахунок того що автономні роботи можуть працювати безперервно, не втрачаючи якість виконання завдань, що призводить до підвищення ефективності та продуктивності виробничого процесу.

- Зниження витрат на робочу силу. Роботи можуть виконувати завдання, які зазвичай потребують людської праці, знижуючи витрати на робочу силу та вивільняючи людей, щоб вони могли зосередитися на інших завданнях.

- Підвищена безпека: автономні роботи можуть виконувати небезпечні завдання у небезпечних умовах, знижуючи ризик травмування людей.

- Послідовність та точність. Роботи можуть виконувати завдання з високим ступенем узгодженості та точності, знижуючи ризик помилок та підвищуючи якість виробничого процесу.

- Гнучкість. Автономних роботів можна запрограмувати на виконання широкого кола завдань, що робить їх легко адаптованими до потреб виробництва.

- Збір та аналіз даних. Роботи можуть збирати та аналізувати дані про виробничий процес, надаючи інформацію, яку можна використовувати для оптимізації виробничого процесу та підвищення якості продукції.

Загалом впровадження автономних роботів у виробництві призводить до підвищення ефективності, зниження витрат, підвищення безпеки та підвищення якості продукції.

3. Моделювання

На етапі проектування вже використовується тривимірне моделювання продуктів, матеріалів та виробничих процесів, але в майбутньому моделювання ширше використовуватиметься безпосередньо в роботі виробництва. Такі моделі будуть використовувати дані в реальному часі для відображення фізичного світу у віртуальній моделі. Це дозволить операторам тестувати та налаштовувати обладнання ще до подання продукту, скорочуючи час налаштування та підвищуючи якість.

Завдяки розвитку нових обчислювальних методів, таких як нейромережі та хмарні обчислення, розробляються нові алгоритми та підходи до моделювання. Це дозволяє більш точно та оперативно передбачати поведінку об'єкта керування.

4. Горизонтальна та вертикальна системна інтеграція

Більшість сучасних виробництв не повністю інтегровані. Функціям підприємства від рівня цеху до рівня підприємства не вистачає повної інтеграції. Але з Індустрією 4.0 компанії, відділи, функції та можливості стануть набагато згуртованішими, оскільки міжкорпоративні універсальні мережі інтеграції даних розвиваються та забезпечують дійсно автоматизовані ланцюжки створення вартості.

5. Індустріальний інтернет речей (ІІоТ)

На сьогодні лише деякі датчики та обладнання виробників об'єднані в мережу та використовують вбудовані обчислення. Як правило, вони організовані у вигляді вертикальної піраміди автоматизації, в якій датчики та польові пристрої підключаються до системи керування виробничим процесом. Але з появою промислового Інтернету речей все більше пристроїв, будуть обладнані вбудованими системами обчислення та підключені з використанням стандартних технологій. Це дозволить польовим пристроям зв'язуватися і взаємодіяти як один з одним, так і більш централізованими контролерами, якщо це необхідно. Використання такого підходу також зменшить ієрархічну залежність дозволивши децентралізувати аналіз та прийняття рішень.

ІІоТ дозволяє збирати та аналізувати дані в режимі реального часу, що дозволяє виявляти недоліки та оптимізувати виробничі процеси; забезпечує велику гнучкість виробничих процесів, оскільки машини та системи можуть бути переконфігуровані в режимі реального часу для адаптації до вимог, що змінюються. Це призводить до підвищення ефективності, безпеки та якості продукції, а також зниження витрат та часу простою.

6. Кібербезпека

З розширенням можливостей підключення та використання стандартних комунікаційних протоколів, пов'язаних з Індустрією 4.0, потреба у захисті критично важливих промислових систем та виробничих ліній від загроз кібербезпеці різко зростає. В результаті необхідний безпечний та надійний зв'язок, а також складне керування ідентифікацією та доступом до машин та користувачів.

7. Хмарні сервіси

Компанії вже зараз використовують хмарне програмне забезпечення для деяких корпоративних та аналітичних додатків, але з Індустрією 4.0 більше пов'язаних із виробництвом завдань вимагатиме ширшого обміну даними. Разом з тим продуктивність хмарних технологій покращиться, досягнувши часу реакції всього в кілька мілісекунд. В результаті машинні дані та функціональні можливості все частіше розташовуватимуться у хмарі, що дозволить надавати більше послуг на основі даних для виробничих систем.

Хмарні сервіси позбавляють компанії необхідності купувати і підтримувати власну ІТ-інфраструктуру. Вони легко масштабуються, що дозволяє легко збільшувати чи зменшувати обчислювальні потужності залежно від потреб. Це може до значної економії коштів, оскільки компанії можуть платити лише за ті послуги та сховища, які їм потрібні. Хмарні сервіси можуть забезпечувати моніторинг виробничих процесів у режимі реального часу, дозволяючи швидко виявляти та усувати будь-які проблеми.

8. Адитивні технології

Компанії вже почали впроваджувати адитивне виробництво, таке як 3D-друк, яке вони використовують в основному для створення прототипів і виробництва окремих компонентів. В Індустрії 4.0 ці методи адитивного виробництва широко використовуватимуться для виробництва невеликих партій спеціалізованих продуктів, які пропонують переваги конструкції, такі як складні та легкі конструкції. Високопродуктивні децентралізовані системи адитивного виробництва скоротять транспортні відстані та необхідне обладнання на складі.

9. Доповнена реальність

Системи на основі доповненої реальності підтримують різні послуги, такі як вибір запчастин на складі та надсилання інструкцій з ремонту через мобільні пристрої. Ці системи нині перебувають у зародковому стані, але у майбутньому компанії набагато ширше використовувати доповнену реальність. Наприклад надавати працівникам інформацію в режимі реального часу для покращення процесу прийняття рішень та робочих процедур. Робітники можуть отримати

інструкції з ремонту про те, як замінити конкретну деталь, коли вони дивляться на реальну систему, яка потребує ремонту. Ця інформація може відображатись безпосередньо в полі зору працівників за допомогою таких пристроїв, як окуляри доповненої реальності.

Ще один приклад використання технологій доповненої реальності – віртуальне навчання. Компанія Siemens розробила віртуальний модуль навчання операторів заводу для свого програмного забезпечення Comos, в якому використовується реалістичне тривимірне середовище на основі даних з окулярами доповненої реальності для навчання персоналу підприємства діям у надзвичайних ситуаціях. У цьому віртуальному світі оператори можуть навчитися взаємодіяти з машинами. Вони також можуть змінювати параметри та отримувати робочі дані та інструкції з технічного обслуговування [20].

Багато з цих технологій вже зараз застосовуються у виробництві, проте з переходом до Індустрії 4.0 вони повністю змінять традиційні виробничі ланки, об'єднавшись в цілісну повністю інтегровану, автоматизовану та оптимізовану систему.

В свою чергу технологічні впровадження продукують ряд проблем, такі як:

- зростання ризику кібернетичних небезпек та вразливостей, у зв'язку з розширенням інформаційного простору, складності задачі з фізичної ізоляції таких систем;
- необхідність високого рівня кваліфікації керуючого та обслуговуючого персоналу. З однієї сторони підвищений рівень автоматизації на підприємстві зменшує кількість необхідного обслуговуючого персоналу, а з іншої потребує більш професійних кадрів для ефективної роботи;
- проблеми з уніфікацією стандартів. Наявні ризики неузгодженості та порушення управління через відсутність єдиної методології стандартизації та уніфікації використання програмних продуктів та цифрового простору;
- інвестиційні ризики пов'язані з впровадженням інновацій та тривалим часом окупності проектів для Індустрії 4.0 [17].

Індустрія 5.0

Індустрія 5.0 вважається наступним етапом промислової еволюції, її мета полягає у тому, щоб використати творчий потенціал людей-фахівців у співпраці з ефективними, інтелектуальними та точними машинами для отримання ресурсоефективних та кращих для користувачів виробничих рішень порівняно з Індустрією 4.0. Таким чином Індустрія 5.0 являється наступним етапом промислової революції [23] (Рис. 1.4).



Рис. 1.4 Етапи промислової революції

Попередні промислові революції демонструють, що виробничі системи та стратегії постійно змінювалися у бік підвищення продуктивності та ефективності. Індустрія 5.0 принесе значні виклики у сфері взаємодії людини та машини, оскільки вона наблизить машини до повсякденного життя будь-якої людини: інтелектуальні роботи та системи проникнуть у ланцюжки постачання та виробничі цехи на безпрецедентному рівні. Очікується, що Індустрія 5.0 об'єднає високошвидкісні та точні машини та критичне когнітивне мислення людей.

В той час як мета Індустрії 4.0 полягає тільки в поліпшенні процесу для збільшення виробництва, Індустрія 5.0 більше зосереджена на поєднанні людського розуму та креативності, не забуваючи при цьому про стійкість та амбівалентність [24].

Одними з передумов що лягли в основу розробки концепції Індустрії 5.0 є виражені асиметричності в дизайні екосистеми Індустрії 4.0 [24]:

- Екстремальна інтеграція

Взаємозв'язки між всіма підсистемами може бути корисною з практичної точки зору, проте таке зосередження на радикальній інтеграції. Не дивлячись на нові додатки та величезний потенціал Індустрії 4.0, поточні проекти інноваційної екосистеми та їхнє глобальне управління слід поставити під сумнів через ризик монолітної опори на єдину епістемологічну структуру, таку як екстремальна інтеграція за будь-яку ціну. Більше того, екстремальна інтеграція та гіперз'єднання створюють уразливості системного масштабу, які на сьогодні мало розглядалися. Високоінтегровані безперервні/пористі мережі вразливі для системних ризиків, таких як повний колапс мережі у разі виходу з ладу однієї з її частин, наприклад, шляхом злому або інтернет-вірусів, які можуть повністю проникнути в інтегровані системи.

- Інформаційні бульбашки

Інформаційні бульбашки потенційно можуть з'явитися як наслідок екстремальної інтеграції, викликаной Індустрією 4.0. Інформаційні бульбашки належать до ситуацій, коли однакові підходи до знайдення рішень формують вузьку епістемологію, цим самим обмежуючи наступний розвиток науки та техніки. Інструментам Індустрії 4.0 притаманні визначальні риси, що створюють ризик виникнення інформаційних бульбашок, такі як відсутність рефлексії та розуміння того як наші власні цінності впливають на тип висновків які ми робимо; та відсутність розуміння соціальних та пов'язаних з людською взаємодією контекстів.

- Навколишнє середовище

Індустрія 4.0 не приділяє особливої уваги захисту навколишнього середовища та не фокусує увагу на технологіях підвищення екологічної стійкості Землі.

- Орієнтування на соціум

Взаємовплив Індустрії 4.0 та суспільства вивчено відносно недостатньо порівняно з технічними дослідженнями Індустрії 4.0 та Інтернету речей. Існують ряд питань на які Індустрія 4.0 не може дати вичерпної відповіді, зокрема це питання нормативних аспектів – відповідальність, етичність та політика, яка визначатиме глобальне управління всієї системи. Такий стан речей був сформований тому що розробка Індустрії 4.0 була сфокусована на пошуку рішень, які б привели до найбільшої продуктивної ефективності виробництва, а не на орієнтації на кінцевого споживача.

Важливим елементом Індустрії 5.0 є масова персоналізація, коли клієнти можуть віддати перевагу персоналізованим та індивідуальним продуктам відповідно до свого смаку та потреб. Концепція Індустрії 5.0 зараз розроблена таким чином, щоб використовувати унікальний творчий потенціал експертів-людей для спільної роботи з потужними, інтелектуальними та точними машинами Індустрія 5.0 значно підвищить ефективність виробництва та створить гнучку взаємодію між людьми та машинами, забезпечивши відповідальність за взаємодію та постійний моніторинг.

Значну роль у реалізації виробництва персоналізованих продуктів відіграють коботи. Коботи (collaborative robots) - це роботи, спеціально розроблені для безпечної співпраці з людьми. Вони призначені для автоматизації повторюваних, складних або небезпечних процесів виробництва, таких як монтаж, упаковка, зварювання тощо. Коботи можуть працювати поряд зі людьми, не завдаючи їм шкоди та не потребуючи додаткових заходів безпеки, таких як огороження або спеціальний одяг. Вони здатні сприяти підвищенню продуктивності та ефективності виробництва, зменшенню відходів та покращенню якості та індивідуальності продукції.

П'ята промислова революція повертає людей на заводські цехи, щоб об'єднати людину та машину для подальшого використання людського розуму та креативності для підвищення ефективності процесів шляхом поєднання робочих процесів з інтелектуальними системами. У той час як головна турбота Індустрія 4.0 пов'язана з автоматизацією, Індустрія 5.0 буде синергією між

людьми та автономними машинами. Автономна робоча сила буде проникливою та поінформованою про людські наміри та бажання.

Люди працюватимуть разом із роботами, знаючи, що їхні роботизовані колеги добре розуміють їх і мають здатність ефективно з ними співпрацювати. Це призведе до надзвичайно ефективного виробничого процесу створення вартості, процвітання автономії та зменшення відходів і пов'язаних з ними витрат. Індустрія 5.0 змінить визначення слова «робот». Роботи будуть не лише програмованою машиною, яка може працювати повторюваних завдань, але також перетвориться на ідеального помічника людини [26].

1.4 Огляд сучасних підходів та літературних джерел щодо віртуальних інструментів контролю

1.4.1 Функції віртуальних інструментів контролю

Деякі змінні та взаємозв'язки між змінними технологічних об'єктів автоматизації не можуть бути виміряні класичними датчиками. Одним з ефективних варіантів вирішення даної проблеми, яка стала можливою з розвитком Industry 4.0, є використання віртуальних сенсорів (ВС). Оскільки ці змінні часто пов'язані з якістю вихідних даних виробництва, вони є дуже важливими для контролю та управління процесом. З цих причин надання додаткової інформації про ці змінні представляє великий інтерес для оптимізації виробництва.

Розробка ВС стала можливою завдяки новим досягненням в галузі науки про дані, обчислювальних та комунікаційних технологій, статистичних інструментів та методів машинного навчання. Наприклад, об'єднання в мережу масштабних систем дозволило обмінюватися великою кількістю інформації в режимі реального часу. Це дозволило організувати надходження розподілених даних технічного процесу для надання інформації ВС, а в свою чергу вихідні дані з інтелектуальних сенсорів можуть досягати та використовуватися контролерами та моніторами процесу. Нові комунікаційні протоколи та методи допомагають

використовувати пропускну здатність каналу зв'язку і водночас знижують ймовірність втрати пакетів та ймовірність зовнішніх атак.

В той час як ВС призначені для оцінки змінних, які важко або неможливо виміряти фізичними датчиками, ВС забезпечують комплексний аналіз, прогнозування та оптимізацію технологічних процесів, надаючи додаткову обробку отриманим даним, розширюючи функціонал віртуальних інструментів контролю завдяки залученню великих обчислювальних ресурсів. Для досягнення переваг у вигляді більшої точності надійності та адаптивності використовуються велика кількість складних моделей, які потребують значних потужностей. Наприклад, глибока нейронна мережа може визначити та оптимізувати тисячі параметрів, і разом з тим на побудову і валідацію ВА подається колосальний об'єм історичних даних із різнорідних джерел та різною частотою дискретизації. Така обробка стала можливою завдяки впровадженню нових обчислювальних технологій, таких як хмарні обчислення, туманні обчислення, паралельні обчислення та нових технологій роботи з базами даних оптимізованих для роботи з великим об'ємом даних [27].

ВС зосереджені на оцінці змінних процесу та забезпеченні базової інформації для подальшого аналізу. Вони здатні:

- Здійснювати непрямі виміри змінних процесу на основі даних від фізичних датчиків.
- Прогнозувати значення змінних, які не піддаються прямому вимірюванню.
- Діагностувати несправності обладнання шляхом оцінки відхилень у даних.

Крім можливості отримання доступу до комплексної інформації ВА мають також інші переваги. Вони здатні [28, 41]:

- Проводити ситуаційний аналіз та вивчення процесу як об'єкту управління.
- Виявляти взаємозалежності технологічних параметрів.

- Прогнозувати якість вихідного продукту для обраного технологічного режиму.
- Визначати оптимальні параметри для заданих критеріїв якості.
- Координувати взаємопов'язані виробництва.
- Діагностувати обладнання та проводити моніторинг статистичної інформації.
- Знаходити найбільш раціональне рішення з управління технологічним процесом для різних критеріїв ефективності.
- Аналізувати значущість та функції впливу окремих технологічних параметрів.
- Виявляти приховані технологічні загрози та своєчасно сповіщати про них.
- Знаходити нові технологічні режими для різних критеріїв ефективності.

1.4.2 Зміст та особливості функціонування віртуальних інструментів контролю

ВС можуть описуватися широким спектром визначень, зокрема в промисловості можна зустріти їх під назвою «soft-sensors» або м'який датчик чи програмний датчик. Цей термін об'єднує два поняття «програмне забезпечення» та оскільки моделі оцінки сигналів датчиків зазвичай реалізуються в комп'ютерних програмах та «датчики», тому що ці моделі видають інформацію аналогічну апаратним датчикам [42].

Визначення «Віртуальні сенсори» можна виразити як програмний рівень, який забезпечує непрямі виміри змінної процесу або абстрактної умови на основі даних, зібраних фізичними (або іншими віртуальними) датчиками, що використовують функцію об'єднання [43].

Віртуальні аналізатори є програмно-алгоритмічними комплексами, які здійснюють описані вище функціональні задачі на базі ідентифікаційного підходу до побудови і налаштування моделей [28].

Впровадження віртуальних інструментів контролю є одним зі способів перевести виробництво, управління та контроль за його перебігом на якісно

новий рівень. Вже зараз ця технологія набуває вирішального значення для високої якості продукції та продуктивності в будь-якій галузі.

Основна ідея ВС полягає у використанні відносно легкодоступних даних в режимі онлайн для оцінки інших змінних, які в іншому випадку або важко виміряти, або вони вимірюються тільки з низькою частотою.

В якості альтернативи ВС можуть також використовуватися для діагностики несправностей [42]. У більш загальних рисах цю область застосування можна описати як виявлення та відновлення несправності датчика. Як тільки несправний датчик виявлено та ідентифіковано, він може бути реконструйований, або апаратний датчик може бути замінений іншим програмним датчиком, який навчений діяти як резервний програмний датчик апаратного вимірювального пристрою. Якщо резервний датчик виявиться адекватною заміною фізичного датчика, цю ідею можна розвинути ще далі і програмний датчик зможе замінити вимірювальний пристрій і в нормальних робочих умовах. Програмний інструмент легше обслуговувати і він не схильний до механічних пошкоджень, тому така заміна може забезпечити фінансову вигоду для власника процесу [45].

Основними цілями використання ВА є: стабілізація якості продукту за допомогою онлайн-оцінки, зниження споживання енергії та матеріалів за допомогою ефективної роботи, яка відповідає специфікаціям, а також перевірка онлайн-аналізаторів шляхом порівняння з ВА [44].

ВА мають певні особливості, що характерні для їх реалізацій в тому чи іншому вигляді, та виділяють їх серед звичайних датчиків та керуючих систем.

Головною такою особливістю ВА є здатність до навчання, яка виступає альтернативою отримання інформації шляхом лабораторних досліджень чи безпосередніх вимірювань. Інша особливість полягає в поєднанні оперативної інформації з тією що надається базою знань підприємства Використовуючи ці дані ВА в режимі реального часу здатні проводити аналіз [28].

Віртуальний датчик поєднує в собі вимірювальну частину (датчик) та програмний алгоритм оцінки для оперативного моніторингу. Датчик надає дані

безпосередньо вимірюваних змінних чи складних багатопараметричних вимірювань. На основі даних датчика програмна частина прогнозує змінні процесу, які не можуть бути виміряними напряму. ВС здатні об'єднувати та обробляти дані від кількох датчиків, вимірюючи різноманітні змінні процесу.

В літературі описано два типи віртуальних інструментів: ті, які управляються моделлю та ті, що керуються даними.

Ті, що управляються моделями, також називають моделями білого ящика, тому що вони мають повне феноменологічне знання про передумови процесу. На відміну від цього, моделі, керовані даними, називаються методами чорного ящика, тому що сама модель не має відомостей про процес і заснована на емпіричних спостереженнях за процесом. Між цими двома крайнощами можлива безліч комбінацій цих двох основних типів моделей. Типовим прикладом такої комбінації є керований моделлю «м'який сенсор», який використовує метод, керований даними, для моделювання дробів, які можуть бути легко змодельовані в термінах феноменологічних моделей. Ці моделі іноді називають гібридними моделями, або моделями сірого ящика.

ВС, що керуються даними (DDM) основані на хеометричних моделях, основаних на архівних даних процесу для прогнозування інших змінних процесу в режимі реального часу. Цей тип датчика використовує хеометричні методи, такі як регресія частково найменших квадратів (partial least-squares), метод головних компонент (principal component analysis) та нейронні мережі для опису внутрішніх взаємозв'язків між виміряними даними та різними змінними процесу.

ВС, що управляються моделями в основному базуються на балансах маси та енергії. Ці моделі описують принципи та основи фізичних та хімічних процесів, наприклад з використанням принципів збереження маси чи енергетичних балансів [50]. Ці моделі розробляються в першу чергу для планування та проектування підприємств. Тому вони зазвичай зосереджені на описі ідеальних стаціонарних станів процесу, що являється одним з їх недоліків, що ускладнює створення на їх основі м'яких датчиків [45].

Моделі, керовані моделями (MDM), або, точніше, FPM, у першу чергу розробляються з метою планування та розробки технологічних установок. Ці моделі засновані на рівняннях, описуючи хімічні та фізичні принципи, що лежать в основі процесу. Типовим прикладом являється використання принципу збереження маси, екзотермічного рівняння, енергетичних балансів, кінетики реакції у вигляді рівняння швидкості реакцій. Недолік цього типу моделей полягає в тому, що їх розробка вимагає більшого обсягу експертних знань про процеси. Ці знання не завжди доступні. Наприклад, для біохімічного процесу часто недостатньо феноменологічних знань для точного опису протікаючих процесів. Друга проблема полягає в тому, що моделі часто описують узагальнену теоретичну основу процесу, а не реальні умови, на які впливає безліч факторів, що виходять за рамки МДМ. Крім того, моделі, керовані моделями, зазвичай фокусуються на описі оптимального стаціонарного стану процесу і, таким чином, не підходять для опису будь-якого перехідного стану [45].

Тем не менше, програмні датчики, керовані моделі, популярні в якості підтримки логічного управління.

Оскільки моделі, що управляються даними, засновані на даних, вимірних безпосередньо на підприємствах, і, таким чином, описують реальні умови процесу, порівняно з програмними датчиками що керуються моделями, вони більшою мірою пов'язані з реальністю і краще описують справжні умови процесу. Ці моделі ґрунтуються на реальних вимірах, які записуються, зберігаються та надаються системі у вигляді історичних даних. Самі моделі є емпіричними методами прогнозування, такими як регресія основних компонентів (PCR), багатошаровий перцептрон (MLP),

Незалежно від принципу керування роботою віртуальних інструментів, в основі їх функціонування всіх лежать математичні моделі. Математичні моделі використовують наявні в них механізми для опису будь-якого реального процесу. Це дозволяє їм оцінювати різноманітні критерії процесу не втручаючись в його роботу, наприклад оцінити ефективність нового алгоритму. Алгоритми керування, протестовані на такій моделі, можуть бути реалізованими

безпосередньо в реальному процесі. Такий підхід дозволяє не порушуючи роботу реального процесу оцінювати налаштування контролера, наслідки збурень та оптимізувати роботу установки. Таким чином динамічні математичні моделі можуть виступати ефективними та потужними інструментами для тестування нових схем управління [51].

ВА можуть як виконувати роль інтелектуальної надбудови контуру управління, так і входити в склад автоматизованої системи управління технологічним процесом (АСУ ТП) (Рис. 1.5) [41].

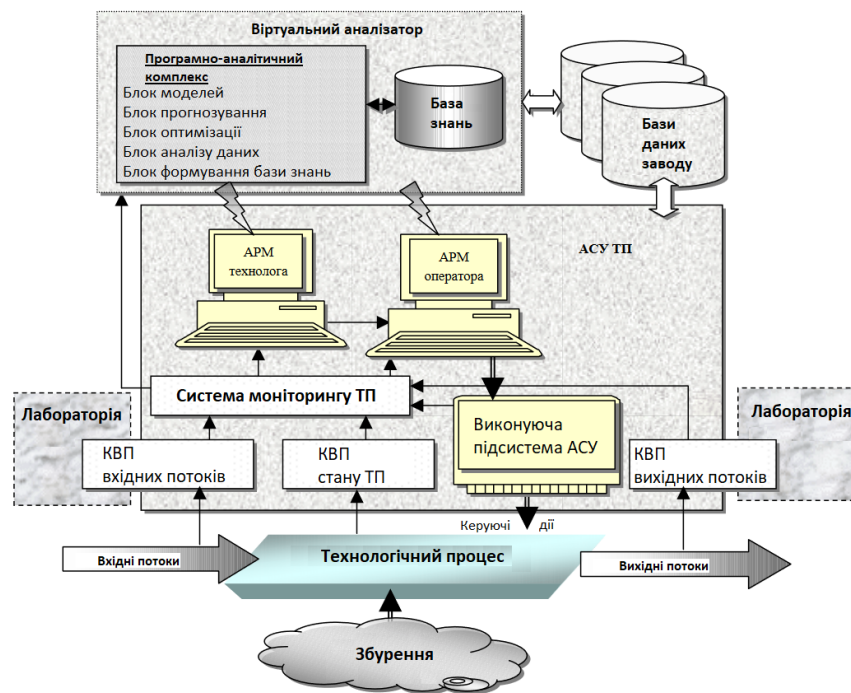


Рис. 1.5 Схема взаємодії ВА з АСУ ТП

Основними джерелами інформації для віртуального аналізу виступають:

- прихована надмірність, що міститься в фізико-хімічних вимірах існуючих систем контрольно-вимірювальних приладів (КВП) і результатах роботи заводської лабораторії;
- ретроспективні технологічні знання, накопичені в процесі управління ТП і збережені в цеховій базі даних (БД).

В основі функціонування ВС, як елемента системи моніторингу ТП, лежить математична модель, що відображає взаємозв'язок вхідних і вихідних параметрів ТП.

В якості вихідної структури для аналізу ТП використовується традиційна кібернетична модель «чорного ящика». Відповідно до методології SADT (Structured Analysis and Design Technique), використовуваної для формування та подання структурних моделей, «чорний ящик» ТП буде мати вигляд, представлений на Рис. 1.6.

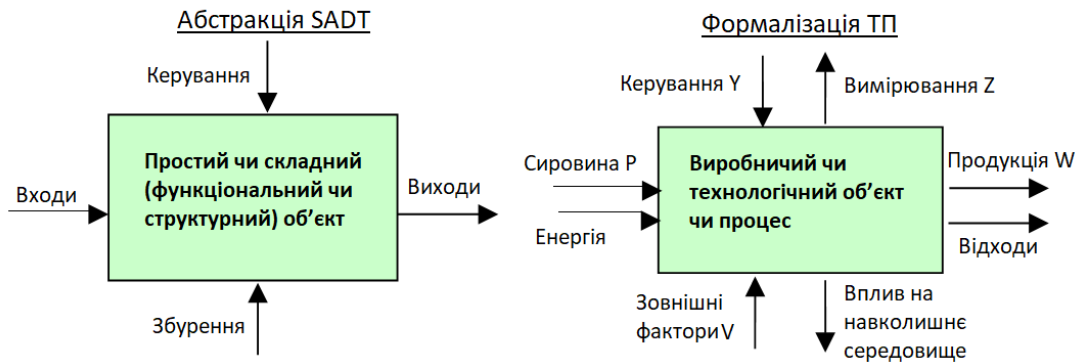


Рис. 1.6 Структурна схема формалізації ТП

Традиційного для кібернетики опис досліджуваного об'єкта O задається у вигляді тріади множин: $O = \langle U, X, Y \rangle$, де $U = \{P_1, P_2, \dots, P_\alpha, Y_1, Y_2, \dots, Y_\chi, V_1, V_2, \dots, V_\beta\}$ – множина вхідних впливів, $X = \{X_1, X_2, \dots, X_m\}$ – множина станів ТП, $Y = \{Z_1, Z_2, \dots, Z_\delta, W_1, W_2, \dots, W_\gamma\}$ – множина вихідних процесів (інформаційних $\{Z\}$ і матеріальних $\{W\}$).

У завданнях синтезу найважливішу роль грає саме структура системи, яку і потрібно визначити. У зв'язку з цим узагальнена модель об'єкта ускладнюється і набуває виду: $O = \langle A, R, U, X, Y, G \rangle$, де A – безліч елементів системи; R – матриця відносин між елементами системи, заданими на A ; G – матриця відносин між множинами A і X (відносини емерджентності). Складова частина системи $\{A, R\}$ називається її структурою, а $\{U, X, Y\}$ – програмою її функціонування. Таким чином, система являє собою об'єднання цих двох складових, а її єдність визначається відношенням емерджентності G [41].

Таким чином, ВС можна використовувати в якості альтернативи та доповнення до стандартних апаратних датчиків, що дозволяє розширити їх функціонал шляхом додавання нових способів визначення та виміру змінних.

1.4.3 Синтез та розробка віртуальних сенсорів

Рівень техніки такого класу аналітичних пристроїв напряму залежить від наявності та можливостей двох основних компонентів – вимірювальних пристроїв та математичної моделі процесів, які при успішному об'єднанні в один функціональний об'єкт, складають програмний сенсор. Тому конфігурація програмної частини ВС має важливу роль для ефективної роботи.

Методологія розробки м'яких сенсорів складається з типових кроків [45]:

- Первинна перевірка даних
- Вибір історичних даних та ідентифікація стаціонарних станів
- Попередня обробка даних
- Вибір моделі, навчання та перевірка
- Обслуговування ВС

1) Первинна перевірка даних

На цьому початковому етапі відбувається перша перевірка даних. Метою цього кроку є отримання загального уявлення про структуру даних та виявлення будь-яких очевидних проблем, які можна вирішити на цьому початковому етапі. Наступною метою цього етапу є оцінка вимог до складності моделі. Прийняття рішення про використання того чи іншого сімейства математичних моделей для створення ВС на цьому етапі може бути невірним, тому моделі та їх характеристики завжди варто оцінювати та порівнювати з альтернативними моделями на більш пізніх етапах розробки.

2) Вибір історичних даних та ідентифікація стаціонарних станів

На цьому етапі обираються дані, які будуть використовуватися для навчання та оцінки моделі. Далі використовуючи ці дані ідентифікуються та обираються стаціонарні їх частини, зазвичай шляхом ручної анотації цих даних.

3) Попередня обробка даних

Ціллю попередньої обробки даних є перетворення даних таким чином, щоб їх можна було більш ефективно обробляти реальною моделлю. Прикладом типової попередньої обробки є нормалізація даних до нульового середнього і одиничній дисперсії. Для складних процесів необхідно проводити декілька циклів попередньої обробки. Зазвичай кроки обробки складаються з таких етапів

як обробка відсутніх даних, виявлення та заміна відхилень, вибір відповідних змінних-ознак, та виявлення затримок між конкретними змінними. Багато з цих кроків в даний час виконуються вручну, або вимагають контрольованої перевірки результатів.

Проблема пропущених значень дуже важлива для розуміння, щоб ефективно управляти даними. Ця проблема виникає, коли для однієї чи кількох змінних у вибірці не зберігається значення. Існують різні підходи до роботи з відсутніми даними. Перший підхід полягає у видаленні вибірок, що містять нульові значення. Цей підхід рекомендується лише тоді, коли кількість пропущених значень невелика, інакше це не дасть очікуваних результатів під час прогнозування висновку. Інший підхід пропонує заповнити пропущені значення за допомогою деяких методів, наприклад обчислити середнє значення, медіану або моду ознаки та замінити ними пропущені значення. Ці два підходи досить поширені для вирішення проблеми пропущених значень [71].

Значення датчиків які відхиляються від типового чи значимого діапазону даних називаються відхиленнями даних чи викидами. Вони можуть бути спричиненими, наприклад, апаратними збоями, помилками зв'язку некоректними зчитуваннями чи умовами роботи. Викиди можуть вплинути на продуктивність ВС. Для того, щоб пом'якшити вплив викидів спочатку їх необхідно виявити, а потім виправити. Існують різноманітні методи виявлення викидів, такі як метод виявлення відхилень 3σ чи ідентифікатор Хампеля, який являється надійною версією методу 3σ [82]. Дані методи засновані на одномірному виявленні викидів. Також для даної мети можуть використовуватися багатомірні методи, які використовуються в контексті м'яких датчиків, та основані на проекції даних, такі як параметри Джолліффа з РСА (principal component analysis) та PLS [83, 84]. В літературі описані різноманітні методи нормалізації, такі як нормалізація min-max нормалізація, нормалізація Z-показника, та нормалізація нульового середнього [83, 85]. Попередня обробка даних - це крок, що вимагає великого обсягу ручної роботи та експертних знань про процес, що лежить в основі [71].

Попередня обробка даних зазвичай виконується ітеративно, після стандартизації та обробки відсутніх значень, які зазвичай виконуються лише один раз, повторно застосовуються видалення відхилень та вибір ознак, поки розробник моделі не визнає дані готовими до використання для навчання та оцінки реальної моделі. Зараз попередня обробка даних є етапом, що вимагає великого обсягу ручної роботи та експертних знань про процес, що лежить в основі.

4) Вибір моделі, навчання та перевірка

Оскільки модель являється основним інструментом ВС, вибір оптимального типу має вирішальне рішення для продуктивності м'якого датчика. Наразі немає єдиного теоретичного підходу до цієї задачі, тому тип моделі та її параметри часто підбираються спеціальним чином для кожного ВС. Цей вибір опирається на особисті вподобання розробника та залежить від минулого досвіду, що в свою чергу може негативно впливати на роботу м'якого датчика.

Багато зразків і складний характер методів моделювання м'якого сприйняття вимагають дуже тривалого навчання. Таким чином, зазвичай використовується поділ вибіркового даних на навчальну та валідаційну або навчальну та тестову вибірки.

Після знаходження оптимальної структури моделі та навчання моделі навчений ВС необхідно ще раз оцінити на незалежних даних. Існує кілька інструментів з метою оцінки продуктивності моделі. У разі числової оцінки продуктивності найбільш популярною є середньоквадратична помилка (MSE), яка вимірює середньоквадратичну різницю між прогнозованим та правильним значенням. Інший спосіб оцінки ефективності є використання візуального подання прогнозів. Недоліком цього метода є залежність від рішення розробника та прийняття людиною суб'єктивного рішення про адекватність моделі.

5) Обслуговування ВА

Після розробки та впровадження ВС його потрібно регулярно підтримувати та налаштовувати. Технічне обслуговування необхідне через дрейф і інші зміни

даних, які спричиняють погіршення характеристик м'якого датчика та повинні бути компенсовані шляхом адаптації або повторної розробки моделі.

Нині більшість ВС не мають жодних автоматизованих механізмів для їх обслуговування. Цей факт разом з постійним дрейфом даних, призводить до необхідності ручного контролю якості та обслуговування м'яких датчиків, що є значним фактором підвищення вартості для застосування ВС. Крім того, часто не існує об'єктивного підходу для оцінки рівня якості м'якого датчика, і рішення про те, добре працює модель чи ні, залежить від суб'єктивного сприйняття оператора моделі, що базується на візуальній інтерпретації відхилення між правильним цільовим значенням та значенням його прогнозу [45].

1.4.4 Моделі віртуальних сенсорів

Для реалізації оптимізаційних функцій у ВС використовується широкий спектр методів аналізу даних та теорії керування. Особливе місце відведено новим комп'ютерним технологіям пов'язаним з парадигмою інтелектуального аналізу даних (Data Mining), що включають в себе еволюційні, нейромережеві, генетичні та інші методи досліджень.

Звичайні методи моделювання м'яких датчиків на основі даних в основному включають широкий спектр методів статистичного виводу та методів машинного навчання, таких як регресія головних компонентів, яка поєднує аналіз головних компонентів з регресійною моделлю, регресію з частково найменшими квадратами (PLS), метод опорних векторів (SVM) та штучною нейронною мережею (ІНС).

При застосуванні інтелектуальних методів для обробки даних та результатів перебігу хімічних процесів доцільно використовувати хемометричний підхід до аналізу даних, який полягає в отриманні хімічних даних за допомогою математичних методів та методів Data mining. Дані, які отримує ВС мають бути співвіднесені з іншими важливими змінними процесу такими як фактичний стан процесу в математичній моделі. Після цього модель можна використовувати для прогнозування змінних чи стану процесу на основі оперативних даних. Для

подібних кореляцій необхідно використовувати багатомірний аналіз даних, оскільки ревалентна інформація розподіляється на весь набір даних. Багатомірний аналіз даних являється частиною хеометрики.

Серед основних методів синтезу ВС можна виділити такі [28, 71]:

Моделі основані на підтримці векторних машинних моделей (SVM)

Найбільш часто використовувані методи моделювання водіння в основному включають методи моделювання на основі SVM. Це контрольований алгоритм машинного навчання, який використовується для задач регресії, класифікації та прогнозування часових рядів. Алгоритм SVM базується на концепції мінімізації структурного ризику. Це метод на основі ядра, який дозволяє використовувати лінійну, поліноміальну та радіально-базисну функціональність (RBF), а також інші, які відповідають умові Мерсера. В основі алгоритму SVM лежить переклад вихідних даних у просторі більш високого розміру, де кожен вхідний вектор формується як послідовність вихідних точок із тимчасовими затримками. Далі ці вектори використовуються в якості опорних зразків, з комбінації яких вираховується регресійна гіперплощина, що описує розподіл даних із заданою точністю. Обчислення являють собою підсумовування за всіма зразками так званих ядер - однакових функцій від вхідних даних. Ці функції можуть бути лінійними або нелінійними, і керуватися параметрами, що впливають на точність регресії. SVM має характеристики які здатні повністю апроксимувати довільно складні нелінійні системи, а також має здатність до самонавчання та адаптації до динамічних характеристик невизначених систем.

У SVM надається навчальний набір даних:

$$d = (x_i, y_i), i = 1, \dots, n, \quad (1.1)$$

де n вказує на кількість вибірок. $x_i \in R^D$ – багатовимірні входи, $y_i \in R$ – безперервні вихідні дані. Ми прагнемо знайти неперервну функцію відображення $f : R^N \rightarrow R$, яка найкраще прогнозує набір точок навчання за допомогою функції $y = f(x)$.

До моделей ВС що базуються на основі SVM можна віднести:

- Метод найменших квадратів опорних векторів (Least-Squares SVM (LS-SVM)).

Він дозволяє замість вихідної нелінійної задачі вирішити еквівалентну як системи лінійних рівнянь.

- Методи з оптимізацією методом рою частинок або particle swarm optimization (PSO-SVM).

В PSO група частинок виконує процес пошуку у просторі рішень. Кожна частка після зміщення в просторі рішень ділиться своєю найкращою позицією з однією або декількома частинками рою. У результаті частка визначає свій наступний крок у просторі рішень, ґрунтуючись на пройденому шляху кожної частинки. Як тільки всі частинки оновлять свої позиції на поточній ітерації, починається наступна ітерація процесу пошуку, освоюючи області, близькі до оптимуму. Зрештою, весь рій наближається до оптимуму цільової функції зі швидкістю збіжності, яка сильно залежить від вибраного варіанта PSO та значень параметрів.

- Покращена версія PSO - improved particle swarm optimization (IPSO-SVM).

Хоча PSO має високу швидкість збіжності та високу універсальність, він має недоліки, такі як легка передчасна збіжність, низька точність пошуку та низька ефективність ітерації. Щоб подолати ці недоліки було введено операцію мутації, тобто повторну ініціалізацію деяких змінних з певною ймовірністю. Операція мутації розширює постійно скорочуваний простір пошуку популяції в ітерації, так що частинки можуть вискочити з попередньо шуканої позиції оптимального значення, і пошук можна здійснювати у більшому просторі, зберігаючи різноманітність популяції, а також покращуючи алгоритм, щоб знайти можливість кращої позиції. Тому вводиться простий оператор мутації на основі звичайного алгоритму PSO, і частинки будуть повторно ініціалізуватися з певною ймовірністю після кожного оновлення частинок [86].

- Метод моделювання м'якого вимірювання, заснований на опорній векторній машині найменших квадратів із кількома вихідними змінними (MLS-SVM).

Такий м'який датчик заснований на поєднанні інверсної системи та теорії SVM [87].

Нечітка логіка

Система нечіткої логіки є унікальною тим, що вона здатна одночасно обробляти числові дані та лінгвістичні знання. Таким чином алгоритми нечіткої логіки здатні поєднувати об'єктивні числові дані з суб'єктивними лінгвістичними для визначення необхідних рішень.

Загалом система нечіткої логіки являє собою нелінійне перетворення вектора вхідних даних в скалярний вихід. При цьому існує велика кількість можливостей які приводять до безлічі різноманітних результатів. Виходячи з алгоритму функціонування нечіткої логіки її можна розглядати як розширення багатозначної логіки. При цьому їх використання та цілі є абсолютно різними.

Таким чином на відміну від класичних логічних систем, нечітка логіка має справу з приблизними, а не точними моделями роздумів, в рамках яких все, включаючи істину, залежить від степенів.

Основними особливостями нечіткої логіки, які відрізняють її від традиційних логічних систем є [88]:

- Наявність проміжних значень істинності.
- Нечітка логіка дозволяє використовувати нечіткі квантори, які дозволяють їм маніпулювати вірогідностями в рамках нечіткої логіки.
- Використання системи обчислень що базується на лінгвістичних змінних, тобто змінних, значеннями яких являються слова, чи речення на природній чи синтетичній мові.

Структура системи нечіткої логіки складається з 5 блоків:

1. Бази правил - містить нечіткі правила.
2. База даних - визначає функції належності нечітких множин.
3. Блок прийняття рішень – виконує операції згідно з правилами.

4. Блок фазифікації – перетворює чіткі множини в нечіткі.

5. Блок дефазифікації – перетворює нечіткі множини в чіткі.

Робота системи нечіткої логіки складається з кількох етапів. Блок фазифікації перетворює чіткі входні дані в нечіткі. На основі цих даних та правил формується база знань, яка пов'язує входні дані з вихідними. Блок дефазифікації перетворює нечіткі вихідні дані в чіткі.

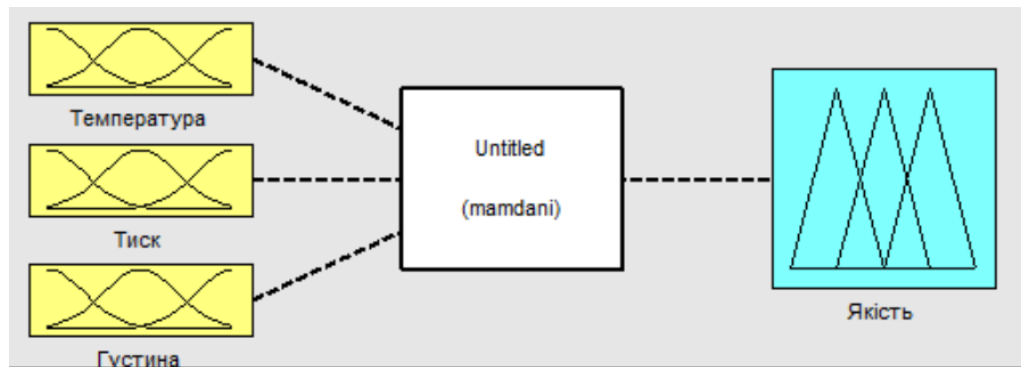


Рис. 1.7 Приклад роботи системи нечіткої логіки. Налаштування структури

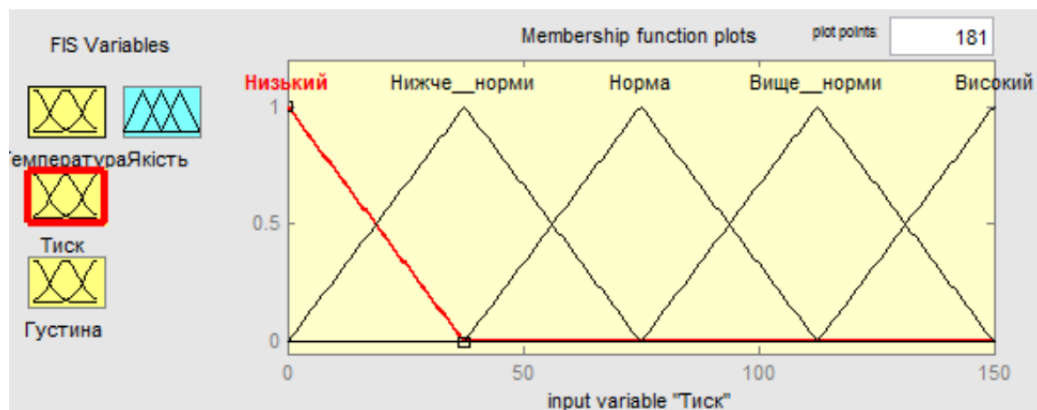


Рис. 1.8 Приклад роботи системи нечіткої логіки. Налаштування правил

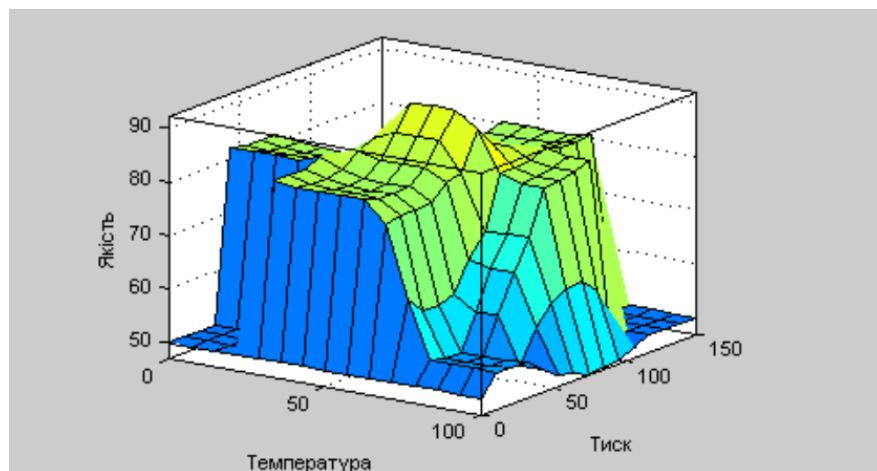


Рис. 1.9 Приклад роботи системи нечіткої логіки. Формування чітких значень вихідної величини алгоритмами системи нечіткої логіки

Використання систем нечіткої логіки має певні обмеження, які пов'язані з такими факторами [28]:

- Відсутність стандартної методики розробки нечітких систем.
- Неможливість математичного аналізу нечітких систем традиційними методами, що обмежує їх використання на неперервних виробництвах.
- Використання нечіткого підходу в порівнянні з вірогіднісним практично не приводить до підвищення точності обчислень.

Нейронні мережі

Більшість сучасних ВС синтезовані на основі нейромережових технологій.

Нейронна мережа складається з вхідного та вихідного рівнів, а також (у більшості випадків) прихованого шару, що складається з одиниць, які перетворюють вхідні дані на те, що може використовувати вихідний рівень. Вони є чудовими інструментами для регресії, класифікації, кластеризації даних, оптимізації та пошуку шаблонів, які є надто складними або численними, щоб програміст-людина міг їх витягнути та навчити машину ідентифікувати.

Основні властивості штучних нейронних мереж – паралельна обробка даних, висока швидкодія, можливість навчання та роботи в умовах сильних шумів та недостатку інформації.

Нейронна мережа являє собою модель фрагменту мозку, оснований на наявних знаннях та представленнях про окремі елементи мозку живих істот. Штучний нейрон являє собою нелінійний пороговий суматор, що має один вихід та безліч входів. Функція, що задає нелінійність перетворення результату підсумовування в штучному нейроні, називається функцією активації нейрона. Функція активації нейрона являє собою нелінійну обмежену монотонно-зростаючу криву.

Штучна нейронна мережа, чи artificial neural network (ANN) являє собою сукупність штучних нейронів, зв'язаних між собою в певному порядку. Набір вагових коефіцієнтів(величин що виражають силу зв'язку між штучними нейронами) являється унікальним та являє собою вміст пам'яті нейронної мережі. Порядок та структура зв'язків у нейронній мережі визначає її топологію.

Механізм процесу навчання нейронної мережі полягає в адаптації вагових коефіцієнтів. Одним з критеріїв навчання являється асимптотичне зменшення середньоквадратичної похибки з кожним новим повтором тестового прикладу. При наявності достатньої кількості нейронів в прихованому шарі і великого набору вхідних та вихідних даних для навчання, нейронні мережі можуть скільки завгодно точно апроксимувати будь-яку неперервну функцію [89].

Завдячуючи своїм універсальним апроксимуючим властивостям, штучні нейронні мережі являють собою потужний інструмент для вирішення задачі ідентифікації нелінійних статичних та динамічних об'єктів керування. Основані на штучних нейронних мережах дискретні ідентифікаційні моделі мають назву нейроемулятори або предиктори.

Навчена нейронна мережа має здатність обробляти нову інформацію інтерпретуючи та узагальнюючи її. Разом з цим нейронні мережі здатні успішно обробляти зашумлену, спотворену та частково пошкоджену інформацію. Проте неможливість прогнозування поведінки нейронної мережі без запуску в роботу робить небезпечним її функціонування на реальних об'єктах в реальному часі [28].

Моделі м'яких датчиків на основі глибокого навчання

Різноманітні статистичні методи та методи машинного навчання ефективно використовуються для керованого даними моделювання ВА, наприклад SVM, ANN, PCR та PLS тощо. SVM виявився ефективним для вирішення проблем із невеликими наборами даних, тоді як для великих навчальних вибірок вартість обчислень становиться надмірною. ANN це техніка машинного навчання, широко застосовувана в розпізнаванні образів, які мають нелінійну структуру та може апроксимувати нелінійну неперервну функцію з довільною точністю, і тому підходить для нелінійного промислового процесу, але проблема полягає в тому, що традиційний метод градієнтного спуску повільно зближуються, а шум у промислових даних часто викликає локальні оптимуми для ANN [71].

Останніми роками моделі, засновані на глибокому навчанні, стають все більш помітними в багатьох областях, таких як комп'ютерне бачення,

розпізнавання мови, класифікація зображень, біоінформатика, обробка природної мови (NLP) і ряді інших областей [90, 91]. Глибоке навчання, підмножина машинного навчання, здатне ефективно вивчати глибокі ієрархічні штучні нейронні мережі. Порівняно з традиційними методами, такими як SVM та ANN, глибоке навчання має більшу глибину рівнів, метод попередньої конвергенції та сильнішу здатність до наближення [92].

Генетичні алгоритми

Генетичні алгоритми (ГА) являють собою методи розв'язання оптимізаційних задач, що імітують процеси еволюційного розвитку в живій природі на основі механізмів генетичної спадковості, мінливості та природного відбору.

ГА один із перших стохастичних алгоритмів на основі популяції. Основними операторами в генетичному алгоритмі являються відбір, схрещення та мутації.

ГА - це алгоритм що оснований на популяції. Кожне рішення відповідає хромосомі, а кожен параметр представляє ген. ГА оцінює пристосовуваність кожної індивідуальності в популяції за допомогою цільової функції пристосовуваності. Для покращення поганих рішень вибираються рішення з найбільшою придатністю. Відбувається процес збереження найкращих рішень в кожному поколінні та використання їх для покращення інших рішень. Таким чином з кожним новим поколінням популяція стає все кращою.

Мутації є однією з ключових особливостей ГА. Оператор мутації випадковим чином змінює гени в хромосомах, що підвищує різноманіття та вибірку для вивчення. Як і в природі оператор мутації може привести до значно кращого рішення і привести інші рішення до глобального оптимуму [93].

Методи ймовірнісного моделювання латентних змінних

Через високу розмірність даних в промисловості завжди потрібно зменшувати розмірність, інакше аналіз даних може бути досить складним. Як рішення, для використання у м'яких сенсорах широко застосовувалися численні методи керованого даними регресійного моделювання, засновані на латентних

змінних, таких як частковий найменший квадрат (PLS) і принципова компонентна регресія (PCR) [94, 95].

Проектуючи дані процесу в простір нижчого виміру з простору вищого виміру, методи моделювання прихованих змінних зможуть витягти важливу інформацію з промислових даних. Сьогодні багато хто перебудовує моделі латентних змінних через імовірнісні рамки дослідження, наприклад, ймовірнісна PCA та ймовірнісна PCR [96, 97].

Імовірнісна модель даних має кілька додаткових переваг порівняно з традиційними методами. По-перше, алгоритм максимізації очікування можна використовувати для оцінки параметрів імовірнісних моделей. По-друге, імовірнісна модель може вирішити проблему відсутніх значень у практичному наборі даних. По-третє, єдина структура імовірнісного моделювання може бути прямо розширена до форми суміші, яку можна використовувати для більш складних випадків. Крім того, імовірнісне моделювання може сформулювати методи баєсівської регуляризації, які можна використовувати для автоматичного визначення розмірності моделі латентної змінної. Методи моделювання латентних змінних використовуються для дискримінантного аналізу, кластеризації, моніторингу даних процесу, регресійного моделювання та класифікації [71].

Гібридні технології

Одна із тенденцій сьогодення є розвиток гібридних технологій які являють собою поєднання нейронних мереж з іншими методами. Такі гібридні технології дозволяють використовувати сильні сторони кількох підходів для отримання ефективного системного моделювання та моніторингу.

Гібридизація використовує хороші властивості комбінованих методів, що часто призводить до точніших моделей процесів. До прикладів гібридних моделей можна віднести:

- Нелінійний PCA (principal component analysis, PCA).

Основним недоліком PCA є його нездатність моделювати нелінійні відносини між змінними. Нелінійна модель ПЛР для оцінки багатовимірних

змінних процесу може бути отримана шляхом застосування нелінійної регресії до прихованих змінних, отриманих шляхом мінімізації критерію [98, 99].

- Нелінійний аналіз головних компонент (nonlinear principal component analysis, NLPCA).

NLPCA зазвичай розглядається як нелінійне узагальнення стандартного аналізу головних компонент (PCA). Він узагальнює головні компоненти від прямих до кривих (нелінійних). Таким чином, підпростір у вихідному просторі даних, який описується всіма нелінійними компонентами, також є викривленим. Нелінійний PCA може бути досягнутий за допомогою нейронної мережі з автоасоціативною архітектурою, також відомою як автоенкодер, мережа реплікатора, мережа вузького місця або мережа типу пісочного годинника. Така автоасоціативна нейронна мережа є багат шаровим персептроном, який виконує відображення ідентичності, а це означає, що вихід мережі повинен бути ідентичним входу. Проте в середині мережі знаходиться шар, який працює як вузьке місце, в якому примусово зменшується розмірність даних. Цей шар вузького місця забезпечує бажані значення компонентів (оцінки).

- Багатомодельні методи

Ансамбль моделей може мати кращі можливості прогнозування порівняно з одиничною моделлю [100]. У методі множинних моделей (ММ) дані діляться різні піддомени, і кожного домену будуються локальні підмоделі. Таким чином, замість використання однієї глобальної моделі розробляється ряд простіших моделей, які потім об'єднуються для отримання прогнозу або стану процесу. Це часто призводить до більш точного ВС при моделюванні нелінійних процесів, оскільки в просторі даних процесу можна розпізнати різні кластери, які відповідають різним робочим станам процесу.

Одним із методів ММ, який можна використовувати для моніторингу процесів, є Multiscale PCA (MSPCA) [101]. Цей метод поєднує вейвлет-аналіз та PCA. В той час, як PCA декорелює вимірювання, вейвлет-аналіз витягує детерміновані ознаки та декорелює приблизно автокорельовані вимірювання. Точніше, в MSPCA вимірювання розкладаються з допомогою вейвлет-аналізу.

Отримані коефіцієнти на кожній шкалі використовуються для розробки моделі PCA для кожної шкали незалежно. Отримані підмоделі потім об'єднуються. Таким чином, MSPCA може бути ефективнішим при моніторингу процесів, ніж класичний (одномасштабний) PCA [99].

Model Reinforcement Learning

Традиційні методи управління, засновані на фіксованих алгоритмах або моделях із заздалегідь визначеними параметрами, часто стикаються з обмеженнями в умовах мінливості сировини, зміни зовнішніх факторів чи непередбачуваних відхилень у процесі. У цьому контексті виникає потреба в інноваційних підходах, які здатні динамічно адаптуватися до складних і нелінійних умов виробництва.

Одним із перспективних напрямів для вирішення цих проблем є використання моделей навчання з підкріпленням (Reinforcement Learning, RL), які дозволяють системі самостійно навчатися оптимальним стратегіям управління через взаємодію з технологічним середовищем. RL системи мають великий потенціал для застосування в промисловості, оскільки вони здатні адаптуватися до складних і змінних умов, оптимізуючи управлінські процеси в реальному часі. Вони дозволяють автоматизувати прийняття рішень на основі аналізу даних, що надходять від сенсорів та систем моніторингу. RL-системи можуть оптимізувати налаштування процесу, знижуючи витрати енергії, збільшуючи продуктивність і покращуючи якість кінцевого продукту. Це допомагає зменшити вплив людського фактору, мінімізувати помилки та забезпечити більш стабільну і ефективну роботу. Завдяки принципу навчання через винагороду, RL-системи здатні постійно вдосконалювати свої стратегії управління, адаптуючись до змінних умов, таких як варіації в якості сировини чи зміни в роботі обладнання. Крім того, вони можуть прогнозувати майбутні стани процесу, що дозволяє приймати рішення з урахуванням довгострокових перспектив. Це особливо важливо в складних виробничих процесах, де необхідно враховувати не лише поточний стан, але й можливі майбутні зміни. RL-системи також здатні масштабуватися, адаптуючись до різних типів

виробничих процесів, що робить їх гнучкими і зручними для використання в різних галузях. Завдяки своїй здатності до оптимізації та автоматизації, RL-системи можуть значно зменшити витрати на енергію, ресурси та матеріали, підвищуючи загальну ефективність виробничих процесів, особливо в умовах невизначеності.

Найбільш популярні агенти для задач з неперервним простором дій, які характерні об'єктам спиртового виробництва наведені в таблиці 1.1. Кожен з цих алгоритмів має свої сильні та слабкі сторони в залежності від специфіки завдання та ресурсів для тренування [102-107].

Таблиця 1.1. Порівняння методів RL-агентів

Агент	Алгоритм	Тип простору дій	Алгоритм оновлення	Переваги	Недоліки
DDPG (Deep Deterministic Policy Gradient)	Середній за описом агент	Неперервний	Оновлення політики за допомогою актор-критика (Actor-Critic)	Підходить для високих вимог до простору дій; добре масштабується для складних завдань	Може мати проблеми з конвергенцією в деяких задачах; потребує значних обчислювальних ресурсів
TD3 (Twin Delayed DDPG)	Покращена версія DDPG	Неперервний	Оновлення політики за допомогою актор-критика (Actor-Critic)	Менший ризик перенавчання та нестабільності в порівнянні з DDPG; надійніший у складних задачах	Такі ж самі проблеми, як і в DDPG, але більш стабільніший
SAC (Soft Actor-Critic)	Активне максимізування ентропії	Неперервний	Оновлення політики з ентропією (Soft Actor-Critic)	Покращена стабільність і ефективність; краще працює з випадковими переходами між станами	Потребує великих обчислювальних витрат для тренування
PPO (Proximal Policy Optimization)	Без моделей	Неперервний/Дискретний	Підхід оновлення політики, що забезпечує стабільність і баланс	Легше налаштовується, ніж DDPG; ефективний в реальних завданнях з високими вимогами	Може мати проблеми при високому розмірі простору дій

A3C (Asynchronous Advantage Actor- Critic)	Актор-Критик	Неперервний/Дискретний	Оновлення актор-критиків із асинхронним навчанням	Ефективний для масштабованих задач з кількома агентами	Може вимагати складного налаштування та високих обчислювальних ресурсів

RL-моделі, базуючись на принципах проб і помилок, можуть знаходити ефективні рішення для регулювання параметрів виробництва етанолу, таких як температура, тиск чи концентрація реагентів, адаптуючись до змін виробничого середовища чи параметрів об'єкту. Інтеграція таких моделей у склад автоматизованої системи виробництва спирту та біоетанолу відкриває можливості для підвищення продуктивності, зниження енергозатрат, забезпечення стабільності та адаптивності процесу.

Роль віртуальних сенсорів при проектуванні RL-систем

ВС є незамінними при проектуванні RL-систем, особливо для управління такими складними процесами спиртового виробництва. По-перше, віртуальні сенсори можуть поєднувати різноманітні джерела інформації та дані з численних вимірювальних систем, таких як температура, тиск, витрати рідини та концентрація компонентів у колоні. Вони можуть здійснювати постійний моніторинг і надавати агенту RL точні дані для прийняття рішень. Це особливо важливо в ситуаціях, коли фізичні сенсори можуть бути дорогими або важкими для встановлення. По-друге, віртуальні сенсори можуть створювати точні моделі прогнозування процесів, наприклад, в ректифікаційній колоні, таких як розподіл компонентів у стовпі чи їх температура. Це дозволяє агенту RL використовувати прогнозовані дані для оцінки результатів своїх дій і вибору найкращої стратегії управління. Прогнозування також може знижувати необхідність у частих фізичних вимірюваннях, що покращує ефективність та швидкість процесу. По-третє, фізичні сенсори можуть бути дорогими у встановленні та обслуговуванні, особливо в умовах високих температур або агресивних середовищах. Віртуальні сенсори дозволяють зменшити витрати на фізичні датчики і зробити систему більш гнучкою, оскільки віртуальні сенсори можуть бути налаштовані для надання потрібних даних без фізичного втручання. По-четверте, віртуальні

сенсори здатні швидко адаптуватися до змін в умовах процесу. Вони можуть миттєво реагувати на зміни у параметрах технологічного процесу, надаючи актуальні дані для агенту RL. Це дозволяє системі оперативно коригувати свої стратегії управління, покращуючи ефективність і якість продукту, а також мінімізуючи енергетичні та матеріальні витрати. По-п'яте, віртуальні сенсори дозволяють моделювати змінні процесу, які складно виміряти за допомогою фізичних сенсорів, такі як концентрації компонентів в певних точках колони або швидкість зміни властивостей рідини. Це розширює можливості агенту RL, оскільки він може використовувати більш точні та повні дані для оптимізації процесу.

Отже, віртуальні сенсори є важливими для ефективного та економічного управління процесами в рамках RL-систем, дозволяючи точно відслідковувати, прогнозувати та оптимізувати поведінку технологічного процесу, що в результаті призводить до підвищення продуктивності та зменшення витрат.

Взаємодія віртуальних аналізаторів з RL-системами

Представляючи більш високий рівень функціоналу та інтерпретації даних ніж ВС, віртуальні аналізатори здатні доповнювати та розширювати функціонал RL-системи наступним чином:

- віртуальні аналізатори як інтерпретатори стану – у RL-системі ВС можуть обробляти дані від сенсорів (наприклад, рівень, температура, концентрація етанолу) і надавати агенту узагальнену оцінку стану середовища для прийняття рішень. На основі цього аналізу агент адаптує свою стратегію управління для оптимізації процесу;
- симуляція та прогнозування – ВС можуть моделювати поведінку системи, прогнозуючи наслідки дій агента на основі наявних даних. Наприклад, замість прямого впливу на реальне середовище, агент використовує моделі для оцінки можливих результатів у симульованих умовах;
- аналіз результатів дій агента – ВС можуть використовуватись для оцінки ефективності кожного кроку чи політики агента. Вони здатні виступати як інструменти для вимірювання результатів і того, як зміни в системі впливають

на кінцеві цілі. Наприклад, якщо агент прийняв рішення про корекцію рівня, ВС можуть надавати відгук про те, як ці зміни позначаються на стані системи;

- ітераційний процес вдосконалення – ВС та моделі процесів можуть допомагати в процесі навчання системи, надаючи агенту дані для розрахунку винагороди або штрафу. Це дозволяє агенту коригувати свою стратегію на основі реального або симульованого зворотного зв'язку, що в кінцевому результаті може призвести до покращення ефективності системи в цілому.

Таким чином, ВС можуть виступати як потужний інструмент для інформаційного забезпечення, аналізу, симуляції та оцінки ефективності рішень у рамках RL-системи, створюючи більш гнучке і точне управління процесами в реальному часі.

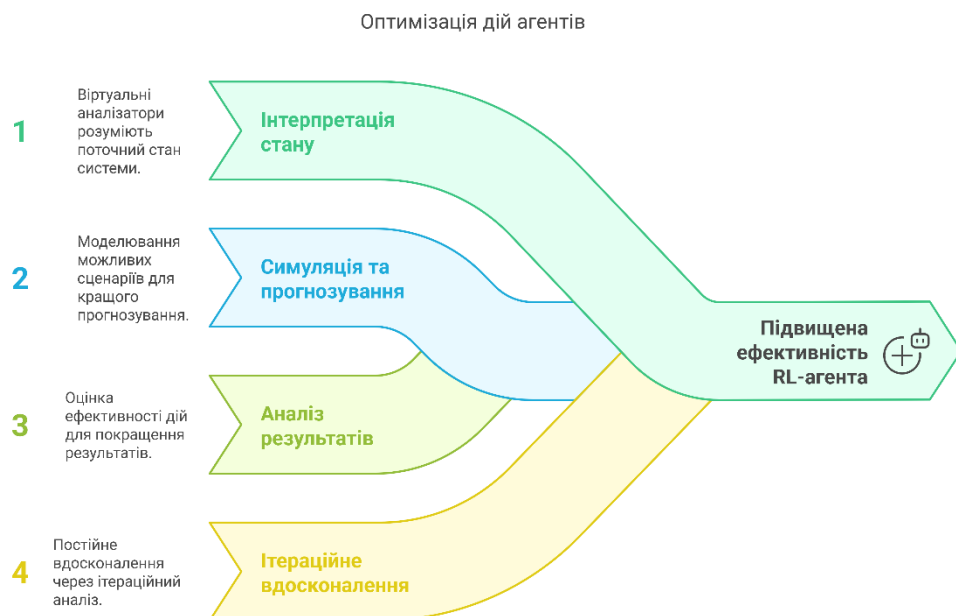


Рис. 1.10. Використання віртуальних аналізаторів для оптимізації RL-системи

1.4.5 Застосування віртуальних сенсорів в контексті виробництва етанолу

Можливість вимірювання всіх змінних процесу має велике значення в області моніторингу та контролю за перебігом процесів, тому дуже важливо мати неперервні вимірювання в режимі реального часу. Чим більш повними та оперативними є вимірювання, тим більш стабільним, відтворюваним та

ефективним може бути процес, що призводить до виробництва продукту вищої якості [50].

Очевидно що можливість вимірювання якнайбільшої кількості параметрів виробництва сильно впливає на керованість перебігу процесів. Це означає, що в процесах де відбуваються складні хімічні процеси, повна реконструкція станів потребує комбінації різноманітних вимірювань та методології їх аналізу.

ВС використовують моделі процесів та алгоритми оцінки для змінних та параметрів, які важко виміряти або які доступні лише після значної затримки в часі від моменту вимірюваних даних в реальному часі. У складних процесах такі змінні часто визначаються за допомогою лабораторних аналізів. Ціна та час, необхідний для проведення такого аналізу, обмежують частоту вибірки. ВС використовують наявні онлайн дані для передбачення таких даних. Ці дані вони також можуть використовувати для визначення неполадок, що вказують на відхилення в роботі системи.

Відсутність можливості надійного моніторингу важливих якісних змінних в режимі реального часу та важкість визначення кількісних взаємозв'язків між поведінкою системи та доступними змінними стану процесів є основними перешкодами в оптимізації керування процесами спиртового виробництва [108].

1.5 Цифрові двійники як ключовий інструмент віртуального контролю у виробництві спирту та біоетанолу

1.5.1 Передумови застосування цифрових двійників

Одним з можливих реалізацій віртуального аналізатора в промисловості Industry 4.0 є цифровий двійник або Digital Twin (DT). Цифровий двійник являє собою віртуальний та комп'ютеризований аналог фізичної системи, який можна використовувати для її моделювання та досягнення різних цілей, використовуючи синхронізацію даних реального часу, отриманих з місця. Така синхронізація стала можливою завдяки підтримувальним технологіям Індустрії 4.0.

В останні роки перед промисловістю у всьому світі стоїть задача стрімкого зростання та розвитку цифрових технологій, що дозволяють легко інтегрувати взаємопов'язані інтелектуальні компоненти всередині виробництва. Цьому значною мірою сприяв розвиток Індустрії 4.0, технологічна основа якої в свою чергу перекликається з Інтернетом речей (IoT), в якому пропонувалося вбудовувати електроніку, програмне забезпечення та мережеве підключення для забезпечення обміну даними в мережі Інтернет. Таким чином IoT можна використовувати на промисловому рівні – пристрої можна підключати через мережеві інфраструктури, що забезпечує більш пряму інтеграцію між фізичним світом та віртуальними системами, що призводить до більш високої ефективності, точності та економічним перевагам [109].

Однією з ключових в Індустрії 4.0 являються кібернетичні системи (CPS) в якості інтелектуальних вбудованих і мережевих систем у виробничих системах. Вони працюють на фізичному та віртуальному рівнях, взаємодіючи з фізичними пристроями та контролюючи їх при цьому впливаючи на реальний світ. Згідно з науковою літературою, щоб повною мірою використовувати потенціал CPS та IoT, потрібно використати необхідні моделі даних, такі як онтології, що являють собою явні, семантичні та формальні концептуалізації понять в предметній області. Вони являються основною семантичною технологією, яка забезпечує інтелект вбудований в CPS, та можуть допомогти в інтеграції та сумісному використанні великих об'ємів даних.

Завдячуючи цим технологіям Індустрія 4.0 відкриває можливість моніторингу в реальному часі та синхронізації реального з віртуальним простором, за допомогою об'єднання елементів в мережу CPS.

1.5.2 Основи функціонування

Цифровий двійник є віртуальним і комп'ютеризованим аналогом фізичної системи, який можна використовувати для її моделювання використовуючи синхронізацію даних в реальному часі [109]. Фактично його можна описати як

повний віртуальний опис фізичного продукту, точне як на мікро, так і на макрорівні.

Одним із ключових понять функціонування віртуального аналізатора є віртуальне середовище. Віртуальне середовище існує в цифровій області і являється дзеркалом фізичного середовища, при цьому дублювання досягається за рахунок фізичної метрології, тобто датчиків, які передають ключові вимірювання із фізичного у віртуальне середовище.

Цифровий двійник реалізується шляхом передачі ключових параметрів фізичної системи в цифрову модель, наприклад через компонент аналого-цифрового перетворювача та додаткових підсистем. Таким чином, цифрова модель являє собою копію фізичного середовища. Така модель обмежує поведінку двійника відтворенням дій виконавчих механізмів фізичної системи, що досягається внаслідок інтеграції параметрів змін об'єкта керування, які надходять від датчиків до цифрового двійника за допомогою алгоритму розпізнавання та оцінки. Запис реальних оперативних даних та ідентифікація їх ключових параметрів можуть виявити певні моменти у виконанні операції, які можна використати для підвищення реалістичності моделювання [110].

Дані є основою роботи цифрового двійника. Вони збираються за допомогою різноманітних датчиків та мають передаватися в режимі реального часу або в режимі близькому до реального часу. Проте дані, необхідні цифровому двійнику, зазвичай мають значний об'єм та різноманітність які важко передати на сервер в режимі реального часу. Тому для забезпечення цього застосовуються граничні обчислення (edge computing), які дозволяють проводити попередню обробку отриманих даних, що зменшує навантаження на мережу [111].

1.5.3 Моделювання DT

Моделювання та симуляція DT є основою практичної реалізації DT. Моделювання DT включає фізичне моделювання, віртуальне моделювання, моделювання зв'язків, моделювання даних та сервісне моделювання [112]:

- Фізичне моделювання забезпечує визначення та опис ключових характеристик фізичного об'єкта.
- Віртуальне моделювання забезпечує створення віртуального представлення фізичної сутності, яке буде відображати ті ж функції та поведінку у віртуальному просторі.
- Моделювання зв'язків забезпечує постійний зв'язок між фізичною моделлю, віртуальною моделлю, моделлю даних та моделлю обслуговування. Типова модель з'єднання охоплює передачу перетворення та захист даних.
- Моделювання даних забезпечує визначення даних та робочих процедур, таких як перевірка безпечності, та їх зберігання. Це забезпечує облегшення обробки даних, оскільки вони зберігаються відповідно до певних логічних критеріїв.
- Сервісне моделювання забезпечує ідентифікацію, аналіз та покращення обслуговування. Воно виконує аналіз операцій, перевірку достовірності віртуальної моделі та забезпечує її рівень шляхом перевірки помилок моделі, алгоритму чи апаратних проблем.

1.5.4 Об'єднання даних в DT

Об'єднання даних в DT складається з трьох процесів попередньої обробки даних, аналізу даних та їх оптимізацію.

Попередня обробка даних необхідна для того зменшити об'єм даних, який необхідно передавати для функціонування DT. Тому сирі дані потребують фільтрації та перетворення. Далі передані дані опрацьовуються методами аналізу даних.

Всі частини DT мають взаємодіяти між собою для вирішення робочих проблем. Виділяють три види взаємодії в структурі DT [112]:

- Фізично-фізична, завдяки якому декілька фізичних об'єктів можуть взаємодіяти один з одним для вирішення складної задачі, яка не може бути виконана окремим пристроєм.

- Віртуально-віртуальна, завдяки якій кілька віртуальних моделей можуть бути зв'язані формуючи спільну мережу для обміну інформацією.
- Віртуально-фізична – віртуальна модель може бути оптимізована синхронно з фізичним об'єктом, а фізичний об'єкт може бути динамічно налаштований на основі прямих вказівок віртуальної моделі.

Висновки до розділу 1 та постановка задач дослідження

Аналізуючи наведену літературу можна підсумувати, що спиртове виробництво є складним технологічним процесом який містить у своєму складі комплексні біохімічні та інерційні процеси. Незважаючи на тривалу історію виробництва етанолу, спиртове виробництво все ще має значну сучасну практичну актуальність, в тому числі етанол може використовуватися як екологічно безпечніша альтернатива паливу, що базується на дефіцитних для України нафтопродуктах. В той час як сировина для виробництва етанолу є доступною завдяки значному аграрному сектору України.

Новітні технології інтелектуального аналізу та збереження і передачі даних дозволяють використовувати більш потужні методи оптимізації та математичного аналізу процесів, які були недоступними до того. Тому незважаючи на множину попередніх наукових розробок в цій області виробництва, можливість подальшої автоматизації та оптимізації та пошук нових технічних рішень для спиртового виробництва залишається актуальною задачею.

Проблеми автоматизації у виробництві етанолу зазвичай пов'язані з пошуком оптимальних умов максимізації виходу та мінімізації виробничих витрат. Серед них:

- Велика інерційність виробництва, що ускладнює пошук оптимальних налаштувань системи автоматизації. Пошук оптимальних налаштувань потребує багато ручної праці та піддається помилкам пов'язаним з людським фактором. Складність визначення оптимальних умов бродіння, пов'язана з залежністю

якості вихідного продукту від великої кількості комплексних змінних, таких як температура, рН, швидкість перемішування для різних сортів дріжджів.

- У процесі виробництва важливу роль відіграють змінні, які неможливо виміряти безпосередньо в режимі реального часу, наприклад, концентрацію етанолу на проміжних стадіях виробництва. До того ж фізичні датчики можуть працювати некоректно або взагалі вийти з ладу. Налаштування таких датчиків у ключових точках контролю потребує часу, що може призвести до втрати якості продукції.

- Складність регулювання процесу ректифікації та дистиляції пов'язана з нелінійністю, багатовимірністю об'єкта, а також значною інерційністю системи, наявністю перехресних зв'язків між параметрами та впливом зовнішніх збурень, що ускладнює забезпечення стабільності й оптимальних умов роботи.

Вище зазначена інформація обумовила вибір теми дисертації та її мети, а саме є підвищення ефективності виробництва спирту та біоетанолу шляхом вдосконалення автоматизованої системи моніторингу та керування, що засновані на віртуальних інструментах контролю. У науковій літературі розглядалися різноманітні методи прогнозування змінних технологічних процесів, алгоритми ідентифікації моделей та адаптивного управління, в тому числі особливості використання м'яких датчиків у біопроцесах. Проте тема інтеграції цих підходів у комплексні системи автоматизованого контролю, особливо в умовах динамічних змін і можливих збоїв фізичних датчиків, залишається відкритою та актуальною.

На основі цього можна зробити наступні висновки:

- Спиртовому виробництву характерна наявність проблем пов'язаних з автоматизацією складних біологічних та хімічних процесів, які важко вирішити класичними методами, таких як інерційність, багатовимірність та нелінійність об'єктів, відсутність можливості вимірювання важливих змінних, наявність великої кількості взаємозалежних змінних.

- Наявність латентних ознак у процесах виробництва спирту та біоетанолу обґрунтовується їх складністю, багатофакторністю та взаємозалежністю фізико-

хімічних явищ. Виявлення й аналіз цих латентних характеристик за допомогою математичних моделей, машинного навчання чи віртуальних сенсорів дозволяє глибше зрозуміти механізми цих процесів, покращити контроль і підвищити якість продукції.

- Віртуальні засоби контролю мають інструменти для вирішення багатьох проблем пов'язаних з продуктивністю спиртового виробництва. Шляхом визначення прихованих зв'язків між відомими даними процесу та очікуваними результатами, м'який датчик може робити передбачення результатів процесу, та слугувати системою підтримки рішень для підвищення ефективності процесу. Крім того, додаткові можливості ВС, такі як аналіз системи на помилки датчиків чи дублювання їх роботи може слугувати додатковим джерелом захисту від помилок на виробництві та зменшувати витрати на дублюючі фізичні сенсорні системи, та попереджуючи про можливу аварійну ситуацію в системі.

- Застосування ВС в складі автоматизованої системи управління є одним з етапів впровадження Індустрії 4.0 на виробництві.

Таким чином, можна сформулювати задачі дослідження:

- провести аналіз сучасного стану розвитку віртуальних інструментів контролю для технологічних комплексів спиртового та біоетанольного виробництва, визначити основні напрями наукових досліджень та їх актуальність, сформулювати основні проблеми у цій галузі та обґрунтувати методи для їх вирішення;

- провести аналіз сучасних математичних моделей технологічних об'єктів з метою формування онтологічної моделі для використання віртуальних сенсорів у моніторингу технологічних процесів;

- виконати синтез та провести моделювання віртуальних сенсорів для визначення та прогнозування ключових технологічних показників спиртового та біоетанольного виробництва;

- здійснити розробку концептуальної структури автоматизованої системи моніторингу та керування технологічними процесами з використанням віртуальних інструментів контролю;

- синтезувати автоматизовану систему керування технологічними процесами з використанням інтелектуальних механізмів, провести її моделювання;

- розробити хмарну архітектуру для системи моніторингу технологічними процесами виробництва спирту та біоетанолу, яка забезпечить високу надійність, гнучкість та масштабованість. Реалізувати запропоновану архітектуру з використанням сучасних методів обробки даних і віртуальних інструментів контролю, а також провести її апробацію.

РОЗДІЛ 2. СИСТЕМНИЙ ПІДХІД ДО АНАЛІЗУ ТА МОДЕЛЮВАННЯ ВИРОБНИЧИХ ПРОЦЕСІВ

2.1 Системний аналіз технологічних відділень виробництва

Спиртове виробництво являє собою систему, що складається із взаємопов'язаних відділень, в яких відбувається послідовний процес перетворення сировини. Цей процес включає в себе ряд технологічних операцій, перебіг кожної з яких впливає на перебіг біохімічних реакцій та відіграє важливу роль у формуванні якості кінцевого продукту.

Структура виробничої системи визначається складом і взаємозв'язками її об'єктів. Стан об'єктів автоматизації характеризується сукупністю технологічних параметрів. Виділення та спостереження за виробничими змінними на різних рівнях виробництва дозволяє прогнозувати їх взаємодію та сукупний вплив на результати перебігу процесів перетворення сировини на кінцеві продукти виробництва (Рис. 2.1).

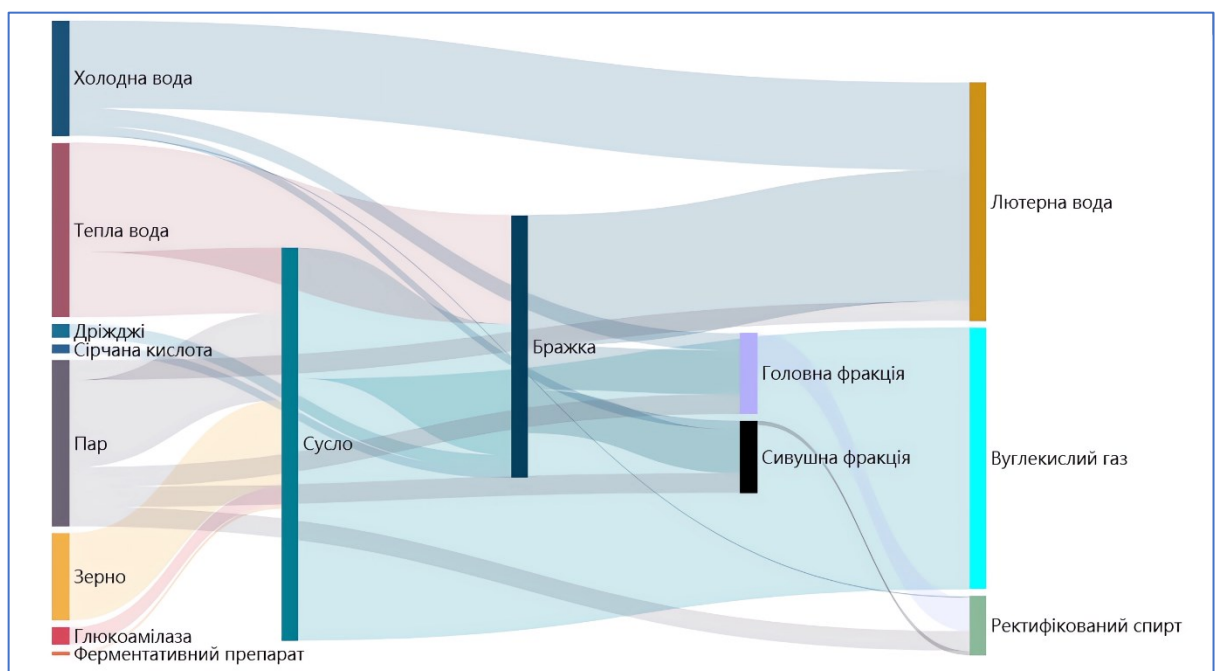


Рис. 2.1. Sankey діаграма перебігу матеріальних потоків спиртового виробництва

Опираючись на рівні в ієрархії керування стандарту ISA-95 в контексті спиртового виробництва можна виділити області в межах яких відбувається аналіз взаємодії параметрів:

1) Рівень підприємства(Enterprise Level)

2) Рівень виробничого управління(Manufacturing Operations Management)

3) Рівень відділень (SCADA Level);

4) Рівень апаратів (Device Level);

Для надання оцінки результатам перебігу виробництва на різних, а особливо вищих рівнях, можуть використовуватися Key Performance Indicators (KPI , ключові показники ефективності) - кількісно вимірювальні стратегічні оцінки, що відображають критичні фактори успіху організації. KPI використовуються для контролю та оцінки плинного стану, прогнозування, аналізу ретроспективних даних.

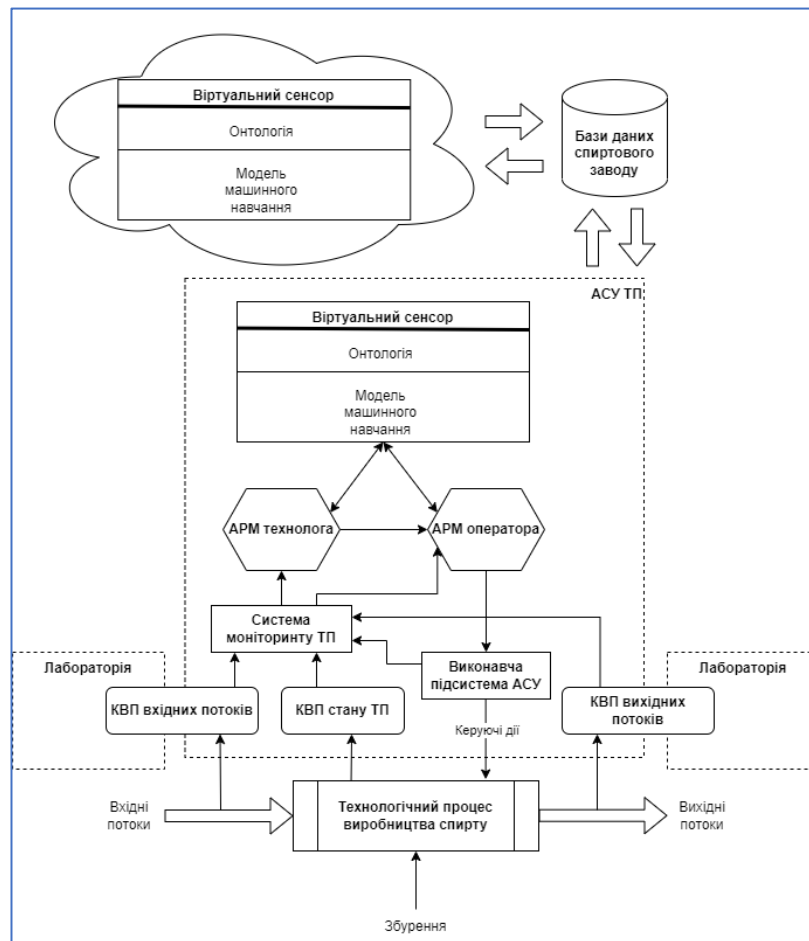


Рис. 2.2. Структурна схема управління виробництвом

Рівень підприємства

На рівні підприємства аналізуються показники ефективності роботи організаційної стратегії ERP (enterprise resource planning, планування ресурсів підприємства), що включає в себе аналіз використання ресурсів та вплив

результати роботи усіх відділів виробництва спирту на бізнес показники підприємства. Вони представляють KPI на цьому рівні та базуються на KPI з рівня виробничого управління.

Рівень виробничого управління

На рівні виробничого управління завдяки узагальненій інформації від всіх відділень стає можливим проведення комплексного аналізу ефективності виробництва. Для виробничих операцій згідно стандарту ISO 22400 застосовуються KPI пов'язані з:

- керуванням операціями виготовлення продукції (виробнича продуктивність, використання обладнання, час циклу виробництва, витрати сировини та енергії на виробництво одиниці продукції);
- керування операціями технічного обслуговування обладнання та ремонтів керування операціями якості (контроль якості, лабораторні вимірювання);
- керування операціями з запасами (постачання сировини, продукту).

Типове спиртове виробництво, в залежності від особливостей технологічних процесів, поділяється на зв'язані між собою відділення між якими розподіляються послідовні задачі по перетворенню сировини (Рис. 2.2):

1) Відділення підготовки сировини та варки сусла

Задачі відділення полягають в обробці крохмалевмісної сировини та перетворення її в сусло шляхом гідролізу полісахаридів для подальшого зброджування.

2) Бродильне відділення

Задачею бродильного відділення є зброджування цукрів, що містяться в суслі за допомогою спиртових дріжджів. В процесі спиртового бродіння отримується бражка – багатокомпонентна суміш, що містить в своєму складі етиловий спирт та домішки у вигляді побічних продуктів бродіння (ефіри, альдегіди і феноли та інші).

3) Брагоректифікаційне відділення

Задачею брагоректифікаційного відділення є ректифікація бражки за допомогою протиточного масообміну між парою та рідиною, це дозволяє відділити етиловий спирт від компонентів що мають іншу температуру кипіння.

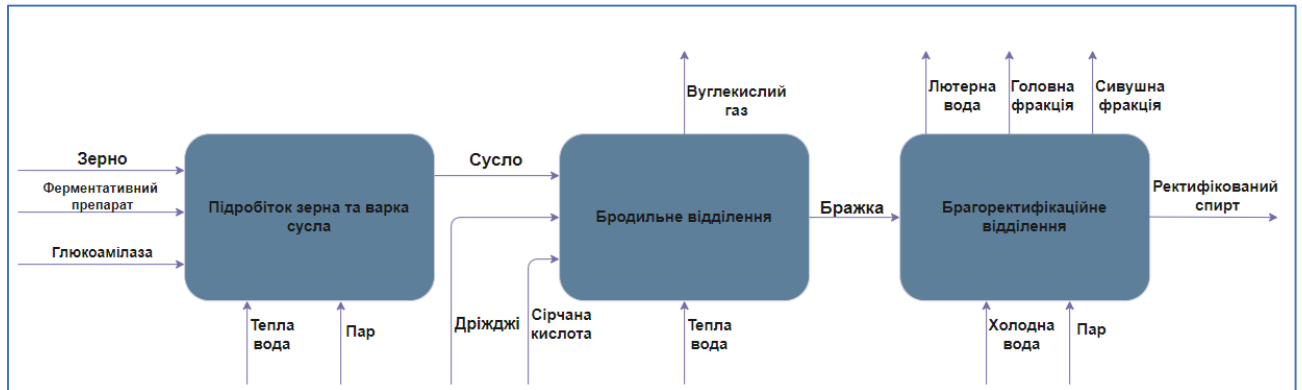


Рис. 2.3. Схема взаємодії матеріальних потоків відділень спиртового виробництва

На рівні підприємства розглядається взаємодія відділень з точки зору впливів змін входних параметрів відділень на вихідні параметри в межах виробництва.

Впровадження інтелектуальних систем аналізу на базі машинного навчання стає можливим завдяки визначенню залежностей між значеннями змінних у межах всього виробництва. Передумовою аналізу такого типу є виділення відстежуваних кількісно-якісних показників відділень (Рис. 2.4).

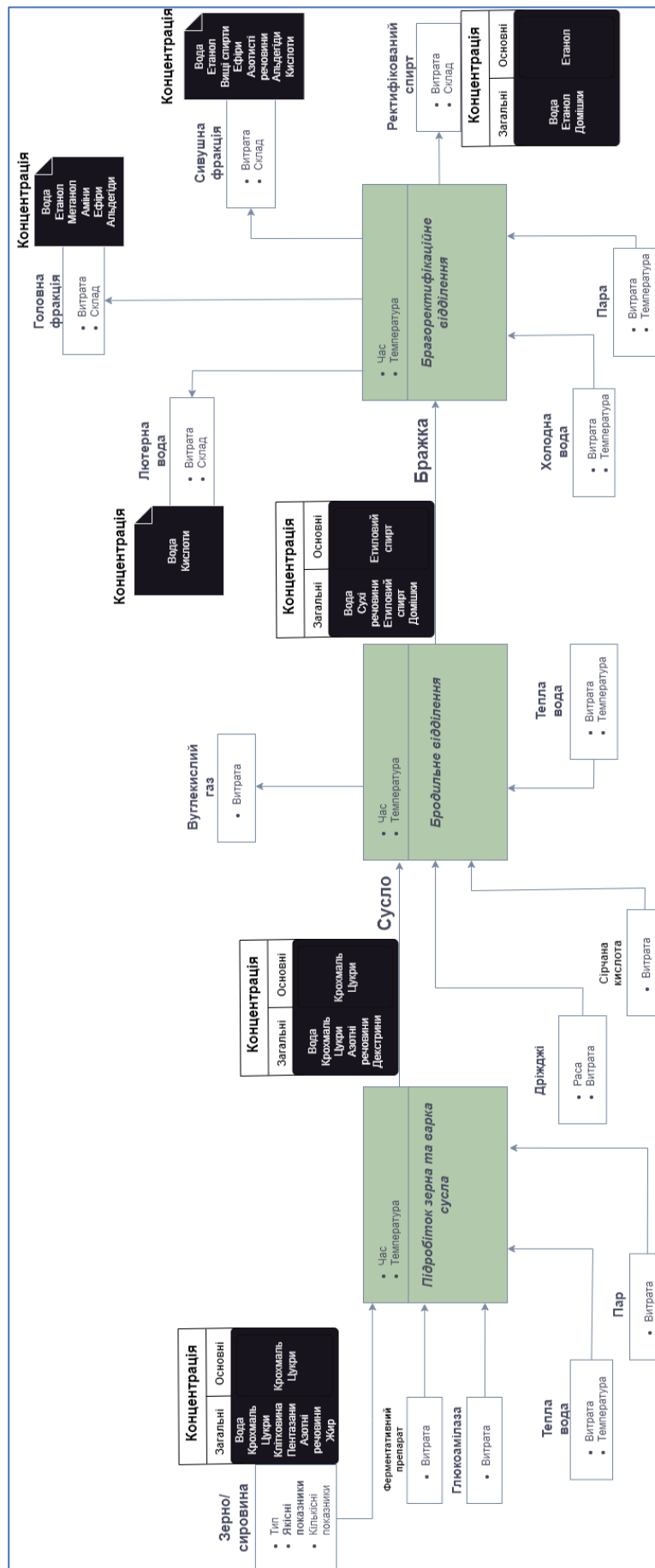


Рис. 2.4. Загальна структурна схема відстежуваних кількісно-якісних показників

Рівень відділень та апаратів

На рівні відділень спиртового виробництва розглядається взаємодія апаратів об'єднаних між собою спільним технологічним потоком з метою обробки та перетворення сировини до певного стійкого проміжного стану. Таким чином аналізу піддаються всі апаратні змінні в межах відділення.

1) Сусловарильне відділення

Сусловарильне відділення відповідає за перетворення крохмаловмісної сировини в сусло. Для проведення цього процесу згідно технології використовуються типові апарати (Рис. 2.5):

- Змішувач;
- Апарат гідродинамічної та ферментативної обробки I ступеня;
- Апарат гідродинамічної та ферментативної обробки II ступеня;
- Стерилізатор;
- Паросепаратор;
- Випарник-зацукрювач.



Рис. 2.5. Апарати сусловарильного відділення

Об'єднані спільним технологічним потоком, дані апарати беруть участь у послідовній переробці крохмаловмісної сировини в сусло, в складі сусловарильного відділення (Рис. 2.6). Аналіз їх вхідних та вихідних змінних дає змогу прогнозувати поведінку в межах сусловарильного відділення.

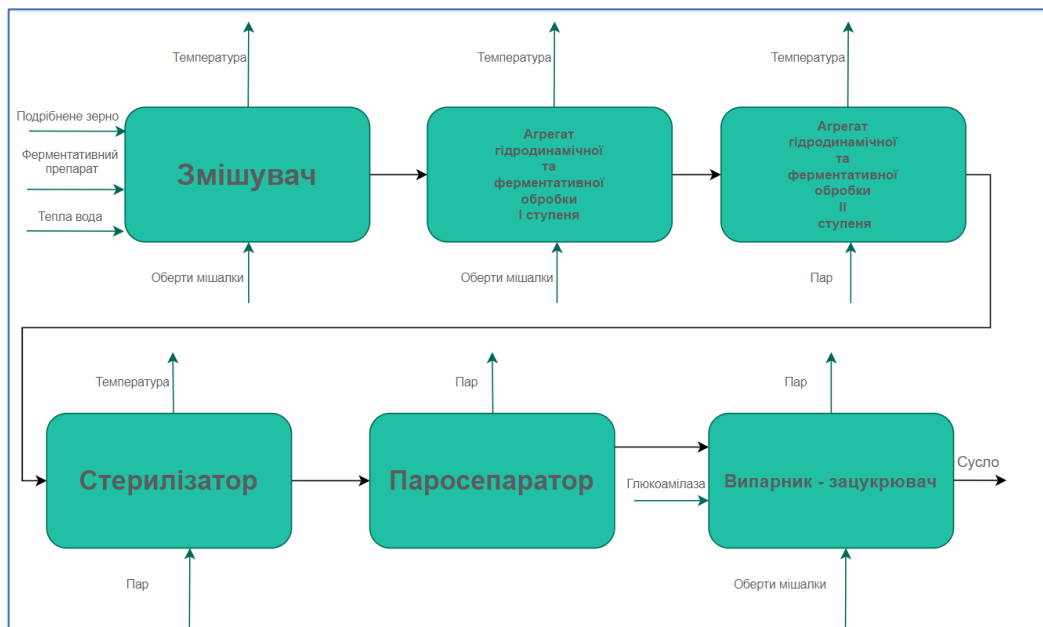


Рис. 2.6. Структурна схема сусловарильного відділення

2) Бродильне відділення

Бродильне відділення відповідає за виробництво бражки шляхом зброджування сусла. Для проведення цього процесу згідно технології використовуються типові апарати (Рис. 2.7):

- Дрожжанки;
- Зброджувачі;
- Головний бродильний агрегат;
- Бродильні агрегати

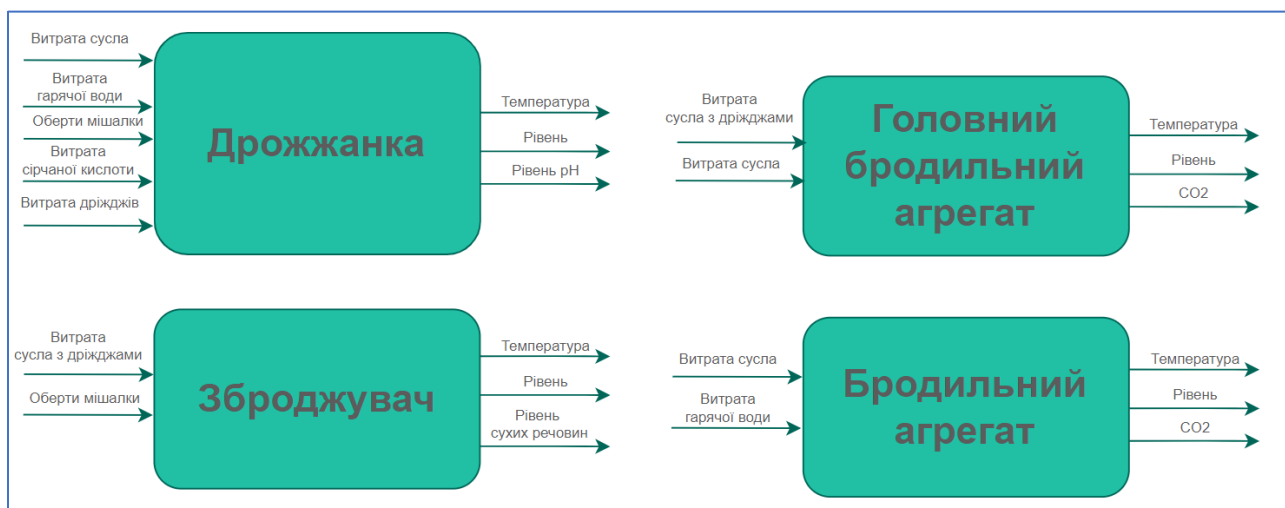


Рис. 2.7. Апарати бродильного відділення

В складі відділення завдяки взаємодії апаратів виконується підготовка та процес зброджування цукрів, від вирощування дріжджів до безпосередньо спиртового бродіння (Рис. 2.8).

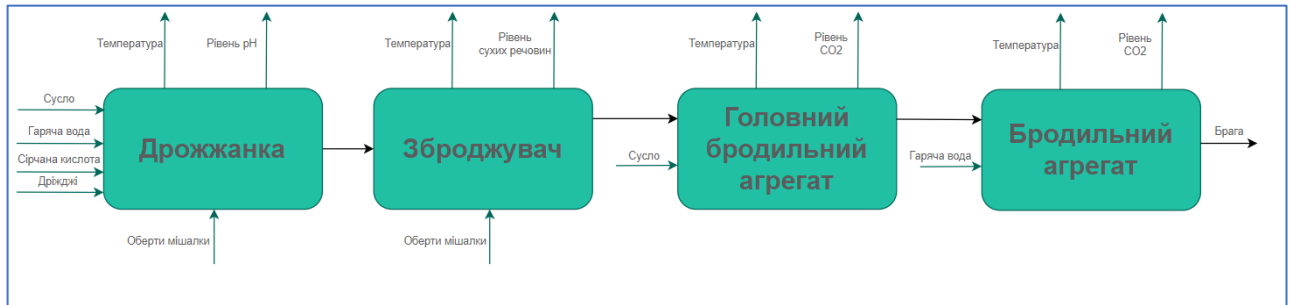


Рис. 2.8. Структурна схема бродильного відділення

3) Брагоректифікаційне відділення

Завданням брагоректифікаційного відділення є ректифікація етанолу з бражки. В основі роботи відділення лежить брагоректифікаційна установка (БРУ). Типова БРУ складається з трьох колон, серед яких (Рис. 2.9):

- Бражна колона;
- Епюраційна колона;
- Спиртова колона.



Рис. 2.9. Апарати брагоректифікаційного відділення

Аналіз об'єднаних між собою в одну систему змінних ректифікаційних колон дозволяє прогнозувати та управляти перебігом процесу спиртової ректифікації (Рис. 2.10).

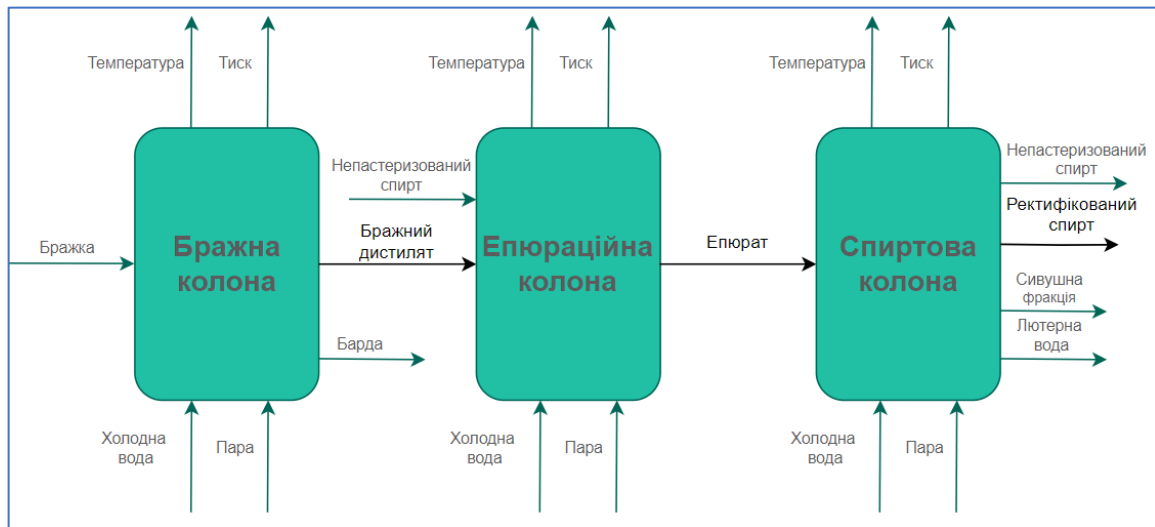


Рис. 2.10. Структурна схема брагоректифікаційного відділення

Проблема вибору моделі для реалізації ВА. Однією з особливостей використання ВА є необхідність вибору математичної моделі (ММ) для забезпечення ефективного функціонування системи. Правильно підібрана модель дозволяє мінімізувати помилки, підвищити надійність аналізу та, в перспективі, оптимізувати роботу всієї системи [123, 124]. Однак неправильний вибір моделі може призвести до помилкових результатів, зниження точності прогнозів і ефективності системи управління в цілому, що нівелює переваги застосування ВА.

Зміст ММ полягає в максимальному наближенні до відтворення поведінки технологічного об'єкту, тому вибір моделі здійснюється на основі аналізу властивостей об'єкта, які відображаються в характеристиках даних, що він продукує [125, 126]. Цей процес потребує значних затрат часу та зусиль для досягнення необхідної відповідності. Одним із варіантів вирішення цих проблем є розробка та представлення різнотипних ММ у вигляді онтологічної бази знань. Така база знань може бути використана як складова системи підтримки прийняття рішень, яка дозволить визначити перелік моделей, здатних ефективно працювати з наявними даними відповідно до їхніх характеристик.

2.2 Розробка та використання прикладної онтології віртуальних сенсорів та їх математичних моделей

2.2.1 Місце математичної моделі в ієрархії моделей

Вибір ММ є складною задачею, що залежить від багатьох факторів, зокрема цілей використання моделі, характеристик та специфіки функціонування об'єкта, вимог до системи та обмежень навколишнього середовища, а також вимог до точності й адекватності. З урахуванням цього, її структура та представлення в онтології узагальнюються у вигляді класів моделей, згрупованих за ключовими ознаками.

Модель є абстрактним або матеріальним відображенням об'єктів або процесів, тотожним досліджуваним об'єктам (процесам) щодо деяких заданих критеріїв. ММ – це наближений опис певного класу явищ зовнішнього світу, вираженого за допомогою математичної символіки.

Так як ММ є одним із похідних «моделі» як такої, то запропонована класифікація починається з класу «модель» (Рис. 2.11).

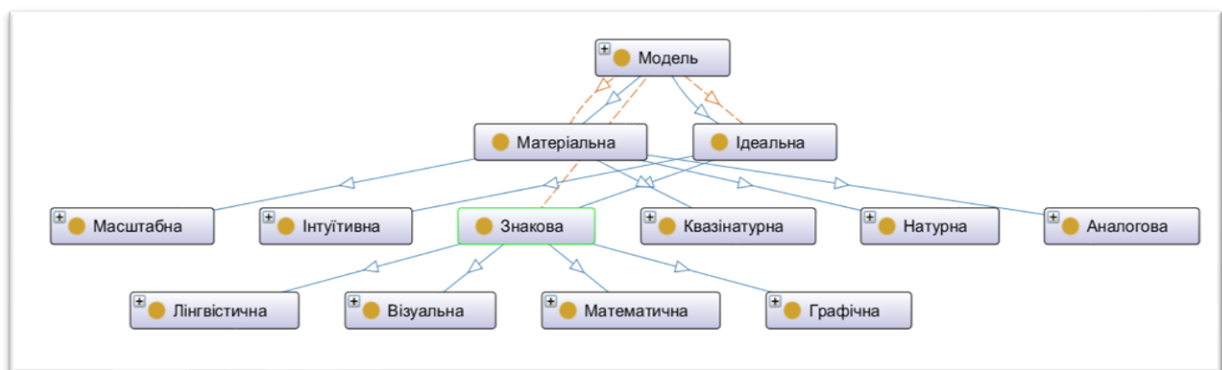


Рис. 2.11. Місце математичної моделі в ієрархії моделей

У свою чергу ММ поділяється на підвиди в залежності від структурних та функціональних особливостей. Такі особливості формують зв'язки або відношення R концептів X онтології, які умовно можна розділити на наступні базові категорії:

- 1) Відношення за структурними особливостями.
- 2) Відношення за способом подання моделі.
- 3) Відношення за об'єктними властивостями моделі.

- 4) Відношення за призначенням моделі.
- 5) Відношення за математичними залежностями.

2.2.2 Відношення та підкласи класу математичної моделі

1) За структурними особливостями

За структурними особливостями ММ можна виділити групи, які формують відношення в залежності від властивостей структури моделі: наявності чи відсутності інформації про внутрішню будову, стохастичних залежностей чи ієрархічних рівнів (табл. 2.1).

Таблиця 2.1. Відношення та підкласи класу математичної моделі за структурними особливостями

№	R (відношення)	X (концепти)
1	В залежності від структури (ступінь наявності інформації про внутрішню структуру моделі)	Чорний ящик Білий ящик Сірий ящик
2	За вмістом ймовірнісних компонентів (наявність стохастичних залежностей)	Стохастична Детермінована
3	За належністю до ієрархічного рівня (характер структурного аналізу)	Мікрорівнева Метарівнева Макрорівнева

Серед описаних груп ММ:

- Моделі Чорного ящика – це моделі системи з невідомою внутрішньою структурою, та відомими вихідними і вхідними параметрами, через дослідження яких можна отримати уявлення про внутрішню будову.
- Моделі Білого ящика – це моделі системи з відомими компонентами (відомими підсистемами, їх зв'язками, функціями системи, організаційно-ієрархічною структурою).
- Моделі Сірого ящика поєднують наявність відомих вихідних і вхідних параметрів з наявною інформацією про часткову теоретичну структуру моделі.
- Детерміновані моделі використовують в описі тільки не випадкові величини, а результати на виході однозначно визначаються керуючими вхідними впливами.

- Стохастичні (імовірнісні) моделі використовують в описі випадкові величини, під впливом яких на виході можуть отримуватися різні результати.

- Мікрорівневе моделювання системи пов'язане з детальним описом кожного компонента системи, дослідженням її структури, функцій та взаємозв'язків.

- Макрорівневе моделювання полягає в ігноруванні детальної структури системи та вивченні лише загальної поведінки системи як єдиного цілого. Метою такого моделювання є побудова моделі системи шляхом дослідження її взаємодії із зовнішнім середовищем.

- Метарівневе моделювання – моделювання системи пов'язане з описом міжсистемної взаємодії.

2) За способом подання

За способом подання виділено групу зв'язків, що включають ступінь абстракції моделі, характер подання інформації та характер даних (табл. 2.2).

Таблиця 2.2. Відношення та підкласи класу математичної моделі за способом подання

№	R (відношення)	X (концепти)	
1	За способом отримання (характер подання інформації про об'єкт)	Формальна Неформальна Емпірична Теоретична	
2	За способом представлення властивостей об'єкта (характер моделі даних)	Графова Аналітична Теоретико-множинна	Імітаційна Алгоритмічна Логічна
3	За способом представлення об'єкту (ступінь абстракції)	Формальна Змістовна	
4	За ступенем деталізації (ступінь деталізації будови моделі)	Повна математична модель Макромодель	

При цьому:

- Формальні моделі – це моделі, що задані на математичній або іншій формалізованій мові.

- Неформальні моделі являють собою набір таких інструментів, які здатні пояснити спостереження, але при цьому визначені недостатньо повно, що не

дозволяє з високою точністю перевірити ступінь їхньої логічної взаємопов'язаності.

- Емпіричні моделі – це математичні вирази, що апроксимують експериментальні дані про залежність параметрів стану системи від значень параметрів факторів, що впливають на них.

- Теоретичні моделі – являють собою певне теоретичне та схематичне уявлення про реальність за допомогою ідеалізованих абстракцій, об'єктів.

- Імітаційні моделі – це різновид математичних моделей, які описують динамічні процеси в реальних системах і піддаються перевірці.

- Графові моделі – це ймовірнісні моделі, в яких графом виражено умовні залежності між випадковими змінними.

- Аналітичні моделі є системою співвідношень між заданими й шуканими величинами, що виражені математичними формулами в явному вигляді.

- Теоретико-множинні моделі – це моделі даних, в яких використовується апарат реляційної алгебри, реляційного обчислення, а операції над даними управляються таблицями.

- Алгоритмічні моделі – це моделі в яких описується алгоритм функціонування системи в певний спосіб.

- Логічні моделі є складними булевими функціями, що залежать від ряду змінних, кожна з яких може приймати значення 0 або 1 у залежності від стану відповідного функціонального елемента об'єкта.

- Формальні моделі – це окремі типи моделей, подані у формальному, описовому вигляді, в який входять головні ознаки, за якими дана модель суттєво відрізняється від інших.

- Змістовні моделі – це моделі, наповнені поняттями даної предметної області. Вони будуються на основі формальних моделей, що служать шаблоном, зразком для побудови змістовних моделей.

- Повні математичні моделі – це моделі, що відображають стан як системи, що моделюється, так і всіх її внутрішніх елементів.

- Макромоделі це моделі які адекватні щодо зовнішніх властивостей об'єкта дослідження.

3) За об'єктними властивостями

Об'єктні властивості моделі включають характери опису, властивостей та елементів об'єкта (табл. 2.3).

Таблиця 2.3. Відношення та підкласи класу математичної моделі за об'єктними властивостями моделі

№	R (відношення)	X (концепти)	
1	За типом значення (характер опису об'єкта)	Чисельна Символьна Змішана	
2	За характером властивостей (характер властивостей об'єкта)	Аналітична Каркасна Топологічна Геометрична макромодель Структурна математична модель Кінематична Гідравлічна	Канонічна Хімічна Геометрична Алгебраїчна Електрична Функціональна математична модель Механічна
3	За характером зміни змінних (характер елементів об'єкта)	Неперервна Дискретно-неперервна Дискретна	

До зв'язків характеру опису належать класи:

- Чисельні моделі – математичні моделі, основними елементами яких є конкретні чисельні значення характеристик системи, що моделюється.

- Символьні моделі – це логічні об'єкти, які замінюють реальний процес і виражають основні властивості його відносин за допомогою певної системи знаків або символів.

- Каркасні моделі – моделі об'єкта в тривимірній графіці, що представляють собою сукупність вершин і ребер, яка визначає форму багатогранного об'єкта, що відображається.

До зв'язків характеру властивостей належать класи:

- Топологічні моделі – є одними з математичних мов моделювання для опису функціональності системи. Зазвичай вони мають вигляд графа, де вузли

відображають функціональні можливості системи, а спрямовані дуги - топологічні відносини між ними.

- Геометричні моделі, засновані на математичних методах аналітичної геометрії, які забезпечують введення і перетворення двовимірних та тривимірних об'єктів з урахуванням обмежень умов, пов'язаних з організацією взаємодії, станом обчислювальної техніки, можливостями засобів відображення.

- Структурні математичні моделі – подають об'єкт моделювання з точки зору його складу та взаємозв'язку елементів системи між собою та з зовнішнім середовищем.

- Кінематичне моделювання – моделювання руху без урахування факторів, що його викликали (тільки наслідки).

- Канонічні моделі визначають структуру об'єкта управління та характеризують його зв'язки із середовищем, які здійснюються через входи та виходи керованої системи.

- Функціональні моделі призначені для вивчення особливостей роботи (функціонування) системи та її призначення у взаємозв'язку з внутрішніми та зовнішніми елементами;

- Алгебраїчні моделі є сукупністю взаємозв'язаних математичних та формально-логічних виразів, як правило, відображаючи реальні процеси та явища.

У залежності від характеру функцій об'єкта, який описують моделі, вони можуть бути механічними, гідравлічними, електричними, хімічними тощо.

За характером зміни змінних моделі поділяються на:

- Неперервні моделі – це математичні моделі, кожне невідоме яких змінюється неперервно в області свого визначення.

- Дискретно-неперервні моделі – математичні моделі, що містять як неперервні елементи, так і дискретні.

- Дискретні моделі – це математичні моделі, всі змінні і параметри яких є дискретними величинами.

4) За призначенням моделі

Таблиця 2.4. Відношення та підкласи класу математичної моделі за призначенням моделей

№	R (відношення)	X (концепти)	
1	Залежно від зв'язку між вхідними даними та досліджуваними властивостями (призначення моделі)	Функціональна Алгоритмічна	
2	Залежно від цілей моделювання (призначення моделі)	Імітаційна Дескриптивна Ігрова Оптимізаційна Модель машинного навчання Контрольовано навчена модель	Самостійно навчена модель Модель класифікації Модель регресії Модель зниження розмірності Модель кластеризації

За призначення математичні моделі поділяються на наступні класи (табл. 2.4):

Імітаційні моделі – передбачають представлення моделі у вигляді деякого алгоритму, виконання якого імітує послідовність зміни станів у системі, таким чином відображаючи поведінку системи або процесу, що моделюється.

Дескриптивні (описові) моделі призначені для опису і пояснення спостережуваних факторів або прогнозування поведінки об'єктів;

Ігрові моделі – це спрощені математичні моделі конфліктних ситуацій з декількома сторонами конфлікту;

Оптимізаційні моделі – це моделі прийняття рішення які необхідно оптимізувати за умови дотримання набору заданих обмежень відповідно до показнику ефективності. До цих моделей також відносяться моделі машинного навчання – додатки штучного інтелекту, які дають можливість автоматично вдосконалюватись та навчатися а основі набутого досвіду. У свою чергу вони поділяються на контрольовано навчені та самостійно навчені моделі.

Контрольоване навчання – це підхід до машинного навчання, заснований на використанні наборів даних. Такі набори даних використовуються для створення алгоритмів, орієнтованих на класифікацію даних або точне прогнозування результатів. Контрольоване навчання залежно від задач поділяється на два типи

– класифікацію та регресію. У задачах класифікації для отримання результатів модель використовує алгоритми класифікації такі як лінійні класифікатори, дерева рішень, випадковий ліс, метод опорних векторів та інші. У моделях регресії використовується алгоритм виявлення взаємозв'язку між залежними і незалежними змінними. До поширених алгоритмів регресійного аналізу належать логістична регресія, лінійна регресія та поліноміальна регресія.

При неконтрольованому (самостійному) навчанні алгоритми машинного навчання використовуються для аналізу та групування наборів нерозмічених даних. Моделі неконтрольованого навчання використовуються для виконання завдань кластеризації та зниження розмірності. Кластеризація – це метод інтелектуального аналізу даних, що застосовується для групування нерозмічених даних виходячи з їх подібностей та відмінностей. Зниження розмірності – це метод навчання, який використовується в тому випадку, коли певний набір даних має дуже багато ознак (або розмірностей). Він скорочує кількість вхідних даних до керованого, зберігаючи їх цілісність.

4) За математичними залежностями

Остання із запропонованих груп зв'язків відноситься до особливих математичних залежностей та включає наступні класи (табл. 2.5):

- Лінійні моделі – моделі, що відображають стан або функціонування системи таким чином, що всі взаємозалежності в ній приймаються лінійними.
- Нелінійні моделі – математична модель, що відображає стан або функціонування таким чином, що всі або деякі взаємозв'язки в ній приймаються нелінійними, тобто містять нелінійні елементи.
- Розподілені моделі – відображають нерівномірність розподілу структурних та фізичних параметрів.
- Моделі із зосередженими параметрами – це моделі в яких основні змінні процесу не змінюються в просторі, а тільки в часі. Також до цієї групи належать статичні моделі, які відображають стан об'єкта в певний фіксований момент часу та динамічні моделі, які описують зміну (динаміку) станів об'єкта.

- Моделі є стаціонарними, якщо вид їх функцій переходів і функцій виходів не змінюється з плином часу. Моделі, для яких не виконуються ці умови, називаються нестаціонарними. У залежності від типу нестаціонарності вони поділяються на параметрично-нестаціонарні та структурно- нестаціонарні.

Таблиця 2.5. Відношення та підкласи класу математичної моделі за математичними залежностями

№	R (відношення)	X (концепти)	
1	Залежно від зв'язку між вхідною та вихідною змінною (наявність нелінійних елементів)	Лінійна Нелінійна	
2	Залежно від параметрів (наявність просторової розподіленості об'єкту)	Зосереджена Розподілена	
3	Залежно від поведінки системи (характер структурної динаміки моделі)	Нестаціонарна Параметрично- нестаціонарна Структурно- нестаціонарна	Статична Динамічна Стаціонарна

Таким чином, онтологія являє собою множину підкласів математичної моделі, яка описує технологічні процеси та комплекси за виділеними ознаками. При цьому деякі сформовані підкласи перетинаються створюючи структуру де певні ознаки можуть відповідати кільком класам. Деякі класи мають незамкнену множину підкласів, що дозволяє в подальшому їх доповнювати залежно від специфіки поставлених задач.

2.2.3 Реалізація онтологічного класифікатора

Описані вище концепти було реалізовано в прикладну онтологію використовуючи інструменти платформи Protege. Protege – це редактор онтології з відкритим кодом та система управління знаннями. Protege є однією з кращих платформ для побудови онтологій завдяки наявності вбудованого semantic reasoner, функціональних розширень, можливостей побудови різносторонніх зв'язків між сутностями, системи візуального відображення бази знань та підтримки структурованих запитів та параметризованих аксіом.

Онтологія у Protege представляється у вигляді графу, що описується кортежем:

$$O = \langle X, R, F \rangle, \quad (2.1)$$

де $X = \{x_i \mid i = \overline{1, n}\}$ – кінцева непуста множина концептів (понять предметної області); $R = \{r_i \mid i = \overline{1, m}\}$ – кінцева множина семантично-значимих відношень між концептами; $F: X \times R$ – кінцева множина функцій інтерпретації, що задані на концептах та відношеннях. До концептів відносяться класи, підкласи та екземпляри онтології. Відношення представлені двома видами: об'єктні відношення (Object Property) та відношення даних (Data Property). Функції інтерпретації розширюються аксіомами, які можна сформулювати для класів, відношень та екземплярів.

Сформовані концепти відношень класу ММ в структурі онтології розміщені в певні групи за певними властивостями (Рис. 2.12), та представлені у вигляді наступних гілок:

- В залежності від структури (Рис. 2.13).
- В залежності від вмісту ймовірнісних компонентів (Рис. 2.14).
- В залежності від ієрархічного рівня (Рис. 2.15).
- В залежності від способу отримання (Рис. 2.16).
- За способом представлення:
 - за способом представлення властивостей об'єкта (Рис. 2.17);
 - за способом представлення об'єкта (Рис. 2.18).
- За ступенем деталізації (Рис. 2.19).
- За типом значення (Рис. 2.20).
- За характером властивостей (Рис. 2.21).
- За характером зміни змінних (Рис. 2.22).
- Залежно від зв'язку:
 - залежно від зв'язку між вхідними даними та досліджуваними властивостями (Рис. 2.23);
 - залежно від зв'язку між вхідною та вихідною змінною (Рис. 2.24).
- Залежно від параметрів (Рис. 2.25).

- Залежно від поведінки системи (Рис. 2.26).
- Залежно від цілей моделювання (Рис. 2.27).

✓	В_залежності_від_структури (Domain>Range)
✓	Використовує (Domain>Range)
✓	За_вмістом_вірогіднісних_компонентів (Domain>Range)
✓	За_належністю_до_ієрархічного_рівня (Domain>Range)
✓	За_способом_отримання (Domain>Range)
✓	За_способом_представлення_властивостей_об'єкта (Domain>Range)
✓	За_способом_представлення_об'єкту (Domain>Range)
✓	За_ступенем_деталізації (Domain>Range)
✓	За_типом_значення (Domain>Range)
✓	За_формою_обробки_та_представленням_інформації (Domain>Range)
✓	За_характером_властивостей (Domain>Range)
✓	За_характером_зміни_змінних (Domain>Range)
✓	Залежно_від_зв'язку_між_вхідними_даними_та_досліджуваними_властивостями (Domain>Range)
✓	Залежно_від_зв'язку_між_вхідною_та_вихідною_змінною (Domain>Range)
✓	Залежно_від_параметрів (Domain>Range)
✓	Залежно_від_поведінки_системи (Domain>Range)
✓	Залежно_від_цілей_моделювання (Domain>Range)

Рис. 2.12. Типи об'єктних відношень, які реалізує онтологія

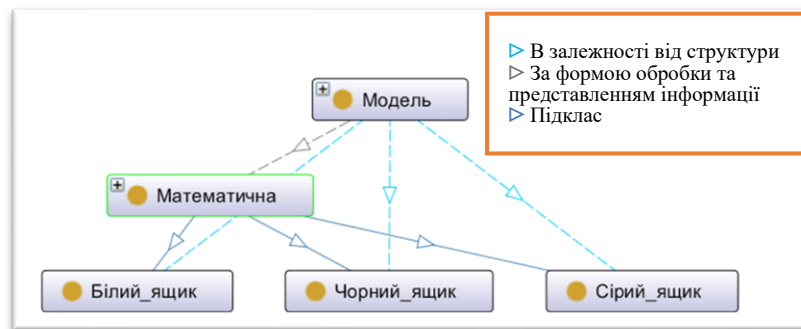


Рис. 2.13. Гілка типів моделей в залежності від структури

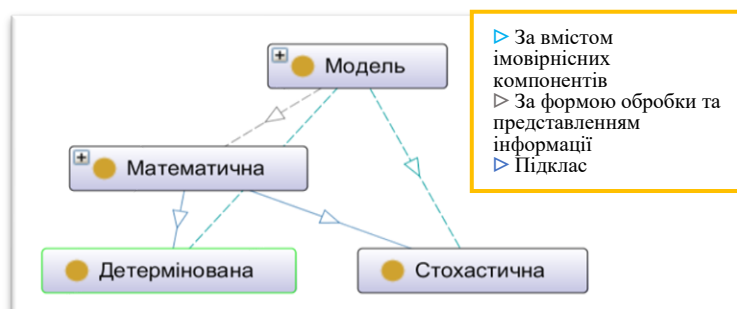


Рис. 2.14. Гілка типів моделей в залежності від вмісту ймовірнісних компонентів

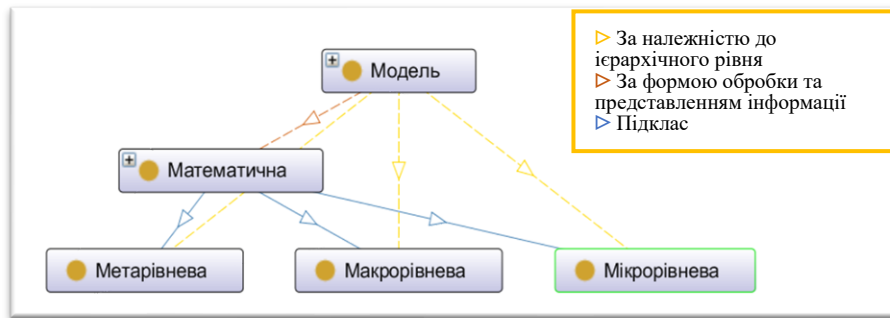


Рис. 2.15. Гілка типів моделей в залежності від ієрархічного рівня

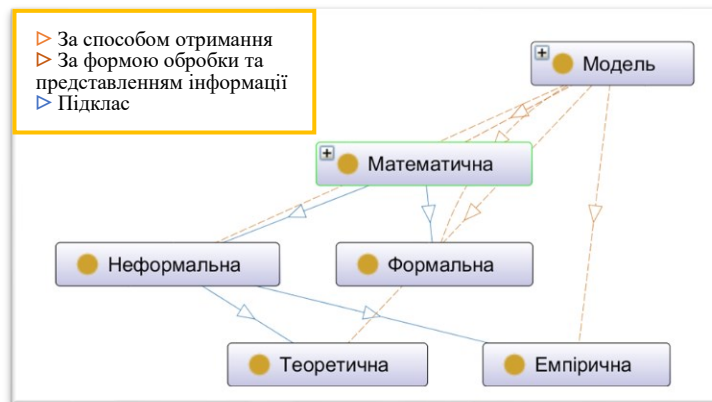


Рис. 2.16. Гілка типів моделей в залежності від способу отримання

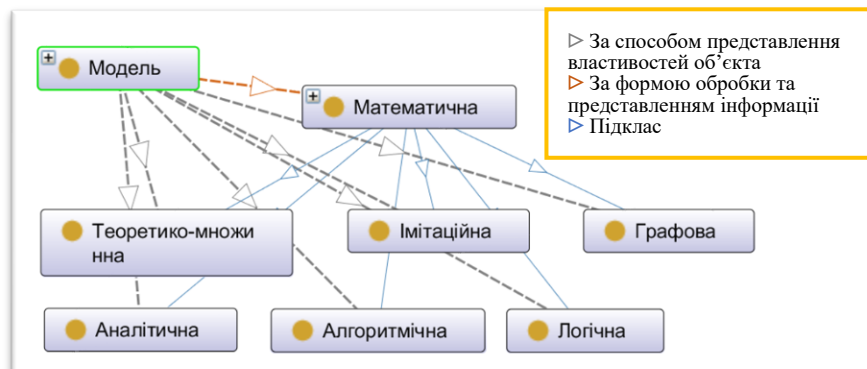


Рис. 2.17. Гілка типів моделей в залежності від способу представлення властивостей об'єкта

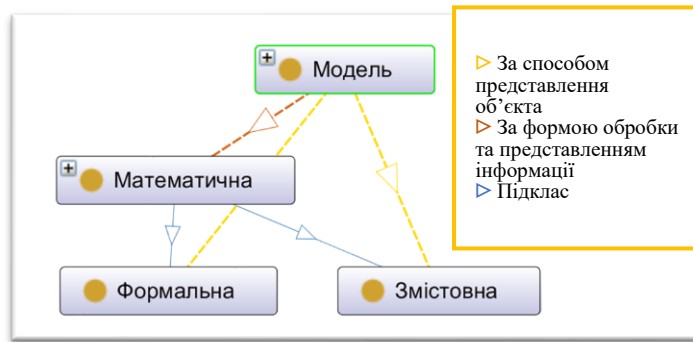


Рис. 2.18. Гілка типів моделей в залежності від способу представлення об'єкта

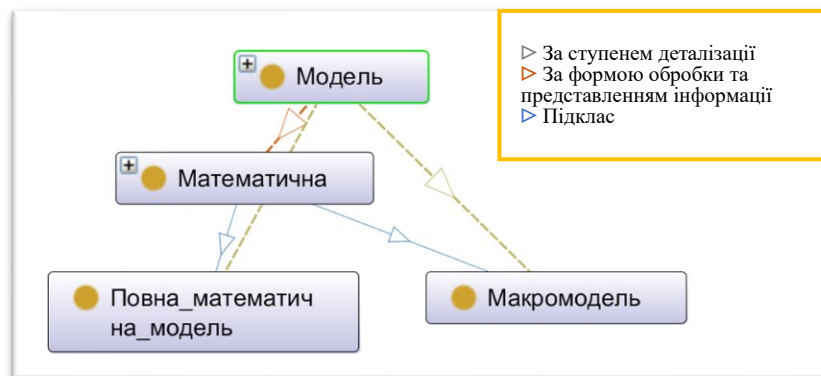


Рис. 2.19. Гілка типів моделей в залежності від ступеню деталізації

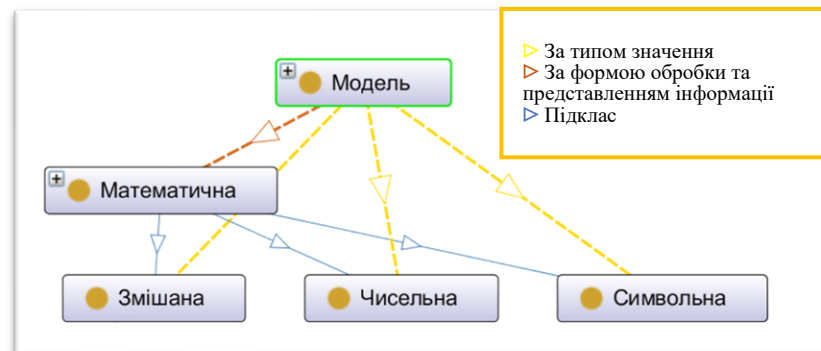


Рис. 2.20. Гілка типів моделей в залежності від типу значення

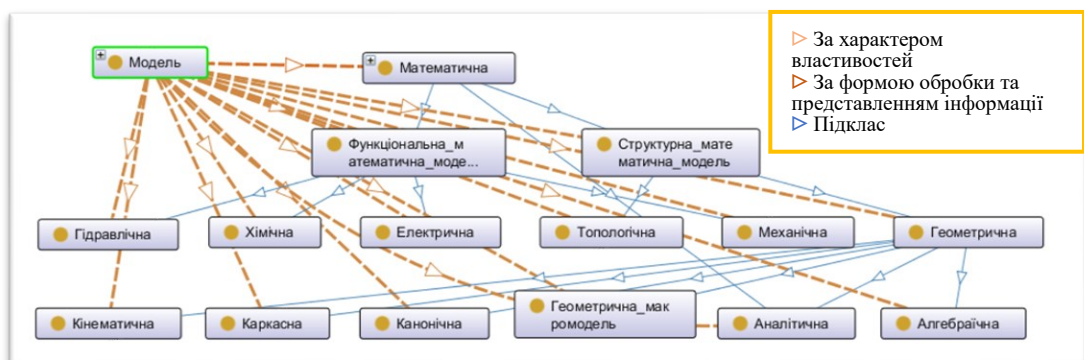


Рис. 2.21. Гілка типів моделей в залежності від характеру властивостей

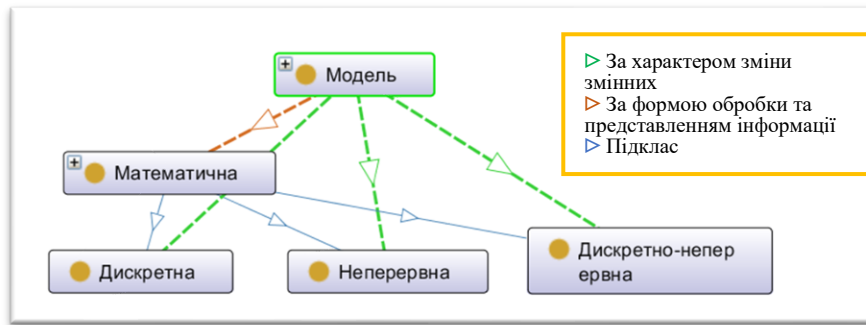


Рис. 2.22. Гілка типів моделей в залежності від характеру зміни змінних

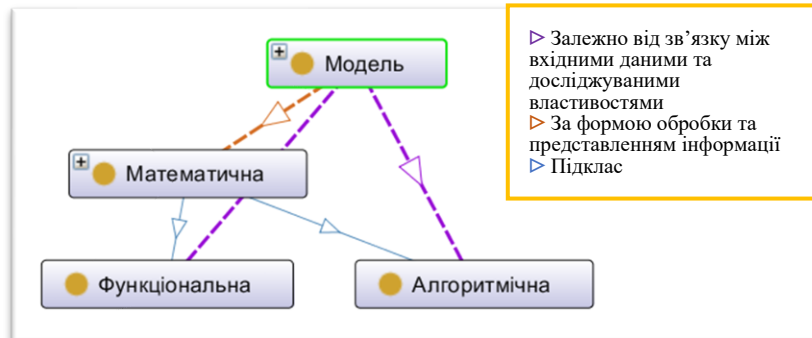


Рис. 2.23. Гілка типів моделей в залежності від зв'язку між вхідними даними та досліджуваними властивостями

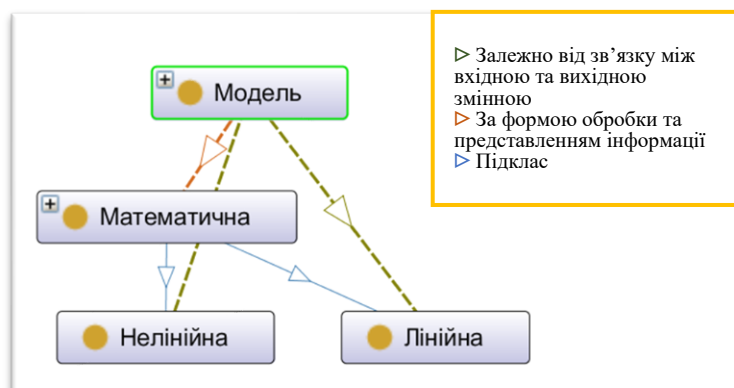


Рис. 2.24. Гілка типів моделей в залежності від зв'язку між вхідною та вихідною змінною

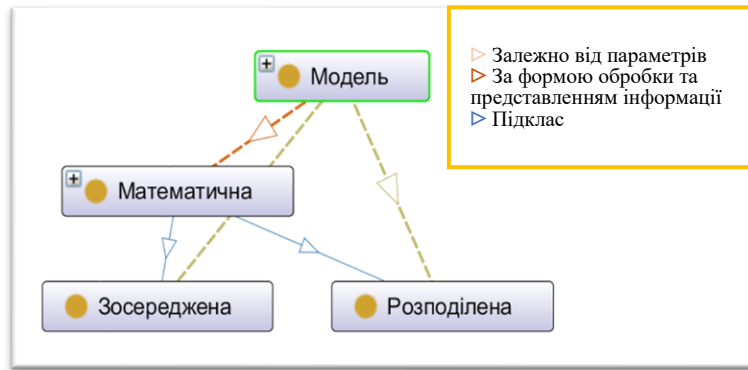


Рис. 2.25. Гілка типів моделей в залежності від параметрів

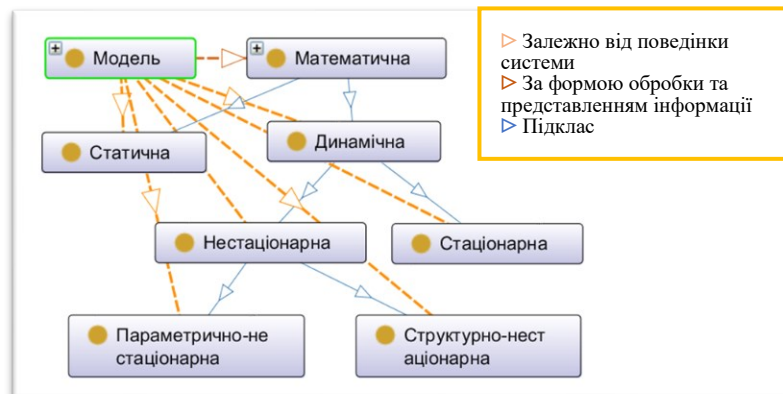


Рис. 2.26. Гілка типів моделей в залежності від поведінки системи

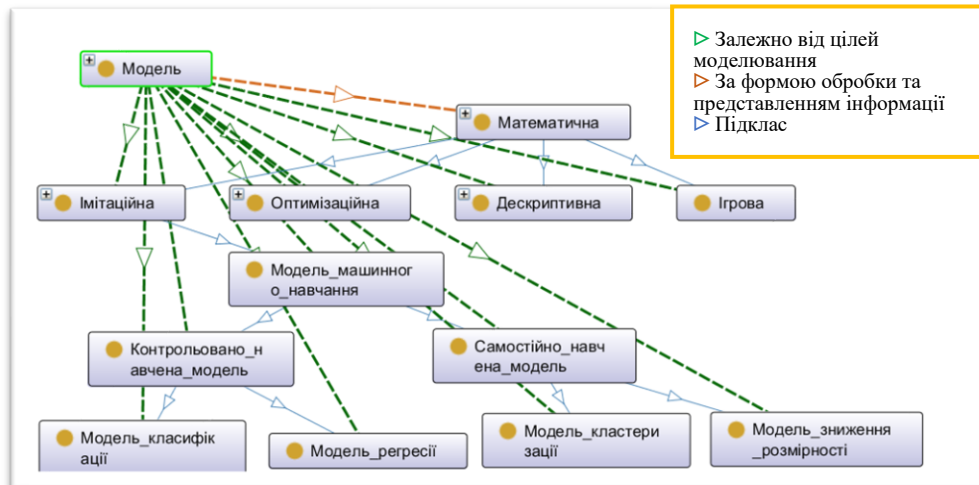


Рис. 2.27. Гілка типів моделей в залежності від цілей моделювання

2.2.4 Аналіз побудованої онтології

Застосування оцінки класичних критеріїв [127–129] дає можливість проаналізувати побудовану онтологію. Серед таких критеріїв оцінки онтології можна виділити:

1. *Узгодженість.*
2. *Класифікацію.*
3. *Тривалість обчислення.*
4. *Повноту.*
5. *Стислість.*
6. *Чутливість.*

Узгодженість оцінює логічну узгодженість онтології шляхом перевірки інваріантів. Перевірка на узгодженість була виконана за допомогою OntoDebug плагіну, відповідно до висновку якого в запропонованій онтології відсутні суперечливі знання, визначення, аксіоми та логічні висновки (Рис. 2.28).



Рис. 2.28. Результат аналізу плагіном OntoDebug

Класифікація означає обчислення всіх супутніх підстановок класів між названими класами та задовольняє розробленій онтології.

Тривалість обчислення є індикатором складності та масштабності онтології. Для онтології математичних моделей середня тривалість обчислення становить близько 400 мс., що вказує на можливість подальшого доповнення онтології без значного підвищення навантаження на роботу резонера.

Онтологія називається *повною*, якщо вся викладена інформація чітко визначена або може бути виведена з інших визначень та аксіом. Дана онтологія розроблена для використання в якості бази знань для системи підтримки прийняття рішень щодо моделей, які використовуються в межах технологічних об'єктів промислових підприємств. Відповідно до цієї цілі, вона є неповною в плані моделей, які не застосовуються до технологічних об'єктів та може бути доповнена специфічними моделями в залежності від області застосування.

Стислість онтології визначається наявністю в її складі зайвих термінів. Як зазначалось вище, в процесі розробки онтології надавалася перевага стислості та практичності онтології зі структурою, яка дозволяє за потреби розширювати базу знань.

Онтологія вважається *чутливою*, якщо мінімальні зміни у визначенні безпосередньо впливають на набір когерентних понять і чітко визначених відносин. З цієї точки зору онтологія математичних моделей є нечутливою, оскільки її структура орієнтована на додавання нових визначень, а зміна набору концепцій впливає на обмежену кількість класів завдяки ієрархічній побудові класифікації.

2.2.5 Використання онтології

Основним способом взаємодії з онтологією є запити (query). Сформовані запити до онтології отримують відповідь, що задовольняє обмеженням, відповідно до вказаних значень Data Property (Рис. 2.29).

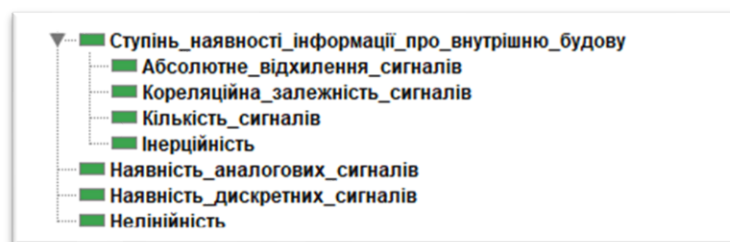


Рис. 2.29. Property для запиту до онтології

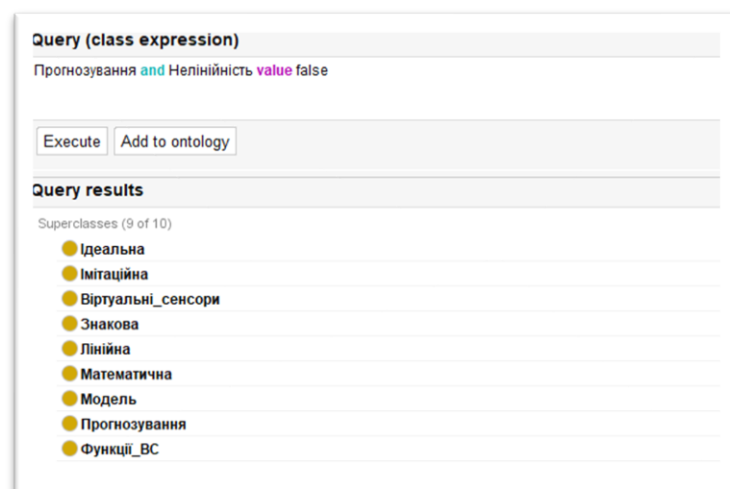


Рис. 2.30. Приклад простого параметризованого запиту до онтології

При збільшенні кількості параметрів запиту онтологія пропонує більш комплексні математичні моделі (Рис. 2.31, 2.32).

Таким чином, виконання запитів до онтології математичних моделей дозволяє отримати рекомендації щодо варіантів вибору моделей в залежності від особливостей досліджуваної системи.

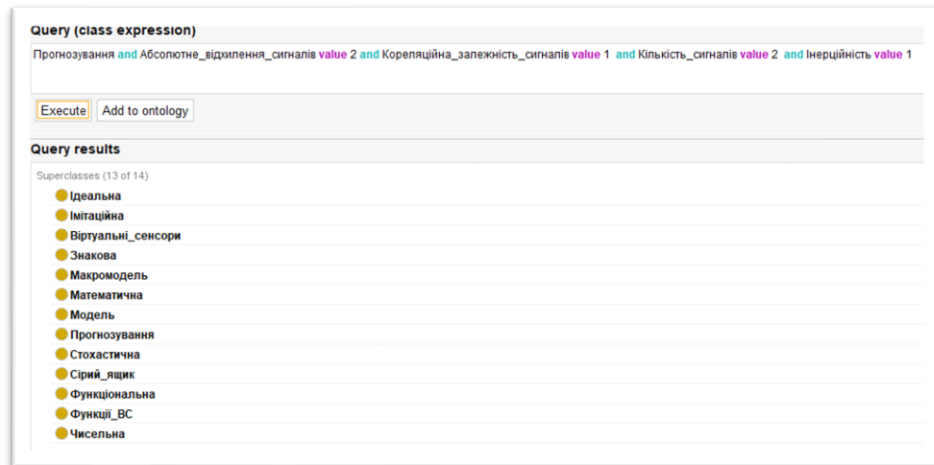


Рис. 2.31. Приклад комплексного запиту до онтології – рекомендація використання моделі Сірого ящика

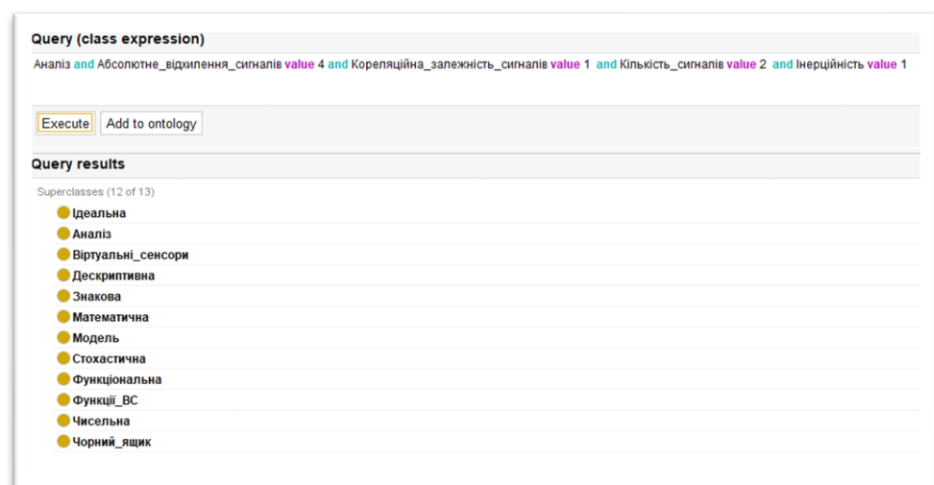


Рис. 2.32. Приклад комплексного запиту до онтології – рекомендація використання моделі Чорного ящика

2.2.6 Місце віртуальних сенсорів в онтології математичних моделей

Особливості віртуальних сенсорів

Віртуальні сенсори представляють собою програмний рівень, який забезпечує непрямі вимірювання змінної процесу або абстрактної умови на основі даних, зібраних фізичними (або іншими віртуальними) датчиками, завдяки використанню функції синтезу. Дані від об'єкта надходять до віртуального сенсору завдяки передачі даних вимірювань за допомогою фізичних сенсорів, або інших віртуальних сенсорів. Завдяки функції синтезу (об'єднання) отримані дані перетворюються у бажану вихідну змінну чи інформацію. Щоб забезпечити кінцевий результат вимірювання функція синтезу може застосовувати до вихідних даних такі методи як масштабування, фільтрацію, лінеаризацію, агрегацію, згладжування, екстраполяцію та інші. Дані, отримані за допомогою функції об'єднання, представляють дані віртуального датчика. Ці дані часового ряду можуть бути будь-якого типу та форми, але мають бути безпосередньо пов'язані з спостережуваним об'єктом, або його частинами. Для збереження цих даних потрібен цифровий двійник. Цифровий двійник — це віртуальний аналог об'єкту (активу), який можна використовувати для цифрового відображення та постійного керування ним. Він поєднує та інтегрує джерела даних активу та контролює його доступність і дійсність. Це включає надання метаданих, семантики та контекстної інформації, яка перетворює дані в інформацію. Крім того, цифровий двійник забезпечує необхідні інтерфейси між віртуальним і реальним світом і забезпечує двонаправлений обмін даними. Таким чином, віртуальні сенсори можуть служити як джерелами даних для цифрових двійників, так і їх інтеграторами, оскільки цифровий двійник також є невід'ємною частиною концепції віртуального сенсора (Рис. 2.33) [130].

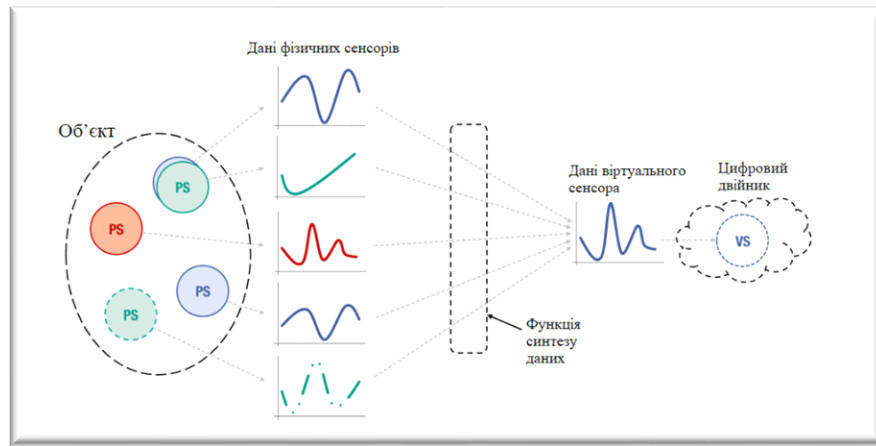


Рис. 2.33. Концепт віртуального сенсору

Віртуальні сенсори та моделі процесів

Моделі процесів і віртуальні сенсори — це два різні підходи, які можуть використовуватися для моніторингу та керування складними системами. Модель процесу — це узагальнене представлення фізичного процесу в системі, яке може використовувати математичні рівняння, симуляції або емпіричні дані для опису її поведінки. Основні відмінності між моделями процесів та віртуальними сенсорами полягають в їхньому призначенні, масштабі, і способі застосування (табл. 2.6).

Таблиця 2.6. Основні відмінності між моделями процесів та віртуальними сенсорами

Характеристика	Моделі процесів	Віртуальні сенсори
Масштаб	Вся система або значна її частина	Окремий параметр або показник
Призначення	Моделюють загальну поведінку системи	Оцінюють окремі параметри системи
Тип даних	Дані процесу, фізичні/хімічні закони	Дані з фізичних сенсорів або інших віртуальних сенсорів
Спосіб побудови	Фізичні рівняння, емпіричні моделі	Алгоритми обробки даних, кореляції
Швидкість	Може бути складною для реального часу	Швидкий розрахунок у реальному часі
Застосування	Оптимізація та управління процесом	Моніторинг і управління параметрами

Обидва підходи можуть доповнювати один одного: моделі процесів можуть бути використані для створення віртуальних сенсорів, а віртуальні сенсори

можуть підвищувати точність моделей процесів за рахунок більш детальних даних.

Віртуальні сенсори в онтології

Модель є ключовим компонентом в структурі віртуального сенсора і виконує роль ядра, яке перетворює сирі дані з фізичних сенсорів або інших віртуальних сенсорів у корисну інформацію. В структурі віртуального сенсору вона виконує роль функції синтезу, яка об'єднує, обробляє та трансформує дані, отримані від сенсорів. Фактично, модель виконує всі необхідні математичні або алгоритмічні операції для отримання бажаних вимірювань чи інформації про об'єкт.

Вибір моделі безпосередньо залежить від функцій, які віртуальний сенсор має виконувати, таких як аналіз, керування чи прогнозування. Кожна функція накладає свої вимоги до того, як дані повинні оброблятися, які математичні або алгоритмічні методи будуть використані, і як буде отримано кінцевий результат. Тому в структурі онтології математичних моделей віртуальні сенсори пов'язані з моделями, як в залежності від властивостей досліджуваного об'єкта, так і в залежності від функцій, що відображають цілі використання ВС (Рис. 2.34).

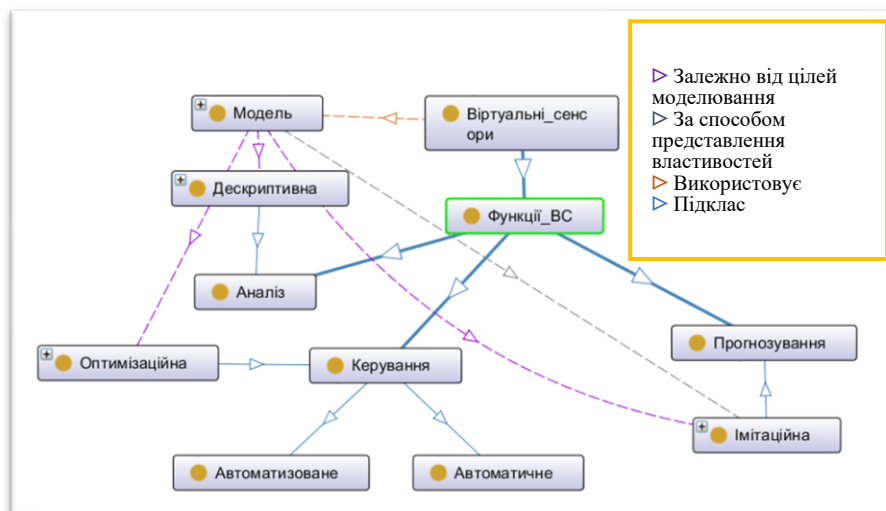


Рис. 2.34 Місце віртуальних сенсорів в онтології

2.2.8 Роль та місце моделі в контексті віртуального сенсора

На основі аналізу розробленої онтології було зроблено висновок про необхідність використання моделей машинного навчання, оскільки вони здатні задовольнити вимогам складності виробництва та забезпечити необхідну швидкість та гнучкість адаптації для реалізації наступних задач в порівнянні з аналітичними моделями:

1. Класифікація стабільності виробничого процесу.
2. Визначення та прогнозування невимірюваних змінних технологічного процесу.
3. М'яке вимірювання змінних для моніторингу стану датчиків та можливості компенсації їх функціоналу в разі збоїв.
4. Оптимізація режимів роботи відділення на основі зворотного моделювання.
5. Автоматичне керування на основі ВС.

Ключовим елементом, що забезпечує функціонування віртуального сенсора, є модель. Від її вибору та налаштувань залежить ефективність його роботи, а також стабільність і надійність результатів, зокрема для нових вхідних даних.

В основі вибору моделі лежать характеристики даних виробничого процесу, їх кількість, якість, наявні залежності між змінними та їх характер. Крім того, важливими є такі фактори, як рівень шуму в даних, складність взаємодії між різними параметрами, специфічні умови роботи системи, в якій модель буде використовуватися. Не менш важливим є врахування можливих змін у процесі чи даних з часом, що може вимагати адаптації моделі чи специфічних налаштувань. Оцінка цих характеристик допомагає вибрати найбільш відповідну модель для досягнення максимальної точності прогнозування та ефективності в умовах реальної експлуатації.

Багато процесів спиртового виробництва характеризуються наявністю складних нелінійних зв'язків між змінними, що разом з неможливістю прямого

вимірювання певних показників створює труднощі для ефективного аналізу роботи системи.

Яскравим прикладом такого процесу є процес ректифікації, який може бути описаний як складний нелінійний процес з численними залежними змінними. Ректифікація передбачає розділення суміші на компоненти з різними температурами кипіння шляхом багаторазового конденсації та випаровування. Цей процес зазвичай включає взаємодію таких факторів як температура, тиск, молекулярний склад і швидкість потоку. Врахування особливостей цих змінних і їхніх взаємозв'язків є необхідним кроком для підбору та налаштування моделі для віртуального сенсора.

2.3 Статистичні та класифікаційні підходи для оцінювання стабільності технологічних процесів

2.3.1 Постановка задачі класифікації

Спиртове виробництво являє собою складний технологічний процес, де кожен етап виробництва залежить від взаємодії множини змінних, таких як температура, тиск, концентрація речовин, швидкість потоків тощо. Навіть перебуваючи у межах технологічних норм, значення цих змінних можуть приховувати комплексні відхилення в просторі станів від стаціонарних режимів функціонування, це може призвести до порушення всієї технології, зниження якості продукції, збільшення витрат сировини або навіть аварійних ситуацій.

Для вирішення проблеми виявлення наявності факторів, що спричиняють нестабільний стан виробництва, за допомогою аналізу взаємозв'язків між параметрами та їх розташування в просторі станів можна застосувати методи статистичного аналізу, для класифікації поточного стану виробництва. Це дозволить виявити закономірності, тренди, залежності і аномалії у даних, а також прийняти обґрунтовані рішення на їх основі, що може бути використано для класифікації, яка ґрунтується на аналізі ймовірностей, статистичних розподілів, залежностей між змінними та математичному моделюванні, з метою оцінки

ймовірності належності об'єкта до певного класу, і на основі цієї оцінки прийняти рішення про класифікацію.

2.3.2 Статистичний аналіз даних роботи дистиляційного відділення

Статистичний аналіз, зокрема його методи було використано на даних роботи спиртзаводу, для виявлення ключові закономірностей та відмінності між режимами. Було визначено статистичні характеристики, такі як середнє, дисперсія, кореляції між різними змінними виробництва, а також виявлено можливі аномалії чи відхилення від нормальних умов. Вони дають змогу зібрати інформацію про типові патерни для кожного з режимів роботи, що дозволяє порівняти їх та з'ясувати, які фактори сприяють переходу між стабільним та нестабільним станом.

Для аналізу були використані архівні дані стабільної та нестабільної роботи дистиляційного відділення спиртового заводу за 12 годин. Візуальне порівняння графіків, залежно від змінних, показує як видиму різницю, так і її відсутність (Рис. 2.35-2.36).

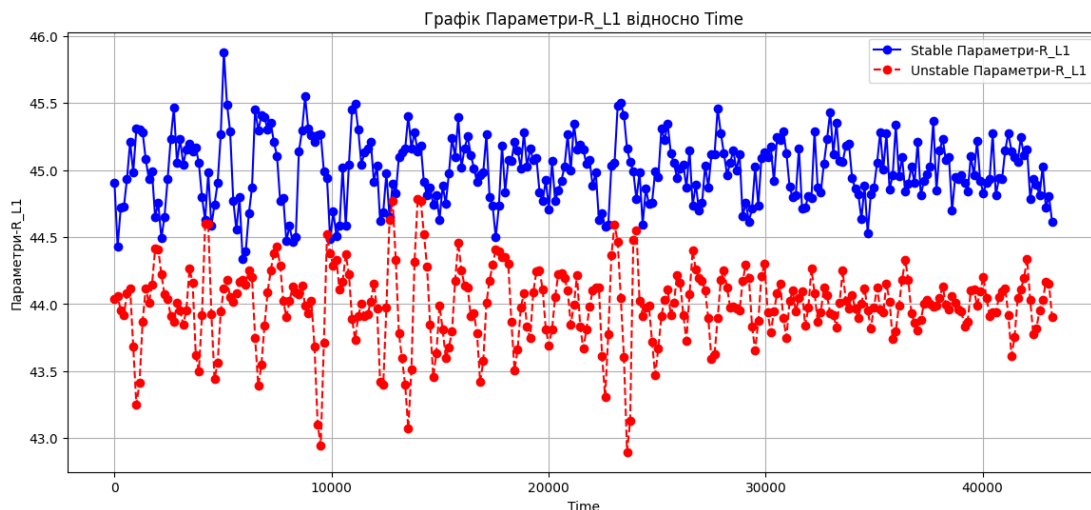


Рис. 2.35. Порівняння графіків параметрів R_L1 для стабільної та нестабільної вибірок

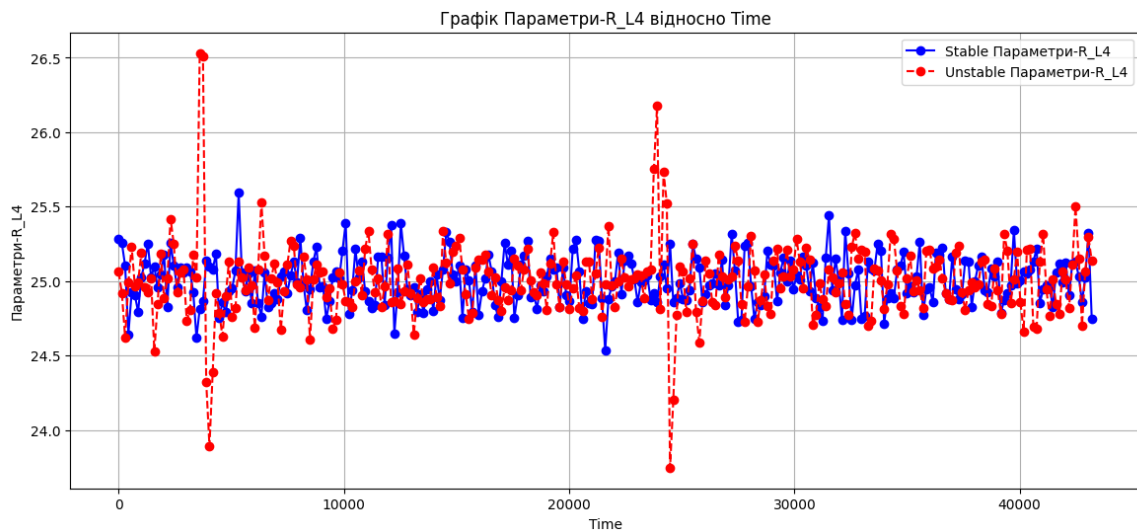


Рис. 2.36. Порівняння графіків параметрів R_L4 для стабільної та нестабільної вибірок

2.3.3 Описова статистика даних

Було використано методи описової статистики для підсумовування, опису й узагальнення даних та отримання основних характеристик набору даних без необхідності аналізу всього масиву. Цей базовий крок в аналізі даних, дозволяє зрозуміти їх структуру, варіації й тенденції.

Використання методів описової статистики дозволило отримати наступну інформацію:

- Середнє значення (mean): Показує середнє значення для кожного параметра за всіма спостереженнями.
- Стандартне відхилення (std): Відображає, наскільки дані розподілені навколо середнього значення. Високе стандартне відхилення означає високу мінливість у даних.
- Мінімальне та максимальне значення (min, max): Показує діапазон значень для кожного параметра.
- Квартилі (25%, 50%, 75%): Показують розподіл даних. Медіана (50%) для Параметри-R_L1 дорівнює 45.015032, що означає, що половина значень цього параметра менша, а половина — більша.

Описова статистика дозволила зрозуміти основні тенденції, варіації та розподіл даних. Наприклад, високі стандартні відхилення для деяких параметрів

свідчать про їхню нестабільність. Порівняння поведінки різних параметрів може вказувати на їх різну роль у процесі (Рис. 2.37).

Описова статистика для стабільного процесу:

	Time	Параметри- R_L1	Параметри- R_L2	Параметри- R_L3	Параметри- R_L4	Параметри- R_L5	Параметри- R_L7	Параметри- R_L8	Параметри- R_L6p	Параметри- R_T1	Параметри- R_T2	Параметри- R_T3	Параметри- R_T4
mean	21600.000000	44.996449	39.991060	75.000321	25.005603	114.113785	35.000026	49.998594	30.016442	178.433194	74.705283	81.651302	78.319602
std	12533.102409	0.245946	1.158173	0.372364	0.162055	0.000000	0.140228	0.058304	0.327608	8.844292	0.125207	0.046026	0.078480
min	0.000000	44.338432	36.826301	74.183350	24.531237	114.113785	34.454652	49.814845	28.838427	156.798885	74.480687	81.540848	77.986304
25%	10800.000000	44.819227	38.894109	74.701791	24.885933	114.113785	34.923853	49.962330	29.808238	174.844913	74.615946	81.615808	78.277203
50%	21600.000000	45.015032	40.183529	75.015777	25.000058	114.113785	35.009915	49.995318	30.000961	181.591514	74.680234	81.644464	78.320832
75%	32400.000000	45.161130	40.996643	75.307156	25.117167	114.113785	35.082803	50.031836	30.209209	184.693691	74.779935	81.677385	78.358833
max	43200.000000	45.881072	42.044929	75.786278	25.594535	114.113785	35.440461	50.237728	31.108289	190.504534	75.166251	81.748122	78.545978

Описова статистика для нестабільного процесу:

	Time	Параметри- R_L1	Параметри- R_L2	Параметри- R_L3	Параметри- R_L4	Параметри- R_L5	Параметри- R_L7	Параметри- R_L8	Параметри- R_L6p	Параметри- R_T1	Параметри- R_T2	Параметри- R_T3	Параметри- R_T4
mean	21600.000000	43.993528	40.006034	79.999157	25.002971	44.999750	40.456879	34.000253	45.141011	178.878274	79.363572	85.667032	82.471754
std	12533.102409	0.283765	0.520769	0.274394	0.260481	0.032837	0.920059	0.101671	2.278830	2.374940	0.788662	0.052938	0.113862
min	0.000000	42.896590	37.539836	79.067620	23.746994	44.909038	39.937957	33.589147	35.952826	173.820390	76.845222	85.572925	82.177876
25%	10800.000000	43.860121	39.776679	79.796742	24.868010	44.982029	39.989702	33.941632	44.190263	177.240973	78.981155	85.620202	82.403751
50%	21600.000000	44.008080	39.992465	80.006000	25.003785	44.998589	40.015980	33.993688	45.110630	178.620359	79.451303	85.661531	82.476745
75%	32400.000000	44.154210	40.242799	80.173657	25.128789	45.017866	40.981921	34.061932	46.183778	180.029224	79.808116	85.710379	82.541972
max	43200.000000	44.781860	42.572794	81.571844	26.531151	45.132637	49.058111	34.296824	51.547501	185.681573	82.187582	85.801067	82.741262

Рис. 2.37. Фрагмент результатів обчислення базових статистик (mean, std, min, max) для стабільного та нестабільного процесів

2.3.4 Порівняльні гістограми

Візуальне порівняння гістограм дозволяє побачити як значення змінних розподілені в стабільному та нестабільному станах. Значна різниця в розподілах може мати високий вплив на прогнозування, наприклад, для класифікації стабільного/нестабільного стану (Рис. 2.38).

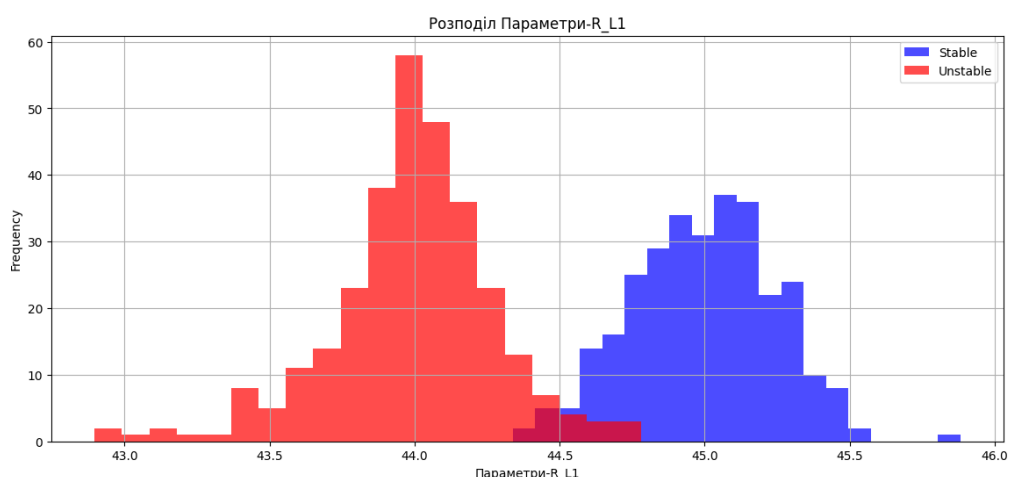


Рис. 2.38. Порівняння гістограм розподілу змінної R_L1 для стабільного та нестабільного процесів

2.3.5 Перевірка нормальності розподілу даних для кожної змінної у двох підмножинах даних за допомогою тесту Шапіро-Вілка

Аналіз нормальності є корисним для розуміння основних характеристик виробничих даних, діагностики процесу та вибору правильних статистичних інструментів. Було проведено тест Шапіро-Вілка для перевірки на нормальний розподіл в змінних процесу (Рис. 2.39):

- Якщо $p\text{-value} < 0.05$: розподіл не є нормальним.
- Якщо $p\text{-value} \geq 0.05$: розподіл можна вважати нормальним.

Для кожної змінної побудовано два типи графіків:

- **Гістограми з лінією KDE (оцінка щільності ядра):** Побудовані окремо для стабільного та нестабільного станів. Дає змогу візуально оцінити, чи змінна має симетричний розподіл (що характерно для нормального розподілу) (Рис. 2.40).

- **Q-Q графіки (Quantile-Quantile):** Порівняно квантилі даних із квантилями нормального розподілу. Якщо точки на графіку лягають на пряму лінію, розподіл наближається до нормального (Рис. 2.41).

Variable	Stable p-value	Unstable p-value	Variable	Stable p-value	Unstable p-value	Variable	Stable p-value	Unstable p-value
Параметри-R_L1	0.477745	0.000001	Параметри-R_T11	0.381334	0.000000	Параметри-R_P9	0.154392	0.000000
Параметри-R_L2	0.000000	0.000000	Параметри-R_T12	0.000053	0.000001	Параметри-R_P10	0.000503	0.000000
Параметри-R_L3	0.000457	0.000007	Параметри-R_T13	0.000000	0.000000	Параметри-R_P11	0.000000	0.000001
Параметри-R_L4	0.277893	0.000000	Параметри-R_T14	0.000000	0.387895	Параметри-R_P12	0.000000	0.000000
Параметри-R_L5	1.000000	0.000000	Параметри-R_T15	0.250301	0.142856	Параметри-R_P13	0.000000	0.000000
Параметри-R_L7	0.000013	0.000000	Параметри-R_T16	0.000000	0.000000	Параметри-R_P14	0.000399	0.000000
Параметри-R_L8	0.004564	0.000106	Параметри-R_T17	0.000169	0.000000	Параметри-R_P15	0.0008952	0.000000
Параметри-R_L6p	0.000005	0.000000	Параметри-R_T18	0.000000	0.000000	Параметри-R_P16	0.007806	0.000000
Параметри-R_T1	0.000000	0.000006	Параметри-R_T19	0.000031	0.000010	Параметри-R_P17	0.001100	0.000004
Параметри-R_T2	0.000000	0.000000	Параметри-R_T20	0.002134	0.011380	Параметри-R_P19	0.000191	0.000000
Параметри-R_T3	0.000001	0.000000	Параметри-R_P1	0.000000	0.001424	Параметри-R_F1	0.000000	0.000003
Параметри-R_T4	0.000041	0.036536	Параметри-R_P2	0.000000	0.000000	Параметри-R_F2p	0.077106	0.000047
Параметри-R_T5	0.000029	0.004176	Параметри-R_P3	0.000000	0.000000	Параметри-R_F3p	0.000000	0.000000
Параметри-R_T6	0.000000	0.000000	Параметри-R_P4	0.000000	0.000000	Параметри-R_T10.1	0.163881	0.000000
Параметри-R_T7	0.000000	0.000000	Параметри-R_P5	0.000516	0.000000	Параметри-R_Ef4	0.000336	0.000000
Параметри-R_T8	0.000000	0.000000	Параметри-R_P6	0.000030	0.000000	Параметри-R_L2.1	0.000000	0.000000
Параметри-R_T9	0.000000	0.000000	Параметри-R_P7	0.011455	0.000000			
Параметри-R_T10	0.183881	0.000000	Параметри-R_P8	0.000000	0.000000			
Параметри-R_T11	0.381334	0.000000	Параметри-R_P9	0.154392	0.000000			

Рис. 2.39. Результат тесту Шапіро-Вілка на нормальний розподіл для стабільного та нестабільного процесів (зелений колір вказує на нормальний розподіл, червоний – ненормальний розподіл)

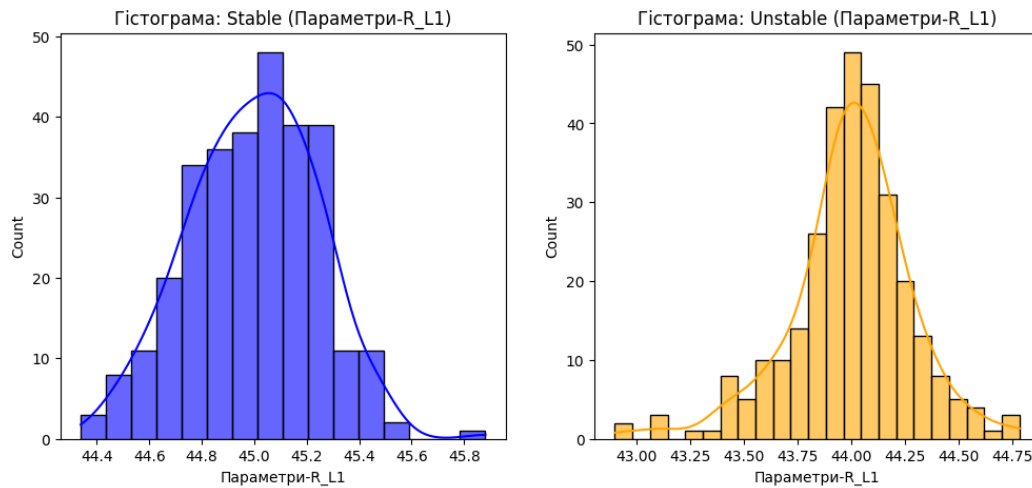


Рис. 2.40. Порівняння гістограм розподілу змінної R_{L1} з лінією KDE для стабільного та нестабільного процесів

Багато статистичних методів (ANOVA, t-тест, регресійні моделі) передбачають, що дані мають нормальний розподіл. Аналіз нормальності дозволяє визначити, чи можна застосовувати ці методи до виробничих даних. Також аналіз на нормальність допомагає виявити статистичні відмінності та аномалії між стабільним і нестабільним станами.

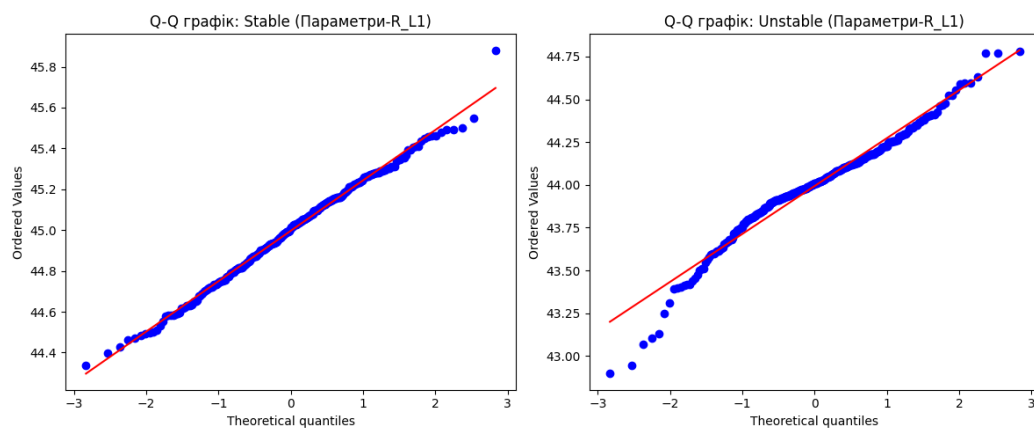


Рис. 2.41. Q-Q графік. Порівняння квантилів даних змінної R_{L1} із квантилями нормального розподілу для стабільного та нестабільного процесів

2.3.6 Матриці кореляції

Для визначення статистичних зв'язків (кореляції) між різними змінними у наборі даних було побудовано матриці кореляції. Матриця кореляції відображає, наскільки сильно зміни однієї змінної пов'язані зі змінами іншої змінної, та

дозволяє побачити, які змінні в кожному наборі даних тісно пов'язані між собою (кореляція близька до 1 або -1) і які змінні не мають значущого зв'язку (кореляція близька до 0). Якщо кореляція між двома змінними близька до 0, це вказує на те, що зміни однієї змінної не сильно пов'язані з іншою змінною. У регресії або інших моделях, коли дві змінні мають високу кореляцію, це може вказувати на мультиколінеарність, що може погіршити якість моделі.

В роботі наведено матриці кореляції для нестабільного (Рис. 2.42) та стабільного (Рис. 2.43) процесів. Їх порівняння дозволяє побачити відмінності в кореляціях між процесами.

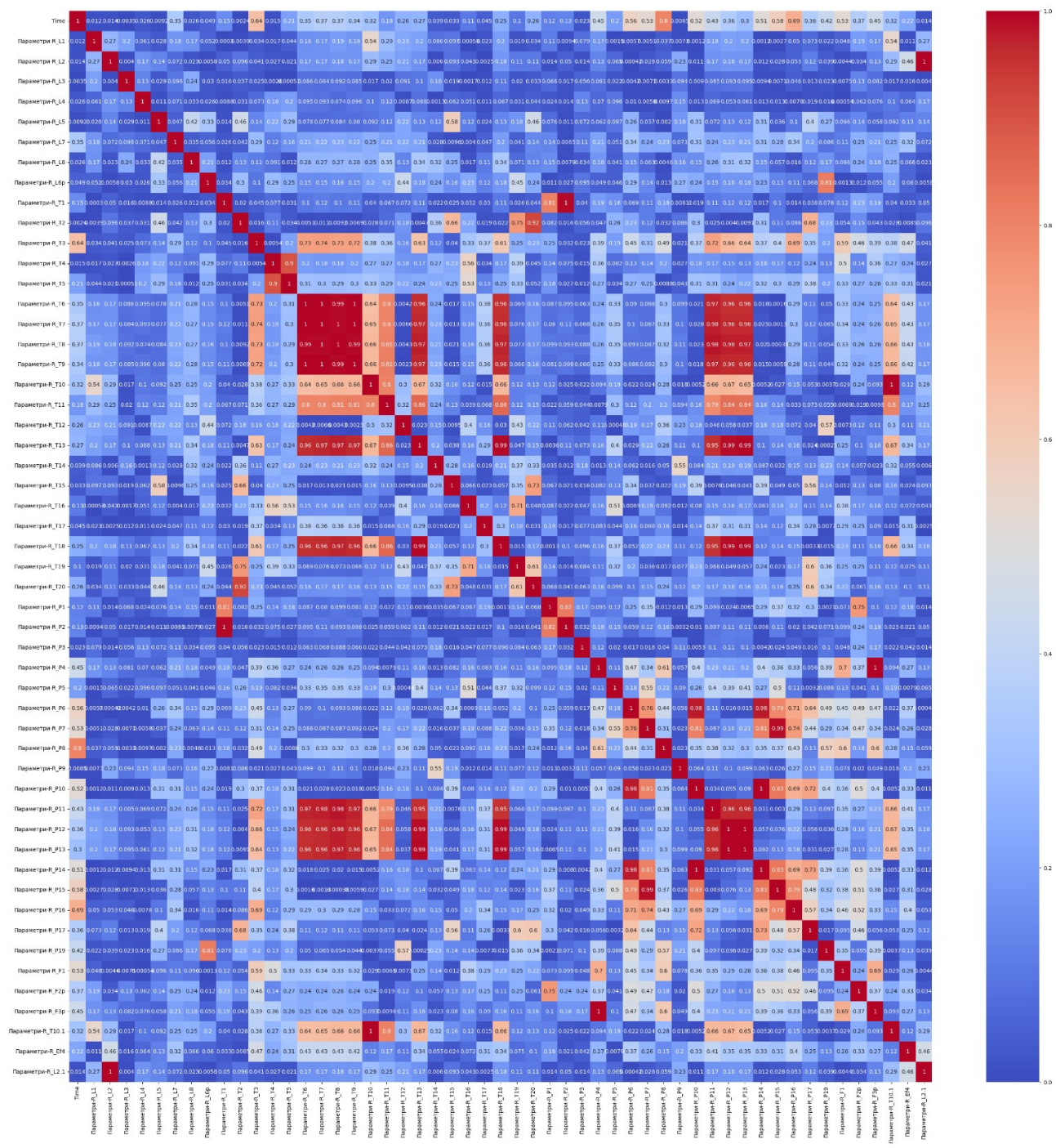


Рис. 2.42. Матриця кореляції для нестабільного процесу



2.3.7 Boxplots та violin plots для порівняння розподілів між групами

Для виявлення різниці у статистичних характеристиках даних (медіана, розмах, наявність викидів) було використано ящиківі діаграми Boxplot :

- Медіана: Boxplot показує лінію всередині кожного "ящика", яка являє собою медіану (центральне значення) для кожної групи даних. Якщо медіана в даних суттєво різниться, це може вказувати на значні відмінності між розподілами в різних умовах.
- Міжквартильний діапазон (IQR): Верхня і нижня межі ящика вказують на 75-й і 25-й процентилі (Q3 та Q1), тобто на те, де зосереджена основна маса даних. Візуально це дозволяє зрозуміти, наскільки розподіли даних перекриваються.
- Викиди: Boxplot також показує викиди, які є точками, що знаходяться за межами "вусів" (границь, визначених за межами 1.5 IQR від меж ящика). Якщо один з розподілів має більше викидів, це може свідчити про те, що дані нестабільніші або мають аномальні значення.

Використання Boxplot дозволяє краще зрозуміти структуру даних і вирішити, чи потрібна додаткова обробка (наприклад, усунення викидів або трансформація даних) для покращення результатів моделей або подальших аналізів. Так порівняння розподілів змінних R-L1 та R-L2 для стабільного та нестабільного процесів показують значну різницю в ширині розподілу та кількості викидів (Рис. 2.44). Наявність значної кількості викидів в даних нестабільного процесу порівняно зі стабільним можуть вказувати на аномалію та допомагати ідентифікувати цей стан.

Violin plot (скрипкові діаграми) поєднують характеристики boxplot з додатковою інформацією про щільність розподілу даних. Лінії "крила" діаграми показують, як багато значень є на кожному рівні, що допомагає оцінити, чи є розподіл симетричним, чи має відмінності на різних ділянках. Це дозволяє оцінити форми розподілу, що не видно у звичайному boxplot.

У violin plots відображаються не тільки медіана та квартилі (як у boxplot), але й повний спектр варіацій та щільності даних. Це дозволило зрозуміти, чи є значущі відмінності в варіативності між двома групами. Звуження розподілу на одній зі сторін може вказувати на концентрацію даних в певному діапазоні.

Violin plot також показує асиметричні розподіли, де одна частина розподілу може бути вище за іншу (правостороння або лівостороння асиметрія). Це є важливим для вибору методів обробки даних і для подальшого аналізу.

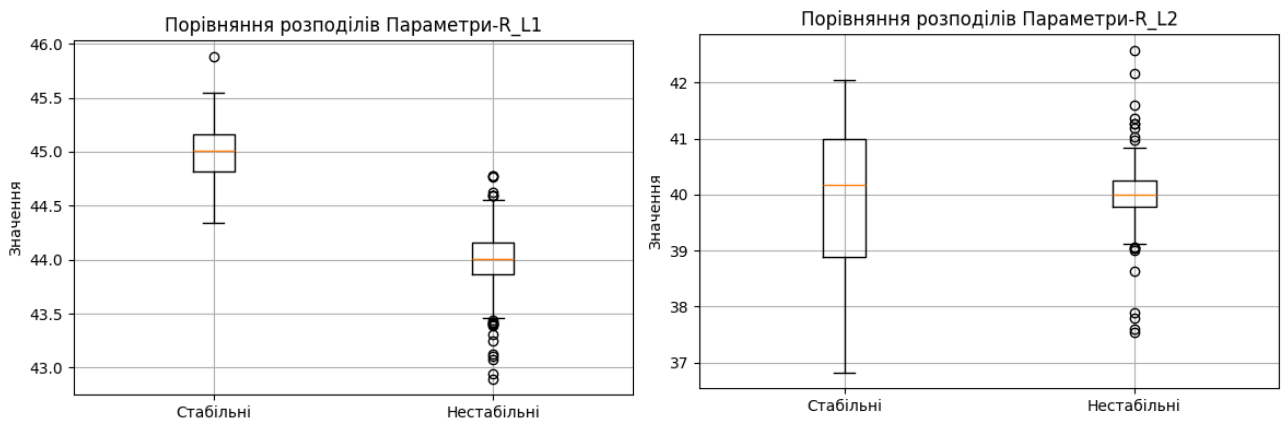


Рис. 2.44. Boxplot порівняння розподілів змінних R-L1 та R-L2 для стабільного та нестабільного процесів

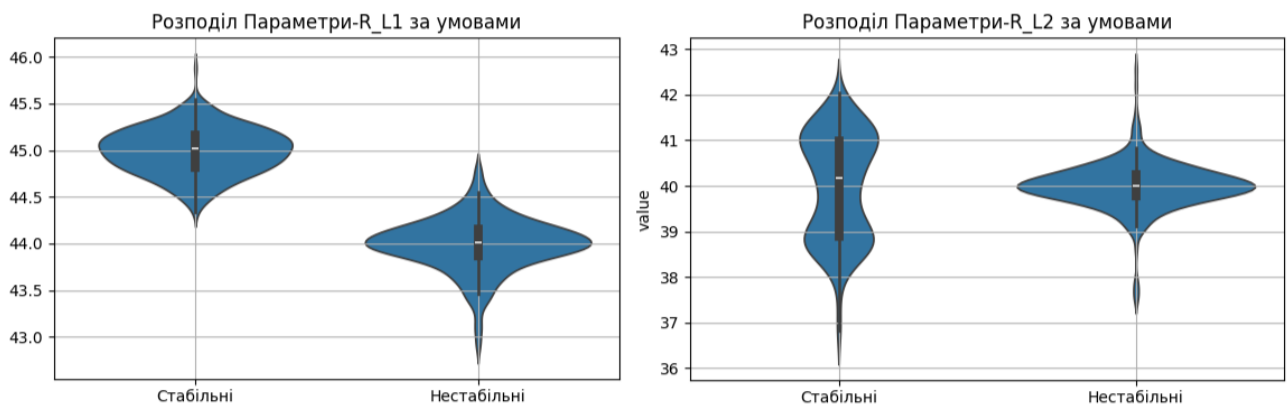


Рис. 2.45. Violin plot порівняння розподілів змінних R-L1 та R-L2 для стабільного та нестабільного процесів

2.3.8 Дисперсійний аналіз ANOVA

Для визначення наявності значущих відмінностей між групами для кожної змінної було виконано дисперсійний аналіз (analysis of variance (ANOVA)) - статистичний метод, який оцінює вплив якісних ознак на результати. Цей підхід базується на використанні F-статистики та р-значень для перевірки наявності суттєвих відмінностей між групами:

- F-статистика вимірює, чи існують значущі відмінності між середніми значеннями стабільних і нестабільних умов для кожної змінної. Великі значення F-статистики свідчать про значущі відмінності між групами.
- р-значення дозволяє оцінити статистичну значущість результату. Якщо р-значення менше за рівень значущості (зазвичай 0.05), то можна стверджувати, що є значущі відмінності між середніми групами.

Результати тестів для стабільного та нестабільного процесів приведені у вигляді таблиць та стовпчикових діаграм (з логарифмічною шкалою) (Рис. 2.46-2.47).

F-статистика: Для кожної змінної побудовано стовпчикову діаграму з логарифмічною шкалою (Рис. 2.46). Логарифмічна шкала дозволяє краще порівняти значення F-статистики, особливо якщо вони значно відрізняються між змінними. Лінія на рівні критичного значення F є умовною межею, що вказує на значущі відмінності. Статистика F є відношенням варіації між групами до варіації всередині груп: Велике значення F вказує на те, що варіація між групами більша, ніж варіація всередині кожної групи, тобто між групами є суттєві відмінності. Це підтримує гіпотезу про наявність значущого ефекту.

р-значення: Побудовано окрему діаграму для р-значень із логарифмічною шкалою (Рис. 2.47). Лінія на рівні 0.05 показує поріг для статистичної значущості. р-значення менше 0.05 означає значущі відмінності між групами.

Якщо р-значення менше за 0.05 (поріг статистичної значущості), це означає, що ймовірність випадковості цих відмінностей дуже мала (менше ніж 5%). Таким чином, ми можемо стверджувати, що відмінності між стабільними та нестабільними умовами не випадкові.

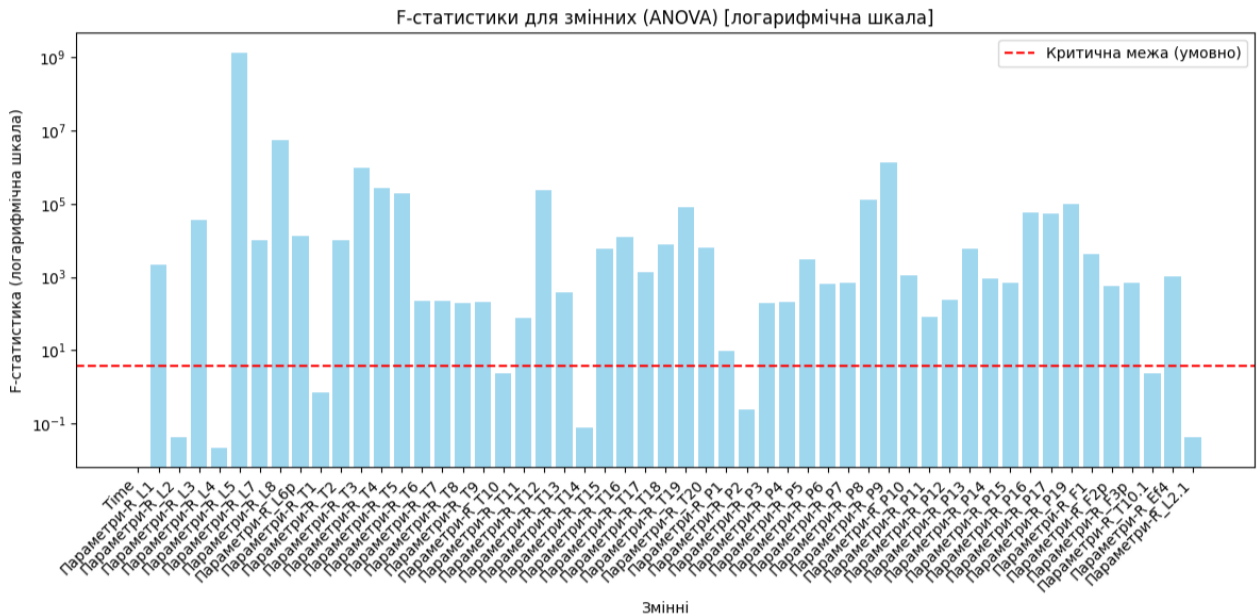


Рис. 2.46. Стовпчикова діаграма для Ф-статистики (з логарифмічною шкалою)

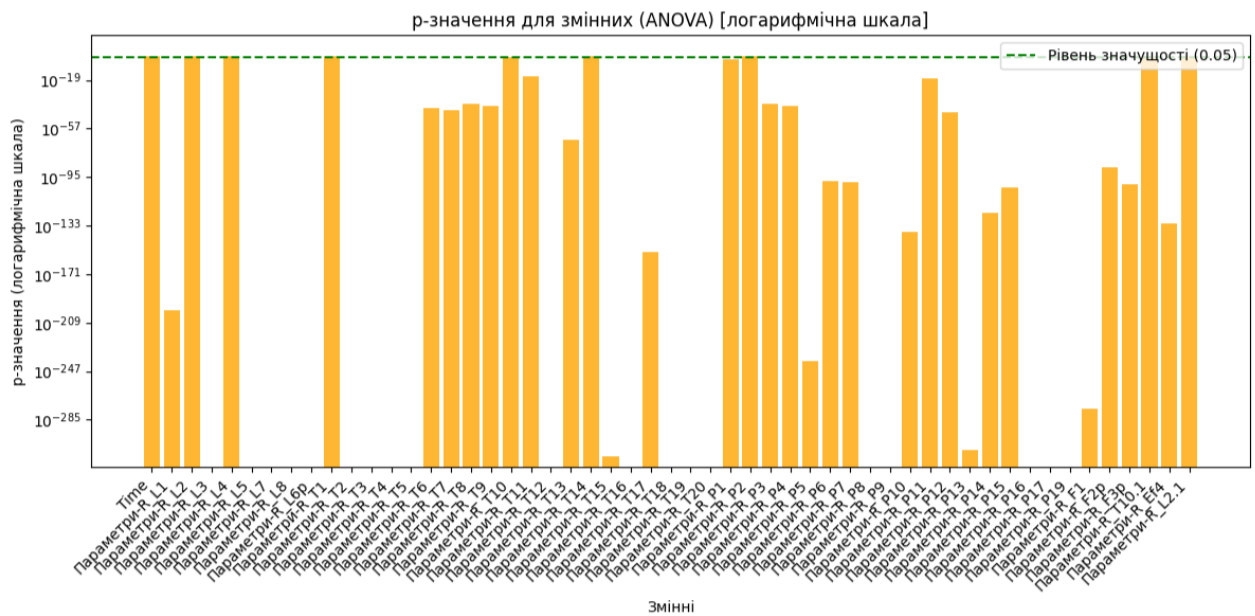


Рис. 2.47. Стовпчикова діаграма для р-значень (з логарифмічною шкалою)

Дисперсійний аналіз даних стабільного та нестабільного процесів вказує на наявність суттєвих відмінностей не випадкової природи між вибірками.

2.3.9 Kernel Density Estimate (перевірка на мультимодальність)

Було проведено оцінку ядра густини (Kernel Density Estimate, KDE) для побудови непараметричних оцінок густини розподілу змінних в стабільних та нестабільних умовах. Це дозволило порівняти розподіли та їх відмінності. Червона та синя лінії на графіку показують оцінку густини для нестабільних та стабільних умов відповідно (Рис. 2.48). Порівняння густини ліній вказує на наявність статистично значущих відмінностей у розподілах.

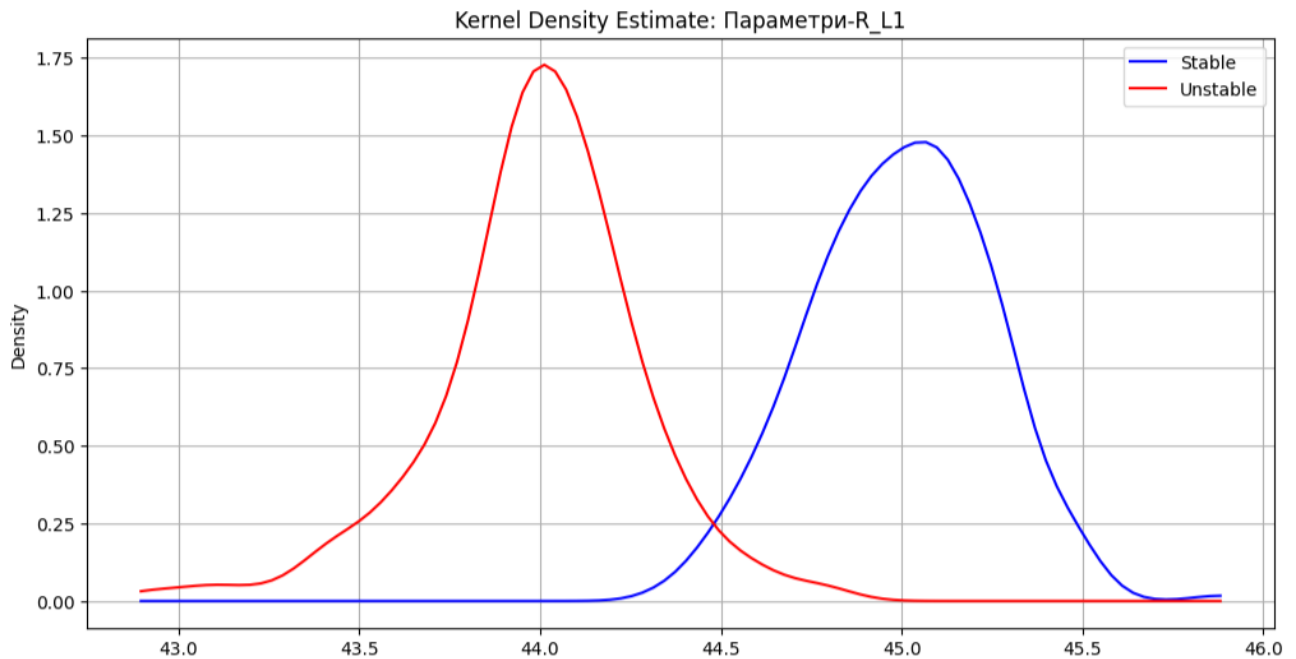


Рис. 2.48. KDE графік для оцінки густини розподілу змінної R-L1 для стабільного та нестабільного процесів

2.3.10 Класифікація стабільності процесу з використанням

Gaussian Mixture Model

На основі проведеного статистичного аналізу було зроблено висновок про статистично значущі відмінності між групами даних. Ці результати вказали на можливість використання Gaussian Mixture Model (GMM) для класифікації виробничих даних на класи стабільного і нестабільного режимів роботи виробництва в режимі реального часу на основі багатовимірних параметрів, які описують стан системи. Це дозволяє оцінювати стабільність системи та визначати прояви нестабільностей на ранніх етапах.

GMM використовує формулу багатовимірного Гаусового розподілу для розрахунку ймовірностей того, що кожна точка належить до певного класу:

$$f(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right), \quad (2.2)$$

де:

- $x \in R^d$ - вектор вхідних даних (точка у d-вимірному просторі).
- $\mu \in R^d$ - середнє значення (вектор, що визначає "центр" розподілу).
- $\Sigma \in R^{d \times d}$ - коваріаційна матриця (визначає форму і розмір розподілу).
- $|\Sigma|$ - визначник коваріаційної матриці.
- Σ^{-1} - обернена матриця до коваріаційної.
- d - кількість вимірів (розмірність простору).
- $\frac{1}{\sqrt{(2\pi)^d |\Sigma|}}$ - нормувальний коефіцієнт, який забезпечує, що сума

ймовірностей для всього розподілу дорівнює 1. Він залежить від розмірності (d) і коваріаційної матриці (Σ).

- $\exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right)$ - експоненціальна частина, яка визначає

ймовірність того, що точка x належить до даного розподілу.

- $(x - \mu)$ - вектор, що показує відстань точки x від центру μ .
- $(x - \mu)^T \Sigma^{-1}(x - \mu)$ - зважена квадратична відстань від точки до центру з урахуванням коваріацій.

Дана функція розраховується для кожного компонента (класу), щоб знайти ймовірність, що точка належить до цього компонента. Потім всі компоненти комбінуються за формулою:

$$p(x) = \sum_{k=1}^K \pi_k f(x_i | \mu_k, \Sigma_k) \quad (2.3)$$

де:

- K - кількість компонентів (класи чи кластери).
- π_k - ваги (ймовірність, що точка належить компоненту k , нормовані так, що $\sum_{k=1}^K \pi_k = 1$).

- $f(x_i | \mu_k, \Sigma_k)$ - значення Гаусового розподілу для компонента k .

Це дозволяє моделі класифікувати дані або визначати ймовірність належності точки до конкретного класу.

Формула Гаусового розподілу лежить в основі розрахунків, що виконуються під час навчання. Навчання моделі відбувається за допомогою ЕМ-алгоритму (Expectation-Maximization). ЕМ-алгоритм складається з двох основних кроків:

1. Е-крок (Expectation)

На цьому кроці модель оцінює ймовірності приналежності кожної точки x_i до кожного компонента k (відповідно до поточних значень параметрів μ_k, Σ_k і π_k).

Для цього обчислюється постеріорна ймовірність для кожного компонента за формулою:

$$\gamma_{ik} = \frac{\pi_k f(x_i | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j f(x_i | \mu_j, \Sigma_j)} \quad (2.4)$$

де:

- γ_{ik} - ймовірність, що точка x_i належить компоненту k .
- $f(x_i | \mu_k, \Sigma_k)$ - формула Гаусового розподілу.

2. М-крок (Maximization)

На цьому кроці параметри моделі μ_k, Σ_k і π_k оновлюються, щоб максимізувати правдоподібність (тобто, зробити модель більш точною). Нові

параметри обчислюються на основі зважених середніх значень (з урахуванням ймовірностей γ_{ik} , знайдених на Е-кроці):

- Оновлення середнього значення (обчислює зважене середнє для k -го компонента (μ_k). Спостереження, які мають вищу ймовірність належності до компонента γ_{ik} , більше впливають на нове середнє значення компонента.):

$$\mu_k = \frac{\sum_{i=1}^N \gamma_{ik} x_i}{\sum_{i=1}^N \gamma_{ik}}, \quad (2.5)$$

де:

- x_i - це i -те спостереження (вектор даних).
- γ_{ik} - це ймовірність (знайдена на Е-кроці), що i -те спостереження належить до k -го компонента.

- Оновлення коваріаційної матриці (Σ_k) (оновлює коваріаційну матрицю, яка описує форму, розмір і орієнтацію k -го Гаусового компонента):

$$\Sigma_k = \frac{\sum_{i=1}^N \gamma_{ik} (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^N \gamma_{ik}} \quad (2.6)$$

- $(x_i - \mu_k)$ - це відхилення кожного спостереження від нового середнього компонента μ_k

- $(x_i - \mu_k)(x_i - \mu_k)^T$ - це матриця, яка відображає, як сильно розподіляються спостереження відносно середнього компонента.

- γ_{ik} - використовується для зважування: чим більше i -те спостереження належить до k -го компонента, тим сильніше воно впливає на форму (коваріацію) цього компонента.

- Оновлення ваг компонента (π_k) (показує, наскільки важливий k -й компонент відносно всіх інших компонентів.):

$$\pi_k = \frac{\sum_{i=1}^N \gamma_{ik}}{N} \quad (2.7)$$

- π_k - це частка спостережень, які належать до k -го компонента (або вага компонента).

- $\sum_{i=1}^N \gamma_{ik}$ - це загальна ймовірність, що всі N спостережень належать до k -го компонента (фактично, "кількість" спостережень у компоненті з урахуванням ймовірностей).

- N - загальна кількість спостережень.

Для реалізації механізму передбачення стану системи GMM навчається на вибірках даних що відповідають стабільним та нестабільним режимам роботи. Під час навчання GMM розраховує параметри (середні значення, коваріаційні матриці та ваги компонентів) для того, щоб математично описати структуру даних та розділити їх на класи або кластери:

- Середнє значення (mean vectors) для кожного класу описує центральну точку розподілу (центроїд) у багатовимірному просторі. Ці значення використовуються для визначення, які параметри найбільше відрізняють класи та побудови меж між класами (Рис. 2.49).

- Коваріаційні матриці (covariance matrices) для кожного класу показують, як змінюються різні параметри (особливості) відносно одне одного: описує форму розподілу даних у просторі та відображає зв'язок між параметрами (чи вони корелюють і наскільки сильно). Коваріаційні матриці допомагають визначити розподіл даних в межах кожного класу (Рис. 2.50).

- Ваги (weights) показують відносну ймовірність приналежності даних до кожного класу. Рівне значення ваг означає, що обидва класи мають однакову початкову ймовірність (Рис. 2.51).

Середні значення для кожного класу:

Стабільний клас: [-0.88407266 0.00835128 0.99157428 -0.00607556 -0.99999978 0.97221104
-0.99994651 0.97767652 0.03440378 0.97193867 0.99969601 0.99889629
0.99844239 0.51441882 0.52001558 0.49471283 0.50082633 -0.0615591
0.33109949 0.99873233 0.62616592 0.01118201 -0.95332006 0.97714766
-0.82977363 0.96233803 0.99629692 0.95638323 0.12401783 -0.01986761
0.49243841 0.50191514 0.91583572 -0.72428679 -0.7264699 0.99768192
0.99977512 -0.80630558 0.34439205 0.53136954 0.95154785 -0.77860641
-0.73532625 0.99472924 0.99429634 0.99702709 0.93749061 0.69475546
0.73042957 -0.0615591 0.7938196 0.00835128]

Нестабільний клас: [0.88407266 -0.00835128 -0.99157428 0.00607556 0.99999978 -0.97221104
0.99994651 -0.97767652 -0.03440378 -0.97193867 -0.99969601 -0.99889629
-0.99844239 -0.51441882 -0.52001558 -0.49471283 -0.50082633 0.0615591
-0.33109949 -0.99873233 -0.62616592 -0.01118201 0.95332006 -0.97714766
0.82977363 -0.96233803 -0.99629692 -0.95638323 -0.12401783 0.01986761
-0.49243841 -0.50191514 -0.91583572 0.72428679 0.7264699 -0.99768192
-0.99977512 0.80630558 -0.34439205 -0.53136954 -0.95154785 0.77860641
0.73532625 -0.99472924 -0.99429634 -0.99702709 -0.93749061 -0.69475546
-0.73042957 0.0615591 -0.7938196 -0.00835128]

Рис. 2.49. Розраховані середні значення для стабільного та нестабільного процесів

Коваріаційні матриці для кожного класу:

Стабільний клас:

```
[[ 2.49446061e-01 -7.95176925e-02 1.10930607e-02 ... -2.25878652e-01
 3.92097427e-03 -7.95176925e-02]
 [-7.95176925e-02 3.36336195e-01 2.55175022e-04 ... 1.43220113e-01
 1.85736415e-01 3.36335195e-01]
 [ 1.10930607e-02 2.55175022e-04 1.18117233e-02 ... -1.52195280e-03
 1.17978948e-03 2.55175022e-04]
 ...
 [-2.25878652e-01 1.43220113e-01 -1.52195280e-03 ... 7.02445548e-01
 6.85231149e-02 1.43220113e-01]
 [ 3.92097427e-03 1.85736415e-01 1.17978948e-03 ... 6.85231149e-02
 4.85064130e-01 1.85736415e-01]
 [-7.95176925e-02 3.36335195e-01 2.55175022e-04 ... 1.43220113e-01
 1.85736415e-01 3.36336195e-01]]
```

Нестабільний клас:

```
[[ 1.87387000e-01 3.62407534e-01 -1.72038341e-03 ... -1.14809847e-01
 1.28883715e-01 3.62407534e-01]
 [ 3.62407534e-01 1.66352632e+00 -1.51818735e-02 ... 2.66746790e-01
 4.96922400e-01 1.66352532e+00]
 [-1.72038341e-03 -1.51818735e-02 2.17511908e-02 ... -2.84613618e-02
 7.31941773e-04 -1.51818735e-02]
 ...
 [-1.14809847e-01 2.66746790e-01 -2.84613618e-02 ... 1.28997741e+00
 -9.44421910e-02 2.66746790e-01]
 [ 1.28883715e-01 4.96922400e-01 7.31941773e-04 ... -9.44421910e-02
 2.54638741e-01 4.96922400e-01]
 [ 3.62407534e-01 1.66352532e+00 -1.51818735e-02 ... 2.66746790e-01
 4.96922400e-01 1.66352632e+00]]
```

Рис. 2.50. Розраховані коваріаційні матриці для стабільного та нестабільного процесів

Ваги для кожного класу:
Стабільний клас: 0.5
Нестабільний клас: 0.5

Рис. 2.51. Розраховані ваги для стабільного та нестабільного процесів

Confusion Matrix (матриця помилок) навченої моделі показує, наскільки добре модель GMM класифікує дані на два класи: стабільний та нестабільний. Вона візуалізує співвідношення між справжніми значеннями та передбаченими значеннями (Рис. 2.52). Відповідно до результатів GMM модель навчилась з 100% результатом визначати належність навчальних даних до відповідної категорії, що вказує на значні відмінності між навчальними вибірками в просторі ознак (Рис. 2.53).

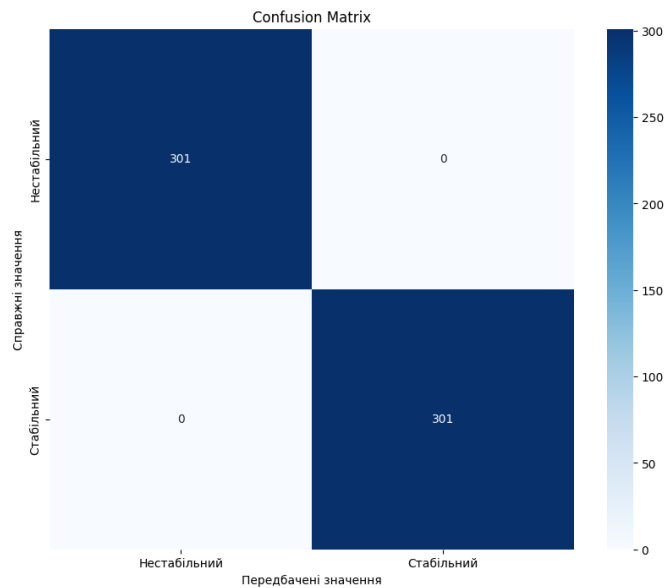


Рис. 2.52. Розраховані ваги для стабільного та нестабільного процесів

Навчена модель GMM може бути використана для визначення поточного стану системи в режимі реального часу. Отримуючи на вхід нові дані GMM визначає вірогідність їх належності до стабільної та нестабільної вибірок (Рис. 2.54). Для зручності аналізу додатково обчислюється «вікно стабільності», яке дозволяє визначити загальну статистику протягом кількох кроків.

Використання запропонованого алгоритму застосування GMM моделі (Рис. 2.55), дозволяє проводити аналіз стану виробництва в режимі реального часу та виявляти порушення режимів роботи на ранніх етапах, що дозволяє вчасно реагувати на них, зменшуючи втрати часу, сировини та якості продукту.

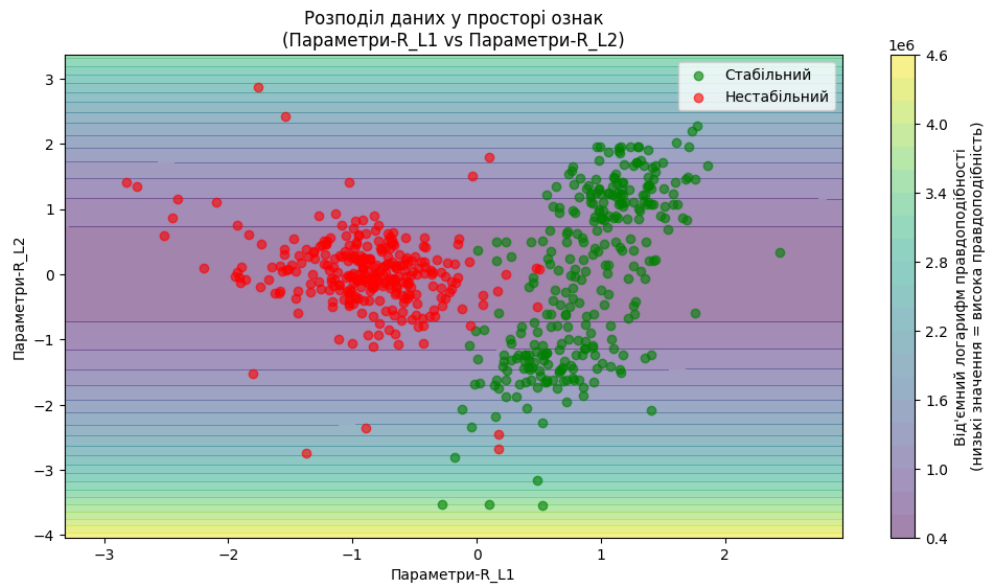


Рис. 2.53. Розподіл даних у просторі ознак для змінних R_L1, R_L2 для стабільної та нестабільної вибірок

```
Крок 179:
Часова мітка: 25632
Ймовірність стабільності: 1.00
Ймовірність нестабільності: 0.00
Співвідношення стабільності у вікні: 1.00
Confusion Matrix для вікна:
[[0 0]
 [0 9]]

Крок 180:
Часова мітка: 25776
Ймовірність стабільності: 0.00
Ймовірність нестабільності: 1.00
Співвідношення стабільності у вікні: 0.90
Confusion Matrix для вікна:
[[0 0]
 [1 8]]

Крок 181:
Часова мітка: 25920
Ймовірність стабільності: 0.00
Ймовірність нестабільності: 1.00
Співвідношення стабільності у вікні: 0.80
Confusion Matrix для вікна:
[[1 0]
 [1 7]]
```

Рис. 2.54. Приклад роботи GMM для аналізу належності стану системи до стабільного чи нестабільного

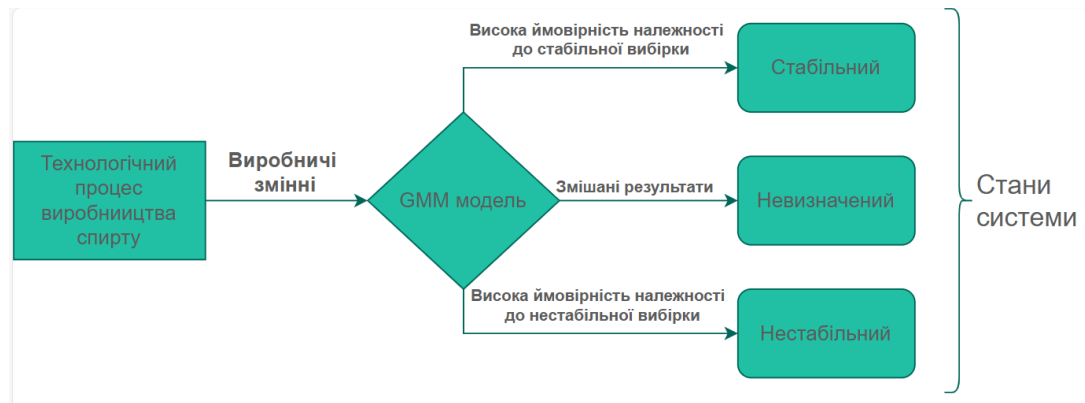


Рис. 2.55. Алгоритм використання GMM для класифікації стану виробництва

Висновки до розділу 2

1. Проведено системний аналіз технологічних відділень спиртового виробництва. Розглянуто взаємодії технологічних змінних та потоків на різних рівнях виробництва для ідентифікації задач дослідження.

2. Розроблено прикладну онтологію математичних моделей та проведено онтологічний аналіз відповідності існуючих моделей до поставлених задач дослідження. На основі онтологічного аналізу було виділено список задач, що можуть бути вирішені з використанням віртуальних інструментів.

3. Проведено статистичний аналіз даних роботи дистиляційного відділення для виявлення закономірностей, трендів та залежностей в даних, що можуть бути використані в задачах класифікації.

4. Розроблено механізм та алгоритм для класифікації стабільності роботи дистиляційного відділення в режимі реального часу з використанням Gaussian Mixture Model, що дозволить оцінювати стабільність системи і визначати прояви нестабільності роботи відділення на ранніх етапах, що дасть змогу вчасно реагувати на них, зменшуючи втрати часу, сировини та якості продукту.

РОЗДІЛ 3. МОДЕЛЮВАННЯ ТА ВІРТУАЛІЗАЦІЯ ПРОЦЕСІВ ВИРОБНИЦТВА

3.1 Моделювання процесу дистиляції з використанням NARX- моделей

3.1.1 Аналіз даних з датчиків дистиляційної колони

Для проведення аналізу та моделювання було використано змодельовані на основі фізичного моделювання фізико-математичних балансів дані роботи дистиляційної колони (Рис. 3.1), представлені у вигляді часових рядів, в яких зазвичай зберігаються виробничі дані.

	Time	Sensor1	Sensor2	Sensor3	Sensor4	Sensor5	Sensor6	Sensor7	Sensor8	Sensor9	Sensor10	Sensor11	Sensor12	Sensor13	Sensor14	Sensor15	MoleFractionTX	MoleFractionHX
0	0	49.718959	101.414959	49.718959	3693.858959	6695.368959	2924.818959	1048.588959	0.054394	0.501001	0.498999	76.803959	104.148959	110.216959	67.038059	0.798463	0.966281	0.033719
1	30	49.561845	100.962045	49.541545	3712.947045	6629.637045	2767.487045	1046.987045	0.054233	0.501001	0.498999	76.437645	103.977045	110.045045	66.866145	0.800811	0.965788	0.034212
2	60	50.189365	101.490865	50.145165	3718.008865	6692.598865	2763.018865	1050.988865	0.053941	0.501001	0.498999	77.041465	104.608865	110.676865	67.497965	0.803544	0.965758	0.034242
3	90	50.183496	101.403296	50.142796	3721.819296	6702.869296	2754.009296	1051.449296	0.053968	0.501001	0.498999	77.016096	104.609296	110.677296	67.498396	0.807065	0.965756	0.034244
4	120	50.428793	101.603693	50.406093	3724.444693	6710.444693	2744.244693	1052.674693	0.054121	0.501001	0.498999	77.252293	104.864693	110.932693	67.753793	0.811774	0.965784	0.034216
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
956	28680	49.604279	100.903579	54.520579	3783.434579	6808.374579	2782.764579	1052.594579	0.054906	0.501001	0.498999	76.467579	104.034579	110.102579	66.923679	0.792092	0.964851	0.035149
957	28710	50.164408	101.465308	55.073908	3783.905308	6809.155308	2782.875308	1053.235308	0.054913	0.501001	0.498999	77.028608	104.595308	110.663308	67.484408	0.792208	0.964852	0.035148
958	28740	50.332178	101.634378	55.234278	3784.033378	6809.343378	2783.673378	1053.403378	0.054925	0.501001	0.498999	77.196878	104.763378	110.831378	67.652478	0.792515	0.964853	0.035147
959	28770	50.080385	101.384285	54.974485	3783.671285	6808.931285	2782.991285	1053.181285	0.054930	0.501001	0.498999	76.944985	104.511285	110.579285	67.400385	0.792525	0.964853	0.035147
960	28800	50.157894	101.462594	55.042694	3783.638594	6808.978594	2783.658594	1053.218594	0.054937	0.501001	0.498999	77.022694	104.588594	110.656594	67.477694	0.792530	0.964854	0.035146

Рис. 3.1. Фрагмент таблиці даних роботи дистиляційної колони

Цільовими для прогнозування змінними в даній вибірці є концентрація фракцій, що відповідають змінним MoleFraction TX та MoleFraction HX. Відповідно до графіків зміни значень цих змінних з часом (Рис. 3.2), можна зробити висновок, що вони мають між собою негативну кореляцію або дзеркальну залежність між собою, що дозволяє нам узагальнювати інформацію про їх характер та про впливи вхідних змінних на них.

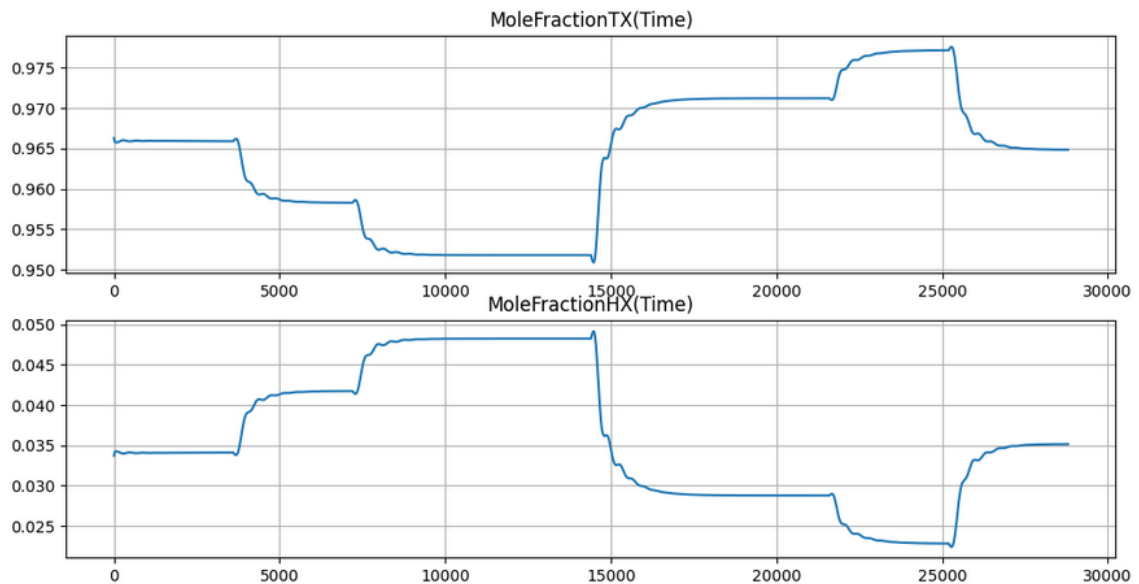


Рис. 3.2. Залежність MoleFraction TX та HX від часу

Одним із критеріїв для вибору моделі є автокореляція, яка показує наскільки значення часового ряду корелюють із самими собою на різних лагах (тобто, зі зсувом на певний період часу). Вона дозволяє оцінити, чи існує залежність між значеннями часового ряду в різні моменти часу, що суттєво впливає на прогнозування значень і вибір методів для цього. Графік автокореляції MoleFractionTX (Рис. 3.3) демонструє досить високу автокореляцію на початку, значне зниження і навіть негативну кореляцію на середніх лагах ("lags" або "запізнення" — це попередні значення змінної, які використовуються для моделювання або прогнозування її поточних чи майбутніх значень), після чого значення кореляції стабілізуються близько до нуля. Це свідчить про те, що змінна має певний ступінь високої автокореляції на коротких часових інтервалах, але на довгих лагах її автокореляція знижується, переходячи у негативну область, а потім стабілізується біля нуля. Це означає, що змінна має трендовий компонент або сезонність на коротких часових відрізках, але цей ефект зникає з часом. Таким чином моделі типу ARIMA (AutoRegressive Integrated Moving Average) та SARIMA (Seasonal ARIMA) можуть переоцінити вплив тренду або сезонності, які фактично зникають в подальших лагах, що призведе до помилкових прогнозів на довших відрізках часу. Що робить недоцільним використання цих моделей для передбачення поведінки цільових змінних виробничого процесу.

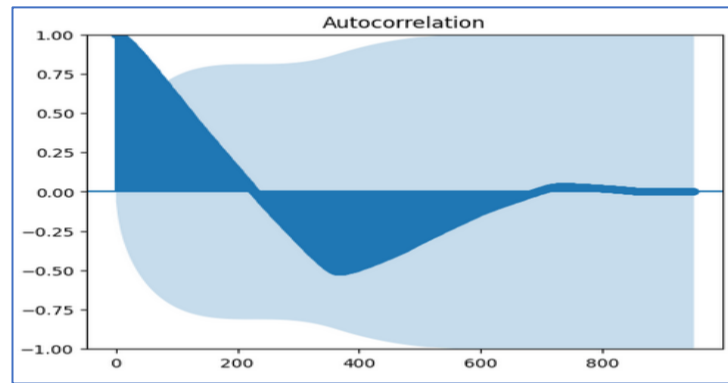


Рис. 3.3. Графік автокореляції MoleFractionTX

Ще одним інформативним показником для вибору моделі є стаціонарність часового ряду. Стаціонарність вказує, що статистичні властивості, такі як середнє відхилення, дисперсія та автокореляція, залишаються постійними в часі. Зміна цих властивостей в часі вказує на нестаціонарність ряду, та обмежує використання методів прогнозування, які потребують стаціонарних даних.

Одним зі способів визначення стаціонарності числового ряду є тест Дікі-Фуллера, який полягає в перевірці ряду на наявність одиничного кореня, що вказує на наявність тенденції або зміни з часом.

Застосування тесту Дікі-Фуллера для наявних даних показало наступні результати (Рис. 3.4):

Змінна	ADF статистика	p-значення	Крит. рівень (1%)	Крит. рівень (5%)	Висновок
Time	-0.4806	0.8957	-3.437237626048241	-2.8645807475403657	Нестаціонарна
Sensor1	-27.5813	0.0	-3.437180074590612	-2.8645553686071326	Стаціонарна
Sensor2	-6.9011	0.0	-3.4372521657769353	-2.864587159166059	Стаціонарна
Sensor3	-2.3055	0.1702	-3.4373407098114765	-2.8646262040163566	Нестаціонарна
Sensor4	-1.5883	0.4895	-3.437180074590612	-2.8645553686071326	Нестаціонарна
Sensor5	-1.273	0.6414	-3.43722314744835	-2.8645743628401763	Нестаціонарна
Sensor6	-1.7232	0.4192	-3.4372087295925664	-2.8645680048962348	Нестаціонарна
Sensor7	-2.3839	0.1463	-3.43731094378832	-2.864613078359397	Нестаціонарна
Sensor8	-6.3386	0.0	-3.4372448882473177	-2.86458394997689	Стаціонарна
Sensor9	Стала	-	-	-	Не тестується
Sensor10	Стала	-	-	-	Не тестується
Sensor11	-3.2287	0.0184	-3.4372887850912175	-2.8646033071530703	Стаціонарна
Sensor12	-29.6303	0.0	-3.437180074590612	-2.8645553686071326	Стаціонарна
Sensor13	-29.6303	0.0	-3.437180074590612	-2.8645553686071326	Стаціонарна
Sensor14	-29.6303	0.0	-3.437180074590612	-2.8645553686071326	Стаціонарна
Sensor15	-1.9724	0.2988	-3.437274090836024	-2.8645968274636933	Нестаціонарна
MoleFractionTX	-1.3916	0.5863	-3.4372887850912175	-2.8646033071530703	Нестаціонарна
MoleFractionHX	-1.3688	0.5971	-3.437274090836024	-2.8645968274636933	Нестаціонарна

Рис. 3.4. Визначення стаціонарності за допомогою тесту Дікі-Фуллера

Висновок про стаціонарність моделі є комплексним результатом порівняння наступних критеріїв:

- ADF (Augmented Dickey-Fuller) статистика показує значення статистики Діккі-Фуллера, яке використовується для перевірки гіпотези про наявність

одиночного кореня в часовому ряді. Чим нижче значення цієї статистики, тим більшою є ймовірність того, що ряд є стаціонарним. Значення порівнюється з критичними рівнями (1%, 5%) для прийняття або відхилення нульової гіпотези.

- р-значення - це статистична величина, яка показує ймовірність того, що отримані дані можуть виникнути випадково за умови, що нульова гіпотеза вірна. Якщо р-значення менше за 0.05, це свідчить про те, що дані мають малу ймовірність виникнути випадково, тому ми можемо відхилити нульову гіпотезу і зробити висновок, що ряд стаціонарний. Якщо р-значення більше за 0.05, це свідчить, що немає достатніх доказів для відхилення нульової гіпотези, тому ми не можемо відхилити її, і робимо висновок, що ряд, ймовірно, нестаціонарний.

- Критичні рівні (1% і 5%) - Це граничні значення ADF статистики для різних рівнів значущості. Якщо ADF статистика менша, ніж ці критичні рівні, то можна відхилити нульову гіпотезу про нестаціонарність.

- 1% критичний рівень: Якщо ADF статистика менша за цей рівень, ми з 99% впевненістю можемо сказати, що ряд стаціонарний.

- 5% критичний рівень: Якщо ADF статистика менша за цей рівень, ми з 95% впевненістю можемо сказати, що ряд стаціонарний.

Результати аналізу вказують на те, що шукані змінні MoleFractionTX, а також певні вхідні змінні є нестаціонарними. Тому використання таких моделей як RNN (Recurrent Neural Networks), NN (Multilayer Perceptron, MLP) та LSTM (Long Short-Term Memory) для їх передбачення може бути неефективним без додаткової обробки даних. Ці моделі зазвичай вимагають стаціонарних вхідних даних для досягнення високої точності, а нестаціонарність може значно знизити якість прогнозів. У промислових умовах обробка таких даних у режимі реального часу може потребувати значних обчислювальних ресурсів, що ускладнює застосування таких моделей без відповідної інфраструктури для підготовки даних.

Аналіз кореляції між змінними допомагає краще зрозуміти структуру даних для вибору моделі, а також відібрати змінні для навчання моделі. Якщо декілька

змінних сильно корелюють між собою, це може призвести до мультиколінеарності, що впливає на точність моделі.

Попередній аналіз кореляційної матриці вказує на наявність змінних з дуже високим рівнем кореляції (Рис. 3.5, а). Результати тесту Дікі-Фуллера також показали, що змінні “Sensor9”, та “Sensor10” є сталими значеннями. Для покращення роботи моделі варто уникнути використання цих виробничих змінних під час навчання моделі. Нова кореляційна матриця вказує на відсутність значної кореляції (Рис. 3.5, б).

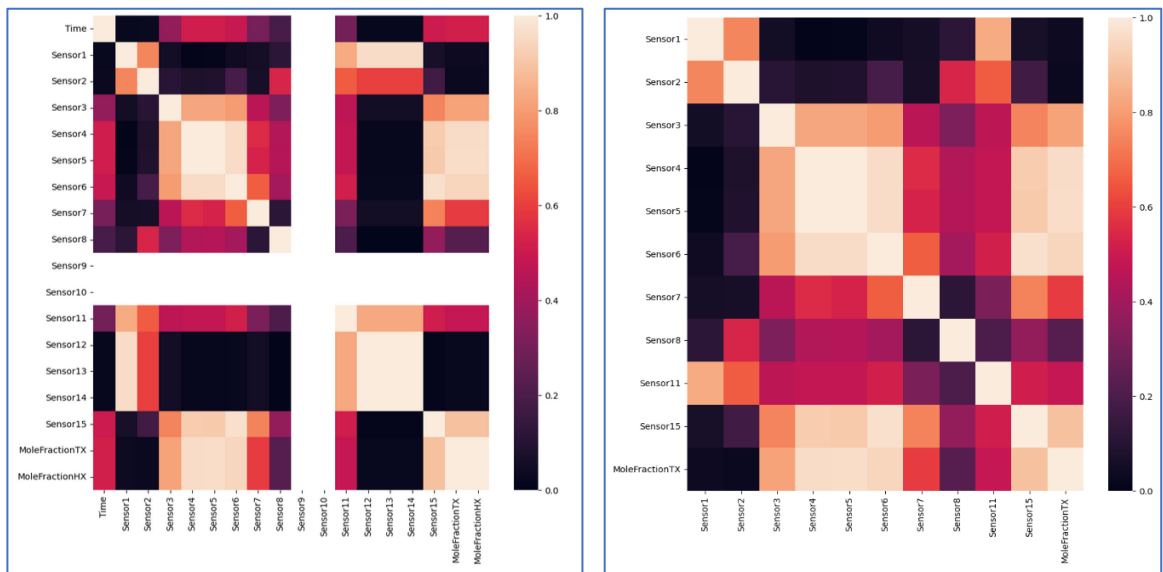


Рис. 3.5. Матриця кореляції до фільтрації значень(а), та після (б)

3.1.2 Прогнозування концентрації MoleFractionTX за допомогою NARX-моделі

NARX (Nonlinear Autoregressive with Exogenous Inputs) — це тип нелінійної моделі, яка використовується для прогнозування часових рядів або інших динамічних систем. Вона базується на використанні попередніх значень цільової змінної (autoregressive) та зовнішніх входів (exogenous inputs) для прогнозування поточного або майбутнього стану системи.

Модель NARX описується як:

$$y(t) = f(y(t-1), y(t-2), \dots, y(t-d_y), x(t-1), x(t-2), \dots, x(t-d_x)) \quad (3.1)$$

Де:

- $y(t)$ -значення цільової змінної в момент часу t ,
- $y(t - 1), y(t - 2), \dots, y(t - d_y)$ - попередні значення цільової змінної (зворотний зв'язок),
- $x(t - 1), x(t - 2), \dots, x(t - d_x)$ - попередні значення вхідних змінних (екзогенні змінні),
- d_x, d_y --кількість затримок для виходу та входу,
- f — нелінійна функція, яка моделює залежності (наприклад, це може бути нейронна мережа).

Відібрані змінні було розділено на навчальний та тестовий набори у співвідношенні 80/20, нормалізовано та використано навчання та перевірки роботи NARX-моделі. Було визначено оптимальні налаштування NARX-моделі ($n\_delay = 11$, $epochs = 100$, $validation\_split = 0.05$, $patience = 16$, $batch\_size = 32$), при яких модель показала наступні результати на тестових даних: середньоквадратична похибка (MSE) = 4×10^{-8} , коефіцієнт детермінації $R^2 = 0.998$ (Рис. 3.6-3.8)

```
# Побудова нейронної мережі
narx_model = keras.Sequential([
    layers.GaussianNoise(0.01, input_shape=(x.shape[1], x.shape[2])),
    layers.Flatten(),
    layers.Dense(32, activation='sigmoid'),
    layers.Dense(1, activation='linear')
])

# Компіляція моделі
narx_model.compile(
    optimizer=tf.keras.optimizers.Adam(),
    loss='mse',
    metrics=['mae']
)

# Навчання моделі
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=16)
history = narx_model.fit(x_train, y_train, epochs=100, validation_split=0.05, verbose=1, callbacks=[callback])
```

Рис. 3.6. Фрагмент коду ініціалізації, налаштування та навчання моделі

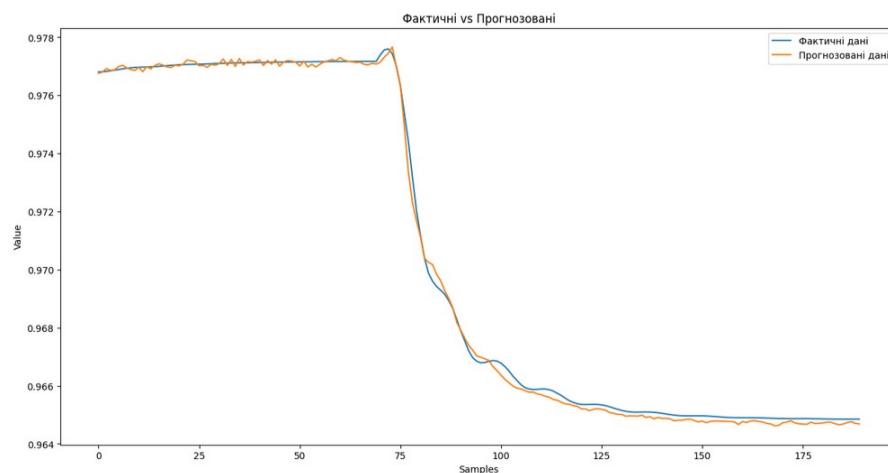


Рис. 3.7. Графік порівняння фактичних та прогнозованих тестових даних

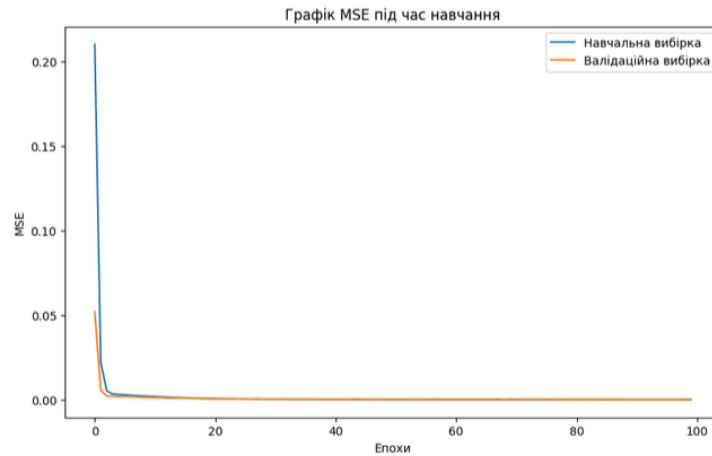


Рис. 3.8. Графік MSE для навчальної та тестової вибірок

Відповідно до результатів, NARX-модель здатна пояснити 99% варіацій цільової змінної, що робить її ефективним інструментом для передбачення результуючої концентрації речовин в дистиляційній колоні у режимі реального часу.

3.1.3 Дослідження впливу змін параметрів на поведінку NARX-моделі та пошук оптимальних налаштувань

Налаштування NARX-моделі є складним процесом, що потребує підбору параметрів навчання згідно особливостей процесу та його даних. Відповідно до цього було проведено дослідження впливу зміни гіперпараметрів моделі на точність її передбачення. За основу було взято підібрані оптимальні налаштування ($n\_delay = 11$, $epochs = 100$, $validation\_split = 0.05$, $patience = 16$, $batch\_size = 32$), та проведено оцінку поведінки моделі при їх змінах в наступних межах:

- $n\_delay\_values = [5, 11, 15, 20, 25]$
- $epochs\_values = [50, 100, 150, 200, 250]$
- $validation\_split\_values = [0.05, 0.1, 0.15, 0.2, 0.25]$
- $patience\_values = [8, 12, 16, 20, 24]$

Параметри що аналізуються включають в себе:

- n_delay - Це кількість часових затримок або розмір "вікна", яке використовується для створення послідовностей з даних. У контексті NARX, цей параметр визначає кількість попередніх значень, які модель враховує при

прогнозуванні. Збільшення значення `n_delay` дозволяє моделі враховувати довші часові залежності, що може бути корисним для процесів із сильними довгостроковими залежностями. Занадто велике значення може призвести до ускладнення моделі і потребувати більше даних для навчання, що збільшує ризик перенавчання.

- `epochs` - Кількість повних ітерацій (epoch), протягом яких модель навчається на наборі навчальних даних. Більше epoch дають більше часу для навчання моделі, що може допомогти досягти кращих результатів. Але після певної кількості epoch модель може почати перенавчатися на навчальні дані. Використання `EarlyStopping` дозволяє завершити навчання раніше, якщо модель перестане покращуватись, що зменшує шанс перенавчання моделі.

- `validation_split` - Частка навчальних даних, яка виділяється для перевірки (валідації) моделі під час навчання. Ці дані не використовуються для коригування ваг моделі, а лише для оцінки її продуктивності на даних, які вона не бачила. Це дозволяє відстежувати можливе перенавчання моделі. Збільшення цього значення збільшує варіативність валідаційних даних, ціною зменшення розміру навчальної вибірки.

- `patience` - Це кількість epoch, протягом яких модель може не покращуватися (в сенсі метрики `mse`), перш ніж `EarlyStopping` зупинить навчання. Занадто велике значення може призвести до надмірного навчання, оскільки модель продовжуватиме тренуватися навіть тоді, коли вже досягла оптимального результату. Занадто мале значення може зупинити навчання завчасно, до того як модель досягне свого максимуму.

Було проаналізовано поведінку NARX-моделі при змінах наведених ключових параметрів, побудовано графіки порівнянь фактичного та прогнозованого результатів (Рис. 3.9) та похибки MSE (Рис. 3.10). Перебір ключових параметрів показав наступні результати (Табл. 3.1-3.4):

Таблиця 3.1. Результати прогнозування при зміні значень n_delay

n_delay	MSE	R ²	AIC
5	1.217398e-06	0.961742	3581.906026
11	1.066237e-07	0.996640	7697.406442
15	2.262636e-07	0.992871	10460.476521
20	7.229676e-08	0.997719	13881.150269
25	1.045785e-07	0.996695	17328.166799

Таблиця 3.2. Результати прогнозування при зміні значень epochs

epochs	MSE	R ²	AIC
50	2.875916e-07	0.990938	7708.321030
100	1.461133e-07	0.995396	7700.872283
150	6.655485e-08	0.997903	7692.222367
200	2.258326e-08	0.999288	7680.333375
250	3.026806e-07	0.990463	7708.883536

Таблиця 3.3. Результати прогнозування при зміні значень validation_split

validation_split	MSE	R ²	AIC
0.05	7.995408e-08	0.997481	7694.240053
0.10	1.261039e-07	0.996027	7699.252241
0.15	4.192601e-07	0.986790	7712.467483
0.20	1.134271e-07	0.996426	7698.086841
0.25	2.156477e-07	0.993205	7705.154180

Таблиця 3.4. Результати прогнозування при зміні значень patience

patience	MSE	R ²	AIC
8	1.969227e-07	0.993795	7704.154998
12	9.896724e-08	0.996882	7696.586754
16	3.651123e-07	0.988496	7710.946329
20	3.882066e-07	0.987768	7711.620990
24	1.109698e-07	0.996503	7697.845914

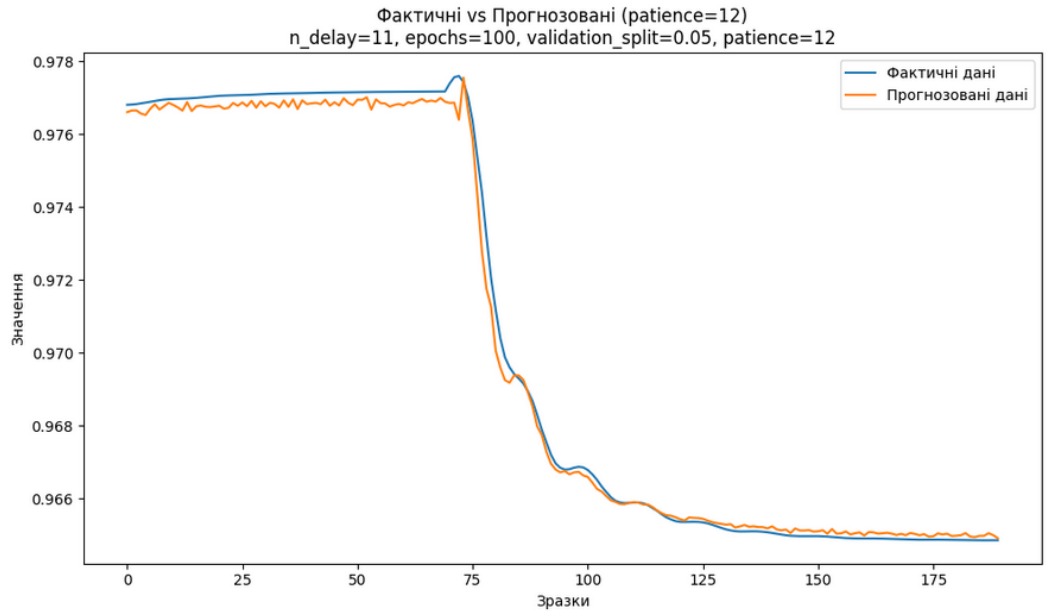


Рис. 3.9. Графік порівняння значень фактичного та прогнозованого результатів для patience=12

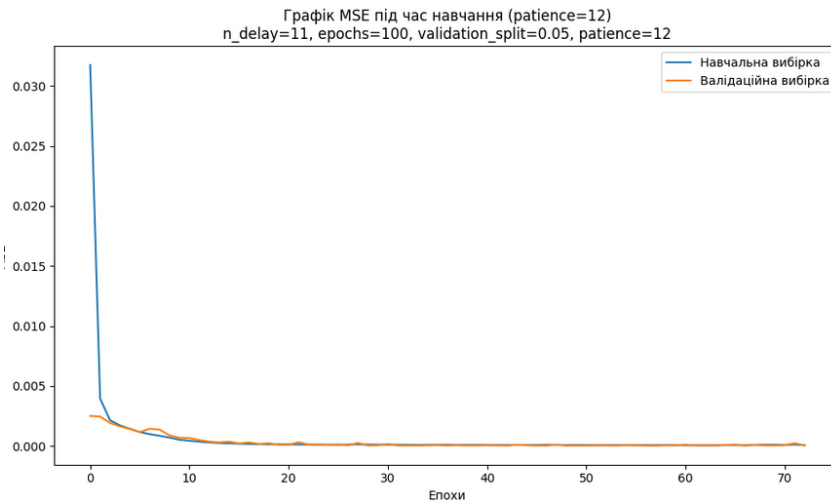


Рис. 3.10. Графік порівняння MSE фактичного та прогнозованого результатів для patience=12

Аналіз візуалізованих результатів дає змогу зробити наступні висновки(Рис. 3.11-3.13):

- Збільшення кількості часових затримок n_delay (кількість попередніх значень, які модель враховує при прогнозуванні) збільшує точність передбачення моделі. Особливо великий приріст спостерігається при значеннях менше 10, при цьому збільшення n_delay призводить до лінійного ускладнення моделі (Рис. 3.11).

- Збільшення кількості epochs збільшує точність моделі, зменшуючи похибку MSE, що в свою чергу позитивно впливає на розрахунок AIC, проте починаючи з значень більше 200 epochs модель перенавчається, навіть незважаючи на наявність EarlyStopping, що призводить до зменшення точності на тестових даних та зростання розрахованої складності моделі (Рис. 3.12).

- Краще значення MSE та AIC досягається при найменшому validation_split (0.05). При збільшенні розміру валідаційної вибірки точність моделі починає знижуватися (R^2 падає). Маленький розмір валідаційного набору дає більше даних для тренування, що підвищує якість моделі. Однак, якщо validation_split стає занадто великим, модель має менше тренувальних даних, що знижує її здатність навчатися (Рис. 3.13).

- Невелике patience дозволяє швидко завершувати навчання, якщо модель не покращується. Однак надто мале patience (наприклад, 8) може зупинити тренування до досягнення оптимальної точки. Занадто велике patience (20+) дає більше шансів на перенавчання (Рис. 3.14).

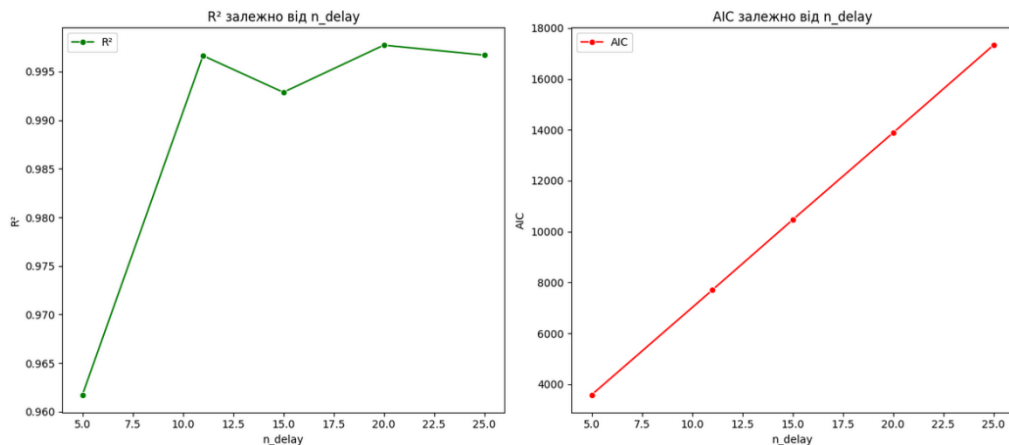


Рис. 3.11. Графіки R2 та AIC під час зміни значень n_delay

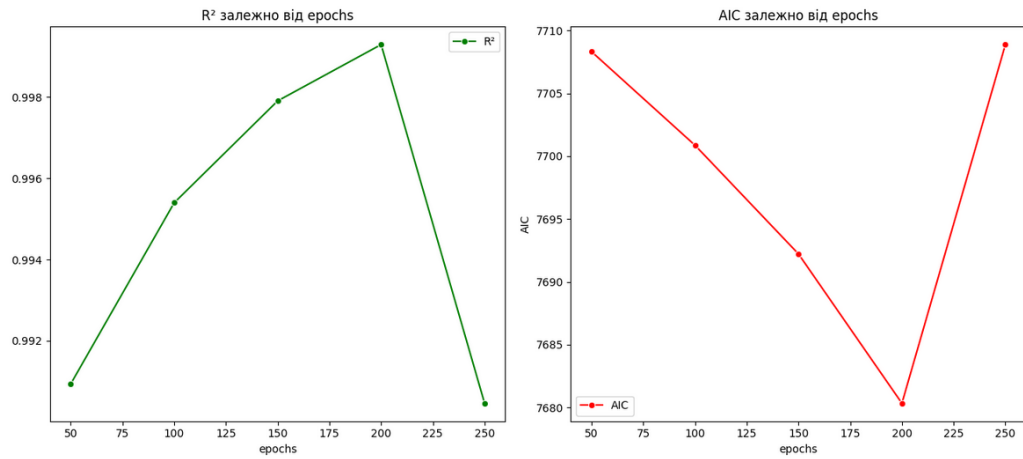


Рис. 3.12. Графіки R2 та AIC під час зміни значень epochs

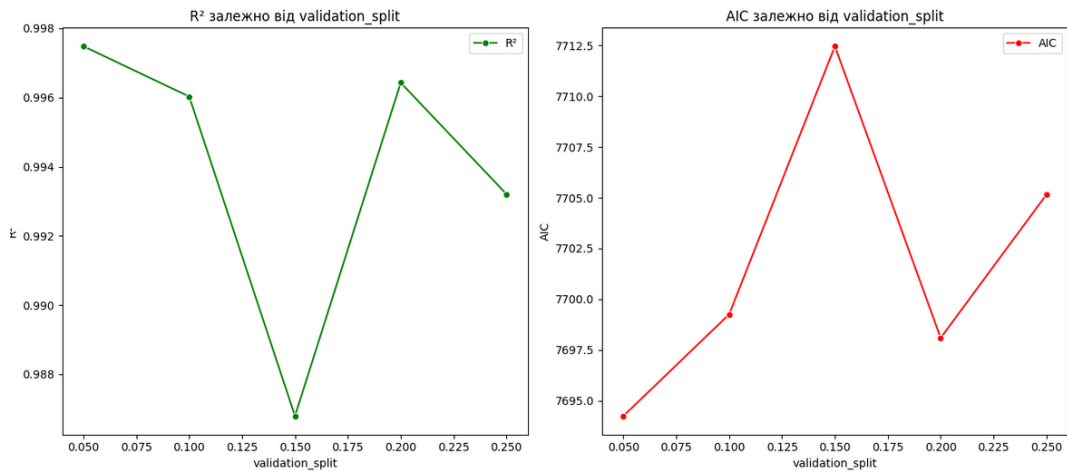


Рис. 3.13. Графіки R2 та AIC під час зміни значень validation_split

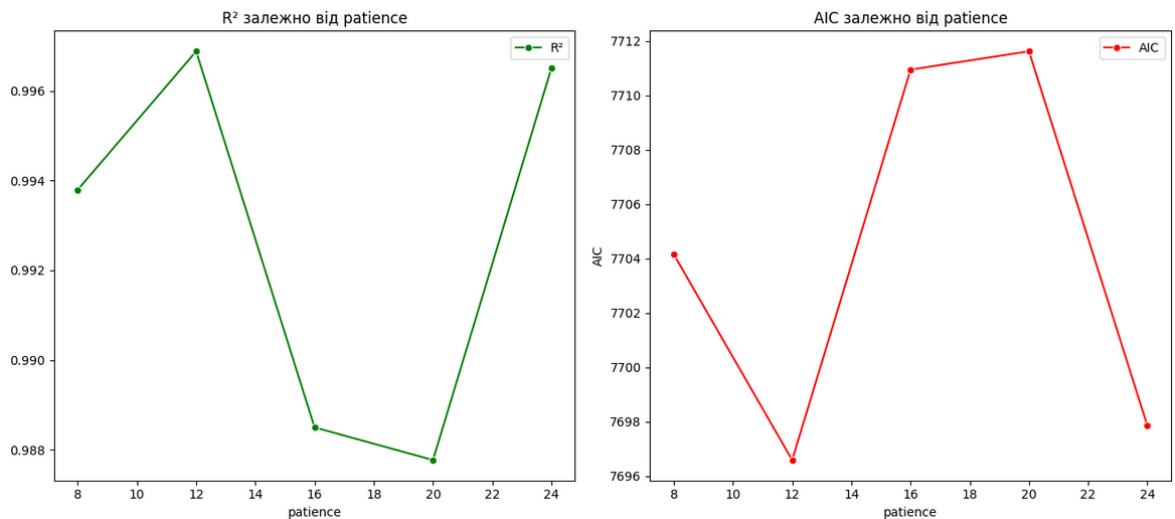


Рис. 3.14. Графіки R2 та AIC під час зміни значень patience

3.1.4 Пошук оптимальних значень з використанням бібліотеки Optuna

Незважаючи на можливість прогнозування впливу зміни окремих параметрів на результат роботи моделі, пошук оптимальних налаштувань часто

вимагає одночасної зміни кількох параметрів. Це пов'язано з тим, що параметри моделі взаємодіють між собою, і зміна одного з них може вплинути на ефективність інших. Наявність кореляцій між параметрами ускладнює процес підбору, оскільки зміна одного параметра може порушити баланс (рівновагу) ваг інших, що призводить до необхідності повторного налаштування. Для подолання цих труднощів та автоматизації процесу пошуку оптимальних налаштувань можна використовувати спеціальні бібліотеки для автоматизованого підбору гіперпараметрів

Однією з таких бібліотек є Optuna. Optuna - це бібліотека для автоматизованого підбору гіперпараметрів (hyperparameter optimization) у задачах машинного навчання. Вона спрощує процес оптимізації гіперпараметрів, за рахунок використання алгоритмів оптимізації, які дозволяють їй ефективно працювати навіть з великою кількістю гіперпараметрів. Серед основних алгоритмів пошуку найкращих параметрів Optuna підтримує:

- **Tree-structured Parzen Estimator (TPE).** TPE використовує ймовірнісні моделі для визначення найкращих значень гіперпараметрів. Він аналізує результати попередніх спроб і вибирає нові параметри, які, ймовірно, дадуть кращий результат.

- **Random Search.** Випадковий пошук, де параметри обираються випадковим чином з заданого діапазону. Цей метод простий, але може бути неефективним для складних задач.

- **Grid Search.** Повний перебір усіх можливих комбінацій параметрів. Використовується рідко через високу обчислювальну складність.

- **CMA-ES (Covariance Matrix Adaptation Evolution Strategy).** Еволюційний алгоритм, який адаптується до форми цільової функції та ефективно досліджує простір параметрів.

- **NSGA-II (Non-dominated Sorting Genetic Algorithm).** Використовується для багатокритеріальної оптимізації, коли потрібно оптимізувати кілька цільових функцій одночасно.

За замовчуванням Optuna використовує ТРЕ алгоритм. Окрім цього бібліотека надає вбудовані інструменти для аналізу результатів, що дозволяє проаналізувати вплив змінних на результат та їх взаємодію.

Optuna за 50 ітерацій запропонувала оптимальні налаштування гіперпараметрів NARX-моделі `{'n_delay': 10, 'epochs': 150, 'validation_split': 0.25, 'patience': 16}`, що забезпечують точність прогнозування на тестових даних з похибкою $(MSE) = 4.6 \times 10^{-8}$. Вбудовані в бібліотеку засоби аналізу результатів дозволяють також відображати:

- Процес підбору налаштувань (Рис. 3.15), що дає змогу побачити процес підбору.
- Відносний вплив кожного гіперпараметра на цільову функцію, оцінюючи наскільки кожен параметр впливає на результат оптимізації (Рис. 3.16).
- Залежність значення цільової функції від окремих гіперпараметрів (Рис. 3.17).
- Вплив на результат комбінації значень параметрів, що дає змогу виявити закономірності, які можуть бути прихованими в багатовимірному просторі параметрів (Рис. 3.18).
- Відображати взаємозв'язки між двома параметрами та вплив їх комбінації на значення MSE (Рис. 3.19).

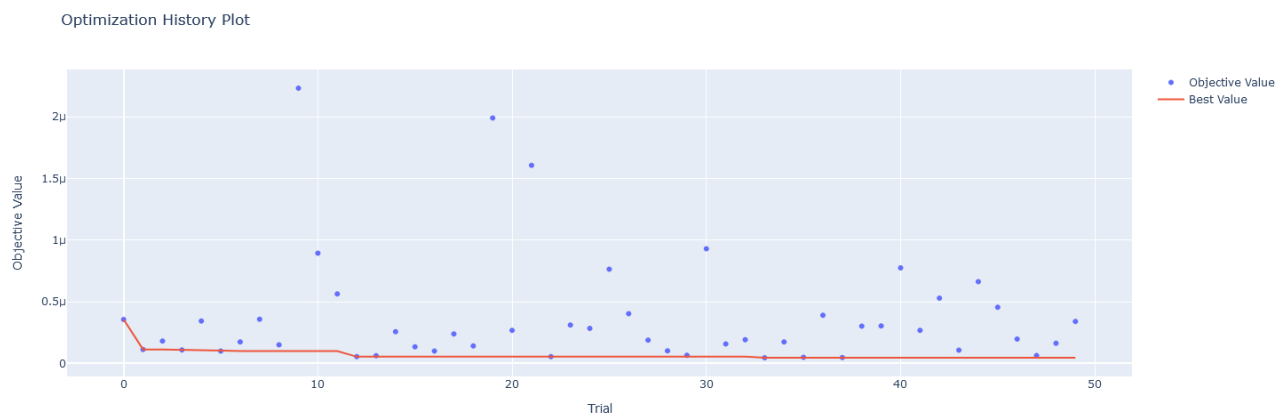


Рис. 3.15. Графік розподілу похибок для набору гіперпараметрів в залежності від ітерації

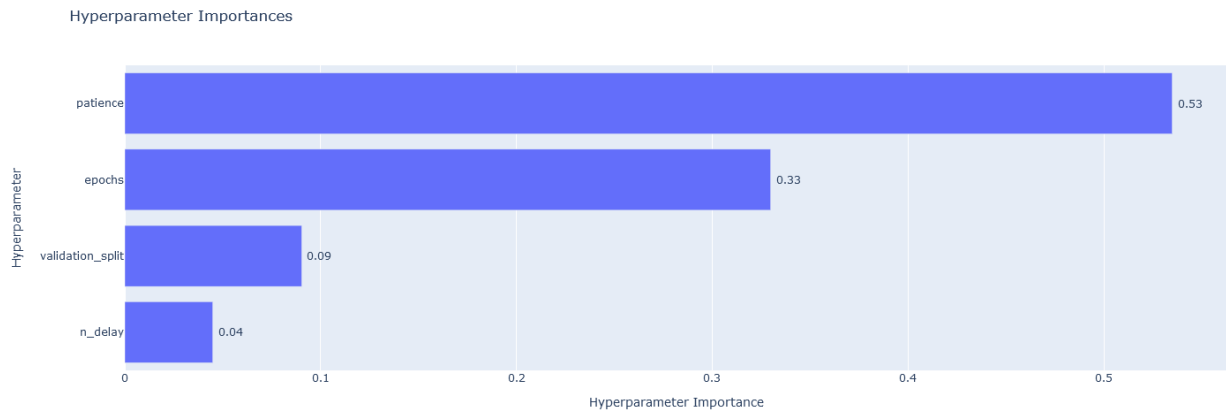


Рис. 3.16. Відносний вплив кожного гіперпараметра на цільову функцію

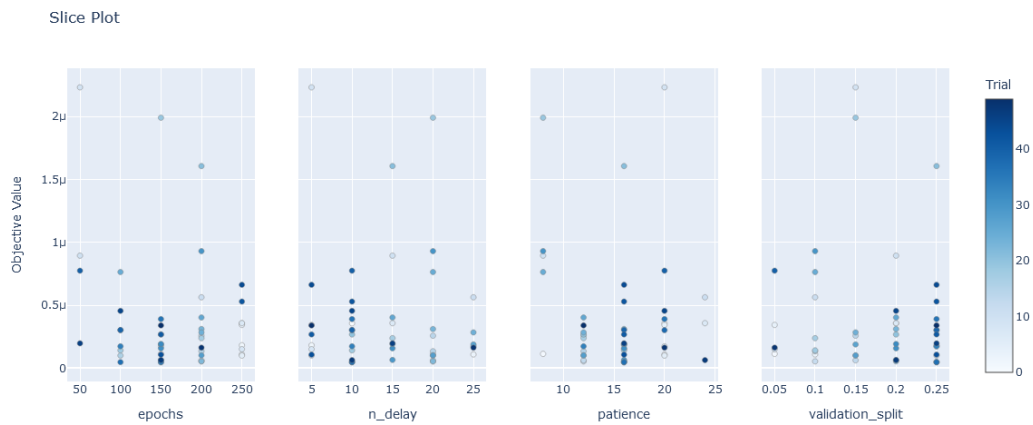


Рис. 3.17. Графік залежності похибки MSE від значень окремих гіперпараметрів, їх розподіл по ітераціям trial

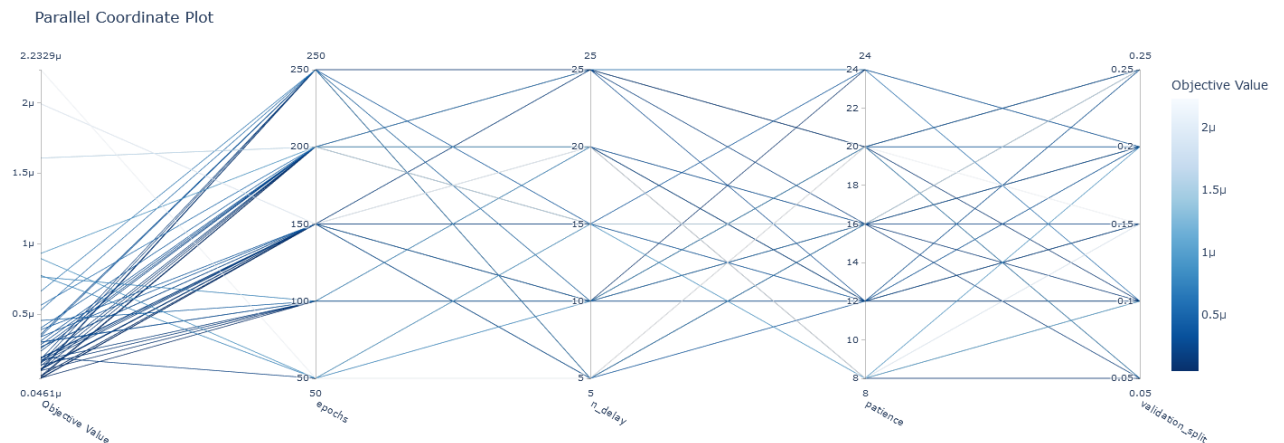


Рис. 3.18. Графік візуалізації взаємозв'язків між різними гіперпараметрами та результатами оптимізації

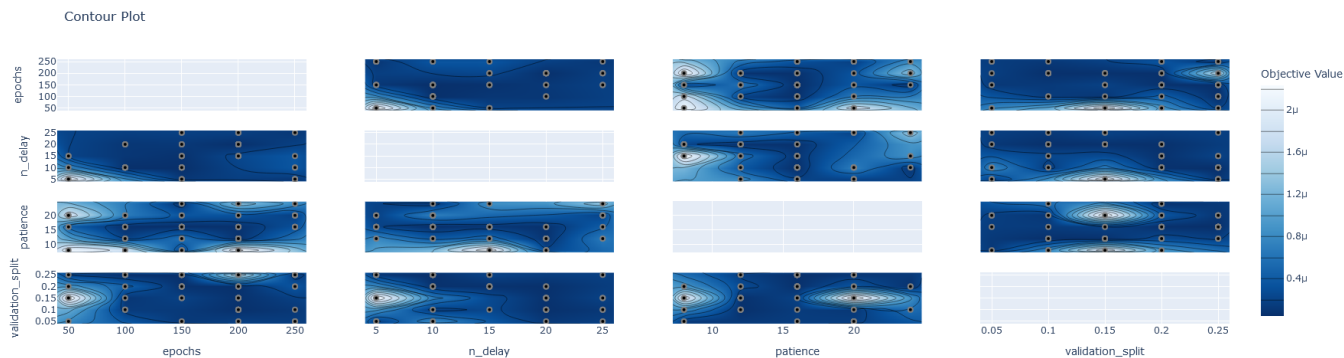


Рис. 3.19. Графік взаємозв'язків між двома параметрами та значенням цільової функції

Висновки: На основі результатів проведеного дослідження, використання NARX-моделі в задачах прогнозування концентрації дозволяє з точністю до 99% передбачити поведінку змінної, в залежності від налаштування моделі. При цьому найбільший вплив на цільову функцію мають гіперпараметри *patience*, що впливає на передчасну зупинку навчання і захист моделі від перенавчання, та кількість епох протягом яких модель навчається. Бібліотеки підбору гіперпараметрів, такі як Optuna здатні спростити процес налаштування моделі та надати розгорнутий аналіз.

Табл. 3.5 демонструє можливі способи використання моделі дистиляційної колони у контексті моніторингу та керування технологічними процесами. Ця таблиця демонструє, як модель дистиляційної колони без регуляторів може бути використана для різних завдань у контексті моніторингу та керування технологічними процесами. Навіть без регуляторів модель залишається потужним інструментом для аналізу, оптимізації та підготовки до роботи з реальною системою.

Таблиця 3.5. Можливості використання моделі процесу

№	Сфера використання	Опис	Приклад
1	Моніторинг процесу	Використання моделі як віртуального сенсора для оцінки концентрацій на основі доступних даних (температура, тиск, витрата сировини).	Візуалізація оцінених концентрацій у графічному інтерфейсі (наприклад, SCADA, Python-бібліотеки).
2	Прогнозування поведінки процесу	Модель використовується для прогнозування змін концентрацій при зміні вхідних параметрів (температура, тиск, витрата сировини).	Аналіз, як зміна температури нагріву вплине на оцінену концентрацію дистиляту.

3	Оптимізація параметрів процесу	Використання моделі для знаходження оптимальних умов роботи колони (температура, тиск, витрата сировини) на основі оцінених концентрацій.	Визначення оптимальної температури для максимізації оціненої концентрації цільового компонента.
4	Тестування сценаріїв "Що якщо"	Модель дозволяє аналізувати різні сценарії роботи колони на основі оцінених концентрацій.	Аналіз, що станеться, якщо витрата сировини збільшиться на 10%.
5	Порівняння з непрямыми даними	Модель використовується для порівняння оцінених концентрацій з іншими доступними даними (наприклад, температура верху і низу колони).	Порівняння оцінених концентрацій з температурним профілем колони для оцінки точності моделі.
6	Додавання віртуальних регуляторів	Модель може бути доповнена віртуальними регуляторами (наприклад, PID) для імітації роботи реальної системи на основі оцінених концентрацій.	Тестування PID-регулятора для підтримки заданої температури в колоні на основі оцінених концентрацій.
7	Підготовка до впровадження регуляторів	Модель використовується для тестування різних стратегій керування перед їх впровадженням у реальну систему.	Перевірка ефективності модельно-прогнозуючого керування (МРС) для дистиляційної колони на основі оцінених концентрацій.
8	Навчання персоналу	Модель використовується як навчальний інструмент для операторів та інженерів, щоб вони могли експериментувати з різними параметрами без ризику для реального обладнання.	Проведення тренінгів зі зміни параметрів процесу та аналізу їх впливу на оцінені концентрації.
9	Аналіз нештатних ситуацій	Модель дозволяє імітувати нештатні ситуації (наприклад, відмова регулятора) та аналізувати їх наслідки на основі оцінених концентрацій.	Аналіз, що станеться, якщо регулятор температури вийде з ладу.
10	Створення віртуального двійника	Модель інтегрується з реальною системою як віртуальний двійник, що дозволяє тестувати різні сценарії та оптимізувати процес на основі оцінених концентрацій.	Використання моделі для прогнозування стану системи в реальному часі на основі доступних даних.
11	Оптимізація енергоспоживання	Модель використовується для аналізу енергоспоживання процесу та пошуку способів його зниження на основі оцінених концентрацій.	Визначення оптимальної температури нагріву для мінімізації енергоспоживання.
12	Підготовка звітів та рекомендацій	На основі результатів моделювання готуються звіти або рекомендації для покращення процесу.	Створення звіту про оптимальні параметри роботи колони для досягнення максимальної ефективності на основі оцінених концентрацій.
13	Інтеграція з системами керування	Модель може бути інтегрована з системами керування (наприклад, DCS, SCADA) для покращення моніторингу та керування процесом на основі оцінених концентрацій.	Використання моделі для прогнозування стану системи та передачі рекомендацій до системи керування.
14	Тестування нових технологій	Модель використовується для тестування нових технологій або обладнання перед їх впровадженням у реальну систему.	Оцінка впливу нового теплообмінника на ефективність дистиляційної

			колони на основі оцінених концентрацій.
15	Використання як віртуального сенсора	Модель використовується для оцінки концентрацій на основі доступних даних (температура, тиск, витрата сировини), якщо реальні вимірювання відсутні.	Оцінка концентрацій верху і низу колони на основі температурного профілю та витрати сировини.
16	Підготовка до впровадження вимірювань	Модель використовується для оцінки того, які вимірювання будуть найбільш корисними для контролю процесу.	Визначення, чи варто встановлювати додаткові сенсори для вимірювання концентрацій на основі результатів моделювання.

3.2 Моніторинг динамічних змінних у дистиляційному відділенні спиртового виробництва

3.2.1 Дослідження моделей ВС для моніторингу виробничих змінних

1) Аналіз даних виробничого процесу. Для дослідження та порівняльного аналізу точності прогнозування поведінки змінної за допомогою моделей ВС було використано дані часових рядів, отримані з датчиків дистиляційного відділення спиртового заводу за 12 годин роботи в умовах стаціонарного режиму. Ці дані охоплюють показники температури, тиску, рівнів і витрат, які характеризуються високим ступенем кореляції та наявністю нестационарності.

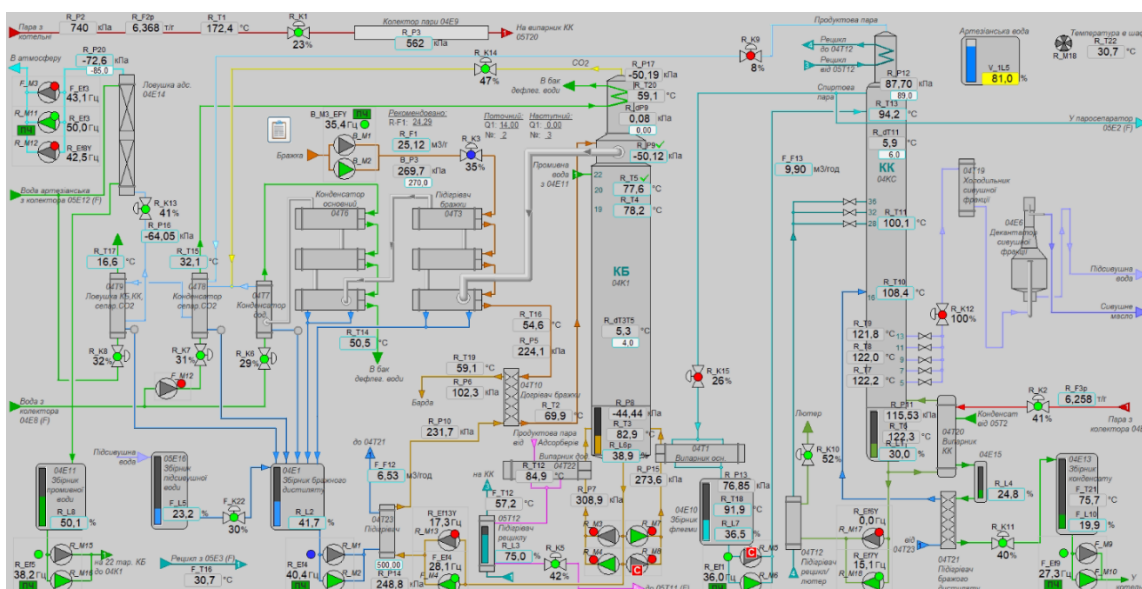


Рис. 3.20. Мнемосхема дистиляційного відділення

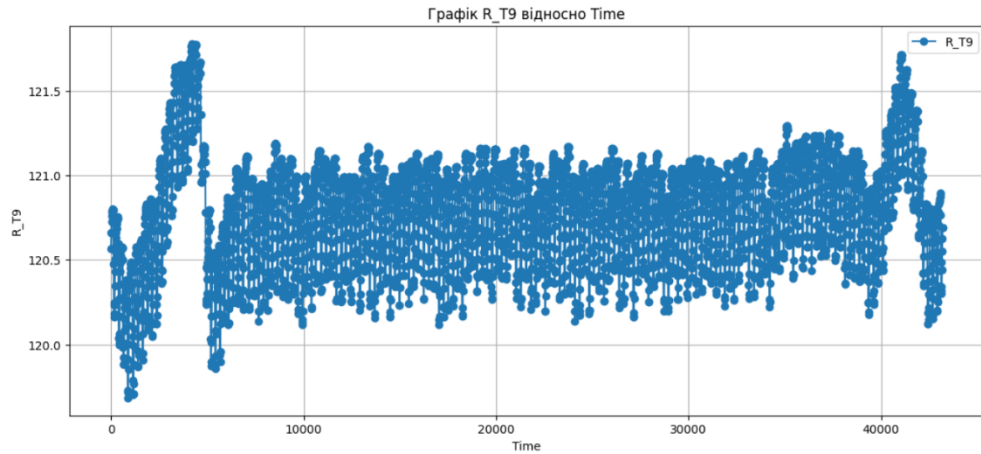


Рис. 3.21. Графік зміни температури над тринадцятою тарілкою колони концентрації

В якості цільової змінної для перевірки ефективності роботи моделей ВС було взято температуру над тринадцятою тарілкою колони концентрації (Рис. 3.21).

2) Вибір критеріїв порівняння. Для порівняння результатів прогнозування математичних моделей було використано наступні показники:

1. MSE (Mean Squared Error) — метрика, що використовується для оцінки якості прогнозу моделі, особливо в задачах регресії. MSE показує середнє значення квадратів помилок між фактичними значеннями (y_i) і прогнозованими значеннями ($\hat{y}_i$):

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (3.2)$$

де n — кількість спостережень, y_i — фактичне (реальне) значення, $\hat{y}_i$ — прогнозоване значення, $(y_i - \hat{y}_i)^2$ — квадрат помилки для кожного спостереження;

R^2 (коефіцієнт детермінації) — статистична метрика, яка показує, здатність моделі пояснювати варіацію цільової змінної у порівнянні з її середнім значенням:

$$R^2 = 1 - \frac{RSS}{TSS} \quad (3.3)$$

де: $RSS = \sum (y_i - \hat{y}_i)^2$ – залишкова сума квадратів (помилки моделі), $TSS = \sum (y_i - \bar{y})^2$ – загальна сума квадратів (варіація фактичних значень навколо середнього), y_i – фактичне значення, $\hat{y}_i$ – прогнозоване значення, $\bar{y}$ – середнє значення фактичних значень.

2. AIC (Akaike Information Criterion) — оцінювач похибки позавибіркового передбачування, і відтак відносної якості статистичних моделей, для заданого набору даних. AIC дозволяє порівнювати моделі та обирати ту, яка найкраще збалансовує точність (якість підгонки) та простоту (число параметрів):

$$AIC = 2k - 2\ln(L), \quad (3.4)$$

де k — кількість параметрів у моделі, L — максимальне значення функції правдоподібності моделі (міра того, наскільки добре модель описує дані).

3) Методологія дослідження. Для перевірки роботи моделей на нових даних у реальних умовах в роботі застосовується TimeSeriesSplit. TimeSeriesSplit – техніка перехресної перевірки (cross-validation), що адаптована для використання у задачах аналізу часових рядів. Її застосування допомагає забезпечити коректне розбиття даних з урахуванням їх послідовності, що важливо для даних часових рядів, які залежать від хронологічного порядку даних. На відміну від стандартного k-fold cross-validation, де поділ даних відбувається випадковим чином, у TimeSeriesSplit модель навчається на попередніх даних і перевіряється на наступних (майбутніх) даних. Це імітує реальний сценарій, коли прогнозується майбутня поведінка системи на основі минулої інформації.

4) Перевірка концепції. Для перевірки працездатності моделі ВС на наявних даних та оцінки базових результатів було проведено попередню перевірку концепції (Proof of Concept) з використанням пакету MATLAB. MATLAB – це програмний пакет для числового аналізу, створений компанією MathWorks. Обрання цього інструменту було зумовлене його численними функціями, необхідними для виконання поставленого завдання:

- забезпечує зручні умови для швидкого створення та тестування моделей завдяки великому набору вбудованих функцій, зокрема для роботи з нейронними мережами;
- дозволяє легко проводити експерименти з архітектурою моделей, кількістю затримок, числом нейронів у прихованому шарі та іншими параметрами;
- має потужні засоби для роботи з TimeSeriesSplit та іншими методами крос-валідації, що дає можливість оцінювати поведінку моделі на різних наборах даних.

Реалізація моделі NARX у MATLAB із використанням TimeSeriesSplit для 5-fold cross-validation продемонструвала результати (Рис. 3.22-3.24), на основі яких можна зробити такі висновки: усі чотири оброблені фолди показують низькі значення Train Performance (варіант критерія 1), що свідчить про ефективне навчання моделі на навчальних даних. Результати Test Performance значно варіюються: похибка на першому фолді значно вища, ніж у наступних, що може бути результатом нових трендів у тестових даних, вплив яких зменшується з ростом обсягу навчальних даних моделі.

```
>> NARX_TimeSeriesSplit
Processing fold 1...
Fold 1 -> Train Performance: 0.000064, Test Performance: 0.075339
Processing fold 2...
Fold 2 -> Train Performance: 0.000063, Test Performance: 0.006470
Processing fold 3...
Fold 3 -> Train Performance: 0.000047, Test Performance: 0.006698
Processing fold 4...
Fold 4 -> Train Performance: 0.000081, Test Performance: 0.001071
Processing fold 5...
Not enough data points left for this fold. Skipping...
Average Train Performance: 0.000064
Average Test Performance: 0.022394
```

Рис. 3.22. Результати роботи NARX в MATLAB з TimeSeriesSplit для 5-fold cross-validation

В цілому, середнє значення Test Performance становить 0.0224, що свідчить про задовільне узагальнення моделі, хоча можливі варіації залежно від частин даних. На четвертому фолді, коли модель отримала достатню варіативність даних для навчання, похибка на тестових даних становить лише 0.001, а

коефіцієнт кореляції — 0.995. Це вказує на те, що модель дуже добре відображає зв'язок між вхідними даними та цільовими значеннями.

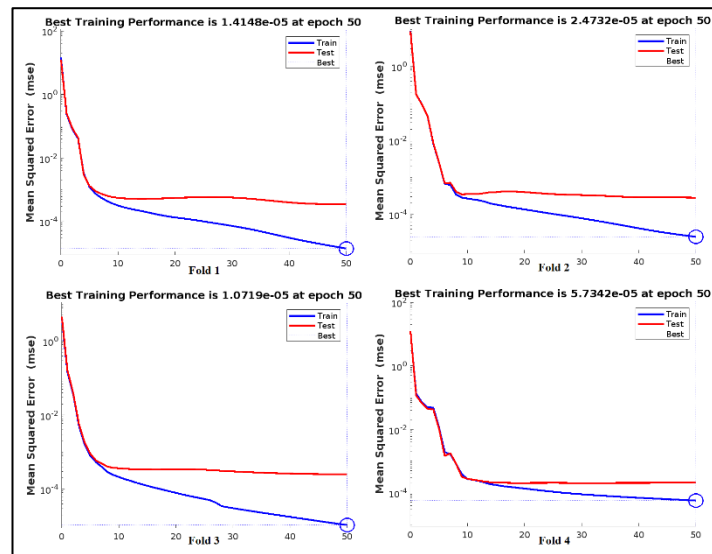


Рис. 3.23. Графіки Performance для NARX в MATLAB

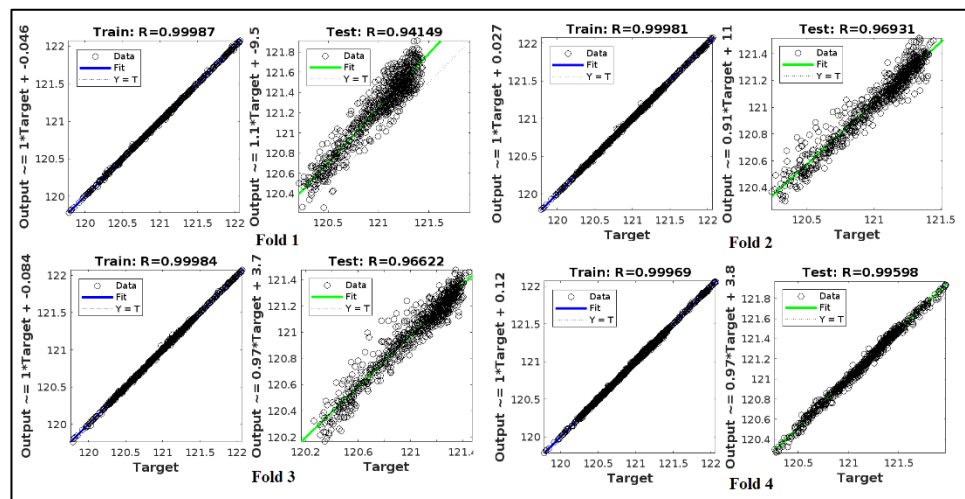


Рис. 3.24. Графіки регресії для NARX в MATLAB

Отримані внаслідок перевірки концепції результати вказують на перспективність подальшого моделювання.

5) Моделювання. Моделювання виконувалося з використанням мови Python та його бібліотек у середовищі Google Colab. Було проведено порівняння ефективності прогнозування моделей XGBoost, Decision Tree, NARX, Gradient Boosting та RNN. Усі вони відповідають таким критеріям:

- Можливість роботи з даними часових рядів.
- Здатність працювати з нелінійними залежностями в даних.

- Здатність виявляти складні залежності між вхідними змінними та цільовими показниками.
- Можливість роботи з багатовимірними даними.

Згідно з методом `TimeSeriesSplit`, дані були розділені на 5 навчальних та тестових вибірок зі збереженням часових послідовностей. З кожним `fold` кількість даних для навчання моделі збільшується, забезпечуючи донавчання моделі (Табл. 3.6). Обрані моделі були навчені та оцінені на основі даних з `TimeSeriesSplit`.

Таблиця 3.6. Розбиття даних на folds

№ фолда	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Розмір навчального набору	603	1202	1801	2400	2999
Розмір валідаційного набору	599	599	599	599	599

XGBoost. XGBoost (eXtreme Gradient Boosting) – це бібліотека з відкритим кодом, що використовується в машинному навчанні, надаючи функції для розв'язання завдань, пов'язаних з регуляризацією градієнтного бустингу. Бібліотека показала наступні результати на тестових даних (Табл. 3.7).

Таблиця 3.7. Результати передбачення XGBoost на тестових даних

№ фолда	MSE	R ²	AIC
Fold 1	0,00048707	0,9719	46431,37
Fold 2	0,00081122	0,9523	46086,68
Fold 3	0,00014358	0,9913	45699,66
Fold 4	0,00019493	0,9876	45882,83
Fold 5	0,00021514	0,9917	45941,92

Нормалізований графік метрик ілюструє відносні значення для кожного показника (Рис. 3.25). Висока точність (низький MSE та високий R²) свідчить про те, що XGBoost успішно виявляє складні патерни в даних. Проте високі значення AIC вказують на значну складність моделі.

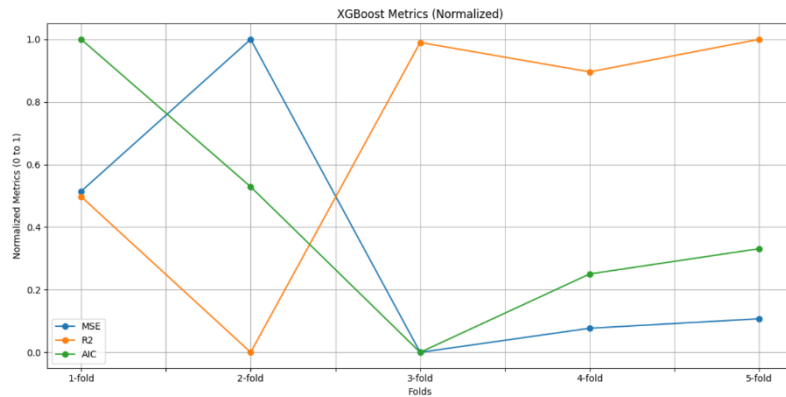


Рис. 3.25. Нормалізований графік метрик XGBoost

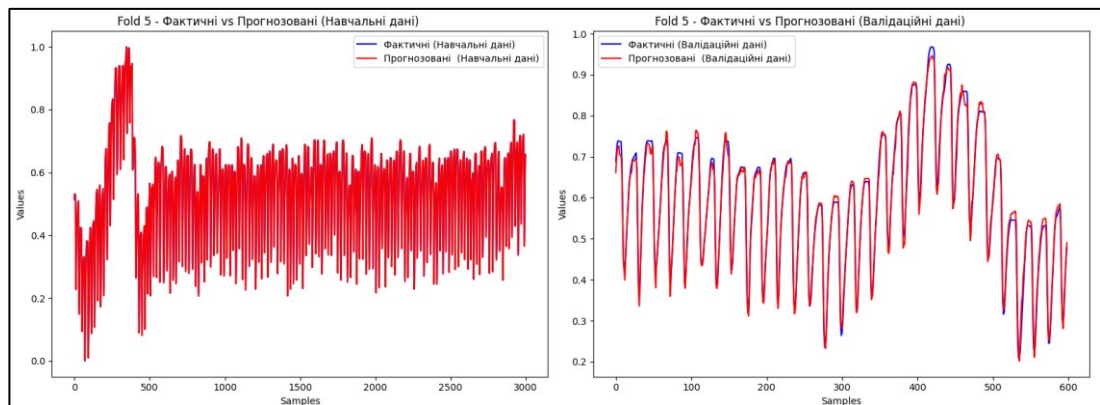


Рис. 3.26. Порівняння фактичних та прогнозованих значень для тестових та валідаційних даних моделлю XGBoost

DecisionTree. DecisionTree (DecisionTreeRegressor) – це регресійна модель на основі дерев рішень (decision trees), яка використовується для прогнозування безперервних цільових змінних. Вона належить до класу жадібних алгоритмів (greedy algorithms), що будують дерево, розділяючи дані на підмножини для мінімізації помилки в кожному вузлі. DecisionTreeRegressor реалізовано в бібліотеці scikit-learn.

Аналіз результатів роботи DecisionTree вказує на її нижчу точність порівняно з XGBoost (Рис. 3.27, 3.28). Від’ємні значення критерію AIC можуть вказувати на непридатність цього критерію для оцінки даного типу моделей, оскільки дерева рішень не належать до ймовірнісних моделей.

Таблиця 3.8. Результати передбачення DecisionTree на тестових даних

№ фолда	MSE	R ²	AIC
Fold 1	0,00143358	0,9174	-3852,00
Fold 2	0,00098765	0,9433	-4075,19
Fold 3	0,00061588	0,9625	-4358,08
Fold 4	0,00059168	0,9625	-4382,09
Fold 5	0,00113648	0,9561	-3991,11

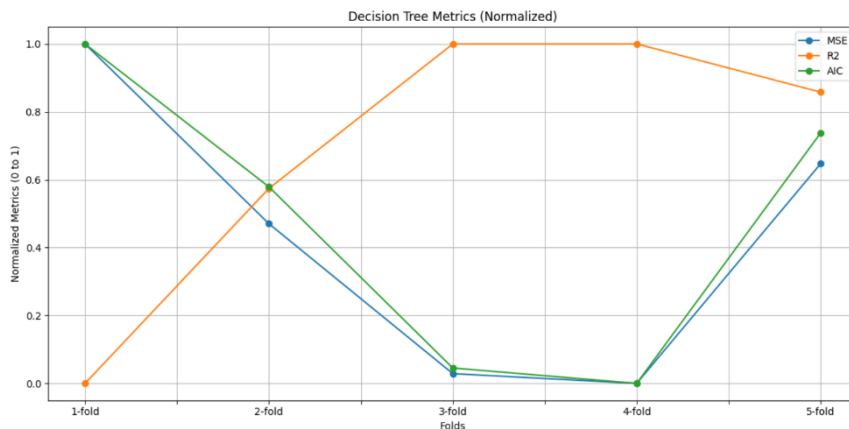


Рис. 3.27. Нормалізований графік метрик DecisionTree

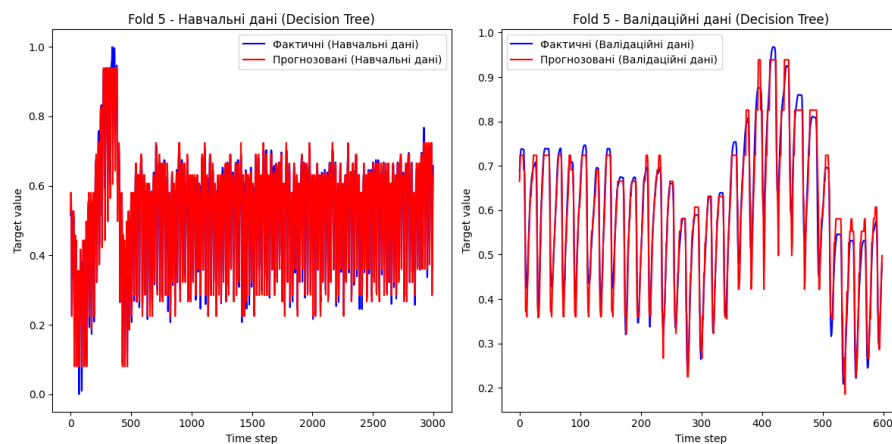


Рис. 3.28. Порівняння фактичних та прогнозованих значень для тестових та валідаційних даних моделлю DecisionTree

NARX. NARX (Nonlinear Autoregressive with Exogenous Inputs) – це нелінійна авторегресійна модель із зовнішніми входами, що використовується для прогнозування часових рядів. Вона ґрунтується на припущенні, що поточне значення залежить як від попередніх значень цільової змінної (авторегресія), так і від попередніх значень зовнішніх входних змінних (екзогенні входи).

Таблиця 3.9. Результати передбачення NARX на тестових даних

№ фолда	MSE	R ²	AIC
Fold 1	0,00130916	0,9246	19577,62
Fold 2	0,00105109	0,9396	19446,10
Fold 3	0,00098448	0,9401	19406,88
Fold 4	0,00080229	0,9491	19284,30
Fold 5	0,00162684	0,9372	19707,75

Низький MSE і високий R² у всіх фолдах свідчать про добру здатність моделі до передбачення з незначною варіабельністю між фолдами. Дані у Fold 5 містять специфічні аномалії, до яких NARX не змогла адаптуватися на основі навчальних залежностей, що призвело до невеликого зниження точності моделі на цьому проміжку.

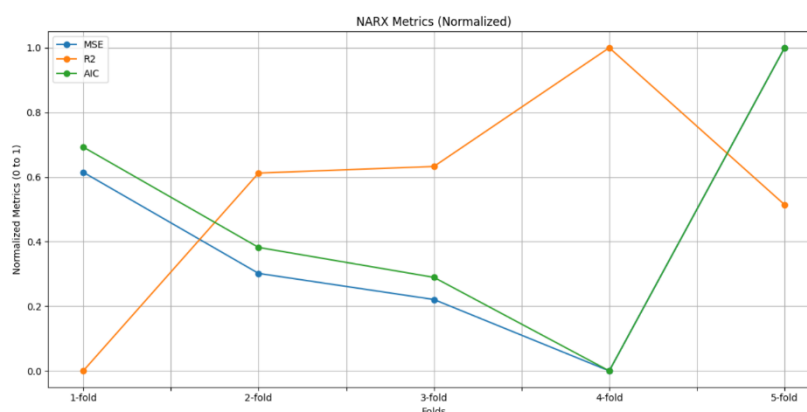


Рис. 3.29. Нормалізований графік метрик NARX Нормалізований графік метрик NARX

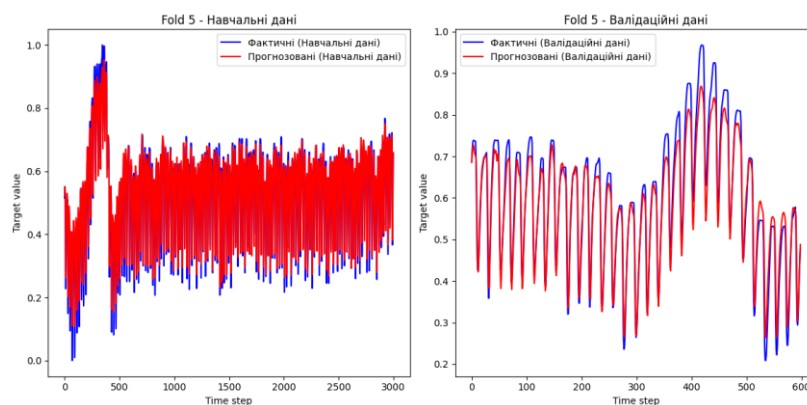


Рис. 3.30. Порівняння фактичних та прогнозованих значень для тестових та валідаційних даних моделлю NARX

RNN. RNN (Recurrent Neural Networks) – це тип нейронних мереж, призначених для обробки послідовних даних, таких як часові ряди, текст або аудіо. Основною особливістю RNN є петлі зворотного зв'язку, які дозволяють моделі "запам'ятовувати" інформацію про попередні елементи послідовності. У Keras модуль SimpleRNN є базовим типом рекурентного шару. Кожен його нейрон пов'язаний із попереднім станом. На кожному кроці SimpleRNN враховує поточний вхід і попередній стан, оновлюючи інформацію про внутрішній стан, що дає змогу моделі враховувати часову залежність.

Модель демонструє значне покращення результатів на кожному наступному фолді, що може свідчити про навчання на різних частинах даних і покращення здатності моделі до узагальнення. Незважаючи на різницю в результатах між фолдами, усі метрики показують досить хороші значення, що свідчить про стабільність моделі.

Таблиця 3.10. Результати передбачення RNN на тестових даних

№ фолда	MSE	R ²	AIC
Fold 1	0,00129290	0,9269	1324,78
Fold 2	0,00068793	0,9611	1046,05
Fold 3	0,00055452	0,9663	818,55
Fold 4	0,00041292	0,9739	642,23
Fold 5	0,00036811	0,9858	573,53

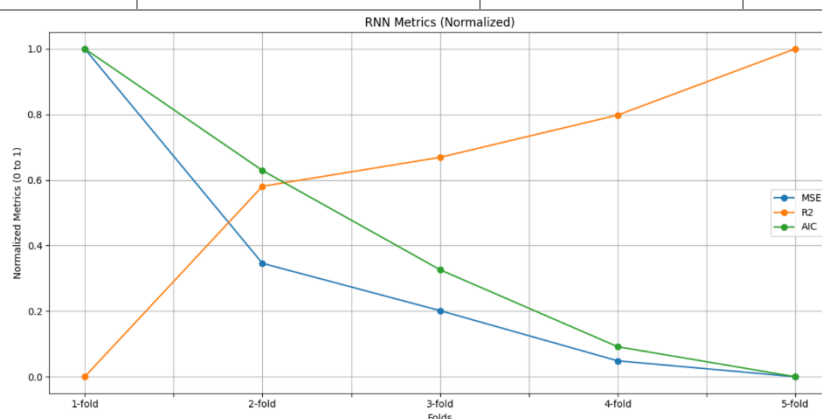


Рис. 3.31. Нормалізований графік метрик RNN

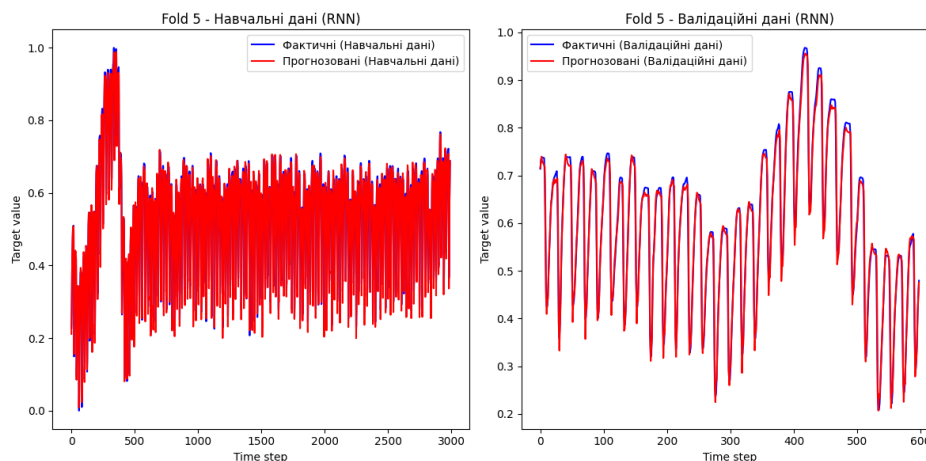


Рис. 3.32. Порівняння фактичних та прогнозованих значень для тестових та валідаційних даних моделлю RNN

GradientBoosting. GradientBoosting – це регресійна модель, що базується на техніці градієнтного бустингу. Вона використовується для створення моделей регресії через послідовне навчання слабких моделей (зазвичай дерев рішень) на помилках попередніх моделей. На кожному етапі тренується нове дерево рішень, яке намагається зменшити помилку попередніх дерев, оптимізуючи градієнт функції втрат.

Модель показує покращення результатів за MSE і R^2 у більшості фолдів, що вказує на підвищення якості прогнозів на кожному наступному етапі. Як і у випадку з моделлю NARX, специфічні аномалії у Fold 5 дещо знижують точність прогнозування моделі.

Таблиця 3.11. Результати передбачення GradientBoosting на тестових даних

№ фолда	MSE	R^2	AIC
Fold 1	0,00048217	0,9722	3258,95
Fold 2	0,00024659	0,9858	3215,35
Fold 3	0,00021137	0,9871	3188,75
Fold 4	0,00015264	0,9903	3176,04
Fold 5	0,00027960	0,9892	3448,75

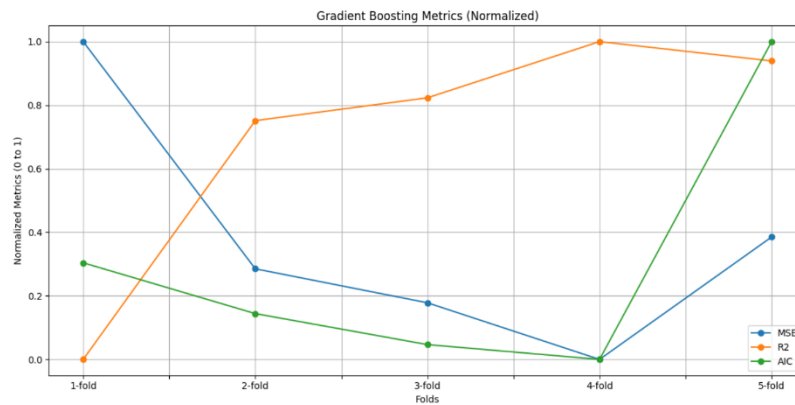


Рис. 3.33. Нормалізований графік метрик GradientBoosting

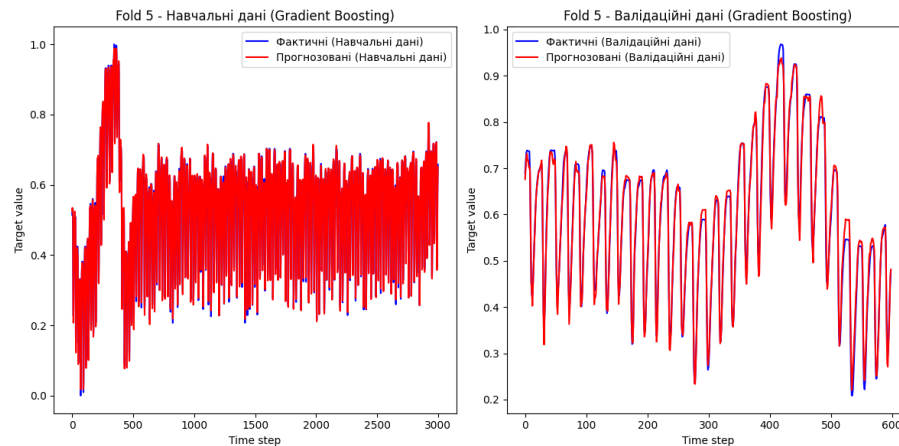


Рис. 3.34. Порівняння фактичних та прогнозованих значень для тестових та валідаційних даних моделлю GradientBoosting

6) Порівняння результатів. Узагальнені результати всіх моделей за всіма фолдами представлені в Табл. 3.12. Хоча всі представлені моделі показали позитивні результати прогнозування, деякі з них продемонстрували суттєві відмінності в точності. Моделі XGBoost, Gradient Boosting та RNN мають значно нижчу середньоквадратичну похибку (Рис. 3.35) та кращу здатність пояснювати варіації даних (Рис. 3.36) порівняно з Decision Tree та NARX.

Таблиця 3.12. Усереднені порівняльні результати моделей за всіма фолдами

Метод	MSE	R ²	AIC
Gradient Boosting	0,00027447	0,9849	3257,57
XGBoost	0,00026294	0,9853	46008,49
RNN	0,00068793	0,9611	881,03
Decision Tree	0,00095305	0,9484	-4131,70
NARX	0,00115477	0,9381	19484,53

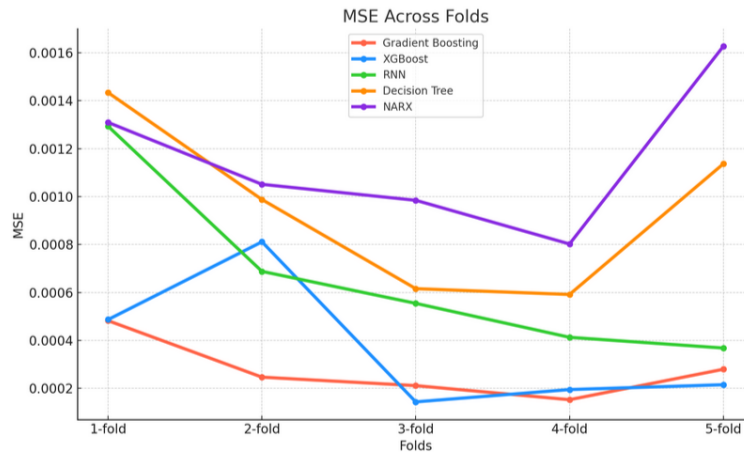


Рис. 3.35. Порівняння значень MSE для всіх моделей

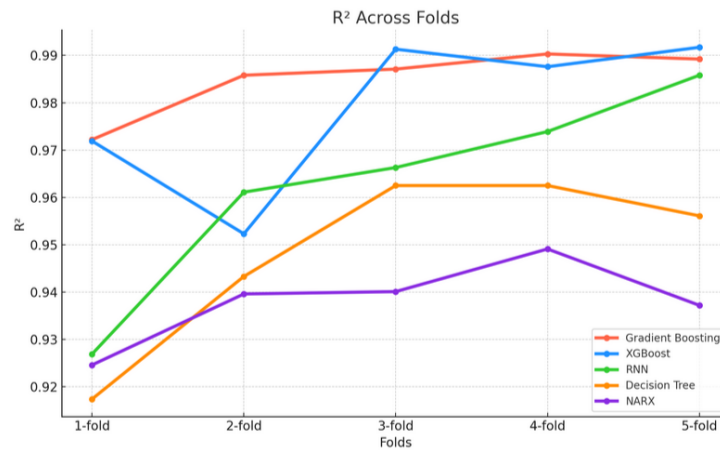


Рис. 3.36. Порівняння значень R^2 для всіх моделей

Порівняння балансу між точністю та складністю моделей за допомогою інформаційного критерію Акаїке показує підвищену складність моделей NARX та XGBoost. Оскільки Decision Tree не відповідає вимогам використання AIC через ймовірнісну природу моделі, що спотворює результати аналізу, найкращими моделями за цим показником є Gradient Boosting та RNN (Рис. 3.37).

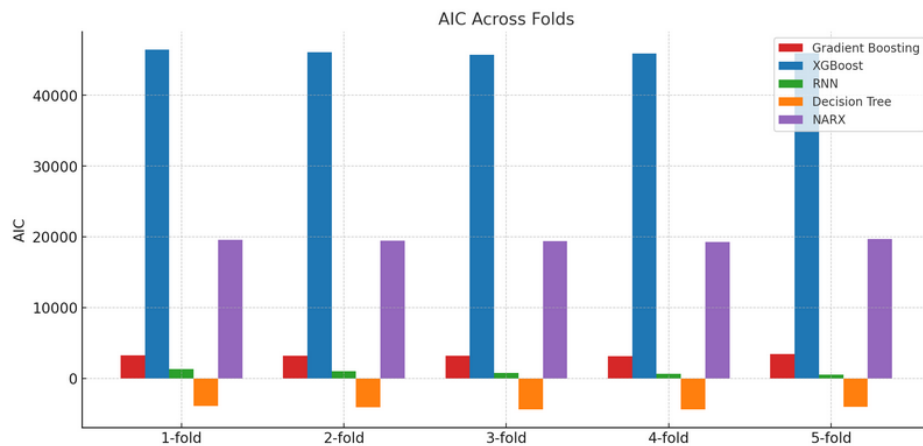


Рис. 3.37. Порівняння значень AIC для всіх моделей

Висновок. На основі результатів дослідження модель ВС здатна пояснити 97.2% варіації цільової змінної з середньоквадратичною похибкою $4,8 \times 10^{-4}$, що дозволяє використовувати її для інтелектуального моніторингу в режимі реального часу.

3.2.2 Моніторинг датчиків, що задіяні в контурах регулювання

Особливе значення можливість інтелектуального моніторингу датчиків відіграє в контурах регулювання. Відсутність інформації про значення змінних в таких контурах не тільки позбавляє інформації про стан системи, а й унеможливорює керування. Завдяки моделюванню значень змінної, ВС дозволяє продовжувати регулювання використовуючи запропоновані моделлю значення для управління.

Використовуючи досліджувані моделі було проведено моделювання цільових змінних для контуру регулювання витрати бражки в дистиляційну колону (Рис. 3.38), та контуру регулювання тиску в колоні концентрації (Рис. 3.39).

Моделювання витрати на основі показників інших сенсорів

Для регулювання подачі бражки на дистиляційну колону використовується датчик витрати R_F1, вихід з ладу якого унеможливорює автоматизоване та ефективне керування процесом (Рис. 3.38). Використання моделей ВС дозволяє з похибкою $2,6 \times 10^{-4}$ прогнозувати витрату бражки на основі показників інших сенсорів та пояснювати 96% варіації змінної, що дає змогу продовжувати роботу навіть при відмові фізичного датчика R_F1 (Табл. 3.13).

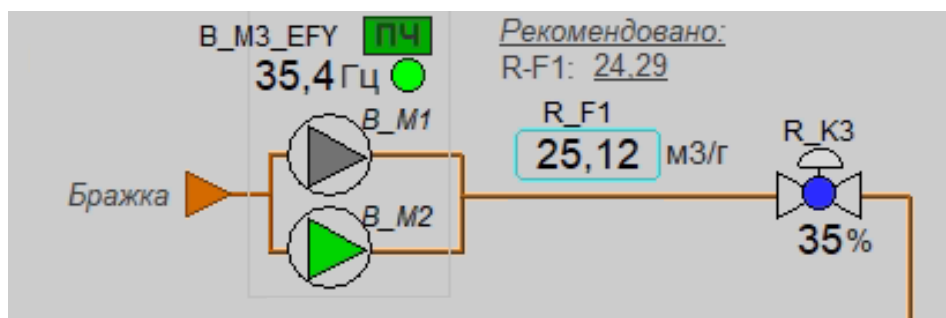


Рис. 3.38. Контур регулювання подачі бражки на дистиляційну колону

Таблиця 3.13. Порівняння метрик моделей для передбачення витрати R_{F1}

Метод	MSE	R ²	AIC
RNN	0.00026666	0.964594	380.741
XGBoost	0,00026294	0.929111	46486.9
GradientBoostingRegressor	0.00056193	0.925459	2727.44
NARX	0.00183831	0.756147	19781
DecisionTreeRegressor	0.00239001	0.682963	-3545.84

Моделювання показників тиску на основі показників інших сенсорів

Датчик тиску R_{P12} у верхній частині колони концентрації забезпечує контроль над тиском пари та дозволяє стабілізувати процес конденсації і забезпечити оптимальну роботу колони (Рис. 3.39). Його поломка призводить до втрати можливості контролювати цей параметр, що може спричинити нестабільність у роботі колони, порушення оптимальних умов процесу з втратою якості кінцевого продукту чи виникнення аварійних ситуацій. Моніторинг та моделювання значень датчика R_{P12} дозволяють продовжувати технологічний процес навіть в разі відмови датчика.

Відповідно до результатів моделювання ВС дозволяє з похибкою $6,3 \times 10^{-4}$ прогнозувати значення тиску та пояснювати 98% варіації змінної R_{P12} (Табл. 3.14).

Також було проведено дослідження зміни точності передбачення R_{P12} за відсутності даних від датчика температури над верхньою тарілкою колони концентрації R_{T13} (Табл. 3.15). Отримані результати вказують на незначне збільшення похибки до $6,7 \times 10^{-4}$ та зменшення значення коефіцієнту детермінації на 0.1% порівняно з моделюванням що включає цю змінну, що демонструє працездатність моделі навіть при відсутності інформації з локаційно близького датчика R_{T13} .

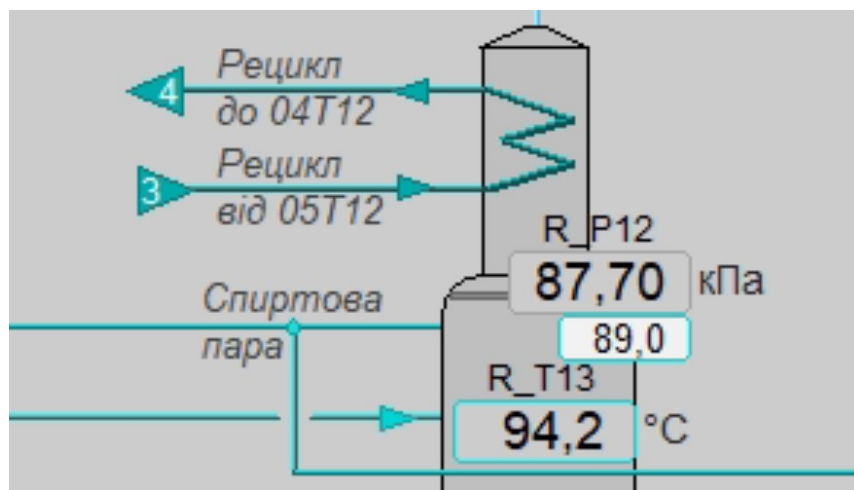


Рис. 3.39. Датчик тиску R_P12 у верхній частині колони концентрації

Таблиця 3.14. Порівняння метрик моделей для передбачення значення тиску R_P12

Метод	MSE	R ²	AIC
XGBoost	0.00063536	0.982343	46590.6
GradientBoostingRegressor	0.00102996	0.971377	3642.26
RNN	0.00133631	0.962543	1344.53
DecisionTreeRegressor	0.00321849	0.910558	-3367.57
NARX	0.00857391	0.76173	20703.3

Таблиця 3.15. Порівняння метрик моделей для передбачення значення тиску R_P12 без даних змінної R_T13

Метод	MSE	R ²	AIC
XGBoost	0.00067615	0.98121	46627.8
GradientBoostingRegressor	0.00102662	0.97147	3643.71
RNN	0.00140524	0.960611	1374.6
DecisionTreeRegressor	0.00232957	0.935261	-3561.18
NARX	0.00917093	0.745139	20359.7

Висновки. У результаті проведених досліджень було продемонстровано ефективність використання моделей ВС для прогнозування критичних параметрів у системах автоматичного регулювання технологічних процесів спиртового виробництва. Зокрема, моделювання витрати бражки на основі

показників інших сенсорів дозволило з коефіцієнтом детермінації $R^2=0,964$ забезпечити безперервність автоматизованого керування навіть у разі відмови датчика витрати R_F1 . Аналогічно, прогнозування тиску в колоні концентрації за відсутності даних від датчика R_P12 здатне пояснити 98% цільової змінної, що підтверджує можливість підтримки стабільної роботи системи при відмові відповідного сенсора.

Отримані результати свідчать про доцільність впровадження інтелектуальних моделей для моніторингу та управління технологічними процесами виробництва спирту, що підвищує надійність та ефективність роботи автоматизованих систем.

3.3 Оптимізація режимів роботи бродильного відділення

3.3.1 Формулювання задачі оптимізації

Бродильне відділення є одним із ключових етапів виробництва етанолу, де відбувається перетворення цукрів у спирт під дією мікроорганізмів. Ефективність цього процесу визначає кінцевий вихід етанолу, енергетичні витрати та має значний вплив на тривалість циклу виробництва. Оптимізація роботи бродильного відділення є водночас важливим та складним завданням, оскільки цей процес має нелінійний, багатofакторний характер та значну інерційність.

Одним із методів вирішення цієї задачі є застосування моделей процесів та зворотного моделювання для отримання оптимізаційних пропозицій щодо зміни налаштувань режимів роботи в залежності від критерію оптимізації та мети. При цьому пропонується застосовувати дерева регресії та ансамблеві методи, тому що саме ці підходи ефективно працюють із нелінійними залежностями та великою кількістю вхідних змінних, що характерно для процесів у бродильному відділенні. Вони дозволяють враховувати взаємозв'язки між різними факторами, такими як температура, час бродіння, концентрація дріжджів тощо. Завдяки цьому можна точно визначити оптимальні налаштування режимів роботи для досягнення бажаної якості продукції та підвищення ефективності процесу. У

даному підході віртуальним сенсором виступатиме модель, побудована на основі дерев регресії або їх ансамблів, яка прогнозує ключові змінні процесу бродіння.

Дерево регресії - це модель машинного навчання, яка використовується для передбачення числових значень. Вона представляє собою деревоподібну структуру, де кожен внутрішній вузол містить умову (поділ), що базується на значеннях вхідних змінних, а листя (кінцеві вузли) представляють передбачене значення.

Основні особливості дерева регресії:

- На кожному рівні дерева здійснюється поділ даних за певною ознакою, що мінімізує похибку прогнозу.
- У кожному листі дерева стоїть значення, яке є середнім для всіх зразків, що потрапляють в цей лист.
- Прогноз для нового зразка отримується шляхом проходження по дереву та обрання значення в листі.

Ансамблі дерев регресії - це методи, які комбінують кілька дерев регресії для покращення точності прогнозів.

Для побудови такої моделі необхідно навчити дерева регресії прогнозувати концентрацію етанолу на основі ключових параметрів процесу бродіння.

Відповідно до даних дослідження Sijing Feng з Університету Саскачеван (Канада) - "Dissolved Carbon Dioxide Driven Repeated Batch Fermentation" Sijing Feng [131] щодо продуктивності раси дріжджів *Saccharomyces cerevisiae* такими параметрами, що впливають на вихідну концентрацію етанолу є початкова концентрація глюкози в суслі, температура процесу, час бродіння, а також рівень розчиненого вуглекислого газу, що підтримується (Табл. 3.16). Матриця кореляцій цих параметрів вказує на значну залежність виходу етанолу від концентрації глюкози та часу процесу, а також помірну залежність від рівня CO₂ (Рис. 3.40).

Таблиця 3.16. Залежність вихідної концентрації етанолу від технологічних параметрів

	~150 g glucose l ⁻¹		~200 g glucose l ⁻¹	
DCO ₂ control level	1000 mg l ⁻¹	750 mg l ⁻¹	1000 mg l ⁻¹	750 mg l ⁻¹
Final ethanol, g l ⁻¹	59.6±2.7	54.2±4.0	70.7±3.4	66.7±1.6
Final biomass, g l ⁻¹	10.7±0.2	10.1±0.3	10.9±0.2	10.1±0.3
Stable cycle time, h	6.6±0.3	6.2±0.2	8.7±0.5	7.3±0.2
Efficiency ¹	0.67 ±0.07	0.62±0.08	0.68±0.06	0.69±0.03
Viability	0.98±0.01	0.99±0.01	0.99±0.01	0.97±0.01
	~250 g glucose l ⁻¹		~300 g glucose l ⁻¹	
Final ethanol, g l ⁻¹	85.1±4.3	83.2±3.0	113.5±7.9	94.3±10.3
Final biomass, g l ⁻¹	9.9±0.7	9.9±0.2	7.4±1.0	8.8±1.5
Stable cycle time, h	14.9±1.9	12.1±1.1	31.5±7.0	21.7±6.0
Efficiency ¹	0.65±0.01	0.69±0.3	0.68±0.02	0.62±0.08
Viability	0.97±0.02	0.99±0.01	0.73±0.08	0.92±0.06

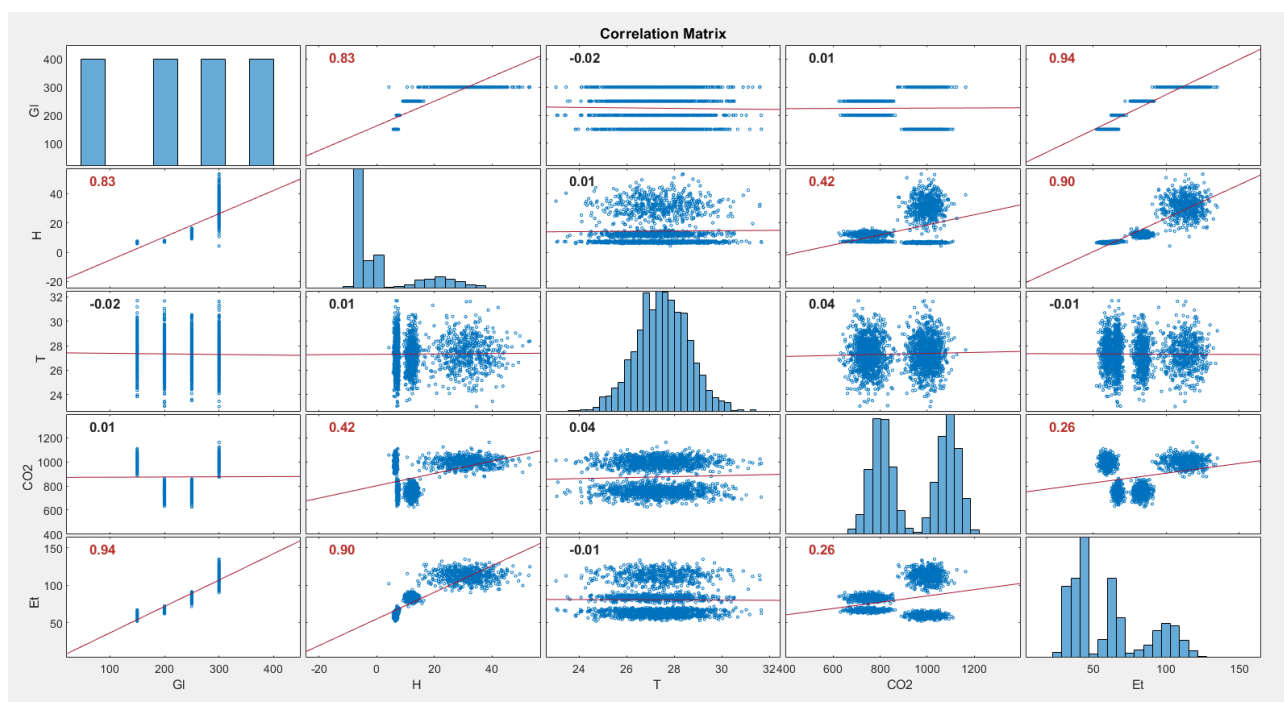


Рис. 3.40. Матиця кореляції процесу спиртового бродіння

3.3.2 Навчання дерев регресії

Засобами середовища Matlab побудовано моделі регресійних дерев на 30, 100 та 400 спостережень (Рис. 3.41-3.43), а також ансамблів за допомогою бустінгу (LSBoost), бегінгу (Bagging) та бегінгу з використанням TreeBagger (Рис. 3.44).

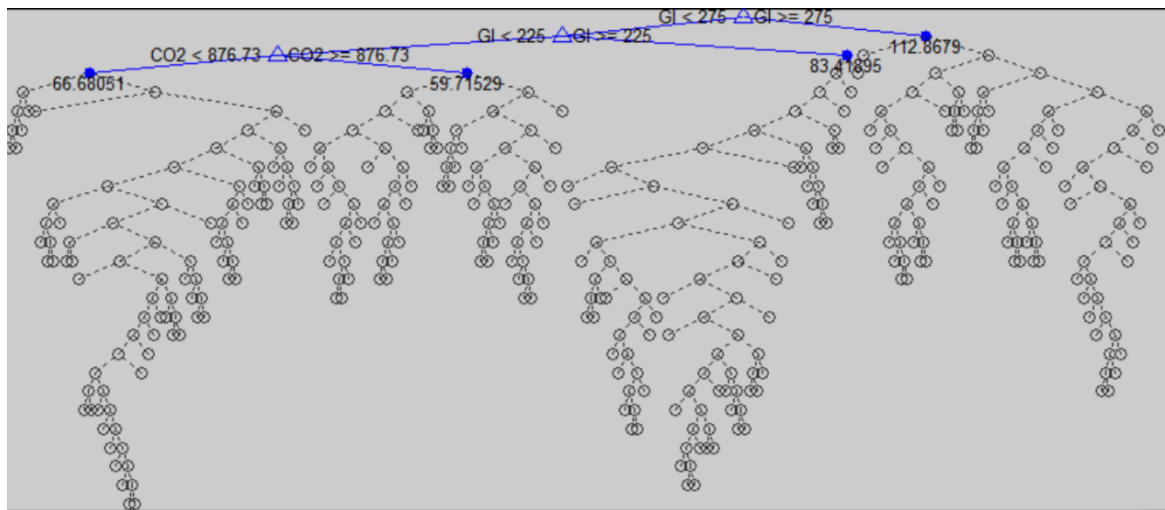


Рис. 3.41. Регресійне дерево на 30 спостережень

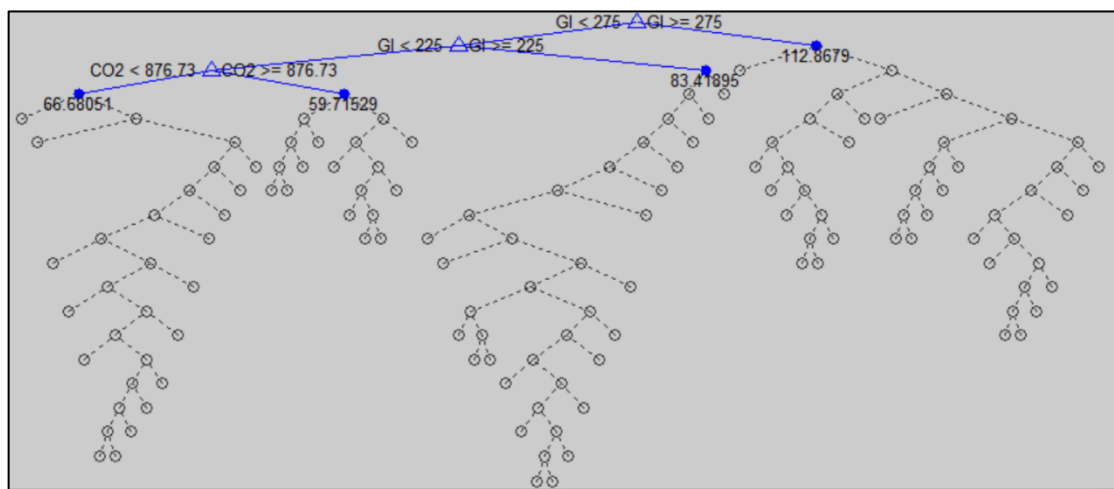


Рис. 3.42. Регресійне дерево на 100 спостережень

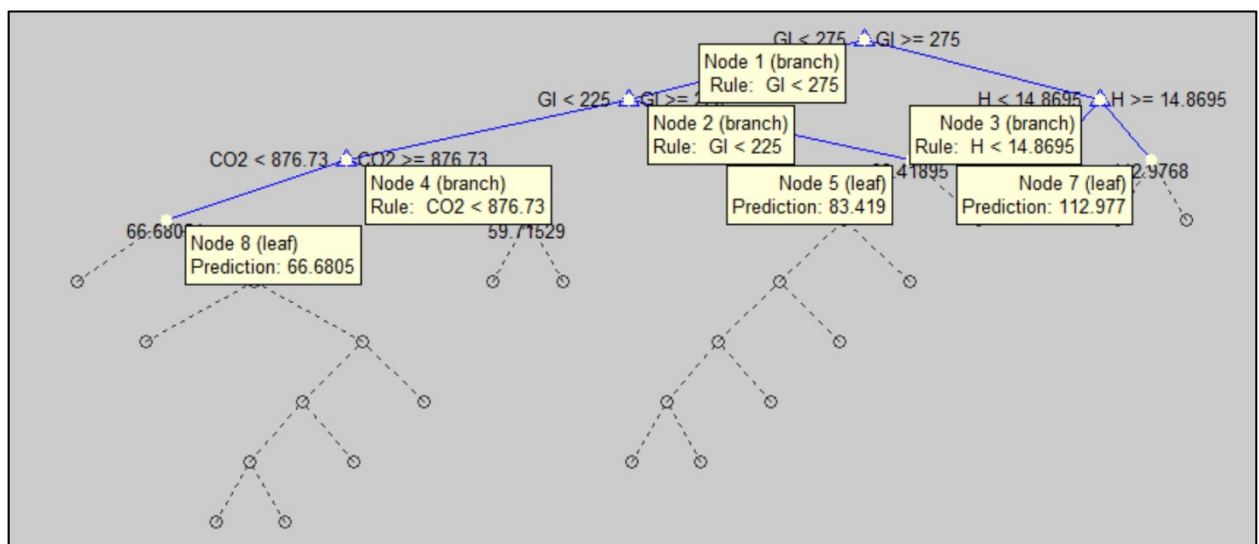


Рис. 3.43. Регресійне дерево на 400 спостережень

Відповідно до оцінки похибок моделей (Табл. 3.17) можна зробити наступні висновки:

- Найкраща модель за помилкою навчання, що найкраще підлаштувалася під навчальні дані є Bagging з помилкою 10.1085
- Найкращою узагальнюючою моделлю (за помилкою перехресної валідації) є дерево регресії з MinParent=400 (помилка 21.2297). Проте LSBoost та Bagging показали дуже близькі результати (21.9267 та 21.9411 відповідно)
- Звичайне дерево регресії (MinParent=30) показує найбільшу різницю між помилками навчання та валідації, що вказує на значне перенавчання. Збільшення параметра MinParent покращує узагальнюючу здатність моделі, але знижує точність на навчальних даних
- Ансамблеві методи (LSBoost та Bagging) показують найкращий баланс між навчанням та узагальненням

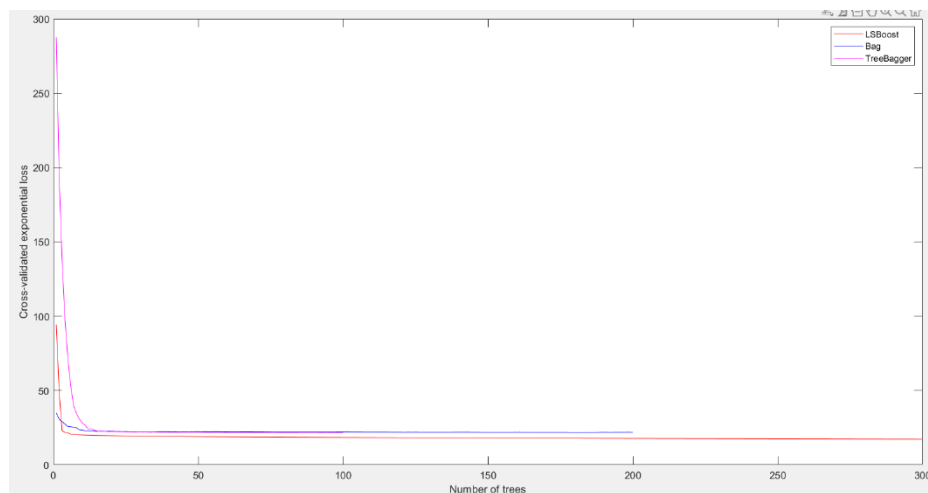


Рис. 3.44. Графіки залежностей перехресних перевірених експоненціальних втрат від кількості дерев регресії для кожного методу побудови ансамблів

Відповідно до отриманих результатів було вирішено обрати Bagging для використання в задачі зворотного моделювання та пошуку оптимізаційних налаштувань оскільки вона враховує баланс між точністю та узагальненням, забезпечує стійкість прогнозів завдяки зменшенню дисперсію прогнозів шляхом усереднення результатів багатьох дерев та при цьому має меншу схильність до перенавчання порівняно з одиночним деревом.

Таблиця 3.17. Залежність вихідної концентрації етанолу від технологічних параметрів

Модель	Помилка навчання	Помилка перехресної валідації
Дерево регресії (MinParent=30)	12.71	27.85
Дерево регресії (MinParent=100)	16.23	23.20
Дерево регресії (MinParent=400)	19.68	21.23
LSBoost	17.27	21.93
Bagging	10.11	21.94
TreeBagger	15.86	29.60

3.3.3 Оптимізація режимів роботи відділення

В рамках бродильного відділення надається приклад реалізації задачі оптимізації для двох можливих критеріїв – максимізація виходу етанолу та мінімізація часу зброджування зі збереженням заданої кількості етанолу. Насправді, критерій може бути й інший.

1) Задача збільшення продуктивності етанолу

Оскільки концентрація глюкози залежить від хімічного складу сировини і не може бути відрегульована напряму то суть задачі оптимізації полягає в підборі таких налаштувань температури, рівня CO₂ та тривалості процесу, щоб максимізувати вихід етанолу.

Для цього було використано алгоритм Patternsearch, який працює шляхом оцінювання функції в точках, що утворюють певний шаблон (pattern) та послідовно адаптує розмір та напрямок пошуку враховуючи задані обмеження (наприклад максимальний час перебігу процесу). Зворотне моделювання з використанням Patternsearch на ансамблі дерев регресії дозволяє підібрати оптимальні параметри при яких досягається збільшення виходу етанолу в середньому на 8.4% (Рис. 3.45).

```
Фінальні результати оптимізації:  
Фіксовані параметри:  
G1: 166.00 (фіксоване значення)  
  
Оптимальні значення керованих параметрів:  
H: 22.66  
T: 26.19  
CO2: 946.77  
Максимальне Et: 88.22  
  
Початкове Et: 77.75  
Покращення: +13.46%
```

Рис. 3.45. Пошук оптимальних параметрів для збільшення результуючої концентрації етанолу за допомогою Patternsearch

2) Задача мінімізації часових затрат на бродіння

Для підбору параметрів, які б дозволили досягнути мінімального часу протікання процесу зі збереженням заданого рівня виходу етанолу було використано алгоритм Fmincon (Find Minimum of Constrained Nonlinear Multivariable Function) з методом SQP (Sequential Quadratic Programming). Метод SQP послідовно розв'язує квадратичні апроксимації початкової задачі, використовуючи градієнтні та другорядні похідні цільової функції та обмежень. На кожному кроці оптимізації розв'язується допоміжна квадратична підзадача, що апроксимує вихідну нелінійну задачу та визначає напрямок руху до оптимального розв'язку.

Модель отримала завдання з обмеженням мінімального рівня втрати кількості етанолу при пошуку оптимальних налаштувань, та показала здатність знаходити оптимальні налаштування що дозволяють зменшити час перебігу процесу в середньому на 12.2% (Рис. 3.46).

```
Фіксовані параметри:  
Gl: 166.00  
  
Оптимальні значення керованих параметрів:  
H: 8.74  
T: 28.10  
CO2: 596.23  
Концентрація Et: 73.45  
  
Порівняння результатів:  
H: 15.00 → 8.74 (зміна: -41.76%)  
T: 26.00 → 28.10 (зміна: 8.06%)  
CO2: 750.00 → 596.23 (зміна: -20.50%)  
Et: 77.75 → 73.45 (зміна: -5.53%)  
  
Перевірка обмежень:  
Цільовий діапазон Et: 73.86 - 81.64  
Отримане значення Et: 73.45
```

Рис. 3.46. Пошук оптимальних параметрів для зменшення тривалості бродіння в межах заданого рівня виходу етанолу

Можливі варіанти застосування оптимізації на базі ВС та дерев регресії в спиртовому та біоетанольному виробництві не обмежуються бродінням та можуть бути використані для наступного переліку задач:

1) Відділення підготовки сировини та варки сусла

а. Оптимізація температурних режимів розварювання - налаштування температурних кривих для максимального розщеплення крохмалю при мінімальних енерговитратах.

б. Оптимізація дозування ферментних препаратів - розрахунок точних доз амілолітичних ферментів залежно від якості сировини для максимізації виходу зброджуваних цукрів.

с. Оптимізація рН середовища - підбір оптимальних значень рН для максимальної активності ферментів на різних етапах підготовки сусла.

2) Бродильне відділення

а. Оптимізація кількості засівних дріжджів - розрахунок оптимальної дози внесення дріжджів залежно від концентрації сусла та умов бродіння.

б. Оптимізація режиму аерації - налаштування інтенсивності та тривалості аерації на початковому етапі бродіння для оптимального розмноження дріжджів.

с. Оптимізація температурного профілю бродіння - створення оптимальної температурної кривої для різних фаз бродіння.

3) Дистиляційне або ректифікаційне відділення

а. Оптимізація режимів роботи колон - підбір параметрів для мінімізації втрат етанолу.

б. Оптимізація відбору головної фракції - визначення оптимального обсягу та швидкості відбору для ефективного видалення домішок.

с. Оптимізація відбору хвостової фракції - розрахунок оптимального обсягу та моменту відбору для балансу між чистотою продукту та виходом спирту.

4) Загальні задачі оптимізації:

а. Оптимізація енергоспоживання - розподіл енергоресурсів між ділянками виробництва та налаштування режимів роботи обладнання для мінімізації загальних енерговитрат.

б. Оптимізація логістики внутрішніх потоків - синхронізація роботи всіх відділень для забезпечення безперервності виробництва.

с. Оптимізація переходу між різними видами сировини - розрахунок оптимальних параметрів для швидкого та ефективного переналаштування всіх процесів.

Висновки до розділу 3

1. У рамках проведеного дослідження розроблено динамічну модель дистиляційної колони з використанням NARX-моделей, яка продемонстрували високу ефективність. Зокрема, NARX-модель змогла пояснити 99% варіацій цільової змінної при прогнозуванні концентрації етанолу, що підтверджує її точність та надійність. Дослідження також включало аналіз впливу зміни параметрів NARX-моделей на їх продуктивність, що дозволило визначити оптимальні налаштування для роботи з технологічними даними. Крім того, було запропоновано та протестовано бібліотеки для автоматизованого підбору налаштувань NARX-моделей, що значно пришвидшило процес моделювання та підвищило точність прогнозування. Ці результати свідчать про великий

потенціал використання NARX-моделей для моніторингу та керування технологічними процесами у спиртовій галузі.

2. Досліджено ефективність інтелектуального моніторингу динамічних змінних у дистиляційному відділенні спиртового виробництва за допомогою методів XGBoost, Decision Tree, NARX, Gradient Boosting та RNN. Проведене моделювання підтвердило можливість компенсації функціоналу сенсорів у разі їхньої відмови або виходу з ладу, що забезпечує безперервність автоматизованого керування процесом. Зокрема:

- Моделювання витрати бражки на основі показників інших сенсорів дозволило компенсувати відмову датчика R_F1 з коефіцієнтом детермінації $R^2=0,964$.

- Прогнозування тиску в колоні концентрації за відсутності даних від датчика R_P12 здатне пояснити 98% цільової змінної, що підтверджує можливість підтримки стабільної роботи системи при відмові відповідного сенсора.

- Компенсація відмови датчика температури R_T9 дозволила пояснити 97,2% варіації цільової змінної, що підтверджує ефективність моделі для інтелектуального моніторингу в режимі реального часу.

Таким чином, використання запропонованих моделей дозволяє підвищити стійкість системи моніторингу та автоматизованого керування процесами дистиляції, забезпечуючи їхню безперервність навіть у разі виходу з ладу окремих фізичних сенсорів.

3. Розроблено алгоритм оптимізації, що визначає технологічні режими на основі моделей регресійних дерев. Наведено реалізацію для оптимізації режимів роботи бродильного відділення, що дає змогу знаходити оптимальні налаштування процесу в умовах зміни характеристик сировини відповідно до критерію оптимізації, що дозволяє збільшити вихід етанолу в середньому на 8.4%, або зменшити час бродіння на 12,2%.

РОЗДІЛ 4. ІНТЕЛЕКТУАЛЬНЕ КЕРУВАННЯ ТА АПРОБАЦІЯ ВІРТУАЛЬНИХ ІНСТРУМЕНТІВ У ВИРОБНИЦТВО

4.1 Концептуальна структура автоматизованої системи моніторингу та керування технологічними процесами

Представлена в роботі концепція системи моніторингу та керування виробництвом спирту та біоетанолу містить інтегровані модулі віртуальних інструментів контролю, які реалізують аналітичний та управляючий функціонал для підвищення ефективності системи. В рамках даного дослідження розглядаються наступні модулі, інтегровані в єдину інформаційну систему моніторингу та керування спиртовим виробництвом (Рис. 4.1):

- 1) **Модуль класифікації стабільності процесу** аналізує поточний стан відділень в реальному часі шляхом перевірки належності розподілу значень змінних процесів до стабільного класу. В разі виявлення відхилень інформує оператора про потенційні проблеми.
- 2) **Модуль моніторингу стану датчиків** порівнює дані від фізичних датчиків з їх розрахованими за допомогою ВС значеннями. У разі виявлення заданої різниці між даними модуль інформує обслуговуючий персонал в необхідності проведення профілактичного обслуговування фізичного сенсора або калібрування моделі. Може використовуватися як ВС у разі поломки фізичного сенсора.
- 3) **Модуль прогнозування та оптимізації режимів** передбачає поведінку окремих змінних (в тому числі невимірюваних, лабораторних показників) та поведінку системи при заданих налаштуваннях. Використовує прогнозовані значення, як завдання модулю регулювання, а також виконує задачу оптимізації шляхом підбору та рекомендації налаштувань для досягнення поставленої виробничої задачі.
- 4) **Модуль автоматичного керування/регулювання** здатний регулювати перебіг технологічного процесу використовуючи модель для попереднього навчання управляючих сигналів. Використовує Reinforcement Learning для

навчання, здатний донавчатися та використовувати передбачені ВС вхідні дані для компенсації відсутньої інформації з фізичних датчиків.

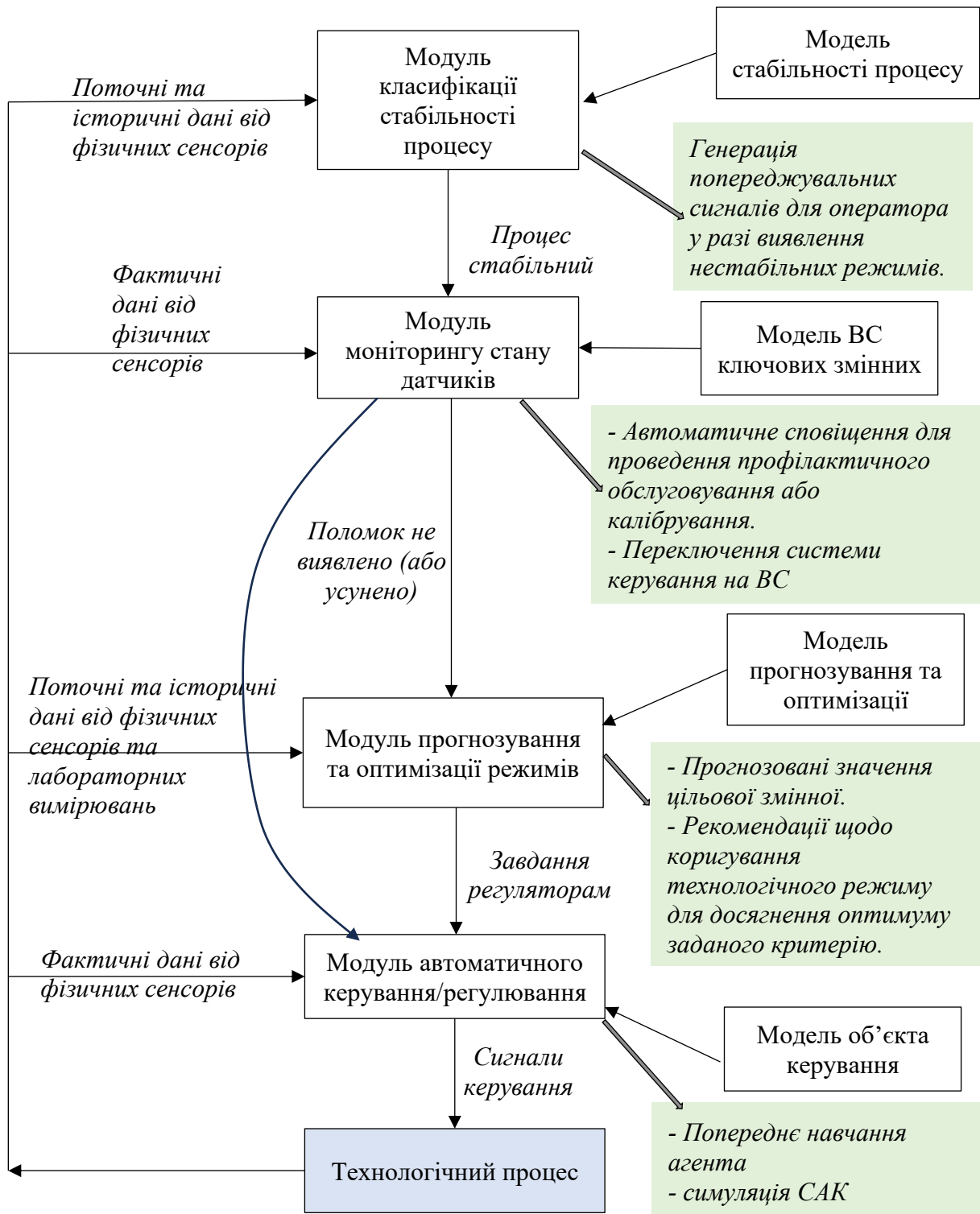


Рис. 4.1. Схема концептуальної структури виробництва спирту та біоетанолу з модулями віртуальних інструментів контролю

4.2.1 Розробка RL-системи керування

4.2.2 Вибір RL-агентів для використання в умовах специфічних для спиртового виробництва

У сфері глибокого навчання з підкріпленням (Deep Reinforcement Learning, DRL) створено низку алгоритмів, які здатні ефективно функціонувати в неперервних просторах дій. Для управління бродильними апаратами, установками для зневоднення, сепараторами та дистиляційними колонами з перехресними зв'язками, де простір дій є неперервним, найкраще використовувати off-policy алгоритми. Оскільки така система має складні взаємодії між входами і виходами, off-policy методи дозволяють агенту навчатися оптимальній політиці незалежно від поточної стратегії дій. Це важливо, оскільки можна використовувати історичні дані або дані з інших політик, що прискорює процес навчання та дозволяє адаптуватися до змінних умов. Крім того, off-policy алгоритми, як TD3, SAC або DDPG, зазвичай демонструють кращу стабільність і швидше сходяться, що важливо для високодинамічних промислових процесів, які характерні для спиртового виробництва. Ці методи дають змогу агенту оптимізувати політику для більш загальних умов, що підходить для задач з багатозмінними процесами, де постійно змінюються вхідні параметри.

Deep Deterministic Policy Gradient (DDPG) є офлайн-алгоритмом, що поєднує підхід актор-критик з детерміністичним підходом до політики і використовує дві нейронні мережі: одну для оцінки політики (актор) та іншу для оцінки значення (критик) [102]. DDPG ефективно працює з неперервними просторами дій, однак може бути чутливим до налаштувань гіперпараметрів та схильним до перенавчання, що сприяло подальшим дослідженням та розробці на його базі алгоритму TD3.

Twin Delayed DDPG (TD3) є вдосконаленням DDPG, яке включає додаткові механізми для покращення стабільності навчання. Він використовує два критики для зменшення переоцінки значень та затримку в оновленні політики, що

допомагає уникнути нестабільності [103-104]. TD3 показує кращі результати в порівнянні з DDPG у багатьох завданнях з неперервними просторами дій.

Soft Actor-Critic (SAC) є офлайн-методом глибокого підкріпленого навчання, який поєднує підхід актор-критик з максимізацією ентропії. Це дозволяє агенту не лише максимізувати очікувану винагороду, але й діяти більш випадковим чином, що сприяє кращій стабільності та ефективності навчання.

Основною особливістю SAC є використання стохастичної політики, що дозволяє агенту ефективно досліджувати середовище та уникати передчасної конвергенції до підоптимальних рішень. Це досягається завдяки максимізації ентропії, що стимулює агентів до більш випадкових дій, сприяючи кращому вивченню середовища. SAC демонструє високу ефективність у завданнях з неперервними просторами дій, перевершуючи попередні методи за швидкістю навчання та якістю результатів. Алгоритм також менш чутливий до налаштувань гіперпараметрів, що спрощує процес налаштування та підвищує надійність результатів. Однак навчання SAC може вимагати значних обчислювальних ресурсів, особливо при роботі з великими та складними середовищами. Імплементація SAC може бути складною через необхідність точного налаштування та інтеграції різних компонентів алгоритму [105-107].

Відповідно до проведеного аналізу, для реалізації RL-системи в спиртовому виробництві було обрано алгоритми TD3 та SAC. TD3 (рис. 4.2) забезпечує стабільність навчання завдяки подвійним критикам і затримці оновлення політики, що ідеально підходить для складних багатозмінних процесів, таких як управління бродильними апаратами та дистиляційними колонами. SAC (рис. 4.3), у свою чергу, завдяки стохастичній політиці та максимізації ентропії, сприяє ефективному дослідженню середовища, що дозволяє адаптуватися до змінних умов виробництва. Ці алгоритми оптимально відповідають вимогам неперервного простору дій і високодинамічних промислових процесів.

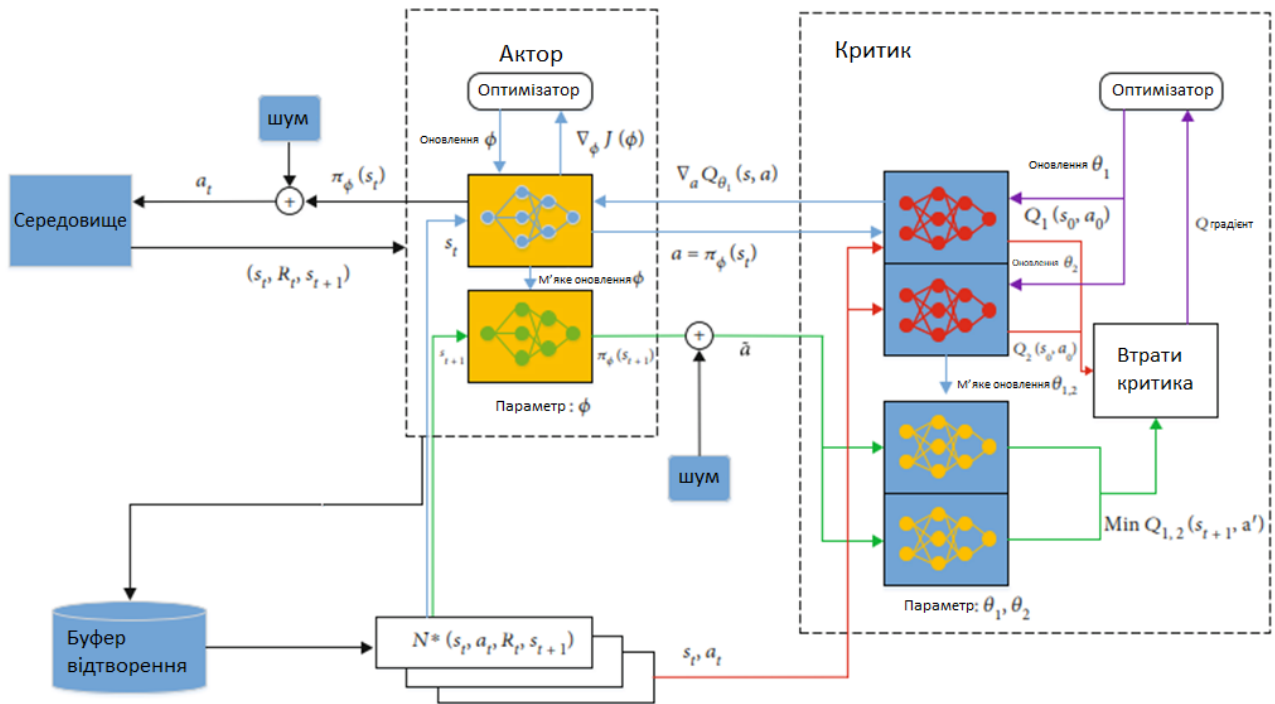


Рис. 4.2. Архітектура TD3 алгоритму

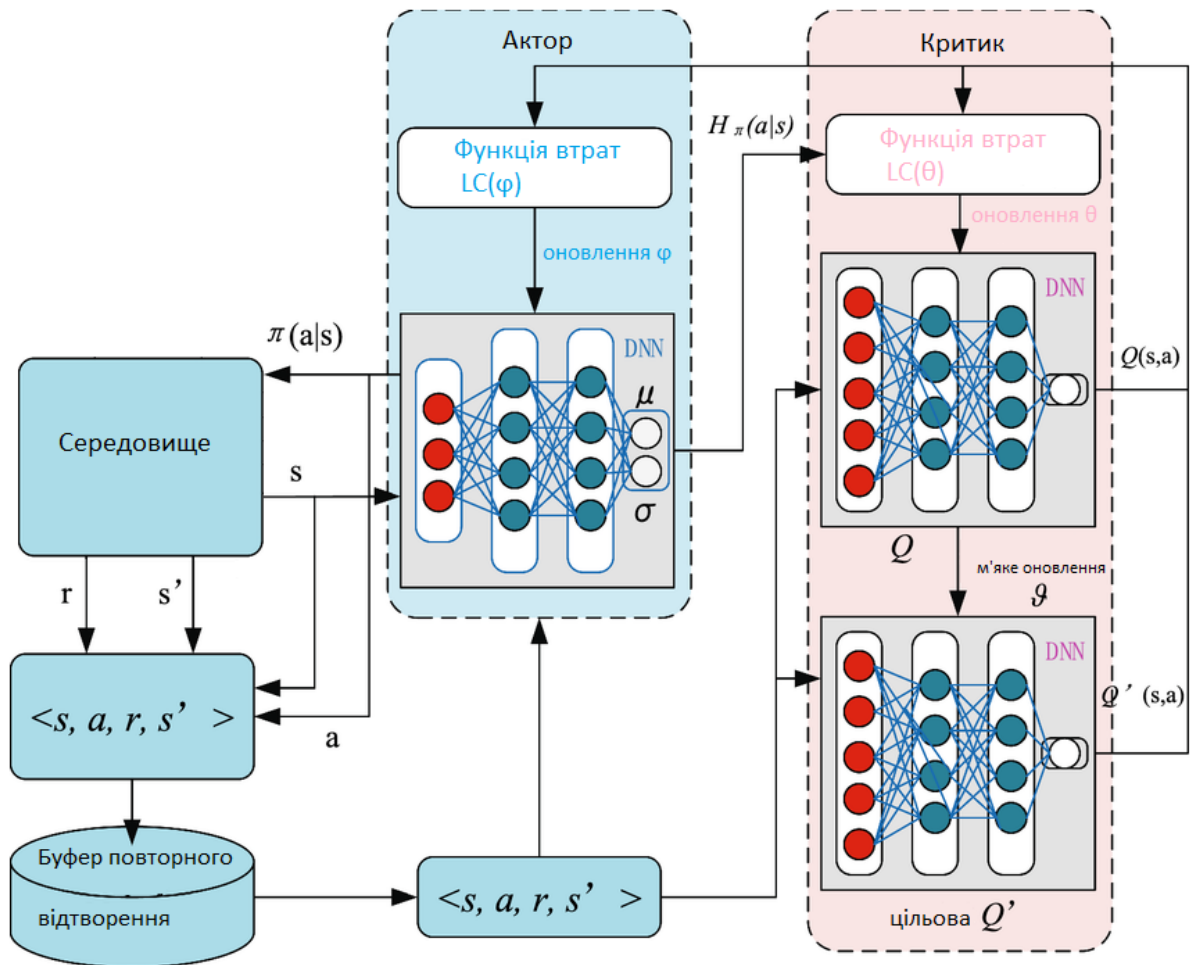


Рис. 4.3. Архітектура SAC алгоритму

4.2.3 Дослідження об'єкту керування

В якості об'єкта для апробації впровадження RL-системи було обрано дистиляційну колону, оскільки вона є складним, багатозв'язним об'єктом із суттєвим запізненням та нелінійностями, що ускладнює її регулювання та формує високі вимоги до регуляторів (рис. 4.4). Математична модель представлена у вигляді [149]:

$$\begin{aligned} y_D(s) &= \frac{12.8}{16.7s + 1} e^{-1s} R(s) + \frac{-18.9}{21s + 1} e^{-3s} S(s) \\ y_B(s) &= \frac{6.6}{10.9s + 1} e^{-7s} R(s) + \frac{-19.4}{14.4s + 1} e^{-3s} S(s) \end{aligned} \quad (4.1)$$

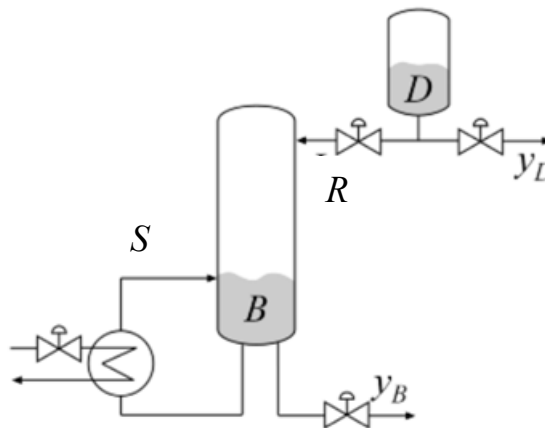


Рис. 4.4. Спрощена структура дистиляційної колони

У даній конфігурації RS (потік флегми і пари) контрольованими змінними є концентрації компонентів (wt.%) у верхньому (дистиллят) і нижньому (продукт на дні) потоках, тобто y_D (концентрація у дистилляті) і y_B (концентрація у продукті на дні). Керованими змінними є потік (витрата lb/min) флегми (рефлюкс R) та пари (S) (Рис. 4.4).

Процес дистиляції в такій конфігурації демонструє кілька важливих аспектів:

- зміни в рефлюксовому потоці R безпосередньо впливають на концентрацію компонентів у дистилляті y_D та продукті на дні y_B . Наприклад, збільшення R може підвищити ефективність розділення компонентів, що змінює їхні

концентрації в обох потоках. Аналогічно, пара (S) також визначає температуру і фазовий склад у колоні, що впливає на розподіл компонентів між рідиною і паром, що, в свою чергу, коригує концентрацію в обох потоках.

- процес демонструє значні коливання у стані стаціонарного підсилення при деяких комбінаціях параметрів R і S. Це може бути спричинено перехідними процесами, які відбуваються при зміні значень рефлюксу та кипіння. Оскільки R і S взаємно впливають один на одного, навіть незначні зміни одного параметра можуть призвести до значних змін у іншому.

В таких умовах важливо точно налаштовувати систему керування для мінімізації коливань і стабілізації цих змінних. Це також може вимагати коригування параметрів моделі, таких як коефіцієнти масопереносу, щоб вони відображали зміни у реальному процесі, особливо при різних значеннях R і S.

Відповідно до опису моделі колони, для деяких поєднань R і S система може демонструвати стабільне підсилення в стаціонарному режимі, коли зміни цих змінних ведуть до постійних відхилень у концентраціях y_D та y_B . Це може бути пов'язано із зниженням або підвищенням ефективності відділення компонентів, що, в свою чергу, змінює робочі умови колони і впливає на стійкість системи.

При значних коливаннях R і S, система може переходити в нові динамічні стани, що може вимагати коригування параметрів моделі для точного опису поведінки.

Невизначеності в об'єкті. Зміни в коефіцієнтах підсилення моделі та в постійних часу можуть бути значними в залежності від умов експлуатації, таких як зміна потоків, температури, тиску, складу вхідного потоку або навіть знос обладнання. Відповідно до розглянутої моделі колони вплив невизначеності може змінити у моделі:

- коефіцієнти підсилення;
- постійні часу;
- структуру моделі (неврахована динаміка або її зміна).

Зміна швидкості потоку флегми R або пари S може призвести до змін в ефективності масопереносу, що безпосередньо відобразиться на коефіцієнтах підсилення. Наприклад, збільшення потоку рефлюксу може призвести до підвищення ефективності масопереносу, що збільшить коефіцієнти, а зменшення потоку може знизити ці коефіцієнти. Орієнтовно зміни в потоці можуть привести до змін коефіцієнтів на 10-30% в залежності від специфіки системи. У деяких випадках зміни можуть бути більшими, особливо якщо система зазнає значних коливань в умовах потоків.

Якщо температура або тиск змінюються, це може змінити летючість компонентів і, відповідно, коефіцієнти фазового розподілу. Це може вплинути на значення коефіцієнтів підсилення. Орієнтовно температурні коливання можуть призвести до змін коефіцієнтів на 5-15%, залежно від розміру зміни температури та її впливу на властивості рідини та пари.

Також зміни в складі вхідного потоку можуть призвести до зміни концентрацій компонентів у коліні, що вплине на ефективність розділення і, відповідно, на коефіцієнти масопереносу. Орієнтовно зміни в складі вхідного потоку можуть змінити коефіцієнти на 5-20% в залежності від характеру змін у складі.

Постійні часу в математичній моделі, визначають час, за який система досягає певної частини стабільного стану після зміни вхідного сигналу. Цей параметр можна змінювати залежно від кількості тепло- та масопереносу в системі, а також від ефективності внутрішніх елементів (тарілок, насадок) колони. Збільшення забруднень або знос внутрішніх елементів колони (наприклад, тарілок чи насадок) може призвести до зниження ефективності масопереносу і теплопереносу, що впливає на динаміку системи і, відповідно, на постійні часу. У разі зниження ефективності процесу постійна часу може збільшитися, що означає повільніше досягнення стабільного стану. Якщо внутрішні елементи колони сильно забруднені або зношені, постійна часу може зрости на 20-50% в залежності від рівня зносу.

Також, збільшення швидкості потоку може зменшити час досягнення стабільного стану, тобто постійні часу можуть зменшитися. Зменшення потоку може призвести до збільшення постійних часу, оскільки система стане менш ефективною в перенесенні маси та тепла. Орієнтовно зміни в потоці можуть змінити постійні часу на 10-25%, в залежності від змін швидкості потоку. Так само температурні та коливання тиску можуть вплинути на швидкість процесів, що відбуваються в колоні (особливо на процеси кипіння та конденсації). Це може призвести до зміни постійних часу, оскільки швидкість переносу маси та тепла змінюється в залежності від цих умов. Орієнтовно температурні та коливання тиску можуть змінити постійні часу на 10-30%.

Таким чином, матриця коефіцієнтів підсилення можуть змінюватися на 10-30% в залежності від змін в потоках, температурі, тиску або складі вхідного потоку, а постійні часу можуть змінюватися на 10-50% в залежності від зносу обладнання, зміни швидкості потоку або температурних коливань. Ці зміни можуть бути адаптовані в реальному часі, використовуючи методи адаптивного або робастного керування, або при застосуванні технологій, таких як reinforcement learning (RL), для оптимізації процесу в колоні дистиляції. Таким чином, найгірший сценарій для системи керування полягає в тому, що коефіцієнти моделі можуть знизитися на 30%, а постійна часу може збільшитися на 50%, що може призвести до значної нестабільності і потреби в постійному коригуванні параметрів системи управління.

Зміни коефіцієнтів в моделі та постійних часу, залежно від умов і характеру збурень, можуть бути як плавними, так і стохастичними. Плавні зміни в системі колони дистиляції зазвичай пов'язані з регулюванням керувальних параметрів від інших автоматичних систем керування, змінами в складі вхідного потоку або змінами температури та тиску. Стохастичні зміни виникають через випадкові збурення, такі як механічні збої в обладнанні або випадкові коливання в складі потоку. У реальних умовах, зміни часто поєднують обидва типи — плавні зміни з коригуванням процесу, а також стохастичні збурення, які виникають через непередбачувані фактори.

4.2.4 Моделювання системи з локальними ПІ-регуляторами

Одним із класичних способів регулювання дистиляційної колони є використання ПІ-регуляторів (PI-). В досліджуємому об'єкті відбувається стабілізація складу дистиляту (y_D або x_D на схемі) і кубового залишку (y_B або x_B на схемі) з використанням регуляторів - двох дискретних ПІ- алгоритмів (Рис. 4.5, 4.6). Вони призначені для керування концентраціями у верхньому і нижньому продуктах шляхом регулювання швидкості потоку флегми (R) і пари (S).

В табл. 4.1 наведені параметри системи керування з ПІ-регуляторами, які отримані з використанням оптимального інструменту Response Optimizer (Matlab), який визначає оптимальні налаштування за видом заданого перехідного процесу, що визначається встановленими межами, з додатковим ручним підлаштуванням. Для покращення якості перехідних процесів ПІ-алгоритмів у об'єкті, що характеризується зв'язаними каналами, використовують компенсатор – автономна система регулювання (розрахунок компенсатора наведений в Додатку Є). А на рис. 4.7 (а, б) та 4.8 (а, б) - сигнал керування та перехідні процеси відносно зміни завдання. Так додавання матриці розв'язування перехресних каналів компенсує взаємний вплив між каналами, що дозволяє значно зменшити статичну та динамічну похибки, а також скоротити час перехідного процесу (Табл. 4.2).

Таблиця 4.1. Налаштування контролера на основі ПІ-регуляторів та компенсатора

Регулятор	Налаштування	Особливості
ПІ-регулятори без компенсатора		
PI_R (потік флегми R)	$K_p = -0.2$ $K_i = -0.09$	Дискретний ПІ-регулятор
PI_S (потік пари S)	$K_p = 0.04$ $K_i = 0.03$	Дискретний ПІ-регулятор
ПІ-регулятори з компенсатором		
PI_R (потік флегми R)	$K_p = -0.2$ $K_i = -0.02$	Дискретний ПІ-регулятор
PI_S (потік пари S)	$K_p = 0.09$ $K_i = 0.16$	Дискретний ПІ-регулятор
DM	$W_{komp} = [2.01 \ -1.96 \ ; \ -1.04 \ 2.01]$	Компенсатор

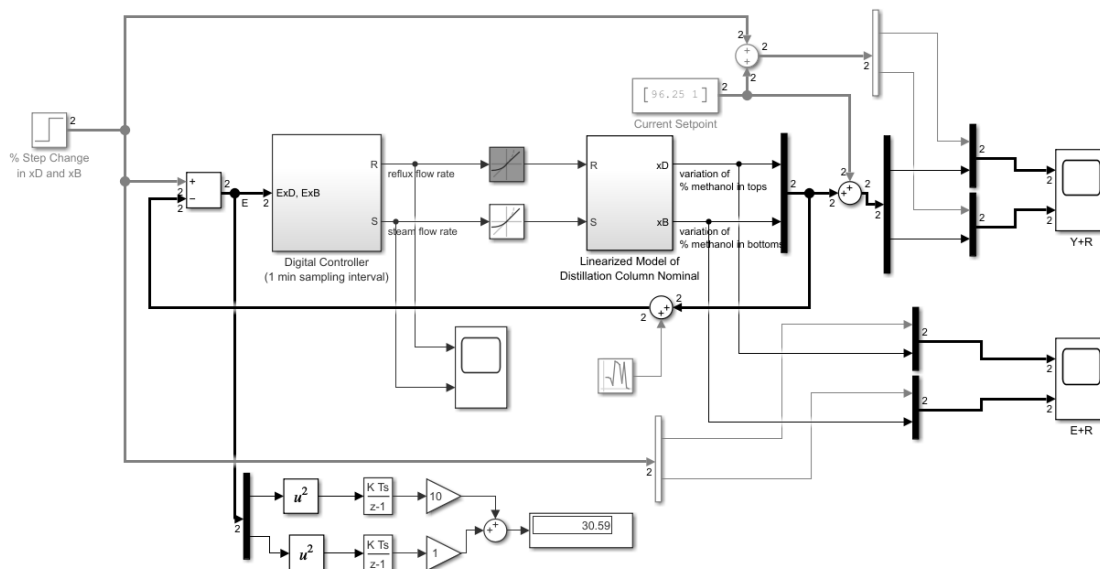


Рис. 4.5. Система керування дистиляційної колони з ПІ-регуляторами

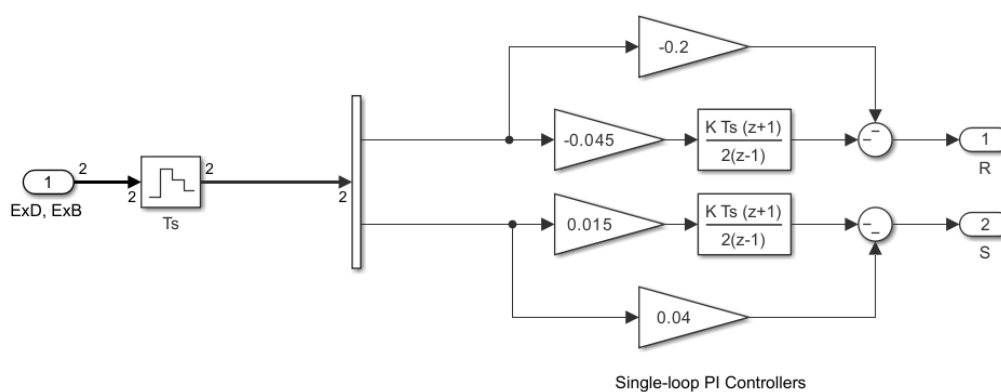


Рис. 4.6 (а). Блок Digital Controller з використанням ПІ-регуляторів

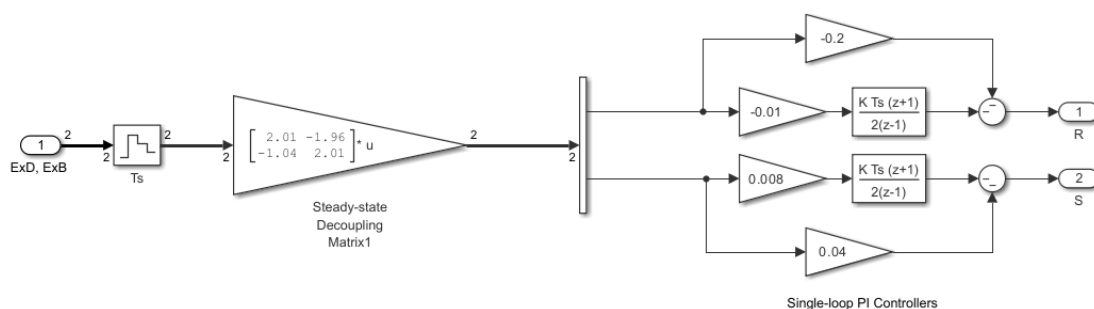


Рис. 4.6 (б). Блок Digital Controller з використанням ПІ-регуляторів та компенсатора

Таблиця 4.2. Порівняння роботи ПІ-регуляторів

Модель	ІКК	Статична похибка	Час перехідного процесу	Затримка	Динамічна похибка
ПІ-регулятори без компенсатора	30,59	0,125	180	3	0,226/0,294
ПІ-регулятори з компенсатором	35,15	0,002	110	3	0,084/0,297

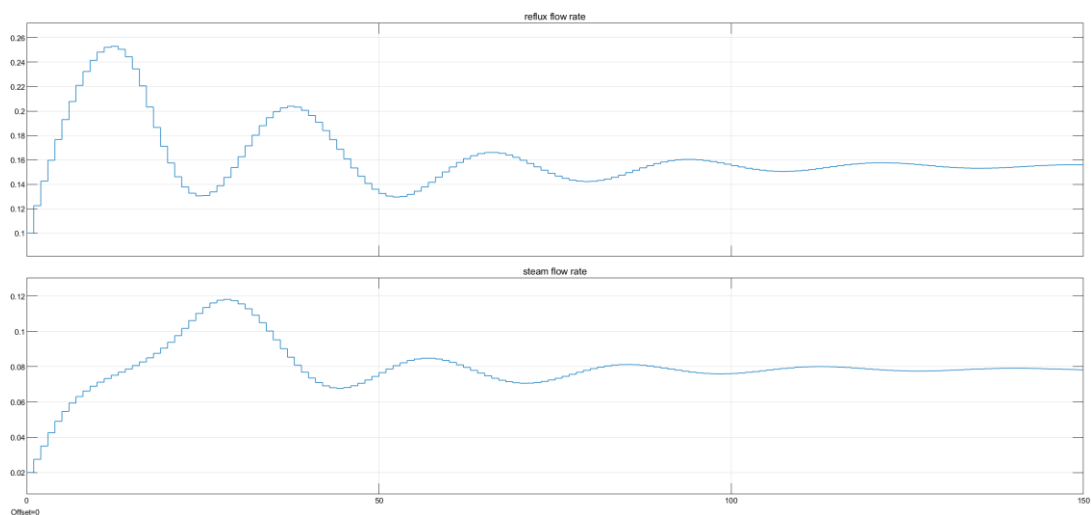


Рис. 4.7 (а). Сигнали управління системи з ПІ-регуляторами

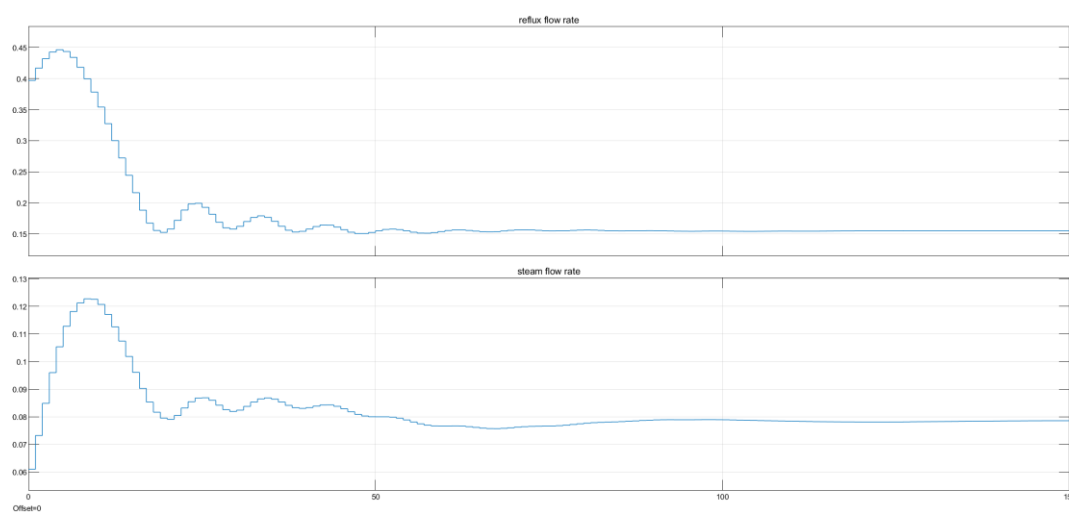


Рис. 4.7 (б). Сигнали управління системи з ПІ-регуляторами з компенсатором

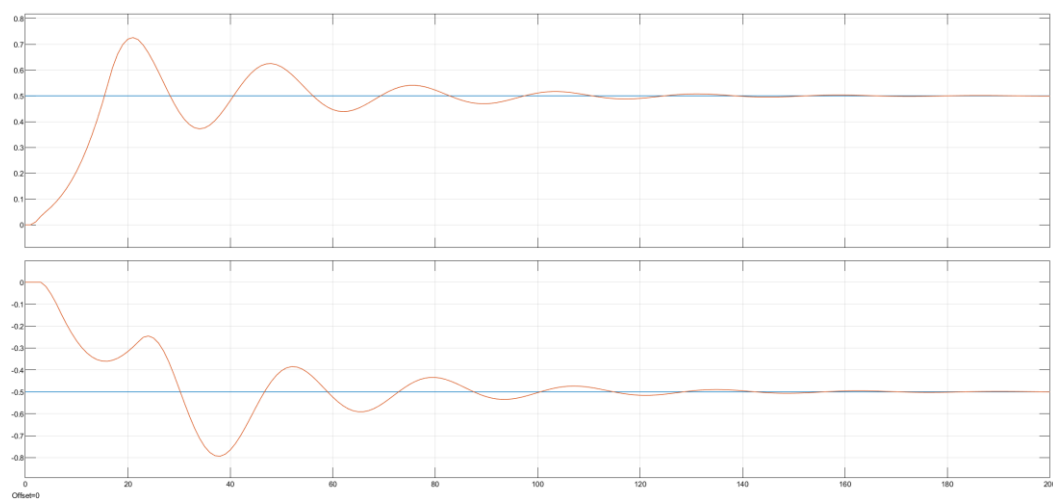


Рис. 4.8 (а). Перехідні процеси системи з ПІ-регуляторами

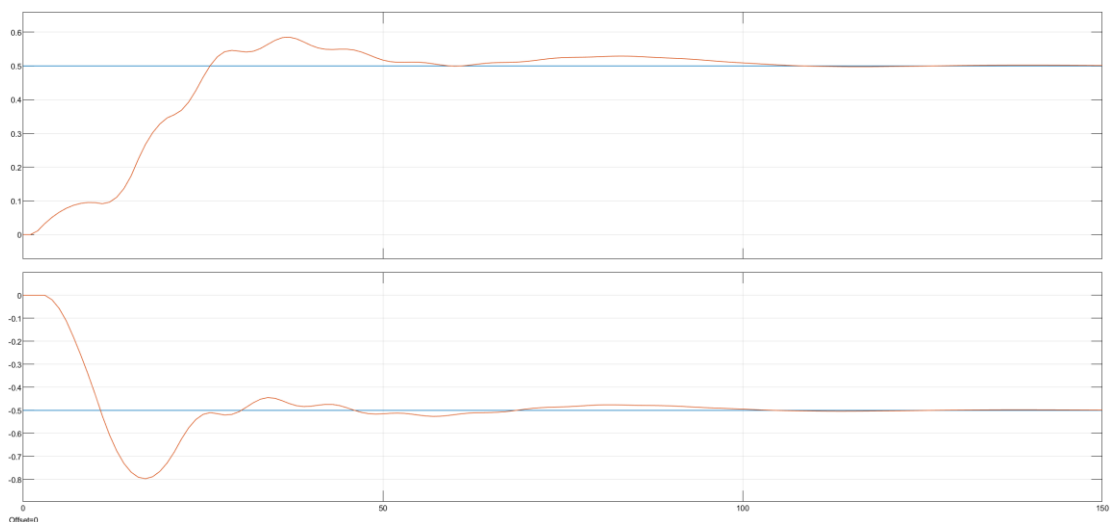


Рис. 4.8 (б). Перехідні процеси системи з ПІ-регуляторами та компенсатором

Відмітимо, що лінеаризована модель дистиляційної колони моделює динаміку процесу дистиляції та описує, як входні змінні - потік рефлюксу (R) і потік пари (S) впливають на концентрації етанолу у верхній (X_D) і нижній (X_B) частинах дистиляційної колони (4.1). Модель представлена у вигляді чотирьох передаточних функцій (transfer functions) першого порядку з запізнюванням, які описують динамічну поведінку системи в частотній області (Рис. 4.9).

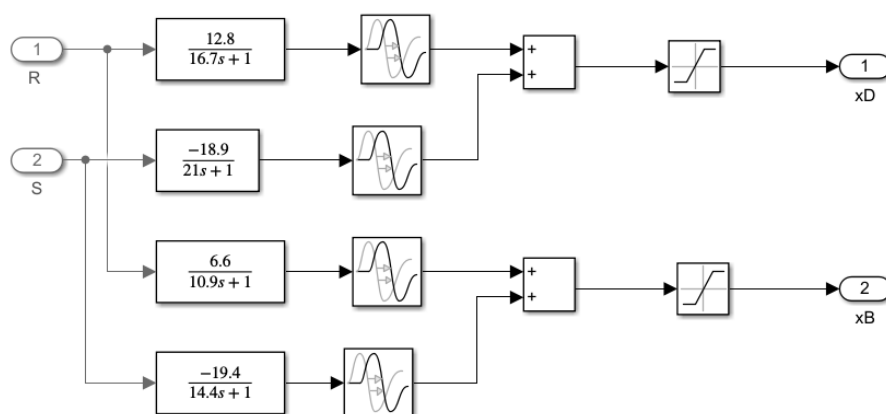


Рис. 4.9. Модель дистиляційної колони

В класичній реалізації за допомогою ПІ-регуляторів отриманий контролер здатний генерувати управляючі сигнали та підтримувати задані значення в колоні, проте дана реалізація робить його вразливим до змін параметрів моделі,

потребуючи ручного підбору налаштувань та втрачаючи якість при змінах в об'єкті чи середовищі.

4.2.5 Моделювання системи з RL-агентом

Формування вектору спостережень. На вхід RL-агента подаються спостереження, що являють собою актуальні дані помилки регулювання та управляючих дій, а також їх попередні значення. Це необхідно для врахування агентом запізнювання, наявних в складі об'єкта управління. Також, разом із цими даними подаються два інтеграли від розузгодження, що дозволяє агенту враховувати накопичення похибки, і зменшувати статичну похибку. Таким чином, в представленому векторі, RL-агент спостерігає за 2-ма змінними помилки регулювання, 14-ма їх архівними значеннями, 14-ма архівними значеннями сигналів управління та 2-ма інтегралами від помилки регулювання (Рис. 4.10).

Формування функції винагороди. Функція винагороди є ключовим елементом RL, оскільки вона визначає, наскільки ефективно агент виконує завдання, і спрямовує його навчання шляхом максимізації загальної винагороди. У контексті управління дистиляційною колоною в спиртовому виробництві функція винагороди розроблена для стабілізації концентрацій Y_D та Y_B (у кубовому залишку) на цільових значеннях, а також для забезпечення ефективного використання ресурсів.

Загальна винагорода формується як сума трьох компонентів:

- 1) Target Reward (винагорода за досягнення цільових значень Y),
- 2) Target Penalty (штраф за відхилення від цільових значень),
- 3) Control Penalty (штраф за надмірні дії U).

Target Reward винагороджує агента за те, що він наближає Y до цільових значень. Control Penalty штрафує агента за надмірні дії (U), щоб уникнути різких змін у L і V , які можуть дестабілізувати колону. Target Penalty штрафує агента за відхилення Y_D і Y_B від цільових значень.

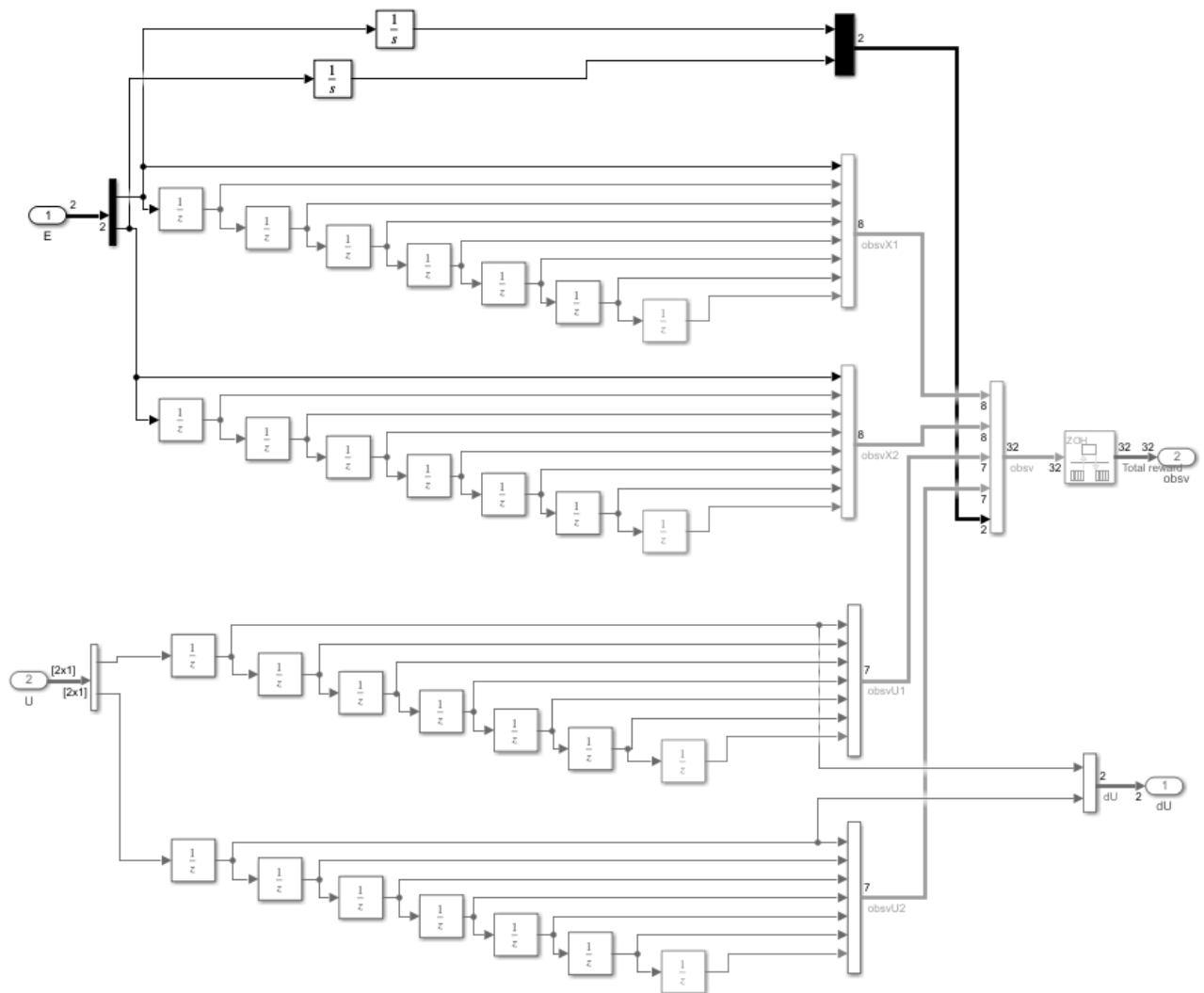


Рис. 4.10. Вектор спостережень для агента

Ці компоненти розраховуються за допомогою відповідних вагових коефіцієнтів, що дозволяє налаштувати пріоритети між стабілізацією концентрацій, енергоефективністю та плавністю дій. Після обчислення вони додаються разом, формуючи Total Reward, який передається агенту для оцінки його дій (рис. 4.11). Додатково, сигнал isDone може бути використаний для завершення епізоду, якщо система виходить за допустимі межі чи досягла цільових значень.

У роботі як функція винагороди використовується обернена величина інтегрально-квадратичного критерію (ІКК) по розузгодженню, оскільки цей підхід продемонстрував найстабільніші результати регулювання (найменші

коливання ІКК і динамічна похибка) під час систематичного порівняння різних варіантів функцій винагороди.

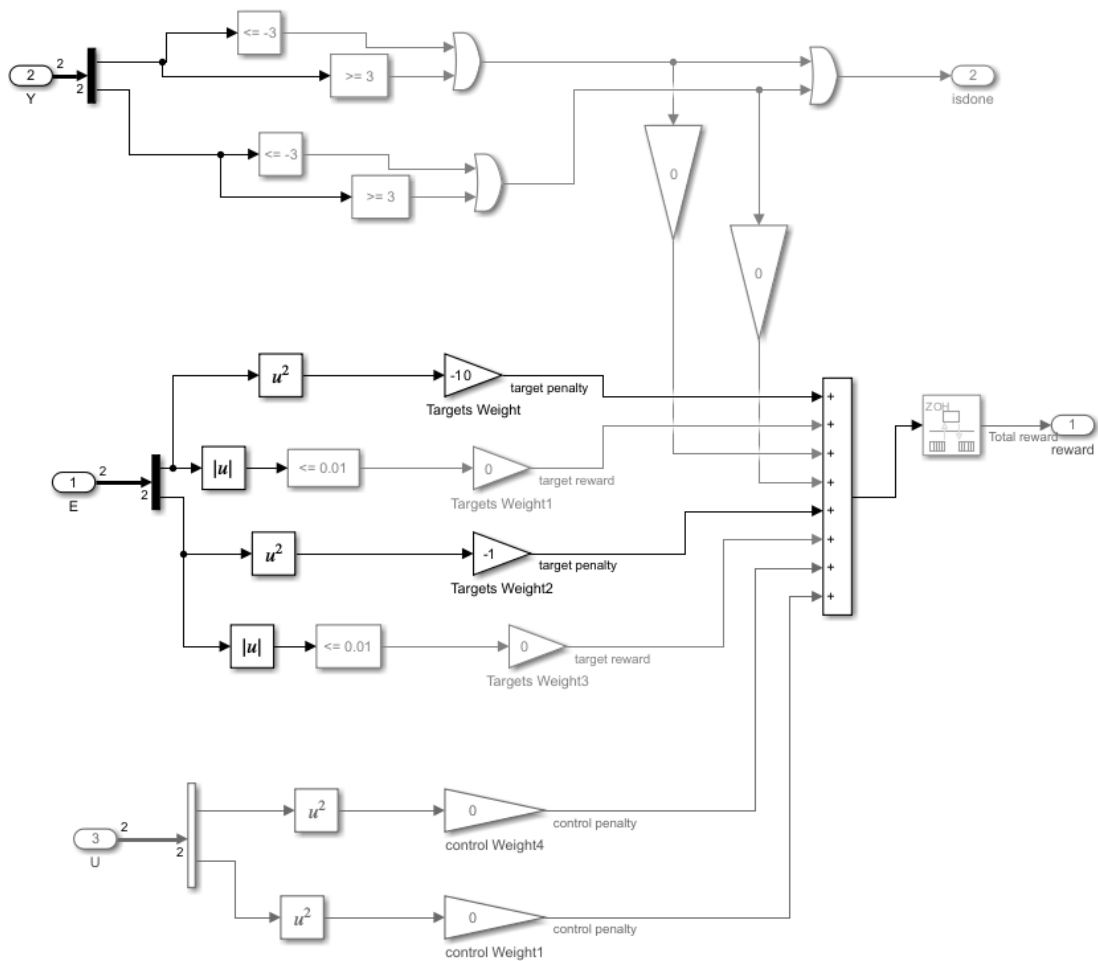


Рис. 4.11. Функція винагороди для агента

Формування мережі актора та критика. Мережа актора та критика в RL-моделях із застосуванням алгоритмів TD3 та SAC є ключовим аспектом їхньої архітектури, оскільки ці алгоритми належать до методів навчання з підкріпленням на основі актор-критик.

TD3 (Twin Delayed DDPG). Актор у TD3 відповідає за визначення оптимальної політики (policy), тобто функції, яка зіставляє стани (states) з діями (actions). Це детермінована політика, що означає, що для кожного стану актор видає конкретну дію. Мережа актора є нейронною мережею, яка приймає на вхід стан s і видає дію a . Щоб уникнути надмірної чутливості до шуму та покращити стабільність, у TD3 до дій актора додається шум (наприклад, гаусівський шум) під час навчання. Це дозволяє досліджувати простір дій (exploration).

Оновлення актора відбувається за допомогою градієнтного підйому за функцією цінності дії (Q-функція), яка оцінюється критиком:

$$\nabla_{\theta} J(\theta) = \mathbb{E}[\nabla_a Q(s, a) \nabla_{\theta} \mu(s)] \quad (4.2)$$

де $\mu(s)$ — політика актора, θ — параметри мережі актора, а $Q(s,a)$ — оцінка критика.

Використовувана мережа приймає на вхід вектор стану розміром 32 ознаки та використовує два повнозв'язні шари по 256 нейронів кожен (fc_1, fc_body). Після кожного з цих шарів застосовується активація ReLU (relu_body після fc_1 і body_output після fc_body). Останній шар (output) видає вектор розміром 2, що відповідає розмірності простору дій. Функція tanh у кінці забезпечує нормалізацію дій у діапазон $[-1, 1]$, щоб обмежити дії в розумних межах (Рис. 4.12).

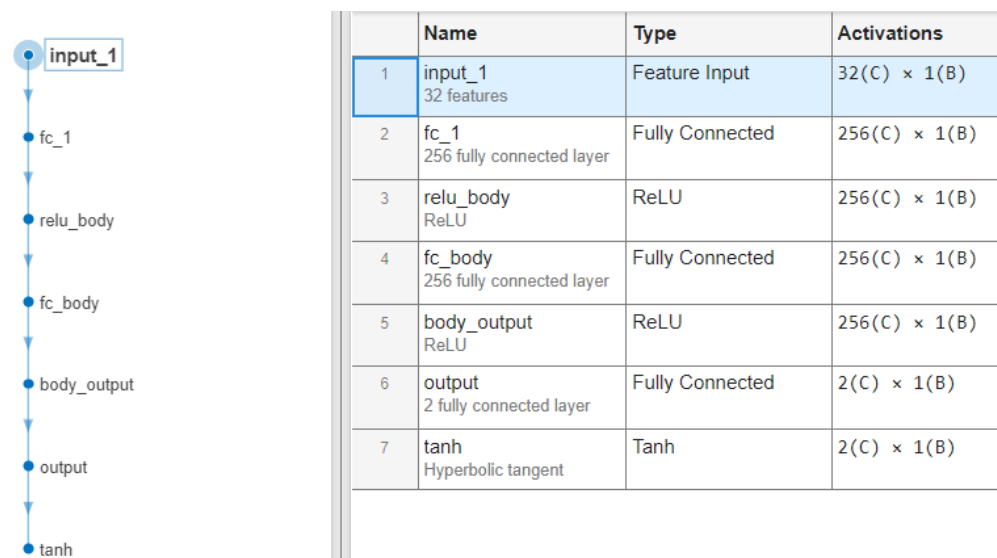


Рис. 4.12. Структура актора TD3

Критик у TD3 оцінює Q-функцію, очікувану кумулятивну винагороду для пари стан-дія (s,a). Унікальною особливістю TD3 є використання двох критиків (twin critics) Q1 і Q2. Це зроблено для зменшення проблеми переоцінки Q-значень, яка була властива попереднику TD3 — DDPG. Обидва критики є нейронними мережами, які приймають на вхід стан s і дію a, а видають числове значення Q. Під час оновлення критика використовується мінімальне значення між Q1 і Q2 для обчислення цільового значення (target value), що допомагає уникнути завищення оцінок:

$$y = r + \gamma \min(Q'_1(s', a'), Q'_2(s', a')) \quad (4.3)$$

де r — винагорода, γ — дисконтний фактор, а a' — дія, отримана з цільової мережі актора з доданим шумом.

Критики оновлюються частіше, ніж актор (затримка оновлення актора — "delayed update"), що підвищує стабільність навчання.

Використовувана мережа критика приймає на вхід два вектори: вектор стану розміром 32 ознаки (input_1) і вектор дій розміром 2 ознаки (input_2). Кожен із цих входів обробляється окремим повнозв'язним шаром по 256 нейронів (fc_1 для стану і fc_2 для дій). Виходи цих шарів об'єднуються через шар конкатенації (concat), утворюючи вектор розміром 512. Далі застосовується активація ReLU (relu_body), після чого йде ще один повнозв'язний шар (fc_body) із 256 нейронами, за яким слідує активація ReLU (body_output). Останній шар (output) видає скалярне значення розміром 1, що відповідає Q-значенню для оцінки пари стан-дія (Рис. 4.13).

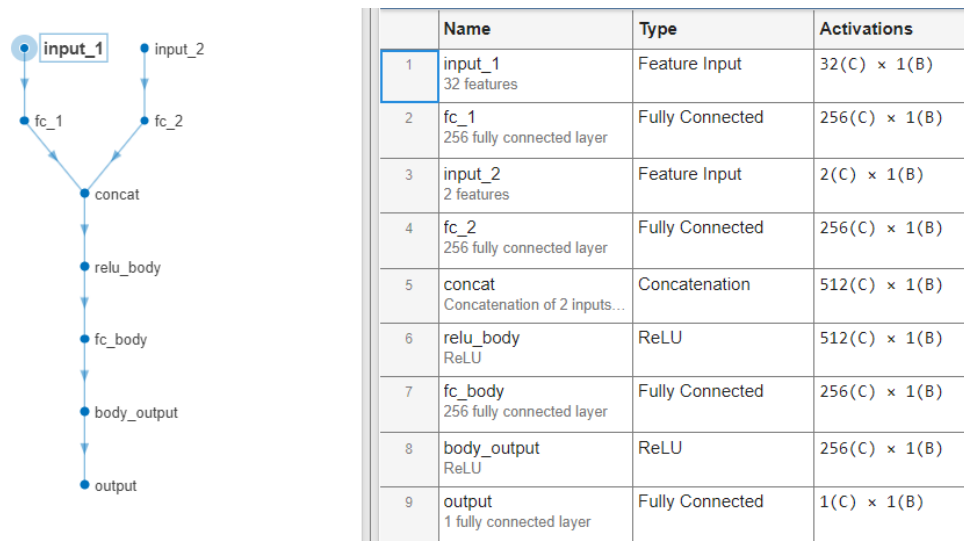


Рис. 4.13. Структура критика TD3

SAC (Soft Actor-Critic). На відміну від TD3, актор у SAC моделює стохастичну політику, тобто замість однієї конкретної дії видає розподіл ймовірностей дій (наприклад, гаусівський розподіл із середнім μ і дисперсією σ). Мережа актора приймає стан s і видає параметри розподілу (наприклад, μ і σ), з якого семплюється дія a . Основна ідея SAC — максимізація не лише очікуваної винагороди, а й ентропії політики. Це стимулює дослідження (exploration)

природним чином, без необхідності додавати шум вручну, як у TD3. Цільова функція актора:

$$J(\theta) = \mathbb{E}[Q(s, a) + \alpha H(\pi(\cdot|s))] \quad (4.4)$$

де $H(\pi)$ — ентропія політики, а α — коефіцієнт, що регулює баланс між винагородою та ентропією (може адаптивно налаштовуватися). Оновлення актора відбувається через градієнтний підйом із використанням оцінки Q-функції від критика.

Використовувана мережа актора SAC приймає на вхід вектор стану розміром 32 ознаки (input_1) та використовує два повнозв'язні шари по 256 нейронів кожен (fc_1, fc_body). Після кожного з цих шарів застосовується активація ReLU (relu_body після fc_1 і body_output після fc_body). Далі мережа розгалужується на два виходи: перший вихідний шар (fc_mean) видає вектор розміром 2, що відповідає середньому значенню (mean) нормального розподілу дій, а другий вихідний шар (fc_std) видає вектор розміром 2, який після застосування Softplus-активації (std) відповідає стандартному відхиленню (std) цього розподілу. Така структура є стандартом для актора в SAC, дозволяючи моделювати стохастичну політику через нормальний розподіл дій (Рис. 4.14).

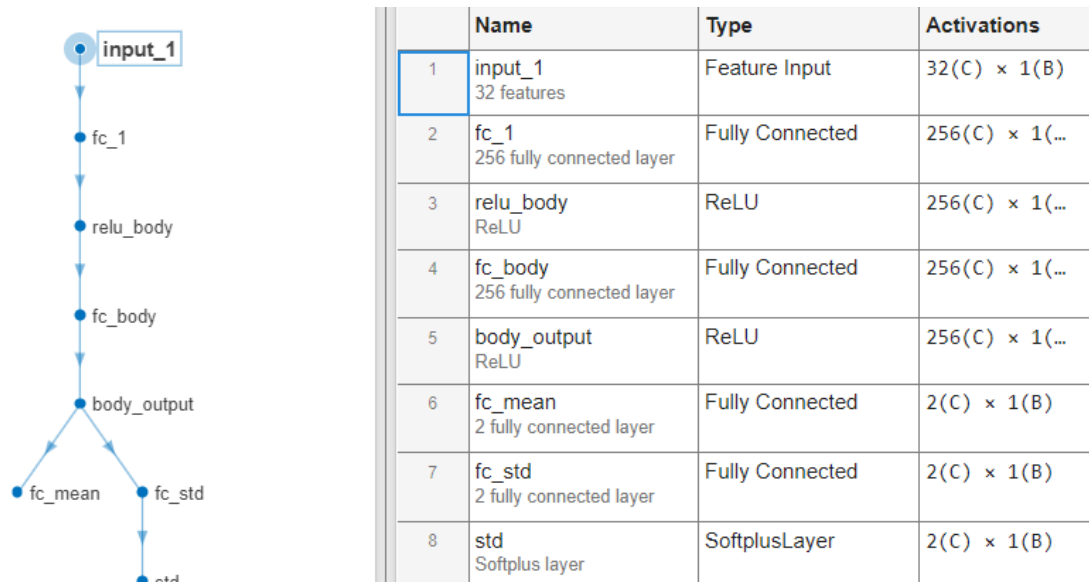


Рис. 4.14. Структура актора SAC

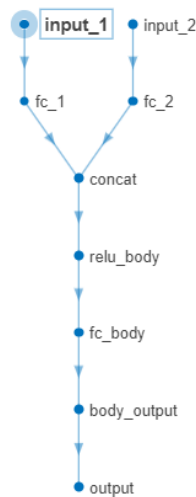
У SAC також використовується два критики (Q1 і Q2), як у TD3, для зменшення переоцінки Q-значень. Критики є нейронними мережами, що

приймають стан s і дію a та видають оцінку Q . Однак, на відміну від TD3, цільове значення для критика враховує ентропію політики через "м'яке" оновлення Беллмана:

$$y = r + \gamma \mathbb{E}[\min(Q'_1(s', a'), Q'_2(s', a')) - \alpha \log \pi(a' | s')] \quad (4.5)$$

де a' семплюється з поточної політики π . Критики оновлюються шляхом мінімізації похибки між передбаченими та цільовими значеннями Q .

Використовувана мережа критика SAC приймає на вхід два вектори: вектор стану розміром 32 ознаки (input_1) і вектор дій розміром 2 ознаки (input_2). Кожен із цих входів обробляється окремим повнозв'язним шаром по 256 нейронів (fc_1 для стану і fc_2 для дій). Виходи цих шарів об'єднуються через шар конкатенації (concat), утворюючи вектор розміром 512. Далі застосовується активація ReLU (relu_body), після чого йде ще один повнозв'язний шар (fc_body) із 256 нейронами, за яким слідує активація ReLU (body_output). Останній шар (output) видає скалярне значення розміром 1, що відповідає Q -значенню для оцінки пари стан-дія (Рис. 4.15).



	Name	Type	Activations
1	input_1 32 features	Feature Input	32(C) × 1(B)
2	fc_1 256 fully connected layer	Fully Connected	256(C) × 1(B)
3	input_2 2 features	Feature Input	2(C) × 1(B)
4	fc_2 256 fully connected layer	Fully Connected	256(C) × 1(B)
5	concat Concatenation of 2 inputs ...	Concatenation	512(C) × 1(B)
6	relu_body ReLU	ReLU	512(C) × 1(B)
7	fc_body 256 fully connected layer	Fully Connected	256(C) × 1(B)
8	body_output ReLU	ReLU	256(C) × 1(B)
9	output 1 fully connected layer	Fully Connected	1(C) × 1(B)

Рис. 4.15. Структура критика SAC

Навчання агентів. Навчання виділених агентів було виконано за допомогою Reinforcement Learning Designer бібліотеки Reinforcement Learning Toolbox середовища Matlab. М-файл environment (зовнішнього середовища) для RL-агента наведена в Додатку Є. Для оцінки та порівняння результатів обрано три RL-агента: TD3, та два SAC з різними значеннями Entropy weight. У

відповідності з особливостями архітектури агентів та об'єкта керування було налаштовано гіперпараметри RL-моделей. Основні гіперпараметри налаштувань наведені в таблиці 4.3.

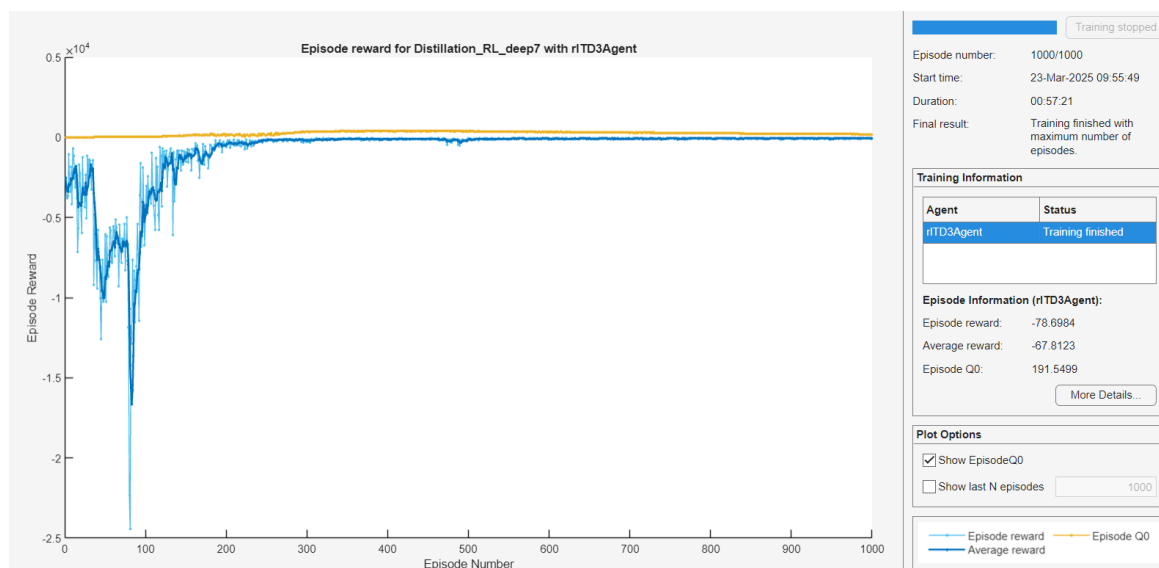


Рис. 4.16. Навчання TD3-агента

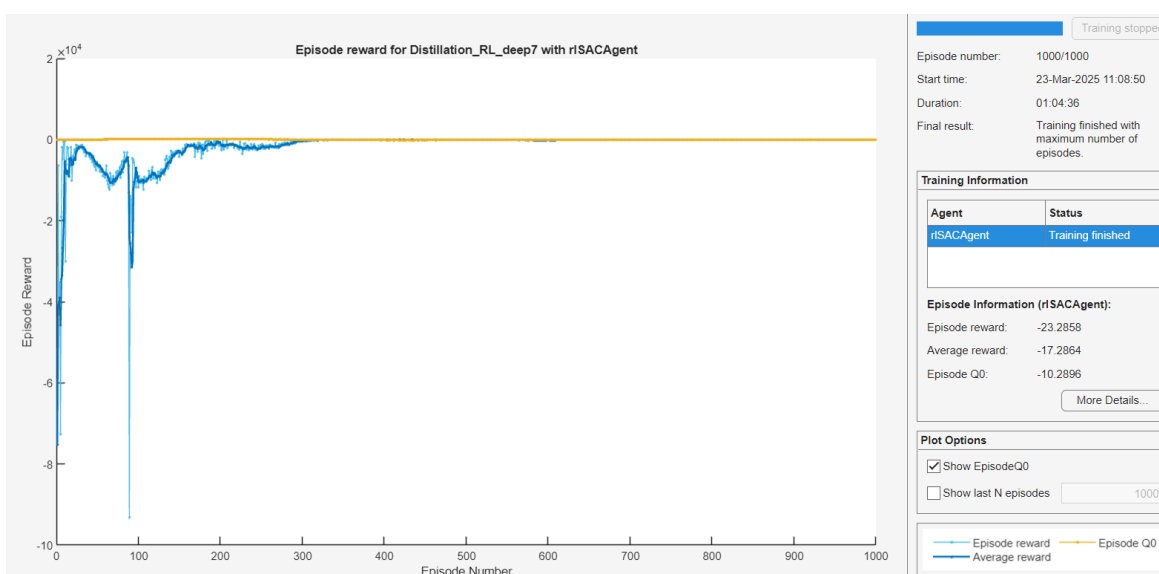


Рис. 4.17. Навчання SAC-агента з Entropy weight 1

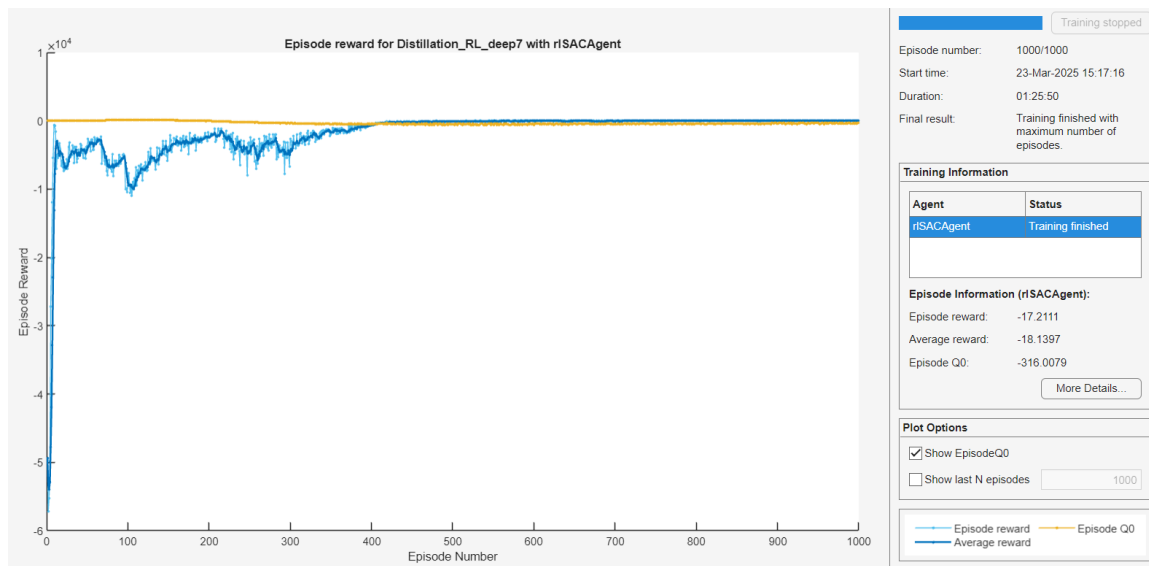


Рис. 4.18. Навчання SAC2-агента з Entropy weight 0,3

Таблиця 4.3. Гіперпараметри RL-агентів

Гіперпараметри	TD3	SAC1	SAC2
Agent Options			
Batch size	256	256	256
Experience buffer length	1e+04 (10,000)	1e+04 (10,000)	1e+04 (10,000)
Policy update frequency	2	1	1
Target smooth factor	0.005	0.001	0.001
Target update frequency	2	1	1
Actor Optimizer Options			
Learn rate	0.0001	0.0001	0.0001
Optimizer	Adam	Adam	Adam
L2 regularization	1e-05	1e-05	1e-05
Critic Optimizer Options			
Learn rate	0.0001	0.0001	0.0001
Optimizer	Adam	Adam	Adam
L2 regularization	1e-05	1e-05	1e-05
Exploration Model			
Standard deviation	0.316227766016	-	-
Entropy Options (SAC-specific)			
Entropy weight	-	1	0.3

Моделювання систем. Побудована система складається з моделі дистиляційної колони, з якою взаємодіє RL-агент. RL-агент отримує спостереження (observation) із системи, обчислює дії (action), які впливають на

модель, і отримує винагороду (reward) на основі ефективності управління. Система включає кілька підсистем, таких як обчислення винагороди, формування вектору спостережень, обмеження швидкості управляючого сигналу за допомогою Rate limiter і модель дистиляційної колони (Рис. 4.19).

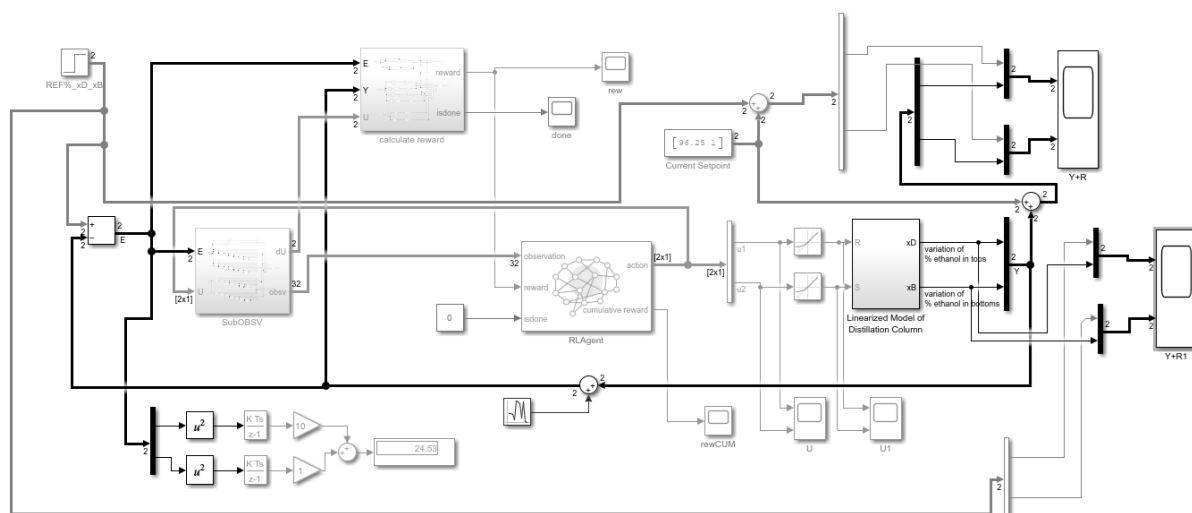


Рис. 4.19. Модель системи керування з RL-агентом

RL-агент взаємодіє з середовищем, яке включає дистиляційну колону, генеруючи сигнали управління для регулювання заданих значень концентрацій (Рис. 4.20 - 4.22).

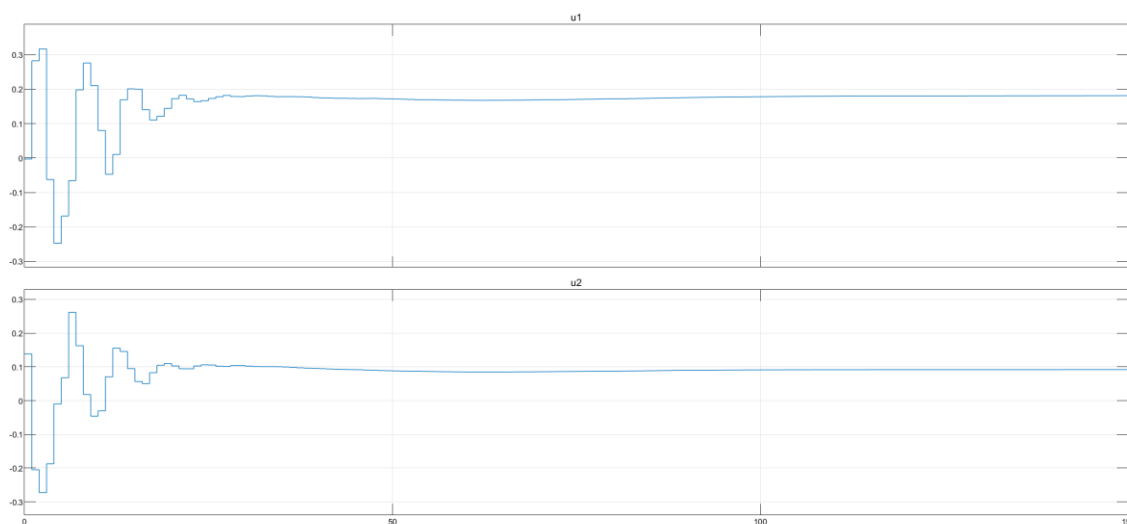


Рис. 4.20. Сигнали управління у системі з RL-агентом (TD3)

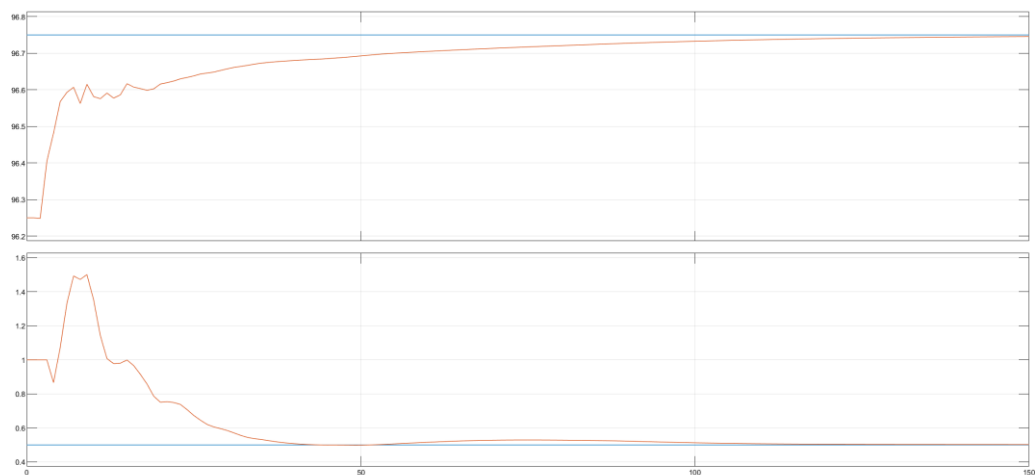


Рис. 4.21. Перехідні процеси концентрацій у верхній та нижній частинах колони (TD3)

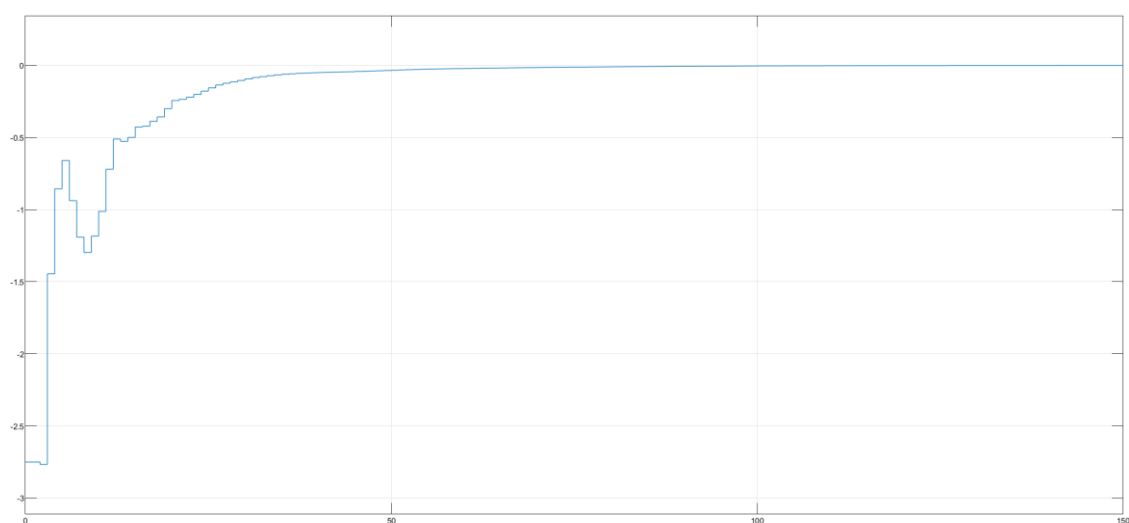


Рис. 4.22. Графік винагороди агента (TD3)

Порівняння результатів. Для оцінки ефективності роботи RL-моделей було проведено порівняння якості перехідних процесів для номінального об'єкта, системи з шумом, та об'єкта при зміні коефіцієнтів дистиляційної колони для оцінки адаптації системи.

Для порівняння використовувалися наступні критерії:

1) Інтегрально-квадратичний критерій по розузгодженню (ІКК) показує, наскільки добре модель управляє дистиляційною колоною, вимірюючи сумарну квадратичну помилку між референтними і фактичними значеннями концентрацій (X_D і X_B) за певний період часу. Менший ІКК вказує на кращу якість регулювання, тобто менші і коротші відхилення від цільових значень.

2) Статична похибка показує, наскільки точно агент підтримує концентрації X_D і X_B на референтних рівнях у сталому режимі.

3) Час перехідного процесу визначає, як швидко агент може привести концентрації до цільових значень після зміни умов.

4) Затримка вказує на час, необхідний для початку реакції системи.

5) Динамічна похибка показує, наскільки перевищують концентрації X_D і X_B цільові значення під час перехідного процесу.

6) Амплітуда усталених коливань відображає величину шумових коливань в усталеному режимі.

Було проведено аналіз роботи регуляторів, навчених та налаштованих відповідно до поточної конфігурації дистиляційної колони для стабілізації концентрацій цільових змінних, для оцінки їх ефективності в стабільних та незмінних умовах функціонування системи.

Серед розглянутих регуляторів SAC1 є найкращим за загальною ефективністю (ІКК = 13,44) та точністю в усталеному режимі (статична похибка = 0,0009), SAC2 – за збалансованістю показників (ІКК = 20,33, статична похибка = 0,0002), ПІ-регулятори – за стабільністю та швидкістю реакції (динамічна похибка = 0,084/0,297, час перехідного процесу = 110). TD3 продемонстрував найгіршу продуктивність через великі відхилення (ІКК = 24,53, статична похибка = 0,004) і повільну адаптацію (час перехідного процесу = 130) (Табл. 4.4).

Таблиця 4.4. Порівняння систем регулювання для номінального об'єкта

Модель	ІКК	Статична похибка	Час перехідного процесу	Затримка	Динамічна похибка
ПІ-регулятори (з компенсатором)	35,15	0,002	110	3	0,084/0,297
TD3	24,53	0,004	130	3	-/0,501
SAC1	13,44	0,0009	70	3	0,2186/-
SAC2	20,33	0,0002	90	3	0,143/0,5

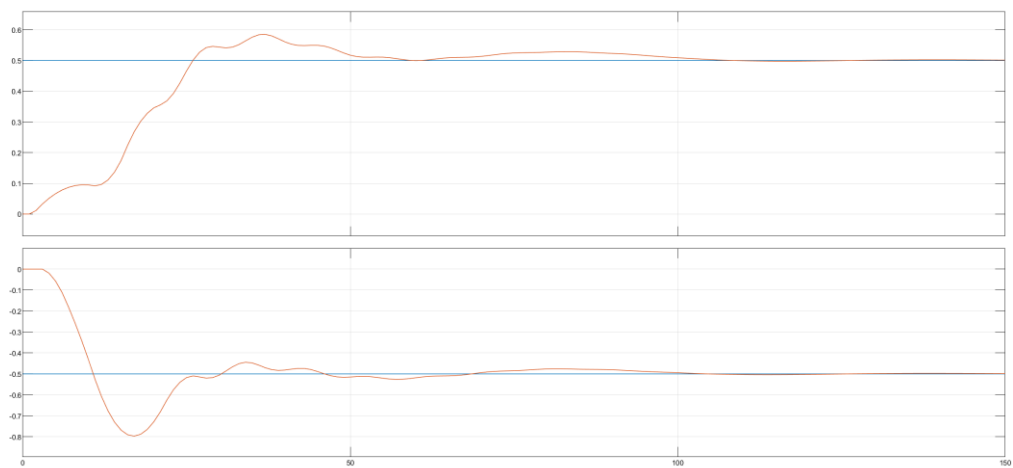


Рис. 4.23 (а). Перехідні процеси з ПІ-регуляторами для номінального об'єкта

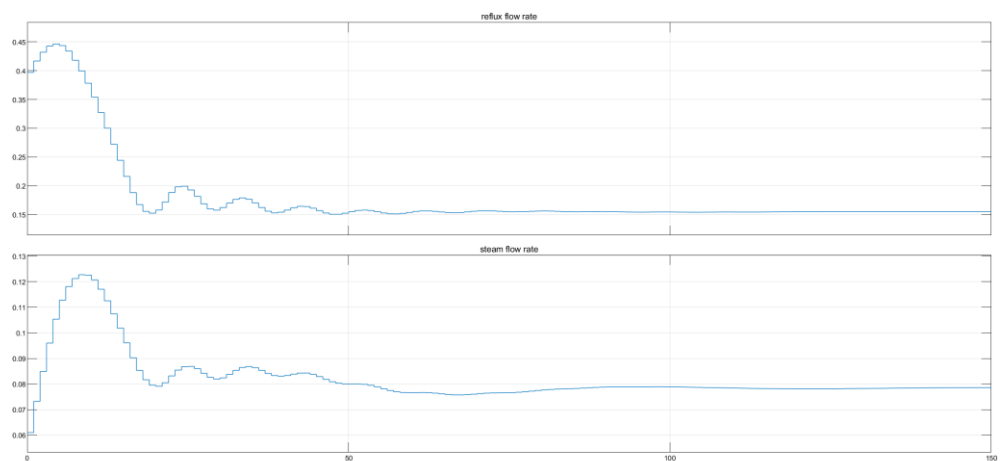


Рис. 4.23 (б). Сигнали управління у системі з ПІ-регуляторами для номінального об'єкта

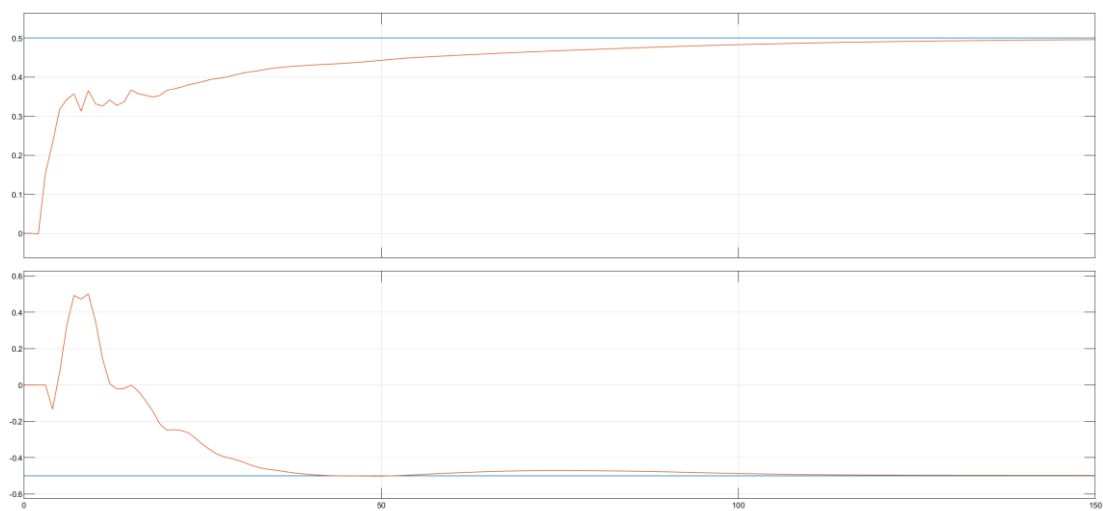


Рис. 4.24 (а). Перехідні процеси з TD3-агентом для номінального об'єкта

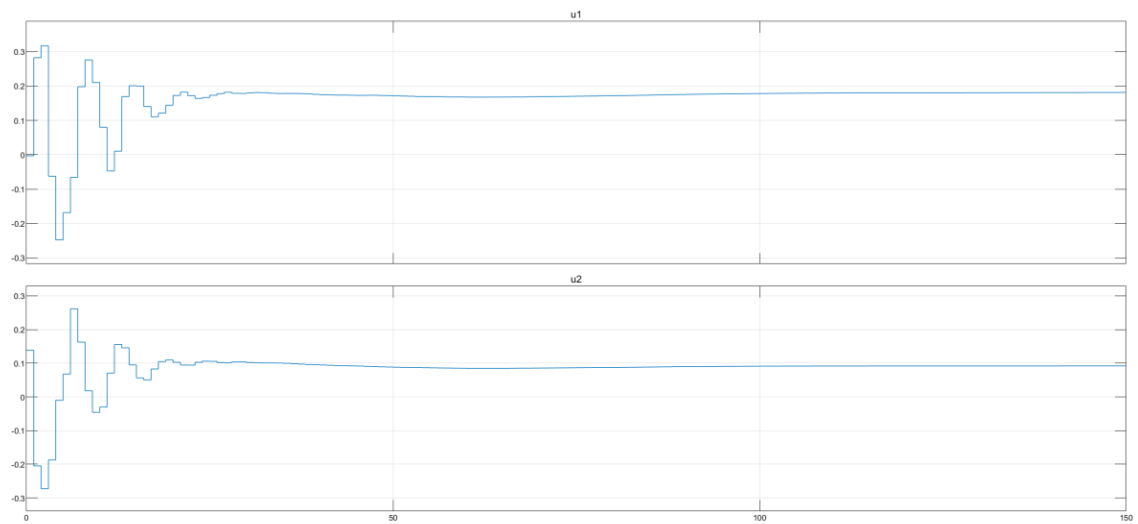


Рис. 4.24 (б). Сигнали управління у системі з TD3-агентом для номінального об'єкта

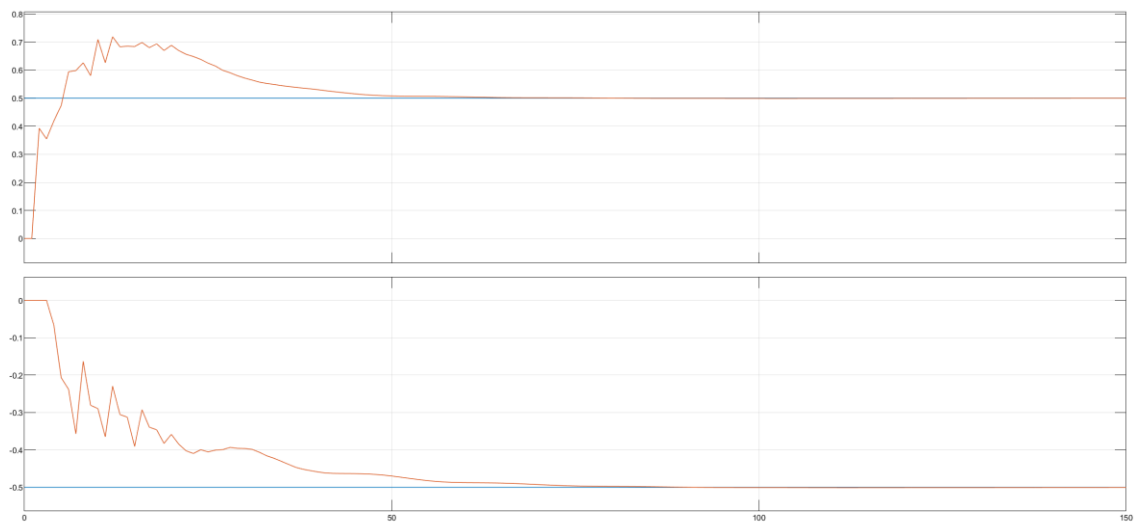


Рис. 4.25 (а). Перехідні процеси з SAC1-агентом для номінального об'єкта

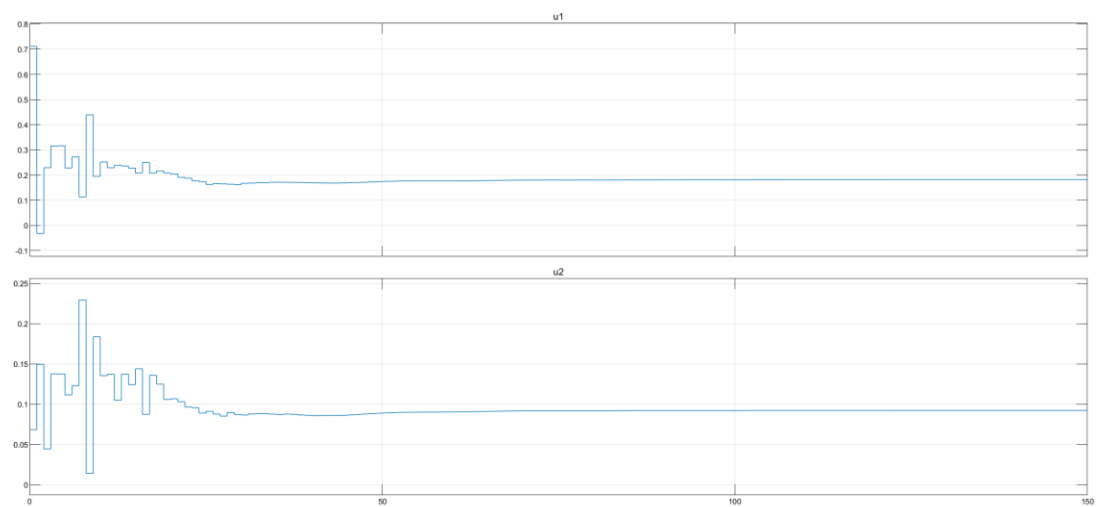


Рис. 4.25 (б). Сигнали управління у системі з SAC1-агентом для номінального об'єкта

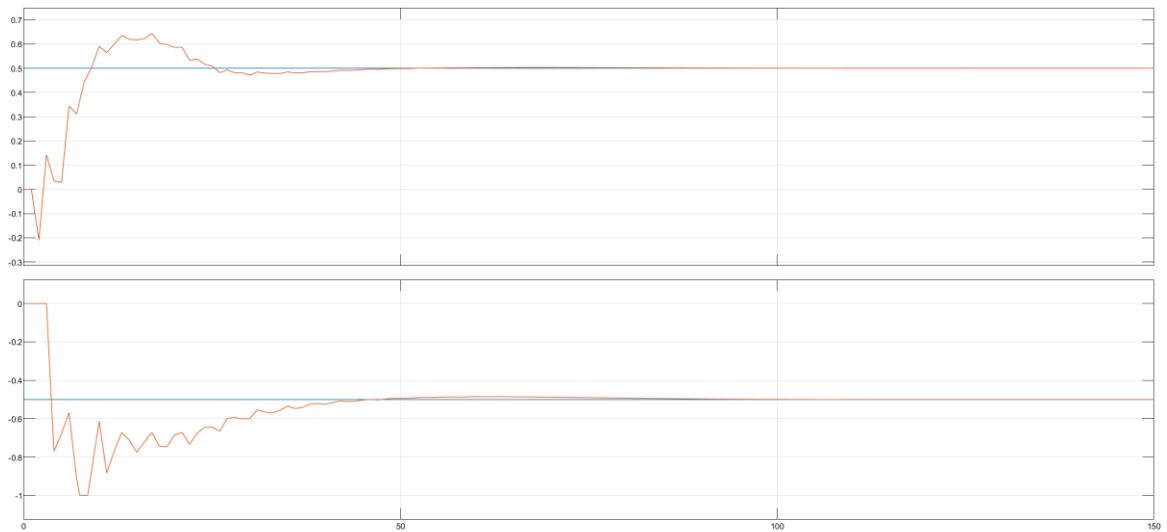


Рис. 4.26 (а). Перехідні процеси з SAC2-агентом для номінального об'єкта

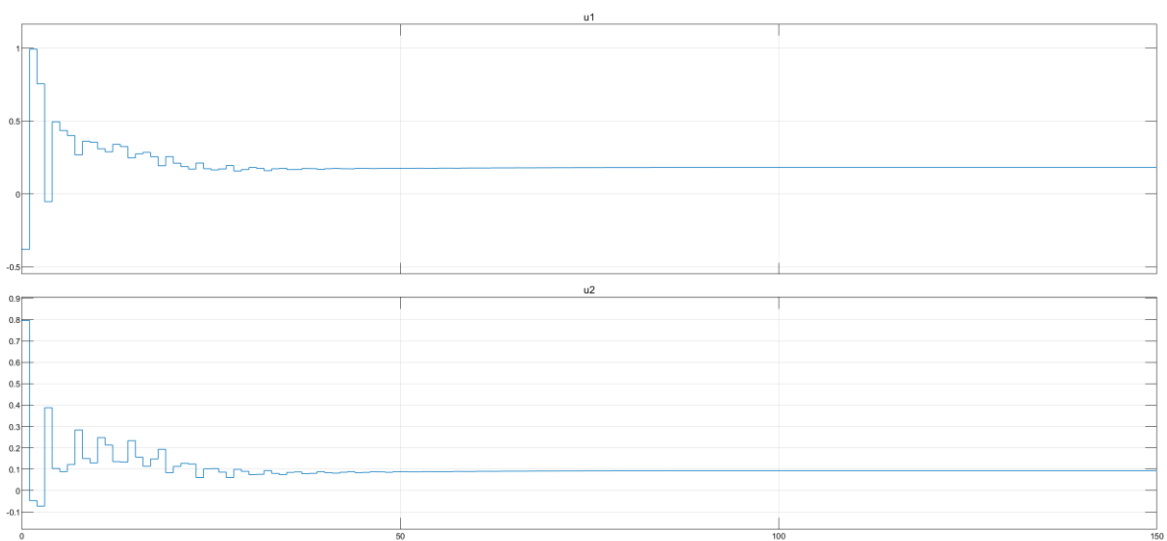


Рис. 4.26 (б). Сигнали управління у системі з SAC2-агентом для номінального об'єкта

Для подальшого дослідження було розглянуто поведінку регуляторів при впливі на систему випадкових шумів. Згідно результатів, при впливі випадкових шумів SAC1 є найкращим агентом за загальною ефективністю (ІКК = 14,42), SAC2 – за швидкістю реакції (час перехідного процесу = 50) та збалансованістю показників (ІКК = 22,45, динамічна похибка = 0,152/0,5). РІ-регулятор вирізняється найменшою амплітудою коливань (0,005) та стабільністю (динамічна похибка = 0,085/0,3). TD3 продемонстрував найгіршу продуктивність через великі відхилення (ІКК = 25,75, амплітуда коливань = 0,0342) і повільну адаптацію (час перехідного процесу = 115) (Табл. 4.5).

Таблиця 4.5. Порівняння систем регулювання для номінального об'єкта для системи з шумом

Модель	ІКК	Час перехідного процесу	Затримка	Динамічна похибка	Амплітуда коливань
PI-регулятори	35,63	110	3	0,085/0,3	0,005
TD3	25,75	115	3	-/0,504	0,0342
SAC1	14,42	70	3	0,21/-	0,054
SAC2	22,45	50	3	0,152/0,5	0,08

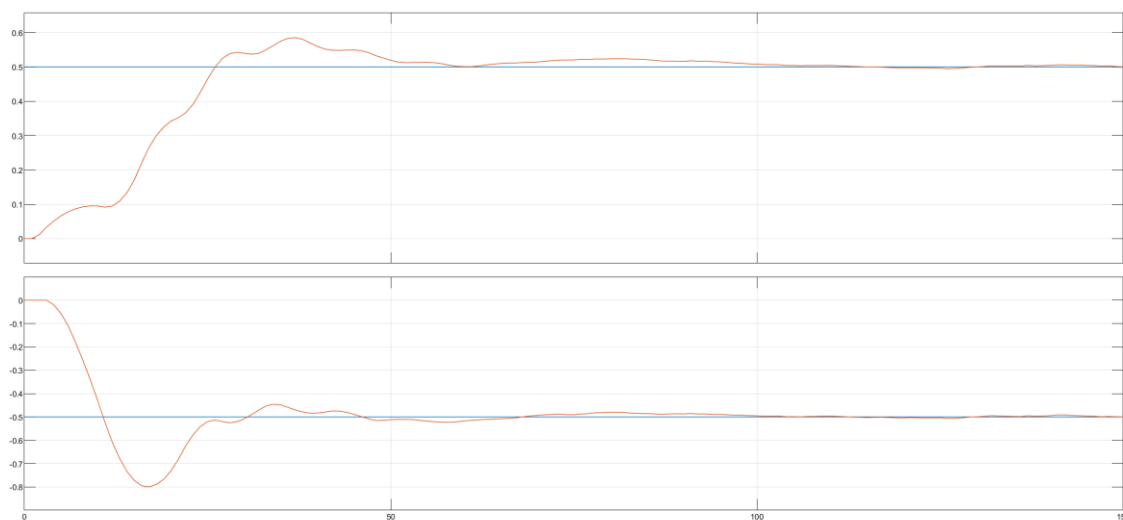


Рис. 4.27 (а). Перехідні процеси з ПІ-регуляторами та шумом

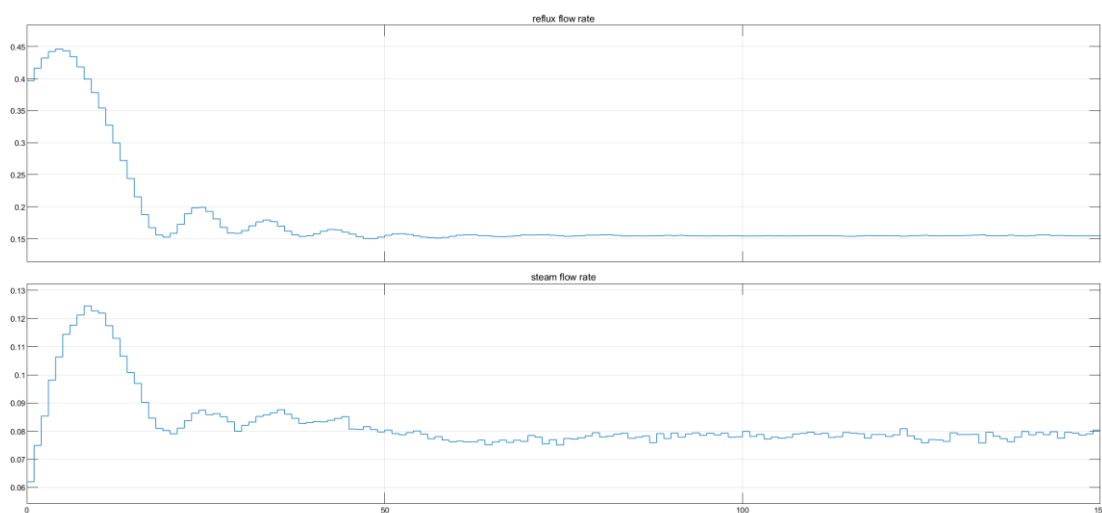


Рис. 4.27 (б). Сигнали управління у системі з ПІ-регуляторами та шумом

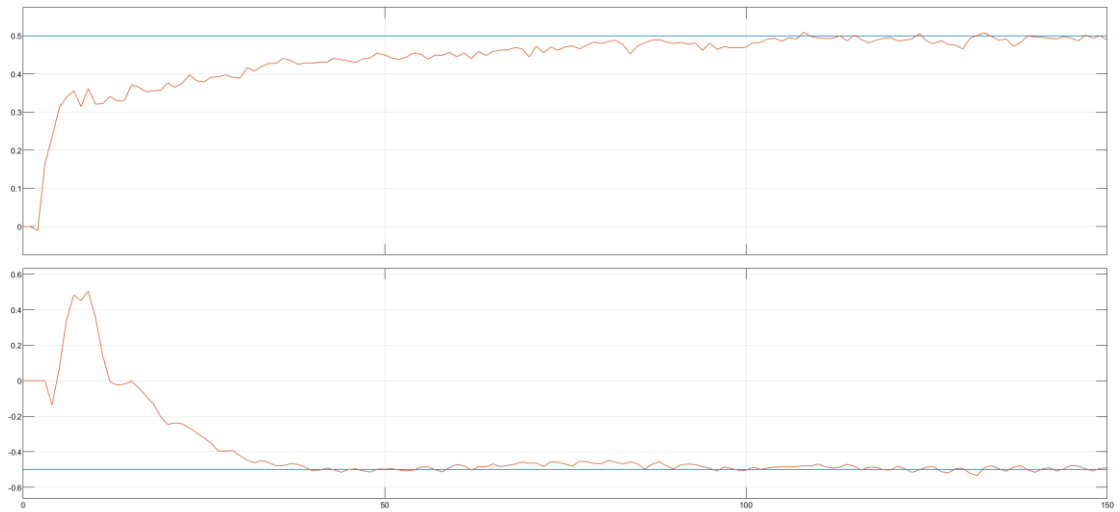


Рис. 4.28 (а). Перехідні процеси з TD3-агентом та шумом

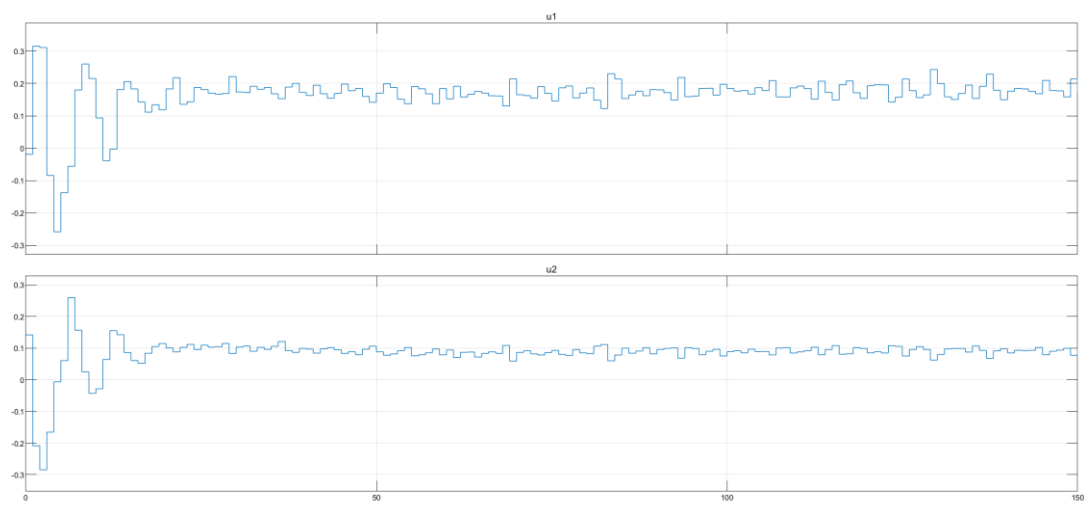


Рис. 4.28 (б). Сигнали управління у системі з TD3-агентом та шумом

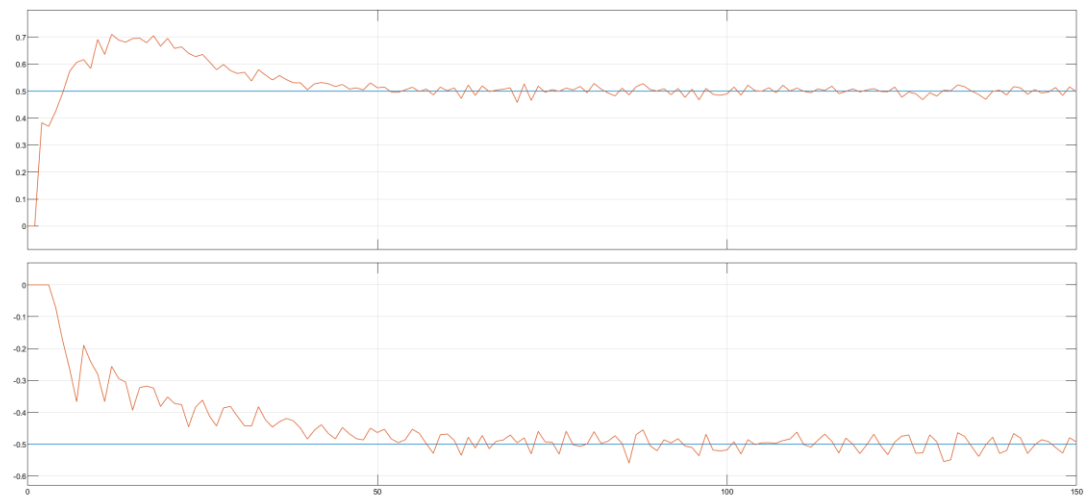


Рис. 4.29 (а). Перехідні процеси з SAC1-агентом та шумом

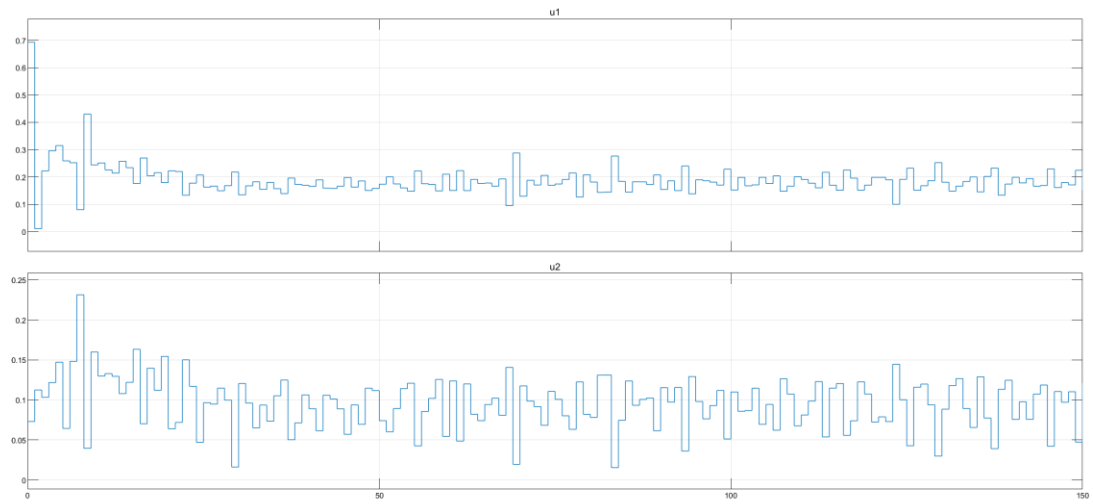


Рис. 4.29 (б). Сигнали управління у системі з SAC1-агентом та шумом

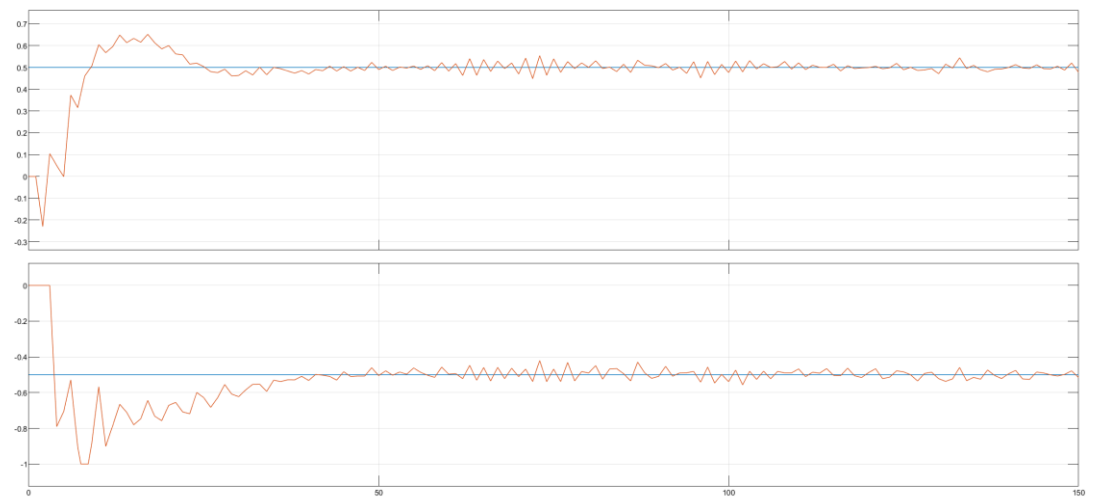


Рис. 4.30 (а). Перехідні процеси з SAC2-агентом та шумом

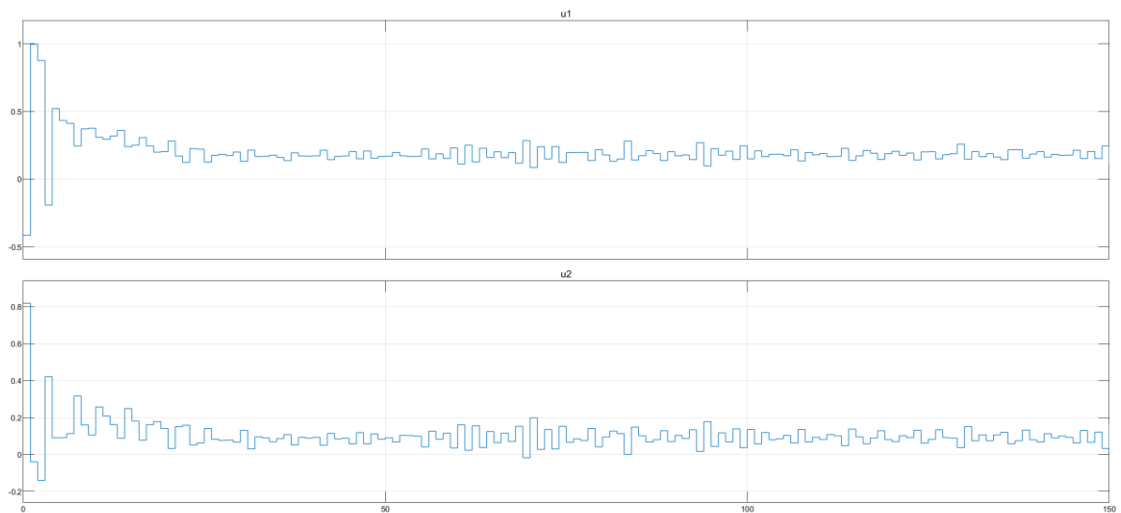


Рис. 4.30 (б). Сигнали управління у системі з SAC2-агентом та шумом

Для перевірки працездатності системи керування у випадку можливих змін в об'єкті внаслідок змін в потоках, температурі, тиску, складі вхідного потоку та зносу обладнання, було змінено коефіцієнти підсилення моделі дистиляційної колони (4.1) та постійні часу на 15% (Рис. 4.31).

Експеримент 1. Збільшення коефіцієнтів передачі та постійних часу на 15%.

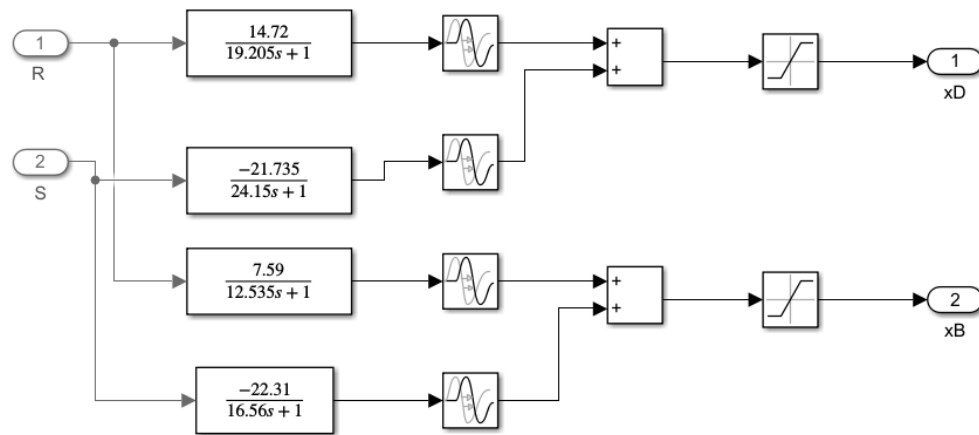


Рис. 4.31. Модель дистиляційної колони зі зміненими параметрами
(Експеримент 1)

В умовах відхилення параметрів моделі дистиляційної колони на 15% (коефіцієнти підсилення та сталі часу) SAC1 є найкращим за загальною ефективністю (ІКК = 13,31), SAC2 – за точністю в усталеному режимі (статична похибка = 0,0000003) та швидкістю реакції (час перехідного процесу = 50). ПІ-регулятор вирізняється стабільністю (динамічна похибка = 0,085/0,32). TD3 продемонстрував найгіршу продуктивність через значні відхилення (ІКК = 21,03, динамічна похибка = 0/0,637) та повільну адаптацію (час перехідного процесу = 120). Загалом агенти SAC1 і SAC2 мають перевагу над ПІ-регулятором у загальній ефективності (нижчий ІКК), а SAC2 додатково вирізняється високою точністю (найменша статична похибка) та швидкістю реакції (найкоротший час перехідного процесу). (Табл. 4.6).

Таблиця 4.6. Порівняння систем регулювання для об'єкта зі зміненими параметрами (Експеримент 1)

Модель	ІКК	Статична похибка	Час перехідного процесу	Затримка	Динамічна похибка
PI-регулятори	34,46	0,003	120	3	0,085/0,32
TD3	21,03	0,0035	120	3	0/0.637
SAC1	13,31	0,003	100	3	0,264/-
SAC2	21,55	0,0000003	50	3	0,212/0,5

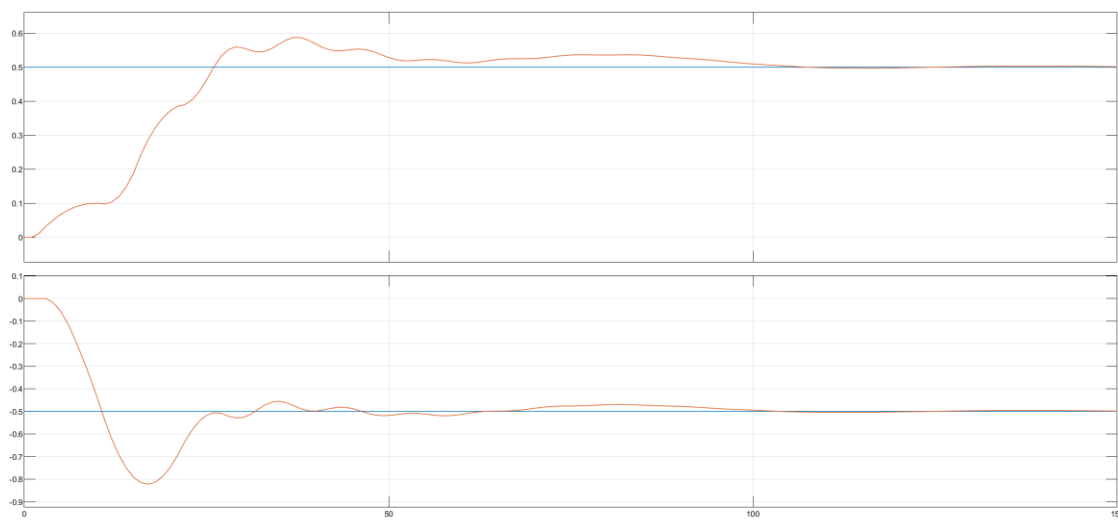


Рис. 4.32 (а). Перехідні процеси з ПІ-регулятором для об'єкта зі зміненими параметрами (Експеримент 1)

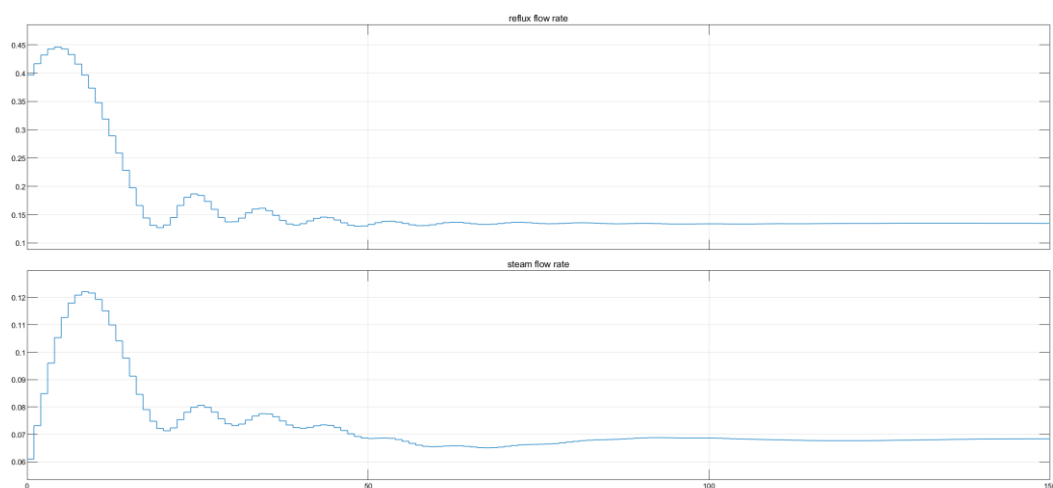


Рис. 4.32 (б). Сигнали управління у системі з ПІ-регулятором для об'єкта зі зміненими параметрами (Експеримент 1)

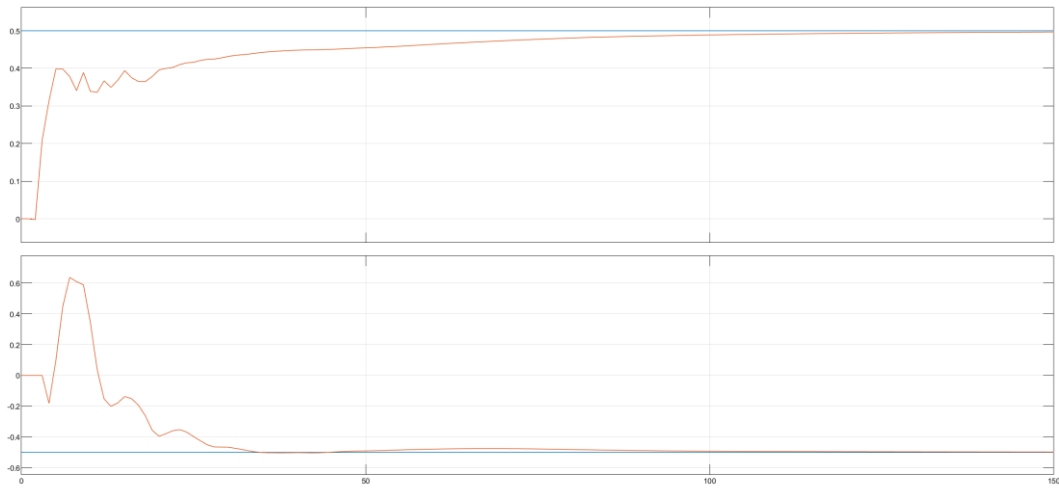


Рис. 4.33 (а). Перехідні процеси з TD3-агентом для об'єкта зі зміненими параметрами (Експеримент 1)

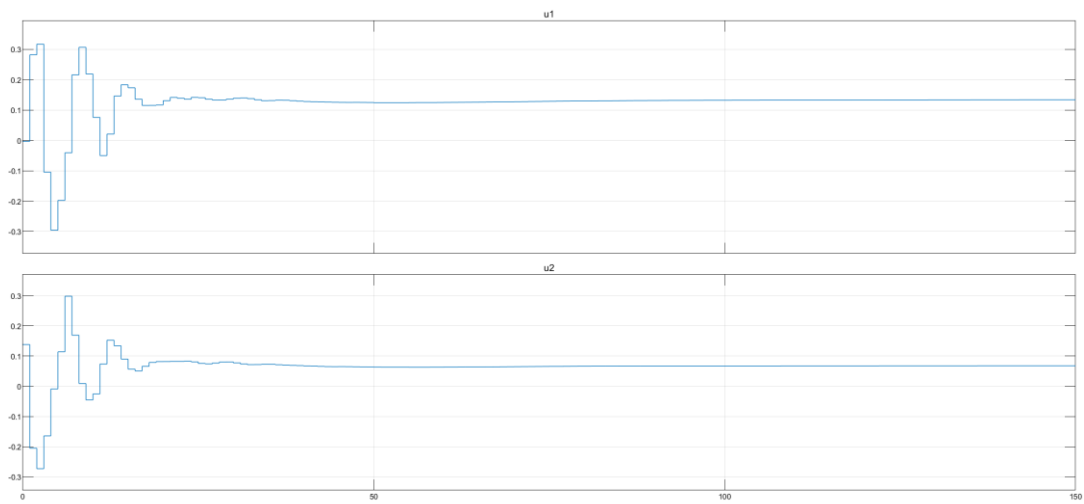


Рис. 4.33 (б). Сигнали управління у системі з TD3-агентом для об'єкта зі зміненими параметрами (Експеримент 1)

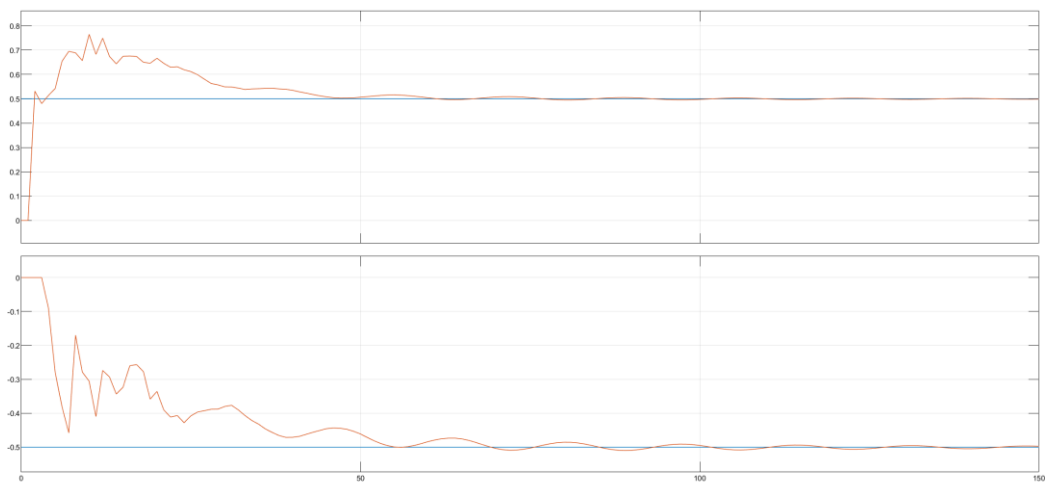


Рис. 4.34 (а). Перехідні процеси з SAC1-агентом для об'єкта зі зміненими параметрами (Експеримент 1)

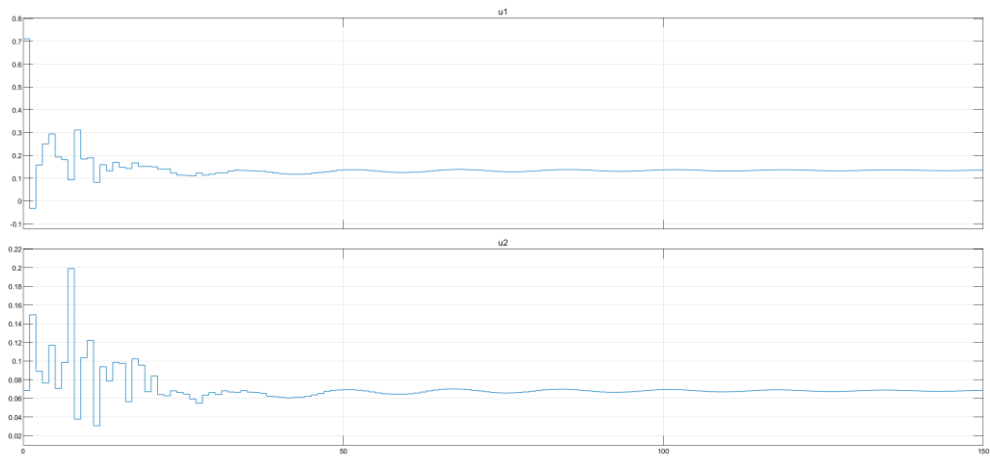


Рис. 4.34 (б). Сигнали управління у системі з SAC1-агентом для об'єкта зі зміненими параметрами (Експеримент 1)

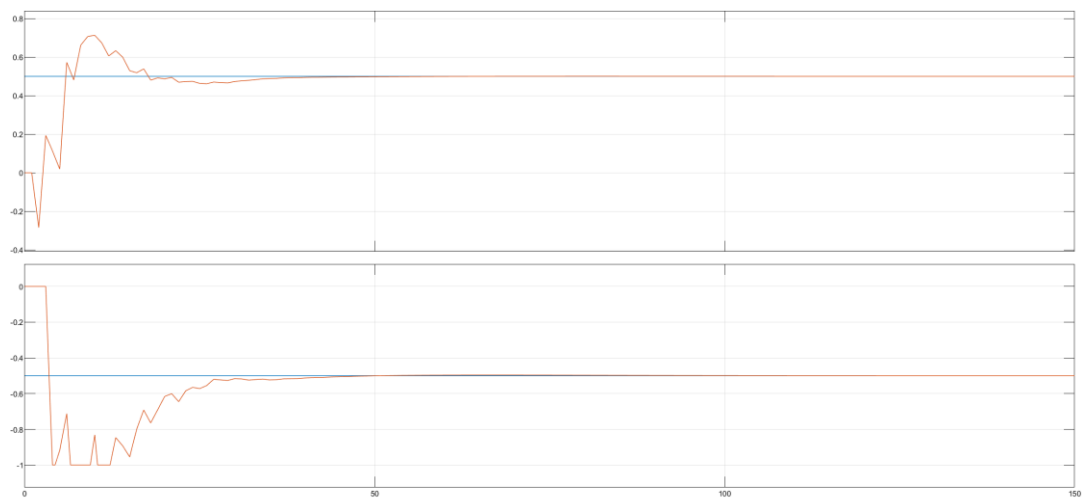


Рис. 4.35 (а). Перехідні процеси з SAC2-агентом для об'єкта зі зміненими параметрами (Експеримент 1)

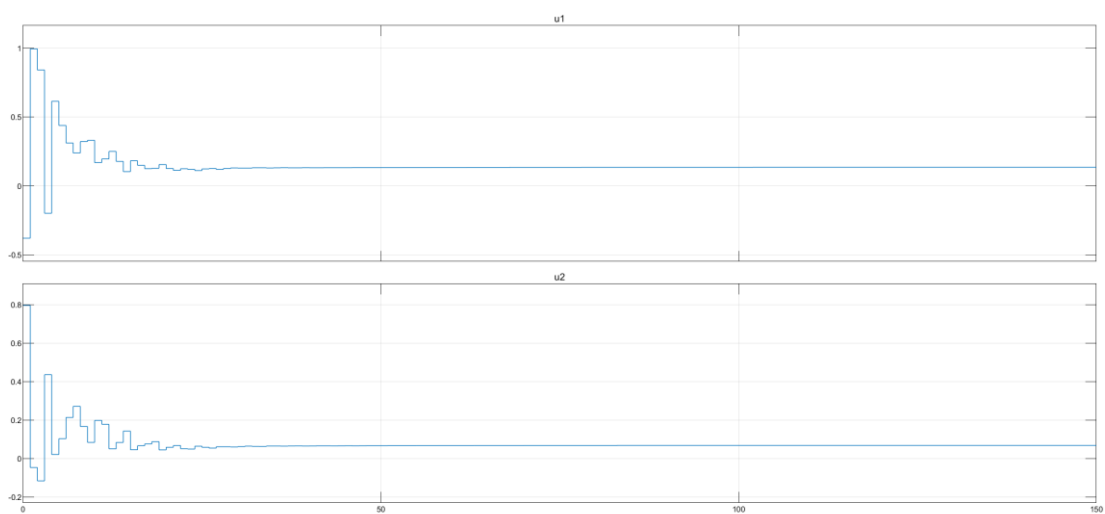


Рис. 4.35 (б). Сигнали управління у системі з SAC2-агентом для об'єкта зі зміненими параметрами (Експеримент 1)

На практиці об'єктам типу дистиляційної колони характерне зменшення коефіцієнтів передачі та збільшення постійних часу внаслідок зміни гідродинамічних режимів, забруднення поверхонь теплообміну, старіння обладнання або коливань у складі сировини. Такі фактори призводять до погіршення динамічних властивостей системи, зменшення швидкості реакції на збурення та ускладнення процесу автоматичного регулювання.

Експеримент 2. Зменшення коефіцієнтів передачі на 15% та збільшення постійних часу на 15% (Рис. 4.36).

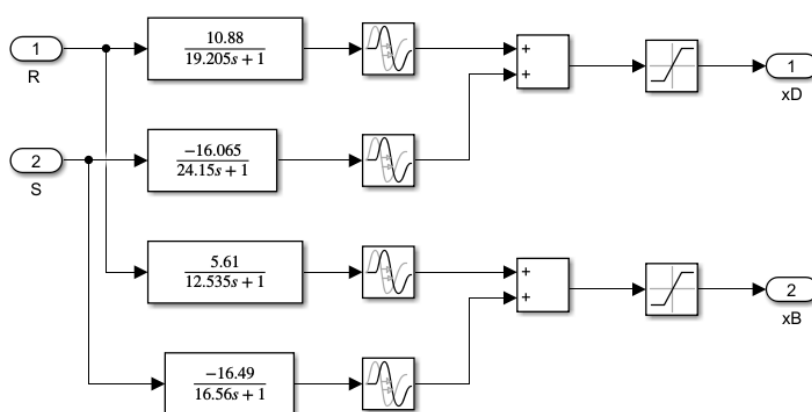


Рис. 4.36. Модель дистиляційної колони зі зміненими параметрами
(Експеримент 2)

У умовах зменшення коефіцієнтів передачі на 15% і збільшення сталих часу на 15%, що відображає типові зміни в дистиляційній колоні через гідродинамічні режими, забруднення, старіння обладнання або коливання сировини, SAC1 є найкращим за загальною ефективністю (ІКК = 13,44) та точністю в усталеному режимі (статична похибка = 0,0005). SAC2 показує збалансовану продуктивність (ІКК = 20,33, статична похибка = 0,00008) і швидку реакцію (час перехідного процесу = 90). ПІ-регулятор вирізняється стабільністю (динамічна похибка = 0,055/0,234), але має гіршу ефективність (ІКК = 40,71). TD3 демонструє найгіршу продуктивність через значні відхилення (ІКК = 24,53, статична похибка = 0,004) і повільну стабілізацію (час перехідного процесу = 140) (Табл. 4.7).

Таблиця 4.7. Порівняння систем регулювання для об'єкта зі зміненими параметрами (Експеримент 2)

Модель	ІКК	Статична похибка	Час перехідного процесу	Затримка	Динамічна похибка
PI-регулятори	40,71	0,0015	135	3	0,055/0,234
TD3	24,53	0,004	140	3	-/0,5
SAC1	13,44	0,0005	80	3	0,218/-
SAC2	20,33	0,00008	90	3	0,143/0,5

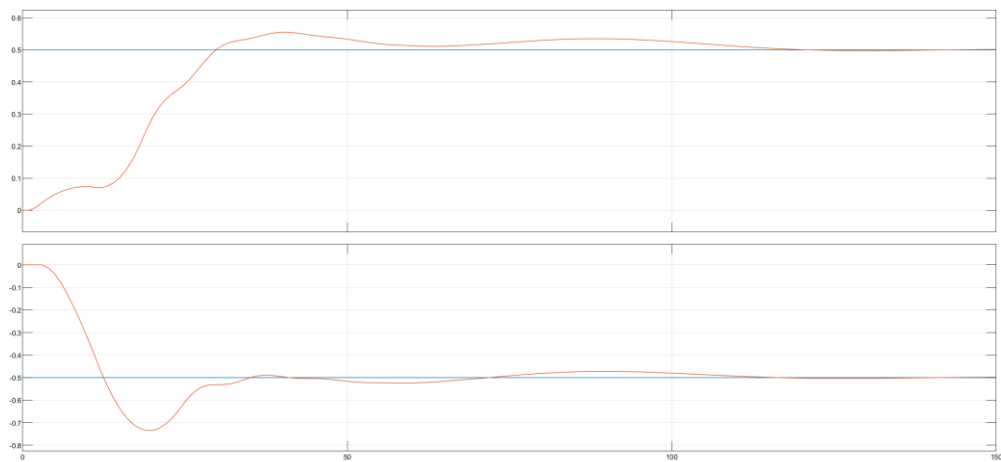


Рис. 4.37 (а). Перехідні процеси з ПІ-регулятором для об'єкта зі зміненими параметрами (Експеримент 2)

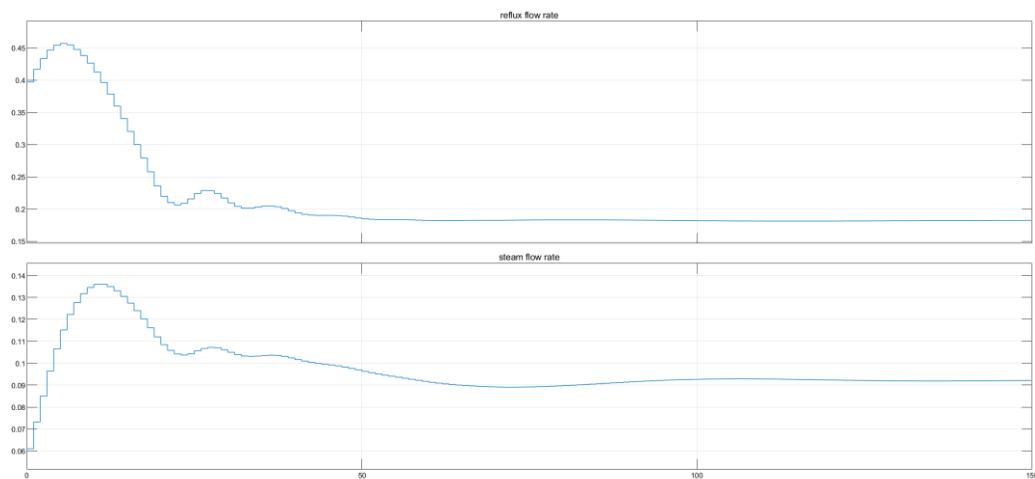


Рис. 4.37 (б). Сигнали управління у системі з ПІ-регулятором для об'єкта зі зміненими параметрами (Експеримент 2)

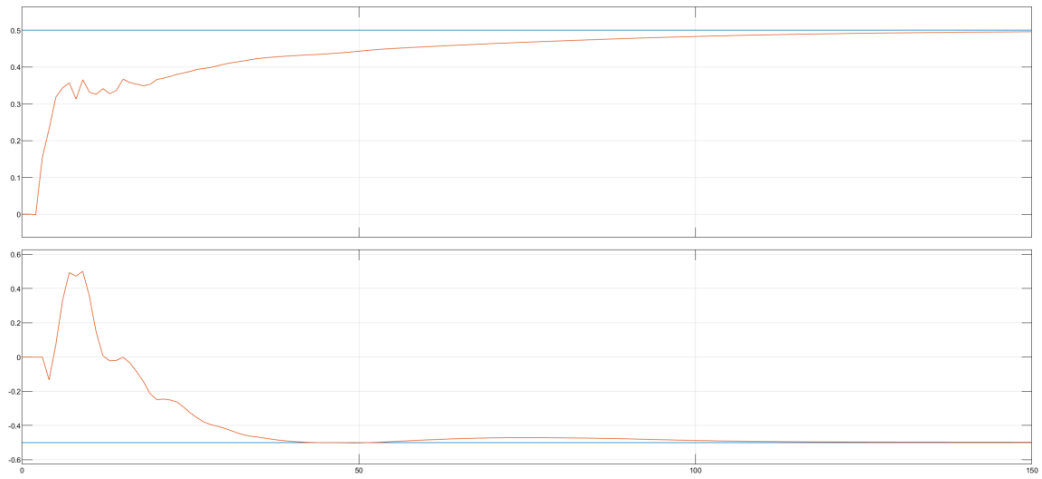


Рис. 4.38 (а). Перехідні процеси з TD3-агентом для об'єкта зі зміненими параметрами (Експеримент 2)

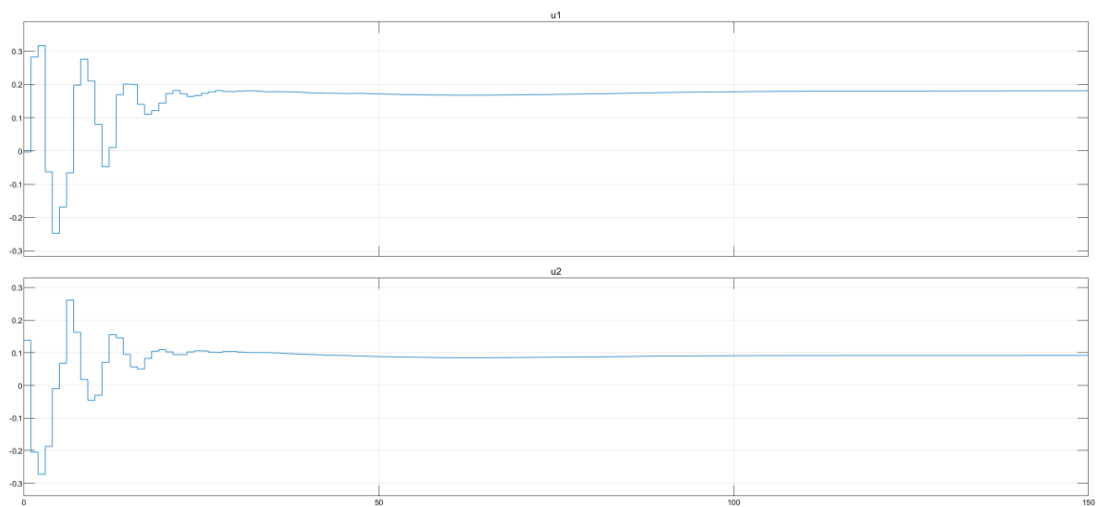


Рис. 4.38 (б). Сигнали управління у системі з TD3-агентом для об'єкта зі зміненими параметрами (Експеримент 2)

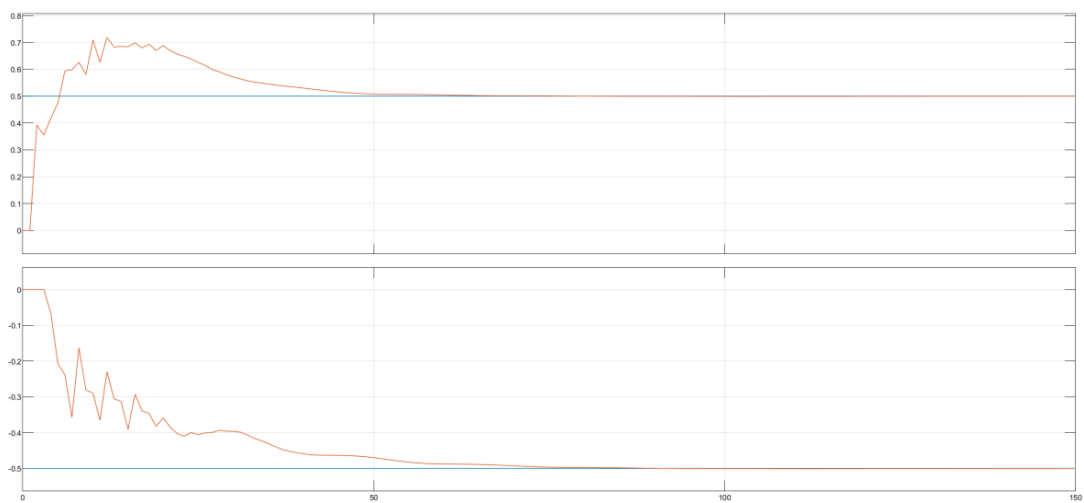


Рис. 4.39 (а). Перехідні процеси з SAC1-агентом для об'єкта зі зміненими параметрами (Експеримент 2)

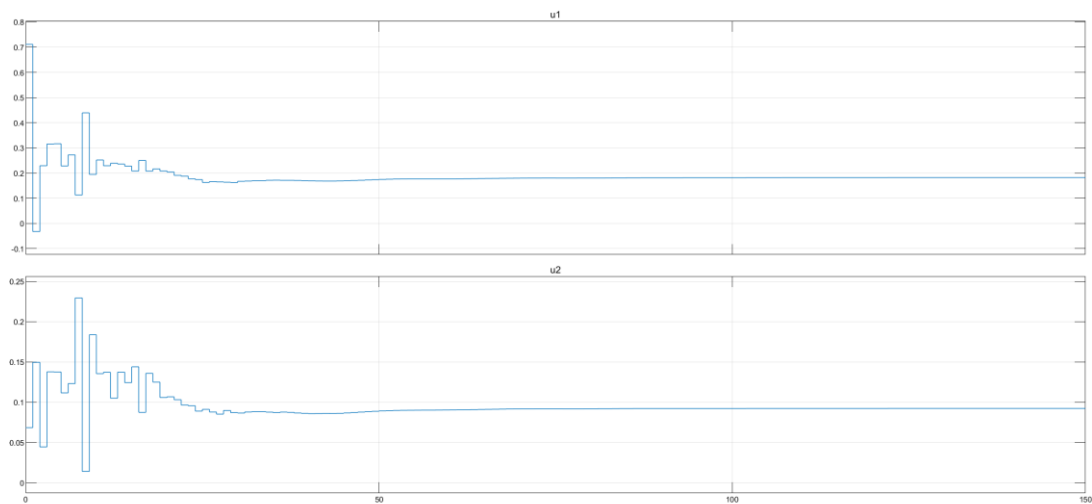


Рис. 4.39 (б). Сигнали управління у системі з SAC1-агентом для об'єкта зі зміненими параметрами (Експеримент 2)

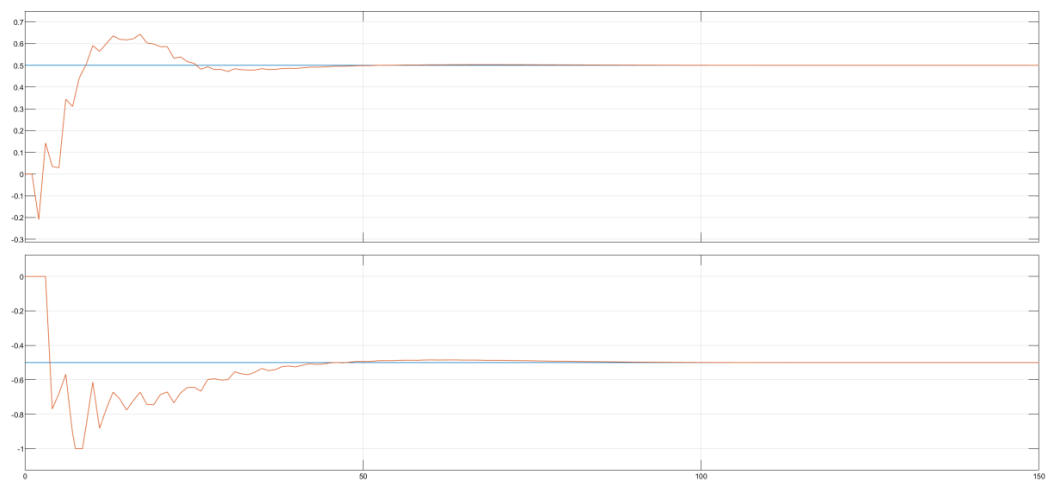


Рис. 4.40 (а). Перехідні процеси з SAC2-агентом для об'єкта зі зміненими параметрами (Експеримент 2)

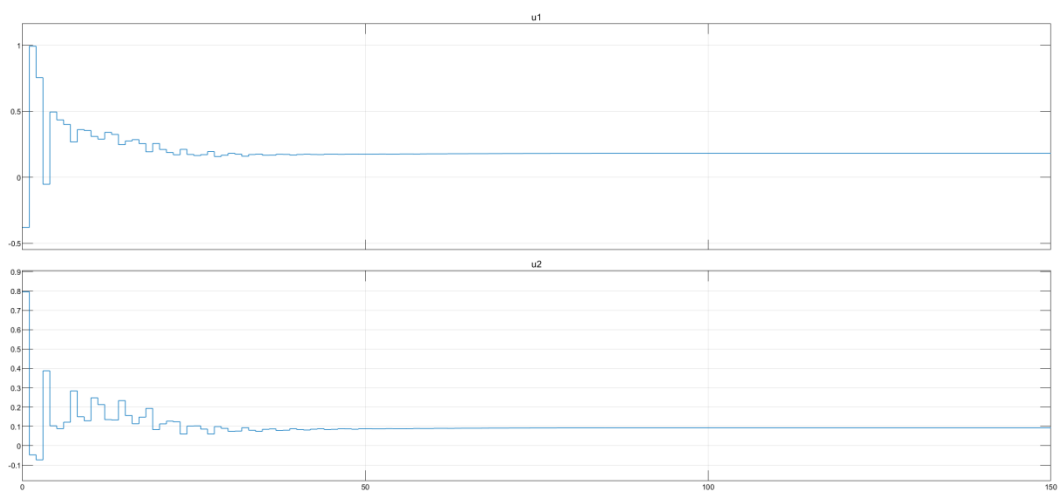


Рис. 4.40 (б). Сигнали управління у системі з SAC2-агентом для об'єкта зі зміненими параметрами (Експеримент 2)

Висновок. Проведені дослідження показали, що SAC1 (Soft Actor-Critic з Entropy weight 1) є найкращим регулятором для управління дистиляційною колоною в усіх протестованих умовах: стабільних (ІКК 13,44), із шумами (ІКК 14,42), при відхиленні параметрів на 15% (ІКК 13,31), а також при зменшенні коефіцієнтів передачі на 15% і збільшенні сталих часу на 15% (ІКК 13,44), демонструючи високу ефективність. SAC2 (Soft Actor-Critic з Entropy weight 0.3) забезпечує кращу точність (статична похибка 0,00008–0,0002) і швидкість реакції (час перехідного процесу 50–90), але поступається SAC1 за ІКК. ПІ-регулятор ефективний за стабільністю (динамічна похибка 0,055–0,085), але втрачає якість при значних змін параметрів об'єкта (ІКК до 40,71 при змінах параметрів об'єкта). TD3 послідовно показує найгірші результати (ІКК до 25,75, динамічна похибка до 0,5), потребуючи суттєвого донавчання.

При ускладненні умов (зменшення коефіцієнтів передачі і збільшеннях сталих часу, зміни нелінійних залежностей) SAC1 і SAC2, зберігають перевагу завдяки здатності донавчатися та адаптуватися, тоді як ПІ-регулятор потребує переналаштування. Для практичного застосування в складних виробничих процесах із високою інерційністю, таких як дистиляційні колони, рекомендується використання SAC-агентів, які забезпечують гнучкість і адаптивність до змін.

4.3 Розробка та апробація віртуальних інструментів у виробництво

4.3.1 Архітектура автоматизованої системи моніторингу

Формулювання вимог щодо впровадження віртуального сенсора. Для впровадження віртуальних інструментів в спиртове виробництво необхідно інтегрувати новий функціонал в існуючу автоматизовану систему управління виробництвом спирту з урахуванням наявних структур, а також стандартів та вимог Індустрії 4.0, ключовим елементом якої виступає промисловий інтернет речей.

Відповідно до концепції ПоТ, отримана система має забезпечувати інтелектуальний та автономний доступ у режимі реального часу, збір, аналіз, комунікацію та обмін процесами, продуктами та службовою інформацією у виробничому середовищі [145], можливість підключення за стандартизованими протоколами обміну інформацією, безпечну передачу даних, надійне зберігання, управління та обробку отриманої інформації [146, 147].

Розробка архітектури системи. Аналіз публікацій в рамках яких розглядається проектування та реалізація ПоТ систем та їх компонентів [148, 150-153], дозволяє виділити перелік ключових елементів на яких будується архітектура інтелектуальної системи такого типу (Рис. 4.41). Серед них:

- 1) Забезпечення збору та фільтрації необхідних даних від об'єкта керування.
- 2) Забезпечення неперервної та безпечної передачі інформації.
- 3) Зберігання та накопичення отриманої інформації в базі даних, яка б забезпечувала високий ступінь ефективності та надійності при роботі з безперервними часовими даними.
- 4) Засоби що дозволяють візуалізацію, взаємодію та аналіз збережених даних.
- 5) Безпосередня реалізація віртуального аналізатору на базі підготовленої інфраструктури.



Рис. 4.41. Структурна схема архітектури для впровадження віртуального аналізатору

4.3.2. Обґрунтування вибору програмних засобів та архітектура системи

Вибір конкретних реалізацій для елементів описаної структури залежить від особливостей об'єкту керування, наявних технічних можливостей та поставлених задач.

В контексті спиртового виробництва об'єктом керування виступає виробниче обладнання технологічного процесу виготовлення спирту. Сучасний завод оснащений автоматизованою системою управління, ключовим елементом якої є промисловий програмований логічний контролер (ПЛК). В процесі своєї роботи ПЛК збирає дані виробничих змінних для опрацювання та генерації керуючих інструкцій. Оскільки ПЛК вже оперує необхідними для віртуального аналізатору даними, доцільно використати його в якості структурного елементу, що постачає зібрану технологічну інформацію для обробки.

Так як виробничі дані надходять у вигляді послідовних масивів змінних, для зручності зберігання та використання доцільно використовувати базу даних часових рядів. В роботі [154] авторами було проведено порівняння БД часових рядів. За результатами дослідження, InfluxDB успішно конкурує з іншими базами даних за низкою ключових параметрів, серед яких пропускна здатність, продуктивність та обсяг використовуваного дискового простору. Являючись однією з найпопулярніших БД для зберігання даних IoT, завдяки наявності безлічі API для взаємодії з іншими платформами, InfluxDB є перспективним кандидатом на роль БД для зберігання даних виробничого процесу.

InfluxDB підтримує взаємодію з Telegraf та Node-RED, що дозволяє отримувати дані з ПЛК. Обидва інструменти можуть бути використані для отримання даних з контролера. Взаємодія та маніпулювання даними в InfluxDB може здійснюватися з використанням таких інструментів як: InfluxDB Query Language (InfluxQL), Chronograf, Grafana, InfluxDB API та ін. В рамках даного дослідження використовується Grafana, оскільки дана програмна система окрім потужних можливостей оперування даними також містить додатковий функціонал у вигляді різноманітних плагінів, в тому числі пов'язаних з машинним навчанням.

4.3.2 Реалізація віртуального сенсора в умовах спиртового виробництва

Враховуючи ключові аспекти побудови IoT архітектури, що були представлені авторами у публікаціях [148, 150-153], а також особливості поставленого завдання щодо впровадження віртуального аналізатора у спиртове виробництво, та результати аналізу програмних засобів, запропонована структура набуває наступного вигляду (Рис. 4.42).

В цій структурі роль PLC виконує симулятор контролера UnityPro, який одночасно відповідає за імітацію роботи технологічної ділянки та промислового контролера. Таким чином обмін даними між структурними складовими набуває наступного вигляду:

- Unity Pro – Node-RED;
- Node-RED – InfluxDB;
- InfluxDB – Grafana.

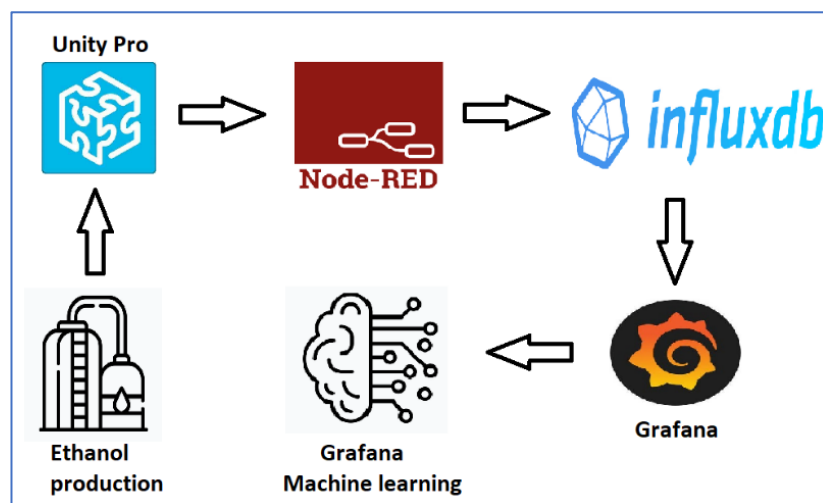


Рис. 4.42. Схема впровадження віртуального аналізатору в склад автоматизованої системи виробництва спирту

Для формування структури використано наступні програмні інструменти:

1) *Unity Pro* - середовище для налаштування, програмування, налагодження та діагностики промислових контролерів Modicon Quantum, Premium, Atrium та M340. Однією з можливостей *Unity Pro*, використаних у даній роботі, є наявність програмного імітатора контролера та здатність доступу до нього через протокол Modbus/TCP.

2) *Node-RED* - інструмент для візуального програмування потоків даних, розроблений компанією IBM для об'єднання різноманітних пристроїв, API та онлайн-сервісів в межах Інтернету речей. Використання Node-RED забезпечує функціонування вузла передачі даних між Unity Pro та InfluxDB.

3) *InfluxDB* - база даних часових рядів з відкритим кодом, написана на мові Go і оптимізована для швидкого запису та отримання даних часових рядів. У даній роботі InfluxDB використовується для зберігання та обробки даних з контролера.

4) *Grafana* - веб-додаток з відкритим кодом для аналізу та візуалізації даних. У цьому проекті Grafana використовується для візуалізації даних з InfluxDB та впровадження плагіну машинного навчання - Grafana Machine Learning, який пропонує користувачам широкий діапазон аналізу даних і генеративних можливостей штучного інтелекту, включаючи створення сповіщень, прогнозування та виявлення відхилень в роботі системи.

Збір та передача даних. Для демонстрації роботи віртуального аналізатору було використано можливості Unity Pro – імітацію промислового контролера та модель симуляції спиртового зброджувача (Рис. 4.43, а).

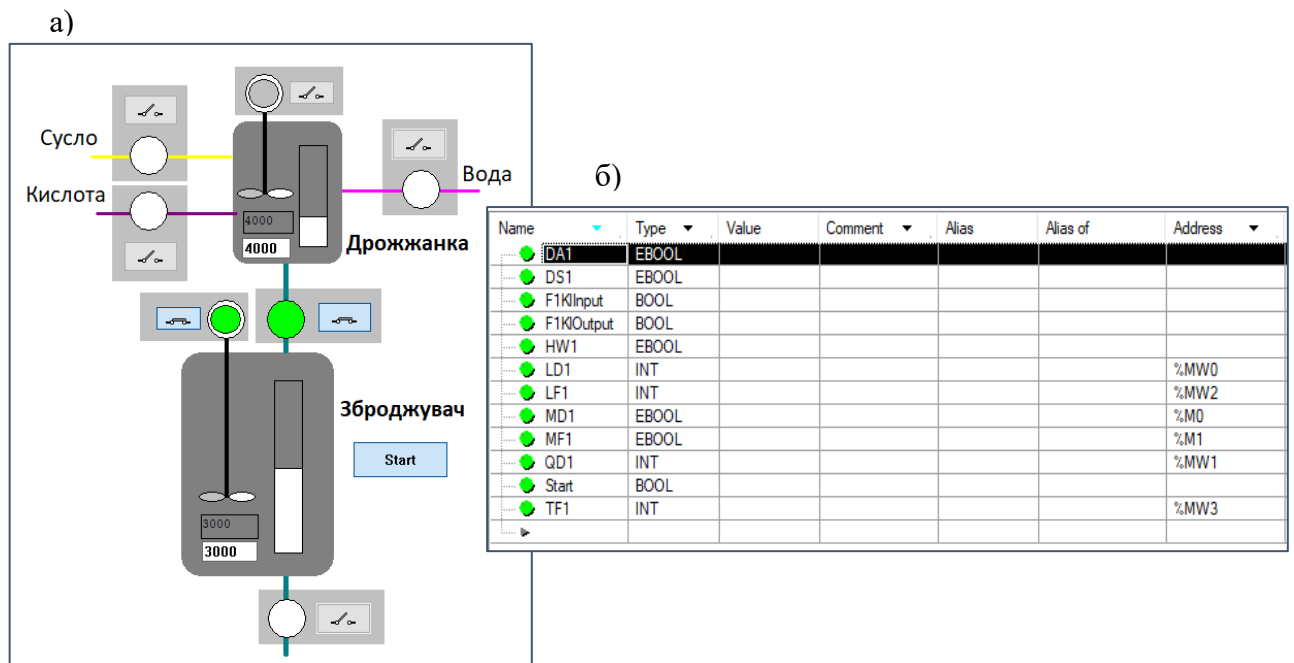


Рис. 4.43. Екран оператора Unity Pro (а) та таблиця змінних PLC (б)

Змінні, обрані для передачі та використання віртуальним аналізатором, мають визначений адрес для їх отримання в Node-RED (Рис. 4.43, б). Це дозволяє налаштувати зв'язок між значеннями в PLC та їх зчитуванням з Holding Registers в Node-RED (Рис. 4.44, а) через підключення через протокол Modbus-TCP (Рис. 4.44, б).

Програма в Node-RED (Рис. 4.45) представляє собою структуру візуального програмування, побудовану з взаємопов'язаних вузлів, що маніпулюють даними під час їх проходження в потоці. Структурно програму можна поділити на такі функціональні блоки:

- Отримання даних з PLC (Рис. 4.44);
- Виділення конкретної змінної з масиву (Рис. 4.46, а);
- Приведення змінної до її діапазону (Рис. 4.46, б);
- Локальна візуалізація отриманих даних (Рис. 4.47);
- Передача відмасштабованих даних до InfluxDB (Рис. 4.49).

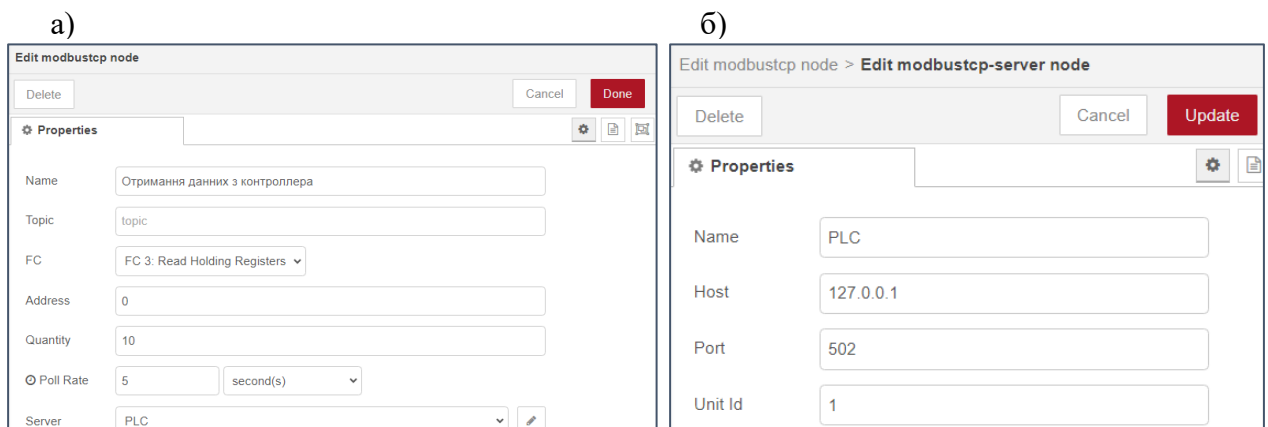


Рис. 4.44 Конфігурація node для отримання даних (а) та з'єднання (б) з PLC

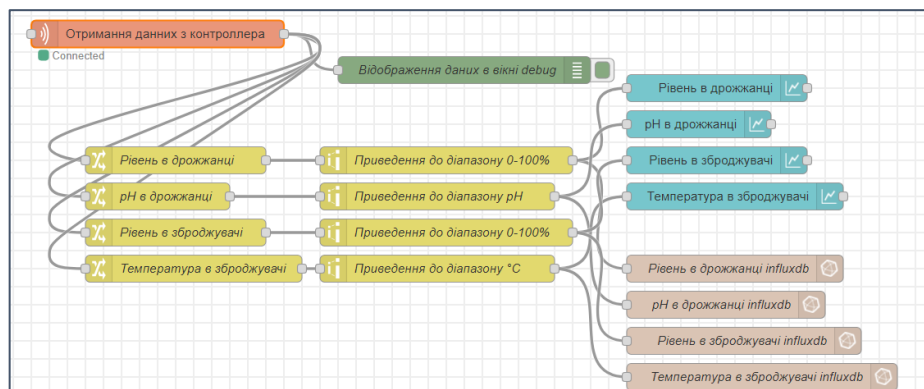


Рис. 4.45. Програма в Node-RED

Дані від контролера надходять у вигляді масиву, що складається з відстежуваних змінних та їх значень, тому першим кроком в обробці є їх розділення на потоки окремих змінних. Для цього в Node-RED використовується change node, який дозволяє відібрати з масиву необхідну змінну (Рис. 4.46, а).

Обрані для передачі та аналізу, відокремлені значення змінних, отримані після обробки в change node, знаходяться в нормалізованому діапазоні контролера, в якому, за замовчуванням, значення виражені у діапазоні від 0 до 10000. Тому для подальшого використання цих даних необхідне їх приведення до відповідного їм природнього діапазону. Використання range node дозволяє масштабувати числові значення під вказані діапазони (Рис. 4.46, б).

Відібрані та відмасштабовані до природнього діапазону дані є готовими для передачі. Використання node-red-dashboard плагіну дозволяє попередньо їх візуалізувати в Node-RED (Рис. 4.47).

Наступним кроком є конфігурування двостороннього зв'язку між Node-RED та InfluxDB. Для конфігурації підключення з боку InfluxDB:

- Використовуються Cluster URL та Organization ID, розміщені на сторінці організації в даних (Рис. 4.48).
- Створюється API Token для надання дозволів на запис чи читання з Buckets InfluxDB (Рис. 4.48)
- Виділяють окремий Bucket для зберігання даних.

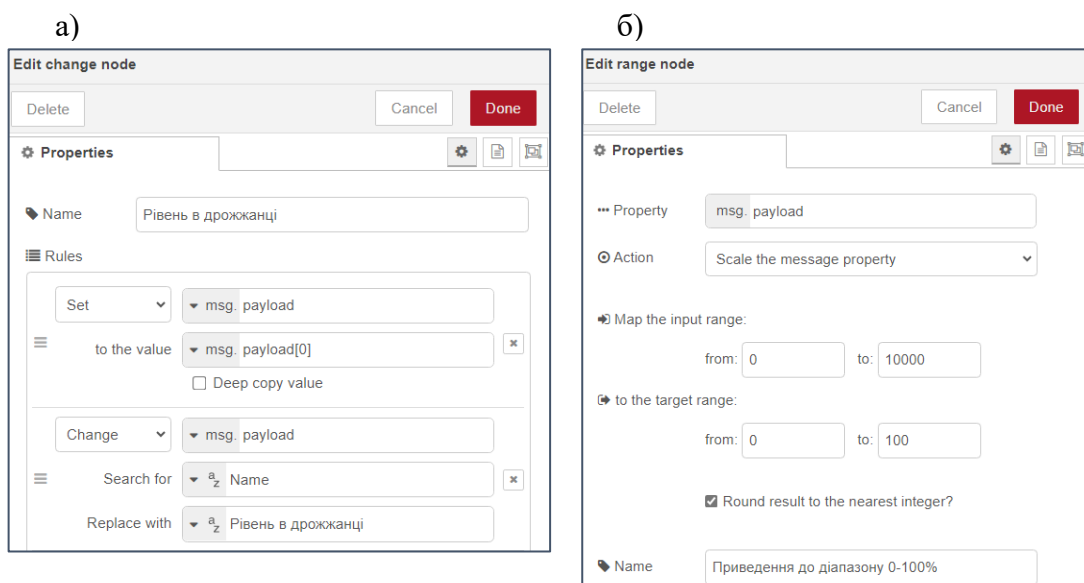


Рис. 4.46. Приклад конфігурації change node (а) та range node (б)

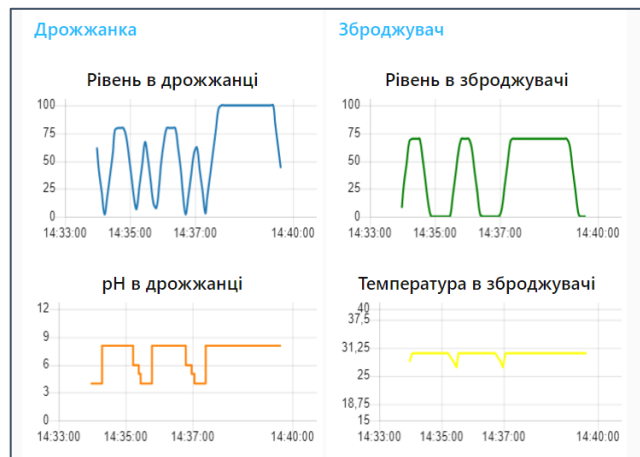


Рис. 4.47. Візуалізація даних в Node-RED

Рис. 4.48. Дані для підключення зі сторони InfluxDB

Конфігураційні дані з InfluxDB використовуються для налаштування підключення з боку Node-RED: Cluster URL та API Token необхідні для конфігурації підключення до серверу (Рис. 4.49, а), Organization ID та дані Bucket – для налаштування зв’язку з спеціально виділеним Bucket (Рис. 4.49, б).

а)

б)

Рис. 4.49. Налаштування підключення до серверу InfluxDB (а) та загальні налаштування взаємодії з InfluxDB (б)

Інтерфейс InfluxDB. Отримані дані з PLC надходять у вказані в налаштуваннях контейнери – Buckets. Кожен Bucket може містити часові ряди (time series data), які відповідають певному контексту або джерелу даних. Вони можуть зберігати значення багатьох змінних одночасно та використовуються для організації даних в InfluxDB та налаштування прав доступу до них.

InfluxDB дозволяє переглядати збережені дані у вигляді таблиці чи графіків використовуючи Data Explorer (Рис. 4.50), а також підтримує створення користувацьких панелей візуалізацій – Dashboard, з можливістю налаштування перегляду необхідних змінних (Рис. 4.51).

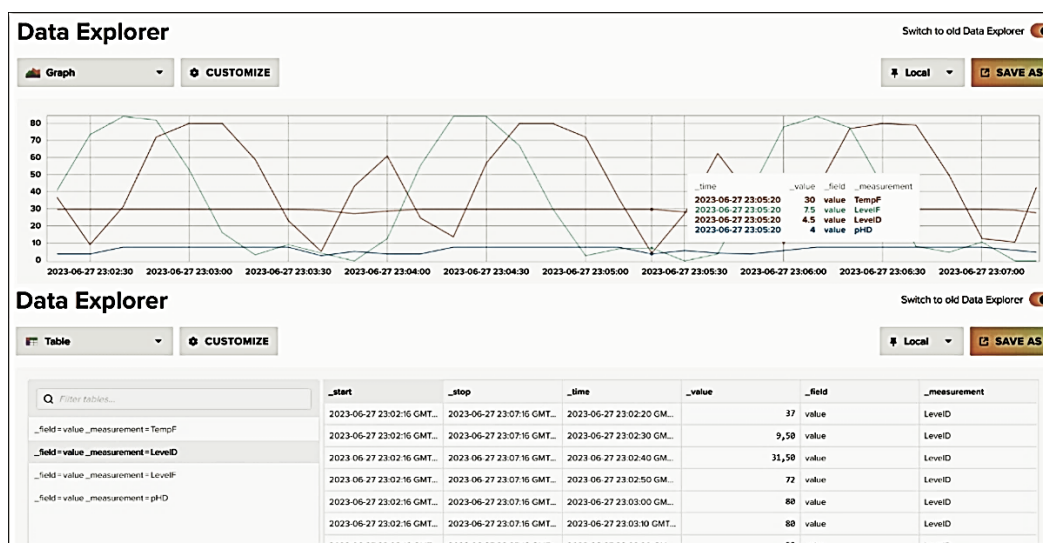


Рис. 4.50. Відображення даних за допомогою Data Explorer InfluxDB

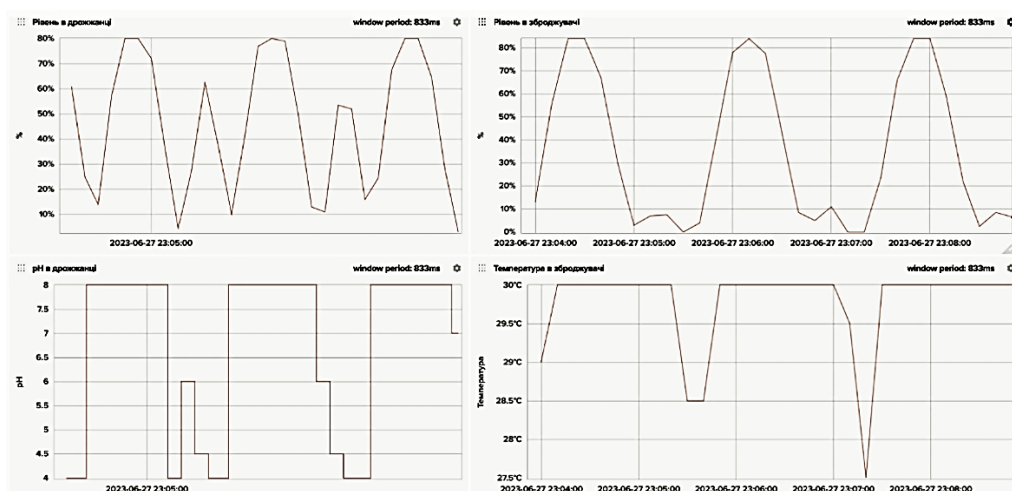


Рис. 4.51. Відображення даних за допомогою Dashboard InfluxDB

Використання Grafana та сервісу машинного навчання Grafana Machine Learning. Grafana є інструментарієм для візуалізації даних з

можливістю взаємодії з InfluxDB та наявністю потужного плагіну аналізу даних, що ґрунтується на методах машинного навчання - Grafana Machine Learning.

Grafana Machine Learning дозволяє прогнозувати поведінку системи, навчаючись на архівних даних роботи системи. Використання цього сервісу потребує встановлення з'єднання між InfluxDB та Grafana, використовуючи дані для підключення зі сторони InfluxDB (Рис. 4.48), та Grafana Data source конфігурацію (Рис. 4.52).

Завдяки сконфігурованому з'єднанню, Grafana отримує доступ до збережених у InfluxDB даних (Рис. 4.53), що дозволяє проводити подальшу обробку цієї інформації.

The image shows the 'InfluxDBData' configuration page in Grafana. It includes fields for 'Name' (InfluxDBData), 'Query Language' (Flux), 'HTTP' settings (URL, Allowed cookies, Timeout), and 'InfluxDB Details' (Organization, Token, Default Bucket, Min time interval, Max series).

Рис. 4.52. Налаштування Grafana Data source для зв'язку з InfluxDB

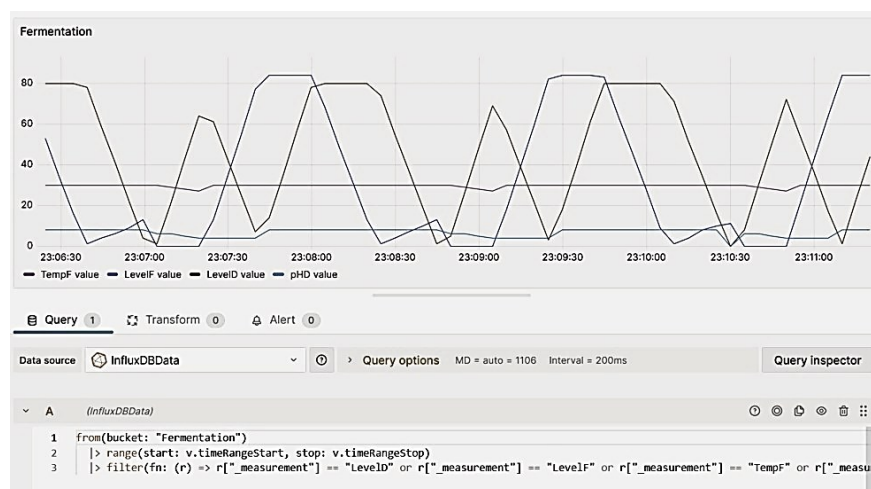


Рис. 4.53. Відображення збережених в InfluxDB даних за допомогою Grafana

Застосування можливостей прогнозування у Grafana Machine Learning вимагає створення та налаштування відповідної метрики прогнозування - Metric Forecast. Це включає формування запиту для вибору необхідної змінної для навчання з відповідного Bucket (Рис. 4.54).

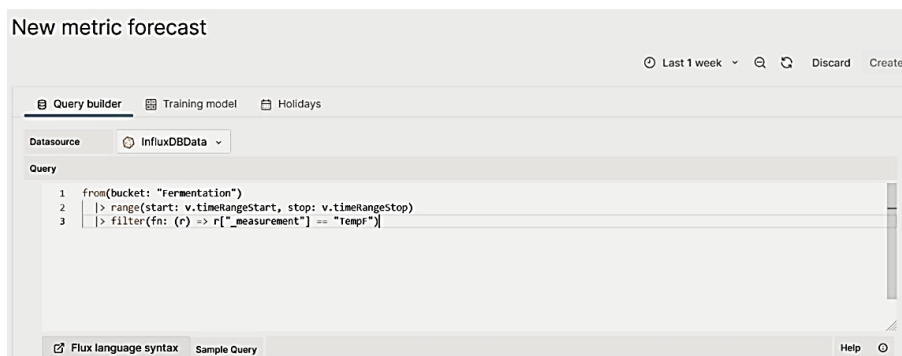


Рис. 4.54. Створення Metric Forecast в Grafana Machine Learning

Сформований Metric Forecast піддається навчанню за допомогою алгоритму Prophet і після завершення надає прогнози щодо поведінки системи у майбутньому, залежно від актуальних даних (Рис. 4.55).

Інтерфейси Grafana Machine Learning дозволяють глибше вивчити прогнозовані набори даних та створювати користувацькі запити до них (Рис. 4.56).

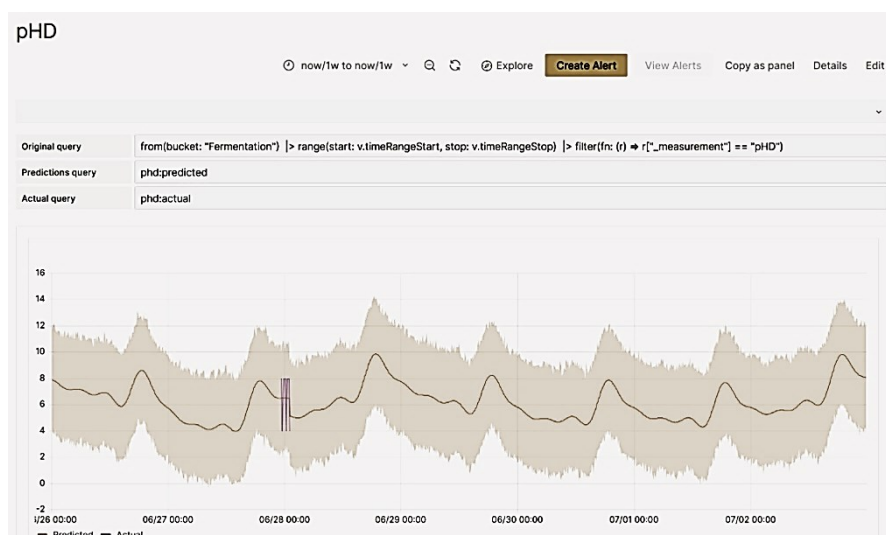


Рис. 4.55. Прогноз зміни поведінки змінної в Grafana Machine Learning

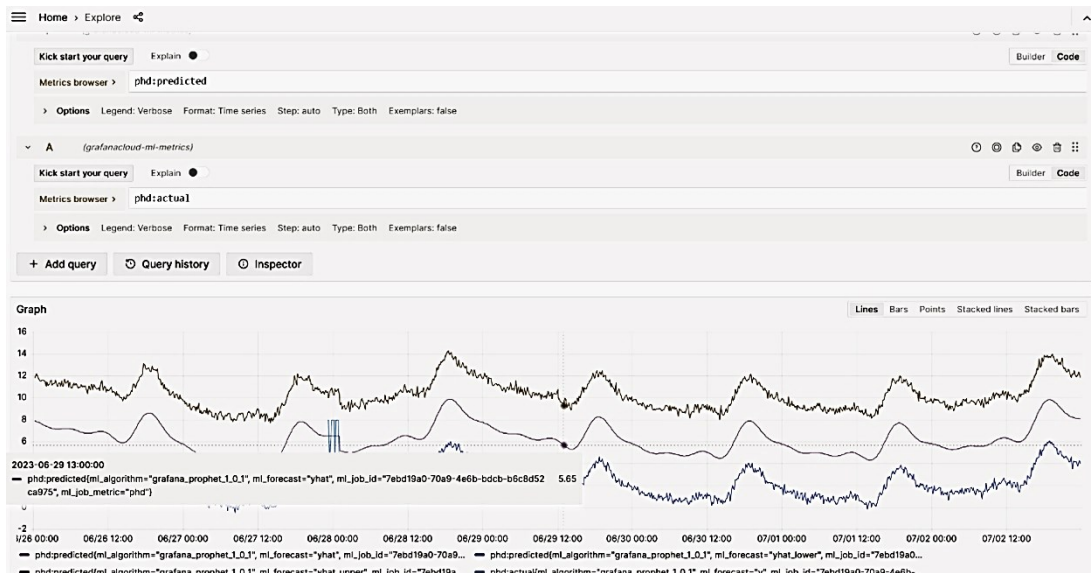


Рис. 4.56. Дослідження прогнозу в Grafana Machine Learning

Використовуючи результати передбачень Grafana Machine Learning, можна відслідковувати зміни довгострокових трендів та отримувати попередження про майбутні зміни технологічних показників процесу спиртового виробництва. Це допомагає зменшити ризик виникнення аварійних ситуацій, пов'язаних з технологічною складовою виробництва, за допомогою своєчасної зміни технологічного режиму та збільшення ритмічності технологічних процесів. Усе це призводить до зменшення матеріальних та енергетичних втрат, зменшення кількості браку та збільшення продуктивності.

4.4 Забезпечення безпеки та стійкості автоматизованої системи управління

4.4.1 Забезпечення функціональної стійкості розробленої системи

Під функціональною стійкістю системи прийнято розуміти властивість системи виконувати свої функції, хоч і ціною втрати певної якості, протягом заданого періоду часу під впливом зовнішніх та внутрішніх дестабілізуючих факторів, таких як експлуатаційні відмови, навмисні пошкодження, втручання в обмін та обробку інформації, а також у разі помилок обслуговуючого персоналу [155].

Типова система автоматизації використовує комплекс заходів для забезпечення стабільності її функціонування. Сюди входять холодне та гаряче

резервування ПЛК, наявність джерел безперебійного живлення, конструювання з передбаченням аварійних ситуацій та ізоляція системи керування від доступу ззовні.

Запропонована система являється функціональною надбудовою над стандартною системою автоматизації спиртового виробництва, тому вихід з ладу її елементів не впливає на перебіг типового технологічного процесу, а лиш нівелює переваги використання віртуального аналізатора на час несправності системи.

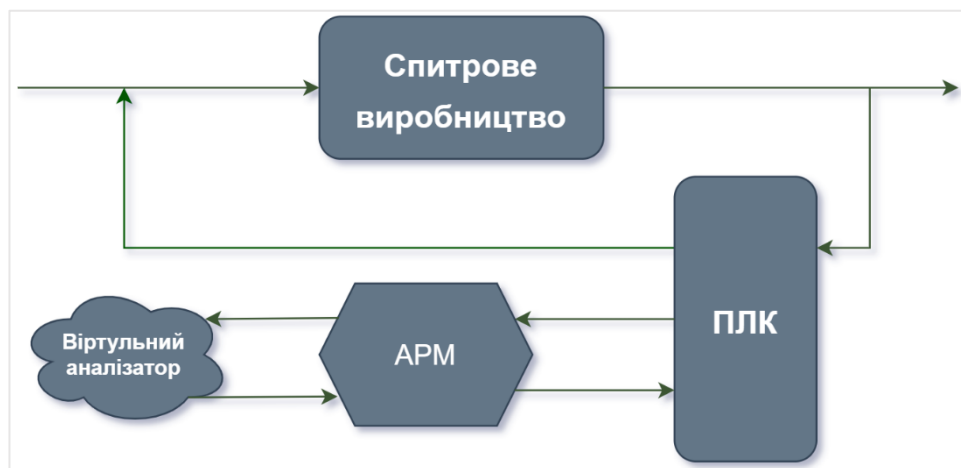


Рис. 4.57. Схема інформаційної взаємодії віртуального аналізатору зі спиртовим виробництвом

Автоматизоване робоче місце оператора слугує елементом, що відмежовує і водночас забезпечує взаємодію між ПЛК, що контролює протікання процесу спиртового виробництва та віртуальним аналізатором, що аналізує перебіг технологічного процесу та надає прогнози та рекомендації стосовно налаштування параметрів та стану обладнання (Рис. 4.52). Завдяки реалізації такої структури забезпечується ізоляція виробничої системи від прямого втручання в її роботу ззовні, та гарантується перевірка експертом результатів інтелектуальної обробки перед їх взаємодією з системою. Таким чином функціональна стійкість системи автоматизації залишається стабільною навіть під час технічних несправностей в контурі віртуального аналізатору.

Функціональна стійкість підсистеми віртуального аналізатору залежить від стану його складових компонентів та зв'язку між ними.

Ключовими елементами в структурі віртуального аналізатору є база даних (БД) часових рядів, що дозволяє акумулювати значні об'єми виробничих даних для аналізу, та сервіс що надає послуги аналізу даних за допомогою методів машинного навчання. У запропонованій реалізації роль БД виконує InfluxDB, а сервіс з функціями машинного навчання Grafana Machine Learning реалізується за допомогою Grafana. Стабільність та доступність InfluxDB та Grafana базується на кількох ключових аспектах:

- Реплікація даних дозволяє створювати копії даних на різних серверах. Це забезпечує надійність та стійкість до відмов: якщо один сервер вийде з ладу, дані все одно будуть доступні з інших реплік.
- Кластеризація. Сервіси можуть бути розгорнуті у кластері для розподілення навантаження та забезпечення високої доступності. Завдяки використанню кластеризації стає можливим автоматично реагувати на відмови серверів та перенаправляти запити на доступні частини системи.
- Системи моніторингу InfluxDB та Grafana дозволяють вчасно виявляти проблеми та надавати повідомлення про них адміністраторам системи.
- Механізми резервного копіювання та відновлення.

Для забезпечення безперервного збору виробничих даних у разі втрати зв'язку з серверами InfluxDB, дані можуть зберігатися локально та надсилатися в хмарну БД при відновленні зв'язку.

Таким чином функціональна стійкість підсистеми віртуального аналізатору забезпечується відповідними інструментами сервісів, що реалізують функції збереження та обробки інформації. При цьому велика увага приділяється ізоляції підсистеми від прямої взаємодії з технологічним процесом, гарантуючи збереження стійкості стандартної системи автоматизації.

4.4.2 Інформаційна безпека системи

Характерним для впровадження інновацій в області інформаційних та комунікаційних технологій, притаманним Індустрії 4.0, є виникнення додаткових ризиків в сфері інформаційної безпеки у зв'язку з інтеграцією нових

систем та розширенням доступу до них. На практиці це означає необхідність визначення адекватних механізмів безпеки, які забезпечать надійність та стабільність роботи, а також забезпечать конфіденційність, цілісність та доступність автоматизованої системи [156].

Серед основних загроз інформаційній безпеці системи віртуального аналізатора можна виділити несанкціонований доступ, витік конфіденційних даних та DoS-атаки (атаки на відмову в обслуговуванні). Для забезпечення належного рівня інформаційної безпеки запропонованої системи віртуального аналізатора необхідно гарантувати захист від несанкціонованого доступу та зовнішніх впливів під час передачі даних до сервісів, а також під час їх обробки та зберігання. Ці завдання значною мірою покладаються на архітектуру сервісів та особливості їх взаємодії.

InfluxDB та Grafana пропонують схожий комплекс інструментів для конфігурації та забезпечення безпеки з'єднання, серед них:

- Можливість передачі даних по протоколу https, який використовує TLS(Transport Layer Security) для шифрування та перевірки цілісності даних , що передаються між клієнтом та сервером.
- Підтримка налаштування аутентифікації користувачів та авторизації їх доступу до баз даних та даних в них.
- Моніторинг доступу та логування подій за допомогою моніторингу безпеки.
- Валідація та фільтрація даних для забезпечення цілісності та безпеки зберігання інформації.

Висновки до розділу 4

1) Розроблено автоматизовану інтелектуальну систему керування дистиляційною колоною з використанням RL-агентів, яка забезпечує стабільне регулювання в динамічних та нестационарних процесах спиртового виробництва, підтримування заданих значень та якісні перехідні процеси, при зміні параметрів об'єкта та середовища спричинених типовими виробничими змінами, такими як

забруднення, старіння обладнання або коливання температури середовища чи складу сировини. Найкращі результати при змінах параметрів дистиляційної колони продемонстрував SAC-агент з Entropy weight 1, забезпечивши високу точність (статична похибка 0,0005–0,0009, що еквівалентно відхиленню концентрації на 0,05–0,09% при нормалізованих значення), мінімальне перевищення (динамічна похибка 0,21–0,264, тобто 21–26,4% від цільового значення, що є допустимим для промислових систем) та високу загальну ефективність (ІКК 13,31–14,42 у різних умовах проти 34,46–40,71 для ПІ-регулятора).

2) Запропоновано та розроблено хмарну архітектуру автоматизованої системи моніторингу виробництвом спирту та біоетанолу, яка дозволяє впроваджувати віртуальні інструменти в структуру виробничої системи та забезпечує обмін технологічною інформацією. Виконано апробацію представленої архітектури з використанням програмно-технічних комплексів Unity Pro, Node-RED і хмарних інструментів InfluxDB, Grafana, Grafana Machine Learning. Така архітектура на базі хмарних інструментів дозволяє легко масштабувати і змінювати систему та її складові, знижує витрати на локальні сервери та забезпечує надійну і зручну взаємодію із виробничими даними.

3) Визначено ключові аспекти забезпечення безпеки та стійкості автоматизованої системи управління, що полягають у реплікації даних, кластеризації сервісів, механізмах моніторингу, резервного копіювання та відновлення, а також у локальному буферизуванні даних для безперервного збору інформації в разі втрати зв'язку з хмарними серверами. Для інформаційної безпеки розглянуто шифрування даних через протокол HTTPS із TLS, аутентифікацію та авторизацію користувачів, моніторинг і логування подій, а також валідацію та фільтрацію даних. Це дає можливість забезпечити високу доступність і надійність системи навіть у разі відмов або атак, гарантувати конфіденційність і цілісність даних, а також ізолювати підсистему віртуального сенсора від прямого впливу на технологічний процес, зберігаючи стійкість локальної системи автоматизації спиртового виробництва.

ОСНОВНІ ВИСНОВКИ ТА РЕЗУЛЬТАТИ

Основні наукові та практичні результати можуть бути узагальнені в таких висновках:

1) Проведений аналіз сучасного стану технологічних комплексів спиртового та біоетанольного виробництва виявив ключові особливості та проблеми автоматизації в цій галузі, що можуть бути вирішені за допомогою використання віртуальних інструментів контролю. Встановлено, що спиртовому виробництву притаманні складні біологічні та хімічні процеси, які характеризуються інерційністю, багатовимірністю, нелінійністю, а також обмеженою можливістю вимірювання важливих змінних через велику кількість взаємозалежних параметрів. Ці особливості ускладнюють застосування класичних методів автоматизації, проте можуть бути вирішені шляхом впровадження віртуальних сенсорів і методів машинного навчання, що сприяє глибшому розумінню процесів, підвищенню якості контролю та кінцевої продукції. Системний аналіз технологічних відділень дозволив ідентифікувати ключові взаємозв'язки між технологічними змінними та потоками на різних рівнях виробництва, що стало основою для визначення задач дослідження.

2) У результаті проведеного аналізу математичних моделей було сформовано прикладну онтологічну модель, призначену для пошуку відповідностей між математичними моделями, задачами та особливостями досліджуваної системи. Онтологічний аналіз відповідності існуючих математичних моделей до поставлених задач показав, що традиційні моделі мають обмеження в адаптивності до динамічних умов виробництва, зокрема через складність врахування нелінійних взаємозалежностей і змін параметрів об'єкта. На основі аналізу сформовано перелік задач, які можуть бути ефективно вирішені з використанням віртуальних інструментів, зокрема прогнозування невимірюваних змінних, оцінка стабільності процесів і компенсація відмов датчиків та інтелектуальне керування. Запропонована онтологічна модель слугує інструментом підтримки прийняття рішень, полегшуючи вибір адекватних моделей для конкретних задач, скорочуючи час їх пошуку та сприяючи інтеграції

нових знань у системи автоматизації. Таким чином, розроблена онтологічна модель створює методологічну основу для ефективного використання віртуальних сенсорів у моніторингу технологічних процесів спиртового виробництва.

3) Розроблено механізм та алгоритм для класифікації стабільності роботи дистиляційного відділення в режимі реального часу на основі статистичних моделей машинного навчання. Запропонований підхід дозволяє виявляти ознаки нестабільності на ранніх етапах та своєчасно реагувати, що зменшить втрати часу, сировини та якості продукту.

4) Проведено моделювання невимірюваних показників на основі NARX-моделей. Отримані результати показали, що модель здатна пояснити 99% варіацій цільової змінної при прогнозуванні концентрації етанолу. Досліджено вплив гіперпараметрів NARX-моделей на їхню продуктивність та можливості автоматизованого налаштування. Оптимізація цих гіперпараметрів дозволяє пришвидшити процес моделювання та підвищити точність прогнозування.

5) Розроблені моделі ВС на основі алгоритмів машинного навчання (XGBoost, Decision Tree, NARX, Gradient Boosting та RNN) та проведено порівняльний аналіз їх ефективності для інтелектуального моніторингу даних дистиляційного відділення. Результати проведених досліджень продемонстрували ефективність використання моделей ВС для прогнозування критичних змінних у системах автоматичного регулювання технологічних процесів спиртового виробництва. Зокрема, моделювання витрати бражки на основі показників інших сенсорів дозволило з коефіцієнтом детермінації $R^2=0,964$ забезпечити безперервність автоматизованого керування навіть у разі відмови датчика витрати. Аналогічно, прогнозування тиску в колоні концентрації за відсутності даних від датчика тиску здатне пояснити 98% цільової змінної, що підтверджує можливість підтримки стабільної роботи системи при відмові відповідного сенсора. Отримані результати вказують на доцільність впровадження інтелектуальних моделей для моніторингу та

управління технологічними процесами виробництва спирту, що підвищує надійність та ефективність роботи автоматизованих систем.

6) Розроблено алгоритми для оптимізації режимів роботи технологічних відділень, зокрема бродильного відділення. Моделювання відділення на основі моделей дерев регресії та алгоритм оптимізації дозволяють знаходити оптимальні налаштування процесу відповідно до заданих критеріїв, що забезпечує збільшення виходу етанолу в середньому на 8,4% або скорочення часу бродіння на 12,2%.

7) Розроблено концептуальну структуру автоматизованої системи моніторингу та керування технологічними процесами з інтеграцією віртуальних інструментів контролю. Представлена структура забезпечує замкнений цикл керування: від збору даних із фізичних і віртуальних сенсорів до формування управляючих дій із урахуванням прогнозованих значень і режимів оптимізації завдяки використанню модулів класифікації стабільності процесу, моніторингу стану датчиків, прогнозування та оптимізації режимів та автоматичного керування. Інтеграція віртуальних інструментів контролю дозволяє компенсувати обмеження традиційних систем, підвищувати надійність моніторингу та забезпечувати адаптивність системи до змін умов експлуатації.

8) Розроблено інтелектуальну систему керування з використанням RL-агентів, яка забезпечує стабільне регулювання в динамічних та інерційних процесах спиртового виробництва, де ПІ-регулятор не здатний ефективно підтримувати задані значення та якісні перехідні процеси через зміни об'єкта й середовища, спричинені типовими виробничими факторами, такими як забруднення, старіння обладнання, коливання температури середовища чи складу сировини. Найкращі результати при змінах параметрів дистиляційної колони продемонстрував Soft Actor-Critic агент (SAC з Entropy weight 1), який забезпечив високу загальну ефективність (низький ІКК 13,31–14,42), високу точність в усталеному режимі (статична похибка $<0,01\%$).

9) Запропоновано та розроблено хмарну архітектуру автоматизованої системи моніторингу виробництвом спирту та біоетанолу, яка дозволяє

впроваджувати віртуальні інструменти в структуру виробничої системи та забезпечує обмін технологічною інформацією. Виконано апробацію представленої архітектури з використанням програмно-технічних комплексів Unity Pro, Node-RED і хмарних інструментів InfluxDB, Grafana, Grafana Machine Learning. Така архітектура на базі хмарних інструментів дозволяє легко масштабувати і змінювати систему та її складові, знижує витрати на локальні сервери та забезпечує надійну і зручну взаємодію із виробничими даними.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Хоружий, Б. В., & Слободян, Н. Г. (2014). Сучасний стан та стратегічні орієнтири спиртової промисловості України. Науковий вісник Херсонського державного університету. Сер.: Економічні науки, 4, 80-83.
2. Шаманська, О. І., & Паламаренко, Я. В. (2017). Дослідження ефективності функціонування спиртової промисловості України. Економіка. Фінанси. Менеджмент: актуальні питання науки і практики, 1, 19-27.
3. Скрипниченко, В. А. (2014). Інноваційні перспективи виробництва біопалива в Україні. Науковий вісник Національного університету біоресурсів і природокористування України. Серія: Економіка, аграрний менеджмент, бізнес, 200(1), 291-298.
4. Kaletnik, G., Prutska, O., & Pryshliak, N. (2014). Resource potential of bioethanol and biodiesel production in Ukraine. Visegrad Journal on Bioeconomy and Sustainable Development, 3(1), 9-12.
5. Кизюн, Г. О., Міщенко, О. С., & Толстопятов, О. М. (2012). Особливості застосування біоетанолу в Україні. Цукор України, (9), 35-36.
6. Антонюк, П. О. (2014). Перспективи розвитку біоетанольного виробництва в Україні. Економіка харчової промисловості, 2, 78-82.
7. Скорук, О. П., & Зубар, І. В. (2014). Пріоритетні напрями виробництва біоетанолу в Україні. Економіка АПК, 2, 36-41.
8. Kyshenko, V., Korobiichuk, I., & Rzeplińska-Rykała, K. (2020). Technological monitoring in the management of the distillation-rectification plant. In Conference on Automation (pp. 165-174). Springer, Cham.
9. Булій, Ю. В., Куц, А. М., Бондар, М. В., & Мукоїд, Р. М. (2020). Підвищення якості спирту етилового ректифікованого в процесі брагоректифікації. Харчова промисловість, 28, 17-24.
10. Li, S.-Z., & Chan-Halbrendt, C. (2009). Ethanol production in (the) People's Republic of China: potential and technologies. Applied Energy, 86, 162-169.
11. Bahrin, M. A. K. (2016). Industry 4.0: A review on industrial automation and robotic. Jurnal Teknologi, 78, 6-13.

12. Українець, А. І., Шиян, П. Л., Булій, Ю. В., & Куц, А. М. (2018). Дослідження руху органічних домішок спирту в процесі брагоректифікації. Наукові проблеми харчових технологій та промислової біотехнології в контексті Євроінтеграції: програма та тези матеріалів VII-ї Міжнародної науково-технічної конференції (с. 97-98).
13. Українець, А. І., Булій, Ю. В., Шиян, П. Л., & Куц, А. М. (2017). Ресурсо- та енергозберігаюча технологія ректифікованого етилового спирту. Наукові праці Національного університету харчових технологій, 6, 213-218.
14. Олійничук, С. Т., & Коткова, Н. С. (2014). Спиртове виробництво та енергетична безпека України. Економіка АПК, (4), 60-64.
15. Balat, M., Balat, H., & Öz, C. (2008). Progress in bioethanol processing. Progress in Energy and Combustion Science, 34(5), 551-573.
16. Aditiya, H. B., Mahlia, T. M. I., Chong, W. T., Nur, H., & Sebayang, A. H. (2016). Second generation bioethanol production: A critical review. Renewable and Sustainable Energy Reviews, 66, 631-653.
17. Тимошенко, О. В. (2019). Виклики та загрози четвертої промислової революції: наслідки для України. Бізнес Інформ, 2(493), 21-29.
18. Gilchrist, A. (2016). Introducing Industry 4.0. In Industry 4.0: The Industrial Internet of Things (pp. 195-215).
19. Brecher, C. (2021). Automation technology as a key component of the Industry 4.0 production development path. The International Journal of Advanced Manufacturing Technology, 117, 2287-2295.
20. Rüßmann, M., et al. (2015). Industry 4.0: The future of productivity and growth in manufacturing industries. Boston Consulting Group, 9(1), 54-89.
21. Yaqoob, I. (2016). Big data: From beginning to future. International Journal of Information Management, 36(6), 1231-1247.
22. Sagioglu, S., & Sinanc, D. (2013). Big data: A review. In 2013 International Conference on Collaboration Technologies and Systems (CTS) (pp. 42-47). IEEE.
23. Maddikunta, P. K. R. (2022). Industry 5.0: A survey on enabling technologies and potential applications. Journal of Industrial Information Integration, 26, 100257.

24. Özdemir, V., & Hekim, N. (2018). Birth of industry 5.0: Making sense of big data with artificial intelligence, "the internet of things" and next-generation technology policy. *Omics: A Journal of Integrative Biology*, 22(1), 65-76.
25. Paschek, D. (2019). Industry 5.0—The expected impact of next industrial revolution. In *Thriving on Future Education, Industry, Business, and Society*, Proceedings of the MakeLearn and TIIM International Conference (pp. 15-17). Piran, Slovenia.
26. Nahavandi, S. (2019). Industry 5.0—A human-centric solution. *Sustainability*, 11(16), 1-13.
27. Jiang, Y. (2020). A review on soft sensors for monitoring, control, and optimization of industrial processes. *IEEE Sensors Journal*, 21(11), 12868-12881.
28. Bakhtadze, N. N. (2004). Virtual analyzers: Identification approach. *Automation and Remote Control*, 65(11), 1691-1709.
29. Harrold, D. (2001). Process control's latest tool: Soft sensors. *Control Engineering Europe*, June/July, 42-45.
30. Ljung, L. (1987). *System identification: Theory for the user*. Englewood Cliffs: Prentice Hall.
31. Bartos, F. J. (1997). Artificial intelligence: Smart thinking for complex control. *Control Engineering*, (7).
32. Zadeh, L. A. (1987). *Fuzzy sets and applications*. New York: Wiley.
33. Masalovich, A. I. (1996). Computer... predicts. *Softmarket*, (23), 6-9.
34. Kosko, B. (1992). *Fuzzy thinking*. New York: Hyperion.
35. Kosko, B. (1992). *Neural networks and fuzzy systems: A dynamic system approach to machine intelligence*. New Jersey: Prentice Hall.
36. Holland, J. (1992). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan.
37. Booker, L. B., Goldberg, D. E., & Holland, J. H. (1989). Classifier systems and genetic algorithms. *Artificial Intelligence*, 40(2), 235-282.
38. Goldberg, D. (1989). *Genetic algorithms in machine learning, optimization and search*. Massachusetts: Addison-Wesley.

39. Jones, A. J. (1993). Genetic algorithms and their applications to the design of neural network. *Neural Computing Applications*, 1(1).
40. Soderstrom, T., & Stoica, P. (1983). *Variable methods for system identification*. New York: Springer.
41. Мусаев, А. А. (2003). Виртуальные анализаторы: концепция построения и применения в задачах управления непрерывными технологическими процессами. *Автоматизация в промышленности*, (8), 28-33.
42. Luttmann, R. (2012). Soft sensors in bioprocessing: A status report and recommendations. *Biotechnology Journal*, 7(8), 1040-1048.
43. Martin, D., Kühl, N., & Satzger, G. (2021). Virtual sensors. *Business & Information Systems Engineering*, 63(3), 315-323.
44. Kano, M., & Fujiwara, K. (2013). Virtual sensing technology in process industries: Trends and challenges revealed by recent industrial applications. *Journal of Chemical Engineering of Japan*, 46(1), 1-17.
45. Kadlec, P., Gabrys, B., & Strandt, S. (2009). Data-driven soft sensors in the process industry. *Computers & Chemical Engineering*, 33(4), 795-814.
46. Hermann, M., Pentek, P., & Otto, B. (2015). Design principles for Industrie 4 scenarios: A literature review. Business Engineering Institute.
47. Kagermann, H., Wahlster, W., & Helbig, J. (2013). Recommendations for implementing the strategic initiative Industrie 4.0: Final report of the Industrie 4.0 Working Group. Ulrike Findeklee: Acatech – National Academy of Science and Engineering.
48. Schlaepfer, R. C., & Koc, M. (2015). Industry 4.0: Challenges and solutions for the digital transformation and use of exponential technologies. Deloitte AG, Audit, Tax, Consulting, Corporate Finance.
49. Michael, R., Markus, L., et al. (2015). Industry 4.0: The future of productivity and growth in manufacturing industries. Boston Consulting Group.
50. Biechele, P., Busse, C., Solle, D., Scheper, T., & Reardon, K. (2015). Sensor systems for bioprocess monitoring. *Engineering in Life Sciences*, 15(5), 469-488.

51. Pachauri, N., Singh, V., & Rani, A. (2017). Two degree of freedom PID based inferential control of continuous bioreactor for ethanol production. *ISA Transactions*, 68, 235-250.
52. Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10, 988-999.
53. Suykens, J. K., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters*, 6, 293-300.
54. Gu, B., & Pan, F. (2015). A soft sensor modelling of biomass concentration during fermentation using accurate incremental online v-Support Vector Regression learning algorithm. *American Journal of Biochemistry and Biotechnology*, 11, 149.
55. Behnasr, M., & Jazayeri-Rad, H. (2015). Robust data-driven soft sensor based on iteratively weighted least squares support vector regression optimized by the cuckoo optimization algorithm. *Journal of Natural Gas Science and Engineering*, 22, 35-41.
56. Yang, Q., & Yan, F. (2012). Soft sensor of biomass in fermentation process based on robust neural network. In *Communications and Information Processing* (pp. 273-280). Springer, Berlin, Germany.
57. Hongyi, L., Haoming, Z., & Yinhua, W. (2010). Study on soft sensor for lysine fermentation based on BP neural network. In *Proceedings of the 2010 8th World Congress on Intelligent Control and Automation* (pp. 4120-4124). Jinan, China.
58. Liu, G., Yu, S., Mei, C., & Ding, Y. (2011). A novel soft sensor model based on artificial neural network in the fermentation process. *African Journal of Biotechnology*, 10, 19780-19787.
59. Wu, J.-Q., Sun, Y.-K., Huang, Y.-H., & Sun, L.-N. (2014). Soft sensor modeling based on GRNN for biological parameters of marine protease fermentation process. In *Proceedings of the 33rd Chinese Control Conference* (pp. 5102-5106). Nanjing, China.
60. Yu, P., Lu, J., Wu, Y., & Sun, Y. (2006). Soft sensor based on kernel principal component analysis and radial basis function neural network for microbiological fermentation. In *Proceedings of the 2006 6th World Congress on Intelligent Control and Automation* (pp. 4851-4855). Dalian, China.

61. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (pp. 1097-1105). Lake Tahoe, NV, USA.
62. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61, 85-117.
63. Gopakumar, V., Tiwari, S., & Rahman, I. (2018). A deep learning based data-driven soft sensor for bioprocesses. *Biochemical Engineering Journal*, 136, 28-39.
64. Yao, L., & Ge, Z. (2017). Deep learning of semisupervised process data with hierarchical extreme learning machine and soft sensor application. *IEEE Transactions on Industrial Electronics*, 65, 1490-1498.
65. Lin, Y., & Yan, W. (2015). Study of soft sensor modeling based on deep learning. In *Proceedings of the 2015 American Control Conference (ACC)* (pp. 5830-5835). Chicago, IL, USA.
66. Zhu, J., Ge, Z., Song, Z., & Gao, F. (2018). Review and big data perspectives on robust data mining approaches for industrial process modeling with outliers and missing data. *Annual Reviews in Control*, 46, 107-133.
67. Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of the 19th International Conference on Computational Statistics* (pp. 177-186). Paris, France.
68. Yukun, S., Bo, W., & Shenping, D. (2008). Soft sensor of Lysine fermentation based on fuzzy support vector machines. In *Proceedings of the 2008 27th Chinese Control Conference* (pp. 280-284). Kunming, China.
69. Fidanova, S., Paprzycki, M., & Roeva, O. (2014). Hybrid GA-ACO algorithm for a model parameters identification problem. In *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems* (Vol. 9, pp. 413-420). Warsaw, Poland.
70. Ahuja, K., & Pani, A. K. (2018). Software sensor development for product concentration monitoring in fed-batch fermentation process using dynamic principal component regression. In *Proceedings of the 2018 International Conference on Soft-computing and Network Security (ICSNS)* (pp. 1-6). Coimbatore, India.

71. Zhu, X. (2020). Modern soft-sensing methods for fermentation processes. *Sensors*, 20(6), 1771.
72. Yan, W., Tang, D., & Lin, Y. (2016). A data-driven soft sensor modeling method based on deep learning and its application. *IEEE Transactions on Industrial Electronics*, 64, 4237-4245.
73. Chérury, A. (1997). Software sensors in bioprocess engineering. *Journal of Biotechnology*, 52, 193-199.
74. Benjamin, K. K., Ammanuel, A., David, A., & Benjamin, Y. K. (2008). Genetic algorithm using for a batch fermentation process identification. *Journal of Applied Sciences*, 8, 2272-2278.
75. Robles-Rodriguez, C. E., Bideaux, C., Roux, G., Molina-Jouve, C., & Aceves-Lara, C. A. (2019). Soft-sensors for lipid fermentation variables based on PSO Support Vector Machine (PSO-SVM). In *Distributed Computing and Artificial Intelligence*, 13th International Conference (pp. 175-183). Avila, Spain.
76. Zhang, X., & Pan, F. (2015). Parameters optimization and application to glutamate fermentation model using SVM. *Mathematical Problems in Engineering*, 2015, 1-7.
77. Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning*, 3, 95-99.
78. Sang, H., Yuan, W., Wang, F., & He, D. (2007). Support vector machines and genetic algorithms for soft-sensing modeling. In *International Symposium on Neural Networks* (pp. 330-335). Nanjing, China.
79. Yang, Z., Yao, L., & Ge, Z. (2020). Streaming parallel variational Bayesian supervised factor analysis for adaptive soft sensor modeling with big process data. *Journal of Process Control*, 85, 52-64.
80. De Assis, A. J., & Maciel Filho, R. (2000). Soft sensors development for on-line bioreactor state estimation. *Computers & Chemical Engineering*, 24, 1099-1103.
81. Jin, H., Chen, X., Yang, J., Zhang, H., Wang, L., & Wu, L. (2015). Multi-model adaptive soft sensor modeling method using local learning and online support vector

regression for nonlinear time-variant batch processes. *Chemical Engineering Science*, 131, 282-303.

82. Pearson, R. K. (2002). Outliers in process modeling and identification. *IEEE Transactions on Control Systems Technology*, 10, 55-63.

83. Fortuna, L., Graziani, S., Rizzo, A., & Xibilia, M. G. (2007). *Soft sensors for monitoring and control of industrial processes*. Springer Science & Business Media.

84. Warne, K., Prasad, G., Rezvani, S., & Maguire, L. (2004). Statistical and computational intelligence techniques for inferential model development: A comparative evaluation and a novel proposition for fusion. *Engineering Applications of Artificial Intelligence*, 17, 871-885.

85. Lee, H. W., Lee, M. W., & Park, J. M. (2007). Robust adaptive partial least squares modeling of a full-scale industrial wastewater treatment process. *Industrial & Engineering Chemistry Research*, 46, 955-964.

86. Zhou, J., Guang, F., & Tang, R. (2018). Scenario analysis of carbon emissions of China's power industry based on the improved particle swarm optimization-support vector machine model. *Polish Journal of Environmental Studies*, 27(1).

87. Wang, B., & Ji, X. (2015). Soft-sensing modeling based on MLS-SVM inversion for L-lysine fermentation processes. *International Journal of Bioautomation*, 19, 207-222.

88. Zadeh, L. A. (1988). Fuzzy logic. *Computer*, 21(4), 83-93.

89. Osorio, D. (2008). Soft-sensor for on-line estimation of ethanol concentrations in wine stills. *Journal of Food Engineering*, 87(4), 571-577.

90. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (pp. 1097-1105). Lake Tahoe, NV, USA.

91. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61, 85-117.

92. Shang, C., Yang, F., Huang, D., & Lyu, W. (2014). Data-driven soft sensor development based on deep learning technique. *Journal of Process Control*, 24, 223-233.

93. Mirjalili, S. (2019). Genetic algorithm. In *Evolutionary Algorithms and Neural Networks* (pp. 43-55). Springer, Cham.
94. Li, G., Liu, B., Qin, S. J., & Zhou, D. (2011). Quality relevant data-driven modeling and monitoring of multivariate dynamic processes: The dynamic T-PLS approach. *IEEE Transactions on Neural Networks*, 22, 2262-2271.
95. Ni, W., Tan, S. K., Ng, W. J., & Brown, S. D. (2012). Localized, adaptive recursive partial least squares regression for dynamic system modeling. *Industrial & Engineering Chemistry Research*, 51, 8025-8039.
96. Ge, Z., Gao, F., & Song, Z. (2011). Mixture probabilistic PCR model for soft sensing of multimode processes. *Chemometrics and Intelligent Laboratory Systems*, 105, 91-105.
97. Li, G., Qin, S. J., & Zhou, D. (2014). A new method of dynamic latent-variable modeling for process monitoring. *IEEE Transactions on Industrial Electronics*, 61, 6438-6445.
98. Qin, S. (1997). Neural networks for intelligent sensors and control-practical issues and some solutions. In *Neural Systems for Control* (pp. 213-234).
99. Slišković, D., Grbić, R., & Hocenski, Ž. (2011). Methods for plant data-based process modeling in soft-sensor development. *Automatika*, 52(4), 306-318.
100. Perrone, M. (1992). When networks disagree: Ensemble methods for hybrid neural networks. In *Neural Networks for Speech and Image Processing* (pp. 126-142).
101. Bakshi, B. (1998). Multiscale PCA with application to multivariate statistical process monitoring. *AIChE Journal*, 44(7), 1596-1610.
102. Sumiea, E. H., Abdulkadir, S. J., Alhussian, H. S., Al-Selwi, S. M., Alqushaibi, A., Ragab, M. G., & Fati, S. M. (2024). Deep deterministic policy gradient algorithm: A systematic review. *Heliyon*.
103. Hu, Z., Xiao, L., Guan, J., Yi, W., & Yin, H. (2023). Intercept guidance of maneuvering targets with deep reinforcement learning. *International Journal of Aerospace Engineering*, 2023(1), 7924190.
104. Liu, S. (2024, February). An evaluation of DDPG, TD3, SAC, and PPO: deep reinforcement learning algorithms for controlling continuous system. In *2023*

International Conference on Data Science, Advanced Algorithm and Intelligent Computing (DAI 2023) (pp. 15-24). Atlantis Press.

105. Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018, July). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International conference on machine learning (pp. 1861-1870). Pmlr.

106. Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., ... & Levine, S. (2018). Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905.

107. Huang, Z., Zhong, W., Li, D., & Lu, H. (2023). Delay constrained SFC orchestration for edge intelligence-enabled IIoT: A DRL approach. *Journal of Network and Systems Management*, 31(3), 53.

108. Meleiro, L. A. C., & Maciel, R. F. (2000). A self-tuning adaptive control applied to an industrial large scale ethanol production. *Computers & Chemical Engineering*, 24(2-7), 925-930.

109. Negri, E., Fumagalli, L., & Macchi, M. (2017). A review of the roles of digital twin in CPS-based production systems. *Procedia Manufacturing*, 11, 939-948.

110. Nikolakis, N. (2019). The digital twin implementation for linking the virtual representation of human-based production tasks to their physical counterpart in the factory-floor. *International Journal of Computer Integrated Manufacturing*, 32(1), 1-12.

111. Liu, M. (2021). Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems*, 58, 346-361.

112. Tao, F. (2018). Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics*, 15(4), 2405-2415.

113. Qi, Q., & Tao, F. (2018). Digital twin and big data towards smart manufacturing and industry 4.0: 360 degree comparison. *IEEE Access*, 6, 3585-3593.

114. Tao, F., et al. (2018). Digital twin and its potential application exploration. *Computer Integrated Manufacturing Systems*, 24(1), 1-18.

115. Schluse, M., & Rossmann, J. (2016). From simulation to experimentable digital twins: Simulation-based development and operation of complex technical systems. In

Proceedings of IEEE International Symposium on Systems Engineering (pp. 1-6). Edinburgh, UK.

116. Schroeder, G., et al. (2016). Visualising the digital twin using web services and augmented reality. In Proceedings of IEEE 14th International Conference on Industrial Informatics (pp. 522-527). Poitiers, France.

117. Schroeder, G., Steinmetz, C., Pereira, C. E., & Espindola, D. B. (2016). Digital twin data modeling with automation ML and a communication methodology for data exchange. IFAC PapersOnLine, 49(30), 12-17.

118. Yun, S., Park, J. H., & Kim, W. T. (2017). Data-centric middleware based digital twin platform for dependable cyber-physical systems. In Proceedings of the 9th International Conference on Ubiquitous Future Networks (pp. 922-926). Milan, Italy.

119. Moreno, G., Velez, G., Ardanza, A., Barandiaran, I., Rodriguez de Infante, A., & Chopitea, R. (2016). Virtualisation process of a sheet metal punching machine within the industry 4.0 vision. International Journal of Interactive Design and Manufacturing, 11(2), 1-9.

120. Cai, Y., Starly, B., Cohen, P., & Lee, Y. S. (2017). Sensor data and information fusion to construct digital-twins virtual machine tools for cyber physical manufacturing. Procedia Manufacturing, 10, 1031-1042.

121. Rosen, R., Von Wichert, G., Lo, G., & Bettenhausen, K. D. (2015). About the importance of autonomy and digital twins for the future of manufacturing. IFAC PapersOnLine, 48(3), 567-572.

122. Jones, D. (2020). Characterising the digital twin: A systematic literature review. CIRP Journal of Manufacturing Science and Technology, 29, 36-52.

123. Illa, N., Sin, T. C., Fathullah, G. M., & Rosmaini, A. (2018). Mathematical modeling of quality and productivity in industries: A review. AIP Conference Proceedings, 2030(1), Article 020126.

124. Zaiets, N., Vlasenko, L., Lutska, N., & Shtepa, V. (2022). Resource efficiency forecasting neural network model for the sugar plant diffusion station. In Automation 2022: New solutions and technologies for automation, robotics and measurement techniques (pp. 151–161). Springer.

125. Gresova, E., & Svetlík, J. (2021). Mathematical modeling of the manufacturing sector's dominant part as a base for automation. *Applied Sciences*, 11(7), Article 3295.
126. Korobiichuk, I., Ladanyuk, A., Vlasenko, L., & Zaiets, N. (2018). Modern development technologies and investigation of food production technological complex automated systems. In *Proceedings of the 2nd International Conference on Mechatronics Systems and Control Engineering (ICMSCE 2018)* (pp. 52–56). Association for Computing Machinery.
127. Gomez-Perez, A. (2004). Ontology evaluation. In S. Staab & R. Studer (Eds.), *Handbook on ontologies* (pp. 251–274). Springer.
128. Lovrencic, S., & Cubrilo, M. (2008). Ontology evaluation – Comprising verification and validation. In *Proceedings of the Central European Conference on Information and Intelligent Systems (CECIIS 2008)* (pp. 24–26). University of Zagreb.
129. Parsia, B., Matentzoglou, N., Goncalves, R. S., Glimm, B., & Steigmiller, A. (2017). The OWL reasoner evaluation (ORE) 2015 competition report. *Journal of Automated Reasoning*, 59(4), 455–482.
130. Martin, D., Köhl, N., & Satzger, G. (2021). Virtual sensors. *Business & Information Systems Engineering*, 63(3), 315–323.
131. Feng, S. (2014). Dissolved carbon dioxide driven repeated batch fermentation (Doctoral dissertation, University of Saskatchewan).
132. Kano, M., & Fujiwara, K. (2013). Virtual sensing technology in process industries: Trends and challenges revealed by recent industrial applications. *Journal of Chemical Engineering of Japan*, 46(1), 1-17.
133. Martin, D., Köhl, N., & Satzger, G. (2021). Virtual sensors. *Business & Information Systems Engineering*, 63, 315-323.
134. Cristaldi, L., Ferrero, A., Macchi, M., Mehrafshan, A., & Arpaia, P. (2020). Virtual sensors: A tool to improve reliability. In *2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT* (pp. 142-145). IEEE.
135. Goodwin, G. C. (1999). Evaluating the performance of virtual sensors. In *1999 Information, Decision and Control. Data and Information Fusion Symposium, Signal*

Processing and Communications Symposium and Decision and Control Symposium. Proceedings (pp. 5-12). IEEE.

136. Li, H., Yu, D., & Braun, J. E. (2011). A review of virtual sensing technology and application in building systems. *HVAC&R Research*, 17(5), 619-645.

137. Zhang, M. Z., Wang, L. M., & Xiong, S. M. (2020). Using machine learning methods to provision virtual sensors in sensor-cloud. *Sensors*, 20(7), 1836.

138. Hirsenkorn, N., Hanke, T., Rauch, A., Dehlink, B., Rasshofer, R., & Biebl, E. (2016). Virtual sensor models for real-time applications. *Advances in Radio Science*, 14, 31-37.

139. Kullaa, J. (2017). Development of virtual sensors to increase the sensitivity to damage. *Procedia Engineering*, 199, 1937-1942.

140. Liu, L., Kuo, S. M., & Zhou, M. (2009). Virtual sensing techniques and their applications. In 2009 International Conference on Networking, Sensing and Control (pp. 31-36). IEEE.

141. Dimitrov, N., & Göçmen, T. (2022). Virtual sensors for wind turbines with machine learning-based time series models. *Wind Energy*, 25(9), 1626-1645.

142. Stavropoulos, G., Violos, J., Tsanakas, S., & Leivadreas, A. (2023). Enabling artificial intelligent virtual sensors in an IoT environment. *Sensors*, 23(3), 1328.

143. Wei, C., Chen, J., Song, Z., & Chen, C. I. (2018). Development of self-learning kernel regression models for virtual sensors on nonlinear processes. *IEEE Transactions on Automation Science and Engineering*, 16(1), 286-297.

144. Kim, W., & Braun, J. E. (2020). Development, implementation, and evaluation of a fault detection and diagnostics system based on integrated virtual sensors and fault impact models. *Energy and Buildings*, 228, 110368.

145. Boyes, H., Hallaq, B., Cunningham, J., & Watson, T. (2018). The industrial internet of things (IIoT): An analysis framework. *Computers in Industry*, 101, 1–12.

146. Tavana, M., Hajipour, V., & Oveisi, S. (2020). IoT-based enterprise resource planning: Challenges, open issues, applications, architecture, and future research directions. *Internet of Things*, 11, Article 100262.

147. Ungurean, I., & Gaitan, N. C. (2020). A software architecture for the Industrial Internet of Things—A conceptual model. *Sensors*, 20(19), Article 5603.
148. Da Silva, A. F., & [others]. (2016). A cloud-based architecture for the internet of things targeting industrial devices remote monitoring and control. *IFAC-PapersOnLine*, 49(30), 108–113.
149. Wood, R. K., & Berry, M. W. (1973). Terminal composition control of a binary distillation column. *Chemical Engineering Science*, 28(9), 1707-1717.
150. Tukade, T. M., & Banakar, R. (2018). Data transfer protocols in IoT—An overview. *International Journal of Pure and Applied Mathematics*, 118(16), 121–138.
151. Mudaliar, M. D., & Sivakumar, N. (2020). IoT based real time energy monitoring system using Raspberry Pi. *Internet of Things*, 12, Article 100292.
152. Chooruang, K., & Meekul, K. (2018). Design of an IoT energy monitoring system. In 2018 16th International Conference on ICT and Knowledge Engineering (ICT&KE) (pp. 1–4). IEEE.
153. Gangadhar, S., & [others]. (2020). The real time environmental time series data analysis using Influx DB. *International Journal of Advanced Scientific Innovation*, 1(1).
154. Naqvi, S. N. Z., Yfantidou, S., & Zimányi, E. (2017). Time series databases and InfluxDB [Unpublished manuscript]. Université Libre de Bruxelles.
155. Сбоев, Р. Ю., & Могилевич, Д. І. (2023). Аналіз факторів, що впливають на функціональну стійкість обладнання електронних комунікаційних систем. *Вчені записки*, Article 5202370.
156. Pereira, T., Barreto, L., & Amaral, A. (2017). Network and information security challenges within Industry 4.0 paradigm. *Procedia Manufacturing*, 13, 1253–1260.

ДОДАТКИ

ДОДАТОК А. Код програми для аналізу даних дистиляції

```
# Імпорт бібліотек
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller

# Завантаження часового ряду з CSV-файлу
df = pd.read_csv('DistillationColumnDataset.csv')
df

# Побудова графіків цільових змінних
plt.figure(figsize=(12, 6))
plt.subplot(2,1,1)
plt.plot(df['Time'], df['MoleFractionTX'])
plt.title('MoleFractionTX(Time)')
plt.grid(True)
plt.subplot(2,1,2)
plt.plot(df['Time'], df['MoleFractionHX'])
plt.title('MoleFractionHX(Time)')
plt.grid(True)
plt.show()

# Побудова графіків
for column in df.columns[1:-2]:
    plt.figure(figsize=(14, 6))
    plt.plot(df['Time'], df[column], marker='o', linestyle='-',
label=column)
    plt.xlabel('Time')
    plt.ylabel(column)
    plt.title(f'Графік {column} відносно Time')
    plt.legend()
    plt.grid(True)
    plt.show()

# Побудова графіків автокореляції
from statsmodels.graphics.tsaplots import plot_acf

plot_acf(df['MoleFractionTX'], lags=950)
```

```

plt.show()
plot_acf(df['MoleFractionHX'], lags=950)
plt.show()

# Тест Дікі-Фуллера для перевірки стаціонарності часового ряду
import pandas as pd
from statsmodels.tsa.stattools import adfuller
from prettytable import PrettyTable

# Функція для виконання тесту Діккі-Фуллера та виведення результатів у
таблиці
def adf_test(series, column_name, table):
    if series.nunique() == 1: # Перевірка на константність
        table.add_row([column_name, "Стала", "-", "-", "-", "Не
тестується"])
    else:
        result = adfuller(series.dropna()) # Тест Діккі-Фуллера
        adf_stat = result[0]
        p_value = result[1]
        conclusion = "Стаціонарна" if p_value < 0.05 else
"Нестационарна"
        table.add_row([column_name, round(adf_stat, 4), round(p_value,
4), result[4]['1%'], result[4]['5%'], conclusion])

# Ініціалізація таблиці для виведення результатів
table = PrettyTable()
table.field_names = ["Змінна", "ADF статистика", "p-значення", "Крит.
рівень (1%)", "Крит. рівень (5%)", "Висновок"]

# Перевірка стаціонарності кожного стовпця в DataFrame
for column in df.columns:
    adf_test(df[column], column, table)

# Виведення таблиці
print(table)

# Аналіз кореляції
import seaborn as sns

fig, ax = plt.subplots(figsize = (10,10))
sns.heatmap(abs(df.corr()), vmin=0, vmax=1)

```

```

plt.show()

# Видалення Sensor 9 - 10
df=df.drop(['Time', 'Sensor9', 'Sensor10', 'Sensor12', 'Sensor13',
'Sensor14', 'MoleFractionHX'],axis=1)
df

# Оновлена кореляція
fig, ax = plt.subplots(figsize = (10,10))
sns.heatmap(abs(df.corr()),vmin=0, vmax=1)
plt.show()

```

ДОДАТОК Б. Код програми для статистичного аналізу та класифікації

Статистичний аналіз даних роботи дистиляційного відділення

1) Описова статистика даних

```

import pandas as pd
import matplotlib.pyplot as plt

# Зчитування CSV файлу з даними в DataFrame
stable_data = pd.read_csv('bio_disel_12h_Stable.csv')
unstable_data = pd.read_csv('bio_disel_12h_Nonstable.csv')

display(stable_data)
display(unstable_data)

# Побудова графіків для кожної колонки
for column in stable_data.columns[1:]:
    plt.figure(figsize=(14, 6))

    # Побудова графіка для stable_data
    plt.plot(stable_data['Time'], stable_data[column], marker='o',
linestyle='-', label=f'Stable {column}', color='blue')

    # Побудова графіка для unstable_data
    if column in unstable_data.columns: # Перевірка, чи колонка є в
unstable_data
        plt.plot(unstable_data['Time'], unstable_data[column],
marker='o', linestyle='--', label=f'Unstable {column}', color='red')

```

```

# Налаштування осей та легенди
plt.xlabel('Time')
plt.ylabel(column)
plt.title(f'Графік {column} відносно Time')
plt.legend()
plt.grid(True)
plt.show()

# Відображення даних у форматованій таблиці без стовпця "count"
def display_formatted_stats_readable(stable_stats, unstable_stats):
    # Видалення першого стовпця "count"
    stable_stats = stable_stats.drop(index="count")
    unstable_stats = unstable_stats.drop(index="count")

    # Форматування та стилізація таблиці
    print("Описова статистика для стабільного процесу:")
    display(
        stable_stats.style.set_properties(**{'background-color':
'#f9f9f9', 'border': '1px solid black'})
        .highlight_between(subset=pd.IndexSlice[:, :], color="#d4e1f5",
axis=0)
    )

    print("\nОписова статистика для нестабільного процесу:")
    display(
        unstable_stats.style.set_properties(**{'background-color':
'#f9f9f9', 'border': '1px solid black'})
        .highlight_between(subset=pd.IndexSlice[:, :], color="#d4e1f5",
axis=0)
    )

    # Розрахунок описової статистики для кожного набору даних
    stable_stats = stable_data.describe()
    unstable_stats = unstable_data.describe()

    # Виклик функції
    display_formatted_stats_readable(stable_stats, unstable_stats)

```

2) Порівняльні гістограми

```

# Побудова гістограм
for column in stable_data.columns[1:]:

```

```

plt.figure(figsize=(14, 6))

# Гістограма для stable_data
plt.hist(stable_data[column], bins=20, alpha=0.7, label='Stable',
color='blue')

# Гістограма для unstable_data
if column in unstable_data.columns:
    plt.hist(unstable_data[column], bins=20, alpha=0.7,
label='Unstable', color='red')

plt.xlabel(column)
plt.ylabel('Frequency')
plt.title(f'Розподіл {column}')
plt.legend()
plt.grid(True)
plt.show()

```

3) Перевірка нормальності розподілу даних для кожної змінної у двох підмножинах даних за допомогою тесту Шапіро-Вілка

```

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import shapiro, probplot

# Перевірка нормальності та візуалізація
for column in stable_data.columns[1:]:
    # Статистичний тест
    stat_stable, p_stable = shapiro(stable_data[column])
    stat_unstable, p_unstable = shapiro(unstable_data[column])

    # Результати тесту
    print(f"Змінна: {column}")
    print(f"  Stable: p-value = {p_stable}")
    print(f"  Unstable: p-value = {p_unstable}")
    print()

    # Візуалізація
    fig, axes = plt.subplots(1, 2, figsize=(12, 5))

```



```

# Гістограми
sns.histplot(stable_data[column], kde=True, ax=axes[0],
color="blue", alpha=0.6)
axes[0].set_title(f"Гістограма: Stable ({column})")
sns.histplot(unstable_data[column], kde=True, ax=axes[1],
color="orange", alpha=0.6)
axes[1].set_title(f"Гістограма: Unstable ({column})")

plt.show()

# Q-Q графіки
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
probplot(stable_data[column], dist="norm", plot=axes[0])
axes[0].set_title(f"Q-Q графік: Stable ({column})")
probplot(unstable_data[column], dist="norm", plot=axes[1])
axes[1].set_title(f"Q-Q графік: Unstable ({column})")

plt.tight_layout()
plt.show()

normality_results = []
for column in stable_data.columns[1:]:
    stable_p = shapiro(stable_data[column])[1]
    unstable_p = shapiro(unstable_data[column])[1]
    normality_results.append({
        'Variable': column,
        'Stable p-value': stable_p,
        'Unstable p-value': unstable_p
    })

normality_df = pd.DataFrame(normality_results)

# Функція для стилізації DataFrame
def highlight_normality(val):
    color = 'green' if val > 0.05 else 'red' # Вибір кольору: зелений
для нормального, червоний для ненормального розподілу
    return f'background-color: {color}'

# Стилiзація p-value
normality_df_styled = normality_df.style.applymap(highlight_normality,
subset=['Stable p-value', 'Unstable p-value'])

```

```
# Виведення таблиці
display(normality_df_styled)
```

4) Матриці кореляції

```
import seaborn as sns
import matplotlib.pyplot as plt

# Матриці кореляції

print("stable_data")
fig, ax = plt.subplots(figsize = (35,35))
sns.heatmap(abs(stable_data.corr()), vmin=0, vmax=1, annot=True,
cmap='coolwarm', ax=ax)
plt.show()

print("unstable_data")
fig, ax = plt.subplots(figsize = (35,35))
sns.heatmap(abs(unstable_data.corr()), vmin=0, vmax=1, annot=True,
cmap='coolwarm', ax=ax)
plt.show()
```

5) Boxplots та violin plots для порівняння розподілів між групами

```
# Boxplots для порівняння розподілів (по два в ряд)
columns = list(stable_data.columns)
for i in range(0, len(columns), 2):
    plt.figure(figsize=(12, 4))

    # Перший графік
    plt.subplot(1, 2, 1)
    data_to_plot = [stable_data[columns[i]], unstable_data[columns[i]]]
    plt.boxplot(data_to_plot, tick_labels=['Стабільні', 'Нестабільні'])
    plt.title(f'Порівняння розподілів {columns[i]}')
    plt.ylabel('Значення')
    plt.grid(True)

    # Другий графік (якщо є)
    if i + 1 < len(columns):
        plt.subplot(1, 2, 2)
        data_to_plot = [stable_data[columns[i+1]],
unstable_data[columns[i+1]]]
        plt.boxplot(data_to_plot, tick_labels=['Стабільні',
'Нестабільні'])
```

```

plt.title(f'Порівняння розподілів {columns[i+1]}')
plt.ylabel('Значення')
plt.grid(True)

plt.tight_layout()
plt.show()

# Violin plots для порівняння розподілів
for i in range(0, len(columns), 2):
    plt.figure(figsize=(12, 4))

    # Перший графік
    plt.subplot(1, 2, 1)
    combined_data = pd.DataFrame({
        'value': pd.concat([stable_data[columns[i]],
unstable_data[columns[i]]]),
        'condition': ['Стабільні']*len(stable_data) +
['Нестабільні']*len(unstable_data)
    })
    sns.violinplot(data=combined_data, x='condition', y='value')
    plt.title(f'Розподіл {columns[i]} за умовами')
    plt.grid(True)

    # Другий графік (якщо є)
    if i + 1 < len(columns):
        plt.subplot(1, 2, 2)
        combined_data = pd.DataFrame({
            'value': pd.concat([stable_data[columns[i+1]],
unstable_data[columns[i+1]]]),
            'condition': ['Стабільні']*len(stable_data) +
['Нестабільні']*len(unstable_data)
        })
        sns.violinplot(data=combined_data, x='condition', y='value')
        plt.title(f'Розподіл {columns[i+1]} за умовами')
        plt.grid(True)

plt.tight_layout()
plt.show()

```

6) Дисперсійний аналіз ANOVA

```
import matplotlib.pyplot as plt
```

```

import numpy as np
from scipy import stats

# Збір результатів ANOVA
anova_results = {
    "Variable": [],
    "F-statistic": [],
    "p-value": []
}

for column in stable_data.columns:
    f_stat, p_val = stats.f_oneway(stable_data[column],
unstable_data[column])
    anova_results["Variable"].append(column)
    anova_results["F-statistic"].append(f_stat)
    anova_results["p-value"].append(p_val)

# Перетворення результатів у DataFrame для зручності
anova_df = pd.DataFrame(anova_results)

# Визначення критичного значення F
alpha = 0.05 # Рівень значущості
number_of_groups = 2 # Дві групи
total_number_of_observations = len(stable_data) + len(unstable_data) #
Загальна кількість спостережень

df1 = number_of_groups - 1
df2 = total_number_of_observations - number_of_groups

# Критичне значення F для рівня значущості 0.05
critical_value = stats.f.ppf(1 - alpha, df1, df2)

print(f"Критичне значення F: {critical_value}")

# Побудова стовпчикової діаграми для F-статистики (з логарифмічною
шкалою)
plt.figure(figsize=(12, 6))
plt.bar(anova_df["Variable"], anova_df["F-statistic"], color="skyblue",
alpha=0.8)
plt.yscale("log") # Логарифмічна шкала

```

```

plt.axhline(y=critical_value, color='red', linestyle='--',
label="Критична межа (умовно)") # Додаткова лінія для орієнтиру
plt.xticks(rotation=45, ha='right')
plt.title("F-статистики для змінних (ANOVA) [логарифмічна шкала]")
plt.ylabel("F-статистика (логарифмічна шкала)")
plt.xlabel("Змінні")
plt.legend()
plt.tight_layout()
plt.show()

# Побудова стовпчикової діаграми для р-значень (логарифмічна шкала)
plt.figure(figsize=(12, 6))
plt.bar(anova_df["Variable"], anova_df["p-value"], color="orange",
alpha=0.8)
plt.yscale("log") # Логарифмічна шкала
plt.axhline(y=0.05, color='green', linestyle='--', label="Рівень
значущості (0.05)") # Лінія рівня значущості
plt.xticks(rotation=45, ha='right')
plt.title("р-значення для змінних (ANOVA) [логарифмічна шкала]")
plt.ylabel("р-значення (логарифмічна шкала)")
plt.xlabel("Змінні")
plt.legend()
plt.tight_layout()
plt.show()

```

7) Kernel Density Estimate (перевірка на мультимодальність)

```

from scipy.stats import gaussian_kde

for column in stable_data.columns[1:]:
    density_stable = gaussian_kde(stable_data[column])
    density_unstable = gaussian_kde(unstable_data[column])
    x_vals = np.linspace(min(stable_data[column].min(),
unstable_data[column].min()),
                        max(stable_data[column].max(),
unstable_data[column].max()), 100)

    plt.figure(figsize=(12, 6))
    plt.plot(x_vals, density_stable(x_vals), label='Stable',
color='blue')
    plt.plot(x_vals, density_unstable(x_vals), label='Unstable',
color='red')
    plt.title(f'Kernel Density Estimate: {column}')

```

```
plt.xlabel(column)
plt.ylabel('Density')
plt.legend()
plt.grid()
plt.show()
```

8) Класифікація стабільності процесу з використанням GMM

```
import pandas as pd
import numpy as np
from sklearn.mixture import GaussianMixture
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

def analyze_stability(stable_data, unstable_data):
    features = stable_data.columns[1:].tolist()
    print("Використовуються наступні колонки:", features)

    X_stable = stable_data[features].values
    X_unstable = unstable_data[features].values

    scaler = StandardScaler()
    X_combined = np.vstack([X_stable, X_unstable])
    X_combined_scaled = scaler.fit_transform(X_combined)

    y_true = np.hstack([np.ones(len(X_stable)),
np.zeros(len(X_unstable))])

    gmm = GaussianMixture(n_components=2, covariance_type='full',
random_state=42)
    gmm.fit(X_combined_scaled)

    # Витягування параметрів моделі для кожного класу
    means = gmm.means_
    covariances = gmm.covariances_
    weights = gmm.weights_

    # Визначення, який компонент відповідає стабільному стану
    # Порівняння передбачення з істинними мітками
    y_pred = gmm.predict(X_combined_scaled)
```

```

if np.mean(y_pred[:len(X_stable)]) < 0.5:
    # Якщо перший компонент частіше передбачає нестабільний стан,
мінємо місцями
    means = means[::-1]
    covariances = covariances[::-1]
    weights = weights[::-1]

print("Середні значення для кожного класу:")
print(f"Стабільний клас: {means[0]}")
print(f"Нестабільний клас: {means[1]}")

print("\nКоваріаційні матриці для кожного класу:")
print(f"Стабільний клас:\n{covariances[0]}")
print(f"Нестабільний клас:\n{covariances[1]}")

print("\nВаги для кожного класу:")
print(f"Стабільний клас: {weights[0]}")
print(f"Нестабільний клас: {weights[1]}")

y_pred = gmm.predict(X_combined_scaled)

if np.mean(y_pred[:len(X_stable)]) < 0.5:
    y_pred = 1 - y_pred

conf_mat = confusion_matrix(y_true, y_pred)
class_report = classification_report(y_true, y_pred)

# Візуалізація confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Нестабільний', 'Стабільний'],
            yticklabels=['Нестабільний', 'Стабільний'])
plt.title('Confusion Matrix')
plt.ylabel('Справжні значення')
plt.xlabel('Предбачені значення')
plt.show()

# Візуалізація розподілу компонент для кожної пари ознак
num_features = X_combined_scaled.shape[1]
for i in range(0, num_features, 2):
    if i+1 >= num_features:

```

```

        break # Якщо кількість ознак непарна, останню ознаку не
ВИВОДИМО

plt.figure(figsize=(12, 6))

# Створення сітки точок для візуалізації
x = np.linspace(X_combined_scaled[:, i].min() - 0.5,
X_combined_scaled[:, i].max() + 0.5, 100)
y = np.linspace(X_combined_scaled[:, i+1].min() - 0.5,
X_combined_scaled[:, i+1].max() + 0.5, 100)
X, Y = np.meshgrid(x, y)
XX = np.array([X.ravel(), Y.ravel()]).T

# Додавання нулів для інших ознак, якщо вони є
if X_combined_scaled.shape[1] > 2:
    XX = np.hstack([XX, np.zeros((XX.shape[0],
X_combined_scaled.shape[1]-2))])

# Обчислення від'ємного логарифм правдоподібності для кожної
точки

Z = -gmm.score_samples(XX)
Z = Z.reshape(X.shape)

# Візуалізація контурів розподілу та точки даних
contour = plt.contourf(X, Y, Z, levels=20, cmap='viridis',
alpha=0.5)
plt.colorbar(contour, label='Від\'ємний логарифм
правдоподібності\n(низькі значення = висока правдоподібність)')

# Візуалізація точок даних
plt.scatter(X_combined_scaled[:len(X_stable), i],
X_combined_scaled[:len(X_stable), i+1],
c='green', label='Стабільний', alpha=0.6)
plt.scatter(X_combined_scaled[len(X_stable):, i],
X_combined_scaled[len(X_stable):, i+1],
c='red', label='Нестабільний', alpha=0.6)

# Налаштування графіка
plt.legend()
plt.title(f'Розподіл даних у просторі ознак\n({features[i]} vs
{features[i+1]})')

```



```

plt.xlabel(features[i])
plt.ylabel(features[i+1])
plt.show()

return gmm, conf_mat, class_report, scaler, features

def predict_stability(gmm, new_data, scaler, training_features,
window_size=10):
    """
    Покроковий аналіз стабільності процесу.

    Parameters:
    gmm: Навчена GMM модель
    new_data: Нові дані для аналізу
    scaler: Навчений StandardScaler
    training_features: Список колонок з тренувальних даних
    window_size: Розмір вікна для аналізу стабільності (за
замовчуванням 10)

    Returns:
    DataFrame з результатами аналізу
    """
    # Підготовка даних
    if set(new_data.columns[1:]) != set(training_features):
        missing_features = set(training_features) -
set(new_data.columns[1:])
        for feature in missing_features:
            new_data[feature] = 0

    X = new_data[training_features].values
    X_scaled = scaler.transform(X)

    # Отримання ймовірності належності до кожного класу
    probabilities = gmm.predict_proba(X_scaled)

    # Створення DataFrame для результатів
    results = pd.DataFrame()
    results['timestamp'] = new_data.iloc[:, 0] # Перша колонка як
часова мітка
    results['probability_stable'] = probabilities[:, 1] # Ймовірність
належності до стабільного класу

```

```

        results['probability_unstable'] = probabilities[:, 0] #
Ймовірність належності до нестабільного класу

# Аналіз стабільності у вікні
stability_metrics = []

for i in range(len(new_data)):
    if i < window_size - 1:
        # Для початкових спостережень
        window_probabilities = probabilities[:i+1]
        window_conf_matrix = None
    else:
        # Для повного вікна
        window_probabilities = probabilities[i-window_size+1:i+1]

        # Розрахунок confusion matrix для вікна
        if i > 0:
            window_conf_matrix = confusion_matrix(
                window_probabilities[:-1, 1] > 0.5, # Попередні
                window_probabilities[1:, 1] > 0.5,   # Поточні
                labels=[0, 1]
            )
        else:
            window_conf_matrix = None

    stability_metrics.append({
        'window_size': len(window_probabilities),
        'stable_ratio': np.mean(window_probabilities[:, 1]),
        'confusion_matrix': window_conf_matrix
    })

# Додавання метрики до результатів
metrics_df = pd.DataFrame(stability_metrics)
results = pd.concat([results, metrics_df], axis=1)

return results

def plot_stability_over_time(results):
    """
    Візуалізація стабільності у часі.

```

```

Parameters:
    results: DataFrame, що містить колонки 'timestamp' та
'stable_ratio'.
"""

# Перевірка, чи потрібні колонки є в DataFrame
if 'timestamp' not in results.columns or 'stable_ratio' not in
results.columns:
    raise ValueError("DataFrame повинен містити колонки 'timestamp'
та 'stable_ratio'")

# Створення графіка
plt.figure(figsize=(12, 6))
plt.plot(results['timestamp'], results['stable_ratio'], marker='o',
label='Стабільність', color='blue')

# Налаштування графіка
plt.axhline(y=0.5, color='red', linestyle='--', label='Попір
стабільності (0.5)')
plt.title('Стабільність процесу у часі', fontsize=16)
plt.xlabel('Час', fontsize=14)
plt.ylabel('Коефіцієнт стабільності (stable_ratio)', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend(fontsize=12)
plt.xticks(rotation=45)

# Відображення графіка
plt.tight_layout()
plt.show()

# Навчання моделі
stable_data = pd.read_csv('bio_disel_12h_Stable.csv')
unstable_data = pd.read_csv('bio_disel_12h_Nonstable.csv')
gmm, conf_mat, report, scaler, training_features =
analyze_stability(stable_data, unstable_data)

# Аналіз нових даних
new_data = pd.read_csv('new_data4.csv')
results = predict_stability(gmm, new_data, scaler, training_features)

# Виведення результатів
print("\nРезультати аналізу стабільності:")

```

```

for i, row in results.iterrows():
    print(f"\nКрок {i + 1}:")
    print(f"Часова мітка: {row['timestamp']}")
    print(f"Ймовірність стабільності: {row['probability_stable']:.2f}")
    print(f"Ймовірність нестабільності:
{row['probability_unstable']:.2f}")
    print(f"Співвідношення стабільності у вікні:
{row['stable_ratio']:.2f}")

    if row['confusion_matrix'] is not None:
        print("Confusion Matrix для вікна:")
        print(row['confusion_matrix'])

# Виклик функції для візуалізації
plot_stability_over_time(results)

```

ДОДАТОК В. Код програми для прогнозування концентрації за допомогою NARX-моделей

1) Побудова моделі

```

# Імпорт бібліотек
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
from tensorflow import keras
from keras import layers
import tensorflow as tf
from sklearn.metrics import mean_squared_error, r2_score

# Зчитування CSV файлу з даними в DataFrame
data = pd.read_csv('DistillationDataset1.csv')

# Вибір тільки необхідних стовпців
data = data.drop(['Time', 'Sensor9', 'Sensor10', 'Sensor12',
'Sensor13', 'Sensor14', 'MoleFractionHX'], axis=1)

# Обрання цільової змінної (Sensor11)
target_column = 'MoleFractionTX'

# Масштабування даних

```

```

scaler = MinMaxScaler()
scaled = scaler.fit_transform(data)
scaled_data = pd.DataFrame(scaled, columns=data.columns)

# Підготовка даних для навчання
def create_sequences(input_data, target_column, sequence_length=1):
    inputs, targets = [], []
    for i in range(len(input_data) - sequence_length):
        inputs.append(input_data[i:i + sequence_length, :])
        targets.append(input_data[i + sequence_length, target_column])
    return np.array(inputs), np.array(targets)

# Створення послідовностей
n_delay = 11
x, y = create_sequences(scaled_data.to_numpy(),
scaled_data.columns.get_loc(target_column), n_delay)

# Розподіл даних на навчальний та тестовий набори
split_ratio = 0.8
split = int(len(x) * split_ratio)

x_train, x_valid, = x[:split], x[split:]
y_train, y_valid, = y[:split], y[split:]

# Побудова нейронної мережі
narx_model = keras.Sequential([
    layers.GaussianNoise(0.01, input_shape=(x.shape[1], x.shape[2])),
    layers.Flatten(),
    layers.Dense(32, activation='sigmoid'), #, input_shape=(x.shape[1],
x.shape[2])),
    layers.Dense(1, activation='linear')
])

# Компіляція моделі
narx_model.compile(
    optimizer=tf.keras.optimizers.Adam(),
    loss='mse',
    metrics=['mae']
)

# Навчання моделі

```

```

callback = tf.keras.callbacks.EarlyStopping(monitor='loss',
patience=16)

history = narx_model.fit(x_train, y_train, epochs=100,
validation_split=0.05, verbose=1, callbacks=[callback])

# Оцінка моделі
narx_model.evaluate(x_train, y_train)
narx_model.evaluate(x_valid, y_valid)

# Прогнозування
predicted_valid = narx_model.predict(x_valid)

# Зворотнє масштабування
y_valid_actual = y_valid *
(scaler.data_max_[scaled_data.columns.get_loc(target_column)] -
scaler.data_min_[scaled_data.columns.get_loc(target_column)]) +
scaler.data_min_[scaled_data.columns.get_loc(target_column)]
predicted_valid_actual = predicted_valid.flatten() *
(scaler.data_max_[scaled_data.columns.get_loc(target_column)] -
scaler.data_min_[scaled_data.columns.get_loc(target_column)]) +
scaler.data_min_[scaled_data.columns.get_loc(target_column)]

# Візуалізація
plt.figure(figsize=(16, 8))
plt.plot(y_valid_actual, label='Фактичні дані')
plt.plot(predicted_valid_actual, label='Прогнозовані дані')
plt.title('Фактичні vs Прогнозовані')
plt.xlabel('Samples')
plt.ylabel('Value')
plt.legend()
plt.show()

# Візуалізація графіку MSE
plt.figure(figsize=(10, 6))
plt.plot(history.history['loss'], label='Навчальна вибірка')
plt.plot(history.history['val_loss'], label='Валідаційна вибірка')
plt.title('Графік MSE для навчальної та тестової вибірок')
plt.xlabel('Епохи')
plt.ylabel('MSE')
plt.legend()
plt.show()

```

```

# Оцінка якості
mse_valid = mean_squared_error(y_valid_actual, predicted_valid_actual)
r2_actual = r2_score(y_valid_actual, predicted_valid_actual)
print(f'Середньоквадратична похибка (MSE): {mse_valid:.8f}')
print(f'R2: {r2_actual:.8f}')

# Розрахунок AIC
def calculate_aic(n, mse, num_params):
    return n * np.log(mse) + 2 * num_params

n_samples = len(y_valid_actual)
num_params = narx_model.count_params()
aic = calculate_aic(n_samples, mse_valid, num_params)
print(f'Критерій Акаїке (AIC): {aic:.8f}')

```

2) Підбір гіперпараметрів NARX-моделі

```

import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
from tensorflow import keras
from keras import layers
import tensorflow as tf
from sklearn.metrics import mean_squared_error, r2_score

# Зчитування CSV файлу з даними в DataFrame
data = pd.read_csv('DistillationDataset1.csv')

# Вибір тільки необхідних стовпців
data = data.drop(['Time', 'Sensor9', 'Sensor10', 'Sensor12',
'Sensor13', 'Sensor14', 'MoleFractionHX'], axis=1)

# Обрання цільової змінної (Sensor11)
target_column = 'MoleFractionTX'

# Масштабування даних
scaler = MinMaxScaler()
scaled = scaler.fit_transform(data)
scaled_data = pd.DataFrame(scaled, columns=data.columns)

# Функція для створення послідовностей

```

```

def create_sequences(input_data, target_column, sequence_length=1):
    inputs, targets = [], []
    for i in range(len(input_data) - sequence_length):
        inputs.append(input_data[i:i + sequence_length, :])
        targets.append(input_data[i + sequence_length, target_column])
    return np.array(inputs), np.array(targets)

# Стабільні значення параметрів
stable_n_delay = 11
stable_epochs = 100
stable_validation_split = 0.05
stable_patience = 16
stable_batch_size = 32

# Параметри для перебору
n_delay_values = [5, 11, 15, 20, 25]
epochs_values = [50, 100, 150, 200, 250]
validation_split_values = [0.05, 0.1, 0.15, 0.2, 0.25]
patience_values = [8, 12, 16, 20, 24]

# Функція для навчання та оцінки моделі
def train_and_evaluate(param_name, param_values, **fixed_params):
    results = []
    for param_value in param_values:
        # Підготовка параметрів
        params = fixed_params.copy()
        params[param_name] = param_value

        # Підготовка даних
        x, y = create_sequences(scaled_data.to_numpy(),
scaled_data.columns.get_loc(target_column), params['n_delay'])
        split_ratio = 0.8
        split = int(len(x) * split_ratio)
        x_train, x_valid = x[:split], x[split:]
        y_train, y_valid = y[:split], y[split:]

        # Побудова моделі
        model = keras.Sequential([
            keras.Input(shape=(x.shape[1], x.shape[2])),
            layers.GaussianNoise(0.01),
            layers.Flatten(),

```



```

        layers.Dense(32, activation='sigmoid'),
        layers.Dense(1, activation='linear')
    ])
    model.compile(
        optimizer=tf.keras.optimizers.Adam(),
        loss='mse',
        metrics=['mae']
    )

    # Навчання моделі
    callback = tf.keras.callbacks.EarlyStopping(monitor='loss',
patience=params['patience'])
    history = model.fit(
        x_train, y_train,
        epochs=params['epochs'],
        validation_split=params['validation_split'],
        batch_size=stable_batch_size,
        verbose=0,
        callbacks=[callback]
    )

    # Оцінка
    predicted_valid = model.predict(x_valid, verbose=0)
    y_valid_actual = y_valid * (
        scaler.data_max_[scaled_data.columns.get_loc(target_column)]
] - scaler.data_min_[scaled_data.columns.get_loc(target_column)]
    ) +
scaler.data_min_[scaled_data.columns.get_loc(target_column)]
    predicted_valid_actual = predicted_valid.flatten() * (
        scaler.data_max_[scaled_data.columns.get_loc(target_column)]
] - scaler.data_min_[scaled_data.columns.get_loc(target_column)]
    ) +
scaler.data_min_[scaled_data.columns.get_loc(target_column)]

    mse_valid = mean_squared_error(y_valid_actual,
predicted_valid_actual)
    r2_actual = r2_score(y_valid_actual, predicted_valid_actual)
    num_params = model.count_params()
    aic = params['n_delay'] * np.log(mse_valid) + 2 * num_params

    # Збереження результатів

```

```

results.append({
    'param': param_name,
    'value': param_value,
    'mse': mse_valid,
    'r2': r2_actual,
    'aic': aic
})

# Візуалізація
fig, axs = plt.subplots(1, 2, figsize=(20, 6))

# Лівий графік: фактичні дані vs прогноз
axs[0].plot(y_valid_actual, label='Фактичні дані')
axs[0].plot(predicted_valid_actual, label='Прогнозовані дані')
axs[0].set_title(f'Фактичні vs Прогнозовані
({param_name}={param_value})\n'
                  f'n_delay={params["n_delay"]},
epochs={params["epochs"]}, '
                  f'validation_split={params["validation_split"]}
, patience={params["patience"]}')
axs[0].set_xlabel('Зразки')
axs[0].set_ylabel('Значення')
axs[0].legend()

# Правий графік: історія навчання
axs[1].plot(history.history['loss'], label='Навчальна вибірка')
axs[1].plot(history.history['val_loss'], label='Валідаційна
вибірка')
axs[1].set_title(f'Графік MSE під час навчання
({param_name}={param_value})\n'
                  f'n_delay={params["n_delay"]},
epochs={params["epochs"]}, '
                  f'validation_split={params["validation_split"]}
, patience={params["patience"]}')
axs[1].set_xlabel('Епохи')
axs[1].set_ylabel('MSE')
axs[1].legend()

plt.tight_layout()
plt.show()

```

```

        return pd.DataFrame(results)

# Перебір параметрів
results_n_delay = train_and_evaluate('n_delay', n_delay_values,
                                     n_delay=stable_n_delay,
                                     epochs=stable_epochs,
                                     validation_split=stable_validation
_split,

                                     patience=stable_patience)

results_epochs = train_and_evaluate('epochs', epochs_values,
                                    n_delay=stable_n_delay,
                                    epochs=stable_epochs,
                                    validation_split=stable_validation_
split,

                                    patience=stable_patience)

results_validation_split = train_and_evaluate('validation_split',
validation_split_values, n_delay=stable_n_delay, epochs=stable_epochs,
validation_split=stable_validation_split, patience=stable_patience)

results_patience = train_and_evaluate('patience', patience_values,
n_delay=stable_n_delay, epochs=stable_epochs,
validation_split=stable_validation_split, patience=stable_patience)

# Виведення всіх результатів
print("Результати для n_delay:")
print(results_n_delay)
print("\nРезультати для epochs:")
print(results_epochs)
print("\nРезультати для validation_split:")
print(results_validation_split)
print("\nРезультати для patience:")
print(results_patience)

import seaborn as sns

# Функція для створення спільних графіків
def plot_results(results_df, param_name):
    fig, axs = plt.subplots(1, 3, figsize=(20, 6))

```

```

sns.lineplot(data=results_df, x='value', y='mse', ax=axes[0],
marker='o', label='MSE')
axes[0].set_title(f'MSE залежно від {param_name}')
axes[0].set_xlabel(param_name)
axes[0].set_ylabel('MSE')

sns.lineplot(data=results_df, x='value', y='r2', ax=axes[1],
marker='o', color='green', label='R²')
axes[1].set_title(f'R² залежно від {param_name}')
axes[1].set_xlabel(param_name)
axes[1].set_ylabel('R²')

sns.lineplot(data=results_df, x='value', y='aic', ax=axes[2],
marker='o', color='red', label='AIC')
axes[2].set_title(f'AIC залежно від {param_name}')
axes[2].set_xlabel(param_name)
axes[2].set_ylabel('AIC')

plt.tight_layout()
plt.show()

# Побудова графіків для кожного параметра
plot_results(results_n_delay, 'n_delay')
plot_results(results_epochs, 'epochs')
plot_results(results_validation_split, 'validation_split')
plot_results(results_patience, 'patience')

```

3) Пошук оптимальних значень з використанням бібліотеки Optuna

```

import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, r2_score
import tensorflow as tf
from keras import layers
!pip install optuna
import optuna
from optuna.visualization import (
    plot_optimization_history,
    plot_param_importances,
    plot_slice,
    plot_parallel_coordinate,
    plot_contour,

```

```

)

# Зчитування даних
data = pd.read_csv('DistillationDataset1.csv')

# Вибір необхідних стовпців
data = data.drop(['Time', 'Sensor9', 'Sensor10', 'Sensor12',
'Sensor13', 'Sensor14', 'MoleFractionHX'], axis=1)
target_column = 'MoleFractionTX'

# Масштабування даних
scaler = MinMaxScaler()
scaled = scaler.fit_transform(data)
scaled_data = pd.DataFrame(scaled, columns=data.columns)

# Функція для створення послідовностей
def create_sequences(input_data, target_column, sequence_length=1):
    inputs, targets = [], []
    for i in range(len(input_data) - sequence_length):
        inputs.append(input_data[i:i + sequence_length, :])
        targets.append(input_data[i + sequence_length, target_column])
    return np.array(inputs), np.array(targets)

# Функція для створення та навчання моделі
def objective(trial):
    # Параметри оптимізації
    n_delay = trial.suggest_int('n_delay', 5, 25, step=5) # Вибір
затримки
    epochs = trial.suggest_int('epochs', 50, 250, step=50) # Кількість
епох
    validation_split = trial.suggest_float('validation_split', 0.05,
0.25, step=0.05) # Валідація
    patience = trial.suggest_int('patience', 8, 24, step=4) # Раннє
зупинення

    # Підготовка даних
    x, y = create_sequences(scaled_data.to_numpy(),
scaled_data.columns.get_loc(target_column), n_delay)
    split_ratio = 0.8
    split = int(len(x) * split_ratio)
    x_train, x_valid = x[:split], x[split:]

```

```

y_train, y_valid = y[:split], y[split:]

# Побудова моделі
model = tf.keras.Sequential([
    tf.keras.Input(shape=(x.shape[1], x.shape[2])),
    layers.GaussianNoise(0.01),
    layers.Flatten(),
    layers.Dense(32, activation='sigmoid'),
    layers.Dense(1, activation='linear')
])
model.compile(optimizer=tf.keras.optimizers.Adam(), loss='mse',
metrics=['mae'])

# Навчання моделі
callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=patience)
history = model.fit(
    x_train, y_train,
    epochs=epochs,
    validation_split=validation_split,
    batch_size=32,
    verbose=0,
    callbacks=[callback]
)

# Оцінка на валідаційних даних
predicted_valid = model.predict(x_valid, verbose=0)
y_valid_actual = y_valid * (
    scaler.data_max_[scaled_data.columns.get_loc(target_column)] -
scaler.data_min_[scaled_data.columns.get_loc(target_column)]
) + scaler.data_min_[scaled_data.columns.get_loc(target_column)]
predicted_valid_actual = predicted_valid.flatten() * (
    scaler.data_max_[scaled_data.columns.get_loc(target_column)] -
scaler.data_min_[scaled_data.columns.get_loc(target_column)]
) + scaler.data_min_[scaled_data.columns.get_loc(target_column)]

mse_valid = mean_squared_error(y_valid_actual,
predicted_valid_actual)

return mse_valid

```

```

# Виконання оптимізації
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=50) # Число ітерацій для пошуку
оптимальних параметрів

# Найкращі параметри
print("Найкращі параметри:", study.best_params)
print("Найкраще MSE:", study.best_value)

# Аналіз результатів
plot_optimization_history(study).show()
plot_param_importances(study).show()
plot_slice(study).show()
plot_parallel_coordinate(study).show()
plot_contour(study).show()

```

ДОДАТОК Г. Код програми для інтелектуального моніторингу динамічних процесів у дистиляційному відділенні

1) Перевірка концепції з використанням TimeSeriesSplit в NARX

Matlab

```

X = all_12h_12s(:, [2:16, 18:end]);
Y = all_12h_12s(:, 17);

X = table2array(X);
Y = table2array(Y);

X = tonndata(X, false, false);
T = tonndata(Y, false, false);

% Визначення параметрів перехресної перевірки
numFolds = 5; % Number of folds for cross-validation
dataLength = length(T); % Length of the data
foldSize = floor(dataLength / numFolds); % Size of each fold

% Ініціалізація змінних для збереження даних продуктивності
trainPerformanceFold = zeros(1, numFolds);
testPerformanceFold = zeros(1, numFolds);

% Параметри NARX-моделі

```

```

trainFcn = 'trainbr'; % Байєсова регуляризація зворотного поширення
inputDelays = 1:4;
feedbackDelays = 1:4;
hiddenLayerSize = 64;

% Цикл по кожному блоку для перехресної перевірки
for fold = 1:numFolds
    fprintf('Processing fold %d...\n', fold);

    % Define the train and test split indices
    train_end = fold * foldSize;
    test_start = train_end + 1;

    % Ensure the test set is large enough to handle delays
    if test_start > dataLength - max(inputDelays)
        fprintf('Not enough data points left for this fold.
Skipping...\n');
        continue;
    end

    test_end = min(test_start + foldSize - 1, dataLength);

    % Prepare training and test data
    X_train = X(:, 1:train_end); % Дані до train_end для навчання
    T_train = T(:, 1:train_end);

    X_test = X(:, test_start:test_end); % Наступний сегмент для
тестування
    T_test = T(:, test_start:test_end);

    % Створення нелінійної авторегресійної мережі з зовнішнім вводом
    net = narxnet(inputDelays, feedbackDelays, hiddenLayerSize, 'open',
trainFcn);
    net.trainParam.epochs = 50; % Limit to 50 epochs

    % Choose Input and Feedback Pre/Post-Processing Functions
    net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
    net.inputs{2}.processFcns = {'removeconstantrows','mapminmax'};

    % Prepare the Data for Training and Simulation

```



```

[x_train, xi_train, ai_train, t_train] = preparets(net, X_train,
{}, T_train);

[x_test, xi_test, ai_test, t_test] = preparets(net, X_test, {},
T_test);

% Налаштування розподілу даних для навчання, валідації та
тестування

net.divideFcn = 'dividerand'; % Випадковий розподіл даних
net.divideMode = 'time'; % Розподіл кожного зразка
net.divideParam.trainRatio = 75/100;
net.divideParam.valRatio = 10/100;
net.divideParam.testRatio = 15/100;

% Вибір функції оцінки
net.performFcn = 'mse'; % Середньоквадратична помилка

% Навчання мережі
[net, tr] = train(net, x_train, t_train, xi_train, ai_train);

% Test the Network on training data
y_train = net(x_train, xi_train, ai_train);
trainPerformanceFold(fold) = perform(net, t_train, y_train);

% Тестування на навчальних даних
y_train = net(x_train, xi_train, ai_train);
trainPerformanceFold(fold) = perform(net, t_train, y_train);

% Тестування на тестових даних
y_test = net(x_test, xi_test, ai_test);
testPerformanceFold(fold) = perform(net, t_test, y_test);

% Конвертація коміркових масивів у числові масиви для побудови графіків
y_train_numeric = cell2mat(y_train);
t_train_numeric = cell2mat(t_train);
y_test_numeric = cell2mat(y_test);
t_test_numeric = cell2mat(t_test);

% Виведення продуктивності для цього fold
fprintf('Fold %d -> Train Performance: %f, Test Performance: %f\n', ...
fold, trainPerformanceFold(fold), testPerformanceFold(fold));

```

```

% Збереження всіх графіків для навчання цього fold
% Графік продуктивності
figure;
plotperform(tr);
saveas(gcf, sprintf('Performance_Fold_%d.png', fold));

% Гістограма помилок (використовуючи числові масиви)
figure;
ploterrhist(y_test_numeric - t_test_numeric);
saveas(gcf, sprintf('ErrorHistogram_Fold_%d.png', fold));

% Графік регресії (використовуючи числові масиви)
figure;
plotregression(t_train_numeric, y_train_numeric, 'Train',
t_test_numeric, y_test_numeric, 'Test');
saveas(gcf, sprintf('Regression_Fold_%d.png', fold));

% Графік фактичних/прогнозованих значень для навчальних даних
figure;
plot(t_train_numeric, 'b', 'LineWidth', 1.5); % Фактичні навчальні дані
hold on;
plot(y_train_numeric, 'r--', 'LineWidth', 1.5); % Прогнозовані
навчальні дані
title('Training Response');
xlabel('Time Steps');
ylabel('Output Value');
legend('Actual', 'Predicted');
grid on;
saveas(gcf, sprintf('TrainingResponse_Fold_%d.png', fold));

% Графік фактичних/прогнозованих значень для тестових даних
figure;
plot(t_test_numeric, 'b', 'LineWidth', 1.5); % Фактичні тестові дані
hold on;
plot(y_test_numeric, 'r--', 'LineWidth', 1.5); % Прогнозовані тестові
дані
title('Testing Response');
xlabel('Time Steps');
ylabel('Output Value');
legend('Actual', 'Predicted');
grid on;

```

```

saveas(gcf, sprintf('TestingResponse_Fold_%d.png', fold));

% Стан навчання (MSE, Градієнт)
figure;
plottrainstate(tr);
saveas(gcf, sprintf('TrainState_Fold_%d.png', fold));

close all; % Закриття всіх графіків, щоб уникнути надмірної кількості
відкритих вікон
end

% Виведення середньої продуктивності по всіх валідних блоках
avgTrainPerformance = mean(trainPerformanceFold(trainPerformanceFold >
0));
avgTestPerformance = mean(testPerformanceFold(testPerformanceFold >
0));
fprintf('Average Train Performance: %f\n', avgTrainPerformance);
fprintf('Average Test Performance: %f\n', avgTestPerformance);

```

2) Моделювання в Colaboratory

2.1) Підготовка та аналіз даних

```

import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller

# Завантаження часового ряду з CSV-файлу
df = pd.read_csv('all_12h_12s.csv')
df_table = df.round(3)
df

# Побудова графіків
for column in df.columns[1:]:
    plt.figure(figsize=(14, 6))
    plt.plot(df['Time'], df[column], marker='o', linestyle='-',
label=column)
    plt.xlabel('Time')
    plt.ylabel(column)
    plt.title(f'Графік {column} відносно Time')
    plt.legend()
    plt.grid(True)

```

```

plt.show()

from statsmodels.graphics.tsaplots import plot_acf

plot_acf(df['R_T9'], lags=len(df['R_T9'])-1)
plt.show()

import pandas as pd
from statsmodels.tsa.stattools import adfuller
from prettytable import PrettyTable

# Функція для виконання тесту Діккі-Фуллера та виведення результатів у
таблиці
def adf_test(series, column_name, table):
    if series.nunique() == 1: # Перевірка на константність
        table.add_row([column_name, "Стала", "-", "-", "-", "Не
тестується"])
    else:
        result = adfuller(series.dropna()) # Тест Діккі-Фуллера
        adf_stat = result[0]
        p_value = result[1]
        conclusion = "Стационарна" if p_value < 0.05 else
"Нестационарна"
        table.add_row([column_name, round(adf_stat, 4), round(p_value,
4), result[4]['1%'], result[4]['5%'], conclusion])

# Ініціалізація таблиці для виведення результатів
table = PrettyTable()
table.field_names = ["Змінна", "ADF статистика", "p-значення", "Крит.
рівень (1%)", "Крит. рівень (5%)", "Висновок"]

# Перевірка стаціонарності кожного стовпця в DataFrame
for column in df.columns:
    adf_test(df[column], column, table)

# Виведення таблиці
print(table)
for column in df.columns:
    adf_test(df[column], column, table)

# Виведення таблиці

```

```

print(table)

# Кореляція
import seaborn as sns

fig, ax = plt.subplots(figsize = (35,35))
sns.heatmap(abs(df.corr()), vmin=0, vmax=1, annot=True,
cmap='coolwarm', ax=ax)
plt.show()

# Видалення 'R_L5'
df=df.drop(['R_L5'],axis=1)
df

# Оновлена кореляція

fig, ax = plt.subplots(figsize = (30,30))
sns.heatmap(abs(df.corr()), vmin=0, vmax=1, annot=True,
cmap='coolwarm', ax=ax)
plt.show()

# 1. Кількість стовпців
num_columns = df.shape[1]
print(f"Кількість стовпців: {num_columns}")

# 2. Функція для отримання номера позиції стовпця по його назві
def get_column_index(df, column_name):
    try:
        # Використовуємо метод get_loc, щоб отримати індекс стовпця
        index = df.columns.get_loc(column_name)
        return index
    except KeyError:
        return f"Стовпець '{column_name}' не знайдено"

column_name = 'R_T9'
target_column = get_column_index(df, column_name)
print(f"Позиція стовпця '{column_name}': {target_column}")

from sklearn.model_selection import TimeSeriesSplit
import numpy as np # Бібліотека для роботи з числовими масивами та
лінійною алгеброю

```

```

import pandas as pd # Бібліотека для обробки даних, роботи з таблицями
та CSV файлами

from sklearn.preprocessing import MinMaxScaler # Функція для
масштабування даних до діапазону [0, 1]

import matplotlib.pyplot as plt # Бібліотека для створення графіків
from tensorflow import keras # Бібліотека для побудови та навчання
нейронних мереж

from keras import layers # Модуль для роботи з шарами нейронних мереж
import tensorflow as tf # Основна бібліотека TensorFlow для машинного
навчання

from sklearn.metrics import mean_squared_error, r2_score # Імпорт
метрик для оцінки якості моделі


# Масштабування даних
scaler = MinMaxScaler() # Створення об'єкта масштабувальника даних
scaler.fit(df) # Обчислення мінімального та максимального значень для
кожного стовпця в даних
scaled = scaler.fit_transform(df) # Масштабування даних до діапазону
[0, 1]
scaled_data = pd.DataFrame(scaled, columns=df.columns) # Створення
нового DataFrame з масштабованими даними


# Підготовка даних для навчання моделі
def create_sequences(input_data, target_column, sequence_length=1,
flatten=False): # Створення послідовності даних для навчання моделі
    inputs = []
    targets = []
    data_size = len(input_data)

    for i in range(data_size-sequence_length):
        sequence = input_data[i:i+sequence_length]
        label_position = i + sequence_length
        label = input_data[label_position][target_column]
        if flatten:
            inputs.append(sequence.flatten())
        else:
            inputs.append(sequence)
        targets.append(label)

    return np.array(inputs), np.array(targets)

```

```

# Створення послідовностей з затримкою
n_delay = 3
x, y = create_sequences(scaled_data.to_numpy(), [target_column],
n_delay, flatten=True)
x.shape, y.shape
# Створення TimeSeriesSplit для 5-fold cross-validation
n_splits = 5
tscv = TimeSeriesSplit(n_splits=n_splits)

# Масиви для збереження результатів для кожного fold
fold_train, fold_valid = [], []

# Цикл по кожному розбиттю
for train_index, valid_index in tscv.split(x):
    x_train, x_valid = x[train_index], x[valid_index]
    y_train, y_valid = y[train_index], y[valid_index]

    # Зберегти результати для кожного fold (для навчання і валідації)
    fold_train.append((x_train, y_train))
    fold_valid.append((x_valid, y_valid))

# Перевірити кількість вибірок у кожному fold
for i, (train, valid) in enumerate(zip(fold_train, fold_valid)):
    print(f"Fold {i+1}:")
    print(f" Розмір навчального набору: {len(train[0])}")
    print(f" Розмір валідаційного набору: {len(valid[0])}")

```

2.2) NARX модель

```

# NARX модель
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Функція для обчислення AIC
def calculate_aic(n, mse, num_params):
    aic = n * np.log(mse) + 2 * num_params
    return aic

# Підрахунок кількості параметрів моделі
def count_model_params(model):

```

```

        return np.sum([np.prod(v.shape) for v in
model.trainable_variables])

# Створення TimeSeriesSplit для 5-fold cross-validation
n_splits = 5
tscv = TimeSeriesSplit(n_splits=n_splits)

# Массиви для зберігання метрик
mse_train_scores, mse_valid_scores = [], []
r2_train_scores, r2_valid_scores = [], []
aic_train_scores, aic_valid_scores = [], []

# Функція для побудови NARX моделі
def build_narx_model():
    model = keras.Sequential([
        layers.Dense(64, activation='tanh',
kernel_regularizer=tf.keras.regularizers.l2(0.01)),
        layers.Dropout(0.2),
        layers.Dense(32, activation='tanh',
kernel_regularizer=tf.keras.regularizers.l2(0.01)),
        layers.Dropout(0.2),
        layers.Dense(1, activation='relu')
    ])
    model.compile(optimizer=tf.keras.optimizers.Adam(), loss='mse',
metrics=['mae'])
    return model

# Цикл по кожному розбиттю
for fold, (train_index, valid_index) in enumerate(tscv.split(x)):
    print(f"Fold {fold + 1}/{n_splits}")

    # Розподіл даних
    x_train, x_valid = x[train_index], x[valid_index]
    y_train, y_valid = y[train_index], y[valid_index]

    # Створення нової NARX моделі для кожного fold-a
    narx_model = build_narx_model()

    # Callback для early stopping
    callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=10, restore_best_weights=True)

```



```

# Навчання моделі
history = narx_model.fit(x_train, y_train, epochs=100,
validation_data=(x_valid, y_valid),
                        callbacks=[callback], verbose=0)

# Прогнозування
predicted_train = narx_model.predict(x_train)
predicted_valid = narx_model.predict(x_valid)

# Обчислення MSE і R² для навчальних та тестових даних
mse_train = mean_squared_error(y_train, predicted_train)
mse_valid = mean_squared_error(y_valid, predicted_valid)
r2_train = r2_score(y_train, predicted_train)
r2_valid = r2_score(y_valid, predicted_valid)

# Обчислення AIC
num_params = count_model_params(narx_model)
aic_train = calculate_aic(len(y_train), mse_train, num_params)
aic_valid = calculate_aic(len(y_valid), mse_valid, num_params)

# Збереження метрик
mse_train_scores.append(mse_train)
mse_valid_scores.append(mse_valid)
r2_train_scores.append(r2_train)
r2_valid_scores.append(r2_valid)
aic_train_scores.append(aic_train)
aic_valid_scores.append(aic_valid)

print(f'MSE для навчальних даних (Fold {fold+1}): {mse_train:.8f}')
print(f'MSE для тестових даних (Fold {fold+1}): {mse_valid:.8f}')
print(f'R² для навчальних даних (Fold {fold+1}): {r2_train:.8f}')
print(f'R² для тестових даних (Fold {fold+1}): {r2_valid:.8f}')
print(f'AIC для навчальних даних (Fold {fold+1}): {aic_train:.8f}')
print(f'AIC для тестових даних (Fold {fold+1}): {aic_valid:.8f}')

# Побудова графіків
plt.figure(figsize=(12, 6))

# Графік для тренувальних даних
plt.subplot(1, 2, 1)

```

```

plt.plot(y_train, label='Фактичні (Навчальні дані)', color='blue')
plt.plot(predicted_train, label='Прогнозовані (Навчальні дані)',
color='red')

plt.title(f'Fold {fold+1} - Навчальні дані')
plt.xlabel('Time step')
plt.ylabel('Target value')
plt.legend()

# Графік для валідаційних даних
plt.subplot(1, 2, 2)
plt.plot(y_valid, label='Фактичні (Валідаційні дані)',
color='blue')
plt.plot(predicted_valid, label='Прогнозовані (Валідаційні дані)',
color='red')

plt.title(f'Fold {fold+1} - Валідаційні дані')
plt.xlabel('Time step')
plt.ylabel('Target value')
plt.legend()

# Відображення графіків
plt.tight_layout()
plt.show()

# Підсумкові середні значення
mean_mse_train = np.mean(mse_train_scores)
mean_mse_valid = np.mean(mse_valid_scores)
mean_r2_train = np.mean(r2_train_scores)
mean_r2_valid = np.mean(r2_valid_scores)
mean_aic_train = np.mean(aic_train_scores)
mean_aic_valid = np.mean(aic_valid_scores)

print(f'\nСередній MSE для навчальних даних: {mean_mse_train:.8f}')
print(f'Середній MSE для тестових даних: {mean_mse_valid:.8f}')
print(f'Середній R² для навчальних даних: {mean_r2_train:.8f}')
print(f'Середній R² для тестових даних: {mean_r2_valid:.8f}')
print(f'Середній AIC для навчальних даних: {mean_aic_train:.8f}')
print(f'Середній AIC для тестових даних: {mean_aic_valid:.8f}')

```

2.3) Модель DecisionTreeRegressor

```

# DecisionTreeRegressor
from sklearn.tree import DecisionTreeRegressor

```

```

from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import matplotlib.pyplot as plt

def calculate_aic(n, mse, num_params):
    aic = n * np.log(mse) + 2 * num_params
    return aic

# Створення TimeSeriesSplit для 5-fold cross-validation
n_splits = 5
tscv = TimeSeriesSplit(n_splits=n_splits)

# Ініціалізація DecisionTreeRegressor
model_RT = DecisionTreeRegressor(criterion='squared_error',
random_state=None, max_leaf_nodes=35, splitter='random')

# Масиви для збереження метрик
mse_train_scores = []
mse_valid_scores = []
r2_train_scores = []
r2_valid_scores = []
aic_train_scores = []
aic_valid_scores = []

# Цикл по кожному фолду
for fold, (train_index, valid_index) in enumerate(tscv.split(x)):
    print(f"\nFold {fold + 1}/{n_splits}")

    # Розподіл даних
    x_train, x_valid = x[train_index], x[valid_index]
    y_train, y_valid = y[train_index], y[valid_index]

    # Навчання моделі на тренувальних даних
    model_RT.fit(x_train, y_train)

    # Прогнозування для навчальних і тестових даних
    y_train_pred_RT = model_RT.predict(x_train)
    y_valid_pred_RT = model_RT.predict(x_valid)

    # Оцінка точності моделі на навчальних і тестових даних

```

```

r2_train = r2_score(y_train, y_train_pred_RT)
r2_valid = r2_score(y_valid, y_valid_pred_RT)
mse_train = mean_squared_error(y_train, y_train_pred_RT)
mse_valid = mean_squared_error(y_valid, y_valid_pred_RT)

# Розрахунок кількості параметрів моделі
# Для дерева рішень це кількість листків (terminal nodes)
num_params = model_RT.get_n_leaves()

# Розрахунок AIC
aic_train = calculate_aic(len(y_train), mse_train, num_params)
aic_valid = calculate_aic(len(y_valid), mse_valid, num_params)

# Збереження метрик
mse_train_scores.append(mse_train)
mse_valid_scores.append(mse_valid)
r2_train_scores.append(r2_train)
r2_valid_scores.append(r2_valid)
aic_train_scores.append(aic_train)
aic_valid_scores.append(aic_valid)

print(f"Fold {fold + 1} - MSE навчальні дані: {mse_train:.8f}")
print(f"Fold {fold + 1} - MSE тестові дані: {mse_valid:.8f}")
print(f"Fold {fold + 1} - R² навчальні дані: {r2_train:.8f}")
print(f"Fold {fold + 1} - R² тестові дані: {r2_valid:.8f}")
print(f"Fold {fold + 1} - AIC навчальні дані: {aic_train:.8f}")
print(f"Fold {fold + 1} - AIC тестові дані: {aic_valid:.8f}")

# Побудова графіків
plt.figure(figsize=(12, 6))

# Графік для тренувальних даних
plt.subplot(1, 2, 1)
plt.plot(y_train, label='Фактичні (Навчальні дані)', color='blue')
plt.plot(y_train_pred_RT, label='Прогнозовані (Навчальні дані)',
color='red')

plt.title(f'Fold {fold+1} - Навчальні дані (Decision Tree)')
plt.xlabel('Time step')
plt.ylabel('Target value')
plt.legend()

```

```

        # Графік для тестових даних
        plt.subplot(1, 2, 2)
        plt.plot(y_valid, label='Фактичні (Валідаційні дані)',
color='blue')
        plt.plot(y_valid_pred_RT, label='Прогнозовані (Валідаційні дані)',
color='red')

        plt.title(f'Fold {fold+1} - Валідаційні дані (Decision Tree)')
        plt.xlabel('Time step')
        plt.ylabel('Target value')
        plt.legend()

        plt.tight_layout()
        plt.show()

    # Виведення середніх значень метрик
    print('\nСередні значення метрик по всіх фолдах:')
    print(f'Середній MSE для навчальних даних:
{np.mean(mse_train_scores):.8f}')
    print(f'Середній MSE для тестових даних:
{np.mean(mse_valid_scores):.8f}')
    print(f'Середній R² для навчальних даних:
{np.mean(r2_train_scores):.8f}')
    print(f'Середній R² для тестових даних:
{np.mean(r2_valid_scores):.8f}')
    print(f'Середній AIC для навчальних даних:
{np.mean(aic_train_scores):.8f}')
    print(f'Середній AIC для тестових даних:
{np.mean(aic_valid_scores):.8f}')

```

2.4) Модель GradientBoostingRegressor

```

# Модель GradientBoostingRegressor

from sklearn.ensemble import GradientBoostingRegressor # Регресор
градієнтного бустингу
from sklearn.model_selection import TimeSeriesSplit # Крос-валідація
часових рядів
from sklearn.metrics import mean_squared_error, r2_score # Метрики для
MSE та R²
import numpy as np # Для операцій з масивами
import matplotlib.pyplot as plt # Для побудови графіків
import warnings

```

```

from sklearn.exceptions import DataConversionWarning

warnings.filterwarnings(action='ignore',
category=DataConversionWarning)

# Ініціалізація TimeSeriesSplit для 5-fold cross-validation
n_splits = 5
tscv = TimeSeriesSplit(n_splits=n_splits)

# Ініціалізація GradientBoostingRegressor
GBreg = GradientBoostingRegressor(validation_fraction=0.15)

# Масиви для зберігання метрик для кожного fold
mse_train_scores, mse_valid_scores = [], []
r2_train_scores, r2_valid_scores = [], []
aic_train_scores, aic_valid_scores = [], []

# Функція для обчислення AIC
def calculate_aic(y_true, y_pred, num_params):
    resid = y_true - y_pred
    rss = np.sum(resid**2)
    n = len(y_true)
    aic = 2 * num_params + n * np.log(rss / n)
    return aic

# Цикл крос-валідації для кожного fold
for fold, (train_index, valid_index) in enumerate(tscv.split(x)):
    print(f"Fold {fold + 1}/{n_splits}")

    # Розділення даних на навчальний та валідаційний набори
    x_train, x_valid = x[train_index], x[valid_index]
    y_train, y_valid = y[train_index], y[valid_index]

    accuracyGB_test = 0
    while accuracyGB_test < 0.4:
        # Навчання моделі на навчальному наборі
        GBreg.fit(x_train, y_train)
        # Оцінка моделі на валідаційному наборі
        accuracyGB_test = GBreg.score(x_valid, y_valid)

    # Оцінка моделі на навчальному наборі

```

```

accuracyGB_train = GBreg.score(x_train, y_train)
print(f"Gradient Boosting R^2 (Навчальні дані):
{accuracyGB_train:.8f}")

print(f"Gradient Boosting R^2 (Валідаційні дані):
{accuracyGB_test:.8f}")

# Прогнозування на навчальному та валідаційному наборах
y_train_pred_GB = GBreg.predict(x_train)
y_valid_pred_GB = GBreg.predict(x_valid)

# Обчислення MSE для тренувального та валідаційного наборів
mse_train_GB = mean_squared_error(y_train, y_train_pred_GB)
mse_valid_GB = mean_squared_error(y_valid, y_valid_pred_GB)
print(f'MSE (Навчальні дані) Gradient Boosting:
{mse_train_GB:.8f}')

print(f'MSE (Валідаційні дані) Gradient Boosting:
{mse_valid_GB:.8f}')

# Кількість параметрів для обчислення AIC
num_trees_GB = GBreg.n_estimators
max_depth_GB = 3
num_params_GB = num_trees_GB * (2**max_depth_GB - 1)

# Обчислення AIC
aic_train_GB = calculate_aic(y_train, y_train_pred_GB,
num_params_GB)
aic_valid_GB = calculate_aic(y_valid, y_valid_pred_GB,
num_params_GB)

print(f"AIC (Навчальні дані) Gradient Boosting:
{aic_train_GB:.8f}")

print(f"AIC (Валідаційні дані) Gradient Boosting:
{aic_valid_GB:.8f}")

# Збереження метрик для кожного fold
mse_train_scores.append(mse_train_GB)
mse_valid_scores.append(mse_valid_GB)
r2_train_scores.append(accuracyGB_train)
r2_valid_scores.append(accuracyGB_test)
aic_train_scores.append(aic_train_GB)
aic_valid_scores.append(aic_valid_GB)

```

```

        # Побудова графіків фактичних та прогнозованих значень для
поточного fold

        plt.figure(figsize=(12, 6))

        # Графік для тренувальних даних
        plt.subplot(1, 2, 1)
        plt.plot(y_train, label='Фактичні (Навчальні дані)', color='blue')
        plt.plot(y_train_pred_GB, label='Прогнозовані (Навчальні дані)',
color='red')
        plt.title(f'Fold {fold + 1} - Навчальні дані (Gradient Boosting)')
        plt.xlabel('Time step')
        plt.ylabel('Target value')
        plt.legend()

        # Графік для валідаційних даних
        plt.subplot(1, 2, 2)
        plt.plot(y_valid, label='Фактичні (Валідаційні дані)',
color='blue')
        plt.plot(y_valid_pred_GB, label='Прогнозовані (Валідаційні дані)',
color='red')
        plt.title(f'Fold {fold + 1} - Валідаційні дані (Gradient
Boosting)')
        plt.xlabel('Time step')
        plt.ylabel('Target value')
        plt.legend()

        plt.tight_layout()
        plt.show()

    # Обчислення та виведення середніх метрик по всіх fold
    mean_mse_train = np.mean(mse_train_scores)
    mean_mse_valid = np.mean(mse_valid_scores)
    mean_r2_train = np.mean(r2_train_scores)
    mean_r2_valid = np.mean(r2_valid_scores)
    mean_aic_train = np.mean(aic_train_scores)
    mean_aic_valid = np.mean(aic_valid_scores)

    print(f"\nСередній MSE для навчальних даних: {mean_mse_train:.8f}")
    print(f"Середній MSE для тестових даних: {mean_mse_valid:.8f}")
    print(f"Середній R2 для навчальних даних: {mean_r2_train:.8f}")
    print(f"Середній R2 для тестових даних: {mean_r2_valid:.8f}")

```



```
print(f"Середній AIC для навчальних даних: {mean_aic_train:.8f}")
print(f"Середній AIC для тестових даних: {mean_aic_valid:.8f}")
```

2.5) XGBoost модель

```
# XGBoost модель

import xgboost as xgb # Бібліотека XGBoost для градієнтного бустингу
дерев рішень

from sklearn.multioutput import MultiOutputRegressor # Регресор з
багатьма виходами

from sklearn.model_selection import TimeSeriesSplit # Поділ на
навчальну та валідаційну вибірку за часовим рядом

from sklearn.metrics import mean_squared_error, r2_score # Метрики для
середньоквадратичної помилки та коефіцієнта детермінації

import numpy as np
import matplotlib.pyplot as plt

# Ініціалізація TimeSeriesSplit для 5-fold cross-validation
n_splits = 5
tscv = TimeSeriesSplit(n_splits=n_splits)

# Ініціалізація моделі XGBoost
modelXGB = MultiOutputRegressor(xgb.XGBRegressor(
    objective='reg:squarederror',
    n_estimators=100,
    max_depth=8,
    learning_rate=0.1,
    subsample=0.8,
    colsample_bytree=0.8
))

# Масиви для зберігання метрик
mse_train_scores, mse_valid_scores = [], []
r2_train_scores, r2_valid_scores = [], []
aic_train_scores, aic_valid_scores = [], []

# Функція для обчислення AIC
def calculate_aic(y_true, y_pred, num_params):
    resid = y_true - y_pred
    sse = np.sum(resid ** 2)
    n = len(y_true)
```

```

aic = 2 * num_params + n * np.log(sse / n)
return aic

# Цикл крос-перевірки
for fold, (train_index, valid_index) in enumerate(tscv.split(x)):
    print(f"Fold {fold + 1}/{n_splits}")

    # Розділення даних на навчальну та валідаційну вибірки
    x_train, x_valid = x[train_index], x[valid_index]
    y_train, y_valid = y[train_index], y[valid_index]

    # Тренування моделі XGBoost
    modelXGB.fit(x_train, y_train)

    # Обчислення коефіцієнта детермінації R2
    r2XGB_train = modelXGB.score(x_train, y_train)
    r2XGB_valid = modelXGB.score(x_valid, y_valid)
    print(f"XGBoost R^2 (Навчальні дані): {r2XGB_train:.8f}")
    print(f"XGBoost R^2 (Валідаційні дані): {r2XGB_valid:.8f}")

    # Отримання прогнозованих значень
    y_train_pred = modelXGB.predict(x_train)
    y_valid_pred = modelXGB.predict(x_valid)

    # Обчислення середньоквадратичної помилки
    mse_train = mean_squared_error(y_train, y_train_pred)
    mse_valid = mean_squared_error(y_valid, y_valid_pred)
    print(f'MSE (Навчальні дані): {mse_train:.8f}')
    print(f'MSE (Валідаційні дані): {mse_valid:.8f}')

    # Обчислення AIC
    num_trees = 100
    max_depth = 8
    num_params = num_trees * (2 ** max_depth - 1)

    aic_train = calculate_aic(y_train.flatten(),
y_train_pred.flatten(), num_params)
    aic_valid = calculate_aic(y_valid.flatten(),
y_valid_pred.flatten(), num_params)
    print(f"AIC (Навчальні дані): {aic_train:.8f}")
    print(f"AIC (Валідаційні дані): {aic_valid:.8f}")

```

```

# Збереження метрик
mse_train_scores.append(mse_train)
mse_valid_scores.append(mse_valid)
r2_train_scores.append(r2XGB_train)
r2_valid_scores.append(r2XGB_valid)
aic_train_scores.append(aic_train)
aic_valid_scores.append(aic_valid)

# Візуалізація фактичних та прогнозованих значень
num_targets = y_valid.shape[1]
plt.figure(figsize=(16, num_targets * 6))

# Графіки для навчальної вибірки
for i in range(num_targets):
    plt.subplot(num_targets, 2, 2 * i + 1) # First column for
training data
    plt.plot(y_train[:, i], label='Фактичні (Навчальні дані)',
color='blue')
    plt.plot(y_train_pred[:, i], label='Прогнозовані (Навчальні
дані)', color='red')
    plt.title(f'Fold {fold + 1} - Фактичні vs Прогнозовані
(Навчальні дані) для Fold {i+1}')
    plt.xlabel('Samples')
    plt.ylabel('Values')
    plt.legend()

# Plot for validation data
for i in range(num_targets):
    plt.subplot(num_targets, 2, 2 * i + 2) # Second column for
validation data
    plt.plot(y_valid[:, i], label='Фактичні (Валідаційні дані)',
color='blue')
    plt.plot(y_valid_pred[:, i], label='Прогнозовані (Валідаційні
дані)', color='red')
    plt.title(f'Fold {fold + 1} - Фактичні vs Прогнозовані
(Валідаційні дані) для Fold {i+1}')
    plt.xlabel('Samples')
    plt.ylabel('Values')
    plt.legend()

```

```

plt.tight_layout()
plt.show()

# Calculate and print average metrics across all folds
mean_mse_train = np.mean(mse_train_scores)
mean_mse_valid = np.mean(mse_valid_scores)
mean_r2_train = np.mean(r2_train_scores)
mean_r2_valid = np.mean(r2_valid_scores)
mean_aic_train = np.mean(aic_train_scores)
mean_aic_valid = np.mean(aic_valid_scores)

print(f"\nСередній MSE (Навчальні дані): {mean_mse_train:.8f}")
print(f"Середній MSE (Валідаційні дані): {mean_mse_valid:.8f}")
print(f"Середній коефіцієнт детермінації R2 (Навчальні дані):
{mean_r2_train:.8f}")
print(f"Середній коефіцієнт детермінації R2 (Валідаційні дані):
{mean_r2_valid:.8f}")
print(f"Середній AIC (Навчальні дані): {mean_aic_train:.8f}")
print(f"Середній AIC (Валідаційні дані): {mean_aic_valid:.8f}")

```

2.6) RNN модель

```

# RNN модель

import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, r2_score
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense
from sklearn.model_selection import TimeSeriesSplit

# Функція для розрахунку AIC
def calculate_aic(n, mse, num_params):
    aic = n * np.log(mse) + 2 * num_params
    return aic

def create_sequences(data, time_steps):
    sequences = []
    labels = []
    for i in range(len(data) - time_steps):

```

```

        sequences.append(data[i:(i + time_steps)])
        labels.append(data[i + time_steps])
    return np.array(sequences), np.array(labels)

def build_rnn_model(time_steps):
    model = Sequential([
        SimpleRNN(units=50, activation='tanh', input_shape=(time_steps,
1)),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mse')
    return model

# Параметри
time_steps = 10
n_splits = 5
epochs = 50
batch_size = 32

# Створення TimeSeriesSplit
tscv = TimeSeriesSplit(n_splits=n_splits)

# Масиви для збереження метрик
mse_train_scores = []
mse_valid_scores = []
r2_train_scores = []
r2_valid_scores = []
aic_train_scores = []
aic_valid_scores = []

# Нормалізація всіх даних
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(y.reshape(-1, 1))

# Створення послідовностей для всього набору даних
X_sequences, y_sequences = create_sequences(scaled_data, time_steps)
X_sequences = X_sequences.reshape((X_sequences.shape[0],
X_sequences.shape[1], 1))

# Розрахунок кількості параметрів моделі
temp_model = build_rnn_model(time_steps)

```

```

num_params = temp_model.count_params()

# Цикл по кожному розбиттю
for fold, (train_index, valid_index) in
enumerate(tscv.split(X_sequences)):
    print(f"\nFold {fold + 1}/{n_splits}")

    # Розділення даних на тренувальні та валідаційні
    X_train, X_valid = X_sequences[train_index],
X_sequences[valid_index]
    y_train_seq, y_valid_seq = y_sequences[train_index],
y_sequences[valid_index]

    # Створення та навчання моделі
    model = build_rnn_model(time_steps)

    # Callback для early stopping
    callback = tf.keras.callbacks.EarlyStopping(
        monitor='val_loss',
        patience=10,
        restore_best_weights=True
    )

    # Навчання моделі
    history = model.fit(
        X_train,
        y_train_seq,
        epochs=epochs,
        batch_size=batch_size,
        validation_data=(X_valid, y_valid_seq),
        callbacks=[callback],
        verbose=0
    )

    # Прогнозування
    y_train_pred = model.predict(X_train)
    y_valid_pred = model.predict(X_valid)

    # Зворотне перетворення для отримання оригінальних значень
    y_train_pred = scaler.inverse_transform(y_train_pred)
    y_valid_pred = scaler.inverse_transform(y_valid_pred)

```

```

y_train_actual = scaler.inverse_transform(y_train_seq.reshape(-1,
1))

y_valid_actual = scaler.inverse_transform(y_valid_seq.reshape(-1,
1))

# Обчислення метрик
mse_train = mean_squared_error(y_train_actual, y_train_pred)
mse_valid = mean_squared_error(y_valid_actual, y_valid_pred)
r2_train = r2_score(y_train_actual, y_train_pred)
r2_valid = r2_score(y_valid_actual, y_valid_pred)

# Розрахунок AIC
aic_train = calculate_aic(len(y_train_actual), mse_train,
num_params)
aic_valid = calculate_aic(len(y_valid_actual), mse_valid,
num_params)

# Збереження метрик
mse_train_scores.append(mse_train)
mse_valid_scores.append(mse_valid)
r2_train_scores.append(r2_train)
r2_valid_scores.append(r2_valid)
aic_train_scores.append(aic_train)
aic_valid_scores.append(aic_valid)

print(f'MSE для навчальних даних (Fold {fold+1}): {mse_train:.8f}')
print(f'MSE для тестових даних (Fold {fold+1}): {mse_valid:.8f}')
print(f'R² для навчальних даних (Fold {fold+1}): {r2_train:.8f}')
print(f'R² для тестових даних (Fold {fold+1}): {r2_valid:.8f}')
print(f'AIC для навчальних даних (Fold {fold+1}): {aic_train:.8f}')
print(f'AIC для тестових даних (Fold {fold+1}): {aic_valid:.8f}')

# Побудова графіків
plt.figure(figsize=(12, 6))

# Графік для тренувальних даних
plt.subplot(1, 2, 1)
plt.plot(y_train_actual, label='Фактичні (Навчальні дані)',
color='blue')
plt.plot(y_train_pred, label='Прогнозовані (Навчальні дані)',
color='red')

```

```

plt.title(f'Fold {fold+1} - Навчальні дані (RNN)')
plt.xlabel('Time step')
plt.ylabel('Target value')
plt.legend()

# Графік для валідаційних даних
plt.subplot(1, 2, 2)
plt.plot(y_valid_actual, label='Фактичні (Валідаційні дані)',
color='blue')
plt.plot(y_valid_pred, label='Прогнозовані (Валідаційні дані)',
color='red')
plt.title(f'Fold {fold+1} - Валідаційні дані (RNN)')
plt.xlabel('Time step')
plt.ylabel('Target value')
plt.legend()

plt.tight_layout()
plt.show()

# Виведення середніх значень метрик
print('\nСередні значення метрик по всіх фолдах:')
print(f'Середній MSE для навчальних даних:
{np.mean(mse_train_scores):.8f}')
print(f'Середній MSE для тестових даних:
{np.mean(mse_valid_scores):.8f}')
print(f'Середній R² для навчальних даних:
{np.mean(r2_train_scores):.8f}')
print(f'Середній R² для тестових даних:
{np.mean(r2_valid_scores):.8f}')
print(f'Середній AIC для навчальних даних:
{np.mean(aic_train_scores):.8f}')
print(f'Середній AIC для тестових даних:
{np.mean(aic_valid_scores):.8f}')

```

2.7) Побудова нормалізованих графіків за результатами моделей

```

import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Data preparation

```



```

models = ["Gradient Boosting", "XGBoost", "RNN", "Decision Tree",
"NARX"]

folds = ["1-fold", "2-fold", "3-fold", "4-fold", "5-fold"]

# Data for each model
data = {
    "Gradient Boosting": {
        "MSE": [0.00048217, 0.00024659, 0.00021137, 0.00015264,
0.00027960],
        "R2": [0.9722, 0.9858, 0.9871, 0.9903, 0.9892],
        "AIC": [3258.95, 3215.35, 3188.75, 3176.04, 3448.75],
    },
    "XGBoost": {
        "MSE": [0.00048707, 0.00081122, 0.00014358, 0.00019493,
0.00021514],
        "R2": [0.9719, 0.9523, 0.9913, 0.9876, 0.9917],
        "AIC": [46431.37, 46086.68, 45699.66, 45882.83, 45941.92],
    },
    "RNN": {
        "MSE": [0.00129290, 0.00068793, 0.00055452, 0.00041292,
0.00036811],
        "R2": [0.9269, 0.9611, 0.9663, 0.9739, 0.9858],
        "AIC": [1324.78, 1046.05, 818.55, 642.23, 573.53],
    },
    "Decision Tree": {
        "MSE": [0.00143358, 0.00098765, 0.00061588, 0.00059168,
0.00113648],
        "R2": [0.9174, 0.9433, 0.9625, 0.9625, 0.9561],
        "AIC": [-3852.00, -4075.19, -4358.08, -4382.09, -3991.11],
    },
    "NARX": {
        "MSE": [0.00130916, 0.00105109, 0.00098448, 0.00080229,
0.00162684],
        "R2": [0.9246, 0.9396, 0.9401, 0.9491, 0.9372],
        "AIC": [19577.62, 19446.10, 19406.88, 19284.30, 19707.75],
    },
}

# Normalizing the data
scaler = MinMaxScaler()

```

```

# Plotting
fig, axes = plt.subplots(len(models), 1, figsize=(12, 30))

for idx, model in enumerate(models):
    df = pd.DataFrame(data[model], index=folds)

    # Normalize each metric for better visualization
    normalized_data = scaler.fit_transform(df)
    normalized_df = pd.DataFrame(normalized_data, columns=df.columns,
index=folds)

    ax = axes[idx]
    normalized_df.plot(ax=ax, marker='o', title=f"{model} Metrics
(Normalized)")
    ax.set_xlabel("Folds")
    ax.set_ylabel("Normalized Metrics (0 to 1)")
    ax.grid(True)

plt.tight_layout()
plt.show()

```

ДОДАТОК Д. Код програми для задачі оптимізації процесу бродіння

```

X=[[VarName1;VarName6;VarName11;VarName16]
[VarName2;VarName7;VarName12;VarName17]
[VarName3;VarName8;VarName13;VarName18]
[VarName5;VarName10;VarName15;VarName20]];
Y=[VarName4;VarName9;VarName14;VarName19];

R0 = corrcoef([X Y])
xnames = {'Gl', 'H', 'T', 'CO2'};
yname = 'Et';
xynames = xnames; xynames(5)={yname};
corrplot([X Y],'testR','on', 'varNames',xynames)

% Статичне (детерміноване) дерево регресії
rtree = fitrtree(X,Y,'MinParent',30,...
    'PredictorNames', xnames,...
    'ResponseName', yname)
view(rtree)
view(rtree,'mode','graph') % graphic description

```

```

rng 'default';
rtree1 = fitrtree(X,Y,'MinParent',100,...
    'PredictorNames', xnames,'ResponseName', yname)
view(rtree1)
view(rtree1,'mode','graph')

rng 'default';
rtree2 = fitrtree(X,Y,'MinParent',400,...
    'PredictorNames', xnames,'ResponseName', yname)
view(rtree2)
view(rtree2,'mode','graph')

resuberror = resubLoss(rtree)

%Втрата resuberror для дерева регресії є середньоквадратичною помилкою.
Отримане значення вказує на те, що типова прогнозована помилка для дерева
становить близько квадратного кореня з 14,5 або трохи більше 3,8.
cvrtree = crossval(rtree);
cvloss = kfoldLoss(cvrtree)

%Перехресна перевірена втрата становить близько 30, що означає, що
типова похибка прогнозування для дерева на нових даних становить близько
5.5. Це демонструє, що перехресна перевірена втрата зазвичай перевищує
просту втрату заміщення.
resuberror1 = resubLoss(rtree1)
cvrtree1 = crossval(rtree1);
cvloss1 = kfoldLoss(cvrtree1)
resuberror2 = resubLoss(rtree2)
cvrtree2 = crossval(rtree2);
cvloss2 = kfoldLoss(cvrtree2)

% Ансамбль дерев регресії - бустінг
RegTreeTemp = templateTree('Surrogate','On');
RegTreeEns =
fitensemble(X,Y,'LSBoost',300,'Tree','PredictorNames',xnames)
cvens = crossval(RegTreeEns);
lossLSBoostCV = kfoldLoss(cvens)
lossLSBoostTrain = loss(RegTreeEns,X,Y)

```

```

% Використовуйте навчений ансамбль регресії для передбачення
pred_Et = predict(RegTreeEns,[250 15 26 750])

% Бегінг
RegTreeEns1 = fitensemble(X,Y,'bag',200,'Tree',...
    'type','regression')
cvens1 = crossval(RegTreeEns1);
lossBaggingCV = kfoldLoss(cvens1)
lossBaggingTrain = loss(RegTreeEns1,X,Y)

% Бегінг 2
rng(1945,'twister');
RegTreeEns2 = TreeBagger(100,X,Y,'Method','R','OOBVarImp','On',...
    'MinLeaf',15)
lossBaggingCV2 = mean(oobError(RegTreeEns2));
predictions2 = predict(RegTreeEns2, X);
lossBaggingTrain2 = mean((Y - predictions2).^2);

disp(['lossBaggingCV2: ', num2str(lossBaggingCV2)]);
disp(['lossBaggingTrain2: ', num2str(lossBaggingTrain2)]);

figure;
plot(loss(RegTreeEns,X,Y,'Mode','cumulative'),'r');
hold on;
plot(oobLoss(RegTreeEns1,'Mode','cumulative'),'b');
plot(oobError(RegTreeEns2,'Mode','cumulative'),'m');
hold off;
xlabel('Number of trees');
ylabel('Cross-validated exponential loss');
legend('LSBoost','Bag','TreeBagger','Location','NE');

%-----Finding the Optimal Leaf Size
leaf = [5 10 20 50 100];
col = 'rbcmy';
figure
for i=1:length(leaf)
    b =
TreeBagger(50,X,Y,'Method','R','OOBPred','On','MinLeaf',leaf(i));
    plot(oobError(b),col(i));
    hold on;
end

```

```

xlabel 'Number of Grown Trees';
ylabel 'Mean Squared Error' ;
legend({'5' '10' '20' '50' '100'}, 'Location', 'NorthEast');
hold off;

% Фіксуємо значення Gl
Gl_fixed = 166;

% Встановлюємо межі для керованих параметрів
lb = [5, 20, 400];      % Нижні межі для [H, T, CO2]
ub = [30, 35, 1500];   % Верхні межі

% Функція цілі для patternsearch (максимізація)
obj_fun = @(x)(-predict(RegTreeEnsl, [Gl_fixed, x(1), x(2), x(3)]));

% Налаштування для patternsearch
options = optimoptions('patternsearch', ...
    'Display', 'iter', ...      % Показувати прогрес
    'MaxIterations', 1000, ...  % Максимальна кількість ітерацій
    'MaxFunctionEvaluations', 2000, ... % Максимальна кількість
    обчислень функції
    'InitialMeshSize', 1, ...    % Початковий розмір сітки
    'MeshTolerance', 1e-6, ...   % Толерантність сітки
    'CompletePoll', 'on', ...    % Повний перебір точок
    'SearchMethod', @searchlhs); % Використання Latin Hypercube для
    початкового пошуку

% Створюємо кілька початкових точок для мултистарту
n_starts = 5;
best_Et = -inf;
best_x = [];

fprintf('Початок оптимізації з використанням patternsearch...\n');

for i = 1:n_starts
    % Генеруємо випадкову початкову точку в межах обмежень
    x0 = lb + rand(1,3).*(ub-lb);

    % Запускаємо patternsearch

```

```

[x_temp, fval_temp, exitflag, output] = patternsearch(obj_fun, x0,
[], [], [], [], ...
    lb, ub, [], options);

current_Et = -fval_temp; % Конвертуємо назад для виведення

if current_Et > best_Et
    best_Et = current_Et;
    best_x = x_temp;
    fprintf('Ітерація %d: Знайдено краще рішення: Et = %.2f\n', i,
current_Et);
end

fprintf('Ітерація %d завершена. Кількість обчислень функції: %d\n',
...
    i, output.funcscount);
end

% Виведення результатів
fprintf('\nФінальні результати оптимізації:\n');
fprintf('Фіксовані параметри:\n');
fprintf('Gl: %.2f (фіксоване значення)\n', Gl_fixed);
fprintf('\nОптимальні значення керованих параметрів:\n');
fprintf('H: %.2f\n', best_x(1));
fprintf('T: %.2f\n', best_x(2));
fprintf('CO2: %.2f\n', best_x(3));
fprintf('Максимальне Et: %.2f\n', best_Et);

% Порівняння з початковою точкою
x0_original = [166, 15, 26, 750];
Et_initial = predict(RegTreeEnsl, x0_original);
fprintf('\nПочаткове Et: %.2f\n', Et_initial);
fprintf('Покращення: +%.2f%%\n', ((best_Et -
Et_initial)/Et_initial)*100);

% Фіксуємо значення Gl
Gl_fixed = 166;

% Встановлюємо межі для керованих параметрів
lb = [5, 25, 400]; % Нижні межі для [H, T, CO2]

```

```

ub = [30, 35, 1500]; % Верхні межі

% Визначаємо допустимий діапазон етанолу
Et_initial = predict(RegTreeEns1, [Gl_fixed, 15, 26, 750]); % Еталонне
значення
Et_min = Et_initial * 0.95; % 95% від початкового значення
Et_max = Et_initial * 1.05; % 105% від початкового значення

% Функція цілі (мінімізація тільки H)
obj_fun = @(x) x(1);

% Нелінійні обмеження для концентрації етанолу
nonlcon = @(x) deal([], ... % Немає нерівностей
    [predict(RegTreeEns1, [Gl_fixed, x(1), x(2), x(3)]) - Et_max;
    Et_min - predict(RegTreeEns1, [Gl_fixed, x(1), x(2), x(3)])]);

% Налаштування оптимізації
options = optimoptions('fmincon', ...
    'Display', 'iter', ...
    'Algorithm', 'sqp', ...
    'MaxIterations', 1000, ...
    'MaxFunctionEvaluations', 2000, ...
    'StepTolerance', 1e-6, ...
    'OptimalityTolerance', 1e-6);

% Додаємо кілька початкових точок для пошуку глобального мінімуму
num_starts = 5;
x_opts = zeros(num_starts, 3);
fvals = zeros(num_starts, 1);

for i = 1:num_starts
    % Генеруємо випадкову початкову точку в межах допустимих значень
    x0 = lb + rand(1,3).*(ub - lb);

    % Виконання оптимізації
    [x_opts(i,:), fvals(i)] = fmincon(obj_fun, x0, [], [], [], [], lb,
ub, nonlcon, options);
end

% Вибираємо найкращий результат
[best_fval, best_idx] = min(fvals);

```

```

x_opt = x_opts(best_idx,:);
Et_opt = predict(RegTreeEns1, [Gl_fixed, x_opt(1), x_opt(2),
x_opt(3)]);

% Виведення результатів
fprintf('\nФінальні результати оптимізації:\n');
fprintf('Фіксовані параметри:\nGl: %.2f\n', Gl_fixed);
fprintf('\nОптимальні значення керованих параметрів:\n');
fprintf('H: %.2f\n', x_opt(1));
fprintf('T: %.2f\n', x_opt(2));
fprintf('CO2: %.2f\n', x_opt(3));
fprintf('Концентрація Et: %.2f\n', Et_opt);

% Порівняння з початковою точкою
x0_original = [Gl_fixed, 15, 26, 750];
Et_initial = predict(RegTreeEns1, x0_original);

% Обчислення змін
H_change = x_opt(1) - 15;
H_change_percent = (H_change / 15) * 100;
T_change = x_opt(2) - 26;
T_change_percent = (T_change / 26) * 100;
CO2_change = x_opt(3) - 750;
CO2_change_percent = (CO2_change / 750) * 100;
Et_change = Et_opt - Et_initial;
Et_change_percent = (Et_change / Et_initial) * 100;

% Виведення порівняльних результатів
fprintf('\nПорівняння результатів:\n');
fprintf('H: %.2f → %.2f (зміна: %.2f%%)\n', 15, x_opt(1),
H_change_percent);
fprintf('T: %.2f → %.2f (зміна: %.2f%%)\n', 26, x_opt(2),
T_change_percent);
fprintf('CO2: %.2f → %.2f (зміна: %.2f%%)\n', 750, x_opt(3),
CO2_change_percent);
fprintf('Et: %.2f → %.2f (зміна: %.2f%%)\n', Et_initial, Et_opt,
Et_change_percent);

% Додаткова перевірка обмежень
fprintf('\nПеревірка обмежень:\n');
fprintf('Цільовий діапазон Et: %.2f - %.2f\n', Et_min, Et_max);

```



```
fprintf('Отримане значення Et: %.2f\n', Et_opt);
```

ДДД

ДОДАТОК Е. Коди додаткових програм для систем автоматичного керування

1) Розрахунок компенсатора

```
% Передаточні функції
s = tf('s');
G11 = 12.8 / (16.7*s + 1);
G12 = -18.9 / (21*s + 1);
G21 = 6.6 / (10.9*s + 1);
G22 = -19.4 / (14.1*s + 1);

% Матриця об'єкта
G = [G11, G12; G21, G22];

% Матриця G_diag
G_diag = [G11, 0; 0, G22];
G0 = [12.8, -18.9; 6.6, -19.4];
D_static = inv(G0);

% Компенсатор по класиці
D = G_diag * D_static;
freqresp(D,0)

Результат:
D =
    2.0094    -1.9576
   -1.0361     2.0094
```

2) Environment для RL-моделі

```
mdl = 'Distillation_RL_deep';
open_system(mdl);
agentblk = [mdl 'RLAgent'];

% ---- Sim space -----
% Observation
numObserv = 32;
u12min = -1*ones(7,1); u12max = ones(7,1);
e12min = -1*ones(8,1); e12max = ones(8,1);
int12min = [-inf;-inf]; int12max = [inf;inf];
```

```

LL = [e12min; e12min; u12min; u12min; int12min];
UL = [e12max; e12max; u12max; u12max; int12max];
observationInfo = rlNumericSpec([numObserv 1], 'LowerLimit', LL,
'UpperLimit', UL);
observationInfo.Name = 'observations';
observationInfo.Description = 'System states';
observationInfo

% Actions
numAction = 2;
actionInfo = rlNumericSpec([numAction 1], 'LowerLimit', [-1; -1],
'UpperLimit', [1; 1]);
actionInfo.Name = 'actions';
actionInfo.Description = 'Agent actions';
actionInfo

% ----- ENV -----
env = rlSimulinkEnv mdl, agentblk, observationInfo, actionInfo);
%set_param mdl, 'StopTime', '100');

% randomize initial values for lateral deviation and relative yaw angle
env.ResetFcn = @(in) localResetFcn(in);

Ts = 1
Tf = 100;
%-----
function in = localResetFcn(in)
% Reset the initial ui
ui_rand = -0.5+randi(10,2,1)/10;
in = setVariable(in,"u1", ui_rand(1));
in = setVariable(in,"u2", ui_rand(2));

ri = -0.5 + randi(10,1)/10;
r = [ri -ri];
timeVector = [0 0];
in = setBlockParameter(in,'Distillation_RL_deep/REF%_xD_xB','Time',
mat2str(timeVector));
in = setBlockParameter(in,'Distillation_RL_deep/REF%_xD_xB','After',
mat2str(r));
end

```

ДОДАТОК Є. Акти впровадження

Міністерство освіти і науки України
Національний університет харчових технологій



АКТ

впровадження результатів науково-дослідних, дослідно-конструкторських і дисертаційних робіт у навчальний процес закладу вищої освіти

Замовник Національний університет харчових технологій в особі проректора з наукової роботи Токарчука Сергія Володимировича, який діє на підставі наказу №87 від 20.10.22р.

Дійсним актом підтверджується, що результати: дербюджетної науково-дослідної роботи «Науково-технічні основи розробки систем автоматизації складних об'єктів з використанням інтелектуальних методів та технологій Індустрії 4.0», державний реєстраційний номер: № 0121U113308

виконаної кафедрою автоматизації та комп'ютерних технологій систем управління НУХТ ім. А.П. Ладанюка

виконуваної з грудня 2023 р. по січень 2025 р.

впроваджені у навчальний процес кафедри автоматизації та комп'ютерних технологій систем управління ім. А.П. Ладанюка

1. Вид впроваджених результатів: розроблені новітні підходи до побудови віртуальних сенсорів харчової промисловості. Результатом є система аналітичного резервування, що може бути використана для підвищення продуктивності та керованості виробництва.

Опубліковано тези доповідей.

2. Форма впровадження: розділи теоретичних тем лекційних занять, лабораторні роботи.

3. Новизна результатів науково-дослідних робіт: імітаційні моделі віртуальних сенсорів процесів брагоректифікації та дисциляції, що можуть бути використані для виявлення несправностей у роботі фізичних сенсорів та для переключення системи керування на віртуальний сенсор, тим самим підвищуючи керованість та стабільність виробництва. Модель побудована методами машинного навчання за даними з автоматичної системи керування.

4. Перелік курсів і дисциплін, у рамках яких викладені результати НДР: результати роботи використано у навчальному процесі кафедри автоматизації та комп'ютерних технологій систем управління під час викладання дисципліни «Математичне та програмне забезпечення систем керування виробничими операціями промислового підприємства» для здобувачів спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» освітньо-наукової програми «Автоматизація та комп'ютерно-інтегровані технології» ступеня «доктор філософії» та у науково-дослідній роботі здобувачів.

5. Соціальний і науково-технічний ефект: розроблені імітаційні моделі віртуальних сенсорів процесів брагоректифікації та дисциляції, що можуть бути використані для виявлення несправностей у роботі фізичних сенсорів, тим самим підвищується інформаційна підтримка оператора-технолога. Крім того, дані моделі можуть бути використані як цифрові двійники процесів в рамках концепції Індустрії 4.0. Здобувачі на конкретних прикладах ознайомлені із науковими досягненнями і сучасними практиками в галузі математичного моделювання харчових процесів.

Керівник НДР

Зав.каф.

«20» 02 2025 р.



Ярослав СМІТЮХ

Виконавець

Аспірант

«20» 02 2025 р.



Олександр ОМЕЛЬЧЕНКО

ЗАТВЕРДЖУЮ

Директор ТОВ «САУТКОМ»

Коломієць С.М.

12.02.2025р.



АКТ

**впровадження результатів дисертаційної роботи
Омельченка Олександра Станіславовича «Автоматизована система
моніторингу процесів виробництва спирту та біоетанолу на основі
віртуальних інструментів контролю»**

Даний акт складений за результатами впровадження, що проводилися на «Марилівському спиртовому заводі» (ТОВ «ВІТАГРО ЕНЕРДЖІ») із залученням наукової установи – кафедри Автоматизації та комп'ютерних технологій систем управління ім. А.П. Ладанюка Національного університету харчових технологій.

Цей акт про те, що 12 жовтня 2024 року на «Марилівському спиртовому заводі» (ТОВ «ВІТАГРО ЕНЕРДЖІ») впроваджено систему моніторингу процесів виробництва спирту та біоетанолу на основі віртуальних інструментів контролю, що дозволяє своєчасно виявляти поломки фізичних датчиків та виконувати переключення автоматичних регуляторів на віртуальні сенсори. Така система підвищує керованість та стабільність виробництва.

В результаті отримано збільшення ефективності функціонування технологічних процесів в порівнянні з попередніми системами керування в загальному на 3,5%.

Завідувач каф.АКТСУ
ім. А.П. Ладанюка, доц.

Смітюх Я.В.

Виконавець науково-дослідних
робіт, аспірант

Омельченко О.С.