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

Фізико-математичний факультет
Кафедра математичного аналізу та теорії ймовірності

«На правах рукопису»
УДК 51-77

До захисту допущено:
Завідувач кафедри
Олег Клесов
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Страхова та фінансова математика»

зі спеціальності 111 «Математика»

на тему: «Застосування штучного інтелекту та Wolfram для автоматичного створення шаблонів тестів з вищої математики»

Виконав:
студент ІІ курсу, групи ОМ-41мн
Кайдалов Семен Вікторович _____

Науковий керівник:
Доцент Київського Політехнічного Інституту іменя Ігоря Сікорського,
Круглова Наталя Володимирівна _____

Рецензент:
старший науковий співробітник відділу механіки повзучості інституту механіки ім. С П. Тимошенка НАН України,
Ушакова Віра Сергіївна _____

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

Київ – 2026 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-математичний факультет
Кафедра математичного аналізу та теорії ймовірності

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою

Спеціальність – 111 «Математика»

Освітньо-наукова програма «Страхова та фінансова математика»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Олег Клесов

Завдання
на магістерську дисертацію студенту
Кайдалов Семен Вікторович

1. Тема дисертації «Застосування штучного інтелекту та Wolfram для автоматичного створення шаблонів тестів з вищої математики», науковий керівник дисертації кандидат фізико-математичних наук, доцент Круглова Наталія Володимирівна, затверджені наказом по університету від «31» березня 2026 р. №1340-с
2. Термін подання студентом дисертації 15.05.2026
3. Об’єктом дослідження є процес автоматизованого формування тестових завдань з вищої математики.
4. Предметом дослідження є методи використання штучного інтелекту та Wolfram Mathematica для генерації, математичні методи аналізу якості тестових завдань.
5. Перелік завдань, які потрібно виконати:
    1) Ознайомитися з літературою та дослідити сучасні методи автоматичної генерації тестових завдань.

2) Дослідити можливості систем штучного інтелекту для генерації математичних задач та текстового наповнення тестів.
3) Опанувати можливості Wolfram Mathematica для символьних обчислень та автоматичної перевірки математичних виразів.
4) Розробити шаблони тестових завдань з теми «Функції багатьох змінних».
5) Реалізувати генерацію параметризованих математичних задач у середовищі Wolfram Mathematica.
6) Реалізувати формування Moodle XML-файлів для автоматичного імпорту тестів у систему Moodle.
7) Дослідити методи психометричного аналізу тестів на основі Класичної теорії тестів.
8) Провести апробацію автоматично згенерованих тестових завдань та виконати статистичний аналіз результатів студентів.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: 19 слайдів.

7. Дата видачі завдання 03.02.2026.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ознайомитися з літературою та дослідити сучасні методи автоматичної генерації тестових завдань	03.02.2026 – 10.02.2026	виконано
2.	Дослідити можливості систем штучного інтелекту та Wolfram	11.02.2026 – 15.02.2026	виконано

	Mathematica для генерації математичних задач		
3.	Розробити шаблони тестових завдань з теми «Функції багатьох змінних»	16.02.2026 – 28.02.2026	виконано
4.	Реалізувати генерацію параметризованих задач та автоматичне формування відповідей у Wolfram Mathematica	01.03.2026 – 31.03.2026	виконано
5.	Реалізувати формування Moodle XML-файлів для імпорту тестових завдань	01.04.2026 – 20.04.2026	виконано
6.	Дослідити методи психометричного аналізу тестів на основі Класичної теорії тестів	20.04.2026 – 30.04.2026	виконано
7.	Провести апробацію тестових завдань на вибірці студентів РТФ	01.05.2026 – 10.05.2026	виконано
8.	Провести статистичний аналіз результатів апробації та оцінити валідність і надійність тестових завдань	10.05.2026 – 13.05.2026	виконано

Студент Кайдалов Семен Вікторович

Науковий керівник Круглова Наталія Володимирівна

Реферат

Магістерська дисертація: 96 сторінок, 15 першоджерел, 19 слайдів презентації, електронні додатки. Робота складається зі вступу, шести розділів, висновків та списку використаної літератури.

У дисертаційній роботі досліджуються методи автоматичного створення тестових завдань з вищої математики із застосуванням систем штучного інтелекту та Wolfram Mathematica. Основну увагу приділено генерації параметризованих математичних задач, автоматичному формуванню Moodle XML-файлів та психометричному аналізу якості тестових завдань.

Метою дисертаційного дослідження є розробка системи автоматичного створення шаблонів тестів з вищої математики на основі штучного інтелекту та Wolfram Mathematica з подальшою перевіркою валідності та надійності сформованих тестових завдань.

Об’єктом дослідження є процес автоматизованого формування тестових завдань з вищої математики.

Предметом дослідження є методи використання штучного інтелекту, Wolfram Mathematica та психометричного аналізу для генерації, перевірки та оцінювання тестових завдань.

Перший розділ містить огляд літератури щодо сучасних методів автоматичної генерації тестових завдань, систем штучного інтелекту, великих мовних моделей та систем комп’ютерної алгебри.

Другий розділ містить опис архітектури розробленої системи, методи генерації параметризованих математичних задач у середовищі Wolfram Mathematica та алгоритми автоматичного формування Moodle XML-файлів.

Третій розділ містить математичні основи психометричного аналізу тестів, методи Класичної теорії тестів, показники складності, дискримінативності, надійності та валідності тестових завдань.

Четвертий розділ містить реалізацію інтеграції систем штучного інтелекту для автоматичного формування текстового наповнення тестових завдань, генерації дистракторів та автоматичної перевірки математичної коректності задач.

П’ятий розділ містить результати апробації тестових завдань на вибірці студентів РТФ, статистичний аналіз результатів тестування, оцінювання психометричних характеристик тесту та перевірку валідності й надійності сформованих тестових завдань.

Ключові слова: штучний інтелект, Wolfram Mathematica, Moodle XML, автоматична генерація тестів, тести з вищої математики, психометричний аналіз, Класична теорія тестів, точково-бісеріальна кореляція, коефіцієнт Кронбаха, дискримінативність, валідність тесту, генерація математичних задач, великі мовні моделі.

Abstract

Master's thesis: 96 pages, 15 primary sources, 19 presentation slides, electronic appendices. The thesis consists of an introduction, six chapters, conclusions, and a list of references.

This thesis explores methods for the automatic generation of higher mathematics test items using artificial intelligence systems and Wolfram Mathematica. The primary focus is on the generation of parameterized mathematical problems, the automatic creation of Moodle XML files, and the psychometric analysis of test item quality.

The aim of the dissertation research is to develop a system for the automatic generation of higher mathematics test templates based on artificial intelligence and Wolfram Mathematica, followed by verification of the validity and reliability of the generated test items.

The object of the study is the process of automated generation of higher mathematics test items.

The subject of the study is the methods of using artificial intelligence, Wolfram Mathematica, and psychometric analysis for the generation, verification, and evaluation of test items.

The first chapter contains a literature review on modern methods of automatic test item generation, artificial intelligence systems, large language models, and computer algebra systems.

The second chapter describes the architecture of the developed system, methods for generating parameterized mathematical problems in the Wolfram Mathematica environment, and algorithms for automatically generating Moodle XML files.

Chapter 3 covers the mathematical foundations of psychometric test analysis, methods of classical test theory, and measures of difficulty, discrimination, reliability, and validity of test items.

The fourth chapter describes the implementation of artificial intelligence systems for the automatic generation of test item text, the creation of distractors, and the automatic verification of the mathematical correctness of the items.

The fifth chapter presents the results of testing the test items on a sample of students at the RTF, statistical analysis of the test results, evaluation of the test's psychometric characteristics, and verification of the validity and reliability of the generated test items.

Keywords: artificial intelligence, Wolfram Mathematica, Moodle XML, automatic test generation, higher mathematics tests, psychometric analysis, classical test theory, point-biserial correlation, Cronbach's alpha, discriminant validity, test validity, generation of mathematical problems, large language models.

ЗМІСТ

Вступ

Актуальність теми

Розвиток цифрових технологій та впровадження систем дистанційного навчання зумовлюють необхідність удосконалення підходів до організації освітнього процесу викладання вищої математики. Головною складовою дистанційного навчання є контроль і оцінювання знань студентів, що потребує створення великої кількості якісних, варіативних та методично обґрунтованих тестових завдань.

Традиційні методи створення тестових матеріалів передбачають значні витрати часу викладача, а також обмежену кількість варіантів завдань. Ручне створення тестів часто призводить до технічних помилок, обмеженої варіативності та можливості списування. У зв’язку з цим актуальним є впровадження автоматизованих систем генерації тестових завдань, які дозволяють:

- забезпечити потрібну кількість варіантів задач за рахунок параметризації;
- підвищити об’єктивність контролю знань;
- зменшити навантаження на викладача;
- створювати великі банки тестових завдань.

Обчислювальні системи, такі як Wolfram Language, забезпечують правильність математичних розрахунків, перевірку коректності умов задач та автоматичне отримання правильних відповідей. Це дозволяє створювати математично коректні задачі.

Розвиток великих мовних моделей (LLM) дозволяє автоматизувати рутинні елементи освітнього процесу. Системи штучного інтелекту можуть використовуватися для генерації текстових формулювань задач,

створення різних варіантів подання матеріалу, формування пояснень до задач та розробки типових помилкових відповідей.

Таким чином, актуальність теми дослідження обумовлена необхідністю поєднання:

- генерації математично правильних задач (за допомогою Wolfram);
- гнучкого створення умов та пояснень до задач (за допомогою штучного інтелекту);
- імпортування в Moodle.

Розробка системи автоматичного створення шаблонів тестів з вищої математики, яка поєднує зазначені компоненти, є важливим кроком оптимізації роботи викладача та впровадження сучасних цифрових технологій у навчальний процес.

Мета і завдання дослідження

Метою даного дослідження є розробка та обґрунтування підходу до автоматизованого створення шаблонів тестових завдань з вищої математики на основі поєднання можливостей системи Wolfram Language та сучасних методів штучного інтелекту, з подальшим імпортом категорій у систему дистанційного навчання Moodle.

Реалізація цієї мети передбачає створення системи, яка забезпечує:

- генерацію математично правильних задач із параметризацією;
- отримання правильних відповідей;
- формування різноманітних коректних формулювань завдань;
- підтримку різних форматів тестових питань.

Завдання дослідження

Для досягнення поставленої мети в роботі послідовно розв’язуються кілька взаємопов’язаних завдань. Насамперед проводиться аналіз сучасних підходів до автоматичної генерації тестових завдань з вищої математики, зокрема досліджуються методи параметризації задач, шаблонні підходи до їх побудови та можливості їх реалізації в системах комп’ютерної математики і середовищах дистанційного навчання.

Окрему увагу приділено вивченню можливостей використання Wolfram Language як інструмента математичної генерації. У цьому контексті розглядаються питання автоматичного формування математичних моделей задач, обчислення правильних відповідей, а також перевірки коректності згенерованих варіантів і уникнення вироджених випадків.

Досліджуються можливості застосування великих мовних моделей для автоматизації текстової складової навчальних матеріалів. Йдеться про генерацію формулювань задач, створення пояснень розв’язків і формування типових дистракторів, що можуть використовуватись у тестових завданнях.

На основі проведеного аналізу розробляється архітектура системи автоматичної генерації тестів, що поєднує математичний модуль на базі Wolfram, модуль текстової генерації із використанням штучного інтелекту, а також засоби формування та імпорту тестових завдань у Moodle.

Практична частина роботи передбачає реалізацію шаблонів задач з теми «Функції багатьох змінних»: задач на знаходження частинних похідних, дослідження функцій на екстремум, знаходження похідної за напрямом, побудову дотичної площини та нормалі, а також задач на умовний екстремум і знаходження мінімальних та максимальних значень функцій у обмежених областях. Для кожного типу задач забезпечується генерація множини варіантів із контролем коректності та рівня складності.

Завершальним етапом є формування тестових матеріалів у форматі Moodle XML та оцінювання ефективності запропонованого підходу з точки зору коректності генерації, варіативності завдань і можливості їх практичного використання в освітньому процесі.

Об’єкт і предмет дослідження

Об’єктом дослідження є процес автоматизації створення та використання тестових завдань з вищої математики у системах дистанційного навчання. Цей процес охоплює формування змісту задач, їх параметризацію, забезпечення математичної коректності, а створення файлів категорій тестових задач.

Предметом дослідження є методи, моделі та програмні засоби автоматичної генерації шаблонів тестових завдань з вищої математики. Розглянуто параметризовані математичні моделі задач, алгоритми їх генерації та перевірки, засоби інтеграції отриманих задач у системи дистанційного навчання.

Предметом дослідження є задачі із розділу «Функції багатьох змінних», для яких розробляються шаблони, що дозволяють здійснювати генерацію завдань із гарантованою коректністю відповідей.

Методи дослідження

В роботі використано сукупність теоретичних, математичних та програмно-інструментальних методів дослідження, що забезпечують обґрунтованість запропонованого підходу до автоматизованого створення тестових завдань з вищої математики.

Теоретичною основою дослідження є методи математичного аналізу та аналітичної геометрії, зокрема апарат функцій багатьох змінних. На основі цих методів здійснюється побудова параметризованих математичних моделей задач, визначення їх структурних властивостей та умов коректності.

Для формування варіативних тестових завдань застосовано метод параметризації, який полягає у введенні змінних коефіцієнтів у математичні моделі задач із подальшим контролем допустимих значень параметрів. Це дозволяє отримувати множину різних варіантів задач одного типу при збереженні їх методичної еквівалентності.

Обчислення правильних відповідей, перевірка коректності згенерованих задач та виключення вироджених випадків здійснюються із застосуванням Wolfram Language.

Для автоматизації текстової складової тестових завдань застосовано методи обробки природної мови, реалізовані за допомогою моделей штучного інтелекту. Зокрема, використовуються підходи до генерації тексту, що дозволяють формувати різноманітні формулювання задач, пояснення розв’язків та типові помилкові відповіді. При цьому математичне ядро задач залишається незалежним від мовної моделі, що забезпечує коректність результатів.

Формування тестових матеріалів здійснюється з використанням методів програмної генерації структурованих даних. Зокрема, застосовується генерація XML-документів, що відповідають формату імпорту в систему дистанційного навчання Moodle. Це дозволяє інтегрувати згенеровані завдання у навчальне середовище.

У процесі дослідження також використано методи алгоритмізації та програмної реалізації, що забезпечують побудову модульної структури системи генерації тестів. Це включає розробку алгоритмів генерації параметрів, обчислення відповідей, формування варіантів задач та їх подальшого експорту.

Таким чином, використаний комплекс методів забезпечує як математичну обґрунтованість, так і практичну реалізацію системи автоматизованого створення тестових завдань з вищої математики.

I. Огляд літератури

1.1. Генерація завдань

Дослідження останніх років демонструють, що штучний інтелект уже придатний для масштабування банків тестових завдань, попереднього контролю якості та грубого прогнозування складності. Водночас проблема безпосередньої генерації файлів формату Moodle XML майже не розглядалась, а вивчалися суміжні питання: генерація завдань, психометрична перевірка та калібрування банку питань [1, 2, 3, 4, 5].

Для задачі побудови XML-файлів категорій у Moodle це означає наступне: штучний інтелект і Wolfram доцільно розглядати як конвеєр по створенню питань, а валідацію якості питань — виносити в окремий психометричний контур, зокрема в середовищі R, через класичну теорію тестів і аналіз латентної структури [6, 7, 8, 9].

Сильна сторона сучасних AI-підходів полягає не лише у пришвидшенні написання окремих тестових питань, а в створенні потокового виробництва завдань. В роботі [1] описано повний конвеєр, де модель генерує тексти, питання та варіанти відповідей, далі працюють автоматичні фільтри, а потім здійснюється перевірка експертами. Після багатокрокового відбору автори отримали операційний банк завдань із прийнятними показниками складності, диференціаціюючої здатності та локальної залежності. Це важливий прецедент для систем, де банк має поповнюватися серійно, а не вручну.

У роботі [2] GPT використано як генератор завдань на розуміння прочитаного усередині більш структурованої схеми автоматизованої генерації завдань (AIG). Основна ідея не в тому, що LLM сама гарантує якість, а в тому, що генеративна модель може підсилити шаблонний або когнітивно спроектований процес створення завдань. Це означає, що AI

варто ставити на етап формування чернеток, варіантів формулювання, дистракторів і групування питань у змістові серії, але не на етап остаточного затвердження банку [2, 4].

Робота [5] підтверджує цей висновок на ширшій вибірці даних: великі мовні моделі (LLM) показали гнучкість на стадіях попередньої генерації, генерації та постгенерації, але в дослідженні мало висвітлено інформацію про перевірку валідності, надійності та немає психометричного обґрунтування.

Найперспективнішим напрямком досліджень останніх років є те, що AI використовується не лише для генерації, а й для попереднього відсіву слабких завдань. У [3] мовні моделі, налаштовані за інструкціями, застосовано як фільтр якості для автоматично згенерованих питань з пропусками. Це важливий крок між генерацією і повним запуском: система може спочатку розділити завдання на ймовірно добрі, сумнівні та слабкі, а вже потім передавати їх на експертну перевірку. Інший напрямок робіт зосереджений на прогнозуванні параметрів завдання за його текстом. У [10] запропоновано багатокрокову схему обробки, де мовна модель не тільки обробляє текст, а й оцінює складність та час відповіді. У [12] складність прогнозується через відповіді віртуальних учасників тестування із подальшим призв’язуванням тестів до людської шкали. У [11] тонке налаштування мовних моделей і підходи на основі латентних ознак дають корисні попередні оцінки складності завдань ще до повного польового тестування.

В усіх цих роботах ключовою є ідея такого типу: перед тим як експортувати згенеровані питання у Moodle XML, доцільно автоматично оцінити їх складність, очікуваний час відповіді та ймовірний ризик низької дискримінації [10, 12, 11].

Згенерований банк завдань не можна вважати валідним лише тому, що питання граматично правильні або тематично доречні. У [6] було змодельовано результати тестування для нових тестових запитань для обчислення показників класичної теорії тестів та IRT, включно з обчисленням частки правильних відповідей, кореляції завдань із загальним результатом, поведінкою дистракторів та перевіркою багатовимірності. Проте автор відмічає, що такі оцінки лишаються лише наближенням до реального польового тестування, а деякі завдання мають нестабільні оцінки диференціації.

У роботі [7] підтверджена ця ідея. Порівняння тестів, створених людьми та згенерованих системами штучного інтелекту, показало, що навіть за схожої зовнішньої форми версія, створена алгоритмом, може змінювати приховану багатовимірність, порядок складності завдань і профіль інформаційної функції тесту. Тому для банку, який згодом буде організовано за категоріями у Moodle, принципово важливо перевіряти не лише якість окремих запитань, а й те, чи весь банк справді вимірює задуману структуру знань, а не випадкову суміш локальних підтематичних блоків або стилістичних шаблонів [7, 9].

Класична теорія тестів (CTT) лишається найзручнішим першим етапом перевірки якості тестових питань. У роботі [8] показано, що така перевірка питань дозволяє визначити, які питання залишити, які переписати, а які відкинути.

Використовують кілька ключових індексів для оцінки якості тестування.

Індекс складності показує частку правильних відповідей. Занадто легкі або занадто важкі завдання зазвичай не є оптимальними для робочого

банку питань, тому цей показник застосовують для первинного розподілу завдань за категоріями складності [8].

Індекс диференціації відображає здатність завдання відділяти сильніших учасників від слабших. Низькі значення сигналізують про двозначність, шум або невдалий дизайн дистракторів. Цей індекс допомагає відсіяти слабкі згенеровані завдання [8].

Ефективність дистракторів показує, чи обираються респондентами неправильні відповіді. Якщо дистрактори нефункціональні, завдання стає занадто легким і втрачає здатність диференціювати учасників. Перевірка цього показника важлива для оцінки якості структури завдань із множинним вибором [8, 6].

Точково-бісеріальна кореляція демонструє, наскільки окреме запитання узгоджується з тестом у цілому. Низький зв’язок може означати, що завдання випадає зі структури знань, яку тест має вимірювати. Це допомагає виявляти питання, що не відповідають категорії [6].

Надійність характеризує узгодженість тесту. Якщо вона низька, це свідчить про розмитість змісту або слабку підбірку завдань. Такий показник застосовують для перевірки якості всієї категорії чи підкатегорії [8].

Таким чином, кожен індекс має свою роль: від первинного розподілу за складністю до перевірки узгодженості всієї категорії, і разом вони забезпечують системну оцінку якості банку завдань.

CTT має особливу цінність з трьох причин: вона проста для масового застосування після пілотування; добре поєднується з ручним редагуванням згенерованих питань; дозволяє перевіряти не лише весь тест, а й окремі категорії, які потім імпортуються у Moodle як логічні блоки.

У практичному сенсі це означає, що після генерації й експорту XML-файл категорій не повинен вважатися фінальним об’єктом. Остаточним буде лише той банк питань, для якого на пілотних даних пораховано принаймні складність, диференціююча здатність, ефективність дистракторів та надійність [8, 6].

1.2. Експлораторний факторний аналіз для перевірки структури банку

Експлораторний факторний аналіз (EFA) потрібен не для заміни класичного аналізу завдань, а для відповіді на інше запитання: чи справді сукупність згенерованих завдань організовується у ті приховані категорії, які були задумані. Саме тут банки, створені за допомогою штучного інтелекту, мають особливий ризик. Генератор може витримувати спільну тему, але непомітно змішувати різні когнітивні операції, рівні складності або дрібні підструктури знань [7, 4].

Тобто EFA допомагає перевірити не лише якість окремих завдань, а й те, чи вся система вимірює саме ту структуру знань, яка була закладена, а не випадкову суміш елементів.

У [9] показано практичну логіку такого аналізу: спершу перевіряється придатність даних до факторизації, далі обирається факторне рішення на основі власних значень, графіка scree pattern та змістовних навантажень, а потім інтерпретується зміст отриманих факторів. Хоча дослідження стосується оцінювання на основі груп завдань (тестлетів), його висновок має ширше значення: емпірична структура банку завдань не завжди збігається з наміром розробника. В [7] демонструється те саме вже на матеріалі тестів, створених за допомогою штучного інтелекту: зміна дизайну запиту впливала на багатовимірність та профіль інформаційної функції тесту. Експлораторний факторний аналіз

допомагає зрозуміти не окремі завдання, а всю структуру банку питань. Він перевіряє кілька речей.

По-перше, кількість факторів: чи не утворилися приховані підшкали всередині однієї категорії. Якщо так – доведеться перегрупувати або розділити категорії у Moodle [9, 7].

По-друге, факторні навантаження: які саме завдання справді вимірюють одну й ту саму компетентність. Це дає підказку, які питання варто перенести в інші категорії або взагалі прибрати, якщо вони лише створюють шум [9].

Третє – частка поясненої дисперсії. Вона показує, наскільки чітко організований банк. Якщо структура розмита, підбанк може виявитися нестабільним [9].

І нарешті, розходження між задумом і даними. Тут ми бачимо, чи відповідає згенерована структура навчальній логіці. Якщо ні – треба переглянути запити, шаблони або правила генерації [7, 4].

Отже, класичний аналіз завдань відповідає на питання «чи добре працює конкретне завдання», а факторний аналіз – «чи правильно організований увесь банк». Для категорій, створених за допомогою штучного інтелекту у Moodle, ці два підходи варто поєднувати, а не ставити один проти одного.

1.3 Використання Wolfram Language для генерації.

Wolfram Language підтримує роботу з XML через спеціальні об’єкти (XMLElement і XMLObject), а також імпорт, трансформацію й експорт XML-документів у стандартному форматі. Офіційна документація Moodle описує XML як формат для імпорту запитань і зазначає, що категорії

можна задавати безпосередньо з файлу під час завантаження у банк запитань. Це робить Wolfram природним середовищем для автоматичної збірки XML-файлу, навіть якщо психометрична перевірка здійснюється окремо в R.

Практичний робочий процес цього дослідження виглядає так:

- у Wolfram створюється схема категорій, підкатегорій і типів запитань;
- система генерує чернетки формулювань завдань, варіантів відповіді, правильних рішень і, за потреби, пояснень;
- у Wolfram виконуються програмні перевірки: чи всі поля заповнені, чи правильно задано формат відповідей, чи унікальні назви, чи логічно побудовані категорії, а також базові фільтри за правилами;
- з тих самих структур даних формується XML-файл із категоріями та запитаннями для Moodle;
- після пілотування відповіді аналізуються в R за класичними показниками та аналізом структури, а результати повертаються у цикл редагування банку.

У літературі наведено аргументи в підтримку генерації, попереднього відбору за якістю та попередньої оцінки складності, але не стверджується, що сам експорт у формат XML автоматично означає готовий якісний банк. Wolfram виступає засобом складання й трансформації, а R — інструментом емпіричної перевірки [3, 6, 10, 11].

У сучасних дослідженнях є кілька важливих прогалин. Є багато робіт про саму генерацію та частковий контроль якості [1, 2, 3, 5], але майже немає повних досліджень, які простежують увесь шлях — від генерації до стабільного робочого банку.

Також розглядається створення завдань і прогнозування їхньої складності [10, 12, 11], але існує мало досліджень, де штучний інтелект будує саме педагогічно осмислену й психометрично перевірену систему категорій.

Хоча є технічна документація Moodle, але майже немає академічних робіт, де експорт категорій у форматі XML розглядається як основний об’єкт аналізу.

Ще одна прогалина стосується оцінювальних конвеєрів на основі Wolfram. Технічно Wolfram добре підходить для створення XML-структур і символьних трансформацій, але майже немає освітньо-психометричних публікацій, де його описано як платформу для повного робочого процесу.

І нарешті, перевірка валідація категорій, згенерованих штучним інтелектом. Є приклади використання класичних показників, факторного аналізу та моделей IRT [6, 7, 8, 9], але мало робіт, які прямо поєднують генерацію, побудову категорій і перевірку після імпорту.

Таким чином, наукова література вже має напрацювання щодо генерації та окремих етапів контролю якості, але бракує комплексних досліджень, які з’єднують усі ці кроки в єдиний, стабільний і педагогічно осмислений процес.

Отже, штучний інтелект уже придатний для генерації великого масиву тестових завдань, попереднього контролю якості, прогнозування складності і часткової перевірки банку питань [1, 3, 10, 5]. Проте наукові джерела не підтримують автоматизм у сенсі повної довіри до генерації.

Найкраще підтверджена модель роботи є багатошаровою: AI генерує і попередньо сортує питання, Wolfram збирає структурований XML-файл

для Moodle, а R виконує психометричну валідацію через CTT і аналіз латентної структури [6, 7, 8, 9].

Це означає, що науково обґрунтований підхід полягає не в заміні розробника одним генератором, а в побудові конвеєра, де генерація, XML-формування і статистична валідація є трьома окремими, але пов'язаними шарами. Саме таке поєднання найкраще узгоджується і з сучасною літературою, і з практикою створення робочого банку тестових питань.

II. Архітектура розробленої системи

2.1. AI-генератор: принцип використання в розробленій системі

AI-генератор у межах розробленої системи доцільно розглядати не як самостійний інструмент створення математично завершених задач, а як допоміжний модуль, що відповідає за мовну, методичну та структурну складову тестового завдання. Його основне призначення полягає у формуванні текстових елементів задачі, адаптації формулювань до навчального контексту, створенні пояснень і підготовці допоміжних матеріалів, тоді як математична коректність забезпечується Wolfram-модулем.

Такий розподіл функцій є принциповим, оскільки великі мовні моделі здатні генерувати змістовно переконливі формулювання, але не завжди гарантують точність математичних обчислень. Тому в архітектурі системи AI-генератор не виконує остаточне обчислення відповідей і не є джерелом істинності математичного результату. Його вихідні дані використовуються лише після формальної перевірки або у тих частинах завдання, де потрібна природномовна генерація.

У запропонованій системі AI-генератор працює у режимі контрольованої генерації, коли мовна модель обмежена заданими параметрами задачі. На вхід він отримує опис типу задачі у вигляді структурованих даних (JSON), математичну тему, очікуваний рівень складності, перелік обов’язкових елементів відповіді та дані, сформовані іншими модулями системи. Наприклад, для задачі на дослідження функції двох змінних на екстремум AI-генератор отримує інформацію про те, що студент має знайти стаціонарну точку, обчислити другі частинні похідні та визначити тип точки за критерієм Гессе. На основі цих даних він формує зрозуміле формулювання задачі, інструкцію до виконання та, за потреби, коротке пояснення методу.

Основні функції AI-генератора в системі можна поділити на кілька напрямів. По-перше, він забезпечує створення формулювання задачі у природній мові. Це дозволяє уникнути надмірно одноманітних умов при генерації великої кількості варіантів. Наприклад, одна й та сама математична структура може бути подана як «знайти стаціонарну точку функції», «дослідити функцію на екстремум» або «визначити характер критичної точки». При цьому математична модель залишається незмінною, але текстове подання варіюється.

По-друге, AI-генератор може використовуватися для формування методичних пояснень. У навчальному режимі після виконання завдання студент може отримати короткий коментар, у якому пояснюється застосоване правило або алгоритм розв'язання. Наприклад, для задачі на неявно задану функцію AI-генератор може сформувати пояснення, що частинні похідні знаходяться за формулами

$$z_x = -\frac{F_x}{F_z}, \quad z_y = \frac{F_y}{F_z},$$

а для задачі на екстремум — що тип стаціонарної точки визначається за знаком визначника матриці Гессе. Однак самі числові значення похідних або координат точки в цьому поясненні підставляються не мовною моделлю, а Wolfram-модулем.

По-третє, AI-генератор може брати участь у створенні типових помилкових відповідей для питань із вибором відповіді. Такі відповіді не мають бути випадковими: вони повинні відповідати типовим помилкам студентів. Наприклад, у задачах на екстремум типовими помилками є неправильне визначення знака визначника Гессе, ототожнення максимуму із сідловою точкою або ігнорування умови $D > 0$. Для задач на неявне диференціювання поширеними помилками є пропуск знака «мінус» у

формулі або ділення на F_x замість F_z. AI-генератор може формувати такі помилки, а система надалі перевіряє їх відповідність заданому типу задачі.

По-четверте, AI-генератор може застосовуватися для класифікації або уточнення рівня складності завдання. Складність у такій системі визначається не лише числовими коефіцієнтами, а й структурними характеристиками задачі: наявністю змішаних членів, кількістю проміжних кроків, видом функції, кількістю полів відповіді та необхідністю пояснення методу. AI-генератор може допомагати сформулювати опис рівня складності, однак фактичні параметри складності задаються алгоритмічно в генераторі варіантів і перевіряються математичним модулем.

Важливою особливістю використання AI-генератора є робота з промптами. Для кожного типу задачі формується службовий опис, у якому задаються межі дії мовної моделі. Наприклад, промпт може містити інструкцію: сформулювати умову задачі українською мовою, не змінювати математичні дані, не обчислювати відповіді самостійно, використовувати лише ті значення, які передані Wolfram-модулем. Такий підхід зменшує ризик появи некоректних тверджень і забезпечує узгодженість між текстом умови та математичними відповідями.

Узагальнену роботу AI-генератора можна подати так. На вхід надходить структурований опис задачі: тема, тип задачі, математична модель, рівень складності, перелік полів відповіді та результати обчислень. На виході формується текст умови, інструкція для студента, за потреби пояснення розв’язання та варіанти неправильних відповідей. Далі ці дані передаються до генератора Moodle-формату, який вбудовує їх у XML-структуру тестового питання.

Таким чином, AI-генератор у розробленій системі виконує функцію інтелектуального текстового й методичного модуля. Його використання дозволяє підвищити варіативність формулювань, зробити тестові завдання

більш природними для сприйняття студентом, автоматизувати створення пояснень і підтримати навчальне структурування матеріалу. Водночас обмеження ролі AI-генератора лише мовною та методичною складовою дозволяє зберегти математичну строгість системи, оскільки всі обчислення та перевірки виконуються формальним Wolfram-модулем.

2.2. Wolfram-модуль: принципи побудови та функціонування

Wolfram-модуль у розробленій системі виступає як центральне обчислювальне ядро, що забезпечує формування математично коректних варіантів задач, обчислення правильних відповідей і перевірку їхньої валідності. Його використання обумовлене необхідністю гарантувати строгість математичних перетворень. На відміну від мовних моделей, які працюють із текстовими структурами, Wolfram-модуль оперує формальними математичними об'єктами, що дозволяє уникнути неоднозначностей.

Функціонування модуля базується на принципі параметризованої генерації задач. Замість того щоб формувати окремі приклади вручну, система використовує узагальнені математичні моделі, які містять змінні параметри. Однією з таких моделей є квадратична функція двох змінних:

$$f(x, y) = ax^2 + by^2 + cxy + dx + ey + g.$$

У цьому випадку коефіцієнти a, b, c, d, e, g не задаються наперед, а генеруються випадково в заданих межах із подальшою перевіркою умов коректності. Важливою особливістю є те, що генерація параметрів здійснюється не довільно, а конструктивно, тобто з урахуванням бажаних властивостей функції. Зокрема, для задач на дослідження екстремуму спочатку обирається точка (x_0, y_0), яка має бути стаціонарною, після чого підбираються такі значення параметрів, щоб ця умова виконувалася.

Стаціонарна точка визначається з умов:

$$f_x = 0;\ f_y = 0,$$

де перші частинні похідні мають вигляд:

$$f_x = 2ax + cy + d;$$

$$f_y = cx + 2by + e.$$

Підставляючи координати точки (x_0, y_0) у ці вирази, модуль обчислює коефіцієнти d та e таким чином, щоб рівності виконувалися тотожно. Це дозволяє гарантувати, що згенерована функція дійсно має стаціонарну точку в заданому місці, що є суттєвою перевагою конструктивного підходу порівняно з випадковою генерацією.

Після визначення стаціонарної точки система переходить до аналізу її типу. Для цього обчислюються другі частинні похідні функції та формується визначник матриці Гессе:

$$D = f_{xx}f_{yy} - f_{xy}^2.$$

Саме значення цього визначника, а також знак другої похідної f_{xx}, дозволяють класифікувати стаціонарну точку як точку мінімуму, максимуму або сідлову. У межах Wolfram-модуля ці обчислення виконуються автоматично, що виключає можливість помилки при визначенні правильного типу екстремуму. Важливо, що модуль також перевіряє умови невиродженості квадратичної форми, зокрема виконується перевірка:

$$4ab - c^2 \neq 0,$$

що гарантує коректність застосування критерію Гессе.

Аналогічний підхід використовується і для задач на неявно задані функції. У цьому випадку базовою моделлю є рівняння:

$$F(x, y, z) = 0,$$

яке задає залежність між змінними. Для забезпечення коректності задачі система формує функцію з параметрами таким чином, щоб задана точка (x_0, y_0, z_0) задовольняла це рівняння. Це досягається шляхом підбору

вільного члена функції, що дозволяє уникнути ситуацій, коли студенту пропонується задача без розв’язку або з некоректними умовами.

Далі Wolfram-модуль виконує символьне диференціювання, обчислюючи частинні похідні функції:

$$F_x = A + Dy + Ez;$$

$$F_y = B + Dx + Gz;$$

$$F_z = C + Ex + Gy.$$

Після цього визначаються похідні функції $z(x, y)$за формулами:

$$\partial z/\partial x = -F_x/F_z;$$

$$\partial z/\partial y = -F_y/F_z.$$

Ключовим моментом є перевірка умови невиродженості:

$$F_z \neq 0$$

у заданій точці. Якщо ця умова не виконується, варіант задачі відкидається і генерується новий. Таким чином забезпечується коректність постановки задачі та можливість її розв’язання стандартними методами.

Важливою характеристикою Wolfram-модуля є його здатність виконувати не лише обчислення, а й логічну перевірку отриманих результатів. У процесі генерації задач система автоматично відкидає варіанти, які не відповідають заданим умовам, наприклад, коли визначник Гессе дорівнює нулю або коли похідні набувають невизначених значень.

Отримані результати — значення параметрів, координати точок, похідні, типи екстремумів — формуються у структурованому вигляді та

передаються до інших компонентів системи. Зокрема, ці дані використовуються AI-генератором для формування тексту задачі, а також модулем генерації Moodle-формату для створення тестових питань із правильними відповідями.

Таким чином, Wolfram-модуль забезпечує формалізацію математичної частини всієї системи. Його використання дозволяє перейти від інтуїтивного або випадкового створення задач до строго контрольованого процесу генерації, у якому кожен варіант має гарантовану математичну коректність. Саме це робить можливим використання автоматично згенерованих задач у реальному освітньому процесі без втрати якості навчального контролю.

2.3. Генератор варіантів задач: принципи побудови та функціонування

Генератор варіантів задач у розробленій системі виконує функцію організації множини тестових завдань на основі математичних шаблонів, сформованих Wolfram-модулем, та текстових структур, підготовлених AI-генератором. Якщо Wolfram-модуль відповідає за коректність окремого математичного екземпляра задачі, то генератор варіантів забезпечує систематичне створення набору таких екземплярів із урахуванням варіативності, рівня складності та дидактичної доцільності.

Основною ідеєю роботи генератора є перехід від одиничного шаблону задачі до множини взаємопов'язаних варіантів, які належать до одного типу, але відрізняються параметрами, структурою або формою подання. При цьому зберігається методична еквівалентність задач, тобто всі варіанти перевіряють однакові знання та навички. Наприклад, для задачі на дослідження функції двох змінних на екстремум використовується загальна модель

$$f(x, y) = ax^2 + by^2 + cxy + dx + ey + g,$$

але конкретні значення коефіцієнтів змінюються від варіанта до варіанта.

Генератор варіантів працює на основі заданого шаблону, який включає математичну модель, обмеження на параметри та вимоги до результату. На кожному кроці генерації він ініціює Wolfram-модуль для отримання нового набору параметрів і відповідних правильних відповідей. Таким чином формується послідовність незалежних варіантів, кожен з яких є повноцінною задачею. Важливо, що цей процес є керованим: викладач або система може задавати кількість варіантів, допустимі діапазони параметрів та умови відбору.

Однією з ключових характеристик генератора є забезпечення варіативності без втрати якості. Це означає, що зміна параметрів не повинна призводити до появи тривіальних або, навпаки, надмірно складних задач. Для цього використовується система обмежень, яка контролює значення параметрів і властивості функції. Наприклад, для задач на екстремум контролюється знак визначника

$$D = f_{xx}f_{yy} - f_{xy}^2,$$

а також виконується умова невиродженості квадратичної форми. Якщо згенерований варіант не відповідає заданим критеріям, він відкидається, і система формує новий.

Важливим аспектом роботи генератора є управління складністю задач. Складність визначається не лише числовими значеннями коефіцієнтів, а й структурними особливостями задачі. Наприклад, у простішому варіанті може бути відсутній змішаний член cxy, тоді як у складнішому варіанті він присутній і ускладнює систему рівнянь для знаходження стаціонарної точки. Крім того, складність може змінюватися за рахунок вибору точки, у якій виконуються обчислення, або за рахунок формату відповіді.

Ще однією важливою функцією генератора варіантів є забезпечення унікальності завдань. Оскільки параметри генеруються випадковим або псевдовипадковим чином у заданих межах, існує ймовірність повторення варіантів. Для уникнення цього система може використовувати механізми перевірки вже згенерованих задач, порівнюючи їхні параметри або ключові характеристики. У разі виявлення збігу варіант відкидається і генерується новий.

Генератор також відповідає за формування повного набору задач для контрольної роботи або тесту. Наприклад, якщо необхідно створити

контрольну роботу, що містить вісім типів задач, генератор послідовно формує варіанти для кожного типу, забезпечуючи однакову структуру для всіх студентів, але різні числові дані. У результаті кожен студент отримує індивідуальний варіант роботи, що знижує можливість списування та підвищує об’єктивність оцінювання.

Генератор варіантів виконує роль проміжного шару між Wolfram-модулем і модулем формування тестового формату. Він агрегує математичні дані, отримані від Wolfram, та передає їх у структурованому вигляді до наступного етапу обробки. При цьому можуть виконуватися додаткові перетворення, наприклад, округлення числових значень або приведення виразів до зручного для відображення вигляду.

Отже, генератор варіантів задач перетворює окремі математично коректні задачі на повноцінний банк тестових завдань, придатний для використання в навчальному процесі. Його функціонування дозволяє автоматизувати створення великої кількості варіантів.

2.4. Генератор Moodle-формату: принципи побудови та функціонування

Генератор Moodle-формату є завершальним етапом формування тестових завдань у розробленій системі та забезпечує перетворення структурованих математичних і текстових даних у формат, придатний для імпорту в систему Moodle. Його основна функція полягає не у створенні змісту задач, а у формалізації вже згенерованих даних у вигляді стандартизованого XML-документа. На цьому етапі система оперує вже повністю визначеним набором даних, що включає математичну модель задачі, конкретні значення параметрів, правильні відповіді, а також текстові формулювання, підготовлені AI-генератором. Завдання генератора Moodle-формату полягає у тому, щоб поєднати ці дані у єдину структуру.

Кожне тестове завдання оформлюється як окремий елемент структури, що містить назву, текст питання, формат відповіді, правильні значення та додаткові параметри, зокрема допустиму похибку для числових відповідей. Така структура дозволяє описати складні задачі, які містять кілька полів введення та комбінують різні типи відповідей.

Особливе значення у цій роботі має використання формату Cloze, який дозволяє інтегрувати кілька підзавдань в одне питання. Наприклад, у задачі на дослідження функції двох змінних студент може окремо ввести координати стаціонарної точки, значення других похідних та визначити тип екстремуму. Однак, генератор Moodle-формату не виконує обчислень, а лише включає вже отримані значення у відповідні поля XML-структури.

При формуванні числових відповідей важливим є врахування допустимої похибки. У Moodle числові відповіді задаються з певним інтервалом точності, що дозволяє враховувати можливі округлення. Наприклад, якщо правильне значення отримане у вигляді десяткового дробу, система може приймати відповіді, що відрізняються на малу величину. Це особливо

актуально для задач, де результати обчислень можуть мати нецілі значення.

Ще одним важливим аспектом є узгодженість між математичною частиною задачі та її текстовим описом. Генератор Moodle-формату забезпечує коректне підставлення числових значень і виразів у текст питання, щоб уникнути розбіжностей між умовою та правильними відповідями.

У процесі формування XML-документа також визначається структура банку питань. Завдання можуть групуватися за категоріями, що відповідають темам або типам задач, наприклад, «Функції багатьох змінних», «Екстремум», «Неявне диференціювання». Це полегшує подальше формування контрольні роботи.

Технічно генератор Moodle-формату реалізується як програмний модуль, що перетворює внутрішнє представлення задач у XML-файл. На цьому етапі виконуються операції форматування, екранування спеціальних символів, перевірки синтаксичної коректності та підготовки файлу до імпорту. Отриманий файл може бути безпосередньо завантажений у Moodle, після чого всі питання стають доступними в банку питань системи. Таким чином, генератор Moodle-формату виконує роль інтерфейсу між внутрішньою логікою системи та зовнішнім навчальним середовищем.

2.5. Опис взаємодії між модулями системи

Розроблена система автоматичного створення тестових завдань має модульну архітектуру, у якій кожен компонент виконує чітко визначену функцію, а їх взаємодія забезпечує цілісний процес генерації, оформлення та використання задач у навчальному середовищі. Такий підхід дозволяє розділити відповідальність між математичною, текстовою та технічною складовими системи.

Взаємодія між модулями організована як послідовний процес обробки даних, у якому результат роботи одного модуля виступає вхідними даними для наступного. Початковим етапом є визначення шаблону задачі, що задає її математичну модель, тип, рівень складності та вимоги до відповіді

Після цього в роботу вступає Wolfram-модуль, який виконує генерацію конкретного математичного екземпляра задачі. На основі шаблону він формує значення параметрів, будує відповідну функцію або рівняння та обчислює всі необхідні величини. Наприклад, для задач на дослідження функції двох змінних модуль визначає стаціонарну точку, обчислює похідні та значення визначника

$$D = f_{xx}f_{yy} - f_{xy}^2,$$

що дозволяє класифікувати тип екстремуму. Результатом роботи цього модуля є структурований набір математичних даних, який включає параметри задачі, правильні відповіді та додаткові характеристики, необхідні для подальшої обробки.

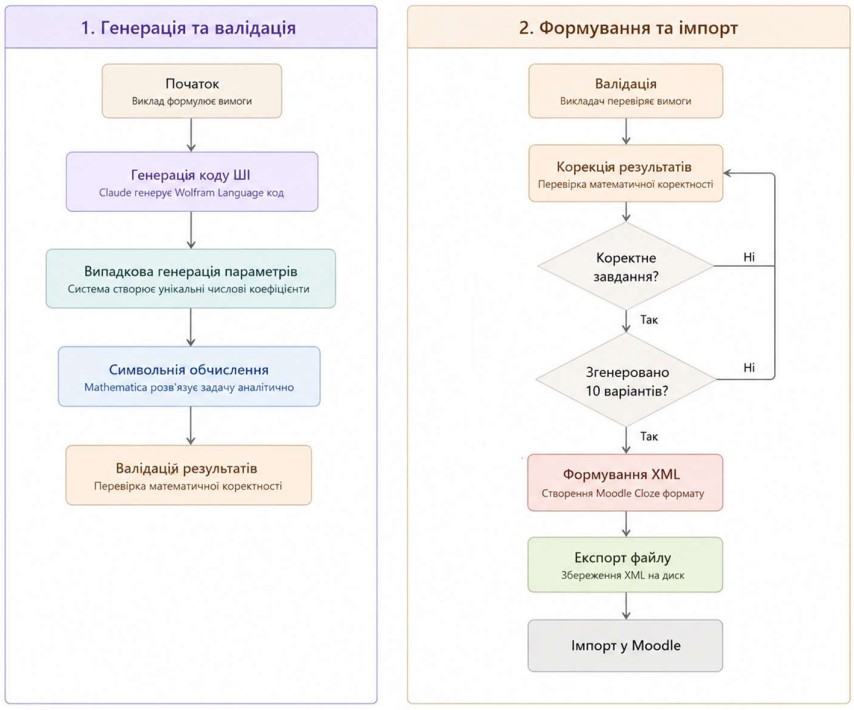
Отримані дані передаються до генератора варіантів задач, який організовує процес створення множини завдань на основі одного шаблону. На цьому етапі забезпечується варіативність, контроль складності та унікальність кожного варіанта. Генератор варіантів може багаторазово

викликати Wolfram-модуль, отримуючи нові набори параметрів, доки не буде сформовано необхідну кількість задач. Таким чином, він перетворює окремі математичні приклади на повноцінний банк завдань.

Паралельно або після формування математичних даних до процесу підключається AI-генератор, який відповідає за текстову інтерпретацію задачі. Він отримує параметри, обчислені значення та опис типу задачі, після чого формує умову, інструкції та, за потреби, пояснення розв’язання. Важливо, що AI-генератор адаптує математичні дані до зручного для сприйняття студентом вигляду. Таким чином забезпечується узгодженість між числовими значеннями, отриманими в Wolfram-модулі, та текстом задачі. Наступним етапом є робота генератора Moodle-формату, який об’єднує математичні та текстові дані у структурований XML-документ. На цьому рівні визначається формат тестового питання, типи відповідей, допустима похибка для числових значень і загальна структура завдання. Генератор забезпечує правильне відображення математичних виразів і відповідність формату вимогам Moodle. У результаті формується файл для імпорту у систему дистанційного навчання.

Завершальним етапом є використання згенерованих завдань у середовищі Moodle. Після імпорту XML-файлу питання додаються до банку питань, де можуть бути використані для створення тестів, контрольних робіт або тренувальних завдань.

Блок-схема алгоритму генерації завдань



III. Теоретичні основи. Перевірка валідності тестових питань

3.1. Складність питання

В класичній теорії тестування складність визначається як частка респондентів, які правильно відповіли на питання.

$$p_i = (1/n) \sum_{j=1}^{n} x_{ji} = N_{прав.}/N_{заг.},$$

де:

- p_i — індекс складності питання i,
- n — загальна кількість студентів,
- x_{ji}— відповідь студента j на питання i (1 або 0),
- $N_{прав.}$ — кількість правильних відповідей,
- $N_{заг.}$ — загальна кількість студентів.

Розподіл питань за складністю

Добре збалансований тест має містити питання різної складності. Типовий розподіл:

Категорія складності	Частка питань	Призначення
Легкі (p > 0.7)	20-30%	Мотивація, базові знання

Середні ($0.3 < p \leq 0.7$)	50-60%	Основна диференціація
Важкі ($p \leq 0.3$)	10-20%	Виявлення сильних студентів

Складність не враховує: чи розрізняє питання сильних і слабких студентів, чи узгоджене питання з іншими питаннями тесту, якість дистракторів у питаннях множинного вибору. Саме тому необхідний аналіз інших показників, зокрема дискримінативності.

Дискримінативність питання

Дискримінативність питання показує, наскільки добре воно відокремлює студентів з високим рівнем знань від студентів з низьким рівнем.

Найбільш поширеним показником дискримінативності є точково-бісеріальна кореляція між відповідями на питання та загальним балом студентів.

$$r_{pbi} = (M_1 - M_0)/S_t\sqrt{pq/n},$$

де:

- r_{pbi} — точково-бісеріальна кореляція для питання і,
- M_1 — середній загальний бал студентів, які правильно відповіли на питання,

- M_0 — середній загальний бал студентів, які неправильно відповіли,
- S_t — стандартне відхилення загальних балів,
- p — частка правильних відповідей,
- q = 1 - p — частка неправильних відповідей,
- n — загальна кількість студентів.

Точково-бісеріальну кореляцію можна також виразити як звичайну кореляцію Пірсона:

$$r_{pbi} = \frac{\mathrm{Cov}(X_i, T)}{\sigma_{X_i}\sigma_T},$$

де:

- X_i — вектор відповідей на питання i,
- T — вектор загальних балів,
- σ_{X_i} — стандартне відхилення відповідей на питання i,
- σ_T— стандартне відхилення загальних балів.

Інтерпретація значень

Значення r_{pb}	Якість питання	Рекомендації
$r_{pb} > 0.40$	Відмінна дискримінація	Залишити без змін

$0.30 < r_{pb} \leq 0.40$	Добра дискримінація	Може бути покращено
$0.20 < r_{pb} \leq 0.30$	Прийнятна дискримінація	Потребує покращення
$0.10 < r_{pb} \leq 0.20$	Погана дискримінація	Рекомендується переробити
$r_{pb} \leq 0.10$	Дуже погана	Вилучити або повністю переробити
$r_{pb} < 0$	Негативна дискримінація	Критична проблема

Від'ємне значення коефіцієнта означає, що на питання частіше правильно відповідають слабкі студенти. Це майже завжди вказує на помилку в ключі відповідей або серйозні проблеми з формулюванням питання.

Для більш точної оцінки дискримінативності використовується кореляція між питанням і загальним балом без цього питання:

$$r_{i(t-i)} = \mathrm{Cor}(X_i, T - X_i).$$

Ця корекція усуває штучне завищення кореляції через включення самого питання в загальний бал. Критерії інтерпретації залишаються подібними до некоригованої кореляції, але значення зазвичай дещо нижчі.

Альтернативним і більш простим для обчислення показником дискримінативності є порівняння частки правильних відповідей у групах сильних та слабких студентів.

Метод верхніх та нижніх 27%

Класичний підхід, запропонований Келлі, передбачає розділення студентів на три групи за загальним балом:

1. Верхня група — 27% студентів з найвищими балами.
2. Середня група — 46% студентів з середніми.
3. Нижня група — 27% студентів з найнижчими балами.

Вибір саме 27% обґрунтований статистично: цей відсоток максимізує різницю між групами при збереженні достатнього розміру вибірки.

Індекс дискримінації D

Індекс дискримінації обчислюється як різниця між часткою правильних відповідей у верхній та нижній групах:

$$D_i = p_U - p_L,$$

де:

- D_i — індекс дискримінації питання i,
- $p_U = N_{U_{correct}}/N_U$ — частка правильних відповідей у верхній групі,

- $p_L = N_{L_{correct}}/N_L$ — частка правильних відповідей у нижній групі.

Інтерпретація індексу D

Значення D	Оцінка	Інтерпретація
$D \geq 0.40$	Відмінно	Дуже хороше питання, залишити
$0.30 \leq D < 0.40$	Добре	Гарне питання, можливі незначні покращення
$0.20 \leq D < 0.30$	Задовільно	Прийнятно, але бажано покращити
$0.10 \leq D < 0.20$	Погано	Слабка дискримінація, потребує перегляду
$0 \leq D < 0.10$	Дуже погано	Практично не дискримінує, переробити
$D < 0$	Неприйнятно	Критична помилка

3.2. Коефіцієнт альфа Кронбаха

Надійність тесту характеризує стабільність та узгодженість вимірювань. Найпоширенішим показником внутрішньої узгодженості є коефіцієнт альфа Кронбаха.

Коефіцієнт альфа Кронбаха визначається через дисперсії окремих питань та загальну дисперсію тесту:

$$\alpha = (k / (k-1)) (1 - \Sigma\sigma^2_i / \sigma^2_T),$$

де:

- α — коефіцієнт альфа Кронбаха,
- k — кількість питань у тесті,
- σ^2_i — дисперсія відповідей на питання i,
- σ^2_T — дисперсія загальних балів студентів,
- $\Sigma\sigma^2_i$ — сума дисперсій всіх питань.

Коефіцієнт альфа Кронбаха можна інтерпретувати як середню кореляцію між усіма можливими способами розділення тесту на дві частини. Формула показує, яка частка загальної варіабельності обумовлена коваріацією між питаннями, а не унікальною варіабельністю окремих питань.

Інтерпретація значень α

Діапазон α	Рівень надійності	Оцінка	Типове застосування

$\alpha \geq 0.90$	Відмінний	Дуже висока надійність	Вступні екзамени
$0.80 \leq \alpha < 0.90$	Добрий	Висока надійність	Більшість стандартизованих тестів
$0.70 \leq \alpha < 0.80$	Прийнятний	Задовільна надійність	Поточний контроль, дослідницькі тести
$0.60 \leq \alpha < 0.70$	Сумнівний	Низька надійність	Пілотні версії, попередні тести
$\alpha < 0.60$	Неприйнятний	Дуже низька надійність	Тест потребує серйозної переробки

Фактори, що впливають на α

Надійність зростає зі збільшенням кількості питань. Формула Спірмена-Брауна показує, як зміниться надійність при зміні кількості питань:

$$\alpha_{новий} = (n \times \alpha_{попередній}) / (1 + (n-1) \times \alpha_{попередній})$$

де n — множник зміни довжини тесту (наприклад, n=2 означає подвоєння кількості питань).

Альфа Кронбаха вимірює внутрішню узгодженість — наскільки питання вимірюють один і той самий показник. Якщо тест охоплює кілька тем, α може бути низьким навіть при хорошій якості окремих питань.

Надто легкі ($p > 0.95$) або надто важкі ($p < 0.05$) питання мають малу дисперсію і знижують α. Оптимальна складність для максимізації α — близько 0.5.

Якщо тест складається з підшкал (наприклад, різні теми або рівні складності), α слід обчислювати окремо для кожної підшкали:

$$\alpha_{підшкали} = (k_s / (k_s-1)) \times (1 - \Sigma\sigma^2_{si} / \sigma^2_S)$$

де індекс s позначає підшкалу.

Стандартна похибка вимірювання

Стандартна похибка вимірювання (SEM) показує типову величину помилки при вимірюванні індивідуального балу студента.

$$\mathrm{SEM} = \sigma_T\sqrt{1-\alpha},$$

де:

- SEM — стандартна похибка вимірювання,
- σ_T — стандартне відхилення загальних балів тесту,
- α — коефіцієнт надійності (наприклад, альфа Кронбаха).

SEM можна інтерпретувати наступним чином: якщо студент пройде тест багато разів за однакових умов, його бали будуть розподілені нормально навколо істинного балу з стандартним відхиленням SEM.

Довірчий інтервал для істинного балу студента:

$$CI = X_{емпіричне} \pm z \times SEM,$$

де:

- $X_{емпіричне}$ — спостережуваний бал студента,
- z — критичне значення для обраного рівня надійності.

В класичній теорії тестування припускається, що SEM однакова для всіх рівнів балів. Проте в реальності точність вимірювання може відрізнятися.

Щоб визначити, чи є різниця між балами двох студентів статистично значущою, використовують стандартну похибку різниці:

Надійність методом розщеплення

Тест розділяється на дві половини (парні/непарні питання або випадково), обчислюється кореляція між половинами:

$$r_{половин} = Cor(T_{парні}, T_{непарні}).$$

Оцінка надійності повного тесту за формулою Спірмена-Брауна:

$$r_{тесту} = (2 \times r_{половин}) / (1 + r_{половин}).$$

3.3. Факторний аналіз

Факторний аналіз — це статистичний метод, який дозволяє виявити приховані (латентні) змінні, що пояснюють кореляції між спостережуваними змінними.

Метою факторного аналізу тестів є виявлення структури, групування питань за подібністю вимірюваних компетентностей, оцінка валідності компетентностей, виявлення проблемних питань, скорочення розмірності даних для подальшого аналізу.

Основна модель факторного аналізу

Спостережувана змінна (відповідь на питання) розкладається на лінійну комбінацію факторів:

$$X_i = \lambda_{i1}F_1 + \lambda_{i2}F_2 + ... + \lambda_{im}F_m + \varepsilon_{i,}$$

де:

- X_i — спостережувана змінна (відповідь на питання i)
- F_j — j-й фактор (латентна змінна)
- λ_{ij} — факторне навантаження питання i на фактор j
- ε_i — унікальна компонента (специфічна для питання i)
- m — кількість факторів

Факторне навантаження λ_{ij} показує силу зв'язку між питанням i та фактором j.

Інтерпретація

$\lvert\lambda_{ij}\rvert$	Інтерпретація
> 0.7	Сильний зв'язок
0.5 - 0.7	Помірний зв'язок
0.3 - 0.5	Слабкий зв'язок
< 0.3	Незначний зв'язок

Комунальність h^2 — частка дисперсії питання, що пояснюється факторами:

$$h^2_i = \lambda^2_{i1} + \lambda^2_{i2} + ... + \lambda^2_{im.}$$

Унікальність u^2 — частка дисперсії, що не пояснюється факторами:

$$u^2_i = 1 - h^2_i.$$

Питання з низькою комунальністю ($h^2 < 0.3$) слабко пов'язані з іншими питаннями тесту і можуть бути кандидатами на вилучення.

Експлораторний факторний аналіз (EFA) — використовується, коли структура тесту невідома. Метод сам визначає кількість факторів та їх зв'язок з питаннями.

В контексті валідації тестів найчастіше використовується EFA для початкового дослідження структури.

Один з найважливіших і найскладніших кроків у факторному аналізі — визначення оптимальної кількості факторів.

Критерій Кайзера

Найпростіший критерій: залишити фактори з власними значеннями $\lambda > 1$.

Власне значення $\lambda_j = 1$ означає, що фактор пояснює стільки ж дисперсії, скільки одна спостережувана змінна. Тому фактори з $\lambda < 1$ вважаються неінформативними.

Має тенденцію переоцінювати кількість факторів, особливо при великій кількості змінних. Більш надійний для 20-40 змінних.

Графічний метод

Будується графік власних значень у порядку спадання. Шукається "злам" кривої — точка, після якої власні значення зменшуються повільно.

Критерій відсотка поясненої дисперсії

Залишають стільки факторів, щоб вони разом пояснювали не менше певного відсотка загальної дисперсії:

$$\frac{\sum_{j=1}^{m} \lambda_j}{k} \cdot 100\%.$$

Типові пороги:

- Соціальні науки: 50-60%.

- Природничі науки: 70-80%.
- Психометрія: 60-70%.

Обертання факторів

Початкове рішення факторного аналізу часто важко інтерпретувати. Обертання спрощує структуру, роблячи навантаження більш екстремальними (близькими до 0 або 1).

Незалежні фактори:

- Varimax — максимізує дисперсію квадратів навантажень.
- Quartimax — мінімізує кількість факторів для кожної змінної.
- Equamax — комбінація varimax та quartimax.

Фактори можуть корелювати:

- Promax — швидкий метод, популярний для великих наборів даних.
- Oblimin — мінімізує кореляцію між факторами.

Інтерпретація факторів

Після обертання кожен фактор інтерпретується на основі питань з найвищими навантаженнями.

Якщо питання має високі навантаження (> 0.4) на два або більше факторів, то це ускладнює інтерпретацію і може вказувати на те, що питання вимірює кілька компетенцій одночасно або фактори слід об'єднати, або питання сформульоване нечітко.

Питання з $h^2 < 0.3$ слабко пов'язані з виділеними факторами. Це може бути пов’язане з тим, що питання вимірює щось унікальне, не пов'язане з

іншими, питання має погану дискримінацію, обрано неправильну кількість факторів.

Перевірка придатності даних

Перед проведенням факторного аналізу необхідно перевірити, чи придатні дані:

Тест сферичності Бартлетта

Перевіряє гіпотезу H_0: кореляційна матриця є одиничною (змінні некорельовані).

$$\chi^2 = -(n - 1 - (2k + 5)/6) \times \ln|R|,$$

де:

- n — розмір вибірки.
- k — кількість змінних.
- |R| — визначник кореляційної матриці.

Якщо $p < 0.05$, гіпотеза H_0 відхиляється — дані придатні для факторного аналізу.

IV. Реалізація

У роботі було реалізовано програмну систему для автоматичного створення тестових завдань з теми “Функції багатьох змінних”. Основна ідея полягає в тому, що математичні обчислення виконує Wolfram, а Python відповідає за керування процесом, обробку даних, формування тексту завдань українською мовою та експорт у формат Moodle.

Система складається з кількох логічних частин:

1. Математичні генератори Wolfram.
2. Python-модуль запуску Wolfram.
3. Модуль генерації текстової частини через штучний інтелект.
4. Модуль формування фінального JSON-файлу.
5. Модуль експорту в Moodle XML.
6. Головний керуючий файл main.py.

Тобто система працює як послідовний конвеєр обробки:

Wolfram → JSON → Python → штучний інтелект → фінальний JSON → Moodle XML

Англійський термін pipeline у роботі можна перекладати як конвеєр обробки або послідовність етапів обробки.

1. Роль Wolfram у системі

Wolfram використовується як математичне ядро системи. Його завдання — генерувати математично коректні варіанти задач та обчислювати правильні відповіді.

Штучний інтелект не використовується для математичних обчислень, тому що мовні моделі можуть помилятися в обчисленнях. Wolfram, навпаки, є системою символьних і числових обчислень, тому він

краще підходить для побудови похідних, розв'язування систем рівнянь, обчислення визначників, значень функцій тощо.

Для кожного типу задачі створено окремий Wolfram-скрипт:

generate_extremum.wls

generate_implicit_derivatives.wls

generate_parametric_derivatives.wls

generate_directional_derivative.wls

generate_second_differential.wls

generate_tangent_plane_normal.wls

generate_conditional_extremum.wls

generate_bounded_region_extrema.wls

Кожен скрипт працює автономно. Він:

- генерує випадкові параметри;
- будує математичну модель;
- перевіряє умови коректності;
- обчислює правильні відповіді;
- записує результат у файл JSON.

У Wolfram свідомо не використовується український текст. Наприклад, тип екстремуму записується не як мінімум, а як:

- minimum
- maximum
- saddle
- undefined

Це зроблено для уникнення проблем з кодуванням символів. Український текст додається вже на рівні Python.

2. Єдина структура даних JSON

Усі типи задач приводяться до єдиної структури. Це важливо, бо надалі Python може однаково обробляти різні типи задач.

Кожен варіант має таку структуру:

```
{
  "task_type": "...",
  "topic": "multivariable_functions",
  "difficulty": "medium",
  "statement_data": {},
  "parameters": {},
  "answers": {},
  "checks": {},
  "valid": true
}
```

Пояснення полів:

task_type — тип задачі;

topic — тема;

difficulty — рівень складності;

statement_data — дані для формування умови;

parameters — випадково згенеровані параметри;

answers — правильні відповіді;

checks — результати перевірок коректності;

valid — ознака придатності задачі.

Це дозволяє зробити систему розширюваною. Якщо надалі потрібно додати новий тип задачі, достатньо створити новий Wolfram-скрипт, який експортує дані в такій самій структурі.

3. Генерація числових відповідей

Для всіх числових відповідей використовується уніфікована схема:

"value": ...,

"value_exact": "...",

"value_rounded": ...

Наприклад:

"zx": 0.7142857142857143,

"zx_exact": "5/7",

"zx_rounded": 0.714

Це зроблено для трьох різних потреб:

Основне поле — зберігає значення, яке обчислив Wolfram.

Поле exact — зберігає точне дробове представлення.

Поле rounded — зберігає округлене значення для Moodle.

У Moodle для числової перевірки використовується округлене значення з допустимою похибкою 0.001.

Це важливо, тому що Moodle не завжди зручно перевіряє дробові вирази, але добре працює з числовими відповідями у форматі:

{1:NUMERICAL:=0.714:0.001}

4. Python-модуль запуску Wolfram

Для запуску Wolfram-скриптів створено модуль:

modules/wolfram_runner.py

Його завдання:

- знайти корінь проекту;
- перевірити наявність Wolfram-скриптів;
- запустити кожен скрипт через wolframscript;
- передати кількість варіантів;
- прочитати створені JSON-файли;
- об’єднати всі задачі в один список.

Кількість варіантів задається в одному місці — у файлі main.py:

VARIANTS_PER_TYPE = 2

Якщо потрібно генерувати 5 варіантів кожного типу, достатньо змінити:

VARIANTS_PER_TYPE = 5

Це значення передається у Wolfram через змінну середовища:
VARIANTS_PER_TYPE
Завдяки цьому не потрібно вручну змінювати кожен із восьми Wolfram-скриптів.
Також у модулі запуску передбачена обробка ситуації, коли Wolfram створив JSON-файл, але сам процес завершився з попередженням або помилкою ліцензії. Якщо JSON-файл існує і коректно читається, система може продовжити роботу.

5. Генерація текстової частини через штучний інтелект

Для створення українського тексту задач використовується модуль:
modules/ai_generator.py

Штучний інтелект не виконує математичних обчислень. Він отримує вже готові математичні дані та формує:
question_text — текст умови;
instruction — інструкцію до розв’язання;
feedback — пояснення після виконання;
distractors — варіанти неправильних відповідей, якщо вони потрібні.
Для звернення до моделі використовується програмний інтерфейс OpenAI. Англійський термін API можна перекладати як програмний інтерфейс доступу.
У системному запиті до моделі явно вказано:

- Не обчислюй математику.
- Не додавай готові відповіді.
- Поверни тільки JSON за заданою схемою.

Тобто модель не має права змінювати правильні відповіді. Вона лише мовно оформлює задачу.

Це важливо з методичної точки зору, бо математична правильність залишається за Wolfram, а штучний інтелект використовується тільки для природного українського формулювання.

6. Захист від розкриття відповіді в тексті задачі

Окремо реалізовано перевірки, щоб штучний інтелект не розкрив студенту правильну відповідь у тексті умови.

Наприклад, для задачі на екстремум заборонено, щоб у тексті з’являлися готові значення:

x0 = ...

y0 = ...

тип точки = ...

Тобто студент має сам знайти ці значення. Умова повинна містити лише функцію та загальне завдання.

Якщо модель випадково повертає текст із готовою відповіддю, Python піднімає помилку і не приймає такий результат.

7. Формування фінального JSON

Після того як Wolfram згенерував математичні дані, а штучний інтелект додав текстову частину, система створює файл:

data/generated_variants.json

Це головний фінальний файл з усіма задачами.

Він містить:

- математичну модель;
- параметри;
- правильні відповіді;
- перевірки;
- український текст задачі;
- інструкції;
- пояснення.

Проміжні JSON-файли після цього прибираються, щоб у теці data залишався фінальний результат.

8. Експорт у Moodle XML

Для експорту задач у Moodle створено модуль:
modules/moodle_xml_generator.py.
Він перетворює generated_variants.json у файл:
data/moodle_questions.xml.
Цей файл можна імпортувати в Moodle через:
Банк питань → Імпорт → Формат Moodle XML.
Для кожної задачі створюється питання типу Cloze.
У XML створюється категорія:
$course$/Generated/Multivariable Functions
Для кожної задачі формується:

- назва питання;
- текст умови;
- підказка щодо формату відповіді;
- підзавдання;
- поля для введення відповідей;
- пояснення після відповіді.

9. Оформлення питань у Moodle

Була додана велика кількість текстових «вкраплень», які допомагають студенту розуміти суть та послідовність виконання підзавдань:

- українські назви питань;
- структуровані підзавдання;
- підказки;
- покращений вигляд формул;
- випадаючі списки для вибору типу точки.

10. Підказки до відповідей

У кожному питанні додається коротка підказка. Наприклад:

Числові відповіді вводьте як число або десятковий дріб з точністю до 0.001.

Для задач із вибором типу точки додається:

Тип точки оберіть зі списку.

Для другого диференціала:

Коефіцієнт при dxdy уже відповідає виразу $2f_{x\gamma}$.

Це потрібно, щоб студент розумів, у якому форматі вводити відповідь.

11. Випадаючі списки для вибору

Для відповідей, де є очевидний вибір з кількох варіантів, використовується не текстове поле, а список.

Наприклад, для типу точки:

- мінімум
- максимум
- сідлова точка
- невизначено

У Moodle це реалізовано через формат MULTICHOICE. Тобто, студент не вводить текст вручну, а обирає правильний варіант зі списку.

12. Обробка кожного з восьми типів задач

У системі реалізовано 8 типів задач:

1. Екстремум функції.
2. Неявно задана функція.
3. Параметрично задана функція.
4. Похідна за напрямом.
5. Другий диференціал.
6. Дотична площина і нормаль.

7. Умовний екстремум.

8. Екстремуми в обмеженій області.

Для кожного типу є власний Wolfram-скрипт і власний шаблон Moodle-питання.

Наприклад, для екстремуму Moodle питає:

- координати стаціонарної точки;
- другі похідні;
- визначник Гессе;
- значення функції;
- тип точки.

Для похідної за напрямом:

- компоненти градієнта;
- норму вектора;
- компоненти одиничного вектора;
- похідну за напрямом.

Для обмеженої області:

- точку мінімуму;
- значення мінімуму;
- точку максимуму;
- значення максимуму.

13. Перевірки коректності

У кожній задачі є блок:

"checks": {}

У ньому зберігаються результати перевірок.

Наприклад:

- чи точка лежить на поверхні;
- чи не дорівнює знаменник нулю;
- чи система має єдиний розв’язок;

- чи не занадто великі відповіді;
- чи обмеження не вироджене;
- чи знайдено глобальні екстремуми.

Поле:

"valid": true означає, що задача придатна до використання.

Під час експорту в Moodle задачі з:

"valid": false пропускаються.

14. Чому система є модульною

Кожен етап винесений в окремий файл:

wolfram_runner.py — запуск Wolfram;

ai_generator.py — робота зі штучним інтелектом;

moodle_xml_generator.py — експорт у Moodle XML;

main.py — головний файл запуску.

Це дає кілька переваг:

легше тестувати окремі частини;

можна додати новий тип задачі без переписування всієї системи;

можна замінити модуль штучного інтелекту;

можна змінити формат експорту;

можна масштабувати кількість варіантів.

15. Як працює main.py

Головний файл main.py запускає весь процес:

1. Запускає всі Wolfram-генератори.
2. Об’єднує всі задачі.
3. Надсилає задачі до модуля штучного інтелекту.
4. Зберігає generated_variants.json.
5. Створює moodle_questions.xml.
6. Видаляє проміжні JSON-файли.

Запуск виконується командою:

py main.py

Після виконання з’являються два головні файли:

data/generated_variants.json;

data/moodle_questions.xml.

Перший потрібний для аналізу та зберігання даних, другий — для імпорту в Moodle.

16. Головна технічна цінність роботи

Технічна цінність полягає в тому, що система автоматизує повний цикл створення тестів:

- генерація математичної задачі;
- обчислення правильної відповіді;
- перевірка коректності;
- створення українського тексту;
- формування Moodle XML;
- імпорт у систему дистанційного навчання.

Тобто викладач не створює кожне питання вручну, а може згенерувати багато варіантів автоматично. При цьому кожен варіант має правильні відповіді, пояснення та готовий формат для Moodle.

V. Апробація

5.1. Загальна характеристика апробації

Апробацію розроблених тестових завдань було проведено на базі РТФ. У дослідженні взяли участь 66 студентів, які виконували ДКР на тему «ФБЗ». Основною метою апробації була перевірка валідності та надійності тестових завдань, автоматично сформованих за допомогою розробленої системи.

Для оцінювання якості тесту використовувалися методи Класичної теорії тестів, теоретичні основи якої були розглянуті у попередніх розділах дипломної роботи. Аналіз проводився для 42 тестових питань, причому враховувалися всі підпитання окремих завдань.

Максимальна кількість балів за тест становила:

$$X_{max} = 42.$$

Середній результат студентів дорівнював:

$$\bar{X} = 20.73.$$

Медіана результатів:

$$Me = 22.$$

Стандартне відхилення:

$$\sigma = 9.15.$$

Мінімальний результат становив 2 бали, максимальний — 38 балів.

Загальні статистичні характеристики тесту та критерії оцінювання:

Завдань 8, але загальна кількість питань (рахуємо підпитання кожного завдання) - 42		
Студентів (n)	**66**	
Питань (k)	**42**	
Максимальний бал	**42**	
Середній бал	**20.73**	
Медіана	**22.0**	
Стд. відхилення (σ)	**9.15**	
Мін. бал студента	**2**	
Макс. бал студента	**38**	
Коеф. Кронбаха (α)	**0.9**	
Стандартна похибка (SEM)	**2.896**	
Дельта Фергюсона (δ)	**20.802**	
Split-half (Спірмен-Браун)	**0.894**	
Оптимальних p∈[0.3,0.9]	**42/42**	
r_pb > 0.2	**42/42**	
D > 0.2	**42/42**	
Кількість факторів (λ>1)	**15**	

1. Складність (p)	p=Σ(Xij)/n. Оптимально: 0.3≤p≤0.9.
2. Дискримінативність (r_pb)	Point-biserial кореляція. r_pb>0.3 — добре, >0.2 — прийнятно.
3. Індекс D (27%)	D=p_upper−p_lower. D>0.4 відмінно, >0.3 добре, >0.2 прийнятно.
4. Альфа Кронбаха (α)	α>0.7 прийнятно, >0.8 добре, >0.9 відмінно.
5. SEM	SEM=σ_T√(1−α). Точність вимірювання.
6. Дельта Фергюсона (δ)	δ>0.9 — тест добре розрізняє студентів.
7. Split-half	r_split=$2r_{12}/(1+r_{12})$. Надійність при поділі навпіл.
8. Корекція на здогадку	p_corr=p−(1−p)/(m−1), m=4.

Аналіз складності тестових питань

У результаті аналізу було встановлено, що всі 42 питання потрапили в допустимий діапазон складності. Це свідчить про відсутність надто легких або надто складних завдань.

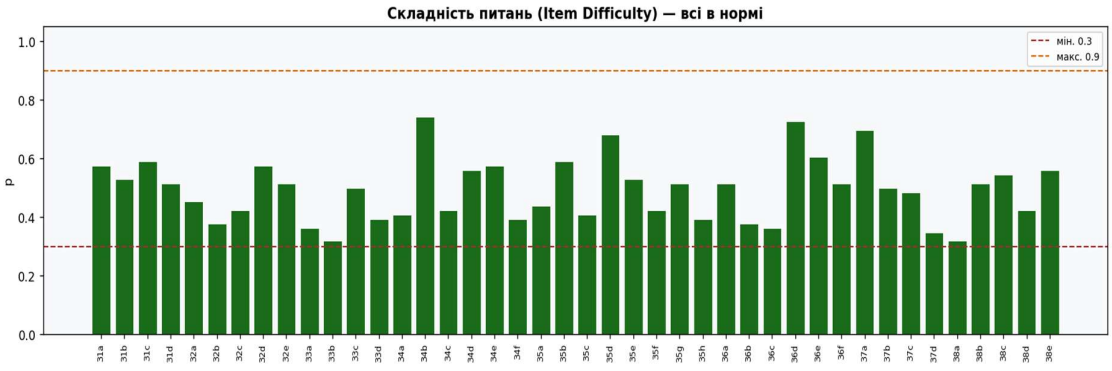
Середня складність тесту становила:

$$\bar{p} = 0.494.$$

Отримане значення є близьким до оптимального рівня 0.5, що вказує на добре збалансований тест.

На графіку складності видно, що значення pдля всіх завдань знаходяться між граничними лініями 0.3 та 0.9. Найвищу складність мали окремі питання типу 34b та 36e, однак навіть вони залишалися в межах норми.

Графік складності тестових питань:



Аналіз дискримінативності питань

Мінімально допустимим значенням для точково-бісеріальної кореляції вважається:

$$r_{pb} > 0.2.$$

Усі 42 тестові питання перевищили цей поріг, що свідчить про достатню дискримінативність.

Середнє значення показника:

$$\bar{r}_{pb} = 0.439.$$

Це означає, що тестові питання добре розділяють студентів із високими та низькими результатами.

Також було проаналізовано індекс дискримінації

Середнє значення:

$$\bar{D} = 0.533.$$

Оскільки:

$$D > 0.4,$$

можна зробити висновок про високу дискримінативність тесту.

Аналіз складності та дискримінативності тестових питань:

31a	0.576	Норма	0.377	Добре	0.389	Добре	0.434	0.899	Залишити
31b	0.53	Норма	0.549	Добре	0.778	Відмінно	0.374	0.896	Залишити
31c	0.591	Норма	0.48	Добре	0.611	Відмінно	0.455	0.897	Залишити
31d	0.515	Норма	0.564	Добре	0.778	Відмінно	0.354	0.896	Залишити
32a	0.455	Норма	0.327	Добре	0.333	Добре	0.273	0.899	Залишити
32b	0.379	Норма	0.375	Добре	0.556	Відмінно	0.172	0.899	Залишити
32c	0.424	Норма	0.431	Добре	0.611	Відмінно	0.232	0.898	Залишити
32d	0.576	Норма	0.316	Добре	0.389	Добре	0.434	0.899	Залишити
32e	0.515	Норма	0.375	Добре	0.5	Відмінно	0.354	0.899	Залишити
33a	0.364	Норма	0.443	Добре	0.556	Відмінно	0.152	0.898	Залишити
33b	0.318	Норма	0.33	Добре	0.389	Добре	0.091	0.899	Залишити
33c	0.5	Норма	0.474	Добре	0.611	Відмінно	0.333	0.897	Залишити
33d	0.394	Норма	0.573	Добре	0.667	Відмінно	0.192	0.896	Залишити
34a	0.409	Норма	0.486	Добре	0.556	Відмінно	0.212	0.897	Залишити
34b	0.742	Норма	0.441	Добре	0.444	Відмінно	0.657	0.898	Залишити
34c	0.424	Норма	0.461	Добре	0.556	Відмінно	0.232	0.897	Залишити
34d	0.561	Норма	0.407	Добре	0.5	Відмінно	0.414	0.898	Залишити
34e	0.576	Норма	0.413	Добре	0.5	Відмінно	0.434	0.898	Залишити
34f	0.394	Норма	0.438	Добре	0.556	Відмінно	0.192	0.898	Залишити
35a	0.439	Норма	0.34	Добре	0.278	Прийнятно	0.253	0.899	Залишити
35b	0.591	Норма	0.511	Добре	0.611	Відмінно	0.455	0.897	Залишити
35c	0.409	Норма	0.352	Добре	0.389	Добре	0.212	0.899	Залишити
35d	0.682	Норма	0.282	Прийнятно	0.389	Добре	0.576	0.9	Залишити
35e	0.53	Норма	0.45	Добре	0.556	Відмінно	0.374	0.897	Залишити
35f	0.424	Норма	0.454	Добре	0.556	Відмінно	0.232	0.897	Залишити
35g	0.515	Норма	0.445	Добре	0.5	Відмінно	0.354	0.898	Залишити
35h	0.394	Норма	0.383	Добре	0.5	Відмінно	0.192	0.898	Залишити
36a	0.515	Норма	0.263	Прийнятно	0.278	Прийнятно	0.354	0.9	Залишити
36b	0.379	Норма	0.296	Прийнятно	0.333	Добре	0.172	0.9	Залишити
36c	0.364	Норма	0.518	Добре	0.667	Відмінно	0.152	0.896	Залишити
36d	0.727	Норма	0.443	Добре	0.389	Добре	0.636	0.898	Залишити
36e	0.606	Норма	0.42	Добре	0.389	Добре	0.475	0.898	Залишити
36f	0.515	Норма	0.481	Добре	0.556	Відмінно	0.354	0.897	Залишити
37a	0.697	Норма	0.604	Добре	0.667	Відмінно	0.596	0.895	Залишити
37b	0.5	Норма	0.54	Добре	0.778	Відмінно	0.333	0.896	Залишити
37c	0.485	Норма	0.41	Добре	0.444	Відмінно	0.313	0.898	Залишити
37d	0.348	Норма	0.411	Добре	0.611	Відмінно	0.131	0.898	Залишити
38a	0.318	Норма	0.454	Добре	0.556	Відмінно	0.091	0.897	Залишити
38b	0.515	Норма	0.551	Добре	0.722	Відмінно	0.354	0.896	Залишити
38c	0.545	Норма	0.442	Добре	0.5	Відмінно	0.394	0.898	Залишити
38d	0.424	Норма	0.501	Добре	0.667	Відмінно	0.232	0.897	Залишити
38e	0.561	Норма	0.634	Добре	0.778	Відмінно	0.414	0.895	Залишити

Аналіз надійності тесту

Надійність тесту оцінювалася за допомогою коефіцієнта Кронбаха.

Отримане значення:

$$\alpha = 0.900.$$

Відповідно до загальноприйнятих критеріїв, тест демонструє дуже високу внутрішню узгодженість.

Додатково оцінювалася split-half надійність за формулою Спірмена–Брауна:

$$r_{split} = \frac{2r_{1/2}}{1 + r_{1/2}}.$$

Отриманий результат:

$$r_{split} = 0.894.$$

Це підтверджує стабільність тесту при поділі питань на дві частини.

Стандартна похибка вимірювання визначалась за формулою:

$$SEM = \sigma_T\sqrt{1 - \alpha}.$$

У результаті:

$$SEM = 2.896.$$

Отримане значення означає, що середня похибка вимірювання результату студента становить приблизно ±3 бали.

Також було розраховано дельту Фергюсона:

$$\delta = 20.802.$$

Цей показник свідчить про високу розрізнювальну здатність тесту.

Узагальнені психометричні характеристики тесту:

Коефіцієнт Кронбаха α	**0.9**	☑ *Добре (норма: >0.7)*
SEM	**2.896**	*σ_T×√(1−α)*
Split-half (Спірмен-Браун)	**0.894**	*Парні/непарні питання*
Дельта Фергюсона δ	**20.802**	☑ *Висока дискримінативність (норма: >0.9)*
Середнє p	**0.494**	*Середня складність*
Середнє r_pb	**0.439**	*Середня дискримінативність*
Середнє D	**0.533**	*Середній індекс дискримінації*
Оптимальних p∈[0.3,0.9]	**42/42**	☑ *Всі в нормі*
r_pb > 0.2	**42/42**	☑ *Всі в нормі*
D > 0.2	**42/42**	☑ *Всі в нормі*
Кореляцій |r|>0.5 між питаннями	**4**	*Потенційно дублюючих пар*
Кількість факторів (λ>1)	**15**	*Критерій Кайзера*

Аналіз розподілу результатів студентів

Для оцінювання якості тесту також було проаналізовано розподіл сумарних балів студентів.

На гістограмі видно, що результати студентів розподілені достатньо рівномірно. Основна частина результатів знаходиться поблизу середнього значення:

$$\bar{X} = 20.73.$$

Медіана:

$$\mathrm{Me} = 22.$$

Розподіл не має вираженого зміщення до мінімальних або максимальних значень, що свідчить про відсутність ефекту «надто легкого» або «надто складного» тесту.

Наявність широкого діапазону результатів підтверджує хорошу розрізнювальну здатність тестових питань.

5.2. Висновки за результатами апробації

Проведена апробація підтвердила високу якість розроблених тестових завдань. Усі 42 питання відповідали нормативним критеріям Класичної теорії тестів за показниками складності, дискримінативності та індексу дискримінації.

Основні результати апробації:

- кількість студентів:

$$n = 66;$$

- кількість тестових питань:

$$k = 42;$$

- середня складність:

$$\bar{p} = 0.494;$$

- середня дискримінативність:

$$\bar{r}_{pb} = 0.439;$$

- середній індекс дискримінації:

$$\bar{D} = 0.533;$$

- коефіцієнт Кронбаха:

$$\alpha = 0.900;$$

- split-half надійність:

$$r_{split} = 0.894;$$

- стандартна похибка вимірювання:

$$\mathrm{SEM} = 2.896.$$

Отримані результати свідчать, що тест є:

- надійним;
- валідним;
- збалансованим за складністю;
- придатним для оцінювання знань студентів.

Таким чином, апробація підтвердила ефективність використання автоматизованої системи генерації тестових завдань на основі методів штучного інтелекту та Wolfram Mathematica для дисциплін математичного циклу.

Висновки

Проведене дослідження показало, що поєднання систем штучного інтелекту та Wolfram Mathematica дійсно дозволяє автоматизувати процес створення тестових завдань з вищої математики без втрати математичної коректності. У ході роботи було проаналізовано сучасні підходи до автоматичної генерації тестів та виявлено, що більшість існуючих рішень або не забезпечують достатньої математичної надійності, або потребують значних часових затрат зі сторони викладача. Саме тому основна ідея роботи полягала у побудові системи, яка б дозволяла автоматично створювати унікальні математичні задачі з уже гарантовано вбудованими правильними відповідями.

У якості основного інструменту для математичної частини було обрано Wolfram Mathematica, оскільки система дозволяє виконувати символьні обчислення, перевіряти еквівалентність виразів та автоматично контролювати коректність згенерованих задач. У процесі роботи було реалізовано механізм генерації параметризованих задач з теми «Функції багатьох змінних», де побудова задачі фактично відбувається «від відповіді». Це дозволило уникнути ситуацій, коли система випадково генерує функцію без коректного або адекватного розв’язку.

Для текстового наповнення задач було використано систему штучного інтелекту, яка відповідала виключно за мовне оформлення умови, варіантів відповідей та допоміжного тексту. При цьому математична частина залишалась повністю контрольованою зі сторони Wolfram. Такий підхід дозволив поєднати математичну стабільність із більш природним та «людяним» формулюванням задач.

У результаті роботи було реалізовано повний ланцюг автоматичної генерації тестів: від створення математичної моделі задачі та знаходження

правильної відповіді до автоматичного формування Moodle XML-файлу, готового до імпорту в систему Moodle. Це дозволяє викладачу фактично одразу отримувати готовий банк тестових завдань без необхідності ручного перенесення задач у систему тестування.

Окрему увагу було приділено перевірці якості сформованих тестів. Для цього проведено апробацію на вибірці із 66 студентів РТФ, які виконували ДКР з теми «Функції багатьох змінних». Психометричний аналіз показав, що тест має хороші характеристики складності, дискримінативності та надійності. Середня складність тесту склала

$$\bar{p} = 0.494,$$

що є практично оптимальним значенням для тестових завдань. Середня дискримінативність становила

$$\bar{r}_{pb} = 0.439,$$

а коефіцієнт Кронбаха дорівнював

$$\alpha = 0.900.$$

Отримані результати свідчать про те, що сформовані питання добре розділяють студентів за рівнем підготовки та забезпечують високу внутрішню узгодженість тесту.

У результаті вдалося побудувати систему, яка не просто генерує випадкові задачі, а створює математично коректні, валідні та придатні до реального використання тестові завдання. Практична цінність роботи полягає у можливості значно скоротити час на підготовку контрольних

матеріалів, зменшити повторюваність варіантів та автоматизувати процес формування банків тестових питань для Moodle.

У подальшому розроблений підхід можна розширити на інші розділи вищої математики, а також інтегрувати психометричні моделі та розробити більш гнучкі механізми генерації задач.

Список використаних джерел

[1] Attali Y. et al. The interactive reading task: Transformer-based automatic item generation. Frontiers in Artificial Intelligence. 2022. Vol. 5. DOI: 10.3389/frai.2022.903077.

[2] Sayın A., Gierl M. J. Using OpenAI GPT to Generate Reading Comprehension Items. Educational Measurement: Issues and Practice. 2024. DOI: 10.1111/emip.12590.

[3] Gorgun G., Bulut O. Instruction-Tuned Large-Language Models for Quality Control in Automatic Item Generation: A Feasibility Study. Educational Measurement: Issues and Practice. 2024. DOI: 10.1111/emip.12663.

[4] Sommer M., Arendasy M. Automatic- and Transformer-Based Automatic Item Generation: A Critical Review. Journal of Intelligence. 2025. Vol. 13. DOI: 10.3390/jintelligence13080102.

[5] Tan B. et al. A review of automatic item generation techniques leveraging large language models. International Journal of Assessment Tools in Education. 2025. DOI: 10.21449/ijate.1602294.

[6] Maeda H. Field-Testing Multiple-Choice Questions With AI Examinees: English Grammar Items. Educational and Psychological Measurement. 2024. Vol. 85. P. 221–244. DOI: 10.1177/00131644241281053.

[7] Angelelli M. et al. Human- vs. AI-generated tests: dimensionality and information accuracy in latent trait evaluation. ArXiv. 2025. abs/2510.24739. DOI: 10.48550/arXiv.2510.24739.

[8] Kumar D. et al. Item analysis of multiple choice questions: A quality assurance test for an assessment tool. Medical Journal, Armed Forces India. 2021. Vol. 77 Suppl 1. P. S85–S89. DOI: 10.1016/j.mjafi.2020.11.007.

[9] Kıyak Y. S. et al. Exploratory Factor Analysis of a Computerized Case-Based F-Type Testlet Variant. Medical Science Educator. 2023. Vol. 33. P. 1191–1196. DOI: 10.1007/s40670-023-01876-y.

[10] Veeramani H. et al. Large Language Model-based Pipeline for Item Difficulty and Response Time Estimation for Educational Assessments. Workshop on Innovative Use of NLP for Building Educational Applications. 2024. P. 561–566.

[11] Tran P. et al. Using item features to calibrate educational test items: Comparing artificial intelligence and classical approaches. American Journal of Education and Learning. 2025. DOI: 10.55284/ajel.v10i2.1543.

[12] Uto M., Tomikawa Y., Suzuki A. Question Difficulty Prediction Based on Virtual Test-Takers and Item Response Theory. EcalLAC@AIED. 2024.

[13] Диховичний О. О., Круглова Н. В., Кайдалов С. В. Застосування додатків, створених у середовищах Shiny та Wolfram, на заняттях з вищої математики та теорії ймовірностей // XX міжнародна наукова конференція імені академіка Михайла Кравчука. – 2025. – с. 191-192.

[14] Круглова Н.В., Диховичний О. О., Кайдалов С. В. Застосування штучного інтелекту та інструментів Wolfram, GeoGebra для створення шаблонів тестових завдань з вищої математики на платформі Moodle. // XX міжнародна наукова конференція імені академіка Михайла Кравчука. – 2025. – с. 193-194.

[15] Кайдалов С. В., Круглова Н. В. Генерація категорій тестових питань з теорії ймовірностей за допомогою Claude та Wolfram // XIV Всеукраїнська наукова конференція молодих математиків. – 2026. – С. 189–190.

Додатки

Main-скрипт:

```
import json,sys
from modules.ai_generator import enrich_variants_openai
from modules.moodle_xml_generator import DEFAULT_OUTPUT_PATH,generate_moodle_xml
from modules.wolfram_runner import FINAL_JSON,cleanup_intermediate_json_files,generate_all_variants
VARIANTS_PER_TYPE=2
def main()->None:
    if hasattr(sys.stdout,"reconfigure"): sys.stdout.reconfigure(encoding="utf-8")
    print("[1/4] Запускаємо Wolfram-генератори...")
    variants=generate_all_variants(VARIANTS_PER_TYPE)
    print(f"[2/4] Згенеровано {len(variants)} задач ({VARIANTS_PER_TYPE} на тип).")
    print("[3/4] Генеруємо українські тексти через OpenAI...")
    enriched=enrich_variants_openai(variants)

FINAL_JSON.write_text(json.dumps(enriched,ensure_ascii=False,indent=2),encoding="utf-8")
    generate_moodle_xml(enriched,DEFAULT_OUTPUT_PATH)
    cleanup_intermediate_json_files()
    print(f"[4/4] JSON: {FINAL_JSON}")
    print(f"Moodle XML: {DEFAULT_OUTPUT_PATH}")
if __name__=="__main__": main()
```

Python-код для генерації описів до завдань (необхідно вставити валідний API-ключ для OpenAI API):

```
import copy,json,os,re
from typing import Any
OPENAI_API_KEY=os.getenv("OPENAI_API_KEY","PUT_YOUR_API_KEY_HERE")
OPENAI_MODEL="gpt-5.4-nano"
TYPE_LABELS_UA={"minimum":"мінімум","maximum":"максимум","saddle":"сідлова точка","undefined":"невизначено"}
AI_TEXT_SCHEMA={"type":"object","properties":{"question_text":{"type":"string"},"instruction":{"type":"string"},"feedback":{"type":"string"},"distractors":{"type":"array","items":{"type":"string"}}},"required":["question_text","instruction","feedback","distractors"],"additionalProperties":False}
def d(v,k): return v.get(k) if isinstance(v.get(k),dict) else {}
def answers(v): return d(v,"answers")
def sd(v): return d(v,"statement_data")
def params(v): return d(v,"parameters")
def func(v):
    f=v.get("function") or sd(v).get("function")
    if not f: raise ValueError("Немає function.")
    return f
def eq(v): return sd(v).get("equation","")
def point(v,*keys): return "(" + ", ".join(str(params(v)[k]) for k in keys) + ")"
def region(v):
    r=sd(v).get("region","")
    m=re.fullmatch(r"x in \[([^,]+),([^\]]+)\], y in \[([^,]+),([^\]]+)\]",r)
    return f"x ∈ [{m[1].strip()}; {m[2].strip()}], y ∈ [{m[3].strip()}; {m[4].strip()}]" if m else r
def distractors(t): return [v for k,v in TYPE_LABELS_UA.items() if k!=t]
```

```
def _build_openai_payload(v:dict[str,Any])->dict[str,Any]:
  t=v.get("task_type"); a=answers(v); f=func(v) if t not in {"implicit_derivatives"} else ""
  if t=="extremum":
    typ=str(a.get("type"))
    return {"task_type":t,"function":f,"required_question_text":f"Дано функцію f(x,y)={f}. Дослідіть її на екстремум.","required_instruction":"Знайдіть стаціонарну точку, обчисліть fxx, fyy, fxy у цій точці, знайдіть значення функції в стаціонарній точці та визначте її тип. Вкажіть координати точки, значення функції та тип точки.","required_feedback":"Стаціонарна точка знаходиться із системи f_x=0, f_y=0. Тип точки визначається за знаком визначника Гессе D=f_xx f_yy - f_xy^2.","required_distractors":distractors(typ)}
  if t=="implicit_derivatives":
    e,p=eq(v),point(v,"x0","y0","z0")
    return {"task_type":t,"equation":e,"point":p,"required_question_text":f"Дано неявно задану поверхню {e} і точку M{p}. Знайдіть частинні похідні z_x та z_y у цій точці.","required_instruction":"Обчисліть Fx, Fy, Fz у заданій точці та використайте формули z_x=-Fx/Fz, z_y=-Fy/Fz. Відповідь можна подати у вигляді дробу або десяткового числа з точністю до 3 знаків.","required_feedback":"Для неявно заданої функції F(x,y,z)=0 частинні похідні знаходяться за формулами z_x=-F_x/F_z і z_y=-F_y/F_z, якщо F_z не дорівнює нулю.","required_distractors":[]}
  if t=="parametric_derivatives":
    xuv,yuv,p=sd(v).get("x_uv",""),sd(v).get("y_uv",""),point(v,"u0","v0")
    return {"task_type":t,"function":f,"x_uv":xuv,"y_uv":yuv,"point":p,"required_question_text":f"Дано функцію z=f(x,y), де f(x,y)={f}, а змінні x та y задано
```

параметрично: x={xuv}, y={yuv}. Знайдіть частинні похідні z_u та z_v при (u,v)={p}.","required_instruction":"Обчисліть значення x0 та y0 у заданій точці, знайдіть f_x, f_y, похідні x_u, x_v, y_u, y_v та застосуйте правило ланцюга. Відповідь можна подати у вигляді дробу або десяткового числа з точністю до 3 знаків.","required_feedback":"Для параметрично заданої функції похідні z_u та z_v обчислюються за правилом ланцюга: z_u=f_x x_u+f_y y_u, z_v=f_x x_v+f_y y_v.","required_distractors":[]}

```
    if t=="directional_derivative":
        p,dv=point(v,"x0","y0"),point(v,"p","q")
        return
```

{"task_type":t,"function":f,"point":p,"direction_vector":dv,"required_question_text":f"Дано функцію f(x,y)={f}. Знайдіть похідну цієї функції в точці M{p} за напрямом вектора v={dv}.","required_instruction":"Обчисліть градієнт функції в заданій точці, нормуйте напрямний вектор та знайдіть скалярний добуток градієнта на одиничний напрямний вектор. Відповідь можна подати у вигляді дробу або десяткового числа з точністю до 3 знаків.","required_feedback":"Похідна за напрямом обчислюється як скалярний добуток градієнта функції на одиничний вектор напряму.","required_distractors":[]}

```
    if t=="tangent_plane_normal":
        p=point(v,"x0","y0")
        return
```

{"task_type":t,"function":f,"point":p,"required_question_text":f"Дано поверхню z=f(x,y)={f}. Знайдіть рівняння дотичної площини та нормаль у точці M{p}.","required_instruction":"Обчисліть частинні похідні f_x, f_y, знайдіть точку на поверхні та побудуйте рівняння дотичної площини. Відповідь можна подати у вигляді дробу або десяткового числа з точністю до 3 знаків.","required_feedback":"Рівняння дотичної площини має вигляд

```
z=z0+f_x(x-x0)+f_y(y-y0). Нормаль задається вектором (-f_x,-f_y,1).","required_distractors":[]}
    if t=="conditional_extremum":
        c=sd(v).get("constraint","")
        return
{"task_type":t,"function":f,"constraint":c,"required_question_text":f"Дано функцію f(x,y)={f}. Знайдіть умовний екстремум цієї функції за умови {c}.","required_instruction":"Складіть функцію Лагранжа, запишіть систему рівнянь для L_x=0, L_y=0 та умови зв’язку, знайдіть точку умовного екстремуму та значення функції в цій точці. Відповідь можна подати у вигляді дробу або десяткового числа з точністю до 3 знаків.","required_feedback":"Умовний екстремум знаходиться методом множників Лагранжа. Для обмеження p*x+q*y=r розв’язується система L_x=0, L_y=0, p*x+q*y=r.","required_distractors":[]}
    if t=="bounded_region_extrema":
        r=region(v)
        return
{"task_type":t,"function":f,"region":r,"required_question_text":f"Дано функцію f(x,y)={f}. Знайдіть її найбільше та найменше значення в області: {r}.","required_instruction":"Знайдіть стаціонарні точки функції, перевірте їх належність області, дослідіть функцію на межах області та порівняйте всі отримані значення. Відповідь можна подати у вигляді дробу або десяткового числа з точністю до 3 знаків.","required_feedback":"Глобальні екстремуми функції в замкненій області досягаються у стаціонарних точках або на межі області.","required_distractors":[]}
    if t=="second_differential":
        p=point(v,"x0","y0")
        return
{"task_type":t,"function":f,"point":p,"required_question_text":f"Дано
```

функцію f(x,y)={f}. Знайдіть другий диференціал цієї функції в точці M{p}.","required_instruction":"Обчисліть другі частинні похідні fxx, fxy, fyy у заданій точці та запишіть другий диференціал у вигляді d²f=fxx dx²+2fxy dxdy+fyy dy². Відповідь можна подати у вигляді дробу або десяткового числа з точністю до 3 знаків.","required_feedback":"Другий диференціал функції двох змінних обчислюється за формулою d²f=fxx dx²+2fxy dxdy+fyy dy², де похідні беруться у заданій точці.","required_distractors":[]}

raise ValueError(f'Непідтримуваний тип задачі: {t}")

def enrich_variant_openai(variant:dict[str,Any])->dict[str,Any]:

if not OPENAI_API_KEY or OPENAI_API_KEY=="PUT_YOUR_API_KEY_HERE": raise RuntimeError("Вкажіть OPENAI_API_KEY у змінній середовища або у файлі ai_generator.py.")

from openai import OpenAI

enriched=copy.deepcopy(variant);

payload=_build_openai_payload(enriched)

system_prompt=("Ти формуєш українські тексти для тестових завдань з вищої математики. "

"Поверни тільки JSON за схемою. Не обчислюй математику. Не додавай готові відповіді у question_text, instruction або feedback. "

"Використай required_question_text, required_instruction, required_feedback і required_distractors без змістових змін.")

user_prompt="Створи ai_text для цього варіанта. Не розкривай студенту відповіді до задачі.\n\n"+json.dumps(payload,ensure_ascii=False,indent=2)

response=OpenAI(api_key=OPENAI_API_KEY).responses.create(model=OPENAI_MODEL,input=[{"role":"system","content":system_prompt},{"role":"u

```
ser","content":user_prompt}],text={"format":{"type":"json_schema","name":"
ai_text","strict":True,"schema":AI_TEXT_SCHEMA}})
    enriched["ai_text"]=json.loads(response.output_text)
    return enriched
def enrich_variants_openai(variants:list[dict[str,Any]])->list[dict[str,Any]]:
    return [enrich_variant_openai(v) for v in variants]
```

Код для генерації у Wolfram:

```
import json,os,subprocess
from pathlib import Path
from typing import Any
PROJECT_ROOT=Path(__file__).resolve().parents[1]
DATA_DIR=PROJECT_ROOT/"data"
FINAL_JSON=DATA_DIR/"generated_variants.json"
TASK_FILES={
    "extremum":("generate_extremum.wls","extremum_variants.json"),

"implicit_derivatives":("generate_implicit_derivatives.wls","implicit_derivativ
es_variants.json"),

"parametric_derivatives":("generate_parametric_derivatives.wls","parametric_
derivatives_variants.json"),

"directional_derivative":("generate_directional_derivative.wls","directional_de
rivative_variants.json"),

"second_differential":("generate_second_differential.wls","second_differential
_variants.json"),
```

```
"tangent_plane_normal":("generate_tangent_plane_normal.wls","tangent_plan
e_variants.json"),

"conditional_extremum":("generate_conditional_extremum.wls","conditional_
extremum_variants.json"),

"bounded_region_extrema":("generate_bounded_region_extrema.wls","bounde
d_region_extrema_variants.json"),
}
def load_variants(path:Path)->list[dict[str,Any]]:
   if not path.exists(): raise FileNotFoundError(f"JSON-файл не знайдено:
{path}")
   data=json.loads(path.read_text(encoding="utf-8"))
   if not isinstance(data,list): raise ValueError(f"Очікувався JSON-масив:
{path}")
   return data
def _json_ok(path:Path)->bool:
   try: return bool(load_variants(path))
   except Exception: return False
def run_wolfram_script(script_path:Path,json_path:Path,variants_per_type:int)-
>None:
   if not script_path.exists(): raise FileNotFoundError(f"Wolfram-скрипт не
знайдено: {script_path}")
   if variants_per_type<=0: raise ValueError("Кількість варіантів має бути
додатною.")
   env=os.environ.copy();
env["VARIANTS_PER_TYPE"]=str(variants_per_type)
   rel=script_path.relative_to(PROJECT_ROOT)
```

```
    r=subprocess.run(["wolframscript","-
file",str(rel)],cwd=PROJECT_ROOT,env=env,capture_output=True,text=True,
encoding="utf-8")
    if r.returncode and not _json_ok(json_path):
        raise RuntimeError(f"Wolfram-скрипт завершився з
помилкою.\nКоманда: wolframscript -file {rel}\nКод:
{r.returncode}\nstdout:\n{r.stdout}\nstderr:\n{r.stderr}")
    if r.returncode: print(f"[Wolfram warning] Код {r.returncode}, але JSON
читається: {json_path.name}")
def generate_task_variants(task_type:str,variants_per_type:int)-
>list[dict[str,Any]]:
    script_name,json_name=TASK_FILES[task_type]
    json_path=DATA_DIR/json_name

run_wolfram_script(PROJECT_ROOT/"wolfram"/script_name,json_path,varia
nts_per_type)
    return load_variants(json_path)
def generate_all_variants(variants_per_type:int=5)->list[dict[str,Any]]:
    variants=[]
    for task_type in TASK_FILES:
variants.extend(generate_task_variants(task_type,variants_per_type))
    return variants
def cleanup_intermediate_json_files()->None:
    for _,json_name in TASK_FILES.values():
        path=DATA_DIR/json_name
        try:
            if path.exists(): path.unlink()
        except OSError: pass
```

Wolfram-скрипт для генерації параметрів:

```
SetDirectory[ParentDirectory[DirectoryName[$InputFileName]]];
If[!DirectoryQ["data"], CreateDirectory["data"]];
ClearAll[roundedValue, exactValue, generateVariant];
variantCount =
ToExpression[Replace[Environment["VARIANTS_PER_TYPE"], $Failed | ""->"5"]];
roundedValue[value_] := N[Round[value, 1/1000], 6];
exactValue[value_] := ToString[InputForm[Rationalize[value, 0]]];
generateVariant[] := Module[
{a, b, c, g, x0, y0, d, e, f, fxx, fyy, fxy, hessianDet,
fValue, pointType, validMagnitude},
validMagnitude = False;
While[!validMagnitude,
a = RandomChoice[DeleteCases[Range[-5, 5], 0]];
b = RandomChoice[DeleteCases[Range[-5, 5], 0]];
c = RandomInteger[{-5, 5}];
While[4 a b - c^2 == 0,
a = RandomChoice[DeleteCases[Range[-5, 5], 0]];
b = RandomChoice[DeleteCases[Range[-5, 5], 0]];
c = RandomInteger[{-5, 5}];
];
x0 = RandomInteger[{-4, 4}];
y0 = RandomInteger[{-4, 4}];
g = RandomInteger[{-10, 10}];
d = -(2 a x0 + c y0);
e = -(2 b y0 + c x0);
f = a*x^2 + b*y^2 + c*x*y + d*x + e*y + g;
fValue = f /. {x->x0, y->y0};
```

```
validMagnitude = Abs[fValue] <= 200;
];
fxx = D[f, {x, 2}];
fyy = D[f, {y, 2}];
fxy = D[D[f, x], y];
hessianDet = fxx*fyy - fxy^2;
pointType = Which[
hessianDet > 0 && fxx > 0, "minimum",
hessianDet > 0 && fxx < 0, "maximum",
hessianDet < 0, "saddle",
True, "undefined"
];
<|
"task_type"->"extremum",
"topic"->"multivariable_functions",
"statement_data"-><|
"function"->ToString[InputForm[f]],
"variables"->{"x", "y"}
|>,
"parameters"-><|
"a"->a,
"b"->b,
"c"->c,
"d"->d,
"e"->e,
"g"->g,
"x0"->x0,
"y0"->y0
|>,
```

```
"answers"-><|
"x0"->x0,
"y0"->y0,
"fxx"->fxx,
"fyy"->fyy,
"fxy"->fxy,
"hessian_det"->hessianDet,
"f_value"->fValue,
"type"->pointType,
"x0_exact"->exactValue[x0],
"y0_exact"->exactValue[y0],
"fxx_exact"->exactValue[fxx],
"fyy_exact"->exactValue[fyy],
"fxy_exact"->exactValue[fxy],
"hessian_det_exact"->exactValue[hessianDet],
"f_value_exact"->exactValue[fValue],
"x0_rounded"->roundedValue[x0],
"y0_rounded"->roundedValue[y0],
"fxx_rounded"->roundedValue[fxx],
"fyy_rounded"->roundedValue[fyy],
"fxy_rounded"->roundedValue[fxy],
"hessian_det_rounded"->roundedValue[hessianDet],
"f_value_rounded"->roundedValue[fValue]
|>,
"checks"-><|
"stationary_point"->True,
"hessian_det_nonzero"->(hessianDet != 0),
"answer_magnitude_limited"->validMagnitude
|>,
```

```
"valid"->True
|>
];
variants = Table[generateVariant[], {variantCount}];
Export["data/extremum_variants.json", variants, "JSON"];
```

Матриця результатів для перевірки тестів на валідність:

	a	b	c	d	a	b	c	d	e	a	b	c	d	a	b	c	d	e	f	a	b	c	d	e	f	g	h	a	b	c	d	e	f	a	b	c	d	a	b	c	d	e	
Ст.01	1	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	1	1	0	1	1	1	0	0	1	1	1	1	1	32
Ст.02	0	1	1	0	0	1	0	1	1	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	1	1	1	0	0	1	0	1	17
Ст.03	1	0	1	1	0	1	0	1	1	0	0	1	1	1	1	0	0	1	0	0	1	1	0	0	1	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	1	1	24
Ст.04	0	0	0	0	1	1	1	0	0	1	0	1	1	1	1	1	0	0	0	1	1	0	1	0	1	1	0	0	1	0	1	1	1	1	0	0	1	0	1	1	1	1	24
Ст.05	0	1	0	0	0	0	0	1	1	0	0	1	0	0	1	0	1	1	0	1	0	0	1	0	0	0	0	1	1	0	1	0	1	1	1	1	0	0	0	0	1	1	18
Ст.06	1	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	1	1	0	1	1	0	1	0	1	1	1	1	1	1	0	1	1	1	0	1	23
Ст.07	1	1	1	1	1	0	1	0	0	1	0	0	1	1	0	1	0	0	0	1	0	0	1	0	1	0	0	0	1	1	0	1	1	1	1	1	0	0	0	1	1	1	23
Ст.08	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	5
Ст.09	0	1	0	1	1	1	0	1	0	1	0	1	1	0	1	0	1	1	1	1	1	0	1	1	0	1	1	0	1	1	1	1	1	1	1	1	0	1	0	1	0	1	30
Ст.10	1	1	1	1	0	0	0	1	1	0	1	1	1	1	0	1	1	0	0	1	1	1	1	1	0	1	1	0	1	0	1	1	1	0	1	1	0	1	1	1	1	0	29
Ст.11	1	0	1	0	1	0	0	1	0	0	1	0	0	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	0	0	1	0	0	1	0	0	1	0	0	0	1	1	18
Ст.12	0	1	0	1	0	0	0	1	0	1	0	0	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	0	1	0	0	1	1	18
Ст.13	0	1	1	0	1	0	0	0	1	1	1	1	1	1	1	0	1	1	0	1	1	0	1	1	0	0	1	1	1	0	1	1	1	0	0	1	1	0	0	1	0	1	26
Ст.14	1	0	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	1	0	1	1	0	1	0	0	0	0	1	0	13
Ст.15	1	0	0	1	1	0	0	1	0	0	0	0	0	0	1	0	1	1	1	1	0	1	0	0	1	0	0	1	0	0	1	1	0	1	0	0	0	0	0	1	0	1	17
Ст.16	0	1	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
Ст.17	0	0	0	1	0	0	1	1	1	1	0	1	1	0	1	1	1	1	1	0	1	1	1	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	1	1	0	1	25
Ст.18	1	0	1	0	1	1	0	1	1	0	0	1	0	1	1	1	0	0	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	1	1	0	0	1	1	0	1	0	22
Ст.19	0	1	1	1	1	1	1	0	1	1	0	0	1	0	1	1	1	1	1	0	1	0	1	1	1	0	0	1	0	0	1	1	1	0	1	1	0	1	1	0	1	1	28
Ст.20	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	8
Ст.21	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	38
Ст.22	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	1	1	1	1	0	1	1	0	0	1	1	1	1	1	0	0	0	1	1	0	0	23
Ст.23	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	1	0	1	0	1	1	0	0	0	1	1	0	1	0	1	0	0	0	1	0	16
Ст.24	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	38
Ст.25	0	0	1	0	1	0	0	1	1	0	0	1	0	0	1	0	0	1	0	1	1	0	1	1	1	1	0	1	1	1	1	1	0	1	1	0	1	0	1	1	0	1	24
Ст.26	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	7
Ст.27	1	1	1	1	0	1	0	1	1	0	0	0	1	0	1	1	1	1	0	1	1	1	0	0	1	1	0	0	0	0	1	1	0	1	0	1	0	0	0	1	1	0	23
Ст.28	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	2
Ст.29	1	1	1	0	0	1	1	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	35
Ст.30	1	0	0	1	0	1	0	0	0	0	1	1	0	0	1	0	1	1	0	0	0	1	1	1	0	0	1	0	0	1	0	1	1	1	0	0	0	0	0	0	1	0	17
Ст.31	1	0	0	0	1	0	0	1	1	0	1	0	0	0	1	0	0	1	0	1	1	0	1	0	0	0	0	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	16
Ст.32	1	1	1	1	1	0	1	1	1	1	0	0	1	1	1	1	1	1	1	0	1	0	0	0	1	1	0	1	0	0	1	0	1	1	1	0	1	0	1	1	1	1	29
Ст.33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Ст.34	1	1	1	1	1	1	1	0	1	1	1	1	0	1	1	0	1	0	1	1	1	1	0	0	1	1	0	0	1	0	0	0	1	1	1	1	0	1	1	0	1	0	29
Ст.35	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	9
Ст.36	1	1	0	0	1	0	1	0	0	0	1	1	0	0	1	0	1	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	0	0	18
Ст.37	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0	1	1	0	1	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	12
Ст.38	0	1	1	1	0	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	34
Ст.39	0	1	1	1	0	1	0	1	1	1	0	1	1	0	1	0	1	1	0	1	0	1	1	1	1	1	0	0	0	1	1	1	0	1	1	1	0	1	1	1	1	1	29
Ст.40	1	1	1	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	0	1	1	1	0	0	1	1	1	0	0	1	1	0	1	0	0	1	0	0	0	18
Ст.41	1	1	1	0	0	0	1	1	0	0	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1	1	1	1	0	1	1	1	1	0	1	1	1	0	1	29
Ст.42	0	1	1	1	0	0	1	1	0	0	0	1	0	1	1	0	0	1	1	0	0	0	1	0	0	1	1	1	0	1	1	1	1	1	0	0	0	0	1	0	0	0	20
Ст.43	0	1	0	1	1	0	1	0	1	1	1	1	0	0	0	0	0	0	0	1	0	1	1	1	1	0	1	1	0	0	1	0	1	1	1	1	0	0	1	1	0	0	22
Ст.44	0	0	1	0	0	1	1	1	0	1	0	0	0	0	1	0	0	1	0	0	1	1	1	1	1	0	0	1	0	0	1	1	0	0	0	0	0	0	1	1	0	0	17
Ст.45	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	3
Ст.46	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	6
Ст.47	1	0	1	0	1	1	1	1	0	0	0	1	0	1	0	0	1	1	1	0	1	0	0	1	1	1	1	0	0	0	1	1	0	0	0	1	0	0	1	1	1	0	22
Ст.48	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	38
Ст.49	1	0	0	1	0	0	1	1	0	0	0	1	0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	0	0	0	0	0	0	1	22
Ст.50	1	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	1	1	1	1	1	1	0	0	0	1	0	1	0	1	0	1	0	1	1	1	20
Ст.51	1	1	1	1	1	0	1	1	1	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	36
Ст.52	1	1	0	1	0	0	1	0	1	0	1	1	1	0	1	1	0	1	1	0	1	0	1	1	1	0	1	1	1	0	1	1	0	1	0	1	1	0	0	1	1	0	26
Ст.53	0	0	1	1	0	0	0	0	1	0	1	1	1	1	0	0	1	1	1	0	1	0	0	0	1	1	1	0	1	1	1	1	1	0	0	1	0	1	1	0	1	0	23
Ст.54	1	1	1	1	1	0	1	1	0	1	0	0	1	1	0	1	1	1	0	0	0	1	0	1	1	1	0	1	0	1	1	1	1	1	0	1	0	0	0	0	1	0	25
Ст.55	0	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	0	0	1	0	1	1	1	0	0	1	0	0	1	1	0	0	1	0	0	0	0	1	0	1	20
Ст.56	1	1	0	1	1	0	1	1	0	1	1	1	0	1	1	0	0	1	1	0	1	0	1	1	0	0	0	1	0	0	1	0	1	0	0	0	0	0	1	0	0	0	20
Ст.57	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	1	0	1	0	1	0	9
Ст.58	1	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	0	0	0	1	1	0	1	1	1	0	1	1	0	1	0	1	1	0	1	0	1	1	0	1	27
Ст.59	1	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	1	1	0	1	1	1	1	0	1	1	1	0	0	1	1	1	0	0	0	0	1	1	22
Ст.60	1	1	1	0	1	1	1	1	1	0	0	1	0	1	1	1	0	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	0	1	0	1	1	32
Ст.61	0	1	0	1	0	1	1	1	1	1	0	0	0	1	1	0	0	1	0	1	1	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1	0	1	0	20
Ст.62	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	5
Ст.63	0	1	0	0	1	0	0	0	1	0	0	1	1	0	1	1	1	0	0	0	1	0	1	0	0	0	1	0	0	0	1	1	0	1	0	0	0	0	0	1	0	0	15
Ст.64	1	1	1	1	1	0	0	1	1	1	1	1	1	0	1	1	1	0	0	1	1	1	1	1	0	1	1	0	0	1	1	0	1	1	1	0	1	0	1	0	1	1	30
Ст.65	1	0	1	0	1	0	1	1	1	1	0	0	1	0	1	0	1	0	0	0	1	1	1	1	1	0	0	0	1	1	0	0	1	0	1	0	0	0	0	0	0	1	19
Ст.66	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	1	0	1	1	0	1	0	0	0	1	1	0	1	0	0	0	1	15
	38	35	39	34	30	25	28	38	34	24	21	33	26	27	49	28	37	38	26	29	39	27	45	35	28	34	26	34	25	24	48	40	34	46	33	32	23	21	34	36	28	37	1368
	58 %	53 %	59 %	52 %	45 %	38 %	42 %	58 %	52 %	36 %	32 %	50 %	39 %	41 %	74 %	42 %	56 %	58 %	39 %	44 %	59 %	41 %	68 %	53 %	42 %	52 %	39 %	52 %	38 %	36 %	73 %	61 %	52 %	70 %	50 %	48 %	35 %	32 %	52 %	55 %	42 %	56 %	20.7