НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Приладобудівний факультет
Кафедра комп'ютерно-інтегрованих оптичних та навігаційних систем

«На правах рукопису»
УДК 004.94

«До захисту допущено»
Завідувач кафедри
_____ Надія БУРАУ
(підпис)
«____» ____грудня_____ 2024 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою
«Комп'ютерно-інтегровані технології та системи в приладобудуванні»
зі спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»
на тему:
«Комбінований підхід та програмне забезпечення для моделювання MATLAB моделей»

Виконав:
студент ІІ курсу, групи ПГ-31мп
Головань Ярослав Валерійович _____
(підпис)

Науковий керівник:
к.т.н.,
Рупіч Сергій Сергійович _____
(підпис)

Консультант з розроблення стартап-проекту:
д.е.н., проф., завідувач кафедри економічної кібернетики
Бояринова Катерина Олександрівна _____
(підпис)

Рецензент:
д.т.н., проф.
Тимчик Григорій Семенович _____
(підпис)

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

Київ – 2024 р.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Приладобудівний факультет

Кафедра комп'ютерно-інтегрованих оптичних та навігаційних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність – 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»

Освітньо-професійна програма «Комп'ютерно-інтегровані системи та технології в приладобудуванні»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Надія БУРАУ
«___»_____2024 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Голованю Ярославу Валерійовичу

1. Тема дисертації «Комбінований підхід та програмне забезпечення для моделювання MATLAB моделей», науковий керівник дисертації Рупіч Сергій Сергійович, к.т.н., затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом дисертації: 9 грудня 2024 року

3. Об'єкт дослідження: процес автоматизованого моделювання інженерних розрахунків і процесів в інтегрованому програмному середовищі

4. Предмет дослідження: методи та програмні засоби автоматизованої багатопотокової симуляції MATLAB моделей шляхом інтеграції з мовою програмування Java

5. Перелік завдань, які потрібно розробити:

5.1 Провести огляд стану проблеми.

5.2 Провести огляд існуючих рішень та підходів у сфері багатопотокового моделювання в MATLAB.

5.3 Розробити архітектуру програмної реалізації, основні модулі та підсистеми.

5.4 Перевірити працездатність системи.

5.5 Провести дослідження ефективності розробленого комбінованого підходу для запуску MATLAB моделей.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: таблиці, графіки, рисунки, схеми, діаграми тощо

7. Орієнтовний перелік публікацій: 1 стаття в матеріалах конференції

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розроблення стартап-проекту			

9. Дата видачі завдання 15 жовтня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Провести огляд стану проблеми	27.10.2024	
2.	Провести огляд існуючих рішень та підходів у сфері багатопотокового моделювання в MATLAB	03.11.2024	
3.	Розробити архітектуру програмної реалізації, основні модулі та підсистеми	10.11.2024	
4.	Перевірити працездатність системи	24.11.2024	
5.	Провести дослідження ефективності розробленого комбінованого підходу для запуску MATLAB моделей	01.12.2024	
6.	Оформлення рукопису дисертації	09.12.2024	

Студент _____ Ярослав ГОЛОВАНЬ
(підпис) (прізвище, ініціали)

Науковий керівник дисертації _____ Сергій РУПІЧ
(підпис) (прізвище, ініціали)

РЕФЕРАТ

Актуальність. Моделювання є невід'ємною складовою наукових та інженерних досліджень, а також важливим інструментом для розробки та оптимізації складних систем у різних галузях, зокрема, машинобудування, електроніка, авіакосмічні системи, тощо. Одним із найпопулярніших інструментів для моделювання є програмне забезпечення MATLAB. Цей інструмент забезпечує високий рівень гнучкості та ефективності для створення математичних моделей та їхнього тестування в динамічних умовах. Однак зростаюча складність сучасних систем і вимоги до їхньої точності та швидкості обчислень підвищують необхідність у пошуку нових підходів для покращення ефективності процесів моделювання.

Одним із перспективних напрямків вдосконалення моделювання в MATLAB є використання комбінованих підходів, що передбачають інтеграцію різних методів для вирішення широкого спектра завдань. Зокрема, особливо актуальним є впровадження паралельних обчислень для оптимізації симуляційних процесів. В останні роки, багатопоточність стає важливим інструментом для роботи зі складними математичними моделями, оскільки дозволяє суттєво прискорити процес обробки даних, використовуючи можливості сучасних багатоядерних процесорів та обчислювальних кластерів.

Таким чином, дослідження нових підходів до моделювання в MATLAB, зокрема, комбінованих методів, є важливим кроком у напрямку створення ефективних інструментів для інженерів та дослідників.

Мета роботи. Розроблення комбінованого підходу та програмного забезпечення для ефективного моделювання MATLAB моделей із використанням багатопотоковості.

Об’єктом дослідження є процес автоматизованого моделювання інженерних розрахунків і процесів в інтегрованому програмному середовищі

Предметом дослідження є методи та програмні засоби автоматизованої багатопотокової симуляції MATLAB моделей шляхом інтеграції з мовою програмування Java.

Апробація результатів дисертації. Результати роботи були обговорені на наступних конференціях:

- XIX Наукова-практична конференція студентів, аспірантів та молодих вчених «Ефективність та автоматизація інженерних рішень у приладобудуванні», 04-05 грудня 2024 року.

Публікації. За матеріалами дисертації було опубліковано 1 тезу та доповідь на науково практичній конференції.

Головань Я. В., студент гр. ПГ-31мп, к.т.н., Рупіч С.С.. КПІ ім. Ігоря Сікорського. Комбінований підхід для моделювання MATLAB моделей. Ефективність та автоматизація інженерних рішень у приладобудуванні (04-05 грудня 2024 року), КПІ ім. Ігоря Сікорського, приладобудівний факультет, Київ, 2024.

Структура та обсяг роботи. Робота складається зі вступу, чотирьох розділів, висновків, у роботі наведено 24 рисунки, 30 таблиць, 7 лістингів, список літературних джерел з 31 найменування. Загальний обсяг роботи 121 аркуш.

Ключові слова: MATLAB, МОДЕЛЮВАННЯ, БАГАТОПОТОЧНІСТЬ, КОМБІНОВАНИЙ ПІДХІД, API, ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ, АВТОМАТИЗАЦІЯ.

ABSTRACT

Relevance. Modeling is an integral component of scientific and engineering research, as well as a crucial tool for developing and optimizing complex systems across various industries, including mechanical engineering, electronics, aerospace systems, and others. One of the most popular tools for modeling is MATLAB software, which provides high flexibility and efficiency for creating mathematical models and testing them under dynamic conditions. However, the increasing complexity of modern systems and the growing demands for accuracy and computational speed emphasize the need to explore new approaches to enhance the efficiency of modeling processes.

One promising direction for improving MATLAB modeling is the use of combined approaches that integrate various methods to solve a wide range of tasks. In particular, the implementation of parallel computing for optimizing simulation processes is especially relevant. In recent years, multithreading has become an important tool for working with complex mathematical models, as it significantly accelerates data processing by leveraging the capabilities of modern multicore processors and computing clusters.

Thus, exploring new approaches to MATLAB modeling, particularly combined methods, is a vital step toward creating efficient tools for engineers and researchers.

The goal. Development of a combined approach and software for efficient modeling of MATLAB models using multithreading.

The object of research is the process of automated modeling of engineering calculations and processes in an integrated software environment.

The subject of research are methods and software tools for automated multithreaded simulation of MATLAB models through integration with the Java programming language.

Approbation of the results of the dissertation. Results of the work were discussed at the following conferences:

- XIX Scientific and Practical Conference of Students, Postgraduates, and Young Scientists "Efficiency and Automation of Engineering Solutions in Instrumentation," December 4-5, 2024.

Publications. Based on the dissertation materials, 1 thesis and a report at a scientific and practical conference were published.

Holovan Ya. V., student of group PG-31mp, Ph.D., Rupich S. S. National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute.” A Combined Approach for MATLAB Model Simulation. Efficiency and Automation of Engineering Solutions in Instrumentation (December 4–5, 2024), National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute,” Instrumentation Faculty, Kyiv, 2024.

Structure and scope of work. The work consists of an introduction, four chapters, conclusions, and includes 24 figures, 30 tables, 7 code listings, and a bibliography of 31 references. The total volume of the work is 121 pages.

Keywords: MATLAB, SIMULATION, MULTITHREADING, COMBINED APPROACH, API, PARALLEL COMPUTATION, AUTOMATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ 11

ВСТУП 12

РОЗДІЛ 1 ОГЛЯД СТАНУ ПРОБЛЕМИ 15

1.1 Математичне моделювання та програмне середовище MATLAB 15

1.2 Основні принципи комбінованого підходу у моделюванні 18

1.2.1 Загальна характеристика паралельних обчислень 18

1.2.2 Комбінований підхід до моделювання MATLAB моделей 19

1.3 Огляд існуючих рішень та підходів у сфері багатопотокового моделювання в MATLAB 21

1.3.1 Паралельні цикли (parfor) 22

1.3.2 Використання spmd 24

1.3.3 Моделювання на основі GPU 26

1.4 Аналіз наукових робіт за тематикою дисертації 31

1.5 Формування мети завдання дослідження 34

РОЗДІЛ 2 ПРОГРАМНА РЕАЛІЗАЦІЯ КОМБІНОВАНОГО ПІДХОДУ ДЛЯ ЗАПУСКУ МОДЕЛЮВАНЬ MATLAB МОДЕЛЕЙ 35

2.1 MATLAB Engine API. Інструмент для інтеграції MATLAB 35

2.2 Архітектура комбінованого підходу для запуску MATLAB моделей 36

2.3 Вибір програмних засобів 39

2.3 Паралельне виконання моделей MATLAB у середовищі Java 41

2.4 Проектування компонентів системи 44

2.4.1 MATLAB-bot мікросервіс 44

2.4.2 Task-proccessor мікросервіс 46

2.4.3 MATLAB-engine мікросервіс 49
2.4.4 Result-handler мікросервіс 51
2.5 Тестування працездатності розробленої системи 53
Висновок до розділу 2 54
РОЗДІЛ 3 ДОСЛІДЖЕННЯ РОЗРОБЛЕНОГО КОМБІНОВАНОГО ПІДХОДУ ДЛЯ ЗАПУСКУ МОДЕЛЮВАНЬ MATLAB МОДЕЛЕЙ 56
3.1 Складність алгоритмів 56
3.2 Опис процесу дослідження комбінованого підходу 57
3.3 Модель для обчислення суми елементів вектора зі складністю $O(n)$ 59
3.4 Модель для алгоритму бінарного пошуку зі складністю $O(log\ n)$ 61
3.5 Модель для сортування масиву методом злиття зі складністю $O(n\ log\ n)$ 62
3.6 Модель для сортування бульбашкою зі складністю $O(n2)$ 64
3.7 Модель для генерації перестановок зі складністю $O(n!)$ 66
3.8 Модель для розбиття множин зі складністю $O(nn)$ 68
3.9 Дослідження використання системних ресурсів під час використанням комбінованого підходу у порівнянні з традиційним 70
Висновок до розділу 3 72
РОЗДІЛ 4 РОЗРОБКА СТАРТАП ПРОЕКТУ «MATLABOT» 74
4.1 Опис та технологічний аудит ідеї стартап-проекту 74
4.2 Аналіз ринкових можливостей запуску стартап проєкту 78
4.3 Розроблення ринкової стратегії та маркетингової програми стартап-проєкту 89
4.4 Розроблення бізнес-моделі стартап-проєкту 96
4.5 Висновки до розділу 4 100
ВИСНОВОК 102
ВИКОРИСТАНІ ДЖЕРЕЛА 104

ДОДАТОК А Скрипт для демонстрації роботи системи 107

ДОДАТОК Б Моделі різної складності 109

ДОДАТОК В Лістинг коду програмної реалізації 113

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

GPU – графічний процесор

CPU – центральний процесор

PCT – Paralel Computing Toolbox

RAM – Random Access Memory

ВСТУП

Зростання обчислювальних потреб у сучасних інженерних задачах висуває нові вимоги до ефективності моделювання. MATLAB, як один із провідних інструментів для виконання складних симуляцій, забезпечує широкі можливості для розробки моделей. Однак, традиційні підходи до використання MATLAB часто обмежуються послідовною обробкою даних, або потребують ручного налаштування для інтеграції з іншими системами, що може створювати додаткові труднощі в роботі з великими обсягами даних.

Впровадження підходів, які об'єднують багатопотокову обробку і інтеграцію зі сторонніми системами, дозволяє значно підвищити швидкість виконання симуляцій. Це відкриває нові можливості для застосування MATLAB у сферах, де потрібна висока продуктивність та автоматизація рутинних процесів.

Попри існування сучасних засобів паралельного та розподіленого обчислення, їх впровадження часто вимагає спеціалізованих знань та ресурсів для налаштування. Водночас, інтеграція MATLAB із зовнішніми інструментами, такими як Java, дозволяє створювати більш адаптивні та функціональні рішення.

Мета роботи. Розроблення комбінованого підходу та програмного забезпечення для ефективного моделювання MATLAB моделей із використанням багатопотоковості.

Завдання дослідження:

- Провести огляд стану проблеми.
- Провести огляд існуючих рішень та підходів у сфері багатопотокового моделювання в MATLAB.
- Розробити архітектуру програмної реалізації, основні модулі та підсистеми.
- Перевірити працездатність системи.

- Провести дослідження ефективності розробленого комбінованого підходу для запуску MATLAB моделей.

Об’єктом дослідження є процес автоматизованого моделювання інженерних розрахунків і процесів в інтегрованому програмному середовищі.

Предметом дослідження методи та програмні засоби автоматизованої багатопотокової симуляції MATLAB моделей шляхом інтеграції з мовою програмування Java.

Наукова новизна дисертації полягає у наступному**:**

- запропоновано комбінований підхід для моделювання MATLAB моделей, який відрізняється від існуючих методів реалізацією на основі мікросервісної архітектури з інтеграцією Telegram-бота в якості інтерфейсу користувача, з поєднанням багатопотокового середовища Java, що надає змогу пришвидшити процес симуляції, а також ефективніше використовувати системні ресурси;
- автоматизовано процес запуску моделювань для підвищення ефективності обробки великої кількості MATLAB моделей.

Практичне значення отриманих результатів визначається у застосуванні сучасних методів розробки, зокрема, асинхронної обробки запитів і розбиття функціоналу системи на окремі мікросервіси, та реалізації програмного забезпечення для нового підходу до запуску моделювань у багатопотоковому середовищі Java за допомогою MATLAB Engine API та інструментів: Spring Framework, пакету java.util.concurrent, Telegram Bot API, що надає можливість ефективно обробляти MATLAB моделі на віддаленому сервері без необхідності встановлення пакету MATLAB.

Апробація результатів дисертації. Результати роботи були обговорені на наступній конференції:

- XIX Наукова-практична конференція студентів, аспірантів та молодих вчених «Ефективність та автоматизація інженерних рішень у приладобудуванні», 04-05 грудня 2024 року.

Публікації. За матеріалами дисертації було опубліковано 1 тезу доповідь на

науково-практичній конференції.

Головань Я. В., студент гр. ПГ-31мп, к.т.н., Рупіч С.С.. КПІ ім. Ігоря Сікорського. Комбінований підхід для моделювання MATLAB моделей. Ефективність та автоматизація інженерних рішень у приладобудуванні (04-05 грудня 2024 року), КПІ ім. Ігоря Сікорського, приладобудівний факультет, Київ, 2024.

РОЗДІЛ 1

ОГЛЯД СТАНУ ПРОБЛЕМИ

1.1 Математичне моделювання та програмне середовище MATLAB

Математичне моделювання є важливим інструментом для вирішення складних наукових, інженерних, економічних та інших завдань. Воно дозволяє будувати абстрактні моделі реальних процесів і систем, які важко або неможливо досліджувати безпосередньо через їх складність, вартість або небезпеку. За допомогою моделювання можна оцінити поведінку системи, проводити експерименти, оптимізувати процеси та приймати важливі рішення на основі результатів аналізу.

З огляду на стрімкий розвиток технологій та зростання складності сучасних задач, моделювання стало невід'ємною частиною багатьох наукових дисциплін і галузей. Воно широко застосовується в інженерії для проектування нових механізмів, в економіці для аналізу ринкових процесів, у фізиці для симуляції явищ на мікро- та макрорівнях, а також в багатьох інших сферах.

Одним з найпоширеніших інструментів для математичного моделювання є програмне середовище MATLAB [1]. Його можливості включають роботу з векторними і матричними обчисленнями, симуляції динамічних систем, обробку сигналів, аналіз даних та побудову графіків. Завдяки широкому спектру вбудованих функцій, цей програмний пакет дозволяє створювати моделі високої складності і точно передбачати результати в різних ситуаціях.

MATLAB є продуктивною інтегрованою системою для виконання числових обчислень, моделювання та візуалізації даних [1]. В сучасних науково-дослідних і технічних галузях даний пакет програм часто використовується для моделювання складних систем і процесів. Програмне забезпечення від розробника MathWorks надає

можливість швидко прототипувати рішення і виконувати їхню валідацію за допомогою широкого спектра інструментів для чисельних обчислень і моделювання. Його інтуїтивний інтерфейс та можливості візуалізації забезпечують гнучкість при дослідженні та аналізі великих обсягів даних.

MATLAB часто використовується в галузях, де важлива точність і швидкість обчислень. Дане програмне забезпечення є незамінним інструментом для проектування і моделювання керуючих систем. Це включає розробку автоматизованих систем керування для промислових і механічних процесів, а також алгоритмів для автономних транспортних засобів і роботизованих систем.

MATLAB пропонує широкий спектр бібліотек і спеціалізованих інструментів для виконання різноманітних задач. Серед них варто виділити [1]:

- Signal Processing Toolbox для аналізу і фільтрації сигналів;
- Simulink для моделювання динамічних систем за допомогою блок-схем
- Optimization Toolbox для вирішення задач оптимізації;
- Statistics and Machine Learning Toolbox для аналізу даних та побудови моделей машинного навчання;
- Control System Toolbox для проєктування та аналізу систем автоматичного керування;
- Image Processing Toolbox для роботи з цифровими зображеннями;
- Deep Learning Toolbox для створення та навчання нейронних мереж.

Завдяки цим інструментам MATLAB є універсальним рішенням для розробки, моделювання та аналізу систем у різних галузях техніки та науки. Він дозволяє ефективно працювати з великими обсягами даних та моделювати складні процеси, забезпечуючи гнучкість і точність обчислень.

Проте, попри значний прогрес у програмному забезпеченні для моделювання, існують певні виклики, пов’язані з ефективністю та продуктивністю таких систем. Особливо це стосується моделювання алгоритмів різної складності, де обчислювальні витрати суттєво зростають із ускладненням математичних моделей.

Однією з головних проблем традиційного моделювання є високі вимоги до обчислювальних ресурсів, особливо коли мова йде про великомасштабні моделі або

складні динамічні системи. У таких випадках для точного моделювання потрібно залучати значну кількість оперативної пам’яті та потужних процесорів для забезпечення швидкої обробки великих обсягів даних. Це створює певні обмеження для дослідників, особливо коли необхідно працювати з моделями, що мають багато параметрів або використовують великі набори даних. Наприклад, моделювання фізичних процесів у реальному часі вимагає швидкого аналізу та обробки даних, що не завжди можливо через апаратні обмеження [2].

Крім того, традиційні методи моделювання часто не справляються з ефективною роботою в умовах паралельних обчислень або з багатоядерними системами. Багато класичних алгоритмів були розроблені для послідовного виконання, що суттєво обмежує продуктивність при спробах розподілити обчислення на кілька процесорів або ядер. Це особливо важливо в умовах зростання потреб у масштабуванні моделей та використанні паралельних обчислень для прискорення симуляцій. Незважаючи на те, що сучасні технології дозволяють використовувати багатоядерні процесори та графічні процесори (GPU) для паралельної обробки, багато програмних середовищ мають обмежену підтримку таких можливостей, або їх інтеграція є складною та трудомісткою [3].

Ще одним суттєвим обмеженням є складність роботи з великими обсягами даних. Коли моделі повинні враховувати величезні масиви даних, наприклад, у задачах машинного навчання, аналізу великих масивів інформації або моделюванні фізичних систем, виникає проблема оптимізації пам'яті та швидкості обробки. Великі дані вимагають не лише потужних обчислювальних ресурсів, а й ефективних алгоритмів, які можуть працювати з ними без перевантаження пам’яті. Це призводить до необхідності використовувати методи оптимізації або розподілених обчислень, що ускладнює процес моделювання [4].

Моделювання в реальному часі також стикається з низкою обмежень через продуктивність. Багато систем потребують миттєвої реакції на зовнішні фактори або зміни в умовах середовища, що вимагає високої швидкості обчислень та мінімізації затримок. У традиційних середовищах моделювання часто не забезпечується достатній рівень продуктивності для таких завдань. Це стає критичним у системах

управління або навігації, де відсутність своєчасної реакції може призвести до значних помилок або навіть до аварійних ситуацій.

Таким чином, традиційне моделювання має ряд важливих обмежень, які обумовлені як технічними аспектами, так і недостатньою адаптацією до сучасних викликів, таких як паралельні обчислення, великі дані та реальний час [5].

1.2 Основні принципи комбінованого підходу у моделюванні

1.2.1 Загальна характеристика паралельних обчислень

Паралельні обчислення є фундаментальним методом, що дозволяє значно підвищити продуктивність і скоротити час виконання обчислювальних завдань шляхом одночасного виконання кількох задач. Цей підхід базується на розподілі великого завдання на менші підзадачі, які можуть бути оброблені одночасно на різних обчислювальних ресурсах. На рис. 1 наведено відмінності в процесах послідовних та паралельних обчислень.

Можливість одночасної обробки багатьох задач дозволяє значно підвищити ефективність та оптимізувати використання ресурсів, однак реалізація паралельних обчислень пов'язана з низкою технічних викликів. Розробка ефективних алгоритмів вимагає врахування таких аспектів, як синхронізація потоків, мінімізація залежностей між задачами, а також оптимальний розподіл навантаження для уникнення простоїв і перевантаження потоків. Крім того, у розподілених системах значну роль відіграють витрати на передачу даних між потоками, які можуть суттєво впливати на загальну продуктивність системи [6].

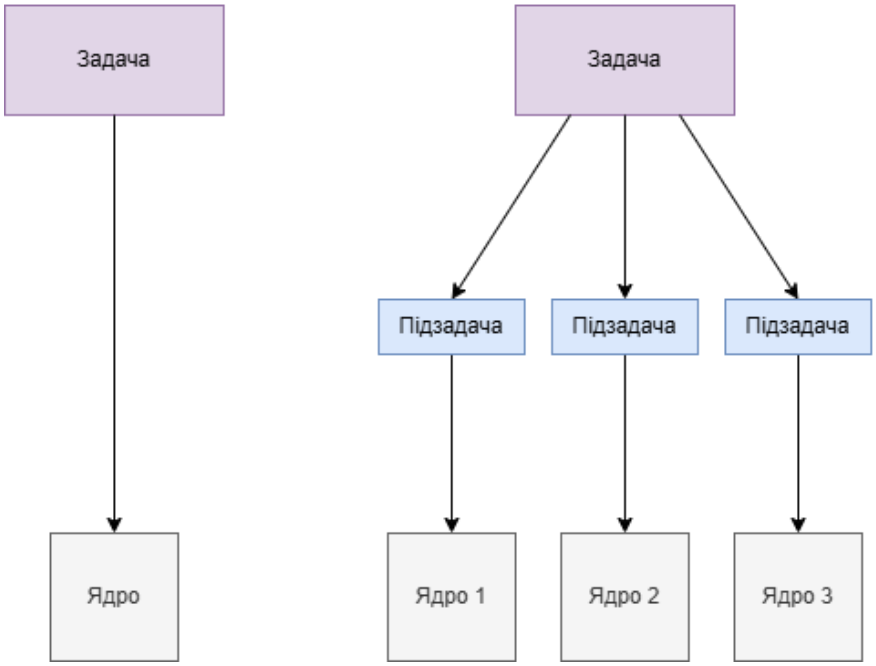

Рисунок 1 – Принципи послідовних та паралельних обчислень (праворуч – паралельне виконання, ліворуч – послідовне виконання)

Попри ці виклики, паралельні обчислення є незамінним інструментом у багатьох галузях. Вони широко використовуються для моделювання складних фізичних явищ, аналізу великих масивів даних, машинного навчання та багатьох інших завдань, де точність і швидкість обчислень є критично важливими показниками. Таким чином, паралельні обчислення є ефективним інструментом, що відкриває нові можливості для оптимізації обчислювальних процесів і підвищення їх продуктивності.

1.2.2 Комбінований підхід до моделювання MATLAB моделей

Зростання потреби в складних обчислювальних моделях стало однією з визначальних тенденцій сучасного розвитку технологій. Відповідно до нових вимог до точності, швидкості та масштабованості, методи моделювання, зокрема за допомогою MATLAB, повинні постійно вдосконалюватися. В умовах швидкого

зростання обсягу даних і складності систем, що потребують моделювання, традиційні підходи часто виявляються недостатньо ефективними. MATLAB, завдяки своїй гнучкості та ефективним функціям, пропонує безліч інструментів для аналізу, візуалізації та реалізації алгоритмів, що робить його незамінним у різних галузях, таких як інженерія, фізика, економіка та інформаційні технології [7].

Комбінований підхід до моделювання MATLAB моделей полягає у поєднанні стандартних можливостей середовища MATLAB із зовнішніми технологіями та інструментами для досягнення високої ефективності, продуктивності та зручності використання. Одним із ключових аспектів цього підходу є інтеграція паралельних обчислень, що дозволяють виконувати кілька моделювань одночасно. Паралельні обчислення сприяють раціональному використанню обчислювальних ресурсів, зменшуючи час виконання задач та підвищуючи загальну продуктивність. У цьому контексті кожне моделювання запускається як незалежний процес. Традиційні алгоритми часто не здатні обробляти великі обсяги даних за короткий час, що може призводити до затримок проведення моделювання. Паралельні обчислення, як новий підхід до вирішення цих проблем, здатні оптимізувати роботу з моделями, дозволяючи виконувати симуляції швидше [8].

Для поєднання функціоналу MATLAB із зовнішніми системами, компанія MathWorks надає спеціальний інструмент – MATLAB Engine API. Цей інструмент підтримує три основні мови програмування: Java, Python та C++. Кожна з цих мов має свої переваги та недоліки, які варто враховувати залежно від поставлених задач. Python забезпечує простоту використання і зручний синтаксис, але його модель багатопоточності, обмежена GIL (Global Interpreter Lock), не дозволяє повноцінно реалізовувати паралельне виконання задач. Це є серйозним недоліком, враховуючи вимогу системи до ефективного паралельного запуску численних моделей MATLAB для підвищення продуктивності. C++, у свою чергу, є надзвичайно продуктивною мовою, що дозволяє реалізовувати високоефективні обчислення, але вимагає значних ресурсів для розробки та підтримки, зокрема через складність синтаксису і необхідність ручного управління пам’яттю.

Java поєднує зручність розробки з високою продуктивністю. Важливо відзначити, що багатопоточність у Java реалізована на основі повноцінної моделі потоків, яка дозволяє ефективно використовувати ресурси багатоядерних процесорів. Завдяки цьому Java забезпечує можливість виконання кількох моделей MATLAB одночасно, що є критично важливим для паралельної обробки даних. Використання сучасних інструментів Java, таких як Spring Framework та його компонентів, зокрема Spring Boot, дозволяє значно скоротити час розробки та забезпечити високу гнучкість архітектури. Також варто зазначити, що Spring Data, який є частиною Spring Framework, забезпечує ефективну взаємодію з базами даних, спрощуючи реалізацію репозиторіїв та операцій над даними [9].

Ще одним важливим елементом комбінованого підходу є інтеграція сторонніх технологій, які усувають необхідність прямої взаємодії користувачів із середовищем MATLAB. Наприклад, використання Telegram-ботів для ініціювання та моніторингу моделювань дозволяє значно спростити процес для кінцевих користувачів. Завдяки цьому підходу, команди для запуску моделей можуть передаватися віддалено, що не тільки розширює функціональність системи, але й усуває потребу у встановленні MATLAB на стороні користувача.

1.3 Огляд існуючих рішень та підходів у сфері багатопотокового моделювання в MATLAB

Одним із ключових інструментів MATLAB для реалізації паралельного моделювання є Parallel Computing Toolbox (PCT) [3]. Цей пакет дозволяє використовувати багатоядерні процесори, GPU та обчислювальні кластери для виконання паралельних обчислень. Основними компонентами PCT є цикл parfor, механізм створення пулу робітників (workers), розподілені обчислення за допомогою конструкції spmd та інтеграція з GPU через функції, оптимізовані для паралельної

роботи. Інструменти PCT спрощують процес перенесення існуючих алгоритмів на багатоядерну архітектуру без потреби значної модифікації вихідного коду.

Іншим ефективним засобом є механізм Batch Processing, який забезпечує можливість виконання обчислень у фоновому режимі [10]. Ця технологія дозволяє запускати процес моделювання на віддалених робочих станціях або у локальному середовищі, мінімізуючи блокування основного потоку MATLAB.

Розподілені масиви (Distributed Arrays) надають можливість виконувати обчислення над великими обсягами даних, які розподіляються між кількома вузлами [11]. Це особливо корисно для задач, що перевищують доступну оперативну пам’ять локальної машини. Функції distributed та codistributed дозволяють організувати ефективний розподіл даних та виконання над ними базових операцій у паралельному режимі.

Для виконання обчислень із використанням графічних процесорів MATLAB підтримує GPU Computing, що дозволяє значно прискорити обробку інтенсивних обчислень [12]. Інтеграція GPU виконується через об'єкти gpuArray, які автоматично адаптують операції під архітектуру графічного процесора.

На масштабованому рівні для роботи з великими обчислювальними кластерами MATLAB пропонує MATLAB Parallel Server, що дозволяє виконувати паралельні задачі в розподілених обчислювальних середовищах.

1.3.1 Паралельні цикли (parfor)

Цикл parfor є одним із ключових інструментів для реалізації паралельних обчислень у MATLAB, що забезпечує ефективне використання багатоядерних процесорів [13]. Він входить до складу Parallel Computing Toolbox і дозволяє виконувати ітерації циклу паралельно на кількох ядрах або робітниках (workers), значно скорочуючи час обчислень.

Принцип роботи parfor полягає в тому, що MATLAB автоматично розподіляє ітерації між доступними потоками (рис. 2), забезпечуючи їх незалежне виконання. Кожна ітерація виконується окремо, а результати об'єднуються після завершення всіх обчислень. Це спрощує програмування, оскільки користувачеві не потрібно вручну управляти потоками чи синхронізувати дані.

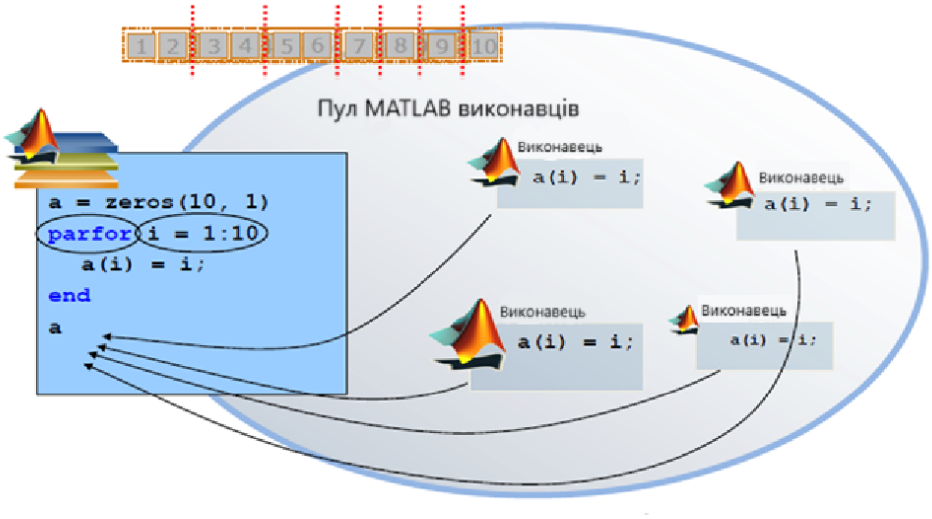

Рисунок 2 – Механізм паралельних циклів [14]

Головною перевагою parfor є простота використання. Для інтеграції паралельного циклу достатньо замінити звичайний цикл for на parfor, що мінімізує необхідність змін у вихідному коді. MATLAB автоматично виконує розподіл ітерацій, управління потоками та синхронізацію даних. Інструмент підходить для розв'язання задач із великим числом незалежних ітерацій, таких як чисельне інтегрування, моделювання фізичних систем або обробка великих наборів даних. Наприклад, у задачах обробки зображень parfor дозволяє паралельно обробляти кадри або виконувати фільтрацію зображень для значного скорочення часу обчислень [13].

Проте використання parfor має певні обмеження, які необхідно враховувати під час розробки моделей. Насамперед усі ітерації циклу повинні бути незалежними, тобто результати однієї ітерації не можуть залежати від попередніх. Це означає, що для адаптації алгоритму до parfor можуть знадобитися значні зміни. MATLAB також накладає обмеження на типи змінних, які використовуються в циклі. Наприклад,

змінні повинні бути класифіковані як sliced (масиви, поділені за індексами) або broadcast (константи, доступні всім потокам), тоді як глобальні змінні у parfor не підтримуються [13].

Окремим викликом є накладні витрати, пов’язані з ініціалізацією пулу робітників, які можуть знизити ефективність для коротких циклів із невеликою кількістю ітерацій. Крім того, синхронізація даних між потоками може створювати додаткові затримки, особливо при роботі з великими обсягами даних. Для забезпечення максимальної ефективності рекомендується мінімізувати кількість змінних, які передаються між потоками, використовувати локальні змінні для зменшення витрат пам’яті та рівномірно розподіляти обчислення між потоками.

Цикл parfor знаходить широке застосування в наукових дослідженнях і технічних задачах, зокрема для моделювання фізичних систем, розрахунку числових інтегралів та інших складних завдань. Наприклад, у задачах числового інтегрування можна паралельно виконувати розрахунки для великої кількості точок, а в обробці зображень – фільтрувати десятки кадрів одночасно [13]. Завдяки автоматичному управлінню потоками parfor забезпечує баланс між продуктивністю та простотою інтеграції в існуючий код.

Таким чином, parfor є ефективним засобом для організації паралельних обчислень, особливо в задачах із незалежними ітераціями. Його використання дозволяє значно прискорити розрахунки та зменшити час виконання моделей, водночас зберігаючи простоту програмування. Проте його ефективність залежить від специфіки задачі, правильної організації даних і мінімізації накладних витрат.

1.3.2 Використання spmd

Конструкція spmd (single program, multiple data) у MATLAB забезпечує ефективну платформу для реалізації паралельних обчислень, коли необхідно виконувати одну програму на кількох потоках із можливістю розподілу та обміну

даними між ними [15]. Основною перевагою цього підходу є його гнучкість, яка дозволяє налаштовувати розподіл даних між потоками та забезпечувати синхронізацію результатів у разі потреби (рис. 3). Завдяки цьому spmd використовується в задачах, що виходять за межі можливостей циклу parfor, зокрема в складних моделюваннях і обчисленнях, де потрібна взаємодія між потоками.

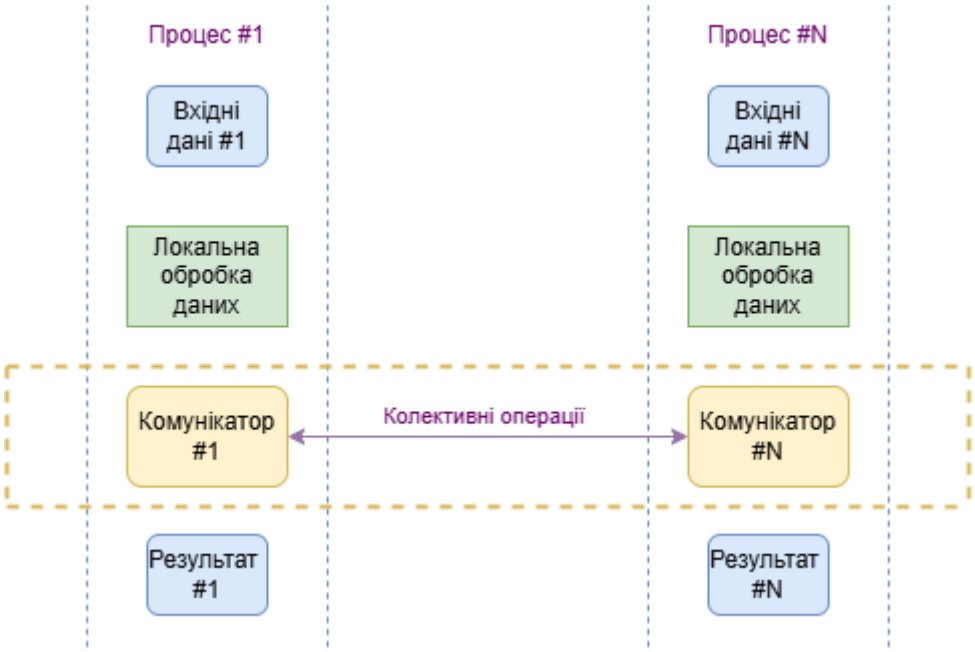

Рисунок 3 – Принцип роботи spmd [16]

Усі робітники, що входять до пулу Parallel Computing Toolbox, виконують той самий код, визначений у блоці spmd. Це забезпечує однаковість виконання завдання, але кожен потік працює зі своєю частиною даних. Такий підхід дозволяє розподіляти великі масиви даних між потоками та зменшувати обсяг пам'яті, необхідної для кожного окремого робітника. Наприклад, у випадках, коли доступний обсяг пам’яті обмежений, spmd забезпечує розподіл даних між кількома обчислювальними вузлами, дозволяючи ефективно виконувати завдання навіть із дуже великими наборами даних [15].

Важливою особливістю конструкції spmd є можливість синхронізації даних між потоками. MATLAB пропонує вбудовані функції для колективних операцій, таких як об’єднання результатів, обчислення середніх значень або підсумовування.

Це дозволяє реалізовувати більш складні алгоритми, які потребують інтеракції між потоками, зберігаючи при цьому контроль над даними на кожному окремому потоці. Завдяки таким функціям spmd стає особливо корисним у задачах із нерівномірним розподілом навантаження, коли кожен потік виконує різні обсяги роботи залежно від специфіки даних.

Однак використання spmd вимагає від розробника більшого рівня розуміння парадигм паралельного програмування. Зокрема, важливо правильно організувати розподіл даних між потоками, щоб уникнути надлишкових обчислень або неефективного використання пам’яті. Слід також враховувати накладні витрати, пов’язані із синхронізацією потоків, особливо якщо обсяги даних великі, а обчислення в кожному потоці — короткотривалі. У таких випадках витрати на комунікацію між потоками можуть перевищувати виграш від паралельного виконання.

Конструкція spmd знаходить широке застосування у складних моделюваннях, таких як симуляція фізичних систем, аналіз великих обсягів даних та обчислення, що потребують високої продуктивності. Її використання дозволяє ефективно масштабувати обчислення на обчислювальних кластерах або великих серверних системах, забезпечуючи при цьому високу гнучкість у розподілі ресурсів. Таким чином, spmd є ефективним засобом для організації паралельних обчислень, що дозволяє реалізовувати складні моделі з високою продуктивністю та ефективністю.

1.3.3 Моделювання на основі GPU

Моделювання на основі GPU у MATLAB є одним із найсучасніших та найбільш продуктивних підходів до виконання обчислень, що вимагають значної кількості паралельних операцій. Графічні процесори, завдяки своїй архітектурі, здатні обробляти тисячі паралельних потоків, що робить їх надзвичайно ефективними для задач із високим ступенем паралелізму, таких як обробка великих масивів даних,

чисельне інтегрування, розв’язання систем рівнянь, моделювання фізичних процесів та навчання нейронних мереж. MATLAB надає розширені засоби для інтеграції обчислень на GPU, забезпечуючи користувачам зручний інструментарій для оптимізації своїх моделей [12].

Графічні процесори мають суттєві відмінності від центральних процесорів (CPU). Основна їхня перевага — це велика кількість ядер, які можуть виконувати прості обчислювальні операції одночасно. У той час як CPU оптимізовані для послідовного виконання складних завдань, GPU орієнтовані на обробку масивних наборів даних. MATLAB інтегрує підтримку GPU через спеціальні інструменти, такі як Parallel Computing Toolbox, що дозволяє переносити обчислення на GPU із мінімальними змінами в коді [17]. Це стало можливим завдяки об'єктам gpuArray, які автоматично адаптують операції для виконання на графічному процесорі, використовуючи потужності NVIDIA CUDA.

Основна перевага використання GPU полягає в значному прискоренні обчислень (рис. 4).

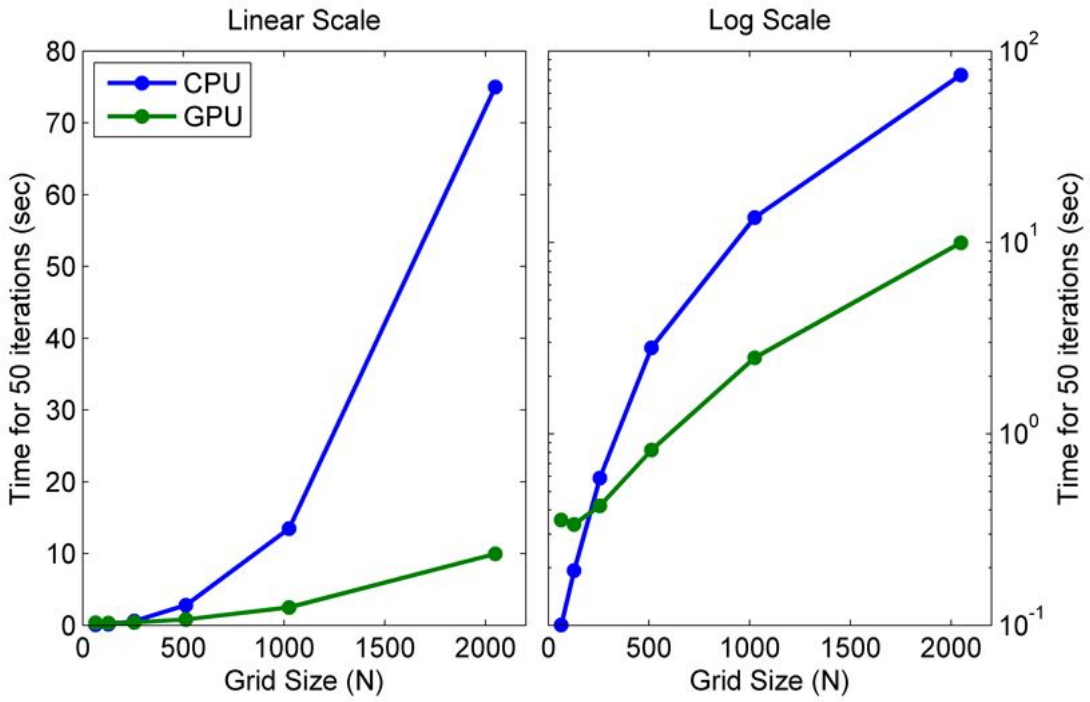

Рисунок 4 – Порівняння швидкості обчислень з використанням CPU та GPU [12]

Для задач, які включають великі паралельні обчислення, такі як множення великих матриць, обробка сигналів або аналіз зображень, графічний процесор може забезпечити прискорення в десятки разів порівняно з CPU. Наприклад, у задачах машинного навчання використання GPU дозволяє швидше тренувати нейронні мережі, обробляючи великі обсяги даних паралельно. MATLAB забезпечує оптимізацію таких обчислень, автоматично використовуючи потужності графічного процесора для базових операцій, таких як лінійна алгебра, FFT (швидке перетворення Фур'є) та операції з великими масивами даних.

Ще однією перевагою GPU у MATLAB є спрощена інтеграція із вже існуючими алгоритмами. Використовуючи gpuArray, користувачі можуть адаптувати свій код до виконання на GPU без необхідності переписувати значні частини програми. Наприклад, матриця, створена як gpuArray, автоматично виконує всі подальші операції на графічному процесорі, що спрощує реалізацію складних обчислювальних моделей. Крім того, MATLAB надає можливість використання власних CUDA-функцій через інтеграцію із CUDA-кодом, що дозволяє реалізовувати користувацькі алгоритми для виконання на GPU.

Проте моделювання на основі GPU має свої обмеження. Насамперед, не всі задачі можуть бути ефективно розпаралелені. Алгоритми із сильно вираженою залежністю між етапами обчислень або з низьким ступенем паралелізму можуть не отримати значного прискорення на GPU [12]. Крім того, передача даних між пам'яттю CPU та GPU створює додаткові накладні витрати, що може знижувати продуктивність, особливо для невеликих задач, які не використовують повністю обчислювальну потужність GPU. Іншим важливим фактором є вимоги до обладнання: для ефективного використання GPU необхідно мати сучасну графічну карту з достатнім обсягом відео пам'яті та підтримкою технологій CUDA [12].

Іншою проблемою є обмежені можливості GPU для роботи з великими обсягами даних, що перевищують обсяг його пам'яті. Для подолання цього обмеження необхідно використовувати розподілені обчислення або методи, які дозволяють обробляти дані поетапно [13]. Крім того, оптимізація алгоритмів для GPU

вимагає від користувачів розуміння архітектури графічного процесора, особливостей управління пам’яттю та організації обчислень.

Незважаючи на значні переваги, які надають підходи до багатопотокового та паралельного моделювання в MATLAB, існують кілька важливих недоліків, які обмежують їх ефективне використання в ряді реальних задач. Основним обмеженням є складність реалізації паралельності та організації одночасного моделювання багатьох моделей, оскільки MATLAB не має вбудованої підтримки для такого сценарію в повному обсязі.

Перше суттєве обмеження полягає в тому, що хоча MATLAB надає потужні інструменти для паралельних обчислень, такі як цикли parfor, конструкції spmd та інтеграцію з GPU, їх застосування вимагає значної адаптації існуючого коду. Моделювання, яке включає розподіл обчислень на кілька процесорів або робітників, часто вимагає зміни структури програми, щоб забезпечити незалежність ітерацій або коректну синхронізацію даних між потоками. У випадку задач, де моделі мають залежності між етапами або вимагають спільної роботи між обчислювальними одиницями, адаптація до паралельного виконання стає складним і трудомістким процесом. Це вимагає від користувача не лише знань паралельного програмування, а й розуміння специфіки алгоритмів, для яких потрібно застосовувати паралелізм.

Другим важливим недоліком є те, що MATLAB не підтримує одночасне моделювання великої кількості моделей без додаткових інструментів або програмних рішень. Хоча існуючі інструменти дозволяють розподіляти завдання серед кількох робітників або виконувати окремі частини моделі паралельно, MATLAB не має вбудованої підтримки для організації одночасного моделювання кількох моделей без суттєвих змін у коді. Це означає, що для кожної окремої моделі потрібно створювати окрему обчислювальну задачу, що ускладнює масштабування на велику кількість моделей, особливо коли необхідно обробляти тисячі або десятки тисяч моделей одночасно. Відсутність підтримки цього функціоналу у MATLAB обмежує використання платформи для задач, що потребують одночасного запуску численних моделей, як у випадку з великими симуляціями або обробкою даних у реальному часі.

Крім того, організація ефективної паралельності часто супроводжується значними накладними витратами на передачу даних між потоками або вузлами. У паралельних обчисленнях важливо мінімізувати затримки, які можуть виникати під час обміну інформацією між робітниками. Проте MATLAB, попри потужні інструменти для розподілених обчислень, все ще має обмеження в організації ефективної синхронізації даних. Це особливо стає очевидним при розподілі великих обсягів даних або обчислень, де часті комунікації між робітниками можуть значно знизити загальну продуктивність. У таких випадках досягнення високої продуктивності вимагає ретельної оптимізації, що є додатковим викликом для розробників.

Однією з найбільших проблем, пов’язаних із паралельним моделюванням у MATLAB, є необхідність налаштування середовища для ефективного використання апаратних ресурсів. MATLAB потребує спеціальної конфігурації для роботи з великими обчислювальними кластерами або GPU, що потребує додаткових зусиль і ресурсів. Крім того, складність розподілених обчислень збільшується при використанні різних типів обчислювальних ресурсів, таких як графічні процесори, що вимагає оптимізації алгоритмів під специфіку кожної платформи.

Загалом, незважаючи на високий рівень інтеграції паралельних обчислень у MATLAB, реалізація багатопотокового та паралельного моделювання залишається складною задачею, яка вимагає значних зусиль для налаштування і оптимізації. Відсутність підтримки для одночасного моделювання великої кількості моделей, обмеження в організації ефективної комунікації між потоками та необхідність специфічної конфігурації для кожної апаратної платформи ускладнюють використання MATLAB для ряду реальних завдань. Тому, для досягнення максимальної ефективності необхідно застосовувати спеціалізовані стратегії паралелізму та ретельно оптимізувати код і обчислювальну архітектуру відповідно до специфіки задачі.

1.4 Аналіз наукових робіт за тематикою дисертації

Для оцінки актуальності даної роботи проведено аналіз робіт по використанню методів паралельних обчислень, розподілених обчислень та оптимізації на основі GPU для підвищення продуктивності симуляцій. Огляд наукових робіт спрямований на вивчення найновіших інструментів і технологій, таких як Parallel Computing Toolbox, MATLAB Distributed Computing Server, а також оптимізаційні алгоритми, що дозволяють значно скоротити час виконання складних моделей. Аналіз також дозволяє оцінити практичні переваги різних методів прискорення, вплив архітектури обчислювальних систем і оптимізаційних параметрів на результати моделювання.

Автори статті «MATLAB and parallel computing» [18], Магдалена та Пйотра Шимчик з AGH Університету науки і технологій, пропонують огляд можливостей MATLAB у контексті паралельних обчислень.

У статті розглядаються можливості MATLAB щодо паралельних обчислень, зокрема використання багатоядерних процесорів та графічних процесорів для прискорення розрахунків. Автори наводять приклади алгоритмів, що працюють у середовищі CUDA, та їх інтеграцію з MATLAB для прискорення обробки зображень. Інструменти Parallel Computing Toolbox та MATLAB Distributed Computing Server дозволяють використовувати до 8 ядер на робочих станціях та масштабувати обчислення на кластери з десятками або сотнями ядер.

Для великих масивів даних це дозволяє значно пришвидшити обчислення. Також описано роботу з розподіленими масивами, що дозволяє використовувати великі масиви даних, розподілені між кількома процесорами. Також описується інтеграція MATLAB з CUDA — мовою програмування для паралельних обчислень на платформах NVIDIA.

Результати експериментів показали, що GPU забезпечує значне прискорення обчислень. Ця публікація є важливою для тих досліджень, які використовують MATLAB для обробки зображень, моделювання та симуляції, особливо у поєднанні з паралельними обчисленнями на GPU.

Стаття Джеремі Кепнера «Parallel MATLAB for Multicore and Multinode Computers» [19] присвячена застосуванню MATLAB для паралельних обчислень на багатоядерних процесорах і багатовузлових комп'ютерних системах. Автор детально описує різні моделі паралельного програмування та їх реалізацію в MATLAB, зокрема використання розподілених масивів, моделі "менеджер/працівник" та передачі повідомлень.

Автор пояснює концепції паралельного програмування, такі як конкуренція та локальність, що мають вирішальне значення для досягнення високої продуктивності на багатоядерних системах. MATLAB спрощує процес створення паралельних програм, дозволяючи зосередитися на алгоритмах, а не на технічних аспектах паралельності.

Однією з ключових моделей, яку використовує автор, є розподілені масиви. Вони дозволяють розподіляти великі набори даних між ядрами або вузлами для прискорення обчислень. Автор підкреслює, що розподілені масиви підходять для 90% задач паралельних обчислень і є базовою моделлю для більшості користувачів MATLAB.

Окрім розподілених масивів, розглядаються моделі "менеджер/працівник" і передача повідомлень, які використовуються для більш складних задач, де потрібна взаємодія між процесами під час виконання.

Автор наводить низку практичних прикладів з реальних програм, які демонструють застосування паралельних моделей у MATLAB. Такі програми, як Mandelbrot, ZoomImage і ParallelIO, використовуються для ілюстрації технік проектування, кодування, налагодження та тестування паралельних програм.

У науковій публікації «GPU Cluster with MATLAB» [20] від авторів A. Guill´en M. Garc´ıa L. J. Herrera H. Pomares I. Rojas представлено інноваційну архітектуру гетерогенного кластера, що наведено на рис. 5, де кожен вузол оснащений одним або декількома графічними процесорами. Основною мотивацією дослідження стала здатність цієї технології демонструвати вражаючі результати у високопродуктивних обчисленнях при збереженні низької вартості та енергоспоживання. Особливу цінність представляє можливість програмування через MATLAB.

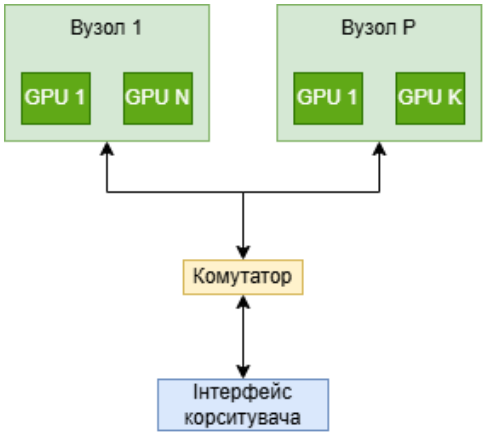

Рисунок 5 – Архітектура кластера, запропонована в дослідженні [20]

Запропонована архітектура базується на комбінації технологій MPI (Message Passing Interface) та CUDA, що забезпечує ефективну взаємодію між вузлами кластера та графічними процесорами. Важливою особливістю є можливість розгортання застосунків за допомогою MATLAB Compiler, що дозволяє створювати автономні програми, які можуть працювати незалежно від встановленого MATLAB. Архітектура підтримує гетерогенні комп'ютери та CPU, що робить її гнучкою у застосуванні.

Технічна реалізація включає використання MPImex для MPI-комунікації між вузлами та пряме вбудовування CUDA-функцій у MATLAB-код. Автори детально описують процес компіляції через nvmex та необхідні налаштування, а також особливості взаємодії між різними компонентами системи. Важливим аспектом є необхідність явного вказування GPU для кожного процесу, що забезпечує коректний розподіл обчислювального навантаження.

Незважаючи на певні обмеження, такі як необхідність розгортання застосунків через MATLAB Compiler та неможливість використання Parallel Computing Toolbox у розгорнутих застосунках, запропонована архітектура демонструє значний потенціал для наукових та інженерних застосувань. Вона забезпечує ефективне поєднання зручності програмування в MATLAB з потужністю GPU-обчислень, дозволяючи створювати масштабовані рішення для високопродуктивних обчислень.

Огляд наукових публікацій показав можливості використання паралельних обчислень, розподілених систем, а також графічних процесорів для підвищення

ефективності моделювань. Впровадження технологій, таких як CUDA, MPI, дозволяє ефективно розподілити обчислювальне навантаження між вузлами кластерів та графічними процесорами, що сприяє значному зниженню часу виконання складних симуляцій. Правильна організація паралельних процесів та оптимізація використання апаратних ресурсів системи можуть значно скоротити час виконання складних математичних моделей, що є важливим фактором у сучасних інженерних і наукових задачах.

1.5 Формування мети та завдання дослідження

Кожен із розглянутих підходів має свої обмеження та недоліки, пов’язані зі складністю реалізації, високими вимогами до апаратного забезпечення та необхідністю глибоких знань цих інструментів для їх правильного і ефективного використання.

Мета роботи. Розроблення комбінованого підходу та програмного забезпечення для ефективного моделювання MATLAB моделей із використанням багатопотоковості.

Завдання дослідження:

- Провести огляд стану проблеми.
- Провести огляд існуючих рішень та підходів у сфері багатопотокового моделювання в MATLAB.
- Розробити архітектуру програмної реалізації, основні модулі та підсистеми.
- Перевірити працездатність системи.
- Провести дослідження ефективності розробленого комбінованого підходу для запуску MATLAB моделей.

РОЗДІЛ 2

ПРОГРАМНА РЕАЛІЗАЦІЯ КОМБІНОВАНОГО ПІДХОДУ ДЛЯ ЗАПУСКУ МОДЕЛЮВАНЬ MATLAB МОДЕЛЕЙ

2.1 MATLAB Engine API. Інструмент для інтеграції MATLAB

MATLAB Engine API представляє собою програмний інтерфейс, що дозволяє розробникам взаємодіяти з MATLAB із зовнішніх додатків, написаних різними мовами програмування. Цей API забезпечує доступ до обчислювального ядра MATLAB для виконання математичних операцій, аналізу даних, моделювання та візуалізації, інтегруючи його з іншими програмними рішеннями. MATLAB Engine API є важливим компонентом сучасної інженерії, яка потребує автоматизації та розширеної обчислювальної функціональності в межах гетерогенних програмних екосистем [21].

Однією з основних характеристик MATLAB Engine API є здатність виконувати MATLAB-команди або скрипти із зовнішнього середовища, зокрема із програм, написаних мовами Java, Python, C++ та іншими. Це дозволяє запускати MATLAB як фоновий процес, виконувати обчислення, передавати параметри та отримувати результати. API підтримує інтеграцію MATLAB з додатками, які працюють у реальному часі, що робить його важливим інструментом для складних наукових і технічних розрахунків.

MATLAB Engine API також забезпечує можливість обміну даними між MATLAB і зовнішніми програмами. Це включає передачу скалярів, масивів, матриць та структур, а також підтримує сучасні формати даних, які використовуються в науці та інженерії. Ця функція є критично важливою для автоматизації обчислювальних задач, де дані генеруються або обробляються поза MATLAB [22].

Крім того, API дозволяє інтерактивно створювати графіки та зберігати їх у різних форматах, що забезпечує зручність візуалізації результатів розрахунків. MATLAB Engine API інтегрується з інструментами для створення звітів, презентацій та наукових публікацій, сприяючи візуалізації даних і представленні результатів у зрозумілій формі [23].

Асинхронна природа API дозволяє виконувати MATLAB-команди без блокування основного потоку програми, що забезпечує підвищену гнучкість і зручність у багатопотокових середовищах. Це відкриває можливість виконання складних обчислень у фоновому режимі без переривання основного процесу програми [21].

MATLAB Engine API знаходить застосування в багатьох сферах, включаючи моделювання, оптимізацію, машинне навчання, аналіз великих даних та обробку сигналів. Його використання дозволяє створювати комбіновані або альтернативні системи для виконання моделювань, оскільки функціонал MATLAB може бути розширений під будь-які потреби.

2.2 Архітектура комбінованого підходу для запуску MATLAB моделей

Запропонована архітектура системи, що показано на рис. 6, заснована на мікросервісному підході, що забезпечує чіткий розподіл функціональних обов’язків між компонентами, гнучкість у розширенні системи та оптимізацію її роботи. Система включає кілька взаємодіючих мікросервісів і використання бази даних для забезпечення надійності збереження даних.

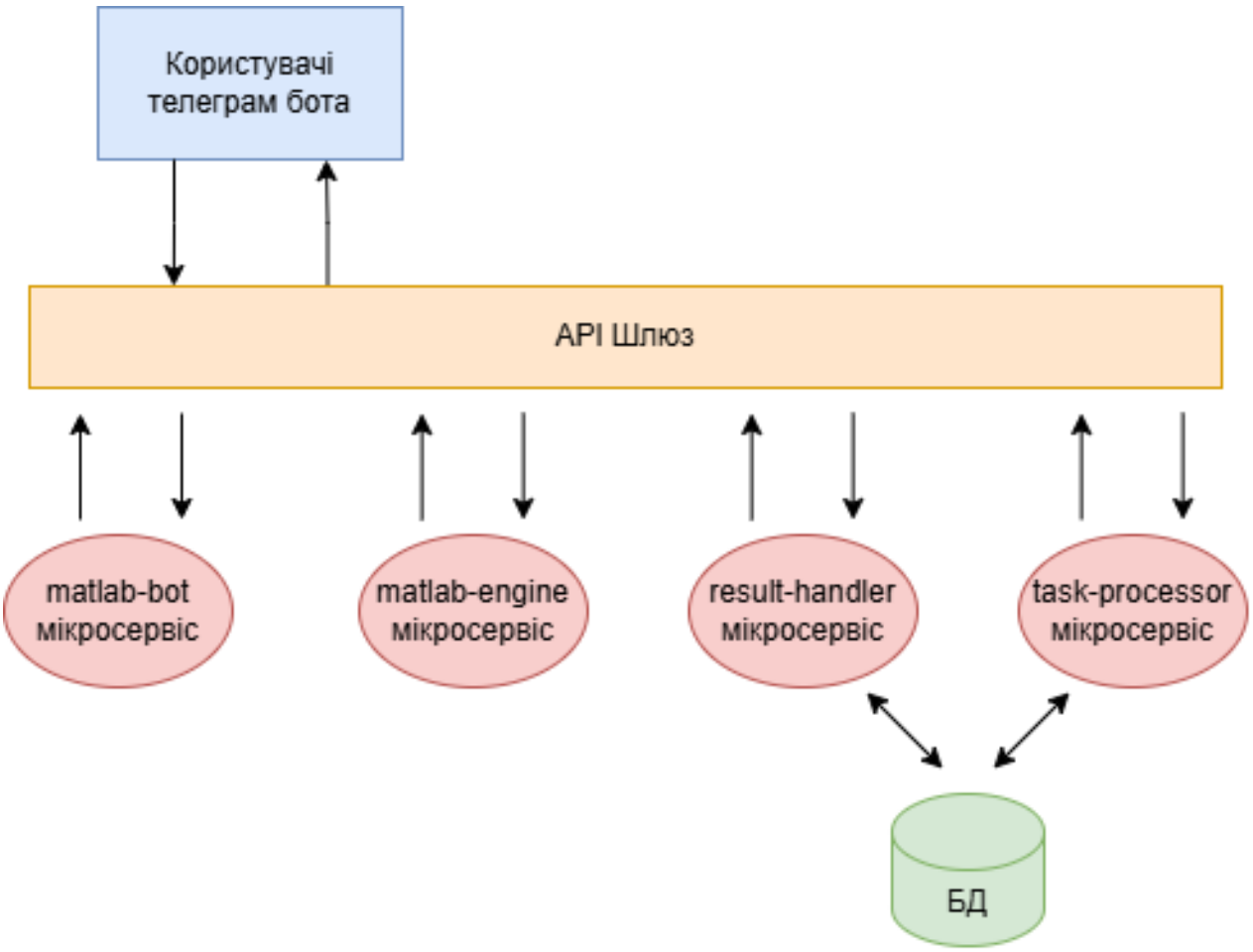

Рисунок 6 – Високорівнева архітектура системи

Для взаємодії мікросервісів між собою розроблено внутрішнє API, яке використовує REST запити для обміну даними. На рис. 7 наведено опис внутрішнього та зовнішнього API. Telegram-клієнт виступає як точка взаємодії користувача із системою. Користувач через Telegram API передає дані на MATLAB-bot microservice по зовнішньому API, який в свою чергу, відповідає за приймання запитів і файлів зі скриптами MATLAB, а також за взаємодію з іншими мікросервісами через API Gateway. API Gateway забезпечує маршрутизацію запитів між мікросервісами, ізолюючи зовнішні компоненти від внутрішньої архітектури системи, а також надає можливість масштабування системи через інтеграцію нових сервісів.

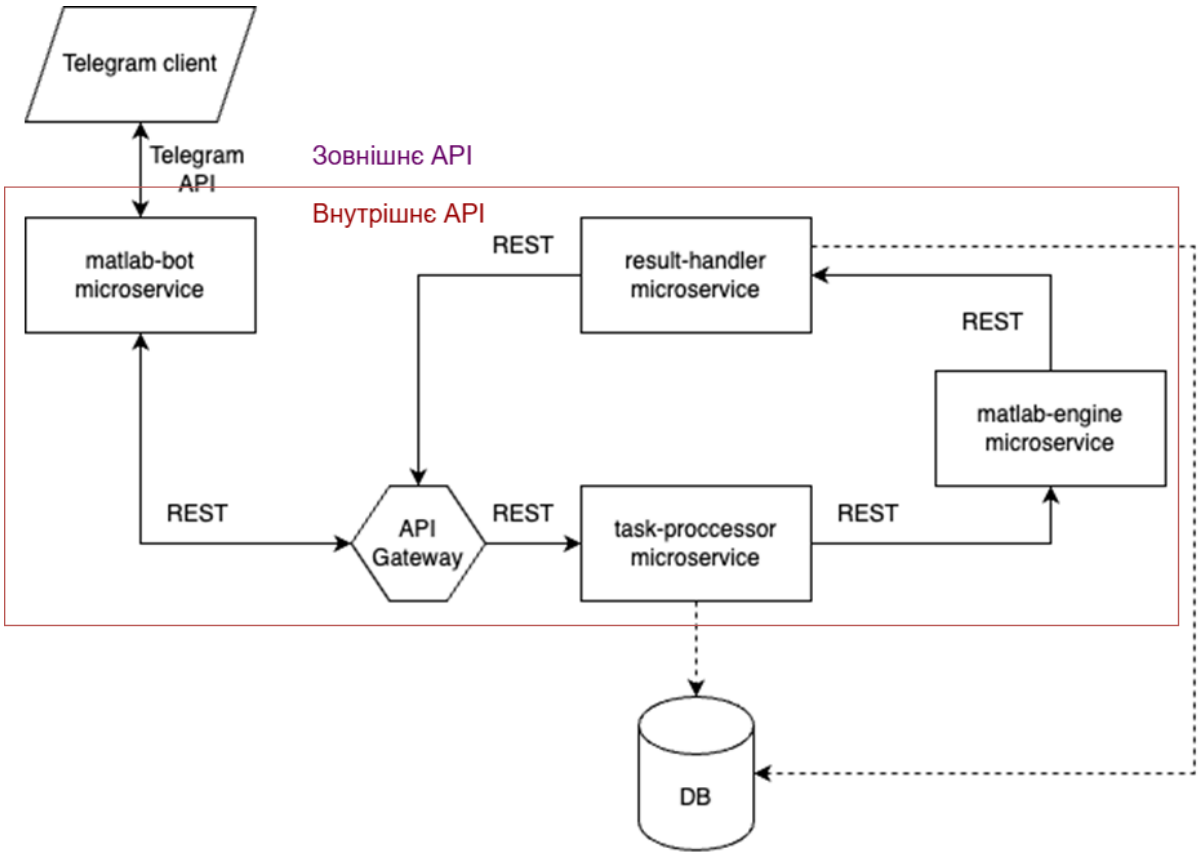

Рисунок 7 – Зовнішнє та внутрішнє API системи

Після передачі скрипта MATLAB від telegram-bot microservice, його обробка передається на task-processor microservice. Цей мікросервіс виконує критично важливі функції, зокрема, модифікацію вхідних скриптів. Task-processor аналізує переданий файл, додаючи до нього додаткові підскрипти для збереження графіків, якщо такі генеруються під час виконання моделі, а також забезпечує збереження метаданих, таких як модель, дані відправника та ідентифікатор моделі (model_id), у базі даних. Це дозволяє гарантувати надійність зберігання даних навіть у разі збоїв на наступних етапах обробки.

Після обробки в task-processor microservice скрипт передається на MATLAB-engine microservice, який відповідає виключно за виконання моделювання. MATLAB Engine, запущений у цьому мікросервісі, обробляє модифікований скрипт і генерує результати моделювання. Ці результати разом із model_id повертаються до result-handler microservice.

Result-handler microservice виконує функцію посередника між MATLAB Engine та Telegram Bot. Оскільки MATLAB-Engine microservice зосереджується виключно на виконанні математичних обчислень, result-handler microservice бере на себе роботу з базою даних. Отримавши model_id разом із результатами моделювання, result-handler звертається до бази даних для отримання інформації про користувача (chat_id), який надіслав модель. На основі отриманих даних result-handler передає результати моделювання разом із chat_id користувача на telegram-bot microservice.

Telegram-bot microservice отримує підготовлені результати та передає їх кінцевому користувачу через Telegram API. Такий підхід дозволяє розділити навантаження між сервісами, застерігаючи MATLAB Engine microservice від обробки зайвої інформації, пов’язаної з користувачами та передачею даних, що підвищує продуктивність і надійність обчислювального компонента системи.

Запропонована архітектура забезпечує чітке розмежування функцій між компонентами, масштабованість за рахунок використання API Gateway і модульність системи, що дозволяє легко інтегрувати нові сервіси або функції без значного впливу на існуючу архітектуру.

2.3 Вибір програмних засобів

До основних компонентів розробки програмної реалізації належать вибір бази даних та мови програмування. Для реалізації цієї системи було обрано MongoDB. Це документно-орієнтоване NoSQL-сховище відмінно підходить для зберігання інформації про моделі, їх статус виконання та користувачів. MongoDB забезпечує високу продуктивність під час масштабування, підтримує горизонтальне масштабування даних і надає зручні інструменти для роботи з JSON-подібними документами. З огляду на те, що система повинна обробляти великі обсяги даних у реальному часі, MongoDB дозволяє досягти високої швидкості запису та зчитування даних, зберігаючи при цьому їхню структурованість [24].

При виборі мов програмування для реалізації мікросервісної архітектури важливо враховувати як загальні вимоги системи, так і специфічні характеристики кожної з доступних мов. Вибір Java як основної мови програмування, у поєднанні з використанням Spring Framework [9] та MongoDB як бази даних, надає можливість створити масштабовану, продуктивну та зручну для підтримки систему. Цей підхід забезпечує як ефективну інтеграцію з MATLAB, так і можливість паралельного виконання моделей, що є ключовим аспектом для реалізації задач автоматизованої симуляції інженерних розрахунків.

Окрім основних компонентів, важливу роль у розробці системи відіграють допоміжні засоби, які забезпечують інтеграцію з зовнішніми інструментами та сервісами. Одним із ключових таких засобів є Telegram Bot API, що дозволяє реалізувати взаємодію між користувачами та системою через зручний інтерфейс Telegram [25]. Telegram Bot API надає широкий набір можливостей для обробки повідомлень, файлів, запитів команд і надсилання відповідей. Це робить Telegram ідеальним рішенням для створення ботів, які можуть отримувати та передавати дані, а також забезпечувати зворотний зв’язок із користувачем.

Для інтеграції Telegram Bot API у систему обрано бібліотеку Spring Telegram Bots, яка є розширенням Spring Framework для роботи з Telegram-ботами. Ця бібліотека спрощує розробку завдяки готовим рішенням для обробки вебхуків, роботи з командами та пересилання повідомлень. Використання Spring Telegram Bots забезпечує ефективне управління ботами через знайомі засоби Spring Framework, такі як залежності, конфігурації та контейнери компонентів [26].

Telegram Bot API також дозволяє працювати з файлами, що є критично важливим для системи, оскільки користувачі будуть надсилати скрипти MATLAB для обробки у вигляді m-файлів. Завдяки Spring Telegram Bots ця взаємодія стає більш простою, наприклад, бібліотека дозволяє автоматично обробляти повідомлення від користувачів, зберігати ідентифікатори (chat_id) та управляти логікою відповіді на основі отриманих даних.

2.3 Паралельне виконання моделей MATLAB у середовищі Java

В даній реалізації для паралельного виконання MATLAB моделей використовується концепція багатопоточності з використанням ExecutorService в Java. ExecutorService є частиною стандартної бібліотеки Java, яка надає високорівневий інтерфейс для асинхронного виконання завдань у фонових потоках. У даній реалізації використовується ThreadPoolExecutor (рис. 8), що дозволяє створювати пул потоків для виконання кількох завдань одночасно.

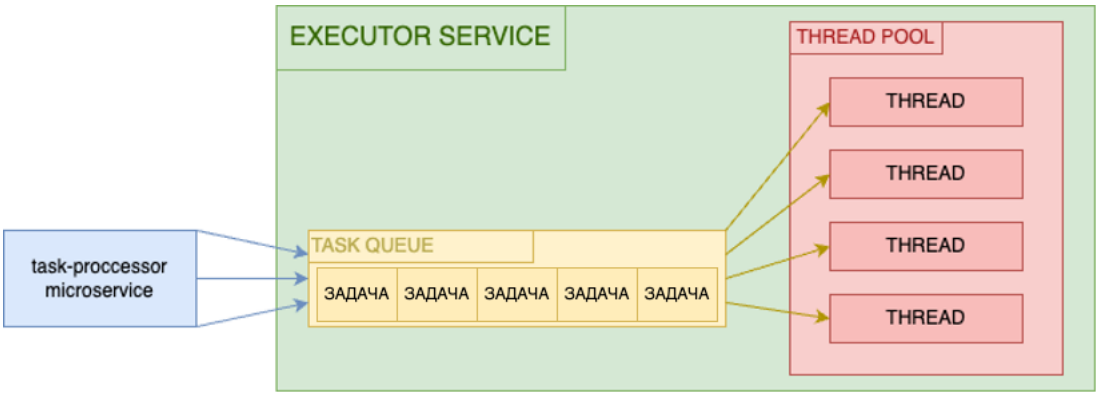

Рисунок 8 – Принцип паралельної реалізації обробки моделей

У процесі реалізації було використано FixedThreadPool, який є статичним пулом потоків із фіксованою кількістю потоків. Кількість потоків визначається параметром threadPoolSize (у цьому випадку встановлено значення 4). Однак для досягнення більшої гнучкості в умовах високого навантаження, реалізація передбачає можливість динамічної зміни кількості потоків за допомогою механізму, доступного через ThreadPoolExecutor. У разі високого навантаження, потік, який не може бути виконаний через обмеження в розмірі пулу, додається в чергу завдань, що дозволяє гнучко масштабувати розмір пулу відповідно до поточної потреби в обчислювальних ресурсах.

ExecutorService дозволяє ефективно керувати потоками, виконуючи завдання паралельно, при цьому зменшується навантаження на основний потік, що дозволяє

більш ефективно використовувати ресурси. У разі високих вимог до обчислювальної потужності, ThreadPoolExecutor дозволяє динамічно масштабувати пул потоків для підвищення продуктивності. Основна перевага використання пулу потоків полягає в зменшенні часу на створення нових потоків і ефективному використанні ресурсів при виконанні великої кількості однотипних завдань.

Лістинг коду, що наведено на рис. 9, дозволяє оцінити ефективність використання паралельного підходу до запуску моделювань MATLAB моделей. В даному лістингу показано два методи для запуску MATLAB моделей, які реалізують різні підходи до виконання обчислювальних задач: паралельне та послідовне моделювання та обробку одних і тих ж моделей.

Перший метод, runMATLABModel, призначений для паралельного виконання окремої MATLAB моделі. Він отримує шлях до моделі як вхідний параметр і реалізує механізм багатопотокового виконання за допомогою сервісу ExecutorService, який описано раніше. Завдання виконання моделі передається в пул потоків через метод submit, що забезпечує асинхронне виконання. Для кожної задачі ініціалізується середовище MATLAB через об'єкт MATLABEngine, після чого виконується команда engine.eval, яка запускає модель на моделювання.

Другий метод, runModelsSequentially, реалізує послідовне виконання списку MATLAB моделей. Вхідними даними для цього методу є список моделей, кожна з яких представлена шляхом до відповідного файлу. Метод виконує послідовну ітерацію за списком моделей, для кожної з яких створюється новий екземпляр MATLABEngine, що ініціалізує середовище MATLAB. Виконання моделі відбувається за допомогою тієї ж команди engine.eval. Послідовний підхід є простішим у реалізації, але менш ефективним у порівнянні з паралельним через необхідність очікування завершення моделювання кожної окремої моделі перед початком виконання наступної.

```java
package com.lentez.diploma.matlabengine.service;

import com.mathworks.engine.MatlabEngine;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;

@Service
public class MatlabEngineTestService {

    private final ExecutorService executorService =
Executors.newFixedThreadPool(4);
    private final String PATH = "/Users/yaroslavholovan/IdeaProjects/bot/matlab-
engine-wrapper/matlab-engine/";
    public Future<String> runMatlabModel(String modelPath) {
        return executorService.submit(() -> {

            try {
                MatlabEngine engine = MatlabEngine.startMatlab();
                System.out.println("Starting to execute model: " + modelPath);
                engine.eval("run('" + PATH + modelPath + "')");
                System.out.println("Matlab model executed successfully");
                System.out.println();
                engine.close();
                return "Model " + modelPath + " completed";
            } catch (Exception e) {
                e.printStackTrace();
                return "Error running model " + modelPath;
            }
        });
    }

    public String runModelsSequentially(List<String> models) {
        for (String model : models) {
            try {
                MatlabEngine engine = MatlabEngine.startMatlab();
                System.out.println("Starting to execute model: " + model);
                engine.eval("run('" + PATH + model + "')");
                System.out.println("Matlab model executed successfully");
                System.out.println();
                engine.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return "All models executed sequentially";
    }


}
```

Рисунок 9 – Лістинг коду послідовного та паралельного способу запуску моделювань

Загалом, метод runMATLABModel забезпечує вищу продуктивність завдяки паралельному виконанню, проте вимагає більшого обсягу апаратних ресурсів, таких як оперативна пам’ять та доступні потоки. У порівнянні з послідовним виконанням завдань, використання паралельного виконання через ExecutorService надає значне

зменшення часу виконання моделювань для кількох моделей. Наприклад, для трьох простих m-файлів, час моделювання за допомогою паралельної реалізації, в середньому, на 3 секунди менше, порівняно з послідовною реалізацією. Цей результат досягається завдяки одночасному виконанню кількох моделювань різними MATLAB Engine інстансами, без очікування завершення кожного попереднього моделювання. Як наслідок, паралельне виконання дозволяє зменшити загальний час обробки та покращити ефективність роботи системи.

Однією з ключових переваг даної реалізації є можливість динамічного масштабування пулу потоків. Якщо навантаження збільшується і кількість паралельних завдань зростає, розмір пулу потоків може бути автоматично збільшений, щоб підтримувати високу продуктивність системи. Це дозволяє адаптувати систему до змінних умов навантаження без необхідності вручну налаштовувати кількість потоків або перевантажувати систему.

Загалом, використання багатопоточності через ExecutorService з динамічним масштабуванням кількості потоків дозволяє значно покращити продуктивність при виконанні кількох MATLAB моделей одночасно. Цей підхід забезпечує високу ефективність у використанні обчислювальних ресурсів і дозволяє значно скоротити час моделювання порівняно з послідовним виконанням завдань. Однак при розробці таких систем необхідно враховувати потенційні обмеження системи, пов'язані з управлінням потоками та їх коректним завершенням.

2.4 Проектування компонентів системи

2.4.1 MATLAB-bot мікросервіс

MATLAB-bot мікросервіс є ключовим компонентом системи, який реалізує взаємодію між кінцевим користувачем та іншими мікросервісами через API Gateway.

Основною задачею цього мікросервісу є обробка вхідних файлів, які надходять від користувачів через Telegram, їх передача на обробку в інші частини системи, а також доставка результатів моделювання назад користувачам.

Мікросервіс отримує файли через Telegram API, де забезпечується попередня обробка даних. Після отримання файлу, він перевіряється на відповідність вимогам, як-от формат «.m», що є стандартним форматом скриптів MATLAB. Кожен файл супроводжується додатковими метаданими, такими як унікальний ідентифікатор моделі (model_id) та ідентифікатор чату користувача (chat_id). Ці метадані дозволяють однозначно ідентифікувати моделі та встановлювати зв'язок між результатами моделювання та кінцевими користувачами.

Оброблений файл разом із супутніми метаданими передається на API Gateway через REST-запити. API Gateway виступає центральним маршрутизатором системи, перенаправляючи запити до відповідних мікросервісів, зокрема до Task-Processor для модифікації скриптів. Використання API Gateway дозволяє легко масштабувати систему та додавати нові точки входу чи обробки без необхідності змін у логіці MATLAB-Bot.

Після завершення моделювання результати передаються назад на MATLAB-Bot через API Gateway. Результати можуть включати текстові дані, які є виводом моделі, а також графічні зображення, створені під час виконання моделювання. MATLAB-Bot отримує ці дані, а потім передає користувачеві через Telegram API. Зображення, такі як графіки чи діаграми, надсилаються як мультимедійні повідомлення з підписами, що включають назву моделі та опис файлу.

Для взаємодії з іншими мікросервісами використовується RestTemplate, який дозволяє зручно здійснювати REST-запити та обробляти відповіді. UML діаграму класів компоненту системи MATLAB-bot наведено на рис. 10.

У проєктуванні MATLAB-Bot також враховано підтримку багатопоточності для обробки великої кількості запитів одночасно. Завдяки використанню пулу потоків забезпечується висока продуктивність навіть за умов значного навантаження, що критично важливо для систем, які повинні обробляти запити в реальному часі. Такий

підхід дозволяє ефективно обслуговувати користувачів і забезпечувати швидку доставку результатів, зберігаючи при цьому стабільність системи.

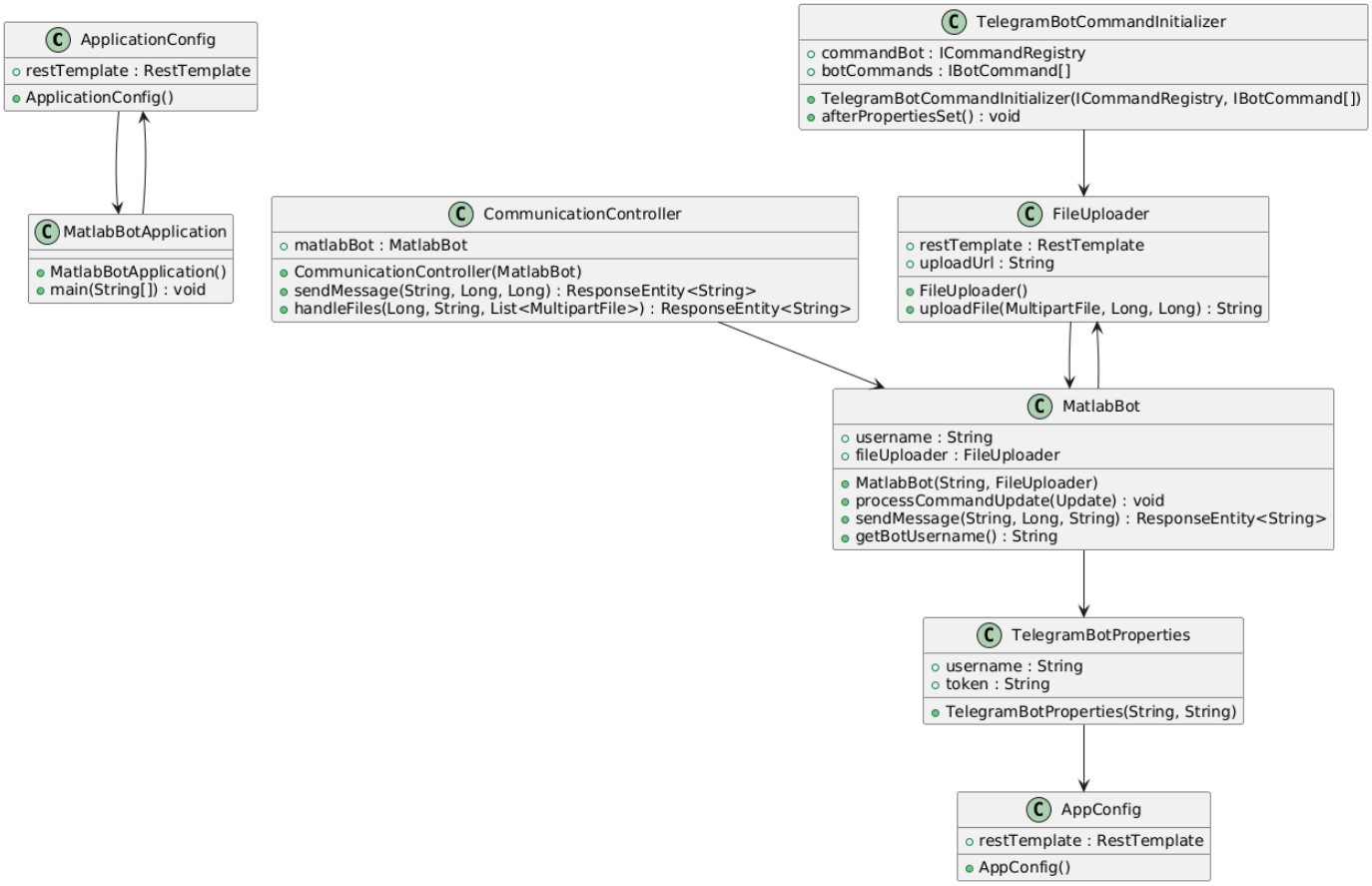

Рисунок 10 – UML діаграма класів MATLAB-bot мікросервісу

2.4.2 Task-proccessor мікросервіс

Task Processor (рис. 11) є одним із ключових компонентів системи, відповідальним за попередню обробку моделей MATLAB, які надходять до системи перед їх виконанням. Основними завданнями цього мікросервісу є прийняття файлів моделей, перевірка їх на відповідність встановленому формату та структурі, модифікація для збереження проміжних результатів моделювання, таких як графіки, а також збереження метаданих у базі даних для забезпечення подальшої взаємодії з іншими компонентами системи. Завдяки такому підходу забезпечується не лише

підготовка моделей для ефективного виконання, але й надійне збереження важливої інформації, пов’язаної з кожною окремою моделлю.

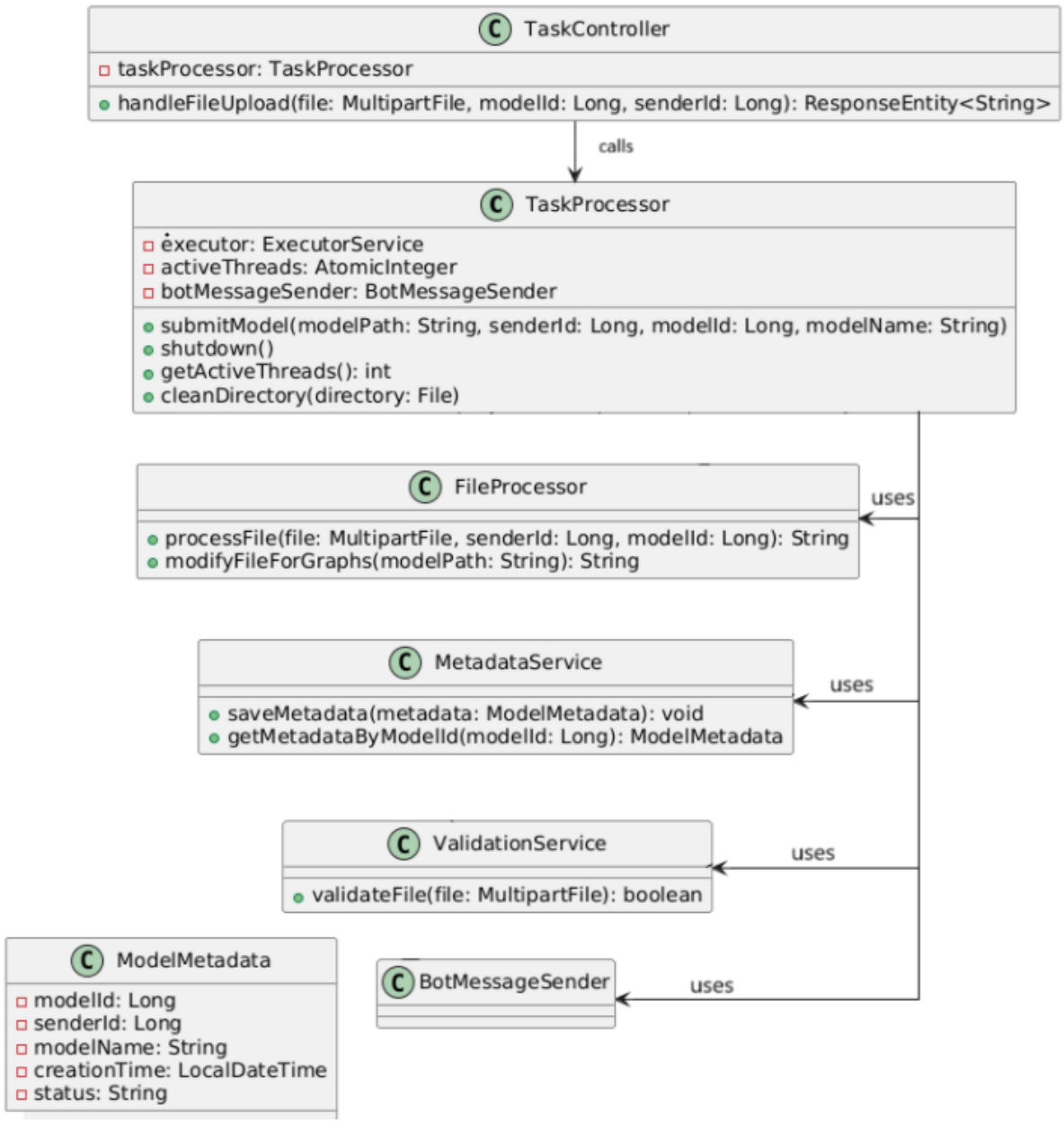

Рисунок 11 – UML діаграма класів task-proccessor мікросервісу

Процес обробки моделей починається з прийняття файлів через REST API. Контролер, реалізований у рамках цього мікросервісу, отримує запити HTTP POST і забезпечує передачу файлів на перевірку. На цьому етапі перевіряється відповідність файлу формату .m (скрипти MATLAB), а також аналізується його структура для виявлення можливих синтаксичних помилок або недопустимих конструкцій, які

могли б ускладнити виконання моделі MATLAB Engine. Якщо файл відповідає вимогам, він проходить наступний етап обробки — модифікацію. До моделі додається спеціальний сабскрипт MATLAB, що дозволяє зберігати ілюстративні ресурси, такі як графіки, створені під час моделювання. Наприклад, автоматично додається команда saveas, яка забезпечує збереження графіків у визначеній директорії з унікальними назвами, прив'язаними до ідентифікатора моделі. Таким чином, користувачі отримують доступ до результатів моделювання, представлених у зручному форматі.

Додатковим аспектом функціональності Task Processor є інтеграція з MongoDB, яка використовується для збереження метаданих моделей. У базі даних зберігаються такі параметри, як унікальний ідентифікатор моделі, ідентифікатор відправника, назва моделі, час створення запису та поточний статус виконання моделі. Збереження цих даних дозволяє ефективно управляти моделями та підтримувати надійну взаємодію між компонентами системи. Метадані серіалізуються у формат JSON і зберігаються у колекції MongoDB, що забезпечує їх доступність для запитів через REST API.

Архітектура Task Processor побудована на основі Spring Boot, що забезпечує модульність і масштабованість системи. Основні компоненти мікросервісу включають контролер для обробки запитів, сервіс перевірки файлів, модуль для збереження та обробки метаданих, а також сервіс для внесення змін у файли моделей. Взаємодія цих компонентів забезпечує узгоджений процес прийняття, перевірки, модифікації та передачі моделей на MATLAB Engine для виконання.

Функціонування Task Processor організовано у вигляді чіткого потоку, що включає прийом моделі, її перевірку, додавання необхідних модифікацій і передачу на наступні етапи обробки. Застосування MongoDB для збереження даних гарантує їх надійність, а використання Spring Boot сприяє спрощенню конфігурації та гнучкості розробки. Усе це створює ефективну основу для інтеграції Task Processor з іншими компонентами системи, такими як Telegram Bot і Result Handler, що дозволяє забезпечити високу продуктивність та надійність всієї системи.

2.4.3 MATLAB-engine мікросервіс

Мікросервіс MATLAB Engine (рис. 12) реалізує функціонал паралельного виконання моделей MATLAB з використанням потокобезпечної архітектури. Основною метою цього компонента є забезпечення ефективного виконання скриптів MATLAB у багатопоточному середовищі та передачі результатів обчислень через REST-запити до інших мікросервісів системи. Цей підхід дозволяє розподіляти навантаження між потоками, забезпечуючи швидку обробку численних запитів.

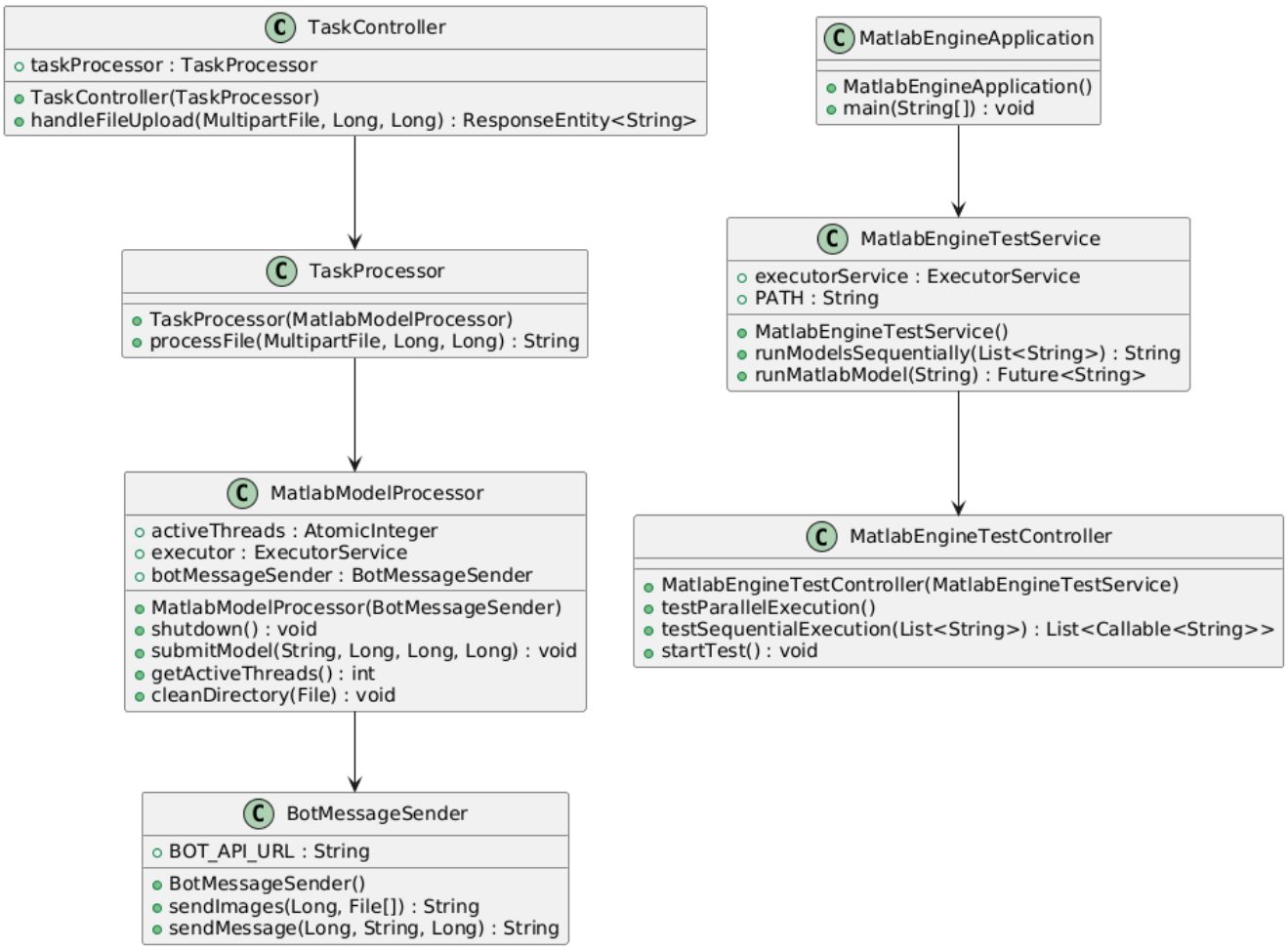

Рисунок 12 – UML діаграма класів MATLAB-engine мікросервісу

Мікросервіс побудований на основі Spring Boot, що забезпечує легкість інтеграції REST API, спрощене конфігурування, а також можливість масштабування.

Основною частиною паралельного виконання є використання пулу потоків ExecutorService, що дозволяє виконувати кілька моделей одночасно, обмежуючи кількість активних потоків для уникнення перевантаження системи. Конструктор класу MATLABModelProcessor ініціалізує пул потоків із розміром, який можна налаштовувати, що дає змогу адаптувати систему до різних рівнів навантаження.

Ключова функція мікросервісу – це метод submitModel, який приймає шлях до файлу моделі, ідентифікатор користувача, ідентифікатор моделі та ім'я моделі. Запуск кожної моделі виконується в окремому потоці, де для кожного виклику створюється інстанція MATLAB Engine через API. MATLAB Engine запускає модель, використовуючи методи evalc для отримання результатів виконання у вигляді рядка, що дозволяє зберігати всі виведені дані MATLAB, включно з потенційними помилками, у форматі тексту. Це спрощує аналіз результатів і подальшу передачу інформації на інші компоненти системи.

Для обробки результатів мікросервіс генерує графіки, які зберігаються як зображення у зазначеній директорії. Дані графіків передаються через REST-запити іншому мікросервісу, що відповідає за обробку результатів (наприклад, Telegram Bot). Для зручності реалізована функція очищення директорії від тимчасових файлів після завершення обробки.

Потокобезпека забезпечується за допомогою атомарних лічильників (AtomicInteger), що дозволяють відслідковувати кількість активних потоків і запобігати їх переповненню. Також для уникнення конфліктів доступу до спільних ресурсів використовується чітке розділення виконання кожної моделі в окремому потоці.

Паралельне виконання моделей значно підвищує продуктивність системи порівняно з послідовною обробкою. Наприклад, запуск трьох простих моделей з часом виконання 10 секунд кожна, в середньому, завершується на 3 секунди швидше у порівнянні з послідовною обробкою, що демонструє ефективність розподілу навантаження між потоками.

Архітектура забезпечує чіткий розподіл відповідальності. Основна логіка роботи мікросервісу розділена між контролером, який приймає запити, класом

обробки завдань TaskProcessor, що відповідає за збереження файлів, і класом MATLABModelProcessor, який запускає моделі. Для відправки даних на інші сервіси використовується BotMessageSender, що реалізує REST-виклики, спрощуючи інтеграцію з іншими мікросервісами.

Загалом, MATLAB Engine мікросервіс є високоефективним компонентом, який забезпечує надійне, потокобезпечне та масштабоване виконання завдань у паралельному режимі.

2.4.4 Result-handler мікросервіс

Архітектура Result Handler мікросервісу, що наведена на рис. 13, спрямована на отримання результатів моделювання від MATLAB Engine, їх обробку та зберігання, а також передачу на Telegram Bot через API Gateway.

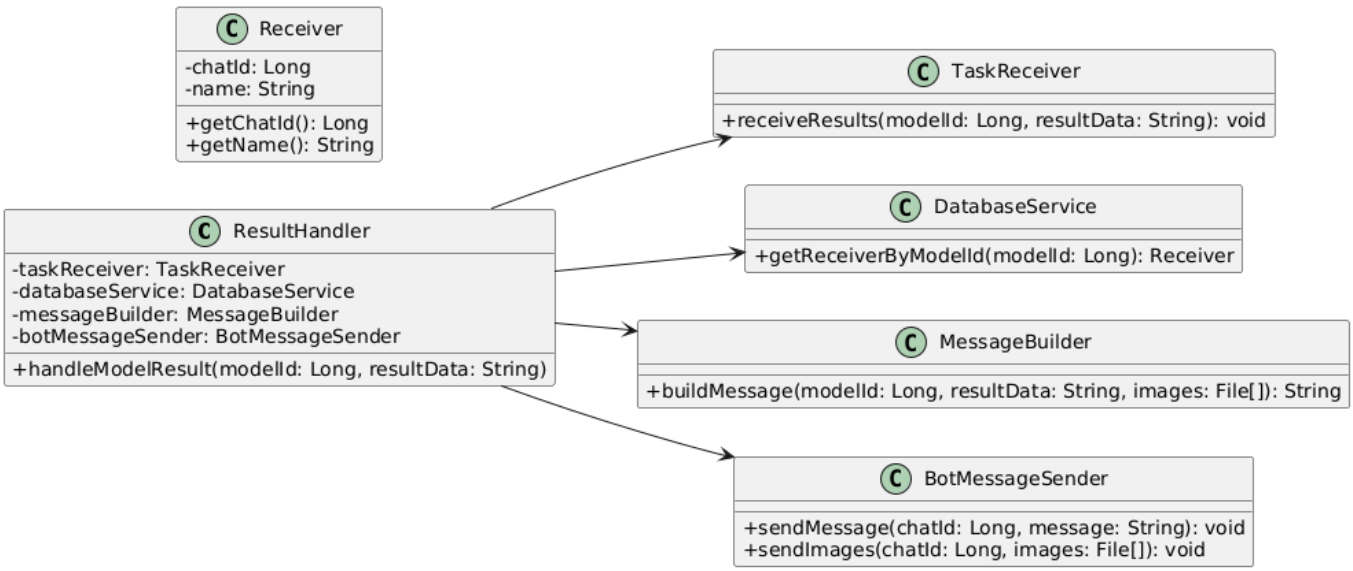

Рисунок 13 – UML діаграма класів result-handler мікросервісу

Основною метою цього мікросервісу є прийом даних про виконання моделі, пошук отримувача результатів у базі даних за допомогою ідентифікатора моделі,

підготовка відповіді для користувача, яка включає текстове повідомлення і збережені графіки, а також відправка цієї інформації на Telegram Bot мікросервіс.

Процес роботи Result Handler включає кілька етапів, кожен з яких виконується відповідним компонентом. Спершу, Result Handler отримує результат виконання моделі від MATLAB Engine через REST API. Після цього система взаємодіє з базою даних MongoDB через компонент DatabaseService, що дозволяє визначити отримувача результатів за допомогою переданого model_id. Цей процес забезпечує точне відправлення результатів користувачеві, який надіслав модель на виконання.

Подальша обробка результатів здійснюється компонентом MessageBuilder, який відповідає за формування відповідного повідомлення. Це повідомлення включає не лише текстовий результат виконання моделі, але й збережені графіки або інші ресурси, що були створені під час моделювання. Після формування повідомлення, результат передається компоненту BotMessageSender, який відповідає за відправку повідомлення через API Gateway на Telegram Bot мікросервіс. Весь процес передачі результатів користувачу завершується через HTTP запит, що гарантує швидку та надійну взаємодію між компонентами.

Основними функціями Result Handler є отримання результатів моделювання від MATLAB Engine, пошук отримувача результатів у базі даних за допомогою model_id, формування повідомлення з результатами, включаючи графіки, та відправка цього повідомлення через Telegram Bot мікросервіс за допомогою API Gateway. Така архітектура дозволяє ефективно організувати процес обробки результатів та їх передачу користувачам, забезпечуючи високий рівень автоматизації та масштабованості системи.

Як результат – реалізовано систему, яка може бути запущена на віддаленому сервері. Інтерфейсом користувача виступає Telegram-бот, через який і відбувається передеча файлів на виконання моделювань, та отримання результатів. Особливість такого підходу закулючається у автоматичному запуску моделювань, як тільки бот отримає файл, виконанні параллельної обробки моделей завдяки використанню багатопоточності.

2.5 Тестування працездатності розробленої системи

Для тестування розробленої системи необхідно перейти за посиланням: https://t.me/bot_MATLABot та надіслати m-файл в чат з ботом. Надсилання моделі для моделювання в бот зображено на рис. 14.

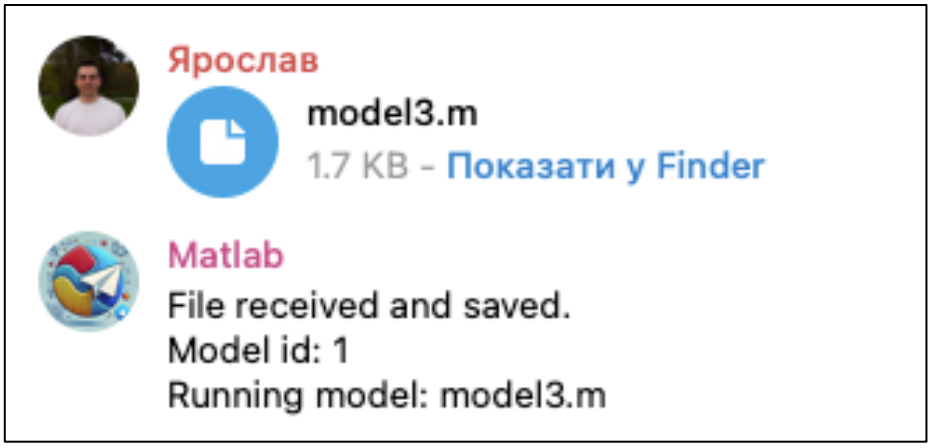

Рисунок 14 – Відправка m-файлу в телеграм бот

Після отримання файлу бот інформує про успішне отримання та збереження файлу, також він надає моделі унікальний ідентифікаційний номер, аби у випадку одночасної обробки великої кількості моделей користувач міг ідентифікувати, до якої саме моделі відносяться отримані результати. В цей час в системі розпочинається запуск моделювання отриманої моделі. Як тільки результат буде отримано в MATLAB Engine, система відправить результат користувачу у вигляді, зображеному на рис. 15.

Лістинг надісланого у m-файлі скрипту наведено в додатку А. Файл містить модель маятника, який піддається впливу сили тертя та зовнішньої періодичної сили. Скрипт виконує складну задачу, включаючи чисельне розв'язання диференціальних рівнянь, аналіз даних та побудову графіків.

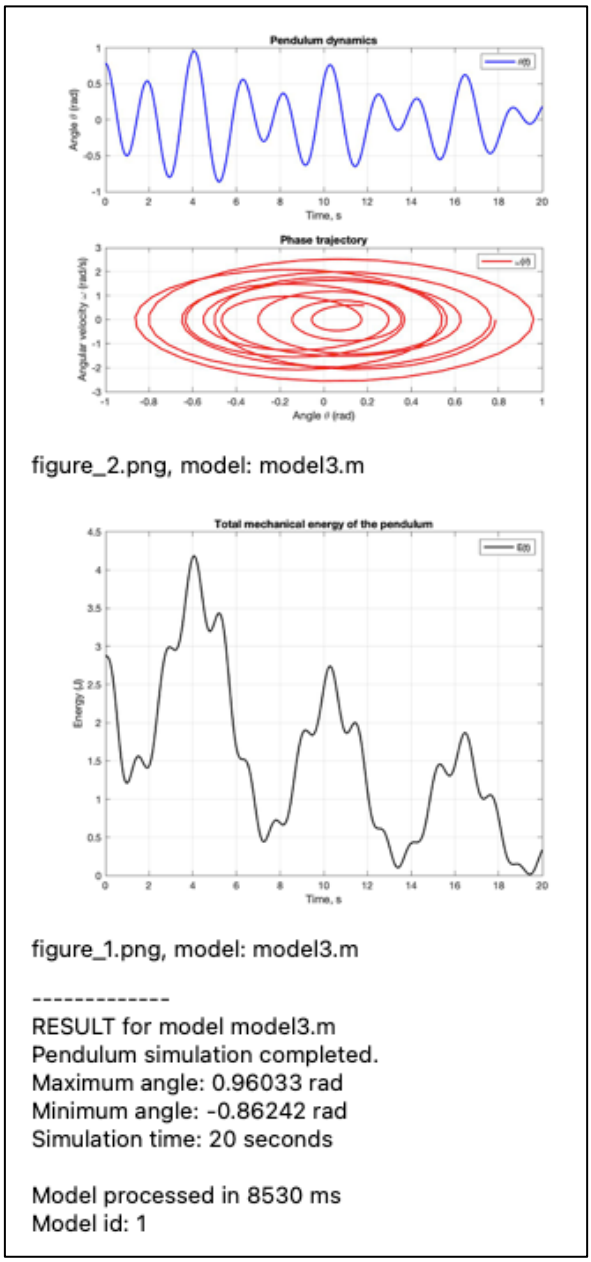

Рисунок 15 – Отримання результату моделювання m-файлу.

Висновок до розділу 2

У розділі 2 здійснено програмну реалізацію комбінованого підходу для запуску моделювань MATLAB моделей. Основною метою даного розділу стало проєктування та реалізація мікросервісної архітектури, яка дозволяє інтегрувати MATLAB API із середовищем Java для виконання багатопотокових обчислень.

У рамках розробки було реалізовано основні мікросервіси системи. Зокрема, розроблено MATLAB-bot мікросервіс для взаємодії з користувачами через Telegram, Task-processor мікросервіс для розподілу завдань, MATLAB-engine мікросервіс для запуску обчислювальних процесів, а також Result-handler мікросервіс для обробки й збереження результатів моделювань. Крім того, реалізовано функціонал для багатопотокового виконання моделей із використанням механізмів ExecutorService у Java.

Було визначено ключові програмні засоби, серед яких фреймворк Spring для організації мікросервісної архітектури та пришвидшення процесу розробки, Telegram Bot API для інтеграції з користувачами, а також MATLAB Engine API для виконання моделей у MATLAB. Система також включає базу даних для зберігання метаданих та результатів моделювань, що забезпечує зручний доступ до історії виконаних завдань.

У процесі тестування працездатності системи підтверджено її функціональність, зокрема здатність виконувати послідовні та паралельні моделювання для різних сценаріїв.

Таким чином, у цьому розділі було повністю реалізовано запропонований комбінований підхід, який дозволяє значно оптимізувати процеси моделювання MATLAB моделей, забезпечуючи автоматизацію, паралельність обчислень та інтеграцію з сучасними комунікаційними платформами. Ця реалізація стала основою для подальшого експериментального дослідження ефективності розробленого підходу.

РОЗДІЛ 3

ДОСЛІДЖЕННЯ РОЗРОБЛЕНОГО КОМБІНОВАНОГО ПІДХОДУ ДЛЯ ЗАПУСКУ МОДЕЛЮВАНЬ MATLAB МОДЕЛЕЙ

3.1 Складність алгоритмів

Складність алгоритмів є ключовим поняттям у теорії обчислень, яке визначає ефективність виконання алгоритмів через оцінку обчислювальних ресурсів, необхідних для їх реалізації [27]. Вона характеризує, як витрати часу або пам'яті змінюються залежно від розміру вхідних даних. Це поняття є фундаментальним для аналізу та оптимізації обчислювальних процесів, особливо у випадках роботи з великими обсягами даних або у високопродуктивних обчисленнях. Основними категоріями складності є часова та просторова [28]. Часова складність оцінює кількість операцій, які алгоритм виконує для обробки вхідних даних певного розміру. Для її опису найчастіше використовується нотація "О-велике" (Big-O), що дозволяє визначити асимптотичну поведінку алгоритму при зростанні обсягу даних. Наприклад, лінійна складність O(n) означає, що кількість операцій зростає прямо пропорційно до розміру вхідних даних, тоді як квадратична складність O(n^2) вказує на те, що витрати часу збільшуються пропорційно квадрату кількості вхідних елементів. Інші класи часової складності включають логарифмічну, полігональну, експоненціальну та факторіальну, кожна з яких відображає різний ступінь залежності витрат часу від розміру вхідних даних. На рис. 16. показана залежність різних видів складності алгоритмів від збільшення розміру вхідних даних n.

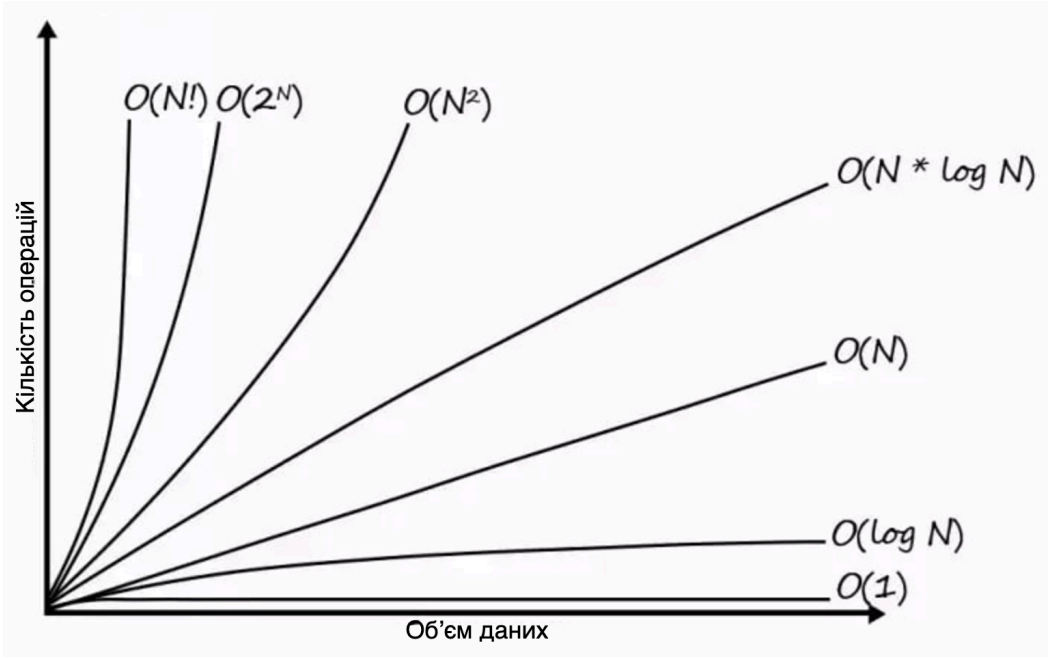

Рисунок 16 – Залежність різних видів складності алгоритмів від збільшення розміру вхідних даних n

Просторова складність, у свою чергу, визначає обсяг пам’яті, необхідної для виконання алгоритму, враховуючи як постійну пам’ять для збереження змінних, так і додаткову пам’ять, яка виділяється динамічно під час виконання [28]. Наприклад, алгоритми сортування можуть вимагати різний обсяг додаткової пам’яті залежно від обраного методу, що також впливає на їхню практичну ефективність. Розуміння складності алгоритмів дозволяє не лише вибирати оптимальні рішення для конкретних завдань, але й оцінювати можливості масштабування обчислень та передбачати поведінку системи за змінних умов. Це є особливо важливим у сучасному контексті, коли обсяг даних і складність задач стрімко зростають.

3.2 Опис процесу дослідження комбінованого підходу

Процес дослідження комбінованого підходу для запуску MATLAB моделей був спрямований на оцінку ефективності паралельної та послідовної обробки моделей

із різними рівнями складності алгоритмів. Для цього було проведено серію експериментів, які охоплювали моделювання алгоритмів шести типів складності: $O(n), O(log\ n), O(n\ log\ n), O(n^2), O(n!), O(n^n)$.

Дослідження складалося з кількох етапів. Спочатку кожен алгоритм моделювався з різною кількістю файлів моделей: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Для кожної кількості моделей запуск моделювання проводився двома підходами:

1. Послідовний запуск – моделі виконувалися одна за одною в однопоточному середовищі.
2. Паралельний запуск – моделі запускалися одночасно, використовуючи багатопотокове середовище.

Для кожного підходу вимірювався загальний час виконання моделювань усіх моделей. Це дозволило оцінити, наскільки ефективним є паралельний запуск в комбінованому підході у порівнянні з послідовним для алгоритмів різних складностей.

Загалом було проведено 6 повних експериментів – по одному для кожного рівня складності алгоритмів. Результати дослідження, а саме час виконання для кожної кількості моделей у послідовному та паралельному режимах, були збережені у форматі CSV для подальшого аналізу.

Для кожного рівня складності були побудовані графіки, які демонструють залежність часу виконання від кількості моделей та їх складності у послідовному та паралельному режимах. Ці результати стали основою для оцінки ефективності розробленого комбінованого підходу.

Дослідження проводилось на Apple MacBook Pro з такими характеристиками:

- Процесор: 10-ти ядерний Apple M2 Pro, базова частота – 2.4 ГГц, максимальна – 3.7 ГГц;
- RAM: LPDDR5, 16 GB;
- Пропускна здатність RAM: 200 ГБ/с.
- Графічний процесор: вбудований 16-ти ядерний Apple M2 Pro;
- Носій: APPLE SSD AP0512Z

3.3 Модель для обчислення суми елементів вектора зі складністю $O(n)$

У додатку Б.1.1 наведено модель для обчислення суми елементів вектора, яка демонструє лінійну складність алгоритму $O(n)$. У цьому випадку алгоритм проходить кожен елемент вектора лише один раз. Він реалізує цикл, у якому всі елементи вектора додаються до змінної sum_result. Загальна формула обчислення суми елементів вектора може бути записана як:

$$S = \sum_{i=1}^{n} a_i ,$$

де S — сума елементів,

a_i — елементи вектора,

n — розмір вектора (в даному випадку n = 1 000 000).

У процесі дослідження було проведено обчислення часу виконання для послідовного та паралельного запусків моделювань з кількістю файлів від 1 до 10. Результати експериментів наведено в табл. 3.1.

Таблиця 3.1 – Результат дослідження моделі зі складністю O(n)

Кількість файлів	Час послідовної обробки (мс)	Час паралельної обробки (мс)
1	3728	2329
2	4287	2489
3	6175	2862
4	8374	3594
5	10858	4227
6	13157	5305
7	15294	5858
8	17035	6558
9	19755	7963
10	21584	10300

За результатами, наведеними в табл. 3.2, можна встановити, що час виконання обчислень у послідовному режимі зростає майже лінійно зі збільшенням кількості

файлів. Це очікувано, адже кожен файл обробляється окремо, а загальний час є сумою часу обробки всіх файлів. У паралельному режимі спостерігається значне скорочення часу виконання, особливо для великої кількості файлів. Це пояснюється тим, що завдання розподіляються між потоками, що дозволяє ефективно використовувати обчислювальні ресурси.

На рис. 17 показано результат дослідження залежності часу виконання від кількості файлів для алгоритму зі складністю O(n) при n = 1 000 000 елементів. Як видно, паралельний підхід забезпечує скорочення часу обробки на 52.3% завдяки одночасному виконанню кількох процесів.

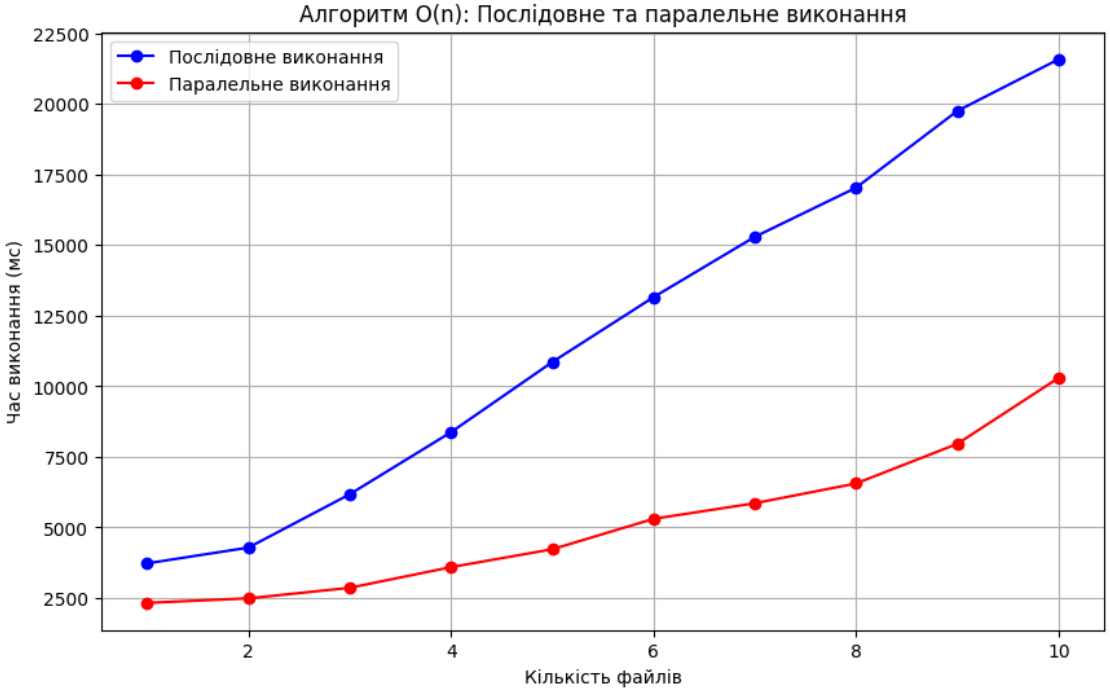

Рисунок 17 – Залежність часу виконання від кількості файлів для алгоритму зі складністю $O(n)$

Дослідження показало, що ефективність паралельного підходу особливо помітна для великої кількості файлів, коли обчислювальні ресурси використовуються максимально ефективно. Для невеликої кількості файлів виграш у часі є менш вираженим через накладні витрати на ініціалізацію паралельних потоків.

3.4 Модель для алгоритму бінарного пошуку зі складністю $O(log\ n)$

Для дослідження ефективності виконання алгоритмів зі складністю $O(log\ n)$ було реалізовано алгоритм бінарного пошуку наведений у додатку Б.1.2, що є одним із класичних прикладів такої складності [29]. Основна ідея цього алгоритму полягає у послідовному поділі відсортованого масиву навпіл, доки не буде знайдено шуканий елемент або не буде визначено, що його в масиві немає.

Моделювання проводилося для різної кількості файлів, які запускалися як послідовно, так і паралельно. Для кожного випадку було визначено час виконання. У дослідженні використовувався MATLAB-скрипт, наведений у додатку В.

Для моделювання було створено масив, розміром 1 000 000 елементів, заповнених випадковими числами, та відсортовано його. Цільовий елемент для пошуку також генерувався випадковим чином. У кожному тесті перевірявся час виконання пошуку як у послідовному, так і в паралельному режимах. Дані, отримані під час дослідження, наведено у табл. 3.2:

Таблиця 3.2 – Результат дослідження моделі зі складністю $O(log\ n)$

Кількість файлів	Час послідовної обробки (мс)	Час паралельної обробки (мс)
1	2291	2110
2	4156	2372
3	6489	3010
4	8549	3642
5	10680	4113
6	13727	4386
7	14942	5109
8	17740	6440
9	20044	6627
10	25176	8066

На рис. 18 показано результат дослідження залежності часу виконання від кількості файлів для алгоритму зі складністю $O(log\ n)$. Паралельна обробка дозволяє провести моделювання 10 файлів за час, на 57% менший ніж при послідовній обробці.

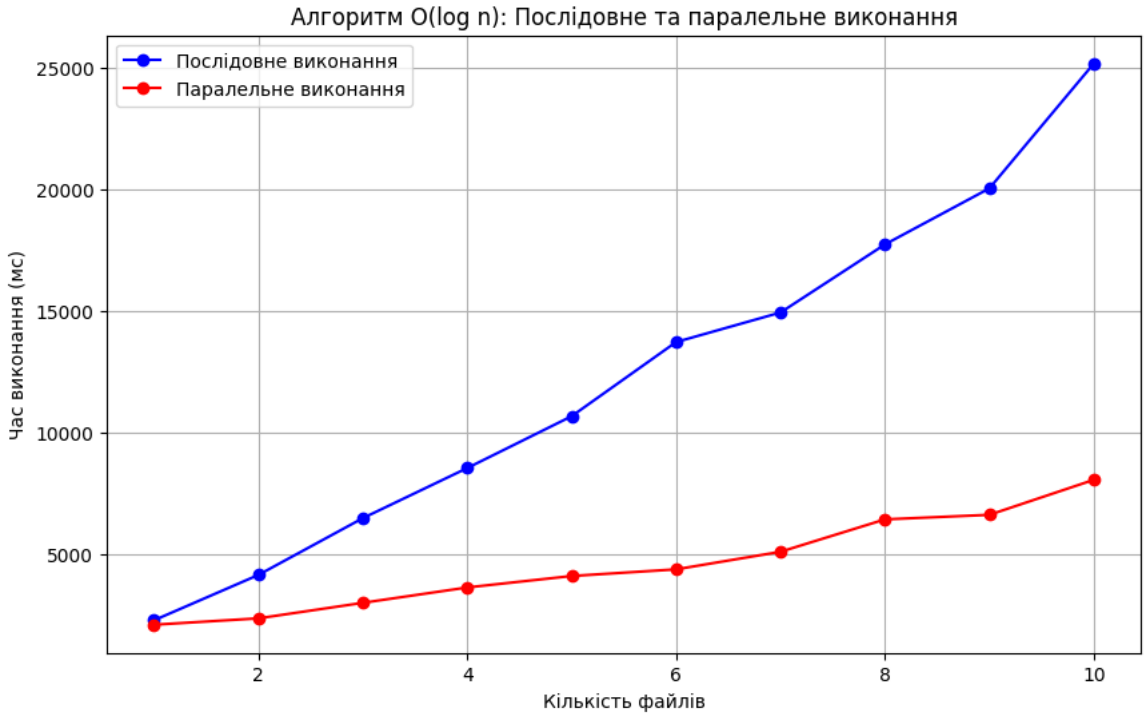

Рисунок 18 – Залежність часу виконання від кількості файлів для алгоритму зі складністю $O(log\ n)$

Як і у попередньому випадку, дослідження показало, що ефективність комбінованого підходу найбільше помітна для великої кількості файлів, коли обчислювальні ресурси використовуються максимально ефективно.

3.5 Модель для сортування масиву методом злиття зі складністю $O(n\ log\ n)$

Алгоритм сортування злиттям (Merge Sort) [30] є одним із найбільш ефективних методів сортування, який демонструє складність $O(n\ log\ n)$. Основна

ідея цього алгоритму полягає у рекурсивному поділі масиву на дві частини, сортуванні кожної з них окремо та об’єднанні результатів у впорядкований масив. Завдяки своїй логарифмічній залежності від кількості елементів, алгоритм Merge Sort широко використовується у випадках, коли необхідно обробити великі масиви даних з високою ефективністю.

Для аналізу ефективності роботи алгоритму було реалізовано MATLAB-скрипт, наведений у додатку Б.1.3, який реалізує сортування злиттям. У дослідженні використовувався масив довжиною 100 000 елементів , заповнений випадковими числами. Усі експерименти виконувалися для різної кількості одночасно запущених моделей, від 1 до 10, як у послідовному, так і в паралельному режимах. Час виконання вимірювався для кожного підходу, а результати зберігалися для подальшого аналізу.

Результати дослідження комбінованого підходу для сортування методом злиття представлено в табл. 3.3.

Таблиця 3.3 – Результат дослідження моделі зі складністю $O(n\ log\ n)$

Кількість файлів	Час послідовної обробки (мс)	Час паралельної обробки (мс)
1	3561	3252
2	4705	3671
3	7669	4021
4	9603	4280
5	12413	4553
6	15362	5173
7	17260	6428
8	20462	7652
9	23805	8390
10	26058	9017

На рис. 19 показано результат дослідження залежності часу виконання від кількості файлів для алгоритму зі складністю $O(n\ log\ n)$. Паралельна обробка

дозволяє провести моделювання 10 файлів за час, на 65.4% менший ніж при послідовній обробці.

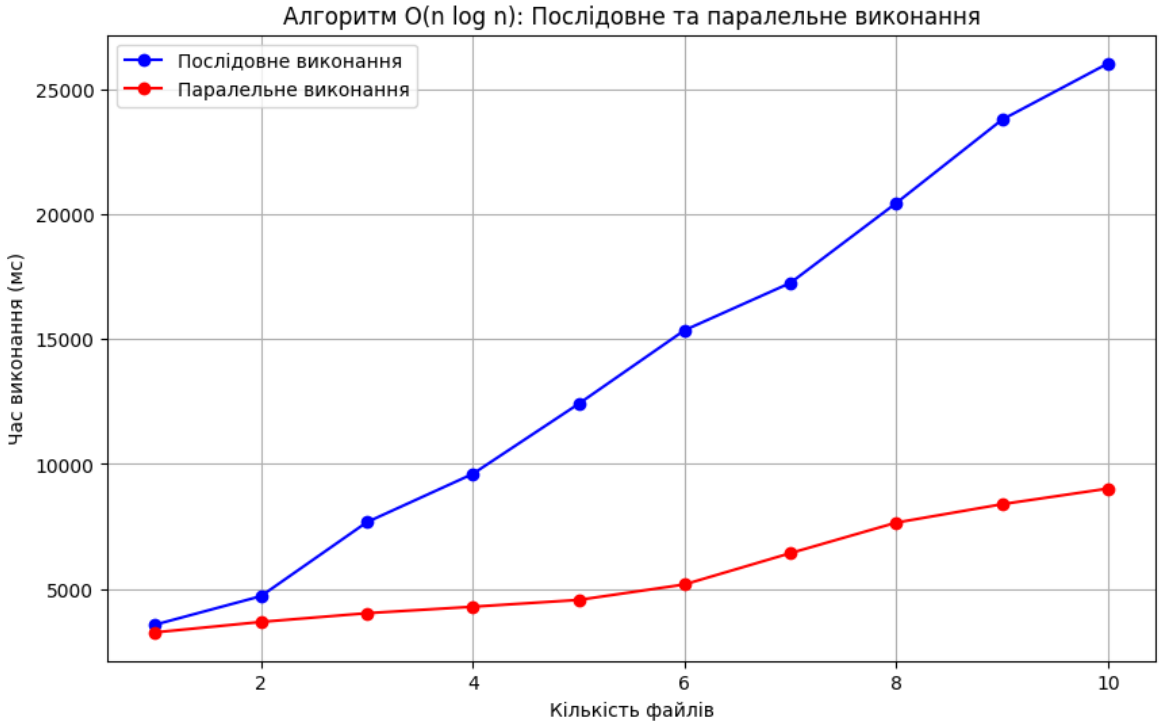

Рисунок 19 – Залежність часу виконання від кількості файлів для алгоритму зі складністю $O(n\ log\ n)$

Аналіз отриманих даних показує, що паралельний підхід демонструє значну ефективність у порівнянні з послідовним виконанням. Із збільшенням кількості одночасно запущених моделей переваги паралельної обробки стають дедалі більш виразними. Для обробки 10 файлів час послідовного виконання склав 26,058 мс, у той час як паралельне виконання зайняло 9,017 мс, що на 65.4% швидше.

3.6 Модель для сортування бульбашкою зі складністю $O(n^2)$

У межах дослідження алгоритмів різної складності було проведено експериментальне моделювання роботи алгоритму сортування бульбашкою, який

належить до класу алгоритмів із квадратичною складністю $O(n^2)$ [31]. Для цього було використано спеціально створений MATLAB-скрипт, наведений у додатку Б.1.4. Алгоритм реалізує базовий метод сортування, що передбачає послідовне порівняння сусідніх елементів масиву та їх перестановку, якщо перший елемент перевищує другий. Цей процес повторюється для всіх елементів масиву до повного упорядкування.

Дослідження включало послідовне й паралельне виконання алгоритму для масивів розміром 1 500 елементів. Генерація даних для масивів відбувалася випадковим чином, з числами в діапазоні від 1 до 1 000 000. Як і в попередніх дослідженнях, ключовою характеристикою є час моделювання різної кількості моделей у послідовному та паралельному режимі. Результати наведені в табл. 3.4.

Таблиця 3.4 – Результат дослідження моделі зі складністю $O(n^2)$

Кількість файлів	Час послідовної обробки (мс)	Час паралельної обробки (мс)
1	3612	2073
2	4160	2369
3	6436	3129
4	9320	3799
5	10550	4526
6	12752	4779
7	15200	6122
8	18582	8002
9	21974	9594
10	24652	11917

На основі отриманих даних побудовано графік залежності часу виконання алгоритму від кількості файлів для послідовного та паралельного підходів (рис. 20).

Час виконання збільшується квадратично зі збільшенням кількості файлів у послідовному режимі. При обробці 10 файлів паралельний підхід надає можливість досягти зменшення часу виконання на 51,6% у порівнянні з послідовним підходом.

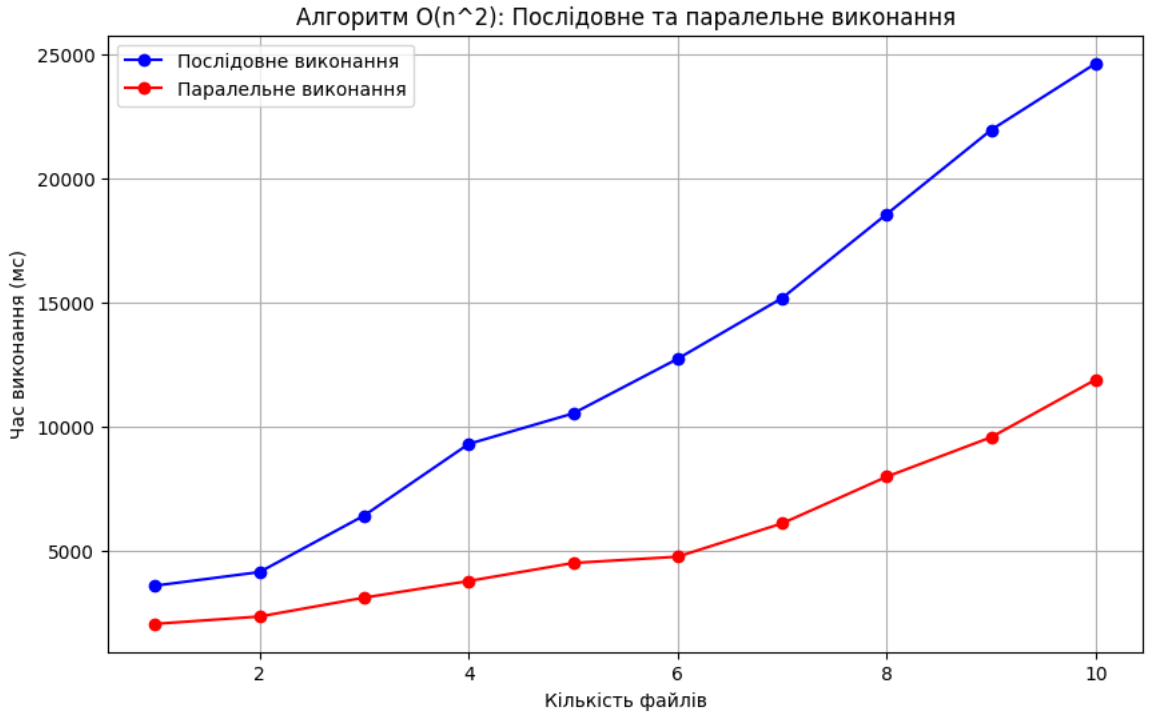

Рисунок 20 – Залежність часу виконання від кількості файлів для алгоритму зі складністю $O(n^2)$

3.7 Модель для генерації перестановок зі складністю $O(n!)$

У межах дослідження алгоритмів високої складності проведено експеримент моделювання роботи алгоритму генерації всіх можливих перестановок елементів масиву, що належить до класу алгоритмів із факторіальною складністю O(n!).

Скрипт, що наведений у додатку Б.1.5, реалізує алгоритм генерації всіх можливих перестановок елементів заданого масиву. Його основою є рекурсивний підхід, який відповідає за створення всіх можливих комбінацій шляхом ітеративного вибору елементів та їхнього поєднання з підперестановками.

Алгоритм складається з рекурсивної функції generatePermutations, яка приймає масив як вхідний параметр. Якщо довжина масиву дорівнює 1, функція просто повертає цей масив, оскільки для одного елемента є лише одна перестановка. Якщо

масив містить більше елементів, алгоритм проходить через кожен із них, вибираючи його як фіксований. Решта елементів формує новий масив, для якого рекурсивно викликається функція, що створює підперестановки. Ці підперестановки об'єднуються з фіксованим елементом, і результат додається до загального списку перестановок.

У основній частині скрипта генерується масив випадкових чисел розміром n=10, де кожен елемент є цілим числом у діапазоні від 1 до 1000. Цей масив передається у функцію generatePermutations, яка повертає всі можливі перестановки.

Для оцінки ефективності було проведено такі ж експерименти, як і для інших алгоритмів. Результати дослідження наведено в табл. 3.5.

Таблиця 3.5 – Результат дослідження моделі зі складністю $O(n!)$

Кількість файлів	Час послідовної обробки (мс)	Час паралельної обробки (мс)
1	3365	3025
2	4352	4362
3	6596	5254
4	8544	6054
5	10788	7333
6	13202	8586
7	15723	9474
8	19534	10304
9	21714	11025
10	27457	13362

На основі отриманих даних побудовано графік, який демонструє залежність часу виконання алгоритму від кількості файлів для послідовного та паралельного підходів (рис. 21). Паралельний підхід суттєво знижує час виконання моделювань. При обробці 10 файлів час виконання скоротився на 37,5% порівняно з послідовним підходом.

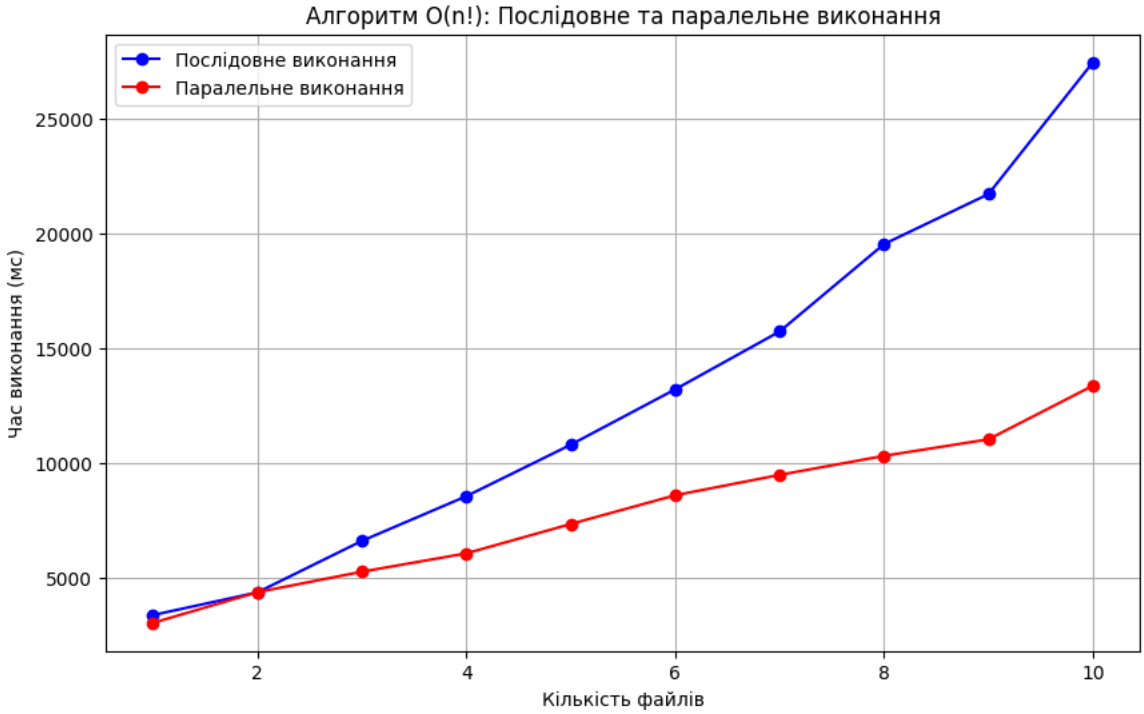

Рисунок 21 – Залежність часу виконання від кількості файлів для алгоритму зі складністю $O(n!)$

3.8 Модель для розбиття множин зі складністю $O(n^n)$

Скрипт, наведений в додатку Б.1.6 реалізує алгоритм генерації всіх можливих комбінаторних розбиттів множини, складність якого визначається як $O(n^n)$. Алгоритм створює всі можливі способи розбиття цілого числа n (яке в даному випадку рівне 7) на доданки. Кожен доданок є цілим числом, а сума всіх доданків у кожному розбитті дорівнює n. Завдяки цій властивості алгоритм охоплює всі можливі комбінаторні структури, що відповідають заданій множині.

Функція generatePartitions служить точкою входу, викликаючи рекурсивну функцію generatePartitionsRecursive. Ця рекурсивна функція обробляє кожен можливий доданок у діапазоні від 1 до n, додаючи його до поточного розбиття. Якщо залишок від n після додавання всіх доданків дорівнює нулю, розбиття вважається завершеним і додається до загального списку. Таким чином, алгоритм ітеративно

зменшує n і будує всі можливі варіанти розбиття, що забезпечує повне охоплення множини можливостей.

Зростання складності $O(n^n)$ зумовлене тим, що для кожного елемента n створюються n варіантів подальших розбиттів, що ускладнює виконання при великих значеннях n. Для зменшення обчислювального навантаження і уникнення дублювань використовується рекурсія з динамічним формуванням підмножин.

Результати дослідження продуктивності алгоритму для послідовної та паралельної обробки представлені в табл 3.6.

Таблиця 3.6 – Результат дослідження моделі зі складністю $O(n^n)$

Кількість файлів	Час послідовної обробки (мс)	Час паралельної обробки (мс)
1	4538	4203
2	5227	5492
3	7636	6618
4	9629	7363
5	12147	8123
6	15335	10119
7	18057	11787
8	20532	13306
9	23658	15288
10	28499	16311

На основі результатів побудовано графік залежності часу виконання від кількості файлів для алгоритму зі складністю $O(n^n)$, що показано на рис. 22.

Як і у попередніх дослідженнях, аналіз результатів показав, що паралельна обробка демонструє переваги над послідовною, особливо для великих значень n.

При обробці 10 файлів час паралельного виконання (16311 мс) виявився значно меншим порівняно з послідовним виконанням (28499 мс), що забезпечує на 46.8 % покращення ефективності.

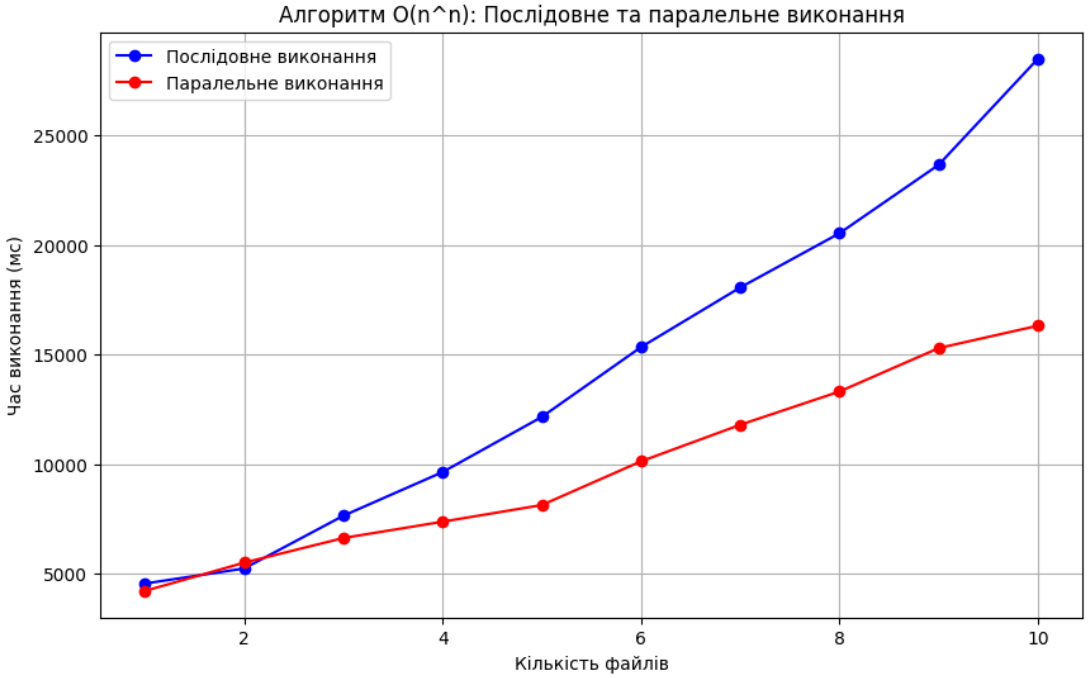

Рисунок 22 – Залежність часу виконання від кількості файлів для алгоритму зі складністю $O(n^n)$

3.9 Дослідження використання системних ресурсів під час використанням комбінованого підходу у порівнянні з традиційним

Ефективність обчислювальних підходів залежить не лише від часу виконання алгоритмів, але й від рівня використання апаратних ресурсів, таких як центральний процесор та оперативна пам’ять.

Дослідження проводилося на шести алгоритмах різної складності: $O(n), O(log\ n), O(n\ log\ n), O(n^2), O(n!), O(n^n)$. Для кожного алгоритму моделювалися обчислення із фіксованою кількістю вхідних файлів (10 файлів), а результати зафіксовані у вигляді найбільшого відсоткового використання CPU та пам’яті на момент моделювання. Послідовний запуск моделей характеризується поступовим зростанням навантаження на CPU залежно від кількості операцій та тривалості його виконання, тоді як паралельний підхід демонструє значно більший рівень завантаження CPU, що пояснюється одночасним виконанням кількох задач.

За результатами досліджень було отримано дані, що наведено в табл. 3.7, використання ресурсів під час послідовного та паралельного запуску. На основі цих даних побудовано графіки, наведені на рис. 23 та рис. 24.

Таблиця 3.7 – Використання апаратних ресурсів під час комбінованого запуску

n	Складність моделі	Послідовний запуск		Паралельний запуск	
		Максимальне навантаження на CPU (%)	Максимальне навантаження на ОЗУ (%)	Максимальне навантаження на CPU (%)	Максимальне навантаження на ОЗУ (%)
1000000	$O(n)$	11.43	33.12	37.81	39.18
1000000	$O(log\ n)$	11.18	30.44	36.10	38.04
100000	$O(n\ log\ n)$	14.01	34.90	47.25	40.73
1500	$O(n^2)$	15.14	37.52	50.63	44.92
10	$O(n!)$	15.86	37.03	52.21	43.66
7	$O(n^n)$	11.22	33.28	48.08	41.10

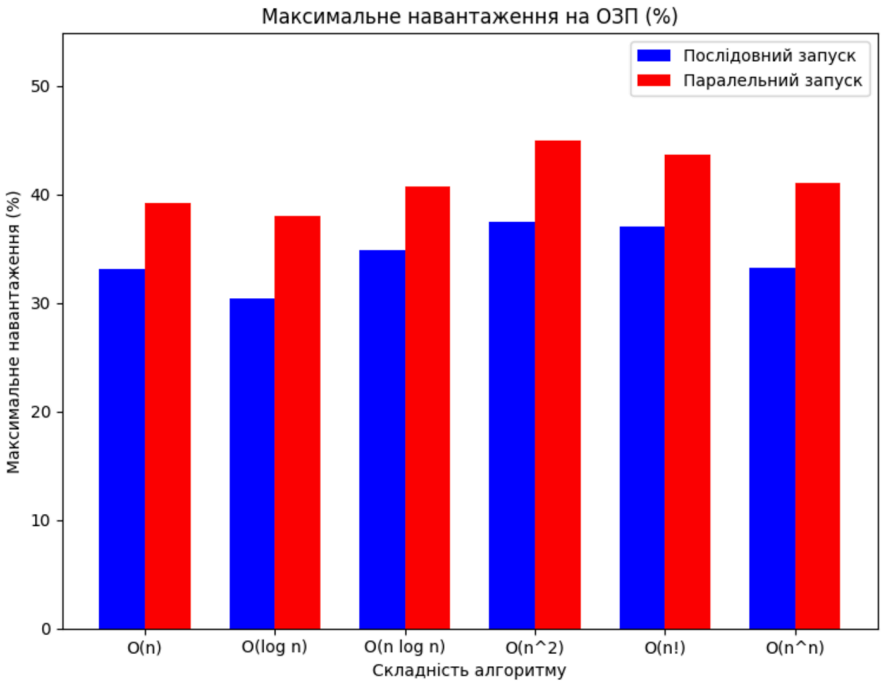

Рисунок 23 – Використання RAM для послідовної та паралельної обробки моделей

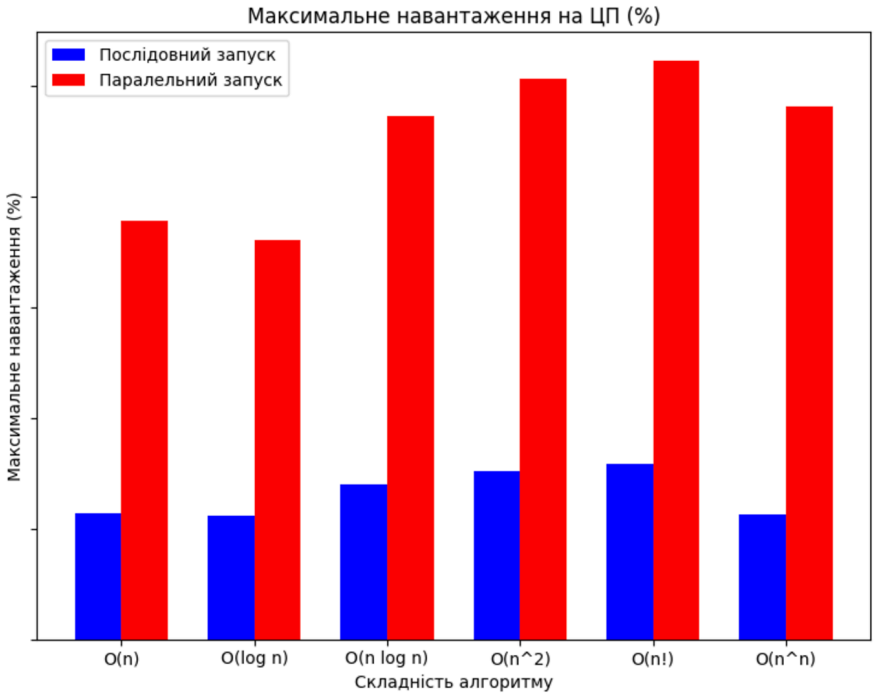

Рисунок 24 – Використання CPU для послідовної та паралельної обробки моделей

Дослідження показує, що паралельна обробка моделей значно знижує час виконання завдань, проте це досягається за рахунок підвищеного використання апаратних ресурсів, таких як CPU та оперативна пам’ять. Зростання навантаження на процесор є очікуваним результатом, оскільки обчислювальні ресурси розподіляються між кількома потоками.

Таким чином, комбінований підхід показав істотні переваги в скороченні часу виконання завдань, особливо для моделей високої складності. Однак ці переваги супроводжуються зростанням вимог до апаратних ресурсів, що є важливим фактором для врахування при масштабуванні та оптимізації обчислювальних систем.

Висновок до розділу 3

У ході проведеного дослідження було проаналізовано ефективність розробленого комбінованого підходу до запуску моделювань MATLAB моделей із

використанням багатопотокового середовища. Було змодельовано та протестовано шість алгоритмів різної складності: $O(n), O(log\ n), O(n\ log\ n), O(n^2), O(n!), O(n^n)$. Дослідження охоплювало оцінку часу виконання моделювань для послідовного та паралельного підходів, а також аналіз використання апаратних ресурсів.

Результати експериментів показали, що паралельна обробка моделей значно знижує час виконання завдань порівняно з послідовною. Найбільші переваги комбінованого підходу були зафіксовані для моделей із високою складністю $O(n^2), O(n!), O(n^n)$, де час виконання в паралельному режимі скоротився, в середньому на 57.97% для 10-ти моделей.

Водночас, комбінований підхід до обробки моделей супроводжувався підвищеним використанням апаратних ресурсів. Зокрема, навантаження на процесор у паралельному режимі значно зростало для всіх моделей. В середньому, навантаження на процесор в паралельному режимі на 32.21% більше, ніж в послідовному, в той час як навантаження на пам’ять відрізняється, в середньому, на 6.89%.

РОЗДІЛ 4

РОЗРОБКА СТАРТАП ПРОЕКТУ «MATLABOT»

4.1 Опис та технологічний аудит ідеї стартап-проекту

У попередніх розділах було досліджено потенційні виклики, пов'язані із запуском MATLAB моделей у розподілених системах. У цьому розділі пропонується аналіз стартап-проєкту, спрямованого на створення інноваційної системи запуску моделювань MATLAB моделей за допомогою Telegram-бота. Ця система поєднує переваги сучасних інформаційних технологій і потреби користувачів у зручних і автоматизованих рішеннях для моделювання, пропонуючи ефективний і простий інструмент для взаємодії з MATLAB через популярний месенджер.

Метою цього проєкту є інтеграція MATLAB Engine та Telegram-бота в єдину систему на мікросервісній архітектурі, що дозволить запускати моделювання з будь яких пристроїв. Такий підхід спрямований на підвищення зручності роботи з MATLAB моделями, розширення доступності інструментів моделювання, а також зменшення часу та зусиль, необхідних для налаштування середовища моделювання.

Ключові переваги запропонованої системи включають інтуїтивно зрозумілий інтерфейс взаємодії через Telegram, автоматизацію запуску моделювань, гнучкість у налаштуванні системи під потреби користувачів і можливість запуску моделювань з будь-якого пристрою, підключеного до Інтернету. Інтеграція Telegram-бота забезпечує легкий доступ до функціоналу MATLAB, навіть для користувачів, які не мають глибоких технічних знань.

Такий підхід до організації процесу моделювання сприяє підвищенню ефективності роботи студентів, викладачів, інженерів та дослідників, які працюють із MATLAB моделями. Запропонована система також має значний потенціал для

масштабування, адаптації до різних сценаріїв використання та інтеграції з іншими системами, що відкриває нові можливості для автоматизації й оптимізації наукових і прикладних досліджень.

У таблиці 4.1 наведено огляд концепції та потенційних ринкових сегментів для розглядуваної системи запуску MATLAB моделей через Telegram-бот.

Таблиця 4.1 – Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Система запуску MATLAB моделей через Telegram-бот	Освітні установи	Зручний доступ до запуску MATLAB моделей для студентів та викладачів без необхідності глибоких технічних знань.
	Приватні дослідники	Використання мобільного або настільного пристрою для тестування й аналізу моделей у реальному часі.
	Інженерні компанії	Швидкий і гнучкий запуск моделей із будь-якого пристрою, зменшення часу на підготовку обчислень.

Отже, запропонована система являє собою інноваційне рішення у сфері автоматизації інженерних обчислень, яке поєднує простоту використання з потужними можливостями MATLAB. Аналіз техніко-економічних характеристик ідеї порівняно з іншими підходами подано в таблиці 4.

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї.

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона	S (сильна сторона)
		Мій проєкт	Конкурент1	Конкурент2	Конкурент3			
1.	Доступність	+	+	-	-	-	-	+
2.	Зручність	+	+	+	+	-	+	-
3.	Інтеграція	+	-	-	-	-	-	+
4.	Вартість, у.о.	2000	10000	1000	5000	-	+	-
5.	Масштабованість	+	-	-	-	-	-	+
6.	Надійність	+	+	-	-	+	-	-

Аналізуючи таблицю 4.2, можна зробити висновок, що основними сильними сторонами проєкту є його зручність, доступність, масштабованість і інтеграція з популярними технологіями. Це забезпечує значні переваги порівняно з традиційними підходами. Нейтральною характеристикою можна вважати вартість системи, яка є прийнятною для цільової аудиторії, але в деяких випадках може поступатися дешевшим альтернативам. Водночас, слабких сторін у системи практично немає, оскільки її архітектура базується на перевірених рішеннях.

Таким чином, запропонована система запуску MATLAB моделей через Telegram-бот має значний потенціал для впровадження у різні галузі, забезпечуючи інноваційний підхід до автоматизації інженерних обчислень і аналізу.

Аудит технологічних рішень є критичним кроком для забезпечення високої продуктивності та ефективності будь-якого проєкту [47]. У контексті створення системи запуску MATLAB моделей через Telegram-бот, такий аудит включає вибір оптимальних мов програмування, фреймворків і бібліотек, які найкраще відповідають потребам проєкту. Крім того, цей процес враховує доступність обраних технологій, їхню підтримку спільнотою розробників та можливість масштабування.

Вибір технологій базувався на аналізі вимог проєкту, зокрема необхідності забезпечення паралельного виконання моделей, інтеграції MATLAB Engine API, зручності використання Telegram API, а також надійності системи зберігання даних.

Підбір інструментів, які оптимально відповідають цим критеріям, дозволяє мінімізувати технічні ризики та забезпечити успішну реалізацію проєкту. У таблиці 4.3 наведено огляд технологій, обраних для реалізації системи.

Таблиця 4.3 – Технологічна здійсненність ідеї проєкту

№ п/п	Ідея проек-ту	Технології її реалізації	Наявність технологій	Доступність техно-логій
1.	Система запуску MATLAB моделей через Telegram-бот	Java	Наявна	Доступна
		Spring Framework	Наявна	Доступна
		Telegram API	Наявна	Доступна
		Mongo DB	Наявна	Доступна
		Telegram-bots	Наявна	Доступна
		MATLAB Engine API	Наявна	Доступна
		MATLAB	Наявна	Доступна
		HTML	Наявна	Доступна
		CSS	Наявна	Доступна
		JS	Наявна	Доступна
Обрані технології реалізації ідеї проєкту: Java, Spring Framework, Telegram API, MATLAB, MATLAB Engine API, MongoDB, Telegram-bots.				

Проаналізувавши таблицю, можна зробити висновок, що обрані технології задовольняють усі потреби та вимоги проєкту. Використання Java як основної мови програмування разом із фреймворками та бібліотеками, такими як Spring Boot Framework, забезпечує створення надійної, масштабованої та легко підтримуваної системи. Java відзначається високою продуктивністю та ефективною реалізацією багатопоточності, що критично важливо для обробки MATLAB моделей у паралельному режимі.

Spring Framework виступає основним фреймворком для створення мікросервісної архітектури, забезпечуючи швидку інтеграцію компонентів, таких як Telegram API і MATLAB Engine API. Telegram API дозволяє реалізувати інтерактивний інтерфейс для взаємодії користувача із системою через месенджер. MATLAB Engine API інтегрує можливості MATLAB для виконання моделей і забезпечує доступ до обчислювальних потужностей безпосередньо з Java.

MongoDB використовується для зберігання метаданих моделей, таких як ідентифікатори користувачів, назви моделей і статуси виконання. Вибір MongoDB обґрунтований її гнучкістю та зручністю роботи з неструктурованими даними, що робить її оптимальним рішенням для мікросервісної архітектури.

Таким чином, запропоновані технологічні рішення дозволяють створити систему, яка відповідає сучасним вимогам до інноваційних стартап-проєктів. Використання мікросервісної архітектури, обраних мов і фреймворків забезпечує ефективну інтеграцію компонентів і високу продуктивність системи запуску MATLAB моделей через Telegram-бот.

4.2 Аналіз ринкових можливостей запуску стартап проєкту

Аналіз ринкових можливостей і потенційних ризиків є важливим етапом стратегічного планування стартап-проєкту, присвяченого запуску MATLAB моделей через Telegram-бот. Оцінка наявного ринку та його динаміки дозволяє краще зрозуміти потенційних користувачів, адаптувати систему до їхніх потреб і створити оптимальну стратегію впровадження проєкту. Зокрема, аналіз ринкових можливостей допомагає визначити основні сегменти цільової аудиторії, зрозуміти ключові потреби користувачів, а також виділити переваги запропонованої системи перед конкурентами.

У цьому контексті система запуску MATLAB моделей через Telegram-бот може знайти застосування в освітніх установах, дослідницьких організаціях та інженерних

компаніях. Попит на таке рішення обумовлений необхідністю спрощення процесу запуску складних обчислень і забезпечення зручного доступу до моделей із мобільних та настільних пристроїв. Аналіз ринкових можливостей також передбачає оцінку бар'єрів для входу на ринок, включаючи технічні та організаційні аспекти, які можуть вплинути на реалізацію проєкту. У табл. 4.4 представлено основні характеристики потенційного ринку.

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проєкту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	4
2	Загальний обсяг продаж, грн/ум.од	150 000
3	Динаміка ринку (якісна оцінка)	зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Технічні обмеження
5	Специфічні вимоги до стандартизації та сертифікації	MATLAB API, Telegram API
6	Середня норма рентабельності в галузі (або по ринку), %	65

Результати аналізу підтверджують високий потенціал для впровадження стартапу на ринку автоматизації інженерних обчислень. Спостерігається стабільний попит і позитивна динаміка розвитку ринку, що робить запропоновану систему конкурентоспроможною. Висока рентабельність у цій галузі підкреслює економічну доцільність реалізації проєкту.

Також було проведено аналіз потреб потенційних користувачів, що дозволило визначити ключові характеристики запропонованого рішення. У табл. 4.5 наведено сегментацію ринку з урахуванням потреб різних груп користувачів.

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проєкту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Зручний доступ до запуску MATLAB моделей	Студенти, викладачі	Залежність від досвіду користувачів	Простота використання, інтеграція з Telegram
2	Гнучке налаштування та запуск моделей	Інженерні компанії	Потребують точного налаштування процесу	Швидкість роботи, надійність

Висновки підтверджують, що запропонована система здатна задовольнити потреби цільових сегментів ринку, зокрема студентів, дослідників і професіоналів у сфері інженерних обчислень. Інтеграція з MATLAB API та Telegram API забезпечує унікальну зручність користування, дозволяючи запускати моделі віддалено та отримувати результати в режимі реального часу.

Загалом, аналіз ринкових можливостей показує, що проєкт має високий потенціал для успіху, проте вимагає уваги до технічних деталей реалізації та забезпечення простоти використання для кінцевих користувачів. Для успішного виходу на ринок необхідно створити ефективну маркетингову стратегію, яка

підкреслюватиме ключові переваги системи та її інноваційний підхід до автоматизації моделювання.

Таблиця 4.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Опоненти симуляції MATLAB	Деякі групи користувачів можуть скептично ставитися до використання MATLAB для моделювання через складність або недовіру до результатів симуляції.	Проведення інформаційної кампанії про переваги MATLAB-симуляцій, таких як точність, гнучкість і швидкість.
2.	Технічні недоліки	Можливі помилки в реалізації алгоритмів чи модулів симуляції через складність коду або нестачу тестування.	Тісна співпраця з користувачами, створення тестових сценаріїв, випуск бета-версій для зворотного зв’язку.
3.	Обмеження апаратних ресурсів	MATLAB-моделі можуть вимагати значних ресурсів для складних обчислень, що обмежує можливість роботи на слабких пристроях.	Оптимізація моделей і використання хмарних обчислень для зменшення навантаження на локальні системи.

Аналіз факторів загроз показує, що впровадження MATLAB-симуляцій у поточні процеси може зіткнутися з викликами, пов’язаними як з технічними, так і соціальними аспектами. Інформаційна робота та технічна оптимізація є ключовими стратегіями для їх подолання.

Окрім потенційних загроз, важливо враховувати наявні можливості, що наведені в таблиці. 4.7.

Враховуючи ці можливості, проєкт MATLAB-симуляцій має значний потенціал для розвитку та впровадження. Особливо актуальним є використання технологічного прогресу та залучення зацікавлених сторін для успіху проєкту

Таблиця 4.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Технологічний розвиток	Поширення високопродуктивних обчислювальних технологій і вдосконалення алгоритмів MATLAB.	Використання новітніх функцій MATLAB та інтеграція з іншими платформами для створення інноваційного ПЗ.
2.	Зростання попиту	Підвищення зацікавленості в автоматизації процесів та точних симуляціях у різних галузях.	Розширення функціональності програмного забезпечення для вирішення задач у нових галузях.
3.	Глобалізація	Необхідність у глобальних стандартах для обчислень і моделювання, особливо для міжнародних проєктів.	Розробка програмного забезпечення, яке підтримує кілька мов і міжнародні стандарти обчислень.
4.	Ефективність	MATLAB-симуляції дозволяють скоротити витрати на реальні експерименти, підвищити швидкість розрахунків.	Оптимізація алгоритмів для мінімізації витрат часу й ресурсів.
5.	Підтримка академічної спільноти	MATLAB активно використовується в університетах і наукових дослідженнях, що створює базу потенційних користувачів.	Створення партнерств із навчальними закладами та науковими центрами для популяризації продукту.

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції - чиста	MATLAB-симуляції мають обмежену кількість аналогів в Україні.	Першими запропонувати комбіновану систему симуляцій з інтеграцією до різних середовищ.
2. За рівнем конкурентної боротьби - міжнародний	MATLAB використовується у всьому світі, конкуренція є глобальною	Розробка унікальних функцій, які враховують локальні особливості українського ринку.
3. За галузевою ознакою - міжгалузева	MATLAB-симуляції можуть застосовуватися в різних галузях, від освіти до промисловості.	Розширення універсальності системи для підтримки специфічних потреб кожної галузі.

Аналіз конкурентного середовища для системи запуску MATLAB-моделей через Telegram-бот в Україні виявляє кілька ключових особливостей. Перш за все, ринок є нішевим і перебуває на етапі формування. Основні конкуренти – це традиційні підходи до моделювання, що включають локальний запуск або використання хмарних платформ, які не мають інтеграції з Telegram. Це створює унікальну можливість для системи стати першопрохідцем і лідером у цій сфері.

Однак система, націлена на користувачів України, має значний потенціал для розширення на міжнародний рівень, зважаючи на глобальний попит на автоматизацію процесів моделювання. Система може знайти своє місце у різних галузях, таких як наукові дослідження, освіта, інженерія та промислове моделювання.

Для успішного позиціонування необхідно інвестувати в активну рекламну кампанію, проведення воркшопів і вебінарів, що демонструють переваги системи перед традиційними методами.

З урахуванням виявлених характеристик конкуренції, доцільно провести подальший аналіз за моделлю М. Портера для глибшого розуміння стратегічних можливостей та викликів у даній галузі. Цей аналіз представлений у таблиці 4.9.

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	В Україні відсутні. У світі: традиційні способи запуску через MATLAB Desktop, MATLAB Online, хмарні сервіси (наприклад, MathWorks Cloud).	Розробники Telegram-ботів для наукових задач або компанії, які автоматизують процеси моделювання MATLAB.	Постачальники програмного забезпечення для MATLAB (MathWorks) та розробники API Telegram.	Розробники, інженери, науковці, викладачі, студенти, які використовують MATLAB у своїй роботі чи навчанні.	Альтернативні рішення для автоматизації: - Хмарні платформи - Сервіси з індивідуальними інтерфейсами для запуску моделей - Локальні скрипти для автоматизації.
Висновки:	Конкуренти пропонують багатий функціонал у традиційних продуктах, але відсутня інтеграція з месенджерами, що є унікальною перевагою нашої системи.	Висока зацікавленість у галузі автоматизації серед IT-спільноти може призвести до появи конкурентних Telegram-ботів або аналогічних платформ.	Постачальники доступні, MATLAB API і Telegram API є безкоштовними або мають низьку вартість	Клієнти очікують простоти використання, інтеграції з їхніми робочими процесами, підтримки паралельного моделювання та низької вартості.	Замінники мають обмежену інтеграцію і складніший інтерфейс, проте можуть бути конкурентоспроможними через доступність і популярність традиційних підходів.

Аналіз, проведений за моделлю М. Портером, дозволяє отримати комплексне розуміння конкурентного середовища для системи запуску MATLAB-моделей через Telegram-бот. Прямі конкуренти у вигляді традиційних підходів мають суттєві обмеження в інтеграції з месенджерами та підтримці паралельного моделювання, що створює унікальні можливості для нашої системи.

Потенційні конкуренти можуть виникнути через зростання інтересу до автоматизації серед ІТ-фахівців, тому важливо швидко зайняти лідируючі позиції на ринку. Постачальники програмного забезпечення для розробки є доступними, що спрощує процес створення продукту.

Клієнти мають високі очікування щодо зручності та ефективності, що визначає ключові напрями вдосконалення системи. Альтернативні рішення не можуть забезпечити такого рівня інтеграції та автоматизації, однак залишаються конкурентами через звичність та доступність.

Наступним кроком може бути вивчення можливостей для позиціонування продукту, а також стратегій для забезпечення його конкурентоспроможності в умовах зростаючої зацікавленості та конкуренції.

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проєктів значущим)
1	Інтеграція з популярними платформами	Легкий у використанні інтерфейс дозволяє запускати MATLAB-моделі з будь-якого місця, де є інтернет, без необхідності встановлювати додаткове ПЗ або мати доступ до потужних комп'ютерів.
2	Простота та зручність використання	Використання Telegram як інтерфейсу забезпечує доступність продукту для широкої аудиторії, адже Telegram є одним із найпопулярніших месенджерів у світі.

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності (продовження)

3	Економічність	Зниження витрат завдяки автоматизації процесу запуску моделей, відсутності потреби в дорогих локальних ресурсах або хмарних сервісах для моделювання.
4	Підтримка паралельного моделювання	Можливість запускати декілька моделей одночасно забезпечує високу продуктивність і значну економію часу для користувачів

Аналіз факторів конкурентоспроможності системи запуску MATLAB-моделей через Telegram-бот дозволяє визначити ключові характеристики, які забезпечують переваги над традиційними способами моделювання. Основні висновки:

- Інтеграція з популярними платформами - використання Telegram забезпечує доступність і простоту використання для широкої аудиторії.
- Простота і зручність - користувачі отримують змогу запускати моделі з будь-якого пристрою, без потреби у складних налаштуваннях.
- Економічність - зниження витрат робить продукт привабливим для освітніх закладів, малих компаній та окремих користувачів.
- Підтримка паралельного моделювання - забезпечує швидке виконання задач і значну економію часу.

Ці фактори формують унікальну конкурентну перевагу продукту. Для закріплення позицій на ринку необхідно зосередитися на підвищенні безпеки, гнучкості та прозорості системи.

.

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з «Система запуску моделювань MATLAB моделей через TG Bot»						
			–3	–2	–1	0	+1	+2	+3
1	Продуктивність обчислень	18							+
2	Швидкість виконання моделей	15							+
3	Гнучкість моделюванн	17							+
4	Інтерфейс користувач	19			+				
5.	Надійність та точність	14				+			

1. Продуктивність обчислень (18 балів): MATLAB має високу продуктивність у розв'язуванні математичних задач і моделюванні складних систем, що робить його потужним інструментом для обробки даних.
2. Швидкість виконання моделей (15 балів): Програма швидко обробляє числові обчислення, проте є можливість для подальшого оптимізування під час роботи з великими даними.
3. Гнучкість моделювання (17 балів): MATLAB підтримує широкий спектр моделей і алгоритмів, що дозволяє створювати різноманітні моделі та експериментувати з різними підходами.
4. Інтерфейс користувача (19 балів): MATLAB має добре розроблений інтерфейс, який дозволяє легко працювати з візуалізацією даних і настройками моделей.
5. Надійність та точність (14 балів): MATLAB гарантує високу точність обчислень, але для деяких специфічних завдань є потенціал для поліпшення надійності.

Загальний висновок: MATLAB має важливі конкурентні переваги в розв'язанні математичних задач, продуктивності та гнучкості. Однак, потрібно звернути увагу на можливість підвищення надійності та швидкості виконання складних моделей.

Таблиця 4.12 – SWOT- аналіз стартап-проєкту

Сильні сторони: 1. Висока точність розрахунків 2. Широкий вибір алгоритмів для обробки даних. 3. Простота використання 4. Гнучкість і масштабованість.	Слабкі сторони: 1. Потреба в потужному апаратному забезпеченні 2. Вартість ліцензії MATLAB для комерційних користувачів. 3. Велика залежність від спеціалізованих бібліотек.
Можливості: 1. Інтеграція з іншими програмними засобами. 2. Поширення використання MATLAB у наукових дослідженнях.	1. Загрози: 2. Зростаюча конкуренція з іншими мовами програмування. 3. Недовіра до результатів через недостатню прозорість алгоритмів. 4. Обмежена підтримка для специфічних обчислювальних платформ

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проєкту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Стратегія оптимізації алгоритмів для підвищення продуктивності	Середня	8 міс.
2	Стратегія інтеграції з іншими системами для більш широкого застосування	Висока	10 міс.
3	Стратегія покращення підтримки платформ для наукових досліджень	Середня	12 міс.

Обираємо стратегію оптимізації алгоритмів для підвищення продуктивності, щоб зменшити час виконання складних моделей і забезпечити ефективне використання системи в реальному часі.

4.3 Розроблення ринкової стратегії та маркетингової програми стартап-проєкту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів.

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Освітні установи	Висока	Середній	Низька	Низька
2	Інженерні компанії	Середня	Високий	Висока	Середня
3	Незалежні дослідники	Висока	Середній	Низька	Низька
Які цільові групи обрано: освітні установи					

Основною групою для впровадження продукту обрано **освітні установи**, оскільки вони демонструють високий рівень готовності прийняти інноваційний продукт і характеризуються значним потенційним попитом.

Аналіз виявив, що саме освітні заклади є найбільш перспективним сегментом для старту проєкту. Вони зацікавлені у використанні системи для підтримки навчального процесу, лабораторних робіт та проведення наукових досліджень. Це середовище з низькою конкуренцією дозволить швидко зайняти свою нішу, запропонувавши інструмент, що суттєво спрощує запуск та аналіз MATLAB моделей.

Разом із тим, інженерні компанії можуть стати важливим сегментом для розвитку продукту на наступному етапі, оскільки вони демонструють стабільний середній рівень попиту, але потребують детальнішого опрацювання функціональності продукту відповідно до їхніх запитів. Незалежні дослідники, хоча

й представляють малий сегмент ринку, також можуть стати лояльною аудиторією завдяки простоті та доступності системи.

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку.

Таблиця 4.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Впровадження системи запуску MATLAB моделей через Telegram-бот	Фокусна стратегія	Технологічна перевага, висока адаптивність до потреб освітніх установ	Стратегія диференціації та інноваційного лідерства

На основі аналізу таблиці 4.15 визначено, що оптимальною стратегією розвитку для стартап-проєкту є "Стратегія диференціації та інноваційного лідерства". Ця стратегія передбачає акцентування уваги на технологічних перевагах продукту, таких як автоматизація запуску моделей, спрощення доступу до моделювань та інтеграція з популярним месенджером Telegram. Основна мета цієї стратегії – створення унікального продукту, який відповідає конкретним потребам освітніх установ та інженерних компаній.

Фокус на інноваціях дозволить стартапу виділитися на ринку, забезпечуючи високу конкурентоспроможність і можливість формування довгострокового партнерства з цільовими клієнтами. Такий підхід сприятиме розширенню ринку завдяки інтеграції новітніх технологій і адаптації продукту до потреб користувачів.

Наступним кроком оберемо стратегію конкурентної поведінки (табл. 4.16).

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проєкт «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Ні	Шукати нових споживачів	Не буде копіювати, але намагатиметься вдосконалити основні характеристики	Стратегія заняття конкурентної ніші

На основі аналізу таблиці 4.16 визначено, що оптимальною стратегією конкурентної поведінки є "Стратегія заняття конкурентної ніші". Ця стратегія зосереджується на створенні унікальної ніші для продукту, орієнтуючись на пошук нових споживачів, таких як освітні установи, дослідницькі організації та інженерні компанії.

Замість прямої конкуренції стартап прагне вдосконалювати свої послуги, забезпечуючи більш високу зручність використання, адаптацію під специфічні потреби та ефективність у порівнянні з існуючими альтернативами. Акцент на унікальності пропозиції дозволяє зміцнити конкурентні позиції компанії, забезпечуючи її стабільне зростання і підвищення впізнаваності бренду.

На основі потреб споживачів у обраному сегменті, а також стратегії розвитку та конкурентної поведінки, розроблено стратегію позиціювання, яка визначає, як споживачі мають сприймати наш проєкт.

У таблиці 4.17 відображено ключові аспекти стратегії позиціонування для стартап-проєкту, який забезпечує автоматизацію запуску MATLAB моделей через Telegram-бот. Основними конкурентоспроможними перевагами є інтеграція з Telegram, що забезпечує простоту доступу з будь-якого пристрою, та можливість паралельного запуску моделей MATLAB, що значно підвищує швидкість і ефективність обчислень. Ці переваги формують у споживачів асоціації з доступністю, високою продуктивністю та зручністю використання.

Таблиця 4.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту (три ключових)
1	Автоматизація запуску моделей, легкість використання	Стратегія диферен ціації	Інтеграція з Telegram для забезпечення доступності та зручності	Інтеграція з Telegram. Доступність з будь-якого пристрою. Простота використання
2	Ефективність, швидкість і точність виконання	Стратегія фокусу	Паралельний запуск моделей MATLAB для зменшення часу обчислень	Паралельність виконання. Ефективність використання ресурсів. Висока продуктивність

Поєднання диференціації та фокусування на потребах цільової аудиторії дозволяє стартапу зайняти стійку позицію на ринку, привертаючи увагу інженерів, освітніх установ та дослідницьких організацій, які потребують швидких і надійних рішень для запуску моделювань.

Надалі проведено аналіз конкурентоспроможності товару. Таблиця 4.18 підсумовує ключові переваги концепції стартап-проєкту, що полягає в розробці системи для запуску MATLAB моделей через Telegram-бот. Основними перевагами є висока безпека і надійність запуску моделей, забезпечені за допомогою паралельної обробки та застосування криптографії для захисту результатів. Прозорість процесу забезпечується завдяки відкритим даним та можливості перевірки результатів, що гарантує довіру користувачів. Крім того, інтуїтивно зрозумілий інтерфейс та легка інтеграція з іншими електронними системами надають додаткові переваги, роблячи продукт доступним і зручним для широкої аудиторії користувачів, включаючи студентів, викладачів, науковців і інженерів.

Таблиця 4.18. Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Безпечність та надійність запуску моделей	Забезпечення безпечного та надійного запуску MATLAB моделей	Паралельне виконання моделей з використанням криптографічних методів та дистрибуції навантаження для підвищення надійності.
2	Швидкість обчислень та виконання моделей	Забезпечення швидкого запуску моделей через Telegram-бот	Паралельність виконання, можливість обробки кількох моделей одночасно для зниження часу обчислень.
3	Легкість використання для різних груп користувачів	Інтуїтивний інтерфейс для запуску моделей та отримання результатів	Інтеграція з Telegram, доступність на будь-якому пристрої, зручність для користувачів без технічних знань.

Таблиця 4.19. Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
I. Товар за задумом	Система запуску MATLAB моделей через Telegram-бот
II. Товар у реальному виконанні	Властивості/характеристики
	1. Зручність 2. Ефективність 3. Легкість використання 4. Уникнення встановлення великих обсягів ПЗ
	Якість: Сертифікація за стандартом ISO 27001
	Пакування: .
	Марка:
III. Товар із підкріпленням	До продажу – Індивідуалізація рішень під конкретні потреби організацій чи наукових установ
	Після продажу – Технічна підтримка, навчання користувачів, система оновлень
Захист системи від плагіату за допомогою патентування	

Модель товару на трьох рівнях для системи запуску MATLAB моделей через Telegram-бот відображає ключові аспекти, що визначають її концепцію, реалізацію та підтримку. На першому рівні товар є системою, яка надає можливість запускати складні моделі MATLAB через зручний інтерфейс Telegram-бота. На рівні реального виконання товар має кілька важливих характеристик, зокрема безпечність,

прозорість, легкість використання, а також високий рівень захисту від атак. Також важливими характеристиками є вартість, незалежність та сертифікація за міжнародними стандартами. На третьому рівні товар надає додаткові послуги, такі як індивідуалізація рішень під конкретні потреби користувачів, технічна підтримка та навчання персоналу, а також захист системи від плагіату через патентування. Такий підхід дозволяє забезпечити повний цикл роботи з клієнтами, що гарантує високу конкурентоспроможність продукту на ринку, роблячи його доступним для широкого кола користувачів та адаптованим до їхніх потреб.

Таблиця 4.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	200-20 000 у.о.	1000 – 5000 у.о.	>1 000 000 у.о.	800-4 000 у.о.

Аналізуючи встановлення цін для системи запуску MATLAB моделей через Telegram-бот, важливо враховувати конкурентну середовище, рівень цін на товари-замінники та товари-аналоги, а також доходи цільової групи споживачів. Зазначені верхні та нижні межі встановлення ціни, які знаходяться в діапазоні від 800 до 4 000 у.о., спроєктовані для забезпечення конкурентоспроможності продукту на ринку та відповідності фінансовим можливостям цільової аудиторії.

Таблиця 4.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Договір на поставку	Навчання персоналу, технічна підтримка, доопрацювання при бажанні замовника	Канал збуту прямий	Прямий збут (продаж власними силами)

Звертаючись до специфіки закупівельної поведінки цільових клієнтів (таблиця 4.21), основні функції збуту для цієї системи включають навчання персоналу, технічної підтримки, можливості вдосконалення за бажання замовника. Обрана стратегія глибини каналу збуту є прямою, без посередників, що підкреслює необхідність прямого взаємодії постачальника з клієнтами. Оптимальний підхід до системи збуту полягає в прямому збуті, де продажі здійснюються безпосередньо компанією, що сприяє ефективності та контролю над процесом збуту.

Таблиця 4.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Рішення використовувати систему запуску MATLAB моделей через Telegram-бот	Онлайн-канали, вебінари, технологічні форуми, спеціалізовані семінари	Безпека, прозорість, надійність, простота використання	Заохочення довіри до системи, інформування про її переваги	Простота використання, швидкість запуску, безпека та надійність роботи з MATLAB моделями через Telegram-бо

Аналізуючи специфіку поведінки цільових клієнтів, які зацікавлені у використанні системи запуску MATLAB моделей через Telegram-бот, можна відзначити, що основними каналами комунікацій для цієї цільової аудиторії є онлайн-платформи, вебінари, технологічні форуми та спеціалізовані семінари. Ключові позиції для позиціонування цього продукту включають безпеку, прозорість, надійність та простоту використання, що є важливими для ефективного запуску моделей через Telegram-бот. Завдання рекламного повідомлення полягає в заохоченні довіри до системи та інформуванні потенційних користувачів про її переваги.

Концепція рекламного звернення акцентує увагу на простоті використання, швидкості запуску моделей та безпеці роботи з MATLAB моделями через Telegram-бот.

4.4 Розроблення бізнес-моделі стартап-проєкту

Розробка бізнес-моделі для стартап-проєкту, що фокусується на створенні системи запуску MATLAB моделей через Telegram-бот, є важливим етапом для забезпечення успіху та стабільності проєкту. Для успішної реалізації необхідно чітко визначити ключові елементи цієї моделі.

Перше, що потрібно зробити, — це визначити цільові групи користувачів, зацікавлених у використанні системи. Це можуть бути освітні установи, дослідники, інженерні компанії та окремі користувачі, які потребують швидкого та зручного запуску MATLAB моделей через Telegram. Усі ці групи мають різні потреби, але об'єднує їх зацікавленість у простоті використання та доступності інструментів для моделювання.

Ключовою перевагою цієї системи є її унікальність і можливість запуску MATLAB моделей через Telegram-бот, що дозволяє користувачам проводити моделювання на будь-якому пристрої без необхідності мати доступ до складного програмного забезпечення. Важливо підкреслити, що цей продукт пропонує зручність, швидкість та доступність, а також дозволяє користувачам працювати з моделями у реальному часі.

Вибір стратегії доходів має включати різні варіанти монетизації, такі як підписка на послугу, оплата за кожне використання моделі або комбінація кількох моделей монетизації. Можна також розглянути створення преміум-версії для корпоративних клієнтів, що потребують додаткових функцій або персоналізації.

Для успішної реалізації проєкту важливо розглянути можливість співпраці з технологічними партнерами, провайдерами послуг, такими як постачальники

серверних потужностей, а також консультантами з безпеки, що допоможуть забезпечити належний рівень захисту даних. Співпраця з освітніми установами або інженерними компаніями може стати важливим фактором для легітимації та впровадження системи.

Процеси, необхідні для розробки, впровадження та підтримки системи, мають включати розробку програмного забезпечення, створення інтеграцій з існуючими платформами, а також регулярне оновлення та підтримку клієнтів. Окрему увагу слід приділяти зручності користування системою та її постійному вдосконаленню з урахуванням зворотного зв'язку від користувачів.

Фінансування проєкту може здійснюватися через різні джерела, такі як інвестори, краудфандинг, гранти на підтримку інновацій у галузі освіти чи технологій, а також через партнерства з великими компаніями та університетами.

Враховуючи ці ключові елементи, розроблення ефективної бізнес-моделі для стартапу, що надає послугу запуску MATLAB моделей через Telegram-бот, дозволить забезпечити стабільність проєкту, залучити інвесторів, партнерів та користувачів, а також реалізувати проєкт на великому масштабі.

<table>
<tr>
<td rowspan="2">Ключові партнери:
Органи влади, виборчі комісії, технологічні партнери
Мотивація для партнерства:
Партнерство для розвитку та інтеграції технологі</td>
<td rowspan="2">Ключові види діяльності:
Розробка системи запуску MATLAB моделей, забезпечення підтримки та постійне вдосконалення програмного забезпечення.</td>
<td rowspan="2">Ціннісна пропозиція
Простота використання системи для запуску MATLAB моделей, доступність через Telegram-бот, висока ефективність та прозорість процесу.</td>
<td>Відносини з клієнтами
Персоналізовані рішення для кожного клієнта, зручний інтерфейс та оперативна технічна підтримка.</td>
<td rowspan="2">Сегменти користувачів:
Користувачі, які потребують швидкого та зручного запуску MATLAB моделей через Telegram..
Ринок
Освітні установи, дослідницькі організації, приватні компанії.</td>
</tr>
<tr>
<td>Канали
Прямий збут, власний відділ по роботі з клієнтами</td>
</tr>
<tr>
<td colspan="2">Структура виплат:
Розробка, підтримка та маркетинг</td>
<td colspan="3">Джерела доходів:
Платні послуги для організаторів моделювань та ліцензійні угоди на використання системи.</td>
</tr>
</table>

Отже, компанія спрямована на розробку та впровадження інноваційної системи для запуску MATLAB моделей через Telegram-бот. Основна мета — забезпечити зручний, безпечний та прозорий процес моделювання для різних категорій користувачів, таких як студенти, викладачі, дослідники та інженери. Завдяки індивідуальному підходу до кожного клієнта та постійному контакту з ними, компанія прагне забезпечити високу якість своїх послуг. Основні джерела доходів включають платні послуги для наукових установ, університетів та приватних компаній, а також ліцензійні угоди на використання розробленої системи.

Зазначена бізнес-модель стартап-проєкту для системи запуску MATLAB моделей через Telegram-бот демонструє чітку стратегію партнерства, спрямовану на взаємодію з освітніми та науковими установами, а також з технологічними партнерами. Мотивація для співпраці ґрунтується на забезпеченні високої безпеки, прозорості та надійності моделювання, що здійснюється за допомогою Telegram-бота.

Ключові види діяльності сфокусовані на розробці та вдосконаленні системи для запуску MATLAB моделей, інтеграції нових методів для забезпечення безпеки та стабільної роботи бота. Ціннісна пропозиція проєкту полягає в наданні користувачам зручного та безпечного інтерфейсу для запуску моделей у реальному часі з використанням Telegram, що відкриває нові можливості для організації досліджень, наукових робіт та інженерних розрахунків.

Відносини з клієнтами базуються на індивідуальному підході до кожної організації, забезпечуючи гнучкість системи та адаптивність до конкретних вимог користувачів. Планування роботи з різними сегментами користувачів, включаючи освітні установи, дослідницькі організації та інженерні компанії, підтверджує гнучкість і масштабованість системи.

Управління каналами розповсюдження включає прямий доступ до клієнтів через власні платформи, а також стратегічне партнерство з технологічними компаніями та іншими ключовими учасниками ринку. Структура витрат розроблена таким чином, щоб ефективно використовувати ресурси на дослідження, розробку, маркетинг та підтримку системи, підкреслюючи пріоритет якості, безпеки та надійності продукту.

4.5 Висновки до розділу 4

Розділ, присвячений стартап-проєкту системи запуску моделювань MATLAB моделей за допомогою месенджера Telegram, детально аналізує ключові стратегічні аспекти позиціонування та впровадження продукту на ринку. Особлива увага приділяється різним цільовим групам користувачів, включаючи студентів, викладачів, дослідників та інженерів, які активно використовують MATLAB у своїй професійній діяльності.

У ході дослідження застосовано такі інструменти, як SWOT-аналіз, аналіз конкурентів, а також оцінка потенційних загроз та ризиків. Ці підходи дозволили глибше зрозуміти динаміку ринку, його особливості та потреби кінцевих користувачів. Особливу увагу приділено аналізу конкурентних переваг, які поєднують зручність, швидкість запуску моделей та інтеграцію з популярним месенджером Telegram, що створює унікальну цінність для користувачів.

Стратегія маркетингового планування розробляється з урахуванням специфіки ринкових умов, сучасних технологічних трендів та регуляторних вимог. Особливий акцент зроблено на визначенні оптимальної моделі ціноутворення, розробці комунікаційних каналів для залучення користувачів та впровадженні механізмів, які забезпечать високу лояльність клієнтів. Вибір Telegram як платформи для взаємодії з користувачами зумовлений його широким поширенням, зручністю у використанні та можливістю адаптації функціоналу під специфічні потреби системи.

Додатковий аналіз показав, що комбінований підхід до запуску моделювань, який включає інтеграцію MATLAB Engine, мікросервісної архітектури та Telegram-бота, забезпечує ефективність, масштабованість і простоту використання. Цей підхід дозволяє не лише виконувати моделювання швидко та зручно, але й організувати чітку взаємодію між користувачами та системою, підвищуючи її конкурентоспроможність на ринку.

Подальший розвиток продукту базується на постійному вдосконаленні функціональності, адаптації до змінних вимог користувачів та впровадженні

інноваційних рішень для забезпечення надійності та продуктивності. Гнучкість системи, у поєднанні з використанням сучасних технологічних рішень, дозволяє не лише адаптуватися до швидких змін на ринку, а й забезпечити її стійке позиціонування.

Таким чином, розроблена система має значний потенціал для успішного впровадження, адже вона відповідає актуальним потребам цільової аудиторії, інтегрує інноваційний підхід до автоматизації моделювань і створює додаткову цінність завдяки своїй зручності та функціональності.

ВИСНОВОК

У цій магістерській дисертації] було розроблено, реалізовано та досліджено комбінований підхід до запуску моделювань MATLAB моделей, що ґрунтується на мікросервісній архітектурі та багатопотоковості. Запропонована система дозволяє значно підвищити ефективність процесу моделювання завдяки інтеграції MATLAB Engine API із сучасними технологіями обчислень і комунікації.

У процесі виконання роботи проведено огляд стану проблеми автоматизованого моделювання, що дозволило визначити основні виклики. Проведений аналіз сучасних підходів, включаючи паралельні та розподілені обчислення, що надало змогу сформулювати основні критерії для розробки комбінованого підходу.

Проведено детальний огляд існуючих рішень у сфері багатопотокового моделювання MATLAB моделей. Проаналізовано існуючі наукові публікації та інструменти, зокрема Parallel Computing Toolbox, MATLAB Distributed Computing Server, а також підходи до багатопотокового виконання моделювань. Розглянуто інструмент, такий як MATLAB Engine API, що дозволило зробити інтеграцію MATLAB з мовою програмування Java.

Розроблено архітектуру програмної реалізації, що включає низку мікросервісів, кожен із яких виконує конекретну функцію: прийом задач, обробка вхідних даних, виконання моделювань, обробка результатів та взаємодія з користувачами через Telegram Bot. Для забезпечення ефективності використано сучасні технології: Spring Framework для розроблення всіх мікросервісів, Telegram Bot API для інтерактивної взаємодії з користувачами, MATLAB Engine API для виконання обчислень в середовищі MATLAB, MongoDB для зберігання даних, а також механізми багатопотоковості в Java для паралельної обробки завдань.

Працездатність системи була перевірена в реальних умовах, зокрема через інтерактивну взаємодію з користувачами через Telegram Bot. Тестування

підтвердило, що система коректно виконує моделювання, обробляє результати та забезпечує паралельне виконання завдань.

Проведені дослідження ефективності розробленого комбінованого підходу включали моделювання алгоритмів із різною складністю: $O(n)$, $O(log\ n)$, $O(n\ log\ n)$, $O(n^2)$, $O(n!), O(n^n)$. Для кожного алгоритму виконувалися моделювання з кількістю завдань від 1 до 10 у послідовному та паралельному режимах. Результати досліджень підтвердили, що паралельний підхід дозволяє суттєво зменшити час (в середньому, на 57.97%) виконання моделювань.

Відповідно до проведених експериментів підтверджено, що запропонований комбінований підхід є ефективним інструментом для прискорення виконання складних завдань. Однак застосування цього підходу вимагає врахування додаткових вимог до апаратних ресурсів, особливо для обробки моделей із високою складністю. Це підкреслює важливість вибору оптимального способу обробки залежно від доступної обчислювальної інфраструктури та особливостей моделювань.

Отримані результати мають практичне значення, оскільки вони демонструють доцільність використання багатопотокових середовищ для підвищення продуктивності обчислювальних систем. Дослідження також підтвердило важливість оптимізації моделювань і раціонального використання апаратних ресурсів, що є особливо актуальним у контексті сучасних наукових і технічних задач, які характеризуються зростанням обсягів даних і складності систем.

Запропонована система може бути використана як основа для подальших досліджень у напрямку масштабування, оптимізації алгоритмів і розширення функціональних можливостей у роботі з динамічними моделями.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Офіційний веб-сайт MathWorks. What Is MATLAB? URL: https://www.mathworks.com/discovery/what-is-MATLAB.html (дата звернення: 15.10.2024).

2. Wilkinson M. D., Dumontier M., Aalbersberg I. J., et al. The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data. 2016. Vol. 3. P. 160018.

3. Офіційний веб-сайт MathWorks. Parallel Computing Toolbox. URL: https://www.mathworks.com/products/parallel-computing.html (дата звернення: 17.10.2024).

4. Dongarra J., et al. Numerical Linear Algebra for High-Performance Computers. SIAM, 1998.

5. Conte S. D., de Boor C. Elementary Numerical Analysis: An Algorithmic Approach. McGraw-Hill, 1980.

6. A Guide to Python Multiprocessing and Parallel Programming. URL: https://www.sitepoint.com/python-multiprocessing-parallel-programming/ (дата звернення: 18.10.2024).

7. Офіційний веб-сайт MathWorks. MATLAB and Simulink for Technical Computing. URL: https://www.mathworks.com/solutions/technical-service-providers.html (дата звернення: 19.10.2024).

8. Gupta B. R. K., et al. High Performance Computing in Engineering and Applied Science. Springer, 2020.

9. Spring Framework. URL: https://spring.io/projects/spring-framework (дата звернення: 13.11.2024).

10. What is batch processing? URL: https://aws.amazon.com/what-is/batch-processing/ (дата звернення: 20.10.2024).

11. Create and Use Distributed Arrays. URL: https://www.mathworks.com/help/parallel-computing/when-to-use-distributed-arrays.html (дата звернення: 21.10.2024).

12. Reese J., Zaranek S. GPU Programming in MATLAB. MathWorks. URL: https://www.mathworks.com/company/technical-articles/gpu-programming-in-MATLAB.html (дата звернення: 23.10.2024).

13. Parfor. Execute for-loop iterations in parallel on workers. URL: https://www.mathworks.com/help/parallel-computing/parfor.html (дата звернення: 24.10.2024).

14. How MATLAB divide the number of iterations of "parfor" on the workers of one computer? URL: https://stackoverflow.com/questions/44164857/how-MATLAB-divide-the-number-of-iterations-of-parfor-on-the-workers-of-one-com (дата звернення: 25.10.2024).

15. Run Single Programs on Multiple Data Sets. URL: https://www.mathworks.com/help/parallel-computing/execute-simultaneously-on-multiple-data-sets.html (дата звернення: 27.10.2024).

16. Distributed computation using SPMD model. URL: https://oneapi-src.github.io/oneDAL/onedal/spmd/index.html (дата звернення: 28.10.2024).

17. GPU Computing in MATLAB. URL: https://www.mathworks.com/help/parallel-computing/gpu-computing-in-MATLAB.html (дата звернення: 29.10.2024).

18. Szymczyk M., Szymczyk P. Image Processing and Communication. 2014. Vol. 17. No. 4. P. 207–216.

19. Kepner J. Parallel MATLAB for Multicore and Multinode Computers. SIAM, 2009.

20. Guill´en A., Garc´ıa M., Herrera L. J., Pomares H., Rojas I. GPU Cluster with MATLAB. URL: https://www.researchgate.net/publication/258789666_Optimization_of_Function_by_using_a_New_MATLAB_based_Genetic_Algorithm_Procedure (дата звернення: 03.11.2024).

21. MathWorks. MATLAB Engine API Documentation. URL: https://www.mathworks.com (дата звернення: 07.11.2024).https://www.mathworks.com

22. Smith J. Integrating MATLAB with External Applications Using MATLAB Engine API. Journal of Computational Tools, 2023.

23. Johnson M. Numerical Analysis and Simulation in MATLAB Environment. Springer, 2021.

24. MongoDB Atlas. URL: https://www.mongodb.com/ (дата звернення: 12.11.2024).

25. Telegram Bot API. URL: https://core.telegram.org/bots/api (дата звернення: 17.11.2024).

26. Singh H., Aibin M. Creating a Telegram Bot with Spring Boot. Baeldung. URL: https://www.baeldung.com/spring-boot-telegram-bot (дата звернення: 18.11.2024).

27. Складність алгоритмів. URL: https://javarush.com/ua/groups/posts/uk.2325.skladnstjh-algoritmv (дата звернення: 25.11.2024).

28. Чому важливо оцінювати складність алгоритму. URL: https://foxminded.ua/skladnist-alhorytmu/ (дата звернення: 25.11.2024).

29. Алгоритм бінарного пошуку. URL: https://www.guru99.com/uk/binary-search.html (дата звернення: 29.11.2024).

30. Merge Sort – Data Structure and Algorithms Tutorials. URL: https://www.geeksforgeeks.org/merge-sort/ (дата звернення: 29.11.2024).

31. Bubble Sort Algorithm. URL: https://www.geeksforgeeks.org/bubble-sort-algorithm/ (дата звернення: 29.11.2024).

ДОДАТОК А

Скрипт для демонстрації роботи системи

```
% Script for simulating a pendulum with damping and external force
clc;
clear;
close all;

%% Pendulum parameters
g = 9.81;              % Gravitational acceleration (m/s^2)
L = 1.0;               % Pendulum length (m)
b = 0.1;               % Damping coefficient
A = 1.5;               % Amplitude of external force
omega = 2.0;           % Frequency of external force (rad/s)
theta0 = pi/4;         % Initial angle (rad)
omega0 = 0;            % Initial angular velocity (rad/s)

%% Time settings
tspan = [0 20];        % Time interval (s)
y0 = [theta0, omega0]; % Initial conditions [angle, angular velocity]

%% Definition of the ODE system
damped_forced_pendulum = @(t, y) [y(2);
    -(g/L)*sin(y(1)) - b*y(2) + A*sin(omega*t)];

%% Numerical solution
[t, y] = ode45(damped_forced_pendulum, tspan, y0);

%% Console output
disp('Pendulum simulation completed.');
disp(['Maximum angle: ', num2str(max(y(:,1))), ' rad']);
disp(['Minimum angle: ', num2str(min(y(:,1))), ' rad']);
disp(['Simulation time: ', num2str(t(end)), ' seconds']);

%% Plotting
figure;

% Angle of pendulum over time
subplot(2, 1, 1);
plot(t, y(:, 1), 'b-', 'LineWidth', 1.5);
grid on;
xlabel('Time, s');
ylabel('Angle \theta (rad)');
title('Pendulum dynamics');
legend('\theta(t)');

% Phase trajectory
subplot(2, 1, 2);
plot(y(:, 1), y(:, 2), 'r-', 'LineWidth', 1.5);
grid on;
xlabel('Angle \theta (rad)');
ylabel('Angular velocity \omega (rad/s)');
```

```
title('Phase trajectory');
legend('\omega(\theta)');

%% Additional computations
energy = 0.5 * (L^2) * y(:, 2).^2 + g * L * (1 - cos(y(:, 1))); % Total energy

% Energy plot
figure;
plot(t, energy, 'k-', 'LineWidth', 1.5);
grid on;
xlabel('Time, s');
ylabel('Energy (J)');
title('Total mechanical energy of the pendulum');
legend('E(t)');
```

ДОДАТОК Б

Моделі різної складності

1.1 Модель для обчислення суми елементів вектора зі складністю $O(n)$

```
% Скрипт для обчислення суми елементів вектора зі складністю O(n)

n = 1000000;
vector = randi([1, 100], 1, n);

tic;
sum_result = 0;

for i = 1:n
    sum_result = sum_result + vector(i);
end

toc;

disp(['Сума елементів вектора: ', num2str(sum_result)]);
```

1.2 Модель для алгоритму бінарного пошуку зі складністю $O(log\ n)$

```
% Алгоритм з складністю O(log n)

function result = binarySearch(sortedArray, target)
    left = 1;
    right = length(sortedArray);

    while left <= right
        mid = floor((left + right) / 2);

        if sortedArray(mid) == target
            result = mid;
            return;
        elseif sortedArray(mid) < target
            left = mid + 1;
        else
            right = mid - 1;
        end
    end

    result = -1;
end

n = 1000000; % Розмір масиву
```

```
sortedArray = sort(randi([1, 1000000], 1, n)); % Генерація рандомного відсортованого
масиву
target = randi([1, 1000000]); % Рандомний елемент для пошуку

tic;
index = binarySearch(sortedArray, target);
toc;

if index ~= -1
    disp(['Element found at index: ', num2str(index)]);
else
    disp('Element not found.');
end
```

1.3 Модель для сортування масиву методом злиття зі складністю $O(n \log n)$

```
% Алгоритм з складністю O(n log n) - Сортування злиттям (Merge Sort)

function sortedArray = mergeSort(array)
    if length(array) <= 1
        sortedArray = array;
        return;
    end

    mid = floor(length(array) / 2);
    leftHalf = mergeSort(array(1:mid));
    rightHalf = mergeSort(array(mid+1:end));

    sortedArray = merge(leftHalf, rightHalf);
end

function mergedArray = merge(left, right)
    mergedArray = [];
    while ~isempty(left) && ~isempty(right)
        if left(1) < right(1)
            mergedArray = [mergedArray, left(1)];
            left(1) = [];
        else
            mergedArray = [mergedArray, right(1)];
            right(1) = [];
        end
    end

    mergedArray = [mergedArray, left, right];
end

n = 100000; % Розмір масиву
array = randi([1, 1000000], 1, n); % Генерація рандомного масиву
disp('Unsorted array:');
disp(array);

sortedArray = mergeSort(array);
disp('Sorted array:');
disp(sortedArray);
```

1.4 Модель для сортування бульбашкою зі складністю $O(n^2)$

```
% Алгоритм з складністю O(n^2) - Сортування бульбашкою (Bubble Sort)

function sortedArray = bubbleSort(array)
    n = length(array);
    for i = 1:n-1
        for j = 1:n-i
            if array(j) > array(j+1)
                % Міняємо місцями елементи
                temp = array(j);
                array(j) = array(j+1);
                array(j+1) = temp;
            end
        end
    end
    sortedArray = array;
end

n = 1500; % Розмір масиву
array = randi([1, 1000000], 1, n); % Генерація рандомного масиву
disp('Unsorted array:');
disp(array);

sortedArray = bubbleSort(array);
disp('Sorted array:');
disp(sortedArray);
```

1.5 Модель для генерації перестановок зі складністю $O(n!)$

```
% Алгоритм з складністю O(n!) - Генерація перестановок

function permsList = generatePermutations(arr)
    if length(arr) == 1
        permsList = arr;
    else
        permsList = [];
        for i = 1:length(arr)
            % Вибір елемента і генерація перестановок для решти масиву
            remainingArr = arr([1:i-1, i+1:end]);
            subPerms = generatePermutations(remainingArr);

            % Додавання вибраного елемента до кожної з підперестановок
            permsList = [permsList; [arr(i) * ones(size(subPerms, 1), 1), subPerms]];
        end
    end
end

n = 10; % Розмір масиву
arr = randi([1, 1000], 1, n); % Генерація рандомного масиву
disp('All permutations:');
permsList = generatePermutations(arr);
disp(permsList);
```

1.6 Модель для розбиття множин зі складністю $O(n^n)$

```
% Алгоритм з складністю O(n^n) - Генерація всіх можливих комбінаторних розбиттів
множини
```

```
function allPartitions = generatePartitions(n)
    allPartitions = {};
    generatePartitionsRecursive(n, [], allPartitions);
end

function generatePartitionsRecursive(n, currentPartition, allPartitions)
    if n == 0
        allPartitions{end+1} = currentPartition; % Якщо залишок = 0, додати поточне
розбиття
    else
        for i = 1:n
            % Додавання нового підмножини до поточного розбиття
            newPartition = [currentPartition, i];
            generatePartitionsRecursive(n - i, newPartition, allPartitions);
        end
    end
end

n = 7; % Розмір множини
partitions = generatePartitions(n);
disp('Partitions:');
disp(partitions)
```

ДОДАТОК В

Лістинг коду програмної реалізації

1.1 Основні класи MATLAB-bot та result-handler мікросервісів

```
package com.lentez.diploma.config;

import org.springframework.boot.context.properties.ConfigurationProperties;

@ConfigurationProperties("bot")
public record TelegramBotProperties(String token, String username) {
}
package com.lentez.diploma.bot;

import com.lentez.diploma.service.FileUploader;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
import
org.telegram.telegrambots.extensions.bots.commandbot.TelegramLongPollingCommandBot;
import org.telegram.telegrambots.meta.api.methods.GetFile;
import org.telegram.telegrambots.meta.api.methods.send.SendPhoto;
import org.telegram.telegrambots.meta.api.objects.Document;
import org.telegram.telegrambots.meta.api.objects.InputFile;
import org.telegram.telegrambots.meta.api.objects.Message;
import org.telegram.telegrambots.meta.api.objects.Update;
import org.telegram.telegrambots.meta.exceptions.TelegramApiException;

import java.io.File;

@Component
@RequiredArgsConstructor
public class MATLABBot extends TelegramLongPollingCommandBot {

    private final String username;
    private final FileUploader fileUploader;

    private static long modelId = 0;

    @Autowired
    public MATLABBot(@Value("${bot.token}") String botToken,
                     @Value("${bot.username}") String username,
                     FileUploader fileUploader) {
       super(botToken);
       this.username = username;
        this.fileUploader = fileUploader;
    }

    @Override
    public String getBotUsername() {
```

```
        return username;
    }

    @Override
    public void processNonCommandUpdate(Update update) {
        Long chatId = update.getMessage().getChatId();
        Message message = update.getMessage();
        if (message.hasDocument()){

            Document document = message.getDocument();

            GetFile getFile = new GetFile();
            getFile.setFileId(document.getFileId());
            try {
                File file = downloadFile(execute(getFile), new
File("/Users/yaroslavholovan/IdeaProjects/bot/MATLAB-engine-wrapper/MATLAB-bot/" +
document.getFileName()));
                sendMessage(chatId, fileUploader.uploadFile(file, chatId, modelId));
                modelId++;
            } catch (TelegramApiException e) {
                e.printStackTrace();
            }
        }
    }

    public void sendMessage(Long chatId, String text) {
        try {

execute(org.telegram.telegrambots.meta.api.methods.send.SendMessage.builder()
                    .chatId(chatId.toString())
                    .text(text)
                    .build());
        } catch (TelegramApiException e) {
            e.printStackTrace();
        }
    }

    public void sendImage(Long chatId, File imageFile, String modelName) {
        try {
            if (imageFile.exists()) {
                InputFile inputFile = new InputFile(imageFile);

                SendPhoto sendPhoto = new SendPhoto();
                sendPhoto.setChatId(chatId.toString());
                sendPhoto.setPhoto(inputFile);
                sendPhoto.setCaption(imageFile.getName() + ", model: " + modelName);
                execute(sendPhoto);
            } else {
                sendMessage(chatId, "The image file does not exist.");
            }
        } catch (TelegramApiException e) {
            e.printStackTrace();
            sendMessage(chatId, "Error sending the image.");
        }
    }

}
package com.lentez.diploma.service;

import org.springframework.core.io.FileSystemResource;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
```

```
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Component;
import org.springframework.util.LinkedMultiValueMap;
import org.springframework.util.MultiValueMap;
import org.springframework.web.client.RestTemplate;

import java.io.File;

@Component
public class FileUploader {

    private final RestTemplate restTemplate;
    private final String uploadUrl = "http://localhost:8084/tasks/upload";

    public FileUploader() {
        this.restTemplate = new RestTemplate();
    }

    public String uploadFile(File file, Long senderId, long modelId) {
        try {
            MultiValueMap<String, Object> body = new LinkedMultiValueMap<>();
            body.add("file", new FileSystemResource(file));
            body.add("model_id", modelId);
            body.add("sender_id", senderId);

            HttpHeaders headers = new HttpHeaders();
            headers.setContentType(MediaType.MULTIPART_FORM_DATA);

            HttpEntity<MultiValueMap<String, Object>> requestEntity = new
HttpEntity<>(body, headers);

            ResponseEntity<String> response = restTemplate.postForEntity(uploadUrl,
requestEntity, String.class);

            return response.getBody();
        } catch (Exception e) {
            e.printStackTrace();
            return "Error uploading file: " + e.getMessage();
        }
    }
}
package com.lentez.diploma.controller;

import com.lentez.diploma.bot.MATLABBot;
import lombok.RequiredArgsConstructor;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;

import java.io.File;
import java.io.IOException;
import java.util.List;

@RestController
@RequestMapping("/bot")
@RequiredArgsConstructor
public class CommunicationController {

    private final MATLABBot MATLABBot;
```

```
    @PostMapping("/send")
    public ResponseEntity<String> sendMessage(@RequestParam("message") String
message, @RequestParam("receiver_id") Long receiverId, @RequestParam("model_id") Long
modelId) {
        String toSend = message.replaceAll("%20", " ").replaceAll("%0A", "\n");
        MATLABBot.sendMessage(receiverId, toSend + "\nModel id: " + modelId);
        return ResponseEntity.ok(message);
    }



    @PostMapping("/images")
    public ResponseEntity<String> handleFiles(
            @RequestParam("receiver_id") Long receiverId,
            @RequestParam("model_name") String modelName,
            @RequestParam("files") List<MultipartFile> files) {

        files.forEach(file -> {
            System.out.println("Received file: " + file.getOriginalFilename());
            File temp = new File("/Users/yaroslavholovan/IdeaProjects/bot/MATLAB-
engine-wrapper/MATLAB-engine/" + file.getOriginalFilename());


            try {
                file.transferTo(temp);
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
            MATLABBot.sendImage(receiverId, temp, modelName);

        });

        return ResponseEntity.ok("Files processed successfully!");
    }
}
```

1.2 Основні класи MATLAB-engine та task-processor мікросервісів

```
package com.lentez.diploma.MATLABengine.controller;



import com.lentez.diploma.MATLABengine.service.TaskProcessor;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;



@RestController
@RequestMapping("/tasks")
public class TaskController {



    private final TaskProcessor taskProcessor;
```

```
    public TaskController(TaskProcessor taskProcessor) {
        this.taskProcessor = taskProcessor;
    }

    @PostMapping("/upload")
    public ResponseEntity<String> handleFileUpload(@RequestParam("file")
MultipartFile file, @RequestParam("model_id") Long modelId,
@RequestParam("sender_id") Long senderId) {
        if (file.isEmpty()) {
            return ResponseEntity.badRequest().body("File is empty!");
        }
        String savedFilePath = taskProcessor.processFile(file, senderId, modelId);
        return ResponseEntity.ok("File received and saved.\nModel id: " + modelId +
"\nRunning model: " + file.getOriginalFilename());
    }
}
```

```
package com.lentez.diploma.MATLABengine.service;

import org.springframework.core.io.FileSystemResource;
import org.springframework.http.*;
import org.springframework.stereotype.Component;
import org.springframework.util.LinkedMultiValueMap;
import org.springframework.util.MultiValueMap;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.util.UriComponentsBuilder;



import java.io.File;


@Component
public class BotMessageSender {

    private static final String BOT_API_URL = "http://localhost:8085/bot";
    public String sendMessage(Long receiverId, String message, Long modelId) {
        RestTemplate restTemplate = new RestTemplate();


        UriComponentsBuilder builder = UriComponentsBuilder.fromHttpUrl(BOT_API_URL +
"/send")

                .queryParam("message", message)
                .queryParam("receiver_id", receiverId)
                .queryParam("model_id", modelId);

        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);

        HttpEntity<String> entity = new HttpEntity<>(headers);

        ResponseEntity<String> response = restTemplate.exchange(
                builder.toUriString(), HttpMethod.POST, entity, String.class);

        return response.getBody();
    }
```

```
    public String sendImages(Long receiverId, File[] images, String modelName) {
        RestTemplate restTemplate = new RestTemplate();

        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.MULTIPART_FORM_DATA);

        MultiValueMap<String, Object> body = new LinkedMultiValueMap<>();
        body.add("receiver_id", receiverId);
        body.add("model_name", modelName);

        for (File image : images) {
            body.add("files", new FileSystemResource(image));
        }

        HttpEntity<MultiValueMap<String, Object>> requestEntity = new
HttpEntity<>(body, headers);

        ResponseEntity<String> response = restTemplate.exchange(
                BOT_API_URL + "/images", HttpMethod.POST, requestEntity,
String.class);

        return response.getBody();
    }


}


package com.lentez.diploma.MATLABengine.service;



import org.springframework.stereotype.Component;
import org.springframework.web.multipart.MultipartFile;


import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;


@Component
public class TaskProcessor {

    private final MATLABModelProcessor MATLABModelProcessor;

    public TaskProcessor(MATLABModelProcessor MATLABModelProcessor) {
        this.MATLABModelProcessor = MATLABModelProcessor;
    }

    public String processFile(MultipartFile file, Long senderId, Long modelId) {
        try {
            String tempDir = System.getProperty("java.io.tmpdir");
            String filePath = tempDir + "/" + file.getOriginalFilename();
            Files.write(Paths.get(filePath), file.getBytes());

            System.out.println("File saved: " + filePath);

            MATLABModelProcessor.submitModel(filePath, senderId, modelId,
file.getOriginalFilename());

            return filePath;
```

```
        } catch (IOException e) {
            throw new RuntimeException("Failed to process file", e);
        }
    }
}
package com.lentez.diploma.MATLABengine.service;

import com.mathworks.engine.MATLABEngine;
import org.springframework.stereotype.Component;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.concurrent.*;
import java.util.concurrent.atomic.AtomicInteger;



@Component
public class MATLABModelProcessor {
    private final ExecutorService executor;
    private final AtomicInteger activeThreads = new AtomicInteger(0);
    private final BotMessageSender botMessageSender;
    public MATLABModelProcessor(BotMessageSender botMessageSender) {
        this.botMessageSender = botMessageSender;
        int threadPoolSize = 4;
        this.executor = Executors.newFixedThreadPool(threadPoolSize);
        System.out.println("MATLABModelProcessor initialized with " + threadPoolSize
+ " threads.");
    }


    public void submitModel(String modelPath, Long senderId, Long modelId, String
modelName) {
        executor.submit(() -> {
            long startTime = System.currentTimeMillis();
            MATLABEngine MATLABEngine = null;
            String result = "-------------\nRESULT for model " + modelName + "\n";
            try {
                System.out.println("Starting MATLAB Engine for: " + modelPath);
                MATLABEngine = MATLABEngine.startMATLAB();
                activeThreads.incrementAndGet();
                System.out.println("Running MATLAB script: " + modelPath);
                MATLABEngine.feval("evalc", "disp('MATLAB is running...');");
                result += (String) MATLABEngine.feval("evalc", "run('" + modelPath +
"');");
                String outputDir = "/Users/yaroslavholovan/IdeaProjects/bot/MATLAB-
engine-wrapper/MATLAB-engine/figures";
                String MATLABCode = """
                        figHandles = findall(0, 'Type', 'figure');
                        if ~isempty(figHandles)
                            for i = 1:length(figHandles)
                                saveas(figHandles(i), fullfile('%s', ['figure_'
num2str(i) '.png']));
                            end
                        end
                        """.formatted(outputDir);
                MATLABEngine.feval("evalc", MATLABCode);
                File tempDir = new File(outputDir);
                File[] files = tempDir.listFiles((dir, name) ->
name.startsWith("figure_") && name.endsWith(".png"));
                if (files != null && files.length > 0) {
```

```
                    botMessageSender.sendImages(senderId, files, modelName);
                    for (File file : files) {
                        System.out.println("Generated figure: " +
file.getAbsolutePath());

                        byte[] imageBytes = Files.readAllBytes(file.toPath());
                    }
                    cleanDirectory(tempDir);
                }

                long endTime = System.currentTimeMillis();
                System.out.println("Model processed in " + (endTime - startTime) + "
ms");
                result += "\nModel processed in " + (endTime - startTime) + " ms";
                botMessageSender.sendMessage(senderId, result, modelId);
                System.out.println("MATLAB Result:\n" + result);
            } catch (Exception e) {
                System.err.println("Error during MATLAB execution: " +
e.getMessage());
                e.printStackTrace();
            } finally {
                if (MATLABEngine != null) {
                    try {
                        MATLABEngine.close();
                        System.out.println("MATLAB Engine closed for: " + modelPath);
                    } catch (Exception e) {
                        System.err.println("Error closing MATLAB Engine: " +
e.getMessage());
                    }
                }
                activeThreads.decrementAndGet();
            }
        });
    }

    public void shutdown() {
        try {
            executor.shutdown();
            if (!executor.awaitTermination(60, TimeUnit.SECONDS)) {
                executor.shutdownNow();
            }
            System.out.println("Executor shut down.");
        } catch (InterruptedException e) {
            executor.shutdownNow();
            System.err.println("Error during shutdown: " + e.getMessage());
        }
    }


    public int getActiveThreads() {
        return activeThreads.get();
    }
    public void cleanDirectory(File directory) {
        if (directory.exists() && directory.isDirectory()) {
            File[] files = directory.listFiles();
            if (files != null) {
                for (File file : files) {
                    if (file.isFile()) {
                        if (file.delete()) {
                            System.out.println("Deleted file: " +
file.getAbsolutePath());
                        } else {
                            System.err.println("Failed to delete file: " +
```

```
file.getAbsolutePath());
                            }
                        }

                    }

                }

            } else {
                System.err.println("The directory does not exist or is not a directory: "
+ directory.getAbsolutePath());
            }
        }
}
```