

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”**

Інститут прикладного системного аналізу

Кафедра Системного проектування

"На правах рукопису"
УДК _____

До захисту допущено:
Завідувач кафедри

(підпис) (ініціали, прізвище)
“ ” _____ 20__ р

**Магістерська дисертація
на здобуття ступеня магістра
за освітньо-професійною програмою «Інтелектуальні сервіс-орієнтовані
розподілені обчислювання»
зі спеціальності 122 «Комп'ютерні науки»
на тему: «Дослідження алгоритму Spike для вирішення великих систем
рівнянь з матрицями стрічкового типу»**

Виконав (-ла): студент (-ка) VI курсу, групи ДА-91мп,

Корчагін Іван Олександрович _____

Керівник: к.т.н., Булах Б. В. _____

Консультант Розробка стартап-проекту к.т.н., Булах Б. В. _____

Рецензент: д.т.н., проф. Данилов В. Я. _____

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

Студент (-ка) _____

Київ – 2020 року

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ А.І.Петренко

(підпис) (ініціали, прізвище)

« 07 » Липня 2020 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

_____ Корчагін Іван Олександрович _____

(прізвище, ім'я, по батькові)

1. Тема дисертації: Використання алгоритму Spike для вирішення систем лінійних рівнянь з матрицями стрічкового типу

науковий керівник дисертації Булах Богдан Вікторович, к.т.н.,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « 02 » листопада 2020 р. № 3182-с

2. Строк подання студентом дисертації: **15 грудня, звіт з практики – 25 жовтня**

3. Об'єкт дослідження: Метод Spike та його модифікації.

4. Предмет дослідження (Вихідні дані – для магістерської дисертації за освітньо-професійною програмою): Програмна реалізація алгоритму Spike для вирішення СЛАР

5. Перелік завдань, які потрібно розробити

1. Дослідження Spike алгоритму

2. Дослідження стандартів розпаралелювання програм

3. Огляд бібліотек паралельного вирішення СЛАР

4. Реалізація та аналіз Spike алгоритму

5. Розробка стартап проекту Spike алгоритм

6. Перелік графічного (ілюстративного) матеріалу:

1.
ки роботи алгоритму

Графі

7. Орієнтовний перелік публікацій _____

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розробка стартап-проекту	Булах Б. В.		

9. Дата видачі завдання 7 липня 2020р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	7 липня 2020	
2	Ознайомлення з тех. літературою	10 вересня 2020	
3	Дослідження Spike алгоритму	15 вересня 2020	
4	Дослідження стандартів розпаралелювання програм	20 вересня 2020	
5	Огляд бібліотек паралельного вирішення СЛАР	1 жовтня 2020	
6	Реалізація та аналіз Spike алгоритму	15 жовтня 2020	
7	Розробка стартап проекту Spike алгоритм	25 жовтня 2020	
8	Оформлення дипломної роботи	15 грудня 2020	

Студент

(підпис)

І. О. Корчагін
(ініціали, прізвище)

Науковий керівник дисертації

(підпис)

Б. В. Булах
(ініціали, прізвище)

АННОТАЦІЯ

РЕФЕРАТ НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ

виконану на тему: «Дослідження алгоритму Spike для вирішення великих систем рівнянь з матрицями стрічкового типу»
студентом: Корчагіним Іваном Олександровичем

Робота виконана на 76 сторінках, містить 17 ілюстрацій, 24 таблиці. При підготовці використовувалась література з 26 джерел.

У даній роботі проводиться аналіз алгоритму для вирішення СЛАР, що містить смугову матрицю, а також технологій для розробки алгоритму. Окремо розглянуті готові рішення на ринку для вирішення даного типу рівнянь.

У ході роботи був виявлений оптимальний стек для розробки алгоритму проведена його безпосередня реалізація. Ефективність реалізації була перевірена під час порівняння алгоритму з вже реалізованими бібліотечними функціями. Підтверджена ефективність роботи алгоритму (як по часу виконання так і по споживаній пам'яті) для розріджених матриць с шириною смуги <12% від ширини матриці. Виявлені переваги та не недоліки алгоритму та запропоновані шляхи розвитку та поліпшення роботи алгоритму.

Ключові слова: SPIKE, СЛАР, смугова матриця, розріджені системи, паралельні алгоритми.

АННОТАЦИЯ

РЕФЕРАТ ДЛЯ МАГИСТЕРСКОЙ ДИССЕРТАЦИИ

выполненной на тему: "Исследование алгоритма SPIKE для решения больших систем уравнений с матрицами ленточного типа"

студент: Корчагин Иван Александрович

Работа выполнена на 76 страницах, содержит 17 иллюстраций, 24 таблицы. При подготовке использовалась литература из 26 источников.

В этой работе анализируется алгоритм решения СЛАУ, содержащий ленточную матрицу, а также технологии разработки алгоритма. Отдельно рассмотрены готовые инструменты на рынке для решения такого рода уравнений.

В ходе работы был обнаружен оптимальный стек для разработки алгоритма. Эффективность реализации была проверена при сравнении алгоритма с уже реализованными библиотечными функциями. Подтверждена эффективность алгоритма (как по времени выполнения, так и по затрачиваемой памяти) для разреженных матриц с шириной полосы 12% ширины матрицы. Найдены преимущества и недостатки алгоритма и предлагаемые способы развития и совершенствования алгоритма.

Ключевые слова: SPIKE, СЛАУ, ленточные матрицы, разреженные системы, параллельные алгоритмы.

ANNOTATION

ABSTRACT FOR THE MASTER DISSERTATION

performed on the topic: " Investigation of the Spike algorithm for solving big systems of equations with band matrices "

student: Korchagin Ivan Alexandrovich

The work is done on 76 pages, contains 17 illustrations, 24 tables. In preparation, literature from 26 sources was used.

In this work, an algorithm for solving a SOLE containing a band matrix is analyzed, as well as technologies for developing an algorithm. The ready-made tools on the market for solving such equations are considered separately.

In the course of the work, the optimal stack for the development of the algorithm was found. The efficiency of the implementation was verified by comparing the algorithm with library functions. The efficiency of the algorithm (both in terms of execution time and required memory) for sparse matrices with a bandwidth of 12% of the matrix width has been confirmed. The advantages and disadvantages of the algorithm and the proposed ways of further developing and improving the algorithm are found.

Keywords: SPIKE, SOLE, banded matrices, sparse systems, parallel algorithms

ЗМІСТ

ВСТУП.....	11
1. ДОСЛІДЖЕННЯ SPIKE АЛГОРИТУ.....	13
1.1 Класичний алгоритм.....	13
1.1.1 Варіації SPIKE алгоритму.....	19
1.2 ТА-SPIKE алгоритм.....	20
1.2.1 Усічений SPIKE алгоритм.....	20
1.2.2 LU/UL стратегія.....	21
1.2.3 Розбиття на частини.....	22
1.3 Висновки.....	26
2. ДОСЛІДЖЕННЯ СТАНДАРТІВ РОЗПАРАЛЕЛЮВАННЯ ПРОГРАМ...	26
2.1 OpenMP.....	26
2.1.1 Паралелізація SMP.....	27
2.1.2 Робота потоків.....	27
2.1.3 Паралелізм SMP та потоки OpenMP.....	28
2.1.4 Змінне масштабування.....	28
2.2 MPI.....	29
2.2.1 Використання MPI.....	29
2.3 OpenMP проти MPI.....	30
2.4 Висновки.....	31
3. ОГЛЯД БІБЛІОТЕК ПАРАЛЕЛЬНОГО ВИРІШЕННЯ СЛАР.....	32
3.1 Intel MKL.....	32
3.1.1 Ефективність.....	32
3.1.2 Функціональні категорії.....	33

3.2	Sun Performance Library.....	35
3.2.1	Модулі Sun Performance.....	36
3.2.2	Математичні функції.....	36
3.3	Eigen.....	37
3.4	GNU Scientific Library.....	38
3.5	Висновки.....	39
4.	РЕАЛІЗАЦІЯ ТА АНАЛІЗ SPIKE АЛГОРИТМУ.....	40
4.1	Деталі реалізації.....	40
4.1.1	Numpy.....	40
4.1.2	SkiPy.....	42
4.1.3	Зберігання матриці.....	43
4.2	Аналіз роботи алгоритму.....	43
4.2.1	Порівняння швидкості роботи.....	44
4.2.2	Аналіз ефективного розбиття системи.....	48
4.2.3	Порівняння рівнів споживання пам'яті.....	51
4.3	Висновки.....	53
5.	РОЗРОБКА СТАРТАП-ПРОЕКТУ «SPIKE АЛГОРИТМ».....	54
5.1	Опис ідеї проекту.....	54
5.2	Технологічний аудит ідеї проекту.....	56
5.3	Аналіз ринкових можливостей.....	57
5.4	Розробка ринкової стратегії проекту.....	65
4.6	Висновки.....	70
	ВИСНОВКИ.....	72
	СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	75

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

СЛАР – Система лінійних арифметичних рівнянь

CPU (Central processing unit) – Центральний процесор

CSR (Compressed Sparse Row) – Формат зберігання розріджених матриць

ВСТУП

Лінійні системи виникають у різних сферах науки та техніки, таких як обчислювальна механіка (рідини, структури та взаємодії рідин-структур), обчислювальна наноелектроніка (квантово-механічне моделювання) і навіть у галузі фінансової математики (геометричний броунівський рух). Деякі програми спеціалізованих типів, такі як "транспортні проблеми" в нанопровідниках і нанотрубках, та "проблеми потоку руху" за своєю суттю вимагають вирішення смугових систем. Більшість застосувань часто породжують дуже великі розріджені лінійні системи, які можуть бути переупорядковані для створення або вузькосмугових систем, або низькорангових збурень вузькосмугових систем, причому системи можуть бути або щільними, або розрідженими всередині смуги. Наприклад, при аналізі кінцевих елементів основні розріджені лінійні системи можуть бути перевпорядковані, щоб отримати смугову систему, в якій ширина смуги складає лише невелику частку від розміру загальної матриці. У свою чергу, специфічні методи можуть бути використані для подальшого зменшення смуги та створення надійного передобробника для ітеративного вирішувача.

Традиційні числові алгоритми та бібліотечні пакети, такі як LAPACK, все ще стикаються з проблемами при вирішенні масштабних проблем моделювання для все більш високого рівня ефективності, точності та масштабованості в сучасних паралельних архітектурах. Таким чином, розробка надійних смугових паралельних вирішувачів, які ефективні як для паралельних архітектур високого класу, так і для недорогих кластерів, має велике значення. З точки зору алгоритміки та реалізації для розробки масштабованого, швидкого та надійного смугового вирішувача, багато факторів вступають у гру; зокрема, складність системи, локалізація даних, арифметичні міркування (реальна проти складної, одинарна проти подвійної точність), архітектурні міркування: - спільна пам'ять (OpenMP), розподілена пам'ять (MPI), графічний процесор,

хмара та різні інші високі -кінцеві обчислювальні платформи. Хоча паралельно розподілені числові пакети дійсно пропонують користувачам НРС цінні інструменти для вирішення великомасштабних проблем, зростаючий розмір кількості ядер в обчислювальних вузлах попереджує ефективність моделі розподіленого програмування (тобто MPI) для багатьох користувачів.

1. ДОСЛІДЖЕННЯ SPIKE АЛГОРИТУ

Алгоритм SPIKE базується на парадигмі дизайну "розділяй і володарюй" і включає наступні кроки:

(a) Попередня обробка:

- (i) розділення вихідної системи на різні процесори або SMP
- (ii) розкладання на множини кожного діагонального блоку та вилучення усіченої системи значно меншого розміру;

(b) Постобробка:

- (iii) вирішення скороченої системи
- (iv) отримання загального рішення. SPIKE має кілька опцій використання, які варіюються від використання його як прямого вирішувача до використання для підготовки даних для ітераційних алгоритмів.

Основною перевагою дизайну алгоритму SPIKE "розділяй і володарюй" є те, що він робить його природним чином пристосованим для виконання в багатопроцесорних машинах, оскільки різні підзадачі можуть виконуватися на різних процесорах. Як додаток, особливо для систем спільної пам'яті, передачу даних між процесорами не потрібно планувати заздалегідь. Крім того, таке проектування зазвичай ефективно використовує кеш пам'яті, тим самим надаючи алгоритму додатковий паралельний вимір.

1.1 Класичний алгоритм

Алгоритм SPIKE призначений для вирішення смугових систем на паралельній машині. Основну ідею ввели Самех і Кук [7], які розглядали тридіагональний випадок, і Чен, Кук і Самех [3], які вивчали трикутний випадок. Лоурі [12] та Самех застосували алгоритм до симетричних позитивно визначених систем, тоді як Донгарра та Самех [10] розглядали суворо діагонально домінуючий випадок. Варіації алгоритмів SPIKE для

тридіагональних систем були введені Sun, Zhang і Ni [24], які також проаналізували похибку усічення тридіагональних систем, які рівномірно домінують по діагоналі. Помилка зрізування для тридіагональних систем Тепліца, які також є суворо діагонально домінантними, а також симетричними або перекошеними симетричними, розглядав Сонце [25]. Іншу варіацію алгоритму SPIKE для строго діагонально домінуючих систем вивчали Ларріба-Пей, Йорба та Наварро [11]. Polizzi та Sameh [17] розширили алгоритми SPIKE на загальносмуговий випадок і розробили пакет SPIKE.

Матриця $A = [a_{ij}]$ діагонально домінантна по рядках якщо:

$$\sum_{i \neq j} |a_{ij}| \leq |a_{ii}| \quad \#(1.1)$$

для всіх i . Якщо нерівність чітка, то A строго діагонально домінантна по рядкам.

Якщо A не сингулярна і діагонально домінантна по рядках, то діагональні елементи не є нульовими, і коефіцієнт домінування визначається наступним чином:

$$e = \max \frac{\sum_{i \neq j} |a_{ij}|}{|a_{ii}|} \quad \#(1.2)$$

Якщо $e > 0$, то ступінь домінування діагоналі d визначається,

$$d = e^{-1} \quad \#(1.3)$$

Ступінь домінування по діагоналі є центральним для аналізу усіченого алгоритму SPIKE, який буде докладно пояснено далі. Для ілюстрації основних концепцій та роботи алгоритму SPIKE ми розглянемо неосингулярну лінійну систему,

$$Ax = f \quad \#(1.4)$$

де $A \in n \times n$ смуговою матрицею, а кількість наддіагоналей k приймається рівною кількості субдіагоналей, а сама матриця вузькосмугова, тобто $k \ll n$. Нехай p позначає кількість процесорів. Для простоти передбачається, що p ділить n націло. Нехай система буде розділена на діагональну форму блоку, показану нижче,

$$Ax = \begin{pmatrix} A_1 & B_1 & & & \\ C_2 & A_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & B_{p-1} \\ & & & C_p & A_p \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_p \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_p \end{pmatrix} \quad \#(1.5)$$

де $A_i, i = 1, 2, \dots, p$ смугова матриця порядку $\mu = n/p$ та з шириною смуги $2k+1$, B_i і C_i матриці розміру $\mu \times k$.

$$B_i = \begin{pmatrix} 0 & 0 \\ b_i & 0 \end{pmatrix}, \quad C_{i+1} = \begin{pmatrix} 0 & c_{i+1} \\ 0 & 0 \end{pmatrix}, \quad i = 1, 2, \dots, p-1 \quad \#(1.6)$$

де b_i і $c_i + 1$ трикутні матриці розміру $k \times k$. Нехай D позначає головний діагональний блок.

$$D = \text{diag}\{A_1, A_2, \dots, A_p\} \quad \#(1.7)$$

Якщо обидві сторони (1.5) попередньо помножити на D^{-1} , то ми отримаємо зменшену систему виду $Sx = g$ як показано нижче.

$$\begin{pmatrix} I_\mu & V_1 & & & \\ W_2 & I_\mu & V_2 & & \\ & \ddots & \ddots & \ddots & \\ & & W_{p-1} & I_\mu & V_{p-1} \\ & & & W_p & I_\mu \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{p-1} \\ x_p \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_{p-1} \\ g_p \end{pmatrix} \quad \#(1.8)$$

Де V_i позначає правий спайк а W_i – лівий спайк зі стовпцями отриманими з

$$V_i = A_i^{-1} \begin{pmatrix} 0 \\ B_i \end{pmatrix}, i W_i = A_i^{-1} \begin{pmatrix} C_i \\ 0 \end{pmatrix} \quad \#(1.9)$$

Та V_i та W_i поділені як

$$V_i = \begin{pmatrix} V_i^t \\ V_i^m \\ V_i^b \end{pmatrix}, i W_i = \begin{pmatrix} W_i^t \\ W_i^m \\ W_i^b \end{pmatrix} \quad \#(1.10)$$

Де V_i^t, V_i^b, W_i^t і W_i^b частини спайку з розмірами $k \times k$. Верхні індекси t, m і b це аббревіатури до слів top, middle і bottom відповідно. Також x_i і g_i розділені таким же чином

$$x_i = \begin{pmatrix} x_i^t \\ x_i^m \\ x_i^b \end{pmatrix}, i g_i = \begin{pmatrix} g_i^t \\ g_i^m \\ g_i^b \end{pmatrix} \quad \#(1.11)$$

Система вище для $p = 3$ продемонстрована далі,

$$\begin{pmatrix}
I_k & & & & & & & & \\
& I_v & & & & & & & \\
& & I_k & & & & & & \\
& & & W_2^{(t)} & & & & & \\
& & & W_2^{(m)} & & I_v & & & \\
& & & W_2^{(b)} & & & I_k & & \\
& & & & & & & W_3^{(t)} & \\
& & & & & & & W_3^{(m)} & \\
& & & & & & & & I_v \\
& & & & & & & & & I_k
\end{pmatrix}
\begin{pmatrix}
x_1^{(t)} \\
x_1^{(m)} \\
x_1^{(b)} \\
x_2^{(t)} \\
x_2^{(m)} \\
x_2^{(b)} \\
x_3^{(t)} \\
x_3^{(m)} \\
x_3^{(b)}
\end{pmatrix}
=
\begin{pmatrix}
f_1^{(t)} \\
f_1^{(m)} \\
f_1^{(b)} \\
f_2^{(t)} \\
f_2^{(m)} \\
f_2^{(b)} \\
f_3^{(t)} \\
f_3^{(m)} \\
f_3^{(b)}
\end{pmatrix} \quad \#(1.12)$$

Де $v = \mu = 2k$.

Саме від цих вузьких блокових колон або шипів, що виходять від основної діагоналі, алгоритм отримав свою назву. Матриця S називається матрицею SPIKE.

Зауважимо, що об'єднання k рівнянь вище та k рівнянь нижче $p - 1$ розділових ліній утворює незалежну підсистему порядку $2k(p - 1)$. Така система називається "скороченою" системою і записується як $Rx_r = g_r$. Вона має вигляд

$$Ax = \begin{pmatrix} E_1 & F_1 & & & \\ G_2 & E_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & F_{p-2} \\ & & & G_{p-1} & E_{p-1} \end{pmatrix} \begin{pmatrix} x_{r,1} \\ x_{r,2} \\ \vdots \\ x_{r,p-1} \end{pmatrix} = \begin{pmatrix} g_{r,1} \\ g_{r,2} \\ \vdots \\ g_{r,p-1} \end{pmatrix} \quad \#(1.13)$$

де,

$$E_i = \begin{pmatrix} I_k & V_i^b \\ W_{i+1}^t & I_k \end{pmatrix}, F_i = \begin{pmatrix} 0 & 0 \\ W_{i+1}^t & 0 \end{pmatrix}, F_i = \begin{pmatrix} 0 & W_i^b \\ 0 & 0 \end{pmatrix} \quad \#(1.14)$$

і

$$x_{r,i} = \begin{pmatrix} x_i^b \\ x_{i+1}^t \end{pmatrix}, g_{r,i} = \begin{pmatrix} g_i^b \\ g_{i+1}^t \end{pmatrix} \quad \#(1.15)$$

Індекс r є абрєвіатурою слова reduced і $i = 1, \dots, p-1$

Зменшена система буде використовувється для отримання часткового рішення, і оскільки блоки системи в основному незалежні, пошук може здійснюватися в ідеальному паралелізмі, використовуючи:

$$x_1 = g_1 - V_1 x_2^t \quad \#(1.16)$$

$$x_i = g_i - W_i x_{i-1}^b - V_i x_{i+1}^t, \quad i = 2, \dots, p-1 \quad \#(1.17)$$

$$x_p = g_p - W_p x_{p-1}^b \quad \#(1.18)$$

Слід також зазначити, що x_0 , x_{p+1} , W_1 і V_p не визначені і вважаються нульовими. Якщо обчислення проводяться з використанням точної

арифметики, то наведений вище вираз дає точне рішення

$$Ax = f.$$

Таким чином, підсумовуючи, алгоритм SPIKE можна розбити на такі етапи:

1. Розкласти на множники незалежно діагональні блоки A . Це набагато ефективніше, ніж розкладання A на пряму.
2. Обчисліть шипи, використовуючи факторизацію, отриману на попередньому кроці, і обчислити модифіковану праву сторону.
3. Сформувати та розв'язати скорочену систему, щоб отримати часткові рішення вище та нижче кожного розділу.
4. Отримайте решту рішення x .

1.1.1 Варіації SPIKE алгоритму

Оригінальний алгоритм SPIKE, пояснений вище, має безліч варіантів.

Ці варіації націлені на системи рівнянь з певними властивостями для зменшення обсягу виконуваних обчислень при вирішенні. Вони також збільшують здатність до паралелізму, доступного на різних етапах алгоритму.

SPIKE представляє різні варіанти обчислень залежно від властивостей і типу матриці та архітектури платформи.

1. SPIKES можна обчислити:

- a) Аналітично
- b) Приблизно

2. Діагональні блоки можна вирішити:

- a) Аналітично (PLU, LU, UL тощо)
- b) Ітеративно

3. Скорочену систему можна вирішити

- a) Аналітично (рекурсивний спайк)
- b) Ітеративно разом із схемою попереднього кондиціонування
- c) Приблизно (усічений SPIKE)

1.2 TA-SPIKE алгоритм

Алгоритм TA-SPIKE, описаний у [19], поєднує в собі як переваги стратегії LU/UL усіченої схеми SPIKE, так і нову нетрадиційну схему розподілу для досягнення ефективного балансування навантаження між процесорами / ядрами.

1.2.1 Усічений SPIKE алгоритм

Урізана схема SPIKE, представлена в [16], є оптимізованою версією алгоритму SPIKE з розширеним використанням паралелізму для обробки діагонально домінуючих систем. Ці системи можуть виникати з декількох наукових та технічних застосувань і визначаються, якщо ступінь домінування по діагоналі d матриці A більша за одиницю, де d , як зазначено раніше, рахується:

$$d = \min_i \frac{|A_{i,i}|}{\sum_{j \neq i} |A_{i,j}|} \quad \#(1.20)$$

Якщо ця властивість задовольняється, можна побачити, що величини елементів правих шипів V_j зменшуються знизу вгору, тоді як елементи лівих шипів W_j зменшуються за величиною зверху вниз [21, 22]. Оскільки розмір діагональних блоків A_j передбачається набагато більшим, ніж розмір m блоків B_j і C_j , нижні блоки лівих шипів W_j^b і верхні блоки правих шипів V_j^t будуть приблизно дорівнювати нулю. Звідси випливає, що отримана «усічена» скорочена система - це просто діагональ блоку, складена з $p - 1$ незалежних блокових систем $2m \times 2m$ (p представляє кількість розділів) наступної форми:

$$\begin{bmatrix} I_m & V_j^b \\ W_{j+1}^t & I_m \end{bmatrix} \begin{bmatrix} X_j^b \\ X_{j+1}^t \end{bmatrix} = \begin{bmatrix} G_j^b \\ G_{j+1}^t \end{bmatrix}, j = 1, \dots, p - 1 \quad (1.21)$$

де X_j і G_j позначають j й розділ рішення X та модифіковану праву сторону рівняння G .

Тому блок G_j^b (відповідно X_j^b) пов'язаний з нижньою верхівкою G_j (відповідно X_j), тоді як блок G_{j+1}^t (відповідно X_{j+1}^t) пов'язаний з верхньою верхівкою G_{j+1} (відповідно X_{j+1}). Потім редуковані лінійні системи роз'єднуються і можуть бути вирішені паралельно.

У рамках усіченої схеми також було запропоновано два інших основні внески для покращення обчислювальної продуктивності та масштабованості факторизації та етапів вирішення:

- а) стратегія LU/UL, та
- б) нова нетрадиційна схема розділення.

1.2.2 LU/UL стратегія

Стратегію LU/UL можна використовувати, щоб уникнути обчислення (генерування) цілих спайків для отримання V_j^b і W_{j+1}^t ($j = 1, \dots, p - 1$). Час вирішення для отримання нижнього спайку V_j^b можна різко зменшити,

використовуючи факторизацію LU на кожному діагональному блоці $A_j = 1, \dots, p-1$. У свою чергу, факторизація UL на кожному діагональному блоці $A_j = 2, \dots, p$ може бути використаний для отримання спайків W_j^t , що включають лише верхні $m \times m$ блоки нових матриць U та L.

Нарешті, слід зазначити, що у випадку двох розділів можна скористатися однією факторизацією LU або UL без повороту відповідно для A_1 і A_2 . Потім (точну) результуючу систему з одним блоком можна отримати та вирішити з мінімальними зусиллями. Як результат, кількість арифметичних операцій SPIKE з використанням стратегії LU/UL для випадку двох розділів і роботи на двох процесорах / ядрах, по суті, ділиться на дві порівняно з послідовним алгоритмом на LAPACK на етапах розкладання та вирішення.

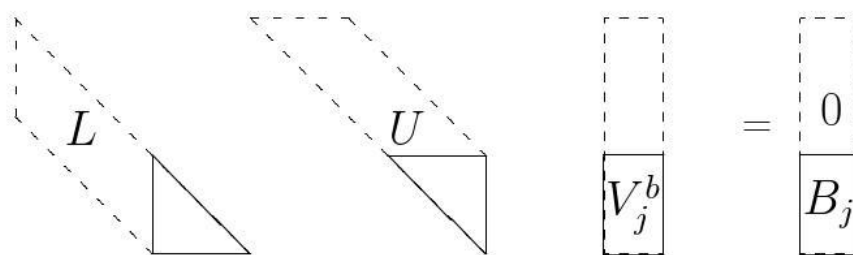


Рисунок 1.1 – Нижня частина спайку V_j може бути порохована використовуючи тільки нижні $m \times m$ блоки L та U . Таким же чином верхня частина спайку W_j може бути отримана за допомогою LU розкладу.

1.2.3 Розбиття на частини

Найчастіше паралельні алгоритми, спрямовані на досягнення лінійної масштабованості для великої кількості процесорів/ядер, мають великі етапи попередньої обробки із частим посиланням на пам'ять та/або складними обчисленнями, що призводить до погіршення продуктивності на невеликій кількості процесорів/ядер. Наприклад, використовуючи описану вище стратегію LU/UL із кількістю розділів, більшою за два, кожен середній розділ $j = 2, \dots, p-1$ тепер повинен виконувати розклад як LU, так і UL. Для того,

щоб зменшити кількість арифметичних операцій, тут розглядається новий паралельний розподіл системної матриці, який передбачає використання меншої кількості розділів, ніж кількість процесорів/ядер k (тобто $p < k$). На практиці нова кількість розділів буде дорівнювати $p = (k + 2)/2$ (де k - парне число процесорів/ядер). Рисунки 1.2 і 1.3 ілюструють новий розподіл правої сторони матриці та рішення відповідно для випадків $k = 4; p = 3$ і $k = 8; p = 5$. У рамках цього нового розділення нові матриці блоків $A_j, j = 1, \dots, p$, пов'язані з першими p процесорами / ядрами, тоді як решта з них, переходячи від $p + 1$ до k , містять ще одну копію блокової матриці $A_j, j = 2, \dots, p - 1$.

$$\begin{array}{c}
 \mathbf{A} = \left[\begin{array}{c|c|c} \begin{array}{c} A_1 \\ B_1 \end{array} & & \\ \hline \begin{array}{c} C_2 \\ A_2 \\ B_2 \end{array} & & \\ \hline \begin{array}{c} C_3 \\ A_3 \end{array} & & \end{array} \right] & \mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} & \mathbf{F} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} \\
 (1) & (2, 4) & (3)
 \end{array}$$

Рисунок 1.2. - Ілюстрація нетрадиційного розділення лінійної системи для алгоритму TA-SPIKE у випадку 4 процесорів/ядер. A_1 надсилається процесору 1, A_2 - процесору 2 і 4, а A_3 - процесору 3.

$$\begin{array}{c}
 \mathbf{A} = \left[\begin{array}{c|c|c|c} \begin{array}{c} A_1 \\ B_1 \end{array} & & & \\ \hline \begin{array}{c} C_2 \\ A_2 \\ B_2 \end{array} & & & \\ \hline \begin{array}{c} C_3 \\ A_3 \\ B_3 \end{array} & & & \\ \hline \begin{array}{c} C_4 \\ A_4 \\ B_4 \end{array} & & & \\ \hline \begin{array}{c} C_5 \\ A_5 \end{array} & & & \end{array} \right] & \mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \end{bmatrix} & \mathbf{F} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \end{bmatrix} \\
 (1) & (2, 6) & (3, 7) & (4, 8) & (5)
 \end{array}$$

Рисунок 1.3. - Ілюстрація нетрадиційного розділення лінійної системи для усіченого алгоритму SPIKE у випадку 8 процесорів/ядер. A_1 надсилається процесору 1, A_5 - процесору 5. A_i для $i = 2, \dots, 4$ до процесора i та $i + 4$.

Як описано вище із використанням стратегії LU/UL, тут можна отримати V_j^b ($j = 1, \dots, p - 1$) з мінімальними обчислювальними зусиллями через крок вирішення LU на процесорах/ядрах 1 до $p - 1$, тоді як W_j^t ($j = 2, \dots, p$) можна отримати подібним чином, виконуючи кроки розв'язання UL на процесорах/ядрах p до k . У прикладі $k = 4$ ми зараз виконуємо незалежно розкладання на множники $L_j U_j \leftarrow A_j$ для розділів $j = 1, 2$, A_j для розділів $j = 2, 3$. Використовуючи цю нову схему розділення, розмір розділів дійсно збільшується, але кількість арифметичних операцій за розділом зменшується разом із розміром усіченої скороченої системи. Порівняно з послідовним алгоритмом LU , пришвидшення етапу розкладання факторів схеми TA-SPIKE тоді ідеально дорівнює новій кількості розділів, тобто 2 на двох процесорах / ядрах, 3 на чотирьох, 5 на восьми, тощо.

Нарешті, слід зазначити, що така ж швидкість роботи очікується і на етапі вирішення. Дійсно можна розрахувати фінальне рішення, використовуючи, по суті, лише одну розгортку вперед і одну розгортку назад за розділами (витрати на комунікації та вирішення усіченої скороченої системи вважаються мінімальними). Малюнок 1.4 ілюструє, зокрема, що G_j^b $j = 1, \dots, p - 1$ блоків, що з'являється у скороченій системі (1.2), може бути сформовано переважно за допомогою лише одного розгортання розкладу LU в розділі j .

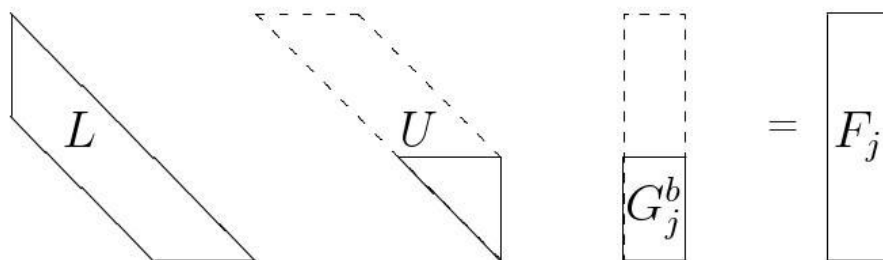


Рисунок 1.4. – Нижню частину модифікованої правої сторони G_j можна обчислити, використовуючи всю нижню частину розгортки на L , і дуже малу частку розгортки назад на U .

Аналогічно, генерація G_{j+1}^t $j = 1, \dots, p-1$ можна отримати самостійно, використовуючи лише одну цілу зворотну розгортку розкладу UL на розділі $j+1$. Після того, як скорочена система буде вирішена (з мінімальними витратами), все рішення можна отримати таким чином:

$$A_j X_j = F_j - \begin{bmatrix} 0 \\ I_m \end{bmatrix} B_j X_{j+1}^{(t)} - \begin{bmatrix} I_m \\ 0 \end{bmatrix} C_j X_{j-1}^{(b)}, \quad j = 2, \dots, p-1, \quad \#(1.22)$$

де третій член (відповідно другий член) правої сторони відсутній у випадку $j = 1$ (відповідно $j = p$). Оскільки для A_j доступні розкладу як LU , так і UL , тобто $A_j = L_j U_j$ та $A_j = U_j L_j$ (в рамках схеми ТА ці факторизації були призначені різним процесорам/ядрам), можна переписати вищеведене рівняння таким чином:

$$X_j = U_j^{-1} \left\{ L_j^{-1} F_j - L_j^{-1} \begin{bmatrix} 0 \\ I_m \end{bmatrix} B_j X_{j+1}^{(t)} \right\} - \hat{L}_j^{-1} \left\{ \hat{U}_j^{-1} \begin{bmatrix} I_m \\ 0 \end{bmatrix} C_j X_{j-1}^{(b)} \right\}. \quad \#(1.23)$$

Як результат, перший частина правої сторони в основному передбачає виконання лише однієї цілої зворотної розгортки на процесорах $1, \dots, p-1$. Дійсно, передня розгортка для отримання $L_j^{-1} F_j$ вже виконана як необхідний крок для генерування кінців модифікованої правої сторони G_j (див. рис. 1.4), тоді як другий член у виразі залучав би лише дуже малу частку розгортки вперед. Подібним чином, другий член правої сторони передбачає в основному виконання лише однієї цілої розгортки вперед на процесорах $p, \dots, k$ (тут також потрібна лише дуже мала частка зворотної розгортки).

1.3 Висновки

Даний розділ був підготовлений у ході виконання переддипломної практики. Були більш точно сформульовані цілі та задачі дипломної роботи, визначено основний обсяг роботи та послідовність кроків для її виконання.

Проведено огляд алгоритму, його основних модифікацій та областей застосування. Також, був обраний варіант модифікації для подальшої реалізації.

2. ДОСЛІДЖЕННЯ СТАНДАРТІВ РОЗПАРАЛЕЛЮВАННЯ ПРОГРАМ

2.1 OpenMP

OpenMP - це набір директив компілятора, бібліотечних процедур та змінних середовища, які призначені для програмування багатопоточних застосунків на багатопроцесорних системах із спільною пам'яттю на мовах C, C++ та Fortran які дозволяють

1. Виділяти пам'ять
2. Визначати робоче навантаження
3. Синхронізувати потоки

OpenMP забезпечує виклики функцій для отримання інформації про потоки. наприклад, *omp_num_thread()*, *omp_get_thread_num()*. Основною ідеєю OpenMP є використання нової форми формалізму програмування, яка побудована в середині як:

1. Існуюча послідовна мова, модифікована для обробки паралелізму, як C / C++ і Fortran.
2. Компілятор з підтримкою розпаралелювання.

3. Бібліотечні підпрограми/директиви компілятора, сумісні з послідовною мовою.

У рамках паралелізації спільної пам'яті в OpenMP всі процесори можуть отримати доступ до всієї пам'яті в паралельній системі (один адресний простір). Час доступу до пам'яті може бути не однаковим для всіх процесорів, що означає, використання пам'яті може бути нерівномірним.

Паралельне виконання досягається шляхом генерації кількох потоків, які виконуються паралельно. Кількість потоків (в принципі) не залежить від кількості процесорів.

2.1.1 Паралелізація SMP

Симетрична багатопроцесорна обробка або SMP передбачає багатопроцесорну архітектуру комп'ютерного обладнання, де два або більше однакових процесорів підключені до однієї спільної основної пам'яті і управляються одним екземпляром ОС. Найбільш поширені багатопроцесорні системи сьогодні використовують архітектуру SMP. Для багатоядерних процесорів архітектура SMP застосовується до ядер, розглядаючи їх як окремі процесори, які можуть бути зв'язані між собою за допомогою шин, поперечних перемикачів або вбудованих мереж. SMP часто використовується в науці, промисловості та бізнесі, де працює на спеціальному програмному забезпеченні для багатопотокової обробки.

2.1.2 Робота потоків

Потоки в SMP не є повними процесами UNIX. Вони являють собою легкі, незалежні «збірники інструкцій», які виконуються в процесі UNIX.

1. Усі потоки, створені одним процесом, мають один і той самий адресний простір. Це може бути як плюсом так і мінусом через те, що спілкування між потоками є дуже зручним інструментом, але існує великий ризик одним потоком зіпсувати роботу у всіх інших.

2. Оскільки вони легкі, їм (відносно) недорого створювати та знищувати, використовуючи той факт, що створення потоку може зайняти на три порядки менше часу, ніж створення процесу.

3. Потоки можна створювати і призначати декільком процесорам.

2.1.3 Паралелізм SMP та потоки OpenMP

Використовуючи компілятор розпаралелювання та його директиви, ми можемо генерувати pthreads, використовуючи стандартні директиви (наприклад !OMP тощо)

1. Усі програми OpenMP починаються як єдиний процес: головний потік.
2. FORK: головний потік створює набір паралельних потоків.
3. Паралельні області операцій, що виконуються паралельно між різними наборами потоків.
4. JOIN: потоки синхронізуються та завершуються, залишаючи лише головний потік.

2.1.4 Змінне масштабування

Найважча частина розподілення спільної пам'яті полягає у встановленні наступного:

1. Пам'ять, для спільного користування.

2. Пам'ять, яка буде приватною (тобто кожен процесор матиме власну копію)

3. І як поводитимуться з приватною пам'яттю по відношенню глобального адресного простору.

Змінні спільні за замовчуванням, за винятком індексу циклу в паралельному `do loop`. Вони також повинні поєднуватися з фортранським видом пам'яті. Широко кажучи, змінні можна класифікувати як:

- Глобальні: спільні для всіх процедур
- Локальні: локальні для даної процедури та збережені порівняно зі незбереженими змінними.

2.2 MPI

Інтерфейс передачі повідомлень (MPI) - це стандарт API, який дозволяє процесам обмінюватися даними, надсилаючи та отримуючи повідомлення. Зазвичай він використовується для реалізації паралельних програм, призначених для виконання на високопродуктивних кластерах та суперкомп'ютерах, де вартість доступу до нелокальної пам'яті висока. MPI контролює власні внутрішні структури даних, тобто підтримує неоднорідність.

Комунікатор MPI - це дескриптор, що представляє групу процесів, які можуть спілкуватися між собою. Всі виклики зв'язку MPI мають аргумент комунікатора, такий як `MPI_COMM_WORLD`, який визначається під час здійснення виклику `MPI_INIT`.

2.2.1 Використання MPI

1. Ініціалізація MPI:

`MPI_INIT` має бути першим викликом MPI (лише один раз)

2. Ранг процесу:

Він використовується для визначення джерела та пункту призначення повідомлення. У комунікаторі є номер ідентифікатора процесу, який починається з нуля і переходить до $n - 1$, де n - кількість запитуваних процесів

3. Вихід з MPI:

Ніяких звернень до MPI після завершення. Наприклад, виклик Fortran, `CALL MPI_FINALIZE` повідомляє компілятору вийти з розподіленого середовища. Кількість процесів, що запускаються після виклику цієї процедури, не визначена.

2.3 OpenMP проти MPI

У наведеній нижче таблиці узагальнено ключові відмінності між OpenMP та MPI:

OpenMP	MPI
Залежить від платформи (працює тільки на архітектурах зі спільною пам'яттю)	Не залежить від платформи
Легко поступово розпаралелювати, але складніше писати високо масштабовані програми	Високо розподілене, тобто все або нічого потрібно розпаралелювати
Невеликий API на основі директив компілятора та обмежених бібліотечних процедур	API на основі величезної колекції бібліотечних процедур
Одна і та ж програма може використовуватися для послідовного та паралельного виконання	Можливо, але складно використовувати одну і ту ж програму як для послідовного, так і для паралельного виконання

Спільні та локальні змінні	Грубе локальне масштабування, бо змінні локальні для кожного процесу
----------------------------	--

2.4 Висновки

Оскільки SPIKE алгоритм повністю розкриває свої переваги в паралельному виконанні, тому мета даного розділу продемонструвати основні стандарти API для паралельних обчислень використовуючи які працюють всі сучасні системи розподілених обчислень та виявлені переваги та недоліки кожного з них.

Виявлено, що OpenMP більше підходить для локальних обчислень, тому як мало важить, легко розпаралелюється ітеративно та потребує архітектуру зі спільною пам'яттю, чого легко можна досягнути на локальному комп'ютері. MPI же в свою чергу більше підходить для написання програм кластерного обчислювання, тому як не залежить від платформи та має грубе масштабування.

Найкращим рішенням для розробки високомасштабованого обчислювального алгоритму може стати комбінація цих двох стандартів, у якій на вищому рівні розподілення даних буде відбуватись за допомогою MPI, а локальні обчислення відбуватись з використанням OpenMP.

3. ОГЛЯД БІБЛІОТЕК ПАРАЛЕЛЬНОГО ВИРІШЕННЯ СЛАР

3.1 Intel MKL

Intel Math Kernel Library (Intel MKL) - це бібліотека оптимізованих математичних процедур для науки, техніки та фінансових застосувань. Основні математичні функції включають BLAS, LAPACK, ScaLAPACK, розріджені розв'язувачі, швидкі перетворення Фур'є та векторну математику.

Бібліотека підтримує процесори Intel і доступна для операційних систем Windows, Linux та macOS.

3.1.1 Ефективність

Intel MKL та інші програми, створені компілятором Intel C ++, покращують продуктивність за допомогою техніки, яка називається функцією багатOVERсійного функціонування: функція компілюється або записується для багатьох розширень набору команд x86, а під час виконання "головна функція" використовує CPUID інструкцію для вибору версії, найбільш підходящої для поточного процесора. Однак, якщо головна функція виявляє процесор, який не є процесором Intel, вона майже завжди вибирає найосновнішу (і найповільнішу) функцію для використання, незалежно від того, які інструкції підтримує ЦП. Це присвоїло системі прізвисько "cripple AMD" з 2009 року. Починаючи з 2020 року, Intel MKL, яка залишається обчислювальною бібліотекою, встановленою за замовчуванням попередньо скомпільованими математичними додатками в Windows (такими як NumPy, SymPy). Хоча, спираючись на MKL, MATLAB реалізував обхідний шлях, починаючи з Release 2020a, який забезпечує повну підтримку AVX2 з боку MKL, а також для процесорів, що не належать до Intel (AMD).

У старих версіях встановлення незадокументованої змінної середовища `MKL_DEBUG_CPU_TYPE = 5` може бути використано для заміни шляху, що залежить від постачальника ЦП, і активації підтримуваних інструкцій до AVX2 в системах на базі процесорів AMD, що призводить до рівної або навіть кращої продуктивності порівняно з процесорами Intel. Принаймні з оновлення 1 2020, змінна середовища більше не працює.

Виявлено принаймні два шляхи підключення внутрішніх процедур MKL для усунення дискримінації. Так процедури можна додати під час компіляції шляхом лінковки або під час виконання шляхом ін'єкції. Агнер Фог виявив, що у двійкових файлах MKL та ICC також є недискримінаційний диспетчер. Функція `mkл_serv_intel_cpu_true` була виявлена в 2019 році. `Intel_dispatch_patch` Fog забезпечує код, який підключає обидва маршрути. За словами Даніеля де Кока, лише виправлення останньої функції в MKL 2020 Update 1 покращує продуктивність AMD Zen.

3.1.2 Функціональні категорії

Intel MKL має такі функціональні категорії:

1. Лінійна алгебра: підпрограми BLAS - це вектор-вектор (рівень 1), матриця-вектор (рівень 2) та матриця-матриця (рівень 3) для реальних та складних даних з однією та подвійною точністю. LAPACK складається з розкладів LU, Холецького та QR, пошуку власних значень та методу найменших квадратів. MKL також включає Sparse BLAS, ScaLAPACK, Sparse Solver, Extended Eigensolver, PBLAS та BLACS.
2. Оскільки MKL використовує стандартні інтерфейси для BLAS та LAPACK, програма, яка використовує інші реалізації, може отримати кращу продуктивність на процесорах Intel та сумісних процесорах шляхом повторного зв'язку з бібліотеками MKL.

3. MKL включає різноманітні швидкі перетворення Фур'є (ШПФ) від 1D до багатовимірних, складні до складних, реальні до складних і реальні до реальних перетворень довільної довжини. Програми, написані з відкритим вихідним кодом FFTW, можна легко перенести на MKL, зв'язавши їх з бібліотеками обгортки інтерфейсу, що надаються як частина MKL для зручності перенесення.
4. Кластерні версії LAPACK та FFT також доступні як частина MKL, щоб скористатися перевагами паралелізму MPI на додаток до паралелізму одного вузла від багатопоточності.
5. Функції векторної математики включають обчислювальні інтенсивні основні математичні операції для реальних та складних типів даних з однією та подвійною точністю. Вони схожі на функції libm з бібліотек компілятора, але працюють на векторах, а не на скалярах, щоб забезпечити кращу продуктивність. Існують різні засоби керування для встановлення точності, режиму помилок та денормалізованої обробки чисел для налаштування поведінки процедур.
6. Статистичні функції включають генератори випадкових чисел та розподіли ймовірностей. оптимізований для багатоядерних процесорів. Також включені інтенсивні обчислення в та позаядерних процедурах для обчислення базової статистики, оцінки залежностей тощо.
7. Функції підбору даних включають сплайни (лінійні, квадратні, кубічні, динамічні, ступінчасті постійні) для одновимірної інтерполяції, які можуть бути використані в аналітиці даних, геометричному моделюванні та наближенні поверхні.
8. Глибокі нейронні мережі
9. Часткові диференціальні рівняння
- 10.Вирішення проблем нелінійної оптимізації

3.2 Sun Performance Library

Sun Performance Library - це набір оптимізованих, високошвидкісних математичних підпрограм для вирішення лінійної алгебри та інших чисельно інтенсивних задач. Бібліотека Sun Performance базується на колекції програм загального користування, доступних від Netlib. Sun вдосконалив ці загальнодоступні програми та об'єднав їх як бібліотеку Sun Performance.

Підпрограми Sun Performance Library можуть підвищити продуктивність додатків як на послідовних, так і на багатопроцесорних платформах (MP), оскільки послідовна швидкість багатьох підпрограм Sun Performance Library була збільшена, а багато підпрограм розпаралелені. Підпрограми Sun Performance Library також мають спеціальні оптимізації SPARC, AMD та Intel, яких немає в базових бібліотеках Netlib. Sun Performance Library забезпечує наступні оптимізації та розширення базових бібліотек Netlib:

- Розширення, які підтримують мовні інтерфейси Fortran 95 та C.
- Мовні особливості Fortran 95, включаючи незалежність від типу, перевірка під час компіляції та необов'язкові аргументи.
- Послідовний API для різних бібліотек Sun Performance Library
- Сумісність із бібліотеками LAPACK 1, LAPACK 2.0 та LAPACK 3.x
- Підвищена продуктивність, а в деяких випадках і більша точність
- Оптимізація для конкретних архітектур набору інструкцій SPARC та x86 / x64
- Підтримка 64-розрядних операційних середовищ Solaris та Linux
- Підтримка параметрів компілятора паралельної обробки для SPARC та x86 / x64 платформ
- Підтримка декількох апаратних можливостей процесора

3.2.1 Модулі Sun Performance

Бібліотека Sun Performance містить вдосконалені версії наступних стандартів бібліотек:

- LAPACK версія 3.1.1 - Для розв'язування задач лінійної алгебри.
- BLAS1 (основні підпрограми лінійної алгебри) - для виконання вектор-вектор операцій.
- BLAS2 - для виконання матрично-векторних операцій.
- BLAS3 - для виконання матрично-матричних операцій

Бібліотека Sun Performance доступна як у статичній, так і в динамічній бібліотечній формі. В операційних системах Solaris 10 існують оптимізовані версії SPARC для архітектур sparcvis, sparcvis2 та sparcfmaf. Також існують оптимізовані версії для архітектур x86/x64 в системах Solaris 10, поряд із системами SuSE Linux Enterprise Server 9 та Redhat Enterprise Linux 4. Всі версії мають підтримку паралельного програмування на багатопроцесорних платформах. Процедури Sun Performance Library LAPACK зібрані за допомогою компілятора Fortran 95 і залишаються сумісними з бібліотекою Netlib LAPACK версії 3.1.1. Версії Sun Performance Library цих підпрограм виконують ті самі операції, що і викличні підпрограми Fortran, і мають той самий інтерфейс, що і стандартні версії Netlib. LAPACK містить драйвери, обчислювальні та допоміжні процедури. Sun Performance Library не підтримує допоміжні підпрограми, оскільки допоміжні підпрограми можуть змінюватися або видалятися з LAPACK без попередження.

3.2.2 Математичні функції

Підпрограми Sun Performance Library використовуються для вирішення таких типів лінійної алгебри та числових задач:

- Елементарні векторні та матричні операції - Векторні та матричні продукти; повороти літака; 1, 2- та норми нескінченності; оновлення рангу 1, 2, k та $2k$
- Лінійні системи - вирішує системи повного рангу, обчислює межі помилок, вирішує рівняння Сильвестра, уточнює обчислюване рішення, вирівнює матрицю коефіцієнтів
- Найменші квадрати - повноцінна, узагальнена лінійна регресія, дефіцитний ранг, обмежена лінійна рівність
- Власні задачі - власні значення, узагальнені власні значення, власні вектори, узагальнені власні вектори, вектори Шура, узагальнені вектори Шура
- Розкладання матриць або розкладання - SVD, узагальнені SVD, QL та LQ, QR та RQ, Холескі, LU, Schur, LDLT та UDUT
- Операції підтримки - Номер умови, транспонування на місці або поза місцем, зворотна, детермінанта, інерція
- Розріджені матриці - Розв'язування симетричних, структурно-симетричних та несиметричних матриць за допомогою прямих методів
- Згортка та кореляція в одному та двох вимірах
- Швидкі перетворення Фур'є, синтез Фур'є, косинус та квартер-хвильові косинусні перетворення, косинусні та четвертиволнові синусові перетворення
- Складні векторні ШПФ та БПФ у двох та трьох вимірах
- Операції сортування

3.3 Eigen

Eigen — це бібліотека шаблонів C++ високого рівня для вирішення задач лінійної алгебри, операцій з матрицями і векторами, чисельного аналізу і пов'язаних з цим алгоритмів. Eigen це бібліотека з відкритим кодом, що має

ліцензію MPL2 починаючи з версії 3.1.1. Більш ранні версії розповсюджувалися під ліцензією LGPL3+.

Бібліотека Eigen завжди відзначалась елегантним API, гнучкими можливостями при роботі зі статично і динамічно створеними матрицями і розрідженими матрицями. Для досягнення високої швидкодії, бібліотека Eigen містить складну векторизацію[en] для наборів інструкцій SSE 2/3/4, ARM NEON і AltiVec.

Бібліотека використовується в багатьох проектах, серед яких:

- TensorFlow
- OpenCV
- MeshLab
- VcgLib
- libmv
- Krita
- Yujin Robot
- Willow Garage
- Avogadro
- OpenBabel
- Google Camera

3.4 GNU Scientific Library

Наукова бібліотека GNU (англ. GSL) - це бібліотека, написана на мові програмування C для численних вичислень у прикладній математиці та науці. GSL є частиною проекту GNU та розповсюджується на умовах ліцензії GPL.

GSL використовується, зокрема, у такому програмному забезпеченні, як PSPP та Perl Data Language.

Бібліотека використовує об'єктно-орієнтоване проектування. Різні алгоритми можуть бути легко підключені або змінені під час виконання без повторних програм компіляції. Він призначений для звичайних наукових користувачів. То, хто знає, що деякі програми програмування С матимуть можливість почати використовувати бібліотеку. Там, де це можливо підпрограми були засновані на надійних державно-доменних пакетах Fortran, таких як FFTPACK і Quadpack, що розробники GSL перевизначили в С із сучасними конвенціями кодування.

3.5 Висновки

У розділі були розглянуті найпотужніші інструменти для роботи з лінійною алгеброю, розібрані їх складові та функціонал. Виявлено, що більшість цих інструментів досягає своєї швидкодії за рахунок використання таких низькорівневих бібліотек як LAPACK та BLAS.

Також, були обрані бібліотеки для порівняння ефективності роботи реалізованого алгоритму в майбутньому. Зокрема Intel MKL, оскільки ця бібліотека передвстановлена у такі високорівневі модулі як NumPy та SymPy, що дозволяє значно полегшити тестування.

4. РЕАЛІЗАЦІЯ ТА АНАЛІЗ SPIKE АЛГОРИТМУ

4.1 Деталі реалізації

Для реалізації алгоритму для реалізації алгоритму було обрано мову програмування Python оскільки вона має багато оптимізованих модулів для роботи з матричною математикою та в той же час мова дуже високорівнева, що дозволяє швидко та зручно вести обробку.

Для написання алгоритму використовувались наступні модулі Python.

4.1.1 Numpy



NumPy - це бібліотека для мови програмування Python, що додає підтримку великих багатовимірних масивів та матриць з великою колекцією математичних функцій високого рівня для роботи з цими масивами. Родоначальник NumPy, Numeric, спочатку був створений Джимом Хугуніном за участю кількох інших розробників. У 2005 році Тревіс Оліфант створив NumPy, включивши функції конкуруючого Numarray в Numeric, із великими змінами. NumPy - це програмне забезпечення з відкритим кодом, у якому є багато авторів.

NumPy націлений на реалізацію CPython з обгорткою Python, яка є не оптимізуючим інтерпретатором байт-кодів. Математичні алгоритми, написані для CPython, часто працюють набагато повільніше, ніж зібрані еквіваленти. NumPy вирішує проблему повільності частково, надаючи багатовимірні масиви та функції та оператори, які ефективно працюють з масивами, вимагаючи переписувати деякий код, переважно внутрішні цикли, за допомогою NumPy.

Використання NumPy в Python надає функціональність, порівнянну з MATLAB, оскільки вони обидва інтерпретуються, і обидва вони дозволяють користувачеві писати швидкі програми, якщо більшість операцій працюють над масивами або матрицями замість скалярів. Для порівняння, MATLAB може похвалитися великою кількістю додаткових наборів інструментів, зокрема Simulink, тоді як NumPy невід'ємно інтегрована з Python, більш сучасною та повною мовою програмування. Більше того, доступні додаткові пакети Python; SciPy - це бібліотека, яка додає більше функціональних можливостей, подібних до MATLAB, а Matplotlib - це пакет для побудови графіків, що забезпечує функціональність, подібну до MATLAB. Внутрішньо як MATLAB, так і NumPy покладаються на BLAS і LAPACK для ефективних обчислень лінійної алгебри.

Прив'язки Python до широко використовуваної бібліотеки комп'ютерного зору OpenCV використовують масиви NumPy для зберігання та роботи з даними. Оскільки зображення з кількома каналами просто представлені у вигляді тривимірних масивів, індексування, нарізка чи маскування за допомогою інших масивів є дуже ефективними способами доступу до певних пікселів зображення. Масив NumPy як універсальна структура даних у OpenCV для зображень, витягнутих точок об'єктів, ядер фільтрів та багатьох інших значно спрощує робочий процес програмування та налагодження.

Основною функціональністю NumPy є його "ndarray", для n-мірного масиву, структура даних. Ці масиви є послідовними видами пам'яті. На відміну від вбудованої структури даних списку Python, ці масиви однорідні за типами значень: всі елементи одного масиву повинні бути одного типу.

Такі масиви також можуть бути переглядами буферів пам'яті, виділених розширеннями C / C ++, Cython та Fortran для інтерпретатора CPython, без необхідності копіювати дані навколо, надаючи певний рівень сумісності з

існуючими числовими бібліотеками. Ця функціональність використовується пакетом SciPy, який обгортає ряд таких бібліотек (зокрема BLAS та LAPACK). NumPy має вбудовану підтримку відображених пам'яті ndarrays.

4.1.2 SciPy



SciPy - це бібліотека Python з відкритим кодом, що використовується для наукових обчислень та технічних обчислень. SciPy містить модулі для оптимізації, лінійної алгебри, інтеграції, інтерполяції, спеціальних функцій, FFT обробки сигналів та зображень, вирішувачів ODE та інших завдань, поширених у науці та техніці. Бібліотека базується на об'єкті масиву NumPy і є частиною стеку NumPy, який включає такі інструменти, як Matplotlib, pandas та SymPy, а також розширюється набір наукових обчислювальних бібліотек. Цей стек NumPy має користувачів, подібних до інших програм, таких як MATLAB, GNU Octave та Scilab. Стек NumPy також іноді називають стеком SciPy.

SciPy також є сімейством конференцій для користувачів та розробників цих інструментів: SciPy (у США), EuroSciPy (у Європі) та SciPy.in (в Індії). Enthought започаткував конференцію SciPy у Сполучених Штатах і продовжує фінансувати багато міжнародних конференцій, а також розміщувати веб-сайт SciPy.

Наразі бібліотека SciPy розповсюджується за ліцензією BSD, і її розвиток фінансується та підтримується відкритою спільнотою розробників. Він також підтримується NumFOCUS, фондом громади для підтримки відтворюваної та доступної науки.

4.1.3 Зберігання матриці

У поточній реалізації доступ до пам'яті на процесорі найкраще досягати за допомогою формату CSR (Compressed Sparse Row), який забезпечує, що ненульові елементи в одному рядку зберігаються суміжно. Для діагональних матриць щільної смуги A_i ми приймаємо "високий і тонкий" накопичувач у великому порядку стовпців (як показано на Рис. 4.1 для матриці з $N = 8$ і $K = 2$).

$$\begin{bmatrix} * & * & a_{11} & a_{21} & a_{31} \\ * & a_{12} & a_{22} & a_{32} & a_{42} \\ a_{13} & a_{23} & a_{33} & a_{43} & a_{53} \\ a_{24} & a_{34} & a_{44} & a_{54} & a_{64} \\ a_{35} & a_{45} & a_{55} & a_{65} & a_{75} \\ a_{46} & a_{56} & a_{66} & a_{76} & a_{86} \\ a_{57} & a_{67} & a_{77} & a_{87} & * \\ a_{68} & a_{78} & a_{88} & * & * \end{bmatrix}$$

Рисунок 4.1 – Діагональна матриця розміром 8x8 та товщиною смуги 5 у форматі CSR.

4.2 Аналіз роботи алгоритму

Для порівняння роботи швидкості алгоритму було обрано 3 бібліотечні функції з пакетів оптимізованих та популярних пакетів numpy, та sclearn.

- `np.linalg.solve()`:

Рішення обчислюються за допомогою підпрограми LAPACK `_gesv`. Матриця повинна бути квадратною та позитивно визначена, тобто всі

рядки (або, що еквівалентно, стовпці) повинні бути лінійно незалежними. Функція розкладання LU з частковим обертанням та обміном рядків використовується для коефіцієнта A як:

$$A = P * L * U$$

Де P - матриця перестановок, L – нижньотрикутна матриця U – верхньотрикутна матриця.

Потім розкладена на множники форма A використовується для розв'язання системи рівнянь $A * X = B$.

- `scipy.linalg.solve()`:

Функція також використовує підпрограми LAPACK `?GESV`, `?SYSV`, `?HESV` і `?POSV`, але має внутрішню передобробку даних.

- `scipy.sparse.linalg.spsolve()`:

Функція використовує спеціальний LU оптимізований під розріджені системи (`scipy.sparse.linalg.SuperLU`)

4.2.1 Порівняння швидкості роботи

Для вхідних даних були випадково згенеровані квадратні смугові матриці розмірами (1200, 1800, 2700, 4050, 6075, 9112, 13668, 20502, 30753) а відносними широтами смуг (1%, 2%, 4%, 8%, 16%, 32%). Тестування проводиться на процесорі Intel® Core(TM) i7-8565U CPU @ 1.80Ghz 1.99Ghz який має 4 ядра та 8 потоків, тому у алгоритмі за замовчуванням стоїть розбиття на 8 частин.

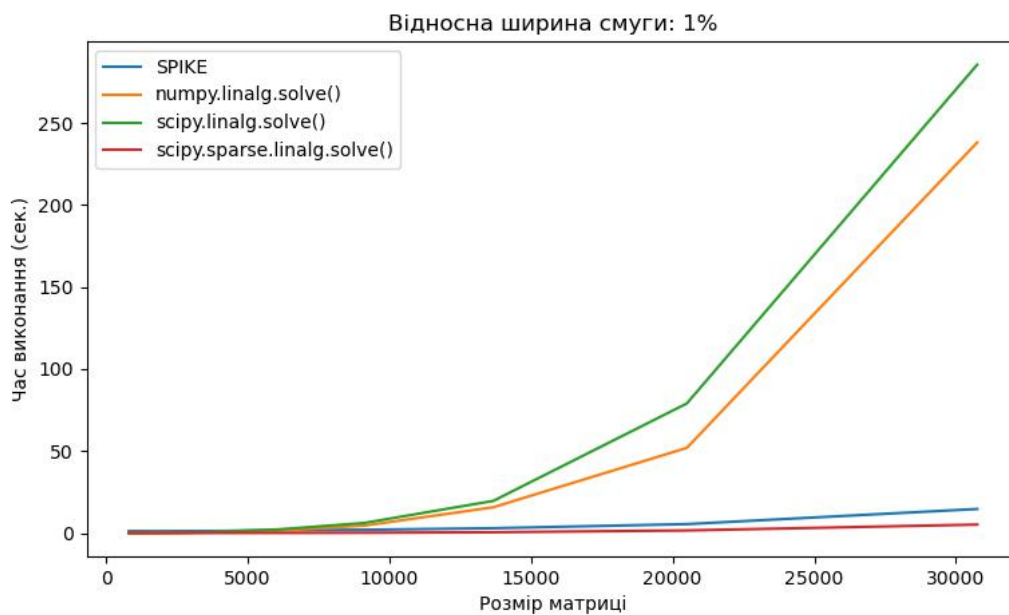


Рисунок 4.2 – Графік швидкості вирішення СЛАР зі смуговою матрицею відносної ширини 1%

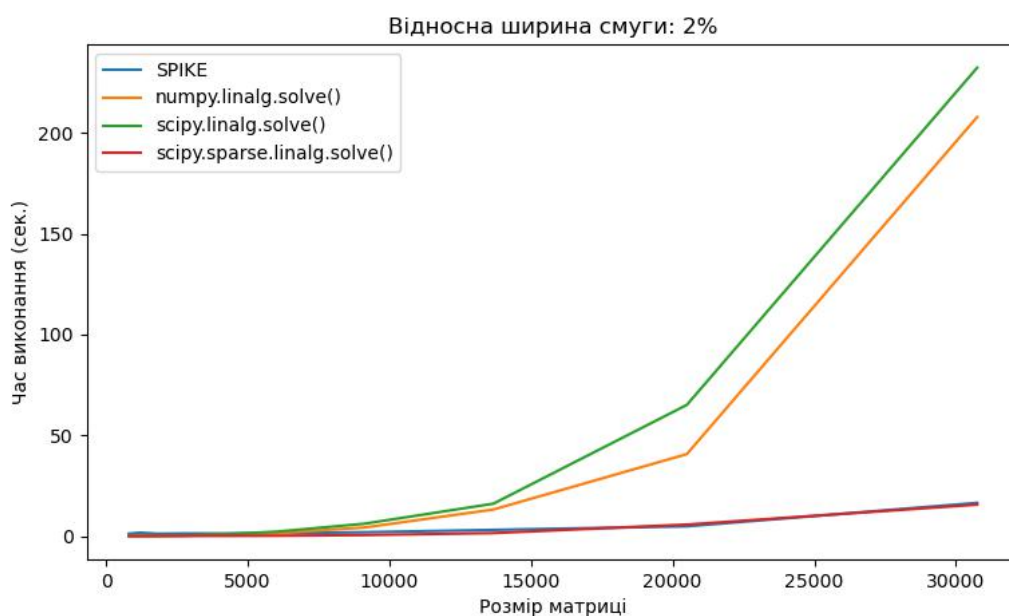


Рисунок 4.3 – Графік швидкості вирішення СЛАР зі смуговою матрицею відносної ширини 2%

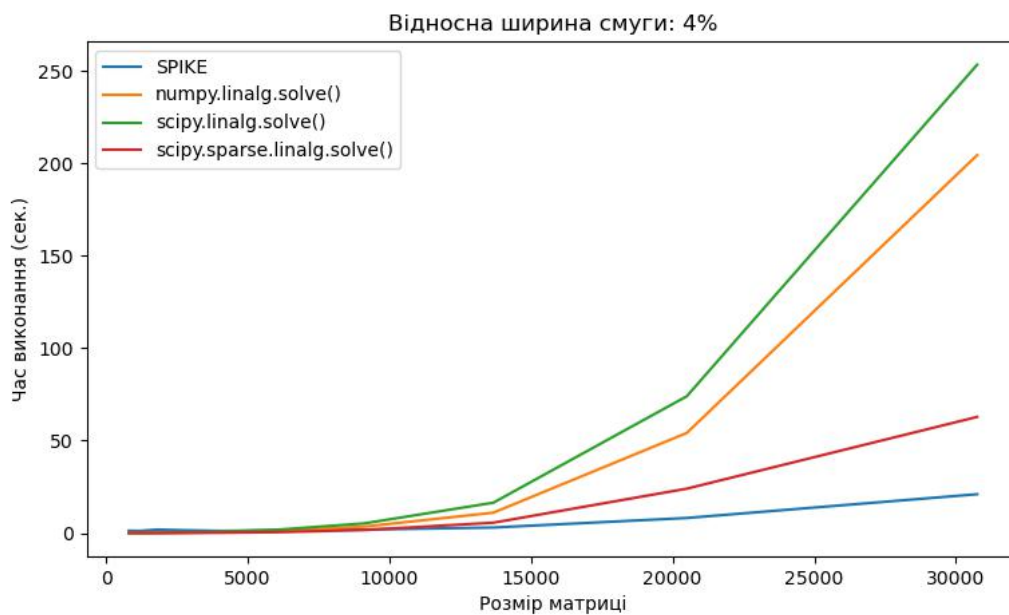


Рисунок 4.4 – Графік швидкості вирішення СЛАР зі смуговою матрицею відносної ширини 4%

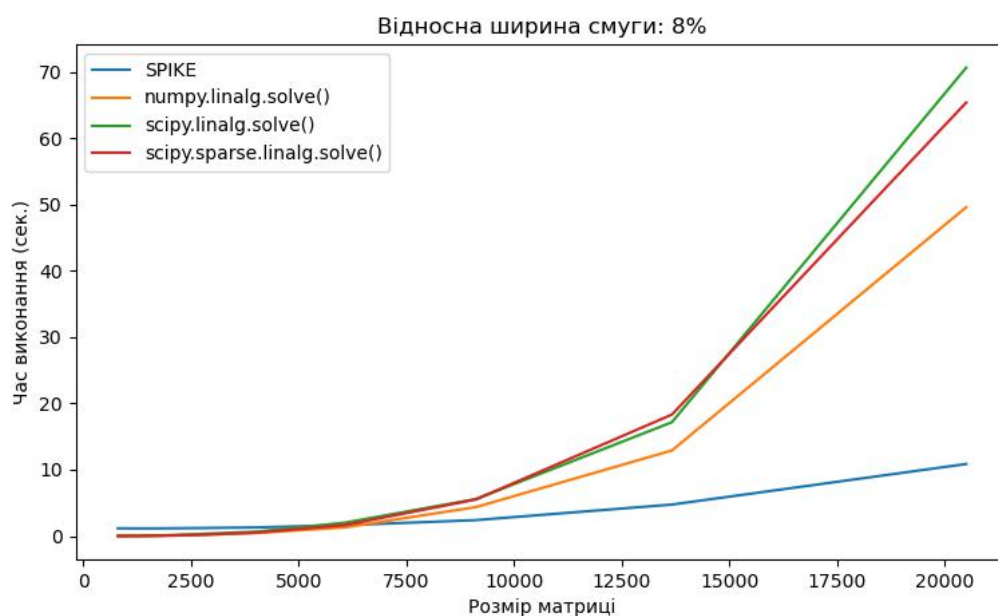


Рисунок 4.5 – Графік швидкості вирішення СЛАР зі смуговою матрицею відносної ширини 8%

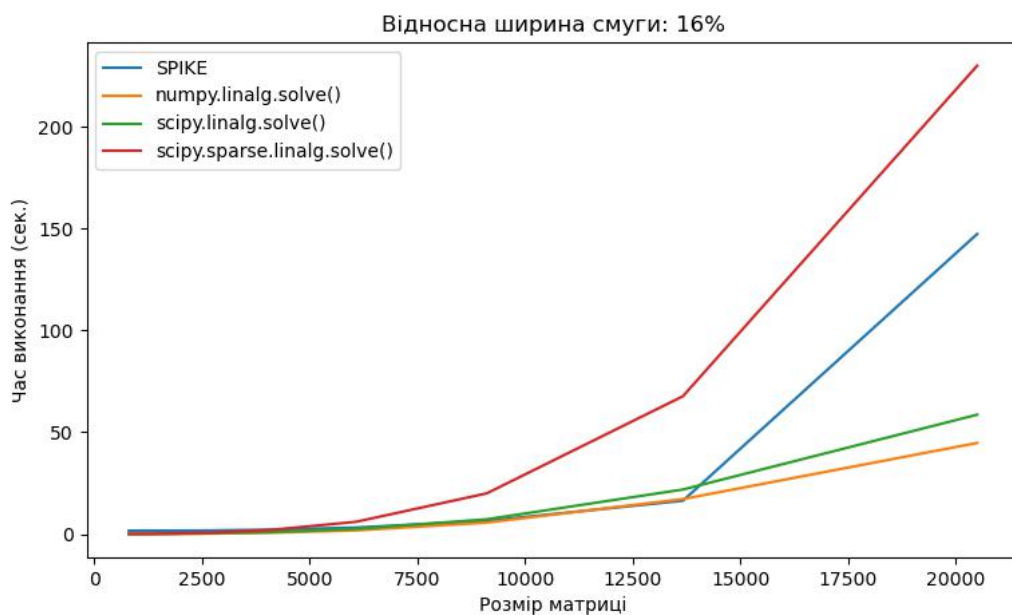


Рисунок 4.6 – Графік швидкості вирішення СЛАР зі смуговою матрицею відносної ширини 16%

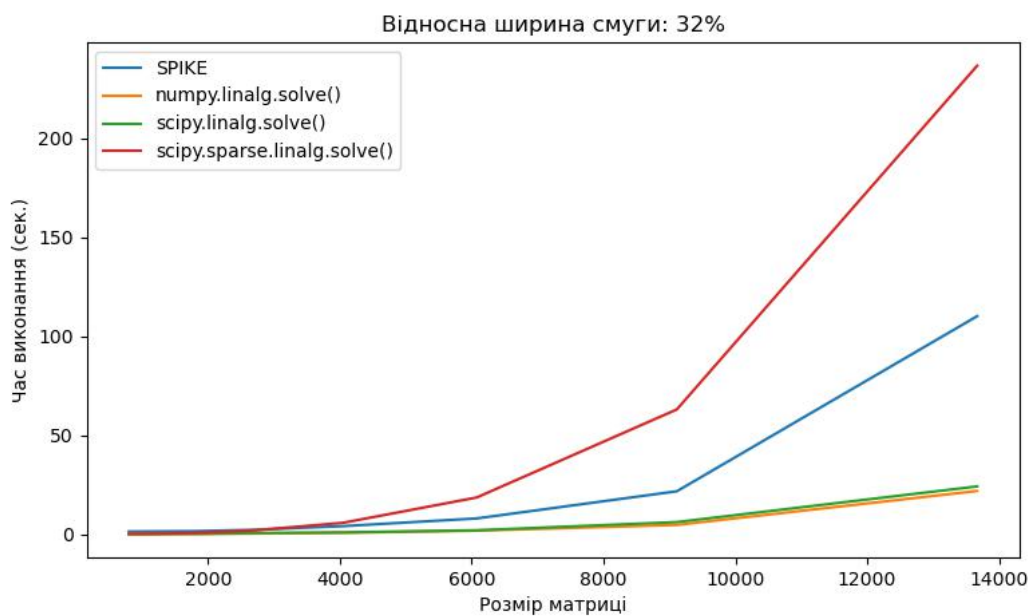


Рисунок 4.7 – Графік швидкості вирішення СЛАР зі смуговою матрицею відносної ширини 32%

4.2.2 Аналіз ефективного розбиття системи

Для дослідження залежності швидкості роботи алгоритму від величини дроблення матриці (в межах 5 до 25 поділів). Були обрані випадково згенеровані квадратні матриці розміром 25,000 та відносними широтами смуги 1%, 4%.

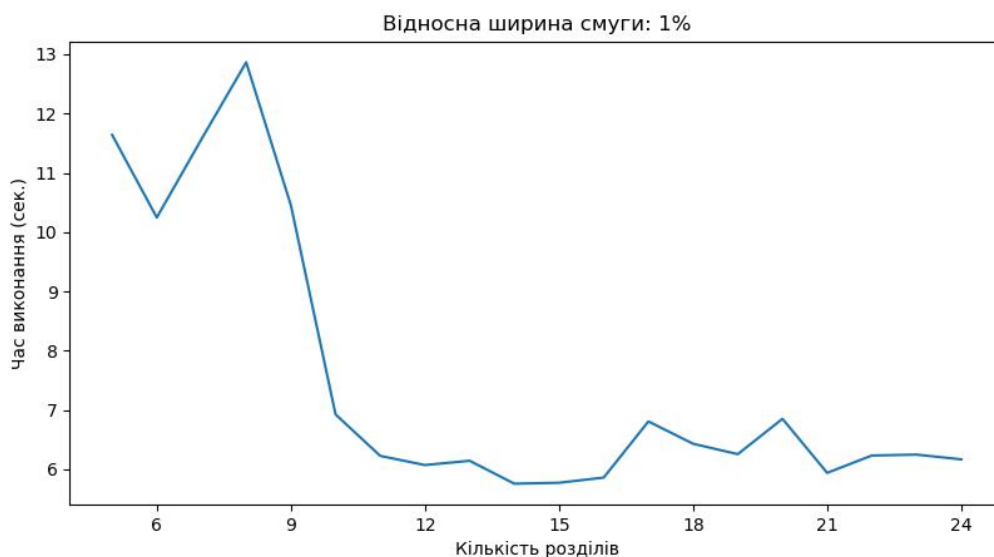


Рисунок 4.8 – Графік швидкості вирішення СЛАР зі смуговою матрицею відносної ширини 1% в залежності від кількості поділів.

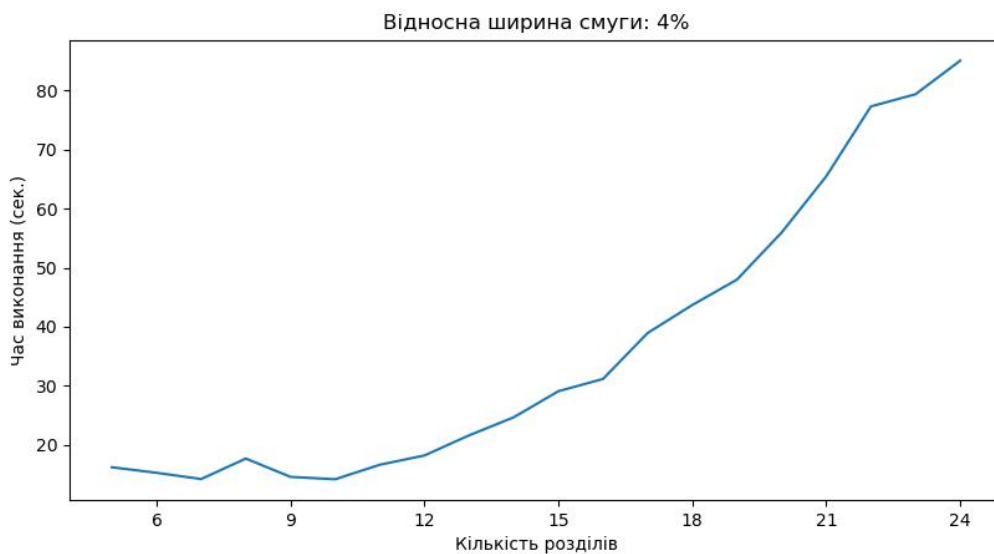


Рисунок 4.9 – Графік швидкості вирішення СЛАР зі смуговою матрицею відносної ширини 4% в залежності від кількості поділів.

Для дослідження залежності спожитої пам'яті від величини дроблення матриці (в межах 3 до 17 поділів). Були обрані випадково згенеровані квадратні матриці розмірами 5000, 8000, 18000, 20000 та відносними широтами смуги 1%, 4%, 10%.

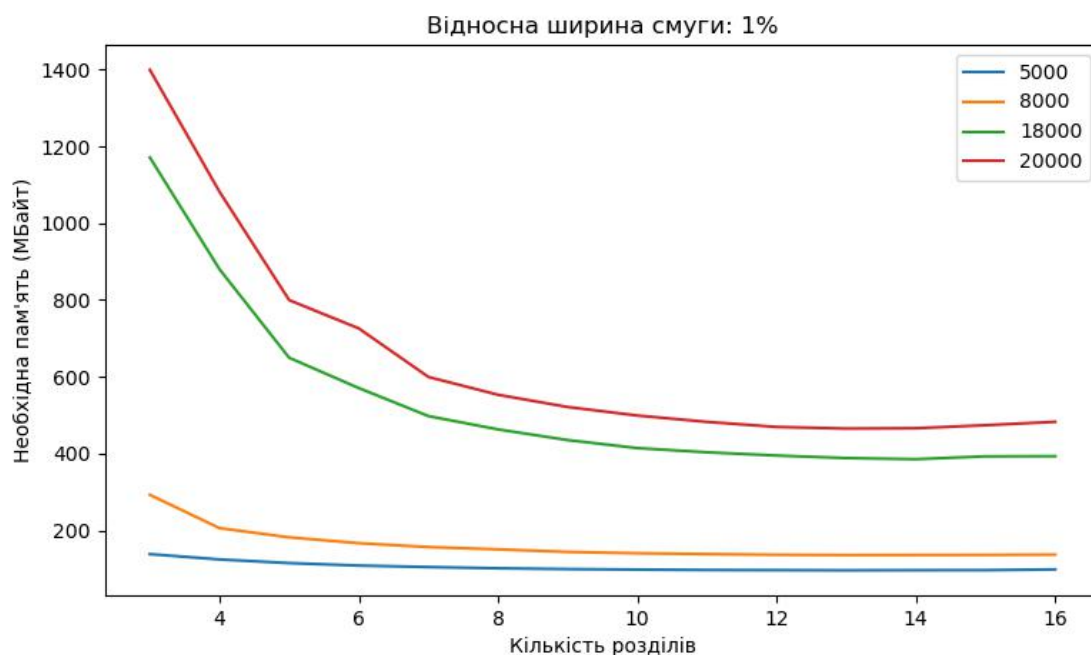


Рисунок 4.10 – Графік залежності спожитої алгоритмом пам'яті від кількості розділів матриці для ширини смуги 1%.

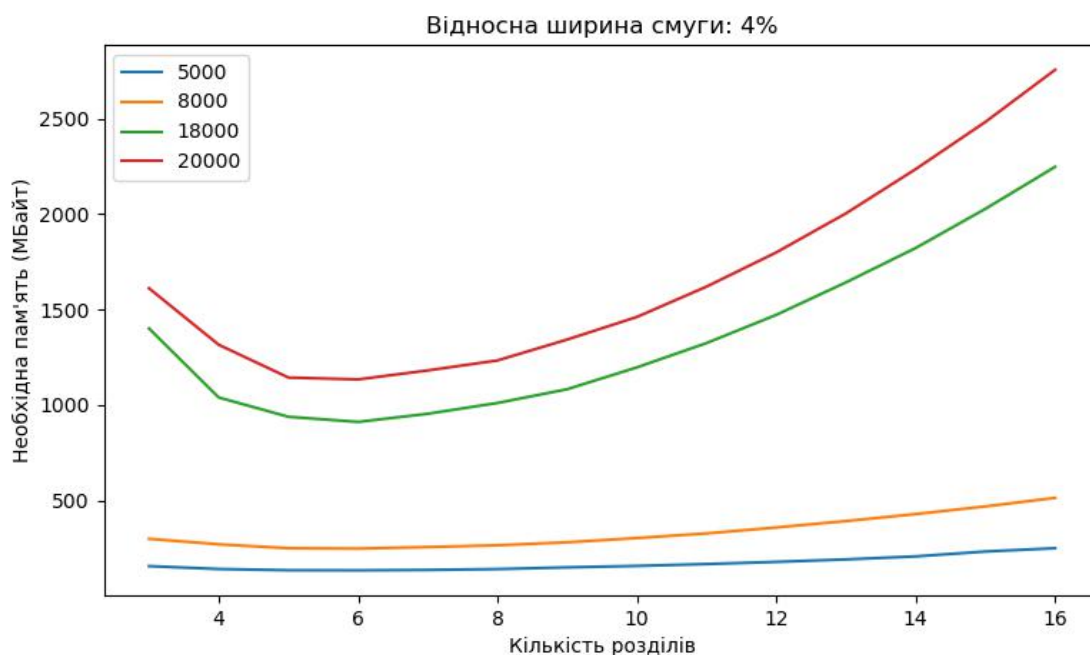


Рисунок 4.11 – Графік залежності спожитої алгоритмом пам'яті від кількості розділів матриці для ширини смуги 4%.

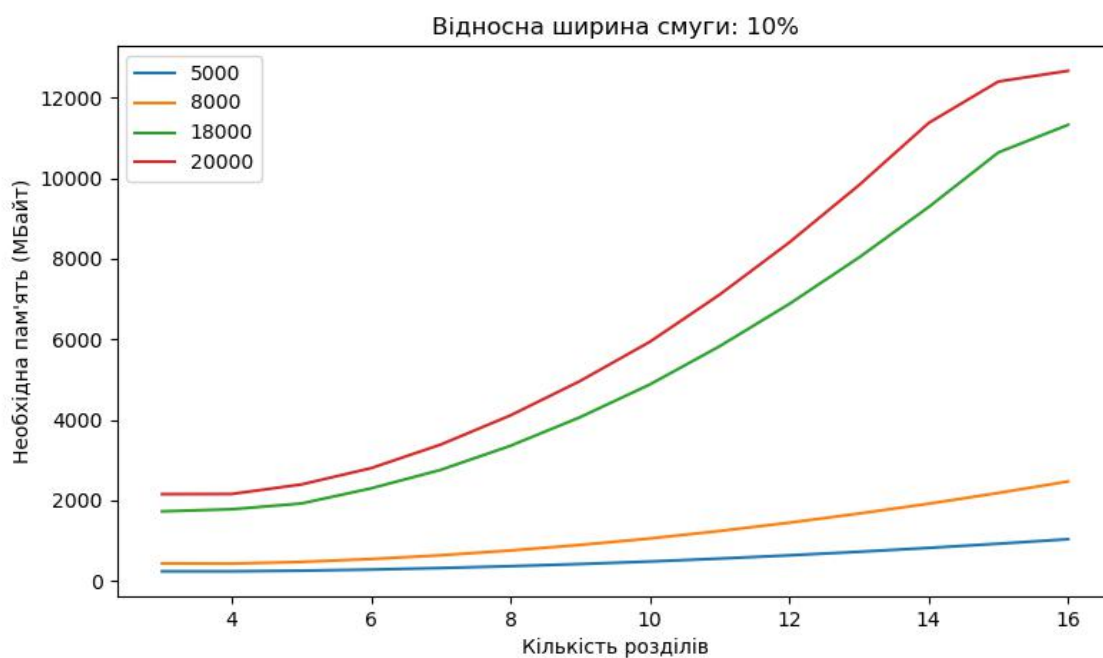


Рисунок 4.12 – Графік залежності спожитої алгоритмом пам'яті від кількості розділів матриці для ширини смуги 10%.

Як видно з графіків, при збільшенні щільності матриці мінімум споживаної пам'яті зміщується вліво.

4.2.3 Порівняння рівнів споживання пам'яті

Заміри зроблені для квадратних матриць розмірами 5000, 8000, 18000, 22000) та відносними широтами смуги 1%, 4% і 16%. Також була обрана оптимальна кількість розділів для кожного порядку матриці, які були отримані у попередньому пункті.

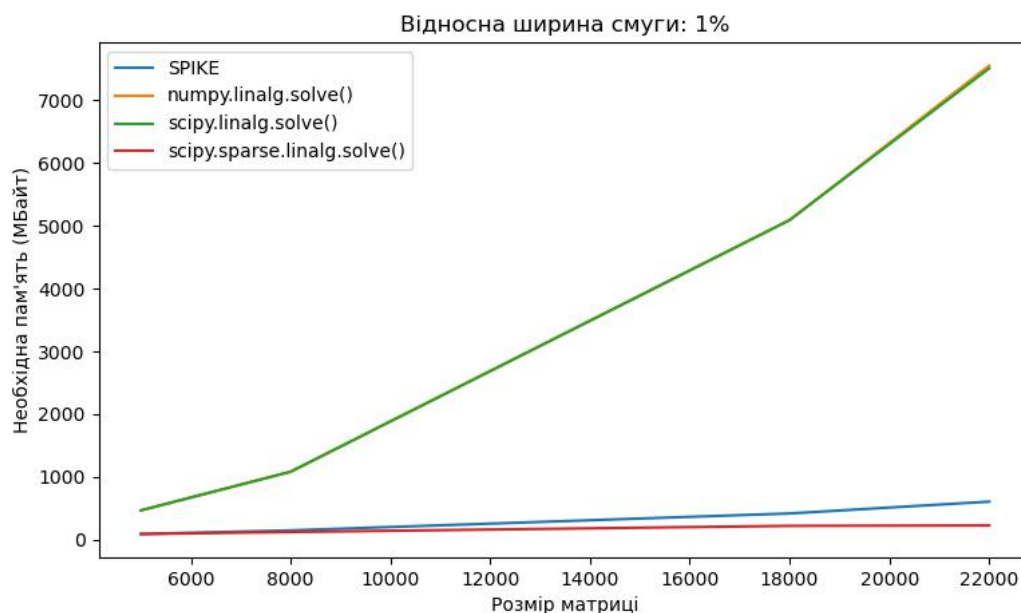


Рисунок 4.13 – Графік використання пам'яті алгоритмами при вирішенні СЛАР зі смуговою матрицею відносної ширини 1%.

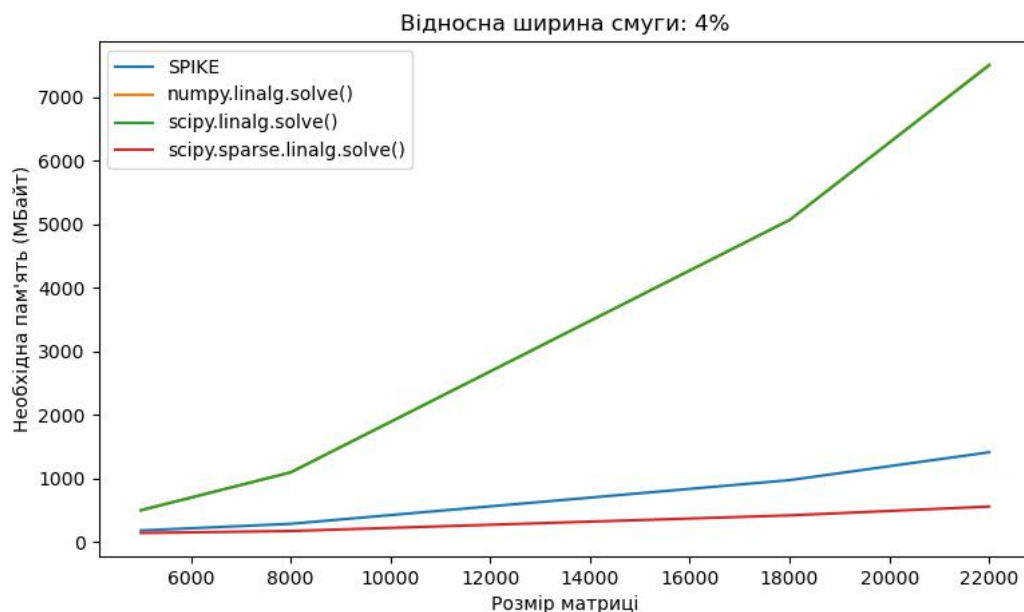


Рисунок 4.14 – Графік використання пам'яті алгоритмами при вирішенні СЛАР зі смуговою матрицею відносної ширини 4%.

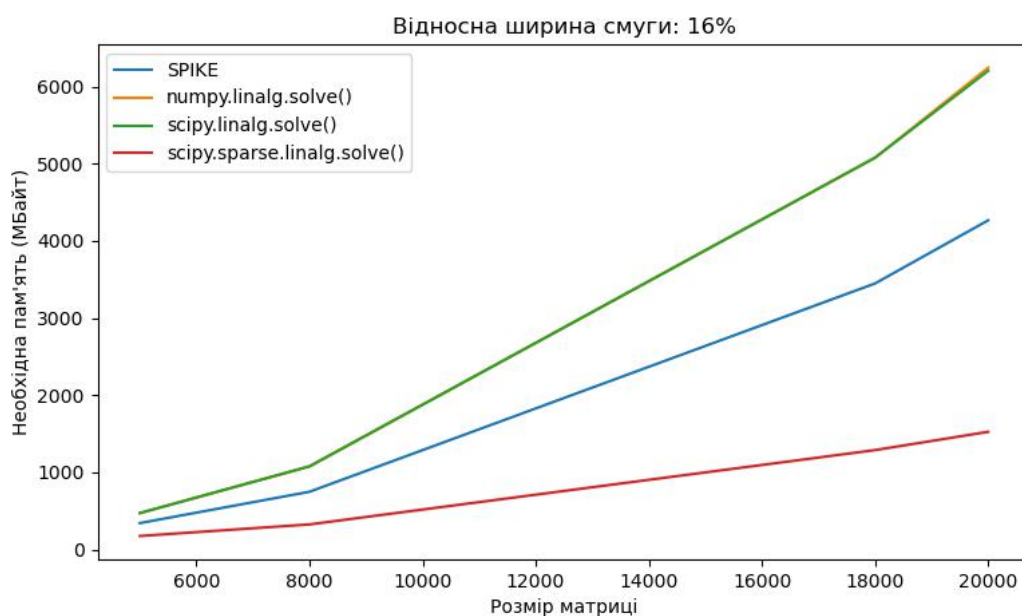


Рисунок 4.15 – Графік використання пам'яті алгоритмами при вирішенні СЛАР зі смуговою матрицею відносної ширини 16%.

4.3 Висновки

В ході даної роботи був розроблений та проаналізований SPIKE алгоритм. Сам програмний модуль був написаний на мові Python з використанням оптимізованих наукових бібліотек NumPy і SciPy. А також, окрім самого алгоритму, був створений модуль для зберігання та обробки смугових матриць в скороченому форматі.

При аналізі швидкості реалізованого алгоритму було виявлено, що алгоритм є дуже ефективним при вирішенні розріджених смугових матриць, с шириною смуги до 8% відносно довжини сторони матриці, і навіть у деяких випадках працює краще ніж функція `scipy.sparse.linalg.spsolve()`, яка створена для вирішення СЛАР з розрідженими матрицями (Рис. 4.5). Але при товщині смуги більше 12% алгоритм втрачає свої переваги і витрачає у рази більше часу на вирішення системи у порівнянні зі звичайними алгоритмами.

При аналізі ефективної кількості розділів зроблений висновок, що при ширині матриці менше 2% збільшення кількості розділів (в певних межах) веде до пришвидшення роботи алгоритму, а також до зменшення споживання пам'яті через зменшення розмірів розділів, та збільшення їх щільності, але при великих розмірах смуги, ефективною кількістю розділень є значення розраховане в пункті 1.2.3 .

Стосовно споживання пам'яті, як зазначалось у розділі 4.1.3 алгоритм використовує скорочений запис смугової матриці, що дозволяє не зберігати велику кількість нульових елементів , та значно знизити використання пам'яті для розріджених матриць (товщина смуги менше 20%).

5. РОЗРОБКА СТАРТАП-ПРОЕКТУ «SPIKE АЛГОРИТМ»

5.1 Опис ідеї проекту

Розділ містить економічне обґрунтування стартап-проекту “SPIKE алгоритм”. Розділ має на меті ознайомлення з економічними та функціональними характеристиками майбутнього проекту, економічними аспектами його реалізації та впровадження у використання. Опис основної ідеї майбутнього стартап-проекту наведено в Таблиці 5.1.

Таблиця 5.1 - Опис ідеї стартап-проекту

<i>Зміст ідеї</i>	<i>Напрямки застосування</i>	<i>Вигоди для користувача</i>
Ідея полягає у створенні SPIKE бібліотеки з реалізацією різних модифікацій алгоритму	1. Вирішення наукових задач по моделюванню.	Зручність використання, висока ефективність роботи, низьке споживання пам'яті
	2. Підвищення ефективності використання розподілених систем.	Алгоритм дуже добре піддається масштабуванню.

Таблиця містить опис ідеї стартап-проекту “SPIKE алгоритм”. Таблиця має на меті ознайомлення з економічними та функціональними вигодами майбутнього проекту. Для оцінювання конкурентоспроможності та можливості і складності виходу стартапу на ринок було виконано порівняння із рядом потенційних конкурентів, до яких можуть бути застосовані обрані для порівняння характеристики (Підтримка декількох мов програмування, Собівартість, Час розробки, Стек технологій і т.д.). Результати порівняння наведені в Таблиці 5.2. В таблиці 5.2 наведено порівняння із рядом потенційних

конкурентів, що надає можливість оцінки конкурентоспроможності та можливості і складності виходу стартапу на ринок.

Таблиця 5.2 - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п / п	Техніко- економічні характери- стики ідеї	Потенційні товари/концепції конкурентів				W (слаб ка стор она)	N (нейтр альна сторон а)	S (сильна сторон а)
		Мій проект т	Конкурен т т1	Конкурен т т2	Конкурен т т3			
1.	Підтримк а популярн их мов програму вання	+	-	-	-			+
2.	Собіварті сть	Низька	Висока	Середня	Висока			+
3.	Час розробки	довгий	коротки й	коротки й	коротки й	+		
4.	Стек технологі й	різнома нітний	різнома нітний	одноман ітний	одноман ітний			+
5.	Додатков і вимоги	Освоєн ня низько рівневи х бібліот	немає	немає	немає	+		

		ек						
--	--	----	--	--	--	--	--	--

Проект має сильні сторони, що відсутні в існуючих аналогах та здатний із ними конкурувати. Наявність пакету з алгоритмом для багатьох мов програмування, низька собівартість. Слабкою стороною є вивчення нових низькорівневих технологій з нуля для оптимізації та покращення алгоритму.

5.2 Технологічний аудит ідеї проекту

Проводиться аудит технологій, за допомогою якої може бути реалізована ідея проекту (технології створення товару).

Таблиця 5.3 - Технологічна здійсненність ідеї проекту

№ n/n	Ідея проекту	Технологія її реалізації	Наявність технології	Доступність технології
1.	Створення SPIKE алгоритму.	BLAS, LAPACK, OpenMP	Наявна	Безплатна, доступна
		Intel MLK	Наявна	Платна, доступна
Обрана технологія реалізації ідеї проекту: для розробки алгоритму планується використовувати рутини бібліотек BLAS і LAPACK які є досить легкими і доступними, сам же алгоритм буде написаний на мові C з використання OpenMP стандарту для його розпаралелювання.				

У таблиці Таблиця 5.3 надано результати огляду основних стеків технологій, що можуть бути використані з метою реалізації системи стартап-проекту, описаного вище. Було обрано стек технологій, що не потребує додаткових витрат але потребує більш глибоких знань предметної області.

5.3 Аналіз ринкових можливостей

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів. Було проведено аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку. Результати аналізу представлені у Таблиці 5.4. У Таблиці 5.4 надано результати аналізу характеристик попиту на ринку. На ринку присутні чотири основних представника, а загальний обсяг продаж є значним. Проте зростання динаміки ринку та відсутність суттєвих обмежень для входу на ринок дозволяють це виконати стартап-проекту з наведеними у попередніх підрозділах характеристиками.

Таблиця 5.4 - Попередня характеристика потенційного ринку стартап-проекту

<i>№ n/n</i>	<i>Показники стану ринку (найменування)</i>	<i>Характеристика</i>
1.	Кількість головних гравців, од	3
2.	Загальний обсяг продаж, грн/ум.од	10000 грн./ум.од
3.	Динаміка ринку (якісна оцінка)	зростає
4.	Наявність обмежень для входу (вказати характер обмежень)	Висока конкуренція
5.	Специфічні вимоги до стандартизації та сертифікації	немає
6.	Середня норма рентабельності в галузі (або по ринку), %	20%

Отже, було проаналізовано наявність попиту, обсяг, динаміку розвитку ринку. Є досить висока конкуренція, динаміка ринку зростає, галузь є рентабельною. Далі визначаємо потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної з груп (табл. 5.5).

Таблиця 5.5 - Характеристика потенційних клієнтів стартап-проекту

<i>№ n/n</i>	<i>Потреба, що формує ринок</i>	<i>Цільова аудиторія (цільові сегменти ринку)</i>	<i>Відмінності у поведінці різних потенційних цільових груп клієнтів</i>	<i>Вимоги споживачів до товару</i>
1.	Потреба у швидких та оптимізованих алгоритмах вирішення смугових СЛАР	Науковці, розробники математичних бібліотек, розробники ПО для моделювання	немає	Рішення повинно бути сприятливим для масштабування, незалежним у розгортанні, швидким у обробці даних, та мати оптимізоване використання пам'яті

Відповідно до виконаної характеристики потенційних клієнтів стартап-проекту впливає, що на ринку запропонована концепція з розробки програмного забезпечення є актуальною. Після визначення потенційних груп

клієнтів проведемо аналіз ринкового середовища: таблиця факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (табл. 5.6, 5.7).

Таблиця 5.6 - Фактори загроз

<i>№ n/n</i>	<i>Фактор</i>	<i>Зміст загрози</i>	<i>Можлива реакція компанії</i>
1.	Конкуренція	Високий рівень фінансування та якості кадрів	Для мінімізації даного ризику необхідно достатньо і безперервно вдосконалювати концепцію стартап-проекту, вводити нові рішення для розв'язання тих чи інших задач.
2.	Ефективність методу	Винайдення більш ефективного методу	Постійно моніторити нові винайдення у сфері та модифікувати бібліотеку згідно з останніми відкриттями.
3.	Держава	Зростання податкового тягаря	Перегляд виконання умов, що зменшують податки Поступове підвищення тарифів

“Відповідно до результатів аналізу можливих факторів загроз стартап-проекту описаного продукту, наведеного у Таблиці 5.6 та необхідного для виходу на ринок, описаний в Таблиці 5.4, існує ряд ризиків які слід враховувати при планування виходу продукту на ринок та мати орієнтовні сценарії їх мінімізації та компенсування їх впливу, наведені в таблиці вище. Аналогічно до загроз стартап-проекту, що можуть спричинити значні проблеми для його розвитку, важливою частиною є огляд можливих сприятливих умов, використання яких може значно покращити становище стартап-проекту та

надати перевагу порівняно із конкурентами. Такі сприятливі умови та відповідні можливості розглянуто в Таблиці 5.7.” [26]

Таблиця 5.7 - Фактори можливостей

<i>№ n/n</i>	<i>Фактор</i>	<i>Зміст можливостей</i>	<i>Можлива реакція компанії</i>
1.	Можливості користувачів	Зростання можливостей потенційних покупців	Таргетована реклама
2.	Конкуренція	Зниження довіри до конкурента 1 внаслідок помилок в функціонуванні	Акцентувати увагу на надійності системи
3.	Технології	Поява нових технологій	Аналіз технології, включення нових модулів (що технологію використовують) до адаптивної системи
4.	Держава	Послаблення обмежень в законодавстві	Оптимізація діяльності для скорочення витрат

“Відповідаючи на результати аналізу можливих несприятливих умов для описаного продукт, введений в Таблиці 5.7 і вкрай бажаного для виходу на ринок, описаний в таблицях 5.4, існує ряд можливих реакцій компанії, правильне застосування яких може надавати значну перевагу в порівнянні з конкурентами Конкуренція на ринку може стати причиною занепаду компанії, так і стимул за допомогою якого стартап-проект значно поліпшив якість послуг

отримав корисний для майбутнього досвіду розвитку. У зв'язку з цим в Таблиці 5.8 проводити аналіз пропозицій: визначають загальні риси конкуренції на ринку стартап-проект. Відповідаючи на результати аналізу пропозицій описаного продукту, зазначеного в Таблиці 5.8 і необхідного для виходу на ринок, описані в таблицях 5.4, визначено загальні риси конкуренції на ринку стартап-проекту і вказано можливий вплив на діяльність підприємства - можливі дії і заходи компанії, спрямовані на підвищення її конкурентоспроможності.” [26]

Таблиця 5.8 - Ступеневий аналіз конкуренції на ринку

<i>№ п/п</i>	<i>Особливості конкурентного середовища</i>	<i>В чому проявляється дана характеристика</i>	<i>Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)</i>
1.	Вказати тип конкуренції: - досконала	Існує 3 фірми конкуренти на ринку	Врахувати ціни конкурентних компаній на початкових етапах створення бізнесу, реклама (вказати на конкретні переваги перед конкурентами)
2.	Конкуренція за видами товарів: - товарно-видова	Однаковий вид товарів (ПЗ) і послуг (сфера медицини)	Створити алгоритм, враховуючи недоліки конкурентів
3.	За характером конкурентних переваг: - нецінова	Вдосконалення концепції, щоб створення ПЗ вимагало	Використання найбільш доступних технологій, що є на ринку, але при цьому найоптимальніших

		менших зусиль	
4.	За інтенсивністю: - не марочна	Бренди відсутні	-

Після аналізу конкуренції проведемо більш детальний аналіз умов конкуренції в галузі (табл. 5. 9) [30].

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

<i>Складові аналізу</i>	<i>Прямі конкуренти в галузі</i>	<i>Потенційні конкуренти</i>	<i>Постачальники</i>	<i>Клієнти</i>	<i>Товари-замінники</i>
	<i>Навести перелік прямих конкурентів</i>	<i>Визначити бар'єри входження в ринок</i>	<i>Визначити фактори сили постачальників</i>	<i>Визначити фактори сили споживачів</i>	<i>Фактори загроз з боку замінників</i>
Висновки	Існує 3 Конкуренти на ринку	Можливості для входу на ринок наявні Надане рішення спрощує та пришвидшує роботу спеціаліста.	Постачальники відсутні	Для користувачі в важливим швидкодія алгоритму	Наявні концепції не несуть загроз даній, але можуть з'явитися найближчим часом

Згідно Підсумком розбору пропозиції описаного продукту, показані в таблиці 5.9 і потрібного для виходу на ринок, описаний в таблиці 4.4, визначено основні особливості конкурентної боротьби на ринку стартап-плану і зазначено висновки по кожній з складових зробленого розбору для вдалої конкурентній боротьби на ринку. На основі аналізу конкурентної боротьби з урахуванням рис

ідеї проекту, вимог покупців до товару і причин рекламної середовища (матриця 5.8 і 5.9) орієнтується і обґрунтовується Список причин конкурентоспроможності ніби Наведені в таблиці 5.10.

Таблиця 5.10 - Обґрунтування факторів конкурентоспроможності

<i>№ n/n</i>	<i>Фактор конкурентоспроможності</i>	<i>Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)</i>
1.	Підтримка різних мов програмування	Дозволяє не ламати голову над вибором інструментів, засобів та способів для роботи з бібліотекою
2.	Масштабованість та адаптивність	Розробник на основі вибраної концепції просто пише код.

“Відповідно до результатів аналізу факторів конкурентоспроможності продукту, наведеного у Таблиці 5.10 та необхідного для виходу на ринок, описаний в Таблиці 5.4, обґрунтовано головні фактори конкурентоспроможності, такі як кросплатформність, масштабованість та адаптивність, переваги яких також було розкрито в Таблицях 5.6 та 5.7. На основі результатів аналізу зазначених вище факторів конкурентоспроможності даного стартап-проекту проведено аналіз сильних та слабких його сторін, результат якого надано в Таблиці 5.11 за використання оцінки конкурентоспроможності за 20-бальною шкалою.”[26]

Таблиця 5.11 - Порівняльний аналіз сильних та слабких сторін проекту

<i>№ n/n</i>	<i>Фактор конкурентоспроможності</i>	<i>Бали 1-20</i>	<i>Рейтинг товарів-конкурентів у порівнянні з нашим підприємством</i>						
			-3	-2	-1	0	+1	+2	+3
1.	Підтримка декількох мов	15		+					

2.	Ефективність алгоритму	20						+	
3.	Невеликі розміри	18							+

“На основі SWOT-аналізу було розроблено альтернативи ринкової поведінки для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок.” [26]

Таблиця 5.12 – SWOT-аналіз стартап-проекту

Сильні сторони: простота використання, заощадження ресурсів, коштів на розгортання та підтримку ПЗ	Слабкі сторони: потрібно постійно оновлювати, підтримувати найновіші технології, аби уникнути застаріння концепції
Таргетована реклама, акцентування уваги на надійності системи, заохочення співробітників конкурентів до зміни компанії	Загрози: Надання додаткових переваг власного продукту лише за появи сильного конкурента, перегляд виконання умов, що зменшують податки, поступове підвищення тарифів

“На основі SWOT-аналізу розроблено альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок. Для визначених альтернатив виконано аналіз з точки зору строків та ймовірності отримання ресурсів, результати якого наведено в Таблиці 5.13. Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту”[26]

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової	Ймовірність отримання	Строки реалізації
----------	--	--------------------------	----------------------

	<i>поведінки</i>	<i>ресурсів</i>	
1.	Створення ПЗ використовуючи MLK	60%	10 місяців
2.	Створення ПЗ використовуючи LAPACK	70%	12 місяців

Відповідно до результатів аналізу з точки зору строків та ймовірності отримання ресурсів, наведеного у Таблиці 5.10, для реалізації проекту більш доцільним є вибір альтернативи 2 - Створення ПЗ використовуючи LAPACK, так як вона передбачає вищу ймовірність отримання ресурсів хоч і програє про строкам реалізації.

5.4 Розробка ринкової стратегії проекту

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

Таблиця 5.14 - Вибір цільових груп потенційних споживачів

№ n/n	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.	Науковці	Пришвидшення підрахунку великої кількості даних	великий	3 конкурента з продуктами відповідної функціональності, проте гіршої гнучкості	Зручність, швидкість розгортання та імплементаци продукту

2.	Розробника систем моделювання	Зменшення часу на розробку алгоритмів	середній		Зменшення витрат, часу та зусиль на підтримку ПЗ
Цільові групи обрано: обираємо стартап проекти та існуючі компанії					

“Підсумки розгляду цільових груп потенційних споживачів стартап-проекту, наведені в Таблиці 5.14, дозволяють визначити які переваги продукту вважаються можливістю використовувати для виходу на даний сегмент ринку, а ще чи доцільна витрата ресурсів для дії на конкретну категорію. Згідно підсумками розбору можливих груп покупців (частин) був обраний 2 головні цільові категорії, яким план стане запропонований для застосування. Для дієвого запровадження препарату в обраних групах зобов'язана існувати винайдена стратегія охоплення ринку.” [26]

Таблиця 5.15 – Визначення базової стратегії розвитку

№ n/n	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1.	Створення концепції побудови інформаційних систем	Ринкове позиціонування	Ефективність, швидкість, легкість використання	Диференціація

Таблиця 5.16 - Визначення базової стратегії конкурентної поведінки

№ n/n	Чи є проект «першопрохідцем»	Чи буде компанія шукати нових	Чи буде компанія копіювати основні	Стратегія конкурентної
----------	---------------------------------	----------------------------------	---------------------------------------	---------------------------

	<i>на ринку?</i>	<i>споживачів, або забирати існуючих у конкурентів?</i>	<i>характеристики товару конкурента, і які?</i>	<i>поведінки</i>
1.	Ні	Так	Ні	Зайняття конкурентної ніші

“За результатами визначення базової конкурентної поведінки, представленими в Таблиці 5.16, було обрано оптимальну стратегію конкурентної поведінки за використання основних характеристик-преваг конкурентів, описаних в Таблиці 5.2, для якої були зазначені наявність агресивного пошуку споживачі та зайняття конкурентної ніші як стратегія безпосередньо, що має надати можливість ефективної боротьби з вищезгаданими конкурентами. На основі вимог споживачів з двох основних сегментів, описаних в Таблиці 5.14, до стартап-компанії та до її системи сервісів, а також зважаючи на диференціальну базову стратегію розвитку та стратегії зайняття конкурентної ніші - стратегії конкурентної поведінки, - розроблено стратегію позиціонування, а саме сформовано ринкову позицію як комплекс асоціацій, для ідентифікації споживача торгівельної марки та проекту.” [26]

Таблиця 5.17 - Визначення стратегії позиціонування

<i>№ п/п</i>	<i>Вимоги до товару цільової аудиторії</i>	<i>Базова стратегія розвитку</i>	<i>Ключові конкурентоспро можні позиції стартап проекту</i>	<i>Вибір асоціацій, які мають сформувану комплексну позицію проекту</i>
1.	Простота реалізації, швидкість	Диференц іація	Швидкіст розгортання	Гнучкість розгортання,

	розгортання, легкість в підтримці		дозволить набути популярності на ринку	масштабування та підтримки систем
--	--------------------------------------	--	--	--------------------------------------

За результатами визначення стратегії позиціонування, представленими в Таблиці 5.17, було обрано оптимальну стратегію позиціонування стартап-проекту за орієнтації на основні вимоги до товару в цільовій аудиторії. З основних ключових конкурентоспроможних позиції стартап-проекту, наданих в Таблиці 5.15, в якості асоціацій було обрано дві, що задовольняють поставленим вимогам: кросплатформеність та мала вага.

Таблиця 5.18 - Визначення ключових переваг концепції потенційного товару

<i>№ n/n</i>	<i>Потреба</i>	<i>Вигода, яку пропонує товар</i>	<i>Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)</i>
1.	Підтримка декількох мов	Фреймворк надає інтерфейс для декількох мов програмування	Перевага у швидкості написання коду який використовує алгоритм
2.	Швидкість	Висока швидкість	Пришвидшення досліджень

“За результатами визначення ключових переваг концепції потенційного товару, представленими в Таблиці 5.18, було розглянуто основні потреби на ринку та проведено оцінку вигоди проекту. Було визначено ключові переваги перед конкурентами, що задовольняють наведеним основним потребам. В Таблиці 5.19 наведено трирівневу маркетингову модель товару. 1-й рівень. Задум товару - засобом вирішення якої потреби / проблеми буде даний товар, яка його основна вигода (основа технічного завдання). 2-й рівень. Рішення

реальної реалізації товару: якість, властивості, упаковка. 3-й рівень. Товар з підкріпленням (супроводом) - додаткові послуги та переваги для споживача, що створюються на основі товару за задумом і товару в реальному виконанні (гарантії якості, доставка, умови оплати та ін). Таблиця 5.19 - Опис трьох рівнів моделі товару.”[26]

Рівні товару	Сутність та складові		
Товар за задумом	Загальна концепція спрощує реалізацію інформаційних систем		
Товар у реальному виконанні	Властивості / характеристики	-	-
	1. Простота використання 2. Невеликі розміри 3. Підтримка різних мов програмування	1. Нм 2. Нм 3. Нм	1-3 Технологічна
	Якість: концепція є добре досліджена та відповідає світовим стандартам. Маркування відсутнє		
Товар із підкріпленням	Концепція не вимагає додаткових витрат		
	Постійна підтримка користувачів		
За рахунок патентування товар буде захищено від копіювання			

“Можна зробити висновок, що основні властивості товару у реальному виконанні це технічні, та не матеріальними.

Наступним кроком є визначення цінових меж експертним методом. Продукція є безкоштовна. Але кожен з користувачів буде мати змогу

підтримати її надіславши кошти. Щодо визначення оптимальної системи збуту, в межах якого приймається рішення – аналогічним чином.

Отже, система не приносить прибуток на постійній основі, але користувачі зможуть щомісячно вносити кошти для подовження досліджень у цій галузі та знаходження нових засобів та інструментів вирішення розріджених систем.

Останньою складовою маркетингової програми є розроблення концепції маркетингових комунікацій.” [26]

Таблиця 5.22 - Концепція маркетингових комунікацій

<i>№ п/ п</i>	<i>Специфіка поведінки цільових клієнтів</i>	<i>Канали комунікацій, якими користуються цільові клієнти</i>	<i>Ключові позиції, обрані для позиціонування</i>	<i>Завдання рекламного повідомлення</i>	<i>Концепція рекламного звернення</i>
1.	Використання за допомогою мережі Інтернет	Інтернет	Швидкість, підтримка багатьох мов програмування	Показати ефективність і швидкість роботи з бібліотекою	Короткий огляд концепції

Таким чином, в табл. 5.22 концепції маркетингових комунікацій, було визначено, що більшу інформацію про запропоновану концепцію буде здійснюватися через мережі Інтернет та у безкоштовному доступі.

4.6 Висновки

Було досліджено, що існує Можливість ринкової комерціалізації запропонованих Концепції. Наявний попит динаміка ринку, а також існують перспективи впровадження. З огляду на потенційні групи клієнтів, бар'єри входження не є значними, а проект має значні переваги перед існуючими

рішеннями.. В рамках даного дослідження були розраховані основні фінансово-економічні показники проекту, а також проведені менеджмент потенційних ризик.

Сильною стороною стартапу є висока швидкість роботи, відмовостійкість, підтримка різних мов програмування та мала вага самої бібліотеки. Слабкою ж стороною стартапу є наявність великих компаній-конкурентів, які поки що не дуже активно розвивають даний напрямок, але це може швидко змінитись.

ВИСНОВКИ

У роботі досліджено алгоритм SPIKE для вирішення СЛАР з діагонально-смуговою матрицею та розібрано його основні модифікації. Також, був реалізований сам алгоритм на мові Python з використанням фреймворків NumPy і SciPy.

Під час виконання роботи були досліджені сучасні стандарти розпаралелювання програм, їх переваги та недоліки. Зроблене припущення щодо доцільності використання комбінації OpenMP та MPI для підвищення ефективності роботи розподіленої системи.

Також, розібрані найпопулярніші бібліотеки для роботи з лінійною алгеброю, зокрема: GNU Scientific Library, Intel MKL, Sun Performance Library, Edge. Виявлено, що своїй швидкодії вони зобов'язані використанню низькорівневих бібліотек LAPACK і BLAS.

Розроблений алгоритм був протестований та порівняний з готовими реалізаціями вирішувачів на мові Python, а саме `numpy.linalg.solve()`, `scipy.linalg.solve()`, `scipy.sparse.linalg.spsolve()`. При тестуванні досліджувались наступні параметри роботи алгоритму:

- Швидкодія

Для аналізу даного показника були випадково згенеровані квадратні смугові матриці розмірами 1200, 1800, 2700, 4050, 6075, 9112, 13668, 20502, 30753 та відносними широтами смуги 1%, 2%, 4%, 8%, 16%, 32%. Виявлено, що розроблений алгоритм є дуже ефективним у вирішенні розріджених систем (<15%), а при ширині смуги 8%- 12% взагалі має найкращі показники серед усіх обраних алгоритмів.

- Використання пам'яті

Для аналізу даного показника були випадково згенеровані квадратні смугові матриці розмірами 5000, 8000, 18000, 22000 та відносними широтами смуги 1%, 4%, 16%. З графіків видно, що при розріджених матрицях, споживання пам'яті у десятки разів менше, аніж у звичайних алгоритмів та приблизно однакове з функціями для роботи з розрідженими матрицями, але при збільшенні товщини смуги ($\approx 20\%$) алгоритм починає споживати значно більше пам'яті та втрачати свої переваги.

- Ефективне розділення вхідних даних

Для цього аналізу дані були обрані аналогічно попереднім та зроблений висновок, що критичним показником при виборі кількості розділів, як і в попередніх тестах є ширина смуги (тобто ступінь розрідженості), із графіків чітко видно, що при збільшенні ширини смуги, ефективна кількість розділів (як для часу роботи, так і для споживання пам'яті) зменшується, що суттєво погіршує здатність алгоритму до розпаралелювання.

Стосовно розробленого алгоритму можна зробити висновок, що він є дуже ефективним для обчислення СЛАР з смуговими матрицями, ширина смуги яких $< 12\%$ відносно розміру матриці. Саме при таких параметрах алгоритм видає найкращі результати по всіх параметрах, а у проміжку 7%-12% навіть показує найкращі результати із всіх протестованих алгоритмів.

До мінусів алгоритму можна віднести сильну залежність від ступеню розрідженості матриці, а також роботу алгоритму тільки з конкретним типом матриць (смуговими).

Подальша робота може бути пов'язана з реалізацією та аналізом різних модифікацій алгоритму, наприклад: рекурсивного SPIKE алгоритму, SPIKE алгоритму з використанням ітеративних методів вирішення СЛАР.

Було проведено дослідження щодо можливості ринкової комерціалізації проекту. В ході складання документації до проекту було сформовано ідею стартап проекту “SPIKE алгоритм”, наведено порівняння із рядом потенційних конкурентів, що надало можливість оцінки конкурентоспроможності та можливості і складності виходу стартапу на ринок. Було виконано обґрунтування факторів конкурентоспроможності та порівняльний аналіз сильних та слабких сторін проекту, результати яких пізніше були використані для SWOT-аналізу стартап-проекту. В рамках даного дослідження були розраховані основні фінансово-економічні показники проекту, а також проведений менеджмент потенційних ризиків.

Отже, в роботі розрита науково-прикладна проблема: використання SPIKE алгоритму.

Відповідно до поставленої мети було:

- Проаналізований SPIKE алгоритм та його модифікації.
- Проаналізовані існуючі інструменти для вирішення такого типу задач.
- Розглянуті інструменти для ефективної реалізації алгоритму.
- Розроблено власну імплементацію алгоритму
- Проведене порівняння розробленого алгоритму з вже реалізованими, високоефективними алгоритмами.

Запропоноване рішення показує найкращі результати із протестованих при певних, визначених у роботі умовах і може використовуватись як бібліотечна функція.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, et al. LAPACK users guide. 3rd ed. Philadelphia, PA: Society for Industrial and Appl. Math.; (1999).
- [2] A. Cleary and J. Dongarra. Implementation in ScaLAPACK of divide and conquer algorithms for banded and tridiagonal linear systems. University of Tennessee Computer Science Technical Report, UT-CS-97-358; (1997).
- [3] S.C. Chen, D.J. Kuck, and A. Sameh, Practical parallel band triangular system solvers, ACM Trans. Math. Software, 4, pp. 270-277 (1978).
- [4] L.S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, et al. ScaLAPACK users guide. Philadelphia, PA: Society for Industrial and Appl. Math.; (1997).
- [5] P. Arbenz, A. Cleary, J. Dongarra, and M. Hegland, A comparison of parallel solvers for diagonally dominant and general narrow banded linear systems II, EuroPar Parallel Processing, p. 1078-1087; (1999)
- [6] A. Sameh. Numerical parallel algorithms: a survey. In: Kuck D, Lawrie D, Sameh A, editors. High speed computer and algorithm organization. New York: Academic Press; p. 207-228 (1977).
- [7] A. Sameh and D. Kuck. On stable parallel linear system solvers. J. ACM, 25, p. 81-91; (1978).
- [8] A. Sameh. On two numerical algorithms for multiprocessors. Proc of NATO advanced workshop on high-speed comp. Series F: computer and systems sciences, vol. 7. Berlin: Springer; p. 31-28, (1983).
- [9] D. Lawrie and A. Sameh. The computation and communication complexity of a parallel banded system solver. ACM Trans Math Software, 10(2); p. 185-195; (1984).

- [10] J. Dongarra and A. Sameh. On some parallel banded system solvers. *Parallel Computing*, 1, p. 223-235; (1984).
- [11] J.-L. Larriba-Pey, A. Jorba, and J.J. Navarro, Spike algorithm with savings for strictly diagonal dominant tridiagonal systems, *Microprocessing and Microprogramming*, 39, pp. 125-128 (1993).
- [12] D.H. Lawrie and A. Sameh, The computation and communication complexity of a parallel banded system solver, *ACM Trans. Math. Software*, 10, pp. 185-195 (1984).
- [13] M. Berry and A. Sameh. Multiprocessor schemes for solving block tridiagonal linear systems. *Int J. Supercomput. Appl.*, 2(3): p. 37-57; (1988).
- [14] N.J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, PA, (2002).
- [15] A. Sameh, and V. Sarin. Hybrid Parallel Linear Solvers. *International Journal of Computational Fluid Dynamics*, Vol 12, p. 213-223, (1999).
- [16] E. Polizzi, A. Sameh. A Parallel Hybrid Banded System Solver: The SPIKE Algorithm, *Parallel Computing*, V. 32, 2, p. 177-194 (2006).
- [17] E. Polizzi, A. Sameh. SPIKE: A parallel environment for solving banded linear systems, *Computers & Fluids*, 36 p. 113-120 (2007).
- [18] A distributed memory version of the SPIKE package can be obtained from: <http://software.intel.com/en-us/articles/intel-adaptive-spike-based-solver/>
- [19] E. Polizzi. The SPIKE software. Book Chapter in *Springer Encyclopedia of Parallel Computing*, D. Padua (Ed.), Springer, to appear. (2010).
- [20] O. Schenk and K. Gartner. Solving Unsymmetric Sparse Systems of Linear Equations with PARDISO. *Journal of Future Generation Computer Systems*, 20(3), p.

475-487; (2004).

[21] S. Demko, W.F. Moss and P.W. Smith. Decay rates for inverses of band matrices. Math Comput., 43(168), p. 491-499; (1984).

[22] C.C.K. Mikkelsen and M. Manguoglu. Analysis of the Truncated Spike Algorithm. SIAM Journal on Matrix Analysis and Applications, 30(4), p.1500-1519; (2008).

[23] <http://www.ecs.umass.edu/~polizzi/feast>; (2009).

[24] X.-H. Sun, H. Zhang, and L.M. Ni, Efficient tridiagonal solvers on multicomputers, IEEE Trans. Comput., 41, p. 286296 (1992).

[25] X.-H. Sun, Application and accuracy of the parallel diagonal dominant algorithm, Parallel Comput., 21, p. 12411267 (1995).

[26] http://cad.kpi.ua/attachments/475_Pipich_magistr.pdf