

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ  
ІНСТИТУТ КІБЕРНЕТИКИ ІМЕНІ В.М. ГЛУШКОВА

Кваліфікаційна наукова  
праця на правах рукопису

Павлюк Антон Володимирович

УДК 519.6

ДИСЕРТАЦІЯ

**ВИСОКОПРОДУКТИВНІ ОБЧИСЛЕННЯ В ЗАДАЧАХ  
МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ ДЛЯ ГІБРИДНИХ  
КОМП'ЮТЕРНИХ АРХІТЕКТУР**

113 – «Прикладна математика»

Галузь знань 11 – «Математика та статистика»

Подається на здобуття наукового ступеня доктора філософії

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

Павлюк А.В.

Науковий керівник:

Попов Олександр Володимирович  
доктор фізико-математичних наук,  
старший науковий співробітник

Київ – 2026

## АНОТАЦІЯ

Павлюк А.В. Високопродуктивні обчислення в задачах математичного моделювання для гібридних комп'ютерних архітектур. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття ступеня доктора філософії за спеціальністю 113 “Прикладна математика”. – Інститут кібернетики імені В.М. Глушкова Національної академії наук України, Київ. – 2026.

**Зміст дисертації.** Дисертаційна робота складається зі вступу, чотирьох розділів, загальних висновків та списку використаних джерел.

У вступі обґрунтовано актуальність теми дослідження, сформульовано мету та завдання роботи, визначено об’єкт і предмет дослідження, наведено наукову новизну та практичне значення отриманих результатів, а також подано відомості про апробацію результатів дисертації.

У першому розділі проведено огляд сучасних методів і алгоритмів розв’язування систем лінійних алгебраїчних рівнянь великих порядків, що виникають у задачах математичного моделювання. Розглянуто особливості математичних моделей з наближеними даними, досліджено можливості ефективного використання матриць різної структури та проаналізовано сучасні бібліотеки і програмні засоби чисельної лінійної алгебри для високопродуктивних комп’ютерних систем.

У другому розділі досліджено методи розв’язування систем лінійних алгебраїчних рівнянь на сучасних гібридних комп’ютерних архітектурах. Розроблено паралельний алгоритм LU-факторизації щільних матриць на основі двовимірного блочно-циклічного розподілу даних для систем із кількома графічними прискорювачами та проведено експериментальні дослідження ефективності запропонованого підходу.

У третьому розділі розроблено плитковий алгоритм факторизації несиметричних стрічкових матриць. Описано структуру алгоритму, виконано оцінку кількості арифметичних операцій, а також отримано оцінки

прискорення та ефективності алгоритму при виконанні паралельних обчислень.

У четвертому розділі розглянуто застосування розроблених методів високопродуктивних обчислень у задачах математичного моделювання технічних конструкцій. Досліджено чисельний аналіз напружено-деформованого стану зварних конструкцій з дефектами локальної втрати металу, запропоновано варіаційно-операторну модель керованої дискретизації та проведено тестування запропонованих підходів.

На підставі проведеного дослідження у розділі висновки узагальнено основні результати дисертаційної роботи та сформульовано основні наукові й практичні результати дослідження.

**Ключові слова:** паралельні алгоритми, розріджені матриці, комп'ютери гібридної архітектури, високопродуктивні обчислення, математичне моделювання.

## ABSTRACT

**Pavliuk A.V. High-Performance Computing in Mathematical Modeling Problems for Hybrid Computer Architectures.** – Qualifying scientific work as a manuscript.

Dissertation for the degree of Doctor of Philosophy in specialty 113 “Applied Mathematics”. – V.M. Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine, Kyiv. – 2026.

**Content of the dissertation.** The dissertation consists of an introduction, four chapters, general conclusions, and a list of references.

**The introduction** substantiates the relevance of the research topic, formulates the aim and objectives of the work, defines the object and subject of the research, presents the scientific novelty and practical significance of the obtained results, and provides information on the approbation of the dissertation results.

**The first chapter** reviews modern methods and algorithms for solving large-scale systems of linear algebraic equations arising in mathematical modeling problems. The features of mathematical models with approximate data are considered, the possibilities of efficient use of matrices with different structures are investigated, and modern libraries and software tools of numerical linear algebra for high-performance computer systems are analyzed.

**The second chapter** investigates methods for solving systems of linear algebraic equations on modern hybrid computer architectures. A parallel LU factorization algorithm for dense matrices based on a two-dimensional block-cyclic data distribution for systems with multiple graphics accelerators is developed, and experimental studies of the efficiency of the proposed approach are carried out.

**The third chapter** develops a tiled algorithm for the factorization of nonsymmetric band matrices. The structure of the algorithm is described, the number of arithmetic operations is estimated, and estimates of the speedup and efficiency of the algorithm during parallel computations are obtained.

**The fourth chapter** considers the application of the developed high-performance computing methods to problems of mathematical modeling of

engineering structures. A numerical analysis of the stress-strain state of welded structures with local metal loss defects is performed, a variational-operator model of controlled discretization is proposed, and the developed approaches are tested.

Based on the research conducted, the conclusions **section summarizes** the main results of the dissertation work and formulates the main scientific and practical results of the research.

**Keywords:** parallel algorithms, sparse matrices, hybrid-architecture computers, high-performance computing, mathematical modeling.

## СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

### *Публікації, в яких відображено основні наукові результати дисертації*

Sydooruk Volodymyr, Anton Pavliuk, Optasyuk Sergey, Heseleva Kateryna. Algorithm for Solving Systems with Band Matrices in the Problems of Predicting the Service Life of Welded Structures. // Scientific Journal Mathematical and computer modelling. Series: Physical and mathematical sciences. - ISSUE 26 - Kamianets-Podilskyi Ivan Ohienko National University, 2024. - P. 62-72. DOI: 10.32626/2308-5878.2024-26.62-72

Олександр Попов, Антон Павлюк. Варіаційно-операторна модель керованої дискретизації для гібридних обчислень // Науковий вісник Ужгородського університету. Серія: Математика і інформатика. Том 47 №2, 2025, С.231-243. DOI: 10.24144/2616-7700.2025.47(2).231-243

Олександр Хіміч, Володимир Сидорук, Антон Павлюк. MULTI-GPU двовимірний блочно-циклічний алгоритм LU-факторизації щільних матриць. // Збірник наукових праць «Кібернетика та комп'ютерні технології» Інституту кібернетики імені В.М. Глушкова НАН України № 4 (24). – 2025. - С. 55-64. DOI: 10.34229/2707-451X.25.4.6

Alexandr Popov, Anton Pavliuk. Multiobjective memory optimization in mathematical modeling for hybrid computer architectures // Scientific Journal Mathematical Modeling. Dniprovsky State Technical University № 2(53), 2025, P.38-49. DOI: 10.31319/2519-8106.2(53)2025.342456

### *Публікації, що засвідчують апробацію матеріалів дисертації*

Володимир Сидорук Антон Павлюк Деякі способи використання паралельних обчислень в прикладних задачах // Сучасні проблеми математичного моделювання, прогнозування та оптимізації: тези доповідей 10-ї Міжнародної наукової конференції. Пам'яті почесного професора Кам'янець-Подільського національного університету імені Івана Огієнка, д.т.н., професора, почесного академіка НАПНУ Анатолія Федоровича ВЕРЛАНЯ [Електронний ресурс] <https://cs.kpnu.edu.ua/optima/>. Кам'янець-

Подільський: Кам'янець-Подільський національний університет імені Івана Огієнка, 2024. – С. 35-37

Олександр Попов, Антон Павлюк Варіаційні механізми адаптивного керування продуктивністю в гібридних системах // Матеріали XIII Міжнародної науково-практичної конференції Математичне та програмне забезпечення інтелектуальних систем 2025 - Дніпровський національний університет імені Олеся Гончара 2025, с. 267-268

***Публікації, які додатково відображають наукові результати дисертації***

Володимир Сидорук, Олексій Чистяков, Антон Павлюк Паралельний алгоритм  $LDL^T$  розвинення для задач механіки // Матеріали Міжнародної наукової конференції “Актуальні проблеми механіки” до 145-річчя від дня народження С.П. Тимошенка [Електронний ресурс] [https://inmech.kyiv.ua/doc/news/2023/actual\\_problems\\_of\\_mechanics/program.pdf](https://inmech.kyiv.ua/doc/news/2023/actual_problems_of_mechanics/program.pdf) - Інститут механіки імені С.П. Тимошенка НАН України, 2023, с. 463-464

Олександр Дученко, Алла Нестеренко, Антон Павлюк До комп'ютерного моделювання нестационарних процесів // Матеріали XIII міжнародної науково-практичної конференції. «ГЛУШКОВСЬКІ ЧИТАННЯ» СУЧАСНА КІБЕРНЕТИКА 2024. Київ. 2024 рік.

Антон Павлюк Моделювання резильєнтності складних систем на гібридних архітектурах // Матеріали III науково-практичної конференції "Резильєнтність динамічних систем". - Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова, 2025. - С. 34-37.

## ЗМІСТ

|   |    |
|---|----|
| ВСТУП   | 10 |
| РОЗДІЛ 1. Теоретичні засади побудови високопродуктивних алгоритмів розв’язування систем лінійних алгебраїчних рівнянь.....                    | 18 |
| 1.1 Постановка та дослідження математичних моделей з наближеними даними.....  | 18 |
| 1.2 Застосування методів та алгоритмічно-програмного забезпечення для розв’язування СЛАР.....   | 24 |
| 1.3 Особливості створення адаптивних методів та алгоритмів розв’язування СЛАР для гібридних комп’ютерів.....                                  | 26 |
| 1.4 Інтелектуалізація алгоритмічно-програмного забезпечення високопродуктивних обчислень з лінійної алгебри на сучасних суперкомп’ютерах..... | 31 |
| 1.5 Постановка завдання дослідження.....  | 34 |
| 1.6 Висновки до розділу.....  | 35 |
| РОЗДІЛ 2. Паралельні алгоритми $LU$ -факторизації для розв’язування СЛАР на гібридних комп’ютерних системах.....                              | 36 |
| 2.1 Комп’ютерне дослідження лінійних систем з наближеними даними.....   | 37 |
| 2.2 Двовимірний блочно-циклічний мульти-GPU алгоритм $LU$ факторизації.....   | 46 |
| 2.3 Апробація розробленого паралельного алгоритму.....  | 56 |
| 2.4 Висновки до розділу.....  | 61 |
| РОЗДІЛ 3. Блочний алгоритм розвинення стрічкових матриць.....   | 62 |
| 3.1 Постановка задачі.....  | 63 |
| 3.2. Блочний алгоритм.....  | 64 |
| 3.2 Прискорення та ефективності алгоритму.....  | 66 |

|  |     |
|--|-----|
| 3.3 Апробація алгоритму .....  | 69  |
| 3.4 Висновки до розділу .....  | 72  |
| РОЗДІЛ 4. Методи математичного моделювання конструкцій на<br>гібридних комп'ютерних системах .....                   | 74  |
| 4.1 Постановка задачі дослідження життєвого циклу<br>зварних конструкцій .....                                       | 74  |
| 4.2 Дефекти локальної втрати металу трубопроводів,<br>методи їх схематизації та критерії допустимості .....          | 77  |
| 4.3 Чисельний аналіз напружено-деформованого стану<br>відповідальних зварних конструкцій з виявленими дефектами..... | 79  |
| 4.4 Апробація алгоритму .....  | 88  |
| 4.5 Варіаційно-операторна модель керованої дискретизації<br>для гібридних обчислень. ....                            | 92  |
| 4.6 Варіаційно-операторний підхід до керованої дискретизації. ....   | 94  |
| 4.7 Енергетично узгоджена декомпозиція та стабільність .....   | 96  |
| 4.8 Варіаційна оптимізація точності та часу обчислень.....   | 101 |
| 4.9 Тестування моделі .....  | 107 |
| 4.10 Висновки до розділу. ....   | 110 |
| ВИСНОВКИ.....  | 111 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....  | 113 |

## ВСТУП

Сучасний розвиток науки і техніки характеризується широким використанням математичного моделювання складних фізичних, технічних та інженерних процесів. Значна частина таких моделей після дискретизації диференціальних або інтегральних рівнянь зводиться до розв'язування систем лінійних алгебраїчних рівнянь (СЛАР) великої розмірності. Подібні системи виникають у задачах механіки суцільних середовищ, теплопровідності, гідродинаміки, електродинаміки, моделювання конструкцій у будівництві та машинобудуванні, а також у задачах обробки великих обсягів даних.

У сучасних задачах математичного моделювання порядки систем лінійних рівнянь можуть досягати сотень тисяч і навіть мільйонів невідомих. При цьому матриці систем, як правило, мають розріджену структуру та складну топологію ненульових елементів. Ефективне розв'язування таких систем є однією з ключових проблем обчислювальної математики та високопродуктивних обчислень.

Розвиток сучасних обчислювальних систем характеризується переходом до багатопроцесорних архітектур, що поєднують багатоядерні центральні процесори (CPU) та графічні прискорювачі (GPU). Зокрема, новий вид архітектур високопродуктивних обчислювальних систем дозволяє встановлювати у вузлі до 8 графічних процесорів, так звані мульти-GPU архітектури. Такі гібридні обчислювальні системи дозволяють досягати надзвичайно високої продуктивності, однак водночас ставлять нові вимоги до алгоритмів чисельної лінійної алгебри. Традиційні алгоритми [1-15], розроблені для послідовних обчислювальних моделей, не забезпечують належної масштабованості на сучасних паралельних архітектурах комп'ютерів [16-26]. Основними обмежувальними факторами є високі витрати на міжпроцесорні комунікації, нерівномірність розподілу обчислювального навантаження між обчислювальними пристроями та недостатньо ефективне використання багаторівневої ієрархії пам'яті GPU.

У зв'язку з цим особливо актуальною є проблема розроблення та дослідження нових паралельних алгоритмів розв'язування систем лінійних алгебраїчних рівнянь, здатних ефективно використовувати обчислювальні ресурси сучасних високопродуктивних комп'ютерних систем, зокрема мульти-GPU архітектури. Значний інтерес у цьому напрямку досліджень становлять блочні алгоритми факторизації матриць, а також підходи, що базуються на використанні двовимірних та одновимірних блочно-циклічних схем розподілу даних між обчислювальними пристроями. Значний розвиток блочних алгоритмів факторизації матриць пов'язаний з роботами A. Buttari, J. Kurzak, J. Dongarra, де запропоновано клас tiled-алгоритмів для багатоядерних та гібридних архітектур [27–36].

Особливого значення набуває створення ефективних алгоритмів факторизації для структурованих розріджених матриць, зокрема стрічкових, які часто виникають у задачах математичного моделювання складних інженерних систем. Використання обчислювальних систем гібридних архітектур із багатьма графічними процесорами (мульти-GPU) [16-21] відкриває нові можливості для прискорення таких обчислень, однак потребує створення спеціалізованих алгоритмів факторизації, орієнтованих на ефективне використання паралелізму, локальності даних та ієрархії пам'яті сучасних високопродуктивних систем.

Додатковою складністю при розв'язуванні систем лінійних рівнянь у задачах математичного моделювання є те, що вихідні дані задані наближено. Похибки вимірювань, дискретизації та заокруглення при комп'ютерних обчисленнях можуть істотно впливати на достовірність отриманих результатів. Тому важливим напрямом досліджень є створення методів аналізу математичних властивостей комп'ютерних моделей та алгоритмів, здатних забезпечити необхідну точність розв'язку.

Отже, розроблення нових паралельних алгоритмів факторизації щільних та стрічкових матриць, орієнтованих на задачі математичного моделювання з наближеними даними та ефективне використання сучасних гібридних

суперкомп'ютерних систем, є актуальною науковою задачею сучасної прикладної математики. Її розв'язання спрямоване на скорочення часу чисельних експериментів, підвищення масштабованості обчислень, забезпечення достовірності отриманих результатів і розширення класу прикладних задач, які можуть бути ефективно розв'язані на сучасних високопродуктивних обчислювальних комплексах. Саме це визначає мету та основні завдання дисертаційного дослідження.

**Мета дисертаційної роботи** — розроблення та дослідження методів та комп'ютерних алгоритмів високопродуктивних обчислень для задач математичного моделювання на комп'ютерах гібридної та мульти-GPU архітектури.

Для досягнення поставленої мети у роботі необхідно розв'язати такі завдання:

- провести аналіз існуючих методів та алгоритмів розв'язування систем лінійних алгебраїчних рівнянь великих порядків на сучасних високопродуктивних обчислювальних системах;
- дослідити особливості розв'язування систем лінійних алгебраїчних рівнянь із щільними, стрічковими та розрідженими матрицями в задачах математичного моделювання з наближеними вихідними даними;
- розробити паралельний алгоритм LU-факторизації щільних матриць на основі двовимірного блочно-циклічного розподілу даних для гібридних обчислювальних систем, мульти-GPU;
- розробити ефективний блочний алгоритм факторизації стрічкових несиметричних матриць з урахуванням особливостей їх структури та мульти-GPU архітектури;
- виконати теоретичний аналіз обчислювальної складності запропонованих алгоритмів, отримати оцінки часу виконання та прискорення;

- провести апробацію розроблених алгоритмів та експериментальне дослідження ефективності в задачах математичного моделювання складних конструкцій.

**Об'єкт дослідження** — високопродуктивні паралельні обчислення для комп'ютерних систем гібридної та мульти-GPU архітектури.

**Предмет дослідження** — методи та комп'ютерні алгоритми високопродуктивних паралельних обчислень для розв'язування задач лінійної алгебри з використанням комп'ютерних систем гібридних та мульти-GPU архітектур.

**Методи дослідження** — методи лінійної алгебри, методи паралельних обчислень, методи математичного моделювання.

**Наукова новизна** полягає у розробленні та теоретичному обґрунтуванні нових паралельних методів і комп'ютерних алгоритмів високопродуктивних обчислень для задач лінійної алгебри, орієнтованих на ефективне використання ресурсів сучасних гібридних і мульти-GPU обчислювальних архітектур.

Зокрема:

- вперше розроблено паралельний алгоритм LU-факторизації щільних матриць на основі двовимірного блочно-циклічного розподілу даних, адаптований до виконання на мульти-GPU архітектурах, що забезпечує рівномірне завантаження графічних прискорювачів і зменшення комунікаційних витрат між ними;
- вперше запропоновано блочно-циклічний алгоритм факторизації несиметричних стрічкових матриць, який враховує зменшення обсягів обчислень і комунікацій на мульти-GPU;
- набули подальшого розвитку методи організації паралельних обчислень для розв'язування систем лінійних алгебраїчних рівнянь великих порядків на гібридних комп'ютерних системах, зокрема за рахунок узгодження схеми розподілу даних, блокової структури алгоритму та особливостей взаємодії CPU і GPU;

- одержано теоретичні оцінки обчислювальної складності, часу виконання, прискорення та ефективності запропонованих паралельних алгоритмів факторизації, які дають змогу обґрунтувати умови їх масштабованості на мульти-GPU системах;
- проведено апробацію розроблених алгоритмів та експериментальне дослідження ефективності в задачах математичного моделювання міцності зварних конструкцій.

**Практичне значення одержаних результатів** полягає в тому, що запропоновані паралельні алгоритми та розроблені на їх основі програмні засоби можуть бути використані при створенні високопродуктивних систем математичного моделювання складних фізичних, технічних і технологічних процесів, чисельна реалізація яких зводиться до розв'язування систем лінійних алгебраїчних рівнянь великих порядків зі щільними, стрічковими та розрідженими матрицями.

Розроблені алгоритми пройшли апробацію та експериментальне дослідження ефективності на задачах математичного моделювання процесів зварювання та споріднених технологій. Отримані результати підтвердили доцільність використання запропонованих підходів для скорочення часу чисельних експериментів, підвищення продуктивності обчислень і забезпечення ефективного використання ресурсів сучасних гібридних та мульти-GPU обчислювальних систем.

Розроблене програмне забезпечення може бути застосоване у наукових дослідженнях, інженерних розрахунках, системах комп'ютерного моделювання складних конструкцій і технологічних процесів, а також при виконанні ресурсоємних обчислень на сучасних суперкомп'ютерних комплексах.

**Зв'язок роботи з науковими програмами, планами, темами.** Дисертаційна робота виконана у відповідності до планів наукових досліджень відділу № 150 Інституту кібернетики імені В.М. Глушкова НАН України в рамках науково-дослідницьких тем:

ВФ.150.26 «Розробити моделі та методи гетерогенних обчислень для задач механіки суцільних середовищ» (№ держреєстрації 0119U002224, 2019–2023 рр.);

ВП.150.4.1230 «Розробити платформу високопродуктивних обчислень на базі суперкомп'ютера СКІТ для задач кібербезпеки, математичного моделювання, інженерії» (№ держреєстрації 0123U101573, 2023 р.);

**Особистий внесок здобувача.** Основні наукові положення, результати та висновки дисертаційної роботи отримані здобувачем особисто. У наукових працях, виконаних у співавторстві, здобувачеві належать постановка задач, розроблення алгоритмічних підходів, програмна реалізація та аналіз отриманих результатів.

У роботі «Algorithm for Solving Systems with Band Matrices in the Problems of Predicting the Service Life of Welded Structures» [58] здобувачем запропоновано підхід до збереження стрічкової структури матриці в процесі LU-факторизації, розроблено алгоритмічну схему обчислень, виконано програмну реалізацію та проведено чисельні експерименти з аналізом точності та обчислювальної ефективності. У роботах «Варіаційно-операторна модель керованої дискретизації для гібридних обчислень» та «Варіаційні механізми адаптивного керування продуктивністю в гібридних системах» здобувачем досліджено можливості застосування запропонованих підходів у задачах гібридних обчислень, обґрунтовано їх інтеграцію з методами розв'язування систем лінійних алгебраїчних рівнянь, а також виконано аналіз впливу параметрів дискретизації на обчислювальну складність, масштабованість та ефективність паралельних алгоритмів.

У роботі «MULTI-GPU двовимірний блочно-циклічний алгоритм LU-факторизації щільних матриць» [20] здобувачем розроблено двовимірний блочно-циклічний алгоритм факторизації щільних матриць, запропоновано ефективну схему розподілу даних і обчислень між графічними прискорювачами, реалізовано алгоритм із використанням технологій CUDA та бібліотек cuBLAS/cuBLASXt, а також проведено експериментальні

дослідження продуктивності та масштабованості на багатопроцесорних обчислювальних системах. У роботі «Multiobjective memory optimization in mathematical modeling for hybrid computer architectures» [60] внесок здобувача полягає у формалізації багатокритеріального підходу до оцінювання витрат пам'яті, дослідженні способів зменшення обсягів збережуваних даних під час реалізації чисельних алгоритмів.

У матеріалах конференцій [61, 62] здобувачем представлено результати дослідження застосування паралельних обчислень у прикладних задачах, виконано аналіз ефективності гібридних обчислювальних підходів та досліджено вплив параметрів алгоритмів на продуктивність обчислень. У «Паралельний алгоритм  $LDL^T$  розвинення для задач механіки» [63] здобувачем запропоновано підхід до побудови паралельного алгоритму  $LDL^T$ -факторизації для задач механіки. У роботі «До комп'ютерного моделювання нестационарних процесів» [64] досліджено особливості комп'ютерного моделювання нестационарних процесів. У «Моделювання резильєнтності складних систем на гібридних архітектурах» [65] розроблено підходи до моделювання резильєнтності складних систем на гібридних обчислювальних архітектурах.

Узагальнюючи отримані результати, слід зазначити, що здобувачем розроблено нові паралельні алгоритми LU- та  $LDL^T$ -факторизації для матриць спеціальної структури, запропоновано двовимірні блочно-циклічні схеми розподілу даних для multi-GPU систем, розвинуто підходи до збереження структурних властивостей матриць при факторизації, створено ефективні програмні реалізації з використанням сучасних GPU-технологій та проведено комплексні експериментальні дослідження продуктивності, масштабованості та точності розроблених алгоритмів, що підтверджують їх ефективність у задачах математичного моделювання.

**Апробація результатів дисертації.** Результати дисертації доповідались та обговорювались на:

- Міжнародній науковій конференції "Актуальні проблеми механіки - 2023" до 145-річчя від дня народження С.П.Тимошенка; Київ, Дніпро, Львів, Харків – 14-16 листопада 2023 р.
- 10-тій міжнародній науковій конференції пам'яті д.т.н., проф А.Ф. Верляня; Кам'янець-Подільський національний університет імені Івана Огієнка 28-29 червня 2024 р.
- XIII Міжнародній науково-практичній конференції "Глушковські читання" Тема: «Сучасна кібернетика 2024»; Інститут кібернетики імені В.М. Глушкова 06 грудня 2024 р.
- III науково-практичній конференції "Резильєнтність динамічних систем"; Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова 12 червня 2025 р.
- XIII Міжнародній науково-практичній конференції «Математичне та програмне забезпечення інтелектуальних систем 2025»; Дніпровський національний університет імені Олеся Гончара 19-21 листопада 2025 р.

**Публікації.** Основні наукові результати дисертаційної роботи у повній мірі викладено у 9 публікаціях, 4 наукові статті опубліковано у фахових виданнях України.

**Структура та обсяг дисертації.** Дисертаційна робота складається зі вступу, чотирьох розділів, загальних висновків, списку використаних літературних джерел, який містить 116 найменувань. Загальний обсяг дисертаційних досліджень викладено на 125 сторінках друкованого тексту, де обсяг основного тексту – 109 сторінок. Дисертація включає 8 рисунків, 7 таблиць.

# **РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ ПОБУДОВИ ВИСОКОПРОДУКТИВНИХ АЛГОРИТМІВ РОЗВ'ЯЗУВАННЯ СИСТЕМ ЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ**

У розділі проведено аналіз проблеми розв'язування систем лінійних алгебраїчних рівнянь великих порядків, що виникають після дискретизації задач математичного моделювання. Розглянуто особливості математичних моделей з наближеними даними, досліджено можливості ефективного використання матриць різної структури, проаналізовано сучасні методи, алгоритми та бібліотеки програм лінійної алгебри. Окрему увагу приділено особливостям побудови адаптивних методів і інтелектуалізації алгоритмічно-програмного забезпечення для гібридних високопродуктивних комп'ютерних систем. Отримані результати аналізу визначають концептуальні засади дисертаційного дослідження та слугують підґрунтям для розроблення нових паралельних алгоритмів.

## **1.1 Постановка та дослідження математичних моделей з наближеними даними**

У більшості задач математичного моделювання процесів у фізиці, механіці, будівництві, гідродинаміці, теплопровідності, електродинаміці та інших галузях науки і техніки використовуються математичні моделі, що описуються диференціальними, інтегральними або інтегро-диференціальними рівняннями. У процесі чисельного розв'язування таких задач виконується дискретизація математичної моделі, внаслідок чого отримуються системи лінійних алгебраїчних рівнянь.

Загальний вигляд такої системи має форму

$$Ax = b,$$

де  $A$  — матриця коефіцієнтів системи,  $x$  — вектор розв'язку,  $b$  — вектор правої частини.

У практичних задачах елементи матриці  $A$  та вектора  $b$  часто визначаються на основі експериментальних даних, результатів вимірювань або наближених математичних моделей. У зв'язку з цим вихідні дані задачі містять похибки [37], які виникають внаслідок похибок вимірювань, апроксимації математичної моделі, дискретизації, а також заокруглень під час комп'ютерних обчислень. Тому реальна комп'ютерна модель задачі відрізняється від її аналітичної постановки, а розв'язування системи фактично здійснюється для задачі з наближеними даними.

Нехай система  $Ax = b$  задана з відносними похибками вихідних даних, які характеризуються величинами  $\varepsilon_A$  та  $\varepsilon_b$ , де  $\varepsilon_A$  — відносна похибка елементів матриці  $A$ , а  $\varepsilon_b$  — відносна похибка елементів вектора правої частини  $b$ . У такому випадку однією з центральних проблем чисельного аналізу є дослідження впливу цих похибок на точність і достовірність обчисленого розв'язку [38-43].

Відомо, що навіть незначні зміни елементів матриці можуть призводити до суттєвих змін розв'язку системи. Це особливо характерно для погано обумовлених задач. Для оцінки чутливості розв'язку до похибок вихідних даних використовується число обумовленості матриці

$$\text{cond}(A) = \|A\| \cdot \|A^{-1}\|.$$

Якщо значення  $\text{cond}(A)$  є великим, то система вважається погано обумовленою, а отже, навіть малі збурення вихідних даних можуть призводити до значних похибок у знайденому розв'язку.

У задачах математичного моделювання часто виникають ситуації, коли математична модель має одні властивості, а її комп'ютерна реалізація — інші. Наприклад, матриця, яка в теоретичній постановці є невинродженою, у комп'ютерній моделі може виявитися близькою до винродженої внаслідок похибок заокруглення або наближеного характеру вихідних даних. У зв'язку з

цим важливо досліджувати не лише математичну модель задачі, а й її машинну реалізацію, що враховує обмежену розрядність комп'ютерної арифметики.

Особливістю сучасних комп'ютерних обчислень є використання стандартів подання чисел з плаваючою комою [44-48], зокрема форматів одинарної, подвійної та підвищеної точності. Проте у багатьох практичних задачах стандартної подвійної точності може бути недостатньо для забезпечення необхідної достовірності результатів. У таких випадках застосовуються методи багаторозрядної арифметики, алгоритми зі змінною розрядністю, а також підходи змішаної точності [49-53].

Ще однією характерною особливістю задач математичного моделювання є великі порядки систем лінійних рівнянь. У більшості випадків матриці таких систем є розрідженими, тобто кількість їх ненульових елементів значно менша за загальну кількість елементів. Якщо  $nnz(A)$  — кількість ненульових елементів матриці  $A$ , то для розрідженої матриці виконується співвідношення

$$nnz(A) \ll n^2.$$

Ця властивість дає змогу використовувати спеціальні формати зберігання розріджених матриць, які істотно зменшують витрати пам'яті та кількість арифметичних операцій.

До найбільш поширених форматів зберігання розріджених матриць належать координатний формат (COO), рядковий формат (CSR), стовпчиковий формат (CSC), формат ELLPACK, а також різні гібридні формати. Ефективність чисельних алгоритмів значною мірою залежить від правильного вибору формату зберігання матриці та способу її обробки.

У сучасних задачах математичного моделювання матриці можуть мати складну розріджену структуру, зокрема стрічкову, профільну, блочно-діагональну, блочно-хмарочосну або довільну розріджену структуру. Вибір ефективного алгоритму розв'язування систем лінійних алгебраїчних рівнянь суттєво залежить від типу структури матриці. Тому важливим етапом дослідження математичних моделей є аналіз структури розрідженості

матриці, її чисельних властивостей та можливостей їх урахування при побудові алгоритмів.

Особливе значення має також дослідження таких властивостей матриці, як симетричність, додатна визначеність, обумовленість, спектральні характеристики та характер заповнення. Саме ці властивості значною мірою визначають вибір чисельного методу та особливості його програмної реалізації.

Таким чином, постановка задач математичного моделювання з наближеними даними передбачає комплексне дослідження властивостей математичної та комп'ютерної моделей, аналіз структури матриць, оцінювання впливу похибок вихідних даних на результати обчислень і врахування обмежень машинної арифметики [44-57]. Результати такого дослідження є основою для побудови ефективних, стійких і достовірних методів розв'язування систем лінійних алгебраїчних рівнянь на сучасних високопродуктивних комп'ютерних системах [5, 58-69].

У задачах математичного моделювання значна частина обчислювальних витрат пов'язана з обробкою матриць великих порядків, що виникають після дискретизації диференціальних або інтегральних рівнянь. Щільні матриці характеризуються тим, що більшість їх елементів є ненульовими. Для таких матриць застосовуються класичні алгоритми чисельної лінійної алгебри, зокрема метод Гауса, *LU*-факторизація, *QR*-факторизація та факторизація Холецького. Основною особливістю ефективної реалізації таких алгоритмів є використання блочних обчислень, що дозволяє максимально використовувати кеш-пам'ять сучасних процесорів і реалізовувати високопродуктивні матрично-матричні операції.

Важливу роль у реалізації алгоритмів для щільних матриць відіграють бібліотеки BLAS [70-75] та LAPACK [76], які містять оптимізовані процедури для базових операцій лінійної алгебри. У паралельних обчислювальних системах широко використовується бібліотека ScaLAPACK

[77-79], яка реалізує паралельні алгоритми для щільних матриць на основі двовимірного блочно-циклічного розподілу даних.

Стрічкові матриці характеризуються тим, що ненульові елементи розташовані у межах певної смуги відносно головної діагоналі. Ширина цієї смуги визначається параметрами верхньої та нижньої півширини стрічки. Для таких матриць ефективно застосовуються спеціалізовані алгоритми факторизації, які враховують стрічкову структуру та дозволяють значно зменшити кількість арифметичних операцій порівняно з повною обробкою матриці. Ефективне використання стрічкових матриць має особливе значення у задачах механіки конструкцій, чисельного розв'язування диференціальних рівнянь і методі скінченних елементів.

У багатьох випадках матриці мають також профільну структуру, коли ненульові елементи розташовані в межах певного профілю відносно головної діагоналі. Для таких матриць застосовуються спеціальні схеми, що дозволяють зберігати лише ненульові елементи профілю та істотно зменшувати витрати пам'яті.

У більшості сучасних задач математичного моделювання матриці систем є розрідженими. Ефективна обробка таких матриць потребує використання спеціальних форматів зберігання даних, які дозволяють значно зменшити обсяг пам'яті та підвищити ефективність обчислень при множенні матриці на вектор, а також при реалізації алгоритмів факторизації. Разом із тим при використанні прямих методів для розріджених матриць виникає проблема заповнення, коли під час факторизації з'являються нові ненульові елементи. Для зменшення цього ефекту застосовуються спеціальні методи впорядкування невідомих, зокрема алгоритми Cuthill–McKee, *minimum degree* та їх модифікації.

Особливо важливими є блочні алгоритми, що застосовуються у сучасних бібліотеках лінійної алгебри для багатоядерних і гібридних комп'ютерів. Такі алгоритми забезпечують кращу локальність доступу до даних, дозволяють

підвищити ступінь паралелізму та краще узгоджуються з ієрархічною структурою пам'яті сучасних процесорів і графічних прискорювачів.

Ефективне використання структури матриць відіграє ключову роль і в паралельних алгоритмах. Для цього застосовуються різні схеми декомпозиції матриць та розподілу даних між обчислювальними вузлами. Одним із найбільш ефективних підходів є двовимірний блочно-циклічний розподіл матриць, який широко використовується в паралельних бібліотеках лінійної алгебри. Такий підхід дозволяє забезпечити рівномірне завантаження процесорів, зменшити обсяг комунікацій між ними та підвищити масштабованість алгоритмів.

У сучасних гібридних обчислювальних системах значна частина обчислень може виконуватися на графічних процесорах. Для таких систем були розроблені спеціалізовані бібліотеки, зокрема cuBLAS [80], cuSPARSE [81] та MAGMA [82], які реалізують високопродуктивні алгоритми обробки щільних і розріджених матриць на архітектурах типу CPU+GPU.

Таким чином, ефективність розв'язування систем лінійних алгебраїчних рівнянь значною мірою визначається повнотою врахування структури матриць, що виникають у задачах математичного моделювання. Рациональний вибір формату зберігання даних, методу факторизації, блочної організації обчислень і схеми розподілу матричних блоків дає змогу істотно зменшити обсяг арифметичних операцій, скоротити витрати пам'яті та підвищити продуктивність програмних реалізацій.

Особливе значення для СЛАР великих порядків мають блочні алгоритми, які забезпечують кращу локалізацію даних, ефективніше використання кеш-пам'яті, векторних інструкцій, багатоядерних процесорів і графічних прискорювачів. Саме тому поєднання структурних властивостей матриць із блочними та паралельними схемами обчислень є одним із ключових напрямів підвищення ефективності розв'язування СЛАР на сучасних високопродуктивних комп'ютерних системах.

## 1.2 Застосування методів та алгоритмічно-програмного забезпечення для розв'язування СЛАР

Інтерес до проблеми побудови ефективних методів розв'язування систем лінійних алгебраїчних рівнянь з матрицями різної структури зумовлений широким колом їх практичних застосувань. Зокрема, системи рівнянь з розрідженими матрицями виникають у задачах аналізу міцності конструкцій у цивільному та промисловому будівництві, в авіа- та ракетобудуванні, суднобудуванні, машинобудуванні та інших галузях. У сучасних умовах сфера застосування методів розв'язування СЛАР з розрідженими матрицями [83-99] постійно розширюється, що зумовлено зростанням складності математичних моделей та потребою у високоточному чисельному моделюванні складних процесів.

Методи розв'язування СЛАР і пов'язаних з ними задач найбільш повно висвітлено у численних працях з лінійної алгебри та обчислювальної математики. Значний внесок у розвиток методів розв'язування СЛАР з розрідженими матрицями зробили А. Джордж, Дж. Лю, Й. Саад, Дж. Голуб, Дж. Донгара, С. Писанецький та інші дослідники [1-15]. У їхніх роботах розглянуто як загальні принципи побудови методів лінійної алгебри, так і спеціалізовані підходи до роботи з розрідженими матрицями, включаючи формати зберігання, схеми попереднього впорядкування та алгоритми факторизації.

На сучасному етапі розвитку обчислювальної техніки паралельні комп'ютери різної архітектури стали основним інструментом чисельного моделювання складних процесів у різних предметних областях. Такі системи дозволяють розв'язувати задачі значно більших порядків та, як правило, за істотно менший час. Значна частина цих задач зводиться саме до розв'язування СЛАР з розрідженими матрицями. У зв'язку з цим побудова паралельних алгоритмів для розв'язування СЛАР з розрідженими матрицями великих порядків на сучасних високопродуктивних суперкомп'ютерах є актуальною науковою проблемою.

Протягом останніх десятиліть було створено значну кількість бібліотек програм, до яких увійшли засоби розв'язування СЛАР з розрідженими матрицями. До таких бібліотек належать SparseBLAS, SparsPak, IMSL, NAG, Aztec, Hypre, MUMPS, PSBLAS, SuperLU, SPARSKIT та інші. Вони реалізують різні підходи до розв'язування систем з розрідженими матрицями та забезпечують високу ефективність обчислень на сучасних обчислювальних системах.

Окремо слід відзначити алгоритмічно-програмне забезпечення з лінійної алгебри, розроблене під керівництвом Дж. Донгари. Найбільш відомими є бібліотеки BLAS, LAPACK і ScaLAPACK, які реалізують блочні алгоритми розв'язування СЛАР для однопроцесорних, багатопроцесорних і розподілених систем. Блочні алгоритми дозволяють узгоджувати розмір блоків матриці з обсягом кеш-пам'яті комп'ютера, що забезпечує суттєве підвищення швидкодії алгоритмів і програм.

Подальший розвиток обчислювальної техніки привів до створення гібридних комп'ютерів, у яких поєднуються центральні процесори та графічні прискорювачі. У зв'язку з цим було розроблено низку бібліотек програм для гібридних обчислювальних систем. Зокрема, для платформи NVIDIA CUDA створено бібліотеку cuBLAS, а для роботи з розрідженими матрицями — бібліотеку cuSPARSE. Для гібридних систем особливо важливою є бібліотека MAGMA, орієнтована на роботу зі щільними матрицями на архітектурах типу CPU+GPU. Такі програмні засоби забезпечують можливість ефективного виконання матричних обчислень з урахуванням архітектурних особливостей графічних прискорювачів.

На основі перелічених програмних продуктів було також створено бібліотеки, адаптовані до конкретних типів процесорів. Наприклад, бібліотека Intel MKL оптимізована для процесорів Intel і підтримує багатопотокове виконання на багатоядерних системах, а її важливою перевагою є наявність механізмів автоматичної диспетчеризації, які дозволяють на етапі виконання вибирати оптимальні процедури залежно від

типу процесора та розмірності задачі. Аналогічно для процесорів AMD розроблялися спеціалізовані оптимізовані бібліотеки математичних підпрограм.

Ефективними засобами для розв'язування розріджених лінійних систем на графічних процесорах є також бібліотеки, орієнтовані на ітераційні методи, зокрема CUSP та Lis. Вони реалізують паралельні алгоритми, включаючи методи спряжених градієнтів, різні типи передобумовлювачів та засоби роботи із задачами на власні значення.

Разом з тим аналіз літератури та практичний досвід використання розглянутих високопродуктивних бібліотек обчислювальної математики показують, що в більшості з них неявно передбачається точність вихідних даних, а питання достовірності отриманих комп'ютерних результатів спеціально не досліджуються. Це обмежує можливості їх застосування у задачах математичного моделювання, де вихідні дані часто задані наближено. У таких випадках важливого значення набуває не лише ефективність алгоритму, а й дослідження достовірності обчисленого розв'язку.

Таким чином, сучасні методи, алгоритми та бібліотеки програм забезпечують широкий інструментарій для розв'язування систем лінійних алгебраїчних рівнянь на комп'ютерах різної архітектури. Проте для задач математичного моделювання з наближено заданими даними актуальною залишається проблема створення нових методів і алгоритмічно-програмних засобів, які поєднують високу продуктивність, адаптивність до структури матриць та архітектури системи, а також контроль достовірності комп'ютерного результату.

### **1.3 Особливості створення адаптивних методів та алгоритмів розв'язування СЛАР для гібридних комп'ютерів**

Аналіз проблем ефективного використання високопродуктивних комп'ютерних систем, архітектурні та технологічні характеристики яких

постійно еволюціонують, приводить до необхідності створення адаптивних алгоритмів розв'язування складних обчислювальних задач. Такі алгоритми повинні забезпечувати автоматичне налаштування обчислювального методу, комп'ютерного алгоритму, моделі паралелізації та обчислювального середовища відповідно до структурних особливостей конкретної задачі та параметрів апаратної платформи.

У сучасних суперкомп'ютерних системах реалізується багаторівнева модель паралельних обчислень, яка передбачає використання декількох рівнів паралелізму. Найбільш поширеною є дворівнева модель, що включає верхній рівень паралелізму, на якому паралельно виконуються макрооперації або підзадачі алгоритму, та нижній рівень паралелізму, на якому здійснюється розпаралелювання обчислень всередині кожної макрооперації.

На верхньому рівні реалізується паралелізм процесів, що відповідає моделі MIMD (Multiple Instruction – Multiple Data). У цьому випадку різні процеси виконують незалежні підзадачі алгоритму, використовуючи як розподілену, так і спільну пам'ять багатопроцесорних систем. Для організації паралельних обчислень у системах із розподіленою пам'яттю найчастіше використовується стандарт MPI [100], тоді як у системах зі спільною пам'яттю ефективним засобом паралелізації є OpenMP [101].

Нижній рівень паралелізму відповідає паралелізму потоків (Thread Level Parallelism) і здебільшого реалізується відповідно до моделі SIMD або SIMT. У цьому випадку однакові операції виконуються над різними елементами даних із використанням великої кількості потоків.

Використання систем із розподіленою пам'яттю породжує специфічні проблеми, пов'язані з обміном даними між процесами. Комунікаційні операції можуть значно перевищувати за тривалістю арифметичні операції та операції доступу до пам'яті. Тому при проектуванні паралельних алгоритмів необхідно забезпечити таку організацію розміщення даних у пам'яті процесорів, за якої співвідношення між обчисленнями та пересиланнями даних буде оптимальним і забезпечуватиме мінімальний час виконання

програми. Важливу роль при цьому відіграє використання асинхронних комунікацій і перекриття комунікацій обчисленнями.

У середовищах зі спільною пам'яттю проблема обмінів даними між потоками значною мірою знімається, проте виникають труднощі, пов'язані з синхронізацією доступу до спільних даних та узгодженням кеш-пам'яті різних рівнів. Крім того, модель OpenMP найбільш ефективна для реалізації синхронного виконання однакових операцій над різними даними, тоді як організація асинхронного виконання різних макрооперацій у цьому середовищі є складнішою.

Суттєве підвищення продуктивності обчислень досягається при використанні графічних процесорів (GPU). Сучасні GPU є масово-паралельними обчислювальними пристроями, які містять тисячі обчислювальних ядер, здатних одночасно виконувати велику кількість потоків. Основу архітектури GPU становлять потокові мультипроцесори (Streaming Multiprocessor, SM), кожен з яких включає десятки або сотні обчислювальних ядер, блоки управління потоками, регістрову пам'ять та швидку спільну пам'ять.

У моделі програмування CUDA паралельні обчислення організуються у вигляді ієрархії потоків. Потоки об'єднуються у блоки (thread blocks), а блоки формують сітку виконання (grid). Усередині кожного потокового мультипроцесора потоки організуються у групи по 32 потоки, які називаються warp. Потоки одного warp виконують одну і ту саму інструкцію над різними даними відповідно до моделі SIMT (Single Instruction Multiple Threads).

Важливою особливістю сучасних GPU є складна ієрархія пам'яті. Вона включає глобальну пам'ять, кеш-пам'ять різних рівнів, спільну пам'ять потокових мультипроцесорів та регістрову пам'ять. Ефективність виконання алгоритмів значною мірою залежить від раціонального використання цієї ієрархії пам'яті, мінімізації доступу до глобальної пам'яті та локалізації обчислень у швидших рівнях пам'яті.

Графічні процесори нових поколінь характеризуються значним збільшенням кількості обчислювальних блоків, наявністю спеціалізованих тензорних ядер (Tensor Cores), а також використанням високошвидкісної пам'яті типу HBM (High Bandwidth Memory). Тензорні ядра призначені для високопродуктивного виконання операцій матричного множення, що дозволяє суттєво прискорити обчислення у задачах чисельної лінійної алгебри та машинного навчання.

У сучасних високопродуктивних системах дедалі ширше застосовуються конфігурації з кількома графічними прискорювачами (мульти-GPU). Використання таких систем дозволяє значно підвищити продуктивність обчислень за рахунок паралельної обробки великих обсягів даних на кількох GPU. При цьому важливе значення має ефективна організація обміну даними між графічними процесорами.

У сучасних обчислювальних системах для взаємодії між GPU використовуються високошвидкісні інтерконекти, зокрема NVLink та NVSwitch, які забезпечують значно більшу пропускну здатність порівняно з традиційною шиною PCI Express. Це дозволяє організовувати швидкий обмін даними між графічними процесорами та ефективно реалізовувати розподілені обчислення.

У multi-GPU системах виникають додаткові проблеми, пов'язані з балансуванням навантаження між графічними процесорами, організацією міжпроцесорних комунікацій та мінімізацією часу передачі даних. Для ефективної роботи таких систем використовуються спеціалізовані комунікаційні бібліотеки, зокрема NCCL, які забезпечують ефективну реалізацію колективних операцій обміну даними між GPU.

Графічні процесори найбільш ефективні для задач, які характеризуються наявністю масового паралелізму за даними, відсутністю частих глобальних синхронізацій та значним переважанням арифметичних операцій над операціями доступу до пам'яті. У таких задачах обчислення можуть бути

розподілені на велику кількість потоків, що забезпечує значне прискорення виконання алгоритмів.

Важливим фактором ефективності паралельних обчислень є також топологія міжпроцесорних зв'язків та ієрархія пам'яті. Раціональне використання кеш-пам'яті та оптимальна організація обміну даними між процесорами дозволяють суттєво зменшити затримки доступу до пам'яті та підвищити продуктивність алгоритмів.

Для задач лінійної алгебри найбільш ефективними з точки зору використання кеш-пам'яті та паралельних обчислень є блочні та плиткові алгоритми. Такі алгоритми дозволяють локалізувати обчислення в окремих підматрицях і забезпечують ефективне використання як центральних процесорів, так і графічних прискорювачів.

При реалізації алгоритмів факторизації матриць важливе значення має схема декомпозиції матриці між обчислювальними елементами. Оскільки у процесі факторизації розмір активної частини матриці поступово зменшується, виникає необхідність забезпечити рівномірне завантаження обчислювальних ресурсів. Для цього широко застосовуються циклічні та блочно-циклічні схеми розподілу даних, які дозволяють зменшити дисбаланс навантаження та підвищити ефективність паралельних обчислень.

Перспективним напрямом підвищення продуктивності є також використання змішаної комп'ютерної розрядності. Обчислення в одинарній точності дозволяють зменшити обсяг пам'яті та підвищити швидкодію, тоді як для забезпечення необхідної точності розв'язку може застосовуватися поєднання одинарної та подвійної точності.

Таким чином, при розв'язуванні систем лінійних алгебраїчних рівнянь на гібридних комп'ютерних системах доцільно використовувати адаптивні алгоритми, які враховують структуру матриці, особливості архітектури обчислювальної системи, характеристики пам'яті та комунікаційного середовища, а також можливості сучасних GPU та multi-GPU систем. Реалізація таких підходів дозволяє підвищити ефективність використання

обчислювальних ресурсів і забезпечити високу продуктивність розв'язування задач математичного моделювання великої розмірності.

#### **1.4 Інтелектуалізація алгоритмічно-програмного забезпечення високопродуктивних обчислень з лінійної алгебри на сучасних суперкомп'ютерах**

Сучасний розвиток високопродуктивних обчислень характеризується не лише постійним зростанням обчислювальної потужності суперкомп'ютерів, але й суттєвим ускладненням їх архітектури. Багатоядерні центральні процесори, графічні прискорювачі, багаторівнева ієрархія пам'яті, високошвидкісні комунікаційні мережі та гібридні моделі обчислень формують складне середовище, у якому традиційні підходи до побудови алгоритмічно-програмного забезпечення вже не забезпечують достатньої ефективності. У цих умовах особливого значення набуває інтелектуалізація алгоритмічно-програмного забезпечення, тобто надання йому властивостей адаптивності, автоматизованого аналізу задачі, вибору методу розв'язування, налаштування параметрів алгоритму та раціонального розподілу обчислень між доступними ресурсами суперкомп'ютера.

Для задач лінійної алгебри ця проблема є особливо актуальною, оскільки значна частина прикладних задач математичного моделювання після дискретизації зводиться до розв'язування систем лінійних алгебраїчних рівнянь, задач на власні значення, а також до виконання великої кількості матрично-векторних і матрично-матричних операцій. Ефективність обчислень у таких задачах суттєво залежить від структури матриці, її порядку, щільності або розрідженості, чисельних властивостей, а також від того, наскільки коректно алгоритм узгоджений з архітектурними особливостями комп'ютерної системи.

Традиційні бібліотеки лінійної алгебри, такі як BLAS, LAPACK, ScaLAPACK, SuperLU, MUMPS, PETSc, Trilinos, cuBLAS, cuSPARSE, MAGMA та інші, забезпечують високий рівень продуктивності для певних

класів задач і певних архітектурних платформ. Проте їх застосування, як правило, передбачає, що користувач або розробник уже знає, який саме метод доцільно використовувати, який формат зберігання даних обрати, яким чином розподілити дані між процесорами, які параметри алгоритму є оптимальними, а також на яких обчислювальних пристроях доцільно виконувати ті чи інші етапи розрахунку. У задачах великих порядків та складної структури такий підхід стає недостатнім, оскільки вимагає від користувача глибоких знань одночасно в галузі лінійної алгебри, паралельного програмування та архітектури суперкомп'ютерів.

У зв'язку з цим одним із перспективних напрямів розвитку високопродуктивних обчислень є створення інтелектуальних систем алгоритмічно-програмної підтримки, які забезпечують автоматизацію вибору чисельного методу та способу його реалізації. Такі системи повинні аналізувати математичні властивості задачі, структуру матриці, характеристики доступного обчислювального середовища та на цій основі формувати ефективну стратегію розв'язування.

Сучасні суперкомп'ютери мають багаторівневу структуру, що поєднує розподілену пам'ять між вузлами, спільну пам'ять всередині вузла та окрему пам'ять графічних прискорювачів. У таких умовах ефективний алгоритм повинен враховувати кількість обчислювальних вузлів, кількість CPU-ядер, кількість GPU та їх характеристики, пропускну здатність пам'яті, топологію мережі, вартість комунікацій, а також співвідношення між продуктивністю CPU та GPU.

Інтелектуалізація програмного забезпечення передбачає автоматичний вибір моделі паралелізації, зокрема визначення, які етапи доцільно виконувати на CPU, які — на GPU, а також як організувати обмін даними між ними. Наприклад, факторизація блоку може виконуватися на CPU, тоді як оновлення підматриці — на GPU, оскільки саме матрично-матричні операції найкраще узгоджуються з архітектурою графічних прискорювачів.

Для задач з розрідженими матрицями надзвичайно важливим є вибір формату зберігання. Один і той самий алгоритм може демонструвати суттєво різну ефективність залежно від того, чи використовується формат CSR, CSC, COO, ELLPACK, HYB або блочний формат. У ряді випадків саме вибір формату визначає можливість ефективної реалізації операцій на GPU або забезпечує кращу локальність доступу до пам'яті. Тому інтелектуальна система повинна не лише класифікувати структуру матриці, але й автоматично добирати формат її подання, який є найбільш ефективним для обраного алгоритму та конкретної обчислювальної платформи.

У багатьох чисельних методах ефективність визначається значеннями параметрів, які складно обрати наперед. До таких параметрів належать розмір блоку, кількість потоків, розмір робочих буферів, параметри передобумовлювача, рівень неповної факторизації, пороги відсікання елементів, стратегія балансування навантаження та схема перекриття комунікацій і обчислень. У традиційних бібліотеках ці параметри часто вибираються емпірично або задаються користувачем. Інтелектуалізація передбачає автоматичне налаштування таких параметрів на основі тестування, моделей продуктивності або попередньо накопичених даних.

Одним із перспективних напрямів інтелектуалізації є застосування методів машинного навчання та штучного інтелекту. Такі методи можуть використовуватися для класифікації структури матриць за їх портретом, прогнозування ефективності різних чисельних методів, вибору формату зберігання даних, налаштування параметрів алгоритму, оцінювання ймовірності появи надмірного fill-in та прогнозування часу виконання алгоритму на конкретній архітектурі. Такий підхід є особливо перспективним у задачах, де структура матриць повторюється або належить до певного класу прикладних моделей.

Ще одним важливим аспектом є оцінювання достовірності отриманих результатів. Більшість сучасних програмних бібліотек не проводить глибокого аналізу похибок, а лише надає механізми контролю залишку або

оцінки числа обумовленості. Проте у задачах з наближеними даними цього недостатньо. Інтелектуальна система повинна враховувати не лише ефективність, але й надійність обчислень. Це може включати аналіз обумовленості, виявлення можливих втрат точності, застосування змішаної або багаторозрядної арифметики, а також вибір більш стійкого методу при виявленні чисельної нестійкості.

Таким чином, інтелектуалізація алгоритмічно-програмного забезпечення охоплює не тільки вибір найшвидшого методу, але й забезпечення достовірності комп'ютерного результату. Поєднання адаптивних підходів, методів машинного навчання та моделей продуктивності створює передумови для побудови інтелектуальних систем підтримки високопродуктивних обчислень у задачах лінійної алгебри на сучасних суперкомп'ютерах.

### **1.5 Постановка завдання дослідження**

- проаналізувати сучасні методи та алгоритми розв'язування систем лінійних алгебраїчних рівнянь великих порядків на високопродуктивних обчислювальних системах;
- дослідити особливості подання та обробки щільних, стрічкових і розріджених матриць у задачах математичного моделювання;
- розробити паралельний алгоритм  $LU$ -факторизації щільних матриць на основі двовимірного блочно-циклічного розподілу даних для мульти-GPU архітектур;
- розробити блочний алгоритм факторизації несиметричних стрічкових матриць з урахуванням їхньої структури, обмеженої області заповнення та особливостей GPU-обчислень;
- отримати теоретичні оцінки обчислювальної складності, часу виконання, прискорення та ефективності запропонованих алгоритмів;
- реалізувати запропоновані алгоритми у вигляді програмних засобів для гібридних обчислювальних систем;

- провести експериментальне дослідження ефективності розроблених алгоритмів на тестових задачах і прикладних задачах математичного моделювання складних конструкцій та технологічних процесів.

## **1.6 Висновки до розділу**

У розділі проведено аналіз особливостей розрахункових задач математичного моделювання для різних галузей інженерії та науки, багато з яких використовують розв'язки СЛАР. Це, як правило, системи великих порядків з наближеними даними. Тому актуальним є розробка методів розв'язування СЛАР з матрицями великих порядків з забезпеченням достовірності комп'ютерних результатів.

Отже, розв'язування та забезпечення достовірності розв'язків СЛАР потребує значних обчислювальних ресурсів, які надаються сучасними високопродуктивними комп'ютерними системами різної архітектури та відповідними програмними засобами.

Встановлено, що ефективність розв'язування СЛАР значною мірою визначається структурою матриці, форматом зберігання даних задачі, чисельним методом і алгоритмом розв'язування та схемою організації обчислень.

Показано доцільність розроблення методів, алгоритмів і програмних засобів високопродуктивних обчислень, орієнтованих на розв'язування систем лінійних алгебраїчних рівнянь з матрицями різної структури для сучасних комп'ютерів гібридної та мульти-GPU архітектури.

Саме ці положення становлять основу подальших досліджень, викладених у наступних розділах дисертаційної роботи.

## РОЗДІЛ 2. ПАРАЛЕЛЬНІ АЛГОРИТМИ *LU*-ФАКТОРИЗАЦІЇ ДЛЯ РОЗВ'ЯЗУВАННЯ СЛАР НА ГІБРИДНИХ КОМП'ЮТЕРНИХ СИСТЕМАХ

У задачах математичного моделювання складних фізичних процесів значна частина обчислювальних витрат пов'язана з розв'язуванням систем лінійних алгебраїчних рівнянь. Такі системи виникають у результаті дискретизації диференціальних та інтегральних рівнянь, що описують процеси в механіці, гідродинаміці, теплопровідності, електродинаміці та інших галузях науки і техніки.

Особливістю практичних задач є те, що вихідні дані математичних моделей часто задані наближено. Елементи матриці системи та вектора правої частини можуть визначатися на основі експериментальних вимірювань, чисельної апроксимації або спрощених математичних моделей. Унаслідок цього виникають похибки, які можуть суттєво впливати на отриманий розв'язок. Тому при побудові чисельних алгоритмів важливо враховувати не лише ефективність обчислень, але й їх чисельну стійкість та достовірність результатів.

Одним із основних інструментів аналізу таких задач є дослідження обумовленості системи лінійних рівнянь. Відомо, що для погано обумовлених систем навіть незначні збурення елементів матриці або правої частини можуть призводити до значних змін розв'язку. У зв'язку з цим важливим є використання стійких чисельних методів, які дозволяють отримувати достовірні результати навіть у випадку систем, заданих з наближеними даними.

Одним із найбільш поширених прямих методів розв'язування систем лінійних алгебраїчних рівнянь є *LU*-факторизація матриці, яка дозволяє представити матрицю системи у вигляді добутку нижньої та верхньої трикутних матриць. Цей метод широко використовується у лінійній алгебрі та лежить в основі багатьох сучасних програмних бібліотек. Разом з тим при

розв'язуванні задач обчислювальна складність  $LU$ -факторизації є дуже високою. Тому ефективна реалізація цього методу потребує використання паралельних алгоритмів, орієнтованих на сучасні високопродуктивні комп'ютерні системи.

Сучасні суперкомп'ютери мають гібридну архітектуру, що поєднує багатоядерні центральні процесори та графічні прискорювачі. Використання GPU дозволяє реалізувати масовий паралелізм обчислень і суттєво підвищити продуктивність виконання матричних операцій.

У зв'язку з цим виникає необхідність розроблення ефективних паралельних алгоритмів факторизації матриць, які враховують особливості сучасних обчислювальних архітектур та забезпечують масштабованість обчислень на системах з кількома графічними прискорювачами.

У цьому розділі розглядаються методи побудови паралельних алгоритмів  $LU$ -факторизації для щільних матриць [20], досліджуються підходи до організації обчислень на гібридних комп'ютерних системах та аналізуються ефективні схеми розподілу даних між обчислювальними пристроями. Особливу увагу приділено використанню блочних алгоритмів та двовимірного блочно-циклічного розподілу матриць, які дозволяють забезпечити ефективне використання ресурсів multi-GPU систем.

## 2.1 Комп'ютерне дослідження лінійних систем з наближеними даними

Розв'язування систем лінійних алгебраїчних рівнянь:

$$Ax = b, \quad (2.1)$$

де в загальному випадку  $A$  – прямокутна матриця розміру  $m \times n$ ;  $b$  – матриця правої частини розміру  $m \times q$ , зводиться в класичному випадку до знаходження такого розв'язку  $x$  (матриці розміру  $n \times q$ ), щоб рівняння (2.1) перетворювалося в тотожність. Якщо матриця  $A$  системи квадратна не вироджена (тобто її визначник  $|A| \equiv \det(A) \neq 0$ ), то розв'язок СЛАР (2.1) існує і єдиний.

Несумісні системи класичного розв'язку не мають, проте в цьому випадку можна знайти псевдорозв'язок  $x$ , який мінімізує евклідову норму  $\|Ax - b\|$ . Вектор  $x$  називається розв'язком за методом найменших квадратів або псевдорозв'язком СЛАР. В загальному випадку існує нескінченна множина псевдорозв'язків. Розв'язок, який має найменшу норму  $\|x\|$ , єдиний і називається нормальним псевдорозв'язком.

При розв'язуванні прикладних задач рідко виникають СЛАР з точними вихідними даними:

$$\bar{A}\bar{x} = \bar{b}. \quad (2.2)$$

Найбільш типовою є постановка задачі (2.1) разом з заданням відповідних похибок у вихідних даних:

$$\|\bar{A} - A\| = \|\Delta A\| \leq \varepsilon_A \|\bar{A}\|, \quad \|\bar{b} - b\| = \|\Delta b\| \leq \varepsilon_b \|\bar{b}\|. \quad (2.3)$$

При цьому передбачається, що структура матриці вихідної задачі (2.2) та збуреної задачі (2.1) не змінюються, тобто, якщо вихідна матриця є симетрична, то і збурена залишається також симетричною, якщо вихідна – стрічкова, то і збурена – стрічкова, тощо.

Тут ми розглядаємо випадок, коли  $A$  – квадратна матриця розміру  $n \times n$ ;  $b$  – вектор правої частини СЛАР розміру  $n$ ,  $x$  – розв'язок (вектор розміру  $n$ ),  $\varepsilon_A$ ,  $\varepsilon_b$  – максимальні відносні похибки елементів матриці та її правої частини відповідно.

Розв'язуючи СЛАР з наближено заданими вихідними даними, необхідно розглядати цілий клас систем рівнянь (2.1), (2.3), що має досить широку множину формально допустимих розв'язків. Тому, розв'язавши задачу (2.1), необхідно оцінити збурення розв'язку в залежності від збурення вихідних даних (2.3). Не завжди близькість елементів матриць  $A$  і  $\bar{A}$  та правих частин  $b$  і  $\bar{b}$  забезпечують достатню близькість розв'язків. Наприклад, при деякому збуренні в межах точності задання елементів матриці і / або правої частини несумісної точно заданої системи (2.2) отримана в комп'ютері збурена

система (2.1), (2.3) може виявитися сумісною і навпаки – сумісна СЛАР може перетворитися в несумісну [1].

Похибку розв'язку  $\Delta x = \bar{x} - x$ , яка обумовлена наближеним заданням вихідних даних, називають спадковою. Її числове значення залежить як від похибки вихідних даних (2.3), так і від властивостей матриці вихідної задачі (2.2).

Наближений розв'язок  $\tilde{x}$  системи (2.1), який отримується в результаті розв'язування задачі на комп'ютері, будемо називати комп'ютерним розв'язком задачі. Через похибки методу розв'язування та комп'ютерних обчислень отриманий розв'язок відрізняється від точного (математичного) розв'язку задачі (2.1). Різниця між цими розв'язками  $\Delta \tilde{x} = x - \tilde{x}$  – це обчислювальна похибка.

Дослідження та розв'язування СЛАР з наближеними даними на комп'ютері складається з таких кроків [7, 38]:

- дослідження математичних властивостей комп'ютерної задачі (2.1), (2.3);
- розв'язування задачі алгоритмом, що відповідає відомим або виявленим математичним властивостям СЛАР з урахуванням архітектурних і технологічних особливостей комп'ютера;
- аналіз достовірності розв'язку задачі – оцінка спадкової та обчислювальної похибок отриманого в комп'ютері розв'язку.

В першу чергу комп'ютерного дослідження СЛАР з наближеними даними проводиться дослідження коректності постановки задачі (2.1), (2.3), тобто визначається існування, єдиність та стійкість класичного розв'язку.

Теоретичним критерієм коректності СЛАР (2.1), (2.3) є виконання умов:

$$\det(A) \neq 0, \quad \|\Delta A\| \|A^{-1}\| < 1$$

для довільного збурення  $\Delta A$  в межах (2.3), які гарантують існування, єдиність та стійкість класичного розв'язку задач в області зміни вихідних даних.

Комп'ютерне дослідження коректності зводиться до перевірки співвідношення:

$$1.0 + \gamma \neq 1.0, \quad (2.4)$$

де  $\gamma = h^{-1}(A)$ ,  $h(A)$  – число обумовленості матриці. Ця умова, яка виконується в арифметиці з плаваючою комою, означає що матриця не вироджена в межах машинної точності (машинно не вироджена), а умова  $(\|\Delta A\|/\|A\|)h(A) < 1$ , – що вона не вироджена в межах точності задання вихідних даних.

При виконанні умови (2.4) розв'язок комп'ютерної задачі існує, єдиний і стійкий. Таку комп'ютерну задачу слід розглядати як коректно поставлену в межах машинної точності. В іншому випадку матриця системи може виявитися матрицею виродженою в межах машинної точності або погано обумовленою.

Для систем з прямокутними та виродженими матрицями обчислюється нормальний псевдорозв'язок який, як зазначалося вище, у випадку сумісної системи співпадає з класичним розв'язком.

Відмітимо, що у формулі (2.4) для перевірки коректності комп'ютерної задачі присутня величина  $\gamma$ , обернена до  $h(A)$ . Тому в разі виникнення великих чисел обумовленості в комп'ютері не настає переповнення за порядком.

Зникнення порядку для  $h^{-1}(A)$  при великих числах обумовленості не фатально: комп'ютерний результат покладається рівним нулю, що дає можливість зробити правильний висновок про машинну виродженість матриці комп'ютерної задачі. У випадку, якщо в процесі розвинення ( $LU$ ,  $LL^T$ ) матриці СЛАР з'явилися нульові діагональні елементи, необхідно покласти  $\gamma$  рівним нулю, і тоді умова (2.4) дає можливість також зробити правильний висновок щодо виродженості матриці.

Таким чином, основним критерієм для визначення властивостей СЛАР, які впливають на достовірність розв'язку задачі, є число обумовленості  $h(A)$  матриці системи.

Для не вироджених матриць, число обумовленості обчислюється, використовуючи обернену матрицю  $A^{-1}$

$$h(A) = \|A\| \|A^{-1}\|, \quad (2.5)$$

а в інших випадках – через псевдообернену матрицю  $A^\#$

$$h(A) = \|A\| \|A^\#\|. \quad (2.6)$$

Для відносної спадкової похибки розв'язку задачі (2.1), (2.3) має місце оцінка [10]

$$\frac{\|x - \bar{x}\|}{\|x\|} \leq \frac{h(A)(\varepsilon_A + \varepsilon_b)}{1 - h(A)\varepsilon_A} \quad (2.7)$$

або

$$\frac{\|x - \bar{x}\|}{\|\bar{x}\|} \leq \frac{h(A)(\varepsilon_A + \varepsilon_b)}{1 - \varepsilon_b} \quad (2.8)$$

за умови  $\|\Delta A\| \|A^{-1}\| < 1$  та природному припущенні  $\varepsilon_b < 1$ .

З оцінок (2.7) та (2.8) видно, що достовірність розв'язку системи визначається величиною числа обумовленості матриці та похибками вихідних даних задачі.

Оцінки мають мажорантний характер на всьому класі матриць. Проте, для невивроджених матриць показана досяжність оцінки для першої матричної норми і тим самим доведена непокращуваність оцінки на класі невивроджених матриць.

Для прямокутних  $(m \times n)$ -матриць повного рангу, тобто у випадку  $r(A) = \min\{m, n\}$ , де  $r(A)$  – ранг матриці  $A$ , відносна спадкова похибка за умови  $\|\Delta A\| \|A^\#\| < 1$  оцінюється для  $m > n$  так [73–77]

$$\frac{\|x - \bar{x}\|}{\|x\|} \leq \frac{h(A)}{1 - h(A)\varepsilon_A} \left( \left( 1 + h(A) \frac{\|r\|}{\|Ax\|} \right) \varepsilon_A + \varepsilon_b \right),$$

де  $r = Ax - b$ , а для  $m < n$  так

$$\frac{\|x - \bar{x}\|}{\|x\|} \leq \frac{h(A)(2\varepsilon_A + \varepsilon_b)}{1 - h(A)\varepsilon_A}.$$

У випадку матриць неповного рангу задача знаходження нормального узагальненого розв'язку в загальному випадку некоректна. Якщо ранг

матриці СЛАР (2.1) є  $r(\bar{A}) = k$ ,  $A = U\Sigma V^T$  – сингулярне розвинення матриці з (2.1), (2.3), а  $A^{(k)} = U\Sigma^{(k)}V^{(T)}$  (причому  $\Sigma^{(k)}$  – квадратна матриця, перші  $k$  діагональних елементів якої відмінні від нуля і співпадають з відповідними елементами матриці  $A$ , а всі інші елементи дорівнюють нулю), то нормальний узагальнений розв’язок  $x^{(k)}$  системи  $A^{(k)}x^{(k)} = b$  є ортогональною проекцією нормального псевдорозв’язку задачі (2.1), (2.3) на головний правий сингулярний підпростір матриці  $A$  розміру  $k$ . Тут необхідно розрізнити три випадки співвідношення рангів матриць  $\bar{A}$  та  $A$ .

Якщо  $\|\Delta A\| \|A^\#\| < 1$  і  $r(\bar{A}) = r(A)$ , то має місце оцінка

$$\frac{\|x - \bar{x}\|}{\|x\|} \leq \frac{h(A)}{1 - h(A)\varepsilon_A} \left( \left( 2 + h(A) \frac{\|r\|}{\|Ax\|} \right) \varepsilon_A + \varepsilon_b \right).$$

Якщо  $\|\Delta A\| \|A^\#\| < 0,5$  і  $r(A) > r(\bar{A}) = k$ , то має місце оцінка

$$\frac{\|x^{(k)} - \bar{x}\|}{\|\bar{x}\|} \leq \frac{h(A)}{1 - 2h(A)\varepsilon_A} \left( \left( 2 + h(A) \frac{\|r\|}{\|Ax\|} \right) \varepsilon_A + \varepsilon_b \right).$$

Якщо  $r(\bar{A}) > r(A) = l$ ,  $\|\Delta A\| \sigma_l^{-1} < 0,5$  і  $\bar{x}^{(l)}$  – проекція нормального псевдорозв’язку задачі (2.2) на правий головний сингулярний підпростір матриці  $\bar{A}$  розміру  $l$ , то має місце оцінка

$$\frac{\|\bar{x}^{(l)} - x\|}{\|\bar{x}\|} \leq \frac{\sigma_{\max} \sigma_l^{-1}}{1 - 2\|A\| \sigma_l^{-1}} \left( \left( 2 + \frac{\sigma_{\max}}{\sigma_l} \frac{\|\bar{b} - \bar{b}^{(l)}\|}{\|\bar{b}^{(l)}\|} \right) \varepsilon_A + \varepsilon_b \right),$$

де  $\bar{b}^{(l)}$  – проекція правої частини  $\bar{b}$  на головний лівий сингулярний підпростір розміру  $l$  матриці  $\bar{A}$ ,  $\sigma_{\max} \equiv \sigma_1$  – максимальне, а  $\sigma_l$  – мінімальне відмінне від нуля сингулярне число матриці  $\bar{A}^{(l)}$ .

Викладене вище свідчить, що для аналізу властивостей комп’ютерної задачі з матрицею неповного рангу в умовах наближених вихідних даних фундаментальну роль відіграє визначення рангу матриці.

Рангом матриці в умовах наближених вихідних даних (ефективним рангом або  $\delta$ -рангом) називають величину, що обчислюється за формулою

$$r_{\delta}(A) = \min_{\|A-B\| \leq \delta} r(B). \quad (2.9)$$

Це означає, що  $\delta$ -ранг матриці дорівнює мінімальному рангу серед рангів всіх матриць  $B$  в околі  $\|A-B\| \leq \delta$ . З [10] слідує, що якщо для  $r(\delta)$  –  $\delta$ -ранг матриці, то для її сингулярних чисел справедливі нерівності

$$\sigma_1 \geq \dots \geq \sigma_{r(\delta)} > \delta \geq \sigma_{r(\delta)+1} \geq \dots \geq \sigma_p, \quad \text{де } p = \min\{m, n\}.$$

Практичний алгоритм для знаходження  $\delta$ -рангу може бути реалізований так: знайти величину  $r$ , рівну найбільшому значенню  $i$  ( $i = 1, 2, \dots$ ), для якого виконується нерівність  $\delta / \sigma_i < 1$ ,  $\sigma_i \neq 0$ . Для аналізу значень рангу матриці в межах машинної точності величину  $\delta$  можна покласти рівною  $\text{macheps} \times \|B\|_{\infty}$  ( $\text{macheps}$  – найбільше число, для якого рівність  $1.0 + \text{macheps} = 1.0$  є справедлива на комп'ютерах при обчисленнях з плаваючою комою).

Оцінити обчислювальну похибку, тобто одержати інформацію про близькість комп'ютерного розв'язку  $\tilde{x}$  до точного розв'язку  $x$  системи (2.1), а в ряді випадків зменшити похибку комп'ютерного розв'язку, можна шляхом його ітераційного уточнення [7].

Ітераційне уточнення розв'язку реалізується за наступною схемою:

$$x^{(0)} = x, \quad r^{(s)} = b - Ax^{(s)}, \quad A \times \Delta x^{(s)} = r^{(s)}, \quad (2.10)$$

$$x^{(s+1)} = x^{(s)} + \Delta x^{(s)}, \quad \text{де } s = 0, 1, 2, \dots \quad (2.11)$$

При обчисленні  $\Delta x^{(s)}$  використовується розвинення матриці, яке вже отримано при розв'язуванні СЛАР прямим методом, тому процедура ітераційного уточнення не потребує значного додаткового часу обчислень. Обчислення нев'язки  $r_i^{(s)}$  виконується дещо з підвищеною довжиною машинного слова відносно основної розрядності. Наприклад, при виконанні основних обчислень з подвійною розрядністю обчислення нев'язки  $r_i^{(s)}$  можна виконати з підвищеною розрядністю.

Оцінка обчислювальної похибки розв'язку СЛАР визначається за формулою

$$E_{\text{обчис.}} \approx \frac{\|\Delta x_1\|}{\|x_2\|}, \quad (2.12)$$

де  $x_2$  – наближення до точного розв’язку, яке отримано за один крок ітераційного уточнення.

Якщо для матриці системи виду (2.1), (2.3) не виконується умова (2.4), то така матриця вважається виродженою в межах машинної точності, тобто в околі машинної похибки може знайтись вироджена матриця. Але такий же ефект може давати і матриця з великим числом обумовленості відносно відповідної розрядності. Такі СЛАР доцільно дослідити та розв’язати з використанням підвищеної розрядності обчислень.

Невиконання умови (2.4) в комп’ютері на послідовності розрядних сіток свідчить про виродженість вихідної матриці. В цьому випадку можна знайти наближення до псевдорозв’язків СЛАР, використовуючи метод регуляризації на основі SVD-розвинення.

Виконання умови (2.4) на підвищеній розрядності у відповідності зі значенням оцінки числа обумовленості матриці вказує на те, що вихідна задача є погано обумовлена відносно попередньої розрядності і є можливість отримати наближення до єдиного розв’язку задачі (2.2), (2.3), використовуючи з підвищеною розрядністю, наприклад, алгоритми  $LU$  або  $LL^T$  – розвинення матриць.

Для реалізації комп’ютерного дослідження математичних властивостей лінійних систем з наближеними даними з матрицями різної структури (щільних, стрічкових, розріджених довільної структури) та розв’язування з оцінками достовірності отриманих розв’язків на багатоядерних комп’ютерах з графічними прискорювачами створено гібридні алгоритми на основі відомих ефективних прямих та ітераційних методів, а саме: для щільних невивроджених та вироджених матриць, прямокутних матриць довільного рангу, розріджених різної структури невивроджених та вироджених матриць.

В кожному гібридному алгоритмі проводиться дослідження математичних властивостей СЛАР з наближеними даними та визначається їх відповідність вибраному гібридному алгоритму.

Як вже зазначалося, для обчислення числа обумовленості матриці за формулами (2.5) або (2.6) в комп'ютері необхідно обчислювати відповідно обернену матрицю  $A^{-1}$  або псевдообернену матрицю Мура–Пенроуза  $A^\#$ . Проте обчислення цих матриць потребує великих витрат часу та пам'яті комп'ютера, тому замість числа обумовленості доцільно використовувати його оцінку, яку в подальшому будемо позначати як  $\text{cond}A$ .

Дослідження математичних властивостей СЛАР з наближеними даними здійснюється на основі визначеної оцінки числа обумовленості матриці системи.

Для квадратної невинродженої матриці, в тому числі розрідженої структури, оцінка числа обумовленості обчислюється так:

1) розв'язати СЛАР (тут, зазвичай, використовується трикутне розвинення матриці  $A = LU$  або  $A^T = U^T L^T$ )

$$A^T y = e, \quad (2.13)$$

де  $e$  – вектор з компонентами  $\pm 1$ , знак яких вибирається так, щоб норма вектора  $y$  була найбільшою;

2) розв'язати СЛАР

$$Az = y; \quad (2.14)$$

3) обчислити оцінку числа обумовленості матриці

$$\text{cond}A = \|A\| \|z\| / \|y\|; \quad (2.15)$$

тут можна використовувати, наприклад, такі норми:

$$\|y\|_1 = \sum_{i=1}^n |y_i|, \quad \|z\|_1 = \sum_{i=1}^n |z_i|, \quad \|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|.$$

Для прямокутної матриці довільного рангу доцільно використовувати спектральне число обумовленості:

$$\text{cond}_s A = \frac{\sigma_{\max}}{\sigma_{\min}},$$

при обчисленні якої використовується сингулярне розв'язання матриці, де  $\sigma_{\max}$  – найбільше сингулярне число,  $\sigma_{\min}$  – найменше відмінне від нульового сингулярне число.

## 2.2 Двовимірний блочно-циклічний мульти-GPU алгоритм LU факторизації

Розглянемо задачу розв'язання системи лінійних алгебраїчних рівнянь на мульти-GPU системі [20].

Розглянемо систему лінійних алгебраїчних рівнянь

$$Ax = b$$

де  $A$  – щільна матриця порядку  $n$ .

Найбільш ефективним прямим методом розв'язання такої задачі є, як відомо, метод Гауса. Розв'язання системи полягає в розв'язанні таких підзадач:

- трикутне розв'язання матриці системи:

$$PA = LU$$

- розв'язання СЛАР з трикутними матрицями:

$$Ly = Pb$$

$$Ux = y \quad .$$

У цьому підрозділі розглядаються особливості реалізації запропонованого алгоритму на обчислювальних системах з мульти-GPU архітектурою [20]. Застосування кількох графічних прискорювачів дає змогу істотно скоротити час виконання обчислювально складних задач, підвищити масштабованість паралельних алгоритмів, забезпечити ефективніше використання апаратних

ресурсів та організувати розподілене зберігання великих обсягів даних у локальних пам'ятях окремих GPU.

Особливе значення такий підхід має для задач лінійної алгебри, зокрема для алгоритмів матричної факторизації, де основна частина обчислень пов'язана з виконанням однотипних операцій над блоками матриці. У цьому випадку мульти-GPU архітектура дозволяє поєднати паралельне виконання обчислювальних ядер з ефективною організацією обміну даними між графічними прискорювачами, що є необхідною умовою досягнення високої продуктивності та хорошої масштабованості алгоритму.

Існує декілька основних конфігурацій мульти-GPU систем: системи з кількома GPU в одному вузлі, системи з прямим GPU-GPU з'єднанням через NVLink або аналогічні інтерконекти та мережеві GPU-кластери. Кожна з цих конфігурацій визначає особливості організації обчислень, обміну даними та вибору програмної моделі.

Однією з ключових проблем при побудові паралельних алгоритмів є балансування навантаження. У методах факторизації активна частина матриці поступово зменшується, тому при послідовному розподілі даних частина процесорів швидко завершує роботу і простоює. Для усунення цього ефекту використовуються циклічні та блочно-циклічні схеми розподілу даних.

Найбільш ефективною для алгоритмів факторизації великих матриць є двовимірною блочно-циклічною схемою, за якої блоки матриці розподіляються між процесорами або GPU у двох напрямках. Якщо GPU організовано у вигляді решітки  $p \times q$ , то блок  $A_{ij}$  призначається GPU з координатами  $(i \bmod p, j \bmod q)$ . Такий підхід забезпечує рівномірний розподіл обчислень, зменшує простої GPU та добре узгоджується з блочним  $LU$ -алгоритмом, де основна частина обчислень припадає на оновлення хвостової підматриці.

Саме тому для реалізації  $LU$ -факторизації на multi-GPU архітектурі використовується двовимірний блочно-циклічний розподіл даних, який

дозволяє поєднати балансування навантаження, локальність обчислень і ефективно використання високопродуктивних матричних операцій.

Нехай  $A$  – квадратна матриця порядку  $n$ . Розіб'ємо її на квадратні блоки розміру  $s \times s$ . Не втрачаючи загальності міркувань, вважатимемо, що  $n/s$  – ціле число.

$$A = (A_{ij}), i, j = \overline{1, 2, 3, \dots, l}, l = n/s. \quad (2.16)$$

На  $k$ -ому кроці алгоритму ( $k = 1, 2, \dots$ ) підматрицю  $A^{(k)}$  порядку  $r = n - (k-1)s$ , яка містить останні  $r$  рядків і  $r$  стовпчиків у правому нижньому куті матриці  $A^{(k)}$ , представимо у вигляді:

$$A^{(k)} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = P \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix} = P \begin{pmatrix} L_{11}U_{11} & L_{11}U_{12} \\ L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22} \end{pmatrix}, \quad (2.17)$$

де  $A_{11}$  – блок, який має розмір  $s \times s$ ,  $A_{12}$  – блок, що має розмір  $s \times (r-s)$ ,  $A_{21}$  – блок розміру  $(r-s) \times s$ , прямокутна підматриця, що містить блоки, розташовані нижче провідного, а  $A_{22}$  – блок, який має розмір  $(r-s) \times (r-s)$ .  $P$  – матриця перестановок у випадку вибору головного елемента (нехай  $P = E$ ).

Таким чином рекурентний алгоритм блочної факторизації на  $k$ -ому блочному кроці ( $k = 1, 2, \dots, r-1$ ) матиме наступний вигляд:

$$A_{11} = L_{11}U_{11}, \quad (2.18)$$

$$U_{12} = (L_{11})^{-1} A_{12}, \quad (2.19)$$

$$L_{21} = A_{21}(U_{11})^{-1}, \quad (2.20)$$

$$\tilde{A}_{22} = A_{22} - L_{21}U_{12}. \quad (2.21)$$

**Розподіл даних.** У реалізації паралельної  $LU$ -факторизації ключову роль відіграє стратегія розподілу вхідної матриці  $A$  розміром  $n \times n$  між обчислювальними ресурсами. Для забезпечення балансування навантаження та ефективної організації комунікацій використовується двовимірна блочно-циклічна схема розподілу даних.

$GPU(i, j)$  це GPU, розташований в  $i$ -му рядку та  $j$ -му стовпчику двовимірної решітки  $p \times q$ , де  $i \in [0, p-1]$ ,  $j \in [0, q-1]$ . Суть блочно-циклічного розподілу полягає у тому, що блоки призначаються GPU згідно з їхніми координатами за модулем розміру мережі:

$$GPU(i, j) \leftarrow A_{kl} \text{ якщо } i = k \bmod p, j = l \bmod q.$$

Такий підхід дозволяє: забезпечити рівномірний розподіл обчислювального навантаження між усіма процесорами у ході факторизації матриці, незалежно від розмірності задачі; зменшити пікове навантаження на окремі GPU; мінімізувати глобальні комунікації та пов'язані з ними затримки; уникнути ситуацій, коли один GPU виконує більшість обчислень на останніх етапах алгоритму.

### Паралельна форма алгоритму

*Означення.* GPU, в якому знаходиться блок  $A_{11}$  у контексті (2.17) (на кожному кроці він може бути різним) назвемо головним GPU.

Стовпчик GPU, в якому знаходиться головний GPU назвемо головним стовпчиком GPU.

Рядок GPU, в якому знаходиться головний процесор назвемо головним рядком GPU.

Тоді паралельна форма алгоритму матиме наступний вигляд:

для  $k = 1, 2, \dots, r-1$  виконуємо:

- в головному GPU факторизуємо  $A_{11}$

$$A_{11} = L_{11}U_{11};$$

- $L_{11}$  розсилаємо вправо і вліво по головному рядку GPU;
- $U_{11}$  розсилаємо вверх і вниз по головному стовпчику GPU;
- одночасно у всіх GPU головного рядка GPU виконуємо

$$U_{12} = (L_{11})^{-1} A_{12};$$

- одночасно у всіх GPU головного стовпчика GPU виконуємо

$$L_{21} = A_{21}(U_{11})^{-1};$$

- всі GPU головного рядка передають вверх і вниз свої частини (панелі)  $U_{12}$ ;
- всі GPU головного стовпчика передають вправо і вліво свої частини (панелі)  $L_{21}$ ;
- одночасно у всіх  $p \times q$  GPU виконуємо

$$\tilde{A}_{22} = A_{22} - L_{21}U_{12}.$$

Для оцінки ефективності паралельного алгоритму  $LU$ -факторизації з двовимірним блочно-циклічним розподілом доцільно розглянути теоретичну модель часу виконання. Нехай  $n_0$  – група одночасно виконуваних арифметичних операцій на GPU  $t_g$  – час виконання однієї групи арифметичних операцій,  $t_o$  – час передачі одного слова між GPU (обмін),  $t_c$  – час однієї глобальної синхронізації. Розмірність матриці –  $n \times n$ , сітка GPU –  $p \times q$ , розмір блоку –  $s \times s$ . Прискорення і ефективність обчислюються за наступними співвідношеннями:

$$S_p = \frac{T_1}{T_p}, \quad E_p = \frac{S_p}{p}, \quad T_p = T + T_o + T_c,$$

де  $T_1$  – час виконання обчислень на 1 GPU,  $T_p$  – час виконання обчислень на  $p$  GPU,  $T = N_p t_g$  – час паралельних обчислень,  $T_o = M t_o$  – час обміну даними,  $T_c = Q t_c$  – час налаштування зв'язку.

**Теорема 2.1** Час виконання  $LU$ -факторизації щільної матриці порядку  $n$  на одному GPU оцінюється формулою

$$T_1 = \frac{2n^3 t_g}{3n_0}.$$

Доведення. Нехай задано щільну квадратну матрицю

$$A = \{a_{ij}\}_{i,j=1}^n \in \mathbb{R}^n.$$

Потрібно виконати її  $LU$ -факторизацію, тобто подати матрицю у вигляді добутку

$$A = LU,$$

де  $L$  — нижня трикутна матриця, а  $U$  — верхня трикутна матриця. Розглянемо прямий хід  $LU$ -факторизації для щільної матриці без урахування міжпроцесорних комунікацій, оскільки у теоремі йдеться про виконання на одному GPU.

На  $k$ -му кроці алгоритму, де  $k = 1, 2, \dots, n - 1$ , вже оброблено перші  $k$  рядків і стовпчиків. Залишається підматриця розміру

$$(n - k) \times (n - k).$$

Кількість її елементів дорівнює

$$(n - k)^2.$$

Кожен елемент цієї підматриці оновлюється за формулою

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik} u_{kj}, \quad i, j = k+1, \dots, n.$$

У цій формулі для кожного елемента виконується одна операція множення  $l_{ik} u_{kj}$  і одна операція віднімання від поточного значення  $a_{ij}^{(k)}$ . Отже, на один елемент припадає дві арифметичні операції.

Тому кількість операцій на  $k$ -му кроці дорівнює:

$$Q_k = 2(n - k)^2.$$

Загальна кількість арифметичних операцій оновлення підматриць визначається сумою:

$$Q(n) = 2 \sum_{k=1}^{n-1} (n - k)^2.$$

Використаємо стандартну формулу суми квадратів та провівши скорочення отримуємо:

$$Q(n) = \frac{(n-1)n(2n-1)}{3} = \frac{2n^3}{3} - n^2 + \frac{n}{3}.$$

Для задач великого порядку головний внесок у кількість операцій дає кубічний член

$$\frac{2n^3}{3}.$$

Тому кількість арифметичних операцій  $LU$ -факторизації щільної матриці оцінюється як

$$Q(n) \approx \frac{2n^3}{3}.$$

Нехай  $t_g$  — середній час виконання однієї арифметичної операції на одному GPU і одночасно виконуються  $n_0$  — арифметичних операцій. Отже, час виконання визначається добутком кількості арифметичних операцій на час однієї операції:

$$T_1 = \frac{2n^3 t_g}{3n_0}.$$

Теорему доведено.

**Теорема 2.2** Час виконання  $LU$ -факторизації щільної матриці порядку  $n$  на  $p^2$  GPU для топології зв'язків «Тор» оцінюється формулою

$$T_p = \frac{2n^3 t_g}{3p^2 n_0} + \frac{n^2}{2} t_o + \frac{2pn}{s} t_c$$

Доведення. Як показано у попередній теоремі, основну частку операцій алгоритму складають операції оновлення підматриці. Підматриця розподілена по сітці GPU  $p \times q$ . Нехай для нашого алгоритму  $p=q$ . Тоді кількість арифметичних операцій виконуваних паралельно оцінюється величиною

$$N_p \approx \frac{2s}{p^2} \sum_{k=1}^l (n - ks)^2 \approx \frac{2}{3p^2} n^3,$$

де  $l = \frac{n}{s}$ .

В процесі виконання алгоритму нам потрібно обмінюватися даними для модифікації підматриці та розсилати факторизований діагональний блок по всій сітці GPU. Обсяг даних які пересилаються оцінюється величиною

$$M \approx s \sum_{k=1}^l (n - sk) \approx \frac{n^2}{2}.$$

Загальна кількість синхронізацій в алгоритмі обчислюється величиною

$$Q \approx 2pl = \frac{2pn}{s}.$$

Підставивши значення кількості операцій, обсягу даних та кількість синхронізацій у формулу для обчислення часу виконання паралельного алгоритму отримаємо:

$$T_p = \frac{2n^3 t_g}{3p^2 n_0} + \frac{n^2}{2} t_o + \frac{2pn}{s} t_c,$$

де  $t_g$  — середній час виконання однієї арифметичної операції на одному GPU і одночасно виконуються  $n_0$  — арифметичних операцій.

Теорему доведено.

**Теорема 2.3** Прискорення двовимірного блочно-циклічного алгоритму  $LU$ -факторизації на  $p \times p$  GPU для топології зв'язків «Тор» оцінюється формулою

$$S_p = \frac{T_1}{T_p} \approx \frac{p^2}{1 + \frac{3p^2}{4n} \tau_o + \frac{3p^2 n}{sn^2} \tau_c},$$

де

$$\tau_o = \frac{t_o}{t_g} n_0, \quad \tau_c = \frac{t_c}{t_g} n_0.$$

Доведення. Використаємо величини з теорем 2.1 та 2.2 та підставимо їх у формулу для обчислення прискорення

$$S_p = \frac{T_1}{T_p} \approx \frac{\frac{2n^3 t_g}{3n_0}}{\frac{2n^3 t_g}{3p^2 n_0} + \frac{n^2}{2} t_o + \frac{2pn}{s} t_c}$$

Поділимо вирази в чисельнику та знаменнику на  $\frac{2n^3 t_g}{3n_0}$

$$S_p = \frac{T_1}{T_p} \approx \frac{1}{\frac{1}{p^2} + \frac{3n^2}{4n} \tau_o + \frac{3n}{sn^2} \tau_c}.$$

Розділимо чисельник та знаменник на  $\frac{1}{p^2}$  та провівши спрощення отримуємо

$$S_p = \frac{T_1}{T_p} \approx \frac{p^2}{1 + \frac{3p^2}{4n} \tau_o + \frac{3p^2 n}{sn^2} \tau_c}$$

Теорему доведено.

Ці теореми дозволяють проаналізувати вплив порядку системи, розміру блоку та швидкодії обміну і синхронізації на масштабованість алгоритму.

**Особливості програмної реалізації.** Програмна реалізація паралельного алгоритму LU-факторизації з блочно-циклічним розподілом даних в обчислювальному середовищі з кількома GPU орієнтована на ефективне використання локальної пам'яті графічних прискорювачів, зменшення міжпристрєвих обмінів та забезпечення рівномірного завантаження обчислювальних ресурсів. Основна увага приділяється узгодженню локальних обчислень, передавання даних і синхронізації між GPU.

Матриця попередньо поділяється на блоки однакового розміру, які розподіляються між GPU за блочно-циклічною схемою. Такий підхід дозволяє рівномірніше розподілити навантаження між пристроями та уникнути концентрації значної частини обчислень на одному GPU. Кожний графічний прискорювач зберігає лише ті блоки, які належать йому відповідно до прийнятої схеми розподілу.

На кожному кроці алгоритму виконується факторизація поточного діагонального блоку, який розміщений у локальній пам'яті одного з GPU. Для цього можуть використовуватися оптимізовані засоби бібліотеки cuSolver, що забезпечують ефективне виконання локальної LU-факторизації. У разі застосування часткового вибору головного елемента інформація про перестановки рядків має бути узгоджена з іншими пристроями.

Після факторизації діагонального блоку виконуються обчислення для відповідних блоків поточного блочного рядка та блочного стовпця. Ці операції зводяться до розв'язування трикутних систем і можуть реалізовуватися засобами бібліотеки cuBLAS. Оскільки відповідні блоки

розподілені між різними GPU, їх обробка виконується паралельно на тих пристроях, де вони локально зберігаються.

Подальше оновлення активних блоків матриці виконується за допомогою блокових матрично-матричних операцій. Для цього використовуються оптимізовані процедури cuBLAS, які забезпечують високу продуктивність на GPU та добре узгоджуються з архітектурою масового паралелізму. Саме ці операції становлять основну частину обчислювального навантаження алгоритму.

Передавання необхідних блоків між GPU здійснюється за допомогою колективних комунікацій або прямих асинхронних копіювань між пристроями. Для цього можуть використовуватися NCCL, CUDA-aware MPI або механізм peer-to-peer доступу між пам'яттю GPU. Така організація дозволяє зменшити участь центрального процесора в обміні даними та уникнути зайвого копіювання через оперативну пам'ять.

Для підвищення ефективності на кожному GPU використовуються кілька асинхронних CUDA-потоків, які обслуговують різні типи операцій: факторизацію, розв'язування трикутних систем, блокові оновлення та передавання даних. Синхронізація між залежними етапами виконується за допомогою CUDA-подій, що дозволяє уникати глобального блокування пристроїв і частково перекривати обчислення з комунікаціями.

Усі робочі буфери для проміжних блоків, комунікаційних даних і службових масивів доцільно виділяти заздалегідь у пам'яті GPU та повторно використовувати протягом усього виконання алгоритму. Це зменшує накладні витрати на динамічне керування пам'яттю та підвищує стабільність часу виконання. Оцінювання продуктивності реалізації виконується за часом основних етапів, продуктивністю у GFLOP/s, прискоренням відносно однієї GPU та паралельною ефективністю.

### 2.3 Апробація розробленого паралельного алгоритму

Апробація розробленого паралельного алгоритму, що базується на ідеології двовимірного блочно-циклічного розподілу даних, проводилася на задачі обробки квадратної матриці порядку  $N = 20000$ . Експерименти виконувалися на одновузловому кластері СКІТ [104] g4301, обладнаному 8 GPU Nvidia GeForce RTX 2080 Ti. Основна мета апробації – визначення ефективності масштабування та встановлення оптимальних параметрів розподілу.

Залежність прискорення від кількості графічних прискорювачів наведено на рис. 2.1. Аналіз отриманих результатів свідчить, що застосування двовимірного блочно-циклічного розподілу даних забезпечує високий рівень паралелізму обчислень і дозволяє ефективно використовувати апаратні ресурси мульти-GPU системи. Спостерігається майже лінійне масштабування, що вказує на домінування локальних обчислень над комунікаційними витратами та підтверджує достатню збалансованість обчислювального навантаження між пристроям.

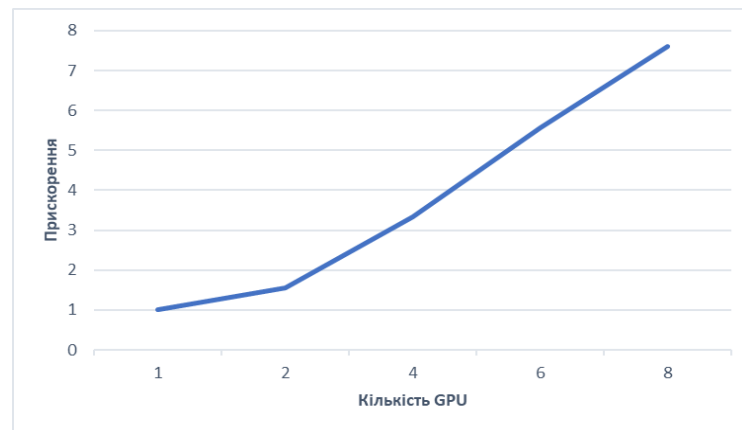


Рисунок 2.1 – Залежність прискорення від кількості GPU

На рис. 2.2 наведено залежність ефективності паралельного виконання від кількості задіяних графічних прискорювачів. Отримані результати демонструють стабільне зростання ефективності при масштабуванні від одного до восьми GPU, ефективність підвищується з приблизно 74% для одного прискорювача до понад 90% при використанні шести та восьми GPU.

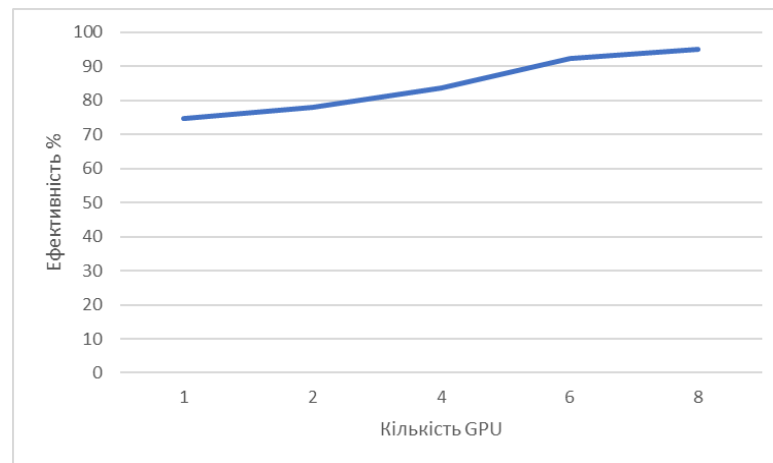


Рисунок 2.2 – Залежність ефективності від кількості GPU

Така поведінка свідчить про високу збалансованість запропонованої двовимірної блочно-циклічної схеми розподілу даних і про здатність алгоритму ефективно залучати додаткові апаратні ресурси без істотного збільшення комунікаційних витрат. Зростання ефективності зі збільшенням числа прискорювачів також підтверджує домінування локальних обчислень у структурі алгоритму та оптимальний характер організації між-GPU обмінів. Таким чином, результати експериментів засвідчують, що запропонований підхід забезпечує продуктивне масштабування навіть за умов значного збільшення кількості обчислювальних пристроїв.

Для підтвердження оптимальності обраної ідеології розподілу було проведено аналіз впливу розміру блоку  $s$  на загальний час виконання алгоритму при фіксованій кількості GPU  $p=1$  (рис. 2.3.).

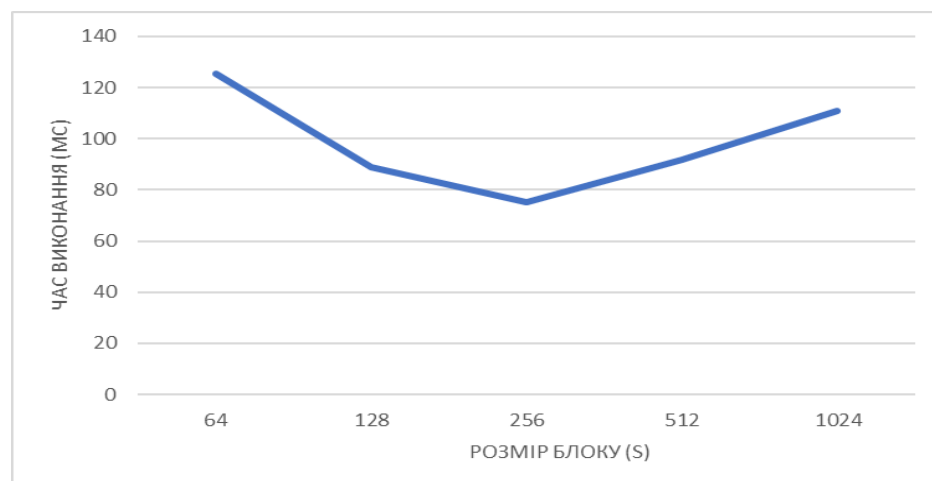


Рисунок 2.3 – Залежність часу виконання від розміру блоку  $s$

Проведені чисельні експерименти повністю підтвердили ефективність розробленої ідеології двовимірного блочно-циклічного розподілу даних та її програмної реалізації. Було досягнуто виняткової ефективності (95.0 %) і визначено оптимальний розмір блоку  $s = 256$ , що дозволяє рекомендувати алгоритм для високопродуктивних обчислень на GPU-акселерованих системах.

### **Ефективність паралельного алгоритму $LU$ -факторизації.**

Ефективність паралельного алгоритму  $LU$ -факторизації в обчислювальному середовищі з кількома GPU визначається не лише формальною структурою алгоритму, а й низкою практичних чинників, пов'язаних з архітектурою апаратного забезпечення, організацією пам'яті, характеристиками міжпристроєвих комунікацій, способом розподілу даних між GPU та ефективністю програмної реалізації.

Обчислювальна складність  $LU$ -факторизації щільної матриці порядку  $n \times n \times n$  є пропорційною  $n^3$ , тоді як обсяг даних, що передаються між обчислювальними пристроями у процесі комунікацій, має порядок  $n^2$ . Отже, зі збільшенням розмірності задачі відносна частка комунікаційних витрат зменшується порівняно з обсягом арифметичних операцій. Саме тому для великих матриць алгоритм демонструє кращу масштабованість на системах з кількома GPU.

Для матриць малої та середньої розмірності, зокрема порядку кількох тисяч, обчислювальне навантаження може бути недостатнім для повного завантаження всіх графічних прискорювачів. У цьому випадку значну частину часу займають передавання даних, синхронізація, очікування завершення комунікаційних операцій та накладні витрати керування. Тому алгоритм набуває ознак communication-bound, коли продуктивність обмежується не швидкістю виконання арифметичних операцій, а швидкістю обміну даними між пристроями.

З практичної точки зору це означає, що ефективне масштабування  $LU$ -факторизації в середовищі мульти-GPU досягається лише для достатньо

великих значень  $n$ , коли обсяг локальних обчислень на кожному GPU істотно перевищує витрати на комунікації. Водночас збільшення розмірності задачі потребує значного обсягу локальної та сумарної GPU-пам'яті, а також високої пропускної здатності каналів зв'язку. Тому реальна ефективність алгоритму визначається компромісом між обчислювальною складністю, доступною пам'яттю та характеристиками комунікаційної підсистеми.

Важливим параметром алгоритму є розмір блоку  $s$ . Він визначає баланс між локальними обчисленнями, обсягом передавання даних, ефективністю використання пам'яті та рівномірністю розподілу навантаження. Оптимальний вибір  $s$  має забезпечувати високе завантаження обчислювальних ядер GPU, ефективне використання ієрархії пам'яті, зменшення частоти міжпристроєвих обмінів та збалансований розподіл блоків у межах двовимірної блочно-циклічної схеми.

Надмірно великий розмір блоку знижує гнучкість розподілу даних між GPU і може призводити до дисбалансу навантаження, коли частина пристроїв простоює в очікуванні завершення обчислень на інших. З іншого боку, занадто малі блоки збільшують кількість синхронізацій і комунікаційних операцій, що призводить до значних накладних витрат. Тому на практиці розмір блоку добирається емпірично або за допомогою автоматичного налаштування з урахуванням архітектури GPU, обсягу локальної пам'яті, швидкодії міжз'єднань і характеру задачі. Для сучасних GPU ефективні значення  $s$  найчастіше належать діапазону 128–512.

Суттєвий вплив на продуктивність має тип інтерконекту між пристроями. Наявність високошвидкісного з'єднання між GPU, зокрема NVLink, дає змогу істотно зменшити час передавання блоків матриці та ефективно реалізовувати колективні операції обміну даними, необхідні під час факторизації й оновлення підматриць. До таких операцій належать передавання діагональних і робочих блоків, розсилка даних між GPU та синхронізація результатів проміжних обчислень.

За відсутності NVLink обмін між GPU зазвичай виконується через PCIe, що обмежує пропускну здатність і збільшує затримки. У таких умовах продуктивність реалізації на кількох GPU може знижуватися, особливо для задач, у яких частка комунікацій є значною. У багатовузлових конфігураціях для зменшення цих втрат використовують високошвидкісні мережі InfiniBand та технологію GPUDirect RDMA, яка забезпечує прямий обмін даними між пам'яттю GPU різних вузлів без зайвого копіювання через оперативну пам'ять CPU.

Для досягнення високої ефективності програмна реалізація алгоритму має враховувати особливості сучасних GPU-архітектур. Локальні обчислювальні операції доцільно виконувати за допомогою оптимізованих бібліотек лінійної алгебри, зокрема cuBLAS та cuSolver, які забезпечують високу продуктивність для операцій факторизації, розв'язування трикутних систем і матрично-матричних оновлень. Колективні обміни між GPU можуть реалізовуватись засобами NCCL або CUDA-aware MPI, що дає змогу ефективно організувати передавання даних між пристроями без зайвих проміжних копій.

Додаткове підвищення продуктивності досягається за рахунок асинхронного виконання операцій у кількох CUDA streams, що дозволяє перекривати обчислення з передаванням даних. Використання look-ahead та double buffering дає змогу починати обробку наступних блоків або оновлення віддалених підматриць ще до повного завершення поточного етапу факторизації. Це зменшує прості графічних прискорювачів, підвищує рівень конвеєризації та покращує масштабованість алгоритму.

Таким чином, ефективність паралельної  $LU$ -факторизації в обчислювальному середовищі з кількома GPU визначається взаємодією трьох основних складових: достатньо великого обсягу арифметичних операцій, раціонального розподілу даних між GPU та мінімізації комунікаційних накладних витрат. Найкращі результати досягаються для великих матриць, коли обчислювальна частина алгоритму домінує над передаванням даних, а

двовимірний блочно-циклічний розподіл забезпечує рівномірне завантаження пристроїв і ефективне використання ресурсів системи з кількома GPU.

## 2.4 Висновки до розділу

У підрозділі 2.1 наведено результати дослідження достовірності розв'язків СЛАР з наближеними даними з врахуванням спадкової та обчислювальної похибки.

У підрозділі 2.2 запропоновано двовимірний блочно-циклічний паралельний алгоритм  $LU$ -факторизації щільних матриць, який забезпечує ефективний розподіл обчислювального навантаження між графічними прискорювачами у мульти-GPU середовищі. При цьому:

- запропоновано двовимірну блочно-циклічну схему розподілу даних, що забезпечує рівномірне балансування навантаження між обчислювальними пристроями та дозволяє зменшити вплив комунікаційних витрат при виконанні алгоритму;
- отримано теоретичні оцінки ефективності та прискорення запропонованого алгоритму для гібридних комп'ютерних систем та проведено експериментальні дослідження ефективності, яке підтвердило теоретичні оцінки;
- проведено апробацію алгоритму на суперкомп'ютері, показано ефективне використання ресурсів графічних прискорювачів.

Отримані результати створюють основу для подальшого розроблення високопродуктивних паралельних алгоритмів розв'язування прикладних задач.

### **РОЗДІЛ 3. БЛОЧНИЙ АЛГОРИТМ РОЗВИНЕННЯ СТРІЧКОВИХ МАТРИЦЬ**

У задачах математичного моделювання складних технічних об'єктів значна частина обчислювальних витрат пов'язана з розв'язуванням систем лінійних алгебраїчних рівнянь. Такі системи виникають при чисельному розв'язуванні диференціальних рівнянь, що описують фізичні процеси у конструкціях, матеріалах та інженерних системах.

Одним із важливих прикладних напрямів є комп'ютерне моделювання життєвого циклу зварних конструкцій. При дослідженні процесів зварювання, термічного навантаження, накопичення пошкоджень та втрати міцності конструкцій використовуються чисельні методи, зокрема метод скінченних елементів. У результаті дискретизації відповідних математичних моделей виникають системи лінійних алгебраїчних рівнянь з розрідженими матрицями спеціальної структури.

Для багатьох таких задач характерною є стрічкова або близька до стрічкової структура матриці коефіцієнтів, що обумовлена локальним характером взаємодії елементів розрахункової сітки. Використання цієї структурної особливості дозволяє суттєво зменшити обсяг пам'яті та кількість арифметичних операцій, необхідних для розв'язування системи.

Одним із перспективних підходів до підвищення ефективності таких обчислень є використання блочних алгоритмів факторизації матриць. У цих алгоритмах матриця розбивається на блоки, що дозволяє локалізувати обчислення у підматрицях, підвищити рівень паралелізму та ефективно використовувати кеш-пам'ять процесорів і ресурси графічних прискорювачів.

Застосування блочних алгоритмів є особливо ефективним у задачах математичного моделювання складних інженерних систем, зокрема при дослідженні напружено-деформованого стану та процесів деградації матеріалів у зварних конструкціях. У таких задачах виникають великі

системи лінійних рівнянь зі стрічковими або близькими до стрічкових матрицями, для яких використання спеціалізованих алгоритмів факторизації дозволяє значно підвищити ефективність обчислень.

У цьому розділі розглядається блочний алгоритм  $LU$ -розвинення стрічкових матриць [58], орієнтований на використання сучасних високопродуктивних комп'ютерних систем. Наведено опис алгоритму, виконано аналіз його обчислювальної складності та досліджено можливості підвищення ефективності обчислень при використанні блочної організації алгоритму.

### 3.1 Постановка задачі

Розглянемо систему лінійних алгебраїчних рівнянь

$$Ax = b \quad (3.1)$$

де матриця  $A$  – стрічкова несиметрична невідроджена,  $n$  – порядок матриці,  $m_l$  – кількість нижніх діагоналей,  $m_u$  – кількість верхніх діагоналей, та  $m_l, m_u \ll n$ .

Найбільш ефективним прямим методом розв'язання такої задачі є, як відомо, метод Гауса. Розв'язання системи (3.1) полягає в розв'язанні таких підзадач:

- трикутне розвинення матриці системи:

$$PA = LU \quad (3.2)$$

- розв'язання СЛАР з трикутними матрицями:

$$Ly = Pb \quad (3.3)$$

$$Ux = y . \quad (3.4)$$

Отже, розглянемо алгоритм  $LU$ -розвинення стрічкової матриці СЛАР (3.1)  $A$  порядку  $n$  з  $m_l$  піддіагоналями та  $m_u$  наддіагоналями на комп'ютерах з багатоядерними процесорами та графічними прискорювачами.

В загальному випадку, як результат перестановок, кількість наддіагоналей у верхній трикутній матриці  $U$  може збільшитись до  $m_u + m_l$ , нижня трикутна матриця  $L$  та вектор  $Pb$  в явному вигляді не формуються, а перестановки враховуються при розв'язуванні СЛАР  $Ly = Pb$ . Матрицю  $A$  поділено на квадратні блоки порядку  $s$  (для спрощення викладу вважатимемо, що  $n = Ns$ ,  $m_l = M_L s$ ,  $m_l + m_u = M_U s$ ):

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,M_U+1} & 0 & 0 & \cdots & 0 \\ A_{2,1} & A_{2,2} & \cdots & A_{2,M_U+1} & A_{2,M_U+2} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{M_L+1,1} & A_{M_L+1,2} & & & & & & \\ 0 & A_{M_L+2,2} & & & & & & \\ 0 & 0 & & & \ddots & \vdots & \vdots & \\ \vdots & \vdots & & & \cdots & A_{N-1,N-1} & A_{N-1,N} & \\ 0 & 0 & & & \cdots & A_{N,N-1} & A_{N,N} & \end{pmatrix} .$$

### 3.2. Блочний алгоритм

Для кожного  $I = 1, \dots, N$  виконуються наступні кроки:

1)  $LU$ -розвинення з частковим вибором головного елемента стовпчика блоків – прямокутної підматриці, яка складається з блоків  $A_{I,I}, A_{I+1,I}, A_{I+2,I}, \dots, A_{K,I}$ , де  $K = \min(I + M_L, N)$  :

$$P_I \begin{pmatrix} A_{I,I} \\ \vdots \\ A_{K,I} \end{pmatrix} = \begin{pmatrix} L_{I,I} \\ \vdots \\ L_{K,I} \end{pmatrix} U_{I,I} . \quad (3.5)$$

2) Перестановка рядків в тих підматрицях, де це необхідно, використовуючи матрицю  $P_I$ .

3) Обчислення рядка блоків  $U_{I,I+1}, U_{I,I+2}, \dots, U_{I,J}$  матриці  $U$ , де  $J = \min(I + M_U, N)$ , як розв'язок матричної СЛАР з нижньою трикутною матрицею:

$$L_{I,I} (U_{I,I+1} \ \dots \ U_{I,J}) = (A_{I,I+1} \ \dots \ A_{I,J}). \quad (3.6)$$

4) Для  $I < Ns$ -рангова модифікація блоків  $A_{M,L}$ , де  $M = I+1, \dots, K$  та  $L = I+1, \dots, J$ , за формулою:

$$A_{M,L} \leftarrow A_{M,L} - L_{M,I} U_{I,L}. \quad (3.7)$$

Зауважимо, що на  $I$ -му кроці цього алгоритму операції проводяться тільки з блоками підматриці розміру  $(J+1)s \times (K+1)s$ , лівий верхній блок якої  $A_{I,I}$ .

Якщо частковий вибір головного елемента проводити лише в провідному діагональному блоці  $A_{I,I}$ , то перший пункт можна розділити на два:

1а)  $LU$ -розвинення  $A_{I,I} = P_I L_{I,I} U_{I,I}$ ;

1б) обчислення стовпчика блоків  $L_{I+1,I}, L_{I+2,I}, \dots, L_{K,I}$  матриці  $L$ , як розв'язок матричної СЛАР з нижньою трикутною матрицею:

$$U_{I,I}^T (L_{I+1,I}^T \ \dots \ L_{K,I}^T) = (A_{I+1,I}^T \ \dots \ A_{K,I}^T).$$

Такий прийом дозволяє зменшити можливу кількість наддіагоналей у верхній трикутній матриці  $U$  – не більше, ніж  $m_u + s$ , що зменшує загальну кількість арифметичних операцій. Аналіз формул (3.5)–(3.7) свідчить, що основна частина обчислювального навантаження в алгоритмі припадає на операції над щільними блоками. Завдяки цьому реалізація відповідних етапів алгоритму може бути здійснена із застосуванням оптимізованих програмних модулів матрично-матричних операцій, наявних у сучасних бібліотеках лінійної алгебри. Такий підхід дає змогу підвищити продуктивність програмної реалізації, забезпечити кращу локалізацію даних у пам'яті та ефективно використовувати обчислювальні можливості сучасних гібридних архітектур.

Для паралельного алгоритму рядки блоків вихідної матриці  $A$  і матриць розвинення  $L$ ,  $U$  розподіляються циклічно між процесами так, щоб кожен процес мав хоча б один рядок блоків, що модифікується згідно (3.7) на даному кроці. Під потоком вважаємо GPU-процес або процес OpenMP.

Порядок і ефективність виконання цього алгоритму великою мірою залежать від множини елементів стовпчика матриці, на якій виконується вибір головного елемента. Найбільш ефективним є варіант, коли головний елемент вибирається тільки серед елементів стовпчика провідного діагонального блоку, менш ефективним є вибір серед елементів стовпчика, які розподілені провідному процесу, а найменш ефективним (з точки зору кількості операцій, обмінів та синхронізацій) є вибір серед всіх ненульових елементів стовпчика.

### 3.2 Прискорення та ефективності алгоритму

**Теорема 3.1.** Час виконання  $LU$ -факторизації стрічкової матриці порядку  $n$  на одному GPU оцінюється формулою

$$T_1 \approx \frac{nm_l(m_l + m_u)t_g}{n_0}$$

**Доведення.** У стрічковій матриці ненульові елементи зосереджені лише в межах обмеженої області навколо головної діагоналі. Тому під час виконання  $LU$ -факторизації обчислення проводяться не над усією матрицею, а лише над елементами, що належать до стрічкового профілю.

Для кожного кроку факторизації кількість активних елементів визначається кількістю піддіагоналей  $m_l$  та кількістю  $m_u$  наддіагоналей. Кількість арифметичних операцій для факторизації одного рядка оцінюється величиною

$$m_l(m_l + m_u).$$

Оскільки факторизація стрічкової несиметричної матриці полягає у факторизації  $n$  її рядків, загальна кількість арифметичних операцій для  $LU$ -факторизації стрічкової матриці оцінюється величиною

$$N \approx nm_l(m_l + m_u).$$

Тому справедлива наступна оцінка часу виконання факторизації на одному GPU

$$T_1 \approx \frac{nm_l(m_l + m_u)t_g}{n_0}$$

Теорему доведено.

**Теорема 3.2.** Час реалізації паралельного алгоритму  $LU$ -факторизації стрічкової матриці порядку  $n$  на  $p$  GPU при топології «Кільце» оцінюється величиною:

$$T_p \approx \frac{nm_l(m_l + m_u)t_g}{pn_0} + \frac{pnt_c}{s} + pn(m_l + m_u)t_o$$

**Доведення.** У паралельній реалізації  $LU$ -факторизації стрічкової матриці обчислення виконуються лише в межах стрічки. Для кожного кроку факторизації кількість активних елементів визначається кількістю піддіагоналей  $m_l$  та наддіагоналей  $m_u$ . Тому загальна кількість арифметичних операцій у послідовному випадку оцінюється величиною

$$N_p \approx nm_l(m_l + m_u).$$

За умови рівномірного розподілу блоків матриці між  $p$  GPU обчислювальне навантаження розподіляється приблизно порівну. З урахуванням коефіцієнта  $n_0$  обчислювальна складова часу виконання має вигляд

$$N_p \approx \frac{nm_l(m_l + m_u)}{pn_0}.$$

Однак у multi-GPU реалізації загальний час визначається не лише арифметичними операціями, а й витратами на організацію обмінів між GPU. Якщо матриця обробляється блоками розміру  $s$ , то кількість блокових кроків приблизно дорівнює

$$n/s$$

У кільцевій топології передавання даних відбувається послідовно між сусідніми GPU. На кожному блоковому кроці в обмінах бере участь  $p$  GPU, тому сумарні накладні витрати на організацію обмінів і синхронізацію оцінюються як

$$M \approx \frac{pn}{s}.$$

Крім того, на кожному етапі необхідно передавати між GPU дані, пов'язані з активною стрірковою областю матриці. Обсяг таких даних пропорційний  $n(m_l+m_u)$ , а з урахуванням проходження інформації по кільцю між  $p$  GPU комунікаційна складова оцінюється величиною.

$$Q \approx pn(m_l+m_u).$$

Отже, загальний час паралельного виконання дорівнює сумі обчислювальної, синхронізаційної та комунікаційної складових:

Підставляючи отримані оцінки, маємо

$$T_p \approx \frac{nm_l(m_l+m_u)t_g}{pn_0} + \frac{pnt_c}{s} + pn(m_l+m_u)t_o$$

Теорему доведено.

Отримана теорема показує, що блочний  $LU$ -алгоритм для стрічкових матриць має високий рівень внутрішнього паралелізму, який лінійно зростає зі збільшенням ширини стрічки. Домінуючі обчислення зводяться до незалежних операцій матрично-матричного множення, що робить алгоритм особливо ефективним для багатоядерних та графічних обчислювальних систем. Водночас максимальне прискорення обмежується геометрією стріркової структури, що є фундаментальною властивістю класу задач.

**Теорема 3.3.** Прискорення блочно-циклічного алгоритму  $LU$ -факторизації на  $p$  GPU на топології «Кільце» має оцінку

$$S_p \approx \frac{p}{1 + \frac{p^2\tau_c}{m_l(m_l+m_u)} + \frac{p^2\tau_o}{m_l}}$$

$$\tau_c = \frac{n_0 t_c}{s t_g} \quad \tau_o = \frac{n_0 t_o}{t_g}$$

де , .

**Доведення.** За означенням:

$$S_p = \frac{T_1}{T_p}$$

де  $T_1$  — час виконання алгоритму на одному процесі,  $T_p$  — на  $p$  процессах.

З попередніх результатів:

$$T_1 \approx \frac{nm_l(m_l + m_u)t_g}{n_0},$$

$$T_p \approx \frac{nm_l(m_l + m_u)t_g}{pn_0} + \frac{pnt_c}{s} + pn(m_l + m_u)t_o.$$

Підставляємо  $T_1$  і  $T_p$  у визначення  $S_p$  отримаємо:

$$S_p = \frac{T_1}{T_p} \approx \frac{p}{1 + \frac{p^2\tau_c}{m_l(m_l + m_u)} + \frac{p^2\tau_o}{m_l}}.$$

Теорема доведена.

Отримана оцінка показує, що блочно-циклічний  $LU$ -алгоритм для стрічкових матриць забезпечує майже лінійне прискорення за умови, що кількість процесів не перевищує величини, пропорційної квадратному кореню з добутку напівширин стрічки. Комунікаційні витрати, зумовлені розсилкою діагональних і наддіагональних блоків, накладають фундаментальне обмеження на подальше зростання прискорення, незалежно від розміру матриці.

### 3.3 Апробація алгоритму

Основною метою апробації було підтвердження коректності запропонованої алгоритмічної схеми, перевірка збереження стрічкової структури матриці у процесі факторизації, а також дослідження впливу порядку матриці, ширини стрічки, розміру блоку та кількості GPU на час виконання алгоритму.

Тестові матриці формувалися як несиметричні стрічкові матриці з нижньою та верхньою півширинами стрічки  $m_l$  і  $m_u$ . Для забезпечення чисельної стійкості тестові матриці формувалися з діагональним переважанням. Вектор правої частини задавався за наперед відомим точним розв'язком  $x$ .

Таблиця 3.1 – Параметри тестових стрічкових матриць

| № тесту | Порядок матриці $n$ | Нижня півширина $m_l$ | Верхня півширина $m_u$ | Розмір блоку $s$ | Кількість GPU |
|---------|---------------------|-----------------------|------------------------|------------------|---------------|
| 1       | 20000               | 512                   | 512                    | 256              | 1, 2, 4, 6, 8 |
| 2       | 40000               | 1024                  | 1024                   | 256              | 1, 2, 4, 6, 8 |
| 3       | 56000               | 1024                  | 1024                   | 256              | 1, 2, 4, 6, 8 |
| 4       | 70000               | 2048                  | 2048                   | 512              | 1, 2, 4, 6, 8 |

Для оцінювання продуктивності використовувалися такі показники: час виконання алгоритму  $T_p$ , прискорення  $S_p$ , ефективність  $E_p$  та продуктивність у GFLOPS.

Таблиця 3.2 – Результати апробації алгоритму для матриці  $n = 56000$ ,  $m_l = m_u = 1024$ ,  $s = 256$ 

| Кількість GPU $p$ | Час виконання $T_p$ , сек. | Прискорення $S_p$ | Ефективність $E_p$ , % | Продуктивність, GFLOPS |
|-------------------|----------------------------|-------------------|------------------------|------------------------|
| 1                 | 118.4                      | 1.00              | 100.0                  | 991.5                  |
| 2                 | 61.2                       | 1.94              | 96.7                   | 1918.2                 |
| 4                 | 32.1                       | 3.69              | 92.2                   | 3656.7                 |
| 6                 | 22.4                       | 5.29              | 88.1                   | 5239.0                 |
| 8                 | 17.6                       | 6.73              | 84.1                   | 6667.9                 |

Аналіз результатів, наведених у таблиці 3.2, показує, що збільшення кількості GPU приводить до істотного зменшення часу виконання алгоритму. При переході від одного до восьми графічних прискорювачів час виконання зменшується з 118,4 сек. до 17,6 сек., що відповідає прискоренню 6,73. Ефективність використання восьми GPU становить 84,1 %, що свідчить про достатньо високий рівень масштабованості алгоритму для матриць великої розмірності зі значною шириною стрічки.

Разом з тим спостерігається поступове зниження ефективності при збільшенні кількості GPU. Це пояснюється зростанням накладних витрат на синхронізацію, передавання даних між графічними прискорювачами та

організацію блокових обмінів. Однак для розглянутої задачі обсяг локальних обчислень залишається достатньо великим, тому комунікаційні витрати не домінують над арифметичними операціями.

Таблиця 3.3 – Вплив ширини стрічки на продуктивність алгоритму при  $n = 40000$ ,  $s = 256$ ,  $p = 4$

| $m_l$ | $m_u$ | Час виконання<br>$T_p$ , с | Прискорення<br>$S_p$ | Ефективність<br>$E_p$ , % | Продуктивність,<br>GFLOPS |
|-------|-------|----------------------------|----------------------|---------------------------|---------------------------|
| 512   | 512   | 14.8                       | 3.31                 | 82.8                      | 1451.0                    |
| 1024  | 1024  | 24.6                       | 3.62                 | 90.5                      | 3409.2                    |
| 2048  | 2048  | 51.3                       | 3.78                 | 94.5                      | 6538.5                    |

Результати таблиці 3.3 показують, що зі збільшенням ширини стрічки зростає загальний час виконання, оскільки збільшується кількість арифметичних операцій. Водночас зростає і продуктивність у GFLOPS, що пояснюється підвищенням обчислювальної інтенсивності алгоритму. Для широких стрічок GPU завантажуються ефективніше, оскільки збільшується розмір активної області обчислень і зменшується відносний вплив накладних витрат.

Таблиця 3.4 – Вплив розміру блоку на час виконання алгоритму при  $n = 56000$ ,  $m_l = m_u = 1024$ ,  $p = 4$

| Розмір блоку $s$ | Час виконання<br>$T_p$ , сек | Прискорення<br>$S_p$ | Ефективність<br>$E_p$ , % | Продуктивність,<br>GFLOPS |
|------------------|------------------------------|----------------------|---------------------------|---------------------------|
| 128              | 36.8                         | 3.22                 | 80.4                      | 3189.5                    |
| 256              | 32.1                         | 3.69                 | 92.2                      | 3656.7                    |
| 512              | 34.5                         | 3.43                 | 85.8                      | 3402.3                    |

З таблиці 3.4 видно, що найкращі результати для розглянутої конфігурації отримано при розмірі блоку  $s = 256$ . При меншому значенні  $s = 128$  збільшується кількість блокових кроків, що приводить до зростання накладних витрат на організацію обчислень. При більшому значенні  $s = 512$

зменшується кількість блоків, однак погіршується гнучкість розподілу обчислень між GPU та зростає навантаження на локальну пам'ять. Отже, вибір розміру блоку є важливим параметром ефективності реалізації алгоритму.

Проведена апробація показала, що запропонований блочний алгоритм *LU*-факторизації стрічкових несиметричних матриць забезпечує коректне розв'язування тестових систем та ефективно використовує структурні властивості матриці. На відміну від алгоритмів для щільних матриць, у запропонованому підході обчислення виконуються лише в межах стрічкового профілю, що дозволяє зменшити кількість арифметичних операцій та обсяг пам'яті.

Результати експериментів підтверджують, що найбільша ефективність мульти-GPU реалізації досягається для матриць великого порядку та значної ширини стрічки. У цьому випадку частка арифметичних операцій переважає над накладними витратами на синхронізацію й обмін даними між GPU. Отримані результати узгоджуються з теоретичними оцінками часу виконання та прискорення, наведеними у попередньому підрозділі, і підтверджують доцільність використання запропонованого алгоритму для задач великої розмірності, що виникають у математичному моделюванні складних технічних і фізичних процесів.

### **3.4 Висновки до розділу**

У третьому розділі дисертаційної роботи розглянуто розв'язування методом Гауса з частковим вибором головного елемента СЛАР із стрічковими несиметричними матрицями.

Основні результати розділу:

- запропоновано блочну декомпозицію матриці, що є передумовою використання матрично-матричних операцій для переважного обсягу обчислень: при цьому застосовано циклічний розподіл рядків блоків між процесорними пристроями;

- запропоновано паралельний блочний циклічний алгоритм  $LU$ -розвинення стрічкової несиметричної матриці для комп'ютерних систем мульти-GPU архітектур;
- отримано оцінки обчислювальної складності алгоритму  $LU$ -розвинення стрічкових матриць; при цьому кількість арифметичних операцій істотно залежить від ширини стрічки матриці, що дозволяє досягти значного скорочення обчислювальних витрат у порівнянні з факторизацією щільних матриць.

Проведений аналіз алгоритму показав, що блочна організація обчислювального процесу дає змогу підвищити рівень використання апаратних ресурсів, зменшити витрати, пов'язані з доступом до даних, та забезпечити передумови для розроблення високопродуктивних паралельних реалізацій на сучасних обчислювальних архітектурах.

Запропонований алгоритм може бути адаптований для  $LL^T$ -розвинення стрічкових симетричних матриць.

## **РОЗДІЛ 4. МЕТОДИ МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ КОНСТРУКЦІЙ НА ГІБРИДНИХ КОМП'ЮТЕРНИХ СИСТЕМАХ**

Основні результати розділу висвітлені в роботах [58, 59, 61]

У задачах математичного моделювання технічних систем значна частина обчислювальних витрат пов'язана з аналізом напружено-деформованого стану складних конструкцій, що містять локальні дефекти та неоднорідності матеріалу. Особливо актуальними є задачі дослідження життєвого циклу зварних конструкцій та трубопроводів, у яких локальні пошкодження можуть призводити до втрати міцності та аварійних руйнувань.

Чисельний аналіз таких систем ґрунтується на використанні методів механіки деформівного твердого тіла та методу скінченних елементів, у результаті чого виникають системи лінійних алгебраїчних рівнянь великої розмірності з розрідженими матрицями складної структури. Ефективне розв'язування таких систем потребує використання спеціалізованих алгоритмів чисельної лінійної алгебри, адаптованих до сучасних високопродуктивних комп'ютерних систем.

У цьому розділі розглянуто застосування розроблених у дисертації методів та алгоритмів високопродуктивних обчислень для задач математичного моделювання життєвого циклу відповідальних зварних конструкцій [58, 61]. Особливу увагу приділено моделюванню дефектів локальної втрати металу трубопроводів, чисельному аналізу напружено-деформованого стану конструкцій та розробленню варіаційно-операторної моделі керованої дискретизації [59], що дозволяє оптимізувати точність і обчислювальні витрати при використанні гібридних комп'ютерних архітектур.

### **4.1 Постановка задачі дослідження життєвого циклу зварних конструкцій**

Поява розсіяної пошкодженості металу може бути обумовлена низкою факторів різної природи: інтенсивною пластичною деформацією, втомним

навантаженням, високою концентрацією дифузійного водню, радіаційним опроміненням тощо. При цьому, накопичення докритичної мікропошкоженості (ДМ) під дією регулярних експлуатаційних впливів враховується як деградація матеріалу при розробці конструктивних рішень, що дозволяє консервативно оцінювати поточний стан конструкцій. Але якщо конструкція піддавалася суттєвому нерегулярному впливу природного або техногенного характеру (зсуви, землетруси, перевантаження при пуско-налагоджувальних роботах тощо), то необхідно оцінити ступінь пошкодження, яке отримав матеріал. Наявність монтажних або ремонтних зварних швів ускладнює такий аналіз, тому що зварні з'єднання є місцями локальної хімічної і структурної неоднорідності, а також характеризуються залишковим напружено-деформованим станом (НДС), обумовленим накопиченими в процесі зварювання пластичними деформаціями, які залежать від технологічних параметрів зварювання.

Отже, розробка математичних моделей накопичення ДМ відповідальних зварних конструкцій, зокрема, для характерних випадків суттєвого зовнішнього статичного, статично змінного або втомного впливу, що спричиняє пластичну течію металу і порушення його суцільності, є актуальною і практично важливою.

Трубопровідні елементи (ТЕ) і посудини тиску (ПТ) є одними з найпоширеніших типів зварних конструкцій, які часто передбачають довгострокову експлуатацію в умовах зовнішнього силового навантаження і агресивної корозійної дії. Допустимість експлуатаційних дефектів, що формуються при цьому, визначається на основі актуальних нормативних документів і стандартів, виходячи з припущень про відомі закономірності деградації властивостей матеріалу конструкції в часі. Надмірне навантаження, яке супроводжується пластичною деформацією металу, може спричинити зародження ДМ по в'язкому механізму руйнування, зменшуючи площу ефективного поперечного перерізу конструкції, що не враховується у відповідних методиках оцінки стану трубопроводів і посудин тиску.

В дослідженнях [105, 106] було запропоновано підходи з чисельного аналізу граничного стану зварних трубопровідних елементів з дефектами локальної корозії металу в області металу шву і зони термічного впливу (ЗТВ). З їх допомогою був досліджений вплив післязварного напруженого стану на схильність конструкції до зародження і розвитку руйнування в області дефекту. Зокрема, показано, що взаємодія полів напруг і пластичних деформацій в області геометричного концентратора і місця зварювання може мати негативний вплив на статичну міцність конструкції. Але важливі, з практичної точки зору, задачі оцінки працездатності трубопровідних елементів з корозійними дефектами несучільності в області монтажного або ремонтного зварювання в умовах ультрамалоциклового та малоциклового навантаження вимагають подальшого розвитку даних методик.

Актуальною є розробка математичних моделей процесів зародження і розвитку ДМ в металі відповідальних зварних конструкцій під впливом ультрамалоциклового та малоциклового навантаження з точки зору оцінки їх стану, а також дослідження, на прикладі трубопровідних елементів з корозійними дефектами несучільності в області монтажного або ремонтного зварного шву, характерних особливостей впливу зварювання на працездатність дефектних конструкцій.

Оцінка працездатності та ресурсу безпечної експлуатації зварних трубопроводів і посудин тиску з виявленим експлуатаційним пошкодженням ґрунтується на аналізі граничного стану матеріалу з урахуванням специфіки напружено-деформованого та пошкодженого станів конструкції. При цьому у випадку простих умов експлуатаційного силового впливу (наприклад, тільки внутрішній тиск) граничний стан характеризується граничним навантаженням, яке характеризує несучу здатність конструкції. Настання граничного стану визначається зародженням та розвитком розсіяних і локальних пошкоджень матеріалу аж до макроскопічного руйнування, специфіку і природу якого необхідно враховувати при аналізі конструкцій відповідного технологічного рішення та при конкретних умовах експлуатації.

У свою чергу, аналіз стану мікро- і макропошкодження сучасних зварних конструкцій є важливим етапом визначення максимальних експлуатаційних навантажень, діагностики їхнього фактичного стану і прогнозування залишкового ресурсу. Таким чином, описання процесів, які призводять до порушення цілісності матеріалу конструкційних елементів, зародження і розвитку характерних дефектів, вимагає спільного використання методів моделювання кінетики напружено-деформованого стану залежно від масштабу і природи зовнішнього силового впливу, теорій механіки руйнування, сучасних концепцій поведінки кристалічних матеріалів під дією граничних навантажень. Зокрема, наявність зварних з'єднань робить необхідним розгляд конструкцій в області зварювання з точки зору залишкового структурного та напружено-деформованого станів, а також розвитку розсіяного пошкодження, які повинні базуватися на моделюванні процесів термопластичності в суцільних середовищах.

Відомо, що механізм втомного руйнування при ультрамалоцикловому та малоцикловому навантаженні пов'язаний з накопиченням і розвитком ДМ на кожному з циклів пластичної деформації металу. При континуальному описі цього процесу прийнято використовувати мезомасштабне наближення, яке базується на моделях в'язкого руйнування суцільного середовища. Для прогнозування даних процесів необхідно коректно описати етапи в'язкого руйнування з урахуванням неізотермічного стану матеріалу при зварюванні. Для цього, моделі руйнування мають бути поєднані з відповідними моделями кінетики температурного і напружено-деформованого стану при зварюванні і циклічному навантаженні, що дозволить отримати повну систему взаємопов'язаних фізико-механічних процесів, що мають місце в металі трубопроводу.

#### **4.2 Дефекти локальної втрати металу трубопроводів, методи їх схематизації та критерії допустимості**

Дефекти корозійної втрати металу можуть бути розділені на локальні та загальні стоншення. Локальні корозійні втрати металу характеризуються

порівняними розмірами (довжину, ширина, глибина) дефектів, а загальні стоншення мають набагато більші довжину та/або ширину в порівнянні із глибиною. Локальна корозійна втрата металу схематизується напівеліптичною геометричною аномалією (рис. 4.1), загальна корозія розглядається як частина конструкції з меншою товщиною стінки.

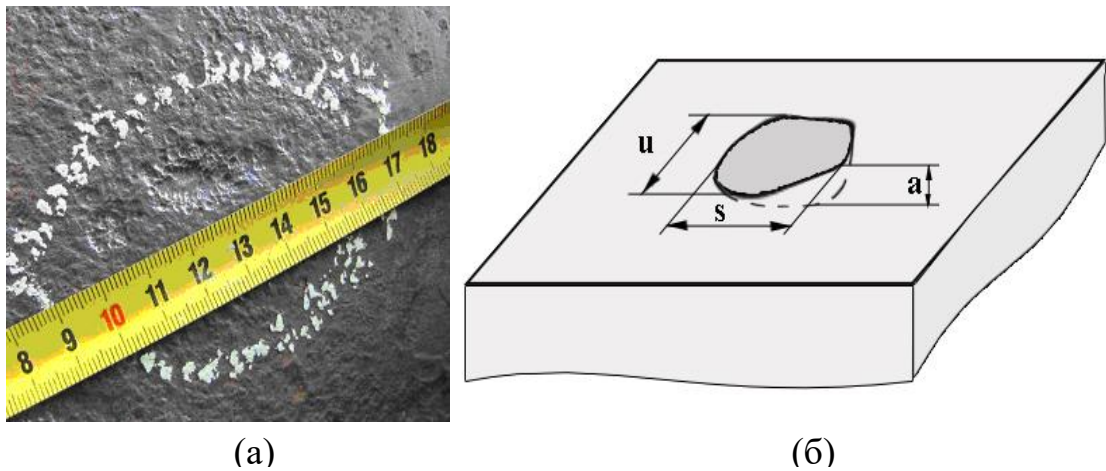


Рисунок 4.1 – Зовнішній вигляд (а) і схематизація (б) типового дефекту локальної корозійної втрати металу

Допустимість дефектів локального стоншення стінки трубопроводів може будуватися на експериментальних дослідженнях граничного стану дефектних конструкцій, що реалізовано в більшості відповідних нормативних документів, або на дослідженні допустимості напружено-деформованого стану конструкційних елементів з урахуванням їх фактичного пошкодження. Відсутність гострих концентраторів в зоні локальної втрати металу дозволяє визначати розподіл напруг з високою точністю. На основі розрахункових полів напруг і деформацій проводиться граничний аналіз стану дефектної зварної конструкції. Можна виділити три основні опції такого аналізу:

- оцінка локальної допустимості напруг і деформацій;
- оцінка допустимості граничного навантаження для фактичного стану конструкції;

– визначення ймовірності руйнування.

Слід зазначити, що перший підхід є максимально консервативним, тому що розглядає граничний стан конструкції як характеристику найбільш навантаженої точки (об'єму). З цієї точки зору інтегральні критерії другої і третьої опції оцінки є більш адекватними конкретному стану пошкодження трубопроводу та дозволяють проводити менш консервативні оцінки.

У рамках другого підходу основним механізмом руйнування зварної конструкції є в'язкий механізм, що визначає граничне експлуатаційне навантаження, яке може витримати трубопровід у фактичному стані корозійного пошкодження. Висновок про допустимість стану конструкції в цьому випадку ґрунтується на відомих частинних коефіцієнтах запасу щодо кожної компоненти експлуатаційного навантаження в рамках граничної оцінки комплексного силового впливу.

Імовірнісний аналіз руйнування проводиться на основі принципів “слабкої ланки”, і полягає в інтегруванні розподілу напруг з використанням функції Вейбулла [106], параметри якої необхідно попередньо визначити з метою одержання адекватних кількісних оцінок. Гранична допустима ймовірність руйнування визначається на основі вимог, що висуваються до конкретної конструкції залежно від її категорійності, умов експлуатації, можливих наслідків аварійної ситуації.

#### **4.3 Чисельний аналіз напружено-деформованого стану відповідальних зварних конструкцій з виявленими дефектами**

У загальному випадку, механізм в'язкого руйнування може бути представлений наступними послідовними етапами [106]:

а) зародження пор в'язкого руйнування при виготовленні конструкції, у тому числі, в зоні локального зварювального нагрівання і при розвиненому пластичному плинні металу конструкції в області фізичних і/або геометричних концентраторів при досягненні певного значення зовнішнього навантаження;

б) збільшення розмірів пор при пластичному деформуванні, взаємодія та об'єднання пор в'язкого руйнування;

в) зародження макродефекту і відповідне зниження несучої здатності як дефектної області, так і конструкції в цілому;

г) розвиток макродефекту.

Кожний із зазначених етапів має різну фізико-механічну природу, а тому їхній опис вимагає розробки відповідних взаємопов'язаних моделей.

Кінетику температурного поля при монтажному і ремонтному дуговому зварюванні може бути описано за допомогою нестационарного рівняння теплопровідності, в багатовимірному випадку у вигляді:

$$c\gamma(T) \frac{\partial T}{\partial t} = \nabla[\lambda(T) \cdot \nabla T] \quad (4.1)$$

де  $c\gamma$ ,  $\lambda$  – об'ємна теплоємність і теплопровідність металу, відповідно;  $T$  – температура конструкції у момент часу  $t$  в точці з координатами  $(r, \beta, z)$ , відповідно до схеми, яку представлено на рис. 4.2.

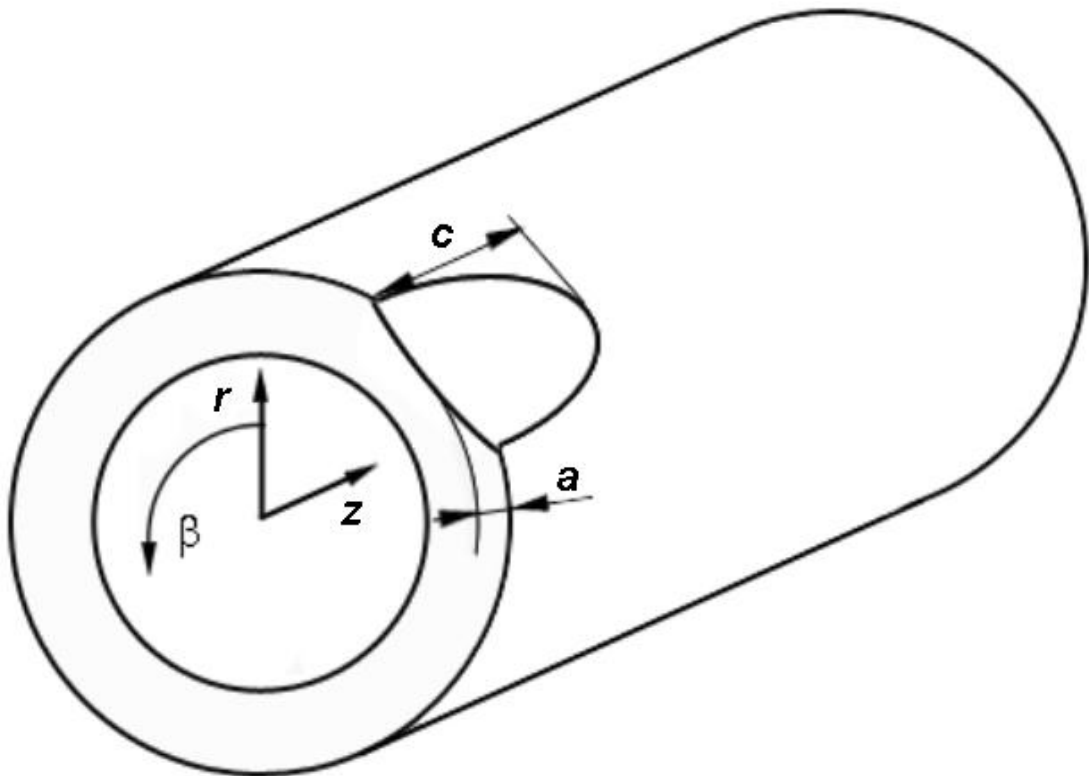


Рисунок 4.2 – Схема ділянки трубопроводу із зовнішнім напівеліптичним дефектом локального стоншення стінки

Для розв'язку рівняння (4.1) і адекватного врахування особливостей технологічного процесу зварювання необхідні граничні умови, які для даного випадку мають наступний вигляд:

$$-\lambda(T) \frac{\partial T}{\partial n} = \alpha_T (T - T_0) + \varepsilon_0 \sigma_{SB} (T^4 - T_0^4) - q, \quad (4.2)$$

де  $n$  – нормаль до поверхні конструкції;  $\alpha_T$  – коефіцієнт поверхневої тепловіддачі;  $\varepsilon_0$  – ступінь чорноти поверхні трубопроводу;  $\sigma_{SB}$  – константа Стефана-Больцмана;  $T_0$  – температура навколишнього середовища;  $q$  – поверхневий потік тепла від джерела зварювального нагріву в даній області поверхні.

Скінченно-різницевий розв'язок задачі (4.1)-(4.2) дозволяє оцінювати розподіл температур в процесі зварювання з урахуванням конкретних технологічних параметрів. Розрахункова кінетика температурного стану конструкції при зварюванні лежить в основі чисельної оцінки напружено-деформованого і пошкодженого станів трубопровідного елемента. Компоненти тензора деформацій  $\varepsilon_{ij}$  ( $i, j = r, \beta, z$ ) в даному випадку представляються у вигляді суперпозиції наступних компонент:

$$d\varepsilon_{ij} = d\varepsilon_{ij}^e + d\varepsilon_{ij}^p + (d\varepsilon_{ij}^T + df/3)\delta_{ij} \quad (4.3)$$

де  $d\varepsilon_{ij}^e$ ,  $d\varepsilon_{ij}^p$ ,  $d\varepsilon_{ij}^T$  – компоненти приросту тензора деформацій, які визначаються пружним механізмом деформації, пластичною течією і температурним розширенням, відповідно;  $f$  – об'ємна концентрація рівномірно розподілених пор в'язкого руйнування.

В (4.3) фігурує об'ємна концентрація рівномірно розподілених пор  $f$ , наявність якої відрізняє розроблені моделі від класичних методів аналізу термодиформованого стану суцільного середовища. Наявність ДМ не тільки має адитивну складову в тензорі деформацій, але і змінює поверхню текучості матеріалу  $\Phi$ , для математичного опису якої знайшли широке

застосування підходи Гурсона, Твергаарда і Нідлмана (так звана ГТН модель):

$$\Phi = (\sigma_i / \sigma_T)^2 - (q_3 f^*)^2 + 2q_1 f^* \cosh\left(q_2 \frac{3\sigma_m}{2\sigma_T}\right) - 1, \quad (4.4)$$

де  $q_1 = 1,5$ ,  $q_2 = 1$ ,  $q_3 = 1,5$  – константи,  $f^*$  – еквівалентна концентрація пор,  $\sigma_T$  – межа текучості матеріалу,  $\sigma_m = (\sigma_{rr} + \sigma_{\beta\beta} + \sigma_{zz})/3$  – середнє значення нормальних компонент тензора напруг,  $\sigma_I = (0,5\sigma_{ij}\sigma_{ij})^{0,5}$  – інтенсивність напруг. Еквівалентна концентрація пор  $f^*$  в (4.4), що враховує взаємодію між окремими несучільностями, оцінюється на основі наступних співвідношень:

$$f^* = \begin{cases} f, & \text{якщо } f \leq f_c \\ f_c + \frac{f_u^* - f_c}{f_F - f_c} (f - f_c), & \text{якщо } f > f_c \end{cases}$$

де  $f_c$  – критична концентрація несучільностей, до якої окремі пори не взаємодіють, прийнято вважати  $f_c = 0,15$ ;  $f_F$  – концентрація пор, при якій відбувається руйнування скінченного елемента,  $f_u^* = 1/q_1$ .

Як видно з (4.4), граничний перехід  $f^* > 0$  переводить ГТН модель в умову текучості Мізеса. Також для коректного опису граничного стану даних конструкцій необхідно враховувати деформаційне зміцнення металу в умовах статичного і циклічного експлуатаційного навантаження, а саме зміна його межі текучості згідно наступного співвідношення:

$$\sigma_T = \sigma_T^0 \left(1 + c_1 \ln(\dot{\epsilon}_p / \dot{\epsilon}_0) + c_2 \ln^2(\dot{\epsilon}_p / \dot{\epsilon}_0)\right) \left(1 + (\dot{\epsilon}_p / \dot{\epsilon}_0)^m\right),$$

де  $c_1 = 2,149 \times 10^{-3}$ ;  $c_2 = 9,112 \times 10^{-2}$ ;  $\epsilon_0 = 1,540 \times 10^{-4}$ ,  $m = 0,14$  – константи; крапкою над змінною позначено диференціювання за часом.

Для оцінки зародження пор в'язкого руйнування при пластичній течії матеріалу в неізотермічному випадку використовувався модифікований критерій Джонсона-Кука, згідно якого в деякому об'ємі металу з'являється початкова пористість з концентрацією  $f_0$  при виконанні наступної умови:

$$\chi_k = \int \frac{d\epsilon_i^p}{\epsilon_c(T)} > 1, \quad (4.5)$$

де  $d\varepsilon_i^p = \frac{\sqrt{2}}{3} \sqrt{d\varepsilon_{ij}^p \cdot d\varepsilon_{ij}^p}$ ,  $\varepsilon_c(T)$  – критична величина пластичних

деформацій. Критичну пластичну деформацію  $\varepsilon_{c,v}$  (4.5) можна обчислити за наступним співвідношенням:

$$\varepsilon_c(T) = (d_1 + d_2 \exp(-d_3 \sigma_m / \sigma_i)) \exp\left(\left(\frac{\sigma_T - \sigma_T(T)}{B_f}\right)^\xi\right),$$

де  $d_1, d_2, d_3, B_f, \xi$  – константи.

Подальше зростання концентрації пор в'язкого руйнування в процесі пластичної деформації металу, зокрема, при експлуатаційному статичному або циклічному навантаженні, підкоряється закону Райса-Трейсі:

$$df = k_{ms} f_0 K_1 \exp(K_2 \sigma_m / \sigma_i) d\varepsilon_i^p, \quad (4.6)$$

де  $k_{ms}$  – коефіцієнт, що враховує пластичну мікродеформацію металу в припущенні лінійної залежності пластичної деформації металу в мікро- і макромасштабі;  $K_1 = 0,28$ ,  $K_2 = 1,5$  – константи.

Математичний розгляд об'єднаної задачі кінетики температурного поля, розвитку напруг і деформацій та формування мікропор базується на скінченно-елементному описі з використанням восьмивузлових скінченних елементах (СЕ, див. рис. 4.3). У рамках об'єму, обмеженого розглянутим елементом, розподіл температур, напруг і деформацій приймається як однорідний.

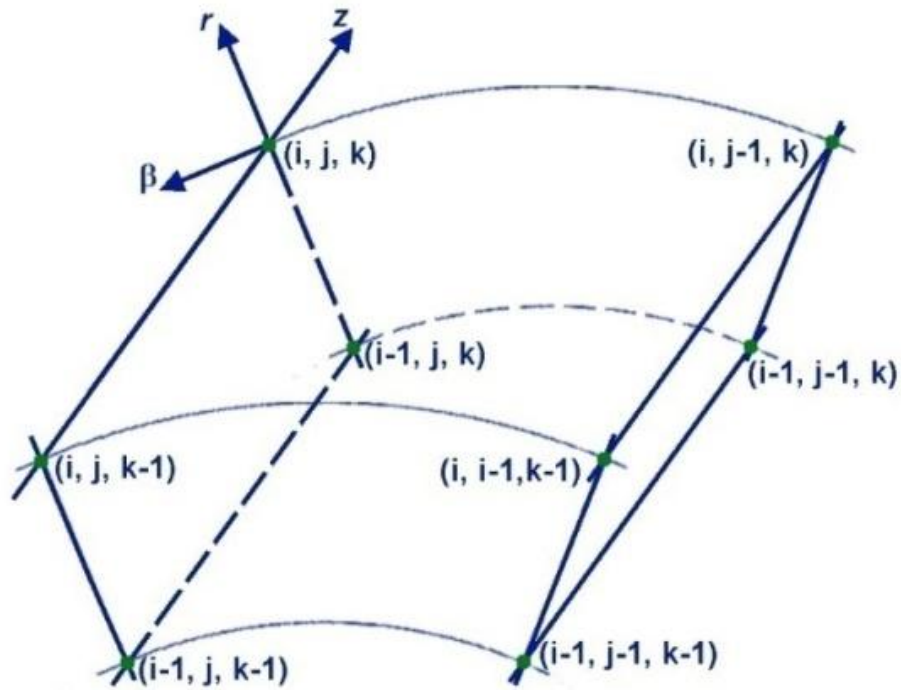


Рисунок 4.3 – Схема скінченного елемента в системі координат  $r, \beta, z$ .

Скінченно-елементне розв'язування крайової задачі нестационарної термопластичності проводилося шляхом дослідження спільного розвитку пружно-пластичних деформацій, докритичного та критичного руйнування за в'язким механізмом. Реалізація такого чисельного дослідження пов'язана з двома нелінійностями за фізичними процесами: пластичній деформації і руйнуванню. Для формального вирішення цих нелінійних задач реалізовано відповідні ітераційні процеси, що дозволяють знаходити стан СЕ, що задовольняє рівнянням рівноваги і умові (4.4). Так, для вирішення нелінійності по пластичній деформації було використано запропонований В.І. Махненко [36] підхід, а саме розгляд функції стану матеріалу  $\Psi$ , що задовольняє наступним умовам на поверхні текучості:

$$\Psi = \frac{1}{2G}, \quad \text{якщо } \sigma_i < \sigma_s, \\ \Psi > \frac{1}{2G}, \quad \text{якщо } \sigma_i = \sigma_s, \quad (4.7)$$

де  $\sigma_s = \sigma_T \sqrt{1 + (q_3 f^*)^2 - 2q_1 f^* \cosh(1,5q_2 \sigma_m / \sigma_T)}$ ,  $G = 0,5E/(1+\nu)$ ,  $E$  – модуль Юнга,  $\nu$  – коефіцієнт Пуассона; а стан  $\sigma_i > \sigma_s$  є недопустимим.

Головна складність при моделюванні циклічного навантаження полягає в тому, що невеликі зміни в металі на одному циклі навантаження, а саме накопичення і зростання ДМ, викликають зміну поверхні текучості згідно (4.4), що спричиняє зміну петлі пластичної деформації. Але при цьому на кожному етапі навантаження необхідно визначити рівноважний стан пошкоженості і відповідний йому розподіл напруг і деформацій. Для цього у припущенні, що стаціонарний стан характеризується нехтуючи малою швидкістю зростання ДМ за (4.6), пропонується проводити такий ітераційний процес по функції  $\Psi_K = f_0 K_1 \exp(K_2 \sigma_m / \sigma_i) d\varepsilon_i^p$  :

$$F = \begin{cases} F + dF, & \text{якщо } \Psi_K \leq \Psi_K^0 \approx 10^{-5}, \\ F, & \text{якщо } \Psi_K > \Psi_K^0, \end{cases} \quad (4.8)$$

де  $F$  – система зовнішніх силових навантажень, що діють на конструкцію;  $dF$  – приріст силових навантажень в процесі чисельного дослідження.

Таким чином, розв'язування задач (4.7), (4.8) на кожному етапі дослідження дозволяє з контрольованою мірою точності визначати ступінь пошкоженості матеріалу зварної конструкції з урахуванням її геометричних особливостей. Як критерій зародження макроскопічного руйнування використовуються умови крихко-в'язкого руйнування, а саме виконання однієї з трьох умов:

$$\left( \Psi - \frac{1}{2G} \right)_{KP} \geq \frac{\varepsilon_f - \varepsilon_p^*}{1,5\sigma_i} \approx \frac{\varepsilon_f - \varepsilon_p^*}{1,5\sigma_s(\varepsilon_p, T)}$$

або

$$f^* \rightarrow f_d^* = 2(q_1/q_3) \cosh(1,5q_2 \sigma_m / \sigma_T) \quad (4.9)$$

або

$$\frac{3\sigma_1}{3-2f} > S_K.$$

де  $S_K$  – величина напруги мікросколу,  $\varepsilon_f$  – гранична деформаційна здатність матеріалу, індекс «\*» відносить змінну до попереднього кроку дослідження.

У випадку, якщо для деякого СЕ виконується одна з умов (4.9), вважається, що даний СЕ втрачає свою несучу здатність і на його місці сформувався макроскопічна несучільність. Подальше навантаження конструкції і розвиток макроруйнування, у результаті призводять до лавиноподібної втрати несучої здатності матеріалу в межах ітераційного процесу (4.8), що можна інтерпретувати, як спонтанне руйнування конструкції. Залежно від точності наявних даних і від виробничої необхідності, граничним станом відповідальної конструкції можна вважати або момент зародження першої макронесучільності, або її (конструкції) спонтанне руйнування.

Залежність деформацій від напруг визначається законом Гука та асоційованим законом пластичного плину, виходячи з наступного співвідношення:

$$\begin{aligned} \Delta\varepsilon_{ij} = & \Psi(\sigma_{ij} - \delta_{ij}\sigma_m) - \frac{1}{2G}(\sigma_{ij} - \delta_{ij}\sigma_m)^* + \\ & + \delta_{ij}(K\sigma_m + \Delta\varepsilon_T + \Delta f/3) + (K\sigma_m)^*, \end{aligned}$$

де  $\delta_{ij}$  – символ Кронекера,  $K = \frac{1-2\nu}{E}$  – модуль об'ємного стиснення.

Пластичні деформації визначаються за співвідношенням:

$$\Delta\varepsilon_{ij} = \left( \Psi - \frac{1}{2G} \right) (\sigma_{ij} - \delta_{ij}\sigma_m).$$

При цьому на кожному кроці ітерації по  $\Psi$  напруги  $\sigma_{ij}$  обчислюються в такий спосіб (за повторюваними індексами відбувається підсумовування):

$$\sigma_{ij} = \frac{1}{\Psi} \left( \Delta\varepsilon_{ij} + \delta_{ij} \frac{\Psi - K}{K} \Delta\varepsilon \right) + J_{ij},$$

де

$$\Delta \varepsilon = \frac{\Delta \varepsilon_{ij}}{3}, \quad J_{ij} = \frac{1}{\Psi} \left( b_{ij} - \delta_{ij} b + \delta_{ij} \left( K \sigma^* - \frac{\Delta \varepsilon_T + \Delta f / 3}{K} \right) \right), \quad b = \frac{b_{ii}}{3}.$$

Співвідношення між компонентами тензора деформацій  $\Delta \varepsilon_{ij}$  і вектора приросту переміщень  $\Delta U_i$  задається наступним виразом (комою позначено диференціювання в межах СЕ):

$$\Delta \varepsilon_{ij} = \frac{\Delta U_{i,j} + \Delta U_{j,i}}{2}.$$

Компоненти тензора напруг задовольняють рівнянням статички для внутрішніх СЕ та граничним умовам – для поверхневих. У свою чергу, компоненти вектора  $\Delta U_i$  задовольняють відповідним умовам на границі.

На кожному кроці дослідження та ітерацій по  $\Psi$  необхідно розв'язувати систему рівнянь в змінних вектора приросту переміщень  $\Delta U_i$  у вузлах СЕ, яка визначається з умов мінімуму наступного функціонала (використовуючи варіаційний принцип Лагранжа):

$$E_I = -\frac{1}{2} \sum_V (\sigma_{ij} + J_{ij}) \Delta \varepsilon_{ij} V_{m,n,r} + \sum_{S_p} P_i \Delta U_i \Delta S_P^{m,n,r}.$$

де  $\sum_V$  – оператор суми по внутрішнім СЕ,  $\sum_{S_p}$  – оператор суми по поверхневим СЕ, на яких задані компоненти силового вектора  $P_i$ . Тобто система нелінійних рівнянь (СНР)

$$\frac{\partial E_I}{\partial \Delta U_{m,n,r}} = 0, \quad \frac{\partial E_I}{\partial \Delta V_{m,n,r}} = 0, \quad \frac{\partial E_I}{\partial \Delta W_{m,n,r}} = 0. \quad (4.10)$$

дозволяє одержати розв'язок в компонентах вектора приросту переміщень на кожному кроці дослідження та ітерацій по  $\Psi$  для конкретного СЕ.

Для розв'язування СНР (4.10) використовуються різноманітні ітераційні методи, загальною рисою яких є те що на кожній ітерації розв'язується відповідна система лінійних алгебраїчних рівнянь, а розв'язки цієї СЛАР використовуються для формування матриці СЛАР наступної ітерації. Якщо використати методологію запропоновану академіком В.І. Махненком, то на кожній ітерації розв'язується СЛАР із стрічковою несиметричною матрицею. Причому розв'язування таких СЛАР потребує значного обчислювального ресурсу і тому визначає ефективність комп'ютерного моделювання.

Отже, для комп'ютерного моделювання об'єктів, що тут розглядаються, доцільно використати паралельні комп'ютери гібридної архітектури. В такому разі розв'язування систем лінійних рівнянь реалізується на multi-GPU архітектурі, використовуючи запропонований у розділі 3 блочний алгоритм *LU*-факторизації стрічкових матриць.

#### 4.4 Апробація алгоритму

На рис. 4.4 наведено загальну схему розв'язування розрахункової задачі.

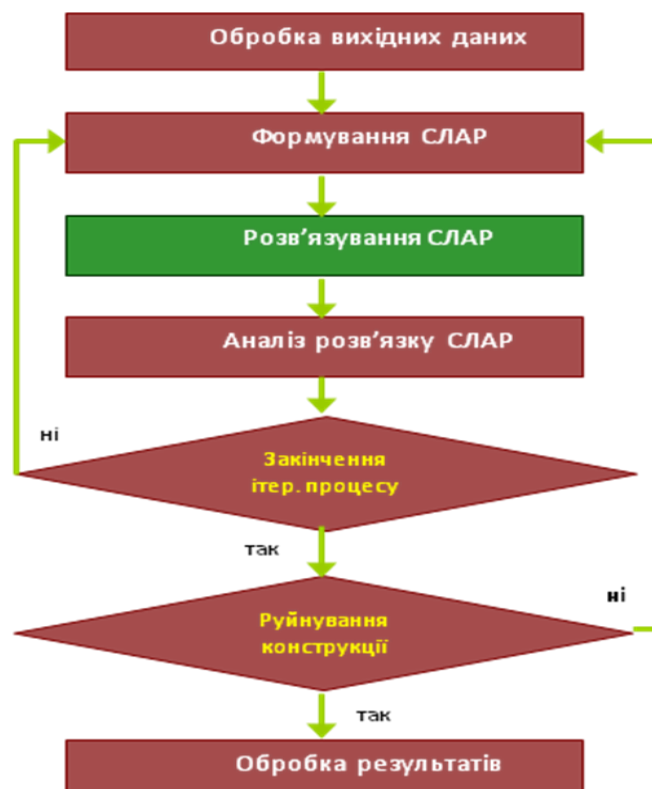


Рисунок 4.4 – Схема розв'язування розрахункової задачі

Відповідно до цієї схеми, на кожному етапі обчислювального процесу після обробки вихідних даних виконується формування системи лінійних алгебраїчних рівнянь, яка описує поточний стан конструкції. Далі здійснюється розв'язування сформованої СЛАР, аналіз отриманого розв'язку та перевірка умов завершення ітераційного процесу. Якщо умови завершення не виконані, виконується наступний цикл формування і розв'язування системи. У разі досягнення критерію руйнування конструкції здійснюється обробка та аналіз кінцевих результатів.

Таким чином, центральним етапом наведеної схеми є багаторазове розв'язування СЛАР, що виникають у процесі чисельного моделювання. Саме тому ефективність усього розрахункового процесу значною мірою визначається швидкістю та надійністю алгоритму розв'язування цих систем. Для цього в роботі використано блочний алгоритм  $LU$ -факторизації стрічкових матриць, розроблений у розділі 3.

Апробація алгоритму проводилася в межах наведеної розрахункової схеми на тестових і прикладних задачах математичного моделювання. Під час чисельних експериментів досліджувалися коректність формування СЛАР, точність отриманих розв'язків, а також обчислювальна ефективність запропонованого алгоритму при багаторазовому використанні в ітераційному процесі. Отримані результати підтвердили, що застосування розробленого алгоритму дозволяє ефективно розв'язувати системи рівнянь, які виникають на кожному кроці аналізу, та забезпечує зменшення загального часу розрахунку.

Чисельні експерименти проводилися у спеціалізованому високопродуктивному обчислювальному середовищі, що забезпечує необхідні ресурси для паралельної обробки великих стрічкових матриць і дозволяє досліджувати ефективність алгоритмів як на багатоядерних CPU, так і на графічних прискорювачах GPU. Апаратна платформа базується на вузлах, оснащених 442-ядерними процесорами архітектури AMD EPYC, які характеризуються високою пропускнуою здатністю пам'яті та здатністю

масштабовано виконувати масивні паралельні навантаження. Кожен вузол містить 256 ГБ оперативної пам'яті, що дозволяє обробляти матриці великої розмірності й виконувати паралельні блокові операції без потреби в зверненні до дискових ресурсів. Великий обсяг оперативної пам'яті є важливою умовою для коректного вимірювання продуктивності  $LU$ -факторизації, оскільки будь-яке використання дискової підкачки спотворювало б результати експериментів.

У складі кластерної інфраструктури використовується високошвидкісний мережевий інтерконект Infiniband зі швидкістю 200 Gbit/s, який забезпечує низьку латентність і високу пропускну здатність міжвузлових комунікацій. Це особливо важливо при масштабуванні  $LU$ -факторизації на багатовузлові системи з використанням MPI або гібридних схем MPI+CUDA, де ефективність алгоритму значною мірою залежить від швидкості обміну панелями та блоками стрічкової матриці. Використання Infiniband дозволяє мінімізувати комунікаційні затрати та наблизити загальну продуктивність до теоретично можливої.

Для прискорення обчислень на графічних процесорах застосовувався професійний GPU Nvidia RTX A6000, що належить до архітектури Ampere та містить понад 11 000 потокових ядер CUDA. Графічний прискорювач оснащено 48 ГБ високошвидкісної відеопам'яті, що забезпечує можливість розміщення великих стрічкових матриць у пам'яті GPU без фрагментації або необхідності порційного перенесення даних. Завдяки цьому вдалося виконувати блокові операції GEMM та TRSM — ключові етапи блочної  $LU$ -факторизації — переважно на GPU, мінімізувавши комунікації між CPU і GPU. Потужність обчислювальних блоків Nvidia A6000 забезпечила можливість одночасного виконання сотень незалежних операцій оновлення блоків стрічкової матриці, що є критично важливим для досягнення високого рівня GFLOPs.

Поєднання потужного багатоядерного процесора AMD EPYC, великого обсягу оперативної пам'яті, високошвидкісної мережевої інфраструктури та

графічного прискорювача Nvidia RTX A6000 створило оптимальні умови для комплексного дослідження паралельних реалізацій блочної  $LU$ -факторизації стрічкових матриць. Таке експериментальне середовище дозволило перевірити ефективність OpenMP-реалізації при різних конфігураціях потоків, дослідити масштабованість GPU-орієнтованих алгоритмів, а також оцінити потенціал гібридних схем CPU+GPU та можливості розширення обчислень на багатовузлові системи.

Запропонований паралельний алгоритм було апробовано на розв'язуванні кількох СЛАР із стрічковими несиметричними матрицями, які виникають при математичному моделюванні різних станів зварних конструкцій. Досліджувалися впливи архітектури комп'ютера та параметрів алгоритму (кількість потоків, розміру блоку тощо), параметри СЛАР (структура, порядок, ширина стрічки тощо) на час розв'язування. Для тестування було використано СЛАР із стрічковими матрицями, параметри яких представлено в табл. 4.1.

Таблиця 4.1 – Набір тестових матриць

| № п/п | Назва    | Порядок $N$ | Кількість піддіагоналей $m_l$ | Кількість наддіагоналей $m_u$ |
|-------|----------|-------------|-------------------------------|-------------------------------|
| 1     | A-126-20 | 126 000     | 2 001                         | 2 001                         |
| 2     | A-126-09 | 126 000     | 902                           | 902                           |
| 3     | A-055-10 | 55 650      | 1 052                         | 1 052                         |
| 4     | A-052-10 | 52 500      | 1 052                         | 1 052                         |
| 5     | A-137-44 | 137 826     | 4 448                         | 4 448                         |
| 6     | A-798-05 | 798 624     | 565                           | 565                           |

В табл. 4.2 показано часи розв'язання задач з тестовими матрицями на описаній платформі з використанням різної кількості потоків в OpenMP реалізації та реалізації на CUDA. Для обчислень використовувався розмір блоку 128.

Таблиця 4.2 – Часи розв’язання задач

| Матриця<br>СЛАР | Час розв’язування (сек.) |            |           |           |       |       |
|-----------------|--------------------------|------------|-----------|-----------|-------|-------|
|                 | 1 потік                  | 16 потоків | 32 потоки | 64 потоки | 1 GPU | 4 GPU |
| A-126-20        | ≈ 2 240                  | 240,23     | 52,32     | 17,79     | 15,23 | 5.12  |
| A-126-09        | ≈ 660                    | 250,35     | 10,13     | 2,5       | 13,46 | 4,5   |
| A-055-10        | ≈ 170                    | 35,02      | 4,08      | 8,45      | 3,21  | 0.87  |
| A-052-10        | ≈ 210                    | 65,28      | 12,28     | 9,34      | 3,74  | 1.21  |
| A-137-44        | ≈ 7 080                  | 420,23     | 185,70    | 64.23     | 45,34 | 12.83 |
| A-798-05        | ≈ 800                    | 186,45     | 35,00     | 23,75     | 9,12  | 1.4   |

Наведені результати як і результати інших експериментів засвідчили, що використання паралельних обчислень дозволяє суттєво скоротити час розв’язування задач – від 15 до 60 раз.

#### **4.5 Варіаційно-операторна модель керованої дискретизації для гібридних обчислень.**

У контексті сучасного обчислювального моделювання гібридних систем, що поєднують континуальні фізичні процеси з дискретними подіями керування, постає фундаментальна проблема узгодження математичної точності з архітектурною ефективністю. При переході до гібридних CPU–GPU архітектур це завдання набуває принципово нового змісту: просторово-часова дискретизація операторів має не лише забезпечувати апроксимаційну збіжність, а й відповідати енергетичним інваріантам системи та топології обчислювального графа.

Класичні схеми дискретизації, зокрема методи скінченних різниць і скінченних елементів фіксованого порядку, не враховують локальної варіації

спектральних властивостей оператора  $L$ , який описує просторово-часову динаміку системи. Унаслідок цього їхня апроксимаційна точність не адаптується до локальної структури поля  $u(x,t)$ , а ефективність реалізації визначається апаратною пропускнуою здатністю пам'яті лише на етапі виконання, без участі самої дискретизаційної схеми.

**Зауваження 1.** Далі під полем  $u(x,t)$  розуміємо функцію стану, що описує розподіл фізичної або модельної величини у просторі та часі; термін *структура поля* використовується для позначення локальної просторово-часової організації або спектральних властивостей  $u(x,t)$ .

У результаті така фіксована дискретизація створює дисбаланс між локальною точністю апроксимації та ефективністю обчислень, оскільки єдина сітка не здатна адаптуватися до областей із різкими змінами просторово-часових масштабів. Цей дисбаланс проявляється як порушення енергетичної узгодженості між континуальною моделлю та її чисельною апроксимацією, що призводить до появи штучних дисипативних або генеративних компонент у спектрі розв'язку й, відповідно, до спотворення фізично коректної динаміки системи.

Для усунення цих обмежень у даній роботі запропоновано метод керованої дискретизації, який розглядає побудову апроксимаційного оператора як задачу оптимізації в параметричному просторі:

$$\theta = (p, r, \varphi), \quad (4.11)$$

де  $p$  – порядок апроксимації;  $r$  – радіус локального шаблону апроксимації;  $\varphi$  – тип фільтраційної схеми, що контролює спектральну характеристику (селективність) оператора  $L$ .

На відміну від класичних схем дискретизації, параметри  $\theta$  у (4.11) не задаються фіксовано, а визначаються з урахуванням локальних характеристик функції стану  $u(x,t)$ , зокрема просторової гладкості, чутливості до збурень і часової жорсткості. Така адаптація дозволяє

реалізувати енергетично узгоджені дискретизації, у яких збережено головні інваріанти системи при мінімізації витрат ресурсів для обчислення на гібридних архітектурах [108].

У межах варіаційно-операторного підходу дискретизація трактується як оптимізаційна процедура наближення оператора  $L$  у просторі дволінійних форм, узгоджених з енергетичними характеристиками системи. У цьому контексті побудова параметризованого дискретного оператора  $L_h(\theta)$  спрямована не лише на мінімізацію норми відхилення  $\|L - L_h(\theta)\|$ , але й на збереження коерцитивності (позитивної визначеності енергетичної форми), дисипативності та коректної енергетичної балансової структури. Відповідно, керована дискретизація розглядається як варіаційний механізм узгодження аналітичних, чисельних та апаратних інваріантів, який формує теоретичну основу для побудови енергетично збалансованих сплітів та адаптивного планування обчислень у гібридних CPU–GPU середовищах [109].

#### 4.6 Варіаційно-операторний підхід до керованої дискретизації.

Дискретний оператор у межах варіаційно-операторного підходу визначається як варіаційне наближення безперервного оператора в енергетичній нормі, що забезпечує узгодженість між аналітичною моделлю та її чисельним представленням.

Нехай оператор  $L:V \rightarrow V'$  визначено в гільбертовому просторі  $V$  з відповідним енергетичним скалярним добутком  $\langle \cdot, \cdot \rangle_E$ , а його дискретна апроксимація задається у вигляді параметризованого відображення:

$$L_h(\theta):V_h(\theta) \rightarrow V'_h(\theta), \quad (4.12)$$

де  $V_h(\theta)$  – параметризований підпростір апроксимації, породжений системою локальних базисних функцій  $\{\phi_j^{(\theta)}\}_{j=1}^{N_h^{(\theta)}}$ , де кожна функція  $\phi_j^{(\theta)}$

асоційована з вузлом або коміркою обчислювальної сітки, а  $N_h^{(\theta)}$  – розмірність апроксимаційного підпростору (кількість базисних функцій) для вибраної конфігурації  $\theta$  (4.11).

Враховуючи (4.12), дискретну форму оператора можна визначити матрицею:

$$[L_h(\theta)]_{ij} = \langle L\phi_j^{(\theta)}, \phi_j^{(\theta)} \rangle_E, \quad (4.13)$$

що дозволяє зберегти енергетичну когерентність між неперервною та дискретною моделями.

У (4.13) оператор  $L_h(\theta)$  наближає  $L$  не лише у метриці норми  $\|L - L_h(\theta)\|$ , але й у просторі дволінійних форм, де збережено позитивну визначеність і коерцитивність енергетичного функціоналу.

Для гібридних архітектур CPU–GPU важливо, щоб дискретний оператор мав структуру, сумісну з асинхронним розпаралелюванням. Тому  $L_h(\theta)$  подається як композиція енергетично узгоджених спліт-операторів:

$$L_h(\theta) = \sum_{k=1}^{m(\theta)} L_{h,k}, \quad (4.14)$$

де  $k$  – індекс, що нумерує локальні енергетично узгоджені спліти, тобто фрагменти оператора, що відповідають окремим фізичним підпроцесам або компонентам обчислювального графа;  $m(\theta)$  – кількість сплітів, що визначається структурою параметрів  $\theta$  (4.11).

Кожен спліт-оператор  $L_{h,k}$  є енергетично самодостатнім, тобто  $\langle L_{h,k}u_h, u_h \rangle_E \geq 0$ , де  $u_h$  – дискретне наближення функції стану,  $u_h \in V_h(\theta)$ ,  $u_h = \sum_j u_j \phi_j^{(\theta)}$ . Така декомпозиція дає змогу виконувати асинхронні оновлення вузлів і мінімізує комунікаційні витрати між пам'яттю CPU та GPU та

гарантує відсутність неузгоджених дисипативних ефектів у процесі розпаралелювання.

Оптимальні параметри  $\theta$  задаються як аргумент мінімуму локальної похибки за умови обмеженої обчислювальної вартості:

$$\theta^* = \arg \min_{\theta} E_{\text{loc}}^{(i)}(\theta), \quad (4.15)$$

за умови

$$T_{\text{GPU}}(\theta) \leq T_{\text{max}}, \quad (4.16)$$

де  $E_{\text{loc}}^{(i)}(\theta)$  – локальна оцінка похибки наближення;  $T_{\text{GPU}}(\theta)$  – прогнозований час виконання обчислень.

Оптимізаційна задача (4.15)-(4.16) формує компроміс між точністю апроксимації та архітектурною ефективністю, не потребуючи глобального задання фіксованої схеми дискретизації для всієї області визначення задачі  $\Omega \subset \square^d$ . У підсумку формується локально адаптивна схема дискретизації, у межах якої кожна підобласть  $\Omega_i \subset \Omega$  має власну конфігурацію  $\theta_i$  та оптимальну кількість обчислювальних сплітів  $m_i = m(\theta_i)$ , що забезпечують мінімальну енергетичну похибку наближення:

$$E_{\text{loc}}^{(i)}(\theta) = \|Lu - L_n(\theta_i)u\|_{\Omega_i}^2. \quad (4.17)$$

Модель (4.11)-(4.17) створює основу для подальшого введення енергетично узгоджених сплітів і двохарової оптимізації точності та часу.

#### 4.7 Енергетично узгоджена декомпозиція та стабільність

Після введення параметризованого оператора  $L_n(\theta)$  (4.14), ключовим етапом є його декомпозиція на спліт-оператори  $\{L_{n,k}\}$ , які забезпечують коерцитивність і стабільність при розпаралелюванні на CPU–GPU архітектурах. Такий підхід дозволяє реалізувати спліт-схеми інтегрування (типу *Strang splitting scheme*, *Alternating Direction Implicit*)[110, 111],

зберігаючи порядок апроксимації та контроль енергетичних інваріантів системи.

Рівняння (4.14) вважається енергетично узгодженим, якщо сукупна дія часткових операторів зберігає коерцитивність енергетичної форми, тобто:

$$\langle L_h u_h, u_h \rangle_E = \sum_{k=1}^{m(\theta)} \langle L_{h,k} u_h, u_h \rangle_E \geq c \|u_h\|_E^2, c > 0, \forall u_h \in V_h(\theta). \quad (4.18)$$

У чисельному інтегруванні еволюційних рівнянь розглядається початкова задача

$$\frac{\partial u_h}{\partial t} = L_h(\theta) u_h, \quad (4.19)$$

для якої часовий крок  $\tau > 0$  задає дискретизацію часової осі  $t_n = n\tau$ . Відповідно, оператор еволюції за один крок часу формально можна визначити як експоненту  $e^{\tau L_h}$ .

**Приклад 1.** Розглянемо побудову  $L_h(\theta)$  для простого енергетично узгодженого оператора типу Лапласа

$$Lu = -\nabla \cdot (a(x) \nabla u), a(x) > 0, x \in \Omega \subset \mathbb{R}^2.$$

На рівномірній сітці з кроком  $h$  і параметрами (4.11) його дискретизація приймає вигляд:

$$\begin{aligned} (L_h(\theta) u_h)_{i,j} = & -\frac{a_{i,j}}{h^2} (\varphi(\theta_x) (u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) + \\ & + \varphi(\theta_y) (u_{i,j+1} - 2u_{i,j} + u_{i,j-1})), \end{aligned}$$

де  $\varphi(\theta)$  задає тип локальної фільтрації (наприклад, центральна схема для  $\varphi = 1$ , або згладжена – для  $\varphi < 1$ ).

Нехай  $C$  – стала апроксимації, тоді енергетичний добуток визначається як:

$$\langle L_h(\theta) u_h, u_h \rangle_E = \sum_{i,j} a_{i,j} \left[ \frac{(u_{i+1,j} - u_{i,j})^2}{h^2} + \frac{(u_{i,j+1} - u_{i,j})^2}{h^2} \right],$$

а це гарантує, що  $\langle L_h(\theta) u_h, u_h \rangle_E \geq 0, \forall u_h$ , тобто коерцитивність і енергетична узгодженість дискретної форми. Збільшення порядку  $p(\theta)$

автоматично розширює радіус  $r(\theta)$  і зменшує похибку  $\|u - u_h(\theta)\| \leq Ch^{p(\theta)}$  та зберігає структуру.

Умова (4.18) означає, що дискретна система не створює штучних джерел або втрат енергії під час роздільного обчислення компонент. Для гібридних систем CPU–GPU такий розклад створює природний механізм асинхронного оновлення: кожен спліт може бути виконаний незалежно за умови, що енергетична форма залишається позитивно напіввизначеною.

Точність симетричних сплітів визначається не самим фактом декомпозиції, а ступенем їхньої некомутативності, що контролюється верхніми оцінками норм комутаторів. Для будь-яких двох енергетично узгоджених компонент  $L_{h,i}$  та  $L_{h,j}$  розглядаємо комутатор Лі [110, 112]:

$$[L_{h,i}, L_{h,j}] := L_{h,i}L_{h,j} - L_{h,j}L_{h,i}. \quad (4.20)$$

Оскільки у загальному випадку  $[L_{h,i}, L_{h,j}] \neq 0$ , спліт-апроксимація наближує  $e^{\tau L_h}$  із похибкою, пропорційною нормі цього комутатора.

**Лема 1.** *Порядок енергетично узгодженого спліту.* Нехай (4.14) – енергетично узгоджений розклад дискретного оператора, побудований згідно з умовами (4.1)–(4.7), де кожен компонент  $L_{h,k}$  є коерцитивним у нормі енергії  $\|\cdot\|_E$  і комутатори (4.20) задовольняють оцінку:

$$\|[L_{h,i}, [L_{h,i}, L_{h,j}]]\|_{E \rightarrow E} = O(h^{p+1}), \quad (4.21)$$

та симетрична схема Странга

$$S_\tau = e^{\frac{\tau}{2}L_{h,1}} e^{\tau L_{h,2}} e^{\frac{\tau}{2}L_{h,1}}, \quad (4.22)$$

з кроком часової дискретизації  $\tau > 0$  є апроксимацією операторної експоненти  $e^{\tau L_h}$  другого порядку за часом у нормі  $\|\cdot\|_{E \rightarrow E}$ , тобто

$$\|e^{t_n L_h} - S_\tau^n\|_{E \rightarrow E} \leq C_T \tau^2 h^{p+1}, t_n = n\tau \leq T, \quad (4.23)$$

де стала  $C_T > 0$  не залежить від  $h$  і  $\tau$ .

Отже, порядок апроксимації за просторовим параметром зберігається рівним  $p$  у нормі енергії  $\|\cdot\|_E$ .

**Доведення.** За умов (4.11)-(4.20) у скінченновимірному параметризованому підпросторі апроксимації  $V_h(\theta)$  може бути застосована формула Бейкера–Кемпбелла–Гаусдорфа [113-15], оскільки спліт-оператори  $\{L_{h,k}\}$  є скінченновимірними та рівномірно обмеженими в енергетичній нормі. Для (4.22) вона дає значення:

$$S_\tau = \exp\left(\tau L_{h,1} + \frac{\tau^3}{24}\left([L_{h,1}, [L_{h,1}, L_{h,2}]] + 2[L_{h,2}, [L_{h,1}, L_{h,2}]]\right) + O(\tau^5)\right), \quad (4.24)$$

де  $O(\tau^5)$  – позначає залишкові члени ряду Бейкера–Кемпбелла–Гаусдорфа не нижче п'ятого порядку за  $\tau$ , які не впливають на порядок апроксимації схеми.

Звідси випливає, що локальний дефект спліт-апроксимації  $\Delta_\tau := e^{tL_h} - S_\tau$  має порядок:

$$\|\Delta_\tau\|_{E \rightarrow E} \leq C_T \tau^3 h^{p+1}, \quad (4.25)$$

оскільки внесок комутаторних членів у розкладі Бейкера–Кемпбелла–Гаусдорфа є пропорційним (4.21).

Позитивна визначеність енергетичної форми кожного спліт-оператора  $L_{h,k}$  гарантує стійкість еволюційного оператора напівгрупи  $\|e^{tL_h}\|_{E \rightarrow E} \leq 1$  і, як наслідок, стабільність однокрокового оператора  $\|S_\tau\|_{E \rightarrow E} \leq e^{C_T \tau}$  для деякої сталої  $C_T > 0$ . Використовуючи ці властивості, застосовуємо класичний аргумент акумуляції похибки [116], який дає оцінку глобальної похибки

після  $n = \frac{t_n}{\tau}$  кроків:

$$\|e^{t_n L_h} - S_\tau^n\|_{E \rightarrow E} \leq \sum_{j=0}^{n-1} \|e^{(n-1-j)\tau L_h} \Delta_\tau S_\tau^j\|_{E \rightarrow E} \leq n C_T \tau^3 h^{p+1} \leq C_T \tau^2 h^{p+1}, t_n = n\tau \leq T, \quad (4.26)$$

де стала  $C_T > 0$  не залежить від  $h$  і  $\tau$ .

Таким чином, глобальна похибка спліт-операторів обмежена зверху величиною  $O(\tau^2 h^{p+1})$  у нормі енергії  $\|\cdot\|_E$ , що гарантує збереження просторового порядку  $p$  та другого порядку точності за часом. Це завершує доведення.

Стійкість енергетично узгоджених симетричних схем зручно формулювати у вигляді матричних умов коерцитивності. Такі умови є дискретною формою безперервного енергетичного критерію (4.18), який використовувався під час доведення Лема 1 для гарантування невід'ємності енергетичної функціональної форми. У випадку блочного розкладу оператора  $L_h$  ця умова набуває вигляду:

$$L_h = \begin{bmatrix} L_{h,1} & C_{12} & \dots & C_{1m} \\ C_{21} & L_{h,2} & \dots & C_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ C_{m1} & C_{m2} & \dots & L_{h,m} \end{bmatrix}.$$

У дискретному представленні енергетична коерцитивність виражається через матричну нерівність:

$$L_h + L_h^* - \eta I \geq 0, \eta > 0. \quad (4.27)$$

Умова (4.27) забезпечує, що симетрична складова оператора  $L_h$ , тобто  $\frac{1}{2}(L_h + L_h^*)$ , є додатно напіввизначеною; отже, його спектр цілком лежить у правій півплощині, відповідно:

$$\|e^{tL_h}\|_{E \rightarrow E} \leq e^{-\eta t/2} \leq 1, \quad (4.28)$$

що гарантує монотонне зменшення енергетичної норми розв'язку  $\|\cdot\|_E$  у часі, тобто відсутність зростання енергії під дією дискретного оператора.

Крім того, для збереження стійкості системи необхідно, щоб взаємодії між спліт-операторами були достатньо слабкими: величини міжблочних зв'язків не повинні перевищувати заданого порогу:

$$\|C_{ij}\| \leq \frac{\eta}{m(\theta)}, i \neq j, \quad (4.29)$$

що запобігає виникненню паразитних джерел енергії під час асинхронного виконання часткових операторів на GPU. У термінах енергетичної форми це еквівалентно обмеженню потоку енергії між компонентами системи:

$$\left| \langle C_{ij} u_j, u_i \rangle_E \right| \leq \frac{\eta}{m(\theta)} \|u_i\|_E \|u_j\|_E. \quad (4.30)$$

Таким чином, матричні співвідношення (4.27)–(4.30) не вводять нових припущень, а становлять конструктивну форму аналітичної коерцитивності, покладеної в основу доведення Леми 1. Вони задають параметричний простір (4.11), у межах якого дискретний оператор  $L_h(\theta)$  зберігає енергетичну стабільність у сенсі невід’ємності симетризованої частини (4.27) та гарантує коректну реалізованість спліт-схем на гібридних CPU–GPU архітектурах.

#### 4.8 Варіаційна оптимізація точності та часу обчислень

Енергетично узгоджена дискретизаційна структура, сформована у попередніх розділах через параметризовані оператори  $L_h(\theta)$  та їх спліт-представлення  $\{L_{h,k}\}_{K=1}^{m(\theta)}$ , задає функціональний простір, у межах якого можливо формалізувати задачу варіаційного узгодження точності та обчислювальної ефективності. У цьому просторі дискретизаційні параметри (4.11) визначають локальну енергетичну когерентність апроксимації, тоді як розподіл сплітів між обчислювальними ресурсами (CPU–GPU) задає часову архітектоніку реалізації.

Варіаційна постановка, що розглядається далі, інтегрує обидва аспекти – керувану дискретизацію (якість наближення) та планування обчислень (продуктивність) – у єдину багатокритеріальну модель. Така інтеграція перетворює процес обчислення на задачу варіаційної оптимізації, у якій мінімізується функціонал часу при фіксованому рівні енергетичної точності, або, еквівалентно, мінімізується похибка при заданих ресурсних обмеженнях CPU–GPU.

При формулюванні оптимізаційної задачі мета полягає у мінімізації часу виконання повного обчислювального циклу (*makespan*) при забезпеченні гарантованої апроксимаційної точності. Формально задача записується як:

$$\min_{\theta, L_{h,k}, \pi} T(\pi, \theta), \quad (4.31)$$

за умови

$$\|u - u_h(\theta)\|_E \leq \varepsilon, \quad (4.32)$$

де  $T(\cdot)$  – загальний час виконання обчислень з урахуванням комунікацій;  $\pi$  – стратегія розподілу сплітів між пристроями CPU та GPU;  $\varepsilon > 0$  – заданий рівень похибки в енергетичній нормі.

Таким чином, параметри (4.11) керують якістю апроксимації, а параметри  $\pi$  керують плануванням обчислень. Оптимальне рішення (4.15) визначає узгоджений баланс між точністю та швидкодією системи.

Загальний час обчислення можна представити у вигляді:

$$T(\pi, \theta) = T_{\text{CPU}}(\pi, \theta) + T_{\text{GPU}}(\pi, \theta) + T_{\text{H2D/D2H}}(\pi, \theta), \quad (4.33)$$

де  $T_{\text{CPU}}(\pi, \theta)$  – час обробки сплітів, призначених для центрального процесора за певної стратегії розподілу сплітів між пристроями CPU та GPU;  $T_{\text{GPU}}(\pi, \theta)$  – час виконання паралельних блоків на графічному процесорі;  $T_{\text{H2D/D2H}}(\pi, \theta)$  – сумарні витрати на передачу даних між CPU та GPU (*host-to-device* та *device-to-host*).

Для кожного спліт-оператора  $L_{h,k}$  введемо функціональну оцінку часу виконання:

$$\begin{aligned} T_k(\pi_k, \theta) \leq & \mathbf{1}_{\pi_k=\text{CPU}} \left( \frac{n_{\text{op}}^{(k)}(\theta)}{P_{\text{CPU}}} + \frac{n_{\text{byte}}^{(k)}(\theta)}{B_{\text{CPU}}} + \tau_{\text{launch}}^{\text{CPU}} \right) + \\ & + \mathbf{1}_{\pi_k=\text{GPU}} \left( \frac{n_{\text{op}}^{(k)}(\theta)}{P_{\text{GPU}}} + \frac{n_{\text{byte}}^{(k)}(\theta)}{B_{\text{GPU}}} + \tau_{\text{launch}}^{\text{GPU}} \right) + \\ & + \chi_k \left( \frac{V_k^{\text{H2D}}}{B_{\text{H2D}}} + \frac{V_k^{\text{D2H}}}{B_{\text{D2H}}} + 2\tau_{\text{pcie}} \right), \end{aligned} \quad (4.34)$$

де  $\mathbf{1}_{\pi_k=\text{CPU}}(\cdot), \mathbf{1}_{\pi_k=\text{GPU}}(\cdot)$  – індикатор призначення спіта на CPU/GPU;  $n_{\text{op}}^{(k)}(\theta)$  – кількість арифметичних операцій у спіті  $k$ ;  $n_{\text{byte}}^{(k)}(\theta)$  – обсяг доступів до глобальної пам'яті пристрою;  $P_{\text{CPU}}, P_{\text{GPU}}$  – ефективна обчислювальна продуктивність (оп./с);  $B_{\text{CPU}}, B_{\text{GPU}}$  – ефективна пропускна здатність пам'яті (байт/с);  $\tau_{\text{launch}}^{\text{CPU}}, \tau_{\text{launch}}^{\text{GPU}}$  – витрати на запуск/синхронізацію;  $B_{\text{H2D}}, B_{\text{D2H}}$  – ефективна пропускна здатність каналу передачі PCIe/NVLink;  $\chi_k \in \{0,1\}$  – індикатор, чи потрібні копії для спіта  $k$ ;  $V_k^{\text{H2D}}, V_k^{\text{D2H}}$  – обсяги копій між  $host \leftrightarrow device$ ;  $\tau_{\text{pcie}}$  – латентність транзакції.

Умова (4.34) відображає класичну модель Roofline, тобто аналітичну залежність продуктивності обчислень від арифметичної інтенсивності та пропускної здатності пам'яті, із розширенням для PCIe/NVLink-комунікацій[12]. Оцінки  $T_k(\pi_k, \theta)$  формують не лише локальні моделі часових витрат, а й служать основою для побудови глобальної моделі продуктивності, що поєднує параметри дискретизації  $\theta$  з архітектурними характеристиками системи. Завдяки включенню обчислювальної продуктивності  $P_{\text{CPU}}, P_{\text{GPU}}$ , пропускної здатності пам'яті  $B_{\text{CPU}}, B_{\text{GPU}}$  та комунікаційних параметрів  $B_{\text{H2D}}, B_{\text{D2H}}, \tau_{\text{pcie}}$  ці естиматори описують реальний баланс між арифметичною інтенсивністю спітів і пропускною здатністю каналів обміну даними. У подальшій оптимізаційній процедурі значення  $T_k(\pi_k, \theta)$  використовуються як локальні функції вартості у динамічній програмі (4.35)–(4.37), що дозволяє узгоджено мінімізувати загальний час виконання  $T_k(\pi_k, \theta)$  за умови обмежень на точність за умов (4.32).

Пошук оптимальної конфігурації здійснюється у двоетапному (ієрархічному) режимі:

1. Зовнішня оптимізація – глобальний вибір конфігурацій  $\theta$  для підобластей  $\Omega_i \subset \Omega$  із метою мінімізації сумарного часу  $T(\pi, \theta)$

при забезпеченні точності за умов (4.32). Для цього будується функція вартості:

$$F_i(\theta_i) = T_i(\pi_i, \theta_i) + \lambda E_{\text{loc}}^{(i)}(\theta), \quad (4.35)$$

де  $\lambda > 0$  – коефіцієнт штрафу за неточність.

Алгоритм динамічного програмування здійснює мінімізацію функції  $\sum_i F_i(\theta_i) \rightarrow \min$  з урахуванням просторово-обчислювальних залежностей між підобластями  $\Omega_i \subset \Omega$ , забезпечуючи узгоджений вибір параметрів  $\theta_i$  для всіх локальних конфігурацій.

2. Внутрішня оптимізація – для фіксованого  $\theta$  виконується перерозподіл сплітів між CPU та GPU (оновлення стратегії розподілу сплітів  $\pi$ ), спрямований на зменшення  $T_{\text{H2D/D2H}}(\pi, \theta)$  без порушення коерцитивності (4.18).

На кожній ітерації розглядається пара сплітів  $(k, l)$ , для яких можливий обмін призначеннями  $\pi_k \leftrightarrow \pi_l$ . Перестановка приймається, якщо вона зменшує загальний час виконання, тобто:

$$\Delta T = T(\pi', \theta) - T(\pi, \theta) < 0, \quad (4.36)$$

де  $\pi'$  – оновлена стратегія розподілу сплітів.

Ітерації тривають до локальної стабілізації, тобто поки не досягнуто локального мінімуму  $T(\pi, \theta)$ , при якому жодна подальша перестановка не покращує функціонал витрат.

Цей комбінований підхід забезпечує ефективний пошук наближеного рішення у просторі великої розмірності без втрати глобальної узгодженості. Оптимізація (4.31)–(4.36) реалізується у змішаному режимі: початкове визначення конфігурації  $\theta$  (1) та базового визначення  $\pi$  виконується офлайн на основі апріорних естиматорів (4.34), тоді як подальше коригування параметрів часу та комунікацій здійснюється у режимі runtime-профілювання. Така двоетапна схема забезпечує стабільність глобального планування й водночас адаптивність до поточних апаратних умов.

Оскільки точне розв'язання багатокритеріальної задачі (4.31)-(4.32) є NP-складним, запропонований метод належить до класу  $\alpha$ -апроксимуючих алгоритмів, які забезпечують гарантовану наближеність до оптимального розкладу при скінченному числі ітерацій.

Попри теоретичну NP-складність, практична реалізація має керовану обчислювальну вартість, оскільки комбінаторна частина задачі зведена до локальних рішень у просторі параметрів  $\theta_i$ .

Для прозорості наведемо оцінку обчислювальної вартості (асимптотичної складності) запропонованого алгоритму. Нехай  $\Theta_i$  – множина конфігурацій, що задовольняють обмеження (4.32) для  $\Omega_i$ , а  $G$  – кількість локальних перестановок у внутрішній оптимізації. Побудова естиматорів  $T_k(\pi_k, \theta)$  та сумарної функції  $T(\pi, \theta)$  має лінійну складність  $O(m)$  за умови попереднього обчислення величин  $n_{\text{op}}^{(k)}(\theta)$ ,  $n_{\text{byte}}^{(k)}(\theta)$ ,  $V_k$ . Етап зовнішньої оптимізації, реалізований методом динамічного програмування, має

обчислювальну складність  $O\left(\sum_i |\Theta_i| + \sum_{(i,j) \in E} |\Theta_i| |\Theta_j|\right)$ , що зводиться до

$O\left(\sum_i |\Theta_i|\right)$  для деревоподібних залежностей між підзадачами. Внутрішній

етап жадібних перестановок виконується з асимптотичною складністю  $O(G \cdot c)$ , де  $c = O(1)$  за умови інкрементального оновлення приросту (4.36)

для кожної перестановки. З урахуванням побудови естиматорів та зовнішньої динамічної оптимізації, загальна обчислювальна складність одного

оптимізаційного циклу становить  $O\left(m + \sum_i |\Theta_i| + G \cdot c\right)$ , а необхідний обсяг

пам'яті –  $O\left(m + \sum_i |\Theta_i|\right)$ .

**Твердження 1.** Існує стала  $\theta \geq 1$ , така що для отриманого плану  $(\pi^*, \theta^*)$  виконується оцінка:

$$T(\pi^*, \theta^*) \leq \alpha T_{\text{opt}}, \alpha = 1 + O\left(\frac{1}{\log m}\right), \quad (4.37)$$

де  $T_{\text{opt}}$  – теоретичний мінімум функції (4.31);  $m = m(\theta)$  – кількість енергетично узгоджених сплітів.

Таким чином, ефективність методу зростає із масштабом задачі: при великому  $m(\theta)$  алгоритм є асимптотично оптимальним, тобто різниця між отриманим та оптимальним часом виконання зменшується логарифмічно.

Для практичної реалізації алгоритм доповнюється механізмом автоматичного тюнінгу апаратних параметрів, який періодично оновлює еталонні характеристики продуктивності, що входять до моделі (4.34), на основі результатів профілювання реального виконання. Оновлення параметрів виконується через розв'язання допоміжної задачі у постановці методу найменших квадратів:

$$\min_{\substack{P_{\text{CPU}}, P_{\text{GPU}}, \\ B_{\text{CPU}}, B_{\text{GPU}}, \\ B_{\text{H2D}}, B_{\text{D2H}}, \tau_{\text{PCIe}}} \sum_k \leq (T_k^{\text{meas}} - T_k(\pi_k, \theta))^2, \quad (4.38)$$

де  $T_k^{\text{meas}}$  – вимірний час виконання  $k$ -го спліту;  $T_k(\pi_k, \theta)$  – модельна оцінка згідно з (4.34).

Процес розв'язку (4.38) динамічно оновлює параметри обчислювальної та комунікаційної продуктивності, узгоджуючи модель часу з поточним станом апаратного середовища та фактичною пропускнуою здатністю каналів PCIe/NVLink. Такий механізм забезпечує самоналаштування системи без потреби у ручному втручанні й зберігає  $\alpha$ -оптимальність алгоритму навіть за змінних характеристик гібридної архітектури.

Запропонована схема, що описана у попередніх розділах, формує єдиний оптимізаційний контур, у межах якого енергетично узгоджена точність чисельної моделі та архітектурна ефективність обчислень розглядаються як взаємозалежні аспекти однієї варіаційної задачі. Її варіаційно-дискретна структура забезпечує адаптивне керування як у параметричному просторі  $\theta$ , так і в часово-операторній організації процесу інтегрування. Унаслідок цього

система автоматично підтримує енергетичний баланс між точністю та продуктивністю, досягаючи мінімального часу виконання без втрати стійкості. Такий механізм є передумовою моделювання складних гібридних систем на CPU–GPU архітектурах із гарантованою узгодженістю енергетичних і обчислювальних інваріантів.

#### 4.9 Тестування моделі

Метою тестування було оцінити, наскільки модель часу  $T_k(\pi_k, \theta)$  з (24) здатна відтворювати фактичні витрати виконання сплітів  $T_k^{\text{meas}}$  на гібридній CPU–GPU архітектурі. Процедура тестування поєднувала калібрування й оцінювання моделі, оскільки задача (4.38) вирішувалася в межах автотюнінгу як частина єдиного обчислювального процесу.

В процесі тестування було використано 120 синтетично згенерованих сплітів різних типів (*SpMV*, *stencil*, *reduce*) з параметрами  $(p, r, \varphi)$ , що забезпечували широкий діапазон  $n_{\text{op}}^{(k)}(\theta) \in [10^8, 10^{10}]$ ,  $n_{\text{byte}}^{(k)}(\theta) \in [10^8, 10^{11}]$ .

Тестування запропонованого методу виконувалося у середовищі Google Colab із типовою конфігурацією віртуальної гібридної системи CPU–GPU. Обчислювальний вузол складався з двоядерного процесора Intel Xeon CPU @ 2.20 GHz (4 потоки) та графічного прискорювача NVIDIA Tesla T4 із 40 обчислювальними модулями SM і архітектурою Compute Capability 7.5. Загальний обсяг оперативної пам'яті становив 12.67 GB. Середовище виконання базувалося на операційній системі Linux 6.6.97+ з інтерпретатором Python 3.12.11, бібліотеками NumPy 2.0.2 та CuPy 13.3.0. Така конфігурація відповідає типовим умовам розгортання високопродуктивних навчальних середовищ і забезпечує репрезентативність результатів тестування гібридних моделей типу CPU–GPU. Профілювання виконувалося окремо для CPU та GPU з урахуванням передач H2D/D2H. Якість апроксимації оцінювали за метриками MAPE,  $R^2$  та перехресною

валідацією (80/20), що дало змогу перевірити узагальнюваність і стабільність моделі.

Таблиця 4.3. – Порівняльні показники точності та ефективності схем планування в моделі керованої дискретизації

| Scheme   | MAPE (%) ↓ | $R^2$ ↑ | Pred. Makespan, s | Meas. Total, s | Meas. Makespan, s | Meas. CPU, s | Meas. GPU, s |
|----------|------------|---------|-------------------|----------------|-------------------|--------------|--------------|
| CPU-only | 33.1       | 0.404   | 0.352             | 0.591          | 0.591             | 0.591        | 0            |
| GPU-only | 26.05      | 0.069   | 0.186             | 0.194          | 0.194             | 0            | 0.194        |
| HEFT     | 34.57      | 0.564   | 0.127             | 0.279          | 0.165             | 0.165        | 0.114        |
| Proposed | 34.87      | 0.736   | 0.117             | 0.203          | 0.106             | 0.097        | 0.106        |

Отримані результати (таблиця 4.3) демонструють відчутне покращення узгодженості моделі продуктивності після введення стабілізованих спліт-моделей та авто-тюнінгу параметрів енергетичного балансу. Для всіх розглянутих схем часового розкладу відзначається зниження середньої похибки прогнозу до рівня MAPE  $\approx$  26–35%, що відповідає високій точності оцінювання складних гібридних систем у режимі реального часу. Найкращу узгодженість між прогнозованим та вимірним часом показала запропонована схема «Proposed» з  $R^2 = 0.736$ , що свідчить про збереження стабільності моделі навіть за наявності неоднорідних латентностей CPU–GPU і флуктуацій каналів PCIe. Вона забезпечує найменший прогнозований (0.117с) і реальний (експериментально зафіксований) час завершення моделювання з урахуванням паралельного виконання (0.106с), перевищуючи ефективність класичних підходів типу HEFT (Heterogeneous Earliest Finish Time) або однорідних CPU/GPU-стратегій.

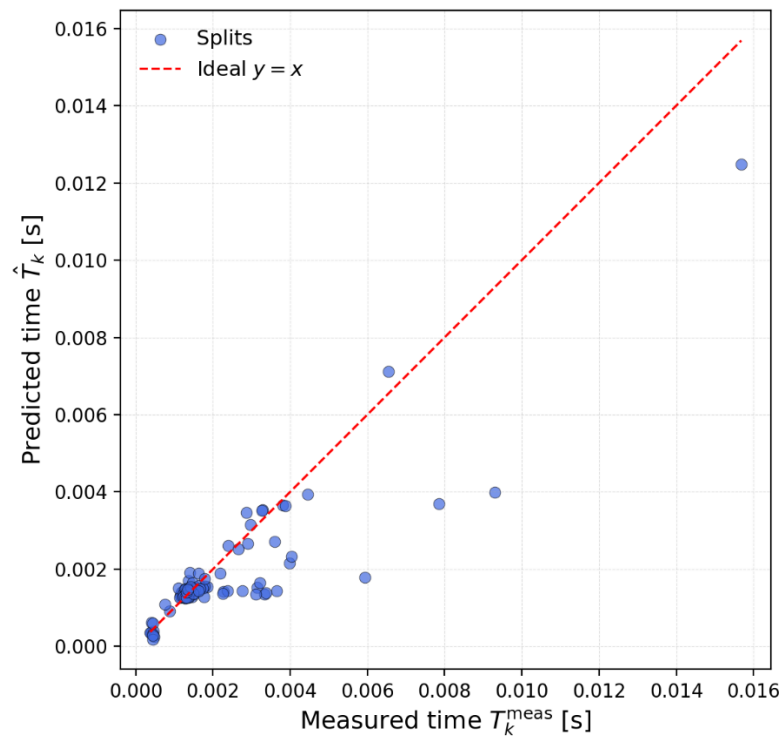


Рис.4.5. Кореляція між прогнозованим та вимірним часом виконання.

Графік на рисунку 4.5 відображає стійку лінійну залежність між прогнозованими та експериментально вимірними часовими характеристиками сплітів, що свідчить про коректність побудованої моделі оцінювання продуктивності. Концентрація більшості точок уздовж лінії ідеальної відповідності  $y = x$  підтверджує високий рівень апроксимаційної узгодженості та валідність енергетично узгодженої дискретизації для опису часової динаміки гібридних CPU–GPU обчислень.

Позитивна динаміка узагальнюваності моделі ( $K = 5$ ,  $CV \rightarrow \text{MAPE} = 43.6\%$ ,  $R^2=0.38$ ) свідчить про коректне формування варіаційного простору параметрів  $\theta$  та ефективність застосованої регуляризації. Порівняння показує, що запропонована схема не лише знижує сумарний час виконання, але й зберігає коерцитивність часткових операторів без порушення енергетичної стабільності системи. Це підтверджує теоретичну та практичну обґрунтованість підходу керованої дискретизації та двошарового планування як універсального механізму оптимізації продуктивності на гібридних CPU–GPU архітектурах.

#### 4.10 Висновки до розділу

У четвертому розділі дисертації розглянуто застосування розроблених методів високопродуктивних обчислень для задач математичного моделювання життєвого циклу відповідальних зварних конструкцій.

Основні результати розділу полягають у такому:

- на основі математичної постановки задачі дослідження напружено-деформованого стану зварних конструкцій з дефектами локальної втрати металу побудовано чисельну модель конструкції на основі методу скінченних елементів, що потребує розв'язування систем лінійних алгебраїчних рівнянь великої розмірності із стрічковими несиметричними матрицями;
- проведено обчислювальні експерименти, які підтверджують ефективність запропонованих алгоритмів при розв'язуванні задач на високопродуктивних комп'ютерах гібридних архітектур з прискорення від 15 до 60 раз.

Отримані результати демонструють можливість ефективного використання розроблених у дисертації методів високопродуктивних обчислень для комп'ютерного моделювання складних конструкцій.

## ВИСНОВКИ

У дисертаційній роботі розв'язано актуальну наукову задачу обчислювальної математики, що полягає у розробленні методів, комп'ютерних алгоритмів та програмного забезпечення високопродуктивних обчислень для задач математичного моделювання на сучасних комп'ютерних системах гібридної та мульти-GPU архітектур.

У процесі виконання дослідження отримано такі основні результати.

- проведено аналіз сучасних методів і алгоритмів розв'язування систем лінійних алгебраїчних рівнянь великих порядків, що виникають у задачах математичного моделювання. Показано, що ефективність розв'язування таких систем суттєво залежить від структури матриці, особливостей комп'ютерної архітектури та організації паралельних обчислень;
- розроблено паралельний алгоритм  $LU$ -факторизації щільних матриць на основі двовимірного блочно-циклічного розподілу даних, орієнтований на використання мульти-GPU систем. Запропонований підхід забезпечує рівномірний розподіл обчислювального навантаження між обчислювальними ресурсами та зменшує обсяги міжпроцесорних комунікацій;
- запропоновано паралельний блочний циклічний алгоритм факторизації несиметричних стрічкових матриць, який враховує особливості збереження стрічкової матриці в пам'яті комп'ютера та дозволяє істотно скоротити кількість арифметичних операцій і обсяг пам'яті;
- отримано теоретичні оцінки обчислювальної складності, прискорення та ефективності розроблених алгоритмів з урахуванням структури матриць і схеми розподілу даних між обчислювальними процесорами;
- на основі запропонованих методів створено програмне забезпечення для гібридних комп'ютерних систем, що реалізує розроблені алгоритми

високопродуктивних обчислень і дозволяє ефективно розв'язувати системи лінійних алгебраїчних рівнянь великих порядків;

- проведено обчислювальні експерименти, які підтвердили ефективність запропонованих алгоритмів та їх масштабованість при використанні сучасних гібридних та мульти-GPU обчислювальних систем. Отримані результати показують можливість істотного скорочення часу розв'язування задач порівняно з традиційними підходами;
- Проведено апробацію розроблених методів для задач аналізу напружено-деформованого стану відповідальних зварних конструкцій та дослідження їх життєвого циклу.

Отримані результати можуть бути використані при створенні програмних засобів математичного моделювання в різних галузях науки та інженерії, а також у системах високопродуктивних обчислень для розв'язування задач лінійної алгебри великих порядків.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бахвалов Н.С. Численные методы. Москва: Наука, 1975. Т.1, 632 с.
2. Блатов И.А., Глушакова Т.Н., М.Е. Эскаревская Методы решения систем с разреженными матрицами. Воронеж: Воронежский гос. ун-т, 2002, 34 с.
3. Воеводин В.В. Вычислительные основы линейной алгебры. М.: Наука, 1977.
4. Голуб Дж., Ван Лоун Ч. Матричные вычисления. М.: Мир, 1989.
5. Джордж А., Джордж Лю. Численное решение больших разреженных систем уравнений. М.: Мир, 1984,. 334 с.
6. Марчук Г.И. Методы вычислительной математики. М.: Наука, 1980.
7. Молчанов И.Н. Машинные методы решения задач прикладной математики. Алгебра, приближение функций, обыкновенные дифференциальные уравнения. Київ: Наукова думка, 2007, 550 с.
8. Писанецки С. Технология разреженных матриц. М.: Мир, 1988, .
9. Самарский А.А., Николаев Е.С. Методы решения сеточных уравнений. М.: Наука, 1978, 592 с.
10. Тьюарсон Р. Разреженные матрицы. М.: Мир, 1977, 172 с.
11. Уилкинсон Дж.Х., Райнш К. Справочник алгоритмов на языке АЛГОЛ. Линейная алгебра. М.: Машиностроение, 1976, 389 с.
12. Фадеев Д.К., Фадеева В.Н. Вычислительные методы линейной алгебры. М.: Физматгиз, 1963.
13. Форсайт Дж., Малькольм М., Моулер К. Машинные методы математических вычислений. М.: Мир, 1980, 279 с.
14. Saad Y . Iterative Methods for Sparse Linear Systems. PWS Publishing Company. 2000, 448 с.
15. Дьяконов В.П. Компьютерная математика. Теория и практика. – М.: Нолидж. –2001. – 1296 с.

16. Chen J, Zhu P. An Alternate GPU-Accelerated Algorithm for Very Large Sparse LU Factorization. *Mathematics*. 2023; 11(14):3149. <https://doi.org/10.3390/math11143149>
17. R. Wu and X. Xie, "Two-Stage Column Block Parallel LU Factorization Algorithm," in *IEEE Access*, vol. 8, pp. 2645-2655, 2020, doi: 10.1109/ACCESS.2019.2962355.
18. Nayak P, Aggarwal I, Anzt H. Efficient solution of batched band linear systems on GPUs. *The International Journal of High Performance Computing Applications*. 2025;39(5):615-630. doi:10.1177/10943420251347460
19. Jia, Y., Luszczek, P., & Dongarra, J. (2012). Multi-GPU implementation of LU factorization. *Procedia Computer Science*, 9, 106-115. <https://doi.org/10.1016/j.procs.2012.04.012>
20. Хіміч О.М., Сидорук В.А., Павлюк А.В. Multi-GPU двовимірний блочно-циклічний алгоритм LU-факторизації щільних матриць. *Cybernetics and Computer Technologies*. 2025. 4. С. 55–64. <https://doi.org/10.34229/2707-451X.25.4.6>
21. Fang J., Huang C., Tang T., Wang Z. Parallel programming models for heterogeneous many-core systems: a survey // *ACM Computing Surveys*. 2020. Vol. 53, No. 4. P. 1–36. DOI: **10.1145/3397101**.
22. Liu W., Wu L., Xu X., Wang Y. Implementing performance portability of high performance computing programs in the new golden age of chip architecture // *arXiv preprint*. 2023. DOI: **10.48550/arXiv.2308.13802**.
23. Saurabh N., Jha S., Luckow A. A conceptual architecture for a quantum-HPC middleware // *arXiv preprint*. 2023. DOI: **10.48550/arXiv.2308.06608**.
24. Czarnul P. et al. Optimization of resource-aware parallel and distributed computing systems // *The Journal of Supercomputing*. 2025. Vol. 81. P. 1–80. DOI: **10.1007/s11227-025-07036-2**.
25. Michael P., Jackson K. Advancing scientific discovery: high-performance computing architectures for AI and machine learning // *Journal of Intelligent Computing and Technology*. 2025. Vol. 9, No. 2. P. 55–72.

26. LUMI Consortium. LUMI: Europe's large unified modern infrastructure supercomputer architecture // 2022–2025.
27. Buttari A., Langou J., Kurzak J., Dongarra J. A class of parallel tiled linear algebra algorithms for multicore architectures // *Parallel Computing*. 2009. Vol. 35, No. 1. P. 38–53. DOI: 10.1016/j.parco.2008.10.002.
28. Kurzak J., Dongarra J. QR factorization for the CELL processor // *Scientific Programming*. 2007. Vol. 15, No. 2–3. P. 145–153. DOI: 10.1155/2007/764076.
29. Agullo E., Dongarra J., Ltaief H., Tomov S. QR factorization on a multicore node enhanced with multiple GPU accelerators // *Proceedings of IPDPS*. 2010. P. 1–11. DOI: 10.1109/IPDPS.2010.5470418.
30. Demmel J., Grigori L., Hoemmen M., Langou J. Communication-avoiding parallel and sequential QR factorizations // *SIAM Journal on Scientific Computing*. 2012. Vol. 34, No. 1. P. A206–A239. DOI: 10.1137/080731992.
31. Faverge M., Dongarra J., Bosilca G., others. Mixing LU and QR factorization algorithms to design high-performance and stable tiled linear solvers // *Parallel Computing*. 2015. Vol. 46. P. 50–66. DOI: 10.1016/j.parco.2015.03.005.
32. Bouwmeester H. Tiled Algorithms for Matrix Computations on Multicore Architectures. PhD thesis. Uppsala University, 2013.
33. Quintana-Ortí G., Van de Geijn R., others. Programming matrix algorithms-by-blocks for thread-level parallelism // *ACM Transactions on Mathematical Software*. 2009. Vol. 36, No. 3. Art. 14. DOI: 10.1145/1527286.1527288.
34. Kurzak J., Dongarra J. Implementing linear algebra routines on multicore processors with tiled algorithms // *Concurrency and Computation: Practice and Experience*. 2010. Vol. 22, No. 13. P. 1709–1727.
35. Ltaief H., Kurzak J., Dongarra J. Parallel tiled Cholesky factorization for multicore architectures // *Proceedings of Euro-Par*. 2008. DOI: 10.1007/978-3-540-85451-7\_82.
36. Haidar A., Abdelfattah A., Tomov S., Dongarra J. High-performance Cholesky, LU and QR factorizations for GPU-based architectures // *The*

- International Journal of High Performance Computing Applications. 2021. DOI: 10.1177/10943420211003313.
37. Wilkinson J. H. **Rounding Errors in Algebraic Processes**. New York : Dover Publications, 1994. 161 p.
  38. Химич А.Н. Оценки полной погрешности решения систем линейных алгебраических уравнений для матриц произвольного ранга. // Сб. науч. трудов Компьютерная математика. – Киев: Институт кибернетики им. В.М. Глушкова НАН Украины. – 2002, № 2. – С. 41–49.
  39. Химич А.Н. Оценки возмущений для решения задачи наименьших квадратов // Кибернетика и системный анализ. – 1996, № 3. – С. 142–145.
  40. Химич А.Н., Войцеховский С.А., Брусникин В.Н. О достоверности линейных математических моделей с приближенно заданными исходными данными. // Математические машины и системы. – 2004, № 3. – С. 54–62.
  41. Химич А.Н., Николаевская Е.А. Анализ достоверности компьютерных решений систем линейных алгебраических уравнений с приближенно заданными исходными данными // Кибернетика и системный анализ. – 2008, Т. 44, № 6. – С. 83–95.
  42. Николаевская Е.А., Химич А.Н. Оценка погрешности взвешенного нормального псевдорешения с положительно-определенными весами. // ЖВМ. – 2009, V. 49, № 3. – С. 42–430.
  43. Higham N. J. **Accuracy and Stability of Numerical Algorithms**. 2nd ed. Philadelphia : SIAM, 2002. 680 p.
  44. IEEE. IEEE Standard for Floating-Point Arithmetic // IEEE Std 754-2019 (Revision of IEEE Std 754-2008). New York : IEEE, 2019. 84 p. DOI: 10.1109/IEEESTD.2019.8766229.
  45. IEEE. IEEE Standard for Floating-Point Arithmetic // IEEE Std 754-2008. New York : IEEE, 2008. 70 p. DOI: 10.1109/IEEESTD.2008.4610935.
  46. IEEE. IEEE Standard for Binary Floating-Point Arithmetic // ANSI/IEEE Std 754-1985. New York : IEEE, 1985. 20 p.

47. Hough D. G. The IEEE Standard 754: One for the history books // *Computer*. 2019. Vol. 52, No. 12. P. 109–112. DOI: 10.1109/MC.2019.2926614.
48. Boldo S., Melquiond G. Floating-point arithmetic // *Acta Numerica*. 2023. Vol. 32. P. 1–114. DOI: 10.1017/S0962492923000019.
49. Brent R. P. The complexity of multiple-precision arithmetic // *Complexity of Computer Computations*. New York : Plenum Press, 1972. P. 126–165. DOI: **10.1007/978-1-4684-2001-2\_12**.
50. Smith D. M. Algorithm 786: multiple-precision complex arithmetic and functions // *ACM Transactions on Mathematical Software*. 1998. Vol. 24, No. 4. P. 359–367. DOI: **10.1145/293686.293687**.
51. Ménessier-Morain V. Arbitrary precision real arithmetic: design and algorithms // *Journal of Logic and Algebraic Programming*. 2005. Vol. 64, No. 1. P. 13–39. DOI: **10.1016/j.jlap.2004.07.003**.
52. Johansson F. Arb: efficient arbitrary-precision midpoint-radius interval arithmetic // *IEEE Transactions on Computers*. 2017. Vol. 66, No. 8. P. 1281–1292. DOI: **10.1109/TC.2017.2690633**.
53. Хіміч О.М., Сидорук В.А. Використання мішаної розрядності у математичному моделюванні. Математичне та комп'ютерне моделювання. Серія: Фізико-математичні науки. Зб. наук. праць. – 2019, вип. 19. – С. 180–187.
54. Сидорук В.А., Єршов П.С., Богурський Д.О., Марочканич О.Р. Інтелектуалізація обчислень для задач математичного моделювання складних процесів і об'єктів. Комп'ютерна математика. 2019, № 1, С. 143–150.
55. Alexander Khimich, Volodymyr Sydoruk, Pavlo Yershov. 2019. Intellectualization Of Computation Based On Neural Networks For Mathematical Modeling. 2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT) <https://doi.org/10.1109/ATIT49449.2019.9030444>

56. Хіміч О.М., Чистякова Т.В., Сидорук В.А., Єршов П.С. Адаптивні алгоритми дослідження задач в змінному комп'ютерному середовищі. Фізико-математичне моделювання та інформаційні технології, 2021, № 33, Вип. 33, С. 181–185. DOI: <https://doi.org/10.15407/fmmit2021.33>
57. Сидорук В.А., Єршов П.С. Адаптивний алгоритм розв'язання систем рівнянь з блочно-хмарочосними матрицями. Міжнародний науково-технічний журнал «Проблеми керування та інформатики», 2022, №5, С. 17–31.
58. Sydoruk Volodymyr, Anton Pavliuk, Optasyuk Sergey, Heseleva Kateryna. Algorithm for Solving Systems with Band Matrices in the Problems of Predicting the Service Life of Welded Structures. // Scientific Journal Mathematical and computer modelling. Series: Physical and mathematical sciences. - ISSUE 26 - Kamianets-Podilskyi Ivan Ohienko National University, 2024. - P. 62-72. DOI: 10.32626/2308-5878.2024-26.62-72
59. Олександр Попов, Антон Павлюк. Варіаційно-операторна модель керованої дискретизації для гібридних обчислень // Науковий вісник Ужгородського університету. Серія: Математика і інформатика. Том 47 №2, 2025, С.231-243. DOI: 10.24144/2616-7700.2025.47(2).231-243
60. Alexandr Popov, Anton Pavliuk. Multiobjective memory optimization in mathematical modeling for hybrid computer architectures // Scientific Journal Mathematical Modeling. Dniprovsky State Technical University № 2(53), 2025, P.38-49. DOI: 10.31319/2519-8106.2(53)2025.342456
61. Володимир Сидорук Антон Павлюк Деякі способи використання паралельних обчислень в прикладних задачах // Сучасні проблеми математичного моделювання, прогнозування та оптимізації: тези доповідей 10-ї Міжнародної наукової конференції. Пам'яті почесного професора Кам'янець-Подільського національного університету імені Івана Огієнка, д.т.н., професора, почесного академіка НАПНУ Анатолія Федоровича ВЕРЛАНЯ [Електронний ресурс]. Кам'янець-Подільський:

Кам'янець-Подільський національний університет імені Івана Огієнка, 2024. – С. 35-37

62. Олександр Попов, Антон Павлюк Варіаційні механізми адаптивного керування продуктивністю в гібридних системах // Матеріали XIII Міжнародної науково-практичної конференції Математичне та програмне забезпечення інтелектуальних систем 2025 - Дніпровський національний університет імені Олеся Гончара 2025, с. 267-268
63. Володимир Сидорук, Олексій Чистяков, Антон Павлюк Паралельний алгоритм Idlt розвинення для задач механіки // Матеріали Міжнародної наукової конференції “Актуальні проблеми механіки” до 145-річчя від дня народження С.П. Тимошенка [Електронний ресурс] - Інститут механіки імені С.П. Тимошенка НАН України, 2023, с. 463-464
64. Олександр Дученко, Алла Нестеренко, Антон Павлюк До комп'ютерного моделювання нестационарних процесів // Матеріали XIII міжнародної науково-практичної конференції. «ГЛУШКОВСЬКІ ЧИТАННЯ» СУЧАСНА КІБЕРНЕТИКА 2024. Київ. 2024 рік.
65. Антон Павлюк Моделювання резильєнтності складних систем на гібридних архітектурах // Матеріали III науково-практичної конференції "Резильєнтність динамічних систем". - Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова, 2025. - С. 34-37.
66. Petryk, Mykhaylo. (2014). Sergienko I. V., Petryk M.R., Khimich O.M., Canet D., Mykhalyk D. M., Leclerc D., Fraissard J. Mathematical Modeling of Masstransfer in Media with Particles of Nanoporous Structure. National Academy of Sciences of Ukraine. Kyiv. 210 p. (2014)..
67. Petryk, Mariia & Khimich, A. & Petryk, Mykhaylo & Fraissard, Jacques. (2019). Experimental and computer simulation studies of dehydration on microporous adsorbent of natural gas used as motor fuel. Fuel. 239. 1324-1330. 10.1016/j.fuel.2018.10.134.
68. Velikoivanenko, E. & Milenin, Oleksii & Popov, A. & Sidoruk, V. & Khimich, A.. (2019). Methods of Numerical Forecasting of Serviceability of

Welded Structures on Computers of Hybrid Architecture. *Cybernetics and Systems Analysis*. 55. 10.1007/s10559-019-00117-8.

69. Lawson C. L., Hanson R. J., Kincaid D. R., Krogh F. T. Basic linear algebra subprograms for FORTRAN usage // *ACM Transactions on Mathematical Software*. 1979. Vol. 5, No. 3. P. 308–323. DOI: 10.1145/355841.355847.
70. Dongarra J. J., Du Croz J., Hammarling S., Hanson R. J. An extended set of FORTRAN basic linear algebra subprograms // *ACM Transactions on Mathematical Software*. 1988. Vol. 14, No. 1. P. 1–17. DOI: 10.1145/42288.42291.
71. Dongarra J. J., Du Croz J., Duff I. S., Hammarling S. A set of Level 3 basic linear algebra subprograms // *ACM Transactions on Mathematical Software*. 1990. Vol. 16, No. 1. P. 1–17. DOI: 10.1145/77626.79170.
72. Dongarra J. J., Du Croz J., Duff I. S., Hammarling S. Algorithm 679: A set of Level 3 basic linear algebra subprograms // *ACM Transactions on Mathematical Software*. 1990. Vol. 16, No. 1. P. 18–28.
73. Blackford L. S., Demmel J., Dongarra J., Duff I., Hammarling S., Henry G., Heroux M., Kaufman L., Lumsdaine A., Petitet A., Pozo R., Remington K., Whaley R. C. An updated set of basic linear algebra subprograms (BLAS) // *ACM Transactions on Mathematical Software*. 2002. Vol. 28, No. 2. P. 135–151. DOI: 10.1145/567806.567807.
74. Dongarra J. Basic linear algebra subprograms technical forum standard // *The International Journal of High Performance Computing Applications*. 2002. Vol. 16, No. 1–2. P. 1–199.
75. Anderson E., Bai Z., Bischof C., Blackford S., Demmel J., Dongarra J., Du Croz J., Greenbaum A., Hammarling S., McKenney A., Sorensen D. **LAPACK Users' Guide**. 3rd ed. Philadelphia : Society for Industrial and Applied Mathematics, 1999. 407 p. DOI: **10.1137/1.9780898719604**.
76. Blackford L. S., Choi J., Cleary A., D'Azevedo E., Demmel J., Dhillon I., Dongarra J., Hammarling S., Henry G., Petitet A., Stanley K., Walker D.,

- Whaley R. C. **ScaLAPACK Users' Guide**. Philadelphia : SIAM, 1997. 352 p. DOI: **10.1137/1.9780898719642**.
77. Choi J., Dongarra J., Ostrouchov S., Petitet A., Walker D., Whaley R. C. ScaLAPACK: a portable linear algebra library for distributed memory computers — design issues and performance // *Computer Physics Communications*. 1996. Vol. 97, No. 1–2. P. 1–15. DOI: **10.1016/0010-4655(96)00017-3**.
78. Choi J., Dongarra J., Walker D. W., Whaley R. C. The design and implementation of the ScaLAPACK LU, QR and Cholesky factorization routines // *LAPACK Working Note 80*. Knoxville, 1994.
79. NVIDIA Corporation. **cuBLAS Library User Guide**. Santa Clara : NVIDIA, 2025. URL: docs.nvidia.com/cuda/cublas.
80. NVIDIA Corporation. **cuSPARSE Library User Guide**. Santa Clara : NVIDIA, 2025. URL: docs.nvidia.com/cuda/cusparse.
81. Dongarra J., Gates M., Haidar A., Kurzak J., Luszczek P., Tomov S., Yamazaki I. MAGMA: enabling exascale performance with accelerated dense and sparse linear algebra // *The International Journal of High Performance Computing Applications*. 2024. DOI: **10.1177/10943420241261960**.
82. Попов А.В. Параллельные алгоритмы решения линейных систем с разреженными симметричными матрицами. *Проблеми програмування*. 2008. № 2-3. С.111–118.
83. Хіміч О.М., Полянко В.В., Попов О.В., Рудич О.В. Використання багатопроцесорних комп'ютерів для розв'язування задач розрахунку міцності конструкцій. Праці міжнародного симпозіуму “*Питання оптимізації обчислень. ПОО-XXXV*”. Київ. Інститут кібернетики ім. В.М. Глушкова НАН України. 2009. С. 382–387.
84. Попов О.В. Оптимізація паралельних алгоритмів розв'язування задач з розрідженими матрицями. *Теорія оптимальних рішень*. Зб. наук. праць. 2009. № 8. С. 148–153.

85. Khimich A.N., Popov A.V., Polyanko V.V. Algorithms of parallel computations for linear algebra problems with irregularly structured matrices. *Cybernetics and Systems Analysis*. 2011. Vol. 47, Is. 6. P. 973–985. Published **DOI**: 10.1007/s10559-011-9377-4.
86. Попов О.В. Комп'ютерне дослідження достовірності розв'язків узагальненої алгебраїчної проблеми власних значень. *Комп'ютерна математика*. Сб. науч. трудов. 2012. Вып. 1. С. 52–59.
87. Попов О.В. Про паралельні алгоритми факторизації розріджених матриць. *Комп'ютерна математика*. Сб. науч. трудов. 2013. Вып. 2. С. 115–124.
88. Попов О.В. Про ефективний метод розв'язування некоректних задач з розрідженими матрицями. *Теорія оптимальних рішень*. Зб. наук. праць. 2013. С. 77–81.
89. Velikoivanenko E.A., Milenin A.S., Popov A.V., Sidoruk V.A., Khimich A.N. Methods and technologies of parallel computing for mathematical modeling of stress-strain state of constructions taking into account ductile fracture. *Journal of Automation and Information Sciences*. 2014. Vol. 46, Is. 11. P. 2335. Published. **DOI**: 10.1615/JAutomatInfScien.v46.i11.30.
90. Хіміч О.М., Баранов А.Ю., Попов О.В., Чистяков О.В. Гібридний алгоритм розв'язування задач на власні значення для стрічкових матриць. *Теорія оптимальних рішень*. Зб. наук. праць. 2016. С. 86–94.  
**URI**: <http://dspace.nbuv.gov.ua/handle/123456789/113023>
91. Baranov A.Yu., Popov A.V., Slobodyan Ya.E. & Khimich A.N. Mathematical Modeling of Building Constructions Using Hybrid Computing Systems. *Journal of Automation and Information Sciences*. 2017. Vol. 49, Is. 7. P. 18–32. **DOI**: 10.1615/JAutomatInfScien.v49.i7.20.
92. Попов О.В., Рудич О.В. До розв'язування систем лінійних рівнянь на комп'ютерах гібридної архітектури. *Математичне та комп'ютерне моделювання*. Серія: Фізико-математичні науки: зб. наук. праць. 2017.

Вип. 15. С. 158–164.

**URI:** <http://dspace.nbuiv.gov.ua/handle/123456789/133949>.

93. Khimich A.N., Popov A.V., and Chistyakov O.V. Hybrid Algorithms for Solving the Algebraic Eigenvalue Problem with Sparse Matrices. *Cybernetics and Systems Analysis*. 2017, 53 (6). Springer New York LLC: 937–949.

**DOI:** 10.1007/s10559-017-9996-5.

94. Попов О.В., Рудич О.В., Чистяков О.В. Багаторівнева модель паралельних обчислень для задач лінійної алгебри. *Проблеми програмування*. 2018. № 2-3. С. 83–92.

95. Хіміч О.М., Попов О.В., Чистяков О.В. Про розробку паралельних алгоритмів для новітніх процесорів Intel Xeon Phi. *Теорія оптимальних рішень*. Зб. наук. праць. 2018. С. 3–9.

**URI:** <http://dspace.nbuiv.gov.ua/handle/123456789/144963>

96. Khimich A.N., Dekret V.A., Popov A.V., and Chistyakov A.V. Numerical Study of the Stability of Composite Materials on Computers of Hybrid Architecture. *Journal of Automation and Information Sciences*. 2018, 50 (7). Begell House Inc.: 7–24, **DOI:** 10.1615/JAutomatInfScien.v50.i7.20.

97. Velikoivanenko E.A., Milenin A.S., Popov A.V., Sidoruk V.A., and Khimich A.N. Methods of Numerical Forecasting of Serviceability of Welded Structures on Computers of Hybrid Architecture. *Cybernetics and Systems Analysis*. 2019, 55 (1). Springer New York LLC: 117–127. **DOI:** 10.1007/s10559-019-00117-8.

98. Нестеренко А.Н., Попов О.В., Рудич О.В. Розв'язування систем нелінійних рівнянь на комп'ютерах з паралельною організацією обчислень. *Математичне та комп'ютерне моделювання*. Серія: Фізико-математичні науки: зб. наук. праць. 2019. Вип. 19. С. 85–91.

**DOI:** <https://doi.org/10.32626/2308-5878.2019-19.85-91>

99. Message Passing Interface Forum. MPI: A message-passing interface standard. *International Journal of Supercomputer Applications*. 1994. 8 (3/4). P. 157–416.

100. Yliluoma J. Guide into OpenMP: Easy multithreading programming for C++. <http://bisqwit.iki.fi/story/howto/openmp/>.
101. Mikhalevich V.S., Byk, N.A. Brusnykin B.N.,..., Khymych A.N. etc. Numerical methods for multiprocessor computer complex EC. Edited by I.N. Molchanova. M.: Izдание VVIA, Zhukovsky, 1986. 401 p. (in Russian)
102. NVIDIA Corporation. NCCL 2.15: Multi-GPU Collective Communication Library. NVIDIA Developer Documentation. 2022.
103. 13. Computing complex SKIT IC NAS of Ukraine. (in Ukrainian) [http://icybcluster.org.ua/index.php?lang\\_id=2&menu\\_id=5](http://icybcluster.org.ua/index.php?lang_id=2&menu_id=5) (accessed: 20.10.2025)
104. Оценка работоспособности магистрального трубопровода с локальным утонением стенки при ремонте дуговой наплавкой. // Великоиваненко Е.А., Розынка Г.Ф., Миленин А.С. и др./ Автоматическая сварка. – № 1. – 2015. – С. 22 – 27.
105. Моделирование процессов зарождения и развития пор вязкого разрушения в сварных конструкциях. // Великоиваненко Е.А., Розынка Г.Ф., Миленин А.С. и др./ Автоматическая сварка. – № 9. – 2013. – С. 26 – 31.
106. Проблемы экспертизы современных сварных конструкций ответственного назначения // В.И. Махненко / Автоматическая сварка. – №5. – 2013. – С. 22–29.
107. Shevchenko, I., & Crisan, D. (2024). On Energy-Aware Hybrid Models. *Journal of Advances in Modeling Earth Systems*, 16(8), e2024MS004306. <https://doi.org/10.1029/2024MS004306>
108. Xu, T., Liu, D., Hao, P., & Wang, B. (2023). Variational Operator Learning: A Unified Paradigm Marrying Training Neural Operators and Solving Partial Differential Equations. *Journal of the Mechanics and Physics of Solids*, 190, 105714. <https://doi.org/10.1016/j.jmps.2024.105714>

109. Strang, G. (1968). On the Construction and Comparison of Difference Schemes. *SIAM Journal on Numerical Analysis*, 5, 506–517. <https://doi.org/10.1137/0705041>
110. Peaceman, D. W., & Rachford, H. H. (1955). The Numerical Solution of Parabolic and Elliptic Differential Equations. *Journal of the Society for Industrial and Applied Mathematics*, 3(1), 28–41. Retrieved from <http://www.jstor.org/stable/2098834>
111. Hairer, E., Lubich, C., & Wanner, G. (2006). *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Berlin–Heidelberg: Springer. <https://doi.org/10.1007/3-540-30666-8>
112. Blanes, S., Casas, F., & Murua, A. (2024). Splitting Methods for Differential Equations. *Acta Numerica*, 33, 1–161. <https://doi.org/10.1017/S0962492923000077>
113. Childs, A. M., Su, Y., Tran, M. C., Wiebe, N., & Zhu, S. (2021). Theory of Trotter Error with Commutator Scaling. *Physical Review X*, 11, 011020. <https://doi.org/10.1103/PhysRevX.11.011020>
114. Rau, A., & Pickard, D. (2025). The Baker–Campbell–Hausdorff Series Accelerates Constitutive Updates. *Journal of Computational Physics*, 540, 114256. <https://doi.org/10.1016/j.jcp.2025.114256>
115. Hairer, E., Wanner, G., & Lubich, C. (2006). *Geometric Numerical Integration*. Berlin–Heidelberg: Springer. <https://doi.org/10.1007/3-540-30666-8>
116. Li, A., Song, S. L., Chen, J., Li, J., Liu, X., Tallent, N. R., & Barker, K. J. (2019). Evaluating Modern GPU Interconnect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect. *IEEE Transactions on Parallel and Distributed Systems*, 31, 94–110. <https://doi.org/10.1109/TPDS.2019.2928289>