Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп’ютерних наук

Кваліфікаційна робота магістра

з теми:

«Паралельний блочний алгоритм розв’язування систем лінійних алгебраїчних рівнянь з стрічковими матрицями»

Виконав: здобувач вищої освіти
групи KN1-M24
спеціальності 122 Комп’ютерні науки
Ігнатов Максим Васильович

Керівник: Іванюк Віталій Анатолійович,
Завідувач кафедри кафедри комп’ютерних наук

Рецензент: Сидорук Володимир Антонович, кандидат фізико-математичних наук

Кам’янець-Подільський – 2025 р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СЛАР – система лінійних алгебраїчних рівнянь.

A – матриця коефіцієнтів СЛАР.

b – вектор правих частин.

x – вектор невідомих.

n – розмірність матриці (кількість рівнянь у системі).

p – кількість потоків / процесорних ядер для паралельних обчислень.

T_p – час виконання алгоритму на p потоках.

T_1 – час виконання послідовного алгоритму.

S_p – прискорення паралельного алгоритму ($S_p = T_1 / T_p$).

E_p – ефективність паралельного алгоритму ($E_p = S_p / p$).

LU-розвинення – розклад матриці *A* на нижню трикутну матрицю *L* і верхню трикутну матрицю *U*.

MPI – Message Passing Interface, стандарт для паралельних обчислень у кластерних системах.

OpenMP – інструмент для паралелізму на спільній пам’яті.

Θ(n) – асимптотична оцінка складності алгоритму.

.

АНОТАЦІЯ

Тема: Паралельний блочний алгоритм розв’язування систем лінійних алгебраїчних рівнянь зі стрічковими матрицями.

Виконавець: Максим Ігнатов, студент спеціальності «Комп’ютерні науки».

Науковий керівник: Іванюк Віталій Анатолійович, завідувач відділення кафедри комп’ютерних наук

Кваліфікаційна робота присвячена розробленню та дослідженню паралельного блочного алгоритму для ефективного розв’язування систем лінійних алгебраїчних рівнянь (СЛАР), матриці яких мають стрічкову структуру.

У роботі виконано аналіз існуючих методів розв’язування СЛАР зі стрічковими матрицями, розглянуто їхні переваги та обмеження у контексті паралельних обчислень. Реалізовано програмний прототип алгоритму з використанням технології OpenMP, проведено тестування на матрицях різної розмірності та ширини стрічки.

Отримані результати свідчать про суттєве прискорення розв’язування задач при збільшенні кількості потоків. Найвищий виграш у продуктивності досягається для матриць великого розміру та помірної ширини стрічки.

Результати роботи можуть бути використані для оптимізації чисельних методів у системах моделювання фізичних процесів, обробки сигналів, розв’язування задач механіки деформацій, а також у програмному забезпеченні для інженерних обчислень.

Ключові слова: стрічкова матриця, СЛАР, паралельні обчислення, блочний алгоритм, OpenMP, оптимізація, прискорення, чисельні методи.

ABSTRACT

Topic: Parallel Block Algorithm for Solving Systems of Linear Algebraic Equations with Band Matrices.
Author: Maksym Ihnatov, student of the “Computer Science” program.
Supervisor:

This qualification work is devoted to the development and investigation of a parallel block algorithm for the efficient solution of systems of linear algebraic equations (SLAE) whose coefficient matrices have a band structure.

The study includes an analysis of existing methods for solving SLAE with band matrices and examines their advantages and limitations in the context of parallel computing. A software prototype of the algorithm was implemented using OpenMP technology, and testing was carried out on matrices of various sizes and bandwidths.

The obtained results indicate a significant acceleration of computations with an increasing number of threads. The highest performance gain is achieved for large matrices with moderate bandwidth.

The results of this work can be used to optimize numerical methods in systems for modeling physical processes, signal processing, solving problems of deformation mechanics, as well as in engineering computational software.

Keywords: band matrix, SLAE, parallel computing, block algorithm, OpenMP, optimization, speedup, numerical methods.

ВСТУП

Системи лінійних алгебраїчних рівнянь (СЛАР) є невід'ємною складовою математичного моделювання об'єктів і процесів у науці, техніці та інших прикладних сферах, зокрема при дослідженні природних явищ і соціально-економічних систем. Проведення чисельних експериментів відкриває можливість отримувати нові знання про складні процеси та прогнозувати поведінку систем у випадках, коли постановка натурних експериментів є надто витратною, утрудненою або фізично неможливою.

Прагнення забезпечити високу достовірність результатів чисельного моделювання закономірно зумовлює потребу розв'язання СЛАР великих порядків. На сучасному етапі часто мають місце задачі, розмірність матриці яких становить декілька мільйонів або навіть десятки мільйонів невідомих.

Ключовою особливістю СЛАР, що виникають при дискретизації крайових задач або задач на власні значення проекційно-різницевими методами (методом скінченних елементів, методом скінченних різниць тощо), є високий рівень розрідженості матриць. Кількість ненульових елементів у таких матрицях зазвичай становить kn, де $k \ll n$, а n — порядок матриці. Таким чином, переважна більшість подібних матриць є розрідженими.

Структура розрідженості матриці безпосередньо залежить від способу нумерації невідомих і специфіки вихідної задачі та часто має характерну організацію — стрічкову, блочно-діагональну з обрамленням, профільну, а також інші спеціальні формати з локалізованою ненульовою ділянкою. Зокрема, стрічкові матриці є типовими для задач аналізу міцності конструкцій у цивільному та промисловому будівництві, авіа-, ракето- і суднобудуванні, а також для задач фільтрації, масообміну, теплопереносу, масопереносу та моделювання фізичних процесів у суцільних середовищах, зокрема зварюванні та спорідненних технологіях.

На сучасному етапі при математичному моделюванні складних явищ і процесів у різних предметних областях активно застосовуються

обчислювальні системи паралельної архітектури. Особливе місце серед них посідають комп’ютери гібридного типу, що поєднують принципи MIMD (Multiple Instruction – Multiple Data) та SIMD (Single Instruction – Multiple Data) й забезпечують виконання обчислень на багатоядерних центральних процесорах (CPU) з додатковим прискоренням на графічних процесорах (GPU). Такий підхід дає змогу суттєво підвищити продуктивність чисельного моделювання, особливо для задач великої розмірності.

Перехід до гібридних обчислювальних платформ закономірно стимулює розвиток нових ефективних алгоритмів, здатних максимально використовувати переваги спільної роботи MIMD- та SIMD-пристроїв. Дослідженням і розробкою паралельних методів розв’язання систем лінійних алгебраїчних рівнянь, включно з алгоритмами для гібридних комп’ютерів, присвячено значний пласт робіт вітчизняних і зарубіжних учених, серед яких Г. Аллан, Дж. М. Ортега, Дж. Донгарра, Е.Ф. Галба, Дж. Голуб, І.Є. Капорін, І.Н. Коньшин, І.М. Молчанов, М.О. Недашковський, Б.Б. Нестеренко, В.А. Марчук, О.В. Попов, О.М. Хіміч, Т. Фрі́ман, К. Філіпс, М. Квін, Е. Чоу та інші.

Створення гібридних алгоритмів для розв’язання СЛАР на багатопроцесорних CPU–GPU комплексах супроводжується низкою актуальних науково-практичних задач. До них належать: ефективна декомпозиція даних, вибір оптимальної кількості процесів і побудова раціональної топології міжпроцесорних комунікацій, забезпечення балансування обчислювального навантаження, організація та синхронізація обмінів між CPU та GPU, мінімізація комунікаційних витрат, а також повне використання можливостей асинхронного й потокового запуску обчислень для приховування затримок обміну та підвищення масштабованості алгоритму.

Кваліфікаційна робота спрямована на розв’язання зазначених проблем у середовищі обчислювальних систем гібридної архітектури та впровадження створеного алгоритмічного й програмного забезпечення у прикладні галузі

науки і техніки. Основний акцент зроблено на розробці ефективних гібридних алгоритмів, побудованих на методі факторизації несетричних стрічкових матриць.

Розглянуті методи мають універсальний характер і застосовуються не лише для розв’язання систем лінійних алгебраїчних рівнянь (СЛАР), але й для широкого спектра інших задач обчислювальної математики. До таких задач, зокрема, належать: знаходження власних значень і власних векторів, розв’язання нелінійних систем, задач оптимізації, а також інші проблеми, що вимагають високопродуктивних і масштабованих чисельних обчислень.

Запропоновані підходи дозволяють ефективно використовувати спільну роботу багатоядерних процесорів (CPU) та графічних прискорювачів (GPU), забезпечуючи прискорення обчислень, балансування навантаження й підвищення достовірності результатів у великорозмірних математичних моделях.

Об'єкт дослідження: Системи лінійних алгебраїчних рівнянь з стрічковими матрицями та методи їх чисельного розв'язування.

Предмет дослідження: Паралельні алгоритми розв'язування систем лінійних алгебраїчних рівнянь з використанням блочних методів для стрічкових матриць.

Мета дослідження: розробка та дослідження паралельних алгоритмів для дослідження та розв’язування СЛАР зі стрічковими матрицями на комп’ютерах гібридної архітектури, а також апробація алгоритмів при математичному моделюванні у прикладних задачах.

Основні завдання дослідження:

- провести огляд сучасних методів розв’язування СЛАР зі стрічковими матрицями;
- розробити паралельний блочно-циклічний алгоритм розв’язання СЛАР з несеметричними стрічковими матрицями;
- Навести теоретичні оцінки прискорення та ефективності для алгоритму

- Провести апробацію алгоритму при математичному моделюванні процесів зварювання та споріднених технологій.

Тема дослідження: паралельний блочний алгоритм розв’язування систем лінійних алгебраїчних рівнянь (СЛАР) зі стрічковими матрицями.

Отримані в межах дослідження результати можуть бути використані для підвищення продуктивності обчислень у математичному моделюванні, машинному навчанні, комп’ютерній графіці, цифровій обробці даних та інших напрямах, де виникає потреба у розв’язанні великомасштабних розріджених систем лінійних рівнянь із структурованою ненульовою областю.

Запропоновані підходи орієнтовані на задачі великої розмірності та спрямовані на вдосконалення масштабованості і швидкодії алгоритмів, що є критично важливим для сучасних обчислювальних платформ і прикладних інженерно-наукових проєктів.

Методи дослідження: у роботі застосовуються методи теорії матриць, паралельних обчислень.

Практичне значення одержаних результатів

Результати дослідження мають практичну цінність для галузі високопродуктивних обчислень. Розроблений паралельний алгоритм може бути використаний у програмних системах, що потребують швидкого розв’язування великих СЛАР зі стрічковими матрицями, зокрема в задачах моделювання фізичних процесів, обробки сигналів, динаміки механічних систем, чисельної оптимізації та інженерних розрахунків.

Розроблений програмний прототип є готовим до інтеграції в наукові пакети або застосунки, що працюють на багатопотокових процесорах. Алгоритм може бути адаптований до кластерних систем із використанням технології MPI для ще більшого прискорення.

Відомості про впровадження результатів дослідження

Результати роботи можуть бути впроваджені в навчальні курси з паралельних обчислень, чисельних методів та алгоритмів лінійної алгебри в

межах освітніх програм спеціальності «Комп’ютерні науки». Також алгоритм може бути інтегрований у прикладні програмні системи для наукових обчислень. За потреби у роботі може бути додано акт про впровадження, підписаний відповідною організацією або кафедрою.

Апробація результатів

Основні положення роботи були представлені на засіданнях наукового гуртка та підготовлені до участі в студентських науково-практичних конференціях. Результати тестування алгоритму були продемонстровані у вигляді доповіді, яка отримала позитивні відгуки та підтвердила актуальність і практичну значимість розробленого підходу.

Структура роботи

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел.

Загальний обсяг роботи становить 60 сторінок, список використаних джерел містить 41 найменування.

Розділ 1. Огляд літературних джерел та основних проблем

1.1. Аналіз досліджень систем лінійних рівнянь

Значна частина розрахункових математичних задач пов’язана з розв’язуванням і дослідженням систем лінійних алгебраїчних рівнянь (СЛАР) [1-9]. У зв’язку з цим проблема ефективного розв’язування СЛАР з матрицями великих порядків є надзвичайно актуальною.

У практичних застосуваннях часто виникають задачі з розрідженими матрицями, що зумовило значну кількість досліджень, присвячених саме таким системам. Однією з перших монографій, присвячених розрідженим матрицям, є праця Тьюарсона [10], де розглянуто фундаментальні аспекти прямих методів розв’язування систем лінійних рівнянь. У книзі Писанецьки [11] досліджуються різні види алгебраїчних задач з розрідженою структурою матриць: формати зберігання, методи розв’язання СЛАР із симетричними та загального вигляду матрицями, аналіз похибок, застосування до задач методу скінченних елементів..

У монографії Джорджа та Лю [12] проведено ґрунтовні дослідження методів розв’язування СЛАР із симетричними додатно визначеними матрицями великих порядків. Робота Естербю та Златєва [13] присвячена несиметричним розрідженим матрицям. Окреме місце посідає задача оптимізації заповнення розріджених матриць. Відомі алгоритми мінімізації заповнення базуються на апараті теорії графів.

Таким чином, починаючи з другої половини XX століття, сформовано потужну теоретичну базу загальних методів розв’язування СЛАР, у тому числі з розрідженими та структурованими матрицями.

1.2. Паралельні обчислення та гібридні архітектури

Зростання розмірності задач і вимог до точності розв’язків при математичному моделюванні призводить до істотного збільшення обсягів обчислень та оброблюваних даних. Традиційне підвищення продуктивності за рахунок збільшення тактової частоти процесорів вичерпало себе; останнє десятиліття характеризується збільшенням кількості обчислювальних ядер на одному процесорі. У результаті алгоритми з одним потоком команд і даних

(типу SISD за класифікацією Фліна [14]) не завжди забезпечують потрібну продуктивність та повне використання апаратних ресурсів.

Основним шляхом підвищення продуктивності стало використання паралельних обчислювальних систем. Найпотужніші сучасні обчислювальні комплекси є багатопроцесорними системами [15]. Вони широко застосовуються при моделюванні складних процесів у біології, медицині, метеорології, хімії, фармакології, геології, а також у машинобудуванні, будівництві, екології, ядерній енергетиці. Ефективне використання таких систем потребує врахування архітектури, функціональних можливостей та характеристик комунікаційної мережі.

У монографії [5] досліджено паралельні алгоритми розв'язування задач лінійної алгебри на комп'ютерах паралельно-послідовної архітектури. У [17] розглянуто чисельні алгоритми та організацію паралельних обчислювальних систем. У [3] розроблено паралельні алгоритми для задач обчислювальної математики на MIMD-комп'ютерах. Принципи організації паралельних обчислень для розв'язування СЛАР на векторних та паралельних комп'ютерах подано в [6].

Подальший розвиток пов'язаний із використанням графічних процесорів (GPU), продуктивність яких істотно зросла за останні десятиріччя. Для роботи з GPU створено спеціальні технології програмування та програмні засоби. Виробниками розроблено бібліотеки cuBLAS, cuSparse, cuRAND, cuFFT тощо, які дозволяють значно скоротити час виконання обчислень. Це зумовлює актуальність побудови ефективних паралельних алгоритмів для гібридних систем, що поєднують багатоядерні процесори MIMD-архітектури з графічними процесорами SIMD-архітектури.

Незважаючи на значний прогрес, не існує універсального паралельного методу, здатного ефективно розв'язувати задачі з матрицями довільної структури. У ряді випадків ефективність розпаралелювання досягається лише на обмеженій кількості процесорів, що додатково мотивує розробку нових спеціалізованих підходів.

1.3. Прикладне програмне забезпечення та бібліотеки

Розвиток прикладного програмного забезпечення для розв’язування СЛАР та задач лінійної алгебри розпочався разом із появою перших комп’ютерів. Важливим етапом стали алгол-процедури, розроблені Уілкінсоном та Райншем [18], які поклали початок створенню математичного забезпечення лінійної алгебри і увійшли до складу багатьох бібліотек прикладних програм.

Зростання розмірності задач та вимог до точності посилило потребу у паралельних програмних засобах для розв’язування систем рівнянь. Більшість із них орієнтована на певні класи задач або структури матриць. У пакетах BlockSolver95 [19] та SuperLU [20] реалізовано прямі методи розв’язування СЛАР. У бібліотеці PSparslib [21] реалізовано ітераційні підходи.

Для матриць розрідженої структури створено низку як прямих (MUMPS [22], PSPASES [23]), так і ітераційних (PSparslib [21], SPARSKIT [24]) методів.

Методи розв’язування СЛАР широко застосовуються в електроніці, будівництві, машинобудуванні та інших прикладних галузях. Кожна галузь висуває специфічні вимоги до структури задач, типу матриць і властивостей розв’язків. Це призводить до того, що, попри наявність великої кількості універсальних бібліотек, зберігається потреба у розробці спеціалізованого програмного забезпечення, орієнтованого на конкретні класи задач (наприклад, СЛАР зі стрічковими або профільними матрицями, задачі міцнісного аналізу конструкцій тощо).

1.4. Проблема достовірності результатів обчислень

Більшість існуючих програмних засобів для розв’язування СЛАР створювалися з припущенням точності вихідних даних. Однак у реальних прикладних задачах математичні моделі містять наближені дані, а результати обчислень зазнають впливу похибок, пов’язаних з обмеженою розрядністю представлення чисел та заокругленнями під час виконання арифметичних операцій.

Питання наближеного характеру даних, похибок обчислень та властивостей машинних моделей детально розглянуто у монографіях [25–30]. Важливою задачею є оцінювання достовірності машинних розв’язків. У деяких програмних комплексах передбачено обчислення оцінки похибки [31] або надано рекомендації щодо її оцінювання [32, 33]. Водночас у більшості програмних засобів відсутні механізми аналізу похибок, зумовлених наближеним характером вхідних даних.

Це зумовлює актуальність розробки алгоритмів, які дозволяють оцінювати достовірність комп’ютерних розв’язків СЛАР, враховуючи як похибки обчислень, так і невизначеність заданих даних. Поєднання таких підходів із паралельними та гібридними алгоритмами розв’язування СЛАР є важливим напрямом подальших досліджень.

1.5. Проблеми розробки паралельних алгоритмів для СЛАР з розрідженими матрицями

Сучасні науково-технічні задачі характеризуються великими обсягами вхідних даних та високою обчислювальною складністю. Подальше підвищення тактової частоти процесорів досягло практичної межі, тому основним шляхом збільшення продуктивності обчислень стало використання паралельних та гібридних комп’ютерних систем [5]. Останні роки супроводжуються також інтенсивним зростанням продуктивності графічних процесорів (GPU) та їх широким застосуванням у науково-технічних обчисленнях.

У роботі розглядаються гібридні обчислювальні системи архітектури MIMD+SIMD (за класифікацією Фліна [14]), які поєднують багатоядерні центральні процесори та графічні прискорювачі. Побудова ефективних масштабованих алгоритмів для таких систем потребує врахування:

- обмеженої кількості процесів і потоків;
- часу обміну даними між CPU та GPU;
- накладних витрат на синхронізацію;
- обмеженого обсягу оперативної та кеш-пам’яті.

У зв’язку з цим до паралельних програм висуваються ключові вимоги [4]:

- **паралелізм** – здатність алгоритму ефективно використовувати множину процесів;
- **масштабованість** – збереження прийнятної ефективності при зміні кількості процесів;
- **локальність** – переважання звернень до локальних даних над зверненнями до віддалених.

Одним із визначальних факторів ефективності є схема декомпозиції даних. Простий блочний розподіл (розбиття матриці на ppp суцільних блоків за кількістю процесів) часто призводить до ефекту «випадання» процесів з обчислень (ефект Гайдна). Рядково-циклічний розподіл усуває цей недолік, але потребує частих синхронізацій. Найбільш поширеним на сучасних системах є двовимірний блочно-циклічний розподіл, за якого матриця розбивається на відносно невеликі блоки й розподіляється між процесами в циклічному порядку. Це дозволяє:

- краще балансувати навантаження;
- організувати обчислення так, щоб обробка кожного блоку відбувалася переважно в межах кеш-пам’яті;
- зменшити обсяги міжпроцесорних обмінів.

Ефективність паралельного алгоритму для p процесів характеризується коефіцієнтом прискорення S_p та ефективності E_p [5]:

$$S_p=T_1/T_p,\ E_p=S_p/p,$$

де T_p – час розв’язування задачі на гібридному комп’ютері з використанням p процесів, а T_1 – час розв’язування тієї ж задачі на гіпотетичному послідовному комп’ютері зі швидкодією одного процесора та обсягом пам’яті, рівним сумарній пам’яті p процесів. Ці величини залежать від середнього часу виконання арифметичних операцій, затрат на обмін даними та встановлення комунікаційних зв’язків.

Важливу роль відіграє й організація обміну даними між процесами. Використовуються як обміни типу «точка–точка», так і колективні операції (мультирозсилка, мультизбірка числа, мультизбірка вектора), що дозволяють оптимізувати обмін у групі процесів.

Для СЛАР з розрідженими матрицями до проблеми розпаралелювання додається вибір формату зберігання даних та оптимізація заповнення. Оскільки обчислення виконуються лише над ненульовими елементами, доцільно використовувати спеціалізовані формати:

- **рядковий розріджений формат (CSR)** – зберігання ненульових елементів в масиві значень та двох масивах індексів; забезпечує низькі вимоги до пам’яті та зручність реалізації операцій типу «матриця–вектор»;
- **стрічковий (band) формат** – зберігання лише елементів в межах кількох діагоналей у двовимірному масиві, що особливо ефективно для стрічкових матриць, дозволяє виконувати основні операції без індексної надбудови.

Розв’язування розріджених систем прямими методами супроводжується появою нових ненульових елементів (заповнення). Для його зменшення використовуються алгоритми переупорядкування, зокрема:

- **алгоритм мінімальної степені**, орієнтований на зменшення заповнення при факторизації симетричних додатно визначених матриць;
- **алгоритм Катхіла–Макі (та його обернений варіант)**, який перетворює портрет симетричної матриці до наближеного стрічкового вигляду та зменшує ширину стрічки і профіль матриці.

У паралельному випадку критерії оптимальності упорядкування доповнюються вимогами до мінімізації міжпроцесорних обмінів і рівномірного завантаження процесів. Отже, для побудови ефективних паралельних алгоритмів розв’язування СЛАР з розрідженими матрицями необхідно поєднувати:

- архітектурно-орієнтований вибір схеми декомпозиції;
- адекватний формат зберігання розріджених даних;
- використання алгоритмів переупорядкування, що зменшують заповнення, обчислювальні витрати та обсяги комунікацій.

Саме на цій основі розробляється та досліджується паралельний алгоритм для СЛАР зі стрічковими матрицями на гібридних обчислювальних системах.

Розділ 2. Постановка задачі та розробка паралельного блочного алгоритму

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 мікропошкодженості (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 спричиняти 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 [34, 35] дослідженнях 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 несуцільності 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 несуцільності 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

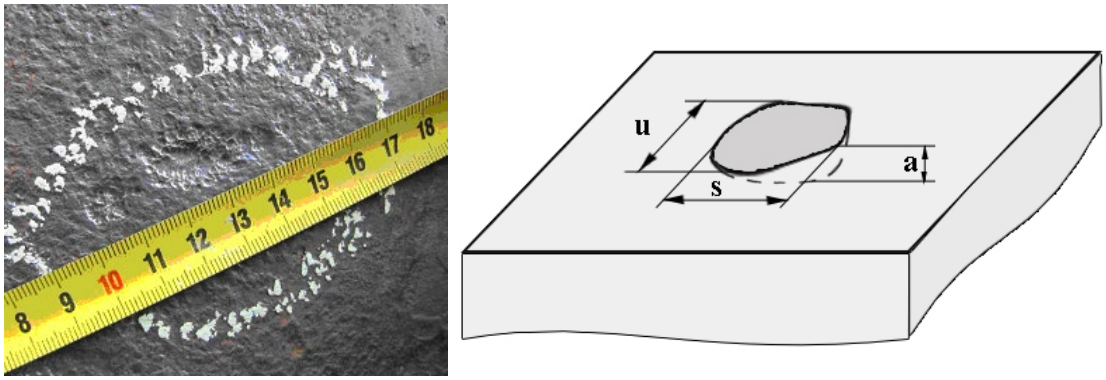
граничного стану визначається зародженням та розвитком розсіяних і локальних пошкоджень матеріалу аж до макроскопічного руйнування, специфіку і природу якого необхідно враховувати при аналізі конструкцій відповідного технологічного рішення та при конкретних умовах експлуатації.

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 [3, 4]. 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 зварювання і циклічному навантаженні, що дозволить отримати повну систему взаємопов’язаних фізико-механічних процесів, що мають місце в металі трубопроводу.

2.1. 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. 1), загальна корозія розглядається як частина конструкції з меншою товщиною стінки.



(а) (б)

Рис. 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 [36], 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.

2.2. Чисельний аналіз напружено-деформованого стану відповідальних зварних конструкцій з виявленими дефектами

У загальному випадку, механізм в’язкого руйнування може бути представлений наступними послідовними етапами [37]:

а) зародження пор в’язкого руйнування при виготовленні конструкції, у тому числі, в зоні локального зварювального нагрівання і при розвиненому пластичному плині 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 нестаціонарного рівняння теплопровідності, в багатовимірному випадку у вигляді:

$$c\gamma(T)\frac{\partial T}{\partial t} = \nabla\left[\lambda(T)\cdot\nabla T\right] \qquad (1)$$

де $c\gamma$, λ – об'ємна теплоємність і теплопровідність металу, відповідно; T – температура конструкції у момент часу t в точці з координатами (r, β, z), відповідно до схеми, яку представлено на рис. 2.

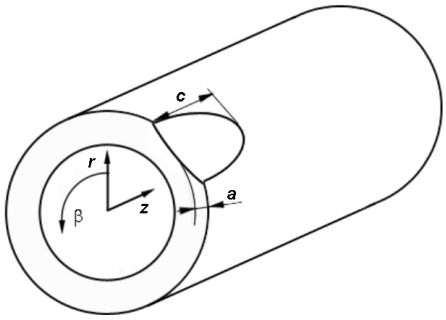


Рис. 2. Схема ділянки трубопроводу із зовнішнім напівеліптичним дефектом локального стоншення стінки

Для розв’язку рівняння (1) і адекватного врахування особливостей технологічного процесу зварювання необхідні граничні умови, які для даного випадку мають наступний вигляд:

$$-\lambda(T)\frac{\partial T}{\partial n} = \alpha_T(T - T_0) + \varepsilon_0\sigma_{SB}(T^4 - T_0^4) - q\,, \qquad (2)$$

де n – нормаль до поверхні конструкції; α_T – коефіцієнт поверхневої тепловіддачі; ε_0 – ступінь чорноти поверхні трубопроводу; σ_{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 (1)-(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 ε_{ij} ($i, j = r, \beta, z$) в даному випадку представляються у вигляді суперпозиції наступних компонент:

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

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 (3) фігурує об'ємна концентрація рівномірно розподілених пор f, наявність якої відрізняє розроблені моделі від класичних методів аналізу термодеформованого стану суцільного середовища. Наявність ДМ не тільки має адитивну складову в тензорі деформацій, але і змінює поверхню текучості матеріалу Ф, для математичного опису якої знайшли широке застосування підходи Гурсона, Твергаарда і Нідлмана (так звана ГТН модель):

$$\Phi = \left(\sigma_i / \sigma_T\right)^2 - \left(q_3 f^*\right)^2 + 2q_1 f^* \cosh\left(q_2 \frac{3\sigma_m}{2\sigma_T}\right) - 1, \qquad (4)$$

де $q_1 = 1{,}5$, $q_2 = 1$, $q_3 = 1{,}5$ – константи, f^* – еквівалентна концентрація пор, σ_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), що враховує взаємодію між окремими несуцільностями, оцінюється на основі наступних співвідношень:

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

де f_c – критична концентрація несуцільностей, 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), 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 співвідношенню:

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

де $c_1 = 2{,}149\times10^{-3}; c_2 = 9{,}112\times10^{-2}$; $\varepsilon_0 = 1{,}540\times10^{-4}$, $m = 0{,}14$ – 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:

$$\chi_к = \int \frac{d\varepsilon_i^p}{\varepsilon_c(T)} > 1, \qquad (5)$$

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 ε_cв (5) можна обчислити за наступним співвідношенням:

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

де d_1, d_2, d_3, B_f, ξ – константи.

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

$$df = k_{ms} f_0 K_1 \exp(K_2 \sigma_m / \sigma_i) d\varepsilon_i^p, \qquad (6)$$

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. 3). У рамках об'єму, обмеженого розглянутим елементом, розподіл температур, напруг і деформацій приймається як однорідний.

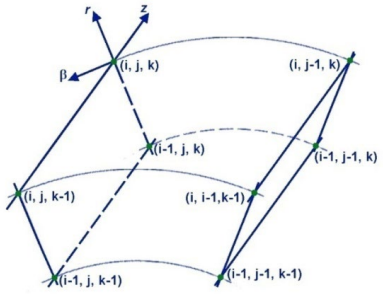


Рис. 3. Схема скінченного елемента в системі координат r, β, z.

Скінченно-елементне розв'язування крайової задачі нестаціонарної 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). Так, для вирішення нелінійності по пластичній деформації було використано запропонований В.І. Махненко [36] підхід, а саме розгляд функції стану матеріалу Ψ, що задовольняє наступним умовам на поверхні текучості:

$$\begin{array}{ll} \Psi = \dfrac{1}{2G}, & \text{якщо} \quad \sigma_i < \sigma_s \\ \Psi > \dfrac{1}{2G}, & \text{якщо} \quad \sigma_i = \sigma_s \end{array}, \qquad (7)$$

де $\sigma_s = \sigma_T \sqrt{1 + (q_3 f^*)^2 - 2q_1 f^* \cosh(1{,}5 q_2 \sigma_m / \sigma_T)}$, $G = 0{,}5E/(1+\nu)$, E – модуль Юнга, ν – 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), що спричиняє зміну петлі пластичної деформації. Але при цьому на кожному етапі навантаження необхідно визначити рівноважний стан пошкодженості і відповідний йому розподіл напруг і деформацій. Для цього у припущенні, що стаціонарний стан характеризується нехтуючи малою швидкістю зростання ДМ за (6), пропонується проводити такий ітераційний процес по функції $\Psi_K = f_0 K_1 \exp(K_2 \sigma_m / \sigma_i) d\varepsilon_i^p$:

$$F = \begin{cases} F + dF, & \text{якщо} \quad \Psi_K \le \Psi_K^0 \approx 10^{-5}, \\ F, & \text{якщо} \quad \Psi_K > \Psi_K^0, \end{cases} \tag{8}$$

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

Таким чином, розв’язування задач (7), (8) на кожному етапі дослідження дозволяє з контрольованою мірою точності визначати ступінь пошкодженості 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:

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

або

$$f^* \to f_d^* = 2(q_1/q_3) \cosh(1{,}5 q_2 \sigma_m / \sigma_T) \tag{9}$$

або

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

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

У випадку, якщо для деякого СЕ виконується одна з умов (9), вважається, що даний СЕ втрачає свою несучу здатність і на його місці сформувалася макроскопічна несуцільність. Подальше навантаження конструкції і розвиток макроруйнування, у результаті призводить до лавиноподібної втрати несучої здатності матеріалу в межах ітераційного процесу (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 макронесуцільності, або її (конструкції) спонтанне руйнування.

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

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

де δ_{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 Ψ напруги σ_{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 ΔU_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 статики для внутрішніх СЕ та граничним умовам – для поверхневих. У свою чергу, компоненти вектора ΔU_i задовольняють відповідним умовам на границі.

На кожному кроці дослідження та ітерацій по Ψ необхідно розв'язувати систему рівнянь в змінних вектора приросту переміщень ΔU_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 рівнянь

$$\frac{\partial E_I}{\partial \Delta U_{m,n,r}} = 0, \quad \frac{\partial E_I}{\partial \Delta V_{m,n,r}} = 0, \quad \frac{\partial E_I}{\partial \Delta W_{m,n,r}} = 0. \quad (10)$$

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

2.3. Блочний циклічний алгоритм LU-розвинення стрічкової матриці.

Аналіз таких технологічних схем та проведене дослідження комп'ютерних програм, які їх реалізують, свідчать, що левова частка часу витрачається на розв'язування СЛАР (10) з несиметричними стрічковими матрицями. Кількість таких СЛАР, як зазначалось вище, досягає декількох сотень, а то і тисяч. Тому доцільно для розв'язування цих СЛАР використати комп'ютери з паралельною організацією обчислень, в тому числі гібридної архітектури, та відповідні алгоритмічно-програмні засоби [37-40].

Отже, розглянемо алгоритм LU-розвинення стрічкової матриці СЛАР (10) A порядку n з m_l піддіагоналями та m_u наддіагоналями на комп'ютерах з багатоядерними процесорами.

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 $n = Ns$, $m_l = M_L s$, $m_l + m_u = M_U s$):

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

Перейдемо до опису алгоритму. Для кожного $I = 1, \ldots, N$ виконуються наступні кроки:

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

$$\begin{pmatrix} A_{I,I} \\ \vdots \\ A_{K,I} \end{pmatrix} = P_I \begin{pmatrix} L_{I,I} \\ \vdots \\ L_{K,I} \end{pmatrix} U_{I,I}. \qquad (11)$$

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

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

$$L_{I,I}\begin{pmatrix}U_{I,I+1} & \cdots & U_{I,J}\end{pmatrix}=\begin{pmatrix}A_{I,I+1} & \cdots & A_{I,J}\end{pmatrix}. \qquad (12)$$

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

$$A_{M,L} \leftarrow A_{M,L} - L_{M,I}\, U_{I,L}\ . \qquad (13)$$

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 $L_{I+1,I}, L_{I+2,I}, \ldots, L_{K,I}$ матриці L, як розв'язок матричної СЛАР з нижньою трикутною матрицею:

$$U_{I,I}^T\begin{pmatrix}L_{I+1,I}^T & \cdots & L_{K,I}^T\end{pmatrix}=\begin{pmatrix}A_{I+1,I}^T & \cdots & A_{K,I}^T\end{pmatrix}.$$

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 (11)-(13) показав, що для реалізації більшості обчислень можна використовувати програмні модулі для матрично-матричних операцій від розробників технічних засобів.

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 (13) на даному кроці.

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.

2.4. Оцінки кількості операцій

Проаналізуємо кількість операцій як у **скалярних термінах** (через *ml*, *mu*), так і у **блочно-плиткових** (через *s, ML, MU*).

1. Скалярна оцінка: LU-факторизація стрічкової матриці

На рівні окремих елементів (без блокування) на кожному кроці класичного методу Гауса для стрічкової матриці:

- кількість елементів під головним елементом у стовпчику — не більше ніж *ml*;
- кількість елементів праворуч у рядку — не більше ніж mu.

На кроці *k*:

- **обчислення множників** (ділення) — не більше ніж *ml* FLOPs;
- **оновлення підматриці**: для кожного з ml рядків і mu стовпців виконується операція виду

$a_{ij} := a_{ij} - l_{ik} u_{kj}$ тобто 2 FLOPs на пару (i,j).

Усього — не більше ніж $2ml \cdot mu$ FLOPs.

Отже, на один крок:

$$F_{step(k)} \lesssim ml + 2ml \cdot mu.$$

Всього кроків приблизно n, тому асимптотична оцінка:

$$F_{fact} \approx nml + 2nml \cdot mu = nml(1 + 2mu).$$

Якщо матриця приблизно «симетрично стрічкова» ($ml \approx mu = b$), то

$$F_{fact} \approx 2nb^2 + nb,$$

тобто

$$F_{fact} = \Theta(n(ml + mu)^2),$$

Операції прямого й зворотного ходу (розв'язування $Ly = Pb$ та $Ux = y$) мають вартість порядку

$$F_{solve} = O(nM),$$

і для великих *M* (широких стрічок) є набагато дешевшими, ніж факторизація.

2. Блочна оцінка: через s,ML,MU

Повернімося до блочної структури, де:

- $N=n/s$— кількість блоків по діагоналі;
- ML — кількість блоків-піддіагоналей;
- MU — кількість блоків у повній блок-стрічці $(\approx(ml+mu)/s)$.

Розглянемо FLOPs на одному кроці I, а потім помножимо на N.

Крок 1. Панельна LU-факторизація

Факторизується прямокутна матриця розміру $(ML+1)s\times s$. Відомо, що вартість LU-факторизації густої матриці розміру $m\times s$ з частковим pivoting оцінюється як

$$F_{panel}(I)=\Theta(MLs^3).$$

Для всіх N кроків:

$$F_{panel}\approx N\cdot F_{panel}(I)=\Theta(NMLs^3)=\Theta(nMLs^2).$$

Якщо підставити $ML\approx ml/s$, маємо:

$$F_{panel}=\Theta\,(n(ml+s)),$$

тобто лінійну залежність від n та майже лінійну від ml.

Крок 2. TRSM для рядка блоків U_I

На кроці I TRSM виконується для кожного блоку $L=I+1,\dots,J$, тобто приблизно для MU правих блоків. Розв'язання $L_{I,I}U_{I,L}=A_{I,L}$ для одного блоку вартує приблизно s^3 FLOPs. Отже:

$$F_{TRSM}(I)\approx c_{trsm}MUs^3,$$

де c_{trsm} — константа (приблизно 1).

Для всіх кроків:

$$F_{TRSM}\approx N\cdot F_{TRSM}(I)=\Theta(NMUs^3)=\Theta(nMUs^2).$$

Приблизно, якщо $MU\sim(ml+mu)/s$, то

$$F_{TRSM}=\Theta\,(n(ml+mu)s).$$

Крок 3. Оновлення підматриці (GEMM)

Для кожної пари індексів $M=I+1,\dots,K$ і $L=I+1,\dots,J$ виконується оновлення:

$$A_{M,L}\leftarrow A_{M,L}-L_{M,I}U_{I,L},$$

що є операцією матричного множення $s \times s$.

Вартість одного GEMM — приблизно $2s^3$ FLOPs.

Кількість пар (M,L) на кроці:

$$\#\{(M,L)\} \approx ML \cdot MU.$$

Тоді

$$F_{upd}(I) \approx 2MLMUs^3.$$

Для всіх кроків:

$$F_{upd} \approx N \cdot F_{upd}(I) = 2NMLMUs^3 = 2nMLMUs^2.$$

Переходячи до скалярних параметрів $ml \approx MLs$, $ml+mu \approx MUs$, маємо:

$$MLMUs^2 \approx ml(ml+mu)s,$$

тобто за порядком

$$F_{upd} = \Theta(n\, ml(ml+mu)).$$

У симетричному випадку $ml \approx mu = b$:

$$F_{upd} = \Theta(nb^2),$$

і це домінуюча частина факторизації, узгоджена з оцінкою зі скалярної моделі.

Сумарно:

$$F_{total} = F_{panel} + F_{TRSM} + F_{upd}.$$

За порядком:

$$F_{panel} = O(nMLs^2), F_{TRSM} = O(nMUs^2), F_{upd} = O(nMLMUs^2).$$

Для реальних задач, де $ML, MU \gg 1$, домінує саме компонент

$$F_{upd} = O(nMLMUs^2) \sim O(n(ml+mu)^2)$$

тобто:

$$F_{total} = \Theta\,(n(ml+mu)^2)$$

— як і для класичного скалярного алгоритму, але з принциповою відмінністю в структурі обчислень: більша частина FLOPs тепер приходиться на операції рівня BLAS-3 (GEMM, TRSM), що:

- суттєво підвищує ефективність використання кеш-пам’яті;
- дозволяє досягати високої продуктивності на багатоядерних CPU;

- робить алгоритм придатним до подальшого перенесення на GPU та гібридні системи.

Теорема. Загальна кількість арифметичних операцій, які **можуть** виконуватися паралельно**,** має асимптотичну оцінку:

$$F_{parallel} = \Theta\left(\frac{nml(ml+mu)}{p}\right)$$

Доведення. На *I*-му кроці LU-алгоритму виконуються такі операції:

Панельна LU-факторизація блок-стовпчика

$$F_{panel}(I) = \Theta(MLs^3).$$

Обчислення рядка блоків матриці U (TRSM):

$$F_{trsm}(I) = \Theta(MUs^3)$$

Рангове оновлення хвостової підматриці:

$$A_{M,L} \leftarrow A_{M,L} - L_{M,I}U_{I,L},$$

де

$$M = I+1,\ldots,K,\ \ L = I+1,\ldots J,\ \ J = \min(I+MU, N)$$

Кількість таких блокових оновлень дорівнює:

$$MLMU$$

і кожне з них є матрично-матричним множенням складності

$$\Theta\left(s^3\right)$$

Отже, обчислювально домінуючий внесок одного кроку:

$$F_{update}(I) = \Theta\left(MLMUs^3\right)$$

З урахуванням $N = n/s$ кроків алгоритму отримуємо:

$$F_{total} = \Theta\left(NMLMUs^3\right) = \Theta\left(\frac{n}{s}\cdot\frac{ml}{s}\cdot\frac{ml+mu}{s}\cdot s^3\right)$$

Після скорочення:

$$F_{total} = \Theta\left(nml(ml+mu)\right)$$

Рангові оновлення блоків $A_{M,L}$ мають такі властивості:

- кожен блок (M,L) оновлюється **незалежно**;
- між операціями оновлення **відсутні залежності за даними**;

- оновлення можуть бути розподілені між процесами циклічно.

Таким чином, на кожному кроці можливе паралельне виконання до:

$$\min(p, MLMU)$$

незалежних блокових операцій.

При рівномірному розподілі блоків між процесами кожен процес виконує:

$$\Theta\left(\frac{MLMUs^3}{p}\right)$$

операцій на одному кроці.

З урахуванням усіх N кроків:

$$F_{parallel} = \Theta\left(\frac{NMLMUs^3}{p}\right) = \Theta\left(\frac{ml(ml+mu)}{p}\right)$$

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 стрічкової структури, що є фундаментальною властивістю класу задач.

2.5 Оцінки прискорення та ефективності алгоритму

Теорема. Прискорення алгоритму з урахуванням комунікацій має асимптотичну оцінку:

$$S_p \approx \Theta\left(\frac{p}{1 + \frac{p^2}{\gamma ml(ml+mu)} + \frac{p^2\beta(ml+mu)}{\gamma ml}}\right)$$

де:

α — латентність комунікацій;

β — час передавання одного слова;

γ — час виконання однієї арифметичної операції.

Доведення. За означенням:

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

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

З попередніх результатів:

$$T_1 = \gamma mml(ml + mu),\ T_p = \gamma \frac{nml(ml + mu)}{p} + p\frac{n}{s}\alpha + p\beta n(ml + mu)$$

Підставляємо T_1 і T_p у визначення S_p:

$$S_p = \frac{\gamma mml(ml + mu)}{\gamma \frac{nml(ml + mu)}{p} + p\frac{n}{s}\alpha + p\beta n(ml + mu)}$$

Виносимо спільний множник γm:

$$S_p = \frac{ml(ml + mu)}{\frac{ml(ml + mu)}{p} + \frac{p\alpha}{\gamma s} + \frac{p\beta(ml + mu)}{\gamma}}$$

Домноживши чисельник і знаменник на p, отримуємо:

$$S_p = \frac{pml(ml + mu)}{ml(ml + mu) + \frac{p^2\alpha}{\gamma s} + \frac{p^2\beta(ml + mu)}{\gamma}}$$

Переходячи до асимптотичної форми, дістаємо:

$$S_p \approx \Theta\left(\frac{p}{1 + \frac{p^2}{\gamma ml(ml + mu)} + \frac{p^2\beta(ml + mu)}{\gamma ml}}\right).$$

Отримана оцінка прискорення показує, що блочно-циклічний LU-алгоритм для стрічкових матриць забезпечує майже лінійне масштабування 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.

У наступному розділі буде розглянуто особливості програмної реалізації алгоритму та наведено результати апробації на обчислювальному вузлі кластера СКІТ [41].

Розділ 3. Реалізація та експериментальне дослідження

3.1. Вибір мови програмування

Вибір мови програмування C++ для проведення високопродуктивних обчислень (HPC) є ключовим фактором у розробці паралельних алгоритмів, зокрема блочної LU-факторизації стрічкових матриць. Високопродуктивні обчислення вимагають поєднання максимальної швидкодії, низькорівневого контролю над пам’яттю, можливості реалізувати масовий паралелізм і повної сумісності з апаратними та програмними засобами паралельного виконання, такими як OpenMP, CUDA, MPI та оптимізовані бібліотеки BLAS/LAPACK. Усі ці вимоги C++ задовольняє найбільш повно і тому є стандартом де-факто в сучасній HPC-індустрії.

Однією з головних причин вибору C++ є здатність генерувати високоефективний машинний код, який оптимізує виконання обчислювально інтенсивних ядер, таких як TRSM та GEMM. Завдяки роботі сучасних компіляторів (GCC, Clang, Intel oneAPI, NVCC) забезпечується автоматична векторизація, оптимізація циклів, ефективне розміщення даних у регістрах і використання інструкцій AVX/AVX-512. Це дає змогу досягти пікової продуктивності багатоядерних процесорів і графічних прискорювачів.

Крім продуктивності, C++ надає гнучкий і детальний контроль над пам’яттю, який є критичним для роботи з нестандартними структурами даних, такими як стрічкові матриці (band-packed) або блочні плитки (tile-based). Можливість явно керувати виділенням, вирівнюванням і розташуванням масивів у пам’яті дозволяє уникнути прихованих витрат продуктивності та реалізувати ефективні формати збереження, що суттєво впливають на швидкість LU-факторизації. Це також забезпечує гнучкість при оптимізації доступу до пам’яті на GPU, де коалесований доступ і використання shared memory визначають ефективність алгоритму.

C++ є універсальною платформою для реалізації різних рівнів паралелізму. На рівні CPU використовується OpenMP, що дозволяє ефективно масштабувати блочні операції TRSM і GEMM. На рівні GPU C++

є базовою мовою для CUDA, забезпечуючи написання власних GPU-ядер і використання високопродуктивних бібліотек, таких як cuBLAS і cuSOLVER. Для кластерного паралелізму C++ повністю сумісний із MPI, що робить можливим реалізацію багатовузлових алгоритмів з розподілом блочних або стрічкових матриць між вузлами кластера. Завдяки цьому C++ дозволяє поєднати OpenMP, CUDA та MPI в межах одного програмного проєкту, що є необхідним для сучасних HPC-додатків.

Важливим аргументом на користь C++ є також сумісність із широким набором високопродуктивних бібліотек. Усі ключові бібліотеки лінійної алгебри — Intel MKL, OpenBLAS, BLIS, LAPACK, ScaLAPACK, cuBLAS, cuSOLVER, MAGMA — орієнтовані саме на C/C++-інтерфейс. Це дає можливість нативно викликати оптимізовані рутинні операції, мінімізуючи накладні витрати та забезпечуючи найвищу можливу швидкодію. Інші мови часто використовують ці бібліотеки через обгортки, що створює додаткові затримки та ускладнює інтеграцію.

3.2. Особливості реалізації алгоритму в OpenMP для блочної LU-факторизації стрічкових матриць

Реалізація блочної LU-факторизації стрічкових матриць у середовищі OpenMP має низку особливостей, пов’язаних зі структурою даних, залежностями між елементами алгоритму та обмеженою областю паралелізму, що визначається шириною стрічки. Оскільки стрічкова матриця містить ненульові елементи лише у діапазоні, що обмежений нижньою та верхньою напівширинами, структура обчислень суттєво відрізняється від класичної LU-факторизації повної матриці. Це впливає на організацію паралельного виконання, розподіл даних між потоками, оптимізацію доступу до пам’яті, а також ефективність масштабування на багатоядерному процесорі.

У блочному алгоритмі факторизація кожного кроку складається з трьох основних частин: панельної факторизації діагонального блока, виконання трикутного розв’язання для блоків поточного рядка та оновлення блоків

хвостової підматриці. У контексті OpenMP панельна частина фактично залишається майже послідовною. Це пов’язано з тим, що операція LU-факторизації одного блоку містить внутрішні залежності між рядками та стовпцями, а також потребує коректного визначення опорних елементів (pivoting). У результаті навіть при застосуванні оптимізованих бібліотек (наприклад, MKL) частка паралельного виконання панелі залишається низькою, що утворює «послідовне ядро» алгоритму та обмежує максимальний можливий прискорюючий ефект згідно із законом Амдала.

Натомість найбільший потенціал паралелізації в OpenMP має етап оновлення хвостової підматриці. На цьому етапі виконуються обчислення типу матричного множення для блоків, які розміщені в межах стрічки, але не належать поточній панелі. Оскільки кожне оновлення блоку є незалежним від інших (за винятком даних, які були щойно отримані на етапі TRSM), ці операції можуть виконуватися паралельно між потоками. Типовою реалізацією є організація циклу за індексами блоків хвостової матриці з використанням директиви #pragma omp parallel for, що забезпечує розподіл блоків між наявними ядрами процесора. Оскільки операція GEMM має високу обчислювальну інтенсивність і добре векторизується, саме вона визначає загальну ефективність OpenMP-реалізації.

При цьому важливо враховувати, що кількість блоків, які потребують оновлення на кожній ітерації, прямо залежить від ширини стрічки. У випадку вузької стрічкової матриці кількість доступних для паралелізації блоків є невеликою, і, відповідно, багатоядерний процесор може бути недовантажений. Якщо ж напівширини стрічки набагато перевищують розмір блоку, загальна кількість незалежних оновлень суттєво збільшується, що дає змогу повністю використати всі доступні ядра процесора. Таким чином, структура стрічки визначає масштабованість алгоритму в OpenMP.

Ще однією особливістю є залежність ефективності від використання багатопотокових BLAS-бібліотек. Оскільки більшість обчислень виконується через виклики dgemm і dtrsm, оптимальною є комбінація OpenMP як

механізму розподілу блоків між потоками та внутрішньої багатопотоковості MKL для прискорення матричних операцій. Однак важливо уникати ситуації, коли одночасно активується паралелізм на рівні OpenMP та паралелізм усередині BLAS-функцій, що може призвести до надмірного перемикання потоків і зниження продуктивності. Тому зазвичай рекомендується або паралелізувати цикл OpenMP і виконувати BLAS-функції в однопотоковому режимі, або, навпаки, викликати BLAS у багатопотоковому режимі, але не дублювати паралелізацію на рівні зовнішніх циклів.

Також значну роль відіграє оптимізація доступу до пам’яті. Стрічкова структура матриці природно підвищує локальність даних, оскільки блоки зберігаються компактно в межах band-packed формату. Завдяки цьому кешування працює ефективніше, а операції над блоками мають менший оверхед звернення до пам’яті. Проте необхідно правильно організувати розміщення блоків у пам’яті, щоб уникнути фрагментації й забезпечити оптимальне передавання даних у BLAS-функціях. У практичній OpenMP-реалізації це означає використання вирівняних буферів та уникнення надлишкових копіювань.

Загалом ефективність OpenMP-реалізації визначається балансом між послідовною панельною частиною та паралельною частиною GEMM/TRSM. При достатній ширині стрічки та правильно налаштованій багатопотоковості OpenMP забезпечує суттєве прискорення виконання алгоритму, хоча й поступається CUDA за піковою продуктивністю через меншу кількість ядер і обмеження пропускної здатності пам’яті. У той же час OpenMP є значно простішим у реалізації, не потребує перенесення даних між CPU і GPU та демонструє стабільну продуктивність навіть для матриць невеликої та середньої розмірності, де GPU-обчислення часто є недоцільними. Завдяки цьому OpenMP-реалізація становить важливу та ефективну частину програмного комплексу для LU-факторизації стрічкових матриць.

3.3. Особливості реалізації алгоритму в CUDA для блочної LU-факторизації стрічкових матриць

Реалізація блочної LU-факторизації стрічкових матриць у середовищі CUDA має свої принципові особливості, зумовлені відмінностями архітектури графічних процесорів та специфічною структурою стрічкових матриць. Алгоритм повинен бути адаптований до масового паралелізму GPU, вузької та предиктивної моделі доступу до пам’яті, особливостей роботи з блоками, а також потреби у високій пропускній здатності при виконанні матричних операцій. Оскільки стрічкова матриця зберігає ненульові елементи лише в межах вузької ділянки навколо головної діагоналі, архітектура GPU може бути використана ефективно лише тоді, коли ця структура допускає достатню кількість незалежних блочних операцій. На відміну від OpenMP, де паралелізм реалізується через розподіл незалежних циклів між ядрами, CUDA передбачає масове, одночасне виконання сотень і тисяч нитей, що накладає зовсім інші вимоги на організацію даних та обчислень.

Однією з центральних особливостей CUDA-реалізації є те, що основна частина продуктивності досягається за рахунок виконання операцій блочного множення матриць (GEMM) та трикутних розв’язань (TRSM) на GPU. Саме ці етапи займають переважну частку загального обсягу обчислень у блочній LU-факторизації і добре відповідають моделі масового паралелізму. Однак ефективність реалізації залежить від правильного формування та розміщення блоків матриці в пам’яті. Стрічкова структура дозволяє уникнути зберігання зайвих нульових блоків, але водночас вимагає обережного формування масивів блоків, щоб забезпечити коалесований доступ нитей до глобальної пам’яті. При неузгодженому розміщенні блоків можливе значне падіння пропускної здатності пам’яті, що призведе до втрати ефективності навіть при використанні потужних ядер BLAS-3.

Панельна частина алгоритму на GPU зазвичай є найменш ефективною, оскільки містить обчислення з вираженими послідовними залежностями та

потребує виконання операцій повної LU-факторизації для діагонального блоку. Розмір такого блоку є порівняно невеликим, а тому не забезпечує достатнього рівня паралелізму для ефективного завантаження GPU. Це зумовлює широке застосування гібридних підходів, коли панельна частина виконується на CPU, а результати передаються на GPU для подальшого використання в TRSM і GEMM. Такий підхід дозволяє уникнути низької продуктивності GPU на малих задачах і водночас мінімізувати загальні накладні витрати. У разі повністю GPU-орієнтованої реалізації панельна факторизація може виконуватися за допомогою одного або невеликої кількості CUDA-блоків з використанням shared memory, що дозволяє приховати частину затримок доступу до глобальної пам’яті, але все одно не забезпечує значного прискорення.

Основний внесок у продуктивність забезпечують операції TRSM та GEMM, які реалізуються через бібліотеку cuBLAS. Ці операції добре оптимізовані для GPU і здатні досягати дуже високої продуктивності. Важливою особливістю є те, що для ефективності GPU необхідна наявність достатньої кількості блочних пар, які підлягають оновленню. У стрічкових матрицях ця кількість визначається шириною стрічки, і тому вузькі стрічки призводять до недостатнього завантаження GPU. При широких стрічках, навпаки, GPU демонструє значне прискорення порівняно з CPU, оскільки може паралельно виконувати велику кількість незалежних операцій GEMM для блоків хвостової підматриці.

Організація CUDA-потоків (streams) є ще одним ключовим елементом оптимізації. Використання кількох потоків дозволяє перекривати копіювання даних і обчислення, а також запускати множину GEMM/TRSM операцій паралельно. Це особливо ефективно в тих етапах, де кількість блоків достатня для повного завантаження всіх потокових мультипроцесорів GPU. Крім того, використання streams дає можливість дробити великі задачі на батчі та оптимально використовувати ресурси GPU при нерівномірному розподілі блоків у стрічковій структурі.

Важливе значення має спосіб реалізації перестановок рядків (pivoting), які є обов'язковими у класичній LU-факторизації. У стрічкових матрицях pivoting зазвичай обмежений межами діагонального блока, щоб не розширювати стрічку. На GPU перестановка рядків, як правило, реалізується через спеціальний kernel, який виконує обмін рядків у межах одного або кількох блоків. Такий підхід зберігає структуру матриці та мінімізує накладні витрати, хоча не забезпечує такої гнучкості, як глобальний pivoting у повних матрицях.

Загалом ефективність CUDA-реалізації визначається балансом між кількістю незалежних блокових операцій та пропускною здатністю пам'яті GPU. Для стрічкових матриць із великою напівшириною GPU досягає суттєвого прискорення порівняно з CPU, тоді як для вузьких стрічок ефективність може бути нижчою через нестачу паралельних задач. Проте навіть у таких випадках GPU часто демонструє кращу продуктивність при виконанні великої кількості TRSM і GEMM, оскільки запуск оптимізованих BLAS-3 ядер cuBLAS дозволяє компенсувати обмеження на рівні алгоритму.

Таким чином, реалізація блочної LU-факторизації стрічкових матриць у CUDA потребує ретельного вибору формату даних, оптимізації доступу до пам'яті, правильного розподілу задач між потоками та максимального використання бібліотек BLAS-3. У разі коректної реалізації GPU забезпечує значний приріст продуктивності для задач великої розмірності та широких стрічкових матриць, тоді як панельні обчислення та вузькі стрічки залишаються потенційними «вузькими місцями», що визначають загальну ефективність алгоритму.

3.4. Порівняння OpenMP та CUDA-реалізацій блочної LU-факторизації стрічкових матриць

Порівняння реалізацій блочної LU-факторизації стрічкових матриць у середовищах OpenMP та CUDA є важливим для визначення оптимальної платформи виконання залежно від розмірності задачі, структури матриці, обчислювальної складності та вимог до продуктивності. Обидва підходи

ґрунтуються на одній і тій самій блочній схемі факторизації, але принципово відрізняються моделлю паралелізму, організацією пам’яті та способом використання апаратних ресурсів. Відмінності між ними визначають їх переваги та обмеження для задач різного масштабу.

У реалізації OpenMP паралелізм організується шляхом розподілу ітерацій циклів між ядрами CPU. Модель OpenMP особливо ефективна для операцій блочного множення матриць, які виконуються на етапі оновлення хвостової підматриці. Цей етап має найбільший рівень паралелізму, оскільки кожен блок, що підлягає оновленню, є незалежною одиницею роботи. При достатній ширині стрічки паралелізація дозволяє повністю завантажити ядра процесора, забезпечуючи помітне прискорення порівняно з послідовною реалізацією. Проте загальна продуктивність OpenMP обмежена невеликою кількістю фізичних ядер CPU та значною часткою послідовних обчислень у панельній частині алгоритму, де виконується факторизація діагонального блока. Через залежності між рядками вона не піддається ефективній паралелізації, що створює природну межу прискорення.

На відміну від CPU, графічні процесори забезпечують масовий паралелізм, завдяки якому CUDA-реалізація може одночасно виконувати сотні або тисячі незалежних блокових операцій. GPU особливо ефективний у виконанні операцій GEMM та TRSM, які становлять основний обсяг FLOPs у блочній LU-факторизації. Оптимізовані бібліотеки cuBLAS дозволяють досягати близьких до пікових характеристик продуктивності. При достатній кількості блоків хвостової підматриці, що притаманно широким стрічковим матрицям, GPU демонструє значно вищу продуктивність порівняно з OpenMP, оскільки може обробляти ці блоки масово та паралельно. У таких випадках прискорення може бути порядку десятикратного або більшого.

Разом із тим CUDA має свої обмеження. Панельна частина алгоритму, як і в OpenMP, залишається слабо паралельною та не забезпечує значного завантаження GPU. Через це часто застосовується гібридний підхід, коли панельна факторизація виконується на CPU, а GPU обробляє лише

високопаралельні блокові оновлення. Іншим обмеженням є недостатнє завантаження GPU у випадку вузьких стрічок, коли кількість блоків у хвостовій підматриці мала і не вистачає паралельної роботи для всіх потокових мультипроцесорів GPU. На малих і середніх розмірностях матриць перенесення даних між CPU та GPU також може нівелювати виграш у продуктивності, роблячи OpenMP-реалізацію більш ефективною.

Загалом ефективність OpenMP-реалізації визначається балансом між часткою паралельних обчислень і числом доступних ядер процесора. Вона є стабільною, передбачуваною і не потребує значних накладних витрат на керування пам’яттю. Навпаки, ефективність CUDA-реалізації залежить від того, наскільки структура задачі здатна забезпечити масову паралельність. При широких стрічкових матрицях GPU значно перевершує CPU, забезпечуючи істотно вищий рівень GFLOPs, тоді як при вузьких стрічках або малих розмірах задач OpenMP може виявитися більш ефективним завдяки меншій латентності та відсутності потреби в копіюванні даних.

Таким чином, обидва підходи мають свої переваги та сфери застосування. OpenMP найбільш ефективний для задач невеликої та середньої розмірності, а також для вузьких стрічкових матриць, де GPU не може бути повністю завантаженим. CUDA забезпечує максимальну продуктивність при великій розмірності задачі та широкій стрічковій структурі, коли кількість блокових операцій достатня, щоб підтримувати повне завантаження графічного процесора. Комбіноване використання CPU та GPU дає змогу поєднати переваги обох підходів: виконувати панельні частини на CPU, а високопаралельні оновлення — на GPU. Такий гібридний підхід є найбільш перспективним для масштабування блочної LU-факторизації стрічкових матриць на сучасних гетерогенних обчислювальних системах, оскільки дозволяє максимально ефективно використовувати ресурси всіх апаратних компонентів.

3.5. Схема програмної архітектури програмного комплексу

Для реалізації паралельного алгоритму блочної LU-факторизації стрічкових матриць було спроєктовано модульну програмну архітектуру, яка забезпечує чітке розділення відповідальностей між окремими компонентами, спрощує супровід коду та дає змогу легко розширювати програмний комплекс за рахунок додавання нових обчислювальних режимів (CPU/OpenMP, GPU/CUDA, MPI тощо). Узагальнена схема програмної архітектури наведена на відповідному рисунку.

На верхньому рівні архітектури розташований головний модуль (driver-програма), який відповідає за взаємодію з користувачем, зчитування параметрів задачі, ініціалізацію структур даних та вибір режиму виконання обчислень. На основі введених параметрів (розмірність матриці, ширина стрічки, розмір блоку, вибір платформи — CPU або GPU) головний модуль формує конфігурацію запуску та ініціює відповідну реалізацію алгоритму: OpenMP-реалізацію на багатоядерному процесорі або CUDA-реалізацію на графічному процесорі. Таким чином, даний компонент виконує функції «координатора» всього програмного комплексу, але не містить обчислювально інтенсивних операцій.

Другий рівень архітектури становить модуль структур даних, який відповідає за представлення стрічкових матриць у пам’яті. У ньому реалізовано спеціалізовані формати зберігання, зокрема band-packed формат, сумісний із процедурою DGBTRF з LAPACK, а також блочний (tile-based) формат, орієнтований на ефективне виконання операцій BLAS-3 як на CPU, так і на GPU. Модуль містить функції для генерації тестових стрічкових матриць із заданими параметрами, конвертації між форматами band та tile, а також засоби для доступу до окремих блоків і елементів матриці з мінімальними накладними витратами. Саме цей модуль забезпечує оптимальну локальність даних та узгодженість структури матриці з обраним алгоритмом.

На наступному рівні розміщено обчислювальні модулі, які реалізують ядро блочної LU-факторизації. CPU-обчислювальний модуль реалізує послідовну та OpenMP-версії алгоритму і використовує високопродуктивні бібліотеки BLAS/LAPACK (наприклад, Intel MKL) для виконання ключових операцій типу матричного множення (GEMM) і трикутних розв’язань (TRSM). У цьому модулі виділено окремі підкомпоненти для панельної факторизації діагональних блоків, виконання TRSM-операцій для блоків поточного рядка та оновлення хвостової підматриці за допомогою GEMM. Модуль GPU-обчислень побудовано аналогічно, але з використанням CUDA і бібліотек cuBLAS/cuSOLVER; він оперує над блочним (tile) представленням матриці, організовуючи копіювання даних на GPU, виклики cublasDtrsm і cublasDgemm, а також, за потреби, запуск спеціалізованих CUDA-ядер для панельної факторизації та перестановок (pivoting). Обидва обчислювальні модулі використовують узгоджений інтерфейс доступу до матриці, що дозволяє легко переключатися між CPU- та GPU-реалізаціями в межах однієї архітектури.

Для підтримки розподілених обчислень і масштабування на кластерні системи архітектура передбачає наявність окремого модуля MPI-координації. Цей модуль відповідає за двовимірний блочно-циклічний розподіл даних між процесами, організацію обміну панелями та блоками матриці між вузлами, а також узгодження кроків факторизації в багатовузловому середовищі. У поточній реалізації цей модуль може розглядатися як перспективний компонент для подальшого розвитку програмного комплексу, що дозволяє переносити напрацьовані методи на розподілені системи з використанням гібридних схем MPI+OpenMP або MPI+CUDA.

Окремо в архітектурі виділено модуль валідації результатів, який реалізує засоби перевірки коректності розв’язку та оцінки чисельної стійкості алгоритму. Він забезпечує обчислення норми залишку для рівняння Ax=b, перевірку відновлення початкової матриці за добутком LU, а також розрахунок відносних похибок у різних нормах. Для цього використовуються

додаткові виклики рутин BLAS (наприклад, операції типу GEMV/GBMV) і стандартні засоби оцінки похибки. Модуль валідації є спільним для CPU- та GPU-реалізацій і працює з уніфікованим інтерфейсом матриці, що забезпечує можливість коректного порівняння результатів між різними платформами.

Ще один важливий компонент архітектури — модуль профілювання та збору статистики. Він відповідає за вимірювання часу виконання окремих етапів алгоритму (панельна факторизація, TRSM, GEMM, копіювання даних між CPU і GPU, MPI-комунікації), підрахунок обсягу виконаних операцій з плаваючою комою (FLOPs), обчислення досягнутої продуктивності в GFLOPs, а також формування табличних і графічних звітів для подальшого аналізу. Цей модуль дозволяє дослідити вплив розмірності задачі, ширини стрічки, розміру блоку та кількості задіяних потоків/блоків на загальну ефективність реалізації.

Отже, схема програмної архітектури програмного комплексу передбачає чітке багаторівневе структурування: верхній рівень керування (головний модуль), рівень структур даних, рівень обчислювальних модулів для різних платформ (OpenMP, CUDA, MPI), а також рівні валідації та профілювання. Такий підхід забезпечує гнучкість, розширюваність і можливість подальшого розвитку програмного комплексу, зокрема шляхом додавання нових форматів матриць, варіантів факторизації або підтримки інших архітектур високопродуктивних обчислень.

3.6. Експериментальне середовище

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 пропускною 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.

3.7 Апробація алгоритму

Запропонований паралельний алгоритм було апробовано на розв'язуванні кількох СЛАР із стрічковими несиметричними матрицями, в тому числі, 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 декілька СЛАР із стрічковими матрицями, параметри яких представлено в табл. 1.

Таблиця 1. Набір тестових матриць

№ п/п	Назва	Порядок *n*	Кількість піддіагоналей *ml*	Кількість наддіагоналей *mu*
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-117-02	117 092	239	239
7	A-798-05	798 624	565	565

В табл.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 2. Часи розв’язання задач

Матриця СЛАР	Час розв’язування (сек.)				
	1 потік	16 потоків	32 потоки	64 потоки	1 GPU
A-126-20	≈ 2 240	240,23	52,32	17,79	15,23
A-126-09	≈ 660	250,35	10,13	2,5	13,46
A-055-10	≈ 170	35,02	4,08	8,45	3,21
A-052-10	≈ 210	65,28	12,28	9,34	3,74
A-137-44	≈ 7 080	420,23	185,70	64.23	45,34
A-117-02	≈ 60	15,49	2,70	8,90	0,98
A-798-05	≈ 800	186,45	35,00	23,75	9,12

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

ВИСНОВОК

У даній кваліфікаційній роботі розв’язано актуальну науково-прикладну задачу підвищення ефективності розв’язування великих систем лінійних алгебраїчних рівнянь, що виникають у задачах чисельного моделювання процесів термопластичності та пошкодження матеріалів. Основну увагу зосереджено на розробці та теоретичному аналізі паралельного блочно-циклічного алгоритму LU-факторизації для несиметричних стрічкових матриць.

У ході виконання роботи отримано такі основні результати:

- Проведено аналіз структури систем лінійних алгебраїчних рівнянь, що виникають при дискретизації задач механіки суцільного середовища методом скінченних елементів. Показано, що відповідні матриці мають несиметричну стрічкову структуру, яка може бути ефективно використана для зменшення обчислювальних витрат та обсягу пам’яті.
- Розроблено блочно-циклічний алгоритм LU-факторизації стрічкових матриць, який поєднує структурну оптимізацію (обмежене заповнення, локальний вибір головного елемента) з використанням високопродуктивних блочних операцій типу BLAS рівня 3. Запропонований підхід забезпечує високу обчислювальну інтенсивність і є придатним для реалізації на сучасних паралельних архітектурах.
- Отримано асимптотичну оцінку загальної кількості арифметичних операцій алгоритму, яка має порядок

  $$\Theta(nml(ml+mu))$$

  що підтверджує істотну перевагу запропонованого методу порівняно з алгоритмами для щільних матриць та демонструє ефективне використання стрічкової структури.
- Виконано теоретичний аналіз паралельної структури алгоритму. Доведено, що домінуючі обчислення зводяться до незалежних операцій матрично-матричного множення, кількість яких на кожному кроці пропорційна добутку напівширин стрічки. Отримано оцінку кількості

паралельно виконуваних операцій, яка показує високий рівень внутрішнього паралелізму алгоритму.

- Виведено аналітичні оцінки паралельного прискорення та ефективності з урахуванням комунікаційних витрат. Показано, що алгоритм забезпечує майже лінійне прискорення на помірній кількості процесів, а масштабованість обмежується латентністю та пропускною здатністю міжпроцесорних обмінів. Встановлено існування оптимальної кількості процесів, що визначається шириною стрічки матриці.
- Показано, що збільшення розміру блоку та ширини стрічки призводить до зростання паралельної ефективності за рахунок підвищення обчислювальної інтенсивності, що робить алгоритм особливо ефективним для реалізації на GPU та multi-GPU системах із використанням оптимізованих бібліотек лінійної алгебри.

Отримані в роботі результати мають як теоретичну, так і практичну цінність. Запропонований алгоритм і наведені оцінки можуть бути використані при розробці високопродуктивних програмних комплексів для розв’язування великих СЛАР у задачах механіки деформівного твердого тіла, теплопровідності та інших галузях обчислювальної фізики.

Подальші напрями досліджень можуть бути пов’язані з розширенням алгоритму на випадок розріджених плиткових матриць, удосконаленням стратегій вибору головного елемента, а також з реалізацією та експериментальною перевіркою алгоритму в середовищах multi-GPU та розподілених обчислень на основі MPI.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Молчанов И.Н. Машинные методы прикладных задач. Алгебра, приближение функций / И.Н. Молчанов. – Киев : Наукова думка, 1987. – 288 с.
2. Нестеренко Б.Б. Основы асинхронных методов параллельных вычислений / Б.Б. Нестеренко, В.А. Марчук – Киев : Наук. думка, 1989. – 176 с.
3. Молчанов И.Н. Введение в алгоритмы параллельных вычислений / И.Н. Молчанов – Киев : Наук. думка, 1990. 128 с.
4. Молчанов И.Н. Машинные методы решения задач прикладной математики. Алгебра, приближение функций, обыкновенные дифференциальные уравнения / И.Н. Молчанов – Київ : Наукова думка, 2007. – 550 с.
5. Параллельные алгоритмы решения задач вычислительной математики / А.Н. Химич, И.Н. Молчанов, А.В. Попов и др. – Киев : Наукова думка, 2008. – 248 с.
6. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем / Дж. Ортега – М. : Мир, 1991. 128 с.
7. Golub G. Scinetific computing. An introduction with parallel computing / G. Golub, J.M. Ortega – Boston : Academic Press, 1993. – 442 p.
8. Воеводин В.В. Параллельные вычисления / Воеводин В.В., Воеводин Вл.В. – СПб. : БХВ-Петербуг, 2002. – 608 с.
9. Freeman T.L. Parallel numerical algorithms / T.L. Freeman, C. Phillips – New York : Prentice Hall International, 1992. – 315 p.
10. Тьюарсон Р. Разреженные матрицы / Р. Тьюарсон – М. Мир, 1977. – 172 с.
11. Писанецки С. Технология разреженных матриц / С. Писанецки – М. Мир, 1988. – 410 с.
12. Джордж А. Численное решение больших разреженных систем уравнений / А. Джордж, Дж. Лю – М. Мир, 1984. – 334 с.

13. Эстребю О. Прямые методы для разреженных матриц / О. Эстребю, З. Златев – М. Мир, 1987. – 118 с.
14. Флинн М.Дж. Сверхбыстродействующие вычислительные системы / М.Дж. Флинн // Труды ИИЭР. – 1966. – Т. 54, № 12. – С. 311–320.
15. Флинн М.Дж. Сверхбыстродействующие вычислительные системы / М.Дж. Флинн // Труды ИИЭР. – 1966. – Т. 54, № 12. – С. 311–320.
16. Statistical lists of supercomputers / H. Meuer, E. Stromhmaier, J. Dongarra, H. Simon – 2009.
17. Quinn M.J. Parallel Computing. Theory and practice / M.J. Quinn – New York : McGraw Hill, 1994. – 446 p.
18. Уилкинсон Дж.Х. Справочник алгоритмов на языке АЛГОЛ. Линейная алгебра / Дж.Х. Уилкинсон, К. Райнш –М. : Машиностроение, 1976. – 389 с.
19. Jones M.T. BlockSolver95: Scalable Software for the Parallel Solution of Sparse Linear Systems [Електронний ресурс] / M.T. Jones, P.E. Plassman. 1997. – Режим доступу : http://www-unix.mcs.anl.gov
20. SUPERLU, a sequential library for the direct solution of large, sparse, nonsymmetric systems of linear equations on high performance machines [Електронний ресурс] / Sherry Li, Jim Demmel, John Gilbert, Laura Grigori. – 2008. – Режим доступу : http://crd.lbl.gov
21. PSparslib, A Portable Library of Parallel Sparse Iterative Solvers [Електронний ресурс] / Yousef Saad, Gen-Ching Lo, Sergey Keznetsov. – 1998. – Режим доступу : http://citeseerx.ist.psu.edu
22. MUMPS, a MUltifrontal Massively Parallel sparse direct Solver [Електронний ресурс] / Iain DUFF. – 2008. – Режим доступу : http://www.cerfacs.fr
23. PSPASES, Parallel SPArse Symmetric dirEct Solver [Електронний ресурс] / Anshul Gupta, Mahesh Joshi, George Karypis, Vipin Kumar, Fred Gustavson. – 1999. – Режим доступу : http://www-users.cs.umn.edu

24. Saad Y. SPARSKIT: a basik tool kit for sparse matrix computations / Y. Saad – 1994.
25. Wilkinson J.H. Rounding Errors in Algebraic Processes / J.H. Wilkinson – London : H.W. Staat. Off, 1963. – 161 p.
26. Воеводин В.В. Матрицы и вычисления / В.В. Воеводин, Ю.А. Кузнецов – М. : Наука, 1984. – 318 с.
27. Воеводин В.В. Ошибки округлений и устойчивость в прямых методах линейной алгебры / В.В. Воеводин – М. : ВЦ МГУ, 1969. – 153 с.
28. Химич А.Н. О достоверности линейных математических моделей с приближенно заданными исходными данными / Химич А.Н., Войцеховский С.А., Брусникин В.Н. // Математические машины и системы. – 2004. – № 3. – С. 54–62.
29. Химич А.Н. О полной погрешности расчета линейных математических моделей итерационными методами // Химич А.Н., Яковлев М.Ф. // Кибернетика и системный анализ. – 2002. – № 5. – С. 132–142.
30. Химич А.Н. Оценки полной погрешности решения систем линейных алгебраических уравнений для матриц произвольного ранга / Химич А.Н. // Компьютерная математика. – 2002. – № 2. – С. 41–49.
31. Гайсарян С.С. Описание пакета программ для решения симметричной алгебраической проблемы собственных / Гайсарян С.С., Теплякова Т.А. // М.: Деп. в ВИНИТИ № 3447-83. – 1983. – 12 с.
32. Dongarra J.J. LINPACK users guide / Dongarra J.J., Moler C.B., Bunch J.R., Stewart G.W. – Philadelphia: SIAM, 1979. – 363 p.
33. Garbow B.S. Matrix Eigensystem Routine / Garbow B.S., Royle J.M., Dongarra J.J., Moller M.M. – EISPACK Guide Extension Lecture Notes in Computer Science, vol. 51. Springer-Verlag, 1977. – 236 p.
34. 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.

35. 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.
36. Проблемы экспертизы современных сварных конструкций ответственного назначения // В.И. Махненко / Автоматическая сварка. – №5. – 2013. – С. 22–29.
37. *Попов А.В., Химич А.Н.* Параллельный алгоритм решения системы линейных алгебраических уравнений с ленточной симметричной матрицей // Компьютерная математика. - 2005. - № 2. - С. 52-59.
38. *Хіміч О.М., Баранов А.Ю.* Гібридний алгоритм розв'язування лінійних систем зі стрічковими матрицями прямими методами. // Комп'ютерна математика. – 2013, Вып. 2. – С. 80-87.
39. *Хіміч О.М., Сидорук В.А.* Плитковий алгоритм факторизації розрідженої матриці // Комп'ютерна математика. – 2015. – Вип. 2. – С. 109-116.
40. *Хіміч О.М., Сидорук В.А.* Комп'ютерна програма «SPARSE_SOLVER» для дослідження та розв'язування систем лінійних алгебраїчних рівнянь з розрідженими матрицями на комп'ютерах з багатоядерними процесорами та графічними прискорювачами // Свідоцтво на реєстрацію авторського права на твір № 65346 від 16.05.2016 р., Держ. служба інтелект. влас. України.
41. Обчислювальний комплекс СКІТ ІК НАН України - Документація - Суперкомп'ютер ІК НАН України http://icybcluster.org.ua/index.php?lang_id=2&menu_id=5
42. Стаття: Ігнатов М. Паралельний блочний алгоритм розв'язування систем лінійних алгебраїчних рівнянь з стрічковими матрицями. Вісник Кам'янець-Подільського національного університету імені Івана Огієнка. Фізико-математичні науки. Кам'янець-Подільський : К-ПНУ ім. І. Огієнка, 2024. № 17. С. 51-53.