КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра прикладної статистики

Кваліфікаційна робота
на здобуття ступеня бакалавра
за спеціальністю 124 Системний аналіз
на тему:

Метод зведення розв'язку багатовимірних диференціальних рівнянь у частинних похідних до розв'язування серії пов'язаних одновимірних задач з подальшим узгодженням їх розв'язків

Виконав студент 4-го курсу
Вітюк Артемій Денис Миколайович


(підпис)

Науковий керівник:
Доктор фізико-математичних наук
Ляшко Сергій Іванович


(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент


(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри прикладної статистики
«____» _____ 202_ р.,
протокол № ___
Завідувач кафедри
Розора І.В. _____
(підпис)

Київ-2023

РЕФЕРАТ

Обсяг роботи 41 сторінка, 5 ілюстрацій, 11 джерел посилання.

ДИФЕРЕНЦІЙНІ РІВНЯННЯ В ЧАСТКОВИХ ПОХІДНИХ, ДЕКОМПОЗИЦІЯ, КООРДИНАЦІЯ, ФУНКЦІОНАЛ, ГРАДІЄНТ, КРОКОВИЙ МНОЖНИК, ГРАФІЧНІ ПРОЦЕСОРИ, РОЗПАРАЛЕЛЕННЯ, СИСТЕМИ ЛІНІЙНИХ РІВНЯНЬ

Об’єктом роботи є розробка метода розпаралелювання чисельних розв’язків дифференційніх рівнянь в часткових похідних шляхом просторової декомпозиції вихідного рівняння з подальшою координацією отримуємих розв”язків відповідних одновимірних задач. Предметом роботи є отримання явного виду градієнту функціоналу відхилення розв”язків систем лінійних рівнянь з заданою функцією координатором та пошук формули для обчислення оптимального крокового множника для пропонуємого градієнтного метода.

Метою роботи є розробка градієнтного метода координації систем лінійних рівнянь та його чисельна веріфікація.

Методи розроблення: метод кінцевіх різніць, градієнтні методи теорії оптимізації.

Інструмент верифікації: мова програмування Python.

Результати роботи: проведена постановка задачі для координації систем лінійних рівнянь загального виду; у випадку координації розв’язків двох систем лінійних рівнянь отримана формула для обчислення градієнту відповідного функціонала через розв’язки спряжених задач, у цьому випадку також отримана формула для розрахунку оптимального крокового множника градієнтного метода; на прикладі конкретних систем лінійних рівнянь був проведен чисельний експеримент мінімізації функціоналу запронованого методу; швидка збіжність доводить перспективність подальших досліджень у цьому напрямку; у двовимірній області традиційним кінцево-різницевим методом був розв’язок рівняння Лапласа з метою можливого подальшого порівняння результатів з тими, що будуть отримані з використанням розпаралелення вирішення цього рівняння на графічних процесорах; був модифікован метод квадратних коренів для пошуку розв’язків систем з трьохдіагональними симетричними матрицями із зберіганням матриць у векторному вигляді для економії пам’яті та виключення множення на нуль.

Проведена робота доводить перспективність проведення подальших досліджень у цьому напрямку.

Зміст

1.Вступ

Широкий спектр прикладних задач, в тому числі захисту навколишнього середовища, вимагає моделювання фізичних процесів в областях великих лінійних розмірів та з урахуванням значної кількості вхідних параметрів. Найчастіше дослідникам, щоб отримати достовірні прогнози для оцінки параметрів екосистем приходиться розв'язувати складні системи нелінійних диференціальних рівнянь в часткових похідних. Очевидно, що розв'язок таких систем вимагатиме залучення значних обчислювальних ресурсів. Насамперед мова йде про вимоги до швидкості обчислень, можливості їх збереження та обробляння, причому бажано з використанням оперативної пам'яті обчислювального комплекса з великими масивами вхідної інформації. Тобто ,мова йде про такі ресурси, які не можуть поки що надати навіть найсучасніші персональні комп'ютери. Такі можливості мають тільки сучасні суперкомп'ютери, але їх використання для пересічного дослідника є проблемою через значну вартість та складності отримання доступу до існуючих з метою проведення на них обчислень.

Останнім часом увагу науковців привернули технології використання графічних процессорів персональних компьютерів (GPU) для організації на них паралельних обчислень. Сучасні GPU зазвичай, на відміну від центрального процесора персонального комп'ютера, мають декілька тисяч обчислювальних ядер. Так, наприклад, один із сучасних графічних процесорів "NVIDIA RTX 3090" має 10496 ядер. Це теоретично дозволяє організовувати та проводити одночасно обробку великої кількості однотипних обчислень. В 2007 році один із світових лідерів розробки та виробництва графічних карт NVIDIA Corporation представив і почав підтримувати свій продукт CUDA (*Compute Unified Device Architecture*) –

програмно-апаратну архітектуру паралельних обчислень. CUDA SDK (*Software Development Kit*) дає змогу розробляти і реалізовувати алгоритми паралельних обчислень на графічних процесорах. NVIDIA Corporation продовжує активно розвивати напрямок використання графічних карт для загальних обчислень. Це означає, що буде і далі активно розвиватись технологія, яка навіть отримала свою окрему назву GPGPU - General Purpose Computation on Graphics Processing Unit. Так останній на сьогодні продукт CUDA Toolkit 12.1 був анонсований компантґю у грудні 2022 року і став доступним на початку 2023 року. Тому особливо актуальною зараз є розробка алгоритмів паралельних обчислень для розв'язання складних прикладних задач.

В даній роботі розвивається напрямок і методика заміни розв'язків багатовимірних задач математичної фізики розв'язкамі відповідних одномірних задач з подальшою їх координацією шляхом мінімізації деякого функціонала. На простому прикладі розв'язку рівняння Лапласа з відповідними граничними умовами у двовимірній прямокутній області пояснюється суть запропонованого метода. Для мінімізації функціоналу отримані алгоритми знаходження його градієнту. Пропонується методика пошуку крокового множника для координації систем лінійних рівнянь. Написана програма розрахунку розподілу тепла у прямокутній пластині. Ці розрахунки можуть бути використані для перевірки рішень скоординованих одновимірних задач. Методика координації систем лінійних рівнянь перевірена на прикладі двух систем лінійних рівнянь. Для розв'язку систем лінійних рівнянь відповідних одновимірних задач пропонується використовувати метод квадратного кореня, а матриці зберігати у векторному виді. Це дозволяє значно економити пам'ять та виключає виконання беззмістовних операцій множення на нуль при пошуках рішень розріжених систем лінійних рівнянь, Проведено огляд методів розпаралелення розв'язків задач математичної фізики. Проведена робота може служити базою для розробки алгоритмів розпаралелення обчислень на чипах графічних карт.

1.1 Огляд деяких методів організації паралельних обчислень для розв'язків рівнянь математичної фізики.

Розпаралелення обчислень на багатопроцесорних обчислювальних комлексах означає, що були розроблені методи та відповідні програми, які виконують кілька задач одночасно і незалежно одна від одної. Після одночасного виконання наперед визначених блоків математичних операцій повинен відбутися обмін інформацією між блоками та керуючим процесором . Технічно це досягається за рахунок об'єднання декількох процесорів, працюючих з загальною або з розподіленою пам'яттю.
Але сподіватись на те, що заміна розрахунків на однопроцессорному обчислювальному комплексі обчислюваннями на багатопроцесорних комплексах, наприклад з М процессорами, призведе до збільшення швидкості обчислень в М раз, найчастіше не приходиться. Може так статися у наслідок недолугості алгоритмів розпаралелення, що збільшення кількості процессорів на яких проводяться обчислення може не прискорити розрахунки, а ,навпаки, взагалі загальмувати роботу всієї обчислювальної системи в цілому. Тому, особливо важливо, розробляти такі алгоритми та методи розпаралелення, які б з урахуванням архітектури обчислювальної системи, її апаратного наповнення давали б найбільший приріст продуктивності. Розв'язок задачі в паралельному режимі потребує правильного вибору моделі розпаралелювання послідовного програмного коду, тісно пов'язаної з апаратними та системними засобами паралельної обробки даних. У цьому обзорі ми будемо використовувати деякі результати [Sorokin, M. and others, 2020] .

Одним із самих розповсюджнних методів чисельного розв'язку задач математичної фізики за допомогою обчислювальних технологій і методів паралельного програмування є метод геометричної декомпозиції розрахункової області на підобласті, кількість яких дорівнює числу

процесорів, розрахунку кожним процесором своєї підобласті та обміні даними між ними на кожному кроці за часом.

Методологічний підхід до розв'язку задачі за допомогою засобів паралельної обробки даних полягає в реалізації наступних етапів [Foster, 1995].

1. Декомпозиція (partitioning). Розбиття задачі обчислення і обробки даних на мінімальні незалежні підзадачі.

2. Комунікації (communication). Визначення структури підзадач та встановлення необхідних зв'язків між ними.

3. Кластерізація (agglomeration). Оцінка структури під задач та їх об'єднання з ціллю мінімізації комунікацій і підвищення продуктивності.

4. Розподілення (mapping). Призначення підзадач конкретним процесорам та забезпечення їх рівномірного навантаження.

Представлена схема відображає відсутність строго формалізованого підходу до паралельного програмування. На кроках 1 і 2 виділяються паралельні гілки обчислювального алгоритму та закладається його масштабованість (гіркість по відношенню до числа процесорів). Реалізація кроків 3 і 4 потребує врахування архітектури багатопроцесорної системи для забезпечення мобільності та локальності алгоритму.

Дуже важливим етапом створення паралельних програм є етап теоретичної розробки глобальної стратегії паралелізації досліджуваного процесу. Ця глобальна стратегія спирається на принцип декомпозиції повної задачі на ряд задач, що виконуються одночасно.

У роботі [Тарнавский и др., 2005] визначаються чотири види декомпозиції: *фізико-математична*, *геометрична* (крупно масштабну та мілкомасштабну), *технологічна* і *декомпозиція* «*за головними вхідними параметрами*». Інші [Dongarra, 2003] – виділяють два типи декомпозиції: геометричну та *декомпозицію за задачами* (task decomposition), об'єднуючи в останню технологічну та фізико-математичну. У статті [Тарнавский и др., 2005], здається, найбільш повно описуються існуючи декомпозиційні методи.

Фізико-математична декомпозиція задачі є одним з глобальних видів структурування високого рівня і пов'язана з розщепленням досліджуваного фізичного процесу за підпроцесами, що його складають, і, відповідно, з сегментацією загального математичного алгоритму розв'язку повної задачі на ряд алгоритмів розв'язку окремих підзадач. При вивченні процесів, що відбуваються в газових та рідких середовищах з врахуванням широкого спектру фізичних явищ, можуть бути промодельовані роздільно (паралелізовані) підпроцеси «динаміка», «термодинаміка», «теплопередача», «динаміка твердої фази» і таке інше.

Геометрична декомпозиція задачі (тут і в подальшому мається на увазі в основному крупномасштабна декомпозиція) і наступна паралелізація обчислювань, також як і фізико-математична, є одним із видів глобального структурування високого рівня і дозволяє суттєво знизити величину часу, потрібного на розв'язок. Геометрична декомпозиція полягає в розділенні всієї області інтегрування на карту підобластей (сегментів), розв'язок в яких проводиться одночасно і незалежно з наступною зшивкою розв'язків на границях сегментів, які прилягають один до одного.

В додаток до фізико-математичної та геометричної декомпозиції повної задачі на ряд підзадач з їх паралельним виконанням може бути розглянуто і використано ще один вид декомпозиції – *технологічна декомпозиція*, яка, в свою чергу, може бути розділена на три підвиди: високого, середнього та низького рівнів. Взагалі кажучи, це достатньо різні типи декомпозиції, об'єднані лише тим фактом, що всі вони представляють тільки технічні засоби проведення паралельного розрахунку. Для оптимізації чисельного розв'язку повної задачі, комплекс програм, що забезпечує цей розв'язок, як правило структурован і складається із набору дій, що мають визначений алгоритмічний сенс, який реалізується замкнутими автономними підпрограмами. Таке структурування і є технологічною декомпозицією (сегментацією) високого рівня.

1.2 Опис метода декомпозиції багатовимірних задач до розв”язків серій одновимірних

В даній роботі пропонуються та частково реалізується деякі нові підходи розробки алгоритмів паралельних обчислень для чисельних розв’язків багатовимірних задач математичної фізики. Метод базується на ідеї заміни пошуку чисельних розв’язків первинного рівняння на дво- чи тривимірній сітці апроксимації, пошукам розв’язків серії одновимірних задач з наступною їх координацією. Під координацією рішень одновимірних задач мається на увазі пошук такої функції координатора, що мінімізує функціонал різниці розв’язків цілком незалежних одновимірних задач у вузлах сітки. Тобто ,ми фактично отримаємо задачу оптимального керування системами лінійних рівнянь.

Ідея цього метода була запропонована у роботах Ліонса Ж.-Л. та Марчука Г.І. ще на початку 80-х років минулого століття та тільки тепер набула своєї актуальності у зв’язку з появою можливостей проводити паралельні обчислення на відносно дешевих графічних процесорах персональних комп’ютерів.

Значний вклад в методики застосування градієнтних методів для розв”яку оптимізаційних задач був зроблен українською математичною школою в роботах Б.М.Пшеничного, Ю.М.Даніліна, С.І.Ляшка, О.Г.Наконечного та багатьох інших представників цієї математичної школи.

Пояснимо суть запропонованого метода на прикладі пошуку розв’язків рівняння Лапласа в прямокутній області

$$L(\varphi) = \frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = 0 \qquad (1)$$

із типовими для цього рівняння граничними умовами.

Замість пошуку чисельного розв'язку рівняння (1), ми будемо розв'язувати сукупність простих одновимірних задач з відповідними граничними умовами та функцією координатора в правій частині.

$$\frac{\partial^2 u}{\partial x^2} = -q(x, y) \qquad (2)$$

$$\frac{\partial^2 v}{\partial y^2} = q(x, y) \qquad (3)$$

Розв'язати поставлену задачу, пропонується як задачу мінімізації функціоналу відхилень функцій u(q) і v(q) у вузлах апроксимаційної сітки для рівняння (1). Тобто ,якщо в результаті застосування якого-небудь кінцево-різнецевого шаблону для апроксимації розв"яку рівняння (1) нам необхідно буде шукати розв'язки в N вузлах сітки, то ,відповідно, нам треба буде знайти таку функцію q(x,y) , яка б в вузлах апроксимаційної сітки мінімізувала б функціонал різниці відхилення розв'язків одновимірних задач одна від іншої в нормі одного із підходящих просторів.

$$J\,(q) = \frac{1}{2} \quad \|u(q) - v(q)\|^2$$

Розглянемо просту стаціонарну задачу находження розподілу тепла T(x,y) в однорідній прямокутній пластині. Нехай задана прямокутна пластина у якої ліва та права бічні границі теплоізольовані , а на верхній та нижній границях постійно підтримується якийсь заданий температурний режим

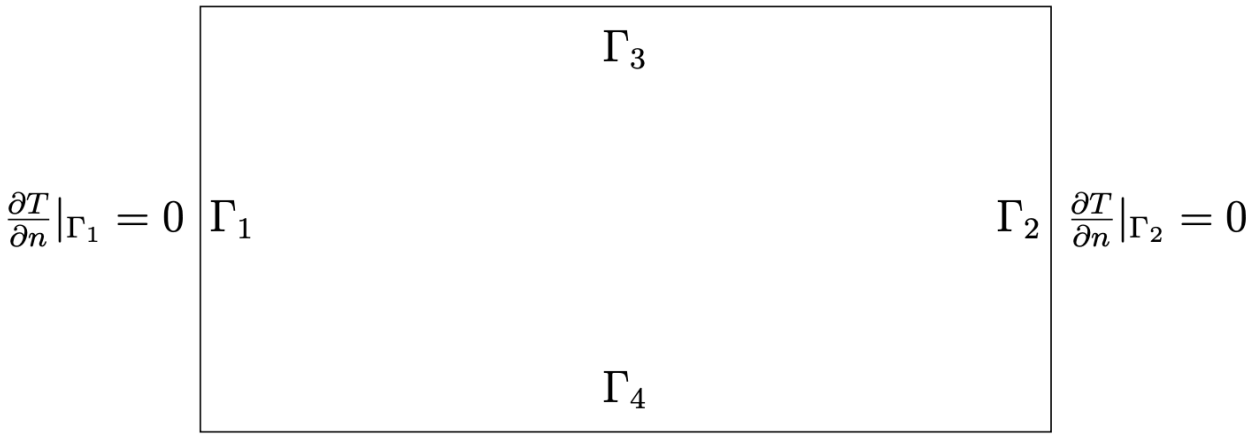

Рис. 1-Однорідна прямокутна пластина з заданими температурами

Тобто, математична модель цієї задачі буде мати наступний вигляд. В прямокутній області треба розв’язати рівняння Лапласа

$$-\frac{\partial^2 \mathrm{T}}{\partial \mathrm{x}^2} - \frac{\partial^2 \mathrm{T}}{\partial \mathrm{y}^2} = 0 \qquad (5)$$

з відповідними граничними умовами

$$\left.\frac{\partial \mathrm{T}}{\partial n}\right|_{\Gamma_1} = 0 \quad \text{і} \quad \left.\frac{\partial \mathrm{T}}{\partial n}\right|_{\Gamma_2} = 0$$

$$T|_{\Gamma_3} = \varphi_1(x, y) \quad \text{та} \quad T|_{\Gamma_4} = \boldsymbol{\varphi_2}(\mathrm{x}, \mathrm{y}) \qquad (5.1)$$

Для чисельного розв’язання рівняння (5) з заданими граничними умовами та більшої наглядності застосуємо найпростійший 5-точковий

шаблон на рівномірній сітці з кроками Δx та Δy у напрямку осі X та осі Y відповідно.

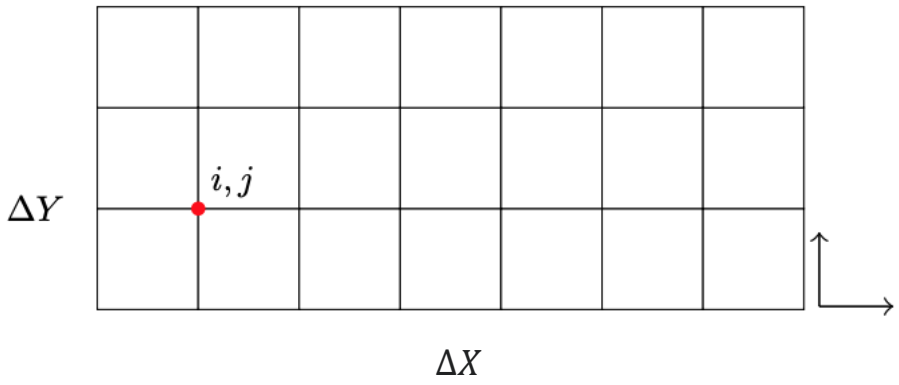

Рис. 2- Найпростійший 5-точковий шаблон на рівномірній сітці з кроками Δx та Δy у напрямку осі X та осі Y відповідно.

Тоді для деякого внутрішнього вузла апроксимаційної сіткі (i , j) отримаємо рівняння

$$-\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{\Delta x^2} - \frac{T_{i,j-1} - 2T_{i,j} + T_{i,,j+1}}{\Delta y^2} = 0 \qquad (6)$$

Замість рівняння (6) для вузла сітки (i , j) розглянемо наступну систему рівнянь

$$\begin{cases} -\frac{u_{i-1,j}-2u_{i,j}+u_{i+1,j}}{\Delta x^2} = -q_{i,j} \\ -\frac{v_{i,j-1}-2v_{i,j}+v_{i,j+1}}{\Delta y^2} = q_{i,j} \end{cases} \quad (7)$$

Якби нам вдалося знайти таке значення $q_{i,j}$ щоб $u_{i,j} = v_{i,j}$ в усіх вузлах сітки, то додавши рівняння системи (7) одне до одного ,ми б отримали рівняння (6).

2.Постановка задачі координації систем лінійних рівнянь

Якщо б ми розв'язували задачу (5) розбиваючи прямокутник по осі X на m точок, а по осі Y на n точок, то з урахуванням граничних умов ми були б змушені розв'язувати систему (m-2) x (n-2) лінійних рівнянь.

Задавши довільні значення параметра $q_{i,j}$ в системі (7) нам треба було б цілком **незалежно** розв'язувати (n-2) системи (m-2) рівнянь по вісях X та (m-2) системи (n-2) рівнянь по осям Y. Незалежність цих систем дає змогу

повністю розпаралелити процес знаходження їх розв'язків. Більше того: у випадку нашої задачі (5) ці рівняння мають 3-х діагональну структуру з симметричною додатньовизначеною матрицею. Застосовуючи для їх розв'язку методи, які не впливають на структуру вихідної матриці та є для них дуже ефективними. Наприклад ,застосувавши метод квадратних коренів або, як його ще називають метод Холецького, та зберігаючи діагональ та допоміжну діагональ у векторному виді ми могли б отримати значну економію пам'яті. Цей підхід до розв'язку систем також автоматично виключає непотрібні множення на нуль, що також значно підвищує швидкість обчислень.

В результаті ми отримуґмо розпаралелення на самому низькому рівні, коли треба буде розв"язувати велику кількість дуже простих задач.

Наприклад, далі ми будемо розв"язувати задачу (5), (5.1) використовуючи рівномірну сітку з 51 х 121 вузлами розбиття нашої області лінійних розмірів 5 на 12. Після урахування граничних умов (5.1) нам довелось би розв"язувати систему із 6000 лінійних рівнян. У випвдку же декомпозиції за нашою методикою треба і можна буде одночасно розв"язувати 50 однотипних систем із 120 рівнянь та 120 систем із 50 рівнян.

В кінцевому випадку ми отримаємо задачу оптимального керування наборами матричних рівнянь, де керуючою функцією буде виступати функція-координатор.

Цю задачу у формальному вигляді, не забуваємо, що у нас сітка з m вузлів по вісі X та n вузлів по вісі Y, можна представити наступним чином:

Знайти такий набір координуючих впливів у вузлах сітки апроксимації

$\boldsymbol{q} = (q_1, \dots\dots, q_N)$, де N=(m-2) x (n-2)

сукупностями матричних рівнянь

(8)

$$[\boldsymbol{A_k}]\overrightarrow{u_k} = -[\boldsymbol{DX_k}]\,\overrightarrow{q_k} + \overrightarrow{FX_k} \quad , \quad k \in [1, n-2]$$

$$[\boldsymbol{B_l}]\overrightarrow{v_l} = \quad [\boldsymbol{DY_l}]\,\overrightarrow{q_l} + \overrightarrow{FY_l} \quad , \qquad l \in [1, m-2] \tag{9}$$

який би мінімізував функціонал

$$J\,(\boldsymbol{q}) = \frac{\mathbf{1}}{\mathbf{2}} \| \boldsymbol{u}(\boldsymbol{q}) - \boldsymbol{v}(\boldsymbol{q}) \|^{\mathbf{2}}_{\mathbb{R}^{\boldsymbol{N}}} \tag{10}$$

В постановці задачі (8) - (10)

$[\boldsymbol{A_k}]$ та $[\boldsymbol{B_l}]$ - симетричні квадратні трьохдіагональні додатньовизначені матриці розмірності (m-2) x (m-2) та (n-2) x (n-2) відповідно.

$[\boldsymbol{DX_k}]$ та $[\boldsymbol{DY_l}]$ – довільні ненульві квадратні матриці розмірності (m-2) x (m-2) та (n-2) x (n-2) відповідно.

$\boldsymbol{q} = (q_{1,}\ldots, q_N)$ – вектор-координатор

$\overrightarrow{q_k}$ та $\overrightarrow{q_l}$ - керуючі вектори совокупностями систем розмірності m-2 та n-2 відповідно.

$\overrightarrow{FX_k}$ та $\overrightarrow{FY_l}$ – вектори правих частин лінійних рівнянь розмірності m-2 та n-2 відповідно.

$\overrightarrow{u_k}$ та $\overrightarrow{v_l}$ - вектори, розв’язки систем (8) та (9), розмірності m-2 та n-2 відповідно.

Зауважимо., що об’єднання індексів *k* по *k* від 1 до n-2 та індексів *l* по *l* від 1 до m-2 дасть нам саме N, як у випадку індесів *k* так і у випадку індексів *l*.

Пропонується розв’язувати задачу (8)-(10), тобто мінімізувати функціонал (10) на розв’язках систем (8) та (9) , якимось із градієнтних методів. Для цього треба спочатку знайти формули для обчислення градієнта функціоналу , а потім розробити алгоритм отримання оптимального крового множника на кожному етапі ітераційної процедури.

Для початку зробимо вивід співідношень для обчислення градієнту на прикладі координації двух систем лінійних рівнянь однакової розмірності.

3. Знаходження градієнту функціоналу на прикладі координації 2-х систем лінійних рівнянь загального виду

Розглянемо дві системи лінійних алгебраїчних рівнянь достатньо загального вигляду

$$[\boldsymbol{A}]\boldsymbol{u} = -[\boldsymbol{DX}]\ \boldsymbol{q} + \boldsymbol{FX} \qquad (11)$$

$$[\boldsymbol{B}]\boldsymbol{v} = \quad [\boldsymbol{DY}]\ \boldsymbol{q} + \ \boldsymbol{FY} \qquad (12)$$

Для якогось **q** на множині відповідних розв'язків систем (10) та (11) задамо функціонал:

$$J(\boldsymbol{q}) = \frac{1}{2}\| \boldsymbol{u}(\boldsymbol{q}) - \boldsymbol{v}(\boldsymbol{q})\|^2_{\mathbb{R}^N} \qquad (13)$$

У співвідношеннях (11) та (12) [**A**] та [**B**] – додатньовизначені симетричні квадратні матриці розмірності N x N.

[**DX**] та [**DY**] – квадратні ненульові матриці тієї ж розмірності

u, v, FX, FY та **q** вектори розмірності N.

Додамо деякий приріст $\Delta\boldsymbol{q} = (\Delta q_1, \ldots, \Delta q_N)$ керуючого впливу на системи (11) та (12), тоді $\mathbf{u}(\mathbf{q}+\Delta\boldsymbol{q})$ та $\mathbf{v}(\mathbf{q}+\Delta\boldsymbol{q})$ будуть розв'язками наступних систем лінійних рівнянь:

$$[\boldsymbol{A}]\mathbf{u}(\mathbf{q} + \Delta\boldsymbol{q}) = -[\boldsymbol{DX}]\boldsymbol{u}(\mathbf{q} + \Delta\boldsymbol{q}) + \boldsymbol{FX} \qquad (14)$$

$$[\boldsymbol{B}]\mathbf{v}(\mathbf{q} + \Delta\boldsymbol{q}) = [\boldsymbol{DY}]\mathbf{v}(\mathbf{q} + \Delta\boldsymbol{q}) + \boldsymbol{FY} \qquad (15)$$

Позначимо $\Delta\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{q} + \Delta\boldsymbol{q}) - \boldsymbol{u}(\boldsymbol{q})$ та $\Delta\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{q} + \Delta\boldsymbol{q}) - \boldsymbol{v}(\boldsymbol{q})$

тоді $\Delta\boldsymbol{u}$ та $\Delta\boldsymbol{v}$ будуть розв'язками наступних систем лінійних рівнянь:

$$[\boldsymbol{A}]\Delta\boldsymbol{u} = -[\boldsymbol{DX}]\Delta\boldsymbol{q} + \boldsymbol{FX} \qquad (15)$$

$$[\boldsymbol{B}]\Delta\mathbf{v} = [\boldsymbol{DY}]\Delta\boldsymbol{q} + \boldsymbol{FY} \qquad (16)$$

Розглянемо відповідну до $\Delta\boldsymbol{q}$ варіацію функціоналу, яка може бути записана наступним чином :

$$J(\boldsymbol{q}+\Delta\boldsymbol{q})-J(\boldsymbol{q})=$$

$$=\quad \frac{\mathbf{1}}{\mathbf{2}}\{\|\mathbf{u}(\mathbf{q}+\Delta\boldsymbol{q})-\mathbf{v}(\mathbf{q}+\Delta\boldsymbol{q})\|^{\mathbf{2}}_{\mathbb{R}^N}-\ \|\boldsymbol{u}(\boldsymbol{q})-\boldsymbol{v}(\boldsymbol{q})\|^{\mathbf{2}}_{\mathbb{R}^N}\}=$$

$$=\frac{\mathbf{1}}{\mathbf{2}}\{\|\Delta\boldsymbol{u}+\mathbf{u}(\mathbf{q})-\Delta\boldsymbol{v}-\boldsymbol{v}(\boldsymbol{q})\|^{\mathbf{2}}_{\boldsymbol{R}^N}-\ \|\boldsymbol{u}(\boldsymbol{q})-\boldsymbol{v}(\boldsymbol{q})\|^{\mathbf{2}}_{\mathbb{R}^N}= \qquad (17)$$

$$=\frac{\mathbf{1}}{\mathbf{2}}\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{N}}(\Delta u_i\,(\boldsymbol{q})-\Delta v_i(\boldsymbol{q})+\boldsymbol{u_i}(\boldsymbol{q})-\boldsymbol{v_i}(\boldsymbol{q}))^{\mathbf{2}}-\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{N}}(u_i\,(\boldsymbol{q})—\boldsymbol{v_i}(\boldsymbol{q}))^2\ =$$

$$=\frac{\mathbf{1}}{\mathbf{2}}\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{N}}((\Delta u_i(\boldsymbol{q})-\Delta v_i(\boldsymbol{q}))^{\mathbf{2}}+$$

$$+\ \mathbf{2}\big(\Delta u_i(\boldsymbol{q})-\Delta v_i(\boldsymbol{q})\big)\big(u_i(\boldsymbol{q})-\ v_i(\boldsymbol{q})\big)\ =\frac{\mathbf{1}}{\mathbf{2}}\ \|\Delta\boldsymbol{u}(\boldsymbol{q})-\Delta\boldsymbol{v}(\boldsymbol{q})\|^{\mathbf{2}}_{\mathbb{R}^N}\ +$$

$$+\langle\Delta\boldsymbol{u}(\boldsymbol{q})-\Delta\boldsymbol{v}(\boldsymbol{q})\,,\,\boldsymbol{u}(\boldsymbol{q})-\boldsymbol{v}(\boldsymbol{q})\rangle$$

де останній доданок $\langle\cdot,\cdot\rangle$ – це скалярний добуток в $\mathbb{R}^N$.

Введемо дві допоміжні задачі:

$$[\boldsymbol{A}]^{\boldsymbol{T}}\boldsymbol{\varphi_x}=-(\boldsymbol{u}(\boldsymbol{q})-\boldsymbol{v}(\boldsymbol{q})) \qquad (18)$$

та

$$[\boldsymbol{B}]^{\boldsymbol{T}}\boldsymbol{\psi_y}=\boldsymbol{u}(\boldsymbol{q})-\boldsymbol{v}(\boldsymbol{q}) \qquad (19)$$

де $[\boldsymbol{A}]^{\boldsymbol{T}}$ та $[\boldsymbol{B}]^{\boldsymbol{T}}$ - матриці, транспоновані до [**A**] та [**B**].

Тоді скалярний добуток в (17) може бути перетворений наступним чином:

$$\langle \Delta \boldsymbol{u}(\boldsymbol{q}) - \Delta \boldsymbol{v}(\boldsymbol{q}) \,,\, \boldsymbol{u}(\boldsymbol{q}) - \boldsymbol{v}(\boldsymbol{q}) \rangle =$$

$$= \quad \langle \, \Delta \boldsymbol{u}(\boldsymbol{q}) \,,\, \boldsymbol{u}(\boldsymbol{q}) - \boldsymbol{v}(\boldsymbol{q}) \rangle \; - \quad \langle \Delta \boldsymbol{v}(\boldsymbol{q}) \,,\, \boldsymbol{u}(\boldsymbol{q}) - \boldsymbol{v}(\boldsymbol{q}) \rangle \; =$$

$$= \quad - \langle \Delta \boldsymbol{u}(\boldsymbol{q}) \,,\; [\boldsymbol{A}]^{\boldsymbol{T}} \boldsymbol{\varphi_x} \rangle \quad - \qquad \langle \Delta \boldsymbol{v}(\boldsymbol{q}),\; [\boldsymbol{B}]^{\boldsymbol{T}} \boldsymbol{\psi_y} \rangle =$$

$$= - \langle [\boldsymbol{A}] \Delta \boldsymbol{u}, \boldsymbol{\varphi_x} \rangle \; - \; \langle [\boldsymbol{B}] \Delta \boldsymbol{v}, \boldsymbol{\psi_y} \rangle \;\; = \;\; \langle [\boldsymbol{DX}] \Delta \boldsymbol{q}, \boldsymbol{\varphi_x} \rangle - [\boldsymbol{DY}] \Delta \boldsymbol{q} \;\; , \boldsymbol{\psi_y} \rangle =$$

$$= \qquad \langle \;\; [\boldsymbol{DX}]^{\boldsymbol{T}} \boldsymbol{\varphi_x} \; - \;\; [\boldsymbol{B}]^{\boldsymbol{T}} \boldsymbol{\psi_y} \,, \Delta \boldsymbol{q} \cdot \rangle$$

Тобто градієнт функціоналу може бути знайдено із співвідношення:

$$\frac{\partial J(\boldsymbol{g})}{\partial q_i} = \sum_{j=1}^{N} (dx_{i,j} \varphi_x^i - \, dy_{i,j} \varphi_y^i) \qquad \forall \; i \in [1, N] \qquad (20)$$

де $dx_{i,j}$ та $dy_{i,j}$ елементи транспонованих матриць [**DX**] $^{\boldsymbol{T}}$ та [**DY**] ^{T} до матриць $[\boldsymbol{DX}]$ **та** $[\boldsymbol{DY}]$ із співвідношень (11) та (12), а φ_x^i та φ_y^i відповідні компоненти розв'язків допоміжних задач (18) та (19).

У випадку,коли матриці [**DX**] та [**DY**] одиничні, а саме так у випадку, коли ми будемо розв'язувати серії одновимірних задач, формула (20) спрощується та набуває виду

$$\frac{\partial J(\boldsymbol{q})}{\partial q_i} = \varphi_x^i - \psi_y^i, \qquad \forall \quad i \in [1, N] \tag{21}$$

4. Вибір крокового множника градієнтного метода координації систем лінійних рівнянь

Якщо ми отримали алгоритми та формули (20) та (21) для обчислення градієнту функціоналу, то очевидно ефективним методом для його мінімізації буде якийсь із градієнтних методів мінімізації. Однією із проблем в даному випадку є вибір оптимального крокового множника на кожному ітераційному кроці. Розглянемо вирішення цієї проблеми на прикладі координації двух систем лінійних рівнянь.

Нехай ми вже маємо значення вектора-координатора $\boldsymbol{q_n}$ наших систем лінійних рівнянь(11) та (12) на n-тому кроці оптимізації. Позначимо градієнт функціоналу (13) через $\boldsymbol{p}_n$

$$\boldsymbol{p}_n = \left(\frac{\partial J(\boldsymbol{q_n})}{\partial q_1}, \ldots, \frac{\partial J(\boldsymbol{q_n})}{\partial q_N}\right) \tag{22}$$

тоді наступне значення вектора-координатора $\boldsymbol{q_{n+1}}$ будемо знаходити за формулою

$$\boldsymbol{q_{n+1}} = \boldsymbol{q_n} - \alpha \boldsymbol{p}_n \ , \ \alpha > 0 \qquad (23)$$

Тоді

$$J(\boldsymbol{q_{n+1}}) = \frac{1}{2} \| \boldsymbol{u}(\boldsymbol{q_{n+1}}) - \boldsymbol{v}(\boldsymbol{q_{n+1}}) \|^2_{\mathbb{R}^N} =$$

$$= \frac{1}{2} \| \boldsymbol{u}(\boldsymbol{q_n} - \alpha \boldsymbol{p}_n) - \boldsymbol{v}(\boldsymbol{q_n} - \alpha \boldsymbol{p}_n) \|^2_{\mathbb{R}^N} \qquad (24)$$

Так як матриці $[\mathbf{A}]$ та $[\mathbf{B}]$ додатньовизначені для них існують відповідні обернені матриці $[\mathbf{A}]^{-1}$ та $[\mathbf{B}]^{-1}$. Запишемо $\boldsymbol{u}(\boldsymbol{q_n} - \alpha \boldsymbol{p}_n)$ та $\boldsymbol{v}(\boldsymbol{q_n} - \alpha \boldsymbol{p}_n)$ наступним чином

$$\boldsymbol{u}(\boldsymbol{q_n} - \alpha \boldsymbol{p}_n) = [\mathbf{A}]^{-1} \boldsymbol{u}(\boldsymbol{q_n} - \alpha \boldsymbol{p}_n) + [\mathbf{A}]^{-1} \mathbf{FX} \qquad (25)$$

та

$$\boldsymbol{v}(\boldsymbol{q_n} - \alpha \boldsymbol{p}_n) = [\mathbf{B}]^{-1} \boldsymbol{v}(\boldsymbol{q_n} - \alpha \boldsymbol{p}_n) + [\mathbf{B}]^{-1} \mathbf{FY} \qquad (26)$$

Введемо допоміжні вектори $\boldsymbol{\varphi}^n$ та $\boldsymbol{\psi}^n$, які є розв"язками наступних систем лінійних рівнянь

$$[\mathbf{A}] \boldsymbol{\varphi}^n = \boldsymbol{p}_n \quad \text{та} \quad [\mathbf{B}] \boldsymbol{\psi}^n = \boldsymbol{p}_n \qquad (27)$$

Тоді останній член рівняння (24) набуває виду

$$\frac{1}{2} \| \boldsymbol{u}(\boldsymbol{q_n} - \alpha \boldsymbol{p}_n) - \boldsymbol{v}(\boldsymbol{q_n} - \alpha \boldsymbol{p}_n) \|^2_{\mathbb{R}^N} =$$

$$= \frac{1}{2} \| \boldsymbol{u}(\boldsymbol{q_n}) - \boldsymbol{v}(\boldsymbol{q_n}) - \alpha (\boldsymbol{\varphi}^n - \boldsymbol{\psi}^n) \|^2_{\mathbb{R}^N} =$$

$$= \frac{1}{2}\sum_{i=1}^{N}[(u_i^n - v_i^n) - \alpha(\varphi_i^n - \psi_i^n)]^2 \quad (28)$$

В рівнянні (28) ми отримали функціонала J від α. Взявши похідну від функціоналу по α та прирівнявши отриманий вираз до нуля ми зможемо знайти шукане значення шагового множника.

Отож

$$J'(\alpha) = \sum_{i=1}^{N}[(u_i^n - v_i^n) - \alpha(\varphi_i^n - \psi_i^n)] \quad (\psi_i^n - \varphi_i^n) = 0 \quad (29)$$

Звідки отримуємо значення α

$$\alpha = \frac{\sum_{i=1}^{N}[(u_i^n - v_i^n)(\varphi_i^n - \psi_i^n)]}{\sum_{i=1}^{N}(\varphi_i^n - \psi_i^n)^2} = \frac{\langle \boldsymbol{u}^n - \boldsymbol{v}^n, \boldsymbol{\varphi}^n - \boldsymbol{\psi}^n \rangle}{\| \boldsymbol{\varphi}^n - \boldsymbol{\psi}^n \|_{\mathbb{R}^N}^2} \quad (30)$$

5.Програмна реалізація деяких аспектів запропонованого метода розпаралелення обчислень

5.1Задача координації двух систем лінійних рівнянь методом мінімізації фукціоналу в нормі простору R^N. Програма coordsys2.py

Координуються дві симметричні додатньовизначені системи лінійних алгебраїчних рівнянь. Розмірність систем задається параметром **NM** у середині програми Розрахунки проводились для значення параметру **NM** = 10. На ітерації 0 виводяться початкові значення координуючого вектора і відповідне йому значення функціоналу (13). Це вектор QSTART та скалярна величина FUNC. На кожному ітераційному кроці виводяться значення функціоналу FUNC та параметру ALFA який визначається співвідношенням (30). Програма закінчила роботу на 26 ітерації, коли значення функціоналу (13) стало меньше < 1.e-20. Післяцього булі виведені значення скоординованих розв”язків систем лінійних рівнянь – вектори X1 та X2, а також кінцеве значення координуючого вектора – вектор QNEW.

ITERATION = 0
QSTART = [25. 25. 25. 25. 25. 25. 25. 25. 25. 25.]
FUNC = 0.43106360312369507

FUNC = 0.08765845860220832 ALFA = 14.033553705198877 ITERATION = 1
FUNC = 0.055121203222138745 ALFA = 3.7070524855401463 ITERATION = 2
FUNC = 0.012007993702326778 ALFA = 3.7727907613134097 ITERATION = 3
FUNC = 0.004151403485464368 ALFA = 19.587019904145507 ITERATION = 4
FUNC = 0.0007245979652935556 ALFA = 5.462055798598421 ITERATION = 5
FUNC = 0.0004615436415633456 6 ALFA = 4.643681006032118 ITERATION = 6
FUNC = 9.682217027086367e-05 ALFA = 3.7021943034338407 ITERATION = 7
FUNC = 5.2845119896304344e-05 ALFA = 67.9927599195488 ITERATION = 8
FUNC = 4.0356777621272936e-06 ALFA = 4.935432904693607 ITERATION = 9
FUNC = 9.55506732062317e-07 ALFA = 4.03134424088514 ITERATION = 10
FUNC = 1.1342951777452956e-07 ALFA = 6.472295355437442 ITERATION = 11
FUNC = 3.8301209484759355e-08 ALFA = 3.801298304327846 ITERATION = 12
FUNC = 3.831574559572521e-09 ALFA = 5.117892157335684 ITERATION = 13
FUNC = 9.72975370963727e-10 ALFA = 7.506364848091667 ITERATION = 14
FUNC = 4.71365360195449e-10 ALFA = 78.95475891028649 ITERATION = 15
FUNC = 1.0789470654162265e-11 ALFA = 11.076001304561071 ITERATION = 16

FUNC = 7.744628564545642e-13 ALFA = 6.700479912306468 ITERATION = 17
FUNC = 3.512713324283265e-14 ALFA = 6.556904553362199 ITERATION = 18
FUNC = 5.583385316140553e-15 ALFA = 30.05896683163577 ITERATION = 19
FUNC = 1.378510167130789e-15 ALFA = 4.8092664352541625 ITERATION = 20
FUNC = 1.2916269884165747e-16 ALFA = 6.0250073875838295 ITERATION = 21
FUNC = 7.344946665211851e-17 ALFA = 3.70945607842233 ITERATION = 22
FUNC = 1.3759148055332914e-17 ALFA = 3.7088055761263683 ITERATION = 23
FUNC = 1.5461740265835085e-18 ALFA = 3.7798150665659738 ITERATION = 24
FUNC = 2.0170394506652053e-19 ALFA = 6.649253868303683 ITERATION = 25
FUNC = 1.583754707191159e-20 ALFA = 6.8390047070461915 ITERATION = 26

X1 = [2.96265416 1.84831991 0.99051933 1.19381638 1.14807454 1.14807454 1.19381638 0.99051933 1.84831991 2.96265416]

X2 = [2.96265416 1.84831991 0.99051933 1.19381638 1.14807454 1.14807454 1.19381638 0.99051933 1.84831991 2.96265416]

QNEW = [26.50991064 25.14794656 28.9420626 30.18443342 29.90489998 29.90489998 30.18443342 28.9420626 25.14794656 26.50991064

Отримані результати обчислень веріфікують правильність розробленого алгоритму.

5.2.Розв’язок рівняння Лапласа. Програма main.py

На мові програмування Python була написана програма розрахунку розподілу температур в прямокутній пластині. Лінійні розміри : по вісі X та

по вісі Y задаються параметрами **ALX** та **ALY** в середині програми. В програмі також задається кількість точок апроксимації по вісі X та по вісі Y параметрами **MX** та **NY** . На верхній границі підтримується задана температура, розподіл якої задається функцією **fii** На нижній границі підтримується задана температура, розподіл якої задається функцією **psi.**

Були знайдені числові розв'язки цієї задачі при **fii = cos(x)** та **psi = sin(x)** з максимальними значеннями **TU = 60** та **TD =20**. Тобто, на вехній границі значення температури змінюється по границі з +60 у лівому верхньому куті і до -60 у правому лівому вехньому куті. На нижній границі від 0 до 0 з максимумом 20 у середині нижньої границі прямокутника.

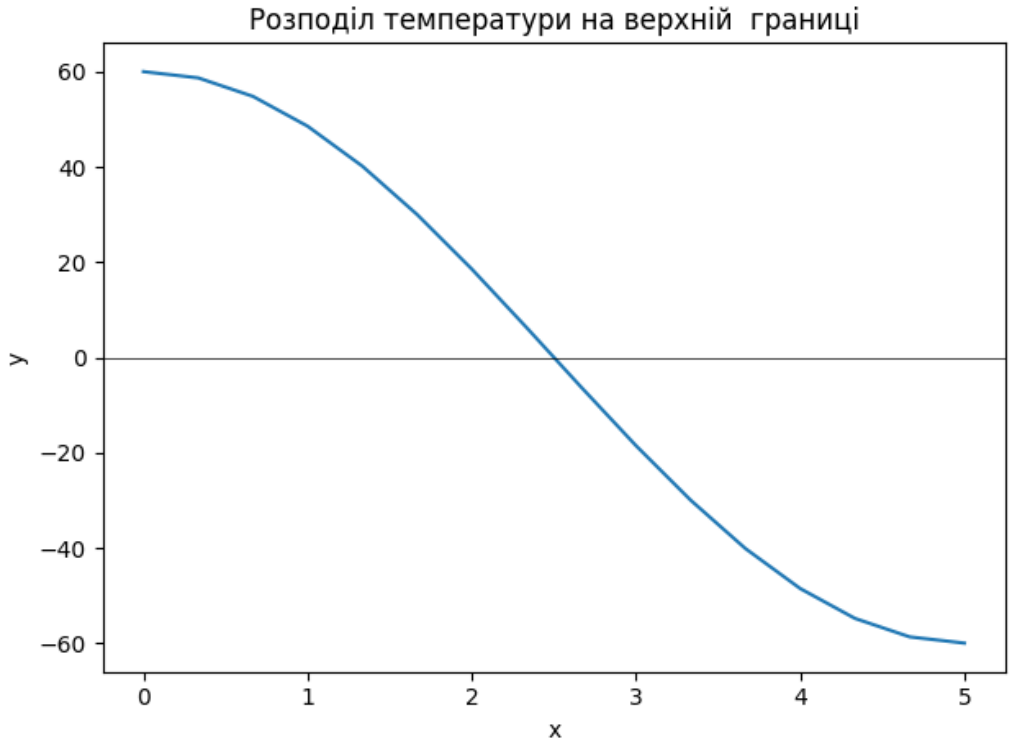

Рис. 3


Розподіл температури на нижній границі
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
y
0
1
2
3
4
5
x

Рис. 4

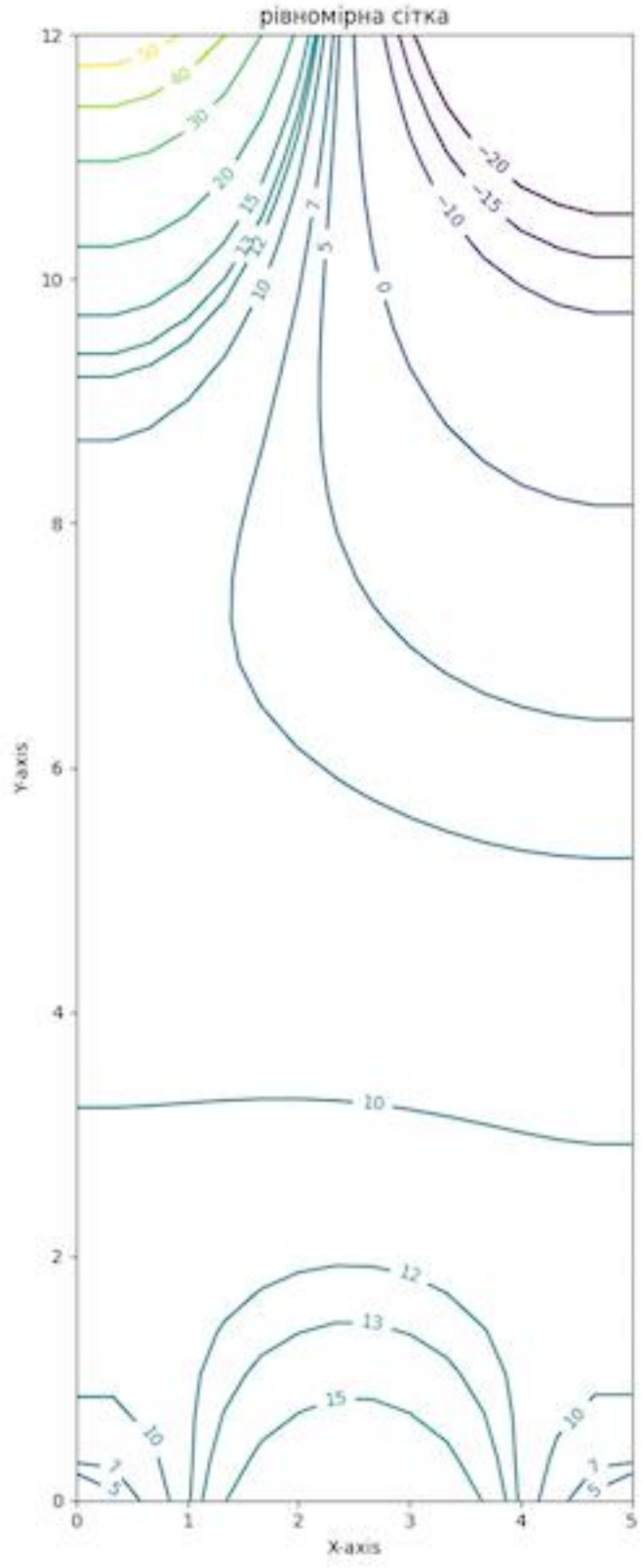

Рис 5 -Розподіл температур в середині прямокутника

Відповідні числові значення представлені нижче:

```
MX = 16
NY = 37
TU = 60.
TD = 20.
```

```
[ 60.   58.69  54.81  48.54  40.15  30.   18.54   6.27  -6.27 -18.54
 -30.   -40.15 -48.54 -54.81 -58.69 -60.  ]
[ 46.6   46.6   43.94  39.13  32.5   24.42  15.25   5.42  -4.64 -14.47
 -23.64 -31.73 -38.35 -43.16 -45.82 -45.82]
[ 37.32  37.32  35.35  31.62  26.41  19.98  12.66   4.8   -3.25 -11.11
 -18.43 -24.85 -30.07 -33.79 -35.76 -35.76]
[ 30.12  30.12  28.61  25.7   21.59  16.48  10.65   4.38  -2.05  -8.32
 -14.15 -19.26 -23.37 -26.28 -27.79 -27.79]
[ 24.53  24.53  23.35  21.06  17.81  13.76   9.12   4.12  -1.01  -6.01
 -10.65 -14.7  -17.96 -20.24 -21.42 -21.42]
[ 20.18  20.18  19.25  17.45  14.87  11.66   7.97   3.98  -0.1   -4.08
  -7.77 -10.99 -13.57 -15.37 -16.3  -16.3 ]
[ 16.81  16.81  16.08  14.66  12.61  10.06   7.13   3.96   0.71  -2.47
  -5.4   -7.95 -10.   -11.42 -12.15 -12.15]
[14.22 14.22 13.64 12.51 10.89  8.87  6.53  4.01  1.43 -1.1  -3.43 -5.46
 -7.08 -8.21 -8.79 -8.79]
[12.25 12.25 11.79 10.89  9.6   7.99  6.14  4.13  2.08  0.07 -1.78 -3.39
 -4.68 -5.57 -6.03 -6.03]
[10.76 10.76 10.39  9.68  8.66  7.38  5.91  4.31  2.68  1.08 -0.39 -1.67
 -2.69 -3.4  -3.77 -3.77]
[ 9.65  9.65  9.36  8.8   7.99  6.97  5.8   4.53  3.23  1.97  0.8  -0.22
 -1.03 -1.6  -1.89 -1.89]
[ 8.86  8.86  8.63  8.18  7.53  6.73  5.8   4.79  3.76  2.75  1.82  1.01
  0.36 -0.08 -0.31 -0.31]
[8.3  8.3  8.12 7.76 7.25 6.61 5.87 5.07 4.25 3.45 2.71 2.07 1.56 1.2
 1.02 1.02]
[7.94 7.94 7.8  7.52 7.11 6.6  6.01 5.37 4.72 4.09 3.5  2.99 2.58 2.3
 2.15 2.15]
[7.74 7.74 7.62 7.4  7.07 6.67 6.2  5.7  5.18 4.67 4.21 3.8  3.48 3.25
 3.14 3.14]
[7.65 7.65 7.56 7.38 7.13 6.8  6.43 6.03 5.62 5.22 4.85 4.52 4.27 4.09
 4.   4.  ]
[7.67 7.67 7.59 7.45 7.25 6.99 6.7  6.38 6.05 5.73 5.44 5.18 4.97 4.83
 4.76 4.76]
[7.75 7.75 7.7  7.58 7.42 7.22 6.99 6.73 6.47 6.22 5.98 5.78 5.62 5.5
 5.45 5.45]
```

[7.9 7.9 7.86 7.77 7.64 7.48 7.3 7.1 6.89 6.69 6.5 6.34 6.21 6.12
 6.07 6.07]
[8.1 8.1 8.07 8.   7.9 7.77 7.62 7.47 7.3 7.14 6.99 6.86 6.76 6.68
 6.64 6.64]
[8.34 8.34 8.31 8.26 8.18 8.08 7.97 7.84 7.71 7.58 7.46 7.35 7.27 7.21
 7.18 7.18]
[8.6 8.6 8.58 8.54 8.48 8.41 8.32 8.22 8.12 8.01 7.91 7.83 7.76 7.71
 7.68 7.68]
[8.89 8.89 8.87 8.85 8.8 8.75 8.68 8.6 8.52 8.44 8.36 8.28 8.22 8.18
 8.16 8.16]
[9.19 9.19 9.18 9.16 9.14 9.1 9.05 9.   8.93 8.86 8.79 8.73 8.67 8.63
 8.61 8.61]
[9.5 9.5 9.5 9.49 9.48 9.47 9.44 9.4 9.34 9.28 9.22 9.16 9.1 9.06
 9.04 9.04]
[9.81 9.81 9.82 9.83 9.84 9.84 9.83 9.81 9.77 9.71 9.65 9.58 9.52 9.47
 9.45 9.45]
[10.12 10.12 10.13 10.16 10.2 10.23 10.24 10.24 10.2 10.15 10.08 10.
  9.92 9.86 9.83 9.83]
[10.4 10.4 10.44 10.5 10.57 10.63 10.68 10.69 10.66 10.6 10.51 10.41
 10.3 10.22 10.18 10.18]
[10.65 10.65 10.71 10.82 10.94 11.06 11.15 11.19 11.17 11.09 10.96 10.81
 10.66 10.54 10.48 10.48]
[10.84 10.84 10.94 11.11 11.32 11.52 11.67 11.74 11.73 11.62 11.44 11.22
 11.   10.81 10.7 10.7]
[10.93 10.93 11.09 11.37 11.71 12.03 12.27 12.4 12.38 12.23 11.97 11.63
 11.28 10.99 10.82 10.82]
[10.84 10.84 11.11 11.58 12.11 12.61 12.99 13.19 13.19 12.97 12.57 12.06
 11.51 11.04 10.76 10.76]
[10.46 10.46 10.94 11.7 12.56 13.33 13.91 14.21 14.21 13.89 13.3 12.52
 11.65 10.88 10.41 10.41]
[9.58 9.58 10.45 11.73 13.08 14.26 15.11 15.56 15.56 15.1 14.24 13.06
 11.7 10.42 9.55 9.55]
[7.79 7.79 9.52 11.7 13.79 15.52 16.75 17.39 17.39 16.75 15.51 13.77
 11.68 9.51 7.77 7.77]
[0.   4.16 8.13 11.76 14.86 17.32 19.02 19.89 19.89 19.02 17.32 14.86
 11.76 8.13 4.16 0.]

Розрахунок температури пластини буде у подальших дослідженнях використовуватися як деякий результат, який потрібно буде отримати при розв’язуванні цієї задачі координацією одновимірних задач.

5.3.Модифікация метода квадратного кореня для пошуку розв'язків систем лінійних рівнянь з 3-х діагональною матрицею. Програма KHOLETSKY.py

Програма шукає розв”язки 3-х діагональних систем лінійних рівнянь з симетричними додатньовизначеними матрицями. Діагоналі матриці зберігаються у виді векторів **A1, A2** –діагоналі, **B –** вектор правих частин

ВИСНОВКИ

В даній роботі запропонован та частково реалізован новий підхід розробки алгоритмів паралельних обчислень для чисельних розв'язків багатовимірних задач математичної фізики. Метод базується на ідеї заміни пошуку чисельних розв'язків первинного рівняння на дво- чи тривимірній сітці апроксимації, пошукам розв'язків серії одновимірних задач з наступною їх координацією.

На першому етапі, що власне і є ця робота, проведена декомпозіція вихідного рівняння Лапласа у двовимірній області до серії одновимірних рівнянь з фукцією-координатором у правій частині цих рівнянь.

Проведена постановка задачі координації розв"язків незалежних одна від іншої сукупностей систем лінійних рівнянь через задачу мінімізації функціонала в нормі простору R^N.

Для задачі координаціх 2-х систем лінійних рівнянь загального вигляду получена формула для обчислення градієнту мінімізуємого функціоналу.

Розроблена процедура находження оптимального щагового множника для використовуємого градієнтного метода.

Проведени чисельні експерименти координації 2-х систем лінійних рівнянь по розробленій в даній роботі методиці. В результаті отримана швидка збіжність пошуку рішень.

Програмно реалізована задача розв"язку рівняння Лапласа в прямокутній області.

Модифікован метод квадратного кореня для пошуку розв"язків систем лінійних рівнянь із зберіганням вихідної матриці у векторному вигляді.

Проведена робота показує перспективність продовження подальших досліджень у цьому напрямку.

При проведенні подальших досліджень треба буде отримати співвідношення для обчислення градієнта функціонала при координації

совокупності великої кількості лінійних рівнянь та підрахунку оптимального крокового множника у цьому випадку. При написанні програм треба буде орієнтуватися на проведенні розрахунків на графічних процессорах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1.S.I.Lyashko Generalized Optimal Control of Linear Systems with Distributed Parameters.// Kluwer Academic Publishers.2002. 464p. DOI10.1007/b130433

2.Лионс Ж.-Л., Марчук Г.И. Методы вычислительной математики, Новосибирск, Наука, Сибирское отделение, 1975 г. 280 с.

3.Васильев Ф.П. Численные методы решения экстремальных задач. Москва: Наука, 1980

4.https://de.wikipedia.org/wiki/General_Purpose_Computation_on_Graphics_Processing_Unit

5.CUDA C++ Programming Guide. [Электронный ресурс] // Документация по CUDA Toolkit. URL:

6.Ортега Дж. (1991) Введение в параллельные и векторные методы решения линейных систем. - М: Мир, 1991. - 367 с. Ортега Дж., Введение в параллельные методы решения линейных систем, М.: Мир, 1991.

7.Поттер Д., Вычислительные методы в физике, Москва: Мир, 1975, 392 с.

8.Sorokin, M. and others : High performance computing of waves, currents and contaminants in rivers and coastal areas of seas on multi-processors systems and GPUs, EGU General Assembly 2020, Online, 4–8 May 2020, EGU2020-11372, https://doi.org/10.5194/egusphere-egu2020-11372, 2020.

9.Тарнавский Г.А., Тарнавский А.Г., Мультипроцессорное компьютерное моделирование в гравитационной газовой динамике, Вычислительные методы и программирование, 2005, т.6, с.71-87.

.

10.Foster I. (1995) Designing and Building Parallel Programs. Addison-Wesley, 95.

11.Dongarra J., et al., Sourcebook of parallel computing, Elsevier Science, 2003

ДОДАТОК

Програма coordsys2.py

```
import numpy as np
from math import sqrt
from scipy.linalg import solve_triangular


def func(u, v, nm):
    fu: float = 0.
    for i in range(0, nm):
        a = (u[i] - v[i])
        fu = fu + a * a
    fu = fu / 2.
    return fu


NM: int = 10

A1 = np.zeros([NM, NM])
B1 = np.zeros([NM])

A2 = np.zeros([NM, NM])
B2 = np.zeros([NM])

Q = np.zeros([NM])
BPR = np.zeros([NM])
GR = np.zeros([NM])
GR1 = np.zeros([NM])

D = np.zeros([NM])
D1 = np.zeros([NM])


for i in range(0, NM):
    A1[i][i] = 25.
    B1[i] = 75.
    A2[i][i] = 55.
    B2[i] = 59.

for i in range(0, 2):
    A1[i][i] = 12.
    A2[i][i] = 22.
    A1[NM - i - 1][NM - i - 1] = 12.
    A2[NM - i - 1][NM - i - 1] = 22.

for i in range(0, NM - 1):
    A1[i][i + 1] = 7.
    A2[i][i + 1] = 11.
    A1[i + 1][i] = A1[i][i + 1]
    A2[i + 1][i] = A2[i][i + 1]

    # print(A1)
    # print(B1)
    # print(A2)
```

```
    # print(B2)

for i in range(0, NM ):
    Q[i] = 25.



L = np.linalg.cholesky(A1)
Y = solve_triangular(L, B1 - Q, lower=True, check_finite=False)
X1 = solve_triangular(L, Y, lower=True, trans=1, check_finite=False)

L = np.linalg.cholesky(A2)
Y = solve_triangular(L, B2 + Q, lower=True, check_finite=False)
X2 = solve_triangular(L, Y, lower=True, trans=1, check_finite=False)

jj: int = 0

print('FUNC =   ', func(X1, X2, NM), 'ITERATION =', jj)

L = np.linalg.cholesky(A1)
Y = solve_triangular(L, -(X1 - X2), lower=True, check_finite=False)
FIX = solve_triangular(L, Y, lower=True, trans=1, check_finite=False)

L = np.linalg.cholesky(A2)
Y = solve_triangular(L, X1 - X2, lower=True, check_finite=False)
FIY = solve_triangular(L, Y, lower=True, trans=1, check_finite=False)

GR = FIX - FIY

amod: float = 0.

for i in range(0, NM):
    amod = amod + GR[i] * GR[i]

ALFA: float = func(X1, X2, NM) / amod

Q = Q - ALFA*GR

for j in range(1, 300):

    for i in range(0, NM):
        GR1[i] = GR[i]

    L = np.linalg.cholesky(A1)
    Y = solve_triangular(L, B1 - Q, lower=True, check_finite=False)
    X1 = solve_triangular(L, Y, lower=True, trans=1, check_finite=False)

    L = np.linalg.cholesky(A2)
    Y = solve_triangular(L, B2 + Q, lower=True, check_finite=False)
    X2 = solve_triangular(L, Y, lower=True, trans=1, check_finite=False)

    print('FUNC =   ', func(X1, X2, NM), 'ITERATION =', jj)

    if (func(X1, X2, NM) < 1.e-20):
        break

    L = np.linalg.cholesky(A1)
    Y = solve_triangular(L, -(X1 - X2), lower=True, check_finite=False)
    FIX = solve_triangular(L, Y, lower=True, trans=1, check_finite=False)

    L = np.linalg.cholesky(A2)
    Y = solve_triangular(L, X1 - X2, lower=True, check_finite=False)
    FIY = solve_triangular(L, Y, lower=True, trans=1, check_finite=False)
```

```
    GR = FIX - FIY

    amod: float = 0.
    amod1: float = 0.
    for i in range(0, NM):
        amod = amod + GR[i] * GR[i]
        amod1 = amod1 + GR1[i] * GR1[i]

    D = -GR + (amod/amod1) * D1
    ALFA = func(X1, X2, NM) / amod
    Q = Q + ALFA * D

    print(' ALFA =', ALFA)



print(' X1 = ', X1)
print(' X2 = ', X2)
print(' QNEW =', Q)
```

Програма main.py

```
import matplotlib.pyplot as plt
import numpy as np
from math import sin, cos, pi
from scipy.linalg import solve_triangular
from matplotlib import rcParams

def fii(alx, tu, x):
    fi: float = tu * cos(x * pi / alx)
    return fi

def psii(alx, td, x):
    psi: float = td * sin(x * pi / alx)
    return psi


ALX = 5.
ALY = 12.
MX = 16
NY = 36
TU = 60.
TD = 20.

DX = ALX / (MX - 1)
DY = ALY / (NY - 1)

start = 0

x_vals = np.arange(start, ALX + DX, DX)
y_vals = np.arange(start, ALY + DY, DY)

XPLOT, YPLOT = np.meshgrid(x_vals, y_vals)

print(np.round(XPLOT, 2))
print(np.round(YPLOT, 2))
```

```
fig, ax = plt.subplots()

ax.scatter(XPLOT, YPLOT, color="black")
ax.set_title('рівномірна сітка')
rcParams['figure.figsize'] = 5, 12
#plt.figure(figsize=(5,1))
plt.show()


# ALX = float(input("Enter LX :  "))
# ALY = float(input("Enter LY :  "))
# MX = int(input("Enter MX  :  "))
# NY = int(input("Enter NY  :  "))
# TU = float(input("Enter max temperature on upper  :  "))
# TD = float(input("Enter max temperature on down   :  "))

NM = (MX - 2) * (NY - 2)
AA = np.zeros([NM, NM])
BR = np.zeros([NM])
ZZ = np.zeros([NY, MX])
VECEXIT = np.zeros([MX])

DXDX = DX * DX
DYDY = DY * DY

# Головна діагональ

for i in range(0, NY - 2):
    AA[i][i] = 2. * DXDX + DYDY

for i in range(NY - 2, (NY - 2) * (MX - 3)):
    AA[i][i] = 2. * (DXDX + DYDY)
for i in range((MX - 3) * (NY - 2), (MX - 2) * (NY - 2)):
    AA[i][i] = 2. * DXDX + DYDY

# Дальня діагональ

for i in range(0, (MX - 3) * (NY - 2)):
    j = i + NY - 2
    AA[i][j] = - DYDY
    AA[j][i] = AA[i][j]

# Ближня діагональ

for i in range(0, (MX - 2) * (NY - 2) - 1):
    AA[i][i + 1] = -DXDX
    AA[i + 1][i] = AA[i][i + 1]

for i in range(1, MX - 2):
    j = (NY - 2) * i -1
    AA[j][j + 1] = 0.
    AA[j+1][j] = 0.

# Формування вектора правої частини

for i in range(0,NM):
    BR[i] = 0.

for i in range(0,MX-2):
    j = (NY-2)*i
    X = DX*(i+1)
    BR[j] = fii(ALX, TU, X)*DXDX
    BR[j+NY-3] = psii(ALX, TD, X) * DXDX
```

```
L = np.linalg.cholesky(AA)
Y = solve_triangular(L, BR, lower = True, check_finite = False)
XX = solve_triangular(L, Y, lower = True, trans = 1, check_finite = False)

for i in range(0,MX):
    XC = i*DX
    VECEXIT[i] = fii(ALX, TU, XC)
    ZZ[NY-1, i] = VECEXIT[i]
print(np.round(VECEXIT, 2))

for j in range(2, NY):
    for i in range(1,MX-1):
        VECEXIT[i] = XX[j-2 + (i-1)*(NY-2)]
        ZZ[NY-j, i] = VECEXIT[i]
    VECEXIT[0] = VECEXIT[1]
    VECEXIT[MX-1] = VECEXIT[MX-2]
    ZZ[NY-j, 0] = VECEXIT[0]
    ZZ[NY-j, MX-1] = VECEXIT[MX-2]
    print(np.round(VECEXIT, 2))

for i in range(0,MX):
    X = i*DX
    VECEXIT[i] = psii(ALX, TD, X)
    ZZ[0, i] = VECEXIT[i]
print(np.round(VECEXIT, 2))


fig = plt.figure()
ax = fig.add_subplot(111)
ax.set_title('рівномірна сітка')
ax.set_xlabel('X-axis')
_ = ax.set_ylabel('Y-axis')


levels = np.array([-20,-15.,-10,0.,5.,7.,10.,12.,13., 15.,20.,30.,40.,50.])
cp = ax.contour(XPLOT, YPLOT, ZZ, levels=levels)
ax.clabel(cp, fontsize=10)

plt.show()
```

Програма KHOLETSKY.py

```
import numpy as np
from math import sqrt

# МЕТОД ХОЛЕЦЬКОГО ДЛЯ 3-Х ДІАГОНАЛЬНОЇ МАТРИЦІ

NM: int = 10

A1 = np.zeros([NM])
A2 = np.zeros([NM - 1])
B = np.zeros([NM])
```

```
L1 = np.zeros([NM])
L2 = np.zeros([NM - 1])

X = np.zeros([NM])
Y = np.zeros([NM])

AA1 = np.zeros([NM])
AA2 = np.zeros([NM - 1])

for i in range(0, NM):
    A1[i] = 25.
    X[i] = 2.

for i in range(0, 2):
    A1[i] = 12.
    A1[NM-i-1] = 12.

for i in range(0, NM - 1):
    A2[i] = 7.

B[0] = A1[0]*X[0] + A2[0]*X[1]

for i in range(1, NM - 1):
    B[i] = A2[i - 1]*X[i - 1] + A1[i]*X[i] + A2[i]*X[i + 1]

B[NM - 1] = A2[NM - 2]*X[NM - 2] + A1[NM - 1]*X[NM - 1]

print('A1 = ', A1)
print('A2 = ', A2)

print('B = ', B)

# ТРЕГОЛЬНИЙ РОЗКЛАД

L1[0] = sqrt(A1[0])

for i in range(1, NM):
    L2[i - 1] = A2[i - 1]/L1[i - 1]
    L1[i] = sqrt(A1[i] - L2[i - 1]*L2[i - 1])

print('L1 = ', L1)
print('L2 = ', L2)


# РОЗВ"ЯЗОК

# ПРЯМИЙ ХІД

X[0] = B[0]/L1[0]

for i in range(1, NM):
    X[i] = (B[i] - L2[i - 1]*X[i -1])/L1[i]

# ОБРАТНІЙ ХІД

Y[NM - 1] = X[NM - 1]/L1[NM - 1]

for j in range (1, NM):
    i = NM - j - 1
    Y[i] = (X[i] - L2[i]*Y[i + 1])/L1[i]
```

```
print('X = ', Y)

# МНОЖЕННЯ МАТРИЦЬ

AA1[0] = L1[0] * L1[0]

for i in range (1, NM):
    AA2[i - 1] = L2[i - 1]*L1[i - 1]
    AA1[i] = L2[i - 1]*L2[i - 1] + L1[i]*L1[i]

print('AA1 = ', AA1)
print('AA2 = ', AA2)
```