ДВНЗ «ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»

Факультет комп'ютерно-інформаційних технологій та автоматизації

Кафедра прикладної математики і інформатики

«До захисту допущено»

Завідувачка кафедри

_____ Ольга ДМИТРІЄВА

«___» _____ 2022 р.

Випускна кваліфікаційна робота
бакалавра
(освітній ступень)

на тему
«Паралельна реалізація методу прямих для рівнянь в частинних похідних колокаційними блоковими різницевими схемами»
зі спецчастиною
«Розробка програмної системи з варіацією розмірності блоків засобами C++, MPI»

Виконав: студент 4 курсу, групи КН-18

Спеціальності 122 Комп'ютерні науки

Рогоза М.С. _____
(прізвище та ініціали) (підпис)

Керівник каф. ПМІ, д.т.н., проф. Ольга ДМИТРІЄВА _____
(посада, науковий ступінь, вчене звання, ім'я та прізвище) (підпис)

Рецензент ав. каф. АТ, д.т.н., доц. Іван ЛАКТІОНОВ _____
(посада, науковий ступінь, вчене звання, ім'я та прізвище) (підпис)

Нормоконтроль:

_____ к.т.н., доц. Ірина НАЗАРОВА
(підпис)

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

Дата 1.06.2022

Студент _____
(підпис)

Луцьк – 2022 р.

ДВНЗ «Донецький національний технічний університет»
Факультет комп'ютерно-інформаційних технологій та автоматизації_____
Кафедра прикладної математики та інформатики_____
Освітньо-кваліфікаційний рівень (освітній ступінь) бакалавр_____
Напрям підготовки (спеціальність) 122 Комп'ютерні науки
(шифр і назва)

ЗАТВЕРДЖУЮ:
Завідувачка кафедри ПМІ
Ольга ДМИТРІЄВА
/_____/
"____" _____2022 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТУ

Михайлу Рогозі
1. Тема проєкту (роботи) «Паралельна реалізація методу прямих для рівнянь в частинних похідних колокаційними блоковими різницевими схемами»
Спецчастина «Розробка програмної системи з варіацією розмірності блоків засобами C++, MPI»

Керівник проєкту (роботи) Ольга ДМИТРІЄВА, д.т.н, проф., зав. каф. ПМІ
(ім'я, прізвище, науковий ступінь, вчене звання, посада)
затверджені наказом від "06" травня 2022 року №_180_____
2. Строк подання студентом проєкту (роботи)_____
3. Вихідні дані до проєкту (роботи) результати науково-дослідної роботи, матеріали переддипломної практики.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1) дослідження предметної області;
2) постановка проблеми;
3) опис методу розв'язання;
4) проєктування системи;
5) реалізація
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
1) екранні форми.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Підпис, дата	Підпис, дата

7. Дата видачі завдання_6 квітня 2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	22.04.22 – 24.04.22	
2	Постановка завдання	25.04.22 – 26.04.22	
3	Огляд тематики роботи, актуальність роботи, формулювання мети на задач дослідження	27.04.22 – 03.05.22	
4	Опис та обґрунтування основних ідей роботи	04.05.22 – 07.05.22	
5	Розробка алгоритмів	08.05.22 – 14.05.22	
6	Програмна реалізація, аналіз результатів дослідження, формулювання висновків та тенденцій	15.05.22 – 31.05.22	
7	Оформлення пояснювальної записки	01.06.22 – 07.06.22	
8	Нормоконтроль пояснювальної записки та додаткових матеріалів	08.06.22 – 12.06.22	
9	Рецензування роботи	13.06.22 – 15.06.22	
10	Захист роботи	22.06.22	

Студент _____ Михайло РОГОЗА
(підпис) (ім'я, прізвище)

Керівник роботи _____ Ольга ДМИТРІЄВА
(підпис) (ім'я, прізвище)

АНОТАЦІЯ

Михайло РОГОЗА. Паралельна реалізація методу прямих для рівнянь в частинних похідних колокаційними блоковими різницевими схемами / Випускна кваліфікаційна робота на здобуття освітньо-кваліфікаційного рівня «бакалавр» за спеціальністю 122 Комп’ютерні науки. – ДВНЗ ДонНТУ, Луцьк, 2022.

Об’єктом дослідження випускної кваліфікаційної роботи є процеси моделювання в комп'ютерних системах з базовими топологічними характеристиками динамічних об’єктів з розподіленими параметрами.

Предметом дослідження є паралельні чисельні методи розв'язання жорстких динамічних завдань великої розмірності, що описуються рівняннями з частинними похідними.

Мета роботи полягає у дослідженні алгоритмів керування кроком і порядком інтегрування при зведенні розв’язання жорстких еволюційних рівнянь до методу прямих з подальшою реалізацією на основі паралельних колокаційних блокових методів.

Методи досліджень базуються на основних положеннях теорії чисельних методів, теорії диференційних рівнянь з частинними похідними, теорії алгоритмів, об’єктно-орієнтованого програмування, теорії обчислювальних систем і паралельних обчислень.

В роботі проведено порівняльний аналіз сучасних методів і алгоритмів паралельного моделювання динамічних систем великої розмірності, теоретично обґрунтований вибір схем реалізації комунікаційних операцій управління кроком, досліджено особливості відображення модифікованих алгоритмів на топологічні поля паралельних обчислювальних систем.

Практична цінність роботи полягає в розробці консольного програмного забезпечення, що здійснює паралельну реалізацію з використанням бібліотеки передачі повідомлень MPI для підтримки паралельних обчислень.

Ключові слова: моделювання динамічних об’єктів, паралельна реалізація, метод прямих, рівняння з частинними похідними, блокові різницеві схеми, завдання Коші, керування кроком, MPI, C++

ANNOTATION

Mykhailo ROHOZA. Parallel implementation of the method of lines for partial differential equations using collocation block difference schemes / Graduate qualification work for bachelor's degree in 122 Computer Science of DonNTU, Pokrovsk, Lutsk, 2022.

The objects of study are the processes of modeling with basic topological characteristics of dynamic objects in computer systems with distributed parameters,

The subject of research is parallel numerical methods for solving rigid dynamic problems of large dimension, described by partial differential equations.

The purpose of the work is to study the algorithms for controlling the step and order of integration in transforming the solution of rigid evolutionary equations to the method of lines with subsequent implementation on the basis of parallel collocation block methods.

Research methods are based on the basic principles of the theory of numerical methods, the theory of partial differential equations, the theory of algorithms, object-oriented programming, the theory of computer systems and parallel computing.

The comparative analysis of modern methods and algorithms of parallel modeling of large-scale dynamic systems is carried out, the choice of schemes of realization of communication operations of step control is theoretically substantiated, features of display of modified algorithms on topological fields of parallel computer systems are investigated.

The practical importance of the work lies in the development of console software that performs parallel implementation using the message passing library to support parallel computing.

Keywords: modeling of dynamic objects, parallel implementation, method of lines, partial differential equations, block difference schemes, Cauchy problems, step control, MPI, C++

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ЗДР	звичайне диференційне рівняння
СЗДР	система звичайних диференційних рівнянь
ДРЧП	диференційні рівняння у частинних похідних
СЛАР	система лінійних алгебраїчних рівнянь
MPI	message passing interface
OpenMP	open multi-processing

ВСТУП

На сьогоднішній день моделювання поведінки динамічних об’єктів з розподіленими параметрами є досить актуальною темою. Майже у будь-якій предметній області можна знайти схожі приклади: більшість законів фізики та хімії, взаємодія частин великих систем у біології або економіці тощо[1-2]. У більшості випадків ці динамічні об’єкти задаються диференційними рівняннями. Кожна окрема проблема зводиться до різних видів рівнянь в залежності від умов цих динамічних об’єктів.

У даній роботі розглядаються диференційні рівняння у частинних похідних (ДРЧП). Аналіз проблеми розв’язання ДРЧП є досить актуальною проблемою, бо досі не існує оптимального методу розв’язання навіть вже відомих законів. Вибір найоптимальнішого методу залежить від типу рівняння, початкових умов, геометрії області розв’язання та багатьох інших характеристиках.

Існує багато підходів та методів вирішення ДРЧП. Аналітичний підхід використовується такими методами, як метод розділення змінних, метод характеристик, інтегральне перетворення тощо[3-4]. Чисельне вирішення задач використовують методи, серед яких є метод прямих, який і буде розглядатись в рамках даної роботи[5].

Метод прямих надає можливість зведення ДРЧП до задачі Коші[6]. Із використанням паралельного обчислення засобами бібліотеки MPI (Message Passing Interface) можна прискорити процес розрахунку[7]. Чим ефективнішим буде розділення задачі на підзадачі на процесорах, тим більше можна отримати прискорення. Тому необхідно проаналізувати як саме можна зробити розпаралелення і яку при цьому можна отримати швидкість обчислень. Із цього робиться висновок щодо обраного підходу та проводиться реалізація паралельного розрахунку із його використанням.

Перший розділ присвячений розгляданню математичної основи задачі – опису проблеми, методів розв’язання, ідеї методу прямих та інструментів і способів створення паралельної системи. У другому розділі описується проєктування системи, третій розділ присвячений програмній реалізації методи і четвертий – його тестуванню.

1 ІДЕЯ РОЗВ'ЯЗАННЯ ДИФЕРЕНЦІЙНИХ РІВНЯНЬ У ЧАСТИННИХ ПОХІДНИХ МЕТОДОМ ПРЯМИХ. ПІДХОДИ ДО РОЗПАРАЛЕЛЮВАННЯ ЗАДАЧІ

1.1 Опис задачі

При розв'язанні звичайних диференційних рівнянь (ЗДР) завданням є знайти таку функцію $y = y(x)$, яка задовольняє (1.1) та початковій умові виду $y(x_0) = y_0$.

$$F\left(\frac{d^n y}{dx^n}, \frac{d^{n-1} y}{dx^{n-1}}, \dots, \frac{dy}{dx}, x\right) = 0. \tag{1.1}$$

ЗДР є підвидом ДРЧП, бо у ньому частинні похідні представляються як звичайні похідні. Головною різницею між двома класами рівнянь є те, що ДРЧП мають хоча б дві незалежні змінні, тож загальний вигляд ДРЧП важко представити у вигляді формули. При порядку рівняння n кожна з частинних похідних (або їх комбінацій) не перевищує похідну того ж порядку.

ДРЧП другого порядку поділяються за своїми властивостями на еліптичні, параболічні та гіперболічні. Їх загальний вигляд можна представити у вигляді (1.2).

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu + G = 0, \tag{1.2}$$

де A, B, C, D, E, F, G – деякі функції від x та y, що є незалежними змінними для цього рівняння.

У літературі часто зустрічається позначення u_{xx} або u_t. Таке позначення ідентичне $\frac{\partial^2 u}{dx^2}$ та $\frac{\partial u}{\partial t}$ відповідно. Надалі буде використовуватися другий варіант.

У даній роботі будуть розглядатися саме параболічні ДРЧП другого порядку. Вони характерні тим, що мають похідну першого порядку за часом та до другого порядку у просторі.

Відповідно, завданням для розв'язання ДРЧП є знаходження такої функції $u(x_1, x_2, \dots, x_n)$, яка б задовольняла заданій початковій умові.

Для таких рівнянь аналітичні методи майже не використовуються через відсутність загального алгоритму розв'язання та потреби у точному розв'язку. Зазвичай необхідно одне конкретне рівняння із сімейства відповідей для даної задачі. Для його знаходження використовуються початкові умови виду (1.3) та граничні умови першого (1.3), другого (1.4) або третього роду (комбінація першого та другого)

$$u(x_1, x_2, \dots, x_0, \dots x_n) = f(x_1, x_2, \dots, x_n) \tag{1.3}$$

$$\frac{\partial u(x_1, x_2, \dots, x_0, \dots x_n)}{\partial x} = f(x_1, x_2, \dots, x_n) \tag{1.4}$$

У прикладних задачах у більшості випадків одна із змінних – часова, а інші – просторові.

1.2 Найпоширеніші методи для розв'язання ДРЧП

Як було сказано раніше, аналітичні підходи до розв'язку таких рівнянь не є найбільш підходящим вибором у прикладних задачах, але у випадку тривіальних рівнянь їх можна використовувати. Це методи розділення змінних, метод характеристик, інтегральне перетворення, заміна змінних, принцип суперпозиції та деякі інші [8-9].

Більш важливими є саме чисельні методи. Метод кінцевих елементів дозволяє розв'язувати рівняння з декількома просторовими змінними шляхом розділення системи на більш прості, «кінцеві елементи» [10]. Метод кінцевих різниць використовує заміну частинних похідних на алгебраїчний вираз, що

дозволяє більш просто представляти рівняння для подальшого програмування його розв’язку [11]. Ідея цього методу частково застосовується і в цій роботі. Також існують спектральний метод, сімейство методів декомпозиції області визначення та безсіткові методи [12-13].

У даній роботі розглядається метод прямих через зручність його використання у паралельній системі.

1.3 Метод прямих для розв’язання ДРЧП

Основна ідея методу полягає у зведенні ДРЧП до задачі Коші шляхом дискретизації однієї із змінних (змінну простору), звідки і назва методу – на графіку точки, що належать функції, будуть знаходитись на відповідних прямих[14]. Таким чином отримується система ЗДР, які розв’язуються іншими способами [15-16]. У даній роботі – це колокаційні блокові методи.

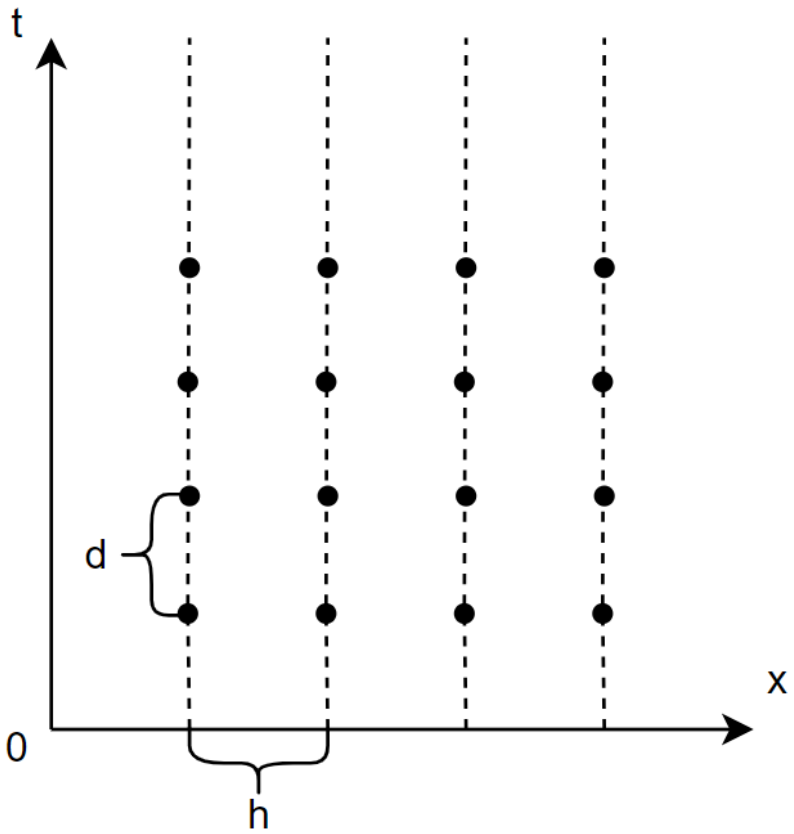


Рисунок 1.1 – Пояснення до методу прямих

Для отримання такої системи рівнянь частинні похідні змінних, що задані граничними умовами, замінюються на алгебраїчний вираз без похідної зазвичай за формулою скінченних різниць. У цій роботі буде використана центральна різниця (1.5), яка виділяється із означення похідної [17].

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \ . \qquad (1.5)$$

Крок h, що наведено у формулах, дискретизує рівняння за однією із змінних, і також використовує сусідні точки графіку для знаходження даних. Саме на цих точках встановлюються прямі (рис. 1.1). Рисунки були виконані за допомогою сервісу diagrams.net [18].

Якщо розглядати одновимірне параболічне рівняння, що має вигляд (1.6), де (1.5) використовується тільки один раз, то система ЗДР буде мати вигляд (1.7).

$$\begin{cases} \dfrac{\partial u}{\partial t} = \alpha \dfrac{\partial^2 u}{\partial x^2} + f(x,t); A \le x \le B, t > 0 \\ u(x,0) = f(x) \\ u(A,t) = u(B,t) = C \end{cases} \qquad (1.6)$$

$$\frac{du}{dt} = \frac{\alpha}{h^2}[u(i+h,t) - 2u(i,t) + u(i-h,t)] + f(x_i,t), \qquad (1.7)$$

де $u(x,t)$ – шукана функція,

h – крок розташування прямих,

i – координата простору поточної прямої.

Таким чином, отримані рівняння залежать тільки від часу (на рис. 1 це видно, як фіксована точка простору), тож це задача Коші. Кількість рівнянь у системі залежить від обраного кроку.

Важливо помітити, що формально у системі є ще два рівняння – граничні умови задачі, але при даному формулюванні вони є константами. У матричному вигляді ця система виглядає, як (1.8).

$$\begin{pmatrix} -2 & 1 & 0 & 0 & \dots & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 \\ 0 & 0 & 1 & -2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix} \tag{1.8}$$

Тоді вектор початкових умов буде рівний (1.9)

$$u_0 = \{f(A), f(A+h), \dots, f(B-h), f(B)\}. \tag{1.9}$$

Кожне рівняння відповідає одній прямій (рис. 1.1), а кожна пряма задає стан однієї точки одномірного об’єкту упродовж часу. Тож відповідь на (1.6) задається як вектор значень шуканого аргументу u у кожній точці просторової змінної у векторі значень часу. Це означає, що результат одновимірної проблеми представляється у трьох вимірах.

1.4 Інструменти для розробки паралельних обчислень

Основними напрямками розробки паралельних обчислень є використання розподіленої пам’яті та загальної пам’яті.

При використанні розподіленої пам’яті через те, що у кожного процесора є своя частина даних, доступ до цієї пам’яті відбувається швидко, однак передача даних між процесорами повільна. Такий підхід є доречним у тих випадках, коли кожен процесор повинен вирішити свою підзадачу і передати кінцевий результат один раз.

Процесори із спільною пам’яттю використовуються тоді, коли завдання передбачає наявність елементів, що часто змінюються протягом виконання алгоритму і потрібні для роботи кожної окремої групи.

Реалізація концепції розподіленої пам’яті в архітектурі комп’ютерів потребує особливих засобів. Такі засоби пропонує бібліотека MPI. Вона має в собі функції, за допомогою яких процесори можуть обмінюватися інформацією один з одним.

MPI підтримується такими мовами, як C/C++ та Fortran (не враховуючи адаптації) [19-20]. Робота буде виконуватися саме засобами C++.

Open Multi-Processing (OpenMP) використовується для реалізації архітектури комп’ютерів із спільною пам’яттю [21]. Через характеристики такого способу програмування OpenMP відрізняється від MPI можливостями та функціями, проте також підтримується тільки C, C++ і Fortran.

У цій роботі буде матися на увазі, що паралельна система містить у собі набір процесорів (вузлів), що напряму зв’язані між собою за якоюсь особливістю (топологія системи), і у кожному процесорі може бути деяка кількість ядер. Таким чином, велику задачу можна розподілити на декілька процесорів, у кожному з яких меншу задачу можна розподілити на ядра (гібридний спосіб розпаралелення).

1.5 Використання колокаційних блоків

Після отримання прямих для знаходження розв’язку ДРЧП на них необхідно кожен інтервал часу знайти своє окреме рішення для даного моменту в часі. Для цього вводяться колокаційні блоки [22].

Основна задача блоків – надати розв’язок на наступний момент часу (або декілька моментів), маючи інформацію про динамічний об’єкт у поточний момент часу (або поточний та декілька попередніх). Так як з умов задачі відома початкова умова за часом, то стан об’єкта на початку дослідження

також є відомим. Таким чином, можна послідовно знайти рішення у будь-який момент часу.

Такі колокаційні методи поділяють на однокрокові та багатокрокові. У процесі присутні опорний блок та розрахунковий блок (рис. 1.2). У випадку із однокроковими методами, опорних точок одна, а у багатокрокових – декілька. Очевидно, що багатокрокові методи дають більш точний результат, але потребують додаткових разрахунків.

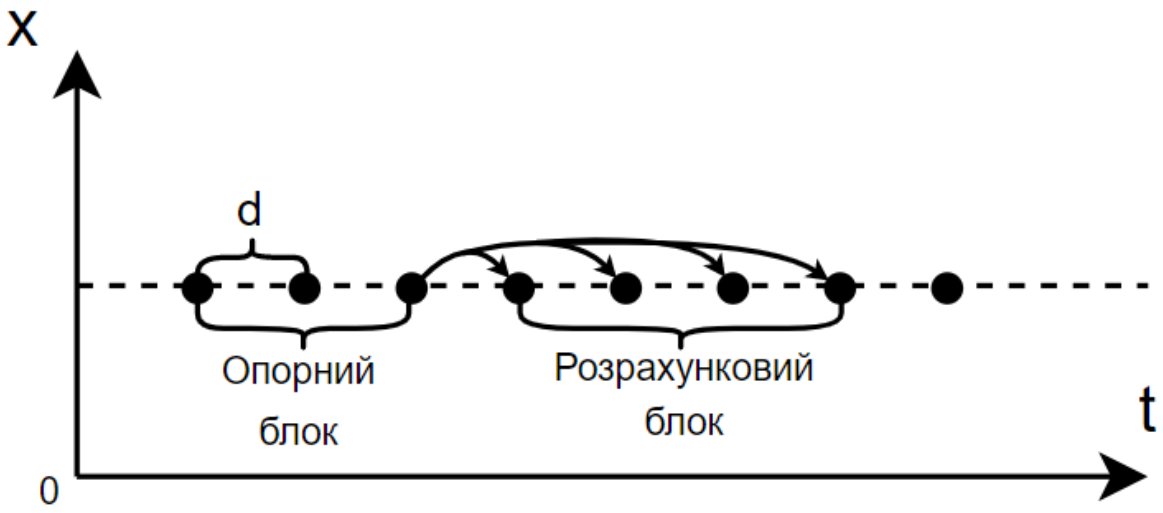


Рисунок 1.2 – Пояснення до колокаційних блоків

На рис. 1.2 зображена одна з прямих методу, але важливо пам’ятати, що сусідні прямі також впливають на стан об’єкта (наглядно видно на (1.7)). Кожна точка на рис. 1.2 означає момент часу для однієї точки динамічного одновимірного об’єкту.

Так як із умов задачі відомо, що $u(x,0) = f(x)$, то точка на початку кожної прямої у $t = 0$ буде мати значення $f(x)$, а тобто відоме значення, бо кожна пряма віддалена від попередньої на заздалегідь відому відстань h, яка може бути постійною або змінною.

Загальний канонічний вигляд колокаційних блокових методів представлено у вигляді (1.10):

$$u_{n,i} = u_{n,0} + d\left(\sum_{j=-(m-1)}^{0} b_{i,j}F_{n,j} + \sum_{j=1}^{s} a_{i,j}F_{n,j}\right), i = 1,2 \dots, s, \qquad (1.10)$$

де $u_{n,i}$ – шукані значення розв'язку у наступних точках колокаційного блоку $t_{n,i}$,

d – крок між точками,

$F_{n,j}$ – відповідна права частина рівняння, що отримується із задачі Коші,

$a_{i,j}$ та $b_{i,j}$ – коефіцієнти блоку,

s та m – число точок у розрахунковому та опорному блоках відповідно.

З цього рівняння невідомими є лише коефіцієнти блоку, які можуть бути визначені із умов апроксимації шляхом математичних перетворень[23]. Отримується система умов для встановлення числових значень коефіцієнтів (1.11).

$$\begin{gathered}\sum_{j=1-m}^{0} b_{i,j} + \sum_{j=1-m}^{0} b_{i,j} = i, i = 1,2, \dots, s, \\ \sum_{j=1-m}^{0} b_{i,j}j^{l-1} + \sum_{j=1}^{s} a_{i,j}j^{l-1} = \frac{i^{l-1}}{l}, l = 2,3, \dots, p.\end{gathered} \qquad (1.11)$$

Отримані коефіцієнти передаються у (1.10) для подальшого розрахунку точок в колокаційних блоках. Таким чином, отримується значення функції для кожного значення часу на кожній прямій – відповідь на початкову задачу.

1.6 Застосування паралельних обчислень для методу прямих

Для даного методу існує декілька способів застосування паралельних обчислень.

Перший з них – застосувати паралельну систему у процесі розрахунку колокаційних блокових схем. Так як кожна з прямих залежить від себе та двох безпосередніх сусідів, ці блоки можна розрахувати майже незалежно одні від одних. При такому підході, обмін інформацією між процесорами виникає тільки при розрахунку всього блока, що може значно зменшити час розрахунку блоків. Більш того, якщо одному процесору надати набір прямих, а його сусідам деякі прямі з цього ж набору, розрахунки стають набагато точніші через можливість порівняти значення на одній точці у багатьох процесорів. А враховуючи, що це виконується паралельно, не витрачається додаткового часу у порівнянні із знаходженням значень на прямих один раз. До того ж, цей спосіб можна використати, маючи будь-яку кількість процесорів. При зміні їх кількості буде змінюватися тільки кількість спільних прямих.

Другий підхід – розпаралелити знаходження коефіцієнтів для систем кожного блоку. Хоч розмірність таких систем співпадає з кількістю застосованих блоків, що в більшості випадків не перевищує 10-12 точок, на це витрачається час, який можна було би використати для розрахування коефіцієнтів іншого блоку.

Третій, та найбільш впливовий у скороченні часу – розпаралелення всього алгоритму. У більшості реальних задач шукана аналітична функція невідома, і тому немає змоги встановити порядок похибки. Щоб дані, отримані чисельними методами все ж можна було використати у якості відповіді на поставлену задачу, метод запускається два (при необхідності більше) рази з різною кількістю опорних та розрахункових точок. Таким чином, незалежно один від одного алгоритми надають різні відповіді. Після цього вони порівнюються у відповідних точках, і, якщо різниця між ними не перевищує

допустиму, той з них, який використовував більше точок, можна вважати наближенням точної відповіді.

Необхідно додати, що проблеми, що присутні усередині алгоритму, наприклад, розв’язання систем лінійних алгебраїчних рівнянь (СЛАР), або нелінійних рівнянь не розглядаються, бо вони є розв’язаними[24-25].

1.7 Висновки за розділом

Для досягнення поставленої мети необхідно виконати наступні завдання дослідження.

Провести об’єктний аналіз предметної області, дослідити сучасні технології розпаралелення та засоби підтримки паралельного програмування;

Розглянути існуючи алгоритми зведення розв’язання жорстких еволюційних рівнянь до методу прямих, алгоритми чисельної реалізації еволюційних рівнянь на основі паралельних колокаційних блокових методів, алгоритми керування кроком і порядком інтегрування;

Провести критичний аналіз розглянутих алгоритмів з метою визначення топологічних особливостей і схем реалізації комунікаційних операцій між паралельними процесами;

Розробити паралельні модифікації алгоритмів чисельного розв’язання еволюційних рівнянь для реалізації на основі паралельних багатокрокових колокаційних блокових методів;

Отримати програмну реалізацію розроблених алгоритмів розв’язання жорстких систем на основі паралельних колокаційних блокових методів з генеруванням коефіцієнтів методу заданого порядку точності;

Провести симуляції динамічних моделей з еволюційними рівняннями на основі розроблених модифікацій і дослідити основні характеристики паралелізму: прискорення та ефективність.

2 СТРУКТУРНИЙ АНАЛІЗ РЕАЛІЗАЦІЇ МЕТОДУ ПРЯМИХ

2.1 Вибір мови програмування та інструментів роботи

Через обмеження на присутність бібліотеки MPI робота може бути виконана тільки на трьох мовах програмування, не рахуючи адаптації. У цій роботі була обрана мова C++ через можливість об’єктно-орієнтованого підходу, існуючим допоміжним бібліотекам та популярності. C++ є мовою середнього рівня, і частіше використовується, якщо ефективність та швидкодія є важливими параметрами у програмі.

Для розробки програми на паралельну систему використовується MPI, що дозволяє реалізувати ідею розподіленої за процесорами пам’яті. При тестуванні програми на одному процесорі (як це буде зроблено у цій роботі) бібліотека має можливість розподілити задачу на ядра із подібною ефективністю [7].

2.2 Вибір оптимальної топології для задачі

Топологія паралельної системи відіграє важливу роль у показниках швидкості розв’язання задачі. Так як обміни між процесами під час виконання програми є невід’ємним, їх зв’язки необхідно встановити правильним чином до початку виконання програми.

Як було сказано раніше, у алгоритму є декілька способів розпаралелення, і, застосувати їх всі буде найоптимальнішим рішенням. Проте такий метод хоч і є найбільш швидким, майже ніколи не може бути застосований. Реальні паралельні системи або не розраховані на розв’язання такої специфічної задачі, або не мають можливості змінювати кількість вузлів або зв’язків між ними. Також варто пам’ятати про можливість гібридного розпаралелення і між системами з розподіленою пам’яттю, і з системами, що мають спільну пам’ять.

Якщо задача – розподілити проблему знаходження значень функції на точках уздовж прямих після отримання (1.6), використовуючи набір процесів для корекції результатів безпосередньо після розрахунку кожного блоку (докладніше в п.п 1.5), найоптимальнішою топологією є найпростіша з можливих – кільце (рис. 2.1).

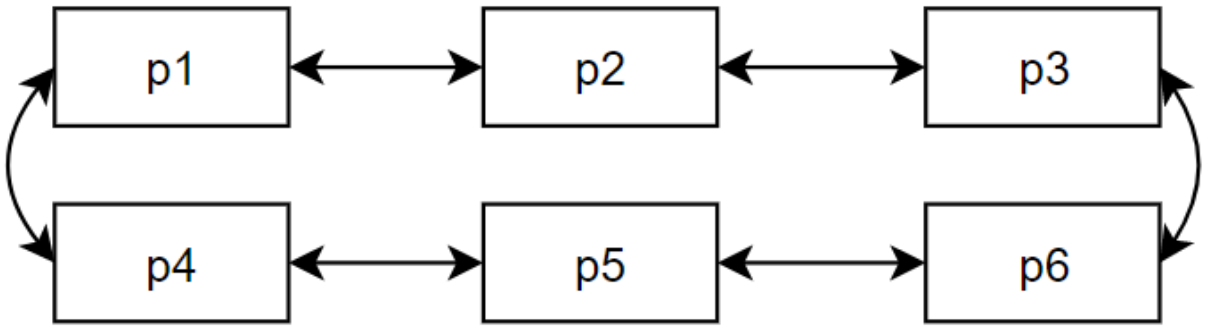


Рисунок 2.1 – Топологія «кільце» на шести процесорах

Так як значення на прямих залежить від безпосередніх сусідів, то додаткових зв’язків не потребується.

При проблемі розпаралелення знаходження коефіцієнтів для колокаційних блоків більш доречно використати системи із загальною пам’яттю у комбінації із попереднім варіантом. Таким чином, залишаються всі переваги, що надає попередній метод, але швидкість розрахунку точок на прямих стає ще швидшою. Проблемою при такому підході є відносна складність паралельної системи.

Найголовнішим скороченням часу є розподілення всього алгоритму на групи процесорів (як мінімум 2 групи). В залежності від кількості процесів у системі можна виділити певну кількість груп. Як було сказано в п.п. 1.5, це є один з єдиних варіантів перевірити правильність розв’язку при відсутності аналітичної відповіді. Також, при відносно невеликих проблемах або великих часових можливостей у дослідників, запускаючи алгоритм декілька разів при різних вхідних параметрах (наприклад, кількість точок у блоковому методі) допоможе надати їх оптимальний варіант.

Для такого способу розпаралелення топологія «кільце» також надає необхідну кількість зв’язків.

Виходячи із цих висновків, можна встановити, що «кільце» – найкраща топологія для даної задачі, але це може бути не так. Така умова виконується тільки на макрорівні, однак для отримання кращих результатів слід звертати увагу на структуру кожної окремої паралельної системи. Якщо, наприклад, надана система спроєктована для розрахунку СЛАУ або розрахунку складної правої частини (1.6), варто використати цю перевагу у правильному розподіленні процесорів на групи, бо ці задачі є частиною проблеми.

2.3 Інтерпретація та візуалізація отриманого результату

Після виконання розрахунків та отримання результату його необхідно правильно зрозуміти та представити користувачу.

Як було вказано у першому розділі, відповідь на таку задачу представляє собою складну структуру – матрицю у найпростішому одновимірному варіанті. Якщо розглядається дво- або тривимірний об’єкт, візуалізація результату є окремою проблемою.

При роботі із ЗДР виду (1.1), відповіддю на нього може бути набір значень незалежного аргументу та відповідний набір значень функції. Це може бути легко представлено у вигляді графіка.

Наприклад, при розв’язанні диференційного рівняння отримано таблицю, що зображено на рис. 2.1. Тоді, ці точки наносяться на графік та проводиться інтерполяційна крива, що наближено надає відповідь на це рівняння (рис. 2.1).

При розв’язанні ДРЧП параболічного типу на кожен момент часу необхідно отримати повний стан об’єкта, тобто набір даних на рис. 2.1 необхідно представити у такому вигляді.

x	y(x)
0,25	0,5
0,50	0,707107
0,75	0,866025
1,00	1
1,25	1,118034
1,50	1,224745
1,75	1,322876
2,00	1,414214
2,25	1,5
2,50	1,581139
2,75	1,658312
3,00	1,732051
3,25	1,802776
3,50	1,870829
3,75	1,936492
4,00	2
4,25	2,061553
4,50	2,12132
4,75	2,179449
5,00	2,236068

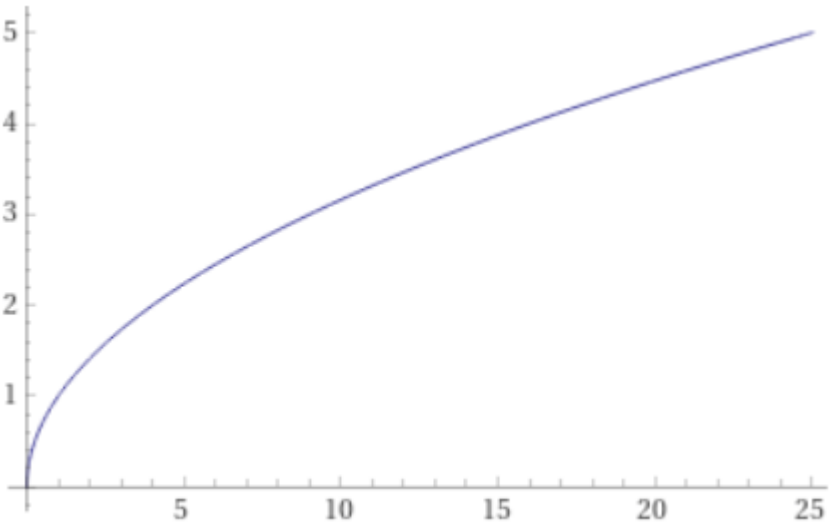


Рисунок 2.1 – Інтерпретація відповіді на ЗДР

Якщо розв’язується рівняння з однією змінною простору, таку відповідь можна зобразити, як тривимірне зображення, де за однією віссю буде значення незалежного аргументу, за іншою – залежного, а за третьою – значення часу (рис. 2.2). На цьому рисунку колір не надає додаткової інформації та служить для простішого розпізнавання графіку.

У площині, де $t = 0$, видно початкові умови задачі, і зі зростанням часу змінюється значення залежного аргументу від зміни незалежного.

Проблема візуалізації стає набагато важче, якщо у початковому рівнянні було дві та більше змінних простору. У такому разі необхідно чотири та більше вимірів для візуалізації відповіді.

У випадку із чотирма змінними є можливість використати методи типу проекції, перетину, розгортки та їм подібним, або у якості змінної часу використовувати реальний час, отримуючи відповідну кількість тривимірних зображень.

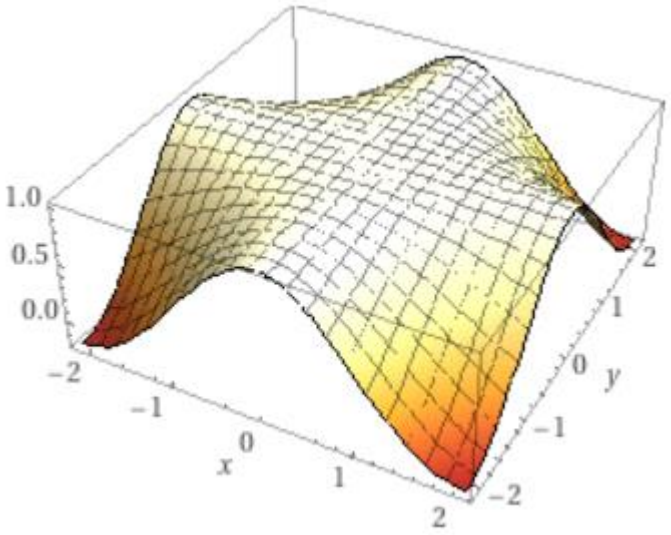


Рисунок 2.2 – Один з варіантів зображення відповіді
на одномірне ДРЧП

Якщо змінних простору 3 і більше, тобто загальна кількість змінних чотири і більше, використовуються таблиці, як на рис. 2.1 для кожної змінної, та кожна із змінних аналізується окремо, або, можливо, у деякій комбінації з іншими.

2.4 Висновки за розділом

У цьому розділі було розглянуто методи та інструменти для реалізації поставленої задачі, а саме обрано:

- мови програмування;
- топологію паралельної системи щодо кожного підходу розпаралелення;
- спосіб представлення результатів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Спосіб програмного завдання математичних структур

ДРЧП параболічного вигляду мають спільний вигляд, тому доцільним буде представлення відразу отриманої системи рівнянь з початкового ДРЧП з додаванням деяких параметрів. Це означає, що користувач вказує у якості параметра тільки додаткові умови з рівняння [26]. Такий спосіб дає можливість користувачу замість опису всього рівняння описати тільки ті його складові, що не співпадають з найпростішим варіантом рівняння – рівняння теплопровідності [27]. Всі подробиці щодо програмного коду знаходяться у додатку Б. Проміжні та кінцеві відповіді знаходяться у одній змінній (двовимірному масиві). Таким чином, якщо необхідно зробити завдання більш складним за допомогою додавання додаткової змінної, достатньо додати вимір у таку змінну.

3.2 Опис методів системи

Необхідними для даної системи є декілька методів:

- завдання ДРЧП;
- підрахунок коефіцієнтів блокових схем;
- застосування блокових схем;
- формування результату.

Для розв’язання системи ЗДР використовуються колокаційні блокові методи. Підрахунок коефіцієнтів блокових схем відбувається за формулою (1.11) за алгоритмом, зображеним на рис. 3.1. На цьому рисунку підпрограма Solve(G,F) означає знаходження рішення на СЛАР виду (3.1).

$$Gx = F. \qquad (3.1)$$

Це векторний вигляд СЛАР та завданням є знайти вектор x. Так як ця система завжди невеликої розмірності, можна використовувати точні методи, проте чисельні методи можуть дати результати достатньої точності за відносно невелику кількість ітерацій.

Після всіх підрахунків коефіцієнти та розв'язки системи ЗДР передаються у функцію, де відбувається безпосереднє застосування алгоритму колокаційного блокового методу на певну кількість точок.

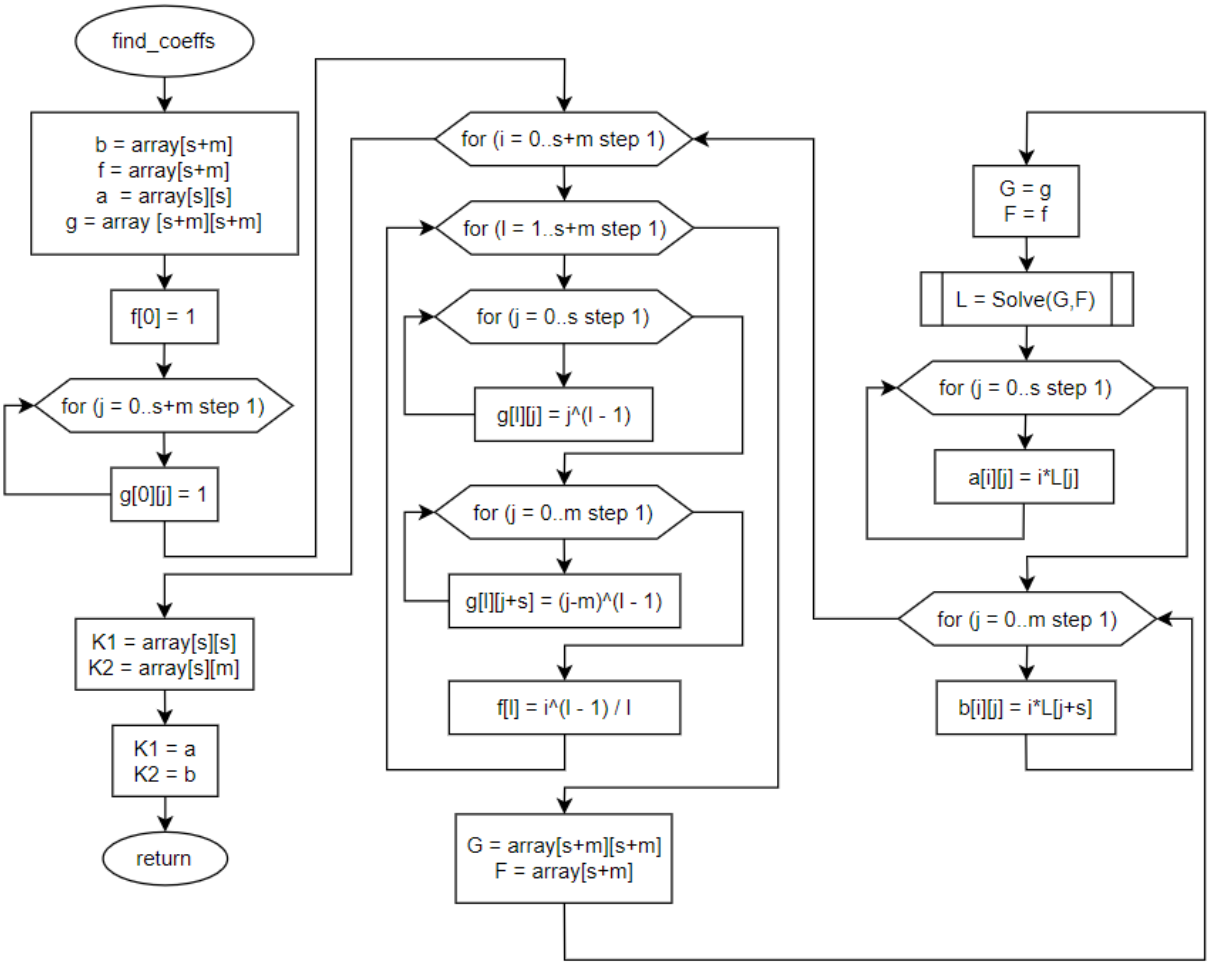


Рисунок 3.1 – Блок-схема розрахунку коефіцієнтів колокаційних блоків

Через відносну складність цього алгоритму його розпаралелення – актуальна проблема для цієї задачі.

Під час розробки програми алгоритм мало відрізняється від математичної моделі. Формула (1.10) описує процес знаходження певної точки

у певному блоці. Використовуючи це співвідношення ітераційно для кожного наступного блоку, отримуються необхідні відповіді.

3.3 Використання бібліотеки MPI

Так як програма тестується на одному процесорі з 8 ядрами, це і є обмеженням на можливе розпаралелення, яке буде використано під час аналізу характеристик паралельної реалізації. Однак у теорії, програма буде написана під оптимально необхідну кількість процесорів (докладно у додатку Б).

Як було сказано раніше, можливості бібліотеки дозволяють майже без втрати у швидкості застосувати ядра одного процесору як паралельну систему, що і буде зроблено у цій роботі.

Для використання MPI достатньо завантажити необхідні файли та виконати деякі налаштування у середовищі програмування.

Паралельно буде весь алгоритм в цілому, тобто необхідно виділити дві групи процесорів, кожна з яких розв’яже задачу з початку до кінця.

3.4 Виведення та візуалізація результатів

Мова C++ ідеально підходить для виконання задачі розв’язання ДРЧП, проте візуалізація та обробка великого об’єму даних – більш підходяща задача для інших мов. У даній роботі для цього використовується Python 3.8 через простоту його використання та наявність таких корисних бібліотек, як matplotlib, numpy, pandas та інших [28-30s].

Візуалізація є вторинним завданням, бо на даному етапі вже не відбувається ніяких обчислень, проте може виникнути можливість перевірити проміжні дані, такі як розв’язання системи ЗДР. Для цього деякі проміжні результати записуються у тимчасовий файл для подальшого їх використання користувачем при потребі.

Python дозволяє отримувати як тривимірні зображення, так і набор двовимірних, наприклад, для порівняння того, яка різниця у зміні температури тільки між двома сусідніми точками. Такий набор виглядає як декілька графіків на одній системі координат.

3.5 Висновки за розділом

Даний розділ було присвячено програмній реалізації системи, а саме:

- аналізу структур, що використовуються у програмі;
- огляду деталей математичної частини задачі;
- формуванню необхідних функцій для роботи програми;
- детальному опису алгоритму задачі;
- способу виведення результату.

Також було розглянуто спосіб використання процесорів паралельної системи для вирішення даної задачі та її частин.

4 ТЕСТУВАННЯ СИСТЕМИ ТА КЕРІВНИЦТВО КОРИСТУВАЧА

4.1 Використання функцій системи

Для використання функції розв’язання методу прямих, користувач повинен задати умови – інтервал інтегрування за просторовою та часовою змінною, початкову умову виду $u(x,0) = f(x)$ та граничні умови. Можливий приклад використання:

```
double x_start = 0.0;
double x_end = M_PI;
double t_end = 1.0;
double h = M_PI/30;
double d = 0.05;
double thermal_diffusivity = 1;
int number_of_lines = (x_end - x_start) / h + 1
double *start_cond = new double[number_of_lines];
for (int i = 0; i < number_of_lines; i++){
    start_cond[i] = sin(k);
    k += h;}
double left_cond = 0;
double right_cond = 0;
```

Рисунок 4.1 – Приклад початкових умов

Всі ці параметри відіграють важливу роль, як буде виглядати розв’язок ДРЧП, тому кожен із них має бути визначений окремо, проте передбачені умови за замовчуванням. Також у загальному випадку існує права частина рівняння, яка залежить від часової та просторової змінної. Так як характер такої функції унікальний, користувач повинен створити її, наприклад, так:

```
double Rhs(double x, double t){
    return sin(x * t);}
```

Рисунок 4.2 – Приклад правої частини

Після цього викликається функція Solve_PDE з усіма зазначеними параметрами та її результат записується у останній з них. Надалі користувач може перенести цю інформацію до файлу, щоб аналізувати отримані дані.

4.2 Тестування програми на прикладі рівняння теплопровідності

Розглядається рівняння теплопровідності у найпростішому випадку (4.1) для полегшення викладу матеріалу. Фізично таке рівняння означає об'єкт, що має різну температуру на початку експерименту та поступово набуває однакової температури на всьому інтервалі. У такій інтерпретації значення u є температурою об'єкта, а друга похідна за x – як сильно сусідні точки більш або менш нагріті. Важливо помітити, що у рівнянні немає сторонніх сил, тобто на температуру кожної точки об'єкта будуть впливати тільки інші точки цього ж об'єкта.

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}; u(x,0) = sin(x); u(0,t) = u(\pi,t) = 0. \qquad (4.1)$$

У даному випадку рівняння є одновимірне і $\alpha = 0.4$. Початкова та граничні умови обрані таким чином, щоб на результаті було більш просто оцінити відповідь.

Відповідно з таких умов початкова координата за простором – 0, кінцева координата за простором – π, додаткова функція зовнішніх впливів відсутня.

Нехай крок за простором обирається $h = \frac{\pi}{30} \approx 0.1$, а за часом для блокового методу – $d = 0.05$. Час моделювання – 5 часових одиниць. Кількість точок у блоці – 2.

Отриманий результат записується в файл, що виглядає, як зображено на рис. 4.3. Файл містить 31 рядок та 101 стовпець, на рисунку видно тільки початок.

Перший та останній рядок відповідають граничним умовам (у даному випадку 0), перший стовпець відповідає координаті об’єкту за простором (місця встановлення прямих), другий – початковим умовам, а третій та подальші – розрахованим значенням у певний момент часу, у даному випадку другий рядок вказує на стан об’єкта у момент часу $t = 0$, третій у $t = 0.05$ і т.д.

0.000000	0.000000	0.000000	0.000000
0.104720	0.104528	0.104070	0.103614
0.209440	0.207912	0.207001	0.206093
0.314159	0.309017	0.307663	0.306314
0.418879	0.406737	0.404954	0.403179
0.523599	0.500000	0.497809	0.495627
0.628319	0.587785	0.585209	0.582645
0.733038	0.669131	0.666198	0.663279
0.837758	0.743145	0.739888	0.736645
0.942478	0.809017	0.805471	0.801942
1.047198	0.866025	0.862230	0.858451
1.151917	0.913545	0.909542	0.905556

Рисунок 4.3 – Частина файлу з результатами

Наприклад, у момент часу $t = 0.05$ у точці $x = 0.733038$ температура мала значення $u = 0.669131$.

Маючи такий набір даних, можна побудувати графіки розв’язання задачі. Проте, як було сказано раніше, це представляється у трьох вимірах, тож, для початку, на рисунку 4.4 представлено графіки залежності температури від простору.

Так як на рисунку зображено 101 графік, важко відрізнити кожен окремий з них, проте видно загальну поведінку функції. Для докладного аналізу можна подивитись у текстовий файл.

Тепер можна зобразити залежність температури відносно часу (рис. 4.5). Тут зображено всього 30 графіків, проте видно тільки 15. Це відбувається

через те, що половина з них симетричні (через початкові умови у вигляді синуса).

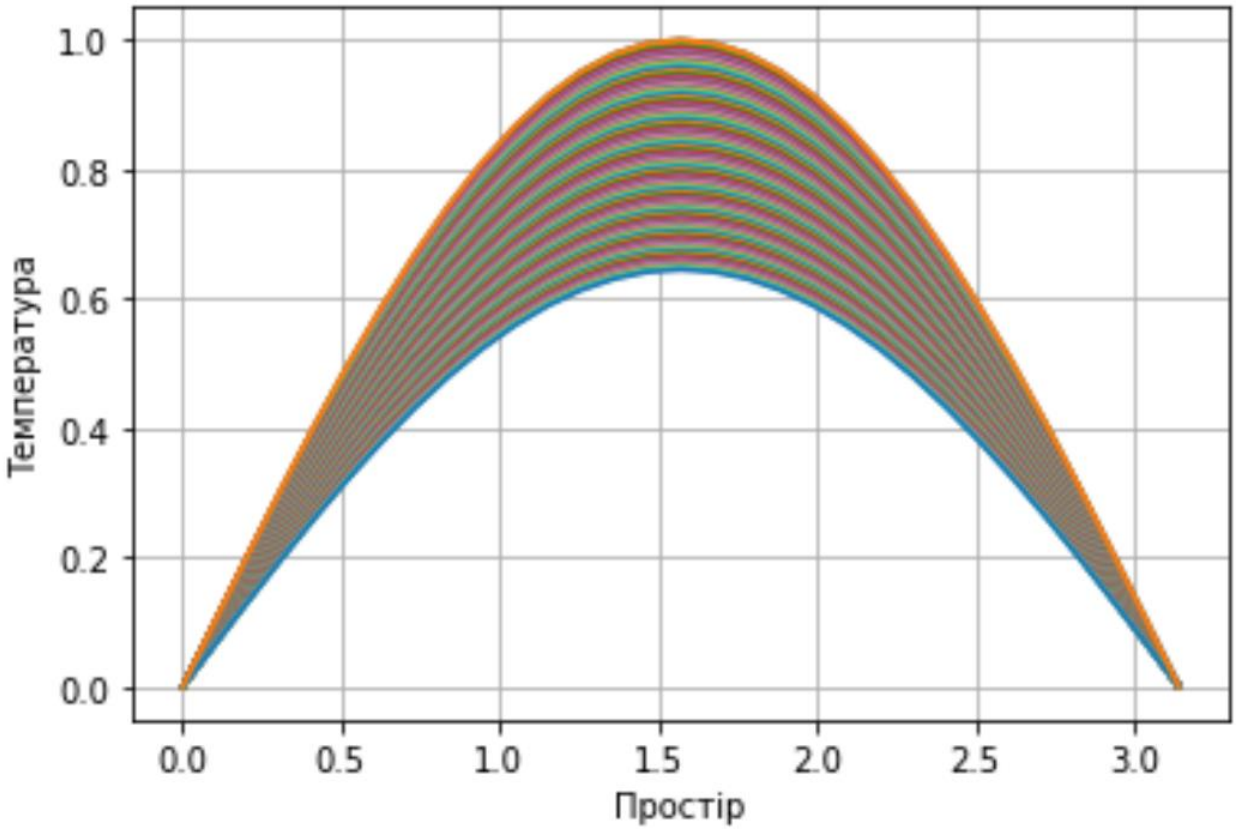


Рисунок 4.4 – Графіки залежності температури від простору

У розв’язанні ДРЧП дуже важливими є початкові та граничні умови. Так як у цьому прикладі немає зовнішньої взаємодії, сумарна температура системи повинна залишатися сталою, але, як видно на графіках, це правило не виконується. Це відбувається через те, що граничні умови є сталими величинами, що змушують систему змінювати свій стан, залишаючись незмінними.

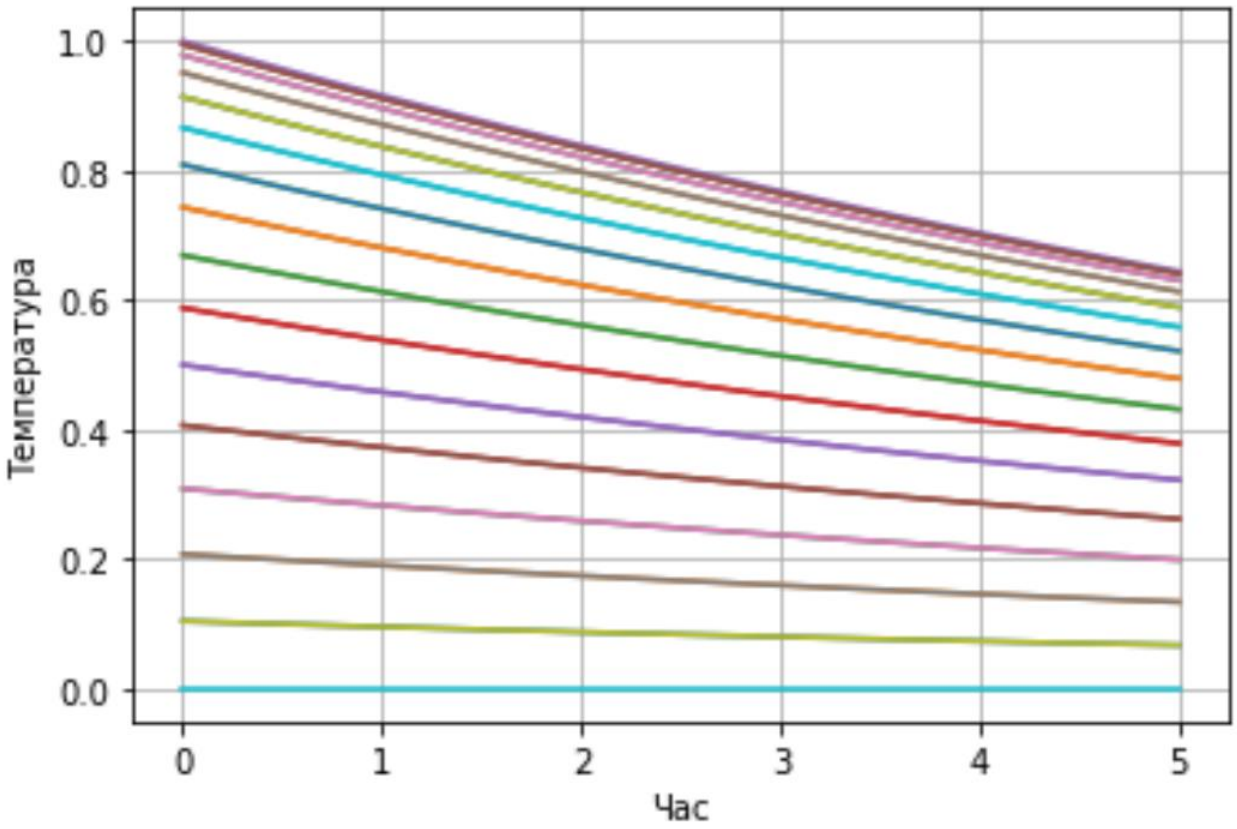


Рисунок 4.5 – Графіки залежності температури від часу

Нехай тепер початкові та граничні умови мають іншу структуру, наприклад:

$$u(x,0) = \frac{x}{2}, u(0,t) = 0, u(\pi,t) = 1. \qquad (4.2)$$

Тепер точка $x = \pi$ відрізняється від попередньої досить сильно через розрив. Результати роботи програми за таких умов наведені на рис. 4.6-4.7.

Виходячи з цих спостережень, можна помітити, що при таких умовах крок є замалим для правильного аналізу правої частини об’єкта, тому щоб правильно проаналізувати його стан, необхідно зменшити крок у частині розриву.

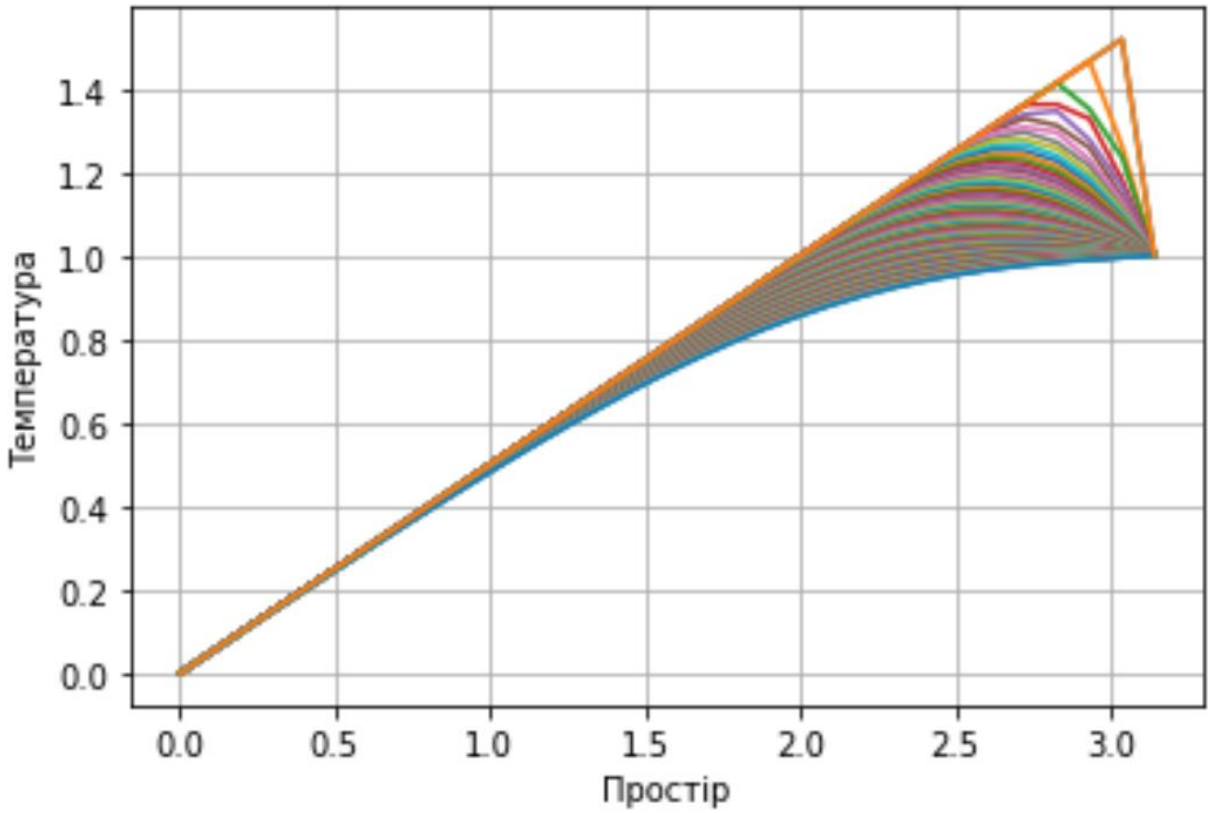


Рисунок 4.6 – Графіки залежності температури від простору за новими умовами

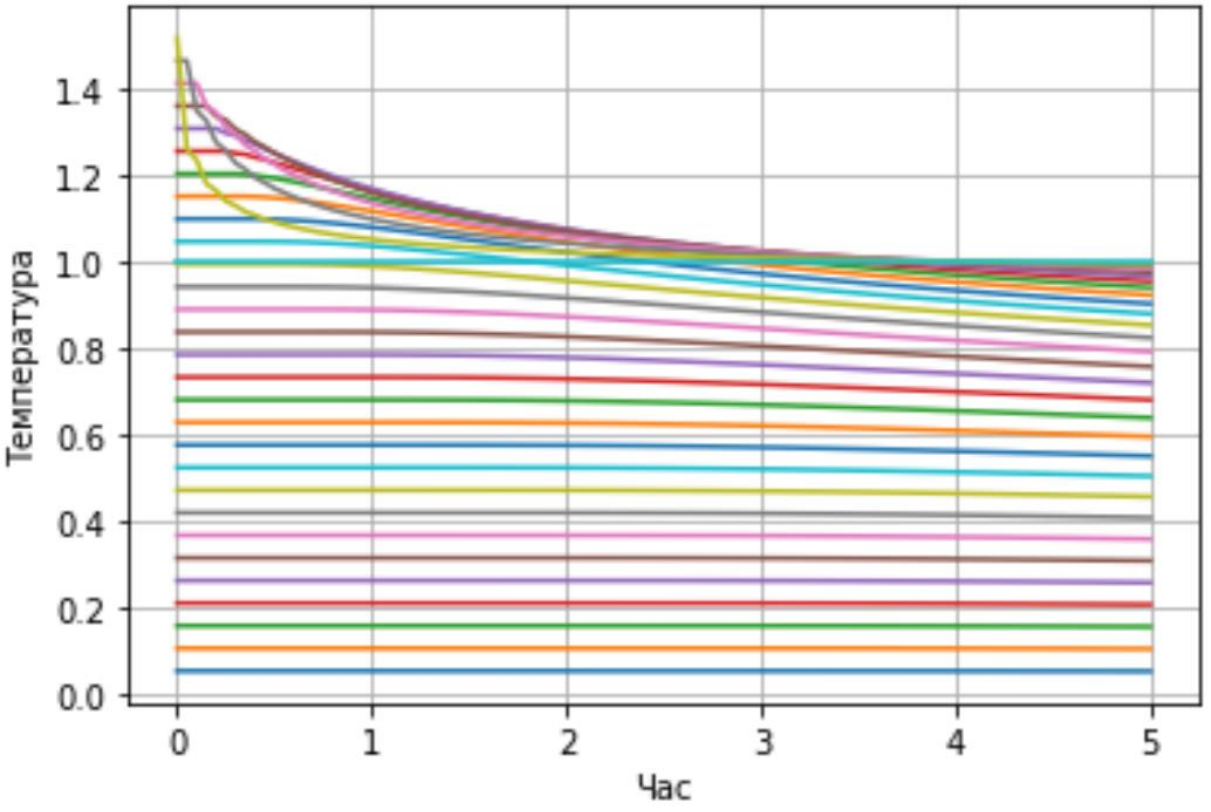


Рисунок 4.7 – Графіки залежності температури від часу за новими умовами

Така задача є більш реальною. Подібна поведінка може виникнути, коли такий одномірний об’єкт із своїм правилом розподілення температури починає взаємодію із іншим, де температура різко змінюється.

4.3 Аналіз точності отриманого результату

Для перевірки правильності отриманого результату необхідно паралельно запустити програму на різну розмірність блоків. Для тестового прикладу обирається однокроковий метод із числом розрахункових точок 2 та 3.

Приклади, наведені вище, були розраховані за допомогою двох блоків без застосування паралельних систем.

Тепер запускається та ж програма із умовами (4.1), але паралельно на 2 і 3 точки у блоці відповідно. Очевидно, отримуються два окремих файли, з яких і можна порівняти точність алгоритму.

Для перевірки точності достатньо знайти різницю між відповідними значеннями у кожній з точок, так як ні крок за часом, ні крок за простором не було змінено. Тож на рисунку 4.8 зображена середня похибка для усіх моментів часу для відповідної точки простору. Крива схожа на графік початкових умов, бо саме у тих точках, які змінюють свої значення більше всього, буде більше похибки.

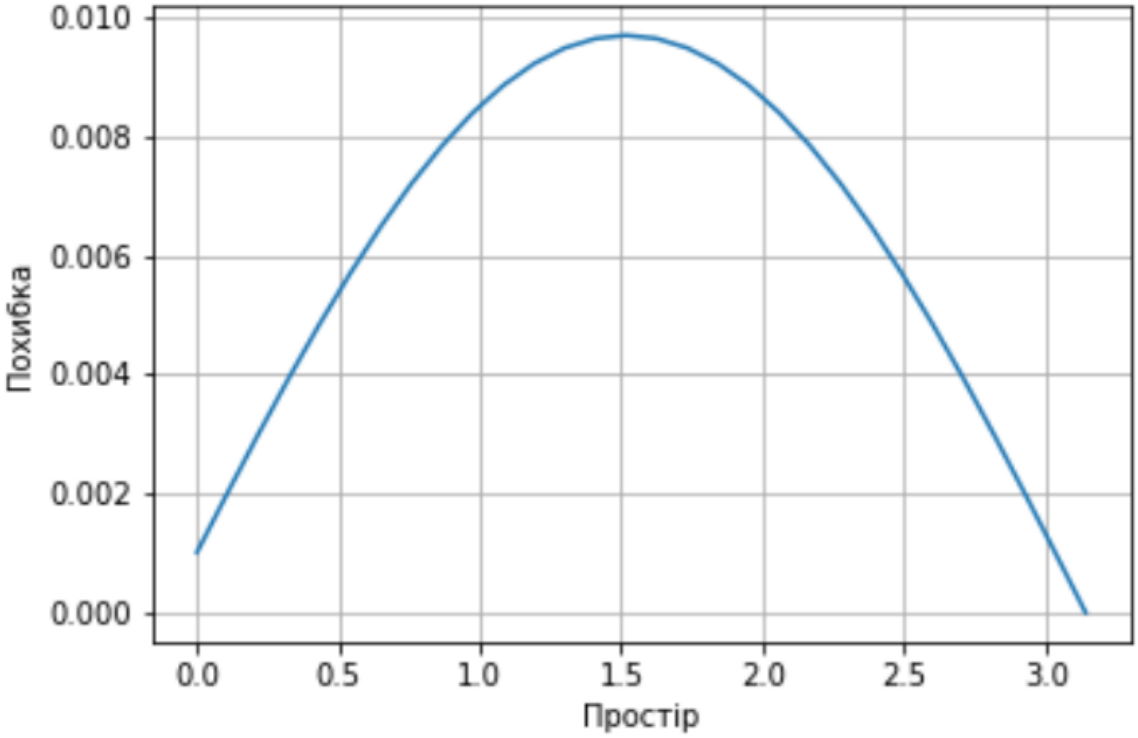


Рисунок 4.8 – Середня похибка у кожній точці простору

Якщо запустити такий алгоритм для другої умови (4.2), то графік похибок виглядає наступним чином (рис. 4.9):

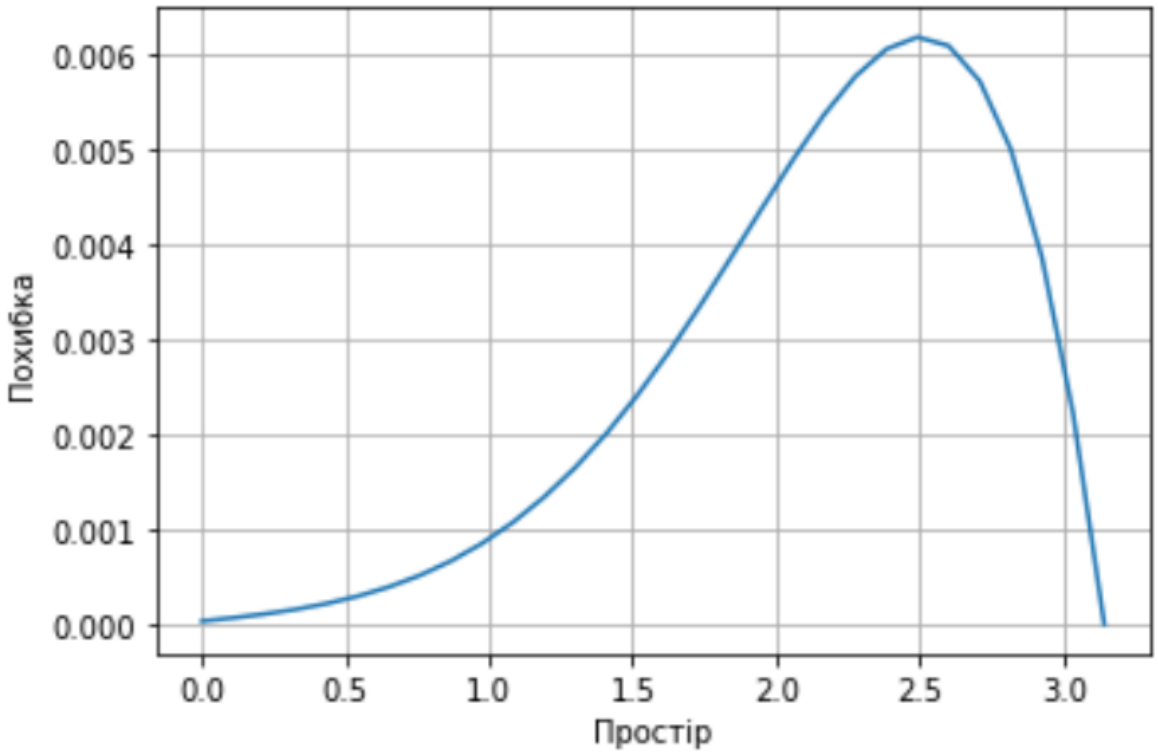


Рисунок 4.7 – Графік похибки для умов (4.9)

Через майже незмінний стан системи у першій половині об’єкта похибка виявляється меншою.

4.4 Динамічні характеристики алгоритму

Основні динамічні характеристики – це прискорення та ефективність. Вони виражаються через більш прості операції.

Нехай $T_{p,comp}$ – час виконання алгоритму із використанням p процесорів без урахування операцій обміну, а T_p – загальний час виконання алгоритму з p процесорами.

Тоді прискорення дорівнює:

$$S_p = \frac{T_1}{T_p}, \tag{4.3}$$

де T_1 – час виконання послідовної програми, а ефективність:

$$E_p = \frac{S_p}{p}. \tag{4.4}$$

Для алгоритму, запропонованого у роботі, розрахунок цих характеристик є досить тривіальним: яким би не був час виконання послідовної проблеми, час виконання паралельної буде у стільки разів швидший, скільки разів було використано алгоритм.

Тобто для випадку із розрахунком двох разів прискорення дорівнює 2, а ефективність – 1.

4.5 Висновки за розділом

У цьому розділі було описано, як необхідно користуватися програмою для розв’язання ДРЧП і на прикладі рівняння теплопровідності показано, як виглядає процес розв’язання та результат.

Також було розглянуто яку похибку дає такий спосіб розрахунку на прикладі однокрокового методу на 2 і 3 точки розрахункового блоку та пораховано динамічні характеристики системи.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розглянуто теоретичні відомості проблеми предметної області, а саме:

- постановку задачі;
- основні та найпоширеніші методи її розв’язання;
- опис метода прямих;
- опис метода колокаційних блоків.

Для розв’язання цієї проблеми було представлено математичний апарат методу прямих, його особливості, конкуренти та переваги, а також аналогічний аналіз для колокаційних блоків.

У роботі наведені способи та методи створення паралельної системи для розв’язання ДРЧП методом прямих, опис необхідних бібліотек та мов програмування, запропоновано набір фунцій, що є необхідними для розв’язання цієї задачі та докладно описано керівництво користувача.

Також виконано аналіз точності та ефективності роботи створеного алгоритму.

Тож було виконано всі поставлені задачі:

1) Проведено аналіз предметної області та досліджено засоби паралельного програмування;

2) Розглянуто алгоритми розв’язання ДРЧП, включаючи зведення до методу прямих та на основі коллокаційних блокових методів;

3) Проведено аналіз з метою визначення топології;

4) Розроблено модифікації чисельних алгоритмів розв’язання еволюційних рівнянь;

5) Виконано програмну реалізацію наведених алгоритмів з урахуванням кроку та порядку точності;

6) Проведено симуляції динамічних моделей та визначено характеристики паралелізму.

Проте залишаються деякі покращення, що виходять за межі мети цієї роботи, це:

– реалізація паралельного обчислення прямих з урахуванням уточнення між блоками;

– більш докладний аналіз маштабованості методу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Хусаінов Д.Я., Харченко І.І. Шатирко А.В. Введення в моделювання динамічних систем: навч. посіб. Київ: КНІ ім. Т. Шевченка, 2010. 132 с.

2. Пріщенко О.П., Черногор Т.Т. Диференціальні рівняння та їх застосування. Харків: НТУ «ХПІ», 2017. 88 с.

3. Адамян В.М., Сушко М.Я. Вступ до математичної фізики. Одеса: Астропринт, 2003. 217 с.

4. Швець В.Т. Вища математика: операційне числення. Одеса: ВМВ, 2015. 138 с.

5. Метод прямих [Електронний ресурс] – Режим доступу: http://www.scholarpedia.org/article/Method_of_lines

6. Бокало М.М. Диференціальні рівняння. Львів: ЛНІ ім. І. Франка, 2014. 230с.

7. Walker D.W. Standarts for Message-Passing in Distributed Memory Environment. Oak Ridge: 1992, 33 p.

8. Гаврилова В.С., Денисова Н.А. Метод характеристик для одномерного волного уравнения. Новгород: 2014. 72 с.

9. Задачин В.М., Конюшенко І.Г. Чисельні методи. Харків: ХНЕУ ім. С. Кузнеця, 2014. 180 с.

10. Легостаєв А.Д. Метод скінченних елементів. Київ: КНУБА, 2004. 112с.

11. Прокопенко Ю.В., Казміренко В.А., Голубєва І.П. Методи математичної фізики. Київ: Видавництво, 2018. 144 с.

12. Ілляшенко Л.М., Стогній Н.П., Нерух О.Г. Спектральний метод граничних інтегральних рівнянь в фотоніці. Харків: ХНУРЕ, 2019. 75 с.

13. Huerta A., Belytschko T., Fernadez-Mendez S., Rabczuk T. Meshfree Methods. Evanston: Northwestern University, 2004, 49 p.

14. Метод прямих [Електронний ресурс] – Режим доступу: http://www.scholarpedia.org/article/Method_of_lines

15. Самкова Г.Є., Н.В. Шарай, О.П. Мойсєєнок. Звичайні диференціальні рівняння та системи звичаних диференціальних рівнянь. Одеса: ОНУ, 2019. 112 с.

16. Гой Т.П., Копач М.І., Федак І.В. Наближені методи розв’язування диференціальних рівнянь. Івано-Франківськ: ПНІ ім. В. Стефанка, 2010. 152 с.

17. Кузнецов В.М., Бусарова Т.М., Агошкова Т.А. Похідна та її застосування. Дніпро: НМВ ДНУЗТ, 2017. 108 с.

18. Diagrams.net [Електронний ресурс] – Режим доступу: https://app.diagrams.net/

19. Адаптація MPI для Python [Електронний ресурс] – Режим доступу: https://sourceforge.net/projects/pydusa/

20. Адаптація MPI для Java [Електронний ресурс] – Режим доступу: http://www.hpjava.org/mpiJava.html

21. Chandra R., Dagum L., Kohr D. Parralel programming in OpenMP. San Diego, CA. 249 p.

22. Дмитриева О.А. Коллокационные блочные методы с контролем на шаге. Системи обробки інформації. 2015. вип. 5 (130), ISSN 1681-7710. С. 78-84

23. Дмитриева О.А. Высокоэффективные алгоритмы управления шагом на основе параллельных колокационных блочных методов. Штучний інтелект 4’2012, 2012. С. 77-88

24. Шахно С.М., Дудикевич А.Т., Левицька С.М. Практична реалізація чисельних методів лінійної алгебри. Львів: ЛНУ ім. І. Франка, 2009. 137 с.

25. Колесницький О.К., Арсенюк І.Р., Месюра В.І. Чисельні методи. Вінниця: ВНТУ, 2017. 131 с.

26. Lee H.J., Schiesser W.E. Ordinary and Partial Differential Equation Routines in C, C++, Fortran, Java, Maple, and MATLAB. Washington: 2004, 515 p.

27. Рівняння теплопровідності [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Heat_equation

28. Python [Електронний ресурс] – Режим доступу: https://www.python.org/

29. Бібліотека Python – Numpy [Електронний ресурс] – Режим доступу: https://numpy.org/

30. Бібліотека Python – Pandas [Електронний ресурс] – Режим доступу: https://pandas.pydata.org/

ДОДАТОК А
ЗАУВАЖЕННЯ НОРМОКОНТРОЛЕРА

Таблиця А.1 – Зауваження нормоконтролера

Сторінка	Позначення	Зміст зауваження
Зміст		Заголовки як у реченні
Вступ		Перед посиланням на джерело - пробіл
ПЗ		Нещільне заповнення сторінок
		Є помилки в оформленні літератури

Додаток Б

Графічна частина

Презентація до пояснювальної записки випускної кваліфікаційної роботи

Автор: Рогоза Михайло Сергійович

Тема: Паралельна реалізація методу прямих для рівнянь в частинних похідних колокаційними блоковими різницевими схемами

Науковий керівник: д.т.н., проф. О.А. Дмитрієва

Рисунок Б.1 – Вступний слайд презентації

Актуальність та об’єкт роботи

Актуальність роботи полягає у розповсюдженості проблем моделювання динамічних об’єктів, особливо у вигляді диференційних рівнянь

Об’єкт дослідження роботи – процеси моделювання динамічних об’єктів з розподіленими параметрами у комп’ютерних системах.

2

Рисунок Б.2 – Слайд з актуальністю роботи

Предмет дослідження та мета роботи

Предмет дослідження – паралельні чисельні методи розв'язання жорстких динамічних завдань великої розмірності, що описуються рівняннями у частинних похідних.

Мета роботи – дослідження алгоритмів керування параметрами інтегрування та реалізація методу на основі паралельних колокаційних блоків.

3

Рисунок Б.3 – Слайд з предметом дослідження

Постановка задачі

Звичайне ДР:	Рівняння у ЧП2 (2 змінні):
Загальний вигляд	
$F\left(\frac{d^n y}{dx^n}, \frac{d^{n-1} y}{dx^{n-1}}, \dots, \frac{dy}{dx}, x\right) = 0.$	$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu + G = 0$
Початкові умови	
$y(x_0) = y_0$	$u(x, y_0) = u(x)$ $u(x_0, y) = u_0, u(x_k, y) = u_1$
Розв'язок	
$y = f(x)$	$u = f(x, y)$

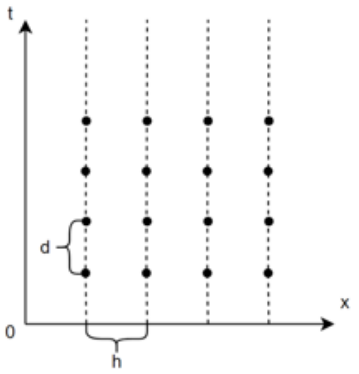
4

Рисунок Б.4 – Слайд із постановкою задачі

Метод прямих: ідея

Рівняння для пояснення методу:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$$ *рівняння теплопровідності*



Формула **скінченних різниць**:

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

Отримана система рівнянь:

$$\frac{du}{dt} = \frac{\alpha}{h^2}[u(i+h,t) - 2u(i,t) + u(i-h,t)]$$

5

Рисунок Б.5 – Слайд із ідеєю методу прямих

Колокаційні блоки: ідея

Рівняння для знаходження точок розрахункового блоку:

$$u_{n,i} = u_{n,0} + d\left(\sum_{j=-(m-1)}^{0} b_{i,j}F_{n,j} + \sum_{j=1}^{s} a_{i,j}F_{n,j}\right), i = 1,2 \dots, s$$

Система умов для знаходження коефіцієнтів:

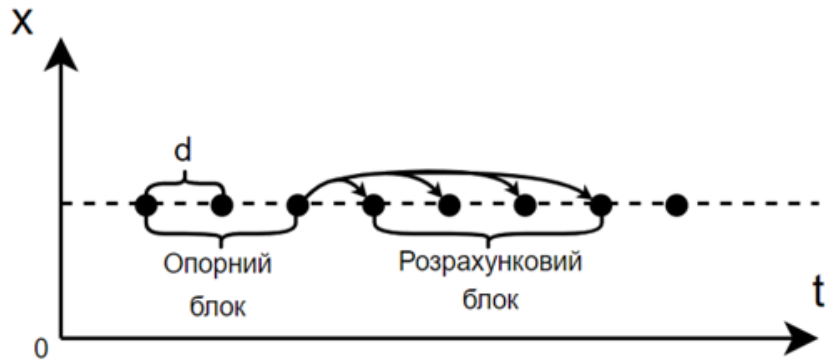
$$\sum_{j=1-m}^{0} b_{i,j} + \sum_{j=1-m}^{0} b_{i,j} = i, i = 1,2, \dots, s,$$

$$\sum_{j=1-m}^{0} b_{i,j}j^{l-1} + \sum_{j=1}^{s} a_{i,j}j^{l-1} = \frac{i^{l-1}}{l}, l = 2,3, \dots, p.$$

6

Рисунок Б.5 – Слайд із ідеєю колокаційних блоків

Колокаційні блоки: ідея. Загальна схема



7

Рисунок Б.7 – Слайд із візуалізацією блокової схеми

Способи застосування паралельних обчислень у алгоритмі

- знаходження точок розрахункового блоку на кожній з прямих;
- знаходження коефіцієнтів розрахункових блоків;
- виконання всього алгоритму при зміні параметрів.

8

Рисунок Б.8 – Слайд із способами застосування паралельних обчислень

Аналіз реалізації алгоритму

Мова програмування – **C++.**

Топологія – **кільце**.

Очікуваний формат відповіді – **таблиця значень**.

	0.05	0.1	0.15	0.2	0.25
0.1	0.104528	0.104070	0.103614	0.103160	0.102708
0.2	0.207912	0.207001	0.206093	0.205190	0.204291
0.3	0.309017	0.307663	0.306314	0.304972	0.303635

9

Рисунок Б.9 – Слайд із аналізом реалізації

Запуск програми: рівняння теплопровідності

1) $\frac{\partial u}{\partial t} = 0.4 * \frac{\partial^2 u}{\partial x^2}; u(x,0) = sin(x); u(0,t) = u(\pi,t) = 0$

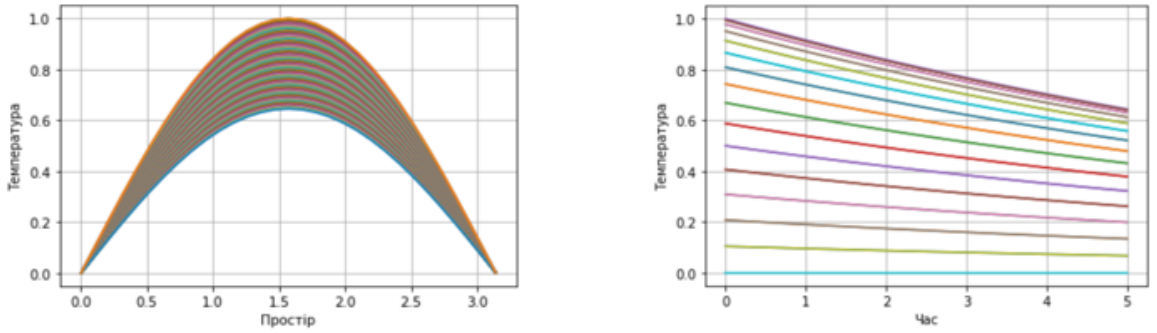
$h = \frac{\pi}{30} \quad d = 0.05 \quad m = 1 \quad s = 2$

2) $\frac{\partial u}{\partial t} = 0.4 * \frac{\partial^2 u}{\partial x^2}; u(x,0) = \frac{x}{2}, u(0,t) = 0, u(\pi,t) = 1$

10

Рисунок Б.10 – Слайд із описом тестового завдання

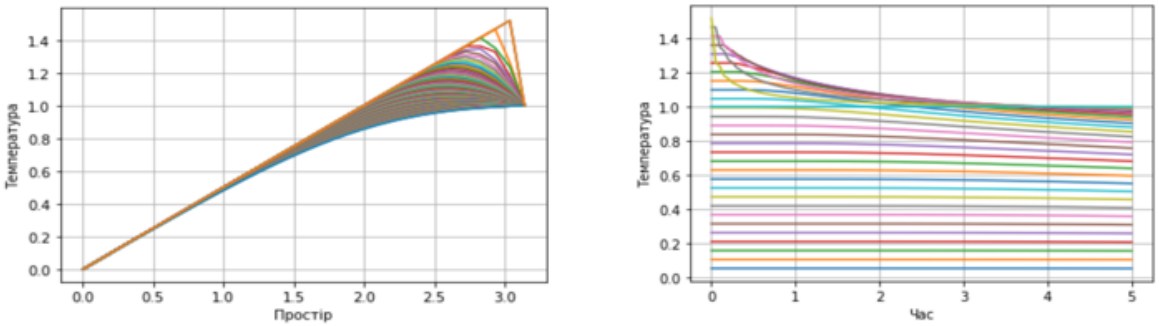
Запуск програми: результат



11

Рисунок Б.11 – Слайд із результатом роботи

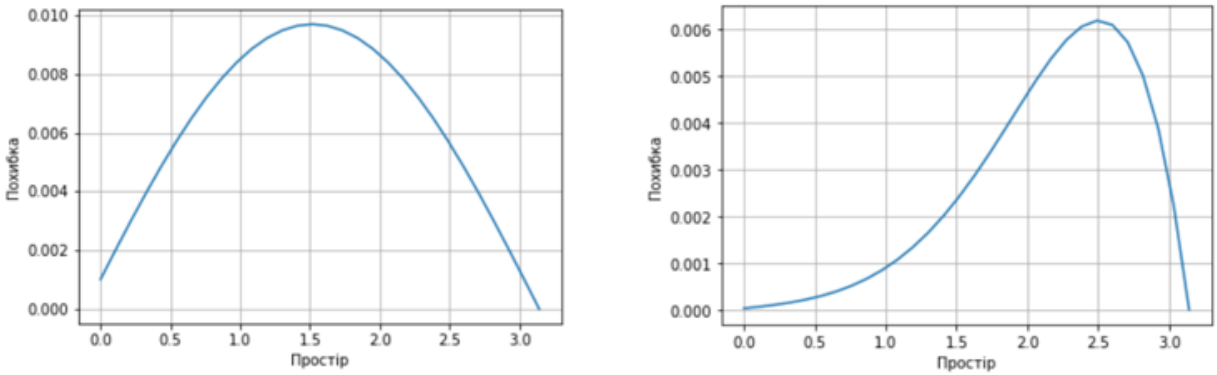
Запуск програми: результат (2)



12

Рисунок Б.12 – Слайд із результатом роботи при інших параметрах

Запуск програми: похибка



13

Рисунок Б.13 – Слайд із аналізом похибки

Висновки

Отже, було досягнуто **поставлені задачі**:

- проведено **аналіз** предметної області;
- досліджено **засоби** паралельного підходу;
- вивчено та реалізовано **метод прямих** та **колокаційні блоки** для рівнянь у частинних похідних;

Подальша робота включає в себе удосконалення програми для застосування **усіх можливих способів розпаралелення**.

14

Рисунок Б.14 – Слайд із висновками

Дякую за увагу!

15

Рисунок Б.15 – Останній слайд

Додаток В

Лістинг програм

В.1 Лістинг основної розрахункової програми

```
#pragma warning(disable : 4996)
#define _USE_MATH_DEFINES
#include <iostream>
#include <cmath>
#include <fstream>
#include <mpi.h>
using namespace std;
void CreateCoeffs(int s, int m, double **k1, double **k2);
void LinearSolve(int size, double **A, double *B);
double Fi(double **u, int curr_x, int curr_t);
double Rhs(double x, double t);
void Solve_PDE(double x_start, double x_end, double t_end, double h, double d, double *start_cond, double left_cond, double right_cond, int points, double **u);
const static double thermal_diffusivity = 0.45;
int main(int argc, char *argv[])
{
	int my_rank, num_procs;
	double x_start = 0.0;
	double x_end = M_PI;
	double t_end = 5.0;
	double h = M_PI/30;
	double d = 0.05;
	int number_of_lines = (x_end - x_start) / h + 1; // extra at x = x_start and x = x_end
	int number_of_t_points = t_end/d + 1; // extra one at t = 0
	double *start_cond = new double[number_of_lines];
	double k = x_start;
	for (int i = 0; i < number_of_lines; i++)
```

```
	{
		/*start_cond[i] = sin(k);*/
		start_cond[i] = k/2;
		k += h;
	}
	double left_cond = 0;
	double right_cond = 1;		// starting conditions
	double **u = new double*[number_of_lines]; // array of answers
	for (int i = 0; i < number_of_lines; i++)
		u[i] = new double[number_of_t_points];
	MPI_Init(&argc, &argv);
	int points;
	MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
	MPI_Comm_size(MPI_COMM_WORLD, &num_procs);
	ofstream file;

	if (my_rank == 0)
	{
		points = 2;
		Solve_PDE(x_start, x_end, t_end, h, d, start_cond, left_cond,
right_cond, points, u);
		file.open("C:\\Users\\user\\desktop\\f_2.txt", ios::out);
		for (int i = 0; i < number_of_lines; i++)
		{
			file << fixed << i * h << '\t';
			for (int j = 0; j < number_of_t_points; j++)
				file << fixed << u[i][j] << '\t';
			file << endl;
		}
	}
```

```
	if (my_rank == 1)
	{
		points = 3;
		Solve_PDE(x_start, x_end, t_end, h, d, start_cond, left_cond,
right_cond, points, u);
		file.open("C:\\Users\\user\\desktop\\f_3.txt", ios::out);
		for (int i = 0; i < number_of_lines; i++)
		{
			file << fixed << i * h << '\t';
			for (int j = 0; j < number_of_t_points; j++)
				file << fixed << u[i][j] << '\t';
			file << endl;
		}
	}
	MPI_Finalize();
}
double Fi(double **u, int curr_x, int curr_t)
{
	double res = u[curr_x][curr_t + 1] - 2 * u[curr_x][curr_t] + u[curr_x][curr_t
- 1] + Rhs(u[curr_x][0], u[curr_x][curr_t]);
	return res;
}
double Rhs(double x, double t)
{
	/*return sin(x * t);*/
	/*return x * t;*/
	return 0.0;
}
void Solve_PDE(double x_start, double x_end, double t_end, double h, double d,
double *start_cond, double left_cond, double right_cond, int points, double **u)
{
```

```
// constant calculations
int number_of_lines = (x_end - x_start) / h + 1;
int number_of_t_points = t_end / d + 1;
// filling zeros
for (int i = 0; i < number_of_lines; i++)
        for (int j = 0; j < number_of_t_points; j++)
                u[i][j] = 0;

// using starting conditions
for (int i = 0; i < number_of_lines; i++)
        u[i][0] = start_cond[i];
for (int i = 0; i < number_of_t_points; i++)
{
        u[0][i] = left_cond;
        u[number_of_lines - 1][i] = right_cond;
}
// solving part
int curr_x = 0;
int curr_t = 0;
for (int i = 0; i < number_of_t_points - 1; i++)
{
        curr_t += 1;
        for (int j = 0; j < number_of_lines - 2; j++)
        {
                curr_x += 1;
                double t1 = u[curr_x - 1][curr_t - 1] - u[curr_x][curr_t - 1];
                double t2 = u[curr_x][curr_t - 1] - u[curr_x + 1][curr_t - 1];
                double tt = t1 - t2;
                if (points == 2)
```

```
                        u[curr_x][curr_t] = u[curr_x][curr_t - 1] + tt *
thermal_diffusivity;
                    else
                    {
                        double avg = (u[curr_x - 1][curr_t - 1] + u[curr_x +
1][curr_t - 1]) / 2;
                        u[curr_x][curr_t] = (u[curr_x][curr_t - 1] + tt *
thermal_diffusivity + avg) / 2;
                    }
                }
                curr_x = 0;
            }
            return;
}
void LinearSolve(int size, double **A, double *B)
{
            double *n = new double[size];
            double *m = new double[size];
            for (int i = 0; i < size; i++)
                m[i] = 0;
            for (int iter = 0; iter < 30; iter++)
            {
                for (int i = 0; i < size; i++) {
                    n[i] = (B[i] / A[i][i]);
                    for (int j = 0; j < size; j++) {
                        if (j == i)
                            continue;
                        n[i] = n[i] - ((A[i][j] / A[i][i]) * m[j]);
                        m[i] = n[i];
                    }
                }
```

```
        }
}
void CreateCoeffs(int s, int m, double **k1, double **k2)
{
        double **b = new double*[s + m];
        for (int i = 0; i < s + m; i++)
                b[i] = new double[s + m];
        double *f = new double[s + m];
        double **a = new double*[s + m];
        for (int i = 0; i < s + m; i++)
                a[i] = new double[s + m];
        double **g = new double*[s + m];
        for (int i = 0; i < s + m; i++)
                g[i] = new double[s + m];
        f[0] = 1;
        for (int j = 0; j < s + m; j++)
                g[0][j] = 1;
        for (int i = 0; i < s + m; i++)
        {
                for (int l = 1; l < s + m; l++)
                {
                        for (int j = 0; j < s; j++)
                        {
                                g[l][j] = pow(j, l - 1);
                        }
                        for (int j = 0; j < m; j++)
                        {
                                g[l][j + s] = pow((j - m), (l - 1));
                        }
                        f[l] = pow(i, (l - 1)) / l;
```

```
            }
            double **G = new double*[s + m];
            for (int x1 = 0; x1 < s + m; x1++)
                G[x1] = new double[s + m];
            for (int x1 = 0; x1 < s + m; x1++)
                for (int x2 = 0; x2 < s + m; x2++)
                    G[x1][x2] = g[x1][x2];
            double *F = new double[s + m];
            for (int x1 = 0; x1 < s + m; x1++)
                F[x1] = f[x1];
            LinearSolve(s + m, G, F);
            for (int j = 0; j < s; j++)
                a[i][j] = i * F[j];
            for (int j = 0; j < m; j++)
                b[i][j] = i * F[j + s];
        }
        for (int i = 0; i < s; i++)
            for (int j = 0; j < m; j++)
            {
                k1[i][j] = a[i][j];
                k2[i][j] = b[i][j];
            }
}
```

В.2 Лістинг програми для створення графіків

```
import matplotlib.pyplot as plt
from copy import deepcopy
import numpy as np

# path = "f_3.txt"
path = "f_2.txt"
f = open(path)
x = list()
x1 = list()

for i in f:
    temp = i.split()
    break
f.close()

n = len(temp)
for i in range(n):
    x.append(deepcopy(x1))

f = open(path)
for i in f:
    temp = i.split()
    for j in range(len(temp)):
        x[j].append(float(temp[j]))
f.close()
# =================
# Графіки температури від простору
```

```
# ===============
for i in range(1, len(x) ):
      plt.plot(x[0],x[i])

plt.plot(x[0], x[1])
plt.grid(True)
plt.xlabel("Простір")
plt.ylabel("Температура")

plt.show()

y = list()
y1 = list()
for i in range(len(x[0])):
    for j in range(1, len(x)):
        y1.append(x[j][i])
    y.append(y1)
    y1 = list()
t = np.linspace(0,5,101)
# ============
# Графіки часу від простору
# ===========
for i in range(1, len(y)):
      plt.plot(t,y[i])

plt.grid(True)
plt.xlabel("Час")
plt.ylabel("Температура")

plt.show()
```

В.3 Лістинг програми порівняльного аналізу

```
import matplotlib.pyplot as plt
import numpy as np

data2 = list()
data_line = list()

f2 = open('f_2.txt')
for i in f2:
    t = i.split()
    for j in t:
        data_line.append(float(j))
    data_line = list()
    data2.append(data_line)
f2.close()

data3 = list()
f3 = open('f_3.txt')
for i in f3:
    t = i.split()
    for j in t:
        data_line.append(float(j))
    data_line = list()
    data3.append(data_line)
f2.close()

x_error = list()
temp_error = list()
```

```
for i in range(len(data2) - 1):
    for j in range(len(data2[0]) - 1):
        temp_error.append(data2[i][j] - data3[i][j])
    x_error.append(sum(temp_error)/len(temp_error))
    temp_error = list()

qq = np.linspace(0,np.pi,30)
plt.xlabel("Простір")
plt.ylabel("Похибка")

plt.grid('True')
plt.plot(qq, x_error)
```