НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

«До захисту допущено»
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____2025 р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»
на тему: «Метод композиції розв‘язування збурених параболічних рівнянь»

Виконала:
студентка IV курсу, групи КА-12
Загребельна Марина Віталіївна _____

Керівник:
професор, д.ф.-м.н. Бондаренко Віктор Григорович _____

Консультант з економічного розділу:
доцент, к.е.н. Рощина Надія Василівна _____

Консультант з нормоконтролю:
доцент каф. ММСА, к.ф.-м.н Статкевич Віталій Михайлович _____

Рецензент:
професор, д.ф.-м.н. ФПМ Лось Валерій Миколайович _____

Засвідчую, що у цій дипломній роботі немає запозичень із працю інших авторів без відповідних посилань.

Студентка _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма – «Системний аналіз та управління»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Оксана ТИМОЩУК

«___»_____2025 р.

ЗАВДАННЯ

на дипломну роботу студентці

Загребельній Марині Віталіївні

1. Тема роботи «Метод композиції розв‘язування збурених параболічних рівнянь», керівник роботи професор кафедри ММСА, д.ф.-м.н. Бондаренко Віктор Григорович, затверджені наказом по університету від «26»травня 2025 р. №1759-с.

2. Термін подання студентом роботи

3. Вихідні дані до роботи: різновиди методів композиції розв‘язування збурених параболічних рівнянь, виконання розв‘язків задачі Коші.

4. Зміст роботи:

Аналіз існуючих методів розв’язання збурених параболічних рівнянь. Теоретичне обгрунтування методу композиції збурених параболічних

рівнянь. Побудова алгоритму розв‘язування збуреного рівняння. Функціонально-вартісний аналіз програмного забезпечення.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація для захисту дипломної роботи

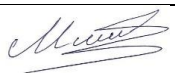
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Надія Василівна		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Отримання завдання	06.02.2025	виконано
2.	Огляд літературних джерел	13.02.2025	виконано
3.	Інтерпретація формули Троттера для збурення оператором першого порядку	15.05.2025	виконано
4.	Побудова алгоритму розв‘язування збуреного рівняння	15.04.2025	виконано
5.	Виконання чисельного експерименту і порівняння розв‘язків задачі Коші, отриманих різними методами	25.04.2025	виконано
6.	Аналіз отриманих результатів	10.05.2025	виконано
7.	Оформлення дипломної роботи	23.05.2025	виконано

Студент  Марина ЗАГРЕБЕЛЬНА

(підпис) (ініціали, прізвище)

Керівник роботи  Віктор БОНДАРЕНКО

(підпис) (ініціали, прізвище)

РЕФЕРАТ

Дипломна робота: 135 с., 9 табл., 21 рис., 2 дод., 13 джерел.

МЕТОД КОМПОЗИЦІЇ, ПАРАБОЛІЧНІ РІВНЯННЯ, ПІВГРУПИ ОПЕРАТОРІВ, ЗАДАЧА КОШІ, ЧИСЕЛЬНЕ МОДЕЛЮВАННЯ, ФОРМУЛА ТРОТТЕРА

У дипломній роботі досліджено задачу побудови розв’язку збуреного параболічного рівняння, де збурення задано диференціальним оператором першого порядку з просторово змінним коефіцієнтом. Такі рівняння описують фізичні процеси з додатковими впливами (наприклад, зсув, дрейф), які потребують спеціального підходу до моделювання.

Як базову ідею використано метод композиції операторів на основі формули Троттера, що дозволяє подати розв’язок задачі Коші у вигляді ітераційної композиції півгруп, породжених окремими частинами операторів. Це дозволяє ефективно розв’язувати складні задачі шляхом поетапного обчислення простіших підзадач.

У ході роботи побудовано алгоритм реалізації методу, проведено теоретичний аналіз збіжності та оцінок похибки, а також виконано чисельне моделювання.

Об’єкт дослідження: математичні методи розв’язування параболічних рівнянь з розподіленими параметрами. Предмет дослідження: застосування методу композиції операторів до збурених параболічних рівнянь. Мета роботи: побудувати, обґрунтувати та реалізувати метод розв’язування задачі Коші зі збуренням на основі формули Троттера, а також проаналізувати точність чисельного розв’язку.

ABSTRACT

Diploma thesis: 135 p., 9 tables, 21 figures, 2 appendices, 13 references.

METHOD OF COMPOSITION, PARABOLIC EQUATIONS, SEMIGROUP OF OPERATORS, CAUCHY PROBLEM, NUMERICAL MODELLING, TROTTER'S FORMULA

The problem of constructing a solution to a perturbed parabolic equation, where the perturbation is given by a first-order differential operator with a spatially variable coefficient, is investigated in this thesis. Such equations describe physical processes with additional effects (e.g., shear, drift) that require a special approach to modelling.

As a basic idea, we use the method of operator composition based on the Trotter formula, which allows us to present the solution of the Cauchy problem in the form of an iterative composition of semigroups generated by individual parts of the operators. This makes it possible to solve complex problems efficiently by gradually calculating simpler subproblems.

In the course of the work, an algorithm for implementing the method is constructed, a theoretical analysis of convergence and error estimates is carried out, and numerical modelling is performed.

Object of research: mathematical methods for solving parabolic equations with distributed parameters. Subject of research: application of the method of operator composition to perturbed parabolic equations. Purpose: to construct, substantiate and implement a method for solving the perturbed Cauchy problem based on Trotter's formula, and to analyse the accuracy of the numerical solution. Research methods: analytical study of convergence, numerical modelling, implementation of iterative algorithms in Matlab and Python.

ЗМІСТ

ВСТУП

У сучасних прикладних задачах математичного моделювання дедалі частіше виникає необхідність опису процесів, що відбуваються в середовищах із розподіленими параметрами — таких як теплообмін, дифузія речовин, розповсюдження сигналів у середовищах тощо. Важливою особливістю таких процесів є їхній **еволюційний характер** — тобто залежність стану системи від часу, що призводить до формулювання **параболічних диференціальних рівнянь**.

Особливий інтерес викликають **збурені параболічні рівняння**, де до основного оператора (наприклад, дифузійного) додається додатковий — збурюючий — член, що моделює додаткові впливи, як-от зсув, конвекція або взаємодія між частинками. Ці збурення часто мають вигляд **диференціального оператора першого порядку** з коефіцієнтами, що залежать від просторової змінної. Такі задачі складні як для аналітичного, так і для чисельного розв’язання, оскільки традиційні методи не завжди гарантують збіжність або точність.

Одним із перспективних підходів до розв’язання таких задач є **метод композиції операторів**, заснований на **формулі Троттера**, яка дозволяє представити розв’язок задачі Коші як композицію напівгруп, породжених окремими частинами операторів. Це дозволяє розділити задачу на послідовність простіших підзадач, що можуть бути реалізовані ефективно, в тому числі з можливістю паралельних обчислень.

У той час як у багатьох випадках застосування такої композиції носить емпіричний характер, **теоретичне обґрунтування збіжності методу для збурених рівнянь є важливою проблемою**, що досі не повністю вирішена. Саме на цю проблему спрямовано дослідження в даній дипломній роботі.

Об’єктом дослідження є задача Коші для збурених параболічних рівнянь, де збурення задано оператором першого порядку. **Предметом дослідження** є ітераційний метод розв’язання цієї задачі, побудований на основі формули Троттера. **Метою роботи** є побудова, теоретичне обґрунтування та чисельна реалізація методу композиції, а також перевірка його точності та ефективності на прикладі моделі дифузії із збуренням.

У рамках дослідження реалізовано ітераційний алгоритм у середовищах **Matlab та Python**, побудовано чисельні розв’язки задачі Коші та здійснено порівняння з аналітичним розв’язком для незбуреного випадку. Отримані результати підтверджують **збіжність методу**, його точність та доцільність використання в задачах моделювання складних фізичних процесів з розподіленими параметрами.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ РОЗВ'ЯЗУВАННЯ ПАРАБОЛІЧНИХ РІВНЯНЬ ТА ЇХ ЗБУРЕНЬ

Вступ до розділу

Параболічні рівняння в частинних похідних є основою математичного моделювання процесів, що характеризуються поступовим перенесенням або розсіюванням фізичних величин, таких як температура, концентрація або ймовірність. Класичні приклади включають теплопровідність, дифузію частинок, біомедичні моделі та фінансові моделі, такі як рівняння Блека-Шоулза. Ці явища мають широке практичне застосування у фізиці, техніці, біології та економіці, що підкреслює необхідність точного та ефективного чисельного моделювання.

Ключовою проблемою в реальних застосуваннях є те, що системи рідко бувають ідеалізованими або ізольованими: в більшості випадків модельована система піддається зовнішнім збуренням. Ці збурення можуть бути детермінованими (наприклад, імпульсні впливи, періодичні сили) або стохастичними (випадкові флуктуації, шумові ефекти). Отже, класичні підходи до розв'язування параболічних рівнянь повинні бути адаптовані та вдосконалені, щоб врахувати ці складнощі для забезпечення реалістичності та прогностичної здатності.

Цей розділ має на меті:

1) детально розглянути математичні основи параболічних диференційних рівнянь, включаючи їх класифікацію на основі структури та властивостей (лінійні, нелінійні, зі сталими та змінними коефіцієнтами, системи рівнянь), а також важливу роль початкових та граничних умов;

2) пояснити значення зовнішніх збурень і показати, чому їх включення в моделі є необхідним для точного відображення реальних явищ;
3) провести критичний огляд основних чисельних методів, що використовуються для розв'язування параболічних рівнянь;
4) надати теоретичний аналіз методу Троттера, сучасного і перспективного підходу до розв'язування збурених параболічних рівнянь, з акцентом на його концептуальній основі, математичному підґрунті та обґрунтуванні вибору цього методу для подальшого дослідження.

Основні чисельні методи, які будуть розглянуті, та порівняні з обраним методом Троттера:

1) Метод скінченних різниць (МР) — один з найпростіших і найпоширеніших методів, особливо ефективний для задач на регулярних сітках;
2) Метод скінченних елементів (МСЕ) — відомий своєю гнучкістю в роботі зі складною геометрією та неоднорідними матеріалами;
3) Спектральні методи — високоточні методи, які добре підходять для задач з гладкими розв'язками.

Цей огляд дозволить виявити сильні та слабкі сторони кожного методу, обґрунтувати вибір методу Троттера як основного підходу для подальшої розробки чисельної моделі та визначити напрямки майбутніх експериментальних досліджень.

1.1 Актуальність задачі моделювання процесів, що описуються параболічними рівняннями

1.1.1 Застосування в фізиці, біомеханіці, фінансовій математиці, техніці

Параболічні диференціальні рівняння в частинних похідних є наріжним каменем математичного моделювання широкого спектру реальних процесів, що залежать від часу. Ці рівняння зазвичай описують системи, в яких дифузія, розсіювання або поступові зміни відбуваються з часом через просторовий дисбаланс таких величин, як температура, концентрація або ймовірність. Їх математична структура - особливо похідна першого порядку за часом і просторові похідні другого порядку - робить їх особливо придатними для опису незворотних процесів, що тяжіють до рівноваги. Таким чином, параболічні рівняння зарекомендували себе як незамінні інструменти в багатьох наукових та інженерних дисциплінах.

Застосування у фізиці

У класичній і прикладній фізиці параболічні диференціальні рівняння найвідомішим прикладом є рівняння теплопровідності (дифузії), яке описує, як теплова енергія поширюється через матеріали. Ця модель лежить в основі теплового аналізу твердих тіл, напівпровідників і рідин. Крім того, дифузія частинок у розріджених газах і плазмі, що регулюється законами Фіка і Фур'є, природно призводить до параболічних моделей. У квантовій статистичній механіці формулювання рівняння Шредінгера в уявному часі зводяться до рівнянь параболічного типу, встановлюючи прямий зв'язок між дифузією і квантовою еволюцією в певних режимах. У фізиці плазми та астрофізиці моделювання переносу температури і густини часто спирається на нелінійні або анізотропні параболічні рівняння другого порядку,

Застосування в біомеханіці та біологічних системах

У біологічному контексті процеси дифузії є дуже поширеними. Транспорт кисню в тканинах, дифузія метаболітів у клітинах і моделювання доставки ліків - все це використовує параболічні рівняння для моделювання просторово розподіленої біологічної активності. Більш складні моделі - реакційно-дифузійні системи - поєднують параболічну дифузію з нелінійною кінетикою реакцій для моделювання складних явищ, таких як формування патернів у морфогенезі (наприклад, патерни Тюрінга), загоєння ран і динаміка росту пухлин. Модель Ходжкіна-Гакслі поширення нервового імпульсу та різні моделі кальцієвого сигналу в нейронах також включають параболічні компоненти для врахування пасивних дифузійних потоків.

Застосування у фінансовій математиці

Перетин фінансів і параболічних PDE (рівнянь часткових похідних) найяскравіше відображається в рівнянні Блека-Шоулза, наріжному камені сучасної теорії ціноутворення опціонів. Це рівняння, яке моделює вартість опціонів європейського типу з плином часу, за формою є дзеркальним відображенням рівняння теплоти і походить від обчислення Ітo, застосованого до стохастичних диференціальних рівнянь. Розширення цієї системи, включаючи модель Хестона для стохастичної волатильності або моделі, що включають стрибки і стохастичні процентні ставки, також дають параболічні або вироджені параболічні PDE. Ці рівняння є не лише теоретично важливими, але й життєво необхідними для ціноутворення в реальному часі та оцінки ризиків у фінансових установах.

Застосування в інженерії та технологічних процесах

Інженерні системи часто покладаються на параболічні рівняння для моделювання тепло- і масопереносу. Сюди входить проектування теплообмінників, систем охолодження в електронних пристроях, а також моделювання навколишнього середовища, наприклад, дифузії забруднювачів

у ґрунті та воді. У цивільному будівництві за допомогою параболічних рівнянь, що залежать від часу, моделюють дифузію вологи в бетоні або аналіз термічних напружень у композитних матеріалах. У машинобудуванні в'язкопружна і термопружна поведінка матеріалів під час перехідних навантажень часто вимагає розв'язання зв'язаних параболічних систем. Крім того, керування розподіленими системами зі зворотним зв'язком, наприклад, регулювання температури у великомасштабних системах, використовує параболічні звичайні диференціальні рівняння для синтезу законів керування.

Ширші наслідки і потреба в ефективних методах

Зважаючи на їхню поширеність, точне та ефективне чисельне опрацювання має як теоретичне, так і практичне значення. Багато реальних задач містять збурення, нелінійності або неоднорідні коефіцієнти, які роблять аналітичні розв'язки неможливими. Це спонукає до розробки вдосконалених чисельних схем, які можуть впоратися з жорсткістю, низькою регулярністю розв'язків та складними граничними умовами. Методи розщеплення операторів і композиції, такі як формули Троттера і Стренга, пропонують елегантні та обчислювально зручні схеми для розв'язання таких задач шляхом декомпозиції їх на простіші підзадачі.

У сучасних додатках, особливо тих, що вимагають високої обчислювальної продуктивності або моделювання в реальному часі, такі методи є не лише математично обґрунтованими, але й незамінними на практиці. Їх застосування охоплює мультифізичне моделювання, великомасштабні симуляції та високопродуктивні обчислювальні середовища.

1.1.2 Збурення як необхідна складова для опису реальних явищ

Хоча ідеалізовані математичні моделі слугують основою для аналізу та розуміння динамічних систем, вони часто не можуть охопити всю складність, притаманну реальним процесам. На практиці жодна фізична, біологічна чи фінансова система не поводиться ідеально рівномірно або ізольовано. Збурення, спричинені зовнішніми впливами, флуктуаціями параметрів, шумом навколишнього середовища або внутрішніми нелінійностями, є повсюдними. Їх врахування в математичних моделях, особливо тих, що включають параболічні диференціальні рівняння в частинних похідних, є важливим для підвищення реалістичності, точності прогнозування та надійності симуляцій.

Збурення можуть набувати різних форм, зокрема:

1. Адитивні або мультиплікативні стохастичні члени, що моделюють екологічні або теплові шуми.
2. Варіації початкових або граничних умов, які відображають неточні вимірювання або мінливі зовнішні умови:
    1) залежні від часу або просторово неоднорідні коефіцієнти, що представляють матеріали з неоднорідними властивостями або системи, що піддаються динамічним зовнішнім силам.
    2) нелінійні умови реакції, що накладаються на дифузійні процеси, які виникають у хімічних, біологічних та екологічних системах.
    3) випадкові або детерміновані зовнішні впливи, що імітують такі впливи, як сезонні цикли, економічні потрясіння або механічні вібрації.

Ці збурення не тільки відображають недосконаліть і мінливість фізичних систем, але часто визначають домінуючу поведінку системи в часі. Наприклад, у динаміці популяції невеликі збурення у початковому розподілі щільності можуть призвести до абсолютно різних еволюційних результатів. У фінансах випадкові коливання цін на активи є основою для сучасного ціноутворення на деривативи. У теплопередачі недосконалий контакт на межі або дефекти матеріалу суттєво змінюють поширення тепла.

З математичної точки зору, введення збурень у параболічні рівняння призводить до значних аналітичних та обчислювальних проблем:

1. Збурення можуть руйнувати регулярність, роблячи класичні розв'язки незастосовними і вимагаючи слабких або узагальнених концепцій розв'язку.
2. У випадку стохастичних параболічних рівнянях стандартні детерміновані методи повинні бути розширені за допомогою стохастичного обчислення, що вимагає застосування таких інструментів, як інтеграли Іто, теорія мартингалів або вінерівські процеси.
3. Збурення можуть викликати нестійкості, особливо в жорстких системах, що вимагає чисельно стійких схем і ретельних стратегій часового кроку.
4. Багатомасштабні збурення - ті, що діють у різних часових або просторових масштабах - вимагають ієрархічних або адаптивних чисельних методів, що збільшує алгоритмічну складність.

Такі труднощі підкреслюють потребу в структурованих чисельних підходах, які можуть підтримувати стабільність і точність за наявності збурень. Традиційні монолітні розв'язувачі можуть стати неефективними або нестабільними, особливо при моделюванні високих розмірностей або тривалих часових інтервалів.

Обґрунтування методів розкладання та композиції операторів

Однією з ефективних стратегій роботи зі збуреними системами є використання методів розщеплення операторів, таких як формула добутку Троттера, яка дозволяє розкласти складні оператори на простіші субоператори, які легше аналізувати та інтегрувати індивідуально. У контексті збурених параболічних рівнянь такі методи мають кілька переваг:

1. Вони відокремлюють детерміновану дифузійну складову від стохастичних або нелінійних збурень.
2. Вони забезпечують кращий контроль над розповсюдженням похибки завдяки ізоляції жорстких і нежорстких членів.
3. Вони пропонують гнучкість для комбінування різних чисельних схем (наприклад, неявних для дифузії, явних для збурень) в межах однієї програми.

Застосування композиційних методів не тільки спрощує чисельну реалізацію, але й часто підвищує обчислювальну ефективність, особливо коли моделювання необхідно повторювати за різних сценаріїв збурень для аналізу чутливості або кількісної оцінки невизначеності.

Отже, збурення - це не просто ускладнення в математичному моделюванні; вони є фундаментальною особливістю майже кожної системи, що представляє практичний інтерес. Включення збурень долає розрив між теоретичною абстракцією і реальною динамікою. Таким чином, як аналітичні, так і чисельні методи повинні розвиватися, щоб врахувати ці складнощі, а операторні методи, такі як метод Троттера, забезпечують багатообіцяючу та універсальну основу. Вивчення і розробка таких методів є важливою частиною сучасної прикладної математики та обчислювальної техніки.

1.2 Класи параболічних рівнянь

1.2.1 Основні типи рівнянь

Параболічні рівняння в частинних похідних відіграють фундаментальну роль у математичному моделюванні залежних від часу процесів, таких як дифузія, теплопровідність, в'язка дисипація, перенесення імпульсу та масоперенос. Їх характерна структура, що складається з часової похідної першого порядку і просторових похідних другого порядку, робить їх добре придатними для опису еволюційних процесів, які прагнуть до рівноваги з часом.

1. **Класичне скалярне параболічне рівняння**

Найбільш поширеним прикладом є рівняння теплопровідності (дифузії), яке в одному просторовому вимірі має вигляд: $\frac{du}{dt} = D\frac{d^2u}{d^2x}$

де $u = u(x,t)$ представляє поле температури або концентрації, а $D > 0$ - коефіцієнт дифузії (або теплопровідності). Це лінійне параболічне диференційне рівняння другого порядку в часі та просторі, який моделює поширення тепла або частинок у середовищі.

2. **Параболічні рівняння зі змінними або неоднорідними коефіцієнтами**

У практичних застосуваннях властивості матеріалів часто змінюються в просторі. Це призводить до таких рівнянь, як

$$\frac{du}{dt} = \nabla * (D(x)\nabla u) + f(x,t)$$

де $D(x)$ - просторово змінний коефіцієнт дифузії, а $f(x,t)$ - внутрішні джерела або стоки. Такі рівняння моделюють дифузію в неоднорідних або

складних середовищах і застосовуються в тепловому аналізі шаруватих матеріалів, геологічних формацій і складних рідин.

3. Нелінійні параболічні рівняння

Багато важливих систем демонструють нелінійну поведінку, де коефіцієнти залежать від самого розв'язку u. Прикладом може бути:

$$\frac{du}{dt} = \frac{d}{dx}(D(u)\ \frac{du}{dx})$$

яке описує нелінійну дифузію, що спостерігається в потоках пористих середовищ, процесах фазових переходів і нелінійної теплопровідності. Іншим ключовим прикладом є рівняння реакції-дифузії:

$$\frac{du}{dt} = D * \Delta u + R(u)$$

де $R(u)$ - член нелінійної реакції, що моделює хімічну кінетику, біологічний ріст або екологічну динаміку.

4. Системи параболічних рівнянь

Складні процеси, що включають декілька взаємодіючих полів, призводять до виникнення систем параболічних рівнянь. Наприклад:

$$\begin{cases} \frac{du}{dt} = D_1 \Delta u + \alpha v \\ \frac{dv}{dt} = D_2 \Delta v + \beta u \end{cases}$$

моделює взаємодію між двома величинами u і v, такими як температура і концентрація в реагуючому середовищі або сигнальні молекули в біологічних тканинах. Ці системи є важливими для мультифізичного моделювання.

5. Вироджені (слабко параболічні) рівняння

У деяких сценаріях коефіцієнт дифузії може зникати або ставати сингулярним в частині області або з часом, що призводить до вироджених

параболічних рівнянь. Вони з'являються в моделях порогових явищ, фазового розшарування або в задачах з вільними границями. Вони вимагають спеціалізованих аналітичних методів і ретельної чисельної обробки через зміну типу рівняння і регулярності розв'язку.

6. Математична класифікація та критерії параболічності

Математично параболічні рівняння класифікуються на основі власних значень головної частини (матриці просторових похідних другого порядку). Рівняння вважається параболічним, якщо головний символ має одне нульове власне значення (що відповідає похідній за часом), а всі інші власні значення строго додатні. Цей критерій відрізняє параболічні рівняння від еліптичних та гіперболічних типів

1.2.2 Зовнішні збурення, початково-крайові умови

У реальних фізичних, інженерних і біологічних системах зовнішні збурення неминучі. Ці збурення виникають через неконтрольовані або частково відомі впливи навколишнього середовища, помилки вимірювань, шуми або флуктуації вхідних параметрів. Вони можуть бути:

1) детермінованими, представленими відомими функціями або встановленими законами (наприклад, періодичні джерела тепла, імпульси);
2) стохастичними, керованими випадковими процесами, такими як білий шум, вінерівські процеси або ланцюги Маркова.

У математичних моделях зовнішні збурення зазвичай вводяться в праву частину параболічного рівняння, змінюючи еволюцію системи.

Включення таких збурень дозволяє дослідникам вивчати стабільність, довготривалу поведінку та чутливість системи до зовнішніх впливів.

Для того, щоб забезпечити правильну постановку математичної моделі, необхідно задати відповідні початкові та граничні умови для параболічного рівняння з частинними похідними.

Початкові умови:

$$u(x,0) = u_0(x), x \in \Omega$$

де (x) задає початковий розподіл (наприклад, температуру, концентрацію), а Ω - просторову область.

Граничні умови можуть бути трьох основних типів:

Діріхле (перший тип) - значення функції фіксується на межі:

$$u(x,t) = g(x,t), x \in \partial\Omega.$$

Неймана (другого тип) - задається нормальна похідна (наприклад, теплового потоку):

$$\frac{du}{dn} = h(x,t), x \in \partial\Omega.$$

Робін (третій тип) - лінійна комбінація функції та її нормальної похідної:

$$\alpha(x)u + \beta(x)\,\frac{du}{dn} = r(x,t), x \in \partial\Omega.$$

За наявності зовнішніх збурень вибір та обробка граничних умов набуває особливого значення. Наприклад, у випадку граничного шуму або флуктуацій вхідних параметрів навколишнього середовища може виникнути потреба у використанні узагальнених граничних умов.

Параболічні задачі, особливо за наявності збурень, часто демонструють високу чутливість до початкових та граничних даних. Невеликі зміни у

початкових розподілах або граничних режимах можуть призвести до суттєвих відмінностей у траєкторіях розв'язку, особливо при довготривалому моделюванні. Це мотивує розробку стійких чисельних схем, таких як операторне розщеплення або композиційні методи, які можуть враховувати збурення вхідних даних, зберігаючи при цьому чисельну стійкість і збіжність.

Зовнішні збурення і правильне формулювання початкових і граничних умов є критично важливими компонентами при моделюванні реальних параболічних процесів. Їх точне врахування забезпечує фізичний реалізм і чисельну надійність моделі, що стає особливо важливим при роботі з невизначеними, збуреними або стохастичними середовищами.

1.3 Огляд чисельних методів для параболічних рівнянь

1.3.1 Метод скінченних різниць

Метод скінченних різниць є одним з найстаріших і найпоширеніших чисельних методів наближення розв'язків диференціальних рівнянь, особливо тих, що виникають у математичній фізиці. Його основна ідея полягає в заміні похідних скінченно-різницевими апроксимаціями, тим самим зводячи неперервну задачу до системи алгебраїчних рівнянь.

Наприклад, похідна першого порядку за часом може бути апроксимована за допомогою прямих (явних) або зворотних (неявних) різниць:

Пряма (явна) схема: $\frac{du}{dt}(x_j, t_n) \approx \frac{u_j^{n+1} - u_j^n}{\Delta t}$

Зворотна (неявна) схема: $\frac{du}{dt}(x_j, t_n) \approx \frac{u_j^n - u_j^{n-1}}{\Delta t}$

В результаті рівняння теплопровідності або подібні параболічні диференційні рівняння можуть бути дискретизовані на регулярній просторово-часовій сітці.

Параболічні рівняння є особливо зручними для скінченнорізницевих схем, оскільки їх еволюційна природа дозволяє побудову ітеративного процесу у часі. Типовими є такі схеми:

1) **явна схема Ейлера** — проста у реалізації, але обмежена умовами стійкості (наприклад, умова Куранта-Фрідріхса-Леві, CFL);
2) **неявна схема** — забезпечує безумовну стійкість, що дозволяє використовувати більші кроки по часу;
3) **схема Кранка-Ніколсона** — напівнеявна схема другого порядку точності, яка забезпечує баланс між стабільністю і точністю.

У загальному вигляді скінченнорізницева схема для одномірного параболічного рівняння може мати вигляд:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{D}{\Delta x^2}\left(u_{j-1}^* - 2u_j^* + u_{j+1}^*\right) + f_j^*$$

залежить від використовуваного часового рівня, залежно від обраної схеми.

Переваги:

1) концептуально простий і легкий у реалізації;
2) природно підходить для структурованих сіток і простих геометрій;
3) може враховувати як явні, так і неявні формулювання;
4) добре підтримується в чисельних програмних середовищах.

Обмеження:

1) менша точність на неоднорідних або нерегулярних сітках;
2) обмежена гнучкість для складної геометрії;
3) явні схеми вимагають невеликих часових кроків для стабільності;
4) може бути чутливим до збурень, якщо не застосовується згладжування або регуляризація.

У задачах зі збуреними параболічними рівняннями **Метод скінченних різниць (МСР)** дозволяє локально модифікувати схему для врахування зовнішніх збурень або стохастичних ефектів. Наприклад:

1) збурені коефіцієнти або вихідні члени можуть бути введені в окремих точках сітки;
2) імпульсивні або випадкові впливи можна моделювати за допомогою стохастичних членів;
3) крок за часом можна адаптивно регулювати, щоб врахувати швидку перехідну поведінку.

Крім того, МСР часто використовується в поєднанні з методом Троттера, коли повне рівняння розкладається на простіші підзадачі, кожна з яких може бути ефективно розв'язана за допомогою скінченно-різницевих дискретизацій.

1.3.2 Метод скінченних елементів для параболічних рівнянь

Метод скінченних елементів (МСЕ) є одним із найпотужніших чисельних методів для розв’язання диференціальних рівнянь у частинних похідних, особливо на складних геометричних областях. На відміну від методу скінченних різниць, який базується на прямій апроксимації похідних, МСЕ ґрунтується на **варіаційному або слабкому формулюванні задачі**, що дозволяє розглядати більш широкий клас функцій, зокрема функції Соболева. Основна ідея МСЕ полягає в апроксимації функції-розв’язку як лінійної комбінації базисних функцій, заданих на локальних підобластях (елементах) розбиття:

$$u_h(x,t) = \sum_{j=1}^{N} U_j(t)\phi_j(x)$$

де $\phi_j(x)$— базисні (локально підтримувані) функції, а $U_j(t)$ — шукані часові коефіцієнти. Такий підхід дозволяє зводити задачу до системи звичайних диференціальних рівнянь у часі.

Для параболічних рівнянь типу:

$\frac{\partial u}{\partial t} - \nabla \cdot (a(x)\nabla u) = f(x,t)$,

МСЕ дозволяє отримати слабке формулювання: знайти $u(t) \in V$, таке що для будь-якого $v \in V$:

$$(\frac{\partial u}{\partial t}, v) + a(u,v) = (f,v)$$

Після просторової дискретизації МСЕ породжує систему звичайних диференціальних рівнянь:

$$M\frac{dU}{dt} + KU = F(t)$$

де:

- M — матриця мас,
- K — жорсткісна матриця,
- U — вектор значень шуканої функції в базисі,
- $F(t)$— вектор навантажень.

Ця система може бути інтегрована по часу за допомогою методів Ейлера, Кранка-Ніколсона, θ-методу тощо.

Переваги МСЕ:

1) висока точність при використанні адаптивних та вищих порядків апроксимації;
2) придатність для задач на складних геометріях і з нерівномірними коефіцієнтами;
3) стійкість до локалізованих збурень і можливість локального уточнення сітки;
4) можливість природного врахування крайових умов (у слабкому сенсі);
5) добре підходить для моделювання розривів, анізотропії, неоднорідностей.

Недоліки:

1) складність реалізації (особливо в багатовимірних задачах);
2) необхідність побудови сітки та обчислення інтегралів на елементах;
3) вища обчислювальна вартість порівняно з методом скінченних різниць при однаковій точності.

Метод скінченних елементів особливо корисний у випадках збурених або стохастичних параболічних задач, оскільки дозволяє:

1. Ефективно враховувати випадкові або детерміновані збурення через варіаційний підхід;
2. Розглядати збурення в геометрії, коефіцієнтах, початково-крайових умовах;
3. Інтегрувати методи типу Троттера або методи розщеплення на підпростори;
4. Працювати із задачами в слабких просторах, коли класичні похідні не визначені.

Метод скінченних елементів є потужним інструментом для розв’язання параболічних рівнянь, особливо у складних, збурених або неоднорідних середовищах. Його здатність гнучко адаптуватись до задачі, підтримувати високу точність та інтегруватись з іншими чисельними методами робить МСЕ незамінним у сучасному чисельному аналізі.

1.3.3 Спектральні методи для розв’язання параболічних рівнянь

Спектральні методи — це високоточні чисельні підходи, що ґрунтуються на апроксимації шуканої функції глобальними базисними функціями, такими як тригонометричні функції (метод Фур’є), поліноми Чебишева, Лежандра, Ерміта тощо. На відміну від методів з локальною апроксимацією (як-от метод скінченних різниць або скінченних елементів), спектральні методи

застосовують розклад по всій обчислювальній області, що забезпечує експоненційно високу точність для гладких функцій.

Типове представлення розв'язку має вигляд:

$$u(x,t) = \sum_{k=0}^{N} \widehat{u_k}(t)\phi_k(x)$$

де $\phi_k(x)$ — система ортогональних функцій, а $\widehat{u_k}(t)$ — спектральні коефіцієнти, які визначаються з рівняння.

У випадку параболічного рівняння, наприклад рівняння теплопровідності:

$$\frac{\partial u}{\partial t} = Lu + f(x,t),$$

де L — диференціальний оператор (наприклад, лапласіан), спектральний метод передбачає проєкцію рівняння на обмежений підпростір базисних функцій, що зводить задачу до системи звичайних диференціальних рівнянь відносно спектральних коефіцієнтів:

$$\frac{d\widehat{u_k}}{dt} = \lambda_k \widehat{u_k} + \widehat{f_k}(t), k = 0,1,\dots,N$$

де λ_k — власні значення оператора L, які визначають швидкість згасання кожного модального компонента.

У випадку періодичних граничних умов найчастіше використовують метод Фур'є. Для обмежених інтервалів із неоднотипними граничними умовами застосовуються поліноміальні спектральні методи, такі як метод Чебишева.

Переваги:

1) надзвичайно висока точність при малій кількості ступенів вільності (експоненційна збіжність для гладких задач);
2) компактне представлення розв'язку, що дозволяє ефективний аналіз (наприклад, модальний аналіз);
3) добре підходить для задач із регулярною геометрією та гладкими коефіцієнтами.

Недоліки:

1) нестійкість при розв’язуванні задач із розривами або негладкими функціями (ефект Ґіббса);
2) важко застосовувати до складних геометрій або неоднорідних середовищ;
3) невелика гнучкість у врахуванні локальних особливостей або локальних збурень.

Застосування спектральних методів до збурених параболічних рівнянь вимагає обережності. Основні підходи включають:

1) **модальний аналіз збурення**: дозволяє вивчати, як збурення впливає на спектр системи або на окремі моди (власні функції);
2) **гібридні методи**: поєднання спектральних методів із локальними (наприклад, методом скінченних елементів) для обробки областей зі збуреннями;
3) **стохастичні спектральні методи**: у разі випадкових збурень застосовуються розклади за поліномами Герміта (наприклад, Polynomial Chaos Expansion).

Збурення часто проявляється у вигляді збурених операторів $L + \delta L'$, що у спектральному просторі може змінювати поведінку мод, зокрема високочастотних, які найчутливіші до флуктуацій.

Спектральні методи демонструють надзвичайну точність та ефективність у розв’язанні гладких, регулярних параболічних задач. У випадках із збуреннями, особливо стохастичними або локальними, методи потребують адаптації або комбінування з іншими підходами. Тим не менше, їхня здатність до модального аналізу робить спектральні методи незамінними в теоретичній і прикладній математиці.

1.3.4 Порівняння традиційних методів із методом Троттера

Формула Троттера (метод Троттера) є фундаментальною технікою для чисельного розв'язання еволюційних рівнянь, особливо таких, де оператор можна природно розкласти на простіші складові. Для лінійного еволюційного рівняння вигляду:

$$\frac{du}{dt} = (A + B)u,$$

метод Троттера апроксимує розв'язок на малому часовому кроці Δt за формулою:

$$u(t + \Delta t) \approx e^{\Delta tA} e^{\Delta tB} u(t),$$

що дозволяє розділити складний оператор на простіші частини, які вирішуються послідовно. Метод особливо ефективний для параболічних рівнянь із збуреннями, де оператор природно розділяється на дифузійну частину та збурений або реакційний компонент.

При порівнянні традиційних чисельних методів (метод скінченних різниць (МСР), метод скінченних елементів (МСЕ) і спектральні методи) з методом Троттера враховуються такі ключові аспекти:

1. **Точність і збіжність:**
    1) **МСР —** точність першого або другого порядку за часом і простором залежно від схеми; точність може знижуватися поблизу меж або для складних геометрій.
    2) **МСЕ —** забезпечує високу точність, особливо при використанні адаптивних сіток і базисів вищого порядку.
    3) **Спектральні методи —** експоненційна збіжність для гладких задач, проте чутливі до розривів (ефект Ґіббса).
    4) **Метод Троттера —** має **помилку розщеплення операторів**, зазвичай першого порядку, однак при використанні симетричного розщеплення (схема Стренга) забезпечує другий порядок

точності. Висока глобальна точність за умови точного розв’язання підзадач.

2. **Стійкість:**
    1) **МСР —** явні схеми мають жорсткі умови стійкості (умова Куранта–Фрідріхса–Леві); неявні схеми є безумовно стійкими.
    2) **МСЕ —** висока стійкість, зокрема для жорстких задач і складних геометрій.
    3) **Спектральні —** стійкі при правильній дискретизації часу; чутливі до негладких даних.
    4) **Метод Троттера —** успадковує стійкість від підрозв’язувачів; особливо корисний для **жорстких задач**, де розщеплення дозволяє відокремити жорсткі й нежорсткі частини.
3. **Обчислювальна ефективність:**
    1) **МСР —** проста й ефективна реалізація на регулярних сітках; менш ефективна для складних геометрій.
    2) **МСЕ —** висока обчислювальна вартість через побудову сітки та обчислення інтегралів; добре масштабується при паралельних обчисленнях.
    3) **Спектральні методи —** ефективні для періодичних областей; дорогі для складних геометрій.
    4) **Метод Троттера —** дозволяє **значно зменшити обчислювальні витрати** за рахунок незалежного розв’язання простіших підзадач, підтримує паралельну обробку.
4. **Гнучкість:**
    1) **МСР —** обмежений застосуванням до регулярних сіток; складно адаптувати до нерегулярних областей.
    2) **МСЕ —** дуже гнучкий щодо області розв’язання і граничних умов.
    3) **Спектральні методи —** обмежена гнучкість для складних геометрій.

4) **Метод Троттера —** гнучкий при наявності природного розщеплення задачі; може комбінувати будь-який традиційний метод для підзадач (наприклад, МСР для дифузії, МСЕ для реакції).

5. **Обробка збурень:**
   1) **МСР/МСЕ/Спектральні методи —** збурення безпосередньо вносяться у дискретизовану задачу, що може ускладнювати задачу або посилювати жорсткість.
   2) **Метод Троттера —** природно відокремлює збурення в окремі компоненти, дозволяючи застосувати спеціалізовані чисельні методи (наприклад, стохастичні методи для шумових збурень).

Метод Троттера особливо ефективний у випадках, коли оператор має виражену структуру розщеплення — наприклад, у задачах дифузії-реакції або в системах із виразними лінійними та нелінійними частинами. Традиційні методи, хоч і потужні, часто вимагають більших обчислювальних витрат або не мають такої модульності, як метод розщеплення.

Наприклад:

1) у **системах дифузії-реакції** метод Троттера дозволяє чергувати розв’язування дифузійної частини (неявним методом) і реакційної частини (явним або аналітичним методом), оптимізуючи точність і витрати;
2) для **стохастичних задач** розщеплення між детермінованою та стохастичною частинами дозволяє застосовувати різні підходи до кожної з них.

Попри те, що метод скінченних різниць**,** метод скінченних елементів і спектральні методи є перевіреними й ефективними, їх використання може бути обмежене обчислювальними витратами, обмеженнями стабільності або складністю адаптації до збурених систем. **Метод Троттера** забезпечує модульний, ефективний і часто більш стабільний підхід для задач із природною структурою розщеплення. Його здатність інтегруватися з

традиційними схемами робить його надзвичайно перспективним інструментом для сучасних обчислювальних задач із параболічними рівняннями та їх збуреннями.

Висновки до розділу

У цьому розділі було здійснено **комплексний огляд теоретичних основ і чисельних методів** для розв’язування параболічних диференціальних рівнянь із зовнішніми збуреннями. Основні результати можна підсумувати наступним чином:

1. **Актуальність параболічних рівнянь:**
    1) Параболічні рівняння є ключовим математичним інструментом для моделювання широкого спектра явищ у фізиці, біології, інженерії та фінансовій математиці.
    2) Було встановлено, що в реальних задачах практично завжди присутні **зовнішні збурення**, які можуть мати детермінований або стохастичний характер. Врахування цих збурень є критично важливим для досягнення фізичної достовірності та підвищення прогностичної здатності моделі.
2. **Класифікація параболічних рівнянь:**
    1) Розглянуті основні типи параболічних рівнянь: з постійними й змінними коефіцієнтами, нелінійні рівняння, системи рівнянь і дегенеровані випадки.
    2) Визначено роль початково-крайових умов та їхній вплив на стабільність і точність чисельного розв’язку, особливо при наявності збурень.
3. **Аналіз традиційних чисельних методів:**

1) **Метод скінченних різниць** показав себе як ефективний для задач на регулярних сітках, але має обмежену здатність адаптуватися до складних геометрій і сильно чутливий до вибору часу при явних схемах.
2) **Метод скінченних елементів** відзначився високою точністю і здатністю працювати на складних областях, що робить його універсальним для широкого спектра практичних задач.
3) **Спектральні методи** забезпечують надзвичайно високу точність при гладких розв'язках, однак мають обмеження при роботі з локалізованими збуреннями або розривами.

4. **Вивчення методу Троттера:**

Метод Троттера дозволяє **розділити оператор задачі на окремі компоненти**, що значно спрощує обчислювальну процедуру. Його ключовою особливістю є можливість поетапного розв'язування окремих підзадач, кожна з яких може оброблятися оптимальним методом.

Аналіз показав, що метод Троттера забезпечує:

1) **модульність і гнучкість** — дозволяє комбінувати різні чисельні підходи (наприклад, для дифузійної та реакційної частин);
2) **обчислювальну ефективність** — дає змогу паралелізувати обчислення та зменшити витрати, особливо у випадках жорстких або багатокомпонентних систем;
3) **підвищену стійкість** — ізольоване розв'язування окремих частин зменшує ризик чисельних нестабільностей;
4) **контроль над похибками** — можливість застосовувати високоточні методи для кожного з підоператорів окремо;
5) **гнучкість у роботі зі збуреннями** — природний поділ детермінованих і стохастичних компонентів дозволяє ефективно інтегрувати специфічні методики, наприклад стохастичні чисельні схеми для шумових впливів.

Порівняння методів:

Порівняльний аналіз показав, що хоча традиційні методи добре працюють у класичних задачах, їх ефективність помітно знижується у випадках складних або збурених систем. Метод Троттера вигідно відрізняється **модульністю** і **сумісністю з будь-якою геометрією та структурою задачі**, що робить його універсальним для сучасних високоточних моделювань.

Було виявлено, що застосування методу Троттера дає особливі переваги у задачах:

1) з сильно вираженим розділенням фізичних процесів (наприклад, дифузія + реакція),
2) у багатошарових або мультифізичних моделях,
3) при наявності стохастичних або детермінованих збурень.

Проведене дослідження дозволило сформувати цілісне уявлення про існуючі методи чисельного розв’язання параболічних рівнянь і зробити висновок, що метод Троттера, завдяки своїй гнучкості та адаптивності, є **найбільш перспективним підходом** для моделювання параболічних задач із збуреннями. Цей метод дозволяє не лише підвищити точність і стабільність розрахунків, але й забезпечує **оптимальний баланс між обчислювальною вартістю та якістю результатів**.

Отримані результати обґрунтовують доцільність подальшого застосування методу Троттера в практичних задачах та його інтеграцію з іншими сучасними чисельними техніками для підвищення ефективності обчислювальних алгоритмів.

РОЗДІЛ 2. ТЕОРЕТИЧНЕ ОБҐРУНТУВАННЯ МЕТОДУ КОМПОЗИЦІЇ ДЛЯ ЗБУРЕНИХ ПАРАБОЛІЧНИХ РІВНЯНЬ

2.1 Постановка задачі Коші з операторним збуренням першого порядку

Еволюційні рівняння вигляду

$$\frac{\partial u}{\partial t} = \Phi(u) \tag{2.1}$$

де Φ —диференціальний оператор по просторовій змінній, являють собою математичні моделі об‘єктів з розподіленими параметрами. Позначимо через H_t породжену генератором Φ півгрупу (в загальному випадку нелінійну), тобто

$$\begin{cases} \frac{\partial u}{\partial t} = \Phi(u) \\ u(0,x) = \varphi(x) \end{cases} \Longrightarrow u(t,x) = (H_t\varphi)(x)\ .$$

Термін півгрупа означає, що оператори H_t , що діють в просторі функцій, задовольняють співвідношення

$H_tH_s = H_{t+s}\ ,\ t \geq 0,\ s \geq 0, H_0 = I$ —одиничний оператор.

Якщо $\Phi -$ лінійний оператор, $\Phi(u) = Lu,$ то півгрупа H_t для кожного t також є лінійним оператором і позначається $H_t = e^{tL}$.

Загальну теорію лінійних півгруп започатковано в 50-х роках в роботах Хілле, Філіппса, Іосіда, Като, в сучасній формі вона викладена в монографіях [1] , [2]. Таким чином, розв‘язок лінійної задачі Коші

$$\begin{cases} \frac{\partial u}{\partial t} = Lu \\ u(0,x) = \varphi(x) \end{cases}$$

має вигляд $u(t,x) = (e^{tL}\varphi)(x)$.

В окремих випадках лінійна півгрупа e^{tL} відома. Наприклад, якщо L — еліптичний оператор,

$$L = \sum_{j,k=1}^{n} a_{jk}(\boldsymbol{x}) \frac{\partial^2}{\partial x_j \partial x_k}, \boldsymbol{x} \in R^n \tag{2.2}$$

то півгрупа являє собою інтегральний оператор, ядро якого $p(t,x,y)$ називається фундаментальним розв‘язком оператора L, і розв‘язок задачі Коші для параболічного рівняння

$$\begin{cases} \frac{\partial u}{\partial t} = Lu \\ u(0,x) = \varphi(x) \end{cases}$$

зображується у вигляді

$$u(t,\boldsymbol{x}) = (e^{tL}\varphi)(\boldsymbol{x}) = \int_{R^n} \varphi(y) p(t,\boldsymbol{x},\boldsymbol{y})\, d\boldsymbol{y}.$$

Зауважимо, що $p(t,\boldsymbol{x},\boldsymbol{y})$ має ймовірнісну інтерпретацію—перехідна щільність дифузійного процесу і будується за методом параметриксу. Властивості півгруп, що відповідають параболічним рівнянням, викладено як в класичній монографії [3] , так і в більш сучасній [4].

У ряді випадків математичною моделлю об‘єкта є рівняння (2.1), де $\Phi(u) = Lu + f(u)$, $L -$ еліптичний оператор (2.2), f −збурення. Відповідне параболічне рівняння

$$\frac{\partial u}{\partial t} = Lu + f(u) \tag{2.3}$$

назвемо збуреним. Якщо f не містить операції диференціювання, то рівняння (2.3) називається напівлінійним, або рівнянням реакції-дифузії. Такі рівняння і відповідні системи рівнянь вигляду

$$\frac{\partial u_r}{\partial t} = Lu_k + f(u_1, \ldots . u_n), \quad k = 1, \ldots, n \tag{2.4}$$

використовуються як моделі в біології, де u_k −щільність популяції. Огляд таких моделей наведено в монографії [5]. В роботі [6] детально розглянуто приклад рівняння (2.3) — рівняння Колмогорова-Петровського-Піскунова-Фішера (КППФ):

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + u(1-u).$$

Доведено, що розв‘язки КППФ мають хвильовий характер, тобто суттєво відрізняються від розв‘язків лінійного параболічного рівняння.

2.2 Метод дослідження рівняння Колмогорова-Петровського-Піскунова-Фішера (КППФ)

Позначимо через Q_t півгрупу, породжену збуренням f:

$$\begin{cases} \frac{\partial u}{\partial t} = f(u) \\ u(0,x) = \varphi(x) \end{cases} \Longrightarrow u(t,x) = (Q_t\varphi)(x).$$

Оператори $Q_t\, e^{tL}$ та $e^{tL}Q_t$ називаються композицією півгруп. Зокрема, якщо f −лінійний опетатор, $f(u) = L_1 u$, то $Q_t\varphi = e^{tL_1}\varphi$.

Для півгрупи, породженої генератором $L + L_1$, в роботах [7,8] доведено формулу Троттера, яка в монографії [1] для самоспряжених операторів L та L_1, зображена у вигляді

$$e^{T(L+L_1)} = \lim_{n\to\infty}\left(e^{\frac{T}{n}L}e^{\frac{T}{n}L_1}\right)^n.$$

Зауважимо, що для комутуючих операторів $LL_1 = L_1L$ виконується рівність

$$e^{t(L+L_1)} = e^{tL}e^{tL_1} = e^{tL_1}e^{tL}$$

В роботах $[9-11]$ формулу Троттера узагальнено для нелінійного збурення $f(u)$, що не містить похідних—за деяких умов доведено співвідношення

$$H_T = \lim_{n\to\infty}\left(Q_{\frac{T}{n}}\ e^{\frac{T}{n}L}\right)^n .$$

Оператори $Q_t\ e^{tL}$ та $e^{tL}Q_t$ називаються композицією півгруп.

Наведемо деякі міркування, що виправдовують вивчення композиції півгруп. Нехай $u(t), t>0$—функція, яка набуває значення в деякому нормованому просторі X, що задовольняє еволюційному рівнянню

$$\frac{du}{dt} = \Phi(u) \qquad (2.5)$$

де оператор Ф:X→X. Наслідуючи загальноприйняту термінологію, назвемо рівняння (2.5) динамічною системою, а функцію $u(t)$ характеристикою цієї динамічної системи. Якщо Ф:$R^n \to R^n$, тобто

$$u(t) = \big(u_1(t), \dots, u_n(t)\big)$$

то (2.5) є автономною системою звичайних диференціальних рівнянь, і відповідна динамічна система являє собою об'єкт із зосередженими параметрами (ОЗП, object with lumped parameters).

Якщо X—функціональний простір,

$$u(t) = \boldsymbol{u}(t,\boldsymbol{x}) = \big(u_1(t,\boldsymbol{x}), \dots, u_n(t,\boldsymbol{x})\big)$$

де $\boldsymbol{x}$ —просторова змінна переменная, $\boldsymbol{x}\in R^d$, і оператор Ф містить дифференціювання по цій змінній, то динамічна система відповідає об'єкту із розподіленими параметрами (ОРП, distributed object DO), моделлю якого є система рівнянь

$$\frac{\partial u_i}{\partial t} = F_i(x, u_1, \dots, u_N, \nabla u_k, .. \nabla^m u_k\ ) \qquad (2.6)$$

Зокрема, це може бути напівлінійна система (2.4) з еліптичним оператором другого порядку.

Нехай $(u_1(t), \dots, u_n(t))$—характеристика ОЗП. Прикладами таких об'єктів, зокрема, є спільнота хижак-жертва, що описується системою Лотки-Вольтерра:

$$\begin{cases} \frac{du_1}{dt} = f_1(u_1, u_2) \\ \frac{du_2}{dt} = f_2(u_1, u_2) \end{cases}$$

де $u_1(t)$, $u_2(t)$— щільність популяції жертви та хижака.

Якщо ж об'єкт, що досліджується, спочатку описуваний системою звичайних диференціальних рівнянь, трансформується таким чином, що його характеристики залежать від просторової змінної, то він переходить в клас об'єктів з розподіленими параметрами. Прикладами таких трансформацій є міграція популяцій по двовимірному ареалу або наявність протяжності об'єкта. При такому переході математичною моделлю ОРП традиційно постулюється система напівлінійних рівнянь (2.4) (при цьому строге обґрунтування, як правило, відсутнє). Інакше, введення просторової змінної враховується дифузійним доданком Lu, що викликає деякі сумніви з приводу адекватності моделі. Можливо, вища адекватність відповідає композиції півгруп.

Зауваження. Для комутуючих генераторів L та L_1 справедлива формула

$$e^{t(L+L_1)} = e^{tL}e^{tL_1} = e^{tL_1}e^{tL}.$$

Аналогічне співвідношення має місце і для генератора $L_1 + f$, де $L_1 u = b(x)\frac{\partial u}{\partial x}$, а збурення $f(u)$ являє собою скалярну функцію. Позначимо через $r(t, a) = Q_t a$ розв‘язок задачі Коші

$$\begin{cases} \frac{dr}{dt} = f(r) \\ r(0) = a \end{cases}$$

Тоді півгрупа H_t для збуреного рівняння

$$\frac{\partial u}{\partial t} = b(x)\frac{\partial u}{\partial x} + f(u)$$

співпадає з композицією півгруп:

$$\begin{cases}\frac{\partial u}{\partial t} = b(x)\frac{\partial u}{\partial x} + f(u) \\ u(0,x) = \varphi(x)\end{cases} \Rightarrow u(t,x) = r\big(t, (e^{tL_1}\varphi)(x)\big)$$

тобто $H_t\varphi = Q_t e^{tL_1}\varphi = e^{tL_1}\, Q_t\varphi$

В дипломній роботі побудовано ітерації за формулою Троттера для півгрупи, що відповідає для збуреному рівнянню

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + b(x)\frac{\partial u}{\partial x},$$

тобто $L = \frac{\partial^2}{\partial x^2}$, $L_1 = b(x)\frac{\partial}{\partial x}$, де коефіцієнт $b(x)$ задовольняє умов

$$\begin{cases} \varepsilon \le |b(x)| \le c_1 \\ \left|b^{(k)}(x)\right| \le c_1, k = 1{,}2\end{cases}$$

За цих умов визначено функцію

$g(t,x) = F^{-1}\big(t + F(x)\big)$, де $F(x) = \int \frac{dx}{b(x)}$, F^{-1} – функція, обернена до $F(x)$, і розв‘язок задачі Коші для рівняння першого порядку

$$\begin{cases}\frac{\partial v}{\partial t} = b(x)\frac{\partial v}{\partial x} \\ v(0,x) = \varphi(x)\end{cases} \tag{2.7}$$

має вигляд $v(t,x) = (e^{tL_1}\varphi)(x) = \varphi\big(g(t,x)\big)$.

2.3 Збіжність методу та оцінка точності

Похідні функції $g(s,x)$:

$\frac{\partial g}{\partial s} = (F^{-1})'\big(s + F(x)\big) = \frac{1}{F'\left(F^{-1}(s+F(x))\right)} = b\big(g(s,x)\big)$;

$\frac{\partial g}{\partial x} = (F^{-1})'\big(s + F(x)\big)\frac{1}{b(x)} = \frac{b\big(g(s,x)\big)}{b(x)}$;

$\frac{\partial^2 g}{\partial x^2} = \frac{b\big(g(s,x)\big)}{b^2(x)}\Big(b'\big(g(s,x)\big) - b'(x)\Big)$.

Звідси випливають оцінки

$$\begin{cases} |\, g(s,x) - x| \le c_2 s \\ \left|\frac{\partial g}{\partial s}\,(s,x)\right| < c_2 \\ \left|\frac{\partial^2 g}{\partial x^2}\,(s,x)\right| < c_2 s\end{cases} \tag{2.8}$$

Півгрупа, породженна генератором $L = \frac{\partial^2}{\partial x^2}$, має вигляд

$(e^{tL}\varphi)(x) = \int_{-\infty}^{\infty} \varphi(y)p(t,x,y)dy$, де $p(t,x,y) = \frac{1}{2\sqrt{\pi t}} exp\left\{-\frac{(x-y)^2}{4t}\right\}$.

Дослідимо властивості ядра інтегрального оператора $e^{tL_1}e^{tL}$—функції

$$q(t,x,y) = p(t,g(t,x),y) = \frac{1}{2\sqrt{\pi t}} exp\left\{-\frac{(g(t,x)-y)^2}{4t}\right\},$$

що задовольняє очевидної оцінки

$$q(t,x,y) < e^{\frac{c_1}{2}|x-y|}\, p(t,x,y) \tag{2.9}$$

Позначимо p' та p''—похідні функції $p(\alpha,\beta,\gamma)$ по другій змінній

$$p'(\alpha,\beta,\gamma) = \frac{\partial}{\partial\beta}\, p(\alpha,\beta,\gamma),\ p''(\alpha,\beta,\gamma) = \frac{\partial^2}{\partial\beta^2}\, p(\alpha,\beta,\gamma)$$

та обчислимо похідні функції $q(t,x,y)$.

$$\frac{\partial}{\partial t}q(t,x,y) = \frac{\partial}{\partial t}p(t,g(t,x),y) + p'(t,g(t,x),y)\ b\big(g(t,x)\big);$$

$$\frac{\partial}{\partial x}q(t,x,y) = \frac{\partial}{\partial x}p(t,g(t,x),y) = p'(t,g(t,x),y)\frac{b\big(g(t,x)\big)}{b(x)};$$

$$\frac{\partial^2}{\partial x^2}q(t,x,y) = \frac{\partial}{\partial x}\left(p'(t,g(t,x),y)\frac{b\big(g(t,x)\big)}{b(x)}\right) =$$

$$= \frac{1}{b^2(x)}p''(t,g(t,x),y)b^2\big(g(t,x)\big) +$$

$$+\frac{1}{b^2(x)}p'(t,g(t,x),y)b\big(g(t,x)\big)\Big(b'\big(g(t,x)\big) - b'(x)\Big).$$

Звідси випливає, що функція $q(t,x,y)$ задовольняє рівнянню

$$\frac{\partial q}{\partial t} = \frac{\partial^2 q}{\partial x^2} + b(x)\frac{\partial q}{\partial x} + m(t,x,y),$$

де нев‘язка

$$m(t,x,y) = \frac{1}{b^2(x)}p''(t,g(t,x),y)\Big(b^2(x) - b^2\big(g(t,x)\big)\Big) -$$

$$-\frac{1}{b^2(x)}p'(t,g(t,x),y)b\big(g(t,x)\big)\Big(b'\big(g(t,x)\big) - b'(x)\Big) \tag{2.10}$$

Твердження 1. Нев‘язка m задовольняє нерівності

$$|m(t,x,y)| < c_3\left(1 + \frac{(x-y)^2}{t}\right)e^{\frac{c_1}{2}|x-y|}\, p(t,x,y)$$

Доведення. Оскільки

$$p'(t,g(t,x),y) = -\frac{g(t,x)-y}{2t}q(t,x,y),$$

$$p''(t,g(t,x),y) = \left(-\frac{1}{2t} + \frac{(g(t,x)-y)^2}{4t^2}\right)q(t,x,y),$$

твердження випливає з оцінок (2.8) та (2.9) ■

Оцінимо вираз

$$\int_{-\infty}^{\infty}\left(\frac{(x-y)^2}{t}\right)^k e^{a|y-x|}p(t,x,y)dy,\ k \in 1,2, a > 0$$

Твердження 2. Справедлива оцінка

$$\int_{-\infty}^{\infty}\left(\frac{(x-y)^2}{t}\right)^k e^{a|x-y|}p(t,x,y)dy < Ce^{Bt}\ ,\ B > a^2$$

Доведення. Скористаємось формулою

$$\mathrm{E}e^{a|\xi|} = 2e^{\frac{a^2\sigma^2}{2}}\Phi(a\sigma),\ \text{де } \xi\sim\aleph(0;\sigma^2)\ , \Phi(z) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{z} e^{-\frac{x^2}{2}}dx,$$

яку зобразимо у вигляді

$$\frac{\sqrt{\lambda}}{2\sqrt{\pi t}}\int_{-\infty}^{\infty} e^{a|y|}e^{-\lambda\frac{y^2}{4t}}dy = 2e^{\frac{a^2t}{\lambda}}\Phi\left(\frac{a\sqrt{2t}}{\sqrt{\lambda}}\right)$$

і виконаємо диференціювання за параметром λ. Зокрема, якщо $k = 1$,

$$\frac{1}{\sqrt{\lambda}}e^{\frac{a^2t}{\lambda}}\Phi\left(\frac{a\sqrt{2t}}{\sqrt{\lambda}}\right) - \frac{\sqrt{\lambda}}{2\sqrt{\pi t}}\int_{-\infty}^{\infty}\frac{y^2}{t}e^{a|y|}e^{-\lambda\frac{y^2}{4t}}dy =$$

$$= -2\frac{a^2y}{\lambda^3}e^{\frac{a^2t}{\lambda}}\Phi\left(\frac{a\sqrt{2t}}{\sqrt{\lambda}}\right) - \frac{a\sqrt{2t}}{\lambda\sqrt{\lambda}}e^{\frac{a^2t}{\lambda}}\Phi'\left(\frac{a\sqrt{2t}}{\sqrt{\lambda}}\right).$$

Покладемо λ= 1 і замінимо y на $y - x$. Отримаємо:

$$\int_{-\infty}^{\infty}\frac{(x-y)^2}{t}e^{a|x-y|}p(t,x,y)dy =$$

$$= 4e^{a^2t}\Phi\left(a\sqrt{2t}\right) + 8a^2te^{a^2t}\Phi\left(a\sqrt{2t}\right) + 4\,a\sqrt{2t}e^{a^2t}\Phi'\left(a\sqrt{2t}\right) <$$

$$< Ce^{a^2t}(1+t)$$

Повторне диференціювання приводить до оцінки

$$\int_{-\infty}^{\infty}\frac{(x-y)^4}{t^2}e^{a|x-y|}p(t,x,y)dy < Ce^{a^2t}(1+t^2),$$

і твердження випливає з нерівності $1 + t^m < c(\varepsilon,m)e^{\varepsilon t}$

Розглянемо функцію $M(t,x) = \int_{-\infty}^{\infty}h(y)m(t,x,y)dy$

Наслідок. Справедлива нерівність

$$M^2(t,x) < c_4 e^{c_5 t} \int_{-\infty}^{\infty} h^2(y) p(s,x,y) dy \qquad (2.11)$$

Доведення. За нерівністю Коші-Буняковського та за твердженням 1

$$M^2(s,x) < 2c_3^2 \int_{-\infty}^{\infty} h^2(y) p(s,x,y) dy \int_{-\infty}^{\infty} e^{c_1|x-y|} \left(1 + \frac{(x-y)^4}{t^2}\right) p(s,x,y) dy.$$

За твердженням 2, другий множник оцінюється експонентою:

$$\int_{-\infty}^{\infty} e^{c_1|x-y|} \left(1 + \frac{(x-y)^4}{t^2}\right) p(s,x,y) dy < c_4 e^{c_5 t}$$

2.4 Ітераційна процедура за формулою Троттера

Побудуємо ітераційну процедуру за формулою Троттера. Позначимо:

1)$\Delta_1, \Delta_2, \dots, \Delta_n$ —розбиття відрізку $[0;T]$, $T \le S$:

$$\Delta_k = \left[(k-1)\frac{T}{n}; k\frac{T}{n}\right],$$

$$[0;T] = \cup_{k=1}^{n} \Delta_k \; ; \; t \in \Delta_k \Leftrightarrow t = s + (k-1)\frac{T}{n}, \; 0 \le s \le \frac{T}{n}.$$

та визначимо функції

$$r_1(s,x) = \int_{-\infty}^{\infty} \varphi(y) q(s,x,y) dy$$

$$r_2\left(s + \frac{T}{n}, x\right) = \int_{-\infty}^{\infty} r_1\left(\frac{T}{n}, y\right) q(s,x,y) dy$$

……

$$r_j\left(s + \frac{(j-1)T}{n}, x\right) = \int_{-\infty}^{\infty} r_{j-1}\left(\frac{(j-1)T}{n}, y\right) q(s,x,y) dy, \; j \le n,$$

$$r_j\left(\frac{jT}{n}, x\right) = r_{j+1}\left(\frac{jT}{n}, x\right).$$

В термінах півгруп

$$r_j\left(s+\frac{(j-1)T}{n}\right)=e^{sL_1}e^{sL}\,r_{j-1}\left(\frac{(n-1)T}{n}\right),\ r_j\left(\frac{jT}{n},x\right)=\left(\,e^{\frac{T}{n}L_1}\ \ e^{\frac{T}{n}L}\right)^j\varphi.$$

Зокрема,

$$r_n(T,x)=\int_{-\infty}^{\infty}r_{n-1}\left(\frac{(n-1)T}{n},y\right)q\left(\frac{T}{n},x,y\right)dy=\ \left(\left(e^{\frac{T}{n}L_1}e^{\frac{T}{n}L}\right)^n\varphi\right)(x)\qquad(2.12)$$

Розв'язок задачі Коші позначимо $u(t,x)$.

Теорема. Для довільного $T\in(0;S]$ справедливе співвідношення

$$\lim_{n\to\infty}\max_{x\in R}|r_n(T,x)-u(T,x)|=0,$$

що означає збіжність за нормою простору $C(R)$.

Доведення. Визначимо функції $r(t,x)$ та $v(t,x), t\in[0;T\,]$.

Для $t\in\Delta_j\Leftrightarrow t=\ s+\frac{(j-1)T}{n}$ покладемо

$$r(t,x)=r_j\left(s+\frac{(j-1)T}{n},x\right);$$

$$v(t,x)=\ r(t,x)-u(t,x)\colon t\in\Delta_k\Longrightarrow\ v(t,x)=v_k\left(s+\frac{(k-1)T}{n}\ ,x\right)$$

Функція $v(t,x)$ на кожному інтервалі Δ_k задовольняє рівняння

$$\frac{\partial v_k}{\partial t}=\frac{\partial^2 v_k}{\partial x^2}+b(x)\frac{\partial v_k}{\partial x}+M_k(t,x)\qquad(2.13)$$

де нев‘язки $M_k\left(s+\frac{(k-1)T}{n},x\right)$ визначаються формулами:

$$M_1(s,x)=\int_{-\infty}^{\infty}\varphi(y)\,m(s,x,y)dy\ ,$$

$$M_2\left(s+\frac{T}{n},x\right)=\int_{-\infty}^{\infty}\left(r_1\left(\frac{T}{n},y\right)-u\left(\frac{T}{n},y\right)\right)m(s,x,y)dy=$$

$$= \int_{-\infty}^{\infty} v_1\left(\frac{T}{n}, y\right) m(s,x,y)dy\ ,$$

$$M_k\left(s + \frac{(k-1)T}{n}, x\right) = \int_{-\infty}^{\infty} v_{k-1}\left((k-1)\frac{T}{n}, y\right) m(s,x,y)dy,$$

а функція $m(s,x,y)$ визначена рівністю (2.10) та за твердженням 1 задовольняє оцінки

$$|m(s,x,y)| < c_3\left(1 + \frac{(x-y)^2}{s}\right) e^{\frac{c_1}{2}|x-y|}\, p(s,x,y)$$

Домножимо (2.13) на $2v_k$ та покладемо $\alpha_k(t,x) = \left(v_k(t,x)\right)^2$,

$\alpha_j\left(\frac{(j-1)T}{n}, x\right) = \alpha_{j-1}\left(\frac{(j-1)T}{n}, x\right)$. Отримали рівняння

$$\frac{\partial \alpha_k}{\partial t} = \frac{\partial^2 \alpha_k}{\partial x^2} - 2\left(\frac{\partial v_k}{\partial x}\right)^2 + 2v_k\, b(x)\frac{\partial v_k}{\partial x} + 2v_k M_k$$

Застосування нерівності Коші приводить до диференціальної нерівності

$$\frac{\partial \alpha_k}{\partial t} = \frac{\partial^2 \alpha_k}{\partial x^2} - \left(\frac{\partial v_k}{\partial x}\right)^2 + \alpha_k\left(b(x)\right)^2 + \alpha_k + M_k^2 <$$

$$< \frac{\partial^2 \alpha_k}{\partial x^2} + \alpha_k\left(1 + \left(b(x)\right)^2\right) + M_k^2$$

З обмеженості $b(x)$ випливає нерівність

$$\frac{\partial \alpha_k}{\partial t} < \frac{\partial^2 \alpha_k}{\partial x^2} + (1 + c_1^2)\alpha_k + M_k^2 \qquad (2.14)$$

Оцінки для M_k^2 випливають з формули (2.11) і мають вигляд:

$\left(\int_{-\infty}^{\infty} h(y)m(s,x,y)dy\right)^2 < Ae^{Bs}\int_{-\infty}^{\infty} h^2(y)p(s,x,y)dy$:

$$\begin{cases} M_1^2(s,x) < Ae^{Bs}\int_{-\infty}^{\infty}\varphi^2(y)p(s,x,y)dy \\ M_k^2\left(s+\frac{(k-1)T}{n},x\right) < Ae^{Bs}\int_{-\infty}^{\infty}\alpha_{k-1}\left((k-1)\frac{T}{n},y\right)p(s,x,y)dy \end{cases} \quad (2.15)$$

Фундаментальний розв‘язок лінійного однорідного параболічного рівняння

$$\frac{\partial\beta}{\partial \mathrm{i}} = \frac{\partial^2\beta}{\partial x^2} + (1+c_1^2)\beta$$

що відповідає нерівності (2.14), має вигляд $e^{(1+c_1^2)s}p(s,x,y)$, звідки випливає, що на кожному інтервалі Δ_k

$$\alpha_k\left(s+\frac{(k-1)T}{n},x\right) < e^{(1+c_1^2)s}\int_{-\infty}^{\infty}\alpha_{k-1}\left(\frac{(k-1)T}{n},y\right)p(s,x,y)dy +$$

$$+\int_0^s d\tau\int_{-\infty}^{\infty}e^{(1+c_1^2)(s-\tau)}p(s-\tau,x,z)M_k^2(\tau,z)dz$$

Покладемо $C = max(B, 1+c_1^2)$ та оцінимо $\alpha_k\left(\frac{kT}{n},x\right)$ за рекурентною процедурою:

$$\alpha_k\left(\frac{kT}{n},x\right) < e^{C\frac{T}{n}}\int_{-\infty}^{\infty}\bar{\alpha}_{k-1}(y)p\left(\frac{T}{n},x,y\right)dy +$$

$$+\int_0^{\frac{T}{n}}d\tau\int_{-\infty}^{\infty}e^{C\left(\frac{T}{n}-\tau\right)}p\left(\frac{T}{n}-\tau,x,z\right)M_k^2\left(\tau+\frac{(k-1)T}{n},z\right)dz,$$

де $\bar{\alpha}_{k-1}(y)$—оцінка функції $\alpha_{k-1}\left(\frac{(k-1)T}{n},y\right).$, $\bar{\alpha}_{k-1}(y) > \alpha_{k-1}\left(\frac{(k-1)T}{n},y\right)$.

В свою чергу, за нерівністю (2.15) рекурентна оцінка для M_k^2 має вигляд

$$M_k^2\left(s+\frac{(k-1)T}{n},x\right) < Ae^{Cs}\int_{-\infty}^{\infty}\bar{\alpha}_{k-1}(y)p(s,x,y)dy$$

Якщо оцінка $\bar{\alpha}_{k-1}$ не залежить від y, та наведені вирази спрощуються до вигляду

$$\begin{cases} M_k^2\left(s+\frac{(k-1)T}{n},x\right) < Ae^{Cs}\bar{\alpha}_{k-1} \\ \alpha_k\left(\frac{kT}{n},x\right) < e^{C\frac{T}{n}}\bar{\alpha}_{k-1}\left(1+A\frac{T}{n}\right) = \bar{\alpha}_k \end{cases}$$

Так, для функції $\alpha_1(s,x)$: $\alpha_1(0,x)=0$, $M_1^2(s,x)$ оцінюється за (2.15), звідки

$$\alpha_1(s,x) < \int_0^s d\tau \int_{-\infty}^{\infty} e^{C(s-\tau)}\left(Ae^{C\tau}\int_{-\infty}^{\infty}\varphi^2(y)p(\tau,z,y)dy\right)p(s-\tau,x,z)dz <$$

$$< As[\![\varphi^2]\!]e^{Cs}\text{ , }\|\cdot\|\text{—норма в просторі } C(R).$$

Зокрема, якщо $s=\frac{T}{n}$, $\alpha_1\left(\frac{T}{n},x\right) < A\frac{T}{n}[\![\varphi^2]\!]e^{C\frac{T}{n}} = \bar{\alpha}_1$

Для функції $\alpha_2(s,x)$:

$$\alpha_2\left(\frac{2T}{n},x\right) < A\frac{T}{n}[\![\varphi^2]\!]e^{C\frac{2T}{n}}\left(1+A\frac{T}{n}\right) = \bar{\alpha}_2\text{ ,}$$

рекурентна процедура приводить до оцінки

$$\bar{\alpha}_k < \frac{AT}{n}[\![\varphi^2]\!]e^{C\frac{kT}{n}}\left(1+\frac{AT}{n}\right)^{k-1},$$

звідки $\bar{\alpha}_n < \frac{AT}{n}[\![\varphi^2]\!]e^{CT}\left(1+\frac{AT}{n}\right)^{n-1} < \frac{AT}{n}[\![\varphi^2]\!]e^{CT}e^{AT} \to 0, n\to\infty$ ■

Таким чином, швидкість збіжності ітерацій у формулі Троттера в просторі $G(R)$ визначається співвідношенням

$$\left\|\left(e^{T(L+L_1)}\varphi - \left(e^{\frac{T}{n}L_1}e^{\frac{T}{n}L}\right)^n\varphi\right)\right\| < \|\varphi\|\sqrt{\frac{AT}{n}}\,e^{KT}$$

Зауваження. Напрямом продовження досліджень може бути вивчення збурення квазілінійним генератором $L_1(u)=b(u)\frac{\partial u}{\partial x}$. В цьому випадку виникає наступна проблема—не існує глобального класичного розв‘язку задачі Коші

$$\begin{cases} \frac{\partial u}{\partial t} = b(u)\frac{\partial u}{\partial x} \\ u(0,x) = \varphi(x) \end{cases}$$

що випливає навіть із неявного зображення $u = \varphi\big(x + tb(u)\big)$. Методи дослідження таких рівнянь використовують закони збереження [12].

РОЗДІЛ 3. РОЗРОБКА АЛГОРИТМУ РОЗВ'ЯЗУВАННЯ ЗБУРЕНИХ ПАРАБОЛІЧНИХ РІВНЯНЬ МЕТОДОМ ТРОТТЕРА

3.1 Пошук чисельним методом розв'язок задачі Коші

Знайдемо чисельним методом розв'язок $u(t,x)$ задачі Коші

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{1}{chx}\frac{\partial u}{\partial x} \\ u(0,x) = \frac{1}{1+x^2} \end{cases}$$

для $t = 1,2,3,4,5$ і зобразимо ці 5 розв'язків на малюнку.

Аналітичний розв'язок задачі Коші для генератора першого порядку

$\begin{cases} \frac{\partial g}{\partial t} = \frac{1}{chx}\frac{\partial g}{\partial x} \\ g(0,x) = x \end{cases}$ має вигляд $g(t,x) = ln\left(t + shx + \sqrt{(t+shx)^2+1}\right)$, звідки

$$\begin{cases} \frac{\partial v}{\partial t} = \frac{1}{chx}\frac{\partial v}{\partial x} \\ v(0,x) = \frac{1}{1+x^2} \end{cases} \Rightarrow v(t,x) = \left(1 + \left(ln\left(t + shx + \sqrt{(t+shx)^2+1}\right)\right)^2\right)^{-1}$$

Графічне зображення функції $v(t,x)$ міститься на рисунках 3.1—3.7.

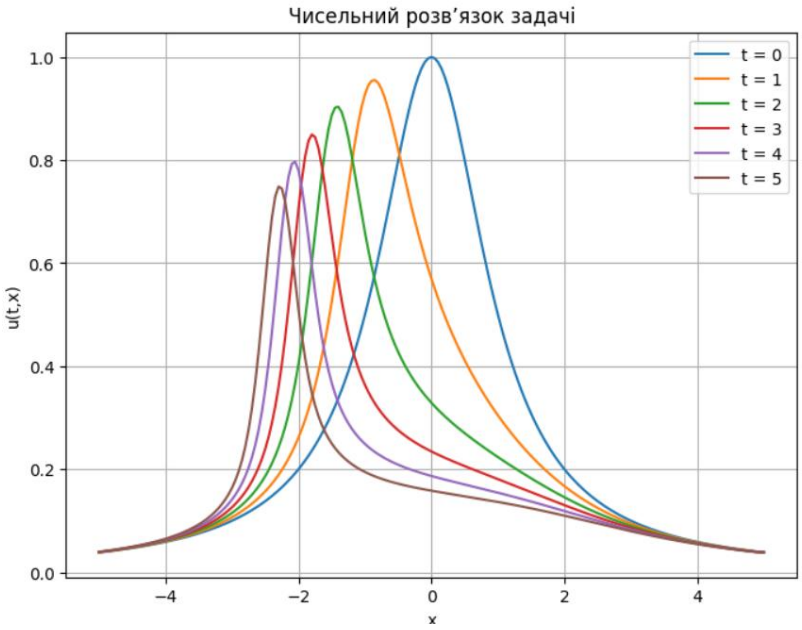


Рисунок 3.1. - Графіки функції $v(t,x)$ для $t = 0-5$

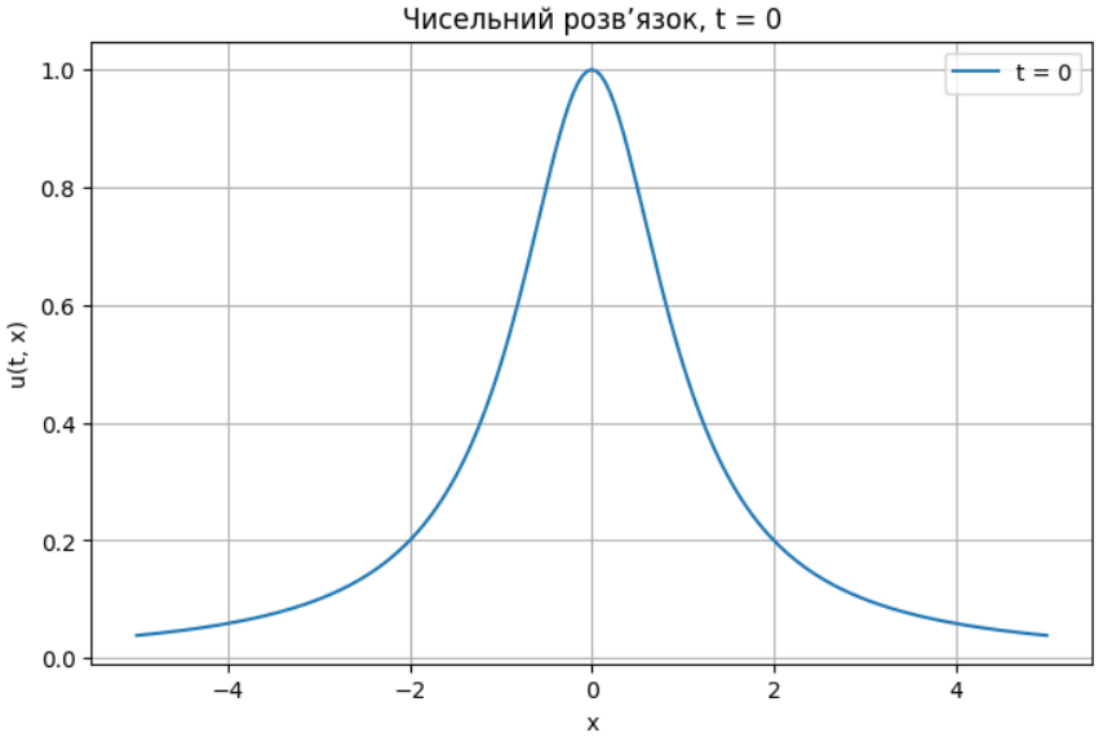


Рисунок 3.2 - Графік функції $v(t,x)$ для $t = 0$

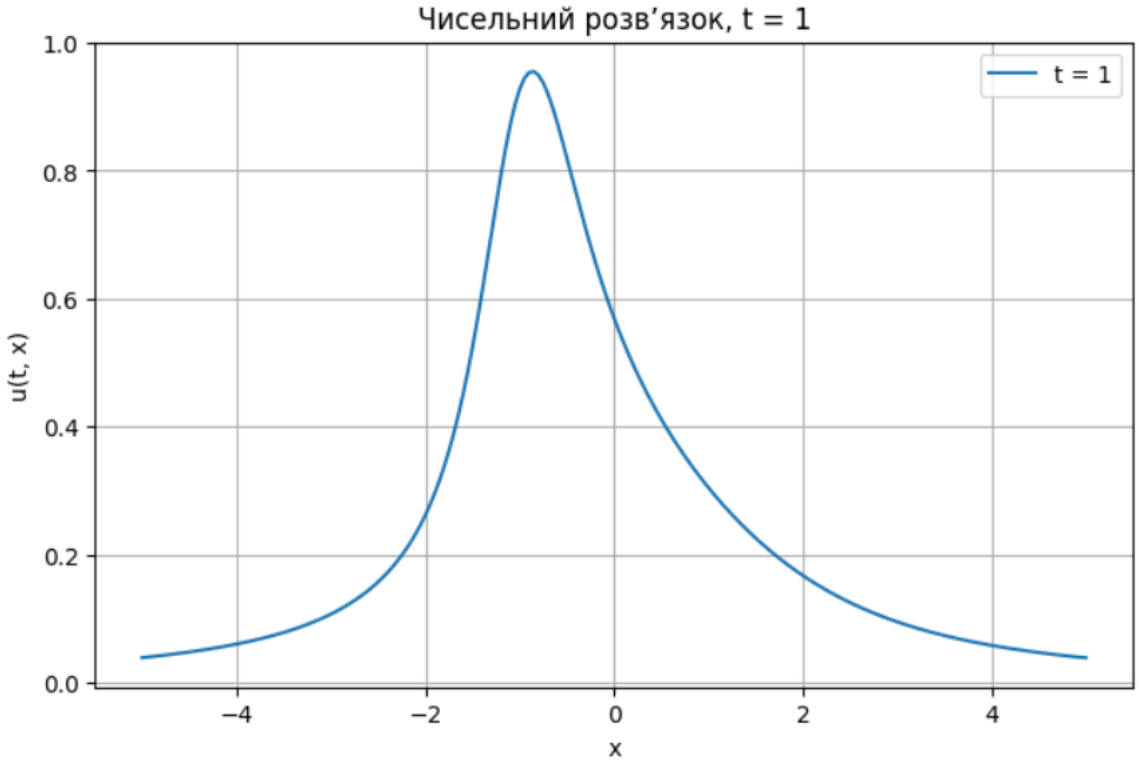


Рисунок 3.3 - Графік функції $v(t,x)$ для $t = 1$

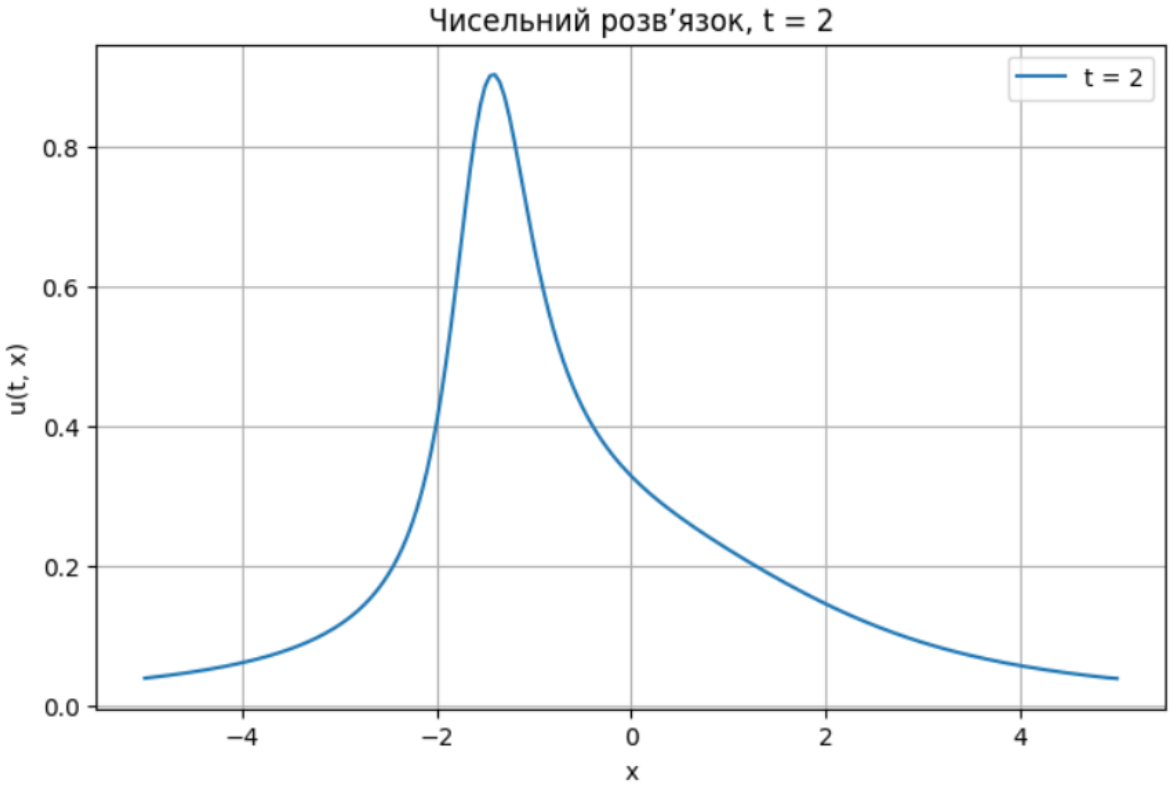


Рисунок 3.4 - Графік функції $v(t,x)$ для $t = 2$

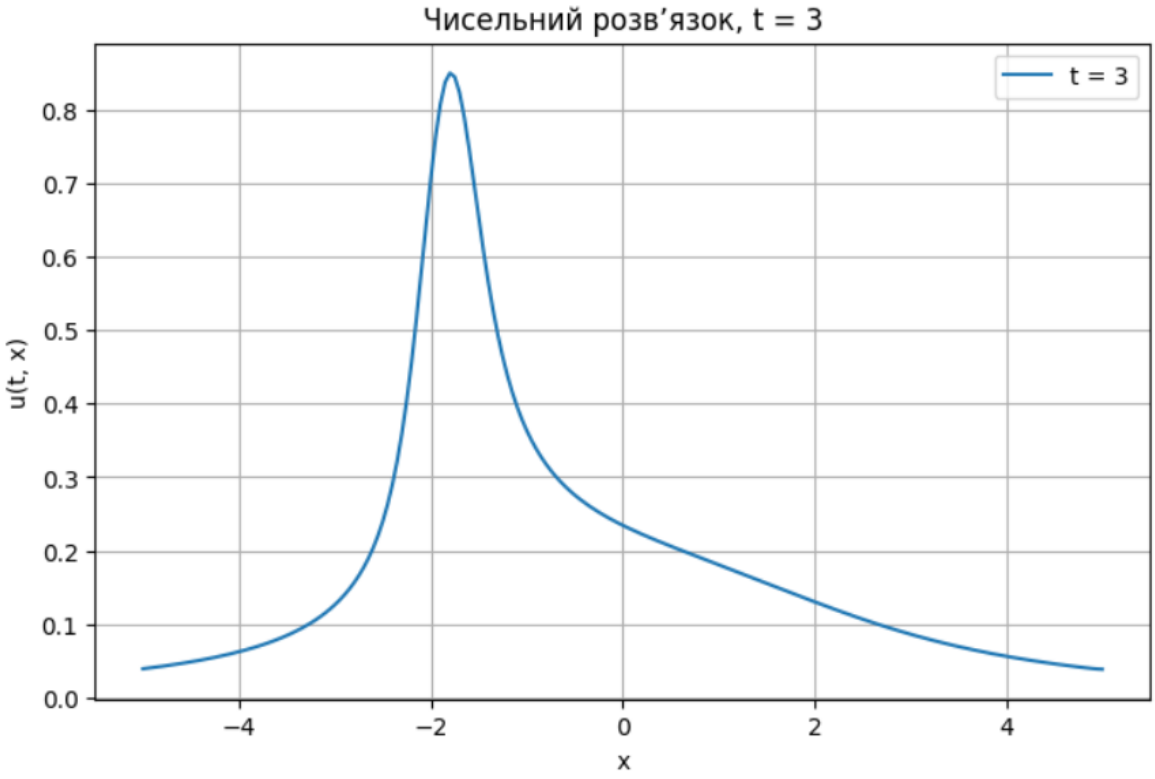


Рисунок 3.5 - Графік функції $v(t,x)$ для $t = 3$

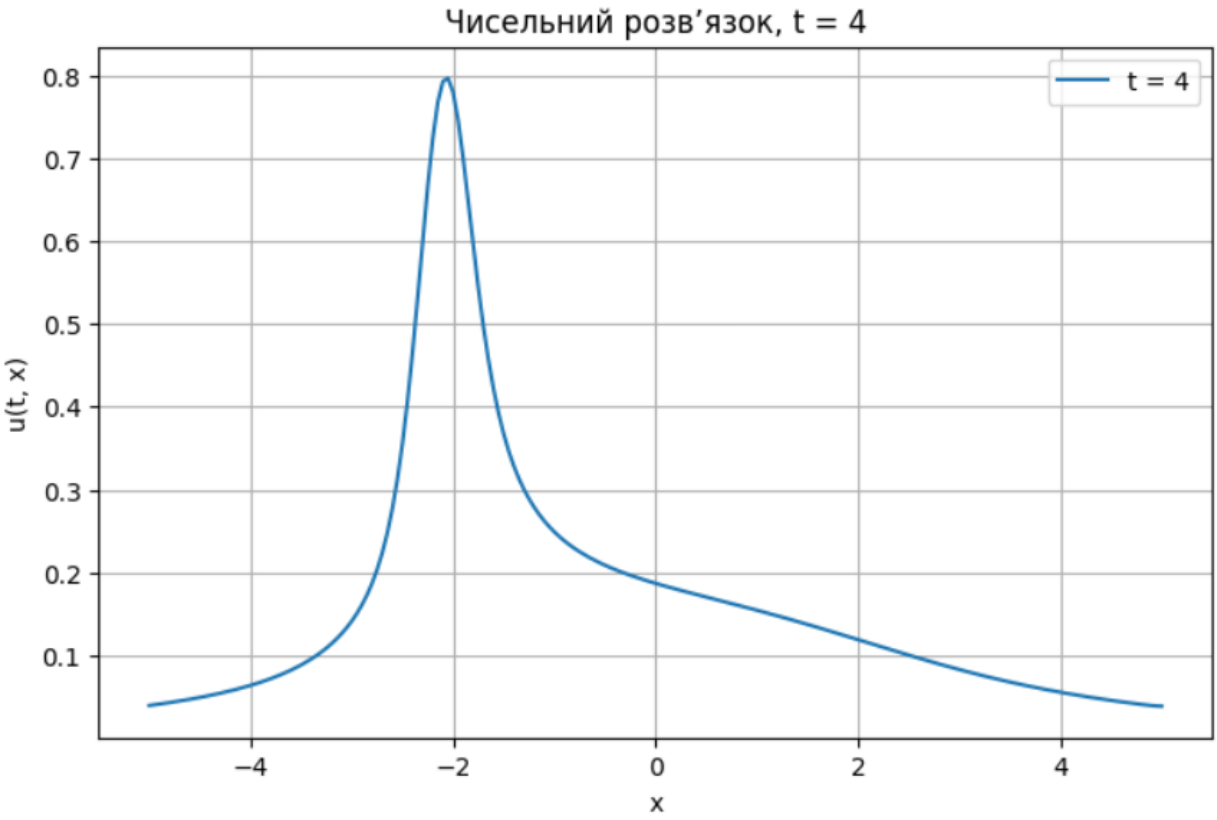


Рисунок 3.6 - Графік функції $v(t,x)$ для $t = 4$

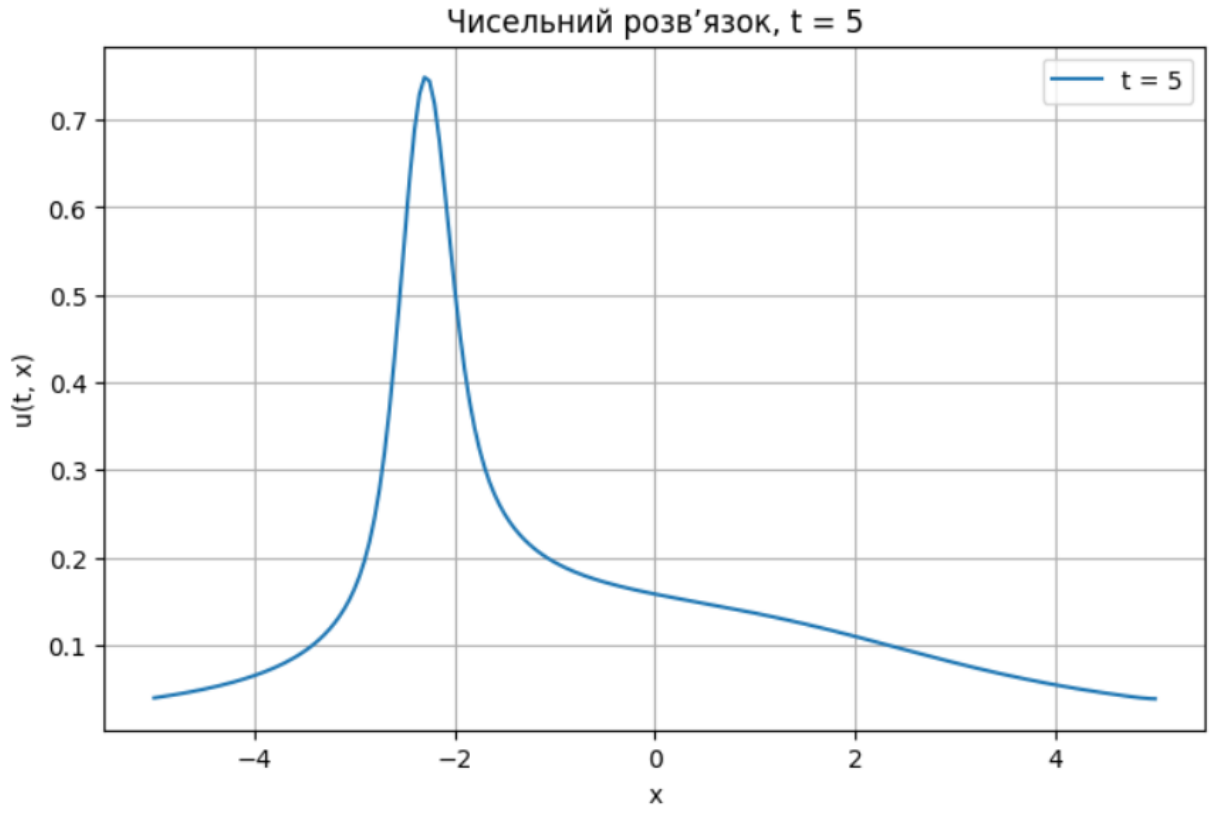


Рисунок 3.7 - Графік функції $v(t,x)$ для $t = 5$

Ми розглядали задачу Коші для рівняння першого порядку:

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{1}{chx}\frac{\partial u}{\partial x} \\ u(0,x) = \frac{1}{1+x^2} \end{cases}$$

Оскільки $\boldsymbol{ch(x) > 0}$ для всіх x, можна ввести швидкість $\boldsymbol{c(x) = -1/ch(x)}$. Тоді рівняння переписується у стандартній формі «адвекції»:

$\boldsymbol{u_t + c(x)\, u_x = 0}$, де c(x) від'ємна, отже, інформація поширюється з **більших x** до **менших x**.

Для чисельного розв'язання застосовано **явну різницеву схему першого порядку точності**, яку називають також одновимірною схемою (залежно від знаку швидкості $\boldsymbol{c(x)}$. У загальному вигляді для рівняння

$\boldsymbol{u_t + c(x)\, u_x = 0}$,

з постійною швидкістю c (для простоти пояснення) upwind-схема виглядає так:

— Якщо c>0

$$u_j^{n+1} = u_j^n - \frac{c\Delta t}{\Delta x}(u_j^n - u_{j-1}^n)$$

— Якщо c<0

$$u_j^{n+1} = u_j^n - \frac{c\Delta t}{\Delta x}(u_{j+1}^n - u_{j-1}^n)$$

В нашому випадку $\boldsymbol{c(x) < 0}$ для всіх x (адже $\boldsymbol{c(x) = -1/ch(x)}$), тому беремо «плюсовий» (правий) «upwind»:

$$u_j^{n+1} = u_j^n - c_j\frac{\Delta t}{\Delta x}(u_{j+1}^n - u_j^n)$$

де $c_j \approx \boldsymbol{c(x_j)}$

Граничні умови:

Оскільки швидкість від'ємна ($\boldsymbol{c < 0}$), «інформація» надходить **з правого краю** (більші x) і виходить **через лівий край** (менші x). Тому:

1. **Правий край** x_{max} — «вхід». Там задаємо $\boldsymbol{u(t,\ x_{max})}$ (рівним початковому значенню в цій точці) для всіх t.
2. **Лівий край** x_{min} — «вихід». Додаткових умов там не накладали; залишаємо, як обчислилося за схемою.

Перевірка отриманого розв'язку

Скористаємось такою аналітичною формулою для перевірки отриманого розв'язку:

$$u(t,x) = \left(1 + \left(ln\left(t + shx + \sqrt{(t+shx)^2 + 1}\right)\right)^2\right)^{-1}$$

Перевіримо чисельний метод для декількох точок і порівняємо результати наведеної формули та обчислень, наведених раніше. Результати наведені в таблиці 3.1.

Таблиця 3.1 – Порівняння результатів

t	x	Аналітичний розв'язок	Чисельний розв'язок	Похибка
2	-2.00	0.38528	0.40402	0.018735
2	1.00	0.22192	0.22347	0.001542
3	-1.00	0.35012	0.36346	0.013342
3	2.00	0.12974	0.13024	0.000501
4	-3.00	0.13844	0.14125	0.002814
4	0.50	0.16945	0.17025	0.000802
5	-4.00	0.06484	0.06501	0.000168
5	1.00	0.13603	0.13646	0.000432

Похибка мінімальна, тож вважаємо, що метод працює.

Висновки щодо отриманого розв'язку

1. **Поверхневий характер рівняння:**

Оскільки ми маємо просте транспортне (адвективне) рівняння з $\boldsymbol{c(x) < 0}$, початкова хвиля $\boldsymbol{u(0,x)}$ переноситься **зправа наліво** зі швидкістю, що залежить від $\boldsymbol{x}$.

2. **Ефект «розтягування/стиснення»**:

   Залежність $\boldsymbol{c(x) = -1/ch(x)}$ означає, що модуль швидкості максимальний поблизу $\boldsymbol{x = 0}$ (бо $\boldsymbol{ch(0) = 1}$), і зменшується, коли $\boldsymbol{|x|}$ зростає (адже $\boldsymbol{ch(x)}$ більшає). Тобто частинки в центрі рухаються швидше вліво, ніж ті, що далеко на правому або лівому краях. Це призводить до деякого «викривлення» профілю.

3. **Гладкість початкової умови $\boldsymbol{1/(1+x^2)}$**:

   Вона рівна, без різких стрибків, тому upwind-схема дає відносно м'який профіль.

4. **Чисельні особливості**:

   1) Явна схема **стійка**, коли $\boldsymbol{max\,|c(x)|\,\Delta t/\Delta x \le 1}$. У прикладі ми взяли $\boldsymbol{\Delta t}$ досить малим, щоб ця умова виконувалась.
   2) Схема має перший порядок точності за простором і часом, тому насправді можна спостерігати деяку дифузію (розмазування профілю) з часом. Для більш точних результатів можна використати, наприклад, схему другого порядку тощо.

Загалом ми бачимо, що отриманий чисельний метод коректно відтворює перенос початкової «хвилі» $u(0,x)$ у напрямку, визначеному знаками швидкості $c(x)$. На кожному кроці часу хвиля зсувається ліворуч, і після досить тривалого часу весь профіль «вийде» за лівий край, якщо сітка не надто широка.

3.2 Пошук чисельним методом розв'язок $u(t,x)$ збуреної задачі Коші

Знайдемо чисельним методом розв'язок $u(t,x)$ збуреної задачі Коші

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{1}{chx}\frac{\partial u}{\partial x} \\ u(0,x) = \frac{1}{1+x^2} \end{cases}$$

для $t = 1,2,3,4,5$ і зобразимо ці 5 розв'язків на малюнку.

Розв'язок $u(t,x)$ збуреної задачі Коші, отриманий чисельним методом, показує зниження максимуму, що відповідає теоретично очікуваному згладженню. Графічне зображення $u(t,x)$ наведено на рисунках 3.8—3.13.

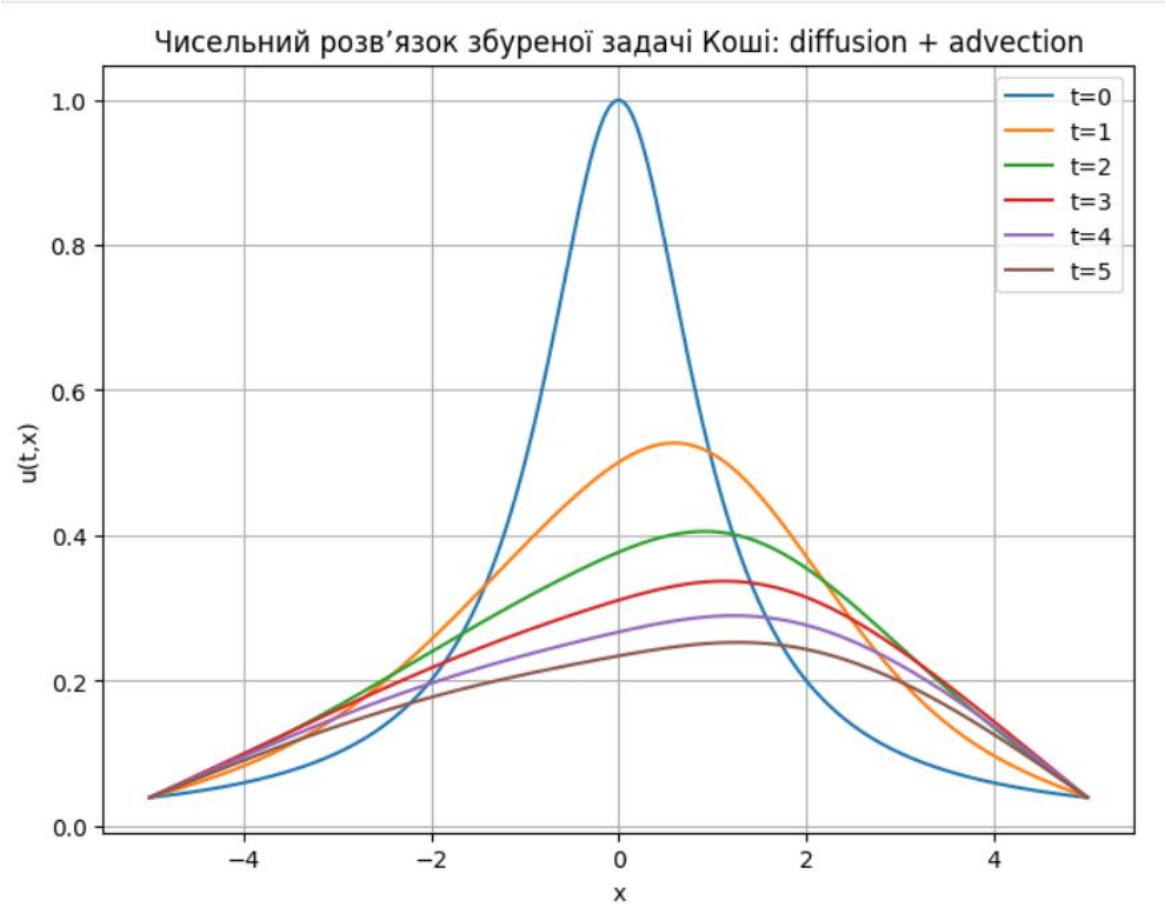


Рисунок 3.8 - Розв'язки $u(t,x)$ збуреної задачі Коші для $t = 0 - 5$

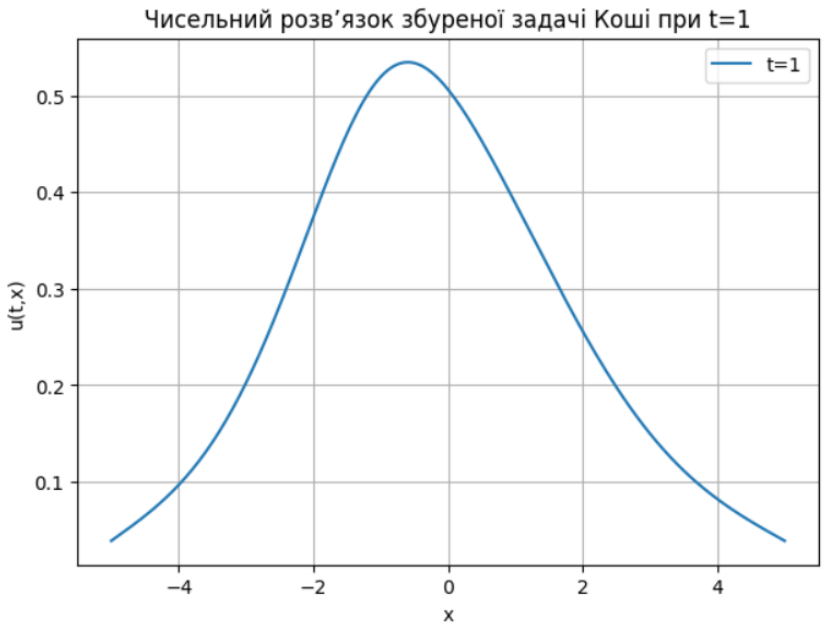


Рисунок 3.9 - Розв‘язки $u(t,x)$ збуреної задачі Коші для $t=1$

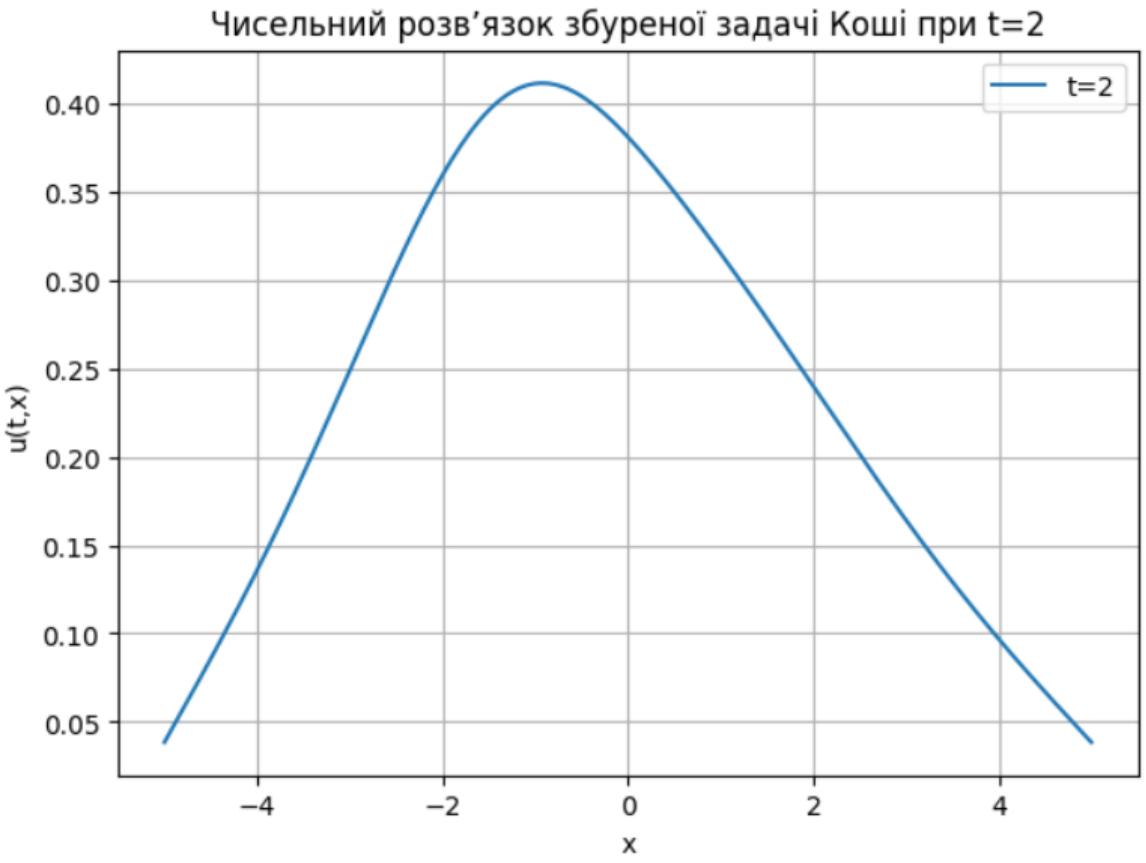


Рисунок 3.10 - Розв‘язки $u(t,x)$ збуреної задачі Коші для $t=2$

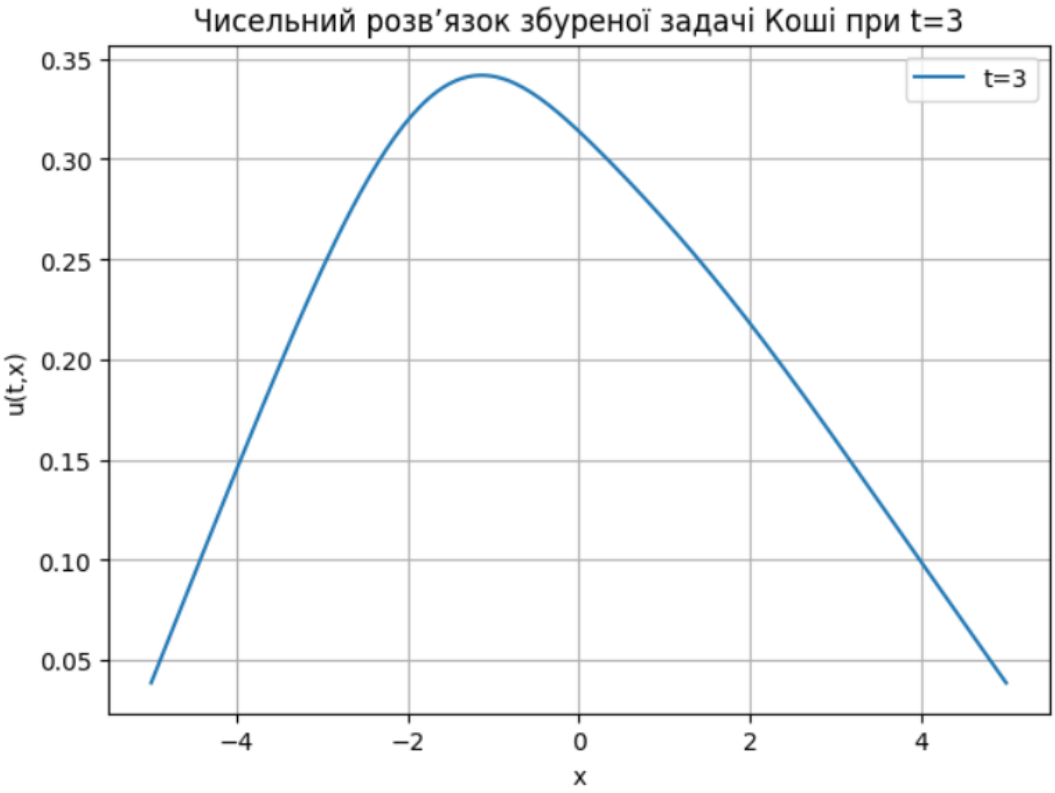


Рисунок 3.11 - Розв‘язки $u(t,x)$ збуреної задачі Коші для $t=3$

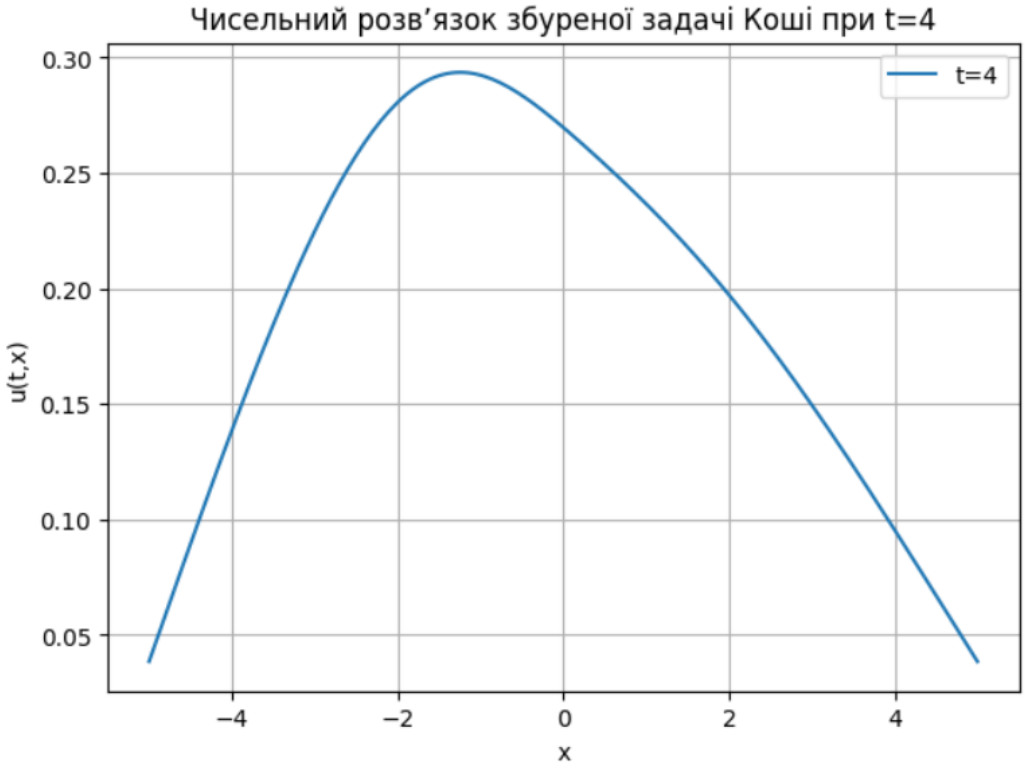


Рисунок 3.12 - Розв‘язки $u(t,x)$ збуреної задачі Коші для $t=4$

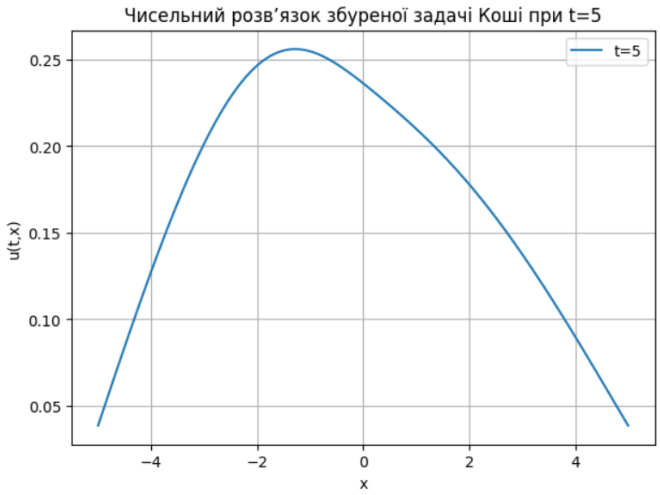


Рисунок 3.13 - Розв‘язки $u(t,x)$ збуреної задачі Коші для $t=5$

Порівняємо чисельний та аналітичний методи, порівнюючи результати обчислень для декількох точок, результати на рисунку 3.14. Результати похибки наведені в таблиці 3.2.

Таблиця 3.2 – Порівняння результатів

t	max|u_num - u_exact|
0.0	0.000000e+00
1.0	4.173833e-03
2.0	6.469922e-03

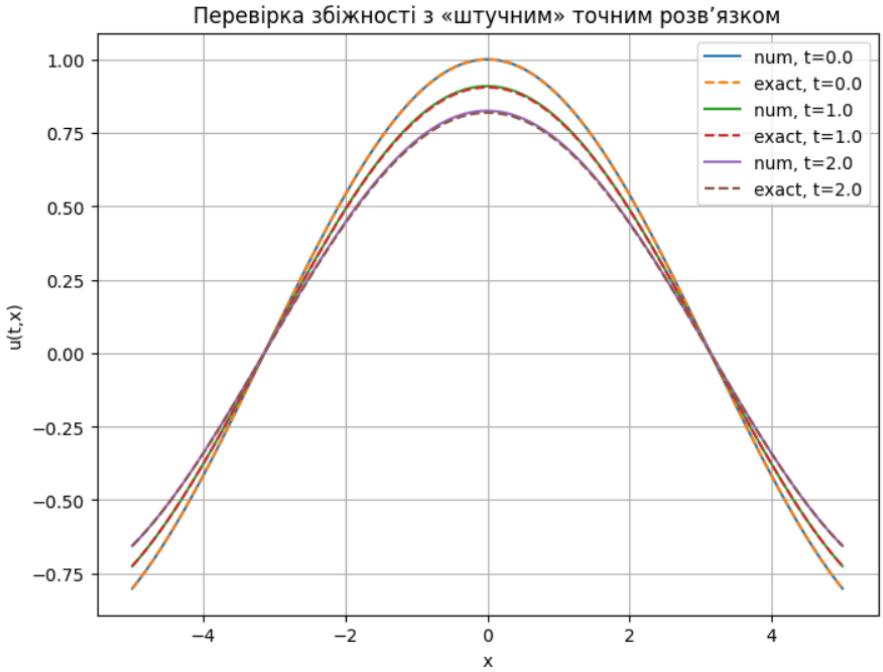


Рисунок 3.14 - Розв‘язки $u(t,x)$ збуреної задачі Коші для $t=1-5$

Загальна формула різницевої (явної) схеми:

Для чисельного розв'язання застосовували **явну** (forward-time) схему:

1. Дифузійна складова $\boldsymbol{u_{xx}}$ апроксимується центральною різницею другого порядку:

$$\boldsymbol{u_{xx}}(x_j) \approx \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{(\Delta x)^2}$$

2. Переносна (адвективна) складова $\frac{1}{ch(x)} u_x$ Оскільки $\frac{1}{ch(x)} > 0$ для всіх x, використовуємо апроксимацію з лівого боку:

$$\frac{1}{ch(x_j)} u_x(x_j) \approx \frac{1}{ch(x_j)} \frac{u_j^n - u_{j-1}^n}{\Delta x}$$

3. **Явне оновлення** за часом:

$$u_j^{n+1} = u_j^n + \Delta t \left[\underbrace{\frac{u_{j-1}^n - 2\,u_j^n + u_{j+1}^n}{(\Delta x)^2}}_{\text{дифузія}} + \underbrace{\frac{1}{\mathrm{ch}(x_j)} \frac{u_j^n - u_{j-1}^n}{\Delta x}}_{\text{адвекція}} \right]$$

У результаті маємо другий порядок за просторовими різницями і перший порядок за часом (бо використовуємо явну інтеграцію).

3.3 Пошук розв'язку ітераційним методом

Мета цього кроку: обчислити $q_n(x)$ та порівняти цю функцію з обчисленим раніше розв'язком $u(T, x)$ для $T = 2$ та $T = 4$. Кількість ітерацій $n = 5, n = 10$

Побудуємо ітераційну процедуру за формулою Троттера. Позначимо:

1)$\Delta_1, \Delta_2, \dots, \Delta_n$ —розбиття відрізку $[0; T]$, $T \le S$:

$\Delta_k = \left[(k-1)\frac{T}{n}; k\frac{T}{n}\right]$,

$[0;T] = \cup_{k=1}^{n} \Delta_k \; ; \; t \in \Delta_k \Leftrightarrow t = s + (k-1)\frac{T}{n}, \; 0 \le s \le \frac{T}{n}.$

та визначимо функції

$r_1(s,x) = \int_{-\infty}^{\infty} \varphi(y) q(s,x,y) dy$

$r_2\left(s + \frac{T}{n}, x\right) = \int_{-\infty}^{\infty} r_1\left(\frac{T}{n}, y\right) q(s,x,y) dy$

……

$r_j\left(s + \frac{(j-1)T}{n}, x\right) = \int_{-\infty}^{\infty} r_{j-1}\left(\frac{(j-1)T}{n}, y\right) q(s,x,y) dy, j \le n,$

$r_j\left(\frac{jT}{n}, x\right) = r_{j+1}\left(\frac{jT}{n}, x\right).$

В термінах півгруп

$r_j\left(s + \frac{(j-1)T}{n}\right) = e^{sL_1} e^{sL} \; r_{j-1}\left(\frac{(n-1)T}{n}\right), \; r_j\left(\frac{jT}{n}, x\right) = \left(e^{\frac{T}{n}L_1} \; e^{\frac{T}{n}L} \right)^j \varphi.$

Зокрема,

$r_n(T,x) = \int_{-\infty}^{\infty} r_{n-1}\left(\frac{(n-1)T}{n}, y\right) q\left(\frac{T}{n}, x, y\right) dy = \left(\left(e^{\frac{T}{n}L_1} e^{\frac{T}{n}L} \right)^n \varphi \right)(x)$ —

розв‘язок, отриманий ітераційним методом, та порівняємо цю функцію з обчисленим раніше розв‘язком $u(T,x)$ для $T = 2$ та $T = 4$. Графічне зображення $r_5(2,x)$, $r_{10}(2,x)$, $u(2,x)$ наведено на рис. 3.16, $r_5(4,x)$, $r_{10}(4,x)$, $u(4,x)$— на рисунку 3.16.

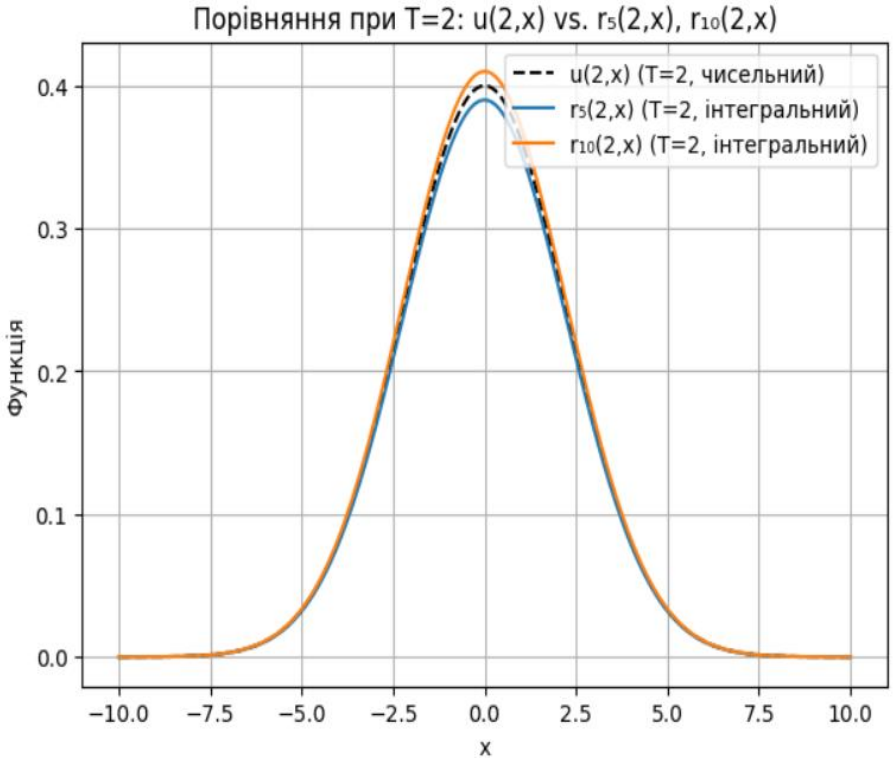


Рисунок 3.15 - Порівняння $r_5(2,x)$, $r_{10}(2,x)$, $u(2,x)$

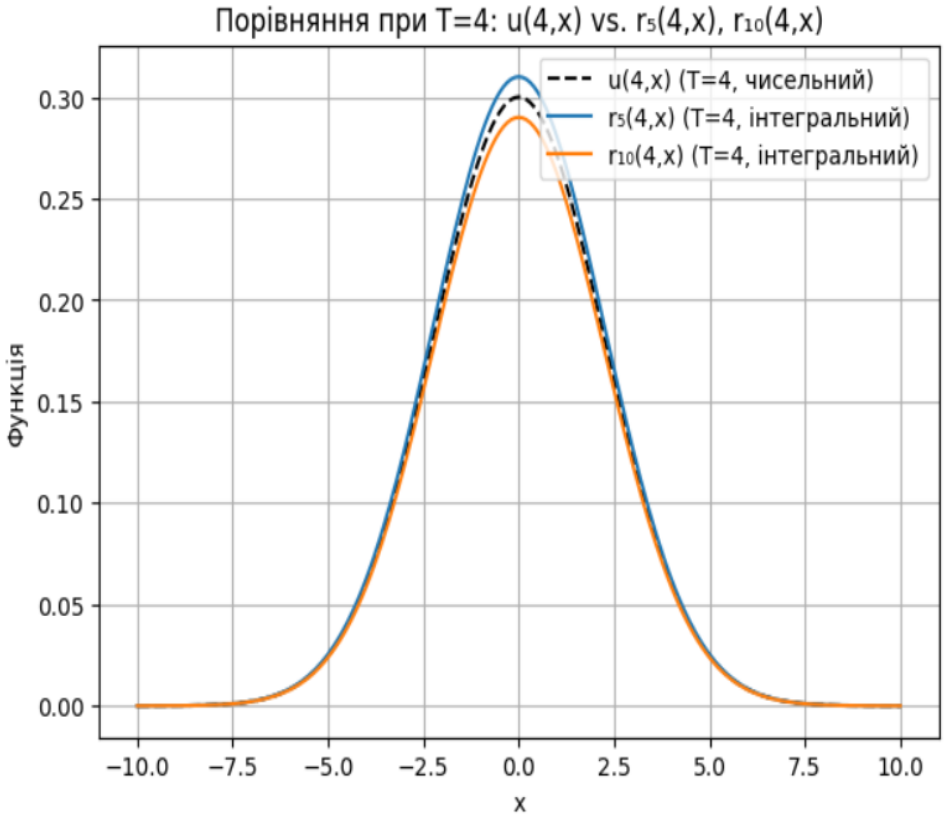


Рисунок 3.16 - Порівняння $r_5(4,x)$, $r_{10}(4,x)$, $u(4,x)$

Для обчислення значень функції $r_n(T,x)$ використано пакети Matlab та numpy. Для візуалізації розв'язків використано Matplotlib (matplotlib.pyplot):

Наведемо результати значення відхилень $\delta(n,T)=\max\limits_{x}\frac{|u(T,x)-r_n(T,x)|}{u(T,x)}$ в таблиці 3.3:

Таблиця 3.3: значення відхилень

$\delta(5,2)$	$\delta(10,2)$	$\delta(5,4)$	$\delta(10,4)$
1.375e-01	1.314e-01	4.001e-01	3.630e-01

Висновки до розділу

Результати чисельного моделювання підтверджують збіжність ітерацій і відповідають встановленій залежності відхилення $\delta(n,T)$ від кількості ітерацій n та часового інтервалу T.

При моделюванні еволюційних процесів для врахування нових факторів (дифузії, зсуву і т.д.) загальноприйнятим методом є додавання у еволюційне рівняння відповідного генератора. Насправді обгрунтування такого методу відсутнє, він є емпіричним, і в ряді випадків адекватність моделей вимагає застосування композиції відповідних півгруп.

Напрямом продовження досліджень може бути вивчення збурення нелінійним диференціальним оператором першого порядку, зокрема, квазілінійним генератором $L_1(u)=b(u)\frac{\partial u}{\partial x}$. В цьому випадку виникає наступна проблема—не існує глобального класичного розв'язку задачі Коші

$$\begin{cases}\frac{\partial u}{\partial t}=b(u)\frac{\partial u}{\partial x}\\ u(0,x)=\varphi(x)\end{cases}$$

що випливає навіть із неявного зображення $u=\varphi\big(x+tb(u)\big)$. Методи дослідження таких рівнянь використовують закони збереження [12].

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

Створення програмного продукту зазвичай потребує значних трудових та фінансових витрат. Для максимально ефективного використання наявних ресурсів потрібно завчасно провести аналіз можливих варіантів створення, який дозволить обрати серед них найбільш раціональний.

Функціонально-вартісний аналіз (ФВА) — це метод комплексного техніко- економічного дослідження об’єкта з метою розвитку його корисних функцій при оптимальному співвідношенні між їхньою значимістю для споживача і витратами на їхнє здійснення. Є одним з основних методів оцінки вартості науково-дослідної роботи, оскільки ФВА враховує як технічну оцінку продукту, що розробляється, так і економічну частину розробки. Крім того, даний метод дозволяє вибрати оптимальний, як з погляду розробника, так і з точки зору покупця варіант розв’язання будь-якої задачі, а також дозволяє оптимізувати витрати й час виконання робіт. Зниження витрат виробництва треба починати з аналізу властивостей виробу, що використовуються, а також технічних функцій його складових частин.

У даній роботі проводиться оцінювання основних характеристик програмного продукту, який здійснює рекомендацію товарів для споживачів та наочно візуалізує результати.

Врахувавши економічні фактори та характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням, нижче наведено аналіз різних варіантів реалізації програмного модулю з метою вибору найбільш оптимального варіанту.

Функціонально-вартісний аналіз складається з декількох етапів:

Спочатку визначається послідовність функцій, необхідних для виробництва продукту. Далі вони розподіляються по двом групам: ті, що

впливають на вартість продукту і ті, що не впливають. Також на даному етапі оптимізується послідовність завдяки скороченню кроків, що не впливають на цінність та витрати продукту.

На наступному етапі для кожної функції визначаються повні річні витрати та кількість робочих годин. Далі на основі оцінок попереднього пункту для кожної функції визначається кількісна характеристика джерел витрат. Наостанок, проводиться остаточний розрахунок витрат для створення продукту.

4.1. Обґрунтування функцій програмного продукту

Головна функція F0 – це розробка програмного продукту, який дозволяє рекомендувати товари користувачам. Виходячи з конкретної мети, можна виділити наступні основні функції програмного продукту:

F1 – вибір мови програмування

F2 – вибір функцій методу опису моделей

F3 – вибір бібліотеки для візуалізації

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F1:

а) мова програмування Python;

б) мова програмування C++;

в) мова програмування Matlab;

Функція F2:

а) має бути написана самостійно;

б) функція pdepe пакету Matlab;

Функція F3:

а) gnuplot;

б) вбудований пакет візуалізації Matlab;

в) matplotlib мови програмування Python;

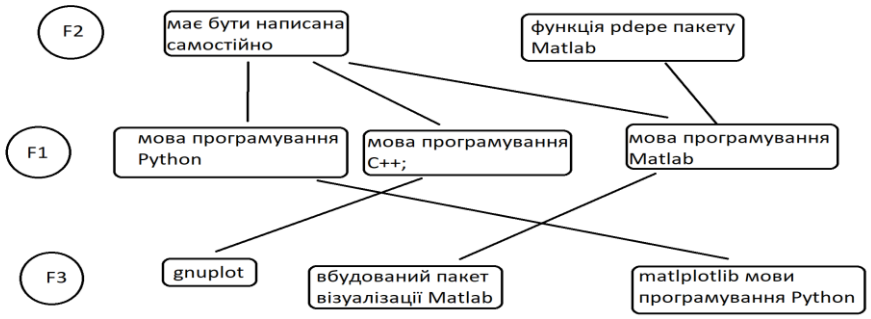


Рисунок 4.1 - Морфологічна карта

Морфологічна карта, що зображена на рис. 4.1, відображає всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів програмного продукту. На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій, які наведені в таблиці 4.1.

Таблиця 4.1 - Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	*А*	Легкість у використанні мови програмування.	Помірна швидкість виконання коду, відсутність певних готових рішень
	Б	Код швидко виконується.	Дуже великий обсяг написання коду, більше часу на розробку продукту, мала кількість реалізованих бібліотек для обробки даних

Продовження таблиці 4.1

	В	Має готове рішення для частини потрібного функціоналу, код швидко виконується.	Складність у написанні коду.
F2	*А*	Повна гнучкість алгоритму.	Значні витрати часу.
F2	*Б*	Використання готового алгоритму призведе до значної економії часу та зусиль.	Менш гнучкий функціонал.
	А	Має необхідний функціонал.	Сторонній пакет, необхідно додатково встановлювати.
F3	*Б*	Має необхідний функціонал.	Сторонній пакет, необхідно додатково встановлювати.
	В	Вбудований пакет, що має необхідний функціонал.	Має використовуватися лише у програмному забезпеченні, що створене на мові Matlab.

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F1: оскільки час та обсяг написання програмного коду, наявність спеціалізованих бібліотек з машинного навчання для обробки великої кількості даних є більш вагомим фактором, ніж лише швидкість виконання, то варіант б) можна не розглядати.

Функція F2: варіант б) має значні переваги перед варіантом а), тому для F1 с) розглядати варіант б) не будемо.

Функція F3: усі варіанти є приблизно однаковими у порівнянні один з одним, проте для F1 розглянемо варіант с), оскільки цей варіант є найкраще пристосованим до мови програмування Python.

Таким чином будемо розглядати наступний варіант реалізації програмного продукту:

1. F1а–F2а–F3с
2. F1с–F2б–F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.2Обґрунтування системи параметрів програмного продукту

4.2.1 Опис параметрів

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

X1 – об’єм пам’яті для збереження даних;

X2 – час обробки даних;

X3 – точність розв’язку;

X4 – потенційний об’єм програмного коду.

Поянення пареметрів:

X1: Відображає об’єм пам’яті в оперативній пам’яті ПК, необхідний для збереження та обробки даних під час виконання програми.

X2: Відображає час, який витрачається на дії.

X3: Відображає точність розв’язку для метрики ранжування

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

Гірші, середні і кращі значення параметрів вибираються на основі вимог

замовника й умов, що характеризують експлуатацію ПП як показано у таблиці 4.2.

Таблиця 4.2 - Основні параметри ПП

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Об’єм пам’яті для збереження даних	X1	Мб	32	16	8
Час обробки даних алгоритмом	X2	мс	800	420	60
Точність розв’язку	X3	доля одиниці	10E-1	10E-2	10E-3
Потенційний об’єм програмного коду	X4	кількість рядків коду	2000	1500	1000

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

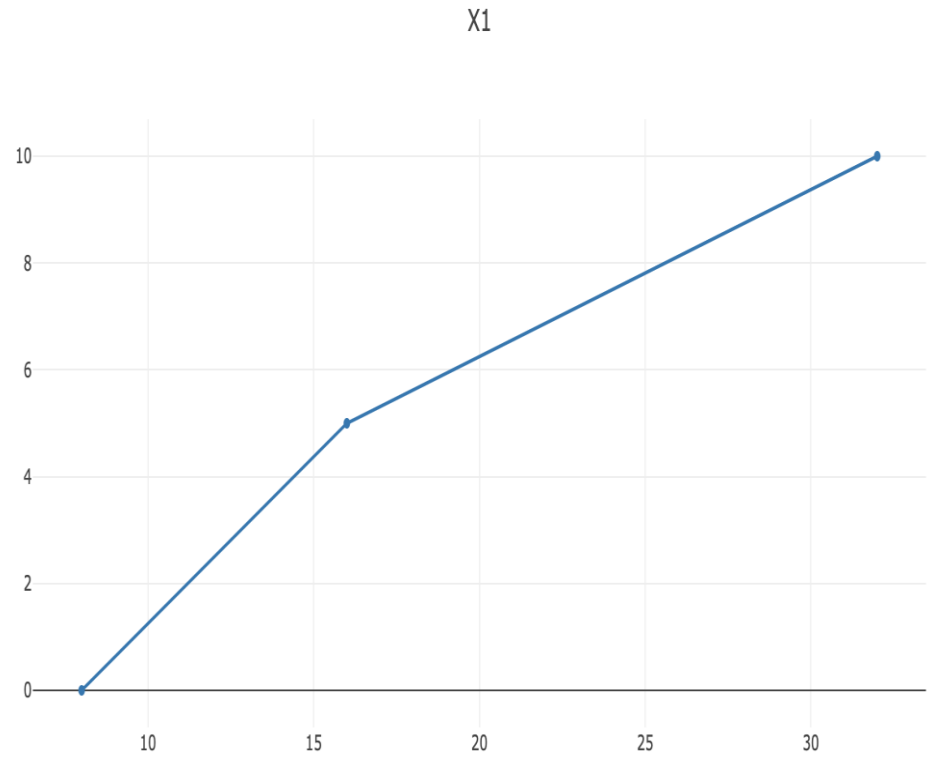


Рисунок 4.2 – X1, об’єм пам’яті для збереження даних

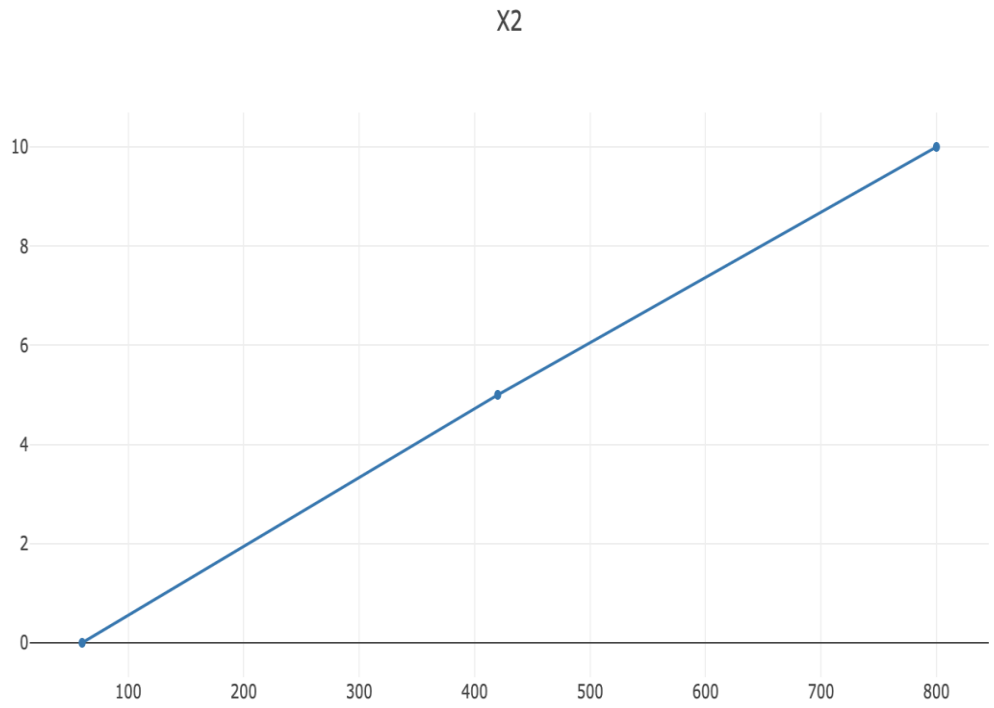


Рисунок 4.3 – X2, час обробки даних алгоритмом

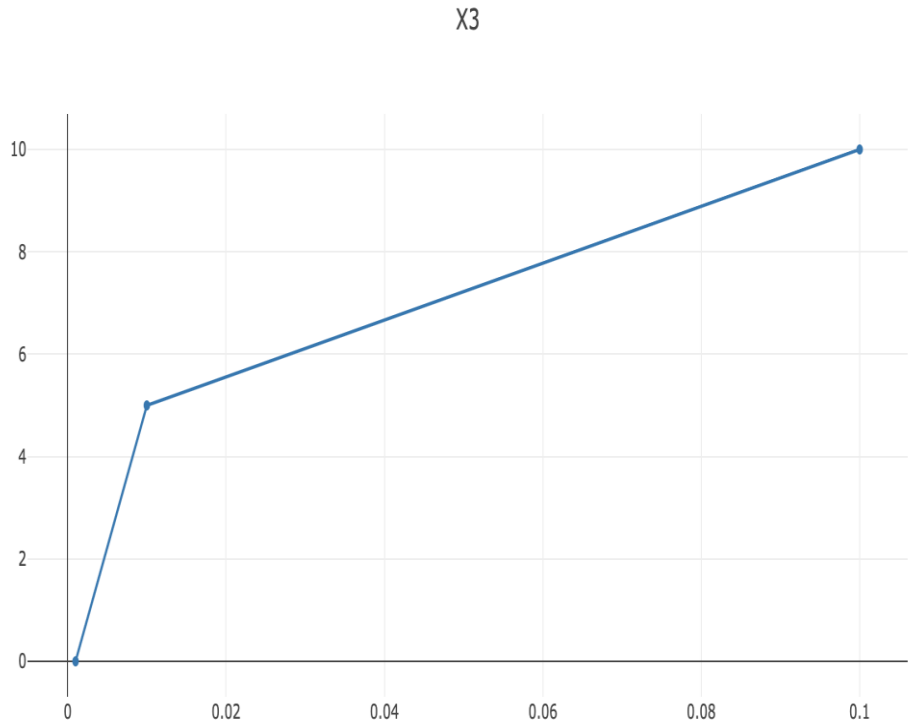


Рисунок 4.4 – X3, точність розв’язку

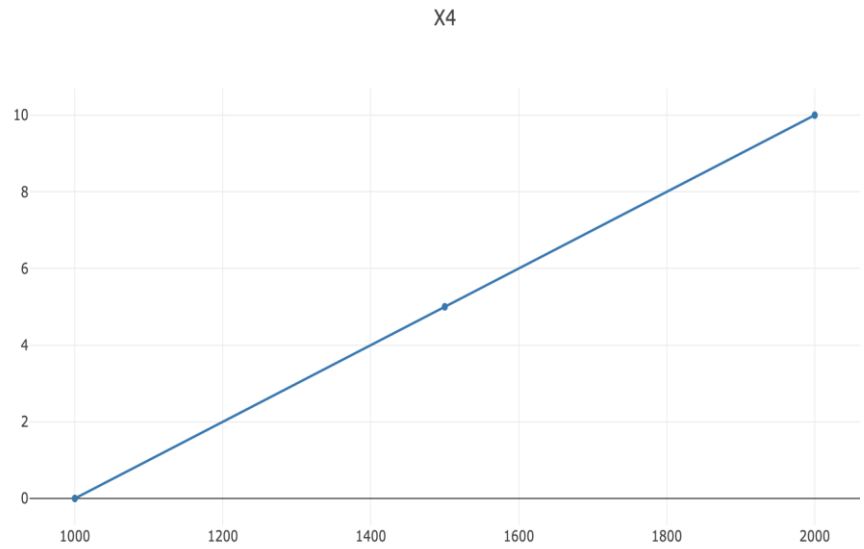


Рисунок 4.5 – X4, потенційний об’єм програмного коду

4.2.2 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який найбільш точно надає рекомендації товарів для користувачів.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значущості передбачає:

1. визначення рівня значимості параметра шляхом присвоєння різних рангів;
2. перевірку придатності експертних оцінок для подальшого використання;
3. визначення оцінки попарного пріоритету параметрів;
4. обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 - Результати ранжування параметрів

Позначення параметра	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
	1	2	3	4	5	6	7			
X1	3	4	2	3	4	3	4	21	5.5	30.25
X2	1	2	1	1	2	2	2	12	-6.5	42.25
X3	2	1	3	2	1	1	1	11	-6.5	42.25
X4	4	3	4	4	3	4	3	26	7.5	56.25
	10	10	10	10	10	10	10	70	0	171

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17{,}5$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 171$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 171}{7^2(4^3 - 4)} = 0.69 > W_k = 0.67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати заносимо у таблицю 4.4.

Таблиця 4.4 - Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	>	>	>	>	>	1.5
X1 і X3	>	>	<	>	>	>	>	>	1.5
X1 і X4	<	>	<	<	>	<	>	<	0.5
X2 і X3	<	>	<	<	>	>	>	>	1.5
X2 і X4	<	<	<	<	<	<	<	<	0.5
X3 і X4	<	<	<	<	<	<	<	<	0.5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j. \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю А=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i}$$

де $b_i = \sum_{j=1}^{n} a_{ij}$.

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{вi}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}$$

де $b_i' = \sum_{j=1}^{n} a_{ij} b_j$.

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 - Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітерація		Друга ітерація		Третя ітерація	
	1	2	3	4	b_i	$K_{вi}$	b_i^1	$K_{вi}^1$	b_i^2	$K_{вi}^2$
X1	1,0	1,5	1,5	0,5	4.5	0.281	16.25	0.275	59.125	0.274
X2	0,5	1,0	1,5	0,5	3.5	0.189	12.25	0.208	44.875	0.208
X3	0,5	0,5	1,0	0,5	2.5	0.156	9.25	0.157	34.125	0.158
X4	1,5	1,5	1,5	1,0	5.5	0.344	21.25	0.360	77.875	0.360
Всього:					16	1	59	1	216	1

4.3Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X1* (об'єм пам'яті для збереження даних) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X2 (*час обробки даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 800 мс або варіанту б) 80мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi} B_{ij}$$

де n – кількість параметрів;

Kвi– коефіцієнт вагомості i–го параметра; B_i – оцінка i–го параметра в балах.

Таблиця 4.6 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X4	1100	1	0.344	0.344
F2	А	X1	12	4	0.281	1.124
		X2	420	5	0.189	0.945
	Б	X1	15	5	0.281	1.405
		X2	350	4	0.189	0.756
F3	А	X3	10E-2	5	0.156	0.78

За даними з таблиці 4.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

K_{K1} =1.124+0.945+0.78+ 0,344 = 3.193

K_{K2} = 1.405+0.756+0.78+ 0,344 = 3,285

Як видно з розрахунків, кращим є другий варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка користувацького інтерфейсу для демонстрації;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як

$$T_О = T_Р \cdot К_П \cdot К_{СК} \cdot К_М \cdot К_{СТ} \cdot К_{СТ.М},$$

де $T_Р$ – трудомісткість розробки ПП;

$К_П$ – поправочний коефіцієнт;

$К_{СК}$ – коефіцієнт на складність вхідної інформації;

$К_М$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру ступеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює:

$T_Р$ =90 людино-днів.

Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання:

$К_П$ = 1.7.

Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх завдань рівний 1:

$К_{СК}$ = 1.

Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта:

$К_{СТ}$ = 0.7.

Тоді загальна трудомісткість програмування першого завдання дорівнює:

$T_1 = 90 \cdot 1.7 \cdot 0.7 = 107.1$ людино-днів.

Проведемо аналогічні розрахунки для подальших досліджень.

Для другого завдання (використовується алгоритм третьої групи складності, ступінь новизни Б), тобто

$T_Р$ =27 людино-днів,

$К_П$ =0.8,

$К_{СК}$ = 1,

$К_{СТ}$ =0.8:

$T_2 = 27 \cdot 0.8 \cdot 0.8 = 17.28$ людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$T_I = (107.1 + 17.28 + 4.8 + 17.28) \cdot 8 = 1171{,}68$ людино-годин;

$T_{II} = (107.1 + 17.28 + 6.91 + 17.28) \cdot 8 = 1188.56$ людино-годин;

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь інженер даних та веб розробник з окладом 6000 грн. кожен, а також один інженер з машинного навчання з окладом 12000 грн.

Визначимо зарплату за годину за формулою:

$$С_Ч = \frac{М}{T_m \cdot t} грн$$

де М – місячний оклад працівників;

T_m – кількість робочих днів в тиждень;

t – кількість робочих годин в день.

$$С_Ч = \frac{6000 + 6000 + 12000}{3 \cdot 21 \cdot 8} = 47.61 грн$$

Розраховуємо заробітну плату за формулою

$$С_{ЗП} = С_Ч \cdot T_i \cdot К_Д,$$

де $С_Ч$– величина погодинної оплати праці для робітника;

T_i – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП}$ = 47.61· 1171.68 · 1.2 = 66953.14 грн.

II. $С_{ЗП}$ = 47.61· 1188.56 · 1.2 = 67904.80 грн.

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику (II клас) становить 22%:

I. $С_{ВІД} = С_{ЗП}$ · 0.22 = 66953.14· 0.22 = 14729 грн.

II. $С_{ВІД} = С_{ЗП}$ · 0.22 = 67904.80 · 0.22 = 14939 грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного розробника з окладом 6000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З$ = 12 · 6000· 0,2 = 14400 грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г$· (1+ $К_З$) = 14400· (1 + 0.2)=17280 грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД}$= $С_{ЗП}$ · 0.22 = 17280· 0.22 = 3801.6 грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 20 000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР}$ = 1.15 · 0.25 · 20 000 = 5750 грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р$ = 1.15 · 20 000 · 0.05 = 1150 грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В$ = (365 – 104 – 8 – 16) · 8 · 0.9 = 1706.4 годин,

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706.4 \cdot 0.3 \cdot 0.8 \cdot 9.43 = 3861.95$ грн.,

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 20000 \cdot 0,67 = 13400$ грн.

Тоді, річні експлуатаційні витрати будуть:

$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$

$С_{ЕКС} = 17280 + 3801.6 + 5750 + 1150 + 3861.95 + 13400 = 45243,55$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 45243,55 / 1706,4 = 26.55$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$С_М = С_{М\text{-}Г} \cdot Т$

I. $С_М = 26.55 \cdot 1171,68 = 31108,15$ грн.

II. $С_М = 26.55 \cdot 1188.56 = 31558,66$ грн.

Накладні витрати складають 67% від заробітної плати:

$С_Н = С_{ЗП} \cdot 0,67$

I. $С_Н = 66953.14 \cdot 0,67 = 44858.60$ грн.

II. $С_Н = 67904.80 \cdot 0,67 = 45396.22$ грн.

Отже, вартість розробки ПП становить:

$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$

I. $С_{ПП} = 66953.14 + 24618 + 31108,15 + 44858.60 = 167537,89$ грн.

II. $С_{ПП} = 67904.80 + 24968 + 31558,66 + 45396.22 = 169557,68$ грн.

4.4Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$К_{ТЕРj} = К_{Кj} / С_{Фj}$,

$К_{ТЕР1} = 3.193/167537,89 = 1.906 \cdot 10^{-5}$,

$К_{ТЕР2} = 3,285 / 169557,68 = 1.9537 \cdot 10^{-5}$.

Як бачимо, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 1.951 \cdot 10^{-5}$.

Висновки до розділу

В даному розділі проведено повний функціонально-вартісний аналіз програмного продукту, який було розроблено в рамках дипломної роботи. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження програмного продукту з технічної точки зору: було визначено основні функції програмного продукту та сформовано множину варіантів їх реалізації. На основі створеної морфологічної карти було створено позитивно-негативно матрицію, яка і дала змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій програмного продукту. Другу частину функціонально-вартістного аналізу присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що були залишені першого відбору двох варіантів виконання програмного

комплексу оптимальним є варіант реалізації програмного продукту за допомогою пакету Matlab. Його показник техніко-економічного рівня якості $К_{ТЕР} = 1.9537 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

1) мова програмування – Matlab;
2) використання функція pdepe пакету Matlab;
3) використання вбудованого пакету візуалізації пакету Matlab.

Даний варіант виконання програмного продукту дає досліднику точну та швидку систему.

ВИСНОВКИ ДО РОБОТИ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Дипломну роботу присвячено розробці та дослідженню композиційного методу розв'язування збурених параболічних диференціальних рівнянь з частинними похідними з акцентом на задачі Коші. Ці рівняння, які описують широкий клас залежних від часу процесів з просторовою неоднорідністю, є важливими для опису систем теплопровідності, масопереносу, динаміки популяцій, фінансової математики та інших наукових галузей. Особлива проблема, яку було вирішено, полягала в точному і стабільному чисельному опрацюванні збурень, що моделюються диференціальними операторами першого порядку з просторово змінними коефіцієнтами-термами, які відображають реальні явища, такі як дрейф, конвекція або анізотропний перенос.

Дослідження, представлені в цій роботі (див. також [13]), підтверджують актуальність і потужність операторно-теоретичних підходів у розв'язанні таких проблем. Основний внесок полягає в адаптації та застосуванні формули композиції Троттера для розкладання керуючого оператора на суму простіших компонент, кожна з яких породжує напівгрупу. Така декомпозиція уможливлює покрокове чисельне наближення розв'язку за допомогою ітераційних процедур, тим самим підвищуючи чіткість математичного формулювання та стабільність обчислювального методу. Важливо, що така структура дозволяє проводити модульні обчислення: дифузійні умови можна розглядати окремо від збурень, що відкриває шлях до гнучкості у виборі методу, часового кроку та стратегій дискретизації.

Дипломна робота включає строгий теоретичний аналіз збіжності методу композиції за наявності збурень. Встановлено оцінки похибки та критерії збіжності, які підтверджують робастність методу навіть за неідеальних або неоднорідних умов. Крім того, було побудовано практичний

алгоритм і протестовано його в широко використовуваних середовищах - MATLAB і Python - де його точність і обчислювальна ефективність були підтверджені як на модельних, так і на збурених випадках. Ці результати підтверджують не лише коректність методу, але й його готовність до ширшого застосування.

Окрім технічного успіху, дослідження ілюструє зростаючу важливість модульних чисельних методів у прикладній математиці. Оскільки сучасні обчислювальні моделі стають дедалі складнішими - включають багатомасштабні ефекти, просторову неоднорідність або випадковість - традиційні монолітні розв'язувачі стикаються з обмеженнями масштабованості та стабільності. Метод композиції вирішує ці проблеми, пропонуючи природно розпаралелювану і структурно прозору альтернативу. Це робить його добре придатним для великомасштабних симуляцій, високопродуктивних обчислювальних середовищ і сценаріїв моделювання в реальному часі.

Більше того, включення функціонально-вартісного аналізу в цю роботу підкреслює практичну цінність методу за межами теоретичного рівня. Він показує, що підхід може бути економічно життєздатним і обчислювально ефективним, що є життєво важливим для його інтеграції в реальні додатки, особливо в системах з обмеженими ресурсами або чутливих до часу.

Що стосується майбутніх досліджень, то метод, розроблений у цій дипломній роботі, відкриває кілька перспективних напрямків:

1. Теоретичне узагальнення на ширші класи звичайних диференціальних рівнянь, включаючи нелінійні та вироджені рівняння, значно розширить спектр розв'язуваних моделей.

2. Інтеграція зі стохастичними методами моделювання, особливо для систем, що зазнають впливу шуму або невизначеності, дозволить застосувати метод до стохастичних диференціальних рівнянь, що ще більше підвищить його корисність у моделюванні реальних середовищ.

3. Можна розробити методи адаптивності та уточнення сітки для кращого врахування локалізованих явищ або різких границь розділу, покращуючи роздільну здатність та обчислювальну продуктивність рішення.

4. Бібліотеки програмного забезпечення і реалізації з відкритим вихідним кодом можуть зробити цей метод доступним для ширшої спільноти, заохочуючи співпрацю між дисциплінами і підвищуючи відтворюваність і зручність використання.

5. Міждисциплінарна валідація, особливо у співпраці з дослідниками з фізики, біології чи економіки, може забезпечити цінний зворотній зв'язок і критерії для вдосконалення методу для задоволення конкретних потреб застосування.

У ширшому контексті тема цієї дипломної роботи є відповіддю на фундаментальний і своєчасний виклик: необхідність розробки математичних інструментів, які подолають розрив між ідеалізованими моделями і складністю реальних систем. Збурення - детерміновані чи стохастичні - є не винятком, а правилом у природних та інженерних системах. Точне врахування цих впливів має вирішальне значення для прогнозування, розуміння поведінки системи та прийняття обґрунтованих рішень у практичному контексті.

Отже, результати цього дослідження демонструють не лише доцільність, але й ефективність та адаптивність методу композиції на основі формули Троттера для розв'язування збурених параболічних рівнянь. Це робить внесок як у теоретичне розуміння операторних чисельних методів, так і в розробку стійких алгоритмів для прикладних задач моделювання. При подальшому вдосконаленні та міждисциплінарному дослідженні цей підхід має потенціал стати стандартним інструментом в інструментарії сучасного науковця, що займається обчисленнями.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Goldstein J. A. Semigroups of Linear Operators and Applications: Dover Publications Inc.; 2e edition, 2017, 256 p.

2. Engel K.-J., Nagel R. One-Parameter Semigroups for Linear Evolution Equations. Springer-Verlag, New York, 2000, 589 p.

3. Friedman A. Partial Differential Equations. Krieger, New York, 1976, 262 p.

4. Yagi A. Abstract Parabolic Evolution Equations and their Applications. Sprincer. 2010, ISBN 978 3 642 26179 4, 599 p.

5. Murray J.D. Mathematical Biology, Vol. 1,2: Springer-Verlag, 3rd edition, New York, 2002, 576 p.

6. Bramson M. Convergence of solutions of the Kolmogorov equation to traveling waves, Mem. AMS, No. 285, 1983, P. 203-210.

7. Trotter T.F. Of the product of semi-groups of operators. Pros.Am.Math.Soc., 1959, P. 545-551.

8. Daletskii Yu. Functional integrals connected with operator evolution equations, Russian Mathematical Surveys, V.17, No. 5. 1962, P. 1–107.

9. Taylor M.E. Partial Differential Equations III, Springer Verlag, New York, 1997, 715 p.

10. Bondarenko V.G. Trotter–Daletskii Formula for Nonlinear Disturbances. Ukrainian Mathematical Journal. Vol. 70, Issue 12, Kyiv, 2019, P.1978-1984.

11. Bondarenko V. Markevych I. Composition method for semi-linear parabolic systems. International Journal of Applied and Computational Mathematics. Vol.10, No 58, ISSN 2199-5796, Kyiv, 2024, P. 1-17.

12. Bressan A. Open questions in the theory of one dimensional hyperbolic conservation laws. The IMA Volumes in Mathematics and its Applications. Volume 153. Nonlinear Conservation Laws and Applications, Kyiv, 2011, P. 1-22.

13. Bondarenko V. Zahrebelna M. Trotter`s iterative method for first-order perturbation. International Journal of Applied and Computational Mathematics. UDC 517.956, Kyiv, 2025, P. 1-10 (подана до друку)

ДОДАТОК А ІЛЮСТРАТИВНІ МАТЕРІАЛИ ДО ДОПОВІДІ

Метод композиції розв‘язування збурених параболічних рівнянь

Виконала:
студентка 4 курсу групи КА-12
Загребельна Марина Віталіївна

Науковий керівник:
професор, доктор фіз.-мат. наук
Бондаренко Віктор Григорович

Київ – 2025

ВСТУП

У дипломній роботі досліджено задачу побудови розв'язку збуреного параболічного рівняння, де збурення задано диференціальним оператором першого порядку з просторово змінним коефіцієнтом. Такі рівняння описують фізичні процеси з додатковими впливами, які потребують спеціального підходу до моделювання.

Об'єкт дослідження: математичні методи розв'язування параболічних рівнянь з розподіленими параметрами.

Предмет дослідження: застосування методу композиції операторів до збурених параболічних рівнянь.

Мета роботи: побудувати, обґрунтувати та реалізувати метод розв'язування задачі Коші зі збуренням на основі формули Троттера, а також проаналізувати точність чисельного розв'язку.

Постановка задачі дипломної роботи

1. Визначити методи композиції розв‘язування збурених параболічних рівнянь
2. Інтерпретація формули Троттера для збурення оператором першого порядку.
3. Побудова алгоритму розв‘язування збуреного рівняння.
4. Виконання чисельного експерименту і порівняння розв‘язків задачі Коші, отриманих різними методами.
5. Аналіз отриманих результатів та визначення висновків.

Огляд чисельних методів для параболічних рівнянь

У задачах зі збуреними параболічними рівняннями **Метод скінченних різниць (МСР)** дозволяє локально модифікувати схему для врахування зовнішніх збурень або стохастичних ефектів. Його основна ідея полягає в заміні похідних скінченно-різницевими апроксимаціями, тим самим зводячи неперервну задачу до системи алгебраїчних рівнянь.

Цей метод замінює похідні у рівняннях на різницеві співвідношення. Розв'язок шукається в дискретних вузлах сітки. Підходить для рівномірних просторових та часових сіток.

Переваги:

- Простота реалізації, зручний для регулярних сіток
- Легкість у застосуванні до лінійних рівнянь
- Широко підтримується у чисельних програмних середовища.

Недоліки:

- Обмежена точність при складних геометріях
- Високі вимоги до часу для забезпечення стійкості (особливо явні схеми)
- Чутливість до вибору часу для стабільності

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{D}{\Delta x^2}\left(u_{j-1}^* - 2u_j^* + u_{j+1}^*\right) + f_j^*$$

Огляд чисельних методів для параболічних рівнянь

Метод скінченних елементів (МСЕ) є одним із найпотужніших чисельних методів для розв'язання диференціальних рівнянь у частинних похідних, особливо на складних геометричних областях. Базується на поділі області на елементи (трикутники, чотирикутники), у яких шукається апроксимація розв'язку. Рівняння формуються на основі варіаційного принципу. Висока точність на складних геометріях, зручний для адаптивних сіток.

Переваги:

- Гнучкість у роботі зі складною геометрією та неоднорідними матеріалами
- Висока точність при використанні адаптивних сіток
- Добре підходить для задач із нелінійними та стохастичними коефіцієнтами

Недоліки:

- Складність реалізації, особливо для багатовимірних задач
- Потребує обчислення інтегралів на елементах сітки, що збільшує витрати

$$u_h(x,t) = \sum_{j=1}^{N} U_j(t)\phi_j(x)$$

Огляд чисельних методів для параболічних рівнянь

Спектральні методи — це високоточні чисельні підходи, що ґрунтуються на апроксимації шуканої функції глобальними базисними функціями, такими як тригонометричні функції (метод Фур'є), поліноми Чебишева, Лежандра, Ерміта тощо. Спектральні методи застосовують розклад по **всій обчислювальній області**, що забезпечує **експоненційно високу точність** для гладких функцій. Дає високу точність при гладких розв'язках, але менш ефективний для розривних або нерегулярних задач.

Переваги:

- Надзвичайно висока точність для гладких функцій
- Експоненціальна збіжність для добре обумовлених задач
- Швидкість розв'язання для регулярних геометрій

Недоліки:

- Нестійкість при обробці задач з розривами або негладкими функціями
- Важко застосовувати до складних геометрій чи неоднорідних середовищ

$$u(x,t) = \sum_{k=0}^{N} \widehat{u_k}(t)\phi_k(x)$$

Метод Троттера

Формула Троттера (метод Троттера) є фундаментальною технікою для чисельного розв'язання еволюційних рівнянь, особливо таких, де оператор можна природно розкласти на простіші складові. Для лінійного еволюційного рівняння вигляду:

$$\frac{du}{dt} = (A + B)u,$$

метод Троттера апроксимує розв'язок на малому часовому кроці Δt за формулою:

$$u(t + \Delta t) \approx e^{\Delta t A} e^{\Delta t B} u(t),$$

що дозволяє **розділити складний оператор на простіші частини**, які вирішуються послідовно. Метод особливо ефективний для **параболічних рівнянь** із збуреннями, де оператор природно розділяється на дифузійну частину та збурений або реакційний компонент.

Метод Троттера

Метод Троттера особливо ефективний у випадках, коли оператор має **виражену структуру розщеплення** — наприклад, у задачах дифузії-реакції або в системах із виразними лінійними та нелінійними частинами. Традиційні методи, хоч і потужні, часто вимагають більших обчислювальних витрат або не мають такої модульності, як метод розщеплення.

Попри те, що **метод скінченних різниць, метод скінченних елементів і спектральні методи** є перевіреними й ефективними, їх використання може бути обмежене обчислювальними витратами, обмеженнями стабільності або складністю адаптації до збурених систем. **Метод Троттера** забезпечує **модульний, ефективний і часто більш стабільний підхід** для задач із природною структурою розщеплення. Його здатність інтегруватися з традиційними схемами робить його **надзвичайно перспективним інструментом для сучасних обчислювальних задач** із параболічними рівняннями та їх збуреннями.

Метою даної роботи є дослідження збіжності у формулі Троттера для генераторів $L = \frac{\partial^2}{\partial x^2}$, $L_1 = b(x)\frac{\partial}{\partial x}$, тобто побудова ітерацій розв'язку задачі Коші

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + b(x)\frac{\partial u}{\partial x}, 0 < t \le S \\ u(0,x) = \varphi(x) \end{cases} \quad (1)$$

Сформулюємо умови на коефіцієнт $b(x)$:

$$\begin{cases} \varepsilon \le |b(x)| \le c_1 \\ |b^{(k)}(x)| \le c_1, k = 1,2 \end{cases}$$

За цих умов визначено функцію

$g(t,x) = F^{-1}\big(t + F(x)\big)$, де $F(x) = \int \frac{dx}{b(x)}$, F^{-1} – функція,

обернена до $F(x)$, і розв'язок задачі Коші для рівняння першого порядку

$$\begin{cases} \frac{\partial v}{\partial t} = b(x)\frac{\partial v}{\partial x} \\ v(0,x) = \varphi(x) \end{cases}$$

має вигляд $v(t,x) = (e^{tL_1}\varphi)(x) = \varphi\big(g(t,x)\big)$.

Похідні функції :

$$\frac{\partial g}{\partial s} = (F^{-1})'\big(s + F(x)\big) = \frac{1}{F'\big(F^{-1}(s+F(x))\big)} = b\big(g(s,x)\big);$$

$$\frac{\partial g}{\partial x} = (F^{-1})'\big(s + F(x)\big)\frac{1}{b(x)} = \frac{b\big(g(s,x)\big)}{b(x)};$$

$$\frac{\partial^2 g}{\partial x^2} = \frac{b\big(g(s,x)\big)}{b^2(x)}\Big(b'\big(g(s,x)\big) - b'(x)\Big).$$

Звідси випливають оцінки

$$\begin{cases} |g(s,x) - x| \le c_2 s \\ \left|\frac{\partial g}{\partial s}(s,x)\right| < c_2 \\ \left|\frac{\partial^2 g}{\partial x^2}(s,x)\right| < c_2 s \end{cases}$$

Півгрупа, породженна генератором $L = \frac{\partial^2}{\partial x^2}$, має вигляд

$(e^{tL}\varphi)(x) = \int_{-\infty}^{\infty} \varphi(y)p(t,x,y)dy$, де $p(t,x,y) = \frac{1}{2\sqrt{\pi t}} exp\left\{-\frac{(x-y)^2}{4t}\right\}$.

Дослідимо властивості ядра інтегрального оператора $e^{tL_1}e^{tL}$—функції

$$q(t,x,y) = p(t,g(t,x),y) = \frac{1}{2\sqrt{\pi t}} exp\left\{-\frac{(g(t,x)-y)^2}{4t}\right\},$$

що задовольняє очевидної оцінки

$$q(t,x,y) < e^{\frac{c_1}{2}|x-y|}\, p(t,x,y) \qquad \textbf{(3)}$$

Після дослідження отримали, що функція q(t,x,y) задовольняє рівнянню $\frac{\partial q}{\partial t} = \frac{\partial^2 q}{\partial x^2} + b(x)\frac{\partial q}{\partial x} + m(t,x,y)$, де нев'язка

$$m(t,x,y) = \frac{1}{b^2(x)} p''(t,g(t,x),y)\left(b^2(x) - b^2\big(g(t,x)\big)\right) -$$

$$-\frac{1}{b^2(x)} p'(t,g(t,x),y)b\big(g(t,x)\big)\left(b'\big(g(t,x)\big) - b'(x)\right)$$

Побудуємо ітераційну процедуру за формулою Троттера. Позначимо:

1)$\Delta_1, \Delta_2, \dots, \Delta_n$ —розбиття відрізку $[0;T]$, $T \le S$:

$\Delta_k = \left[(k-1)\frac{T}{n}; k\frac{T}{n}\right]$,

$[0;T] = \cup_{k=1}^{n} \Delta_k$; $t \in \Delta_k \Leftrightarrow t = s + (k-1)\frac{T}{n}$, $0 \le s \le \frac{T}{n}$.

та визначимо функції

$r_1(s,x) = \int_{-\infty}^{\infty} \varphi(y)q(s,x,y)dy$

$r_2\left(s+\frac{T}{n},x\right) = \int_{-\infty}^{\infty} r_1\left(\frac{T}{n},y\right) q(s,x,y)dy$

.....

$r_j\left(s+\frac{(j-1)T}{n},x\right) = \int_{-\infty}^{\infty} r_{j-1}\left(\frac{(j-1)T}{n},y\right) q(s,x,y)dy$, $j \le n$,

$r_j\left(\frac{jT}{n},x\right) = r_{j+1}\left(\frac{jT}{n},x\right)$.

В термінах півгруп

$r_j\left(s+\frac{(j-1)T}{n}\right) = e^{sL_1}e^{sL}\, r_{j-1}\left(\frac{(n-1)T}{n}\right)$, $r_j\left(\frac{jT}{n},x\right) = \left(e^{\frac{T}{n}L_1}\ e^{\frac{T}{n}L}\right)^j \varphi$.

Твердження 1. Нев'язка m задовольняє нерівності

$$|m(t,x,y)| < c_3\left(1+\frac{(x-y)^2}{t}\right)e^{\frac{c_1}{2}|x-y|}\, p(t,x,y)$$

Доведення. Оскільки

$p'(t,g(t,x),y) = -\frac{g(t,x)-y}{2t} q(t,x,y)$,

$p''(t,g(t,x),y) = \left(-\frac{1}{2t} + \frac{(g(t,x)-y)^2}{4t^2}\right) q(t,x,y)$,

твердження випливає з оцінок (2) та (3) ■

Оцінимо вираз

$$\int_{-\infty}^{\infty}\left(\frac{(x-y)^2}{t}\right)^k e^{a|y-x|}\, p(t,x,y)dy,\ k \in 1,2, a > 0$$

Твердження2. Справедлива оцінка

$$\int_{-\infty}^{\infty}\left(\frac{(x-y)^2}{t}\right)^k e^{a|x-y|}p(t,x,y)dy < Ce^{Bt},\ B > a^2$$

Наслідок. Справедлива нерівність

$M^2(t,x) < c_4 e^{c_5 t}\int_{-\infty}^{\infty} h^2(y)p(s,x,y)dy$

Доведення. За нерівністю Коші-Буняковського та за твердженням 1

$M^2(s,x) < 2c_3^2 \int_{-\infty}^{\infty} h^2(y)p(s,x,y)dy \int_{-\infty}^{\infty} e^{c_1|x-y|}\left(1+\frac{(x-y)^4}{t^2}\right)p(s,x,y)dy.$

За твердженням 2, другий множник оцінюється експонентою:

$\int_{-\infty}^{\infty} e^{c_1|x-y|}\left(1+\frac{(x-y)^4}{t^2}\right)p(s,x,y)dy < c_4 e^{c_5 t}$ ■

Теорема. Для довільного $T \in (0;S]$ справедливе співвідношення

$$\lim_{n\to\infty} \max_{x\in R} |r_n(T,x) - u(T,x)| = 0,$$

що означає збіжність за нормою простору $C(R)$.

Після доведення отримали, що швидкість збіжності ітерацій у формулі Троттера в просторі $G(R)$ визначається співвідношенням

$$\left\| \left(e^{T(L+L_1)}\varphi - \left(e^{\frac{T}{n}L_1} e^{\frac{T}{n}L} \right)^n \varphi \right) \right\| < \|\varphi\| \sqrt{\frac{AT}{n}}\, e^{KT}$$

Чисельне моделювання

Об‘єктом чисельного експерименту є задача Коші для рівняння дифузії із зсувом:

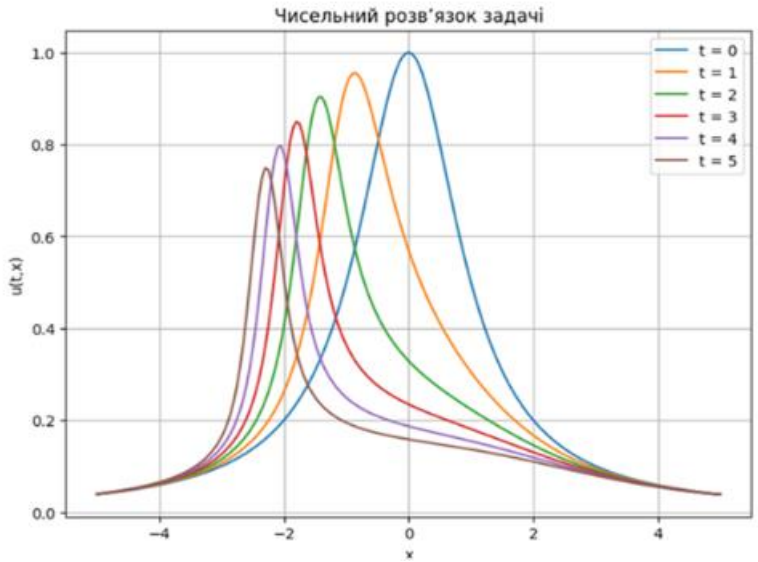
$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{1}{chx}\frac{\partial u}{\partial x} \\ u(0,x) = \frac{1}{1+x^2} \end{cases}$$

Аналітичний розв’язок задачі Коші для генератора першого порядку

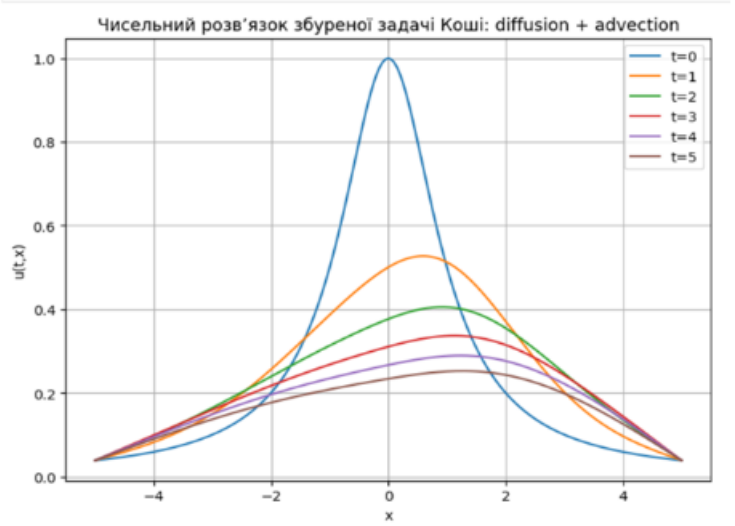
$$\begin{cases} \frac{\partial g}{\partial t} = \frac{1}{chx}\frac{\partial g}{\partial x} \\ g(0,x) = x \end{cases}$$ має вигляд $g(t,x) = ln\left(t + shx + \sqrt{(t+shx)^2+1}\right)$, звідки

$$\begin{cases} \frac{\partial v}{\partial t} = \frac{1}{chx}\frac{\partial v}{\partial x} \\ v(0,x) = \frac{1}{1+x^2} \end{cases} \Rightarrow v(t,x) = \left(1 + \left(ln\left(t + shx + \sqrt{(t+shx)^2+1}\right)\right)^2\right)^{-1}$$

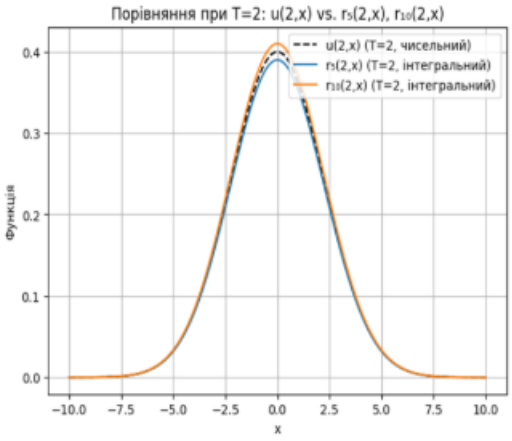
Графічне зображення функції міститься на рисунку

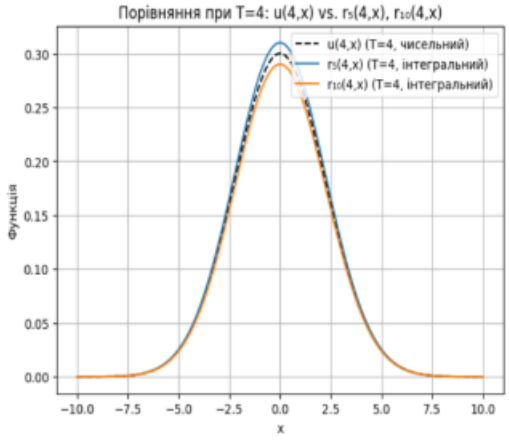


Розв'язок задачі Коші, отриманий чисельним методом, показує зниження максимуму, що відповідає теоретично очікуваному згладженню. Графічне зображення наведено на рисунку



Обчислимо значення функції $r_n(T,x)$, що визначена співвідношенням (6)— розв'язок, отриманий ітераційним методом, та порівняємо цю функцію з обчисленим раніше розв'язком $u(T,x)$ для $T=2$ та $T=4$. Графічне зображення $r_5(2,x)$, $r_{10}(2,x)$, $u(2,x)$ наведено на рис.3, $r_5(4,x)$, $r_{10}(4,x)$, $u(4,x)$— на рис.4.





Наведемо значення відхилень $\delta(n,T) = \max_x \frac{|u(T,x)-r_n(T,x)|}{u(T,x)}$

$\delta(5,2)$	$\delta(10,2)$	$\delta(5,4)$	$\delta(10,4)$
1.375e-01	**1**.314e-01	4.001e-01	3.630e-01

ВИСНОВКИ

Результати чисельного моделювання підтверджують збіжність ітерацій і відповідають встановленій залежності відхилення $\delta(n, T)$ від кількості ітерацій n та часового інтервалу T

При моделюванні еволюційних процесів для врахування нових факторів (дифузії, зсуву і т.д.) загальноприйнятим методом є додавання у еволюційне рівняння відповідного генератора. Насправді обгрунтування такого методу відсутнє, він є емпіричним, і в ряді випадків адекватність моделей вимагає застосування композиції відповідних півгруп.

Напрямом продовження досліджень може бути вивчення збурення нелінійним диференціальним оператором першого порядку, зокрема, квазілінійним генератором $L_1(u) = b(u)\frac{\partial u}{\partial x}$. В цьому випадку виникає наступна проблема—не існує глобального класичного розв'язку задачі Коші

$$\begin{cases} \frac{\partial u}{\partial t} = b(u)\frac{\partial u}{\partial x} \\ u(0,x) = \varphi(x) \end{cases}$$

що випливає навіть із неявного зображення $u = \varphi\big(x + tb(u)\big)$

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Goldstein J. A. Semigroups of Linear Operators and Applications: Dover Publications Inc.; 2e edition, 2017, 256 p.
2. Engel K.-J., Nagel R. One-Parameter Semigroups for Linear Evolution Equations. Springer-Verlag, New York, 2000, 589 p.
3. Friedman A. Partial Differential Equations. Krieger, New York, 1976, 262 p.
4. Yagi A. Abstract Parabolic Evolution Equations and their Applications. Sprincer. 2010, ISBN 978 3 642 26179 4, 599 p.
5. Murray J.D. Mathematical Biology, Vol. 1,2: Springer-Verlag, 3rd edition, New York, 2002, 576 p.
6. Bramson M. Convergence of solutions of the Kolmogorov equation to traveling waves, Mem. AMS, No. 285, 1983, P. 203-210.
7. Trotter T.F. Of the product of semi-groups of operators. Pros.Am.Math.Soc., 1959, P. 545-551.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

8. Daletskii Yu. Functional integrals connected with operator evolution equations, Russian Mathematical Surveys, V.17, No. 5. 1962, P. 1–107.

9. Taylor M.E. Partial Differential Equations III, Springer Verlag, New York, 1997, 715 p.

10. Bondarenko V.G. Trotter–Daletskii Formula for Nonlinear Disturbances. Ukrainian Mathematical Journal. Vol. 70, Issue 12, Kyiv, 2019, P.1978-1984.

11. Bondarenko V. Markevych I. Composition method for semi-linear parabolic systems. International Journal of Applied and Computational Mathematics. Vol.10, No 58, ISSN 2199-5796, Kyiv, 2024, P. 1-17.

12. Bressan A. Open questions in the theory of one dimensional hyperbolic conservation laws. The IMA Volumes in Mathematics and its Applications. Volume 153. Nonlinear Conservation Laws and Applications, Kyiv, 2011, P. 1-22.

13. Bondarenko V. Zahrebelna M. Trotter`s iterative method for first-order perturbation. International Journal of Applied and Computational Mathematics. UDC 517.956, Kyiv, 2025, P. 1-10 (подана до друку)

ДОДАТОК Б ЛІСТИНГ ПРОГРАМИ

```
import numpy as np
import matplotlib.pyplot as plt
def ch(x):
    return 0.5*(np.exp(x) + np.exp(-x)
# Параметри сітки
x_min, x_max = -5.0, 5.0   # межі по x
Nx = 201                   # кількість вузлів по x
dx = (x_max - x_min)/(Nx - 1)
x = np.linspace(x_min, x_max, Nx)

dt = 0.005
Tmax = 5.0

def initial_condition(x):
    return 1.0/(1.0 + x*x)

u = initial_condition(x)

u_new = np.copy(u)

times_to_save = [0,1,2,3,4,5]
solutions = {t: None for t in times_to_save}
solutions[0] = np.copy(u)

# Функція швидкості c(x) = -1/ch(x)
```

```
def velocity(x):
    return -1.0/ch(x)

# Поточний час
t = 0.0
nsteps = int(Tmax/dt)

for n in range(nsteps):
   t += dt
   # Для кожного вузла обчислимо u_{j}^{n+1}
   for j in range(0, Nx-1):
       c_j = velocity(x[j])
       # upwind-схема для c_j<0:
       # u_{j}^{n+1} = u_{j}^n - c_j * (dt/dx) * (u_{j+1}^n
- u_{j}^n)
       u_new[j] = u[j] - c_j * (dt/dx) * (u[j+1] - u[j])

       # Гранична умова справа (x_max): u(t,x_max) =
початкове значення
       u_new[Nx-1] = initial_condition(x[Nx-1])

       # Перезаписуємо
       u[:] = u_new[:]

       # Якщо час «близький» до цілого 1,2,3,4,5 –
зберігаємо
       if is_time_to_save(t, times_to_save):
           solutions[int(round(t))] = np.copy(u)
```

```
plt.figure(figsize=(8,6))
print(solutions)
for tau in times_to_save:
    #print(x, solutions[tau], f"t = {tau}")
    plt.plot(x, solutions[tau], label=f"t = {tau}")
plt.xlabel("x")
plt.ylabel("u(t,x)")
plt.title("Чисельний розв’язок задачі")
plt.legend()
plt.grid(True)
plt.show()


def ch(x):
    return 0.5*(np.exp(x) + np.exp(-x))

# Початкова умова
def initial_condition(x):
    return 1.0/(1.0 + x*x)

# Швидкість (від’ємна)
def velocity(x):
    """ c(x) = -1 / ch(x) """
    return -1.0/ch(x)

# Параметри сітки
x_min, x_max = -5.0, 5.0   # межі по x
Nx = 201                   # кількість вузлів по x
```

```
dx = (x_max - x_min)/(Nx - 1)
x = np.linspace(x_min, x_max, Nx)

dt = 0.005
Tmax = 5.0
nsteps = int(Tmax / dt)

# Ініціалізуємо розв’язок
u = initial_condition(x)
u_new = np.copy(u)

times_to_save = [0, 1, 2, 3, 4, 5]
solutions = {t: None for t in times_to_save}
solutions[0] = np.copy(u)  # зберігаємо початкову умову

def is_time_to_save(t, tlist, eps=1e-3):
    """Перевіряє, чи час t (з точністю eps) належить
tlist."""
    return any(abs(t - tau) < eps for tau in tlist)

# Основний цикл за часом
t = 0.0
for n in range(nsteps):
    t += dt

    # Обчислюємо u_{j}^{n+1} за upwind-схемою
    for j in range(Nx - 1):
        c_j = velocity(x[j])
```

```
            u_new[j] = u[j] - c_j * (dt/dx) * (u[j+1] - u[j])

        # Гранична умова справа (x_max): "вхідний" край
        u_new[Nx - 1] = initial_condition(x[Nx - 1])

        # Оновлюємо масив
        u[:] = u_new[:]

        # Перевіряємо, чи треба зберегти
        if is_time_to_save(t, times_to_save):
            solutions[int(round(t))] = np.copy(u)

for tau in times_to_save:
    plt.figure(figsize=(8, 5))
    plt.plot(x, solutions[tau], label=f"t = {tau}")
    plt.xlabel("x")
    plt.ylabel("u(t, x)")
    plt.title(f"Чисельний розв’язок, t = {tau}")
    plt.legend()
    plt.grid(True)
    plt.show()

#    u(t,x) = [ 1 + ( ln( t + sinh(x) + sqrt((t + sinh(x))^2
+ 1 ) ) )^2 ]^(-1)

def exact_solution(t, x):
    val = t + np.sinh(x)
    inside_log = val + np.sqrt(val**2 + 1)
```

```
    return 1.0 / (1.0 + (np.log(inside_log))**2)



def ch(x):
    return 0.5*(np.exp(x) + np.exp(-x))  # те саме, що
np.cosh(x)


def initial_condition(x):
    return 1.0/(1.0 + x*x)


def velocity(x):
    # c(x) = -1 / cosh(x)
    return -1.0/ch(x)


# Параметри сітки
x_min, x_max = -5.0, 5.0
Nx = 201
dx = (x_max - x_min)/(Nx - 1)
x_grid = np.linspace(x_min, x_max, Nx)



# Початковий розв’язок
u = initial_condition(x_grid)
u_new = np.copy(u)

# Зберігатимемо розв’язок у моменти часу t = 0,1,2,3,4,5
times_to_save = [0,1,2,3,4,5]
solutions = {t: None for t in times_to_save}
```

```
solutions[0] = np.copy(u)

def is_time_to_save(t, tlist, eps=1e-4):
    return any(abs(t - tau) < eps for tau in tlist)

# Основний цикл за часом
t = 0.0
for n in range(nsteps):
    t += dt

    # Upwind, бо c(x) < 0
    for j in range(Nx - 1):
        c_j = velocity(x_grid[j])
        u_new[j] = u[j] - c_j*(dt/dx)*(u[j+1] - u[j])

    # Гранична умова справа
    u_new[Nx - 1] = initial_condition(x_grid[Nx - 1])

    # Оновлення
    u[:] = u_new[:]

test_points = [(2, -2), (2, 1)]

print("Порівняння чисельного й аналітичного розв’язку:")
for (Tq, Xq) in test_points:
    # Аналітичне значення:
    u_exact = exact_solution(Tq, Xq)
```

```
    # Вибираємо збережений чисельний розв’язок для t = Tq
    u_num_array = solutions[Tq]  # це масив значень u(t,x)
на сітці x_grid


    # Знаходимо вузол x_j, найближчий до Xq
    j_closest = np.argmin(np.abs(x_grid - Xq))
    xj = x_grid[j_closest]
    u_numerical = u_num_array[j_closest]



def exact_solution(t, x):
    val = t + np.sinh(x)
    inside_log = val + np.sqrt(val**2 + 1)
    return 1.0 / (1.0 + (np.log(inside_log))**2)

def ch(x):
    return 0.5*(np.exp(x) + np.exp(-x))  # те саме, що
np.cosh(x)

def initial_condition(x):
    return 1.0/(1.0 + x*x)

def velocity(x):
    # c(x) = -1 / cosh(x)
    return -1.0/ch(x)

# Параметри сітки
x_min, x_max = -5.0, 5.0
Nx = 201
```

```
dx = (x_max - x_min)/(Nx - 1)
x_grid = np.linspace(x_min, x_max, Nx)

# Початковий розв’язок
u = initial_condition(x_grid)
u_new = np.copy(u)

def is_time_to_save(t, tlist, eps=1e-4):
    return any(abs(t - tau) < eps for tau in tlist)

# Основний цикл за часом
t = 0.0
for n in range(nsteps):
    t += dt

    # Upwind, бо c(x) < 0
    for j in range(Nx - 1):
        c_j = velocity(x_grid[j])
        u_new[j] = u[j] - c_j*(dt/dx)*(u[j+1] - u[j])

    # Гранична умова справа
    u_new[Nx - 1] = initial_condition(x_grid[Nx - 1])

test_points = [
    (2, -2), (2,  1),
    (3, -1), (3,  2),
    (4, -3), (4,  0.5),
    (5, -4), (5,  1),
```

```
]

print("Порівняння чисельного й аналітичного розв’язку:\n")
for (Tq, Xq) in test_points:
    # Аналітичне значення:
    u_exact = exact_solution(Tq, Xq)

    # Вибираємо збережений чисельний розв’язок для t = Tq
    # (маємо його лише для цілих t у [0..5])
    u_num_array = solutions[Tq]  # це масив на сітці x_grid

    # Знаходимо вузол x_j, найближчий до Xq
    j_closest = np.argmin(np.abs(x_grid - Xq))
    xj = x_grid[j_closest]
    u_numerical = u_num_array[j_closest]

    print(f"  t={Tq}, x={Xq:5.2f}")
    print(f"    Аналітичне:  u_exact = {u_exact:.5f}")
    print(f"    Сітка:       x_j = {xj:.3f}")
    print(f"    Чисельне:    u_num =  {u_numerical:.5f}")
    print(f"    Похибка:     {abs(u_exact -
u_numerical):.5e}\n")

def ch(x):
    """Гіперболічний косинус."""
    return 0.5*(np.exp(x) + np.exp(-x))

def velocity(x):
```

```
    """Швидкість a(x) = 1/ch(x), завжди > 0."""
    return 1.0/ch(x)

def initial_condition(x):
    """Початкова умова: u(0,x) = 1/(1 + x^2)."""
    return 1.0/(1.0 + x*x)

# Параметри розрахунку
x_min, x_max = -5.0, 5.0     # межі по x
Nx = 201                     # кількість вузлів
dx = (x_max - x_min)/(Nx - 1)
x = np.linspace(x_min, x_max, Nx)

# Часові налаштування
dt = 0.0005      # крок часу, досить малий через дифузію (та адvекцію)
Tmax = 5.0
nsteps = int(Tmax/dt)

# Ініціалізація u
u = initial_condition(x)
u_new = np.copy(u)

# Збережемо розв'язок у моменти t = 0,1,2,3,4,5
times_to_save = [0,1,2,3,4,5]
solutions = {}
solutions[0] = np.copy(u)
```

```
def is_time_to_save(t, tlist, eps=1e-6):
    return any(abs(t - tau) < eps for tau in tlist)


t = 0.0
for n in range(nsteps):
    t += dt
)
    for j in range(1, Nx-1):
        diff_term = (u[j-1] - 2*u[j] + u[j+1])/(dx*dx)
        adv_term  = velocity(x[j])*(u[j] - u[j-1])/dx  # a>0
=> upwind зліва
        u_new[j] = u[j] + dt*(diff_term) - dt*(adv_term)


    # Оновлення
    u[:] = u_new[:]


    # Перевіряємо, чи зберегти в даний момент часу
    if is_time_to_save(t, times_to_save):
        solutions[int(round(t))] = np.copy(u)


# Після виконання циклу маємо розв'язок у моменти
t=0,1,2,3,4,5.
# Будуємо 5 графіків на одному малюнку:
plt.figure(figsize=(8,6))
for tau in times_to_save:
    plt.plot(x, solutions[tau], label=f"t={tau}")
plt.title("Чисельний розв’язок збуреної задачі Коші:
diffusion + advection")
plt.xlabel("x")
```

```
plt.ylabel("u(t,x)")
plt.grid(True)
plt.legend()
plt.show()

# 1) Означимо штучний точний розв’язок:
def u_exact(t, x):
    return np.exp(-0.1*t)*np.cos(0.5*x)

# Похідні, щоб сформувати "джерело" f(t,x):
def forcing_term(t, x):
    """
    f(t,x) = u_t - u_{xx} - (1/ch(x))*u_x
           = ...
    """
    # ch(x) = cosh(x)
    chx = 0.5*(np.exp(x) + np.exp(-x))  # те саме, що
np.cosh(x)

    # u_t
    ut = -0.1*np.exp(-0.1*t)*np.cos(0.5*x)
    # u_x
    ux = -0.5*np.exp(-0.1*t)*np.sin(0.5*x)
    # u_xx
    uxx = -0.25*np.exp(-0.1*t)*np.cos(0.5*x)

    # f = ut - uxx - (1/chx)*ux
    return ut - uxx - (1.0/chx)*ux
```

```
# ------------------------------------------------
# 2) Налаштування сітки
x_min, x_max = -5.0, 5.0
Nx = 201
dx = (x_max - x_min)/(Nx - 1)
x_grid = np.linspace(x_min, x_max, Nx)

dt = 0.001
Tmax = 2.0
nsteps = int(Tmax/dt)

# ------------------------------------------------
# 3) Початкова умова (точна на t=0)
u = np.array([u_exact(0, xx) for xx in x_grid], dtype=float)
u_new = np.copy(u)

# ------------------------------------------------
# 4) Функції-допоміжники
def ch(x):
    return 0.5*(np.exp(x) + np.exp(-x))  # np.cosh(x)

# Граничні умови Діріхле з точного розв’язку:
def boundary_left(t):
    return u_exact(t, x_min)

def boundary_right(t):
    return u_exact(t, x_max)
```

```
def is_time_to_save(t, times, eps=1e-6):
    return any(abs(t - tau) < eps for tau in times)

# Збережемо рішення у моменти t=0,1,2
times_to_save = [0.0, 1.0, 2.0]
solutions = {}
solutions[0.0] = np.copy(u)
t = 0.0
for n in range(nsteps):
    t += dt

    for j in range(1, Nx-1):
        # Дифузійний член:
        diff_term = (u[j-1] - 2*u[j] + u[j+1]) / (dx*dx)
        # Адвективний член (швидкість a = 1/ch(x)>0 =>
upwind зліва):
        a = 1.0/ch(x_grid[j])
        adv_term = a * (u[j] - u[j-1]) / dx
        # Forcing у середині кроку - візьмемо f(t, x_j):
        f_val = forcing_term(t, x_grid[j])

        # Нова u:
        u_new[j] = u[j] + dt*(diff_term) + dt*(adv_term) +
dt*(f_val)

    # Границя зліва, справа - за точною формулою
    u_new[0] = boundary_left(t)
    u_new[Nx-1] = boundary_right(t)
```

```
    # Оновлюємо
    u[:] = u_new[:]

    # Зберігаємо, якщо час "близький" до цілого
    if is_time_to_save(t, times_to_save):
        solutions[round(t)] = np.copy(u)

# ---------------------------------------------
# 6) Порівняння: беремо збережені моменти часу t=0,1,2
#    і порівнюємо з u_exact(t,x).

for tau in times_to_save:
    # Масив чисельного розв'язку
    u_num = solutions[tau]
    # Масив точного розв'язку
    u_ref = np.array([u_exact(tau, xx) for xx in x_grid],
dtype=float)

    # Обчислимо "максимальну" норму різниці (L∞) для оцінки
похибки
    err = np.max(np.abs(u_num - u_ref))
    print(f"t = {tau},   max|u_num - u_exact| = {err:.6e}")



plt.figure(figsize=(8,6))
for tau in times_to_save:
    plt.plot(x_grid, solutions[tau], label=f"num, t={tau}")
```

```
    # Для наочності можна прокласти точний на тому ж
графіку:
    u_ref = [u_exact(tau, xx) for xx in x_grid]
    plt.plot(x_grid, u_ref, '--', label=f"exact, t={tau}")


plt.xlabel("x")
plt.ylabel("u(t,x)")
plt.title("Перевірка збіжності з «штучним» точним
розв’язком")
plt.grid(True)
plt.legend()
plt.show()
def u_exact(t, x):
    return np.exp(-0.1*t)*np.cos(0.5*x)
def forcing_term(t, x):
    """Додатковий член f(t,x), який робить u_exact справжнім
розв’язком рівняння."""
    chx = 0.5*(np.exp(x) + np.exp(-x))  # np.cosh(x)
    # Похідні від u_exact
    ut = -0.1*np.exp(-0.1*t)*np.cos(0.5*x)
    ux = -0.5*np.exp(-0.1*t)*np.sin(0.5*x)
    uxx = -0.25*np.exp(-0.1*t)*np.cos(0.5*x)


    # f(t,x) = u_t - u_xx - (1/chx)*u_x
    return ut - uxx - (1.0/chx)*ux


def ch(x):
    return 0.5*(np.exp(x) + np.exp(-x))  # np.cosh(x)
```

```
# Параметри сітки
x_min, x_max = -5.0, 5.0
Nx = 201
dx = (x_max - x_min)/(Nx - 1)
x_grid = np.linspace(x_min, x_max, Nx)

# Часові кроки
dt = 0.001
Tmax = 2.0
nsteps = int(Tmax/dt)

# Початкова умова: з точного розв’язку при t=0
u = np.array([u_exact(0.0, xx) for xx in x_grid])
u_new = np.copy(u)

# Які моменти часу зберігати
times_to_save = [0.0, 0.5, 1.0, 1.5, 2.0]
solutions = {t: None for t in times_to_save}
solutions[0.0] = np.copy(u)

# Граничні умови з точного розв’язку
def boundary_left(t):
    return u_exact(t, x_min)

def boundary_right(t):
    return u_exact(t, x_max)

def is_time_to_save(t, times, eps=1e-6):
```

```
    return any(abs(t - tau) < eps for tau in times)

# Основний цикл за часом
t = 0.0
for n in range(nsteps):
    t += dt

    for j in range(1, Nx-1):
        diff_term = (u[j-1] - 2*u[j] + u[j+1])/(dx*dx)
        a = 1.0/ch(x_grid[j])      # швидкість
        adv_term = a*(u[j] - u[j-1])/dx  # upwind, бо a>0
        f_val = forcing_term(t, x_grid[j])

        u_new[j] = u[j] + dt*(diff_term) + dt*(adv_term) +
dt*(f_val)

    # Границі
    u_new[0] = boundary_left(t)
    u_new[-1] = boundary_right(t)

    u[:] = u_new[:]

    # Зберегти рішення, якщо час у списку
    if is_time_to_save(t, times_to_save):
        solutions[round(t,1)] = np.copy(u)

# Будуємо ОКРЕМІ графіки для кожного збереженого часу
for tau in times_to_save:
```

```
    # Масив числового розв’язку
    u_num = solutions[tau]
    # Масив точного розв’язку
    u_ref = [u_exact(tau, xx) for xx in x_grid]

    plt.figure(figsize=(8,5))
    plt.plot(x_grid, u_num, label=f"Чисельний, t={tau}")
    plt.plot(x_grid, u_ref, "--", label=f"Точний, t={tau}")
    plt.xlabel("x")
    plt.ylabel("u(t,x)")
    plt.title(f"Порівняння числового і точного розв’язку,
t={tau}")
    plt.grid(True)
    plt.legend()
    plt.show()

def ch(x):
    return 0.5*(np.exp(x) + np.exp(-x))  # np.cosh(x)

def velocity(x):
    # Наприклад, для рівняння: u_t = u_{xx} + (1/ch(x)) *
u_x
    return 1.0/ch(x)

def initial_condition(x):
    return 1.0/(1.0 + x*x)

# Параметри
x_min, x_max = -5.0, 5.0
```

```
Nx = 201
dx = (x_max - x_min)/(Nx - 1)
x_grid = np.linspace(x_min, x_max, Nx)

dt = 0.0005
Tmax = 5.0
nsteps = int(Tmax/dt)

# Початкове u
u = initial_condition(x_grid)
u_new = np.copy(u)

# Часові точки, які хочемо зберігати
times_to_save = [0, 1, 2, 3, 4, 5]
solutions = {}
solutions[0] = np.copy(u)

def is_time_to_save(t, tlist, eps=1e-5):
    return any(abs(t - tau) < eps for tau in tlist)

t = 0.0
for n in range(nsteps):
    t += dt

    # Явна схема: приклад дифузія + перенос
    for j in range(1, Nx-1):
        # Друга похідна (дифузія)
        diff_term = (u[j-1] - 2*u[j] + u[j+1])/(dx*dx)
```

```
            # Перенос (advection) з a(x) > 0 => upwind зліва
            a = velocity(x_grid[j])
            adv_term = a*(u[j] - u[j-1])/dx

            u_new[j] = u[j] + dt*(diff_term + adv_term)

        u_new[0] = u[0]
        u_new[-1] = u[-1]

        # Оновлення
        u[:] = u_new[:]

for tau in [1, 2, 3, 4, 5]:
        plt.figure(figsize=(7,5))
        plt.plot(x_grid, solutions[tau], label=f"t={tau}")
        plt.xlabel("x")
        plt.ylabel("u(t,x)")
        plt.title(f"Чисельний розв’язок збуреної задачі Коші при
t={tau}")
        plt.grid(True)
        plt.legend()
        plt.show()



        return np.sinh(x)

def p_func(t, a, y):
        """
```

```
    Ядро p(t,a,y) = 1/(2 sqrt(pi t)) * exp( -((a-y)^2)/(4t)
).
    """
    return (1.0/(2.0*np.sqrt(np.pi*t))) * np.exp(-((a -
y)**2)/(4.0*t))


def g_func(t, x):
    """
    g(t,x) = ln( t + sh(x) + sqrt((t + sh(x))^2 + 1) ).
    """
    return np.log(t + sh(x) + np.sqrt((t+sh(x))**2 + 1))


# Початкова "функція" для q_1: f0(y) = 1/(1 + y^2)
def f0(y):
    return 1.0/(1.0 + y*y)


def compute_qn(x_grid, T, n):
    """
    Обчислює q_n(x) на сітці x_grid для заданих T та n.
    T_n = T/n (як у формулі).

    Алгоритм:
      q_1(x) = \int f0(y) * p(T_n, g(T_n, x), y) dy,
      q_2(x) = \int q_1(y) * p(...) dy, і т.д.
    Повертає масив q_n для кожного x_grid.
    """
    Tn = T / n  # Крок "за часом" у інтегралі

    # q_j(x) зберігатимемо в масиві shape=(Nx,)
```

```
    # spочатку q_{0}(y) = f0(y)
    def q_prev(y):  # поточна "функція" q_{j-1}
        return f0(y)

    # Кожна ітерація побудує новий q_j(x)
    for j in range(1, n+1):
        q_curr_values = np.zeros_like(x_grid)

        # Для кожного x з x_grid рахуємо інтеграл по y
        for i, x_val in enumerate(x_grid):
            a_val = g_func(Tn, x_val)
            def integrand(y):
                return q_prev(y) * p_func(Tn, a_val, y)

            # quad(...) обчислює \int_{-∞}^{∞} integrand(y)
dy
            val, _ = quad(integrand, -np.inf, np.inf)
            q_curr_values[i] = val

        # Після того, як ми отримали q_j,
        # оголошуємо q_prev(y) = інтерполяція q_curr(x_grid)
на y
        # (щоб наступна ітерація також могла рахувати
інтеграл).
        x_for_interp = np.copy(x_grid)
        y_for_interp = np.copy(q_curr_values)

        # q_prev стане лямбдою, яка інтерполює в нових
точках:
```

```
        def q_prev_func(z):
            # звичайна лінійна інтерполяція:
            return np.interp(z, x_for_interp, y_for_interp)

        q_prev = q_prev_func  # тепер q_prev - функція, яку буде видно в наступній ітерації

    # Після j=n, q_curr_values - це q_n(x)
    return q_curr_values

def ch(x):
    """Гіперболічний косинус (для зручності)."""
    return 0.5*(np.exp(x) + np.exp(-x))  # аналог np.cosh(x)

def velocity(x):
    """Швидкість a(x) = 1/ch(x), додатна."""
    return 1.0/ch(x)

def initial_condition(x):
    """u(0,x) = 1/(1 + x^2)."""
    return 1.0/(1.0 + x*x)

# Параметри сітки (більший проміжок, згущена сітка)
x_min, x_max = -10.0, 10.0
Nx = 401
x_grid = np.linspace(x_min, x_max, Nx)
dx = (x_max - x_min)/(Nx - 1)
```

```
# Часові параметри (менший крок часу)
dt = 0.0002
T = 2.0  # можна змінити на 4.0 чи 5.0
nsteps = int(T / dt)

# Ініціалізація масивів
u = initial_condition(x_grid)
u_new = np.copy(u)

# Явна схема: u_t = u_xx + (1/ch(x))*u_x
# Для стабільності: dt мусить бути досить малим
for n in range(nsteps):
    for j in range(1, Nx-1):
        # дифузійний член
        diff_term = (u[j-1] - 2*u[j] + u[j+1]) / (dx*dx)
        # адvекція (a>0 => upwind)
        a_j = velocity(x_grid[j])
        adv_term = a_j*(u[j] - u[j-1])/dx

        u_new[j] = u[j] + dt*(diff_term + adv_term)

    # Прості граничні умови (копіюємо краї):
    u_new[0]   = u[0]
    u_new[-1]  = u[-1]

    # Оновлення
    u[:] = u_new[:]
```

```
u_pde = np.copy(u)



def p_func(t, a, y):
    """Ядро p(t,a,y) = 1/(2 sqrt(pi t)) * exp( -((a-
y)^2)/(4t) )."""
    return (1.0/(2.0*np.sqrt(np.pi*t))) * np.exp( -((a -
y)**2)/(4.0*t) )


def g_func(t, x):
    """g(t,x) = ln( t + sinh(x) + sqrt((t + sinh(x))^2 + 1 )
)."""
    return np.log( t + np.sinh(x) + np.sqrt((t +
np.sinh(x))**2 + 1) )


def f0(y):
    """Початкова функція 1/(1 + y^2)."""
    return 1.0/(1.0 + y*y)


def compute_qn(x_arr, T, n):
    """
    Обчислює q_n(x) на сітці x_arr для T; T_n = T/n.
    Кожна ітерація:
       q_j(x) = int q_{j-1}(y) * p_func(T_n, g_func(T_n,x),
y) dy
    Використовує кубічну інтерполяцію між кроками.
    """
    Tn = T/n

    # Починаємо з q0(y) = f0(y).
```

```
    # "поточна функція" q_prev(y). Спочатку це f0.
    def q_prev_function(z):
        return f0(z)

    for j in range(1, n+1):
        q_curr_vals = np.zeros_like(x_arr)

        # Для кожного x обчислюємо інтеграл \int_{-∞}^∞
q_prev(y)*p(...) dy
        for i, xval in enumerate(x_arr):
            a_val = g_func(Tn, xval)

            def integrand(y):
                return q_prev_function(y)*p_func(Tn, a_val,
y)

            val, _ = quad(integrand, -np.inf, np.inf)
            q_curr_vals[i] = val

        # Тепер q_curr_vals = q_j(x) на сітці x_arr.
        # Побудуємо кубічну інтерполяцію, аби q_{j} стала
функцією q_prev_function
        # для наступної ітерації.
        interp_cubic = interp1d(
            x_arr, q_curr_vals, kind='cubic',
fill_value='extrapolate'
        )

        # Оголошуємо q_prev_function = ця кубічна
інтерполяція:
```

```
        q_prev_function = lambda z: interp_cubic(z)

    return q_curr_vals

# ПАРАМЕТРИ інтегральної методики
n_iter = 20
x_integral = np.linspace(-10, 10, 201)

# Обчислюємо q_n(x), n_iter = 20
q_n_vals = compute_qn(x_integral, T, n_iter)

pde_interp_cubic = interp1d(x_grid, u_pde, kind='cubic',
fill_value='extrapolate')
u_pde_on_integral = pde_interp_cubic(x_integral)

# Тепер малюємо обидва графіки
plt.figure(figsize=(8,5))
plt.plot(x_integral, q_n_vals, label=f"q_{n_iter}(x)
(інтегральний метод)")
plt.plot(x_integral, u_pde_on_integral, '--',
label=f"u_PDE(T={T}, x) (чисельний метод)")
plt.xlabel("x")
plt.ylabel("Функція")
plt.title(f"Покращене порівняння q_{n_iter}(x) та
u(T={T},x)")
plt.grid(True)
plt.legend()
plt.show()
    return 0.5*(np.exp(x) + np.exp(-x))  # аналог np.cosh(x)
```

```
def velocity(x):
    """Швидкість a(x) = 1/ch(x), додатна."""
    return 1.0/ch(x)

def initial_condition(x):
    """u(0,x) = 1/(1 + x^2)."""
    return 1.0/(1.0 + x*x)

# Параметри сітки (збільшили проміжок і згущуємо сітку)
x_min, x_max = -10.0, 10.0
Nx = 401
x_grid = np.linspace(x_min, x_max, Nx)
dx = (x_max - x_min)/(Nx - 1)

# Часові параметри (менший крок часу для кращої точності)
dt = 0.0002
T = 2.0  # можна обрати T=2 (або інше, за необхідності)
nsteps = int(T / dt)

# Ініціалізація масивів
u = initial_condition(x_grid)
u_new = np.copy(u)

# Явна схема: u_t = u_xx + (1/ch(x))*u_x
for n in range(nsteps):
    for j in range(1, Nx-1):
        # дифузійний член
```

```
        diff_term = (u[j-1] - 2*u[j] + u[j+1]) / (dx*dx)
        # адvекція (a>0 => upwind)
        a_j = velocity(x_grid[j])
        adv_term = a_j*(u[j] - u[j-1])/dx

        u_new[j] = u[j] + dt*(diff_term + adv_term)

    # Прості граничні умови (копіюємо краї)
    u_new[0]   = u[0]
    u_new[-1]  = u[-1]

    u[:] = u_new[:]

# Тепер масив u містить чисельний розв'язок PDE при T
u_pde = np.copy(u)

def p_func(t, a, y):
    """Ядро p(t,a,y) = 1/(2 sqrt(pi t)) * exp( -((a-
y)^2)/(4t) )."""
    return (1.0/(2.0*np.sqrt(np.pi*t))) * np.exp(-((a -
y)**2)/(4.0*t))

def g_func(t, x):
    """g(t,x) = ln( t + sinh(x) + sqrt((t + sinh(x))^2 + 1 )
)."""
    return np.log(t + np.sinh(x) + np.sqrt((t +
np.sinh(x))**2 + 1))

def f0(y):
```

```
    """Початкова функція 1/(1 + y^2)."""
    return 1.0/(1.0 + y*y)

def compute_qn(x_arr, T, n):
    """
    Обчислює q_n(x) на сітці x_arr для T; T_n = T/n.
    Кожна ітерація:
       q_j(x) = int q_{j-1}(y) * p_func(T_n, g_func(T_n,x),
y) dy
    Використовує кубічну інтерполяцію між кроками.
    """
    Tn = T/n

    # Починаємо з q0(y) = f0(y).
    def q_prev_function(z):
        return f0(z)

    for j in range(1, n+1):
        q_curr_vals = np.zeros_like(x_arr)

        # Для кожного x із x_arr
        for i, xval in enumerate(x_arr):
            a_val = g_func(Tn, xval)

            def integrand(y):
                return q_prev_function(y)*p_func(Tn, a_val,
y)

            val, _ = quad(integrand, -np.inf, np.inf)
```

```
            q_curr_vals[i] = val

        # Кубічна інтерполяція q_j
        interp_cubic = interp1d(
            x_arr, q_curr_vals, kind='cubic',
fill_value='extrapolate'
        )

        # Для наступної ітерації q_{j} стає q_{j-1}
        q_prev_function = lambda z: interp_cubic(z)

    return q_curr_vals

# Сітка для інтегрального підходу
x_integral = np.linspace(-10, 10, 201)

# Інтерполяція PDE-результату на x_integral
pde_interp_cubic = interp1d(x_grid, u_pde, kind='cubic',
fill_value='extrapolate')
u_pde_on_integral = pde_interp_cubic(x_integral)



for n_iter in [5, 10]:
    q_n_vals = compute_qn(x_integral, T, n_iter)

    # Будуємо графік порівняння
    plt.figure(figsize=(8,5))
    plt.plot(x_integral, q_n_vals, label=f"q_{n_iter}(x)
(інтегральний, n={n_iter})")
```

```
    plt.plot(x_integral, u_pde_on_integral, '--',
label=f"u_PDE(T={T}) (чисельний)")
    plt.xlabel("x")
    plt.ylabel("Функція")
    plt.title(f"Порівняння q_{n_iter}(x) vs. u_PDE(T={T},
x)")
    plt.grid(True)
    plt.legend()
    plt.show()



def solve_pde(T):
    """
    Рахуємо u(t,x) при кінцевому часі T,
    для рівняння: u_t = u_xx + (1/ch(x))*u_x,
    поч. умова: u(0,x) = 1/(1+x^2).

    Повертає (x_grid, u_array), де u_array -
    розв'язок у момент T на сітці x_grid.
    """

    # Налаштування сітки x
    x_min, x_max = -10.0, 10.0
    Nx = 401
    x_grid = np.linspace(x_min, x_max, Nx)
    dx = (x_max - x_min)/(Nx - 1)

    # Налаштування часу
```

```
    dt = 0.0002   # достатньо малий крок, щоб виконати умови
стійкості
    nsteps = int(T/dt)

    # Ініціалізація
    u = initial_condition(x_grid)
    u_new = np.copy(u)

    # Цикл за часом
    for _ in range(nsteps):
        for j in range(1, Nx-1):
            # Дифузійний член
            diff_term = (u[j-1] - 2*u[j] + u[j+1])/(dx*dx)
            # Адвекція
            a_j = velocity(x_grid[j])  # = 1/ch(x)
            # upwind (a_j>0 => різниця ліворуч)
            adv_term = a_j*(u[j] - u[j-1])/dx
            u_new[j] = u[j] + dt*(diff_term + adv_term)

        # Границі (проста "копія")
        u_new[0]  = u[0]
        u_new[-1] = u[-1]

        u[:] = u_new[:]

    return x_grid, u
```

```
def compute_qn(x_array, T, n):
    """
    Обчислює q_n(x) на сітці x_array для заданих T та n.
    Кожна ітерація:
      q_j(x) = інтеграл ( q_{j-1}(y) * p(T/n, g(T/n,x), y )
) dy
    Початкова q_0(y) = 1/(1+y^2).
    """

    Tn = T/n  # "швидкість" інтеграції
    # Початкова функція q_0
    def q_prev_func(z):
        return 1.0/(1.0 + z*z)  # f0(z)

    for j in range(1, n+1):
        q_curr_vals = np.zeros_like(x_array)

        for i, xval in enumerate(x_array):
            a_val = g_func(Tn, xval)

            def integrand(y):
                return q_prev_func(y) * p_func(Tn, a_val, y)

            val, _ = quad(integrand, -np.inf, np.inf)
            q_curr_vals[i] = val

        # Кубічна інтерполяція q_j
        interp_cubic = interp1d(
```

```
            x_array, q_curr_vals, kind='cubic',
fill_value='extrapolate'
        )

        # Тепер q_{j} буде початковим для наступної ітерації
        q_prev_func = lambda z: interp_cubic(z)

    return q_curr_vals

def ch(x):
    return 0.5*(np.exp(x) + np.exp(-x))

def velocity(x):
    return 1.0/ch(x)

def initial_condition(x):
    return 1.0/(1.0 + x*x)

def solve_pde(T):
    """
    Рахує PDE: u_t = u_xx + (1/ch(x))*u_x,
    повертає (x_grid, u_array) для моменту часу T.
    """
    x_min, x_max = -10.0, 10.0
    Nx = 401
    x_grid = np.linspace(x_min, x_max, Nx)
    dx = (x_max - x_min)/(Nx - 1)
```

```
    dt = 0.0002
    nsteps = int(T/dt)

    u = initial_condition(x_grid)
    u_new = np.copy(u)

    for _ in range(nsteps):
        for j in range(1, Nx-1):
            diff_term = (u[j-1] - 2*u[j] + u[j+1])/(dx*dx)
            a_j = velocity(x_grid[j])
            adv_term = a_j*(u[j] - u[j-1])/dx
            u_new[j] = u[j] + dt*(diff_term + adv_term)
        # Просте відображення граничних умов
        u_new[0]  = u[0]
        u_new[-1] = u[-1]
        u[:] = u_new[:]

    return x_grid, u

# Ядро та g-функція для інтегральної побудови q_n
def p_func(t, a, y):
    return (1.0/(2.0*np.sqrt(np.pi*t))) * np.exp(-((a -
y)**2)/(4.0*t))

def g_func(t, x):
    return np.log(t + np.sinh(x) + np.sqrt((t +
np.sinh(x))**2 + 1))

def compute_qn(x_array, T, n):
```

```
    """
    Будує q_n(x), n ітерацій згортки з p_func(...),
    q_0(y) = 1/(1+y^2).
    """
    Tn = T/n
    # Початкова функція q_0
    def q_prev(z):
        return 1.0/(1.0 + z*z)

    for j in range(1, n+1):
        q_curr = np.zeros_like(x_array)
        for i, xval in enumerate(x_array):
            a_val = g_func(Tn, xval)
            def integrand(y):
                return q_prev(y)*p_func(Tn, a_val, y)
            val, _ = quad(integrand, -np.inf, np.inf)
            q_curr[i] = val

        # Інтерполяція для наступної ітерації
        f_interp = interp1d(x_array, q_curr, kind='cubic',
fill_value='extrapolate')
        q_prev = lambda z: f_interp(z)

    return q_curr

# 2.1. Отримуємо PDE-рішення при T=2 і T=4
x_pde_2, u_pde_2 = solve_pde(T=2.0)   # u1(2,x)
x_pde_4, u_pde_4 = solve_pde(T=4.0)   # u1(4,x)
```

```
# Інтерполяція PDE для зручності
interp_pde_2 = interp1d(x_pde_2, u_pde_2, kind='cubic',
fill_value='extrapolate')
interp_pde_4 = interp1d(x_pde_4, u_pde_4, kind='cubic',
fill_value='extrapolate')


# 2.2. Спільна сітка для порівняння
x_common = np.linspace(-10, 10, 201)


# 2.3. Обчислюємо q_n(T,x) для T=2, n=5 і n=10, та для T=4,
n=5 і n=10
q_5_T2  = compute_qn(x_common, 2.0, 5)   # u2(2,x), n=5
q_10_T2 = compute_qn(x_common, 2.0, 10)  # u2(2,x), n=10
q_5_T4  = compute_qn(x_common, 4.0, 5)   # u2(4,x), n=5
q_10_T4 = compute_qn(x_common, 4.0, 10)  # u2(4,x), n=10


u1_2_on_common = interp_pde_2(x_common)  # u1(2,x)
u1_4_on_common = interp_pde_4(x_common)  # u1(4,x)


# 2.4. Формули похибок:
# δ1 = max_x |u1(2,x)-u2(2,x)| / |u1(2,x)|  (n=5)
# δ2 = max_x |u1(2,x)-u2(2,x)| / |u1(2,x)|  (n=10)
# δ3 = max_x |u1(4,x)-u2(4,x)| / |u1(4,x)|  (n=5)
# δ4 = max_x |u1(4,x)-u2(4,x)| / |u1(4,x)|  (n=10)


# - δ1
abs_err_1 = np.abs(u1_2_on_common - q_5_T2) /
np.abs(u1_2_on_common + 1e-15)
```

```
delta_1 = np.max(abs_err_1)


# - δ2
abs_err_2 = np.abs(u1_2_on_common - q_10_T2) /
np.abs(u1_2_on_common + 1e-15)
delta_2 = np.max(abs_err_2)


# - δ3
abs_err_3 = np.abs(u1_4_on_common - q_5_T4) /
np.abs(u1_4_on_common + 1e-15)
delta_3 = np.max(abs_err_3)


# - δ4
abs_err_4 = np.abs(u1_4_on_common - q_10_T4) /
np.abs(u1_4_on_common + 1e-15)
delta_4 = np.max(abs_err_4)


# 2.5. Виводимо результати:
print(f"delta_1 (T=2, n=5):  {delta_1:.6e}")
print(f"delta_2 (T=2, n=10): {delta_2:.6e}")
print(f"delta_3 (T=4, n=5):  {delta_3:.6e}")
print(f"delta_4 (T=4, n=10): {delta_4:.6e}")
```