**Міністерство освіти і науки України**

**Національний технічний університет України**

**“Київський політехнічний інститут ІМ. І. СІКОРСЬКОГО”**

ННК «ІПСА» НТУУ “Київський Політехнічний Інститут”

(назва факультету, інституту)

Системного проектування

(назва кафедри)

**Пояснювальна записка**

до курсової роботи на тему: “Паралельне розв’язання диференційного рівняння у частинних похідних за допомогою методів кінцевих різниць ”

**Студент групи** ДА-61 Кравченко Богдан Євгенович \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр групи) (прізвище, ім’я, по батькові) (підпис)

**Керівник проекту** *ас. Яременко В. С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(вчені ступінь та звання, прізвище, ініціали) (підпис)

|  |  |
| --- | --- |
| **Керівник:** | **Виконавець:** |
| ***Яременко В. С.*** | **ст . Кравченко Богдан**  **гр. ДА-61** |
| **Допущений до захисту** | **Зал. книжка № ДА-6117** |
| **Захищено із оцінкою** |  |

Київ – 2019

Форма № У-6.01

Затв. наказом УРСР

від 3 серпня 1984р. № 253

ННК «ІПСА» НТУУ “Київський Політехнічний Інститут ім. І. Сікорського

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва вищого навчального закладу)

Системного проектування

Кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Технології розподілених систем і паралельних обчислень

Дисципліна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6.050101

Спеціальність\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6

3

ДА - 61

Курс\_\_\_\_\_\_\_\_\_\_Група\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Семестр\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ЗАВДАННЯ**

**на курсову роботу студента**

Кравченко Богдан Євгенович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

1. Тема курсової роботи

Паралельне розв’язання диференційного рівняння у частинних похідних за допомогою методів кінцевих різниць \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Строк здачі студентом закінченого проекту (роботи)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кравченко Богдан Євгенович

(підпис)

(підпис)

Керівник\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Яременко В. С.*

« \_\_ » \_ \_ р.

**Зміст**

[**Завдання** 5](#_Toc6654615)

[**Теоретичні відомості** 7](#_Toc6654616)

[**Багатопотоковість** 7](#_Toc6654617)

[**Метод Сітки** 11](#_Toc6654618)

[Явний метод 12](#_Toc6654619)

[**Розв’язання** 13](#_Toc6654620)

[**Результати** 14](#_Toc6654621)

[**Модулі програми** 16](#_Toc6654622)

[**Висновки** 17](#_Toc6654623)

[**Додаток. Лістинг програми** 19](#_Toc6654624)

**Завдання**

**Варіант 11. Задача №11**

Знайти розв’язок рівняння , де -деякі константи. Дане рівняння використовується в задачах тепло- і масопереносу, теорії горіння, біології та екології та має назву рівняння Колмогорова-Петровського-Піскунова.

Точний розв’язок:  
, де - деякі константи.

**Теоретичні відомості**

Багатопотоковість

Багатопотоковість — властивість операційної системи або застосунку, яка полягає в тому, що процес, породжений в операційній системі, може складатися з кількох нитей, що виконуються паралельно, або навіть одночасно на багатопроцесорних системах. При виконанні деяких завдань таке розділення може досягти ефективнішого використання ресурсів комп'ютера. Такі нитки виконання ще називають потоками.

Суттю багатонитковості є квазі-багатозадачність на рівні одного виконуваного процесу, тобто всі нитки виконуються в адресному просторі процесу. Окрім цього, всі нитки процесу мають не тільки спільний адресний простір, але і спільні дескриптори файлів. Процес, що виконується, має як мінімум одну (головну) нитку.

Багатонитковість (як доктрину програмування) не слід плутати ані з багатозадачністю, ані з багато-процесорністю, не зважаючи на те, що операційні системи, що реалізовують багатозадачність, як правило, реалізують і багатонитковість.

Переваги в багатонитковості такі:

• Спрощення програми в деяких випадках, за рахунок використання загального адресного простору

• Менші відносно процесу часові витрати на створення ниті і взаємодію між нитями

• Підвищення продуктивності процесу за рахунок розпаралелювання процесорних обчислень і операцій вводу/виводу

Розпарале́лювання програ́м — процес адаптації алгоритмів, записаних у вигляді програм, для їх ефективного виконання на обчислювальній системі паралельної архітектури (останнім часом, як правило, на багатопроцесорній обчислювальній системі). Полягає або в переписуванні програм на спеціальну мову, яка описує паралелізм і яку розуміють транслятори цільової обчислювальної системи, або до вставки спеціальної розмітки (наприклад, інструкцій MPI або OpenMP).

Розпаралелювання може бути ручним, автоматизованим або напівавтоматизованим.

При розпаралелюванні важливо брати до уваги не лише формальний паралелізм структури алгоритму, але й те, що обмінні операції в паралельних комп'ютерах є, як правило, набагато повільнішими від арифметичних. З цим пов'язане існування левової частки накладних витрат на організацію паралелізму.

OpenMP(Open Multi-Processing) — це набір директив компілятора, бібліотечних процедур та змінних середовища, які призначені для програмування багатопоточних застосунків на багатопроцесорних системах із спільною пам'яттю.

OpenMP можна розглядати як високорівневу надбудову над Pthreads (або аналогічними бібліотеками потоків). POSIX-інтерфейс для організації потоків Pthreads підтримується широко (практично на всіх UNIX-системах).

OpenMP реалізує паралельні обчислення за допомогою багатопотоковості, в якій «головний» (master) потік створює набір підлеглих (slave) потоків і завдання розподіляється між ними. Передбачається, що потоки виконуються паралельно на машині з декількома процесорами (кількість процесорів не обов'язково має бути більше або дорівнювати кількості потоків).

Завдання, що виконуються потоками паралельно, так само як і дані, необхідні для виконання цих завдань, описуються за допомогою спеціальних директив препроцесора відповідної мови — прагм.

Кількість створюваних потоків може регулюватися як самою програмою за допомогою виклику бібліотечних процедур, так і ззовні, за допомогою змінних оточення.

Переваги

За рахунок ідеї «інкрементального розпаралелювання» OpenMP ідеально підходить для розробників, що прагнуть швидко розпаралелювати свої обчислювальні програми з великими паралельними циклами. Розробник не створює нову паралельну програму, а просто послідовно додає в текст програми OpenMP-директиви.

При цьому, OpenMP — досить гнучкий механізм, що надає розробникові великі можливості контролю над поведінкою паралельної програми.

Передбачається, що OpenMP-програма на однопроцесорній платформі може бути використана як послідовна програма, тобто немає необхідності підтримувати послідовну та паралельну версії. Директиви OpenMP просто ігноруються послідовним компілятором, а для виклику процедур OpenMP можуть бути підставлені заглушки (stubs), текст яких приведений в специфікаціях.

Однією з переваг OpenMP розробники вважають підтримку так званих «orphan» (відірваних) директив, тобто директиви синхронізації і розподілу роботи можуть не входити безпосередньо в лексичний контекст паралельної області.

Крім послідовних потоків Stream API підтримує паралельні потоки. Розпаралелювання потоків дозволяє задіяти кілька ядер процесора (якщо цільова машина многоядерная) і тим самим може підвищити продуктивність і прискорити обчислення. У той же час говорити, що застосування паралельних потоків на багатоядерних машинах однозначно підвищить продуктивність - не зовсім коректно. У кожному конкретному випадку треба перевіряти і тестувати.

Щоб зробити звичайний послідовний потік паралельним, треба викликати в об'єкта Stream метод parallel . Крім того, можна також використовувати метод parallelStream () інтерфейсу Collection для створення паралельного потоку з колекції.

У той же час якщо робоча машина не є многоядерной, то потік буде виконуватися як послідовний.

Застосування паралельних потоків у багатьох випадках буде аналогічно.

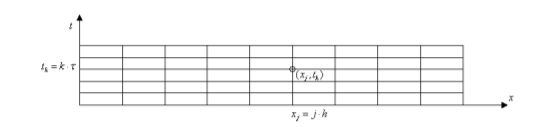
Питання продуктивності в паралельних операціях

Фактично застосування паралельних потоків зводиться до того, що дані в потоці будуть розділені на частини, кожна частина обробляється на окремому ядрі процесора, і в кінці ці частини з'єднуються, і над ними виконуються фінальні операції. Розглянемо деякі критерії, які можуть вплинути на продуктивність в паралельних потоках:

* Розмір даних. Чим більше даних, тим складніше спочатку розділяти дані, а потім їх з'єднувати.
* Кількість ядер процесора. Теоретично, чим більше ядер в комп'ютері, тим швидше програма буде працювати. Якщо на машині одне ядро, немає сенсу застосовувати паралельні потоки.
* Чим простіше структура даних, з якою працює потік, тим швидше будуть відбуватися операції. Наприклад, дані з ArrayList легко використовувати, так як структура даної колекції передбачає послідовність незв'язаних даних. А ось колекція типу LinkedList - не найкращий варіант, так як в послідовному списку всі елементи пов'язані з попередніми / подальшими. І такі дані важко распараллелить.
* Над даними примітивних типів операції будуть проводитися швидше, ніж над об'єктами класів

Метод Сітки

Поняття сітки і гратчастої функції виникають в теорії різницевих схем в зв'язку з вивченням чисельних методів розв'язання крайових задач математичної фізики. Для побудови чисельного методу необхідно написати разностную схему, наближено описує диференціальне рівняння (або систему). Цей етап пов'язаний з заміною області безперервного зміни аргументу областю дискретного його зміни і з заміною диференціального оператора деяким його різницевим аналогом, крім того, записується разностная апроксимація для початкових і граничних умов. Результатом цієї процедури є алгебраїчна система рівнянь або система різницевих рівнянь . Чисельне рішення крайової задачі для вихідного лінійного диференціального рівняння зводиться до вирішення отриманої системи алгебри. Відзначимо, що чисельні методи дають обмежену і наближену інформацію про рішення, але зате є універсальними.



Розглядається деякий безліч функцій y (x, t) двох змінних x t,. Нехай область визначення цих функцій є, наприклад, прямокутник.



Розбиваємо відрізки [0,1] і [0, T ] осі Ox і Ot відповідно на N і M рівних частин, причому h і t.

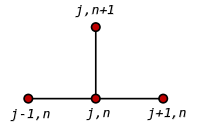
Через точки поділу проводяться прямі, паралельні осях Ox і Ot відповідно. Точки перетину цих прямих утворюють вузли (x, t) сітки. 

Сусідніми вузлами отриманої сітки називають вузли, розташовані на одній і тій же прямій (горизонтальної та вертикальної), відстань між якими дорівнює кроку сітки (h або t).

### Явний метод

Використовуючи праву різницю **по часу** ** t_n ** та центральну різницю другого порядку по простору в точці  x_j  ([FTCS](http://en.wikipedia.org/wiki/FTCS_scheme)) отримаємо рекурентне рівняння:

 \frac{u_{j}^{n+1} - u_{j}^{n}}{k} = \frac{u_{j+1}^n - 2u_{j}^n + u_{j-1}^n}{h^2}. \, 



*Шаблон найбільш поширеного явного*

*методу для рівняння теплопровідності*

Це явний метод розв’язку одновимірного рівняння теплопровідності.

Отримати значення  u_j^{n+1}  можна так:

 u_{j}^{n+1} = (1-2r)u_{j}^{n} + ru_{j-1}^{n} + ru_{j+1}^{n}  

де  r=k/h^2. 

Таким чином, маючи це рекурентне співвідношення та знаючи значення в момент часу *n*, можна отримати відповідні значення в момент часу *n*+1.  u_0^n  і  u_J^n  треба замінити граничними умова, які в даному прикладі дорівнюють нулю.

Відомо, що явний метод чисельно стійкий і збіжний при  r\le 1/2 . Похибка пропорційна часовому кроку та квадрату просторового кроку:

 \Delta u = O(k)+O(h^2)  \, 

Розв’язання

Оберемо такі коефіцієнти:

a = 5, b = -5, C = 5,  λ=1.

Рівняння набуде такого вигляду:

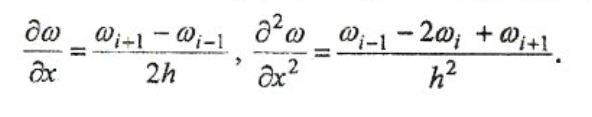
,

Точний розв’язок:

Початкова умова:

Граничні умови:

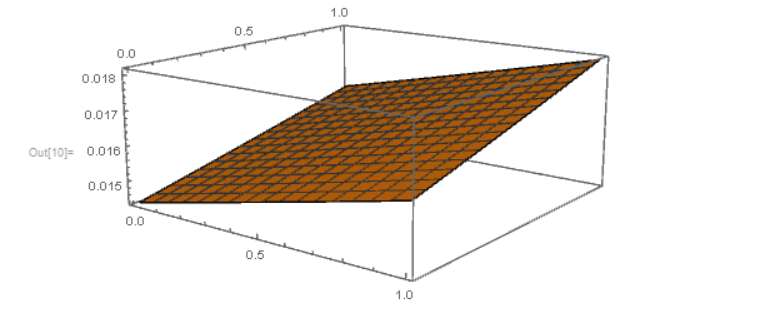
**Явний метод.** Замінимо похідні:



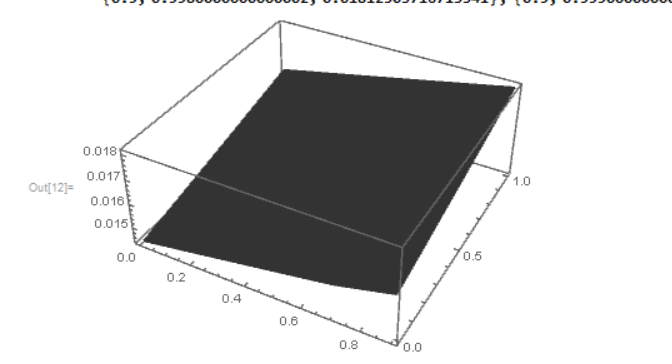
Спростимо вираз:

**Результати**

Візуалізація точного розв’язку:



Візуалізація обчисленого розв’язку:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | X = 10,T=1000 | X = 20,T=1000 | X = 30,T=1000 | X = 1000,T=1000 |
| Послідовно | 5705800 | 6522900 | 7675400 | 167626000 |
| OpenMP | 131299500 | 141890800 | 210994000 | 236170100 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Х=10 | Х=20 | Х=30 | Х=1000 |
| Абсолютна | 1.9918231E-4 | 9.477695883686E-5 | 6.259768982700E-5 | 1.865096717328 |
| Відносна | 1.098813509819 | 0.5686560953672897 | 0.3815848736629564 | 0.01151509164216 |

**Модулі програми**

static int wX - кількість точок або клітинок по осі Ох.

static int wT - кількість точок або клітинок по осі Оt.

static double a,b,C,lam - деякі константи обрані навмання.

static double h – крок у напрямку Ох

static double tau – крок у напрямку Оt.

static double[][] W - двовимірний масив точок отриманих з рішення рівняння

static double func(double x, double t) – функція точного розв’язку, яка дає точку по координатам - x та t .

static double nextIteration(int i,int k) – наступна точка отримана в результаті розв’язання рівняння явним методом.

double Absolute – абсолютна похибка.

double Relative – відносна похибка.

**Висновки**

У ході даної курсової роботи було розв’язано диференційне рівняння у частинних похідних за допомогою неявного методув кінцевих різниць. Паралельна реалізація виконана за допомогою технологій OpenMP і MPI.

У роботі наведено результати роботи розробленої програми, виконано візуалізацію обчисленого та точного розв’язків. Також наведено час роботи програми та коефіцієнт прискорення.

Явний метод, звичайно, є найпростішим для реалізації і достатньо точним, але умова стійкості ускладнює його практичне застосування. Неявний метод має такий самий порядок точності, що й явний, проте є безумовно стійким.

**СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ**

1. Турчак Л.И. Основы численных методов. - М.: Наука, 1987. - 320 с.
2. Петренко А.І. Обчислювальна математика. Суми.: ВМУРоЛ “Україна”,
3. Лекції по технології паралельного обчислення і росподільчих систем КПІ Яременко В. С.

Посилання на GitHub: https://github.com/Bodr1kk/course-work

**Додаток. Лістинг програми**

**Class Main.java**

import java.io.FileWriter;

import java.io.IOException;

import java.util.stream.IntStream;

public class Main {

static int wX = 10;

static int wT = 1000;

static double a = 5;

static double b = -5;

static double С = 5;

static double lam = 1;

static double h = (double) 1/wX;

static double tau = (double) Math.pow(h,2)\*(0.1);

static double[][] W = new double[wX][wT];

static double func(double x, double t){

return Math.pow((С\*Math.exp((-lam/(2\*a))\*(x+lam\*t)))-(2\*b)/(3\*lam),-2);

}

static double nextIteration(int i,int k){

return W[i][k]+tau\*(5\*((W[i-1][k]-2\*W[i][k]+W[i+1][k])/(h\*h))+5\*Math.pow(W[i][k],0.5)\*((W[i+1][k]-W[i-1][k])/(2\*h)));

}

public static void main(String[] args) throws IOException {

long startSerialTime = System.nanoTime();

for(int i=0;i<W.length;i++){

W[i][0] = func(i\*h,0);

}

for(int i=0;i<W[0].length;i++){

W[0][i] = func(0,i\*tau);

W[W.length-1][i] = func(1,i\*tau);

}

for (int k = 0; k < W[0].length-1; k++){

for (int i = 1; i < W.length-1; i++){

W[i][k+1] = nextIteration(i,k);

}

}

long endSerialTime = System.nanoTime();

long timeSerialSpent = endSerialTime - startSerialTime;

FileWriter FileSerial = new FileWriter("serial.txt");

double Absolute = 0;

double Relative = 0;

FileSerial.write("ListPlot3D[{"+"{"+(0.0)+","+(0.0)+","+W[0][0]+"}");

for(int i=0;i<W.length;i++){

for(int j=1;j<W[0].length;j++){

// file.write(String.format("%.3f",W[i][j])+"("+i+" "+j+"), ");

FileSerial.write(",{"+(i\*h)+","+(j\*tau)+","+W[i][j]+"}");

if(Absolute < Math.abs(W[i][j]-func(i\*h,j\*tau))){

Absolute = Math.abs(W[i][j]-func(i\*h,j\*tau));

Relative = Absolute/W[i][j]\*100;

}

}

}

FileSerial.write("}, Mesh -> All]");

FileSerial.close();

long startTime = System.nanoTime();

IntStream.range(0,W.length).parallel().forEach(i-> W[i][0] = func(i\*h,0));

IntStream.range(0,W[0].length).parallel().forEach(i->{

W[0][i] = func(0,i\*tau);

W[W.length-1][i] = func(1,i\*tau);

});

for (int k = 0; k < W[0].length-1; k++) {

int finalK = k;

IntStream.range(1, W.length - 1).parallel().forEach((i) -> W[i][finalK + 1] = nextIteration(i, finalK));

}

long endTime = System.nanoTime();

long totalTime = endTime - startTime;

FileWriter FileParallel = new FileWriter("parralel.txt");

double AbsoluteParallel = 0;

double RelativeParallel = 0;

FileParallel.write("ListPlot3D[{"+"{"+(0.0)+","+(0.0)+","+W[0][0]+"}");

for(int i=0;i<W.length;i++){

for(int j=1;j<W[0].length;j++){

FileParallel.write(",{"+(i\*h)+","+(j\*tau)+","+W[i][j]+"}");

if(AbsoluteParallel < Math.abs(W[i][j]-func(i\*h,j\*tau))){

AbsoluteParallel = Math.abs(W[i][j]-func(i\*h,j\*tau));

RelativeParallel = AbsoluteParallel/W[i][j]\*100;

}

}

}

FileParallel.write("}, Mesh -> All]");

FileParallel.close();

System.out.println("При послідовному вирішенні програма виконувалась " + timeSerialSpent + " наносекунд");

System.out.println("При паралельному вирішенні програма виконувалась " + totalTime + " наносекунд");

System.out.println("Абсолютна похибка = "+ Absolute);

System.out.println("Відносна = "+ Relative);

}

}