**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 ”

**Студентка групи** ДА-62 Петренко Марія Миколаївна \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(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 . Петренко Марія**  **гр. ДА-62** |
| **Допущений до захисту** | **Зал. книжка № ДА-6222** |
| **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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Паралельні обчислення

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

6.050101

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

ДА - 62

6

3

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

**ЗАВДАННЯ**

**на курсову роботу студента**

Петренко Марія Миколаївна

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

(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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Петренко Марія Миколаївна

(підпис)

(підпис)

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

« \_\_ » \_ \_ р.

Зміст

[Завдання 6](#_Toc9101146)

[Теоретичні відомості 7](#_Toc9101147)

[Поняття паралелелізму 7](#_Toc9101148)

[Метод Сітки 8](#_Toc9101149)

[Явний метод 9](#_Toc9101150)

[Розв’язання 10](#_Toc9101151)

[Результати 11](#_Toc9101152)

[Модулі програми 14](#_Toc9101153)

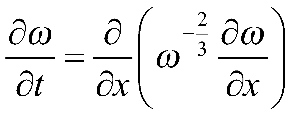
[Висновки 16](#_Toc9101154)

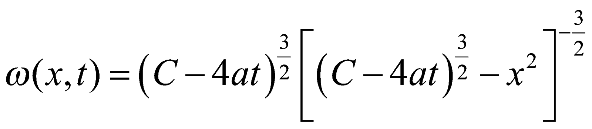
[Додаток. Лістинг програми 18](#_Toc9101155)

Завдання

Варіант 17. Задача №17

Знайти розв’язок рівняння



Точний розв’язок:  
, де С – деяка константа.

Теоретичні відомості

Поняття паралелелізму

Використання декількох потоків у застосуванні означає внесення в нього паралелізму (concurrency). Паралелізм – це одночасне (з погляду прикладного програміста) виконання дій різними фрагментами коду застосування. Така одночасність може бути реалізована на одному процесорі шляхом перемикання задач (випадок псевдопаралелізму), а може ґрунтуватися на паралельному виконанні коду на декількох процесорах (випадок справжнього паралелізму). Потоки абстрагують цю відмінність, даючи можливість розробляти застосування, які в однопроцесорних системах використовують псевдопаралелізм, а при доданні процесорів – справжній паралелізм (такі застосування масштабуються зі збільшенням кількості процесорів).

Види паралелізму:

· паралелізм багатопроцесорних систем;

· паралелізм операцій введення-виведення;

· паралелізм взаємодії з користувачем;

· паралелізм розподілених систем.

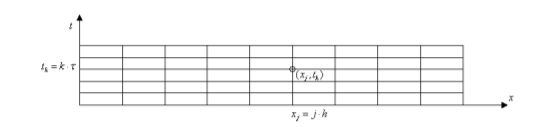
OpenMP(Open Multi-Processing) — це набір директив компілятора, бібліотечних процедур та змінних середовища, які призначені для програмування багатопоточних застосунків на багатопроцесорних системах із спільною пам'яттю.

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).

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.

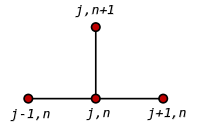
Через точки поділу проводяться прямі, паралельні осях Ox і Ot відповідно. Точки перетину цих прямих утворюють вузли (x, t) сітки. 

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.

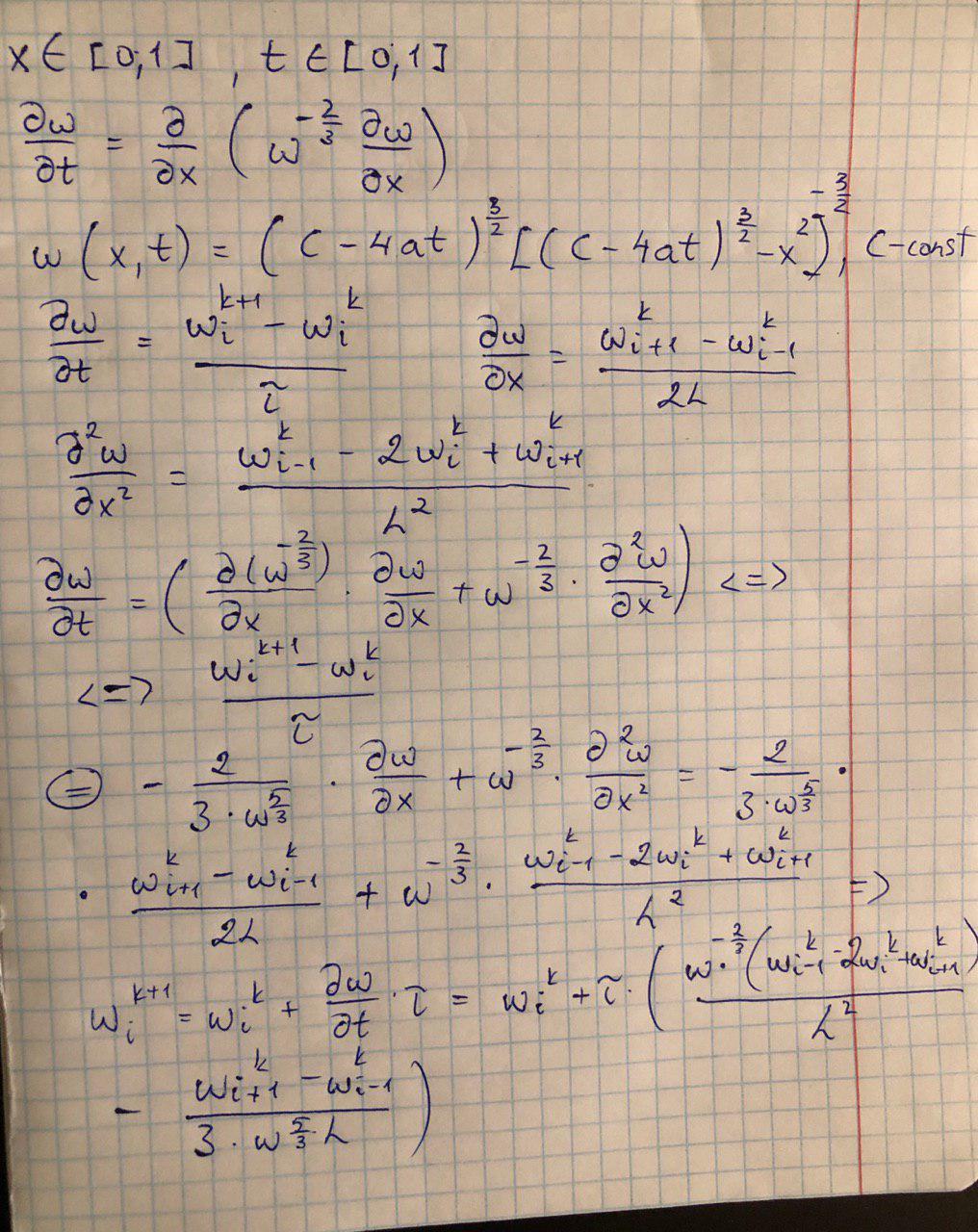
Відомо, що явний метод чисельно стійкий і збіжний при  r\le 1/2 . Похибка пропорційна часовому кроку та квадрату просторового кроку:

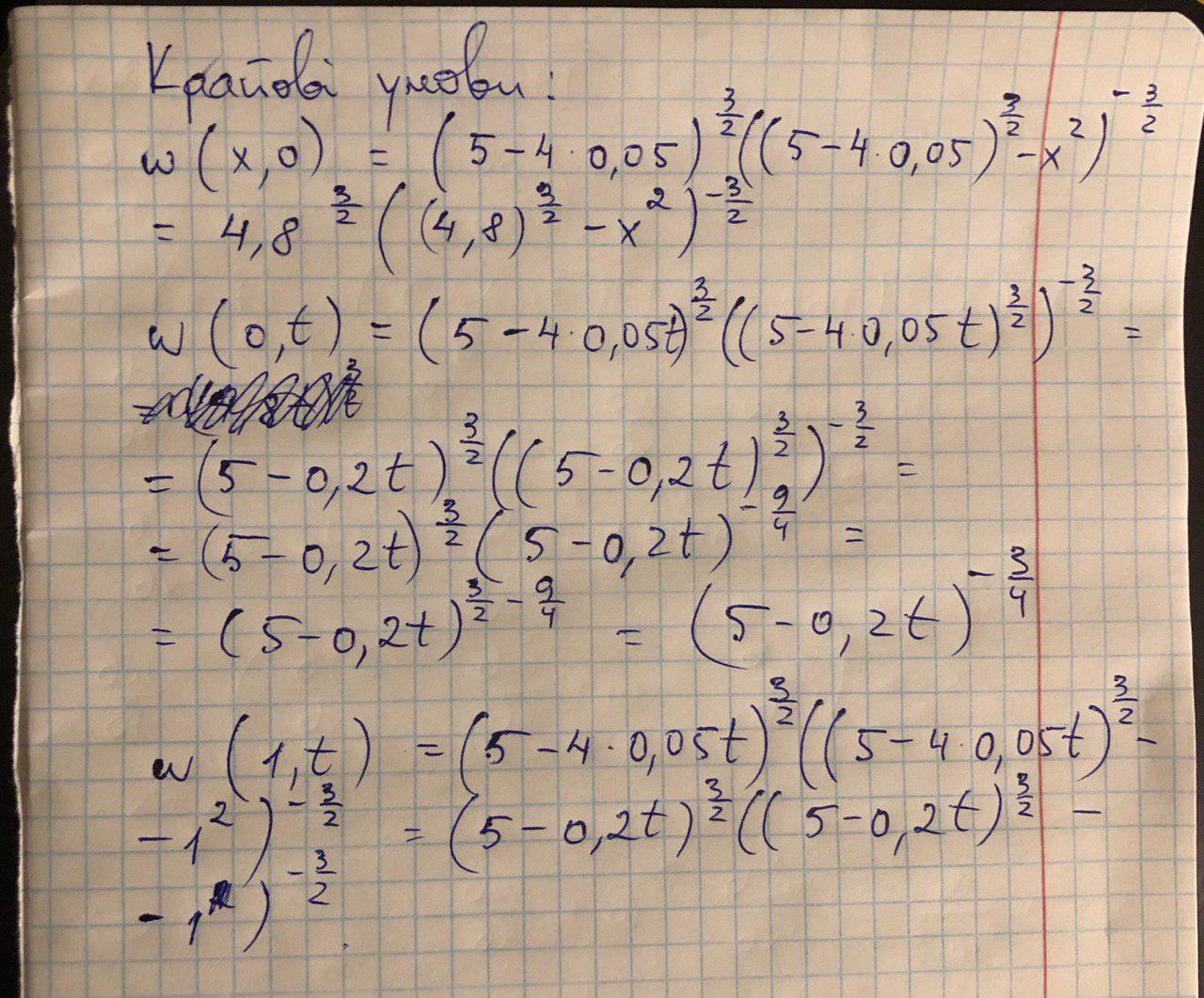
 \Delta u = O(k)+O(h^2)  \, 

Розв’язання

Оберемо такі коефіцієнти:

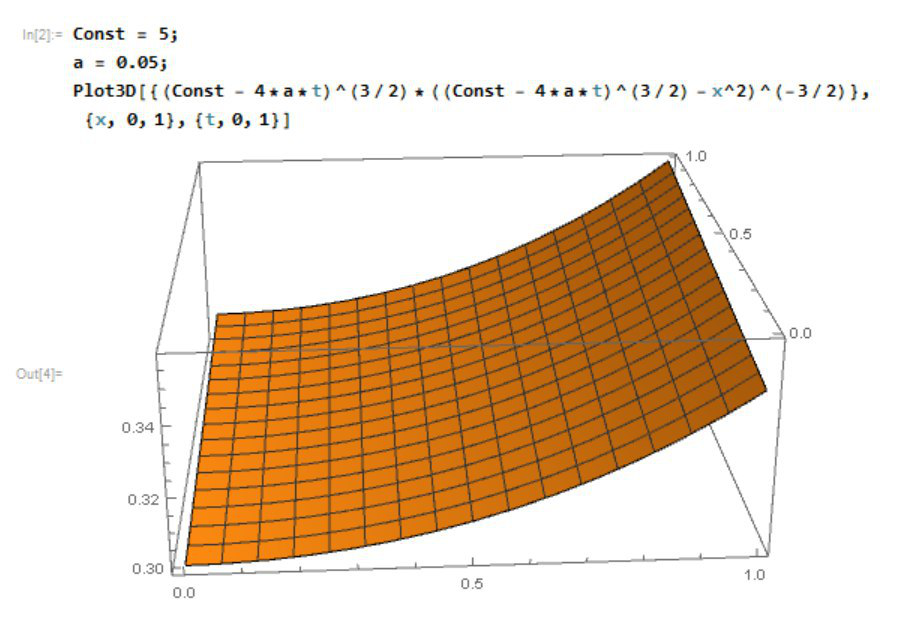
a = 0.05, C = 5



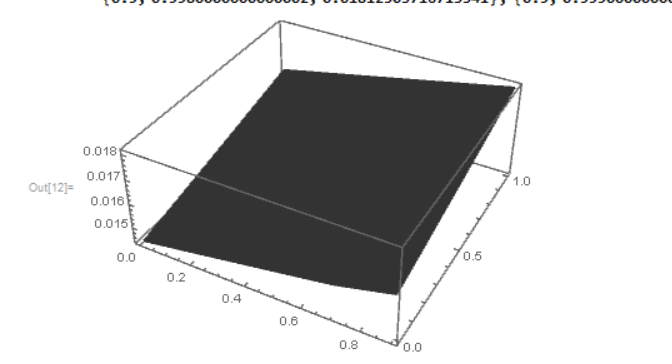


Результати

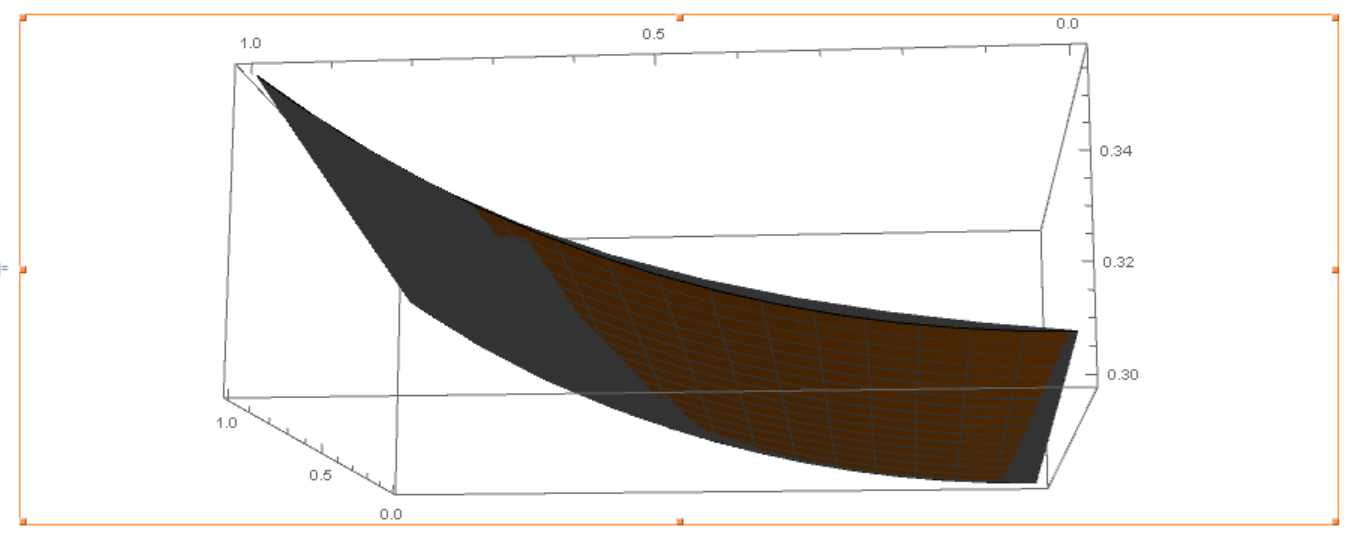
Візуалізація точного розв’язку:

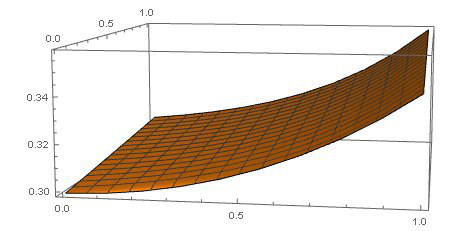


Візуалізація обчисленого розв’язку:



2 графіка на одному:





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 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 |

Висновки

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

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.

Згідно з отриманими результатами: із збільшенням кількості точок у сітці розв’язок стає більш точним.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Петренко А.І. Обчислювальна математика. Суми.: ВМУРоЛ “Україна”,
2. Лекції по технології паралельного обчислення і росподільчих систем КПІ Яременко В. С.

Посилання на GitHub: <https://github.com/Blackkspark/super_course_work>

Додаток. Лістинг програми

#include <iostream>

#include <cmath>

#include <fstream>

#include <thread>

#include <string>

using namespace std;

double C = 5;

double a = 0.05;

const int iterations\_x = 10;

const int iterations\_t = 1000;

double approximated\_solution(double w\_minus\_1, double w, double w\_plus\_1, double x\_move, double t\_move)

{

return w + t\_move \* (pow(w, -2. / 3.) \* ((w\_minus\_1 - 2 \* w + w\_plus\_1) / pow(x\_move, 2))

- (w\_plus\_1 - w\_minus\_1) / (3 \* x\_move \* pow(w, 5. / 3.)));

}

double correct\_solution(double x, double t)

{

double mult1 = pow((C - 4 \* a \* t), (double)3 / 2);

double mult2 = mult1 - pow(x, 2);

return mult1 \* pow(mult2, (double)-3 / 2);

}

void write\_to\_file(string name, double matrix[iterations\_t][iterations\_x], double dx, double dt)

{

ofstream file(name);

double x = 0.0;

double t = 0.0;

file << '{';

for (int i = 0; i < iterations\_t; ++i)

{

x = 0.0;

for (int j = 0; j < iterations\_x; ++j)

{

file << '{' << x << ", " << t << ", " << matrix[i][j] << '}' << ',';

x += dx;

}

t += dt;

}

file << '}';

file.close();

}

void calc\_correct\_matrix(double matrix[iterations\_t][iterations\_x], double iter\_grid\_of\_x, double iter\_grid\_of\_t, double move\_grid\_x, double move\_grid\_t)

{

for (int i = 0; i < iterations\_t; ++i)

{

iter\_grid\_of\_x = 0.0;

for (int j = 0; j < iterations\_x; ++j)

{

matrix[i][j] = correct\_solution(iter\_grid\_of\_x, iter\_grid\_of\_t);

iter\_grid\_of\_x += move\_grid\_x;

}

iter\_grid\_of\_t += move\_grid\_t;

}

}

void calc\_partial\_derivatives(double matrix[iterations\_t][iterations\_x], double iter\_grid\_of\_x, double iter\_grid\_of\_t, double move\_grid\_x, double move\_grid\_t)

{

for (int j = 0; j < iterations\_x; ++j)

{

matrix[0][j] = correct\_solution(iter\_grid\_of\_x, 0);

iter\_grid\_of\_x += move\_grid\_x;

}

for (int i = 0; i < iterations\_t; ++i)

{

matrix[i][0] = correct\_solution(0, iter\_grid\_of\_t);

matrix[i][iterations\_x - 1] = correct\_solution(1, iter\_grid\_of\_t);

iter\_grid\_of\_t += move\_grid\_t;

}

}

void successively\_calc(double calc\_matrix[iterations\_t][iterations\_x], double correct\_matrix[iterations\_t][iterations\_x], double move\_grid\_x, double move\_grid\_t, double &error, int &error\_place\_i, int &error\_place\_j)

{

for (int i = 1; i < iterations\_t; ++i)

{

for (int j = 1; j < iterations\_x - 1; ++j)

{

calc\_matrix[i][j] = approximated\_solution(calc\_matrix[i - 1][j - 1], calc\_matrix[i - 1][j],

calc\_matrix[i - 1][j + 1], move\_grid\_x, move\_grid\_t);

if (fabs(calc\_matrix[i][j] - correct\_matrix[i][j]) > error)

{

error = fabs(calc\_matrix[i][j] - correct\_matrix[i][j]);

error\_place\_i = i;

error\_place\_j = j;

}

}

}

}

void parallel\_calc(double calc\_matrix[iterations\_t][iterations\_x], double correct\_matrix[iterations\_t][iterations\_x], double move\_grid\_x, double move\_grid\_t, double &error, int &error\_place\_i, int &error\_place\_j)

{

for (int i = 1; i < iterations\_t; ++i)

{

#pragma omp parallel for

for (int j = 1; j < iterations\_x - 1; ++j)

{

calc\_matrix[i][j] = approximated\_solution(calc\_matrix[i - 1][j - 1], calc\_matrix[i - 1][j],

calc\_matrix[i - 1][j + 1], move\_grid\_x, move\_grid\_t);

if (fabs(calc\_matrix[i][j] - correct\_matrix[i][j]) > error)

{

error = fabs(calc\_matrix[i][j] - correct\_matrix[i][j]);

error\_place\_i = i;

error\_place\_j = j;

}

}

}

}

int main()

{

setlocale(0, "rus");

double x\_iter\_grid = 0.0, t\_iter\_grid = 0.0;

double x\_move = 1. / (iterations\_x - 1);

double t\_move = 1. / (iterations\_t - 1);

double correct\_matrix[iterations\_t][iterations\_x];

calc\_correct\_matrix(correct\_matrix, x\_iter\_grid, t\_iter\_grid, x\_move, t\_move);

write\_to\_file("super\_correct\_result.txt", correct\_matrix, x\_move, t\_move);

x\_iter\_grid = 0;

t\_iter\_grid = 0;

double approximated\_matrix\_sequential[iterations\_t][iterations\_x];

calc\_partial\_derivatives(approximated\_matrix\_sequential, x\_iter\_grid, t\_iter\_grid, x\_move, t\_move);

double error\_value = 0.0;

int error\_i\_pos = 0;

int error\_j\_pos = 0;

successively\_calc(approximated\_matrix\_sequential, correct\_matrix, x\_move, t\_move, error\_value, error\_i\_pos, error\_j\_pos);

cout << "Последовательное исчисление:" << endl;

cout << "Значение абсолютной погрешности: " << error\_value << endl;

cout << "Относительная погрешность: " <<

fabs((approximated\_matrix\_sequential[error\_i\_pos][error\_j\_pos] - correct\_matrix[error\_i\_pos][error\_j\_pos])

/ correct\_matrix[error\_i\_pos][error\_j\_pos]) \* 100 << "%" << endl << endl;

write\_to\_file("super\_approximated\_result\_sic.txt", approximated\_matrix\_sequential, x\_move, t\_move);

double approximated\_matrix\_parallel[iterations\_t][iterations\_x];

calc\_partial\_derivatives(approximated\_matrix\_parallel, x\_iter\_grid, t\_iter\_grid, x\_move, t\_move);

error\_value = 0.0;

error\_i\_pos = 0;

error\_j\_pos = 0;

parallel\_calc(approximated\_matrix\_parallel, correct\_matrix, x\_move, t\_move, error\_value, error\_i\_pos, error\_j\_pos);

cout << "Паралельное исчисление:" << endl;

cout << "Значение абсолютной погрешности: " << error\_value << endl;

cout << "Относительная погрешность: " <<

fabs((approximated\_matrix\_parallel[error\_i\_pos][error\_j\_pos] - correct\_matrix[error\_i\_pos][error\_j\_pos])

/ correct\_matrix[error\_i\_pos][error\_j\_pos]) \* 100 << "%" << endl << endl;

write\_to\_file("super\_approximated\_result\_paral.txt", approximated\_matrix\_parallel, x\_move, t\_move);

system("pause");

return 0;

}