**Міністерство освіти і науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут ім. Ігоря Сікорського»**

Інститут Прикладного системного аналізу

(назва факультету, інституту)

Кафедра 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-62 Білоус Євгенія Юріївна

(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. Білоус Є.Ю.,

гр. ДА-62

Допущений до захисту Зал. книжка

№ ДА-6206

Захищено із оцінкою

Київ – 2019

Форма No У-6.01   
Затв. наказом УРСР

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

(назва вищого навчального закладу)

Кафедра Системного проектування

Дисципліна Паралельні обчислення

Спеціальність 6.050101

Курс 3 Група ДА-62 Семестр 6

**ЗАВДАННЯ**

**на курсову роботу студента**

Білоус Євгенії Юріївни

(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) до 20.05.2019 р.

Студент Білоус Є.Ю.

(підпис)

Керівник ас. Яременко В.С. (підпис)

«20» травня 2019 р.

Зміст

[ЗАВДАННЯ 4](#_Toc9084717)

[ТЕОРЕТИЧНІ ВІДОМОСТІ 5](#_Toc9084718)

[МЕТОД СКІНЧЕННИХ РІЗНИЦЬ 5](#_Toc9084719)

[ЯВНИЙ МЕТОД 6](#_Toc9084720)

[РОЗВ’ЯЗОК 7](#_Toc9084721)

[РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ 9](#_Toc9084722)

[ОБЧИСЛЕННЯ ПОХИБОК 9](#_Toc9084723)

[ПОВЕРХНЯ ЗА ЯВНИМ МЕТОДОМ 10](#_Toc9084724)

[ВИСНОВОК 12](#_Toc9084725)

[ДОДАТОК 1. ЛІСТИНГ ПРОГРАМИ 13](#_Toc9084726)

# **ЗАВДАННЯ**

Задача № 4.

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 – деяка константа.

# **ТЕОРЕТИЧНІ ВІДОМОСТІ**

## МЕТОД СКІНЧЕННИХ РІЗНИЦЬ

Розглянемо розв’язання лінійного одновимірного рівняння теплопровідності:

з крайовими

та початковою

умовами.

Одним зі способів чисельного розв’язання такого типу рівнянь є апроксимація всіх похідних рівняння їхніми скінченими різницями. Розділимо ділянку простору, використовуючи сітку , а часовий інтервал – сітку . Нехай крок сітки буде рівномірний і в часі, і в просторі. Тоді різницю між двома послідовними точками в просторі позначимо , а в часі – , а чисельною апроксимацією значення буде .

### ЯВНИЙ МЕТОД

Використовуючи праву різницю по часу та центральну різницю по простору для апроксимації похідних, отримаємо рекурентне рівняння:

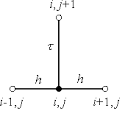


Рис. 1 – Шаблон найбільш поширеного явного методу

Тоді для значення отримуємо:

, де

Таким чином, маючи останнє рекурентне співвідношення та знаючи значення функції в момент часу , можна отримати відповідні значення в момент часу .

Відомо, що явний метод чисельно стійкий і збіжний при , а його похибка пропорційна часовому кроку та квадрату просторового кроку:

# **РОЗВ’ЯЗОК**

Оберемо такі коефіцієнти:

a = 5, b = -5, C = 1

Рівняння набуде такого вигляду:

,

Точний розв’язок:

Початкова умова:

Граничні умови:

Зробимо такі заміни:

Тоді різницеве рівняння матиме наступний вигляд:

звідки

Отримаємо формулу для кожного з (n-1) внутрішніх вузлів поточного часового шару.

# **РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ**

## ОБЧИСЛЕННЯ ПОХИБОК

Таблиця 1 Похибки для явної різницевої схеми

|  |  |  |  |
| --- | --- | --- | --- |
| **Явна різницева схема** | | | |
| Похибки |  |  |  |
| Середня абсолютна | 8.473398782231165E-4 | 8.987995678378125E-4 | 9.156241912536458E-4 |
| Максимальна абсолютна | 0.002157898565984629 | 0.002161937855445023 | 0.0021626980157375165 |
| Середня відносна, % | 4.765637179402645 | 5.05374260121234 | 5.148217019996143 |
| Максимальна відносна, % | 8.912916279974736 | 8.953327443262456 | 8.963694981844117 |

Таблиця 2 Час виконання у наносекундах

|  |  |  |
| --- | --- | --- |
|  | **Послідовний** | **Паралельний** |
|  | 1700246 ns | 165952725 ns |

### ПОВЕРХНЯ ЗА ЯВНИМ МЕТОДОМ

На рисунках 1-3 зображено накладання поверхні точного рішення та поверхні, знайденої явним кінцево-різницевим методом, із кроками і . Графік було побудовано за допомогою пакету Wolfram Mathematica.

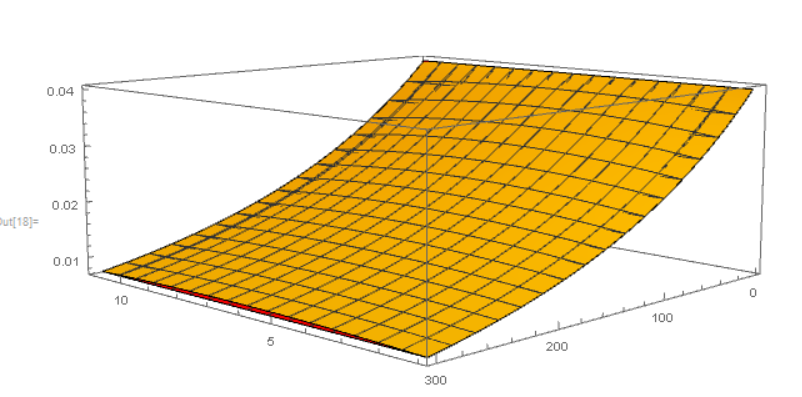


Рисунок 1

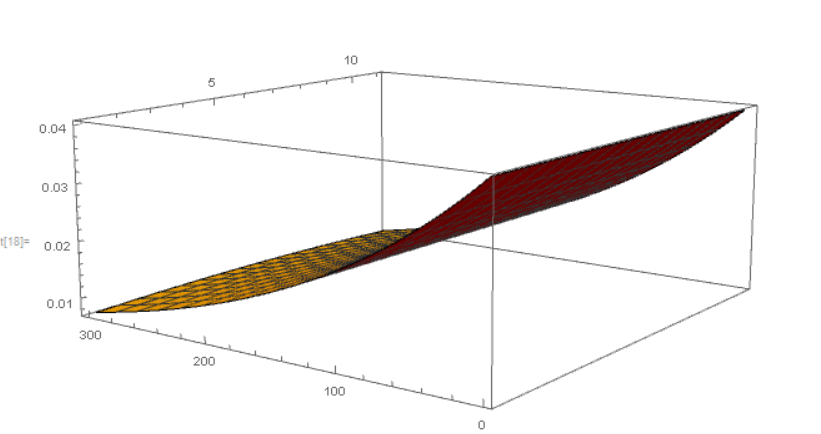


Рисунок 2

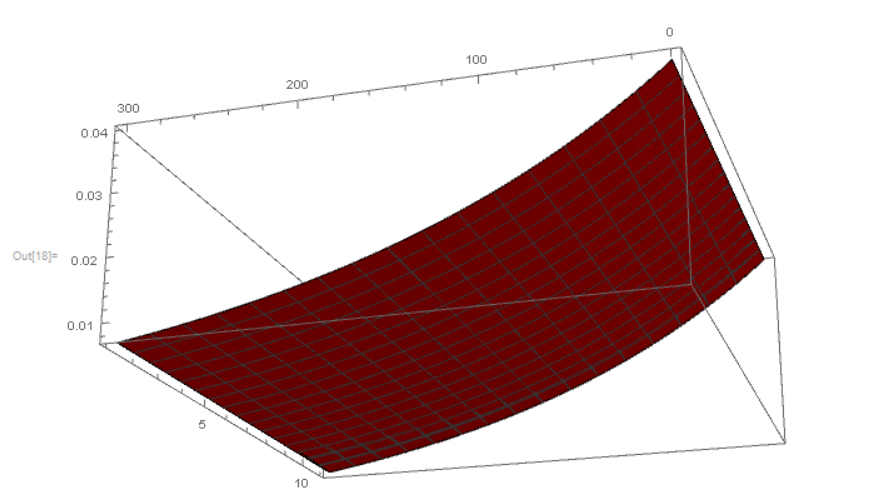


Рисунок 3

# **ВИСНОВОК**

У ході даної курсової роботи було розв’язано диференційне рівняння у частинних похідних за допомогою явного методу кінцевих різниць. Паралельна реалізація виконана за допомогою засобів IntStream. У роботі наведено результати роботи розробленої програми, виконано візуалізацію обчисленого та точного розв’язків.

Явний метод є найпростішим для реалізації і достатньо точним, але умова стійкості ускладнює його практичне застосування.

# **ДОДАТОК 1. ЛІСТИНГ ПРОГРАМИ**

**Main.java**

|  |
| --- |
| package courseWork;  public class Main {   public static void main(String[] args) {  System.out.println("SERIAL RESULT");  Dyffur dyffur = new Dyffur();  long startTimeSerial = System.nanoTime();  double[][] serialSolve = new SerialSolve(dyffur).solve();  long endTimeSerial = System.nanoTime();  long executeTimeSerial = endTimeSerial - startTimeSerial;  System.out.println("Approximate result");  dyffur.printMatrix(serialSolve);    System.out.println();  System.out.println("\nPARALLEL RESULT");  long startTimeParallel = System.nanoTime();  double[][] parallelSolve = new ParallelSolve(dyffur).solve();  long endTimeParallel = System.nanoTime();  long executeTimeParallel = endTimeParallel - startTimeParallel;  for (int i = 0; i < dyffur.getTPointsQuantity(); ++i) {  for (int j = 0; j < dyffur.getHPointsQuantity(); ++j) {  // parallelSolve[i][j] = Math.abs(parallelSolve[i][j] - serialSolve[i][j]);   System.out.print(String.format("%.7f\t", parallelSolve[i][j]));  }  System.out.println();  }    System.out.println("\nExecuted time for serial result " + executeTimeSerial + " ns");  System.out.println("\nExecuted time for parallel result " + executeTimeParallel + " ns");    printResult(dyffur, serialSolve, calculateExactResult(dyffur));  }   private static double[][] calculateExactResult(Dyffur dyffur) {  double trueMatrix[][] = new double[dyffur.getTPointsQuantity()][dyffur.getHPointsQuantity()];  double t = dyffur.getT0();  for (int i = 0; i < dyffur.getTPointsQuantity(); i++) {  double x = dyffur.getX0();  for (int j = 0; j < dyffur.getHPointsQuantity(); j++) {  trueMatrix[i][j] = dyffur.calculateTrueSolution(x, t);  x += dyffur.getH();  }  t += dyffur.getTau();  }  System.out.println("\nExact result");  dyffur.printMatrix(trueMatrix);   return trueMatrix;  }   private static void printResult(Dyffur dyffur, double[][] serialSolve, double[][] trueMatrix) {  System.out.println("Average absolute error: ");  System.out.println(averageAbsoluteError(serialSolve, trueMatrix, dyffur));  System.out.println("Max Absolute Error: ");  System.out.println(maxAbsoluteError(serialSolve, trueMatrix, dyffur));  System.out.println("Average Relative Error: ");  System.out.println(averageRelativeError(serialSolve, trueMatrix, dyffur));  System.out.println("Max Relative Error: ");  System.out.println(maxRelativeError(serialSolve, trueMatrix, dyffur));  }    private static double averageAbsoluteError(double[][] approximateResult, double[][] exactMatrix, Dyffur dyffur) {  double[][] matrix = new double[dyffur.getTPointsQuantity()][dyffur.getHPointsQuantity()];  double error = 0;  double errorResult = 0;   for (int i = 0; i < dyffur.getTPointsQuantity(); i++) {  for (int j = 0; j < dyffur.getHPointsQuantity(); j++) {  matrix[i][j] = Math.abs(approximateResult[i][j] - exactMatrix[i][j]);  }  }   for (int i = 0; i < dyffur.getTPointsQuantity(); i++) {  for (int j = 0; j < dyffur.getHPointsQuantity(); j++) {  error += matrix[i][j];  }  }  errorResult = error / (dyffur.getTPointsQuantity() \* dyffur.getHPointsQuantity());   return errorResult;  }   private static double maxAbsoluteError(double[][] approximateResult, double[][] exactMatrix, Dyffur dyffur) {  double[][] matrix = new double[dyffur.getTPointsQuantity()][dyffur.getHPointsQuantity()];  double maxError = 0;   for (int i = 0; i < dyffur.getTPointsQuantity(); i++) {  for (int j = 0; j < dyffur.getHPointsQuantity(); j++) {  matrix[i][j] = Math.abs(approximateResult[i][j] - exactMatrix[i][j]);  }  }   maxError = matrix[0][0];  for (int i = 0; i < dyffur.getTPointsQuantity(); i++) {  for (int j = 0; j < dyffur.getHPointsQuantity(); j++) {  if (maxError < matrix[i][j]) {  maxError = matrix[i][j];  }  }  }    return maxError;  }    private static double averageRelativeError(double[][] approximateResult, double[][] exactMatrix, Dyffur dyffur) {  double[][] matrix = new double[dyffur.getTPointsQuantity()][dyffur.getHPointsQuantity()];  double error = 0;  double errorResult = 0;   for (int i = 0; i < dyffur.getTPointsQuantity(); i++) {  for (int j = 0; j < dyffur.getHPointsQuantity(); j++) {  matrix[i][j] = (Math.abs(approximateResult[i][j] - exactMatrix[i][j]) / exactMatrix[i][j]) \* 100;  error += matrix[i][j];  }  }   errorResult = error / (dyffur.getTPointsQuantity() \* dyffur.getHPointsQuantity());   return errorResult;  }   private static double maxRelativeError(double[][] approximateResult, double[][] exactMatrix, Dyffur dyffur) {  double[][] matrix = new double[dyffur.getTPointsQuantity()][dyffur.getHPointsQuantity()];  double maxError = 0;   for (int i = 0; i < dyffur.getTPointsQuantity(); i++) {  for (int j = 0; j < dyffur.getHPointsQuantity(); j++) {  matrix[i][j] = (Math.abs(approximateResult[i][j] - exactMatrix[i][j]) / exactMatrix[i][j]) \* 100;   }  }   maxError = matrix[0][0];  for (int i = 0; i < dyffur.getTPointsQuantity(); i++) {  for (int j = 0; j < dyffur.getHPointsQuantity(); j++) {  if (maxError < matrix[i][j]) {  maxError = matrix[i][j];  }  }  }    return maxError;  }   }  **Dyffur.java**  package courseWork;  public class Dyffur {   private final double x0 = 0;  private final double x1 = 1;  private final double h = 1.0 / 10;  private final double t0 = 0;  private final double t1 = 1;  private final double tau = 1.0 / 300;  private final double a = -1;  private final double b = 0.00008;  private final double C = 5;   public double getX0() {  return x0;  }    public double getH() {  return h;  }   public double getT0() {  return t0;  }    public double getTau() {  return tau;  }   public int getTPointsQuantity() {  return (int) Math.ceil((t1 - t0) / tau) + 1;  }   public int getHPointsQuantity() {  return (int) Math.ceil((x1 - x0) / h) + 1;  }   public double calculateTrueSolution(double x, double t) {  return Math.pow(-Math.sqrt(-b / a) + C \* Math.exp((-5 \* a \* t) / 6 + Math.sqrt((a + 1) / 6) \* x), -2);  }   public double calculateBottomBorder(double x) {  return Math.pow(-Math.sqrt(-b / a) + C \* Math.exp(Math.sqrt((a + 1) / 6) \* x), -2);  }   public double calculateLeftBorder(double t) {  return Math.pow(-Math.sqrt(-b / a) + C \* Math.exp((-5 \* a \* t) / 6), -2);  }   public double calculateRightBorder(double t) {  double x = 1;  return Math.pow(-Math.sqrt(-b / a) + C \* Math.exp((-5 \* a \* t) / 6 + Math.sqrt((a + 1) / 6) \* x), -2);  }   public double calculateApproximateSolution(double wLeft, double wCurrent, double wRight) {  double sigma = tau / Math.pow(h, 2);  return wCurrent + sigma \* (wRight - 2 \* wCurrent + wLeft) + a \* tau \* wCurrent + b \* tau \* Math.pow(wCurrent, 2);  }   public void printMatrix(double[][] matrix) {  for (int i = 0; i < matrix.length; ++i) {  for (int j = 0; j < matrix[i].length; ++j) {  System.out.print(String.format("%.7f\t", matrix[i][j]));  }  System.out.println();  }  }  }  **SerialSolve.java**  package courseWork;  public class SerialSolve {   private int TPointsQuantity;  private int HPointsQuantity;  private double x0;  private double h;  private double t0;  private double tau;  private Dyffur dyffur;   public SerialSolve(Dyffur dyffur) {  this.dyffur = dyffur;  this.TPointsQuantity = dyffur.getTPointsQuantity();  this.HPointsQuantity = dyffur.getHPointsQuantity();  this.x0 = dyffur.getX0();  this.h = dyffur.getH();  this.t0 = dyffur.getT0();  this.tau = dyffur.getTau();  }   public double[][] solve() {  double t = t0 + tau;  double x = x0;  double w[][] = new double[TPointsQuantity][HPointsQuantity];  for (int j = 0; j < HPointsQuantity; j++, x += h) {  w[0][j] = dyffur.calculateBottomBorder(x);  }   for (int i = 1; i < TPointsQuantity; ++i, t += tau) {  w[i][0] = dyffur.calculateLeftBorder(t);  for (int j = 1; j < HPointsQuantity - 1; j++) {  w[i][j] = dyffur.calculateApproximateSolution(w[i-1][j-1],w[i-1][j],w[i-1][j+1]);  }  w[i][HPointsQuantity - 1] = dyffur.calculateRightBorder(t);  }  return w;  } }  **ParallelSolve.java**  package courseWork;  import java.util.concurrent.atomic.AtomicInteger; import java.util.stream.IntStream;  public class ParallelSolve {   private int TPointsQuantity;  private int HPointsQuantity;  private double x0;  private double h;  private double t0;  private double tau;  private Dyffur dyffur;   public ParallelSolve(Dyffur dyffur) {  this.dyffur = dyffur;  this.TPointsQuantity = dyffur.getTPointsQuantity();  this.HPointsQuantity = dyffur.getHPointsQuantity();  this.x0 = dyffur.getX0();  this.h = dyffur.getH();  this.t0 = dyffur.getT0();  this.tau = dyffur.getTau();  }   public double[][] solve() {  double t = t0 + tau;  double x = x0;  double w[][] = new double[TPointsQuantity][HPointsQuantity];  for (int j = 0; j < HPointsQuantity; j++, x += h) {  w[0][j] = dyffur.calculateBottomBorder(x);  }   for (int i = 1; i < TPointsQuantity; ++i, t += tau) {  w[i][0] = dyffur.calculateLeftBorder(t);  AtomicInteger ai = new AtomicInteger(i);  IntStream.range(1, dyffur.getHPointsQuantity()-1).parallel().forEach(j -> {  int m=ai.get();  w[m][j] = dyffur.calculateApproximateSolution(w[m-1][j-1],w[m-1][j],w[m-1][j+1]);  });  w[i][HPointsQuantity - 1] = dyffur.calculateRightBorder(t);  }  return w;  } }  Посилання на GitHub:  <https://github.com/Eugenia0000/Course_Work> |
|  |  |
|  |  |
|  |  |