МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

**Паралельні обчислення**

**Курсова робота**

Виконав:

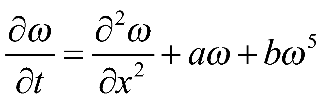
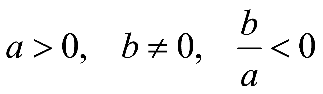
Студент групи ДА-62

Григорян Б. В.

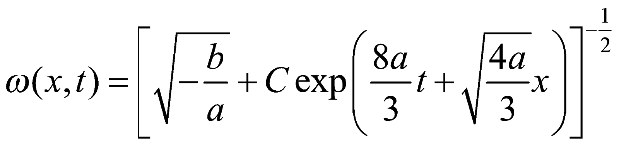
Перевірив:

Яременко В. С.

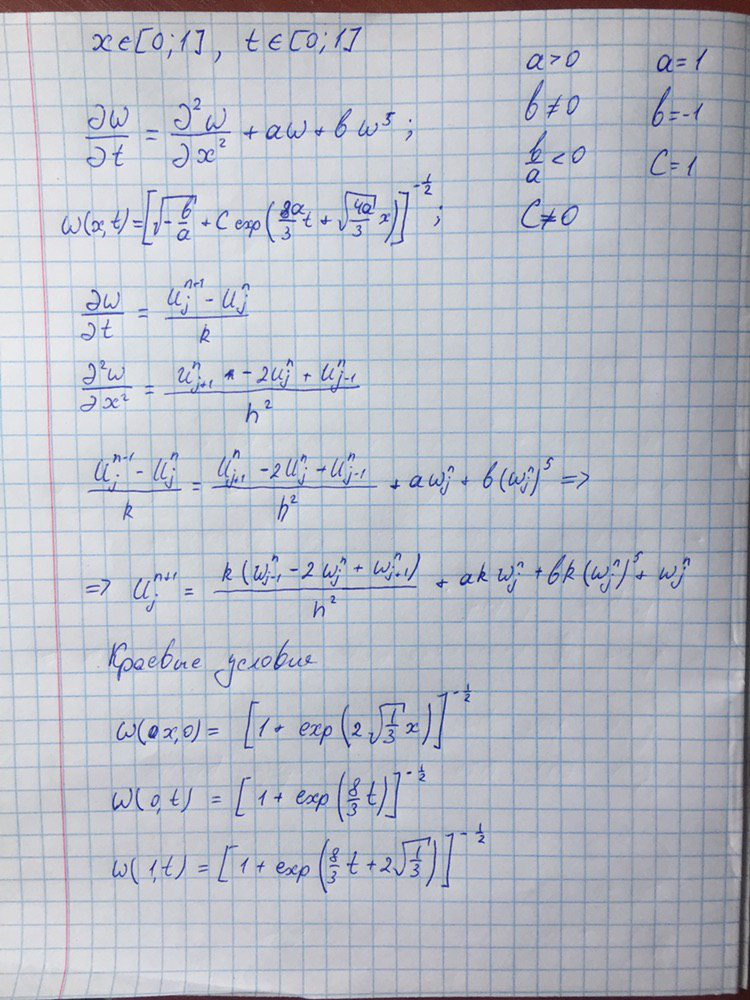
Задача №7

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 https://lh3.googleusercontent.com/mvHsH8oA3xKct_tFxv5tfObZK22heWAc3eGGZEJGMS_Ht6nMnA2KQQhJzIGtXHmgV9rYw6g8iIh-bg1ZNLr7vjwcVC_BnXQT2fZ-W6d23Cwk7bZvX2fj-geaG59mGb0vfzkBz5c8FATAqAb-4Q- деяка константа.

**Розв’язок**



**Лістинг програми**

***pch.h***

#ifndef PCH\_H

#define PCH\_H

class ApproximationObj

{

public:

ApproximationObj(double paramA, double paramB, double paramC) :

a(paramA), b(paramB), c(paramC) {}

double getApproximation(const double omegaLeft, const double omegaRight, const double omegaCenter, const double tauStep, const double hStep);

double getAccurate(double x, double t);

double bottomLimit(double x);

double leftLimit(double t);

double rightLimit(double t);

private:

const double a;

const double b;

const double c;

};

#endif

***pch.cpp***

#include "pch.h"

#include <cmath>

double ApproximationObj::getApproximation(const double omegaLeft, const double omegaRight, const double omegaCenter, const double tauStep, const double hStep)

{

return tauStep \* (omegaLeft - 2 \* omegaCenter + omegaRight) / pow(hStep, 2) + tauStep \* a \* omegaCenter + tauStep \* b \* pow(omegaCenter, 1 / 2) + omegaCenter;

}

double ApproximationObj::getAccurate(double aX, double aT)

{

double x = pow(sqrt(-b / a) + c \* std::exp((8.0 \* a \* aT) / 12.0 + sqrt((4.0 \* a) / 12.0) \* aX), - 0.5);

return x;

}

double ApproximationObj::bottomLimit(double aX)

{

return getAccurate(aX, 0);

}

double ApproximationObj::leftLimit(double aT)

{

return getAccurate(0, aT);

}

double ApproximationObj::rightLimit(double aT)

{

return getAccurate(1, aT);

}

***C++.cpp***

#include "pch.h"

#include <fstream>

#include <iostream>

#include <vector>

#include <omp.h>

struct Error

{

double absolute, relative;

size\_t row, col;

Error() : absolute(0), relative(0) {}

};

Error getError(double\*\* accurateMatrix, double\*\* approximatedMatrix, size\_t rows, size\_t columns);

int main(int argc, char \*argv[])

{

ApproximationObj approxObj(1, -1, 1);

size\_t xIterations, tIterations;

std::cout << "Enter x iterations: ";

std::cin >> xIterations;

std::cout << "Enter t iterations: ";

std::cin >> tIterations;

double tauStep = 1.0 / (tIterations - 1);

double hStep = 1.0 / (xIterations - 1);

double\*\* accurateMatrix = new double\*[tIterations];

double\*\* approximateMatrix = new double\*[tIterations];

for (size\_t i = 0; i < tIterations; ++i) {

accurateMatrix[i] = new double[xIterations];

approximateMatrix[i] = new double[xIterations];

}

double currentX = 0, currentT = 0;

std::ofstream AccurateFile("Accurate Matrix.txt");

if (AccurateFile.is\_open())

{

std::cout << std::endl << "Accurate Matrix is in file: Accurate Matrix.txt";

#pragma omp parallel for private(currentX, currentT)

for (size\_t i = 0; i < tIterations; ++i) {

currentT = tauStep \* i;

for (size\_t j = 0; j < xIterations; ++j) {

currentX = hStep \* j;

accurateMatrix[i][j] = approxObj.getAccurate(currentX, currentT);

//std::cout << accurateMatrix[i][j] << " ";

AccurateFile << "{" << currentX << ", " << currentT << ", " << accurateMatrix[i][j] << "}," << "\n";

//AccurateFile << accurateMatrix[i][j] << " ";

}

}

AccurateFile.close();

}

else std::cout << "Unable to open file";

currentX = 0;

currentT = 0;

for (size\_t i = 0; i < tIterations; ++i) {

approximateMatrix[i][0] = approxObj.leftLimit(currentT);

approximateMatrix[i][xIterations - 1] = approxObj.rightLimit(currentT);

currentT += tauStep;

}

//currentX += hStep;

for (size\_t i = 1; i < xIterations - 1; ++i) {

approximateMatrix[0][i] = approxObj.bottomLimit(currentX);

currentX += hStep;

}

currentX = 0;

currentT = 0;

std::cout << std::endl << "Approximate Matrix is in file: Approximate Matrix.txt" << std::endl;

for (size\_t i = 1; i < tIterations; ++i) {

currentX = 0;

#pragma omp paralel for

for (size\_t j = 1; j < xIterations - 1; ++j) {

approximateMatrix[i][j] = approxObj.getApproximation(approximateMatrix[i - 1][j - 1], approximateMatrix[i - 1][j + 1], approximateMatrix[i - 1][j], tauStep, hStep);

//std::cout << approximateMatrix[i][j] << " ";

currentX += hStep;

}

currentT += tauStep;

}

std::ofstream ApproximateFile("Approximate Matrix.txt");

currentX = 0;

currentT = 0;

if (ApproximateFile.is\_open())

{

std::cout << std::endl << "Approximate Matrix is in file: Approximate Matrix.txt" << std::endl;

for (size\_t i = 1; i < tIterations; ++i) {

currentX = 0;

for (size\_t j = 0; j < xIterations; ++j) {

//std::cout << approximateMatrix[i][j] << " ";

ApproximateFile << "{" << currentX << ", " << currentT << ", " << approximateMatrix[i][j] << "}," << "\n";

currentX += hStep;

}

currentT += tauStep;

}

ApproximateFile.close();

}

else std::cout << "Unable to open file";

Error error = getError(accurateMatrix, approximateMatrix, tIterations, xIterations);

std::cout << std::endl;

std::cout << "Absolute error: " << " i[" << error.col << "] - j[" << error.row << "] - [" << error.absolute << "] " << std::endl;

std::cout << "Relative error: " << " i[" << error.col << "] - j[" << error.row << "] - [" << error.relative << "] " << std::endl;

system("pause");

}

Error getError(double\*\* accurateMatrix, double\*\* approximatedMatrix, size\_t rows, size\_t columns)

{

Error error;

for (size\_t i = 0; i < rows; ++i) {

for (size\_t j = 0; j < columns; ++j) {

if (fabs(approximatedMatrix[i][j] - accurateMatrix[i][j]) > error.absolute) {

error.absolute = fabs(approximatedMatrix[i][j] - accurateMatrix[i][j]);

error.row = i;

error.col = j;

}

}

}

error.relative = error.absolute / accurateMatrix[error.row][error.col];

return error;

}

**Візуалізація точного розв’язку з отриманим розв’язком** 