**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС**

**«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»**

**НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

**КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ**

**Лабораторна робота №9**

**з курсу «Чисельні методи»**

**тема: «Диференціальні рівняння**

**у частинних похідних»**

**Виконав: студент 3 курсу**

**групи КА-23**

**Деундяк О.В.**

**Прийняла: Кузнєцова Н. В.**

**Київ – 2014р.**

***Рівняння коливань струни.***  
Розв'язати рівняння гіперболічного типу  ***u tt = u xx + F(t,x),     0 < x < L = 1,***     (2)   
для функції ***u(t,x)*** з початковими ***u(0,x) = u0(x), ut(0,x) = 0***  
та крайовими ***u(t,0) = u1(t); u(t,L) = u2(t)***  
умовами. 



**Текст програми:**

**TMA.h**

#pragma once

#include <vector>

class TMA

{

std::vector<double> c, d, r;

size\_t i; // number of iterations

size\_t n; // size of system

public:

TMA(size\_t n) : i(0), c(n), d(n), r(n)

{

this->n = n;

}

std::vector<double> Result();

void Iterate(double a, double b, double c, double d);

};

**TMA.cpp**

#include "TMA.h"

void TMA::Iterate(double A, double B, double C, double D)

{

if (i == n)

return;

if (i == 0)

{

c[i] = C / B;

d[i] = D / B;

}

else

{

c[i] = C / (B - A\*c[i - 1]);

d[i] = (D - A\*d[i - 1]) / (B - A\*c[i - 1]);

}

i++;

}

std::vector<double> TMA::Result()

{

if (i != n)

return std::vector<double>();

r[n - 1] = d[n - 1];;

for (size\_t i = n - 1; i > 0; i--)

r[i - 1] = d[i - 1] - c[i - 1] \* r[i];

return r;

};

**Hyper.h**

#pragma once

#define \_USE\_MATH\_DEFINES

#include <cmath>

#include <functional>

#include <vector>

typedef std::function<double(double)> func;

typedef std::function<double(double, double)> func2;

class Hyper

{

public:

static std::vector<std::vector<double>> Process(double s, func2 F, func u0, func u1, func u2, func v0, size\_t Nt, size\_t Nl, double period, double length);

};

**Hyper.cpp**

#include "Hyper.h"

#include "TMA.h"

extern func2 answ;

#include <fstream>

#include <algorithm>

std::ofstream sf("s.txt");

std::vector<std::vector<double>> Hyper::Process(double s, func2 F,

func u0, func u1, func u2, func v0,

size\_t Nt, size\_t Nl, double period, double length)

{

std::vector<std::vector<double>> u(Nt+1, std::vector<double>(Nl+1)); // first time, next x

double h = length / Nl;

double dt = period / Nt;

double q = h / dt\*h / dt;

for (size\_t n = 0; n <= Nl; n++)

u[0][n] = u0(n\*h);

u[1][0] = u1(dt);

u[1][Nl] = u2(dt);

for (size\_t n = 1; n < Nl; n++)

u[1][n] = u[0][n] + dt\*v0(h\*n) + dt\*dt / 2 \* ((u[0][n + 1] - 2 \* u[0][n] + u[0][n - 1]) / h / h +F(0, h\*n));

//u[1][n] = answ(dt, n\*h);

double err = 0;

for (size\_t k = 1; k < Nt; k++) // time loop

{

TMA tma(Nl + 1);

tma.Iterate(0, 1, 0, u1((k + 1)\*dt));

for (size\_t n = 1; n < Nl; n++)

{

tma.Iterate(-s, q + 2 \* s, -s,

2 \* u[k][n]\*q - u[k - 1][n]\*q + (1 - 2 \* s)\*(u[k][n + 1] - 2 \* u[k][n] + u[k][n - 1]) + s\*(u[k - 1][n + 1] - 2 \* u[k - 1][n] + u[k - 1][n - 1])

+ (s\*F((k + 1)\*dt, n\*h) + (1 - 2 \* s)\*F(k\*dt, n\*h) + s\*F((k - 1)\*dt, n\*h))\*h\*h);

//2 \* answ(k\*dt, n\*h) \* q - answ((k - 1)\*dt, n\*h) \* q + (1 - 2 \* s)\*(answ(k\*dt, (n + 1)\*h) - 2 \* answ(k\*dt, n\*h) + answ(k\*dt, (n - 1)\*h))

//+ s\*(answ((k - 1)\*dt, (n + 1)\*h) - 2 \* answ((k - 1)\*dt, n\*h) + answ((k - 1)\*dt, (n - 1)\*h))

//+ F(k\*dt, n\*h) \*h\*h);

}

tma.Iterate(0, 1, 0, u2((k + 1)\*dt));

u[k+1] = tma.Result();

}

return u;

}

**main.cpp**

// utt=uxx+F(t,x) 0<x<L=1

// u(t,x) u(0,x)=u0(x), ut(0,x)=0

// u(t,0)=u1(t), u(t,L)=u2(t)

// u0=u(0,x)=(x+0.2)sin(M\_PIx/2)

// u1=u(t,0)=0

// u2=u(t,L)=u(t,1)=1.2cos(M\_PI\*t)

// F(t,x)=-M\_PI/2cos(M\_PI\*t)\*(2cos(M\_PI\*x/2)+M\_PI\*(x+0.2)sin(M\_PI\*x/2)

// v0=du/dt(x,0)=0

// ANSWER = (x+0.2)\*sin(M\_PI\*x/2)\*cos(M\_PI\*t)

#include "Hyper.h"

#include <iostream>

#include <algorithm>

#include <iomanip>

#include <fstream>

func2 answ = [](double t, double x)->double{ return (x + 0.2)\*sin(M\_PI\*x / 2)\*cos(M\_PI\*t); };

int Test()

{

size\_t const Nt = 1000;

size\_t const Nl = 1000;

func2 F = [](double t, double x)->double{ return -M\_PI\*cos(M\_PI\*t)\*(cos(M\_PI\*x / 2) + 3 / 4 \* M\_PI\*(x + 0.2)\*sin(M\_PI\*x / 2)); };

func u0 = [](double x)->double{ return (x + 0.2)\*sin(M\_PI\*x / 2); };

func u1 = [](double t)->double{ return 0.; };

func u2 = [](double t)->double{ return 1.2\*cos(M\_PI\*t); };

func v0 = [](double x)->double{ return 0.; };

double period = 2, length = 1;

double h = length / Nl;

double dt = period / Nt;

std::vector<std::vector<double>> res = Hyper::Process(0.75, F, u0, u1, u2, v0, Nt, Nl, period, length);

std::ofstream out("output.txt");

out << std::left;

/\*{

for (size\_t k = 0; k < Nt + 1; k++)

{

double m = 0;

for (size\_t n = 0; n < Nl + 1; n++)

{

double a1 = answ(k\*dt, n\*h);

double a2 = res[k][n];

double err = abs(a1-a2);

m = (m\*n + err) / (n + 1);

}

out << std::setw(4) << k\*dt << '\t'

<< std::setw(12) << m << std::endl;

}

}\*/

//{

// for (size\_t k = 0; k < Nt + 1; k++)

// {

// double m = 0;

// for (size\_t n = 0; n < Nl + 1; n++)

// if (abs(res[k][n]) > m)

// m = abs(res[k][n]);

// out << std::setw(4) << k\*dt << '\t'

// << std::setw(12) << m << std::endl;

// }

//}

/\*{

for (size\_t n = 0; n < Nl + 1; n++)

out << std::setw(4) << n\*h << '\t'

<< std::setw(12) << res[Nt][n] << std::endl;

}\*/

{

for (size\_t n = 0; n < Nl + 1; n++)

out << std::setw(4) << n\*h << '\t'

<< std::setw(12) << answ(Nt\*dt, n\*h) - res[Nt][n] << std::endl;

}

out << std::endl;

return 0;

}

int main()

{

return Test();

}

**Результати роботи програми**