Московский Авиационный Институт (национальный исследовательский университет)

**Факультет прикладной математики и физики**

Кафедра Вычислительной математики и программирования

**Лабораторная работа по курсу**

**«Численные методы»**

Студент: Богачев П.С.

Группа: М8О-402Б-20

Руководитель: Пивоваров Д. Е.

Москва 2023

**Лабораторная работа №6**

Используя явную схему крест и неявную схему, решить начально-краевую задачу для дифференциального уравнения гиперболического типа. Аппроксимацию второго начального условия произвести с первым и со вторым порядком. Осуществить реализацию трех вариантов аппроксимации граничных условий, содержащих производные: двухточечная аппроксимация с первым порядком, трехточечная аппроксимация со вторым порядком, двухточечная аппроксимация со вторым порядком. В различные моменты времени вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением . Исследовать зависимость погрешности от сеточных параметров .

Вариант №3

,



,

.

Аналитическое решение: 

**Код программы**

#include <iostream>

#include <fstream>

#include <vector>

#include <cmath>

double phi0(double t) {

return sin(t \* 2.);

}

double phi1(double t) {

return -sin(t \* 2.);

}

double psi0(double x) {

return 0.;

}

double psi1(double x) {

return cos(x) \* 2.;

}

double solution(double x, double t) {

return cos(x) \* sin(t \* 2);

}

std::vector<std::vector<double>> analytical\_solution(double x\_begin, double x\_end, double t\_begin,

double t\_end, double h, double sigma) {

double tau = sigma \* h \* h;

int length\_x = static\_cast<int>((x\_end - x\_begin) / h) + 1;

int length\_t = static\_cast<int>((t\_end - t\_begin) / tau) + 1;

std::vector<double> x(length\_x), t(length\_t);

x[0] = x\_begin;

for (int i = 1; i < length\_x; ++i) {

x[i] = x[i - 1] + h;

}

t[0] = t\_begin;

for (int i = 1; i < length\_x; ++i) {

t[i] = t[i - 1] + tau;

}

std::vector<std::vector<double>> result(length\_t, std::vector<double>(length\_x));

for (int i = 0; i < length\_x; ++i) {

for (int j = 0; j < length\_t; ++j) {

result[j][i] = solution(x[i], t[j]);

}

}

return result;

}

std::vector<std::vector<double>> explicit\_finite\_difference\_method(double x\_begin, double x\_end,

double t\_begin, double t\_end, double h, double sigma) {

double tau = sigma \* h \* h;

int length\_x = static\_cast<int>((x\_end - x\_begin) / h) + 1;

int length\_t = static\_cast<int>((t\_end - t\_begin) / tau) + 1;

std::vector<double> x(length\_x), t(length\_t);

x[0] = x\_begin;

for (int i = 1; i < length\_x; ++i) {

x[i] = x[i - 1] + h;

}

t[0] = t\_begin;

for (int i = 1; i < length\_x; ++i) {

t[i] = t[i - 1] + tau;

}

std::vector<std::vector<double>> result(length\_t, std::vector<double>(length\_x));

for (int i = 0; i < length\_x; ++i) {

result[0][i] = psi0(x[i]);

result[1][i] = psi0(x[i]) + tau \* psi1(x[i]);

}

for (int i = 2; i < length\_t; ++i) {

result[i][0] = phi0(t[i]);

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

result[i][j] = sigma \* (result[i - 1][j + 1] - result[i - 1][j] \* 2. + result[i - 1][j - 1]) +

(2. - tau \* tau \* 3.) \* result[i - 1][j] - result[i - 2][j];

}

result[i][length\_x - 1] = phi1(t[i]);

}

return result;

}

std::vector<double> tridiagonal\_solve(std::vector<std::vector<double>>& A, std::vector<double>& b) {

int n = A.size();

std::vector<double> v(n), u(n), x(n);

v[0] = A[0][1] / -A[0][0];

u[0] = b[0] / A[0][0];

for (int i = 1; i < n - 1; ++i) {

v[i] = A[i][i + 1] / (-A[i][i] - A[i][i - 1] \* v[i - 1]);

u[i] = (A[i][i - 1] \* u[i - 1] - b[i]) / (-A[i][i] - A[i][i - 1] \* v[i - 1]);

}

u[n - 1] = (A[n - 1][n - 2] \* u[n - 2] - b[n - 1]) / (-A[n - 1][n - 1] - A[n - 1][n - 2] \* v[n - 2]);

x[n - 1] = u[n - 1];

for (int i = n - 1; i > 0; --i) {

x[i - 1] = v[i - 1] \* x[i] + u[i - 1];

}

return x;

}

std::vector<std::vector<double>> implicit\_finite\_difference\_method(double x\_begin, double x\_end,

double t\_begin, double t\_end, double h, double sigma) {

double tau = sigma \* h \* h;

int length\_x = static\_cast<int>((x\_end - x\_begin) / h) + 1;

int length\_t = static\_cast<int>((t\_end - t\_begin) / tau) + 1;

std::vector<double> x(length\_x), t(length\_t);

x[0] = x\_begin;

for (int i = 1; i < length\_x; ++i) {

x[i] = x[i - 1] + h;

}

t[0] = t\_begin;

for (int i = 1; i < length\_x; ++i) {

t[i] = t[i - 1] + tau;

}

std::vector<std::vector<double>> result(length\_t, std::vector<double>(length\_x));

for (int i = 0; i < length\_x; ++i) {

result[0][i] = psi0(x[i]);

result[1][i] = psi0(x[i]) + tau \* psi1(x[i]);

}

for (int i = 2; i < length\_t; ++i) {

std::vector<std::vector<double>> A(length\_x - 2, std::vector<double>(length\_x - 2));

A[0][0] = -(tau \* tau \* 3 + sigma \* 2. + 1.);

A[0][1] = sigma;

for (int j = 1; j < A.size() - 1; ++j) {

A[j][j - 1] = sigma;

A[j][j] = -(tau \* tau \* 3 + sigma \* 2. + 1.);

A[j][j + 1] = sigma;

}

A[length\_x - 3][length\_x - 4] = sigma;

A[length\_x - 3][length\_x - 3] = -(tau \* tau \* 3 + sigma \* 2. + 1.);

std::vector<double> b(result[i - 1].size() - 2);

for (int j = 0; j < b.size(); ++j) {

b[j] = result[i - 2][j + 1] - result[i - 1][j + 1] \* 2.;

}

b[0] -= sigma \* phi0(t[i]);

b[b.size() - 1] -= sigma \* phi1(t[i]);

result[i][0] = phi0(t[i]);

std::vector<double> interior = tridiagonal\_solve(A, b);

std::move(interior.begin(), interior.end(), result[i].begin() + 1);

result[i][result[i].size() - 1] = phi1(t[i]);

}

return result;

}

double max\_abs\_error(std::vector<std::vector<double>>& A, std::vector<std::vector<double>>& B) {

int n = A.size(), m = A[0].size();

double max = 0.;

for (int i = 0; i < n; ++i) {

for (int j = 0; j < m; ++j) {

max = std::max(max, std::abs(A[i][j] - B[i][j]));

}

}

return max;

}

double mean\_abs\_error(std::vector<std::vector<double>>& A, std::vector<std::vector<double>>& B) {

int n = A.size(), m = A[0].size();

double mean = 0., prod = static\_cast<double>(n \* m);

for (int i = 0; i < n; ++i) {

for (int j = 0; j < m; ++j) {

mean += std::abs(A[i][j] - B[i][j]) / prod;

}

}

return mean;

}

void output\_to\_file(std::string filepath, const std::vector<std::vector<double>>& arr) {

std::ofstream fout(filepath);

int n = arr.size(), m = arr[0].size();

for (int i = 0; i < n; ++i) {

for (int j = 0; j < m; ++j) {

fout << arr[i][j] << " ";

}

fout << "\n";

}

fout.close();

}

int main()

{

double x\_begin = 0., x\_end = acos(-1), t\_begin = 0., t\_end = 5., h = 0.01, sigma = 1;

std::vector<std::vector<double>> as = analytical\_solution(x\_begin, x\_end, t\_begin, t\_end, h, sigma);

std::vector<std::vector<double>> efdm = explicit\_finite\_difference\_method(x\_begin, x\_end, t\_begin, t\_end, h, sigma);

std::vector<std::vector<double>> ifdm = implicit\_finite\_difference\_method(x\_begin, x\_end, t\_begin, t\_end, h, sigma);

std::cout << "Max abs error between analytical solution and explicit finite difference method solution: " << max\_abs\_error(as, efdm) << "\n";

std::cout << "Mean abs error between analytical solution and explicit finite difference method solution: " << mean\_abs\_error(as, efdm) << "\n";

std::cout << "Max abs error between analytical solution and implicit finite difference method solution: " << max\_abs\_error(as, ifdm) << "\n";

std::cout << "Mean abs error between analytical solution and implicit finite difference method solution: " << mean\_abs\_error(as, ifdm) << "\n";

output\_to\_file("analytical\_solution.txt", as);

output\_to\_file("explicit\_finite\_difference\_method.txt", efdm);

output\_to\_file("implicit\_finite\_difference\_method.txt", ifdm);

return 0;

}

Вывод программы

График функций:

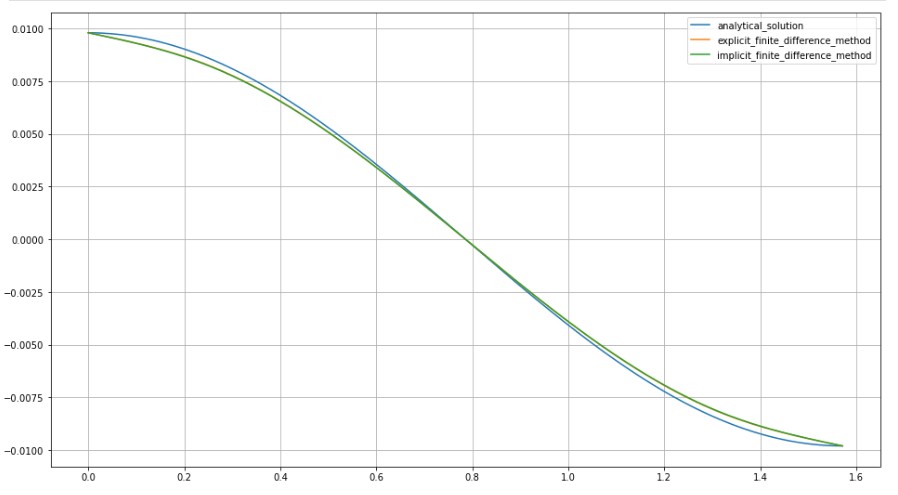
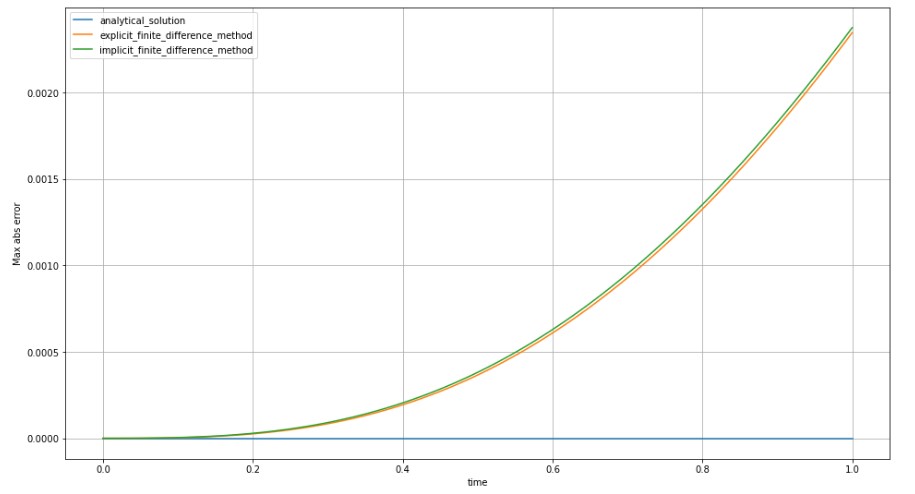


График ошибок:



**Вывод**

В лабораторной работе решена начально-краевая задача для дифференциального уравнение гиперболического типа. Произведена аппроксимация второго начального условия с первым и вторым порядком. Реализованы три метода аппроксимации граничных условий: двухточечная аппроксимация с первым порядком, трехточечная аппроксимация со вторым порядком, двухточечная аппроксимация со вторым порядком. Вычислена погрешность решения.