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 №8**

Используя схемы переменных направлений и дробных шагов, решить двумерную начально-краевую задачу для дифференциального уравнения параболического типа. 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 y, double t, double a) {

return (exp(y) + exp(-y)) / 2. \* exp(-a \* t \* 3.);

}

double phi1(double y, double t, double a) {

return 0.;

}

double phi2(double x, double t, double a) {

return cos(x \* 2.) \* exp(-a \* t \* 3.);

}

double phi3(double x, double t, double a) {

return cos(x \* 2.) \* exp(-a \* t \* 3.) \* 1.25;

}

double psi(double x, double y) {

return cos(x \* 2.) \* (exp(y) + exp(-y)) / 2.;

}

double solution(double x, double y, double t, double a) {

return cos(x \* 2.) \* (exp(y) + exp(-y)) / 2. \* exp(-a \* t \* 3.);

}

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

double t\_begin, double t\_end, double hx, double hy, double tau, double a) {

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

int length\_y = static\_cast<int>((y\_end - y\_begin) / hy) + 1;

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

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

x[0] = x\_begin;

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

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

}

y[0] = y\_begin;

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

y[i] = y[i - 1] + hy;

}

t[0] = t\_begin;

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

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

}

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

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

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

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

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

}

}

}

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<std::vector<double>>> variable\_directions\_method(double x\_begin, double x\_end,

double y\_begin, double y\_end, double t\_begin, double t\_end, double hx, double hy, double tau, double a) {

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

int length\_y = static\_cast<int>((y\_end - y\_begin) / hy) + 1;

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

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

x[0] = x\_begin;

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

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

}

y[0] = y\_begin;

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

y[i] = y[i - 1] + hy;

}

t[0] = t\_begin;

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

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

}

std::vector<std::vector<std::vector<double>>> result(length\_t, std::vector<std::vector<double>>(length\_x, std::vector<double>(length\_y, 0)));

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

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

result[0][i][j] = psi(x[i], y[j]);

}

}

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

std::vector<std::vector<double>> U(length\_x, std::vector<double>(length\_y));

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

result[i][j][0] = phi2(x[j], t[i], a);

result[i][j][length\_y - 1] = phi3(x[j], t[i], a);

U[j][0] = phi2(x[j], t[i] - tau / 2., a);

U[j][length\_y - 1] = phi3(x[j], t[i] - tau / 2., a);

}

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

result[i][0][j] = phi0(y[j], t[i], a);

result[i][length\_x - 1][j] = phi1(y[j], t[i], a);

U[0][j] = phi0(y[j], t[i] - tau / 2., a);

U[length\_x - 1][j] = phi1(y[j], t[i] - tau / 2., a);

}

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

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

A[0][0] = hx \* hx \* hy \* hy \* 2. + a \* tau \* hy \* hy \* 2.;

A[0][1] = -a \* tau \* hy \* hy;

for (int k = 1; k < length\_x - 3; ++k) {

A[k][k - 1] = -a \* tau \* hy \* hy;

A[k][k] = hx \* hx \* hy \* hy \* 2. + a \* tau \* hy \* hy \* 2.;

A[k][k + 1] = -a \* tau \* hy \* hy;

}

A[length\_x - 3][length\_x - 4] = -a \* tau \* hy \* hy;

A[length\_x - 3][length\_x - 3] = hx \* hx \* hy \* hy \* 2. + a \* tau \* hy \* hy \* 2.;

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

for (int k = 0; k < length\_x - 2; ++k) {

b[k] = result[i - 1][k + 1][j - 1] \* a \* tau \* hx \* hx + result[i - 1][k + 1][j] \*

(hx \* hx \* hy \* hy \* 2. - a \* tau \* hx \* hx \* 2) + result[i - 1][k + 1][j + 1] \* a \* tau \* hx \* hx;

}

b[0] -= (-a \* tau \* hy \* hy) \* phi0(y[j], t[i] - tau / 2., a);

b[length\_x - 3] -= (-a \* tau \* hy \* hy) \* phi1(y[j], t[i] - tau / 2., a);

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

for (int k = 0; k < length\_x - 2; ++k) {

U[k + 1][j] = interior[k];

}

}

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

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

A[0][0] = hx \* hx \* hy \* hy \* 2. + a \* tau \* hx \* hx \* 2.;

A[0][1] = -a \* tau \* hx \* hx;

for (int k = 1; k < length\_y - 3; ++k) {

A[k][k - 1] = -a \* tau \* hx \* hx;

A[k][k] = hx \* hx \* hy \* hy \* 2. + a \* tau \* hx \* hx \* 2.;

A[k][k + 1] = -a \* tau \* hx \* hx;

}

A[length\_y - 3][length\_y - 4] = -a \* tau \* hx \* hx;

A[length\_y - 3][length\_y - 3] = hx \* hx \* hy \* hy \* 2. + a \* tau \* hx \* hx \* 2.;

std::vector<double> b(length\_y - 2);

for (int k = 0; k < length\_y - 2; ++k) {

b[k] = U[j - 1][k + 1] \* a \* tau \* hy \* hy + U[j][k + 1] \* (hx \* hx \* hy \* hy \* 2. - a \* tau \* hy \* hy \* 2.)

+ U[j + 1][k + 1] \* a \* tau \* hy \* hy;

}

b[0] -= (-a \* tau \* hx \* hx) \* phi2(x[j], t[i], a);

b[length\_y - 3] -= (-a \* tau \* hx \* hx) \* phi3(x[j], t[i], a);

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

for (int k = 0; k < length\_y - 2; ++k) {

result[i][j][k + 1] = interior[k];

}

}

}

return result;

}

std::vector<std::vector<std::vector<double>>> fractional\_steps\_method(double x\_begin, double x\_end,

double y\_begin, double y\_end, double t\_begin, double t\_end, double hx, double hy, double tau, double a) {

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

int length\_y = static\_cast<int>((y\_end - y\_begin) / hy) + 1;

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

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

x[0] = x\_begin;

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

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

}

y[0] = y\_begin;

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

y[i] = y[i - 1] + hy;

}

t[0] = t\_begin;

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

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

}

std::vector<std::vector<std::vector<double>>> result(length\_t, std::vector<std::vector<double>>(length\_x, std::vector<double>(length\_y, 0)));

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

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

result[0][i][j] = psi(x[i], y[j]);

}

}

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

std::vector<std::vector<double>> U(length\_x, std::vector<double>(length\_y));

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

result[i][j][0] = phi2(x[j], t[i], a);

result[i][j][length\_y - 1] = phi3(x[j], t[i], a);

U[j][0] = phi2(x[j], t[i] - tau / 2., a);

U[j][length\_y - 1] = phi3(x[j], t[i] - tau / 2., a);

}

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

result[i][0][j] = phi0(y[j], t[i], a);

result[i][length\_x - 1][j] = phi1(y[j], t[i], a);

U[0][j] = phi0(y[j], t[i] - tau / 2., a);

U[length\_x - 1][j] = phi1(y[j], t[i] - tau / 2., a);

}

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

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

A[0][0] = hx \* hx + a \* tau \* 2.;

A[0][1] = -a \* tau;

for (int k = 1; k < length\_x - 3; ++k) {

A[k][k - 1] = -a \* tau;

A[k][k] = hx \* hx + a \* tau \* 2.;

A[k][k + 1] = -a \* tau;

}

A[length\_x - 3][length\_x - 4] = -a \* tau;

A[length\_x - 3][length\_x - 3] = hx \* hx + a \* tau \* 2.;

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

for (int k = 0; k < length\_x - 2; ++k) {

b[k] = result[i - 1][k + 1][j] \* hx \* hx;

}

b[0] -= (-a \* tau) \* phi0(y[j], t[i] - tau / 2., a);

b[length\_x - 3] -= (-a \* tau) \* phi1(y[j], t[i] - tau / 2., a);

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

for (int k = 0; k < length\_x - 2; ++k) {

U[k + 1][j] = interior[k];

}

}

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

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

A[0][0] = hy \* hy + a \* tau \* 2.;

A[0][1] = -a \* tau;

for (int k = 1; k < length\_y - 3; ++k) {

A[k][k - 1] = -a \* tau;

A[k][k] = hy \* hy + a \* tau \* 2.;

A[k][k + 1] = -a \* tau;

}

A[length\_y - 3][length\_y - 4] = -a \* tau;

A[length\_y - 3][length\_y - 3] = hy \* hy + a \* tau \* 2.;

std::vector<double> b(length\_y - 2);

for (int k = 0; k < length\_y - 2; ++k) {

b[k] = U[j][k + 1] \* hy \* hy;

}

b[0] -= (-a \* tau) \* phi2(x[j], t[i], a);

b[length\_y - 3] -= (-a \* tau) \* phi3(x[j], t[i], a);

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

for (int k = 0; k < length\_y - 2; ++k) {

result[i][j][k + 1] = interior[k];

}

}

}

return result;

}

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

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

double max = 0.;

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

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

for (int k = 0; k < l; ++k) {

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

}

}

}

return max;

}

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

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

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

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

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

for (int k = 0; k < l; ++k) {

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

}

}

}

return mean;

}

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

std::ofstream fout(filepath);

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

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

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

for (int k = 0; k < l; ++k) {

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

}

fout << "\n";

}

}

fout.close();

}

int main()

{

double x\_begin = 0., x\_end = acos(-1) / 4., y\_begin = 0., y\_end = log(2), t\_begin = 0., t\_end = 1.,

hx = 0.01, hy = 0.01, tau = 0.01, a = 1.;

std::vector<std::vector<std::vector<double>>> as = analytical\_solution(x\_begin, x\_end, y\_begin, y\_end, t\_begin, t\_end, hx, hy, tau, a);

std::vector<std::vector<std::vector<double>>> vdm = variable\_directions\_method(x\_begin, x\_end, y\_begin, y\_end, t\_begin, t\_end, hx, hy, tau, a);

std::vector<std::vector<std::vector<double>>> fsm = fractional\_steps\_method(x\_begin, x\_end, y\_begin, y\_end, t\_begin, t\_end, hx, hy, tau, a);

std::cout << "Max abs error between analytical solution and variable directions method solution: " << max\_abs\_error(as, vdm) << "\n";

std::cout << "Mean abs error between analytical solution and variable directions method solution: " << mean\_abs\_error(as, vdm) << "\n";

std::cout << "Max abs error between analytical solution and fractional steps method solution: " << max\_abs\_error(as, fsm) << "\n";

std::cout << "Mean abs error between analytical solution and fractional steps method solution: " << mean\_abs\_error(as, fsm) << "\n";

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

output\_to\_file("variable\_directions\_method.txt", vdm);

output\_to\_file("fractional\_steps\_method.txt", fsm);

return 0;

}

Вывод программы

График функций:

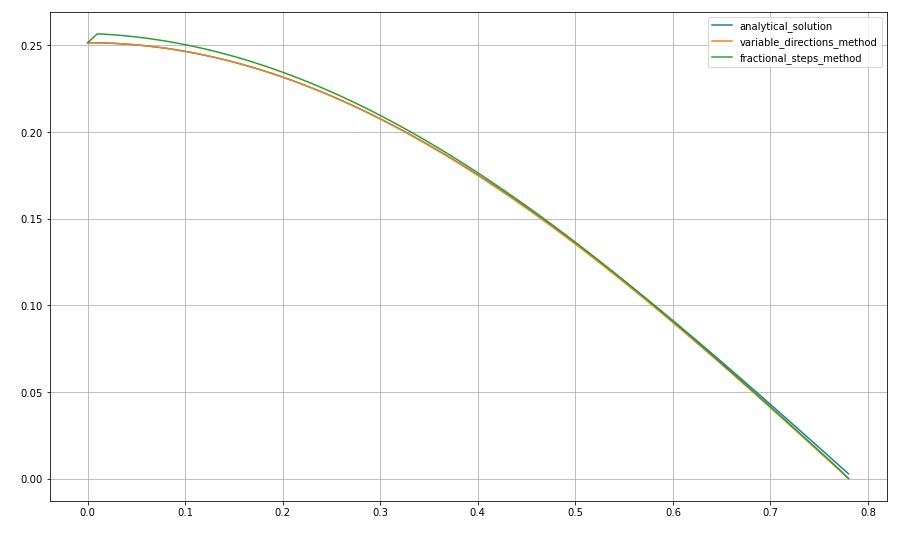
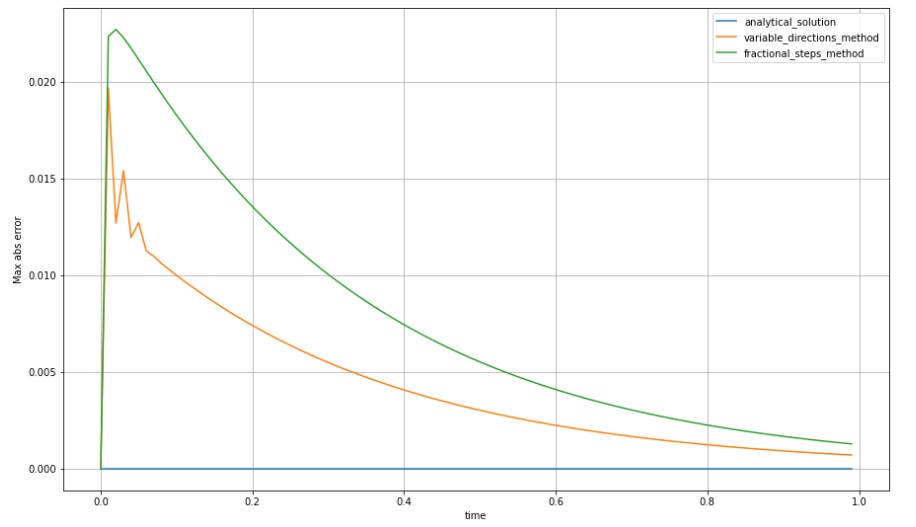


График ошибок:



**Вывод**

В лабораторной работе решена двумерная начально-краевая задача для дифференциального уравнение параболического типа. Вычислена погрешность численного решения от аналитического в зависимости от времени.