|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |
| --- | --- |

ФАКУЛЬТЕТ Информатики и систем управления

КАФЕДРА Теоретической информатики и компьютерных технологий

**ЛАБОРАТОРНАЯ РАБОТА № 4**

**ПО КУРСУ:**

***«Численные методы»***

Студент *Виленский С.Д.*

Преподаватель *Домрачева А.Б.*

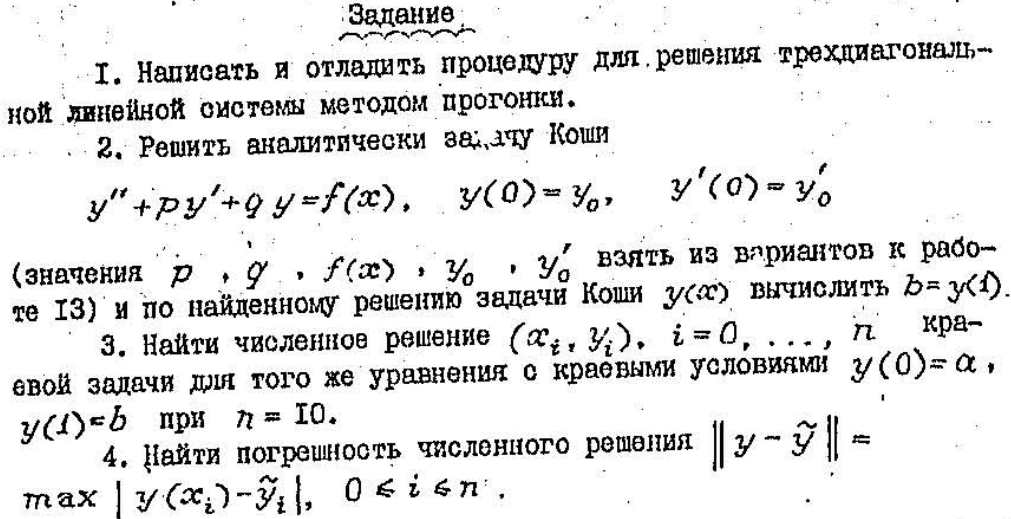
*Москва, 2024 г.*

**ОГЛАВЛЕНИЕ**

[1. Постановка задачи 3](#_heading=h.30j0zll)

[2. Практическая реализация 4](#_heading=h.1fob9te)

# 1. Постановка задачи



# 2. Практическая реализация

#include <cmath>

#include <numbers>

#include <iostream>

#include <iomanip>

#include "../lab1/lab1.cpp"

// *lab 14*

// *test:*

// *y = eˣ, x ∈ [0, 1], y(0)=e⁰=1, y(1)=e*

// *y"(x) + 5y'(x) - 4y(x) = 2eˣ*

#define N 10

auto main() -> int {

auto y = [](double x){return std::pow(std::numbers::e, x);};

double a = y(0);

double b = y(1);

// *y"(x) + p(x)\*y'(x) + q(x)\*y(x) = f(x)*

auto p\_x = [](double x){return 5;};

auto q\_x = [](double x){return -4;};

auto f\_x = [](double x){return 2 \* std::pow(std::numbers::e, x);};

double h = 1. / N;

auto x\_i = [&](std::size\_t i){return i \* h;};

auto p\_i = [&](std::size\_t i){return p\_x(x\_i(i));};

auto q\_i = [&](std::size\_t i){return q\_x(x\_i(i));};

auto f\_i = [&](std::size\_t i){return f\_x(x\_i(i));};

ThreeDiagonalMatrix<double, N - 1> SLAE{};

Vector<double, N - 1> SLAEright{};

for (std::size\_t i = 1; i != N; ++i) {

if (i != 1) {

SLAE.at(i - 1, i - 2) = 1 - h / 2 \* p\_i(i);

}

if (i != N - 1) {

SLAE.at(i - 1, i) = 1 + h / 2 \* p\_i(i);

}

SLAE.at(i - 1, i - 1) = h \* h \* q\_i(i) - 2;

SLAEright.at(i - 1) = h \* h \* f\_i(i);

}

SLAEright.at(0) -= a \* (1 - h / 2 \* p\_i(1));

SLAEright.at(N - 2) -= b \* (1 + h / 2 \* p\_i(N - 1));

Vector<double, N - 1> SLAEsolve = findSolve(SLAE, SLAEright);

Vector<double, N + 1> y\_i{};

y\_i.at(0) = a;

y\_i.at(N) = b;

for (std::size\_t i = 1; i != N; ++i) {

y\_i.at(i) = SLAEsolve.at(i - 1);

}

// *||y - y˜|| = max|y(xᵢ) - yᵢ|*

// *double solve\_error = 0;*

std::cout

<< std::setw(4) << " i"

<< std::setw(12) << " y(x\_i)"

<< std::setw(12) << " y\_i"

<< std::setw(16) << " |y(x\_i) - y\_i|" << '\n';

std::cout << std::setw(45) << std::setfill('-') << '\n';

std::cout << std::setfill(' ');

for (std::size\_t i = 0; i != N + 1; ++i) {

double error\_i = std::abs(y(x\_i(i)) - y\_i.at(i));

// *solve\_error = std::max(solve\_error, error\_i);*

std::cout

<< std::setw(4) << i

<< std::setw(12) << y(x\_i(i))

<< std::setw(12) << y\_i.at(i)

<< std::setw(16) << error\_i << '\n';

}

return 0;

}