|  | **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 № 1.1**

**ПО КУРСУ:**

***«Численные методы»***

Студент *Виленский С.Д.*

Преподаватель *Домрачева А.Б.*

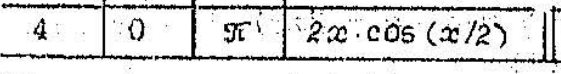
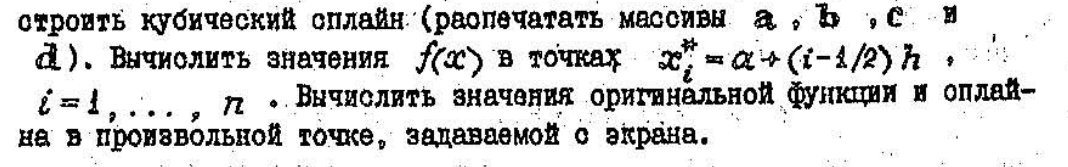
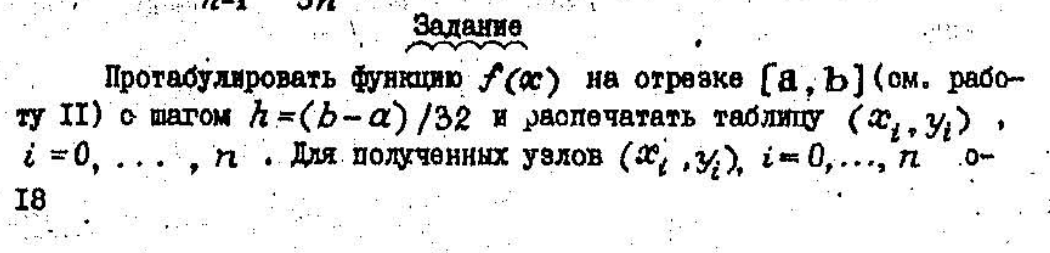
*Москва, 2024 г.*

**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 "../lab1/lab1.cpp"

// *Вариант 4. [0, π]. f(x) = 2x\*cos(x/2)*

#define LOOKAHEAD 1

#define N (10 + LOOKAHEAD)

auto f(double x) -> double {

return 2 \* x \* std::cos(x / 2);

}

auto main() -> int {

double a = 0, b = std::numbers::pi;

double h = (b - a) / (N - LOOKAHEAD);

b += h \* LOOKAHEAD;

Matrix<double, N + 1, 2> table{};

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

table.at(i, 0) = a + h \* i;

table.at(i, 1) = f(table.at(i, 0));

}

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

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

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

if (i != N - 1) {

SLAE.at(i, i - 1) = SLAE.at(i - 1, i) = 1;

}

SLAE.at(i - 1, i - 1) = 4;

SLAEright.at(i - 1) =

(table.at(i + 1, 1) - 2 \* table.at(i, 1) + table.at(i - 1, 1))

/ (h \* h);

}

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

Vector<double, N + 1> A, B, C, D{};

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

C.at(i) = SLAEsolve.at(i - 1);

}

C.at(0) = C.at(N) = 0;

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

A.at(i) = table.at(i, 1);

B.at(i) = (table.at(i + 1, 1) - table.at(i, 1)) / h -

(h / 3) \* (C.at(i + 1) + 2 \* C.at(i));

D.at(i) = (C.at(i + 1) - C.at(i)) / (3 \* h);

}

A.at(N - 1) = table.at(N - 1, 1);

B.at(N - 1) = (table.at(N, 1) - table.at(N - 1, 1)) \* C.at(N - 1) \* 2 / 3;

D.at(N - 1) = - C.at(N) / (3 \* h);

auto spline = [&](double x) {

int i = (x - a) \* N / (b - a);

double delta\_x = x - table.at(i, 0);

double res = D.at(i);

res = res \* delta\_x + C.at(i);

res = res \* delta\_x + B.at(i);

res = res \* delta\_x + A.at(i);

return res;

};

std::cout << "x\t\tS(x)\t\ty(x)\t\t|S(x)-y(x)|\n";

for (std::size\_t i = 0; i != 2 \* (N - LOOKAHEAD) + 1; ++i) {

double x = a + h \* i / 2 + .0001;

double s = spline(x);

double y = f(x);

std::cout << x << " \t";

std::cout << s << " \t";

std::cout << y << " \t";

std::cout << abs(s - y) << '\n';

}

}

# 3. Вывод

x S(x) y(x) |S(x)-y(x)|

0.0001 0.000198357 0.0002 1.64269e-06

0.15718 0.311455 0.313389 0.00193417

0.314259 0.620769 0.620776 6.49168e-06

0.471339 0.910898 0.916621 0.00572231

0.628419 1.19529 1.1953 1.1074e-05

0.785498 1.44211 1.45138 0.00927531

0.942578 1.67963 1.67964 1.51991e-05

1.09966 1.86272 1.87517 0.0124449

1.25674 2.03335 2.03337 1.86937e-05

1.41382 2.13496 2.15006 0.0150984

1.5709 2.22145 2.22147 2.14014e-05

1.72798 2.22722 2.24433 0.017112

1.88506 2.21584 2.21586 2.32114e-05

2.04214 2.11543 2.13385 0.0184182

2.19921 1.99663 1.99665 2.3976e-05

2.35629 1.78437 1.80321 0.0188469

2.51337 1.55309 1.55311 2.38401e-05

2.67045 1.22782 1.24655 0.0187347

2.82753 0.884346 0.884368 2.20251e-05

2.98461 0.451381 0.468043 0.0166618

3.14169 2.96344e-05 -0.000314169 0.000343804

В результате выполнения данной лабораторной работы был приобретен навык построения кубического сплайна по заданным узлам интерполяции, выбранным с равным шагом.