Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

—

Институт компьютерных наук и технологий

Высшая школа программной инженерии

**Курсовая работа**

**по дисциплине «Вычислительная математика»**

Выполнил

Студент группы 5130904/20004 Шелковников Д.С.

Преподаватель Устинов С.М.

Оглавление

[Задание 2](#_Toc160965485)

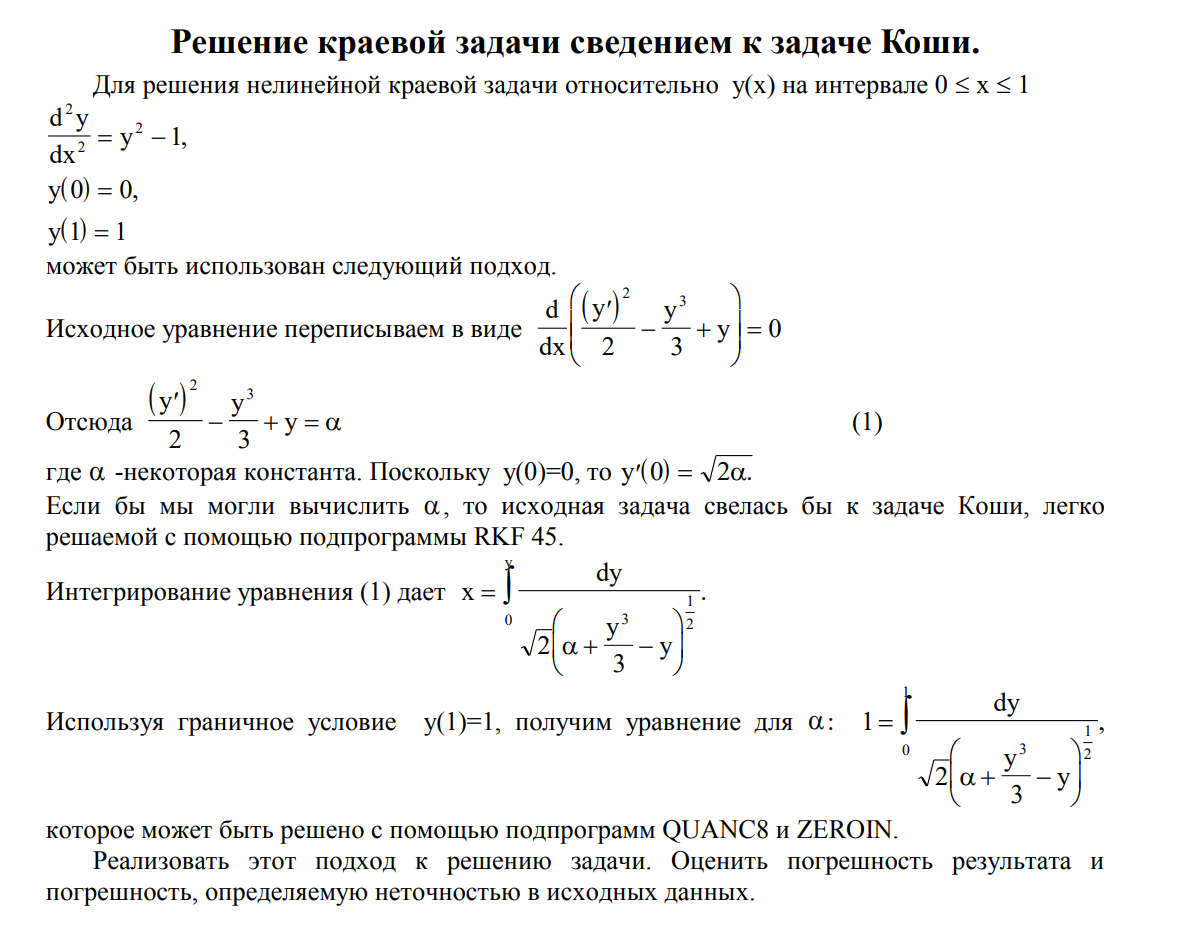
[Результаты 3](#_Toc160965486)

[Вывод 7](#_Toc160965487)

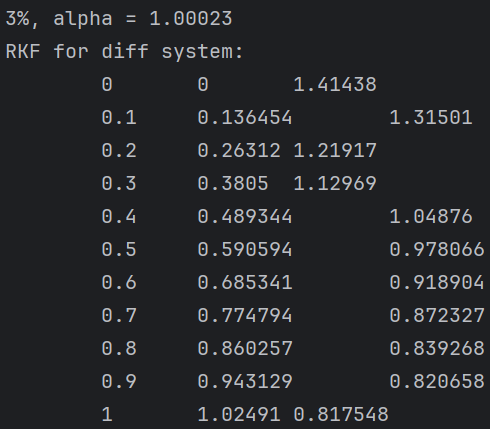
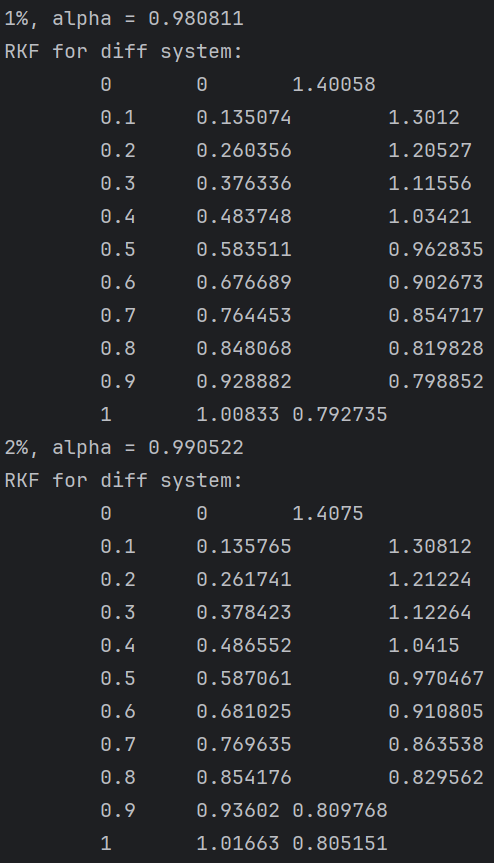
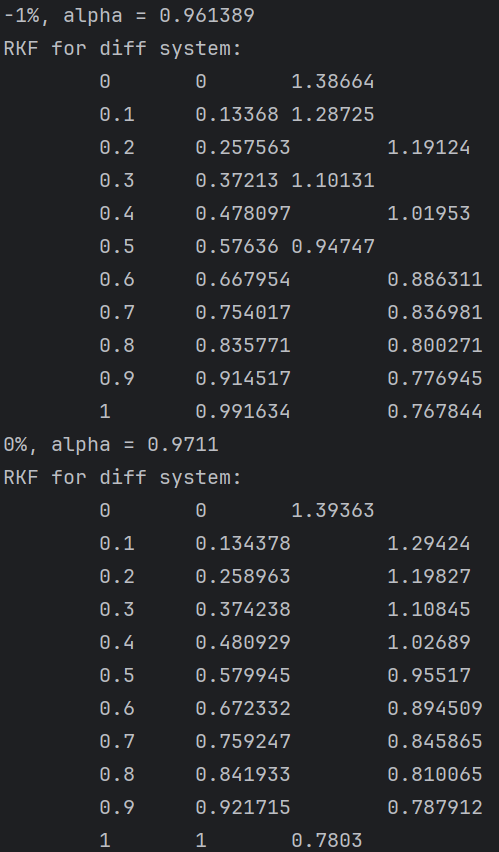
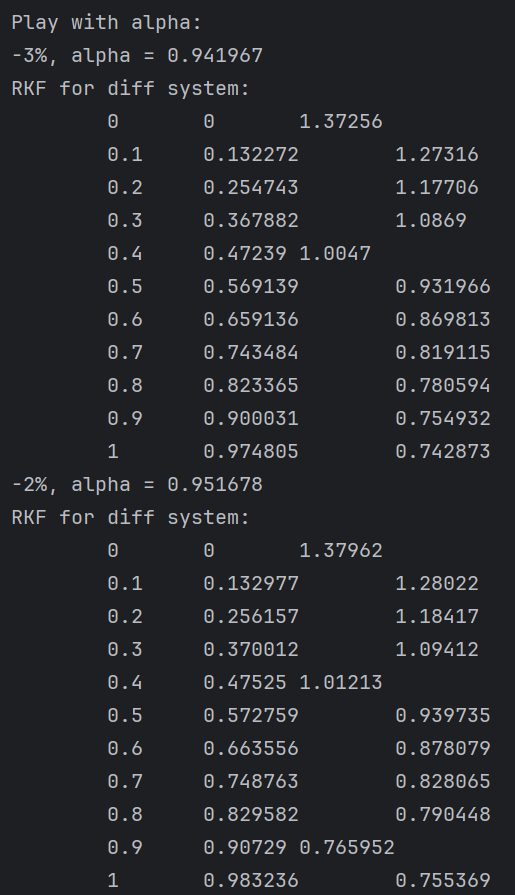
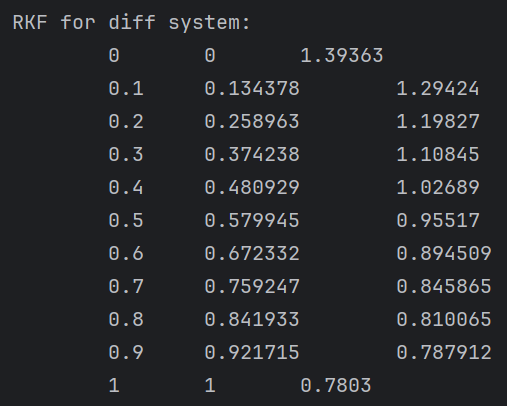
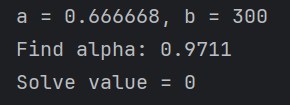
[Код программы 8](#_Toc160965488)

[<DIR>/computational\_mathematics/coursework/main.cpp 8](#_Toc160965489)

# Задание



# Результаты



# Вывод

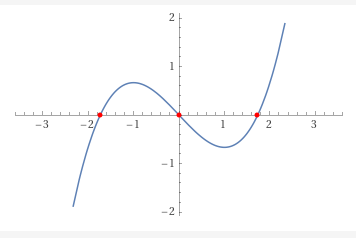
Определим сначала интервалы для поиска решения. 

График знаменателя (подкорневого выражения) имеет такой вид. По условию интегрирования нас интересует часть от 0 до 1. Решим неравенство, чтобы часть от 0 до 1 была неотрицательной. Это выполнится при добавлении . В таком случае знаменатель будет обращаться в 0, что нам не подходит. Тогда в коде программы укажем минимальную границу для поиска решения, как 2.0/3+0.000001, а верхнюю без ограничений. Дальше с помощью программы ZEROIN найдем решение.

Для решения дифференциального уравнения введем замену dy = x и получим систему из двух дифференциальных уравнений, где начальные параметры: 0 и , где α мы нашли ранее.

Проанализируем систему дифференциальное уравнение на устойчивость. При изменении на 1% в большую или меньшую сторону, решение уравнения изменяется на 0.833% в соответствующую сторону. При изменении на 2% в большую или меньшую сторону, решение уравнения изменяется на 1.663% в соответствующую сторону. При изменении на 3% в большую или меньшую сторону, решение уравнения изменяется на 2.491% в соответствующую сторону. Можно сделать вывод, что система не устойчивая, потому что при изменении параметра , решение изменяется примерно на такое же значение в процентах.

# Код программы

## <DIR>/computational\_mathematics/coursework/main.cpp

#include <iostream>

#include <cmath>

#include <functional>

#include "../common/Quanc8.h"

#include "../common/Rkf45.h"

#include "zeroin.h"

double integrand(double y, double alpha) {

return 1.0 / sqrt(2 \* (alpha + pow(y, 3) / 3 - y));

}

int diff\_func(int n, double t, double \*x, double \*dx) {

dx[0] = x[1];

dx[1] = x[0] \* x[0] - 1;

return 0;

}

double to\_solve(double alpha) {

double a = 0, b = 1;

double abserr = 1e-6, relerr = 1e-6;

auto func = std::bind(integrand, std::placeholders::\_1, alpha);

dimkashelk::Quanc8 quanc8(func, a, b, abserr, relerr);

return 1 - quanc8.getResult();

}

double find\_alpha(double a, double b) {

double tol = 1e-12;

int flag = 0;

double res = zeroin(a, b, to\_solve, tol, &flag);

return res;

}

int main() {

double a = 2.0 / 3 + 0.000001, b = 300;

double alpha = find\_alpha(a, b);

std::cout << "a = " << a << ", b = " << b << "\nFind alpha: " << alpha << "\nSolve value = " << to\_solve(alpha) << "\n\n";

std::cout << "RKF for diff system:\n";

double data[2]{0.0, std::sqrt(2 \* alpha)};

dimkashelk::Rkf45::calculate(diff\_func, data, 0, 1);

std::cout << "\n\nPlay with alpha: \n";

for (double i = 0.97; i < 1.04; i += 0.01) {

double new\_alpha = alpha \* i;

std::cout << (i - 1) \* 100 << "%, alpha = " << new\_alpha << "\n";

std::cout << "RKF for diff system:\n";

data[0] = 0.0;

data[1] = std::sqrt(2 \* new\_alpha);

dimkashelk::Rkf45::calculate(diff\_func, data, 0, 1);

}

return 0;

}

В программе также использовались файлы Quanc8.cpp и Quanc8.h, Rkf45.cpp и Rkf45.h из предыдущих лабораторных работ и файлы zeroin.h, rkf.h из стандартной библиотеки.