МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА



Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Лабораторная работа №6

«Приближенное решение обыкновенных дифференциальных

уравнений методом Эйлера с пересчетом»

по дисциплине

Вычислительная математика

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Суркова А.С.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сухоруков В.А.

19-ИВТ-3

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2021

Оглавление

**Элементы оглавления не найдены.**

# Цель работы

Закрепление знаний и умений по численному решению обыкновенных дифференциальных уравнений методом Эйлера и методом Адамса.

# Постановка задачи

**Задание 1**

Используя метод Эйлера и метод Эйлера с пересчетом, составить таблицу приближенных значений интеграла дифференциального уравнения y΄=f(x,y), удовлетворяющего начальным условиям y(x0)=y0 на отрезке [a,b]; шаг h=0.1. Все вычисления вести с четырехзначными знаками. Проверить полученные значения, используя метод Рунге-Кутты 4 порядка.



**Задание 2**

Используя метод Адамса с третьими разностями составить таблицу приближенных значений интеграла дифференциального уравнения y΄=f(x,y), удовлетворяющего начальным условиям y(x0)=y0 на отрезке [0,1]; шаг h=0.1. Все вычисления вести с четырехзначными знаками. Начальный отрезок определить методом Рунге-Кутты. Проверить полученные значения, используя метод Эйлера с пересчетом.



# Теоретические сведения

Постановка задачи Коши:

Найти решение дифференциального уравнения

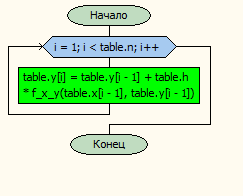
y΄=f(x,y), удовлетворяющее начальному условию у(х 0 )=у 0 . При численном решении уравнения задача ставится так: в точках x 0 , x 1 , х 2, ..., х n найти приближения у k (k = 0,1,2,..., n) для значений точного решения у(х k ). Разность Δx k =х k+1 –х k во многих случаях принимают постоянной h и называют шагом сетки, тогда x k = x 0 +kh, k=0,1,…,n.

## Метод Эйлера

Метод Эйлера для решения указанной задачи Коши основан на непосредственной замене производной разностным отношением по приближенной формуле: dy/dx=f(x,y) , Δy/ Δx=f(x,y) если обозначить h= Δx, то:

y(x+h)=y(x)+hf(x,y)

Приближенные значения y k в точках x K =х 0 +hk вычисляются по формуле yk+1 =y k +hf(x k ,y k )



Нахождение y’ методом Эйлера

## Метод Адамса

Широко распространенным семейством многошаговых методов решения дифференциальных уравнений являются методы Адамса. В практических расчетах чаще всего используется вариант метода Адамса, имеющий четвертый порядок точности и использующий на каждом шаге результаты предыдущих четырех. Именно его и называют обычно методом Адамса.

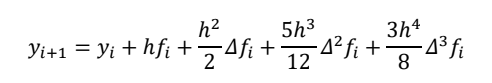
Рассмотрим этот метод. Пусть найдены значения y i-3 ,y i-2 ,y i-1 ,y i - в четырех последовательных узлах и значения правой части - f i-3 ,f i-2 ,f i-1 ,f i , где f i =f(x i ,y i ). В качестве интерполяционного многочлена P 3 (x) можно взять многочлен Ньютона. В случае постоянного шага конечные разности для правой части в узле имеют вид:

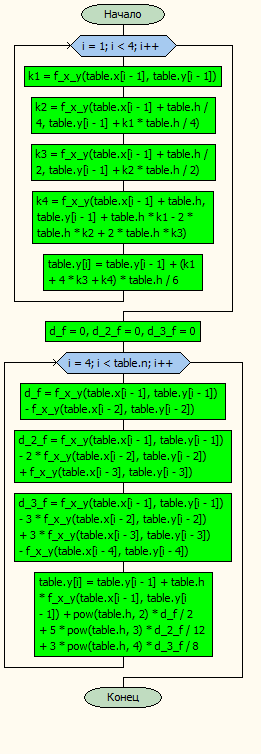
Δf i =f i -f i-1

Δ 2f i =f i -2f i-1 +f i-2

Δ3f i =f i -3f i-1 +3f i-2 -f i-3

Тогда разностную схему четвертого порядка метода Адамса можно записать после необходимых преобразований в виде :



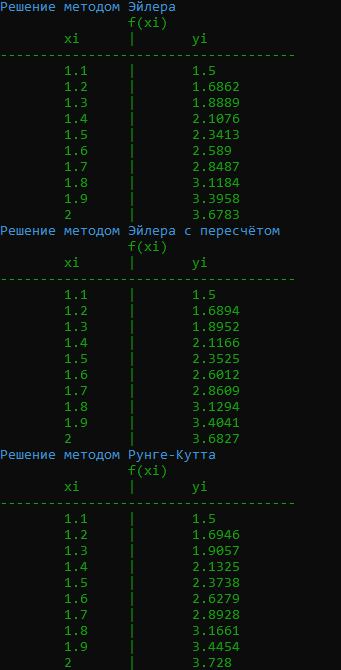


Нахождение y’ методом Адамса. Для нахождения первых 3 значений используется метод Рунге-Кутта

# Расчетные данные

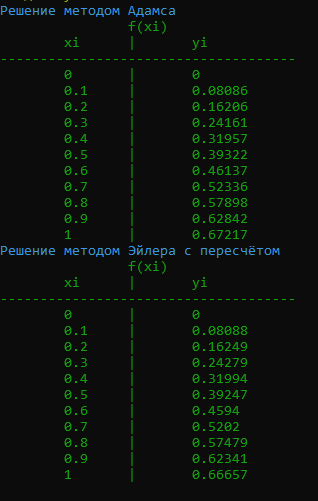
## Задание 1





## Задание 2





# Код программы

## Value\_function\_table.h

#pragma once

#include<vector>

#include<iostream>

#include"Colors.h"

using namespace std;

/\*Класс для описания таблицы значений функции\*/

class Value\_function\_table{

public:

vector<double>x; //Координаты x точек

vector<double>y; //Координаты y точек

size\_t n; //Количество точек

double a; //Левая граница для х

double b; //Правая граница для х

double h; //Шаг

Value\_function\_table() {

n = 0;

a = 0;

b = 0;

h = 0;

}

//Функция заполнения таблицы

void set\_value() {

setlocale(LC\_ALL, "Russian");

cout<<Yellow << "X:[a,b]\nВведите а ";

cin >> this->a;

cout << "\nВведите b ";

cin >> this->b;

cout << "\nВведите шаг h ";

cin >> this->h;

cout << "\nВведите y0 ";

double y0; cin >> y0;

this->y.push\_back(y0);

for (double i = a; i <=b; i=i+h){

this->x.push\_back(i);

this->y.push\_back(0);

}

this->n = this->x.size();

}

};

## solution methods.h

#pragma once

#include<cmath>

#include<vector>

#include<iostream>

#include<iomanip>

#include"Value\_function\_table.h"

/\*Библиотека для решения дифференциального уравнения методом Эйлера и Рунге-Кутта\*/

/\*Нахождение значения функции при заданных x и y

\*Параметр equation отвечает за выбор функции

\*Если equation=1, то находится значение функции f(x,y)=x+sin(y/3^(1/2))

\*Если equation=2, то находится значение функции f(x,y)=(0.8-y^2)\*cos(x)+0.3y \*/

double f\_x\_y(double x, double y,int equation) {

double res=0;

if (equation == 1) {

res = x + sin(y / pow(3, 0.5));

}

if (equation == 2) {

res = (0.8 - y \* y) \* cos(x) + 0.3 \* y;

}

return res;

}

/\*Решение уравнения методом Эйлера\*/

void Euler(Value\_function\_table table, int equation) {

for (size\_t i = 1; i < table.n; i++){

table.y[i]=table.y[i - 1] + table.h \* f\_x\_y(

table.x[i - 1], table.y[i - 1],equation);

}

cout<<Blue<<"Решение методом Эйлера\n"

<<Green << "\t\tf(xi)\n"

<< "\txi\t|\tyi\n"

<<"-------------------------------------\n";

for (size\_t i = 0; i < table.n; i++){

cout << "\t" << table.x[i] << "\t|\t"

<< setprecision(5)<< table.y[i] << "\n";

}

}

/\*Решение уравнения методом Эйлера с пересчётом\*/

void Euler\_recount(Value\_function\_table table, int equation) {

double y;

for (size\_t i = 1; i < table.n; i++){

y = table.y[i-1]+table.h\*f\_x\_y(table.x[i - 1],

table.y[i - 1],equation);

table.y[i]=table.y[i - 1] + table.h / 2 \*

(f\_x\_y(table.x[i - 1], table.y[i - 1],equation) +

f\_x\_y(table.x[i - 1], y,equation));

}

cout << Blue << "Решение методом Эйлера с пересчётом\n"

<< Green << "\t\tf(xi)\n"

<< "\txi\t|\tyi\n"

<< "-------------------------------------\n";

for (size\_t i = 0; i < table.n; i++) {

cout << "\t" << table.x[i] << "\t|\t"

<< setprecision(5) << table.y[i] << "\n";

}

}

/\*Решение методом Рунге-Кутта\*/

void Runge\_Kutt(Value\_function\_table table, int equation) {

double k1, k2, k3, k4;

for (size\_t i = 1; i < table.n; i++){

k1 = f\_x\_y(table.x[i - 1], table.y[i - 1],equation);

k2 = f\_x\_y(table.x[i - 1] + table.h / 4, table.y[i - 1]

+ k1 \* table.h / 4, equation);

k3 = f\_x\_y(table.x[i - 1] + table.h / 2, table.y[i - 1]

+ k2 \* table.h / 2, equation);

k4 = f\_x\_y(table.x[i - 1] + table.h, table.y[i - 1]

+ table.h \* k1 - 2 \* table.h \* k2 + 2 \* table.h \*

k3, equation);

table.y[i]=table.y[i - 1] + (k1 + 4\*k3 + k4) \*

table.h / 6;

}

cout << Blue << "Решение методом Рунге-Кутта\n"

<< Green << "\t\tf(xi)\n"

<< "\txi\t|\tyi\n"

<< "-------------------------------------\n";

for (size\_t i = 0; i < table.n; i++) {

cout << "\t" << table.x[i] << "\t|\t"

<< setprecision(5) << table.y[i] << "\n";

}

}

/\*Решение методом Адамса\*/

void Adams(Value\_function\_table table, int equation) {

//Нахождение первых 3 значений методом Рунге-Кутта

double k1, k2, k3, k4;

for (size\_t i = 1; i < 4; i++) {

k1 = f\_x\_y(table.x[i - 1], table.y[i - 1], equation);

k2 = f\_x\_y(table.x[i - 1] + table.h / 4, table.y[i - 1]

+ k1 \* table.h / 4, equation);

k3 = f\_x\_y(table.x[i - 1] + table.h / 2, table.y[i - 1]

+ k2 \* table.h / 2, equation);

k4 = f\_x\_y(table.x[i - 1] + table.h, table.y[i - 1]

+ table.h \* k1 - 2 \* table.h \* k2 + 2 \* table.h \*

k3, equation);

table.y[i] = table.y[i - 1] + (k1 + 4 \* k3 + k4)

\* table.h / 6;

}

//Нахождение остальных значений методом Адамса

double d\_f = 0, d\_2\_f = 0, d\_3\_f = 0;

for (size\_t i = 4; i < table.n; i++){

d\_f = f\_x\_y(table.x[i - 1], table.y[i - 1], equation) –

f\_x\_y(table.x[i - 2], table.y[i - 2], equation);

d\_2\_f= f\_x\_y(table.x[i - 1], table.y[i - 1], equation)

-2\*f\_x\_y(table.x[i - 2], table.y[i - 2],

equation) + f\_x\_y(table.x[i - 3],

table.y[i - 3], equation);

d\_3\_f = f\_x\_y(table.x[i - 1], table.y[i - 1], equation)

-3 \* f\_x\_y(table.x[i - 2], table.y[i - 2],

equation) +3 \* f\_x\_y(table.x[i - 3],

table.y[i - 3], equation) -

f\_x\_y(table.x[i - 4], table.y[i - 4],

equation);

table.y[i] = table.y[i - 1] +table.h \*

f\_x\_y(table.x[i - 1], table.y[i - 1],

equation) +pow(table.h, 2) \* d\_f / 2 +5 \* pow(table.h, 3) \* d\_2\_f / 12 +3 \* pow(table.h, 4) \* d\_3\_f / 8;

}

cout << Blue << "Решение методом Адамса\n"

<< Green << "\t\tf(xi)\n"

<< "\txi\t|\tyi\n"

<< "-------------------------------------\n";

for (size\_t i = 0; i < table.n; i++) {

cout << "\t" << table.x[i] << "\t|\t"

<< setprecision(5) << table.y[i] << "\n";

}

}

## Main.cpp

#include "solution methods.h"

#include "Value\_function\_table.h"

#include "Colors.h"

#include<iomanip>

using namespace std;

int main() {

setlocale(LC\_ALL, "Russian"); //Включение русского языка в консол

cout<<Blue << "Решение уравнения y'=f(x,y)=x+sin(y/3^(1/2))\n";

Value\_function\_table table\_f,table\_s;

table\_f.set\_value();

Euler(table\_f,1);

Euler\_recount(table\_f,1);

Runge\_Kutt(table\_f,1);

cout << Blue

<< "Решение уравнения y'=f(x,y)=(0.8-y^2)\*cos(x)+0.3y\n";

table\_s.set\_value();

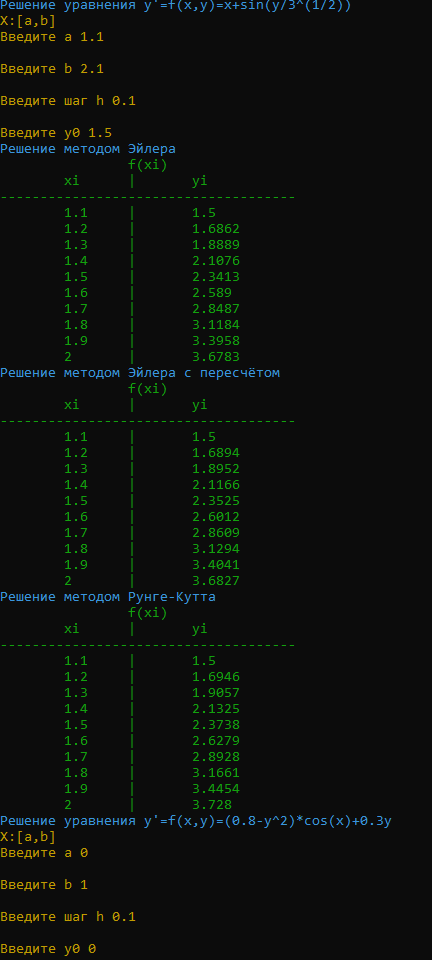
Adams(table\_s, 2);

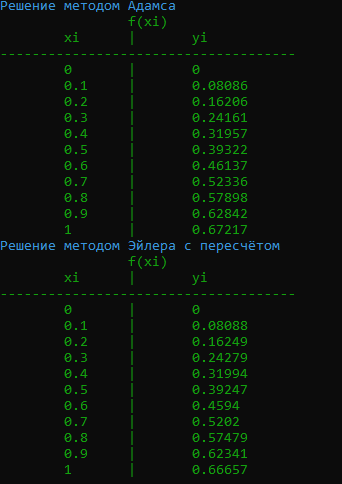
Euler\_recount(table\_s, 2);

return 0;

}

# Результаты работы программы





# Вывод

Закрепил знания и умения по численному решению обыкновенных дифференциальных уравнений методом Эйлера и методом Адамса. Сравнил значения, полученные разными способами решения, они получились равны в пределах погрешности.