Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический университет им. А.Н. Туполева»

Институт компьютерных технологий и защиты информации

Кафедра прикладной математики и информатики имени Ю.В.Кожевникова

**Лабораторная работа № 1**

«Приближенное решение обыкновенных дифференциальных уравнений»

по дисциплине «Численные методы»

Выполнила:

студентка группы 4318

Андреева Мария

Казань 2024

**Вариант 2**

**Цель работы:** научиться решать обыкновенные дифференциальные уравнения методом Эйлера, методом Рунге-Кутта с постоянным и переменным шагом и методом Адамса с помощью ЭВМ.

**Содержание работы:**

1. Изучить методы Эйлера, Рунге-Кутта и Адамса для приближенного

решения задачи Коши.

2. На конкретном примере усвоить порядок решения обыкновенного

дифференциального уравнения указанными методами с помощью ЭВМ.

3. Составить программу на любом языке программирования,

реализующую процесс приближенного решения обыкновенного

дифференциального уравнения указанными методами.

4. Сделать вывод о точности используемых методов.

5. Составить отчет о проделанной работе.  
**Выполнение работы**

Найдем аналитическое решение:

;  *dy* ;

Из условия *y*(1) = -1 следует:

Решение задачи Коши:

1. Метод Эйлера

Для построения формулы метода Эйлера разделим отрезок на *n* равных частей и получим систему равноотстоящих точек:

*x1= x0 + h ,*

*……*

*xi= x0 + ih= xi-1 + h ,*

*……*

*xn= xn-1+h= xi + nh , i=*

Шаг

Рабочая формула метода Эйлера в общем случае имеет вид:

*Решение методом Эйлера.*

*Формула метода Эйлера для заданного примера:*

1. Метод Рунге-Кутта четвертого порядка

Для вычисления по методу Рунге-Кутта четвертого порядка точности необходимо вычислить коэффициенты:

Формула метода Рунге-Кутта четвертого порядка точности примет вид:

*Формула метода Рунге-Кутта четвертого порядка точности для заданного примера:*

*,* где

Метод Адамса

*Экстраполяционная формула Адамса:*

*,*

*Формула метода Адамса для заданного примера:*

**Листинг программы**

#include <iostream>

#include <string>

#include <math.h>

using namespace std;

double F(double x) { return -exp((x \* x) / 2 + x - 1.5); } // частное решение

double f(double x, double y) { return (x + 1) \* y; } // исходная дифф. функция

int main() {

setlocale(LC\_ALL, "Russian");

const double eps1 = 0.001;

const double eps2 = 0.00001;

double a = 1.0; double b = 2.0; double n1 = 10.0; double n2 = 20.0;

double h1 = (b - a) / n1; double h2 = (b - a) / n2;

double X1[100]; double Y\_E1[100]; double Y\_RK1[100];

double K1\_1[100]; double K2\_1[100]; double K3\_1[100]; double K4\_1[100];

double Y\_A1[100];

double X2[100]; double Y\_E2[100]; double Y\_RK2[100]; double Y\_A2[100];

X1[0] = a; Y\_E1[0] = -1.0; Y\_RK1[0] = -1.0; Y\_A1[0] = -1.0;

X2[0] = a; Y\_E2[0] = -1.0; Y\_RK2[0] = -1.0; Y\_A2[0] = -1.0;

for (int i = 1; i <= n1; i++) { X1[i] = a + i \* h1; }

for (int i = 1; i <= n2; i++) { X2[i] = a + i \* h2; }

for (int i = 0; i <= 99; i++) {}

int i = 0;

printf("Приближенное решение обыкновенных дифференциальных уравнений\n\n");

printf("dy/dx = (x+1)y, y(1) = -1 на [1, 2]\n\n");

printf("Из условия y(1) = -1 => y = -exp(x^2/2+x-1,5)\n\n");

printf("Метод Эйлера при n = 10:\n");

printf("--------------------------------------------------------------------\n");

printf("| i | x\_i | y\_i | y\_i\_E | |y\_i - y\_i\_E| |\n");

printf("--------------------------------------------------------------------\n");

printf("| %2d | %12.9lf | %12.9lf | %12.9lf | %12.9lf |\n", 0, X1[0], F(X1[0]), Y\_E1[0], abs(F(X1[0]) - Y\_E1[0]));

for (int i = 0; i < n1; i++) {

Y\_E1[i + 1] = Y\_E1[i] + h1 \* f(X1[i], Y\_E1[i]); // Метод Эйлера 1

printf("--------------------------------------------------------------------\n");

printf("| %2d | %12.9lf | %12.9lf | %12.9lf | %12.9lf |\n", i + 1, X1[i + 1], F(X1[i + 1]), Y\_E1[i + 1], abs(F(X1[i + 1]) - Y\_E1[i + 1]));

}

printf("--------------------------------------------------------------------\n\n");

h1 = (b - a) / n1;

printf("Метод Рунге-Кутта при n = 10:\n");

printf("--------------------------------------------------------------------\n");

printf("| i | x\_i | y\_i | y\_i\_RK | |y\_i - y\_i\_RK| |\n");

printf("--------------------------------------------------------------------\n");

printf("| %2d | %12.9lf | %12.9lf | %12.9lf | %12.9lf |\n", 0, X1[0], F(X1[0]), Y\_RK1[0], abs(F(X1[0]) - Y\_RK1[0]));

for (int i = 0; i < n1; i++) {

K1\_1[i] = h1 \* f(X1[i], Y\_RK1[i]);

K2\_1[i] = h1 \* f((X1[i] + h1 / 2.0), (Y\_RK1[i] + K1\_1[i] / 2.0));

K3\_1[i] = h1 \* f((X1[i] + h1 / 2.0), (Y\_RK1[i] + K2\_1[i] / 2.0));

K4\_1[i] = h1 \* f((X1[i] + h1), (Y\_RK1[i] + K3\_1[i]));

Y\_RK1[i + 1] = Y\_RK1[i] + 1.0 / 6.0 \* (K1\_1[i] + 2.0 \* K2\_1[i] + 2.0 \* K3\_1[i] + K4\_1[i]); // Метод Рунге-Кутта 1

printf("--------------------------------------------------------------------\n");

printf("| %2d | %12.9lf | %12.9lf | %12.9lf | %12.9lf |\n", i + 1, X1[i + 1], F(X1[i + 1]), Y\_RK1[i + 1], abs(F(X1[i + 1]) - Y\_RK1[i + 1]));

}

printf("--------------------------------------------------------------------\n\n");

printf("Метод Адамса при n = 10:\n");

printf("--------------------------------------------------------------------\n");

printf("| i | x\_i | y\_i | y\_i\_A | |y\_i - y\_i\_A| |\n");

printf("--------------------------------------------------------------------\n");

printf("| %2d | %12.9lf | %12.9lf | %12.9lf | %12.9lf |\n", 0, X1[0], F(X1[0]), Y\_A1[0], abs(F(X1[0]) - Y\_A1[0]));

for (int i = 0; i < n1; i++) {

if (i < 3) {

Y\_A1[i + 1] = Y\_A1[i] + 1.0 / 6.0 \* (K1\_1[i] + 2.0 \* K2\_1[i] + 2.0 \* K3\_1[i] + K4\_1[i]);

}

else {

Y\_A1[i + 1] = Y\_A1[i] + h1 / 24.0 \* (55.0 \* f(X1[i], Y\_A1[i]) - 59.0 \* f(X1[i - 1], Y\_A1[i - 1]) + 37.0 \* f(X1[i - 2], Y\_A1[i - 2]) - 9.0 \* f(X1[i - 3], Y\_A1[i - 3])); // Метод Адамса 1

}

printf("--------------------------------------------------------------------\n");

printf("| %2d | %12.9lf | %12.9lf | %12.9lf | %12.9lf |\n", i + 1, X1[i + 1], F(X1[i + 1]), Y\_A1[i + 1], abs(F(X1[i + 1]) - Y\_A1[i + 1]));

}

printf("--------------------------------------------------------------------\n\n");

printf("\nМетод Эйлера при n = 20:\n");

printf("--------------------------------------------------------------------\n");

printf("| i | x\_i | y\_i | y\_i\_E | |y\_i - y\_i\_E| |\n");

printf("--------------------------------------------------------------------\n");

printf("| %2d | %12.9lf | %12.9lf | %12.9lf | %12.9lf |\n", 0, X2[0], F(X2[0]), Y\_E2[0], abs(F(X2[0]) - Y\_E2[0]));

for (int i = 0; i < n2; i++) {

Y\_E2[i + 1] = Y\_E2[i] + h2 \* f(X2[i], Y\_E2[i]); // Метод Эйлера 2

printf("--------------------------------------------------------------------\n");

printf("| %2d | %12.9lf | %12.9lf | %12.9lf | %12.9lf |\n", i + 1, X2[i + 1], F(X2[i + 1]), Y\_E2[i + 1], abs(F(X2[i + 1]) - Y\_E2[i + 1]));

}

printf("--------------------------------------------------------------------\n\n");

h2 = (b - a) / n2;

printf("Метод Рунге-Кутта при n = 20:\n");

printf("--------------------------------------------------------------------\n");

printf("| i | x\_i | y\_i | y\_i\_RK | |y\_i - y\_i\_RK| |\n");

printf("--------------------------------------------------------------------\n");

printf("| %2d | %12.9lf | %12.9lf | %12.9lf | %12.9lf |\n", 0, X2[0], F(X2[0]), Y\_RK2[0], abs(F(X2[0]) - Y\_RK2[0]));

for (int i = 0; i < n2; i++) {

K1\_1[i] = h2 \* f(X2[i], Y\_RK2[i]);

K2\_1[i] = h2 \* f((X2[i] + h2 / 2.0), (Y\_RK2[i] + K1\_1[i] / 2.0));

K3\_1[i] = h2 \* f((X2[i] + h2 / 2.0), (Y\_RK2[i] + K2\_1[i] / 2.0));

K4\_1[i] = h2 \* f((X2[i] + h2), (Y\_RK2[i] + K3\_1[i]));

Y\_RK2[i + 1] = Y\_RK2[i] + 1.0 / 6.0 \* (K1\_1[i] + 2.0 \* K2\_1[i] + 2.0 \* K3\_1[i] + K4\_1[i]); // Метод Рунге-Кутта 2

printf("--------------------------------------------------------------------\n");

printf("| %2d | %12.9lf | %12.9lf | %12.9lf | %12.9lf |\n", i + 1, X2[i + 1], F(X2[i + 1]), Y\_RK2[i + 1], abs(F(X2[i + 1]) - Y\_RK2[i + 1]));

}

printf("--------------------------------------------------------------------\n\n");

printf("Метод Адамса при n = 20:\n");

printf("--------------------------------------------------------------------\n");

printf("| i | x\_i | y\_i | y\_i\_A | |y\_i - y\_i\_A| |\n");

printf("--------------------------------------------------------------------\n");

printf("| %2d | %12.9lf | %12.9lf | %12.9lf | %12.9lf |\n", 0, X2[0], F(X2[0]), Y\_A2[0], abs(F(X2[0]) - Y\_A2[0]));

for (int i = 0; i < n2; i++) {

if (i < 3) {

Y\_A2[i + 1] = Y\_A2[i] + 1.0 / 6.0 \* (K1\_1[i] + 2.0 \* K2\_1[i] + 2.0 \* K3\_1[i] + K4\_1[i]);

}

else {

Y\_A2[i + 1] = Y\_A2[i] + h2 / 24.0 \* (55.0 \* f(X2[i], Y\_A2[i]) - 59.0 \* f(X2[i - 1], Y\_A2[i - 1]) + 37.0 \* f(X2[i - 2], Y\_A2[i - 2]) - 9.0 \* f(X2[i - 3], Y\_A2[i - 3])); // Метод Адамса 2

}

printf("--------------------------------------------------------------------\n");

printf("| %2d | %12.9lf | %12.9lf | %12.9lf | %12.9lf |\n", i + 1, X2[i + 1], F(X2[i + 1]), Y\_A2[i + 1], abs(F(X2[i + 1]) - Y\_A2[i + 1]));

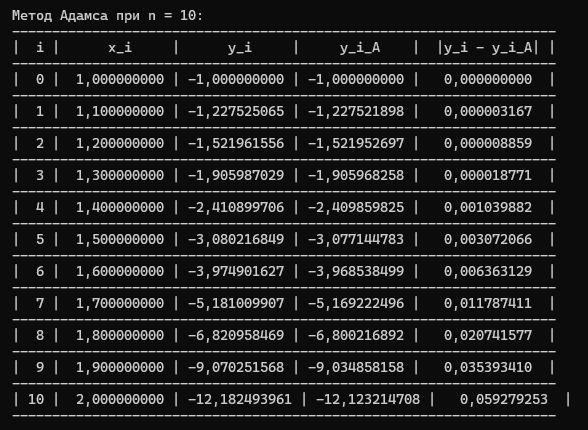
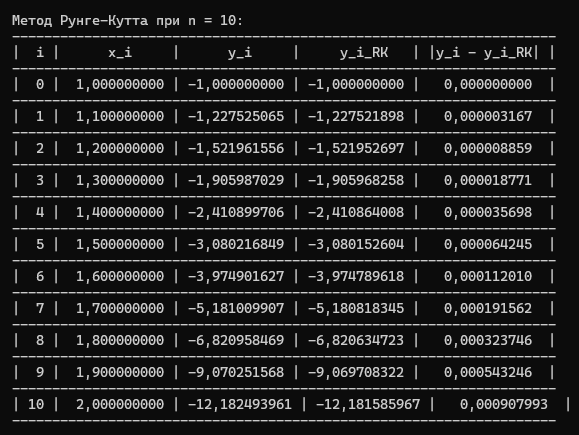
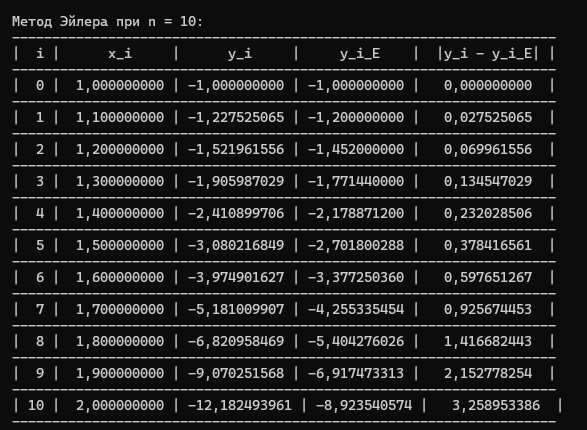
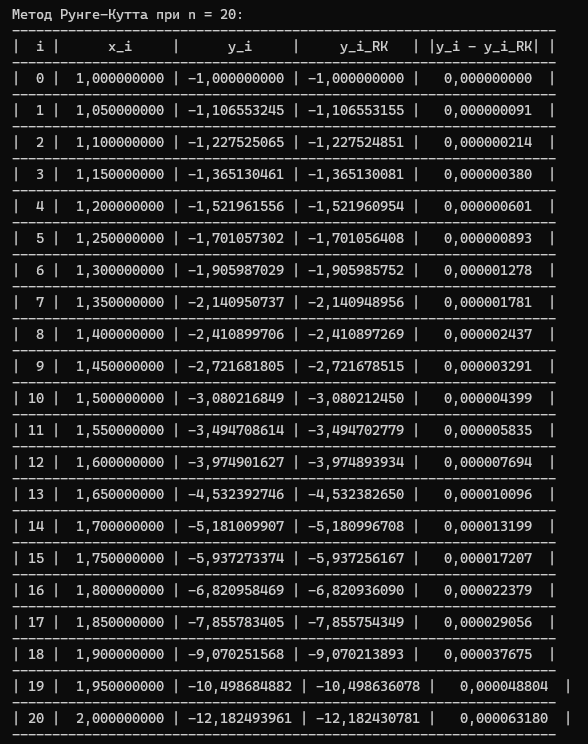
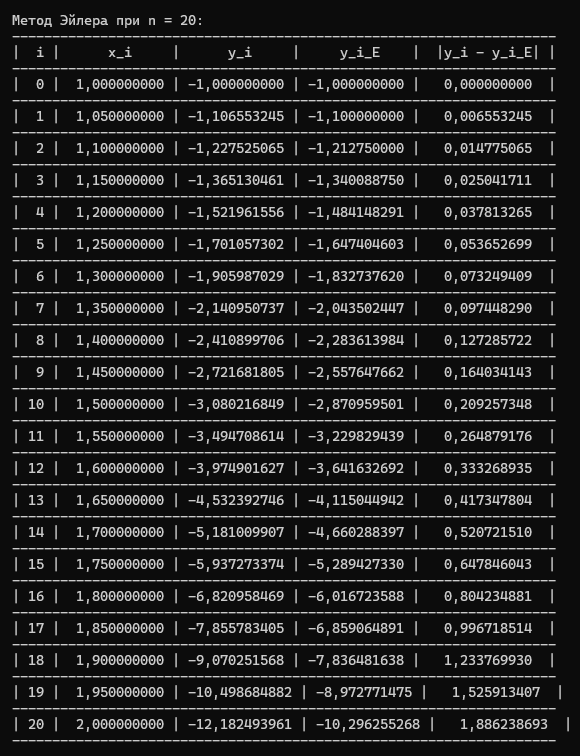
}

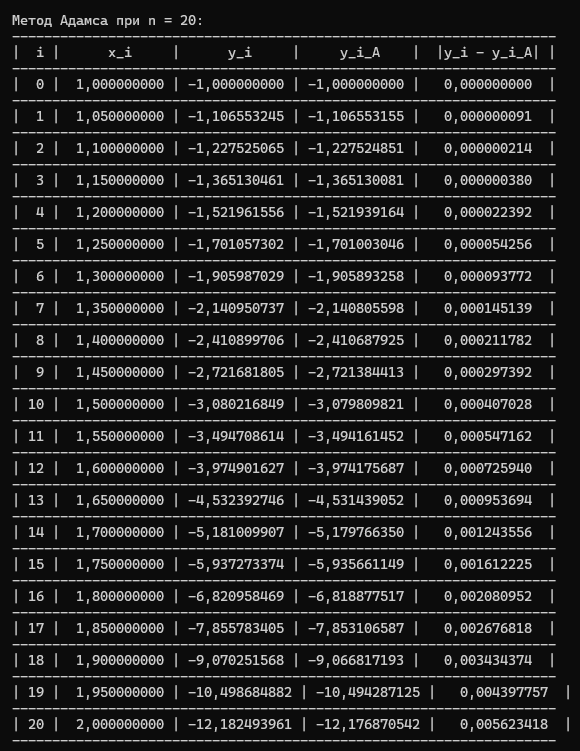
printf("--------------------------------------------------------------------\n");

}

}

**Результат выполнения программы**

**  
**

****

**Выводы**

1. Самым точным методом является метод Рунге-Кутта.

2. Чем меньше шаг, тем больше точность.