МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

Высшего профессионального образования

**«Вятский государственный университет»**

**(ФГБОУ ВПО «ВятГУ»)**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Численное интегрирование и решение обыкновенных дифференциальных уравнений

Отчет по лабораторной работе №4 дисциплины

«Вычислительная математика»

Выполнил студент группы ИВТб-21\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Седов М.Д./

Проверил преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Исупов К.С./

Киров 2018

1 Постановка задачи

1) Вычислить определённый интеграл с точностью до 0,0001. Выбрать значение n, обеспечивающее заданную точность, из формулы остаточного члена.

Задание:

Определённый интеграл от функции: 1/sqrt(x^2+0,6)

Пределы интегрирования: [2,2;2,6]

Использовать формулу Симпсона.

2) Вычислить определённый интеграл с точностью до 0,0001 по другой квадратурной формуле:

Задание:

Определённый интеграл от функции: x^2\*lg(x)

Пределы интегрирования: [1,4;3]

Использовать формулу трапеций.

В качестве начального шага взять число, близкое к Е^(1/m), где m=2. Для приближённой оценки погрешности применить принцип Рунге.

3) Вычислить определённый интеграл по квадратурной формуле Гаусса. Для оценки погрешности взять различное количество узлов:

n1=5; n2=8.

Квадратурная формула Гаусса с 5 узлами:

x1=-x5=-0,90618 A1=A5=0,23698

x2=-x4=-0,538469 A2=A4=0,47863

x3=0 A3=0,56889

Квадратурная формула Гаусса с 8 узлами:

x1=-x8=-0,96028986 A1=A8=0,10122854

x2=-x7=-0,79666648 A2=A7=0,22238103

x3=-x6=-0,52553242 A3=A6=0,31370664

x4=-x5=-0,18343464 A4=A5=0,36268378

Задание:

Определённый интеграл от функции: x/sqrt(x^2+2)

Пределы интегрирования: [0,8;2,0]

4) Решить обыкновенное дифференциальное уравнение. Решение представить в табличной и графической формах. Для оценки погрешности выполнить расчёт с шагом h и с шагом h/2.

Задание:

По формуле 2-го порядка точности решить дифференциальное уравнение

y'=x+y\*sqrt(x)

a=1; y(0)=0; h=0,1; 0<=x<=1

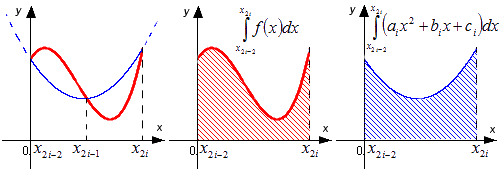
2 Краткие теоретические сведения

2.1 Формула Симпсона

Пусть задана функция вида y = f(x), имеющая непрерывность на интервале [a; b]. Необходимо произвести вычисление определенного интеграла .

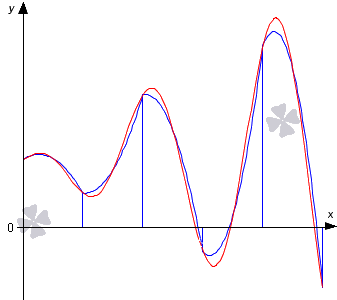
Необходимо разбить отрезок [a; b] на n отрезков вида [x2i-2; x2i], i = 1, 2, …, n с длиной и точками a = x0 < x2 < x4 < … < x2i=b. Тогда точки x2i-1 считаются серединами отрезков [x2i-2; x2i]. Данный случай показывает, что определение узлов производится через xi = a + 2 \* i \* h.

Каждый интервал [x2i-2; x2i] подынтегральной функции приближен при помощи параболы, заданной y=aix2+bix+ci, проходящей через точки с координатами (x2i-2; f(x2i-2)), (x2i-1; f(x2i-1)), (x2i; f(x2i)). Данные действия выполняются для того, чтобы интеграл в взять в качестве приближенного значения . Можем вычислить при помощи формулы Ньютона-Лейбница. Рассмотрим рисунок, проведенный ниже.



Графическая иллюстрация метода парабол (Симпсона).

При помощи красной линии изображается график функции y=f(x), синей – приближение графика y=f(x) при помощи квадратичных парабол.



Исходная формула метода Симпсона для целочисленного интегрирования имеет следующий вид:

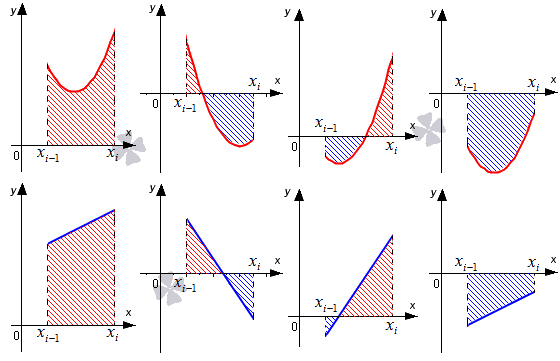
Формула оценки абсолютной погрешности имеет вид:

2.2 Формула трапеций

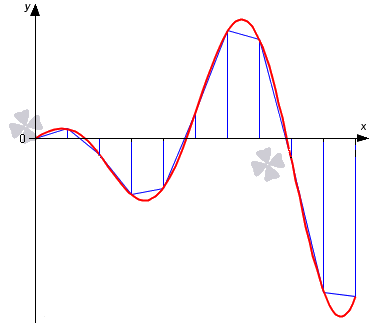
Пусть задана функция вида y = f(x), имеющая непрерывность на интервале [a; b]. Необходимо произвести вычисление определенного интеграла .

Для этого разделим отрезок [a; b] на несколько равных интервалов длины h точками a = x0 < x1 < x2 < … < xn = b. Обозначим количество полученных интервалов как n.

Найдем шаг разбиения: . Узлы определяются следующей формулой: xi = a + i \* h, i = 0, 1, …, n.



Суть метода трапеций заключается в следующем: мы можем представить определенный интеграл в виде суммы интегралом вида на каждом элементарном отрезке и в последующей приближенной замене: .

**

Исходная формула метода трапеций для целочисленного интегрирования имеет следующий вид:

Оценка абсолютной погрешности метода трапеций высчитывается следующим образом:

2.3 Формула Гаусса

Если квадратурная формула с n узлами является интерполяционной, то ее алгебраическая степень точности не меньше n – 1; при любых заданных узлах построение интерполяционной квадратурной формулы осуществляется за счёт выбора ее n коэффициентов. За счет же выбора n узлов интерполяционной квадратурной формулы можно добиться того, чтобы она имела возможно более высокую алгебраическую степень точности, а именно 2n-1. Задача построения такой квадратурной формулы рассматривалась Карлом Фридрихом Гауссом, им была доказана ее разрешимость.

Квадратурная формула с n узлами, алгебраическая степень которой равна 2n-1, называется квадратурной формулой Гаусса или квадратурной формулой наивысшей алгебраической степени точности.

В случае отрезка [-1,1] квадратурная формула

является квадратурной формулой Гаусса тогда и только тогда, когда она является интерполяционной, а ее узлы ti являются корнями многочлена Лежандра

В частности, при n=2 и n=3 квадратурный формулы Гаусса для отрезка [-1;1] таковы:

Чтобы получить квадратурную формулу Гаусса для произвольного отрезка [a;b], следует сделать замену переменной

в результате которой

Воспользовавшись квадратурной формулой Гаусса для отрезка [-1;1], получим квадратурную формулу Гаусса для отрезка [a;b]:

где xi = 0.5(a+b+(b-a)ti), ti – корни многочлена Лежандра Pn(t)

Для погрешности квадратурной формулы Гаусса с n узлами справедлива оценка

Коэффициенты квадратурных формул Гаусса положительны. Поэтому использование квадратурных формул Гаусса с большим числом узлов не приводит к тем осложнениям, которые возникают при использовании формул Ньютона-Котеса.

2.4 Метод Рунге-Кутта

Данный метод численного решения дифференциальных уравнений позволяет решить задачу Коши для обыкновенных дифференциальных уравнений первого порядка.

Дадим формулировку задачи Коши для дифференциального уравнения первого порядка:

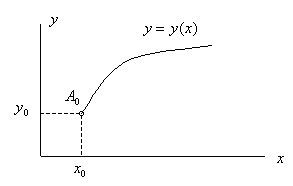
Дано обыкновенное дифференциальной уравнение первого порядка, разрешенное относительно производной:

y' = f(x, y).

Необходимо найти решение уравнения, удовлетворяющее начальному условию:

y(x0) = y0.

То есть в задачи Коши необходимо найти кривую y(x), проходящую через заданную точку (x0, y0).



Решение задачи Коши является частным решением дифференциального уравнения при заданном начальном условии.

Метод Рунге-Кутта первого порядка (метод Эйлера)

Это простейший численный метод. Рассмотрим задачу Коши

y' = f(x, y), a <= x <= b; y(a) = y0.

Зададим равномерную сетку xi = a + i \* h, i = 0, 1, …, n.

Введем обозначения y(xi) = yi. Тогда имеем вычислительную формулу для метода Рунге-Кутта первого порядка

yi+1 = yi + h \* f(xi, yi), i = 0, 1, …, n-1.

Данная формула позволяет, начиная от начального условия (x0, y0) найти последовательно величины y1, y2, …, yn с шагом h и, таким образом, решить задачу Коши.

Погрешность метода

Здесь y(x) – точное решение; yi – численное решение;

Метод Эйлера есть метод первого порядка точности

Метод Рунге-Кутта второго порядка точности

Вычислительная формула метода Рунге-Кутта второго порядка имеет следующий вид:

Погрешность метода: ε ~ (b-a)M3h2

Метод Рунге-Кутта четвертого порядка точности

Это наиболее распространенный метод решения задачи Коши. Вычислим интеграл по формуле Симпсона. Получим следующую вычислительную формулу:

Погрешность метода: ε ~ (b-a)M5h4

Схемы Рунге-Кутта имеют ряд достоинств:

1. Все они (кроме метода Эйлера) имеют хорошую точность;
2. Они являются явными, т.е. значения yi+1 вычисляются по ранее найденный значениям y1, y2, …, yi.
3. Схемы допускают введение переменного шага h

3 Выполнение задания

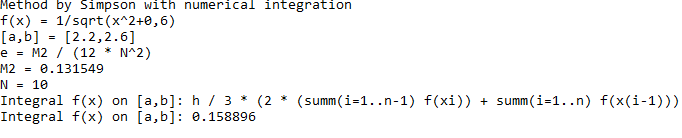


Рисунок 1 – формула Симпсона

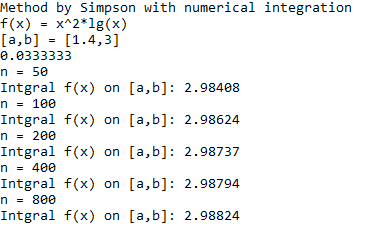


Рисунок 2 – формула трапеций

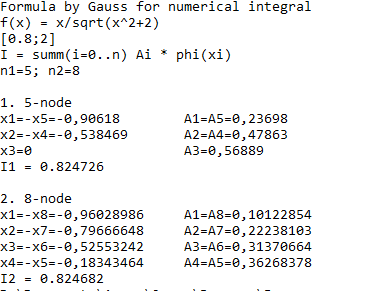


Рисунок 3 –формула Гаусса

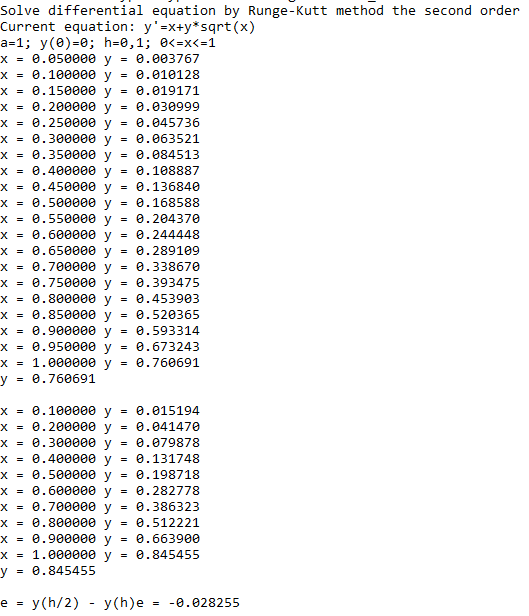


Рисунок 4 – метод Рунге-Кутта

Приложение А

Листинг программы #1

#include <iostream>

#include <iomanip>

#include <cmath>

#include <algorithm>

using namespace std;

double f(double x) {

return 1 / sqrt(x \* x + 0.6);

}

int main(int argc, char const \*argv[])

{

cout << "Method by Simpson with numerical integration\n";

cout << "f(x) = 1/sqrt(x^2+0,6)\n";

cout << "[a,b] = [2.2,2.6]\n";

double a = 2.2, b = 2.6, e = 0.0001;

cout << "e = M2 / (12 \* N^2)\n";

double M2 = max((3 \* a \* a / (a \* a + 0.6) - 1) / pow(sqrt(a \* a + 0.6),3),

(3 \* b \* b / (b \* b + 0.6) - 1) / pow(sqrt(b \* b + 0.6),3));

cout << "M2 = " << M2 << endl;

int N = round(sqrt(M2 / 12 / e));

cout << "N = " << N << endl;

int n = N / 2;

double h = (b - a) / N;

double integral = 0, s1 = 0, s2 = 0;

integral += f(a);

for (int i = 1; i <= n - 1; i++) s1 += f(a + 2 \* i \* h);

for (int i = 1; i <= n; i++) s2 += f(a + (2 \* i - 1) \* h);

integral += 2 \* s1 + 4 \* s2 + f(b);

integral \*= h / 3;

cout << "Integral f(x) on [a,b]: h / 3 \* (2 \* (summ(i=1..n-1) f(xi)) + summ(i=1..n) f(x(i-1)))\n";

cout << "Integral f(x) on [a,b]: " << integral << endl;

return 0;

}

Листинг программы #2

#include <iostream>

#include <iomanip>

#include <cmath>

#include <algorithm>

using namespace std;

double f(double x) {

return x \* x \* log10(x);

}

int main(int argc, char const \*argv[])

{

cout << "Method by Simpson with numerical integration\n";

cout << "f(x) = x^2\*lg(x)\n";

cout << "[a,b] = [1.4,3]\n";

double a = 1.4, b = 3, e = 0.0001;

double integral\_1 = 0, integral\_2 = 0.1;

int n = 50;

cout << 1.0 / 3 \* (integral\_2 - integral\_1) << endl;

while (1.0 / 3 \* (integral\_2 - integral\_1) >= e) {

integral\_1 = integral\_2;

a = 1.4 + pow(e, 0.5);

cout << "n = " << n << endl;

double h = (b - a) / n, s1 = 0;

integral\_2 = 0;

integral\_2 += f(a) / 2 \* (a - 1.4) + f(b) / 2 \* (b - (a + (n - 1) \* h));

for (int i = 1; i <= n - 1; i++)

s1 += f(a + i \* h) / 2 \* (a + (i + 1) \* h - (a + (i - 1) \* h));

integral\_2 += s1;

cout << "Intgral f(x) on [a,b]: " << integral\_2 << endl;

n \*= 2;

}

return 0;

}

Листинг программы #3

#include <iostream>

#include <iomanip>

#include <cmath>

#include <algorithm>

using namespace std;

double f(double x) {

return x / sqrt(x \* x + 2);

}

int main(int argc, char const \*argv[])

{

cout << "Formula by Gauss for numerical integral\n";

cout << "f(x) = x/sqrt(x^2+2)\n";

cout << "[0.8;2]\n";

cout << "I = summ(i=0..n) Ai \* phi(xi)\n";

double a = 0.8, b = 2.0;

cout << "n1=5; n2=8\n";

cout << "\n1. 5-node\n";

cout << "x1=-x5=-0,90618 A1=A5=0,23698" << endl <<

"x2=-x4=-0,538469 A2=A4=0,47863" << endl <<

"x3=0 A3=0,56889\n";

double x1[5] = {-0.90618, -0.538469, 0, 0.538469, 0.90618};

double A1[5] = {0.23698, 0.47863, 0.56889, 0.47863, 0.23698};

double summ = 0;

for (int i = 0; i < 5; i++) {

summ += A1[i] \* (b - a) / 2 \* f((a + b) / 2 + (b - a) / 2 \* x1[i]);

}

cout << "I1 = " << summ;

cout << endl << "\n2. 8-node\n";

cout << "x1=-x8=-0,96028986 A1=A8=0,10122854" << endl <<

"x2=-x7=-0,79666648 A2=A7=0,22238103" << endl <<

"x3=-x6=-0,52553242 A3=A6=0,31370664" << endl <<

"x4=-x5=-0,18343464 A4=A5=0,36268378\n";

double x2[8] = {-0.96028986, -0.79666648, -0.52553242, -0.18343464, 0.18343464, 0.52553242, 0.79666648, 0.96028986};

double A2[8] = {0.10122854, 0.22238103, 0.31370664, 0.36268378, 0.36268378, 0.31370664, 0.22238103, 0.10122854};

summ = 0;

for (int i = 0; i < 8; i++) {

summ += A2[i] \* (b - a) / 2 \* f((a + b) / 2 + (b - a) / 2 \* x2[i]);

}

cout << "I2 = " << summ;

return 0;

}

Листинг программы #4

#include <iostream>

#include <iomanip>

#include <cmath>

#include <algorithm>

using namespace std;

double f(double x, double y) {

return x + y \* sqrt(x);

}

int main(int argc, char const \*argv[])

{

cout << "Solve differential equation by Runge-Kutt method the second order\n";

cout << "Current equation: y'=x+y\*sqrt(x)\n";

cout << "a=1; y(0)=0; h=0,1; 0<=x<=1\n";

double a = 0, h[2] = {0.05, 0.1}, x, y[2] = {0, 0};

for (int j = 0; j < 2; j++) {

x = a;

for (int i = 0; i < 1 / h[j]; i++) {

x += h[j];

y[j] += h[j] \* f(x + h[j] / 2, y[j] + h[j] / 2 \* f(x, y[j]));

cout << fixed << setprecision(6) << "x = " << x << " y = " << y[j] << endl;

}

cout << "y = " << y[j] << endl << endl;

}

cout << "e = y(h/2) - y(h)";

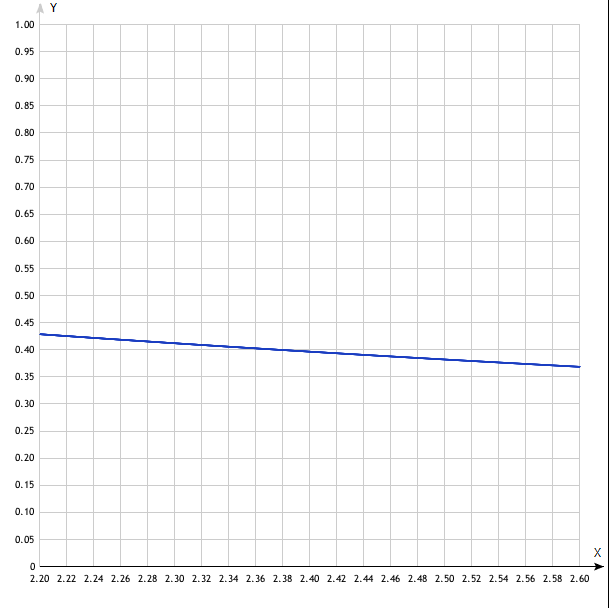
cout << "e = " << (y[0] - y[1]) / 3 << endl;

return 0;

}

Приложение Б

График функции и численный точный результат #1



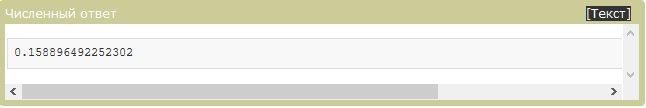
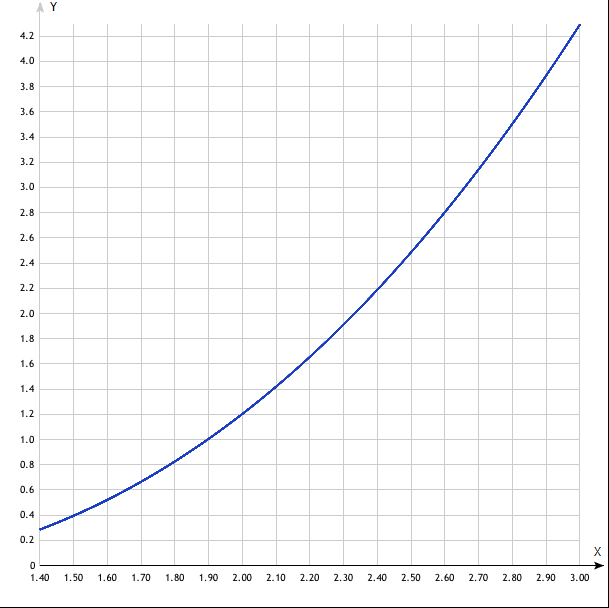


График функции и численный точный результат #2



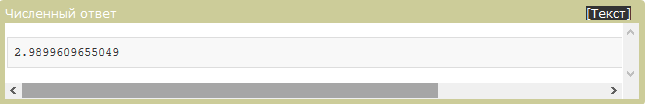
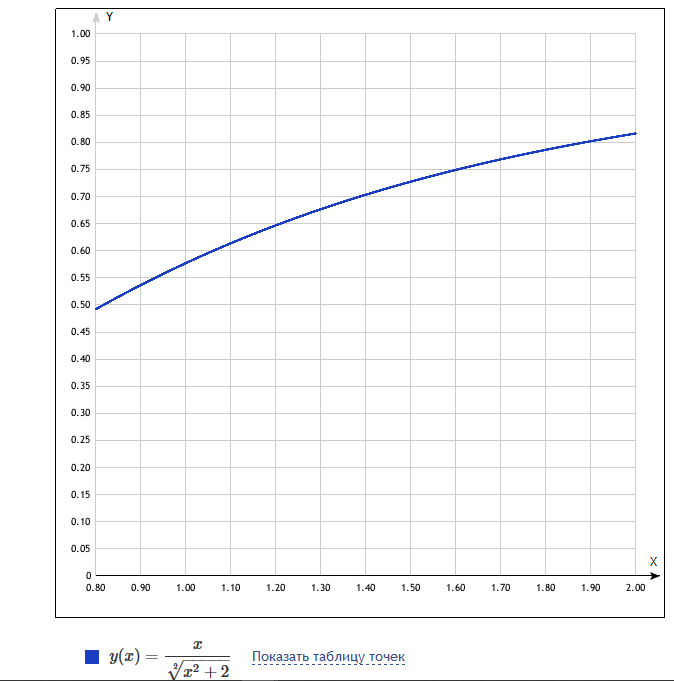


График функции и численный точный результат #3



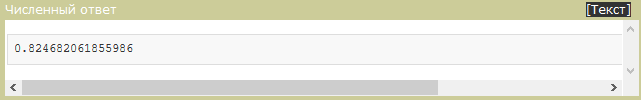
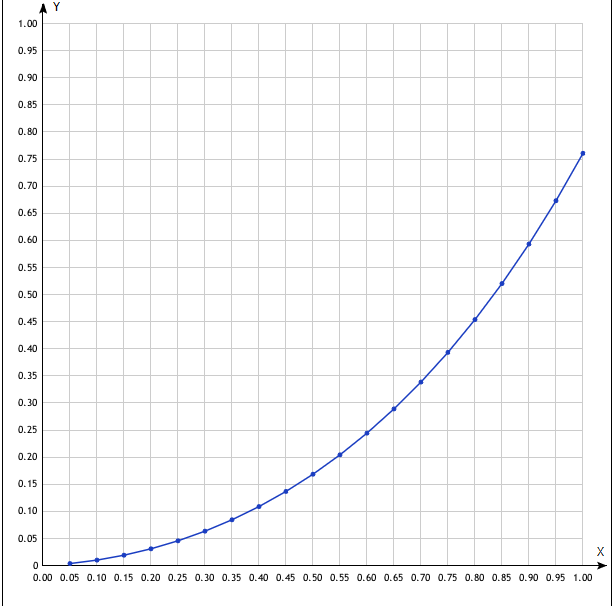


График функции #4



Вывод

В ходе данной лабораторной работы были рассмотрены методы численного интегрирования и решения обыкновенных дифференциальных уравнений: метод Симпсона, метод трапеций, формула Гаусса, метод Рунге-Кутта. В каждом из методов численного интегрирования были изучены основные формулы для приближенного вычисления определенного интеграла заданной функции. Так же были посчитаны значения интегралов, и их погрешности относительно точных результатов. Были изучены квадратурные формулы Гаусса, которые для определенного числа узлов давали более точную формула для вычисления. В совокупности формула Гаусса включает в себя целое семейство различных квадратурных формул для интерполяционной функции. Были изучены основные формулы метода Рунге-Кутта для различных степеней точности для численного решения обыкновенных дифференциальных уравнений. В каждой подзадачи ответы совпали с точными с учётом заданной в условии погрешности.