**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Методы численного анализа

Лабораторная работа №1

Крученкова Евгения Андреевича

студента 3 курса,

специальность «Прикладная математика»

Преподаватель:

доцент кафедры ВычМат,

кандидат физико-математических наук

Будник А.М.

Минск, 2022

ОГЛАВЛЕНИЕ

Постановка задачи. 3

Предварительные Математические вычисления 4

Листинг программы. 5

1. 7

2. 8

3. 10

4. 13

Постановка задачи.

Для вычисления интеграла

с точностью необходимо:

1. Применить правило Рунге, используя квадратурную формулу левых прямоугольников. Определить величину h шага разбиения исходного отрезка интегрирования, достаточного для достижения точности
2. Пользуясь выражением для погрешности интегрирования, определить шаги h в двух нижеуказанных крадратурных формулах, которые обеспечивают требуемую точность результата:

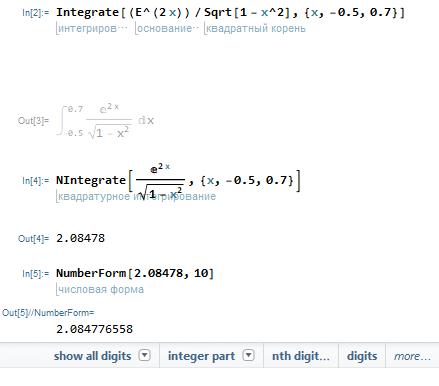
А) Составная квадратурная формула трапеций.

B) Составная квадратурная формула средних прямоугольников.

3. Применить квадратурную формулу НАСТ Гаусса при n = 3. Оценить погрешность интегрирования через формулу остаточного члена

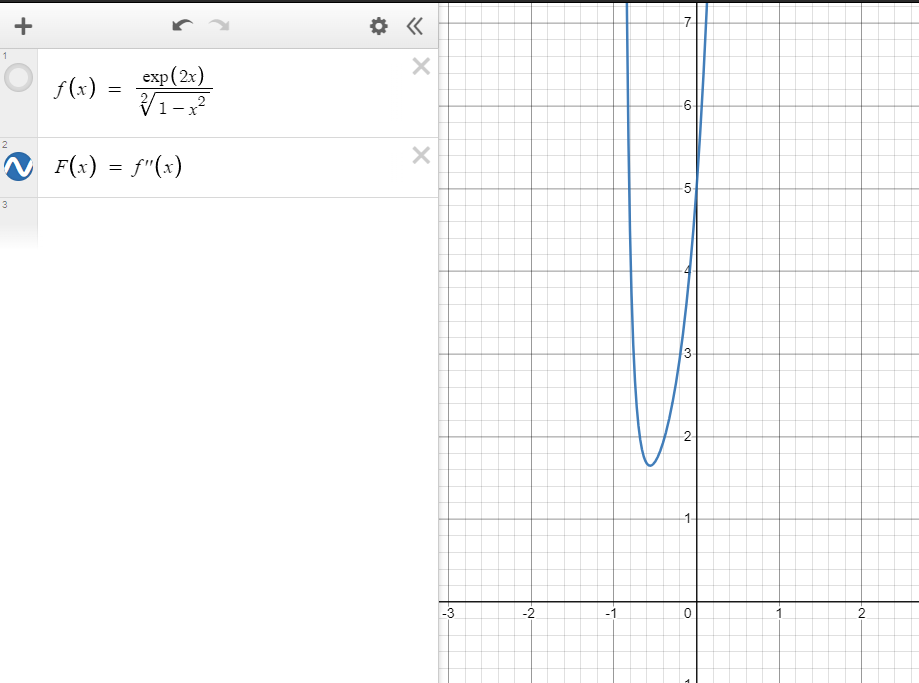
4. Провести сравнительный анализ полученных в п.п. 1-3 результатов.

Предварительные Математические вычисления

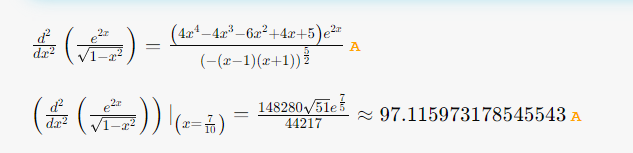


Таким образом, значение I = 2.084776558 будем в дальнейшем рассматривать как точное значение интеграла.

Вычислим необходимые производные:

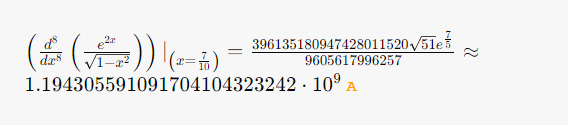


Вторая производная возрастает на промежутке интегрирования, как видно из графика, поэтому за максимум второй производной возьмём её значение на правом конце отрезка.



Положим = 97.2

Для третьего задания нам потребуется максимум производной восьмого порядка. Максимум будет достигаться на правом конце отрезка.



Положим = 1.2\*10^9

Листинг программы.

**Header.h**

#pragma once

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include<cmath>

#include <stdio.h>

#include <iomanip>

#include <fstream>

#include <string>

#include <time.h>

using namespace std;

const double E = pow(10, -5);

const double e = M\_E;

const double M1 = 97.12;

const double M2 = 1.2 \* pow(10, 9);

const double I\_t = 2.084776558;

double F(double x) {

return (pow(e,2\*x)/sqrt(1-x\*x));

}

void LP\_Runge(double a, double b) {

int N = 2;

double h = (b - a) / N;

double Q\_h, Q\_h\_2;

double x\_i;

do {

Q\_h = 0;

x\_i = a;

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

x\_i = a + h \* i;

Q\_h += F(x\_i);

}

Q\_h \*= h;

h /= 2;

N \*= 2;

Q\_h\_2 = 0;

x\_i = a;

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

x\_i = a + h \* i;

Q\_h\_2 += F(x\_i);

}

Q\_h\_2 \*= h;

} while (abs(Q\_h\_2 - Q\_h) > E);

cout << "Полученное приближенное значение:\nI = " << fixed << showpoint << setprecision(9) << Q\_h\_2 << endl;

cout << "Значение шага h: " << scientific << h << endl;

cout << "Количество разбиений N: " << N << endl;

cout << "Невязка: " << scientific << abs(Q\_h\_2 - I\_t) << endl;

}

void Trap(double a, double b) {

int N = 1184;

double h = (b - a) / N;

double I = 0;

for (int i = 1; i < N; i++) {

I += F(a + i \* h);

}

I += (F(a) + F(b)) / 2;

I \*= h;

cout << "Полученное приближенное значение:\nI = " << fixed << showpoint << setprecision(9) << I << endl;

cout << "Значение шага h: " << scientific << h << endl;

cout << "Количество разбиений N: " << N << endl;

cout << "Невязка: " << scientific << abs(I\_t - I) << endl;

}

void SP(double a, double b) {

int N = 837;

double h = (b - a) / N;

double I = 0;

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

I += F(a + i \* h + h/2);

}

I \*= h;

cout << "Полученное приближенное значение:\nI = " << fixed << showpoint << setprecision(9) << I << endl;

cout << "Значение шага h: " << scientific << h << endl;

cout << "Количество разбиений N: " << N << endl;

cout << "Невязка: " << scientific << abs(I\_t - I) << endl;

}

void NAST(double a, double b) {

int N = 4;

double\* X = new double[4];

double\* A = new double[4];

X[0] = -0.8611363115;

X[1] = -0.3399810435;

X[2] = -X[1];

X[3] = -X[0];

A[0] = 0.3468548451;

A[1] = 0.6521451548;

A[2] = A[1];

A[3] = A[0];

double I = 0;

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

double x\_i = (a + b + (b - a) \* X[i])/2;

I += A[i] \* F(x\_i);

}

I \*= (b-a)/2;

cout << "Полученное приближенное значение:\nI = " << fixed << showpoint << setprecision(9) << I << endl;

cout << "Количество разбиений N: " << N << endl;

cout << "Невязка: " << scientific << abs(I\_t - I) << endl;

}

**Main.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#define \_USE\_MATH\_DEFINES

#include<cmath>

#include <iostream>

#include "Header.h"

#include <fstream>

#include<time.h>

using namespace std;

int main()

{

setlocale(LC\_ALL, "ru");

cout << "1.\nКФ ЛП:" << endl;

LP\_Runge(-0.5, 0.7);

cout << "2.\nКФ Т:" << endl;

Trap(-0.5, 0.7);

cout << "КФ CП:" << endl;

SP(-0.5, 0.7);

cout << "2.\nКФ НАСТ при n=3:" << endl;

NAST(-0.5, 0.7);

system("pause");

return 0;

}

1.

Краткие теоретические сведения.

Квадратурная формула ЛП:

Применение правила Рунге для КФ ЛП:

,.

1. Считаем и по выбранной квадратурной формуле.
2. , где m - порядок точности квадратурной формулы (для формулы левых прямоугольников m = 1)
3. Сравниваем с точностью . Если , то полагаем значение интеграла равным . Иначе повторить алгоритм.

Результаты

Полученное приближенное значение:

Значение шага h:

Количество разбиений N:

Невязка:

Выводы.

В данном пункте задания мы применили правило Рунге, позволяющее приближенно вычислить остаток и оценить погрешность при приближённом вычислении интегралов. Из результатов видно, что с помощью этого правила получилось вычислить интеграл с точностью даже большей, чем неоходимая, что подтверждает действенность этого метода. Тем не менее для достижения необходимой точности понадобилось довольно большое количество разбиений, но это может быть обусловлено применением формулы левых прямоугольников, чья алгебраическая степень точности – 0.

2.

Краткие теоретические сведения

Составная квадратурная формула трапеций:

Оценим необходимое для достижения требуемой точности число разбиений:

Cоставная квадратурная формула средних прямоугольников:

Оценим необходимое для достижения требуемой точности число разбиений:

Результаты.

Составная квадратурная формула трапеций:

Полученное приближенное значение:

Значение шага h:

Количество разбиений N:

Невязка:

Cоставная квадратурная формула средних прямоугольников:

Полученное приближенное значение:

Значение шага h:

Количество разбиений N:

Невязка:

Выводы.

В обоих случаях – и в формуле трапеций, и в формуле средних прямоугольников, мы получили точность даже большую, чем заданная. Это объясняется тем, что количество разбиений было оценено сверху, следовательно, оно могло быть и меньшим.

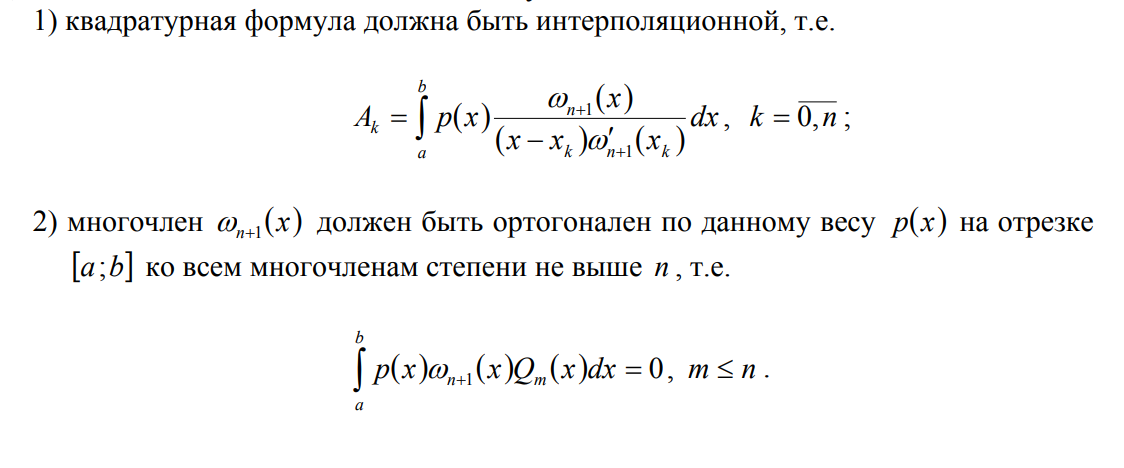
Как видно, количество разбиений зависит от вида остатка, и так как остаток для формулы средних прямоугольников меньше, то потребовалось и меньшее число разбиений. Но обеих формул одинаковая алгебраическая степень точности, поэтому это количество различается не сильно.

При этом для достижения необходимой точности потребовался куда больший шаг, чем при истользовании формулы левых прямоугольников, что опять же обусловлено более высокой АСТ – 1.

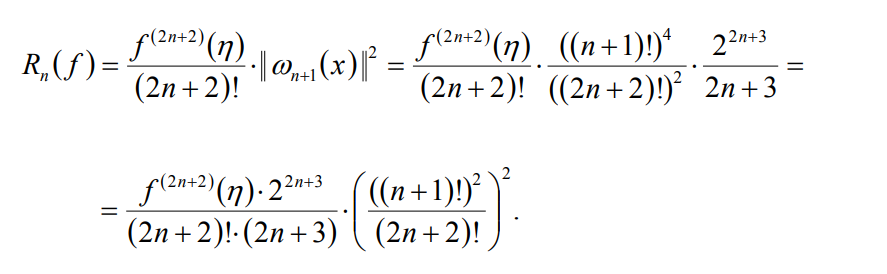
3.

Краткие теоретические сведения

Для достижения наивысшей алгебраической степени точности, т.е. чтобы формула с n+1 узлами была точной для любых алгебраических многочленов до степени (2n+1) включительно, необходимо и достаточно:



Формула для остатка КФ на отрезке [-1; 1]:



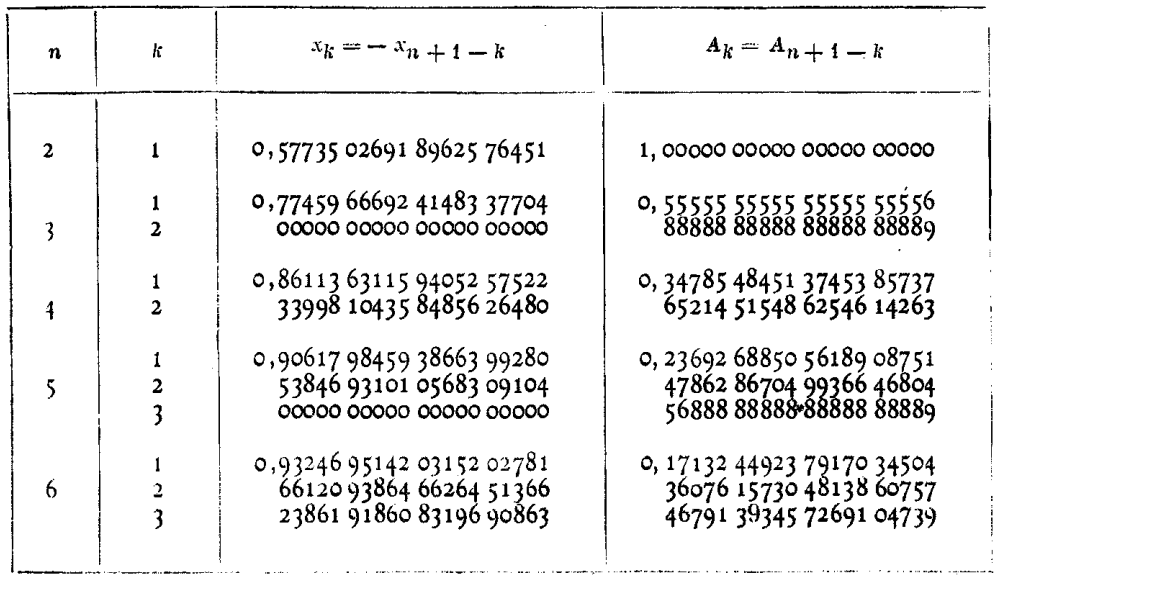
Для вычисления остатка на промежутке [a; b] необходимо умножить остаток на отрезке [-1; 1] на

Итак, в задании необходимо применить КФ НАСТ при n = 3, т.е. для четырёх узлов.

Найдём узлы, удовлетворяющие условию (2) критерия для НАСТ, как:

где

А так же найдём коэффициенты . Все значения возьмём из таблицы:



Т.к. в таблице предоставлены значения и для отрезка [-1; 1], поэтому при вычисление произведем необходимую замену:

Оценим остаток по формуле для n = 3:

Результаты.

Полученное приближенное значение:

Количество разбиений N:

Невязка:

Выводы.

Используя формулу НАСТ Гаусса, удалось всего при четырех разбиениях достичь точность порядка . По сравнению со всеми предыдущими формулами для нахождения достаточно точного решения необходимо очень мало разбиений, что обусловлено высокой алгебраической степенью точности – при n = 3 АСТ = 7. Однако главный недостаток метода состоит в нахождении необходимых коэффициентов и узлов, что весьма трудоёмко.

Заметим, что оценка остатка сверху получилась крайне грубой:

Столь грубую оценку сверху можно объяснить только поведением восьмой производной, которая на правом конце отрезка принимает значение .

4.

Таким образом, сравнивая результаты пунктов 1-3, можно убедиться в том, насколько сильно влияет алгебраическая степень точности на величину шага и количество разбиений, необходимых для достижения определённой точности.