Министерство образования и науки Российской Федерации

Санкт-Петербургский Государственный Политехнический Университет

—

Факультет Технической Кибернетики

Кафедра «Компьютерные системы и программные технологии»

ЛАБОРАТОРНАЯ РАБОТА № 1

Вариант №4

по дисциплине «Вычислительная математика»

Выполнил:

студент гр. 3530904/10001 Петров В.Д.

Руководитель:

профессор С.М. Устинов

Санкт-Петербург

2023

**Задание**

**Изображение выглядит как текст

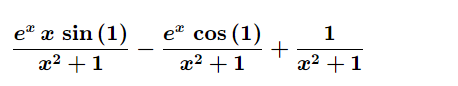
Автоматически созданное описание**

**Ход работы**

Находим нужные точки через программу QUANC8.

Через функцию spline() находим коэффициеты b, c, d.

Находим аналитически интеграл по t. Его вид:



По формуле Xk подставлять в seval, lagrange, и в саму найденную аналитически функцию. По результатам построить графики и выяснить, какая функция выдает более точные значения.

**Листинг программы**

#include <cmath>

#include <iostream>

#include <iomanip>

#include "cmath.h"

#include "spline.c"

#include "quanc8.c"

using namespace std;

static double Xvar\_base = 0.0;

double baseFunc(double t) { return exp(Xvar\_base \* t) \* sin(t); }

double func\_Analytic(double x) { return (exp(x)\*x\*sin(1)-exp(x)\*cos(1)+1) / (pow(x,2)+1); }

double lagrange(double\* Fi, double\* Xi, double X, int m)

{

double result = 0.0;

for (int i = 0; i <= m; i++)

{

double temp = 1.0;

for (int j = 0; j <= m; j++)

{

if (i != j)

{

temp \*= (X - Xi[j]) / (Xi[i] - Xi[j]);

}

}

result += temp \* Fi[i];

}

return result;

}

int main()

{

double h = 0.2;

double end1 = 0.0;

double end2 = 1.0;

double epsabs = 1.0e-18;

double epsrel = 1.0e-18;

double errest, posn = 0.0;

int nfe, flag = 0.0;;

double Xi[11];

double Fi[11];

int n = 0;

for (double x = 0.0; x <= 2.0; x += h)

{

double result = 0.0;

Xvar\_base = x;

quanc8(&baseFunc, end1, end2, epsabs, epsrel, &result, &errest, &nfe, &posn, &flag);

Xi[n] = x;

Fi[n] = result;

n++;

}

cout << "xk \tf(x) \tspline \tlagrange\n\n";

double b[11], c[11], d[11];

int iflag = 0;

spline(n, 0, 0, 0, 0, Xi, Fi, b, c, d, &iflag);

int last = -1;

for (int k = 1; k <= 10; k++)

{

double xk = h \* (k - 0.5);

cout << xk << " \t" << func\_Analytic(xk) << " \t"

<< seval(n, xk, Xi, Fi, b, c, d, &last) << " \t"

<< lagrange(Fi, Xi, xk, 10) << "\n";

}

}

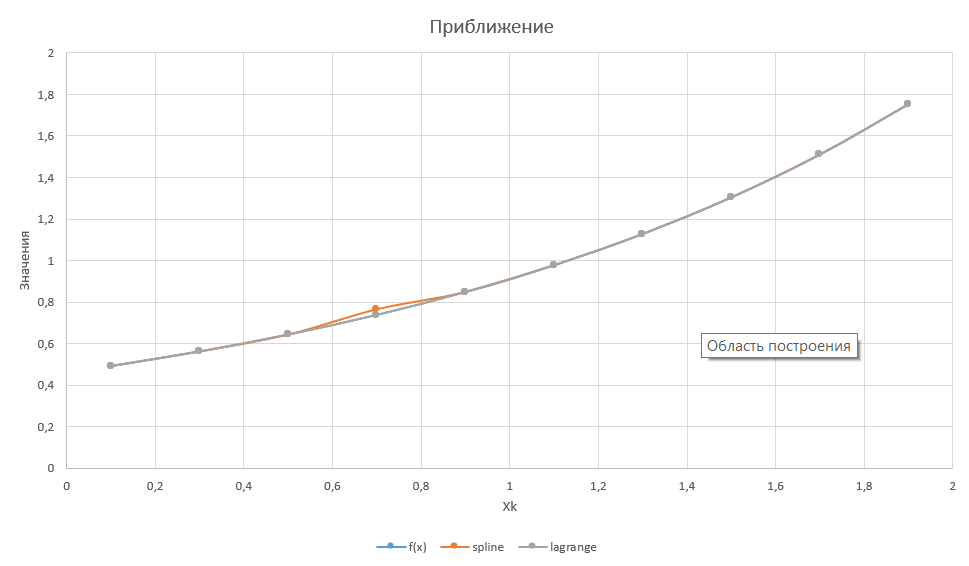
**Результаты работы программы**

**Изображение выглядит как текст

Автоматически созданное описание**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Сравнение Результатов**

****

По результатам программы значения функций lagrange и func\_Analytic(она же f(x) – вычисленный аналитически интеграл) полностью совпали. По графику видно небольшое отклонение результатов сплайн-интерполяции.

**Вывод**

По результатам работы программы видно, что наиболее точные значения получены при помощи использования полинома Лагранжа. SPLINE-функция показала небольшие отклонения в четвертой точке. Следовательно, для небольшого числа точек целесообразнее использовать полином Лагранжа.