|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  |
| МИНОБРНАУКИ РОССИИ | | | |
| Федеральное государственное бюджетное образовательное учреждениевысшего образования"Российский технологический университет"МИРЭА | | | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Институт Кибернетики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
| Кафедра программного обеспечения систем радиоэлектронной аппаратуры  при АО «Концерн радиостроения «ВЕГА» | | | |

|  |  |
| --- | --- |
| **Отчет по лабораторной работе №5** | |
| **по дисциплине** | |
| **«**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Численные методы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**»** | |
| **Вариант № 7** | |
| Студент 3-го курса  группы КМБО-02-16 | Миронов Д.А. |
| Преподаватель | Даева С.Г |
|  |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| Работа представлена к защите | «\_\_»\_\_\_\_\_\_\_201\_\_\_ г. |  |
|  |  |  |
| «Допущен к защите» | «\_\_»\_\_\_\_\_\_\_201\_\_\_ г. |  |

Москва 2019

Содержание

Оглавление

[Задание 3](#_Toc10533776)

[Теоретическая часть 4](#_Toc10533777)

[Практическая часть 12](#_Toc10533781)

[Приложение 1 13](#_Toc10533782)

[Приложение 2 13](#_Toc10533783)

# Задание

1. **Пункт 1**
2. С помощью интерполяционных формул а) Ньютона, б) Стирлинга и в) Ла-
3. гранжа найти значения первой и второй производных при данных значени-
4. ях аргумента для функции, заданной таблично.

|  |  |
| --- | --- |
| Таблица значений функции | |
| X | Y |
| 2.4 | 3.526 |
| 2.6 | 3.782 |
| 2.8 | 3.945 |
| 3.0 | 4.043 |
| 3.2 | 4.104 |
| 3.4 | 4.155 |
| 3.6 | 4.222 |
| 3.8 | 4.331 |
| 4.0 | 4.507 |
| 4.2 | 4.775 |
| 4.4 | 5.159 |
| 4.6 | 5.683 |

|  |  |  |
| --- | --- | --- |
| Таблица аргументов | | |
| Формула Ньютона | Формула Стирлинга | Многочлен Лагранжа |
| х=3.32+0.02\*n  n=1,3,...11 | х=3.82+0.02\*n  n=1,3,...11 | х=4.32+0.02\*n  n=1,3,...11 |

**Пункт 2**

1) Вычислить определенный интеграл в пределах от *a* до *b* по формуле

трапеций с точностью 10\*\*(-3) для приведенных в варианте подинтеграль-

ных функций.

2) Вычислить определенный интеграл в пределах от *a* до *b* по формуле

Симпсона для n=8, оценив погрешность результата для приведенных в вари-

анте подинтегральных функций.

3) Вычислить опредеденный интеграл в пределах от *a* до *b* по формуле

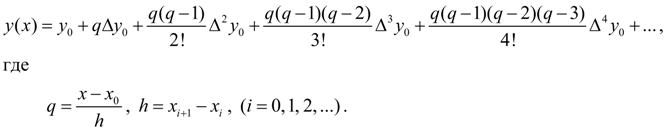
Гаусса, применяя для оценки точности двойной пересчет при n=4 и n=5 для

приведенных в варианте подинтегральных функций.

|  |  |  |
| --- | --- | --- |
| Таблица подинтегральных функций и пределов интегрирования | | |
| Формула трапеций | Формула Симпсона | Формула Гаусса |
| dx/(3x\*\*2-1)  a = 1.4; b = 2.1 | lg(x\*\*2+1)dx/x  a = 0.8; b = 1.6 | sqrt(x\*\*2+1)dx/(x+2)  a = 0.2; b = 2.4 |

# Теоретическая часть

**Дифференцирование на основе первой интерполяционной формулы Ньютона**

Для нахождения первой и второй производных функции http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn001.png функцию у, заданную в равноотстоящих точках http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn002.png (i = 0, 1, 2, …, n) отрезка [a, b] значениями http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn003.png, приближенно заменяют интерполяционным многочленом Ньютона, построенным для системы узлов http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn004.png :(1)

Раскрывая скобки и учитывая, что

http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn006.png

получим:

http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn007.png.        (2)

Аналогично, учитывая

http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn008.png

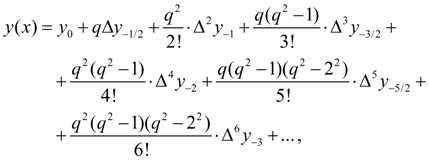
получим:

http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn009.png .              (3)

Таким же образом можно при необходимости вычислить производную функции http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn010.png любого порядка. Заметим, что при вычислении производных в фиксированной точке х в качестве http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn011.png следует брать ближайшее табличное значение аргумента.  
Можно также вывести формулы численного дифференцирования, основанные на второй интерполяционной формуле Ньютона.

**Дифференцирование на основе интерполяционной формулы Стирлинга**

Пусть http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn012.png– система равноотстоящих точек с шагом http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn013.png и http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn014.pngсоответствующие значения данной функции http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn015.png. Полагая  http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn016.png и заменяя функциюhttp://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn017.png интерполяционным полиномом Стирлинга, получим:

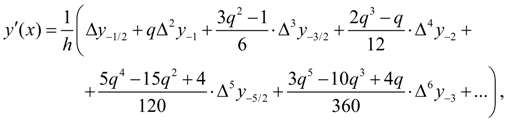
              (1)

где для краткости записи введены следующие обозначения:

http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn019.png

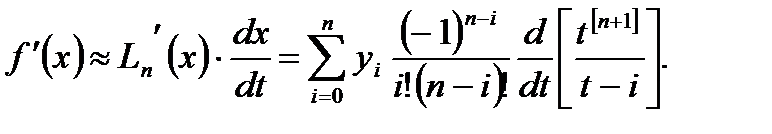
и т.д.

Из (8) с учетом того, что http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn020.png , следует:

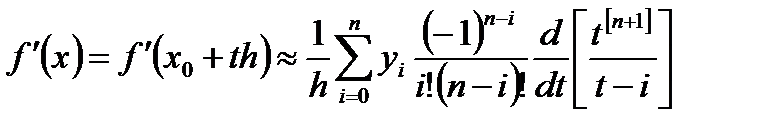
             (2)http://www.simumath.net/library/materials/Num_Dif_formulas/images/Eqn022.png (3)

**Дифференцирование на основе интерполяционного многочлена Лагранжа**

Будем дифференцировать многочлен Лагранжа по *х* как функцию от *t*:



Учитывая, что согласно (6.9) , а также  , получим окончательно:

(1)

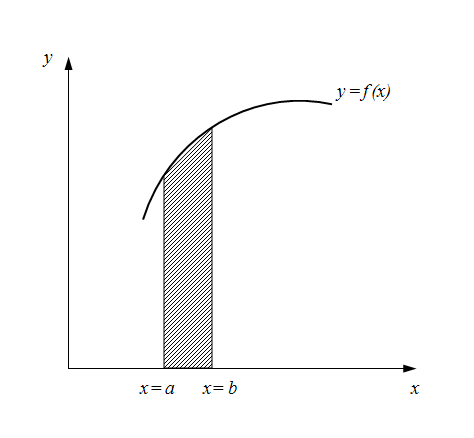
Пользуясь формулой (1), можно вычислять приближенные значения производной функции *f(x),* если она задана на отрезке [*a;b*] значениями в равноотстоящих узлах. Аналогично могут быть найдены производные функции *f(x)*высших порядков.

Формула для второй производной

(2)

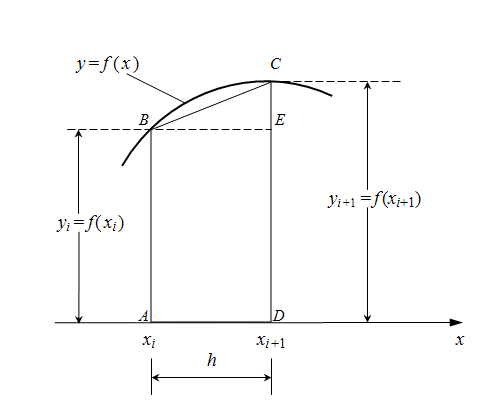
# Формула трапеций

Рассмотрим интеграл, представляющий собой, как известно, площадь под кривой http://www.simumath.net/library/materials/Num_Integr_formula_trap/images/Eqn001.pngна отрезке http://www.simumath.net/library/materials/Num_Integr_formula_trap/images/Eqn002.png (рис. 1)

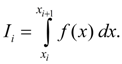
  
Рис. 1. Геометрическое представление численного интегрирования

Разобьем теперь интервал интегрирования  (a, b)  на  n  равных частей длиной http://www.simumath.net/library/materials/Num_Integr_formula_trap/images/Eqn003.png каждая ( h называется шагом интегрирования).

Рассмотрим один из этих интервалов (рис. 2).

  
Рис. 2. Один интервал численного интегрирования по методу трапеций

Площадь под кривой http://www.simumath.net/library/materials/Num_Integr_formula_trap/images/Eqn004.png между http://www.simumath.net/library/materials/Num_Integr_formula_trap/images/Eqn005.png равна:



Предположим, что шаг интегрирования  h достаточно мал, тогда эту площадь без существенной погрешности можно приравнять к площади трапеции ABCD. Так как http://www.simumath.net/library/materials/Num_Integr_formula_trap/images/Eqn007.png, получим:

http://www.simumath.net/library/materials/Num_Integr_formula_trap/images/Eqn008.png             (3)

Поскольку интеграл от суммы равен сумме интегралов (свойство аддитивности), то

http://www.simumath.net/library/materials/Num_Integr_formula_trap/images/Eqn009.png,             (4)

где http://www.simumath.net/library/materials/Num_Integr_formula_trap/images/Eqn010.png.

Подставляя (3) в (4), окончательно получим [3, 4]:

http://www.simumath.net/library/materials/Num_Integr_formula_trap/images/Eqn011.png.            (5)

Это и есть формула трапеций. Правило трапеций – один из простейших методов численного интегрирования,и хотя погрешности вычислений этим способом больше, чем в других методах, он пользуется большим спросом благодаря своей наглядности и простоте.

# Формула Симпсона

Правило Симпсона – один из наиболее широко известных и применяемых методов численного интегрирования. Он аналогичен правилу трапеций, поскольку также базируется на разбиении общего интервала интегрирования на более мелкие отрезки. Однако его отличие в том, что для вычисления площади через каждые три последовательные ординаты разбиения проводится квадратная парабола. Окончательный вид формулы Симпсона :

http://www.simumath.net/library/materials/Num_Integr_formula_Simp/images/Eqn001.png                (1)

Здесь *n* - четное число. Эта формула гораздо точнее формулы трапеций. Так, при интегрировании многочленов степени не выше третьей метод Симпсона дает точные значения интеграла.

# Квадратурная формула Гаусса

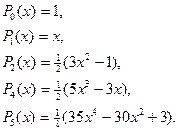
Полиномы вида https://helpiks.org/helpiksorg/baza9/485989017051.files/image2264.gifназываются полиномами Лежандра.

Свойства этих полиномов:

1. https://helpiks.org/helpiksorg/baza9/485989017051.files/image2266.gif , https://helpiks.org/helpiksorg/baza9/485989017051.files/image2268.gif ;

2. https://helpiks.org/helpiksorg/baza9/485989017051.files/image2270.gif , где https://helpiks.org/helpiksorg/baza9/485989017051.files/image2272.gif - любой полином степени k, меньшей n;

3. полином Лежандра https://helpiks.org/helpiksorg/baza9/485989017051.files/image2274.gif имеет n различных и действительных корней, которые расположены на интервале https://helpiks.org/helpiksorg/baza9/485989017051.files/image2276.gif .

Первые пять полиномов Лежандра: 

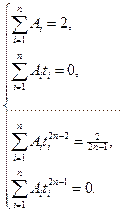
Рассмотрим функцию https://helpiks.org/helpiksorg/baza9/485989017051.files/image2280.gif , заданную на стандартном промежутке https://helpiks.org/helpiksorg/baza9/485989017051.files/image2282.gif . Нужно подобрать точки https://helpiks.org/helpiksorg/baza9/485989017051.files/image2195.gif и коэффициенты https://helpiks.org/helpiksorg/baza9/485989017051.files/image2284.gif , чтобы квадратурная формула

https://helpiks.org/helpiksorg/baza9/485989017051.files/image2286.gif (1)

была точной для всех полиномов https://helpiks.org/helpiksorg/baza9/485989017051.files/image2288.gif возможной наивысшей степени *N*. Так как в нашем распоряжении имеются *2n* постоянных https://helpiks.org/helpiksorg/baza9/485989017051.files/image2178.gif и https://helpiks.org/helpiksorg/baza9/485989017051.files/image2284.gif , а полином степени *2n-1*определяется *2n* коэффициентами, то эта наивысшая степень полинома в общем случае равна *N=2n-1*.

Для обеспечения равенства (1) необходимо и достаточно, чтобы оно было верным при https://helpiks.org/helpiksorg/baza9/485989017051.files/image2291.gif . Действительно, полагая https://helpiks.org/helpiksorg/baza9/485989017051.files/image2293.gif и https://helpiks.org/helpiksorg/baza9/485989017051.files/image2295.gif , будем иметь https://helpiks.org/helpiksorg/baza9/485989017051.files/image2297.gif .

Таким образом, учитывая соотношения https://helpiks.org/helpiksorg/baza9/485989017051.files/image2299.gif , заключаем, что для решения поставленной задачи достаточно определить постоянные https://helpiks.org/helpiksorg/baza9/485989017051.files/image2178.gif и https://helpiks.org/helpiksorg/baza9/485989017051.files/image1890.gif из системы *2n* уравнений:

 (2)

Система (2) нелинейная, и ее решение обычным путем представляет большие трудности.

Рассмотрим полиномы https://helpiks.org/helpiksorg/baza9/485989017051.files/image2304.gif , где https://helpiks.org/helpiksorg/baza9/485989017051.files/image2306.gif - полином Лежандра. Т.к. степени этих полиномов не превышают *2n-1*, то на основании системы (2) для них должны быть справедлива формула (1) и https://helpiks.org/helpiksorg/baza9/485989017051.files/image2308.gif .

С другой стороны, в силу свойства ортогональности полиномов Лежандра выполнены неравенства:

https://helpiks.org/helpiksorg/baza9/485989017051.files/image2310.gif при https://helpiks.org/helpiksorg/baza9/485989017051.files/image2312.gif ,

поэтому

https://helpiks.org/helpiksorg/baza9/485989017051.files/image2314.gif (3).

Равенства (3) будут обеспечены при любых значениях https://helpiks.org/helpiksorg/baza9/485989017051.files/image1890.gif , если положить https://helpiks.org/helpiksorg/baza9/485989017051.files/image2316.gif , т.е. для достижения наивысшей точности квадратурной формулы (1) в качестве точек https://helpiks.org/helpiksorg/baza9/485989017051.files/image2178.gif достаточно взять нули соответствующего полинома Лежандра. Как известно, из свойства 3, эти нули действительны, различны и расположены на интервале https://helpiks.org/helpiksorg/baza9/485989017051.files/image2318.gif. Зная абсциссы https://helpiks.org/helpiksorg/baza9/485989017051.files/image2178.gif , легко можно найти из линейной системы первых *n* уравнений системы (2) коэффициенты *Аi* (i = 1, 2, …, n). Определитель этой подсистемы есть определитель Вандермонда

https://helpiks.org/helpiksorg/baza9/485989017051.files/image2320.gif

и, следовательно, https://helpiks.org/helpiksorg/baza9/485989017051.files/image1890.gif определяются однозначно.

Формула (1), где https://helpiks.org/helpiksorg/baza9/485989017051.files/image2178.gif - нули полинома Лежандра https://helpiks.org/helpiksorg/baza9/485989017051.files/image2306.gif и https://helpiks.org/helpiksorg/baza9/485989017051.files/image2284.gif определяются из системы (2), называется квадратурной формулой Гаусса.

Рассмотрим теперь использование квадратурной формулы Гаусса для вычисления общего интеграла https://helpiks.org/helpiksorg/baza9/485989017051.files/image2323.gif . Делая замену переменной https://helpiks.org/helpiksorg/baza9/485989017051.files/image2214.gif , получим https://helpiks.org/helpiksorg/baza9/485989017051.files/image2325.gif . Применяя к последнему интегралу, квадратурную формулу Гаусса получим:

https://helpiks.org/helpiksorg/baza9/485989017051.files/image2327.gif , (4)

где https://helpiks.org/helpiksorg/baza9/485989017051.files/image2329.gif , https://helpiks.org/helpiksorg/baza9/485989017051.files/image2178.gif - нули полинома Лежандра https://helpiks.org/helpiksorg/baza9/485989017051.files/image2306.gif , т.е. https://helpiks.org/helpiksorg/baza9/485989017051.files/image2331.gif .

Остаточный член формулы Гаусса (4) с n узлами выражается следующим образом:

https://helpiks.org/helpiksorg/baza9/485989017051.files/image2333.gif .

Отсюда получаем:

https://helpiks.org/helpiksorg/baza9/485989017051.files/image2335.gif ,

https://helpiks.org/helpiksorg/baza9/485989017051.files/image2337.gif ,

https://helpiks.org/helpiksorg/baza9/485989017051.files/image2339.gif ,

https://helpiks.org/helpiksorg/baza9/485989017051.files/image2341.gif ,

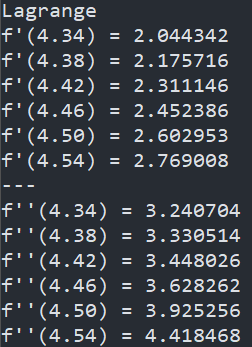
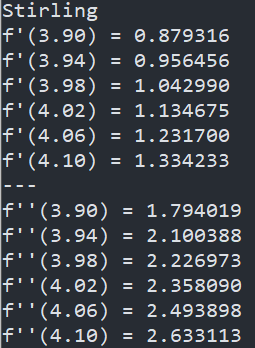
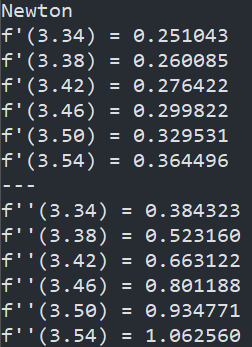
https://helpiks.org/helpiksorg/baza9/485989017051.files/image2343.gif .

# Практическая часть

**Задание1**

Для выполнения первого задания была написана программа “Lab5.1”. Язык «С++».

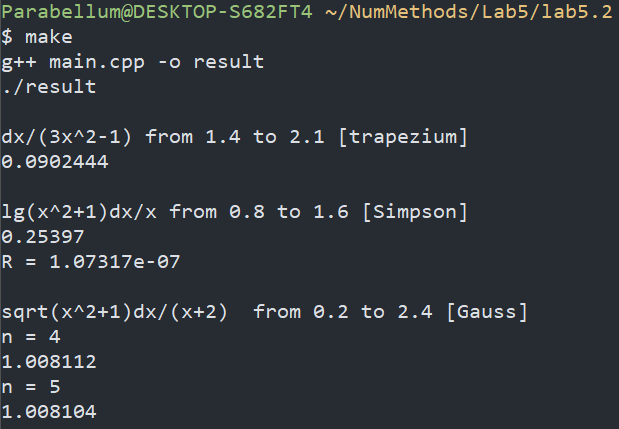
Результат работы программы:



**Задание 2**

Для выполнения первого задания была написана программа “Lab5.2”. Язык «С++».

Результат работы программы:



**Вывод**

Интерполяционные формулы Ньютона, Стирлинга и Лагранжа позволяют найти значения первой и второй производных функции, заданной таблично, с некоторой точностью. Формулы трапеции, Симпсона и Гаусса позволяют вычислить определенный интеграл с некоторой точностью.

# Приложение 1

#include <iostream>

#include <cmath>

#include <stdlib.h>

#include <iomanip>

#include <stdio.h>

#include <string.h>

using namespace std;

double X[] = {2.4, 2.6, 2.8, 3, 3.2, 3.4, 3.6, 3.8, 4, 4.2, 4.4, 4.6};

double Y[] = {3.526 ,3.782, 3.945, 4.043, 4.104, 4.155, 4.222, 4.331, 4.507 ,4.775, 5.159, 5.683};

unsigned factorial (unsigned n){

if (1==n || 0==n)

return 1;

else

return n\*factorial(n-1);

}

double C\_nk(int n, int k)

{

return factorial(n)/(factorial(k)\*(factorial(n-k)));

}

double F\_r(int i, int n, double \*mas)

{

double result = 0;

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

result += pow(-1,j)\*C\_nk(n,j)\*mas[i+n-j];

}

return result;

}

double q\_d1(double q, int n)

{

double buf1 = 0;

double buf2 = 1;

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

{

buf2 = 1;

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

if (i != j)

buf2 \*= (q-j);

buf1 += buf2;

}

return buf1;

}

double Newton\_d1(double x, double n, double \*mas)

{

double h = X[1] - X[0];

double q = (x - X[0])/h;

double result = 0;

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

double cur;

cur = q\_d1(q,i) \* F\_r(0, i+1, mas)/factorial(i+1);

result += cur;

}

result /= h;

return result;

}

double q\_d2(double q, int n)

{

double buf1 = 0;

double buf2 = 1;

int val = factorial(n)/(2\*factorial(n-2));

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

{

buf2 = 1;

int m = i/n + 1;

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

buf2 \*= (q-(m\*j + i)%n);

}

buf1 += buf2;

}

return buf1;

}

double Newton\_d2(double x, int n, double \*mas)

{

double h = X[1] - X[0];

double q = (x - X[0])/h;

double result = 0;

for (int i = 2; i <= n; i++){

double cur;

cur = 2\*q\_d2(q,i) \* F\_r(0, i, mas)/factorial(i);

result += cur;

}

result /= pow(h,2);

return result;

}

double new\_f\_q(int j, double q, int k)

{

double cur = 1;

for (int i = j; i < k; i++)

cur \*= (pow(q, 2) - pow(i ,2));

return cur;

}

double Stirling(double x, int n, double\* mas)

{

n = (n-1)/2;

double q = (x - X[n])/(X[1] - X[0]);

double y = 0;

for (int k = 0; k <= n; k++)

{

y += new\_f\_q(0, q,k) \* F\_r(n-k, 2\*k, mas) / factorial(2\*k);

if (k != n)

y += q \* new\_f\_q(1, q, k+1) \* (F\_r(n-k-1, 2\*k+1, mas) + F\_r(n-k, 2\*k+1, mas))/(2\* factorial(2\*k+1));

}

return y;

}

double Stirling\_d1(int k, int n, double\* mas, double h)

{

double result = 0;

n = (n-1)/2;

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

result += new\_f\_q(1,0,i+1)\*(F\_r(k-i-1,2\*i+1,mas) + F\_r(k-i, 2\*i+1, mas))/(2\*factorial(2\*i+1));

}

result /= h;

return result;

}

double Stirling\_d2(int k, int n, double\* mas, double h)

{

double result = 0;

n = (n-1)/2;

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

result += 2\*new\_f\_q(1,0,i)\*F\_r(k-i,2\*i,mas)/factorial(2\*i);

}

result /= pow(h,2);

return result;

}

double Lagrange\_d1(double x, int n, double \*masx, double \*masy)

{

double result = 0;

double buf1, buf2;

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

{

buf1 = 0;

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

{

if (i != j)

{

buf2 = 1;

for (int k = 0; k < n; k++)

{

if (k != i && k!= j)

buf2 \*= (x-masx[k])/(masx[i]-masx[k]);

}

buf1 += buf2/(masx[i] - masx[j]);

}

}

result += masy[i]\*buf1;

}

return result;

}

double Lagrange\_d2(double x, int n, double \*masx, double \*masy)

{

double result = 0;

double buf1, buf2, buf3;

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

{

buf1 = 0;

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

{

if (i != j)

{

buf2 = 0;

for (int l = 0; l < n; l++)

{

if (i != l && l != j)

{

buf3 = 1;

for (int k = 0; k < n; k++)

{

if (k != i && k != j && k != l)

buf3 \*= (x-masx[k])/(masx[i]-masx[k]);

}

buf2 += buf3/(masx[i] - masx[l]);

}

}

buf1 += buf2/(masx[i] - masx[j]);

}

}

result += masy[i]\*buf1;

}

return result;

}

int main()

{

cout << "Newton" << endl;

int n = sizeof(X)/sizeof(double);

for (int i = 1; i <= 12; i+=2)

{

double x = 3.32 + 0.02\*i;

printf("f'(%.2f) = %f\n",x , Newton\_d1(x,n,Y));

}

cout << "---" << endl;

for (int i = 1; i <=12; i+=2)

{

double x = 3.32 + 0.02\*i;

printf("f''(%.2f) = %f\n",x , Newton\_d2(x,n,Y));

}

cout << "\nStirling" << endl;

n = sizeof(X)/sizeof(double);

double Stirling\_dx[21];

double Stirling\_dy[21];

for (int i = 1; i <= 11; i++)

{

Stirling\_dx[i] = 3.82 + 0.02\*i;

Stirling\_dy[i] = Stirling(Stirling\_dx[i], n, Y);

}

double h = Stirling\_dx[1] - Stirling\_dx[0];

for (int i = 1; i <= 11; i+=2)

{

printf("f'(%.2f) = %f\n", Stirling\_dx[i], Stirling\_d1(i, n, Stirling\_dy, h));

}

cout << "---" << endl;

for (int i = 1; i <= 11; i+=2)

{

printf("f''(%.2f) = %f\n", Stirling\_dx[i], Stirling\_d2(i, n, Stirling\_dy, h));

}

cout << "\nLagrange" << endl;

n = sizeof(X)/sizeof(double);

for (int i = 1; i <= 11; i+=2)

{

double x = 4.32 + 0.02\*i;

printf("f'(%.2f) = %f\n", x, Lagrange\_d1(x,n, X, Y));

}

cout << "---" << endl;

for (int i = 1; i<=11; i+=2)

{

double x = 4.32 + 0.02\*i;

printf("f''(%.2f) = %f\n", x, Lagrange\_d2(x,n, X, Y));

}

return 0;

}

# Приложение 2

#include <iostream>

#include <cmath>

#include <stdlib.h>

#include <iomanip>

#include <stdio.h>

#include <string.h>

using namespace std;

double f1(double x)

{

return 1/(3\*x\*x-1);

}

double Trapezium()

{

double a = 1.4;

double b = 2.1;

int n = 66;

double h = (b-a)/n;

double result = 0;

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

{

result += h\*(f1(a + i\*h) + f1(a + (i+1)\*h))/2;

}

return result;

}

double f2(double x)

{

return (log10(x\*x+1))/x;

}

double Simpson(int k)

{

double a = 0.8;

double b = 1.6;

int n = k;

double h = (b-a)/n;

double result = 0;

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

{

switch (i%2)

{

case 0:

result += 2\*f2(a + i\*h);

break;

case 1:

result += 4\*f2(a + i\*h);

break;

}

}

result += f2(a) + f2(b);

result \*= h/3;

return result;

}

double f3(double x)

{

return sqrt(x\*x+1)/(x+2);

}

double Gauss\_4()

{

double x[4];

x[0] = -0.861136; x[1] = -0.339981;

x[2] = 0.339981; x[3] = 0.861136;

double a = 0.2;

double b = 2.4;

double result = 0;

for (int i = 0; i < 4; i++)

{

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

}

return (0.34785\*(f3(x[0]) + f3(x[3])) + 0.65215\*(f3(x[1]) + f3(x[2])));

}

double Gauss\_5()

{

double x[5];

x[0] = -0.90618; x[1] = -0.53847; x[2] = 0;

x[3] = 0.53847; x[4] = 0.90618;

double a = 0.2;

double b = 2.4;

double result = 0;

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

{

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

}

return (0.236927\*(f3(x[0]) + f3(x[4])) + 0.478629\*(f3(x[1]) + f3(x[3])) + 0.568889\*f3(x[2]));

}

int main()

{

cout << endl;

cout << "dx/(3x^2-1) from 1.4 to 2.1 [trapezium]" << endl;

cout << Trapezium() << endl;

cout << endl;

cout << "lg(x^2+1)dx/x from 0.8 to 1.6 [Simpson]" << endl;

cout << Simpson(8) << endl;

cout << "R = ";

cout << (Simpson(8) - Simpson(4))/15 << endl;

cout << endl;

cout << "sqrt(x^2+1)dx/(x+2) from 0.2 to 2.4 [Gauss]" << endl;

cout << "n = 4" << endl;

printf("%.6f\n", Gauss\_4());

cout << "n = 5" << endl;

printf("%.6f\n", Gauss\_5());

return 0;

}