МИНИСТЕРСТВО НАУКИ И ОБРАЗОВАНИЯ РФ

ПЕРМСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Отчет

По лабораторной работе №6 «Численное интегрирование»

Работу выполнили студенты группы

ПМИ-4-2020

Неверова Е.А. и Пуховкин В.И

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

Работу проверил заведующий кафедрой прикладной математики и информатики механико-математического факультета ПГНИУ

Русаков С.В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

« » ­­­­­­­­­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023

Пермь 2023

**Задание**

1. Найти значение интеграла с максимально возможной точностью

 (1)

1. Вычислить интеграл (1), используя обобщенные квадратурные формулы и метод Рунге, с точностью (относительной погрешностью) с помощью обобщенных формул

* трапеций;
* трапеций (модифицированной с помощью сплайна);
* Симпсона;
* Гаусса (трехточечной).

Оценить порядок аппроксимации квадратурной формулы с помощью метода Рунге. Итоговый результат (значение интеграла) выдавать с максимально допустимой точностью.

1. Провести оценку эффективности рассмотренных формул по точности и вычислительным затратам, учитывая общее число обращений к подынтегральной функции.

Вычисления произвести на отрезке [1,2] для функций, использованных в лабораторной №5.

Замечание.

При использовании формулы Гаусса выполнить не менее 3-х итераций (N=1, N=2, N=4).

**Исходные данные**

Вариант 1.

image001

Вариант 23.

image021

**Теоретическая справка**

**Составные квадратурные формулы:**

1. **Составная формула трапеции.**



*при*









1. **Составная формула трапеций (модифицированная).**

Если подынтегральную функцию приближенно заменить кубическим сплайном, то получим:



Нетрудно показать, что полученная квадратурная формула имеет четвертый порядок точности, как и формула Симпсона.

В случае равномерной сетки и  эта составная квадратурная формула принимает вид:



То есть, в этом случае, мы получаем небольшую модификацию составной формулы трапеций.

1. **Формула Симпсона.**



При







1. **Формулы Гаусса.**





Условия для определения параметров формул:



*Частные случаи.*

1) *n=0, A0,x0=?*



2) *n=1, A0,A1,x0,x1=?*

**

1. *n=2;*

**

*Свойства узлов:*

При *n* – четном,



При *n* – нечетном



*Погрешность*

**

**Метод Рунге оценки погрешности**





Пусть



Оценка погрешности



Получение эмпирической оценки порядка аппроксимации





**Решение**

Вариант 1.

Формула трапеций 0

N | h | Interral | Погрешность |Оценка погрешности (Рунге) | k |

1 | 1.0000 | 7.50000000 |0.11460991822207|

2 | 0.5000 | 7.41421356 |0.02882348059516|-0.02859547920896835654370|

4 | 0.2500 | 7.39260675 |0.00721667216369|-0.00720226947715740806188|1.9893 |

8 | 0.1250 | 7.38719493 |0.00180484497627|-0.00180394239580768567417|1.9973 |

16 | 0.0625 | 7.38584134 |0.00045125358088|-0.00045119713179649539913|1.9993 |

32 | 0.0312 | 7.38550290 |0.00011281604171|-0.00011281251305620779179|1.9998 |

64 | 0.0156 | 7.38541829 |0.00002820417584|-0.00002820395529094336251|2.0000 |

128 | 0.0078 | 7.38539713 |0.00000705105430|-0.00000705104051442380798|2.0000 |

256 | 0.0039 | 7.38539184 |0.00000176276422|-0.00000176276335957699826|2.0000 |

512 | 0.0020 | 7.38539052 |0.00000044069109|-0.00000044069104099264678|2.0000 |

1024| 0.0010 | 7.38539019 |0.00000011017278|-0.00000011017277164645141|2.0000 |

2048| 0.0005 | 7.38539011 |0.00000002754319|-0.00000002754319498402916|2.0000 |

Результат 7.38539010932112

Kobr = 2049

Формула трапеций 1

N | h | Interral | Погрешность |Оценка погрешности (Рунге) | k |

1 | 1.0000 | 7.38447547 |0.00091461187125|

2 | 0.5000 | 7.38533243 |0.00005765192817|0.000057130662872554674480|

4 | 0.2500 | 7.38538647 |0.00000361096714|0.000003602730735074999302|3.9871 |

8 | 0.1250 | 7.38538986 |0.00000022580644|0.000000225677380131609562|3.9968 |

16| 0.0625 | 7.38539007 |0.00000001411480|0.000000014112776073697356|3.9992 |

Результат 7.38539006766313

Kobr = 17+2

Формула Симпсона

N | h | Interral | Погрешность |Оценка погрешности (Рунге)| k |

1| 1.0000 | 7.38561808 |0.00022800138620|

2| 0.5000 | 7.38540448 |0.00001440268653|-0.00001423991331085024577|

4| 0.2500 | 7.38539098 |0.00000090258046|-0.00000090000707153592430|3.9839 |

8| 0.1250 | 7.38539014 |0.00000005644908|-0.00000005640875849384959|3.9959 |

Результат 7.38539013822701

Kobr = 17

Формула Гаусса

N | h | Interral | Погрешность |Оценка погрешности (Рунге) | k |

1| 1.0000 | 7.38538993 |0.00000015617116|

2| 0.5000 | 7.38539008 |0.00000000247010|0.000000002439699320873279|

4| 0.2500 | 7.38539008 |0.00000000003872|0.000000000038593382458073|5.9822 |

Результат 7.38539008173921

Kobr = 21

Вариант 23.

Формула трапеций 0

N | h | Interral | Погрешность |Оценка погрешности (Рунге) | k |

1 | 1.0000 | 6.00000000 |0.53856464023898|

2 | 0.5000 | 5.59807621 |0.13664085159230|-0.13397459621556132014497|

4 | 0.2500 | 5.49572391 |0.03428854834609|-0.03411743441540189547556|1.9734 |

8 | 0.1250 | 5.47001557 |0.00858021202520|-0.00856944544029728839940|1.9932 |

16 | 0.0625 | 5.46358092 |0.00214555853956|-0.00214488449521225499686|1.9983 |

32 | 0.0312 | 5.46197178 |0.00053642124402|-0.00053637909851319176720|1.9996 |

64 | 0.0156 | 5.46156947 |0.00013410728679|-0.00013410465241155594728|1.9999 |

128 | 0.0078 | 5.46146889 |0.00003352694519|-0.00003352678053225588428|2.0000 |

256 | 0.0039 | 5.46144374 |0.00000838174402|-0.00000838173372367615779|2.0000 |

512 | 0.0020 | 5.46143746 |0.00000209543649|-0.00000209543584261950855|2.0000 |

1024| 0.0010 | 5.46143588 |0.00000052385916|-0.00000052385911194126789|2.0000 |

2048| 0.0005 | 5.46143549 |0.00000013096479|-0.00000013096478745922013|2.0000 |

4096| 0.0002 | 5.46143539 |0.00000003274120|-0.00000003274119819707266|2.0000 |

Результат 5.46143539250222

Kobr = 4097

Формула трапеций 1

N | h | Interral | Погрешность |Оценка погрешности (Рунге) | k |

1 | 1.0000 | 5.45069386 |0.01074150409507|

2 | 0.5000 | 5.46074968 |0.00068568449122|0.000670387973590476837424|

4 | 0.2500 | 5.46139227 |0.00004308567479|0.000042839921095290854570|3.9680 |

8 | 0.1250 | 5.46143266 |0.00000269648002|0.000002692612984430316818|3.9919 |

16| 0.0625 | 5.46143519 |0.00000016858674|0.000000168526218580211672|3.9980 |

32| 0.0312 | 5.46143535 |0.00000001053755|0.000000010536612589836143|3.9995 |

Результат 5.46143534922347

Kobr = 33+2

Формула Симпсона

N | h | Interral | Погрешность |Оценка погрешности (Рунге)| k |

1 | 1.0000 | 5.46410162 |0.00266625537673|

2 | 0.5000 | 5.46160647 |0.00017111393069|-0.00016634276306971192421|

4 | 0.2500 | 5.46144613 |0.00001076658490|-0.00001068982305246398252|3.9598 |

8 | 0.1250 | 5.46143603 |0.00000067404435|-0.00000067283603678210586|3.9898 |

16| 0.0625 | 5.46143540 |0.00000004214551|-0.00000004212658944643029|3.9975 |

Результат 5.46143540190653

Kobr = 33

Формула Гаусса

N | h | Interral | Погрешность |Оценка погрешности (Рунге) | k |

1| 1.0000 | 5.46143079 |0.00000457361996|

2| 0.5000 | 5.46143529 |0.00000007366436|0.000000071427866739469887|

4| 0.2500 | 5.46143536 |0.00000000115987|0.000000001150864997657303|5.9557 |

Результат 5.46143535860115

Kobr = 21

**Вывод**

**Вариант 1.**

|  |  |  |  |
| --- | --- | --- | --- |
| Формула | Необходимое n | Значение интеграла | Число обращений к f |
| Трапеции | 2048 | 7.38539010932112 | 2049 |
| Трапеции (мод.) | 16 | 7.38539006766313 | 19 |
| Симпсона | 8 | 7.38539013822701 | 17 |
| Гаусса (трехточечной) | 4 | 7.38539008173921 | 21 |
| Решение в Mathcad | - | 7.38539008177793 | - |

**Вариант 23.**

|  |  |  |  |
| --- | --- | --- | --- |
| Формула | Необходимое n | Значение интеграла | Число обращений к f |
| Трапеции | 4096 | 5.46143539250222 | 4097 |
| Трапеции (мод.) | 32 | 5.46143534922347 | 35 |
| Симпсона | 16 | 5.46143540190653 | 33 |
| Гаусса (трехточечной) | 4 | 5.46143535860115 | 21 |
| Решение в Mathcad | - | 5.46143535976102 | - |

По данным таблиц можно увидеть, что наиболее точное решение получилось по трехточечной формуле Гаусса. Самым неточным оказалось решение по формуле Симпсона.

Формула трапеции требует наибольшее число отрезков разбиения для достижения заданной точности и, отсюда, также и вычисления исходной функции во многих точках, что делает ее наименее эффективной.

**Код программы**

#include <iostream>

#include <fstream>

#include <iomanip>

#include <cmath>

using namespace std;

ofstream output;

const double I1 = 7.38539008177793;

const double I23 = 5.46143535976102;

// Исходные функции

double f1(double x, int var)

{

if (var == 1)

return (pow(2, x) + 5 \* x - 3);

if (var == 23)

return (pow(3, x) + 2 \* x - 3);

}

// Первые производные

double f1\_proiz(double x, int var)

{

if (var == 1)

return (pow(2, x) \* log(2) + 5);

if (var == 23)

return (pow(3, x) \* log(3) + 2);

}

//Формула трапеций 0

void trapezoid\_formula\_0(double a, double b, int var)

{

int N = 1,

kobr = 0;

double x0 = a, x1 = b;

double h = 1,

h\_prev = 1,

ot\_pogr,

pogr\_runge = 0.1,

pogr = 0,

k1 = 2,

k = 0,

alpha = 0,

s\_prev\_2 = 0,

s\_prev\_1 = 0;

output << endl << "Формула трапеций 0" << endl;

output << setw(6) << "N |" << setw(9) << "h |"

<< setw(14) << "Interral |" << setw(17) << "Погрешность |"

<< setw(27) << "Оценка погрешности (Рунге)|" << setw(8) << "k |" << endl;

double s0 = 0,

s1 = 0,

s = 0,

sum = 0,

eps = 0.00000001;

s0 = f1(x0, var) + f1(x1, var);

kobr += 2;

do

{

sum = s1;

s1 = 0;

h = (b - a) / N;

for (int j = 1; j <= N - 1; j += 2)

{

s1 += f1(a + j \* h, var);

kobr++;

}

s1 += sum;

s = h \* (s0 / 2 + s1);

if (N > 1)

{

alpha = h / h\_prev;

if (N > 2)

{

k = (1 / log(alpha)) \* log((s - s\_prev\_2) / (s\_prev\_1 -

s\_prev\_2) - 1);

}

pogr\_runge = (pow(alpha, k1) / (1 - pow(alpha, k1))) \* (s -

s\_prev\_1);

}

if (var == 1)

pogr = abs(s - I1);

else pogr = abs(s - I23);

if (N == 1)

output << setw(5) << N << "|" << fixed << setprecision(4) << setw(7)

<< h << " |" << setprecision(8) << setw(12) << s << " |"

<< fixed << setprecision(14) << setw(16) << pogr << "|" << endl;

if (N == 2)

output << setw(5) << N << "|" << fixed << setprecision(4) << setw(7)

<< h << " |" << setprecision(8) << setw(12) << s << " |"

<< fixed << setprecision(14) << setw(16) << pogr << "|"

<< setw(25) << fixed << setprecision(23) << pogr\_runge << "|" << endl;

if (N > 2)

output << setw(5) << N << "|" << fixed << setprecision(4) << setw(7)

<< h << " |" << setprecision(8) << setw(12) << s << " |"

<< fixed << setprecision(14) << setw(16) << pogr << "|"

<< setw(25) << fixed << setprecision(23) << pogr\_runge << "|"

<< setw(5) << fixed << setprecision(4) << k << " |" << endl;

N \*= 2;

h\_prev = h;

s\_prev\_2 = s\_prev\_1;

s\_prev\_1 = s;

ot\_pogr = abs(pogr\_runge) / abs(s);

} while (ot\_pogr > eps);

output << "Результат " << fixed << setprecision(14) << s << endl;

output << "Kobr = " << kobr << endl;

}

//Формула трапуций 1

void trapezoid\_formula\_1(double a, double b, int var)

{

int N = 1,

kobr = 0;

double x0 = a, x1 = b;

double h = 1,

h\_prev = 1,

ot\_pogr,

pogr\_runge = 0.1,

pogr = 0,

k1 = 4,

k = 0,

alpha = 0,

s\_prev\_2 = 0,

s\_prev\_1 = 0;

output << endl << "Формула трапеций 1" << endl;

output << setw(6) << "N |" << setw(9) << "h |"

<< setw(14) << "Interral |" << setw(17) << "Погрешность |"

<< setw(27) << "Оценка погрешности (Рунге)|" << setw(8) << "k |" << endl;

double s0 = 0,

s1 = 0,

s = 0,

sum = 0,

s0\_proiz = 0,

eps = 0.00000001;

s0 = f1(x0, var) + f1(x1, var);

s0\_proiz = f1\_proiz(x0, var) - f1\_proiz(x1, var);

kobr += 2;

do

{

sum = s1;

s1 = 0;

h = (b - a) / N;

for (int j = 1; j <= N - 1; j += 2)

{

s1 += f1(a + j \* h, var);

kobr++;

}

s1 += sum;

s = h \* (s0 / 2 + s1) + h \* h / 12 \* s0\_proiz;

if (N > 1)

{

alpha = h / h\_prev;

if (N > 2)

{

k = (1 / log(alpha)) \* log((s - s\_prev\_2) / (s\_prev\_1 -

s\_prev\_2) - 1);

}

pogr\_runge = (pow(alpha, k1) / (1 - pow(alpha, k1))) \* (s -

s\_prev\_1);

}

if (var == 1)

pogr = abs(s - I1);

else pogr = abs(s - I23);

if (N == 1)

output << setw(5) << N << "|" << fixed << setprecision(4) << setw(7)

<< h << " |" << setprecision(8) << setw(12) << s << " |"

<< fixed << setprecision(14) << setw(16) << pogr << "|" << endl;

if (N == 2)

output << setw(5) << N << "|" << fixed << setprecision(4) << setw(7)

<< h << " |" << setprecision(8) << setw(12) << s << " |"

<< fixed << setprecision(14) << setw(16) << pogr << "|"

<< setw(25) << fixed << setprecision(24) << pogr\_runge << "|" << endl;

if (N > 2)

output << setw(5) << N << "|" << fixed << setprecision(4) << setw(7)

<< h << " |" << setprecision(8) << setw(12) << s << " |"

<< fixed << setprecision(14) << setw(16) << pogr << "|"

<< setw(25) << fixed << setprecision(24) << pogr\_runge << "|"

<< setw(5) << fixed << setprecision(4) << k << " |" << endl;

N \*= 2;

h\_prev = h;

s\_prev\_2 = s\_prev\_1;

s\_prev\_1 = s;

ot\_pogr = abs(pogr) / abs(s);

} while (ot\_pogr > eps);

output << "Результат " << fixed << setprecision(14) << s << endl;

output << "Kobr = " << kobr << "+" << 2 << endl;

}

// Формула Симпсона

void simpson\_formula(double a, double b, int var)

{

int N = 1,

kobr = 0;

double x0 = a, x1 = b;

double h = 1,

h\_prev = 1,

ot\_pogr,

pogr\_runge = 0.1,

pogr = 0,

k1 = 4,

k = 0,

alpha = 0,

s\_prev\_2 = 0,

s\_prev\_1 = 0;

output << endl << "Формула Симпсона" << endl;

output << setw(6) << "N |" << setw(9) << "h |"

<< setw(14) << "Interral |" << setw(17) << "Погрешность |"

<< setw(27) << "Оценка погрешности (Рунге)|" << setw(8) << "k |" << endl;

double s0 = 0,

s1 = 0,

s = 0,

sum = 0,

s1\_prev = 0,

s2 = 0,

eps = 0.00000001;

sum = f1(x0, var) + f1(x1, var);

kobr += 2;

do

{

s1 = 0;

h = (b - a) / N;

double x = a + 0.5 \* h;

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

{

s1 += f1(x + i \* h, var);

kobr++;

}

s1 = s1 \* 4;

sum += s1 - 0.5 \* s1\_prev;

s = sum \* (b - a) / (N \* 2) / 3;

s1\_prev = s1;

if (N > 1)

{

alpha = h / h\_prev;

if (N > 2)

{

k = (1 / log(alpha)) \* log((s - s\_prev\_2) / (s\_prev\_1 -

s\_prev\_2) - 1);

}

pogr\_runge = (pow(alpha, k1) / (1 - pow(alpha, k1))) \* (s -

s\_prev\_1);

}

if (var == 1)

pogr = abs(s - I1);

else pogr = abs(s - I23);

if (N == 1)

output << setw(5) << N << "|" << fixed << setprecision(4) << setw(7)

<< h << " |" << setprecision(8) << setw(12) << s << " |"

<< fixed << setprecision(14) << setw(16) << pogr << "|" << endl;

if (N == 2)

output << setw(5) << N << "|" << fixed << setprecision(4) << setw(7)

<< h << " |" << setprecision(8) << setw(12) << s << " |"

<< fixed << setprecision(14) << setw(16) << pogr << "|"

<< setw(25) << fixed << setprecision(23) << pogr\_runge << "|" << endl;

if (N > 2)

output << setw(5) << N << "|" << fixed << setprecision(4) << setw(7)

<< h << " |" << setprecision(8) << setw(12) << s << " |"

<< fixed << setprecision(14) << setw(16) << pogr << "|"

<< setw(25) << fixed << setprecision(23) << pogr\_runge << "|"

<< setw(5) << fixed << setprecision(4) << k << " |" << endl;

N \*= 2;

h\_prev = h;

s\_prev\_2 = s\_prev\_1;

s\_prev\_1 = s;

ot\_pogr = abs(pogr) / abs(s);

} while (ot\_pogr > eps);

output << "Результат " << fixed << setprecision(14) << s << endl;

output << "Kobr = " << kobr << endl;

}

// формула Гаусса

void gauss\_formula(double a, double b, int var)

{

int N = 1,

kobr = 0;

double h = 1,

h\_prev = 1,

ot\_pogr,

pogr\_runge = 0.1,

pogr,

k1 = 6,

k = 0,

alpha = 0,

s\_prev\_2 = 0,

s\_prev\_1 = 0,

eps = 0.00000001;

output << endl << "Формула Гаусса" << endl;

output << setw(6) << "N |" << setw(9) << "h |"

<< setw(14) << "Interral |" << setw(17) << "Погрешность |"

<< setw(27) << "Оценка погрешности (Рунге)|" << setw(8) << "k |" << endl;

double s = 0,

c1 = 5 / 9.0, c3 = 5 / 9.0, c2 = 8 / 9.0, x1 = -0.7745966692414834, x3 = 0.7745966692414834;

do

{

s = 0;

h = (b - a) / N;

int i = 0;

while (i < N)

{

s += f1(a + h \* i + h / 2 + h \* x1 / 2, var) \* c1 + f1(a + h \*

i + h / 2, var) \* c2 + f1(a + h \* i + h / 2 + h \* x3 / 2, var) \* c3;

i++;

kobr += 3;

}

s \*= h / 2;

if (N > 1)

{

alpha = h / h\_prev;

if (N > 2)

{

k = (1 / log(alpha)) \* log((s - s\_prev\_2) / (s\_prev\_1 -

s\_prev\_2) - 1);

}

pogr\_runge = (pow(alpha, k1) / (1 - pow(alpha, k1))) \* (s -

s\_prev\_1);

}

if (var == 1)

pogr = abs(s - I1);

else pogr = abs(s - I23);

if (N == 1)

output << setw(5) << N << "|" << fixed << setprecision(4) << setw(7)

<< h << " |" << setprecision(8) << setw(12) << s << " |"

<< fixed << setprecision(14) << setw(16) << pogr << "|" << endl;

if (N == 2)

output << setw(5) << N << "|" << fixed << setprecision(4) << setw(7)

<< h << " |" << setprecision(8) << setw(12) << s << " |"

<< fixed << setprecision(14) << setw(16) << pogr << "|"

<< setw(25) << fixed << setprecision(24) << pogr\_runge << "|" << endl;

if (N > 2)

output << setw(5) << N << "|" << fixed << setprecision(4) << setw(7)

<< h << " |" << setprecision(8) << setw(12) << s << " |"

<< fixed << setprecision(14) << setw(16) << pogr << "|"

<< setw(25) << fixed << setprecision(24) << pogr\_runge << "|"

<< setw(5) << fixed << setprecision(4) << k << " |" << endl;

N \*= 2;

h\_prev = h;

s\_prev\_2 = s\_prev\_1;

s\_prev\_1 = s;

ot\_pogr = abs(pogr) / abs(s);

} while (ot\_pogr > eps || N <= 4);

output << "Результат " << fixed << setprecision(14) << s << endl;

output << "Kobr = " << kobr << endl;

}

int main()

{

double a = 1, b = 2;

output.open("output.txt", ios::out);

trapezoid\_formula\_0(a, b, 23);

trapezoid\_formula\_1(a, b, 23);

simpson\_formula(a, b, 23);

gauss\_formula(a, b, 23);

}