|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Министерство науки и высшего образования РФ | | | | | | | | |  | |
|  | | | | |  | | |  | | | | |
|  | | | | ФГБОУ ВО «Пермский государственный национальный исследовательский университет» | | | | |  | | | |
|  | | | | |  | | |  | | | | |
|  | | | **Численные методы**  Лабораторная работа №6  «Численное интегрирование»  Варианты: №21b, №25b | | | | | | |  | | |
|  | | | | |  | | |  | | | | |
|  | Работу выполнили  Студенты гр. ПМИ-4:  Колесников А.С  Пухов Н.А. | | | | |  | Проверил  профессор, доктор физико-математических наук  Русаков С.В  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г. | | | | |  | |
|  | | | | |  | | |  | | | | |
|  | | | | | Пермь 2020 | | |  | | | | |

СОДЕРЖАНИЕ

[Задание 3](#_Toc57316028)

[Исходные данные 4](#_Toc57316029)

[Теоретическая справка 5](#_Toc57316030)

[Тестирование 11](#_Toc57316031)

[Краткие выводы 13](#_Toc57316032)

[Текст программы 14](#_Toc57316033)

Задание

1. Вычислить интеграл,



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.

Исходные данные

Вариант №21:



Вариант №25:



Теоретическая справка

Задача:

 (1)

p(x) – весовая функция,

n - порядок квадратурной формулы,

xi – узлы квадратурной формулы,

Ai – коэффициенты квадратурной формулы.

**Формулы Ньютона-Котеса**

Случай 

Идея: приблизим f(x) интерполяционным многочленом.







где



Погрешность интерполирования:



Частные случаи.

**Формула «левых» прямоугольников:**

n=0, x0=a



**Формула «правых» прямоугольников:**

n=0, x0=b



**Формула трапеций:**

n=1, x0=a, x1=b



**Формула Симпсона:**





**Алгебраический подход**

Обозначим



тогда



Пусть квадратурная формула дает точный результат для многочлена k-го порядка:



Получили СЛАУ относительно неизвестных коэффициентов квадратурной формулы.

В общем случае



**Квадратурные формулы Гаусса**





Условия для определения параметров формул:



Частные случаи.

1) n=0; A0, x0=?



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



n=2;



Свойства узлов:

При n – четном,



При n – нечетном



Погрешность



**Повышение точности интегрирования за счет разбиения отрезка на равные части (составные квадратурные формулы)**



**Составная формула трапеций**



При









**Составная формула Симпсона**



При







**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.

Тестирование

Вариант 21

**Формула трапеций**

|  |
| --- |
| | N | h| Integral| Оценка. погр.| k| |
| | 1| 1| 0.5| |
| | 2| 0.5| 0.414214| -0.02859547920896836| |
| | 4| 0.25| 0.392607| -0.0072022694771575564| 1.9893| |
| | 8| 0.125| 0.387195| -0.0018039423958077598| 1.9973| |
| | 16| 0.0625| 0.385841| -0.00045119713179638438| 1.9993| |
| | 32| 0.03125| 0.385503| -0.00011281251305600426| 1.9998| |
| | 64| 0.01563| 0.385418| -2.8203955291128402e-05| 2| |
| | 128| 0.007813| 0.385397| -7.0510405142942822e-06| 2| |
| | 256| 0.003906| 0.385392| -1.7627633591884202e-06| 2| |
| | 512| 0.001953| 0.385391| -4.4069104175129919e-07| 2| |
| |1024|0.0009766| 0.38539| -1.1017277308974134e-07| 2| |
| |2048|0.0004883| 0.38539| -2.7543194169865615e-08| 2| |
| |4096|0.0002441| 0.38539| -6.8857985424664037e-09| 2| |
| |8192|0.0001221| 0.38539| -1.7214499825612961e-09| 2| |
| Результат 0.385390083499375 |
| Kobr = 8193 |
| **Формула трапеций (модифицированная с помощью сплайна)** |
| | N | h| Integral| Оценка. погр.| k| |
| | 1| 1| 0.384475| |
| | 2| 0.5| 0.385332| 5.7130662872536169e-05| |
| | 4| 0.25| 0.385386| 1.8013653675208463e-05| 3.9871| |
| | 8| 0.125| 0.38539| 1.1283869004360032e-06| 3.9968| |
| | 16| 0.0625| 0.38539| 7.0563880664546261e-08| 3.9992| |
| | 32| 0.03125| 0.38539| 4.4108632672272092e-09| 3.9998| |
| | 64| 0.01563| 0.38539| 2.7568867559205995e-10| 3.9999| |
| Результат 0.385390081722789 |
| Kobr = 65 |
| **Формула Симпсона** |
| | N | h| Integral| Оценка. погр.| k| |
| | 1| 0.5| 0.385618| |
| | 2| 0.25| 0.385404| -1.4239913310791035e-05| |
| | 4| 0.125| 0.385391| -9.0000707156553022e-07| 3.9839| |
| | 8| 0.0625| 0.38539| -5.6408758516054057e-08| 3.9959| |
| | 16| 0.03125| 0.38539| -3.5280285026750845e-09| 3.999| |
| Результат 0.385390085306586 |
| Kobr = 33 |
| **Формула Гаусса (трехточечная)** |
| | N | h| Integral| Оценка. погр.| k| |
| | 1| 1| 0.38539| |
| | 2| 0.5| 0.38539| 2.4396993191110203e-09| |
| | 4| 0.25| 0.38539| 3.8593385982590151e-11| 5.9822| |
| Результат 0.385390081739214 |
| Kobr = 21 |

Вариант 25

**Формула трапеций**

|  |
| --- |
| | N | h| Integral| Оценка. погр.| k| |
| | 1| 1| 6.5| |
| | 2| 0.5| 6.09808| -0.13397459621556132| |
| | 4| 0.25| 5.99572| -0.034117434415401902| 1.9734| |
| | 8| 0.125| 5.97002| -0.008569445440297585| 1.9932| |
| | 16| 0.0625| 5.96358| -0.0021448844952119592| 1.9983| |
| | 32| 0.03125| 5.96197| -0.00053637909851289578| 1.9996| |
| | 64| 0.01563| 5.96157| -0.00013410465241155597| 1.9999| |
| | 128| 0.007813| 5.96147| -3.3526780532848001e-05| 2| |
| | 256| 0.003906| 5.96144| -8.3817337242682779e-06| 2| |
| | 512| 0.001953| 5.96144| -2.0954358420273897e-06| 2| |
| |1024|0.0009766| 5.96144| -5.2385911253338691e-07| 2| |
| |2048|0.0004883| 5.96144| -1.3096478834739855e-07| 2| |
| |4096|0.0002441| 5.96144| -3.2741195236477928e-08| 2| |
| Результат 5.961435392502223 |
| Kobr = 4097 |
| **Формула трапеций (модифицированная с помощью сплайна)** |
| | N | h| Integral| Оценка. погр.| k| |
| | 1| 1| 5.95069| |
| | 2| 0.5| 5.96075| 0.00067038797359047684| |
| | 4| 0.25| 5.96139| 0.00021419960547645425| 3.968| |
| | 8| 0.125| 5.96143| 1.3463064921855524e-05| 3.9919| |
| | 16| 0.0625| 5.96144| 8.4263109319711782e-07| 3.998| |
| | 32| 0.03125| 5.96144| 5.2683063245240191e-08| 3.9995| |
| Результат 5.961435349223467 |
| Kobr = 33 |
| **Формула Симпсона** |
| | N | h| Integral| Оценка. погр.| k| |
| | 1| 0.5| 5.9641| |
| | 2| 0.25| 5.96161| -0.00016634276306971192| |
| | 4| 0.125| 5.96145| -1.0689823052582408e-05| 3.9598| |
| | 8| 0.0625| 5.96144| -6.7283603666368208e-07| 3.9898| |
| | 16| 0.03125| 5.96144| -4.2126589446430292e-08| 3.9975| |
| Результат 5.961435401906528 |
| Kobr = 33 |
| **Формула Гаусса (трехточечная)** |
| | N | h| Integral| Оценка. погр.| k| |
| | 1| 1| 5.96143| |
| | 2| 0.5| 5.96144| 7.1427866725371818e-08| |
| | 4| 0.25| 5.96144| 1.1508650117553727e-09| 5.9557| |
| Результат 5.961435358601155 |
| Kobr = 21 |

Краткие выводы

**Вариант № 21**

|  |  |  |
| --- | --- | --- |
| Формула | Результат | Количество обращений |
| Трапеций | 0.385390083499375 | 8193 |
| Трапеций (модифиц.) | 0.385390081722789 | 65 |
| Симпсона | 0.385390085306586 | 33 |
| Гаусса | 0.385390081739214 | 21 |



Самый точный результат дала формула Гаусса и формула трапеций с модификацией, самый менее точный – формула Симпсона.

**Вариант №25**

|  |  |  |
| --- | --- | --- |
| Формула | Результат | Количество обращений |
| Трапеций | 5.961435392502223 | 4097 |
| Трапеций (модифиц.) | 5.961435349223467 | 33 |
| Симпсона | 5.961435401906528 | 33 |
| Гаусса | 5.961435358601155 | 21 |



Самый точный результат дала формула трапеций с модификацией, самый менее точный - формула Симпсона.

Таким образом, формула Гаусса и модифицированная формула трапеций являются наиболее точными, формула Симпсона оказалась наименее точна.Формула Гаусса вычисляет интеграл наиболее быстро *(с позиции вычислительных затрат эта формула также и наиболее точная, так как достигает требуемой точности быстрее остальных)*, формула трапеций работает медленнее остальных.При этом все методы реализуются достаточно просто, нет особой разницы в трудности реализации для какого-либо определенного метода.

Текст программы

#include <iostream>

#include <algorithm>

#include <math.h>

#include <fstream>

#include <functional>

#include <iomanip>

#include <cstdarg>

#include <list>

#include <vector>

using namespace std;

//Таблица

class Table

{

list<string> m\_colnames;

list<int> m\_precisions;

public:

Table(const list<string>& colnames, const list<int>& precisions)

: m\_colnames(colnames), m\_precisions(precisions)

{}

//распечатка заголовка таблицы

void printTableHeader()

{

cout << "|" << setw(4) << "N |";

auto it = m\_precisions.begin();

for (const auto& colname : m\_colnames)

{

cout << setw(\*it \* 2) << colname << "|";

it++;

}

cout << "\n";

}

//распечатка строки таблицы

void printTableRow(int N, const vector<double>& values)

{

cout << "|" << setw(4) << N << "|";

auto it = m\_precisions.begin();

for (const auto& value : values)

{

cout << setw(\*it \* 2) << setprecision(\*it) << value << "|";

it++;

}

cout << "\n";

}

};

//Функция и её производная

class Function

{

std::function<double(double)> m\_func;

std::function<double(double, int)> m\_derivative;

public:

int m\_callingCount = 0;

Function(std::function<double(double)> func, std::function<double(double, int)> derivative)

: m\_func(func), m\_derivative(derivative)

{}

double func(double x) {

m\_callingCount++;

return m\_func(x);

}

double derivative(double x, int n = 1) {

return m\_derivative(x, n);

}

};

//Отрезок

struct Interval

{

double m\_A;

double m\_B;

Interval() = default;

};

//Метод трапеций

class TrapezMethod

{

Function\* m\_func;

Interval m\_interval;

bool m\_modificiated;

Table\* m\_outTable;

double m\_eps = 0.00000001;

public:

TrapezMethod(Function\* func, Interval interval, bool modificiated , Table\* outTable)

: m\_func(func), m\_interval(interval), m\_modificiated(modificiated), m\_outTable(outTable)

{}

double calculate() {

double Sh0; //сумма на предпредыдущем шаге

double Sh1; //сумма на предыдущем шаге

double Sh2; //сумма на текущем шаге

int N = 1;

double h = m\_interval.m\_B - m\_interval.m\_A;

double sum = (m\_func->func(m\_interval.m\_A) + m\_func->func(m\_interval.m\_B)) / 2.0;

double dS = m\_modificiated \* (m\_func->derivative(m\_interval.m\_A) - m\_func->derivative(m\_interval.m\_B)) / 12.0;

Sh0 = h \* (sum + dS \* h);

m\_outTable->printTableRow(N, { h, Sh0 });

N = 2;

h /= 2.0;

sum += m\_func->func((m\_interval.m\_A + m\_interval.m\_B) / 2.0);

Sh1 = h \* (sum + dS \* h);

double err = (Sh1 - Sh0) / (4 \* (1 + m\_modificiated \* 3.0) - 1);

m\_outTable->printTableRow(N, { h, Sh1, err });

double k;

do {

N \*= 2;

h /= 2.0;

//сумма значений в новых вершинах

for (int i = 1; i < N; i += 2) {

sum += m\_func->func(m\_interval.m\_A + i \* h);

}

Sh2 = h \* (sum + dS \* h);

err = (Sh2 - Sh1) / 3.0;

k = log((Sh2 - Sh0) / (Sh1 - Sh0) - 1) / log(0.5);

m\_outTable->printTableRow(N, { h, Sh2, err, k });

Sh0 = Sh1;

Sh1 = Sh2;

} while (abs(err / Sh2) >= m\_eps);

return Sh2;

}

};

//Метод Симпсона

class SimpsonMethod

{

Function\* m\_func;

Interval m\_interval;

Table\* m\_outTable;

double m\_eps = 0.00000001;

public:

SimpsonMethod(Function\* func, Interval interval, Table\* outTable)

: m\_func(func), m\_interval(interval), m\_outTable(outTable)

{}

double calculate() {

double Sh0 = -1; //сумма на предпредыдущем шаге

double Sh1 = -1; //сумма на предыдущем шаге

double Sh2; //сумма на текущем шаге

int N = 1;

double h = m\_interval.m\_B - m\_interval.m\_A;

double sum0 = m\_func->func(m\_interval.m\_A) + m\_func->func(m\_interval.m\_B);

double err = 100000.0;

double k;

double sum = 0.0; //сумма старых вершин

do {

N \*= 2;

h /= 2.0;

//сумма значений в новых вершинах с множителем 4

auto sum1 = 0.0;

for (int i = 1; i < N; i += 2) {

sum1 += 4.0 \* m\_func->func(m\_interval.m\_A + i \* h);

}

Sh2 = h \* (sum0 + sum / 2.0 + sum1) / 3; //делим на 2 сумму значений в старых вершинах

sum += sum1;

if (Sh1 != -1) {

err = (Sh2 - Sh1) / 15.0;

if (Sh0 != -1) {

k = log((Sh2 - Sh0) / (Sh1 - Sh0) - 1) / log(0.5);

m\_outTable->printTableRow(N / 2, { h, Sh2, err, k });

}

else {

m\_outTable->printTableRow(N / 2, { h, Sh2, err });

}

}

else {

m\_outTable->printTableRow(N / 2, { h, Sh2 });

}

Sh0 = Sh1;

Sh1 = Sh2;

} while (abs(err / Sh2) >= m\_eps);

return Sh2;

}

};

//Метод Гаусса-3

class GaussMethod

{

Function\* m\_func;

Interval m\_interval;

Table\* m\_outTable;

double m\_eps = 0.000000001;

public:

GaussMethod(Function\* func, Interval interval, Table\* outTable)

: m\_func(func), m\_interval(interval), m\_outTable(outTable)

{}

double calculate() {

double Sh0 = -1; //сумма на предпредыдущем шаге

double Sh1 = -1; //сумма на предыдущем шаге

double Sh2; //сумма на текущем шаге

int N = 1;

double h = m\_interval.m\_B - m\_interval.m\_A;

double err = 100000.0;

double k;

do {

auto sum = 0.0;

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

double a = m\_interval.m\_A + h \* i;

double b = a + h;

double x\_left = linearTransformation(-sqrt(3.0 / 5.0), a, b);

double x\_center = linearTransformation(0.0, a, b);

double x\_right = linearTransformation(sqrt(3.0 / 5.0), a, b);

sum += 5.0 \* (m\_func->func(x\_left) + m\_func->func(x\_right)) + 8.0 \* m\_func->func(x\_center);

}

Sh2 = h \* sum / (2.0 \* 9.0);

if (Sh1 != -1) {

err = (Sh2 - Sh1) / 63.0;

if (Sh0 != -1) {

k = log((Sh2 - Sh0) / (Sh1 - Sh0) - 1) / log(0.5);

m\_outTable->printTableRow(N, { h, Sh2, err, k });

}

else {

m\_outTable->printTableRow(N, { h, Sh2, err });

}

}

else {

m\_outTable->printTableRow(N, { h, Sh2 });

}

N \*= 2;

h /= 2.0;

Sh0 = Sh1;

Sh1 = Sh2;

} while (abs(err / Sh2) >= m\_eps);

return Sh2;

}

private:

//линейное преобразование из [-1, 1] в отрезок [a, b] для точки x

double linearTransformation(double x, double a, double b) {

return ((1 - x) \* a + (1 + x) \* b) / 2.0;

}

};

int main()

{

system("chcp 1251");

Interval interval;

interval.m\_A = 1.0;

interval.m\_B = 2.0;

//функция и её n-ая производная

Function func21(

[](double x) { return pow(2, x) + 2 - 3 \* x; },

[](double x, int n) { return pow(2, x) \* pow(log(2), n) + (n == 1 ? -1 : 0); }

);

Function func25(

[](double x) { return pow(3, x) + 2 - x; },

[](double x, int n) { return pow(3, x) \* log(3) - 1; }

);

Function func = func21; //указать функцию (вариант)

Table table({ "h", "Integral", "Оценка. погр.", "k" }, { 4, 6, 17, 5 });

//1

printf("Формула трапеций 0\n");

table.printTableHeader();

TrapezMethod trapezMethod(&func, interval, false, &table);

auto result = trapezMethod.calculate();

printf("Результат %.15f\nKobr = %i\n", result, func.m\_callingCount);

func.m\_callingCount = 0;

//2

printf("Формула трапеций 1\n");

table.printTableHeader();

TrapezMethod trapezMethodMod(&func, interval, true, &table);

result = trapezMethodMod.calculate();

printf("Результат %.15f\nKobr = %i\n", result, func.m\_callingCount);

func.m\_callingCount = 0;

//3

printf("Формула Симпсона\n");

table.printTableHeader();

SimpsonMethod simpsonMethod(&func, interval, &table);

result = simpsonMethod.calculate();

printf("Результат %.15f\nKobr = %i\n", result, func.m\_callingCount);

func.m\_callingCount = 0;

//4

printf("Формула Гауса-3\n");

table.printTableHeader();

GaussMethod gaussMethod(&func, interval, &table);

result = gaussMethod.calculate();

printf("Результат %.15f\nKobr = %i\n", result, func.m\_callingCount);

func.m\_callingCount = 0;

}