Лабораторная работа №4

Вычисление определенного интеграла по методу прямоугольников, трапеций и Симпсона

Балахонский Дмитрий Михайлович 22928/3

Вариант 7

# Задание

A picture containing text

Description automatically generatedA picture containing text

Description automatically generatedДля каждого варианта в данной работе даны:

- функция на интервале . Границы интервала являются одновременно нижним и верхним пределом интегрирования;

- график этой функции (приводится для справки);

A picture containing text

Description automatically generated- точное значения для определенного интеграла от этой функции в заданных пределах.

Для второй функции этого же варианта приводятся аналогичные данные.

A picture containing text

Description automatically generatedA picture containing text

Description automatically generatedВ ходе выполнения лабораторной работы необходимо выполнить следующее.

Для каждой из двух функций и

A picture containing graphical user interface

Description automatically generatedText, letter

Description automatically generated

вычислить определенный интеграл на заданном интервале ,

используя последовательно три метода приближенного численного интегрирования:

- метод прямоугольников;

- метод трапеций;

- метод Симпсона.

Для каждого метода вычисления выполнять при следующих числах разбиения интервала:

10, 50, 100, 500 и 1000.

Рассчитать погрешность вычислений как разность между полученным и точным значениями определенного интеграла, взятую по модулю.

В программном коде предусмотреть отдельные функции для вычисления заданных в условии формул.

Функции вычисления интеграла различными методами не должны зависеть от вида подынтегральной функции, для этого использовать указатель на функцию.

Массивы в коде не использовать.

Схема построения алгоритма может быть следующая.

В главной функции программы объявить переменные для нижнего и верхнего предела интегрирования, для точного значения интеграла; объявить указатель на функцию.

Далее имеет смысл создать цикл на две итерации, соответственно числу заданных (подынтегральных) функций.

Внутри каждой итерации присвоить указателю на функцию имя соответствующей функции, задать значение точного интеграла, нижнего и верхнего пределов интегрирования.

Там же вызвать функцию вычисления интеграла для разных методов при разных числах разбиения интервала интегрирования, передав ей в качестве аргументов указатель на функцию, пределы интегрирования и значение точного интеграла. В теле этой функции перебираются все методы интегрирования, внутри каждого метода перебираются числа разбиений, производится обращение к другим функциям, реализующим методы интегрирования (метод прямоугольников, метод трапеций и метод Симпсона), и полученные таким образом результаты выводятся на консоль.

Пример вывода на консоль:

Первая функция:

Table

Description automatically generated with medium confidence

Table

Description automatically generated

Text

Description automatically generated

Вторая функция:

Table

Description automatically generated with low confidence

Table

Description automatically generated with medium confidence

Table

Description automatically generated

О методах численного интегрирования.

A picture containing text, watch, clock, gauge

Description automatically generatedИзображение выглядит как текст, датчик

Автоматически созданное описаниеДля каждого из трех упомянутых методов приближенного численного интегрирования выполняется разбиение интервала (где *a* – нижний предел интегрирования, *b* – верхний) на *N* частей.

При этом вычисляется величина шага по оси *X* как .

A picture containing clock, watch

Description automatically generatedText

Description automatically generated with low confidenceA picture containing clock

Description automatically generatedЕсли не рассматривать начальную точку интервала с координатой *a*, то получаем *N* точек деления с координатами , где . Таким образом получаем *N* **прямоугольников** с короткой горизонтальной стороной, равной шагу *s*, и длинной вертикальной стороной, соответствующей значению функции в этой точке .

Text

Description automatically generatedПроизведение сторон даст площадь элементарного прямоугольника, а суммирование всех площадей по даст приближенную площадь под кривой, то есть значение определенного интеграла.

A picture containing clock

Description automatically generatedA picture containing clock, watch

Description automatically generatedАналогичным образом проводятся вычисления по **методу трапеций**. Только в этом случае рассматривается элементарная прямоугольная трапеция с короткой горизонтальной высотой *s*, и вертикальными длинными основаниями, равными значениям функции в соседних точках разбиения и . Произведение полсуммы оснований на высоту (сторона *s*) дает площадь элементарной трапеции, а суммирование всех площадей по *k* даст искомый результат – приближенное значение определенного интеграла.

Изображение выглядит как часы

Автоматически созданное описаниеПри вычислении определенного интеграла по **методу Симпсона** число интервалов разбиения *N* должно быть четным (это соблюдено в условиях) и может быть представлено как

При этом рассматриваются прямоугольные криволинейные трапеции, у которых одна сторона образована куском параболы. После выполнения всех преобразований, получаем формулу для площади под кривой – определенного интеграла – по методу Симпсона:

Chart

Description automatically generated with medium confidence

A picture containing text

Description automatically generated

Первым слагаемым можно инициализировать переменную для интеграла, второе слагаемое вычисляется, естественно, в цикле.

# Ход выполнения

1. Для начала я импортировал библиотеки и объявил константы и входные данные

#include <iostream>  
#include "cmath"  
  
using namespace std;  
  
const double pi = 3.1415926535;  
  
const int N[5] = {10, 50, 100, 500, 1000};  
  
//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//  
// y1 = x^2\*e^(-x^2)  
const double a1 = 0;  
const double b1 = 3;  
const double I\_01 = 0.442918559423631;  
  
double function\_n1(double x) {  
 return pow(x, 2)\*exp(-pow(x, 2));  
}  
//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//  
  
//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//  
// y2 = (1+sin(x))^2  
const double a2 = -pi/2;  
const double b2 = 3/2\*pi;  
const double I\_02 = 9.424777960769379;  
  
double function\_n2(double x) {  
 return pow((1 + sin(x)), 2);  
}  
//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//

1. Функция для установки конкретного числа пробелов

string set\_spaces(int spaceQ){ // Табуляция для красивого вывода))  
 string result = "";  
 for (int i = 0; i < spaceQ; i++){  
 result += " ";  
 }  
 return result;  
}

1. Функции для подсчета интеграла функции

double rectangle\_method(unsigned short int n, double a, double b, double (\*f\_x)(double)) {  
 double integral = 0;  
 double s = (b-a)/n;  
 for (int k = 1; k<=n; k++) {  
 double w = s;  
 double h = f\_x(a + s \* k);  
 integral += w \* h;  
 }  
 return integral;  
}  
  
double trapezoid\_method(unsigned short int n, double a, double b, double (\*f\_x)(double)) {  
 double integral = 0;  
 double s = (b-a)/n;  
 for (int k = 1; k<=n; k++) {  
 double base\_1 = f\_x(a + s \* (k-1));  
 double base\_2 = f\_x(a + s \* k);  
 double h = s;  
 integral += (base\_1 + base\_2) / 2 \* h;  
 }  
 return integral;  
}  
  
double simpson\_s\_method(unsigned short int n, double a, double b, double (\*f\_x)(double)) {  
 double s = (b-a)/n;  
 int m = n/2;  
 double sum = 0;  
 for (int i = 1; i < m; i++) {  
 sum += 4\*f\_x(a+s\*(2\*i-1))+2\*f\_x(a+s\*2\*i);  
 }  
 double integral = s/3\*(f\_x(a)+f\_x(b)+4\*f\_x(a+s\*(2\*m-1)))+s/3\*sum;  
 return integral;  
}

1. Вывод результатов

int main() {  
 cout.fill(' ');  
 cout.precision(8);  
 // function\_n1  
 cout << endl << "Rectangle method" << endl;  
 cout << "Number of Partitions" << "\t" << "The Value of the Integral" << "\t" << "Calculation Error" << endl;  
 for (auto n:N) {  
 double integral = rectangle\_method(n, a1, b1, function\_n1);  
 cout.width(10); cout << n << set\_spaces(15);  
 cout.width(15); cout << integral << set\_spaces(17);  
 cout.setf(ios::scientific);  
 cout.width(15); cout << I\_01 - integral << endl;  
 cout.setf(ios::fixed);  
 }  
  
 cout << endl << "Trapezoid method" << endl;  
 cout << "Number of Partitions" << "\t" << "The Value of the Integral" << "\t" << "Calculation Error" << endl;  
 for (auto n:N) {  
 double integral = trapezoid\_method(n, a1, b1, function\_n1);  
 cout.width(10); cout << n << set\_spaces(15);  
 cout.width(15); cout << integral << set\_spaces(17);  
 cout.setf(ios::scientific);  
 cout.width(15); cout << I\_01 - integral << endl;  
 cout.setf(ios::fixed);  
 }  
  
 cout << endl << "Simpson's method" << endl;  
 cout << "Number of Partitions" << "\t" << "The Value of the Integral" << "\t" << "Calculation Error" << endl;  
 for (auto n:N) {  
 double integral = simpson\_s\_method(n, a1, b1, function\_n1);  
 cout.width(10); cout << n << set\_spaces(15);  
 cout.width(15); cout << integral << set\_spaces(17);  
 cout.setf(ios::scientific);  
 cout.width(15); cout << I\_01 - integral << endl;  
 cout.setf(ios::fixed);  
 }  
  
 // function\_n2  
 cout << endl << "Rectangle method" << endl;  
 cout << "Number of Partitions" << "\t" << "The Value of the Integral" << "\t" << "Calculation Error" << endl;  
 for (auto n:N) {  
 double integral = rectangle\_method(n, a2, b2, function\_n2);  
 cout.width(10); cout << n << set\_spaces(15);  
 cout.width(15); cout << integral << set\_spaces(17);  
 cout.setf(ios::scientific);  
 cout.width(15); cout << I\_02 - integral << endl;  
 cout.setf(ios::fixed);  
 }  
  
 cout << endl << "Trapezoid method" << endl;  
 cout << "Number of Partitions" << "\t" << "The Value of the Integral" << "\t" << "Calculation Error" << endl;  
 for (auto n:N) {  
 double integral = trapezoid\_method(n, a2, b2, function\_n2);  
 cout.width(10); cout << n << set\_spaces(15);  
 cout.width(15); cout << integral << set\_spaces(17);  
 cout.setf(ios::scientific);  
 cout.width(15); cout << I\_02 - integral << endl;  
 cout.setf(ios::fixed);  
 }  
  
 cout << endl << "Simpson's method" << endl;  
 cout << "Number of Partitions" << "\t" << "The Value of the Integral" << "\t" << "Calculation Error" << endl;  
 for (auto n:N) {  
 double integral = simpson\_s\_method(n, a2, b2, function\_n2);  
 cout.width(10); cout << n << set\_spaces(15);  
 cout.width(15); cout << integral << set\_spaces(17);  
 cout.setf(ios::scientific);  
 cout.width(15); cout << I\_02 - integral << endl;  
 cout.setf(ios::fixed);  
 }  
 return 0;  
}

Изображение выглядит как текст, табличка, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, табличка, снимок экрана

Автоматически созданное описание