**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по практическому заданию №6**

**по дисциплине «Вычислительная математика»**

Тема: **Численное интегрирование. Составные формулы прямоугольников, трапеций, Симпсона**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Виноградов К.А. |
| Преподаватель |  | Лисс А.Р. |

Санкт-Петербург

2017

**Цель работы.**

Изучение составных формул прямоугольников, трапеций и Симпсона как способов повышения точности численного интегрирования.

**Основные теоретические положения.**

Повышения точности численного интегрирования добиваются путем применения составных формул. Для этого при нахождении определенного интеграла отрезок  разбивают на четное  число отрезков длины

 (1)

и на каждом из отрезков длины  применяют соответствующую формулу. Таким образом получают составные формулы прямоугольников, трапеций и Симпсона.

На сетке , , , составные формулы имеют следующий вид:

* формула прямоугольников

; (2)

* формула трапеций

; (3)

* формула Симпсона

, (4)

где  - остаточные члены. При  приближенные значения интегралов для всех трех формул (в предположении отсутствия погрешностей округления) стремятся к точному значению интеграла.

Для практической оценки погрешности квадратурной можно использовать правило Рунге. Для этого проводят вычисления на сетках с шагом  и , получают приближенные значения интеграла  и  и за окончательные значения интеграла принимают величины:

* Для формулы прямоугольников:

; (5)

* Для формулы трапеций:

; (6)

* Для формулы Симпсона:

. (7)

За погрешность приближенного значения интеграла для формул прямоугольников и трапеций тогда принимают величину , а для формулы Симпсона .

**Постановка задачи.**

Используя квадратурные формулы прямоугольников, трапеций и Симпсона, вычисление значения интеграла

 (8)

и, применив правило Рунге, нахождение наименьшего значение  (наибольшее значение шага ), при котором каждая из указанных формул дает приближенное значение интеграла с погрешностью , не превышающей заданную.

**Выполнение работы.**

Cоставим программы-функции для вычисления интегралов по формулам прямоугольников, трапеций и Симпсона:

* + Для прямоугольников: double Rectangle(double a, double b, int n);
  + Для трапеций: double Trapeze(double a, double b, int n);
  + Для Симпсона: double Simpson(double a, double b, int n);

Составим подпрограмму-функцию для вычисления подынтегральной функции.

Составим головную программу, содержащую оценку по Рунге погрешности каждой из перечисленных ранее квадратурных формул, удваивающих  до тех пор, пока погрешность не станет меньше , и осуществляющих печать результатов значения интеграла и значения  для каждой формулы.

Все программы представлены в приложении А.

Проведём вычисления по программе с различной . Результаты приведены в табл. 1,2,3.

Таблица 1 – Результаты вычисления программы при 

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Результат | n | Точность Рунге |
| Прямоугольников | 1.63195 | 2 | 7.61517e-04 |
| Трапеций | 1.63178 | 2 | 1.43527e-03 |
| Симпсона | 1.63166 | 2 | 1.14396e-04 |

Таблица 2 – Результаты вычисления программы при 

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Результат | n | Точность Рунге |
| Прямоугольников | 1.63195 | 2 | 7.61517e-04 |
| Трапеций | 1.63186 | 4 | 3.36877e-04 |
| Симпсона | 1.63166 | 2 | 1.14396e-04 |

Таблица 3 – Результаты вычисления программы при 

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Результат | n | Точность Рунге |
| Прямоугольников | 1.63187 | 8 | 4.16173e-05 |
| Трапеций | 1.63187 | 8 | 8.29114e-05 |
| Симпсона | 1.63186 | 4 | 5.85084e-06 |

Сравним значения n для каждого метода при каждом рассмотренном . Результаты приведены в табл. 4.

Таблица 4 – Значения n из всех опытов

|  |  |  |  |
| --- | --- | --- | --- |
|  | n для прямоугольников | n для трапеций | n для Симпсона |
| 0.1 | 2 | 2 | 2 |

окончание таблицы 4

|  |  |  |  |
| --- | --- | --- | --- |
| 0.01 | 2 | 4 | 2 |
| 0.001 | 8 | 8 | 4 |

**Выводы.**

В ходе данной практической работы была составлена программа, позволяющая вычислять значение интеграла (8) с помощью составных формул прямоугольников, трапеций и Симпсона при различных значениях заданного .

Помимо этого, были найдены наименьшее значение  (наибольшее значение шага ) и погрешности вычислений по правилу Рунге.

Таким образом, наиболее точным методом для вычисления определённого интеграла является метод Симпсона, который совершает наименьшее число делений и имеет малую погрешность.

Приложение А

Код программы

#include <iostream>

#include <math.h>

#include <locale.h>

using namespace std;

extern double F(double x);

double Rectangle(double a, double b, int n);

double Trapeze(double a, double b, int n);

double Simpson(double a, double b, int n);

int main()

{

setlocale(LC\_ALL,"rus");

double I1, I2, eps, epsR;

int n;

cout << "\nВведите точность: eps=";

cin >> eps;

cout << "\nМетод прямоугольников";

n = 1;

do

{

n = n \* 2;

I1 = Rectangle(0, 1, n);

I2 = Rectangle(0, 1, n \* 2);

epsR = fabs(I1 - I2) / 3;

}while (eps<epsR);

I1 = I2 + epsR;

cout << "\nРезультат: " << I1 << "\nn = " << n << "\nТочность Рунге: " << epsR<<endl;

cout << "\nМетод трапеций";

n = 1;

do

{

n = n \* 2;

I1 = Trapeze(0, 1, n);

I2 = Trapeze(0, 1, n \* 2);

epsR = fabs(I1 - I2) / 3;

}while (eps<epsR);

I1 = I2 - epsR;

cout << "\nРезультат: " << I1 << "\nn = " << n << "\nТочность Рунге: " << epsR<<endl;

cout << "\nМетод Симпсона ";

n = 1;

do

{

n = n \* 2;

I1 = Simpson(0, 1, n);

I2 = Simpson(0, 1, n \* 2);

epsR = fabs(I1 - I2) / 15;

} while (eps<epsR);

I1 = I2 - epsR;

cout << "\nРезультат: " << I1 << "\nn = " << n << "\nТочность Рунге: " << epsR<<endl;

return 0;

}

double F(double x)

{

return exp(sin(x));

}

double Rectangle(double a, double b, int n)

{

double h, x, s; int i;

h = (b - a) / n;

x = a; s = 0;

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

{

s = s + F(x + h / 2);

x = x + h;

}

s = s\*h;

return (s);

}

double Trapeze(double a, double b, int n)

{

double h, x1, x2, s; int i;

h = (b - a) / n;

x1 = a; x2 = a + h; s = 0;

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

{

s = s + F(x1) + F(x2);

x1 = x2;

x2 = x2 + h;

}

s = s\*h / 2;

return (s);

}

double Simpson(double a, double b, int n)

{

double h, x1, x2, x3, s; int i;

h = (b - a) / n;

x1 = a; x2 = a + h; x3 = x2 + h; s = 0;

for (i = 0; i < n / 2; i++)

{

s = s + F(x1) + 4 \* F(x2) + F(x3);

x1 = x3;

x2 = x3 + h;

x3 = x2 + h;

}

s = s\*h / 3;

return (s);

}