 Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования «Казанский (Приволжский) федеральный университет» Институт вычислительной математики и информационных технологий

Кафедра прикладной математики и искусственного интеллекта



**Отчет**

 ПО СЕМЕСТРОВОЙ РАБОТЕ №2

«Вычисление интеграла с помощью квадратурных формул»

Вариант 15

Выполнил: студент гр. 09-121

Макарова К. Ю.

 Проверил: aссистент

Глазырина О.В.

Казань 2023

**СОДЕРЖАНИЕ**

[1. ЦЕЛЬ РАБОТЫ 3](#_Toc128325025)

[2. ИСХОДНЫЕ ДАННЫЕ И ЗАДАНИЕ 3](#_Toc128325026)

[3. ХОД РАБОТЫ 5](#_Toc128325027)

[4. ЗАКЛЮЧЕНИЕ 9](#_Toc128325028)

[5. ЛИСТИНГ 9](#_Toc128325029)

# **ЦЕЛЬ РАБОТЫ**

Изучить и сравнить способы приближенного вычисления одной из специальной функции математической физики с помощью составных квадратурных формул.

# **ИСХОДНЫЕ ДАННЫЕ И ЗАДАНИЕ**

Одна из специальных функций математической физики – функция ошибок, которая определяется следующим образом:

= .

Для данной функции необходимо:

1. Протабулировать на отрезке [a, b] с шагом с точностью ε, основываясь на ряде Тейлора, предварительно вычислив его:

Ряд Тейлора функции f(x) по степеням (x-*a*):

Разложим в ряд Тейлора функцию :

Так как ряд бесконечен, точно вычислить его невозможно, а возможно с точностью с *ε =* 10-6. Вычислим ряд Тейлора, как qn = *a*n+1/*a*n:

Получим одиннадцать значений функции, с которыми и будем сравнивать значения, которые получены с помощью квадратурных формул.

1. Вычислить интеграл с помощью пяти составных квадратурных формул:
2. Квадратурная формула левых прямоугольников:

(1)

*Ln* - составная формула левых прямоугольников.

1. Квадратурная формула центральных прямоугольников:

(2)

где *Ln* - составная формула центральных прямоугольников.

1. Квадратурная формула трапеции – квадратурная формула с 2-мя узлами *x1 = a, x2 = b:*

.

Составная квадратурная формула трапеции

. (3)

1. Квадратурная формула Симпсона:

) =*Ln*. (4)

1. Составная квадратурная формула Гаусса:

=*Ln*. (5)

Интеграл вычисляется от 0, как задано в самом условии до каждой из 11 точек, увеличивая с каждым разом количество разбиений между точками в два раза до тех пор, пока не выполнится условие |*Ln-Ln+1*| ≤ ε, то есть значение в двух точках почти равны друг другу.

# **ХОД РАБОТЫ**

* 1. **Квадратурная формула левых прямоугольников**

Для вычисления интеграла была применена квадратурная формула (1), в результате получили таблицу 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *xi* | *L0(x)* | *Ln(x)* | *| L0(x)- Ln(x)|* | *N* |
| 0 | 0.0000000 | 0.0000000 | 0.0000000000 | 2 |
| 0.2 | 0.2227029 | 0.2227026 | 0.0000003454 | 4096 |
| 0.4 | 0.4283924 | 0.4283923 | 0.0000000669 | 32768 |
| 0.6 | 0.6038559 | 0.6038564 | 0.0000005036 | 131072 |
| 0.8 | 0.7421015 | 0.7421008 | 0.0000006938 | 262144 |
| 1 | 0.8427009 | 0.8427005 | 0.0000004901 | 524288 |
| 1.2 | 0.9103134 | 0.9103138 | 0.0000004212 | 524288 |
| 1.4 | 0.9522847 | 0.9522847 | 0.0000000633 | 1048576 |
| 1.6 | 0.9763481 | 0.9763476 | 0.0000004750 | 1048576 |
| 1.8 | 0.9890901 | 0.9890896 | 0.0000005633 | 1048576 |
| 2 | 0.9953217 | 0.9953212 | 0.0000005115 | 1048576 |

Таблица 1. Левые прямоугольники

Из таблицы 1 видно, что квадратурная формула (1) требует большое количество разбиений.

* 1. **Квадратурная формула центральных прямоугольников**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *xi* | *L0(x)* | *Ln(x)* | *| L0(x)- Ln(x)|* | *N* |
| 0 | 0.0000000 | 0.0000000 | 0.0000000000 | 2 |
| 0.2 | 0.2227029 | 0.2227026 | 0.0000003385 | 512 |
| 0.4 | 0.4283924 | 0.4283924 | 0.0000000155 | 512 |
| 0.6 | 0.6038559 | 0.6038561 | 0.0000001642 | 1024 |
| 0.8 | 0.7421015 | 0.7421011 | 0.0000003935 | 512 |
| 1 | 0.8427009 | 0.8427009 | 0.0000000180 | 512 |
| 1.2 | 0.9103134 | 0.9103140 | 0.0000006446 | 1024 |
| 1.4 | 0.9522847 | 0.9522851 | 0.0000003804 | 2048 |
| 1.6 | 0.9763481 | 0.9763485 | 0.0000004330 | 512 |
| 1.8 | 0.9890901 | 0.9890908 | 0.0000006950 | 256 |
| 2 | 0.9953217 | 0.9953225 | 0.0000007552 | 256 |

Таблица 2. Центральные прямоугольники

Из таблицы 2 видно, что квадратурная формула (2) требует большое количество разбиений.

* 1. **Квадратурная формула трапеций**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *xi* | *L0(x)* | *Ln(x)* | *| L0(x)- Ln(x)|* | *N* |
| 0 | 0.0000000 | 0.0000000 | 0.0000000000 | 2 |
| 0.2 | 0.2227029 | 0.2227012 | 0.0000017529 | 32 |
| 0.4 | 0.4283924 | 0.4283922 | 0.0000001916 | 256 |
| 0.6 | 0.6038559 | 0.6038560 | 0.0000000426 | 512 |
| 0.8 | 0.7421015 | 0.7421008 | 0.0000006840 | 512 |
| 1 | 0.8427009 | 0.8427005 | 0.0000004139 | 512 |
| 1.2 | 0.9103134 | 0.9103137 | 0.0000003142 | 512 |
| 1.4 | 0.9522847 | 0.9522851 | 0.0000003024 | 1024 |
| 1.6 | 0.9763481 | 0.9763475 | 0.0000005893 | 256 |
| 1.8 | 0.9890901 | 0.9890998 | 0.0000002881 | 256 |
| 2 | 0.9953217 | 0.9953218 | 0.0000001245 | 256 |

Таблица 3. Трапеции

Из таблицы 3 видно, что квадратурная формула (3) требует большое количество разбиений.

* 1. **Квадратурная формула Симпсона**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *xi* | *L0(x)* | *Ln(x)* | *| L0(x)- Ln(x)|* | *N* |
| 0 | 0.0000000 | 0.0000000 | 0.0000000000 | 2 |
| 0.2 | 0.2227029 | 0.2227027 | 0.0000002530 | 2 |
| 0.4 | 0.4283924 | 0.4283925 | 0.0000001085 | 4 |
| 0.6 | 0.6038559 | 0.6038569 | 0.0000009117 | 4 |
| 0.8 | 0.7421015 | 0.7421010 | 0.0000004832 | 16 |
| 1 | 0.8427009 | 0.8427009 | 0.0000000094 | 8 |
| 1.2 | 0.9103134 | 0.9103128 | 0.0000005390 | 2 |
| 1.4 | 0.9522847 | 0.9522848 | 0.0000001003 | 8 |
| 1.6 | 0.9763481 | 0.9763483 | 0.0000002782 | 16 |
| 1.8 | 0.9890901 | 0.9890904 | 0.0000003056 | 32 |
| 2 | 0.9953217 | 0.9953222 | 0.0000004749 | 32 |

Таблица 4. Симпсон

Из таблицы 4 видно, что квадратурная формула (4) требует малое количество разбиений.

* 1. **Квадратурная формула Гаусса**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *xi* | *L0(x)* | *Ln(x)* | *| L0(x)- Ln(x)|* | *N* |
| 0 | 0.0000000 | 0.0000000 | 0.0000000000 | 2 |
| 0.2 | 0.2227029 | 0.2227025 | 0.0000004000 | 2 |
| 0.4 | 0.4283924 | 0.4283923 | 0.0000001308 | 4 |
| 0.6 | 0.6038559 | 0.6038556 | 0.0000003568 | 4 |
| 0.8 | 0.7421015 | 0.7421010 | 0.0000004950 | 16 |
| 1 | 0.8427009 | 0.8427007 | 0.0000002437 | 8 |
| 1.2 | 0.9103134 | 0.9103138 | 0.0000003981 | 4 |
| 1.4 | 0.9522847 | 0.9522853 | 0.0000005528 | 8 |
| 1.6 | 0.9763481 | 0.9763484 | 0.0000003469 | 16 |
| 1.8 | 0.9890901 | 0.9890905 | 0.0000004084 | 16 |
| 2 | 0.9953217 | 0.9953223 | 0.0000005917 | 16 |

Таблица 5. Гаусс

Из таблицы 5 видно, что квадратурная формула (5) требует малое количество разбиений.

# **ЗАКЛЮЧЕНИЕ**

В ходе работы вычислили интеграл с помощью пяти квадратурных

формул. Сравнили получившиеся значения, пришли к выводу, что самый эффективный метод – метод Гаусса.

# **ЛИСТИНГ**

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

void Taylor(int b, long n, double epsilon, double an, double sum, double sum1, int i, double\* xvalues, double\* yvalues, int col, int kol);

double Fx(double epsilon, double x);

double Integral(double t);

double CentralRectangle(double x, int n);

double LeftRectangle(double x, int n);

double Trapecia(double x, int n);

double epsilon = pow(10, -6);

int main() {

double an = 0, sum = 0, sum1 = 0;

int i = 0;

int col = 10;

long n = 0;

double a = 0, b = 2;

double yvalues[100];

double xvalues[100];

printf("Task1.\n");

for (int kol = 11; kol < 12; kol++) {

// printf("%d\n", kol);

Taylor(b, n, epsilon, an, sum, sum1, i, xvalues, yvalues, col, kol);

}

printf("\n\n");

printf("Task2.\n");

return 0;

}

double Fx(double epsilon, double x) {

double sum = 0;

int n = 0;

double an = x;

double q = 0;

while (fabs(an) > epsilon) {

sum += an;

q = -(x \* x \* (2 \* n + 1)) / ((n + 1) \* (2 \* n + 3));

an \*= q;

n++;

}

sum \*= 2 / sqrt(M\_PI);

return sum;

}

double Integral(double t) { return exp(-t \* t) \* 2 / sqrt(M\_PI); }

double LeftRectangle(double x, int n) {

double s1 = 0, s = 0;

double c = 0.0;

n = 1;

double h;

do {

s = s1;

n \*= 1048576;

h = (x - c) / n;

s1 = 0;

for (double temp = 0; temp < x; temp += h) {

s1 += h \* (Integral(temp + h));

}

// printf("%d\n", n);

} while (abs(s1 - s) > epsilon);

{ return s1; }

}

double CentralRectangle(double x, int n) {

double s1 = 0, s = 0;

double c = 0.0;

n = 1;

double h;

do {

s = s1;

n \*= 1024;

printf("%d\n", n);

h = (x - c) / n;

s1 = 0;

for (double temp = 0; temp < x; temp += h) {

s1 += h \* (Integral(temp + h / 2));

}

} while (abs(s1 - s) > epsilon);

{ return s1; }

}

double Trapecia(double x, int n) {

double s1 = 0, s = 0;

double c = 0.0;

n = 1;

double h;

do {

s = s1;

n \*= 512;

printf("%d\n", n);

h = (x - c) / n;

s1 = 0;

for (double temp = 0; temp < x; temp += h) {

s1 += h \* ((Integral(temp + h) + Integral(temp)) / 2);

}

} while (abs(s1 - s) > epsilon);

{ return s1; }

}

double Simpson(double x, int n) {

double s1 = 0, s = 0;

double c = 0.0;

n = 1;

double h;

do {

s = s1;

n \*= 32;

printf("%d\n", n);

h = (x - c) / n;

s1 = 0;

for (double temp = 0; temp < x; temp += h) {

s1 += h / 6 \*

(Integral(temp) + 4 \* Integral(temp + h / 2) +

Integral(temp + h));

}

} while (abs(s1 - s) > epsilon);

{ return s1; }

}

double Gauss(double x, int n) {

double s1 = 0, s = 0;

double c = 0.0;

n = 1;

double h;

do {

s = s1;

n \*= 16;

printf("%d\n", n);

h = (x - c) / n;

s1 = 0;

for (double temp = 0; temp < x; temp += h) {

s1 += h / 2 \*

(Integral(temp + (h / 2) \* (1 - 1 / sqrt(3))) +

Integral(temp + h / 2 \* (1 + 1 / sqrt(3))));

}

} while (abs(s1 - s) > epsilon);

{ return s1; }

}

void Taylor(int b, long n, double epsilon, double an, double sum, double sum1,

int i, double\* xvalues, double\* yvalues, int col, int kol) {

int a = 0;

double max = -100;

// int n=1;

double h = (double)(b - a) / (kol - 1);

double h1 = (double)(b - a) / 10;

int o = 1;

for (int top = 0; top < kol; top++) {

double xi = a + top \* h;

xvalues[top] = xi;

// printf("%lf\n", xvalues[top]);

}

// printf("%d ", kol);

// printf(" x f(x)\n");

for (i = 0; i < kol; i += 1) {

double x = a + i \* h;

yvalues[i] = Fx(epsilon, x);

// printf("%.10lf\n", yvalues[i]);

}

for (int i = 0; i \* h <= b; i++) {

printf("%.10lf\n", fabs(LeftRectangle(i \* h, n) - Fx(epsilon, i \* h)));

// printf("%.7lf\n", LeftRectangle(i\*h, n));

}

return;

}

**Вввввввв**

**1. ЦЕЛЬ РАБОТЫ**

Изучить и сравнить способы приближенного вычисления одной из специальной функции математической физики

на отрезке [a, b], где а = 0, b=2, h = 0.2, ε = 10-6 при помощи квадратурных формул.

**2. ИСХОДНЫЕ ДАННЫЕ И ЗАДАНИЕ**

Одна из специальных функций математической физики – функция ошибок, которая определяется следующим образом

Для данной функции рассмотрим следующие методы вычисления интеграла с помощью квадратурных формул:

1. Квадратурная формула левых прямоугольников.
2. Квадратурная формула центральных прямоугольников.
3. Квадратурная формула трапеции.
4. Квадратурная формула Симпсона.

5. Квадратурная формула Гаусса с 2-мя узлами

**3. ХОД РАБОТЫ**

**1. ЦЕЛЬ РАБОТЫ**

Изучить методы приближенного вычисления определенных интегралов с помощью квадратурных формул.

**2. ИСХОДНЫЕ ДАННЫЕ И ЗАДАНИЕ**

Одна из специальных функций математической физики – функция ошибок, которая определяется следующим образом

на отрезке [a, b], где a=0, b=2, h=0.2, =10-6 при помощи квадратурных формул.

Для этого рассмотрим следующие методы:

1. Квадратурная формула левых прямоугольников.

где Ln – это составная формула левых прямуогольников.