

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

Направление подготовки: 01.03.04 – Прикладная математика

**ОТЧЁТ**

По дисциплине «Численные методы»

на тему:

«Вычисление интеграла с помощью квадратурных формул»

Выполнил:  
студент группы 09-221  
Саитов М.А.  
Проверил:  
ассистент Глазырина О.В.

Казань, 2024 год

## Содержание

1	Постановка задачи	3
2	Ход работы	5
3	Выводы	9
4	Список литературы	10
5	Листинг программы	11

# 1 Постановка задачи

Необходимо изучить и сравнить различные способы приближённого вычисления функции ошибок

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \quad (1)$$

1. Протабулировать  $\operatorname{erf}(x)$  на отрезке  $[a, b]$  с шагом  $h$  и точностью  $\varepsilon$ , основываясь на ряде Маклорена, предварительно вычислив его. Получив таким образом таблицу из 11 точек вида:

$$\begin{array}{cccc} x_0 & x_1 & x_2 & \dots \\ f_0 & f_1 & f_2 & \dots \end{array}$$

$$f_i = \operatorname{erf}(x_i), \quad x_i = a + ih, \quad i = 0, \dots, n.$$

2. Вычислить  $\operatorname{erf}(x)$  при помощи пяти составных квадратурных формул при  $h = (x_{i+1} - x_i)$ ,

$$g(x) = \frac{2}{\sqrt{\pi}} e^{-x^2} \quad (2)$$

2.1. Формула правых прямоугольников:

$$J_N(x) = \sum_{i=1}^n h g(x_i) \quad (3)$$

2.2. Формула центральных прямоугольников:

$$J_N(x) = \sum_{i=1}^n h g\left(\frac{x_i + x_{i+1}}{2}\right) \quad (4)$$

2.3. Формула трапеции:

$$J_N(x) = \sum_{i=1}^n h \frac{g(x_i) + g(x_{i+1})}{2} \quad (5)$$

2.4. Формула Симпсона:

$$J_N(x) = \sum_{i=1}^n \frac{h}{6} \left[ g(x_i) + 4g\left(\frac{x_i + x_{i+1}}{2}\right) + g(x_{i+1}) \right] \quad (6)$$

2.5. Формула Гаусса:

$$J_N(x) = \sum_{i=1}^n \frac{h}{2} \left[ g\left(x_i + \frac{h}{2} \left(1 - \frac{1}{\sqrt{3}}\right)\right) + g\left(x_i + \frac{h}{2} \left(1 + \frac{1}{\sqrt{3}}\right)\right) \right] \quad (7)$$

Вычисления проводятся от начала интегрирования до каждой из 11 точек, увеличивая количество разбиений между точками в 2 раза до тех пор, пока погрешность больше  $\varepsilon$ .

## 2 Ход работы

Для того чтобы найти значение функции в точке, необходимо протабулировать искомый интеграл на отрезке  $[a, b]$  с шагом  $h = 0.2$  и точностью  $\varepsilon$ . Для этого:

1. Найдём разбиение подинтегральной функции  $e^{-t^2}$  в ряд Маклорена, подставив в стандартное разбиение функции  $e^x$  в ряд Маклорена  $x = -t^2$ :

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \Rightarrow e^{-t^2} = \sum_{n=0}^{\infty} \frac{(-1)^n t^{2n}}{n!} \quad (8)$$

2. Проинтегрируем полученное выражение на интеграле  $[0, x]$ :

$$\frac{2}{\sqrt{\pi}} \int_0^x \sum_{n=0}^{\infty} \frac{(-1)^n t^{2n}}{n!} dt = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n t^{2n+1}}{(2n+1)n!} \Big|_0^x = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)n!} \quad (9)$$

3. Выделим два общих члена  $a_n$ ,  $a_{n+1}$  из полученного выражения и найдём  $q_n = \frac{a_{n+1}}{a_n}$ :

$$a_n = \frac{(-1)^n x^{2n+1}}{n!(2n+1)}, \quad a_{n+1} = \frac{(-1)^{n+1} x^{2n+3}}{(n+1)!(2n+3)}. \quad (10)$$

$$q_n = \frac{-x^2(2n+1)}{(n+1)(2n+3)}. \quad (11)$$

Таким образом,

$$a_n = a_0 \prod_{n=0}^{n-1} q_n. \quad (12)$$

Для каждой точки  $x_i = a + ih$  найдём значение  $erf(x_i)$  и составим таблицу результатов

$x_i$	$erf(x_i)$
0,2	0,2227025926
0,4	0,4283923805
0,6	0,6038561463
0,8	0,7421009541
1,0	0,8427006602
1,2	0,9103140831
1,4	0,9522852302
1,6	0,9763484001
1,8	0,9890906215
2,0	0,9953226447

Таблица 1 - точки  $x_i$  и значения разложения в ряд Маклорена функции  $erf(x_i)$

После нахождения значений разложения в ряд Маклорена в точках, вычислим значение  $\text{erf}(x)$  при помощи 5 составных квадратурных формул. Для каждой формулы составим свою таблицу. В таблицах будут находиться значения точки, для которой производились расчёты, значение разбиения в ряд Маклорена в точке, значение найденного с помощью формулы интеграла в точке, модуль разницы между значениями найденного интеграла и разбиения, количества разбиений, которые пришлось совершить для нахождения значения интеграла с нужной точностью.

1. Левые прямоугольники:

$x_i$	$J_0(x_i)$	$J(x_i)$	$ J_0(x_i) - J_N(x_i) $	$N$
0.2	0.222703	0.222707	4.32204e-06	1024
0.4	0.428392	0.428425	3.25766e-05	1024
0.6	0.603856	0.603956	9.9908e-05	1024
0.8	0.742101	0.742309	0.000208343	1024
1.0	0.842701	0.843049	0.000348225	1024
1.2	0.910314	0.910818	0.000504366	1024
1.4	0.952285	0.952948	0.00066258	1024
1.6	0.976348	0.977162	0.00081329	1024
1.8	0.989091	0.990043	0.000952791	1024
2.0	0.995322	0.996404	0.00108161	1024

Таблица 2 - таблица значений для формулы Левых прямоугольников

2. Правые прямоугольники:

$x_i$	$J_0(x_i)$	$J(x_i)$	$ J_0(x_i) - J_N(x_i) $	$N$
0.2	0.222703	0.222698	4.31945e-06	1024
0.4	0.428392	0.42836	3.25944e-05	1024
0.6	0.603856	0.603756	9.99763e-05	1024
0.8	0.742101	0.741893	0.000208371	1024
1.0	0.842701	0.842352	0.00034833	1024
1.2	0.910314	0.909809	0.000504659	1024
1.4	0.952285	0.951622	0.000662823	1024
1.6	0.976348	0.975535	0.000813508	1024
1.8	0.989091	0.988138	0.000953008	1024
2.0	0.995322	0.99424	0.00108189	1024

Таблица 3 - таблица значений для формулы Правых прямоугольников

3. Центральные прямоугольники:

$x_i$	$J_0(x_i)$	$J(x_i)$	$ J_0(x_i) - J_N(x_i) $	$N$
0.2	0.222703	0.222703	1.80774e-07	64
0.4	0.428392	0.428393	3.14933e-07	128
0.6	0.603856	0.603856	2.09718e-07	256
0.8	0.742101	0.742101	1.31198e-07	512
1.0	0.842701	0.842701	1.45324e-07	512
1.2	0.910314	0.910314	7.42506e-08	512
1.4	0.952285	0.952285	8.73705e-08	512
1.6	0.976348	0.976348	6.19822e-08	512
1.8	0.989091	0.989091	2.63449e-07	256
2.0	0.995322	0.995322	9.90524e-08	256

Таблица 4 - таблица значений для формулы Центральных прямоугольников

4. Формула трапеций:

$x_i$	$J_0(x_i)$	$J(x_i)$	$ J_0(x_i) - J_N(x_i) $	$N$
0.2	0.222703	0.222703	8.39077e-08	128
0.4	0.428392	0.428392	1.55199e-07	256
0.6	0.603856	0.603856	1.16408e-07	512
0.8	0.742101	0.742101	1.59468e-07	512
1.0	0.842701	0.842701	2.50953e-07	512
1.2	0.910314	0.910314	3.70479e-07	512
1.4	0.952285	0.952285	3.2974e-07	512
1.6	0.976348	0.976348	2.80325e-07	512
1.8	0.989091	0.98909	2.31385e-07	512
2.0	0.995322	0.995322	2.17437e-07	512

Таблица 5 - таблица значений для формулы Трапеций

## 5. Формула Симпсона

### 5.1. Вывод формулы Симпсона через интегральный полином Лагранжа:

Формула для полинома Лагранжа:

$$L_n(x) = \sum_{i=0}^n f(x_i) \prod_{i \neq j, j=0}^n \frac{x - x_j}{x_i - x_j} \quad (13)$$

По трём узлам ( $x_1 = a, x_2 = \frac{a+b}{2}, x_3 = b$ ):

$$L_2 = f(a) \left( \frac{x - \frac{a+b}{2}}{a - \frac{a+b}{2}} \right) \left( \frac{x - b}{a - b} \right) + f\left(\frac{a+b}{2}\right) \left( \frac{x - a}{\frac{a+b}{2} - a} \right) \left( \frac{x - b}{\frac{a+b}{2} - b} \right) + f(b) \left( \frac{x - \frac{a+b}{2}}{b - \frac{a+b}{2}} \right) \left( \frac{x - a}{b - a} \right).$$

Проинтегрируем выражение по интервалу [a,b]:

$$\int_a^b L_2(x) dx = f(a)c_1 + f\left(\frac{a+b}{2}\right)c_2 + f(b)c_3 \quad (14)$$

где  $c_1 = \frac{b-a}{6}, c_2 = \frac{2}{3}(b-a), c_3 = \frac{b-a}{6}$ .

Тогда:

$$\int_a^b L_2(x) dx = \frac{b-a}{6} \left( f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right) \quad (15)$$

### 5.2. Значения полученные для формулы Симпсона:

$x_i$	$J_0(x_i)$	$J(x_i)$	$ J_0(x_i) - J_N(x_i) $	$N$
0.2	0.222703	0.222703	1.20186e-08	4
0.4	0.428392	0.428392	1.23149e-08	8
0.6	0.603856	0.603856	4.87389e-08	8
0.8	0.742101	0.742101	4.40568e-08	16
1.0	0.842701	0.842701	3.49238e-08	16
1.2	0.910314	0.910314	6.97319e-08	8
1.4	0.952285	0.952285	7.4049e-08	16
1.6	0.976348	0.976348	9.72274e-08	16
1.8	0.989091	0.98909	1.38559e-07	16
2.0	0.995322	0.995322	1.00857e-07	32

Таблица 6 - таблица значений для формулы Симпсона



## 6. Формула Гаусса:

$x_i$	$J_0(x_i)$	$J(x_i)$	$ J_0(x_i) - J_N(x_i) $	$N$
0.2	0.222703	0.222703	9.76378e-11	4
0.4	0.428392	0.428392	9.54019e-09	8
0.6	0.603856	0.603856	3.76879e-08	8
0.8	0.742101	0.742101	2.71688e-08	16
1	0.842701	0.842701	5.74231e-09	16
1.2	0.910314	0.910314	9.95342e-08	8
1.4	0.952285	0.952285	4.75374e-08	16
1.6	0.976348	0.976348	9.31051e-09	16
1.8	0.989091	0.989091	2.59625e-08	16
2	0.995322	0.995322	4.66851e-08	16

Таблица 7 - таблица значений для формулы Гаусса

## 3 Выводы

Проделав все вычисления, можно сделать выводы, что более комплексные методы вычисления интеграла, как формула Гаусса и Симпсона, показывают наилучшие результаты за меньшее количество разбиений. В это же время худшие результаты вычисления показывают методы правых прямоугольников и метод трапеций, приводя к довольно большому значению ошибки.

## 4 Список литературы

1. Глазырина Л.Л., Карчевский М.М. Численные методы: учебное пособие. — Казань: Казан. ун-т, 2012. — 122
2. Глазырина Л.Л.. Практикум по курсу «Численные методы». Решение систем линейных уравнений: учеб. пособие. — Казань: Изд-во Казан. ун-та, 2017. — 52 с.

## 5 Листинг программы

```
1 #include <math.h>
2 #include <iostream>
3 #include <iomanip>
4
5 #define Eps pow(10, -6)
6 #define a 0
7 #define b 2
8 #define H 0.2
9
10 enum FuncType{
11     leftRec, rightRec, centralRect,
12     trapezoid, simpson, gauss
13 };
14
15 class Function{
16 private:
17     FuncType type;
18 public:
19     Function() = default;
20
21     void set_FuncType(FuncType type);
22
23     double Qn(double n, double x);
24     double erf(double x);
25
26     double calculatedFunction(int n, double x);
27     void calculateAndWrite(double x, double y);
28     void printTable();
29
30     double Left_Rect(int n, double x);
31     double Right_Rect(int n, double x);
32     double Central_Rect(int n, double x);
33     double Trapezoid(int n, double x);
34     double Simpson(int n, double x);
35     double Gaus(int n, double x);
36 };
```

```
1 #include "Function.h"
2
3 void Function::set_FuncType(FuncType typeToCopy){
4     type = typeToCopy;
```

```

5 }
6
7 double Function::Qn(double n, double x){
8     return -(((x * x) / (n + 1)) * ((2 * n + 1) / (2 * n + 3)));
9 }
10 double Function::erf(double x){
11     int n = 1;
12     double prevA = x;
13     double currentA = x;
14     double result = x;
15     while (abs(prevA) >= Eps){
16         currentA = Qn(n - 1, x) * prevA;
17         result += currentA;
18         prevA = currentA;
19         n++;
20     }
21     result *= 2 / sqrt(M_PI);
22     return result;
23 }
24
25 float func(float t) { return (2 / sqrt(M_PI)) * pow(std::exp(1.0), -(
    t * t)); }
26
27 double Function::Left_Rect(int n, double x){
28     double h = x/n;
29     double sum = 0.0;
30     for(int i = 0; i < n; i++){
31         double xi = a + i*h;
32         sum += h * func(xi);
33     }
34     return sum;
35 }
36
37 double Function::Right_Rect(int n, double x){
38     double h = x/n;
39     double sum = 0.0;
40     for(int i = 1; i <= n; i++){
41         double xi = a + i*h;
42         sum += h * func(xi);
43     }
44     return sum;
45 }
46

```

```

47 double Function::Central_Rect(int n, double x){
48     double h = x/n;
49     double sum = 0.0;
50     double xi = h/2;
51     for(int i = 0; i <= n-1; i++){
52         sum += h * func(xi);
53         xi+=h;
54     }
55     return sum;
56 }
57
58 double Function::Trapezoid(int n, double x){
59     double h = x/n;
60     double sum = 0.0;
61     double xi = 0.0;
62     for(int i = 0; i <= n-1; i++){
63         sum += h*(func(xi) + func(xi+h))/2;
64         xi += h;
65     }
66     return sum;
67 }
68
69 double Function::Simpson(int n, double x){
70     double h = x/n;
71     double sum = 0.0;
72     double xi = 0.0;
73     for(int i = 0; i <= n-1; i++){
74         sum += (func(xi) + 4 * func(xi + h / 2) + func(xi + h)) * h /
75             6;
76         xi += h;
77     }
78     return sum;
79 }
80
81 double Function::Gaus(int n, double x){
82     double h = x/n;
83     double num1 = (1 - 1.0 / sqrt(3)) * h / 2;
84     double num2 = (1 + 1.0 / sqrt(3)) * h / 2;
85     double sum = 0;
86     double xi = 0.0;
87     for(int i = 0; i <= n-1; i++){
88         sum += (func(xi + num1) + func(xi + num2)) * h/2;
89         xi += h;

```

```

89     }
90     return sum;
91 }
92
93 void Function::calculateAndWrite(double x, double y){
94     double lastJ = 0;
95     double J = 0;
96     int n = 1;
97     do{
98         n *= 2;
99         lastJ = J;
100        J = calculatedFunction(n, x);
101    }
102    while (abs(lastJ - J) > Eps && n < 1024);
103    double accuracy = abs(J - y);
104    std::cout << std::setw(3) << x << " & " << std::setw(9) << y << "
105        & " << std::setw(9) << J << " & " << std::setw(11) <<
106        accuracy << " & " << n << std::endl;
107 }
108 double Function::calculatedFunction(int n, double x){
109     double result = 0.0;
110     switch(type){
111         case leftRec:{
112             result = Left_Rect(n, x);
113             break;
114         }
115         case rightRec:{
116             result = Right_Rect(n, x);
117             break;
118         }
119         case centralRect:{
120             result = Central_Rect(n, x);
121             break;
122         }
123         case trapezoid:{
124             result = Trapezoid(n, x);
125             break;
126         }
127         case simpson:{
128             result = Simpson(n, x);
129             break;
130         }
131         case gauss:{

```

```

130         result = Gaus(n, x);
131         break;
132     }
133 }
134 return result;
135 }
136 void Function::printTable(){
137     double* x = new double[11];
138     double* y = new double[11];
139     x[0] = 0;
140     y[0] = erf(x[0]);
141     for (int i = 1; i < 11; i++){
142         x[i] = x[i - 1] + H;
143         y[i] = erf(x[i]);
144     }
145     std::cout << "\033[1m" << "\033[3m" << "Left Rectangle\n" << "
        \033[0m";
146     set_FuncType(leftRec);
147     for (int i = 1; i < 11; i++){
148         calculateAndWrite(x[i], y[i]);
149     }
150     std::cout << "\033[1m" << "\033[3m" << "Right Rectangle\n" << "
        \033[0m";
151     set_FuncType(rightRec);
152     for (int i = 1; i < 11; i++){
153         calculateAndWrite(x[i], y[i]);
154     }
155     std::cout << "\033[1m" << "\033[3m" << "Central Rectangle\n" << "
        \033[0m";
156     set_FuncType(centralRect);
157     for (int i = 1; i < 11; i++){
158         calculateAndWrite(x[i], y[i]);
159     }
160     std::cout << "\033[1m" << "\033[3m" << "Trapezoid\n" << "\033[0m";
161     set_FuncType(trapezoid);
162     for (int i = 1; i < 11; i++){
163         calculateAndWrite(x[i], y[i]);
164     }
165     std::cout << "\033[1m" << "\033[3m" << "Simpson\n" << "\033[0m";
166     set_FuncType(simpson);
167     for (int i = 1; i < 11; i++){
168         calculateAndWrite(x[i], y[i]);
169     }

```

```
170     std::cout << "\033[1m" << "\033[3m" << "Gauss\n" << "\033[0m";
171     set_FuncType(gauss);
172     for (int i = 1; i < 11; i++){
173         calculateAndWrite(x[i], y[i]);
174     }
175 }
```

```
1 #include "Function.h"
2
3 int main(){
4     Function f;
5     f.printTable();
6     return 0;
7 }
```