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

Направление подготовки: 01.03.04 – Прикладная математика

ОТЧЁТ
По дисциплине «Численные методы»
на тему:
«Вычисление интеграла с помощью квадратурных формул»

Выполнил:
Романов И.И. 09-222 группа.

Руководитель:
Глазырина О.В.

Казань 2024

Содержание

1 Постановка задачи

Необходимо изучить и сравнить различные способы приближённого вычисления функции ошибок

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int\limits_0^x e^{-t^2} dt \tag{1}$$

1. Протабулировать erf(x) на отрезке [a,b] с шагом h и точностью ε, основываясь на ряде Тейлора,
   предварительно вычислив его

   $$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \sum_{n=1}^{\infty} (-1)^n \frac{x^{2n+1}}{n!(2n+1)} \tag{2}$$

   где a = 0, b = 2, h = 0.2, $\varepsilon = 10^{-6}$. Получив таким образом таблицу из 11 точек

   $x_0\ x_1\ x_2\ \dots$
   $f_0\ f_1\ f_2\ \dots$

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

2. Вычислить erf(x) при помощи пяти составных квадратурных формул:

   2.1. Формула Левых прямоугольников:

   $$J_N(x) = \sum_{i=1}^{n} h \cdot g(x_i), \quad h = (x_{i+1} - x_i) \tag{3}$$

   2.2. Формула Центральных прямоугольников:

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

   2.3. Формула трапеции:

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

   2.4. Формула Симпсона:

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

   2.5. Формула Гаусса:

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

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

2 Ход работы

Выберем точки на отрезке [a,b] с шагом h.

$$x_i = a + i \cdot h.$$

Для каждой точки x_i найдём значение $f(x_i)$ и составим таблицу результатов (Таблица 1).

x_i	$f(x_i)$
0,0	0,0000000000
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 Тейлора $f(x_i)$

После нахождения значений разложения в ряд Тейлора в точках вычислим значение erf(x) при помощи 5 составных квадратурных формул. Для каждой формулы составим свою таблицу. В таблицах будут находится значения точки, для которой производились расчёты, значение разбиения в ряд Тейлора, 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. Правые прямоугольники:

x_i	$J_0(x_i)$	$J_(x_i)$	$\|J_0(x_i) - J_N(x_i)\|$	N
0,0	0,0000000000	0,0000000000	0,0000000000	2
0,2	0,2227025926	0,2226983160	0,0000042766	1024
0,4	0,4283923805	0,4283596873	0,0000326931	1024
0,6	0,6038561463	0,6037563682	0,0000997782	1024
0,8	0,7421009541	0,7418920994	0,0002088547	1024
1,0	0,8427006602	0,8423525691	0,0003480911	1024
1,2	0,9103140831	0,9098084569	0,0005056262	1024
1,4	0,9522852302	0,9516219497	0,0006632805	1024
1,6	0,9763484001	0,9764841199	0,0001357198	1024
1,8	0,9890906215	0,9891686440	0,0000780225	1024
2,0	0,9953226447	0,9953628182	0,0000401735	1024

Таблица 2 - таблица значений для формулы Правых прямоугольников

2. Центральные прямоугольники:

x_i	$J_0(x_i)$	$J_(x_i)$	$\lvert J_0(x_i) - J_N(x_i)\rvert$	N
0,0	0,0000000000	0,0000000000	0,0000000000	2
0,2	0,2227025926	0,2227027565	0,0000001639	64
0,4	0,4283923805	0,4283923209	0,0000000596	256
0,6	0,6038561463	0,6038563848	0,0000002384	256
0,8	0,7421009541	0,7421010733	0,0000001192	512
1,0	0,8427006602	0,8427013755	0,0000007153	512
1,2	0,9103140831	0,9103139043	0,0000001788	512
1,4	0,9522852302	0,9522854686	0,0000002384	512
1,6	0,9763484001	0,9763489366	0,0000005364	256
1,8	0,9890906215	0,9890908003	0,0000001788	512
2,0	0,9953226447	0,9953227639	0,0000001192	256

Таблица 3 - таблица значений для формулы Центральных прямоугольников

3. Формула Трапеций:

x_i	$J_0(x_i)$	$J_(x_i)$	$\lvert J_0(x_i) - J_N(x_i)\rvert$	N
0,0	0,0000000000	0,0000000000	0,0000000000	2
0,2	0,2227025926	0,2229140997	0,0002115071	128
0,4	0,4283923805	0,4287676811	0,0003753006	512
0,6	0,6038561463	0,6043169498	0,0004608035	512
0,8	0,7421009541	0,7425656319	0,0004646778	512
1,0	0,8427006602	0,8431062102	0,0004055500	512
1,2	0,9103140831	0,9106265903	0,0003125072	512
1,4	0,9522852302	0,9525024891	0,0002172589	512
1,6	0,9763484001	0,9764841199	0,0001357198	512
1,8	0,9890906215	0,9891686440	0,0000780225	512
2,0	0,9953226447	0,9953628182	0,0000401735	512

Таблица 4 - таблица значений для формулы Трапеций

4. Формула Симпсона

4.1. Вывод формулы Симпсона через интегральный полином Лагранжа:
Формула для полинома Лагранжа:

$$L_n(x) = \sum_{i=0}^{n} f(x_i) \prod_{i\neq j, j=0}^{n} \frac{x - x_j}{x_i - x_j} \tag{8}$$

По трём узлам $(x_1 = a, x_2 = \dfrac{a+b}{2}, x_3 = b) : L_2 = f(a)\left(\dfrac{x - \dfrac{a+b}{2}}{a - \dfrac{a+b}{2}}\right)\left(\dfrac{x-b}{a-b}\right) +$
$f\left(\dfrac{a+b}{2}\right)\left(\dfrac{x-a}{\dfrac{a+b}{2} - a}\right)\left(\dfrac{x-b}{\dfrac{a+b}{2} - b}\right) + f(b)\left(\dfrac{x - \dfrac{a+b}{2}}{b - \dfrac{a+b}{2}}\right)\left(\dfrac{x-a}{b-a}\right).$

Проинтегрируем выражение по интервалу [a,b]:

$$\int_a^b L_2(x)\mathrm{d}x = f(a)c_1 + f\left(\frac{a+b}{2}\right)c_2 + f(b)c_3 \tag{9}$$

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

Тогда:

$$\int\limits_a^b L_2(x)\mathrm{d}x = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right) \tag{10}$$

4.2. Значения полученные для формулы Симпсона:

x_i	$J_0(x_i)$	$J_(x_i)$	$\|J_0(x_i) - J_N(x_i)\|$	N
0,0	0,0000000000	0,0000000000	0,0000000000	2
0,2	0,2227025926	0,2227026075	0,0000000149	2
0,4	0,4283923805	0,4283923805	0,0000000000	4
0,6	0,6038561463	0,6038562059	0,0000000596	8
0,8	0,7421009541	0,7421009541	0,0000000000	8
1,0	0,8427006602	0,8427007794	0,0000001192	16
1,2	0,9103140831	0,9103139639	0,0000001192	16
1,4	0,9522852302	0,9522852302	0,0000000000	8
1,6	0,9763484001	0,9763483405	0,0000000596	16
1,8	0,9890906215	0,9890906215	0,0000000000	32
2,0	0,9953226447	0,9953221679	0,0000004768	32

Таблица 5 - таблица значений для формулы Симпсона

5. Формула Гаусса:

x_i	$J_0(x_i)$	$J_(x_i)$	$\|J_0(x_i) - J_N(x_i)\|$	N
0,0	0,0000000000	0,0000000000	0,0000000000	2
0,2	0,2227025926	0,2227025777	0,0000000149	2
0,4	0,4283923805	0,4283923209	0,0000000596	4
0,6	0,6038561463	0,6038560867	0,0000000596	8
0,8	0,7421009541	0,7421008945	0,0000000596	8
1,0	0,8427006602	0,8427007794	0,0000001192	16
1,2	0,9103140831	0,9103140235	0,0000000596	16
1,4	0,9522852302	0,9522851706	0,0000000596	8
1,6	0,9763484001	0,9763484001	0,0000000000	16
1,8	0,9890906215	0,9890905023	0,0000001192	32
2,0	0,9953226447	0,9953223467	0,0000002980	32

Таблица 6 - 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 Листинг программы

```
#include <algorithm>
#include <cmath>
#include <iostream>
```

```cpp
#include <vector>

using namespace std;

namespace constans {
const int STEPS = 1024;
const float LEFT_BORDER = 0;
const float EPSILON = 1e-6;
const float E = 2.71828182846;
}  // namespace constans

float Tfunc(float x) {
  int n = 0;
  float node_0 = x;
  float ans = x;
  while (fabs(node_0) > 1e-6) {
    float q = (-1)*(((2*n + 1)*x*x)/(2*n*n + 5*n + 3));
    node_0 *= q;
    ans += node_0;
    n++;
  }
  return (2/sqrt(M_PI))*ans;
}

float func(float t) { return (2 / sqrt(M_PI)) * pow(constans::E, -(t * t)); }

float leftRectangles(float (*func)(float), const float &a, float b,
                     int steps) {
  float result = 0.0;
  float h = (b - a) / steps;
  float x_i = 0.0;
  for (int i = 0; i < steps; i++) {
    x_i = a + h * i;
    result += h * func(x_i);
  }
  return result;
}

float rightRectangles(float (*func)(float), const float &a, float b,
                      int steps) {
  float result = 0.0;
  float h = (b - a) / steps;
  float x_i_1 = 0.0;
  for (int i = 1; i <= steps; i++) {
    x_i_1 = a + h * i;
    result += h * func(x_i_1);
  }
  return result;
}

float middleRectangles(float (*func)(float), const float &a, float b,
                       int steps) {
  float result = 0.0;
  float h = (b - a) / steps;
  float x_i = 0.0;
  float x_i_1 = 0.0;
  for (int i = 1; i <= steps; i++) {
    x_i = a + h * (i - 1);
    x_i_1 = a + h * i;
    result += h * func((x_i + x_i_1) / 2);
  }
  return result;
}

float trapezeFormula(float (*func)(float), const float &a, float b,
                     int steps) {
```

```
  float result = func(a) + func(b);
  float h = (b - a) / steps;
  float x_i_1 = 0.0;
  for (int i = 1; i <= steps; i++) {
    x_i_1 = a + h * i;
    result += 2 * func(x_i_1);
  }
  result *= h / 2;
  return result;
}

float SypmsonsFormula(float (*func)(float), const float &a, float b,
                      int steps) {
  float h = (b - a) / steps;
    float result = 0;
    float x = 0;
    for (int i = 0; i < steps; i++)
    {
        result += (func(x) + 4 * func(x + h / 2) + func(x + h)) * h / 6;
        x += h;
    }
    return result;
}

float GaussFormula(float (*func)(float), const float &a, float b, int steps) {
    float h = (b - a) / steps;
    float ad1 = (1 - 1.0 / sqrt(3)) * h / 2;
    float ad2 = (1 + 1.0 / sqrt(3)) * h / 2;
    float result = 0;
    float x = 0;
    for (int i = 0; i < steps; i++)
    {
        result += (func(x + ad1) + func(x + ad2)) * h / 2;
        x += h;
    }
    return result;
}

void CalculateFunc(vector<float> points,
                   float (*function)(float (*func)(float), const float &,
                                     float, int)) {
  for (auto point : points) {
    int i = 1;
    float last_j = 0.0;
    float j = 0.0;
    do {
      i *= 2;
      last_j = j;
      j = function(func, constans::LEFT_BORDER, point, i);
    } while (abs(last_j - j) > constans::EPSILON && i < constans::STEPS);

    float difference = abs(Tfunc(point) - j);

    printf(
        "x_i = %.1lf | J_o = %.10lf | J_n = %.10lf | |J_o - J_n| = %.10lf | N "
        "= %d\n",
        point, Tfunc(point), j, difference, i);
  }
}


int main() {
  vector<float> points = {0.0, 0.2, 0.4, 0.6, 0.8, 1.0,
                          1.2, 1.4, 1.6, 1.8, 2.0};
  cout << "Правые прямоугольники\n";
  CalculateFunc(points, rightRectangles);
```

```
  cout << "Левые прямоугольники\n";
  CalculateFunc(points, leftRectangles);
  cout << "Центральные прямоугольники\n";
  CalculateFunc(points, middleRectangles);
  cout << "Трапеции\n";
  CalculateFunc(points, trapezeFormula);
  cout << "Симпсон\n";
  CalculateFunc(points, SypmsonsFormula);
  cout << "Гаус\n";
  CalculateFunc(points, GaussFormula);
  return 0;
}
```