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

Исследование приближения функций при помощи квадратурных формул

Выполнил:
студент группы 09-222 Шпак В.С.
Проверил:
ассистент Глазырина О.В.

Казань, 2024 год

Оглавление

ПОСТАНОВКА ЗАДАЧИ

Одна из специализированных функций математической физики – интегральный синус, определяется следующим образом $\mathrm{Si(x)} = \int\limits_0^x \frac{sint}{t}\,dx$

Для вычисления погрешности нам понадобиться ее вычислить разложение в ряд Тейлора

$\mathrm{Si(x)} = \frac{(-1)^n x^{2n+1}}{(2n+1)(2n+1)!}$

Цель задания – вычислить интеграл с помощью квадратурных формул.

1. Левых прямоугольников
2. Правых прямоугольников
3. Трапеции
4. Симпсона
5. Гаусса
6. Центральных прямоугольников

где $h = (x_{i+1} - \ x_i),\ f(x) = \frac{sin(x)}{x}$;

ХОД РАБОТЫ

Найдем разложение интегрального синуса в ряд Тейлора. Для этого воспользуемся разложением функции синуса в ряд Тейлора

$$\mathrm{Si(x)} = \int\limits_0^x \frac{\sin t}{t}\,dt = \int\limits_0^x \frac{\sum_{k=0}^{\infty} \frac{(-1)^k t^{2k}}{(2k+1)!}}{t}\,dt = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{(2k+1)(2k+1)!} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)(2n+1)!}$$

Для каждой точки из ряда Тейлора будем вычислять значение до тех пор пока не выполнится условие $\mid \mathrm{J}_N(x) - \ J_{2N}(x) \ \mid < \ \varepsilon$. Далее находим погрешность как модуль разности полученного данным методом значения и найденного через разложение в ряд Тейлора. Составляем таблицу, где $J_0(x)$ - найденное значение интеграла, $J_N(x)$ - настоящее значение интеграла, N - количество разбиений отрезка.

Составная квадратурная формула левых прямоугольников: $J_N(x) = \sum_{i=0}^{n-1} h\ f(x_i)$

x_i	$J_0(x)$	$J_N(x)$	$\mid J_0(x) - \ J_N(x) \mid$	N
0.4	0.396461	0.396467	5.16524e-06	1024
0.8	0.772096	0.772136	4.03406e-05	1024
1.2	1.10805	1.10818	0.000130801	1024
1.6	1.38918	1.38947	0.000293099	1024
2	1.60541	1.60595	0.00053243	1024
2.4	1.75249	1.75333	0.000841865	1024
2.8	1.8321	1.8333	0.00120339	1024
3.2	1.8514	1.85299	0.00159075	1024
3.6	1.82195	1.82392	0.00197365	1024
4	1.7582	1.76053	0.00232251	1024

Составная квадратурная формула правых прямоугольников: $J_N(x) = \sum_{i=1}^{n} h\ f(x_i)$

x_i	$J_0(x)$	$J_N(x)$	$\mid J_0(x) - J_N(x) \mid$	N
0.4	0.396461	0.396456	5.16841e-06	1024
0.8	0.772096	0.772055	4.03664e-05	1024
1.2	1.10805	1.10792	0.00013088	1024
1.6	1.38918	1.38889	0.000293255	1024
2	1.60541	1.60488	0.00053271	1024
2.4	1.75249	1.75164	0.000842253	1024
2.8	1.8321	1.83089	0.00120385	1024
3.2	1.8514	1.84981	0.00159125	1024
3.6	1.82195	1.81997	0.00197412	1024
4	1.7582	1.75588	0.0023228	1024

Составная квадратурная формула трапеции: $J_N(x) = \sum_{i=0}^{n-1} h \quad \frac{f(x_i) + f(x_{i+1})}{2}$

x_i	$J_0(x)$	$J_N(x)$	$\mid J_0(x) - J_N(x) \mid$	N
0.4	0.396461	0.396461	1.06701e-07	128
0.8	0.772096	0.772096	2.03634e-07	256
1.2	1.10805	1.10805	1.57927e-07	512
1.6	1.38918	1.38918	3.27103e-07	512
2	1.60541	1.60541	1.40058e-07	1024
2.4	1.75249	1.75249	1.93854e-07	1024
2.8	1.8321	1.8321	2.29813e-07	1024
3.2	1.8514	1.8514	2.50917e-07	1024
3.6	1.82195	1.82195	2.37024e-07	1024
4	1.7582	1.7582	1.43659e-07	1024

Составная квадратурная формула Симпсона: $J_N(x) = \sum_{i=0}^{n-1} \frac{h}{6} \quad \left[f(x_i) + 4f\left(\frac{x_i + x_{i+1}}{2}\right) + f(x_{i+1}) \right.$

Формула для полинома Лагранжа:

$$L_n(x) = \sum_{i=0}^{n} f(x_i) \prod_{i \neq j, j=0}^{n} \frac{x - x_j}{x_i - x_j} \tag{1}$$

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

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

Проинтегрируем выражение по интервалу [a,b]:

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

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

Тогда:

$$\int_a^b L_2(x)\mathrm{d}x = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right) \tag{3}$$

x_i	$J_0(x)$	$J_N(x)$	$\mid J_0(x) - J_N(x) \mid$	N
0.4	0.396461	0.396461	2.80595e-09	4
0.8	0.772096	0.772096	4.94881e-09	8
1.2	1.10805	1.10805	3.54972e-08	8
1.6	1.38918	1.38918	1.35524e-08	16
2	1.60541	1.60541	1.84437e-08	16
2.4	1.75249	1.75249	3.86081e-08	16
2.8	1.8321	1.8321	6.33216e-08	16
3.2	1.8514	1.8514	6.24795e-08	16
3.6	1.82195	1.82195	2.6337e-08	16
4	1.7582	1.7582	2.6013e-08	16

Составная квадратурная формула Гаусса: $J_N(x) = \sum_{i=0}^{n-1} \dfrac{h}{2}\ \left[f\left(x_i + \dfrac{h}{2}\ (1 - \dfrac{1}{\sqrt{3}})\right) + f\left(x_i + \dfrac{h}{2}\ (1\right.\right.$

x_i	$J_0(x)$	$J_N(x)$	$\mid J_0(x) - J_N(x) \mid$	N
0.4	0.396461	0.396461	1.73704e-09	4
0.8	0.772096	0.772096	5.50997e-08	4
1.2	1.10805	1.10805	2.34466e-08	8
1.6	1.38918	1.38918	1.40096e-10	16
2	1.60541	1.60541	1.50446e-08	16
2.4	1.75249	1.75249	2.49499e-08	16
2.8	1.8321	1.8321	3.14221e-08	16
3.2	1.8514	1.8514	4.44494e-08	16
3.6	1.82195	1.82195	4.36041e-08	16
4	1.7582	1.7582	2.39866e-08	16

Составная квадратурная формула центральных прямоугольников: $J_N(x) = \sum_{i=0}^{n-1} h \ f\left(\frac{x_i + x_{i+1}}{2}\right)$

x_i	$J_0(x)$	$J_N(x)$	$\mid J_0(x) - J_N(x) \mid$	N
0.8	0.772096	0.772096	1.01524e-07	256
1.2	1.10805	1.10805	3.16249e-07	256
1.6	1.38918	1.38918	1.71809e-07	512
2	1.60541	1.60541	2.75169e-07	512
2.4	1.75249	1.75249	9.76376e-08	1024
2.8	1.8321	1.8321	1.24621e-07	1024
3.2	1.8514	1.8514	1.22943e-07	1024
3.6	1.82195	1.82195	9.50707e-08	1024
4	1.7582	1.7582	2.99268e-07	512

ВЫВОДЫ ПО РАБОТЕ

:

Из представленных 6 методов самыми эффективными оказались методы Гаусса и Симпсона. Для метода Гаусса потребовалось меньше итераций, следовательно, он является самым результативным.

ЛИСТИНГ

```
#include <iostream>
#include <Math.h>
#include <string>
#include <vector>

static double step = 0.4;
static double Limit(double x)
{
	if (x != 0) return (sin(x) / x);
	return 1; //первый замечательный предел
}

static double Tabulate(double x)
{
	double a = x;
	double res = x;
	double q;
	int n = 0;
	do
	{
		q = (-1) * x * x * (2 * n + 1) / ((2 * n + 2) * (2 * n + 3) * (2 * n + 3));
		a *= q;
		res += a;
		n++;
	} while (abs(a) > 0.000001);
	return res;
}
static std::vector<double> Tabulate(std::vector<double> x)
{
	std::vector<double> result;
	for (int i = 0; i < x.size(); i++)
	{
		result.push_back(Tabulate(x[i]));
	}
	return result;
}
static double Left_Rectangle_Method(int N, double x0)
{
	double h = x0 / N;
	double result = 0;
	double x = 0;
	for (int i = 0; i < N; i++)
	{
		result += h * Limit(x);
		x += h;
	}
	return result;
```

```
}
staticdoubleRight_Rectangle_Method(intN, doublex0)
{
        double h = x0 / N;
        double sum = 0;
        double x = h;
        for (int i = 0; i < N; i++)
        {
                sum += h * Limit(x);
                x += h;
        }
        return sum;
}
staticdoubleCentral_Rectangles_Method(intN, doublex0)
{
        double h = x0 / N;
        double sum = 0;
        double x = h / 2;
        for (int i = 0; i < N; i++)
        {
                sum += h * Limit(x);
                x += h;
        }
        return sum;
}
staticdoubleSimpson_method(intN, doublex0)
{
        double h = x0 / N;
        double sum = 0;
        double x = 0;
        for (int i = 0; i < N; i++)
        {
                sum += (Limit(x) + 4 * Limit(x + h / 2) + Limit(x + h)) * h / 6;
                x += h;
        }
        return sum;
}
staticdoubleTrapezoid_method(intN, doublex0)
{
        double h = x0 / N;
        double result = 0;
        double x = 0;
        for (int i = 0; i < N; i++)
        {
                result += h * (Limit(x) + Limit(x + h)) / 2;
                x += h;
        }
        return result;
}
staticdoubleGauss_method(intN, doublex0)
{
        double h = x0 / N;
        double ad1 = (1 - 1.0 / sqrt(3)) * h / 2;
        double ad2 = (1 + 1.0 / sqrt(3)) * h / 2;
        double sum = 0;
        double x = 0;
        for (int i = 0; i < N; i++)
        {
                sum += (Limit(x + ad1) + Limit(x + ad2)) * h / 2;
                x += h;
```

```
        }
        return sum;
}
staticvoidCalculateAndWrite(doublex, doubley, double(*Integtal_Variant)(int, double))
{
        double lastJ = 0;
        double J = 0;
        int n = 1;
        do
        {
                if (n == 1024)
                {
                        break;
                }
                n *= 2;
                lastJ = J;
                J = Integtal_Variant(n, x);
        } while(abs(lastJ - J) > 0.000001);
        doubleerror = abs(J - y);
        std :: cout << | < x << "/t|| < y << "/t|| < J << "/t|| < error << "/t|| < n << "/n";
}
```