Шахназарова Хадижат Ильясовна

214 группа

Отчет по курсу

“Введение в численные методы”

Вариант 2-1

A. *Постановка задачи*

Найти приближенное значение интеграла методом трапеций, разбив интервал интегрирования на n частей, где n = 16,32,64.

Интеграл : $\int_a^b \frac{e^{-x}}{\sqrt{x}} dx$

Рассмотрим два отрезка интегрирования

1. a = 1, b = 2
2. a = 0, b = 1

Сравните результат с аналитическим значением интеграла.

Подберите более эффективный численный метод вычисления интеграла для второй задачи.

B. Основной метод решения

Опишем метод трапеций. Для начала рассмотрим интеграл $\int_a^b f(x)dx$, где f(x) определена и непрерывна на [a,b].

Теперь сам метод трапеций:
“Разрежем” отрезок [a,b] на n равные отрезки [x_{i-1} , x_i]. Распишем формулу нахождения x_i : x_i = a + i*h, где 0 ≤ i ≤ n и h - это ход, который вычисляется по формуле h = $\frac{(b-a)}{n}$.

Интеграл методом трапеций вычисляется по формуле:
$I \approx In = h * (0,5 * F_0 + F_1 + \cdots + F_{n-1} + 0.5 * F_n) + R$

Где

$$F_i = F(a + i * h), i = 0,1, \dots, n$$

Где R - это остаточный член($R = -\frac{(n \cdot h^3)}{12} f''(\beta),\ \beta \epsilon (a, b)$)

При каких условиях выполняется метод трапеций:

Функция должна быть определена и непрерывна на всем отрезке [a, b]. Пусть функция f(x) не определена в какой-либо точке, принадлежащей отрезку [a,b] ($x \in [a, b]$). И так как эта точка не определена на выделенном отрезке, то и функция F_i не определена на этом отрезке. А если не определена F_i, то и интеграл I_n не определен. Соответственно, метод трапеций будет не применим. А также функция должна быть непрерывно дифференцируема на отрезке [a,b](это нужно для нахождения остаточного члена).

C. Программная реализация

В моем отчете метод реализован на языке С.

```
#include <stdio.h>
#include <math.h>

double func(double x)
{
  return exp(-x)/sqrt(x);
}

double integral_trap(int n, int a, int b)
{
  double h = (b - a)/(double)n;
  double I = 0.5*(func(a) - func(b));

  for (int i = 1; i < n; i++)
  {
    double x = a + i * h;
    I += func(x);
  }
  return I*h;
}

int main()
{
  printf("16: integral  = %lf\n",integral_trap(16,0,1));
  printf("32: integral  = %lf\n",integral_trap(32,0,1));
  printf("64: integral  = %lf\n",integral_trap(64,0,1));
  printf("difference((I-In)): %f\n", (/*0.19815846732034428571*/ 1.49364826562485405079 - integral_trap(16,0,1)));
  printf("difference((I-In)): %f\n", (/*0.19815846732034428571*/ 1.49364826562485405079 - integral_trap(32,0,1)));
  printf("difference((I-In)): %f\n", (/*0.19815846732034428571*/ 1.49364826562485405079 - integral_trap(64,0,1)));

}
```

Где

- func(double x) - это функция, данная при условии

- Integral_trap(int n, int a, int b) вычисляет искомый интеграл методом трапеций

D. Анализ основного метода

1) При a = 1, b = 2

Так как данная функция определена и непрерывна на отрезке [1,2], то метод применим.

Интеграл:

I ≈ 0.19815846732034428571

Выводы основного метода:

```
16: integral  = 0.192318
32: integral  = 0.195203
64: integral  = 0.196672
difference((I-In)): 0.005840
difference((I-In)): 0.002955
difference((I-In)): 0.001486
```

Мы можем сделать небольшой вывод: чем больше n, тем точнее результат мы получаем.

2) При a = 0, b = 1

Данная функция не определена в точке 0. Вот какой результат должен выводиться:

I ≈ 1.49364826562485405079

И вот какой выводится:

```
16: integral  = inf
32: integral  = inf
64: integral  = inf
difference((I-In)): -inf
difference((I-In)): -inf
difference((I-In)): -inf
```

Вывод: Метод трапеций неприменим к функциям, не определенных в 0.

E. Метод Гаусса - Лежандра

Для более точного вычисления интеграла мы используем метод Гаусса - Лежандра. (I = $\int_a^b f(x)dx \ = \ \sum_{i=1}^{n} w_i f(a_i) + R_n$) – квадратурная формула).

Что из себя представляет данный метод?

Этот метод основан на выборе оптимальных узлов и весов. Узлы a_i и веса w_i выбираются так, чтобы аппроксимировать интеграл с высокой точностью для полиномов Лежандра. ($P_n(x) = \frac{1}{2^n n!}\frac{d}{dx^n}(x^2 \ - \ 1)^n$)

Для удобства вычисления рассмотрим интеграл на отрезке, поиск результатов для других отрезков интегрирования будет осуществляться простым пересчетом. Рассмотрим:

$\int_{-1}^{1} x^m dx = \frac{1-(-1)^{m+1}}{m+1} = \sum_{i=1}^{n} w_i f(a_i)$

Но эту систему уравнений сложно решить для произвольного числа узлов. Поэтому представим, что нам известен некоторый полином степени

$S = \ \prod_{i=1}^{n}(x - a_i)$

Корнями которого являются необходимые нам для построения квадратурной формулы узлы. Возьмем другой полином(любой) степени и выполним следующие вычисления:

$$\int_{-1}^{1} Q(x)S(x)dx = \sum_{i=1}^{n} w_i Q(a_i)S(a_i) = 0$$

Выражение не несет в себе противоречий. А также из него следует что полином ортогонален всякому полиному степени. Полином совпадает с полиномом Лежандра соответствующей степени. Полином Лежандра образует систему (ортогональную). Известно, что всякая система полиномов, которая обладает ортогональностью, определена однозначно с точностью до множителя. Поэтому мы можем сказать, что искомый полином представим в виде:

$$S(x) = c_n P_n$$

А это значит, что узлы, необходимые при вычислении, могут быть найдены как корни соответствующего полинома Лежандра.

Теперь получим веса. Используем для этого систему некоторых полиномов степени, для которых выполняется:

$$T_k(a_i) = \begin{cases} 0, i <> k \\ 1, i = k \end{cases}$$

Отсюда:

$$\int_{-1}^{1} T_k(x)dx = \sum_{i=1}^{n} w_i T_k(a_i) = w_k$$

А чтобы получить все весовые коэффициенты мы представим:

$$T_k(x) = \frac{(x - a_1) \dots (x - a_{k-1})(x - a_{k+1}) \dots (x - a_n)}{(a_k - a_1) \dots (a_k - a_{k-1})(a_k - a_{k+1}) \dots (a_k - a_n)}$$

Тогда:

$$w_i = \int_{-1}^{1} T_i(x)dx$$

Чтобы найти интеграл на отрезке [a,b] мы используем формулы:

$$\int_a^b f(x)dx => \text{Подставим} \begin{cases} x = \frac{a+b}{2} + \frac{b-a}{2} * t \\ dx = \frac{b-a}{2} dt \end{cases} => \int_{-1}^{1} f\left(\frac{a+b}{2} + \frac{b-a}{2} * t\right) * \frac{b-a}{2} dt$$
$$= \frac{b-a}{2} \int_{-1}^{1} f(\frac{a+b}{2} + \frac{b-a}{2} * t)dt$$

Метод Гаусса-Лежандра обеспечивает высокую точность при использовании относительно маленького числа узлов. Так как метод использует веса и узлы, оптимально подобранные для полиномов Лежандра, это позволяет достичь высокой точности даже с маленьким количеством узлов. В то время как в методе трапеций точность зависит от количества разбиений.

Алгебраическая точность:
У этого метода точность до (2n – 1), где n - количество узлов. Это значит, что метод точен для полиномов степени до (2n – 1).

F. Программная реализация

```c
// Функция, которую нужно проинтегрировать
double func(double x) {
    return exp(-x)/sqrt(x);
}

// Вычисление узлов и весов для метода Лежандра
void computeNodesAndWeights(double *nodes, double *weights, int n) {
    int i, j;
    double p1, p2, p3, pp, z, z1;

    // Цикл для вычисления узлов и весов
    for (i = 1; i <= n; i++) {
        z = cos(M_PI * (i - 0.25) / (n + 0.5)); // Узел - корень полинома Лежандра
        z1 = z + 1.0;
        while (fabs(z - z1) > 1e-10) {
            p1 = 1.0;
            p2 = 0.0;
            // Вычисление полинома Лежандра
            for (j = 0; j < n; j++) {
                p3 = p2;
                p2 = p1;
                p1 = ((2.0 * j + 1.0) * z * p2 - j * p3) / (j + 1);
            }
            // Вычисление производной полинома Лежандра
            pp = n * (z * p1 - p2) / (z * z - 1.0);
            z1 = z;
            z = z1 - p1 / pp; // Следующая итерация
        }
        // Узел метода Лежандра
        nodes[i - 1] = z;
        // Веса метода Лежандра
        weights[i - 1] = 2.0 / ((1.0 - z * z) * pp * pp);
    }
}
```

```
// Метод Лежандра для вычисления интеграла
double integral_Gaus(int n, double a, double b) {
    double nodes[n], weights[n];
    double result = 0.0;

    // Вычисление узлов и весов метода Лежандра
    computeNodesAndWeights(nodes, weights, n);

    // Вычисление интеграла методом Лежандра
    for (int i = 0; i < n; i++) {
        // Преобразование интервала [-1, 1] к интервалу [a, b]
        double x = ((b - a) * nodes[i] + (a + b)) / 2.0;
        result += weights[i] * func(x);
    }

    // Масштабирование результата к размеру интервала [a, b]
    result *= (b - a) / 2.0;

    return result;
}
```

G. Анализ вывода данных

1) При a = 1, b = 2

Интеграл: I ≈ 0.19815846732034428571

Вывод:

```
16: integral  = 0.198158
32: integral  = 0.198158
64: integral  = 0.198158
difference((I-In)): 0.000000
difference((I-In)): 0.000000
difference((I-In)): 0.000000
```

Чтобы увидеть хоть какую-то разницу умножим разницу еще на 10^5:

```
16: integral  = 0.198158
32: integral  = 0.198158
64: integral  = 0.198158
difference((I-In)*10^5): 0.000001
difference((I-In)*10^5): 0.000000
difference((I-In)*10^5): 0.000001
```

Заметим, что данный метод намного точнее вычисляет интеграл, в отличии от метода трапеций.

2) При a = 0, b = 1

Интеграл:

I ≈ 1.49364826562485405079

Выводы программы:

```
16: integral  = 1.440853
32: integral  = 1.466854
64: integral  = 1.480149
difference((I-In)): 0.052795
difference((I-In)): 0.026794
difference((I-In)): 0.013500
```

Данный метод вычислил интеграл, который не вычислялся методом трапеций. А так же дал приближенное значение интеграла.

H. Выводы

Использованные в этом отчете методы интегрирования позволяют приближённо вычислить определенные интегралы без формулы Ньютона-Лейбница. Также мы выяснили, что метод Гаусса-Лежандра эффективнее метода трапеций при одинаковом количестве разбиений. Подобранный мною метод также позволяет вычислить интегралы функций, которые не определены в 0.

Таблицы результатов:

Метод трапеций		
	a = 1, b = 2	a = 0, b = 1
Интеграл(I)	0.19815846732034428571	1.49364826562485405079
Интеграл(In) n = 16	0.192318	inf
Интеграл(In) n = 32	0.195203	inf
Интеграл(In) n = 64	0.196672	inf
Остаток(I-In)		

При n = 16	0.005840	- inf
При n = 32	0.002955	- inf
При n = 64	0.001486	- inf

Метод Гаусса-Лежандра		
	a = 1, b = 2	a = 0, b = 1
Интеграл(I)	0.19815846732034428571	1.49364826562485405079
Интеграл(In) n = 16	0.198158	1.440853
Интеграл(In) n = 32	0.198158	1.466854
Интеграл(In) n = 64	0.198158	1.480149
Остаток(I-In)		
При n = 16	0.000000(*10^5 = 0.000001)	0.052795
При n = 32	0.000000 (*10^5 = 0.000000)	0.026794
При n = 64	0.000000 (*10^5 = 0.000001)	0.013500

Литература, использовавшаяся в написании данного отчета:

1. Вводные лекции по численным методам, Д.П. Костомаров, А.П. Фаворский
2. Введение в численные методы, методическое пособие для 2 курса, Д.П. Костомаров
3. Лекции математического анализа 1 курс(2 семестр)