ПРАКТИЧЕСКАЯ работа № 7

МЕТОДЫ ПРИБЛИЖЕННОГО ВЫЧИСЛЕНИЯ

ИНТЕГРАЛОВ

Постановка задачи. Разработать функцию вычисления определенного интеграла с заданной точностью. Использовать разработанную функцию при вычислении интеграла от заданной подынтегральной функции. Вычисление подынтегральной функции оформить в виде функции. Результаты вычисления интеграла для заданных значений параметра представить в виде таблицы. Для вариантов 1, 2, 3, 5, 7, 12, 13, 14 включить в таблицу с результатами дополнительный столбец с количеством элементарных отрезков, которое использовалось для получения значений интеграла с заданной точностью. Варианты заданий приведены в табл. 8.

Таблица 8

**Варианты заданий**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Метод интегрирования | Подынтегральная функция | Пределы интегрирования (a и b) | Примечание |
| 1, 16 | Метод правых прямоугольников | xe-x | a=0, b - параметр | b=0.5, 1.0, 1.5, 2.0 |
| 2 | Метод Симпсона | xe-x | a=0, b - параметр | b=0.5, 1.0, 1.5, 2.0 |
| 3, 17 | Метод трапеций | xke-x  где k – параметр функции | a=0, b=1.5 | k=0, 1, 2, 3, 4 |
| 4 | Метод Веддля | cos(πx2/2) | a=0, b - параметр | b=0.5, 1.0, 1.5, 2.0 |
| 5 | Метод левых прямоугольников | x2e-tx, где t – параметр функции | a=-1, b=1  t=0.5, 1.0, 1.5, 2.0 | t=0.5, 1.0, 1.5, 2.0 |
| 6, 18 | Метод Ньютона-Котеса (m=3) | xk/(ex) , где k – параметр функции | a=0, b=1 | k=1, 2, 3, 4 |
| 7, 19 | Метод средних | e-x/(x2) | a=0, b - параметр | b=0.5, 1.0, 1.5, 2.0 |
| 8 | Метод Ньютона-Котеса (m=4) | sin(πx2/2) | a=0, b - параметр | b=0.5, 1.0, 1.5, 2.0 |
| 9, 20 | Метод Бодэ | e-x/(xk), где k – параметр функции | a=1, b=5 | k=1, 2, 3, 4 |
| 10 | Метод Ньютона-Котеса (m=5) | (1-k\*sin2(x))1/2, где k – параметр функции | a=0, b=π/2 | k=0.2, 0.3, 0.4, 0.5 |
| 11 | Метод Ньютона-Котеса (m=6) | (1-k\*sin2(x))1/2, где k – параметр функции | a=0, b=π/2 | k=0.1, 0.2, 0.3, 0.4 |
| 12 | Метод средних | x\*e-x | a=0, b - параметр | b=1, 2,…,10 |
| 13 | Метод Симпсона | x2e-0.5x | a=0, b - параметр | b=1, 2,…,10 |
| 14 | Метод правых прямоугольников | sin(πx2/2) | a=0, b - параметр | b=0.5, 1.0, 1.5, 2.0 |
| 15 | Метод Ньютона-Котеса (m=4) | xk/(ex) , где k – параметр функции | a=0, b=1 | k=1, 2, 3, 4 |

Методические указания

1. Формулы численного интегрирования

Метод левых прямоугольников



где n – количество элементарных отрезков, на которые разделяется отрезок интегрирования [a,b], h=(b-a)/n, х1=a, хi=хi-1+h, i=2, 3, …,n

Метод правых прямоугольников



где n – количество элементарных отрезков, на которые разделяется отрезок интегрирования [a,b], h=(b-a)/n, х1=a+h, хi=хi-1+h, i=2, 3, …,n

Метод средних



где n – количество элементарных отрезков, на которые разделяется отрезок интегрирования [a,b], h=(b-a)/n, х1=a+h/2, хi=хi-1+h, i=2, 3, …,n

Метод трапеций



где n – количество элементарных отрезков, на которые разделяется отрезок интегрирования [a,b], h=(b-a)/n, х1=a+h, хi=хi-1+h, i=1, 3, …,n-1

Метод Симпсона



где n (четное)– количество элементарных отрезков, на которые разделяется отрезок интегрирования [a,b], h=(b-a)/n

Метод Веддля

Метод базируется на применении к каждому из n элементарных отрезков длиной h=(b-a)/n формулы



где z=h/6, x1 – левая граница элементарного отрезка, х7 – правая граница элементарного отрезка, xi=xi-1+z, i=2, 3, 4, 5, 6, 7

Метод Ньютона-Котеса (для m=3)

Метод базируется на применении к каждому из n элементарных отрезков длиной h=(b-a)/n формулы



где z=h/3, x1 – левая граница элементарного отрезка, х4 – правая граница элементарного отрезка, xi=xi-1+z, i=2, 3, 4

Метод Ньютона-Котеса (для m=4)

Метод базируется на применении к каждому из n элементарных отрезков длиной h=(b-a)/n формулы



где z=h/4, x1 – левая граница элементарного отрезка, х5 – правая граница элементарного отрезка, xi=xi-1+z, i=2, 3, 4, 5

Метод Ньютона-Котеса (для m=5)

Метод базируется на применении к каждому из n элементарных отрезков длиной h=(b-a)/n формулы



где z=h/5, x1 – левая граница элементарного отрезка, х6 – правая граница элементарного отрезка, xi=xi-1+z, i=2, 3, 4, 5, 6

Метод Ньютона-Котеса (для m=6)

Метод базируется на применении к каждому из n элементарных отрезков длиной h=(b-a)/n формулы



где z=h/6, x1 – левая граница элементарного отрезка, х7 – правая граница элементарного отрезка, xi=xi-1+z, i=2, 3, 4, 5, 6, 7

Метод Бодэ

Метод базируется на применении к каждому из n элементарных отрезков длиной h=(b-a)/n формулы



где z=h/4, x1 – левая граница элементарного отрезка, х5 – правая граница элементарного отрезка, xi=xi-1+z, i=2, 3, 4, 5

2. Для достижения заданной точности вычисления интеграла использовать алгоритм последовательного удвоения количества элементарных отрезков. Алгоритм состоит в вычислении приближения выбранным методом с шагом h, а затем с шагом h/2, и сравнением разности двух приближений с заданной точностью вычисления интеграла.

3. Для отладки алгоритма использовать простую подынтегральную функцию, для которой интеграл можно вычислить аналитически, например, f(x)=x или f(x)=x2. После отладки эту функцию заменить на функцию Вашего варианта.

Пример программы

Программа вычисления корня нелинейного уравнения f(x)=0 на интервале (a, b), на котором функция f(x) меняет знак. В программе используется функция *metod,* котораявычисляет корень уравнения с точностью *eps* методом деления отрезка пополам. Заголовок функции:

*float metod (float a, float b, float eps, float(\* f)(float x))*

Программа использует функцию *metod* для вычисления корня уравнения:

*x3-2x-5=0*

Корень находится между 2 и 3.

#include <iostream.h>

#include <conio.h>

//Функция вычисления корня уравнения

float metod (float a, float b, float eps, float(\* f)(float x));

float f(float x); //функция задает нелинейное уравнение

void main( )

{

float a=2, b=3; //левая и правая границы интервала нахождения корня

float eps; //точность вычисления корня

cout<<”eps? “;

cin>>eps;

cout<<metod(a,b,eps,f);

getch();

}

float metod (float a, float b, float eps, float(\* f)(float x))

{

float x; //приближение корня

float u,v;//значение функции f(x) на левой и правой границах интервала

u=f(a);

do

{

x=(a+b)/2;

v=f(x);

if (u\*v<0) //функция имеет разные знаки на интервале [a,x]

b=x; //выбирается левый интервал [a, x] для поиска корня

else //выбирается правый интервал [x, b] для поиска корня

{

a=x; u=v;

}

}while (b-a>eps);

return x;

}

float f(float x)

{

return x\*x\*x-2\*x-5;

}