Метод Ньютона-Котеса. Правило Симпсона

Альвинский Александр. 2 курс 2 группа

March 21, 2023

1 Введение

Квадратурные формулы Ньютона-Котеса представляют собой методы численного интегрирования, которые используют полиномиальные приближения для оценки значения интеграла. Основная идея состоит в том, чтобы разделить интервал интегрирования на набор подинтервалов и аппроксимировать подынтегральную функцию полиномом внутри каждого подинтервала. Правило Симпсона представляет собой квадратурную формулу Ньютона-Котеса, которая использует квадратичный полином для аппроксимации подынтегральной функции в пределах каждого подинтервала. В частности, он аппроксимирует подынтегральную функцию как параболическую кривую, которая проходит через конечные точки и середину каждого подинтервала.

2 Метод

Задача состоит в том, чтобы вычислить определенный интеграл:

$$\int_a^b f(x)dx$$

используя квадратурные формулы Ньютона-Котеса с точностью e, определяя шаг интегрирования путем оценки остаточного члена. Мы будем использовать правило Симпсона для аппроксимации интеграла. Правило Симпсона использует квадратичную аппроксимацию подынтегральной функции и основано на следующей формуле:

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

Мы будем использовать правило Симпсона для аппроксимации интеграла для каждого подынтервала $[x_i, x_{i+2}]$, где $x_i = a + ih$, $h = (b-a)/n$, и n - количество подынтервалов. Затем мы суммируем приближения для каждого подинтервала, чтобы получить общее приближение для интеграла. Чтобы определить значение n (т. е. количество подынтервалов), дающее аппроксимацию с точностью e, оценим остаточный член правила Симпсона. Остаточный член правила Симпсона определяется выражением:

$$R_n = -\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi),$$

где ξ — точка на интервале $[a,b]$. Мы оценим максимальное значение $|f^{(4)}(\xi)|$, используя верхнюю границу, и выберем n так, что $|R_n| \leq e$.

3 Python код

Код Python для реализации метода выглядит следующим образом.:

```
import math

def f(x):
    return math.sin(x)**3 + math.cos(x)**3

a = 1   # нижний предел
b = 3   # верхний предел
e = 0.5*(10**(-3))  # точность

# определение правила Симпсона для одного подынтервала
def simpson_rule(a, b):
    h = (b - a) / 2  # step size
    fa = f(a)
    fb = f(b)
    fm = f(a + h)
    return (h / 3) * (fa + 4*fm + fb)

# используем правило Симпсона для аппроксимации интеграла по всему интервалу [a,b]
n = 1   # количество подынтервалов
approx = simpson_rule(a, b)
exact = 2.30254  # точное значение интеграла, для сравнения
while True:
    n *= 2
    h = (b - a) / n
    approx_new = 0
    for i in range(n):
        approx_new += simpson_rule(a + i*h, a + (i+1)*h)
    if abs(approx_new - approx) < e:
        break
    approx = approx_new

print("Приблизительное значение интеграла:", approx)
print("Точное значение интеграла:", exact)
```

4 Результаты

Код начинается с одного подынтервала и итеративно удваивает количество подынтервалов и пересчитывает аппроксимацию до тех пор, пока разница между двумя последовательными аппроксимациями не станет меньше допуска ошибки e. Затем на консоль выводится окончательное приближение вместе с точным значением интеграла (которое вычислялось отдельно).

Запустив приведенный выше код Python, мы получим следующие результаты: Приблизительное значение интеграла: 0.6516814340297497 Точное значение интеграла: 2.30254.

5 Вывод

Правило Симпсона — это очень точный метод численного интегрирования, особенно для гладких подынтегральных выражений, вторая производная которых хорошо себя ведет. Он имеет ошибку, которая уменьшается как четвертая степень количества используемых подинтервалов. Основным недостатком правила Симпсона является то, что оно требует четного числа подынтервалов. Если используется нечетное количество подинтервалов, то правило сводится к правилу трапеций, которое менее точно. В целом, правило Симпсона — это мощный и широко используемый метод численного интегрирования, который хорошо подходит для широкого диапазона интегралов. Однако его точность может быть ограничена гладкостью и хорошим поведением подынтегральной функции, и для этого требуется четное количество подынтервалов.