БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

**ЛАБОРАТОРНАЯ РАБОТА №3**

**“ Приближенное вычисление интегралов ”**

*Полевиков Виктор Кузьмич*

*Ахрамович Виктория Вячеславовна*

Минск, 2022

**Поставленная задача**

Вычислить интеграл с точностью , оценивая точность по правилу Рунге, с помощью следующих квадратурных формул:

1. составная КФ правых прямоугольников
2. составная КФ средних прямоугольников
3. составная КФ Симпсона

Вычислить интеграл с помощью КФ НАСТ при

**Краткая теория**

**Вариант №4**

Доопределим подынтегральную функцию в точках неопределенности

Найдем значения и .

1. **составная КФ правых прямоугольников**

Т.к. остаток КФ правых прямоугольников , порядок точности .

1. **составная КФ средних прямоугольников**

Т.к. остаток КФ средних прямоугольников , порядок точности .

1. **составная КФ Симпсона**

Т.к. остаток КФ Симпсона , порядок точности .

**Правило Рунге**

Остаток любой из рассмотренных составных КФ имеет вид:

, где не зависящая от шага h.

Как только выполнится неравенство:

значение считается искомым приближенным значением интеграла, найденным с точностью

Точность по правилу Рунге:

1. – **метод правых прямоугольников**
2. - **метод** **средних прямоугольников**
3. - **метод** **Симпсона**

**Вычисление интегралов с помощью КФ НАСТ**

Для вычисления требуется построить КФ НАСТ :

В общем случае эта задача является сложной и не всегда разрешимой. Однако для многих весовых функций построение КФ НАСТ удается значительно упростить.

Приведем исходный интеграл к виду весом

Тогда — корни многочлена Лежандра , имеющего вид

Вычислим корни

Коэффициенты находятся по следующей формуле

**Листинг программы (Python)**

import math  
  
def myfun(x):  
 if x == 0:  
 return 0  
 else:  
 return math.log(1+x\*x)/x  
  
  
def myfunt(x):  
 return math.log(1+(x+1)\*\*2/4)/(x+1)  
  
  
def eps(I1, I2, m):  
 return math.fabs(I1 - I2) / (2 \*\* m - 1)  
  
  
def rightrectint(fun, a, b, N):  
 res = 0  
 h = (b - a) / N  
 for i in range(1, N + 1):  
 res += fun(a + h \* i)  
 return h \* res  
  
  
def middlerectint(fun, a, b, N):  
 res = 0  
 h = (b - a) / N  
 for i in range(N):  
 xi = a + h \* i  
 xi1 = a + h \* (i + 1)  
 res += fun((xi + xi1) / 2)  
 return h \* res  
  
  
def simpson(fun, a, b, N):  
 res = 0  
 h = (b - a) / N  
 for i in range(N):  
 xi = a + h \* i  
 xi1 = a + h \* (i + 1)  
 res += fun(xi) + 4 \* fun((xi + xi1) / 2) + fun(xi1)  
 return h \* res / 6  
  
  
def runge(func, formula, a, b, e, m):  
 N = 1  
 x = formula(func, a, b, N)  
 y = 0  
 while eps(x, y, m) >= e:  
 y = x  
 N \*= 2  
 x = formula(func, a, b, N)  
 print(y)  
 print(N)  
 return y  
  
a = 0  
b = 1  
e = 10 \*\* (-6)  
x1 = runge(myfun, rightrectint, a, b, e, 1)  
x2 = runge(myfun, middlerectint, a, b, e, 2)  
x3 = runge(myfun, simpson, a, b, e, 4)  
print()  
b = 1  
  
A00 = 2  
A20 = 0.55555555555555556  
A21 = 0.55555555555555556  
A22 = 0.88888888888888889  
A40 = 0.5688888888888889  
A41 = 0.23692688505618878  
A42 = 0.23692688505618878  
A43 = 0.47862867049936675  
A44 = 0.47862867049936675  
  
nast0 = A00 \* myfunt(0)  
nast2 = A20 \* myfunt(0.6 \*\* 0.5) + A21 \* myfunt(-0.6 \*\* 0.5) + A22 \* myfunt(0)  
nast4 = (A40 \* myfunt(0) + A41 \* myfunt(((70 + 1120 \*\* 0.5) / 126) \*\* 0.5) + A42 \* myfunt(-((70 + 1120 \*\* 0.5) / 126) \*\* 0.5) +  
A43 \* myfunt(((70 - 1120 \*\* 0.5) / 126) \*\* 0.5) + A44 \* myfunt(-((70 - 1120 \*\* 0.5) / 126) \*\* 0.5))  
  
print(nast0)  
print(nast2)  
print(nast4)

**Результаты**

**Значение интеграла:** 0.644934066848

**КФ правых прямоугольников:**

**=** 0.644935388920

= 524288

= 1.322072488207926 \*

**КФ средних прямоугольников:**

**=** 0.644932832563

= 256

= 1.234285829582938 \*

**КФ Симпсона:**

**=** 0.644938393575

= 8

= 4.326727749888072 \*

**КФ НАСТ:**

= 0.642298708257

= 0.02635358591636254

= 0.644974905887

= 4.0839039048148322\*

= 0.644934126436

= 5.958891630131191\*