**Министерство образования Республики Беларусь**

**Белорусский государственный университет**

Факультет прикладной математики и информатики

Лабораторная работа №3

По курсу «Численные методы»

**Численное интегрирование**

Вариант №2

Работу выполнил:

студент 3 курса 7 группы

**Шатерник Артём**

Преподаватель:

**Будник А. М**.

**Минск 2024**

1. **Постановка задачи.**

Необходимо вычислить интеграл

с точностью

1. Применяя правило Рунге и составную квадратурную формулу правых прямоугольников определить величину шага для достижения точности .
2. Пользуясь выражениями для погрешностей интегрирования, определить шаги в следующих случаях:

* Составная квадратурная формула средних прямоугольников.
* Составная квадратурная формула Симпсона.

1. Применить формулу НАСТ Гаусса при . Оценить погрешность интегрирования.
2. **Алгоритм решения.**
3. Составная квадратурная формула правых прямоугольников.

где величина разбиения, шаг разбиения,

Для получения заданной точности будем использовать правило Рунге. В качестве начального шага возьмём следующий шаг будем брать по формуле:

При каждом шаге будем вычислять остаток вида:

И продолжать процесс до те пор, пока не выполниться условие:

1. Интегрирование с использованием априорной оценки.

* Составная квадратурная формула средних прямоугольников.

Формула имеет вид:

Оценивать количество разбиений будем из соотношения:

* Составная квадратурная формула Симпсона.

Формула имеет вид:

где а это значения разбиения.

Оценивать количество разбиений будем из соотношения:

1. Формула НАСТ Гаусса при n = 4.

Для построения формулы требуется многочлен Лежандра с индексом

*.* Он имеет вид:

Его корни, найденные с помощью Wolfram Mathematica, имеют вид:

Далее находим коэффициенты по формуле:

После нужно применить к преобразование, для того чтобы применять вышеприведённые формулы на отрезке Оно имеет вид:

Теперь можно строить квадратурную формулу Гаусса, которая, учитывая отрезок интегрирования, примет вид:

Остаток квадратурной формулу можно оценить по формуле:

где

**3. Листинг программы.**

**import math  
import numpy as np**

**# Условие  
a, b = 0, 1  
real\_res = -0.1978168271761323678264082  
def func(x):  
 return (x\*\*2 - 1) \* 10\*\*(-2 \* x)  
  
# Разбиение отрезка [a, b]  
N = 10  
h = (b - a) / N  
split = [h \* i for i in range(N + 1)]  
print(f"Разбиение:\n{split}")**

**# Составные правые прямоугольники по правилу Рунге  
def r\_triangle\_comp(f, a, b, epsilon):  
 h1 = 0.1  
 Ih1 = 0  
 N = int((b - a) / h1)  
 for i in range(N):  
 Ih1 += f(a + (i + 1) \* h1)  
 Ih1 \*= h1  
 while True:  
 h2 = 0.5 \* h1  
 N = int((b - a) / h2)  
 Ih2 = 0  
 for i in range(N):  
 Ih2 += f(a + (i + 1) \* h2)  
 Ih2 \*= h2  
 if abs((Ih2 - Ih1) / (1 - h2 / h1)) <= epsilon:  
 return Ih2, h2  
 h1, Ih1 = h2, Ih2  
  
  
eps = 1e-5  
res\_r\_tr\_comp = r\_triangle\_comp(func, a, b, eps)  
print(f"Значение интеграла: {res\_r\_tr\_comp[0]}\nШаг разбиения: {res\_r\_tr\_comp[1]}")  
print(f"Невязка с реальным решением: {abs(res\_r\_tr\_comp[0]) - abs(real\_res)}")**

**# Оцениваем разбиения  
def func\_der(x):  
 return 2 \* x \* pow(10, -2 \* x) + (x\*\*2 - 1) \* math.log(10) \* pow(10, -2 \* x) \* (- 2)  
  
def func\_der\_2(x):  
 return (2 - 8 \* math.log(10) \* x + 4 \* math.log(100) \* x\*\*2 - 4 \* math.log(100)) \* pow(10, -2 \* x)**

**# Максимум второй производной  
x = np.arange(a, b, 0.00001)  
M2 = max([abs(func\_der\_2(i)) for i in x])  
# M2 = 19.2076  
print(F"M2: {M2}")  
N\_mid = math.ceil(math.sqrt((b - a)\*\*3 / (24 \* eps) \* M2))  
print(f"Число разбиений для средних прямоугольников: {N\_mid}")  
h\_mid = (b - a) / N\_mid  
print(f"Шаг для средних прямоугольников: {h\_mid}")**

**M4 = 195.27086  
N\_simp = math.ceil(pow((b - a)\*\*5 / (2880 \* eps) \* M4, 1 / 4))  
print(f"Число разбиений для Симпсона: {N\_simp}")  
h\_simp = (b - a) / N\_simp  
print(f"Шаг для Симпсона: {h\_simp}")**

**# Средние прямоугольники  
def m\_triangle\_comp(f, a, b, h):  
 N = int((b - a) / h)  
 sum = 0  
 for i in range(N):  
 sum += f(a + (i + 0.5) \* h)  
 return sum \* h  
  
res\_mid\_tr\_comp = m\_triangle\_comp(func, a, b, h\_mid)  
print(f"Значение интеграла: {res\_mid\_tr\_comp}")  
print(f"Невязка с реальным решением: {abs(res\_mid\_tr\_comp) - abs(real\_res)}")**

**# Симпсон  
def simp\_comp(f, a, b, h):  
 N = int((b - a) / h)  
 split = [h \* i for i in range(N + 1)]  
 temp1 = sum([f(split[i]) for i in range(0, N - 1, 2)])  
 temp2 = sum([f(split[i]) for i in range(1, N, 2)])  
 return h / 3 \* (split[0] + split[N] + 2 \* temp1 + 4 \* temp2)  
  
  
res\_simp\_comp = simp\_comp(func, a, b, h\_simp)  
print(f"Значение интеграла: {res\_simp\_comp}")  
print(f"Невязка с реальным решением: {abs(res\_simp\_comp) - abs(real\_res)}")**

**# НАСТ Гаусса при n = 4  
# Корни n + 1 многочлена Лежандра  
roots = [0, 1 / 3 \* math.sqrt(5 - 2 \* math.sqrt(10 / 7)), - 1 / 3 \* math.sqrt(5 - 2 \* math.sqrt(10 / 7)),  
 1 / 3 \* math.sqrt(5 + 2 \* math.sqrt(10 / 7)), - 1 / 3 \* math.sqrt(5 + 2 \* math.sqrt(10 / 7))]  
print(roots)**

**# Коэффициенты Ak  
def p\_der\_5(x):  
 return 15 / 8 \* (21 \* x\*\*4 - 14 \* x\*\*2 + 1)  
  
A = []  
for root in roots:  
 A.append(2 / ((1 - root\*\*2) \* p\_der\_5(root)\*\*2))  
print(A)**

**# Преобразуем корни для отрезка [a, b]  
for i in range(len(roots)):  
 roots[i] = (b - a) / 2 \* roots[i] + (a + b) / 2  
print(roots)**

**# Квадратурная формула  
sum = 0  
for i in range(len(roots)):  
 sum += A[i] \* func(roots[i])  
  
res\_gauss = sum \* (b - a) / 2  
print(f"Значение интеграла: {res\_gauss}")  
print(f"Невязка с реальным решением: {abs(res\_gauss) - abs(real\_res)}")**

**# Остаток квадратурной формулы  
# Нужна производная от функции степени 2 \* n + 2. При n = 4: 10 степени.  
# Берём max 10 производной на отрезке [a, b]  
M\_10 = 1.39157 \* 10\*\*7  
n = 4  
r\_gauss = 2\*\*(2 \* n + 3) / ((2 \* n + 3) \* math.factorial(2 \* n + 2)) \  
 \* (math.factorial(n + 1)\*\*2 / math.factorial(2 \* n + 2))\*\*2 \* M\_10  
r\_gauss = r\_gauss \* ((b - a) / 2)\*\*(2 \* n + 3)  
print(f"Оценка погрешности: {r\_gauss}")**

1. **Результат и его анализ.**

В качестве эталонного значения интеграла будем использовать значение, полученное используя Wolfram Mathematica с точностью больше 5 знаков:

**1.** **Составная квадратурная формула правых прямоугольников и правило Рунге.**

Значение интеграла:

Число разбиений:

Шаг:

Невязка с реальным решением:

Вывод:

Получили значение интеграла с точностью что даже превышает заданную нами точность. Связанно это с тем, что при каждой итерации мы уменьшаем шаг вдвое, в результате мы слегка перешагнули и в теории можно взять шаг слегка больше, чтобы иметь меньшее разбиение.

**2.** **Интегрирование с использованием априорной оценки.**

Составная квадратурная формула средних прямоугольников.

Значение найдено в Wolfram Mathematica.

Число разбиений:

Шаг:

Значение интеграла:

Невязка с реальным решением:

Вывод:

Получили значение интеграла с точностью что опять-таки превышает заданную точность. На этот раз это связанно с тем, что мы используем лишь оценку погрешности, а не её точное значение. Если сравнивать данный метод с прошлым, то он является заметно более выгодным с точки зрения числа разбиений.

Составная квадратурная формула Симпсона.

Число разбиений:

Шаг

Значение интеграла:

Невязка с реальным решением:

Вывод:

На этот раз мы получили значение интеграла с заданной точностью . Получили довольно малое число разбиений, особенно если сравнивать с прошлыми методами. Однако для вычисления числа разбиений понадобилось вычислять производную уже 4 порядка.

**3. Формула НАСТ Гаусса при n = 4.**

Корни до преобразования:

].

Корни после преобразования:

.

После преобразования корни лежат на отрезке [0, 1].

Коэффициенты

.

Значение интеграла:

Невязка с реальным решением: .

Оценка погрешности: .

Вывод:

Получили значение интеграла с точностью теоретическое значение погрешности больше, чем реальное, что нормально, так как мы используем оценку производной при подсчёте погрешности. Данный метод дал нам наибольшую точность среди представленных методов, связанно это с тем, что он имеет наибольшую АСТ. Сложность подсчёта при этом значительно не выросла. Нам нужно предварительного рассчитать коэффициенты и корни многочлена Лежандра, но требуется сделать это всего один раз.