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

$$\int_0^1 (x^2 - 1) * 10^{-2x} dx$$

с точностью $\varepsilon = 10^{-5}$.

1. Применяя правило Рунге и составную квадратурную формулу правых прямоугольников определить величину шага h для достижения точности ε.
2. Пользуясь выражениями для погрешностей интегрирования, определить шаги h в следующих случаях:
   - Составная квадратурная формула средних прямоугольников.
   - Составная квадратурная формула Симпсона.
3. Применить формулу НАСТ Гаусса при $n = 4$. Оценить погрешность интегрирования.

2. **Алгоритм решения.**

1. Составная квадратурная формула правых прямоугольников.

$$I_{\text{пс}}(f) = h * \sum_{k=0}^{N-1} f(a + (k+1) * h),$$

где $N -$ величина разбиения, $h -$ шаг разбиения, $f(x) = (x^2 - 1) * 10^{-2x}$, $a = 0,\ b = 1$.

Для получения заданной точности будем использовать правило Рунге. В качестве начального шага возьмём $h1 = 0.1$, следующий шаг будем брать по формуле: $h2 = 0.5 * h1$.

При каждом шаге будем вычислять остаток вида:

$$R(h, f) = \frac{I_{h2} - I_{h1}}{1 - \frac{h2}{h1}}.$$

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

$$|R(h, f)| \leq \varepsilon.$$

2. Интегрирование с использованием априорной оценки.
   - Составная квадратурная формула средних прямоугольников.

   Формула имеет вид:

$$I_{cc}(f) = h * \sum_{k=0}^{N-1} f\left(a + \left(k + \frac{1}{2}\right) * h\right).$$

Оценивать количество разбиений будем из соотношения:

$$N_c \geq \sqrt[2]{\frac{(b-a)^3}{24\varepsilon} * M_2}, \quad M_2 = \max_{[a,b]} |f''(x)|.$$

- Составная квадратурная формула Симпсона.

Формула имеет вид:

$$I_{\text{симп с}}(f) = \frac{h}{3}\big(f_0 + f_N + 2(f_2 + f_4 + \dots + f_{N-2}) + 4(f_1 + f_3 + \dots + f_{N-1})\big),$$

где $f_i = f(x_i)$, а x_i − это значения разбиения.

Оценивать количество разбиений будем из соотношения:

$$N_c \geq \sqrt[4]{\frac{(b-a)^5}{2880\varepsilon} * M_4}, \quad M_4 = \max_{[a,b]}\left|f^{(4)}(x)\right|.$$

3. Формула НАСТ Гаусса при n = 4.

Для построения формулы требуется многочлен Лежандра с индексом $n + 1 = 5$. Он имеет вид:

$$P_5(x) = \frac{1}{8}(63x^5 - 70x^3 + 15x).$$

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

$X_5 = [0, \quad 0.538469310105683, \quad -0.538469310105683, \quad 0.906179845938664, \quad -0.906179845938664].$

Далее находим коэффициенты A_k по формуле:

$$A_k = \frac{2}{2(1 - x_k^2) * \left(P'_{n+1}(x_k)\right)^2}, k = \overline{0, n}.$$

После нужно применить к x_k преобразование, для того чтобы применять вышеприведённые формулы на отрезке $[a, b] = [0,1]$. Оно имеет вид:

$$x = \frac{b-a}{2}x' + \frac{b+a}{2}, x' \in [-1,1].$$

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

$$I_g = \frac{b-a}{2} * \sum_{0}^{n} A_k f(x_k).$$

Остаток квадратурной формулу можно оценить по формуле:

$$R_n(f) = \left(\frac{b-a}{2}\right)^{2n+3} * \frac{2^{2n+3}}{(2n+3)(2n+2)!} * \left(\frac{(n+1)!^2}{(2n+2)!}\right)^2 * M_g,$$

где

$$M_g = \max_{[a,b]}\left|f^{(2n+2)}(x)\right|.$$

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}")
```

4. Результат и его анализ.

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

$$-0.197816827176132367826408 2.$$

1. Составная квадратурная формула правых прямоугольников и правило Рунге.

Значение интеграла: -0.19781072371744327.
Число разбиений: 81920.
Шаг: $1.220703125e-05$.
Невязка с реальным решением: $-6.10345868909401e-06$.

Вывод:
Получили значение интеграла с точностью 10^{-6}, 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.

$$M_2 = \max_{[0,1]} |f''(x)| = 16.420680743952367.$$

Значение найдено в Wolfram Mathematica.
Число разбиений: 262.
Шаг: 0.003816793893129771.
Значение интеграла: -0.19781404401175506.
Невязка с реальным решением: $-2.783164377295755e-06$.

Вывод:
Получили значение интеграла с точностью 10^{-6}, 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.

$$M_4 = \max_{[0,1]} \left|f^{(4)}(x)\right| = 195.27086.$$

Число разбиений: 10.
Шаг: 0.1.
Значение интеграла: -0.1978550914154395.
Невязка с реальным решением: $3.8264239307139736e-05$.

Вывод:
На этот раз мы получили значение интеграла с заданной точностью 10^{-5}. 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 x_k до преобразования:
$[0, \quad 0.538469310105683, \quad -0.538469310105683, \quad 0.906179845938664, \quad -0.906179845938664]$.
Корни x_k после преобразования:
$[0.5, \quad 0.7692346550528415, \quad 0.2307653449471585, \quad 0.9530899229693319, \quad 0.04691007703066802]$.

После преобразования корни лежат на отрезке [0, 1].
Коэффициенты A_k:
$[0.5688888888888889,$
$0.47862867049936636,$
$0.47862867049936636,$
$0.23692688505618922,$
$0.23692688505618922]$.

Значение интеграла: -0.1978171088237481.
Невязка с реальным решением: $2.8164761572968544e-07$.

$$M_{10} = \max_{[0,1]} \left|f^{(10)}(x)\right| = 1.39157 \, * \, 10^{7}.$$
Оценка погрешности: $5.4896955256250204e-06$.

Вывод:
Получили значение интеграла с точностью 10^{-7}, теоретическое значение погрешности больше, чем реальное, что нормально, так как мы используем оценку производной M_{10} 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.