МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №4
«Численные методы»
Вариант 1

Бобовоза Владислава Сергеевича
студента 3 курса, 6 группы
специальность «прикладная математика»

Преподаватель:
Репников В.И.

Минск, 2024

Постановка задач

Задача 1. Построить квадратурную формулу максимально возможной степени точности вида

$$\int_0^1 \frac{f(x)}{\sqrt{x}}dx \approx A_0 f(x_0) + A_1 f(x_1).$$

Задача 2. Определить алгебраическую степень точности указанной квадратурной формулы

$$\int_0^1 f(x)\,dx \approx \frac{5}{18} f\left(\frac{1}{2} - \sqrt{\frac{3}{5}}\right) + \frac{4}{9} f\left(\frac{1}{2}\right) + \frac{5}{18} f\left(\frac{1}{2} + \sqrt{\frac{3}{5}}\right).$$

Задача 3. Используя правило Рунге, провести сравнительный анализ квадратурных формул левых прямоугольников и Симпсона на примере вычисления интеграла $I = \int_1^2 \frac{e^x}{x+1}dx$.

Задача 4. Вычислить (используя квадратурную формулу типа Гаусса) с точностью $\varepsilon = 10^{-4}$ интеграл $I = \int_0^5 e^{-x^2+x}dx$.

Задача 5. Найти с точностью $\varepsilon = 10^{-4}$ решение уравнения $\frac{1}{\sqrt{2\pi}}\int_0^x e^{-t^2}dt = 0.3$.

Решение задачи 1

Воспользуемся следующей системой:

$$\begin{cases} \int_a^b p(x)\, x^i dx = \sum_{k=0}^{n} A_k x_k^i, \quad i = \overline{0, m} \\ \int_a^b p(x)\, x^{m+1} dx \neq \sum_{k=0}^{n} A_k x_k^i \end{cases}$$

в этом случае, считаем, что квадратурная формула имеет АСТ = m.

Из условия видно, что у нас 4 неизвестные, тогда построим систему из 4-ех уравнений:

$$\begin{cases} \int_0^1 \frac{1}{\sqrt{x}}dx = A_0 + A_1 = 2 \\ \int_0^1 \frac{x}{\sqrt{x}}dx = A_0x_0 + A_1x_1 = \frac{2}{3} \\ \int_0^1 \frac{x^2}{\sqrt{x}}dx = A_0x_0^2 + A_1x_1^2 = \frac{2}{5} \\ \int_0^1 \frac{x^3}{\sqrt{x}}dx = A_0x_0^3 + A_1x_1^3 = \frac{2}{7} \end{cases}$$

Домножим на $-x_0$ первые три уравнения и прибавим их к последним трем соответственно, тогда получим следующую систему с 3-мя неизвестными:

$$\begin{cases} A_1x_1 - A_1x_0 = \frac{2}{3} - 2x_0 \\ A_1x_1^2 - A_1x_1x_0 = \frac{2}{5} - \frac{2}{3}x_0 \\ A_1x_1^3 - A_1x_1^2x_0 = \frac{2}{7} - \frac{2}{5}x_0 \end{cases}$$

Домножим на $-x_1$ первые два уравнения и прибавим их к последним двум соответственно, тогда получим следующую систему с 2-мя неизвестными:

$$\begin{cases} x_0 + x_1 - 3x_0x_1 = \frac{3}{5} \\ x_0 + x_1 - \frac{5}{3}x_0x_1 = \frac{5}{7} \end{cases}$$

Вычтем из второго первое, тогда получим:

$$\frac{4x_0x_1}{3} = \frac{4}{35}$$

Отсюда имеем:

$$x_0x_1 = \frac{3}{35}$$

Положим что $x_1 \neq 0$, тогда можем выразить x_0:

$$x_0 = \frac{3}{35x_1}$$

Подставим в первое уравнение и решим его, тогда получим:

$$x_1 = \frac{3}{7} + \frac{2\sqrt{30}}{35}, \quad x_1 = \frac{3}{7} - \frac{2\sqrt{30}}{35}$$

Отсюда получаем:

$$x_0 = \frac{3}{7} - \frac{2\sqrt{30}}{35}, \quad x_0 = \frac{3}{7} + \frac{2\sqrt{30}}{35}$$

Будем рассматривать следующий случай:

$$x_0 = \frac{3}{7} - \frac{2\sqrt{30}}{35}, \quad x_1 = \frac{3}{7} + \frac{2\sqrt{30}}{35}$$

Подставим полученные значения в первые два уравнения системы, чтобы найти A_0, A_1:

$$\begin{cases} A_0 + A_1 = 2 \\ \dfrac{\left(15 - 2\sqrt{30}\right) A_0 + \left(15 + 2\sqrt{30}\right) A_1}{35} = \dfrac{2}{3} \end{cases}$$

Тогда можем выразить из первого уравнения $A_0 = 2 - A_1$ и подставим во второе:

$$\frac{\left(15 - 2\sqrt{30}\right)(2 - A_1) + \left(15 + 2\sqrt{30}\right) A_1}{35} = \frac{2}{3}$$

Раскроем скобки, преобразуем и решим данное уравнение, в итоге получим:

$$A_1 = 1 - \frac{\sqrt{30}}{18}$$

Отсюда найдем A_0:

$$A_0 = 1 + \frac{\sqrt{30}}{18}$$

Тогда имеем:

$$\begin{cases} A_0 = 1 + \dfrac{\sqrt{30}}{18} \\ A_1 = 1 - \dfrac{\sqrt{30}}{18} \\ x_0 = \dfrac{3}{7} - \dfrac{2\sqrt{30}}{35} \\ x_1 = \dfrac{3}{7} + \dfrac{2\sqrt{30}}{35} \end{cases}$$

Таким образом была построена квадратурная формула, имеющая минимум АСТ = 3. Проверим, может быть она имеет АСТ = 4:

$$\frac{2}{9} = \left(1 + \frac{\sqrt{30}}{18}\right)\left(\frac{3}{7} - \frac{2\sqrt{30}}{35}\right)^4 + \left(1 - \frac{\sqrt{30}}{18}\right)\left(\frac{3}{7} + \frac{2\sqrt{30}}{35}\right)^4 ?$$

Данное равенство неверно, поэтому АСТ = 3.

Решение задачи 2

Воспользуемся следующей системой:

$$\begin{cases} \int_a^b p\,(x)\,x^i dx = \sum_{k=0}^{n} A_k\,x_k^i, \quad i = \overline{0, m} \\ \int_a^b p\,(x)\,x^{m+1} dx \neq \sum_{k=0}^{n} A_k\,x_k^i \end{cases}$$

в этом случае, считаем, что квадратурная формула имеет АСТ = m.

В нашем случае, из квадратурной формулы выписывается следующее:

$$\begin{cases} p(x) = 1 \\ A_0 = \dfrac{5}{18} \\ A_1 = \dfrac{4}{9} \\ A_2 = \dfrac{5}{18} \\ x_0 = \dfrac{1}{2} - \sqrt{\dfrac{3}{5}} \\ x_1 = \dfrac{1}{2} \\ x_2 = \dfrac{1}{2} + \sqrt{\dfrac{3}{5}} \end{cases}$$

Теперь будем постепенно проверять соотношения:

$$i = 0 : \ \frac{5}{18} + \frac{4}{9} + \frac{5}{18} = \int_0^1 dx = 1$$

это равенство истинно.

$$i = 1 : \ \frac{5}{18}\left(\frac{1}{2} - \sqrt{\frac{3}{5}}\right) + \frac{4}{9}\left(\frac{1}{2}\right) + \frac{5}{18}\left(\frac{1}{2} + \sqrt{\frac{3}{5}}\right) = \int_0^1 x dx = \frac{1}{2}$$

это равенство также истинно.

$$i = 2 : \ \frac{5}{18}\left(\frac{1}{2} - \sqrt{\frac{3}{5}}\right)^2 + \frac{4}{9}\left(\frac{1}{2}\right)^2 + \frac{5}{18}\left(\frac{1}{2} + \sqrt{\frac{3}{5}}\right)^2 = \int_0^1 x^2 dx = \frac{1}{3}$$

а это равенство не выполняется, тогда можно сделать вывод, что АСТ = 1.

Решение задачи 3

Правило Рунге.

Пусть имеет место разложение остатка составной квадратурной формулы

$$R(h,f) = ch^m + o\left(h^{m+1}\right), \ c = const.$$

Если точное значение интеграла обозначить как I, а приближенное, соответствующее квадратурной формуле с шагом h - I_h, то получим:

$$\begin{cases} I \approx I_{h_1} + ch_1^m \\ I \approx I_{h_2} + ch_2^m \end{cases}$$

Тогда подставляю в $R(h,f)$, получим:

$$R(h,f) \approx \frac{I_{h_2} - I_{h_1}}{h_1^m - h_2^m} h_1^m = \frac{I_{h_2} - I_{h_1}}{1 - \left(\frac{h_2}{h_1}\right)^m}.$$

Тогда:

$$I \approx I_{h_1} + \frac{I_{h_2} - I_{h_1}}{1 - \left(\frac{h_2}{h_1}\right)^m}.$$

В качестве h_1 возьмем $h_1 = b - a$, тогда $h_2 = \frac{h_1}{2}$.

Получим в итоге следующую функцию на Python:

```
def runge_rule(m, compound_quadrature_formula, f, a, b, epsilon, N_divisions=False):
    h_1 = b-a
    h_2 = h_1 / 2
    iterations = 1
    I_h1 = compound_quadrature_formula(f, a, b, h_1)
    I_h2 = compound_quadrature_formula(f, a, b, h_2)
    R = (I_h2 - I_h1) / (1 - (h_2/h_1)**m)
    I = I_h1 + R
    while abs(R) >= epsilon:
        h_1 = h_2
        h_2 = h_1 / 2
        I_h1 = compound_quadrature_formula(f, a, b, h_1)
        I_h2 = compound_quadrature_formula(f, a, b, h_2)
        R = (I_h2 - I_h1) / (1 - (h_2/h_1)**m)
        I = I_h1 + R
        iterations += 1
    if N_divisions:
        return I, iterations
    return I
```

Составная квадратурная формула левых прямоугольников.

$$I_{\text{лс}}(f) = h\sum_{k=0}^{N-1} f(x_k) = h\sum_{k=0}^{N-1} f(a + kh).$$

Составная квадратурная формула Симпсона.

Составная квадратурная формула Симпсона:

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

$$f_i = f(x_i).$$

Реализуем квадратурные формулы на Python:

```
def left_rectangles(f, a, b, h):
    I = 0
    N = int((b-a)/h)
    for k in range(N):
        I += f(a + k*h)
    I *= h
    return I

def simpson(f, a, b, h):
    I = f(a) + f(b)
    N = int((b-a)/h)
    x_k = a
    for k in range(2,N, 2):
        I += 2*f(a + k * h)
    for k in range(1,N, 2):
        I += 4*f(a + k * h)
    I *= h/3
    return I
```

Определим подынтегральную функцию на Python:

```
def f(x):
    return np.exp(x) / (x + 1)
```

Сравним составные формулы:

```
%%time
runge_rule(1, left_rectangles, f, 1, 2, 1e-3, True)
```
✓ 0.0s Python

```
CPU times: user 7.36 ms, sys: 0 ns, total: 7.36 ms
Wall time: 7.29 ms

(1.831891770210002, 11)
```

```
%%time
runge_rule(2, simpson, f, 1, 2, 1e-3, True)
```
✓ 0.0s Python

```
CPU times: user 31 µs, sys: 20 µs, total: 51 µs
Wall time: 54.4 µs

(1.8318294576260676, 2)
```

Видно что оба метода дали ответ довольно быстро, но формула Симпсона проводила меньшее число делений отрезка.

```
%%time
runge_rule(1, left_rectangles, f, 1, 2, 1e-4, True)
```
✓ 0.0s Python

```
CPU times: user 43.1 ms, sys: 0 ns, total: 43.1 ms
Wall time: 42.5 ms

(1.8318918078564406, 14)
```

```
%%time
runge_rule(2, simpson, f, 1, 2, 1e-4, True)
```
✓ 0.0s Python

```
CPU times: user 68 µs, sys: 0 ns, total: 68 µs
Wall time: 71 µs

(1.8318877676375942, 3)
```

Здесь также формула Симпсона дала ответ быстрее и провела меньше делений отрезка.

```
%%time
runge_rule(1, left_rectangles, f, 1, 2, 1e-5, True)
```

✓ 0.2s Python

```
CPU times: user 317 ms, sys: 0 ns, total: 317 ms
Wall time: 316 ms

(1.8318918084446687, 17)
```

```
%%time
runge_rule(2, simpson, f, 1, 2, 1e-5, True)
```

✓ 0.0s Python

```
CPU times: user 95 µs, sys: 0 ns, total: 95 µs
Wall time: 99.7 µs

(1.8318915533643054, 4)
```

```
%%time
runge_rule(1, left_rectangles, f, 1, 2, 1e-6, True)
```

✓ 4.6s Python

```
CPU times: user 4.69 s, sys: 0 ns, total: 4.69 s
Wall time: 4.69 s

(1.831891808454127, 21)
```

```
%%time
runge_rule(2, simpson, f, 1, 2, 1e-6, True)
```

✓ 0.0s Python

```
CPU times: user 129 µs, sys: 0 ns, total: 129 µs
Wall time: 136 µs

(1.8318917924697014, 5)
```

Тогда можно сделать следующий вывод:

Алгоритм Рунге с использованием составной формулы левых прямоугольников оказался значительно медленнее алгоритма с использованием составной формулы Симпсона.

Решение задачи 4

Для применения формулы Гаусса необходимо использовать линейное преобразование:

$$x = \frac{b-a}{2}t + \frac{a+b}{2}, \quad t \in [-1; 1],$$

которое в нашем случае примет вид:

$$x = 2.5t + 2.5,\ t \in [-1; 1].$$

Тогда имеем:

$$t = 0.4x - 1, \quad x \in [0; 5].$$

Воспользуемся этой заменой, тогда:

$$I = \frac{b-a}{2}\int_{-1}^{1} e^{-6.75t^2 - 10t - 3.75} dt.$$

Квадратурная формула Гаусса:

$$I(f) = \frac{b-a}{2}\int_{-1}^{1} f(x)\,dx \approx \frac{b-a}{2}\sum_{k=0}^{n} A_k f(x_k)$$

Системой многочленов отрогональных по весу $p(x) = 1$ на отрезке [-1; 1] является система многочленов Лежандра. Поэтому узлами x_k в формуле Гаусса, будут являться корни многочлена Лежандра, т.е. $P_{n+1}(x) = 0$. Коэффициенты $A_k = \frac{2}{(1-x_k^2)[P'_{n+1}(x_k)]^2}, \quad k = \overline{0, n}$.

В нашем же случае, квадратурная формула Гаусса примет вид:

$$I(f) \approx 2.5\sum_{k=0}^{n} A_k e^{-6.75x_k^2 - 10x_k - 3.75}.$$

Реализуем на Python:

```
import cmath
import sympy as sp
import numpy as np
import math


epsilon = 1e-4


def f(x):
    return cmath.exp(-6.75*x**2-10*x-3.75)


def P(n):
    y = sp.Symbol('y')
    pol = (1/((2**n)*math.factorial(n)))*sp.diff((y**2-1)**n, y, n)
    return pol
```

```
def subs(x, pol):
    y = sp.Symbol('y')
    return pol.subs(y, x)


def find_roots(pol):
    y = sp.Symbol('y')
    pol=sp.simplify(pol)
    roots = sp.solve(pol, y)
    return roots


def gauss_quadrature_formula(f, epsilon):
    y = sp.Symbol('y')
    n = 1
    I_new = np.inf
    I_last = 0
    while abs(I_new - I_last) >= epsilon:
        I_last = I_new
        I_new = 0
        pol=P(n)
        def_pol=sp.diff(pol, y, 1)
        arr_x=find_roots(pol)
        for i in range(len(arr_x)):
            A_k=2/(1-(arr_x[i])**2)*(subs(arr_x[i], def_pol))**2
            I_new+=A_k*f(arr_x[i])
        I_new*=2.5
        n += 1
    return I_new, n-1
```

Тогда $I \approx 1.730234$, а $n = 7$ узлов.

Решение задачи 5

Обозначим $f(x) = \frac{1}{\sqrt{2\pi}}\int_0^x e^{-t^2}dt - 0.3$. Тогда нужно найти решению нелинейного уравнения $f(x) = 0$.

Отделение корня.

$$f'(x) = \frac{1}{\sqrt{2\pi}}e^{-x^2} > 0,$$

это следует из того, что e^{-x^2} строго положительна на [0; +inf).

Чтобы определить знак функции $f(x)$ в точке, приближенно посчитаем интеграл используя составную квадратурную формулу Симпсона:

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

$$f_i = f\,(x_i)\,.$$

Воспользуемся реализацией, написанной в задании 3.

Вычислим значение интеграла на концах отрезка. Для этого определим функцию на Python, которая будет применять построенный алгоритм Симпсона для вычисления $f(x)$:

```
def g(t):
    return np.e**((-t)**2)


def f(x, epsilon):
    return (1 / np.sqrt(2 * np.pi))*runge_rule(3, simpson, g, 0, x, epsilon) - 0.3
```

Тогда, при $\varepsilon = 10^{-9}$, получим, что $f(2) > 0$, а $f(1) < 0$. Отсюда следует, что на отрезке [1; 2] имеется единственный корень. Уточним отрезок методом дихотомии, в результате чего, получим отрезок [1; 1.125].

Для решения нелинейного уравнение воспользуемся методом Ньютона:

```
def phi(x):
    return x - f(x, epsilon) / (( 1 / np.sqrt(2 * np.pi))*g(x))


x_k = 1
x_k1 = phi(1)
iterations = [[x_k1, np.absolute(x_k1 - x_k)]]
while np.absolute(x_k1 - x_k) >= 1e-4:
    x_k = x_k1
    x_k1 = phi(x_k)
    iterations.append([x_k1, np.absolute(x_k1 - x_k)])


newton_table = pd.DataFrame(iterations, columns=[('Метод Ньютона', 'x_k'), ('Метод Ньютона', '|x_k+1 - x_k|')])
newton_table.columns = pd.MultiIndex.from_tuples(newton_table.columns, names=['', ''])
```

В результате получим:

	Метод Ньютона	
	x_k	|x_k+1 - x_k|
0	0.738563	0.261437
1	0.653932	0.084631
2	0.648431	0.005501
3	0.648411	0.000020

Отсюда видно:

$$x^* \approx 0.648411$$