

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ БЕЛОРУССКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет прикладной математики и информатики
Кафедра вычислительной математики

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 4
"ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ"
ВАРИАНТ 5

Выполнил:
Карпович Артём Дмитриевич
студент 3 курса 7 группы

Преподаватель:
Репников Василий Иванович

Минск, 2024

Численное интегрирование

Постановка задач

1. Построить квадратурную формулу максимально возможной степени точности вида

$$\int_a^b f(x)dx \approx A_0 f(a) + A_1 f(b) + A_2 f'(a) + A_3 f'(b).$$

2. Определить алгебраическую степень точности указанной квадратурной формулы

$$\int_{-1}^1 f(x)dx \approx \frac{1}{6}[f(-1) + f(1)] + \frac{5}{6}[f(-\frac{1}{\sqrt{5}}) + f(\frac{1}{\sqrt{5}})].$$

3. Используя правило Рунге, провести сравнительный анализ квадратурных формул средних прямоугольников и трапеций на примере вычисления интеграла

$$I = \int_1^3 \frac{\ln(\sin^2 x + 3)}{x^2 + 2x - 1} dx.$$

4. Вычислить с точностью $\epsilon = 10^{-4}$ интеграл

$$I = \int_{-1}^1 \frac{1}{\sqrt{(1-x^2)}} \frac{\sin x^2}{1 + \ln^2(x+1)} dx.$$

5. Найти с точностью до $\epsilon = 10^{-4}$ решение уравнения

$$\int_0^X (t-1)^6 (\lg \sqrt{t^2+1} + 2) dt = 5.$$

Задача 1

Для построения квадратурной формы с алгебраической степенью точности m необходимо составить соотношения

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

Из этих соотношений можно составить систему для нахождения коэффициентов A_0, A_1, A_2, A_3

$$\begin{cases} x^0 : \int_a^b 1 dx = A_0 + A_1, \\ x^1 : \int_a^b x dx = A_0 a + A_1 b + A_2 + A_3, \\ x^2 : \int_a^b x^2 dx = A_0 a^2 + A_1 b^2 + 2A_2 a + 2A_3 b, \\ x^3 : \int_a^b x^3 dx = A_0 a^3 + A_1 b^3 + 3A_2 a^2 + 3A_3 b^2. \end{cases}$$

Раскроем интегралы и получим

$$\begin{cases} b - a = A_0 + A_1, \\ \frac{(b-a)^2}{2} = A_0a + A_1b + A_2 + A_3, \\ \frac{(b-a)^3}{3} = A_0a^2 + A_1b^2 + 2A_2a + 2A_3b, \\ \frac{(b-a)^4}{4} = A_0a^3 + A_1b^3 + 3A_2a^2 + 3A_3b^2. \end{cases}$$

Найдем решение системы программно, используя инструменты языка Python

```
[4]: import math
import numpy as np
import sympy as sp
import pandas as pd

a, b, A0, A1, A2, A3 = sp.symbols('a b A0 A1 A2 A3')

eq1 = sp.Eq(b - a, A0 + A1)
eq2 = sp.Eq(((b - a)**2)/2, A0*a + A1*b + A2 + A3)
eq3 = sp.Eq(((b - a)**3)/3, A0*a**2 + A1*b**2 + 2*A2*a + 2*A3*b)
eq4 = sp.Eq(((b - a)**4)/4, A0*a**3 + A1*b**3 + 3*A2*a**2 + 3*A3*b**2)

solution = sp.solve((eq1, eq2, eq3, eq4), (A0, A1, A2, A3))

simplified_solution = {key: sp.simplify(value) for key, value in solution.
    ↪items()}

for key, value in simplified_solution.items():
    print(f"{key}: {value}")
```

```
A0: (-3*a**3 - a**2*b - a*b**2 + b**3)/(2*(a**2 - 2*a*b + b**2))
A1: (a**3 + 7*a**2*b - 5*a*b**2 + b**3)/(2*(a**2 - 2*a*b + b**2))
A2: (7*a**3 + 3*a**2*b + 3*a*b**2 - b**3)/(12*(a - b))
A3: (17*a**3 - 3*a**2*b - 3*a*b**2 + b**3)/(12*(a - b))
```

Получили, что

$$\begin{aligned} A_0 &= \frac{(-3)a^3 - a^2b - ab^2 + b^3}{2(a^2 - 2ab + b^2)}, \\ A_1 &= \frac{a^3 + 7a^2b - 5ab^2 + b^3}{2(a^2 - 2ab + b^2)}, \\ A_2 &= \frac{7a^3 + 3a^2b + 3ab^2 - b^3}{12(a - b)}, \\ A_3 &= \frac{17a^3 - 3a^2b - 3ab^2 + b^3}{12(a - b)}. \end{aligned}$$

Проверим полученное решение с помощью тех же инструментов Python

```
[6]: eq1_sub = sp.simplify(eq1.subs(solution))
eq2_sub = sp.simplify(eq2.subs(solution))
eq3_sub = sp.simplify(eq3.subs(solution))
eq4_sub = sp.simplify(eq4.subs(solution))

print(eq1_sub)
print(eq2_sub)
print(eq3_sub)
print(eq4_sub)
```

True
True
True
True

Таким образом, решение получено правильно и, соответственно, квадратурная форма будет иметь вид:

$$\int_a^b f(x)dx \approx \frac{(-3)a^3 - a^2b - ab^2 + b^3}{2(a^2 - 2ab + b^2)}f(a) + \frac{a^3 + 7a^2b - 5ab^2 + b^3}{2(a^2 - 2ab + b^2)}f(b) + \\ + \frac{7a^3 + 3a^2b + 3ab^2 - b^3}{12(a-b)}f'(a) + \frac{17a^3 - 3a^2b - 3ab^2 + b^3}{12(a-b)}f'(b)$$

Найдем АСТ для получившейся квадратурной формулы, для этого построим соотношение x^4 :

$$\int_a^b x^4 dx = A_0a^4 + A_1b^4 + 4A_2a^3 + 4A_3b^3.$$

Подставим коэффициенты и вычислим интеграл

$$\frac{(b-a)^5}{5} = \frac{(-3)a^3 - a^2b - ab^2 + b^3}{2(a^2 - 2ab + b^2)}a^4 + \frac{a^3 + 7a^2b - 5ab^2 + b^3}{2(a^2 - 2ab + b^2)}b^4 + \\ + 4\frac{7a^3 + 3a^2b + 3ab^2 - b^3}{12(a-b)}a^3 + 4\frac{17a^3 - 3a^2b - 3ab^2 + b^3}{12(a-b)}b^3$$

Посмотрим, выполняется ли равенство При подстановке мы получаем следующее

$$\frac{(a-b)^5}{5} \stackrel{?}{=} -\frac{10a^6 - 12a^5b - 18a^4b^2 + 23a^3b^3 - 27a^2b^4 + 15ab^5 - 3b^6}{12(a-b)}.$$

Раскроем левую часть

$$\frac{(a-b)^5}{5} = \frac{a^5 - 5a^4b + 10a^3b^2 - 10a^2b^3 + 5ab^4 - b^5}{5} \\ \frac{a^5 - 5a^4b + 10a^3b^2 - 10a^2b^3 + 5ab^4 - b^5}{5} \neq -\frac{10a^6 - 12a^5b - 18a^4b^2 + 23a^3b^3 - 27a^2b^4 + 15ab^5 - 3b^6}{12(a-b)}$$

Таким образом, алгебраическая степень точности равна тому i , для которого равенство выполнялось, в нашем случае это значение равно 3.

Задача 2

Рассмотрим общий вид квадратурной формулы

$$I(f) = \int_a^b \rho(x)f(x)dx \approx A_0f(x_0) + A_1f(x_1) + A_2f(x_2) + A_3f(x_3).$$

Для построения квадратурной формы с алгебраической степенью точности m необходимо составить соотношения

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

Таким образом, в нашем примере мы имеем

$$\begin{aligned} [a, b] &= [-1, 1], \quad \rho(x) = 1, \\ A_0 &= A_1 = \frac{1}{6}, \quad A_2 = A_3 = \frac{5}{6}, \\ x_0 &= -1, \quad x_1 = 1, \quad x_2 = -\frac{1}{\sqrt{5}}, \quad x_3 = \frac{1}{\sqrt{5}}. \end{aligned}$$

Для определения алгебраической степени точности, необходимо строить по одному уравнению из нашего соотношения до тех пор, пока равенство не обратится в неравенство.

Найдем решение интеграла для любого i :

$$\int_{-1}^1 x^i dx = \left. \frac{x^{i+1}}{i+1} \right|_{-1}^1 = \frac{1 - (-1)^{i+1}}{i+1}$$

Подставим соотношение для i -го порядка:

$$\frac{1 - (-1)^{i+1}}{i+1} = \frac{1}{6} \cdot (-1)^i + \frac{1}{6} \cdot 1 + \frac{5}{6} \cdot \left(-\frac{1}{\sqrt{5}}\right)^i + \frac{5}{6} \cdot \left(\frac{1}{\sqrt{5}}\right)^i$$

Нетрудно заметить, что при нечетных i и левая, и правая части будут равны 0, поэтому сразу будем рассматривать четные i , при которых получим

$$\frac{2}{i+1} = \frac{1}{3} + \frac{5}{3 \cdot (\sqrt{5})^i}$$

Начнем с $i = 0$:

$$x^0 : 2 \stackrel{?}{=} \frac{1}{6} + \frac{1}{6} + \frac{5}{6} + \frac{5}{6} = 2 \Rightarrow \text{Равенство выполняется.}$$

$$x^2 : \frac{2}{3} \stackrel{?}{=} \frac{1}{3} + \frac{5}{3 \cdot 5} = \frac{2}{3} \Rightarrow \text{Равенство выполняется.}$$

$$x^4 : \frac{2}{5} \stackrel{?}{=} \frac{1}{3} + \frac{5}{3 \cdot 5 \cdot \sqrt{5}} = \frac{1}{3} + \frac{1}{3 \cdot \sqrt{5}} \neq \frac{2}{5} \Rightarrow \text{Равенство не выполняется.}$$

Таким образом, АСТ нашей квадратурной формулы равна 3, поскольку, как ранее было отмечено, при $i = 3$ мы получим равенство $0 = 0$.

Задача 3

Зададим нашу функцию $f(x) = \frac{\ln(\sin^2 x + 3)}{x^2 + 2x - 1}$ программно

```
[11]: def f(x):  
        return math.log(math.sin(x)**2 + 3) / (x**2 + 2*x - 1)  
  
a, b, n = 1, 3, 10
```

Рассмотрим общий вид составной квадратурной формулы средних прямоугольников

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

и зададим его программно.

```
[13]: def mean_rect(a, b, f, h):  
        I = 0  
        N = int((b - a) / h)  
  
        for k in range(N):  
            I += f(a + (k + 1/2) * h)  
  
        return h * I
```

Рассмотрим так же составную квадратурную формулу трапеций

$$I_{tc} = h \left[\frac{f(a) + f(b)}{2} + \sum_{k=0}^{N-1} f(x_k) \right]$$

```
[15]: def trap(a, b, f, h):  
        I = 0  
        N = int((b - a) / h)  
  
        x = np.linspace(a, b, N)  
  
        for k in range(N):  
            I += f(x[k])  
  
        return h * ((f(a) + f(b)) / 2 + I)
```

Для использования правила Рунге используем выражение для главной части остатка квадратурной формулы

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

m — алгебраическая степень точности методов, которая в нашем случае равна 1 для каждого метода.

Подбирать шаги будем следующим образом

$$h_1 = \frac{b-a}{N}, \quad h_2 = \frac{h_1}{2}$$

Посмотрим, для какой квадратурной формулы мы быстрее сможем подобрать такие шаги h_1, h_2 , при которых

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

Погрешность в этом задании возьмем $\epsilon = 10^{-4}$, начальные шаги

$$h_1 = b - a, h_2 = \frac{h_1}{2}$$

подбор будем делать по правилу

$$|R(h, f)| \not\leq \epsilon \Rightarrow h_1 = h_2, h_2 = \frac{h_1}{2}$$

Если же неравенство будет выполняться, то мы подобрали шаг при котором, достигается нужная точность. Зададим правило Рунге программно

```
[17]: def runge_rule(m, a, b, I, f, epsilon = 1e-4):
    h1 = b - a
    h2 = h1 / 2

    R = (I(a, b, f, h1) - I(a, b, f, h2)) / (1 - (h2 / h1)**m)
    array = [R]
    h_1 = [h1]
    h_2 = [h2]

    while abs(R) > epsilon:
        h1 = h2
        h2 = h1 / 2

        R = (I(a, b, f, h1) - I(a, b, f, h2)) / (1 - (h2 / h1)**m)

        array.append(R)

        h_1.append(h1)
        h_2.append(h2)

    return h_1, h_2, array, (I(a, b, f, h1) + R)
```

```
[18]: m = max(len(runge_rule(1, a, b, mean_rect, f)[2]), len(runge_rule(2, a, b, trap, f)[2]))

h1 = runge_rule(1, a, b, trap, f)[0]

df = pd.DataFrame(columns = ["Количество узлов", "Трапеции"])
```

```

N = [0] * m

for i in range(m):
    N[i] = int((b - a) / h1[i])

df["Количество узлов"] = N
df["Трапеции"] = runge_rule(2, a, b, trap, f)[2]

df = pd.concat([df, pd.DataFrame(runge_rule(1, a, b, mean_rect, f)[2],
    ↪columns=["Средние прямоугольники"])], axis=1, verify_integrity=True).fillna('')
df

```

```

[18]:
Количество узлов  Трапеции  Средние прямоугольники
0                1    1.257883          -0.121278
1                2    0.461537          -0.048862
2                4    0.194290          -0.015571
3                8    0.089161          -0.004272
4               16    0.042803          -0.001098
5               32    0.020992          -0.000277
6               64    0.010399          -0.000069
7              128    0.005176
8              256    0.002582
9              512    0.001290
10             1024    0.000644
11             2048    0.000322
12             4096    0.000161
13            8192    0.000081

```

Как можно заметить, с увеличением количества узлов значение остатка по правилу Рунге уменьшается как при использовании составной КФ средних прямоугольников, так и составной КФ трапеций, однако можно увидеть, что для КФ средних прямоугольников мы изначально начинали с меньшего по модулю значения остатка и практически в два раза быстрее нам удалось достигнуть нужной точности.

Задача 4

Для вычисления значения интеграла

$$I = \int_{-1}^1 \frac{1}{\sqrt{(1-x^2)}} \frac{\sin x^2}{1 + \ln^2(x+1)} dx$$

с точностью $\epsilon = 10^{-4}$ воспользуемся квадратурными формулами наивысшей алгебраической степени точности, или квадратурные формулы типа Гаусса.

Поскольку мы сразу можем выделить весовую функцию $p(x) = \frac{1}{\sqrt{(1-x^2)}}$ и мы имеем отрезок интегрирования $[a, b] = [-1, 1]$, то мы сразу можем перейти к использованию квадратурной

формулы Эрмита

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

В качестве узлов x_k выбираются

$$x_k = \cos \frac{2k+1}{2n+2} \pi$$

Для

$$A_k = \frac{\pi}{n+1}$$

Остаток имеет вид

$$R_n(f) = \frac{\pi}{2^{2n+1}(2n+2)!} \cdot f^{2n+2}(\eta), \eta \in [-1, 1].$$

Таким образом, получаем, что КФ имеет вид

$$I(f) = \int_{-1}^1 \frac{1}{\sqrt{1-x^2}} f(x) dx \approx \frac{\pi}{n+1} \sum_{k=0}^n f(\cos \frac{2k+1}{2n+2} \pi)$$

Подставим нашу функцию

$$f(x) = \frac{\sin x^2}{1 + \ln^2(x+1)}$$

и получим

$$I(f) = \int_{-1}^1 \frac{1}{\sqrt{1-x^2}} f(x) dx \approx \frac{\pi}{n+1} \sum_{k=0}^n \frac{\sin(\cos \frac{2k+1}{2n+2} \pi)^2}{1 + \ln^2(\cos \frac{2k+1}{2n+2} \pi + 1)}$$

Погрешность будем оценивать следующим образом

$$\Delta = |I(f, n) - I(f, n-1)| < \epsilon = 10^{-4}$$

Из этого неравенства путем перебора найдем подходящее n .

Перейдем к программной реализации.

```
[21]: def u(x):
      return np.sin(x**2)/(1+np.log(x+1)**2)
```

```
[22]: def hermite_quadrature_formula(f, epsilon=10e-4):
      n = 1

      I_new = np.inf
      I_last = 0

      while abs(I_new - I_last) >= epsilon:
          I_last = I_new
          I_new = 0

          for k in range(n + 1):
              x_k = np.cos((2 * k + 1) / (2 * n + 2) * np.pi)

              A_k = np.pi / (n + 1)
```

```

        I_new += A_k * f(x_k)

    n += 1

    return I_new, n

```

[23]: hermite_quadrature_formula(u)

[23]: (0.6831692972144928, 7)

Таким образом, получаем, что приближенное значение интеграла равно

$$I(f) \approx 0.6832$$

и получено оно было при $n = 7$ узлах.

Задача 5

Перепишем наше уравнение

$$\int_0^x (t-1)^6 (\lg \sqrt{t^2+1} + 2) dt = 5$$

в виде

$$\int_0^x (t-1)^6 (\lg \sqrt{t^2+1} + 2) dt - 5 = f(x).$$

Таким образом, решение этого уравнения совпадает с решением нелинейного уравнения

$$f(x) = 0,$$

которое будем решать методом Ньютона.

Вспомним первую лабораторную работу, уравнение будет иметь единственный корень на отрезке $[a, b]$, если функция на концах отрезка имеет разные по знаку значения и является монотонной.

Начнем с монотонности функции, для этого рассмотрим ее производную, вычисленную по теореме Барроу

$$f'(x) = (x-1)^6 (\lg \sqrt{x^2+1} + 2).$$

Поскольку функции $(x-1)^6$ и $\lg \sqrt{x^2+1} + 2$ являются строго положительными на отрезке $[0, +\infty)$, тогда и

$$f'(x) > 0, \quad x \in [0, +\infty],$$

следовательно, наша исходная функция монотонна на нашем множестве.

Для определения знака на концах отрезка, необходимо приближенно вычислить интеграл

$$\int_0^x (t-1)^6 (\lg \sqrt{t^2+1} + 2) dt.$$

Для этого определим подынтегральную функцию

$$g(x) = (t-1)^6 (\lg \sqrt{t^2+1} + 2).$$

```
[27]: def g(x):  
       return (x - 1)**6 * (np.log(np.sqrt(x**2 + 1)) + 2)
```

Для вычисления интеграла воспользуемся правилом Рунге, а именно приближенное значение будем искать в виде

$$\int_0^x (t-1)^6 (\lg \sqrt{t^2+1} + 2) dt \approx I_h(f) + R(h, f),$$

где $I_h(f)$ - квадратурная формула, $R(h, f)$ - остаток составной квадратурной формулы.

Опираясь на заданное 3, будем использовать составную формулу средних прямоугольников, поскольку этот метод сошелся быстрее в рассматриваемом случае, для $\epsilon = 10^{-4}$.

Возьмем значение $x = 2$

```
[29]: def f(x):  
       return runge_rule(1, 0, x, mean_rect, g)[3] - 5
```

```
[30]: print(f(2))
```

-4.319155993134725

То есть

$$f(2) < 0.$$

Так как мы помним, что наша функция является монотонно возрастающей, то все значения слева будут точно отрицательными.

Для рассмотрения правого конца отрезка возьмем значение из отрезка $(2, +\infty)$, например, $x = 2.5$

```
[32]: print(f(2.5))
```

2.420954617125342

Таким образом,

$$f(2.5) > 0,$$

что, опираясь на монотонность функции гарантирует, что во всех точках находящихся правее, значения функции будут положительны.

Следовательно, мы получили, что на отрезке $[2, 2.5]$ наша функция монотонна и имеет разные по знаку значения на его концах, что говорит о том, что на данном отрезке есть единственный корень.

Сведем наш отрезок к минимум, применив метод деления отрезка пополам

```
[34]: a, b = 2, 2.5
epsilon_2 = 1e-1

dichotomy_table = [[a, b, b - a, f(a), f(b) , (a + b)/2]]
c = 0

while b - a > epsilon_2:
    c = (a + b) / 2

    if f(c) * f(a) >= 0:
        a = c

    else:
        b = c

    dichotomy_table.append([a, b, b - a, f(a), f(b) , (a + b) / 2])

pd.DataFrame(dichotomy_table, columns = ['a', 'b', 'b-a', 'f(a)', 'f(b)', '(a+b)/
↪2'])
```

```
[34]:      a      b      b-a      f(a)      f(b)  (a+b)/2
0  2.000  2.5000  0.5000 -4.319156  2.420955  2.25000
1  2.250  2.5000  0.2500 -2.777547  2.420955  2.37500
2  2.375  2.5000  0.1250 -0.886425  2.420955  2.43750
3  2.375  2.4375  0.0625 -0.886425  0.546916  2.40625
```

Таким образом, получаем, что корень находится на отрезке $[2.375, 2.4375]$.

Вспоминая метод Ньютона, имеем следующую формулу

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}, \quad k = 0, 1, \dots; \quad x_0$$

Так же рассмотрим теорему о сходимости метода Ньютона

Теорема 1 (о сходимости метода Ньютона) Пусть выполняются следующие условия:

1. Функция $f(x)$ определена и дважды непрерывно дифференцируема на отрезке

$$s_0 = [x^0; x^0 + 2h_0], \quad h_0 = -\frac{f(x^0)}{f'(x^0)}.$$

При этом на концах отрезка $f(x)f'(x) \neq 0$.

2. Для начального приближения x^0 выполняется неравенство

$$2|h_0|M \leq |f'(x_0)|, \quad M = \max_{x \in s_0} |f''(x)|.$$

Тогда справедливы следующие утверждения:

1. Внутри отрезка s_0 уравнение $f(x) = 0$ имеет корень x^* и при этом этот корень единственный.
2. Последовательность приближений x^k , $k = 1, 2, \dots$ может быть построена по формуле с заданным приближением x^0 .
3. Последовательность x^k сходится к корню x^* , то есть $x^k \xrightarrow[k \rightarrow \infty]{} x^*$.
4. Скорость сходимости характеризуется неравенством

$$|x^* - x^{k+1}| \leq |x^{k+1} - x^k| \leq \frac{M}{2|f'(x^*)|} \cdot (x^k - x^{k-1})^2, \quad k = 0, 1, 2, \dots \quad (4)$$

Выберем x_0 из нашего отрезка, $x_0 = 2.4$ и перейдем к проверке условий теоремы

```
[36]: x0 = 2.4
      print(f'Начальное приближение: {x0}')

      h0 = -f(x0) / g(x0)
      print(f'h_0: {h0}')

      s0 = np.linspace(x0, x0 + 2 * h0, 1000)
      print('s_0 = [', s0[0], ';', s0[-1], ']')
```

Начальное приближение: 2.4

h_0: 0.016168261001675316

s_0 = [2.4 ; 2.4323365220033506]

Проверяем, чтобы на концах отрезка значение функции и ее производной не обращались в ноль одновременно:

```
[38]: f(s0[0]) * g(s0[0])
```

```
[38]: -8.00691153839716
```

```
[39]: f(s0[-1]) * g(s0[0])
```

```
[39]: 9.194062522206684
```

Функция на концах отрезка не обращается в ноль.

Вычислим вторую производную для функции $f(x)$:

$$f''(x) = 3(x-1)^5 \ln(x^2+1) + \frac{(x-1)^5(12x^2+12) + (x-1)^6x}{x^2+1}$$

```
[41]: def f_second_derivative(x):
      return 3 * (x - 1)**5 * np.log(x**2 + 1) + ((x - 1)**5 * (12 * x**2 + 12) +
      ↪ (x - 1)**6 * x) / (x**2 + 1)
```

Найдем

$$M = \max_{x \in s_0} |f''(x)|$$

и сразу проверим выполнение условия

```
[43]: M = np.max(np.absolute(f_second_derivative(s0)))  
print(f'M: {M}')  
  
2 * np.absolute(h0) * M <= np.absolute(g(x0))
```

M: 110.35791665977399

[43]: True

Таким образом, получаем $M = 110.35$, и необходимое условие выполняется. Что говорит о том, что все условия теоремы выполняются, следовательно, метод Ньютона на отрезке $[2.375, 2.475]$ сойдется.

Перейдем непосредственно к реализации метода Ньютона

```
[45]: def newton_method(x0, epsilon=1e-4, max_iterations=100):  
    x_prev = x0  
    x_next = x_prev - f(x_prev) / g(x_prev)  
    iterations = 1  
  
    x_k = []  
  
    while abs(x_next - x_prev) > epsilon and iterations < max_iterations:  
        x_k.append([x_next, abs(x_next - x_prev)])  
  
        x_prev = x_next  
        x_next = x_prev - f(x_prev) / g(x_prev)  
        iterations += 1  
  
    if iterations == max_iterations:  
        print("Максимальное количество итераций достигнуто!")  
  
    x_k.append([x_next, abs(x_next - x_prev)])  
    return x_k, x_next  
  
x_k, x = newton_method(x0)  
  
df = pd.DataFrame(x_k, columns=['Решение', 'Погрешность'])  
df
```

```
[45]:      Решение  Погрешность  
0  2.416168  1.616826e-02  
1  2.415621  5.471388e-04  
2  2.415620  6.594857e-07
```

Таким образом, мы смогли достичь нужной степени точности за три итерации, и корнем нашего уравнения является значение

$$x \approx 2.415620.$$

Выполним проверку

[73]: `f(2.415620)`

[73]: `-1.1026360226651377e-05`

Полученное решение действительно является решением уравнения при заданной точности $\epsilon = 10^{-4}$.