Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №3
по дисциплине «Вычислительная математика»
Вариант 16

Преподаватель:

Машина Е.А.

Выполнила:

Шайхутдинова Н.В.

P3208

Санкт-Петербург

2024 г.

Цель лабораторной работы

Найти приближенное значение определенного интеграла с требуемой точностью различными численными методами.

Рабочие формулы методов

$$\int_a^b f(x)dx = h\sum_{i=1}^{n} y_{i-1}$$

$$\int_a^b f(x)dx = h\sum_{i=1}^{n} y_i$$

$$\int_a^b f(x)dx = h\sum_{i=1}^{n} f(x_{i-1/2})$$

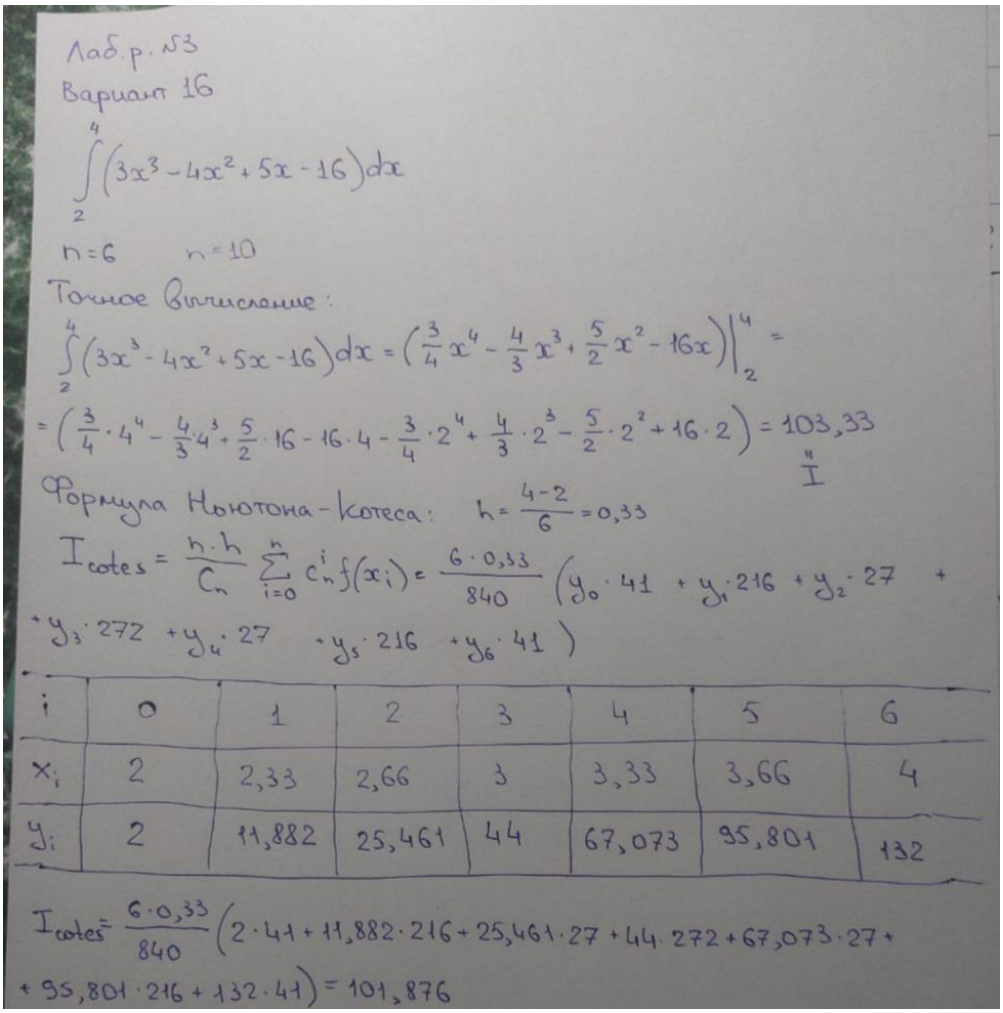
$$\int_a^b f(x)dx = h\cdot\left(\frac{y_0+y_n}{2}+\sum_{i=1}^{n-1} y_i\right)$$

или $$\int_a^b f(x)dx = \frac{h}{2}\cdot\left(y_0+y_n+2\sum_{i=1}^{n-1} y_i\right)$$

$$\int_a^b f(x) = \frac{h}{3}\left[(y_0+4(y_1+y_3+\cdots+y_{n-1})+2(y_2+y_4+\cdots+y_{n-2})+y_n)\right]$$

Порядок выполнения работы

Вычислительная реализация задачи



Лаб.р. №3

Вариант 16

$$\int_2^4 (3x^3 - 4x^2 + 5x - 16)\,dx$$

$n = 6 \qquad n = 10$

Точное вычисление:

$$\int_2^4 (3x^3 - 4x^2 + 5x - 16)\,dx = \left(\frac{3}{4}x^4 - \frac{4}{3}x^3 + \frac{5}{2}x^2 - 16x\right)\Big|_2^4 =$$

$$= \left(\frac{3}{4}\cdot 4^4 - \frac{4}{3}\cdot 4^3 + \frac{5}{2}\cdot 16 - 16\cdot 4 - \frac{3}{4}\cdot 2^4 + \frac{4}{3}\cdot 2^3 - \frac{5}{2}\cdot 2^2 + 16\cdot 2\right) = 103{,}33$$

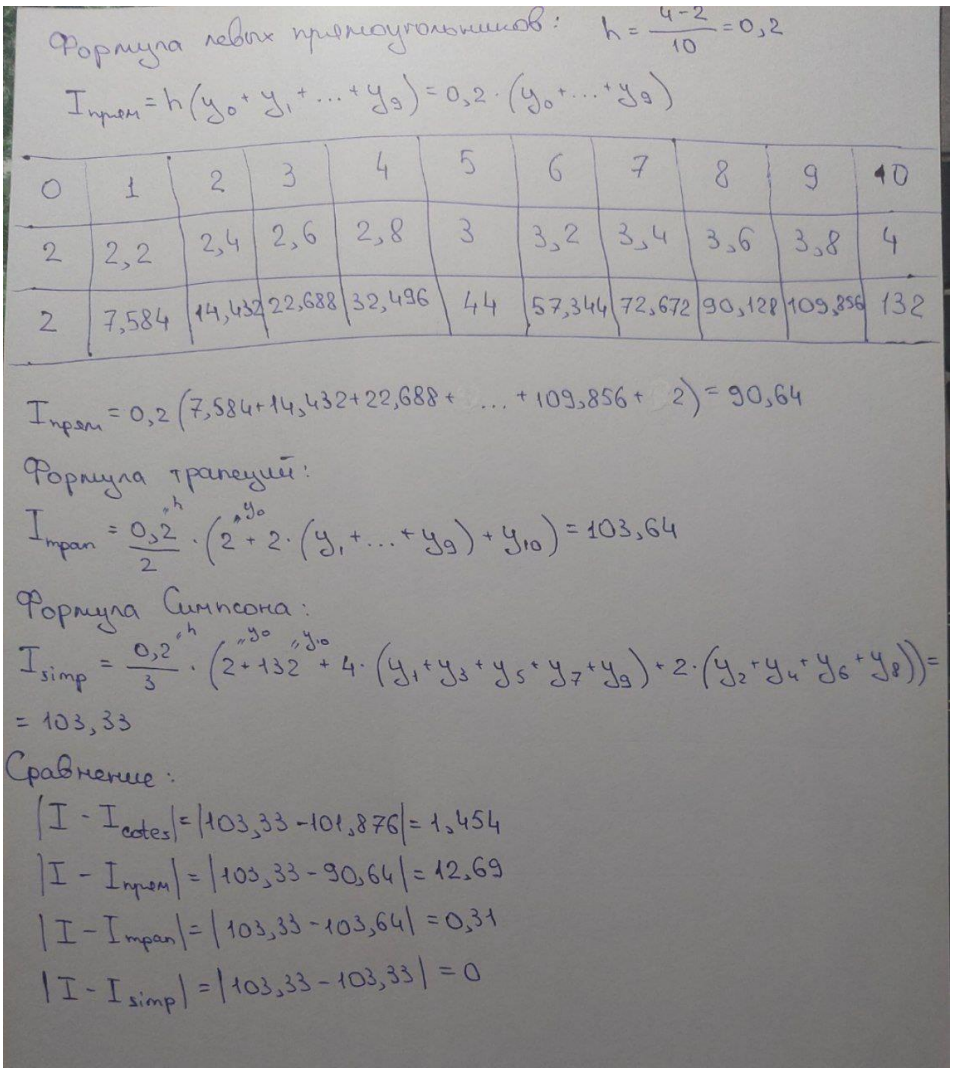
I

Формула Ньютона-Котеса: $h = \frac{4-2}{6} = 0{,}33$

$$I_{cotes} = \frac{n\cdot h}{C_n}\sum_{i=0}^{n} c_n^i f(x_i) = \frac{6\cdot 0{,}33}{840}\left(y_0\cdot 41 + y_1\cdot 216 + y_2\cdot 27 + y_3\cdot 272 + y_4\cdot 27 + y_5\cdot 216 + y_6\cdot 41\right)$$

i	0	1	2	3	4	5	6
x_i	2	2,33	2,66	3	3,33	3,66	4
y_i	2	11,882	25,461	44	67,073	95,801	132

$$I_{cotes} = \frac{6\cdot 0{,}33}{840}\left(2\cdot 41 + 11{,}882\cdot 216 + 25{,}461\cdot 27 + 44\cdot 272 + 67{,}073\cdot 27 + 95{,}801\cdot 216 + 132\cdot 41\right) = 101{,}876$$



Программная реализация задачи

```
def checkConvergence(userChoice, x):
    try:
        y = eval(userChoice[0].replace("x", "(" + str(x) + ")"))
        return True
    except Exception:
        print("Интеграл не существует!")
        userChoice.append(x - userChoice[2])
        userChoice.append(x + userChoice[2])
        return False


def simpsonMethod(userChoice):
    h0 = (userChoice[1][1] - userChoice[1][0]) / userChoice[3]
```

```
n = userChoice[3]
x0 = userChoice[1][0]
x0 += h0
i0 = 0
i0 += eval(userChoice[0].replace("x", "(" + str(userChoice[1][1]) + ")")) + eval(
    userChoice[0].replace("x", "(" + str(userChoice[1][0]) + ")")
)
while x0 < userChoice[1][1]:
    if checkConvergence(userChoice, x0):
        y0 = eval(userChoice[0].replace("x", "(" + str(x0) + ")"))
        i0 += 4 * y0
        x0 += 2 * h0
    else:
        userChoice1 = copy.deepcopy(userChoice)
        userChoice1[1][1] = userChoice[4]
        answer1 = simpsonMethod(userChoice1)
        userChoice2 = copy.deepcopy(userChoice)
        userChoice2[1][0] = userChoice[5]
        answer2 = simpsonMethod(userChoice2)
        answer = []
        answer.append(answer1[0] + answer2[0])
        answer.append(answer1[1] + answer2[1])
        return answer

x0 = userChoice[1][0] + 2 * h0
while x0 < userChoice[1][1]:
    if checkConvergence(userChoice, x0):
        y0 = eval(userChoice[0].replace("x", "(" + str(x0) + ")"))
        i0 += 2 * y0
        x0 += 2 * h0
    else:
        userChoice1 = copy.deepcopy(userChoice)
```

```
        userChoice1[1][1] = userChoice[4]
        answer1 = simpsonMethod(userChoice1)
        userChoice2 = copy.deepcopy(userChoice)
        userChoice2[1][0] = userChoice[5]
        answer2 = simpsonMethod(userChoice2)
        answer = []
        answer.append(answer1[0] + answer2[0])
        answer.append(answer1[1] + answer2[1])
        return answer


i0 *= h0 / 3
y0 = i0
print("Значение интеграла ->", y0)
print("Число разбиения интеграла ->", n)
n *= 2
h1 = (userChoice[1][1] - userChoice[1][0]) / n
x1 = userChoice[1][0]
x1 += h1
i1 = 0
i1 += eval(userChoice[0].replace("x", "(" + str(userChoice[1][1]) + ")")) + eval(
    userChoice[0].replace("x", "(" + str(userChoice[1][0]) + ")")
)
while x1 < userChoice[1][1]:
    if checkConvergence(userChoice, x1):
        y1 = eval(userChoice[0].replace("x", "(" + str(x1) + ")"))
        i1 += 4 * y1
        x1 += 2 * h1
    else:
        userChoice1 = copy.deepcopy(userChoice)
        userChoice1[1][1] = userChoice[4]
        answer1 = simpsonMethod(userChoice1)
        userChoice2 = copy.deepcopy(userChoice)
```

```
        userChoice2[1][0] = userChoice[5]
        answer2 = simpsonMethod(userChoice2)
        answer = []
        answer.append(answer1[0] + answer2[0])
        answer.append(answer1[1] + answer2[1])
        return answer

x1 = userChoice[1][0] + 2 * h1
while x1 < userChoice[1][1]:
    if checkConvergence(userChoice, x1):
        y1 = eval(userChoice[0].replace("x", "(" + str(x1) + ")"))
        i1 += 2 * y1
        x1 += 2 * h1
    else:
        userChoice1 = copy.deepcopy(userChoice)
        userChoice1[1][1] = userChoice[4]
        answer1 = simpsonMethod(userChoice1)
        userChoice2 = copy.deepcopy(userChoice)
        userChoice2[1][0] = userChoice[5]
        answer2 = simpsonMethod(userChoice2)
        answer = []
        answer.append(answer1[0] + answer2[0])
        answer.append(answer1[1] + answer2[1])
        return answer
i1 *= h1 / 3
y1 = i1
print("Значение интеграла ->", y1)
print("Число разбиения интеграла ->", n)
di = (y0 - y1) / (2**4 - 1)

while abs(di) >= userChoice[2]:
    n *= 2
```

```
h = (userChoice[1][1] - userChoice[1][0]) / n
x = userChoice[1][0]
x += h
i = 0
i += eval(userChoice[0].replace("x", "(" + str(userChoice[1][1]) + ")")) + eval(
    userChoice[0].replace("x", "(" + str(userChoice[1][0]) + ")")
)
while x < userChoice[1][1]:
    if checkConvergence(userChoice, x):
        y = eval(userChoice[0].replace("x", "(" + str(x) + ")"))
        i += 4 * y
        x += 2 * h
    else:
        userChoice1 = copy.deepcopy(userChoice)
        userChoice1[1][1] = userChoice[4]
        answer1 = simpsonMethod(userChoice1)
        userChoice2 = copy.deepcopy(userChoice)
        userChoice2[1][0] = userChoice[5]
        answer2 = simpsonMethod(userChoice2)
        answer = []
        answer.append(answer1[0] + answer2[0])
        answer.append(answer1[1] + answer2[1])
        return answer

x = userChoice[1][0] + 2 * h
while x < userChoice[1][1]:
    if checkConvergence(userChoice, x):
        y = eval(userChoice[0].replace("x", "(" + str(x) + ")"))
        i += 2 * y
        x += 2 * h
    else:
        userChoice1 = copy.deepcopy(userChoice)
```

```
            userChoice1[1][1] = userChoice[4]
            answer1 = simpsonMethod(userChoice1)
            userChoice2 = copy.deepcopy(userChoice)
            userChoice2[1][0] = userChoice[5]
            answer2 = simpsonMethod(userChoice2)
            answer = []
            answer.append(answer1[0] + answer2[0])
            answer.append(answer1[1] + answer2[1])
            return answer
        i *= h / 3
        y = i
        print("Значение интеграла ->", y)
        print("Число разбиения интеграла ->", n)
        di = (y1 - y) / (2**4 - 1)
        y1 = y
    answer = []
    answer.append(y1)
    answer.append(n)
    return answer


def trapezoidMethod(userChoice):
    h0 = (userChoice[1][1] - userChoice[1][0]) / userChoice[3]
    n = userChoice[3]
    x0 = userChoice[1][0]
    x0 += h0
    i0 = 0
    while x0 < userChoice[1][1]:
        if checkConvergence(userChoice, x0):
            y0 = eval(userChoice[0].replace("x", "(" + str(x0) + ")"))
            i0 += y0
            x0 += h0
```

```
    else:
        userChoice1 = copy.deepcopy(userChoice)
        userChoice1[1][1] = userChoice[4]
        answer1 = trapezoidMethod(userChoice1)
        userChoice2 = copy.deepcopy(userChoice)
        userChoice2[1][0] = userChoice[5]
        answer2 = trapezoidMethod(userChoice2)
        answer = []
        answer.append(answer1[0] + answer2[0])
        answer.append(max(answer1[1], answer2[1]))
        return answer
i0 += (
    eval(userChoice[0].replace("x", "(" + str(userChoice[1][1]) + ")"))
    + eval(userChoice[0].replace("x", "(" + str(userChoice[1][0]) + ")"))
) / 2
i0 *= h0
y0 = i0
print("Значение интеграла ->", y0)
print("Число разбиения интеграла ->", n)
n *= 2
h1 = (userChoice[1][1] - userChoice[1][0]) / n
x1 = userChoice[1][0] + h1
i1 = 0
while x1 < userChoice[1][1]:
    if checkConvergence(userChoice, x1):
        y1 = eval(userChoice[0].replace("x", "(" + str(x1) + ")"))
        i1 += y1
        x1 += h1
    else:
        userChoice1 = copy.deepcopy(userChoice)
        userChoice1[1][1] = userChoice[4]
        answer1 = trapezoidMethod(userChoice1)
```

```
        userChoice2 = copy.deepcopy(userChoice)
        userChoice2[1][0] = userChoice[5]
        answer2 = trapezoidMethod(userChoice2)
        answer = []
        answer.append(answer1[0] + answer2[0])
        answer.append(max(answer1[1], answer2[1]))
        return answer
i1 += (
    eval(userChoice[0].replace("x", "(" + str(userChoice[1][1]) + ")"))
    + eval(userChoice[0].replace("x", "(" + str(userChoice[1][0]) + ")"))
) / 2
i1 *= h1
y1 = i1
di = (y0 - y1) / (2**2 - 1)
print("Значение интеграла ->", y1)
print("Число разбиения интеграла ->", n)
while abs(di) >= userChoice[2]:
    n *= 2
    h = (userChoice[1][1] - userChoice[1][0]) / n
    x = userChoice[1][0] + h
    i = 0
    while x < userChoice[1][1]:
        if checkConvergence(userChoice, x):
            y = eval(userChoice[0].replace("x", "(" + str(x) + ")"))
            i += y
            x += h
        else:
            userChoice1 = copy.deepcopy(userChoice)
            userChoice1[1][1] = userChoice[4]
            answer1 = trapezoidMethod(userChoice1)
            userChoice2 = copy.deepcopy(userChoice)
            userChoice2[1][0] = userChoice[5]
```

```
            answer2 = trapezoidMethod(userChoice2)
            answer = []
            answer.append(answer1[0] + answer2[0])
            answer.append(max(answer1[1], answer2[1]))
            return answer
        i += (
            eval(userChoice[0].replace("x", "(" + str(userChoice[1][1]) + ")"))
            + eval(userChoice[0].replace("x", "(" + str(userChoice[1][0]) + ")"))
        ) / 2
        i *= h
        y = i
        print("Значение интеграла ->", y)
        print("Число разбиения интеграла ->", n)
        di = (y1 - y) / (2**2 - 1)
        y1 = y
    answer = []
    answer.append(y1)
    answer.append(n)
    return answer


def mediumRectanglesMethod(userChoice):
    h0 = (userChoice[1][1] - userChoice[1][0]) / userChoice[3]
    n = userChoice[3]
    x0 = userChoice[1][0] + h0 / 2
    i0 = 0
    while x0 < userChoice[1][1]:
        if checkConvergence(userChoice, x0):
            y0 = eval(userChoice[0].replace("x", "(" + str(x0) + ")"))
            i0 += y0
            x0 += h0
        else:
```

```
        userChoice1 = copy.deepcopy(userChoice)
        userChoice1[1][1] = userChoice[4]
        answer1 = mediumRectanglesMethod(userChoice1)
        userChoice2 = copy.deepcopy(userChoice)
        userChoice2[1][0] = userChoice[5]
        answer2 = mediumRectanglesMethod(userChoice2)
        answer = []
        answer.append(answer1[0] + answer2[0])
        answer.append(max(answer1[1], answer2[1]))
        return answer
i0 *= h0
y0 = i0
print("Значение интеграла ->", y0)
print("Число разбиения интеграла ->", n)
n *= 2
h1 = (userChoice[1][1] - userChoice[1][0]) / n
x1 = userChoice[1][0] + h1 / 2
i1 = 0
while x1 < userChoice[1][1]:
    if checkConvergence(userChoice, x1):
        y1 = eval(userChoice[0].replace("x", "(" + str(x1) + ")"))
        i1 += y1
        x1 += h1
    else:
        userChoice1 = copy.deepcopy(userChoice)
        userChoice1[1][1] = userChoice[4]
        answer1 = mediumRectanglesMethod(userChoice1)
        userChoice2 = copy.deepcopy(userChoice)
        userChoice2[1][0] = userChoice[5]
        answer2 = mediumRectanglesMethod(userChoice2)
        answer = []
        answer.append(answer1[0] + answer2[0])
```

```
        answer.append(max(answer1[1], answer2[1]))
        return answer
i1 *= h1
y1 = i1
di = (y0 - y1) / (2**2 - 1)
print("Значение интеграла ->", y1)
print("Число разбиения интеграла ->", n)
while abs(di) >= userChoice[2]:
    n *= 2
    h = (userChoice[1][1] - userChoice[1][0]) / n
    x = userChoice[1][0] + h / 2
    i = 0
    while x < userChoice[1][1]:
        if checkConvergence(userChoice, x):
            y = eval(userChoice[0].replace("x", "(" + str(x) + ")"))
            i += y
            x += h
        else:
            userChoice1 = copy.deepcopy(userChoice)
            userChoice1[1][1] = userChoice[4]
            answer1 = mediumRectanglesMethod(userChoice1)
            userChoice2 = copy.deepcopy(userChoice)
            userChoice2[1][0] = userChoice[5]
            answer2 = mediumRectanglesMethod(userChoice2)
            answer = []
            answer.append(answer1[0] + answer2[0])
            answer.append(max(answer1[1], answer2[1]))
            return answer
    i *= h
    y = i
    print("Значение интеграла ->", y)
    print("Число разбиения интеграла ->", n)
```

```
        di = (y1 - y) / (2**2 - 1)
        y1 = y
    answer = []
    answer.append(y1)
    answer.append(n)
    return answer


def leftRectanglesMethod(userChoice):
    h0 = (userChoice[1][1] - userChoice[1][0]) / userChoice[3]
    n = userChoice[3]
    x0 = userChoice[1][0]
    i0 = 0
    while x0 < userChoice[1][1]:
        if checkConvergence(userChoice, x0):
            y0 = eval(userChoice[0].replace("x", "(" + str(x0) + ")"))
            i0 += y0
            x0 += h0
        else:
            userChoice1 = copy.deepcopy(userChoice)
            userChoice1[1][1] = userChoice[4]
            answer1 = leftRectanglesMethod(userChoice1)
            userChoice2 = copy.deepcopy(userChoice)
            userChoice2[1][0] = userChoice[5]
            answer2 = leftRectanglesMethod(userChoice2)
            answer = []
            answer.append(answer1[0] + answer2[0])
            answer.append(max(answer1[1], answer2[1]))
            return answer
    i0 *= h0
    y0 = i0
    print("Значение интеграла ->", y0)
```

```
print("Число разбиения интеграла ->", n)
n *= 2
h1 = (userChoice[1][1] - userChoice[1][0]) / n
x1 = userChoice[1][0]
i1 = 0
while x1 < userChoice[1][1]:
    if checkConvergence(userChoice, x1):
        y1 = eval(userChoice[0].replace("x", "(" + str(x1) + ")"))
        i1 += y1
        x1 += h1
    else:
        userChoice1 = copy.deepcopy(userChoice)
        userChoice1[1][1] = userChoice[4]
        answer1 = leftRectanglesMethod(userChoice1)
        userChoice2 = copy.deepcopy(userChoice)
        userChoice2[1][0] = userChoice[5]
        answer2 = leftRectanglesMethod(userChoice2)
        answer = []
        answer.append(answer1[0] + answer2[0])
        answer.append(max(answer1[1], answer2[1]))
        return answer

i1 *= h1
y1 = i1
di = (y0 - y1) / (2**2 - 1)
print("Значение интеграла ->", y1)
print("Число разбиения интеграла ->", n)
while abs(di) >= userChoice[2]:
    n *= 2
    h = (userChoice[1][1] - userChoice[1][0]) / n
    x = userChoice[1][0]
    i = 0
```

```
        while x < userChoice[1][1]:
            if checkConvergence(userChoice, x):
                y = eval(userChoice[0].replace("x", "(" + str(x) + ")"))
                i += y
                x += h
            else:
                userChoice1 = copy.deepcopy(userChoice)
                userChoice1[1][1] = userChoice[4]
                answer1 = leftRectanglesMethod(userChoice1)
                userChoice2 = copy.deepcopy(userChoice)
                userChoice2[1][0] = userChoice[5]
                answer2 = leftRectanglesMethod(userChoice2)
                answer = []
                answer.append(answer1[0] + answer2[0])
                answer.append(max(answer1[1], answer2[1]))
                return answer

        i *= h
        y = i
        print("Значение интеграла ->", y)
        print("Число разбиения интеграла ->", n)
        di = (y1 - y) / (2**2 - 1)
        y1 = y
    answer = []
    answer.append(y1)
    answer.append(n)
    return answer


def rightRectanglesMethod(userChoice):
    h0 = (userChoice[1][1] - userChoice[1][0]) / userChoice[3]
    n = userChoice[3]
```

```
x0 = userChoice[1][0]
x0 += h0
i0 = 0
while x0 <= userChoice[1][1]:
    if checkConvergence(userChoice, x0):
        y0 = eval(userChoice[0].replace("x", "(" + str(x0) + ")"))
        i0 += y0
        x0 += h0
    else:
        userChoice1 = copy.deepcopy(userChoice)
        userChoice1[1][1] = userChoice[4]
        answer1 = rightRectanglesMethod(userChoice1)
        userChoice2 = copy.deepcopy(userChoice)
        userChoice2[1][0] = userChoice[5]
        answer2 = rightRectanglesMethod(userChoice2)
        answer = []
        answer.append(answer1[0] + answer2[0])
        answer.append(max(answer1[1], answer2[1]))
        return answer

i0 *= h0
y0 = i0
print("Значение интеграла ->", y0)
print("Число разбиения интеграла ->", n)
n *= 2
h1 = (userChoice[1][1] - userChoice[1][0]) / n
x1 = userChoice[1][0] + h1
i1 = 0
while x1 <= userChoice[1][1]:
    if checkConvergence(userChoice, x1):
        y1 = eval(userChoice[0].replace("x", "(" + str(x1) + ")"))
        i1 += y1
```

```
            x1 += h1
        else:
            userChoice1 = copy.deepcopy(userChoice)
            userChoice1[1][1] = userChoice[4]
            answer1 = rightRectanglesMethod(userChoice1)
            userChoice2 = copy.deepcopy(userChoice)
            userChoice2[1][0] = userChoice[5]
            answer2 = rightRectanglesMethod(userChoice2)
            answer = []
            answer.append(answer1[0] + answer2[0])
            answer.append(max(answer1[1], answer2[1]))
            return answer

i1 *= h1
y1 = i1
di = (y0 - y1) / (2**2 - 1)
print("Значение интеграла ->", y1)
print("Число разбиения интеграла ->", n)
while abs(di) >= userChoice[2]:
    n *= 2
    h = (userChoice[1][1] - userChoice[1][0]) / n
    x = userChoice[1][0] + h
    i = 0
    while x <= userChoice[1][1]:
        if checkConvergence(userChoice, x):
            y = eval(userChoice[0].replace("x", "(" + str(x) + ")"))
            i += y
            x += h
        else:
            userChoice1 = copy.deepcopy(userChoice)
            userChoice1[1][1] = userChoice[4]
            answer1 = rightRectanglesMethod(userChoice1)
```

```
            userChoice2 = copy.deepcopy(userChoice)
            userChoice2[1][0] = userChoice[5]
            answer2 = rightRectanglesMethod(userChoice2)
            answer = []
            answer.append(answer1[0] + answer2[0])
            answer.append(max(answer1[1], answer2[1]))
            return answer
        i *= h
        y = i
        print("Значение интеграла ->", y)
        print("Число разбиения интеграла ->", n)
        di = (y1 - y) / (2**2 - 1)
        y1 = y
    answer = []
    answer.append(y1)
    answer.append(n)
    return answer
```

Результат выполнения работы программы

Выберите функцию, интеграл которой нужно вычислить:

"1" -> 5.74x^3-2.95x^2-10.28x+4.23

"2" -> x^3-x+4

"3" -> 1/x:

1

Введите через пробел пределы интегрирования:

1 2

Введите погрешность вычислений:

0.01

Выберите метод:

"1" -> метод левых прямоугольников

"2" -> метод правых прямоугольников

"3" -> метод средних прямоугольников

"4" -> метод трапеций

"5" -> метод Симпсона:

5

Значение интеграла -> 3.4516666666666698

Число разбиения интеграла -> 4

Значение интеграла -> 3.451666666666669

Число разбиения интеграла -> 8

Ответ:

Значение интеграла -> 3.451666666666669

Число разбиения интеграла -> 8

Вывод

В ходе выполнения лабораторной работы я изучила различные численные методы и с помощью них нашла значение определенного интеграла с требуемой точностью.