Отчет по Лабораторной работе №3
по курсу “Вычислительная математика”

Вариант №4

Выполнил:
Студент группы P32082
Дробыш Дмитрий Александрович

Преподаватель:
Машина Екатерина Алексеевна

СКА Санкт-Петербург, 2023

Лабораторная работа №3. Численное интегрирование

Цель работы: найти приближенное значение определенного интеграла с требуемой точностью различными численными методами.

№ варианта определяется как номер в списке группы согласно ИСУ.

Лабораторная работа состоит из двух частей: вычислительной и программной.

Обязательное задание (до 80 баллов)

Исходные данные:

1. Пользователь выбирает функцию, интеграл которой требуется вычислить (3-5 функций), из тех, которые предлагает программа.
2. Пределы интегрирования задаются пользователем.
3. Точность вычисления задается пользователем.
4. Начальное значение числа разбиения интервала интегрирования: n=4.
5. Ввод исходных данных осуществляется с клавиатуры.

Программная реализация задачи:

1. Реализовать в программе методы по выбору пользователя:
   - Метод прямоугольников (3 модификации: левые, правые, средние)
   - Метод трапеций
   - Метод Симпсона
2. Методы должны быть оформлены в виде отдельной(ого) функции/класса.
3. Вычисление значений функции оформить в виде отдельной(ого) функции/класса.
4. Для оценки погрешности и завершения вычислительного процесса использовать правило Рунге.
5. Предусмотреть вывод результатов: значение интеграла, число разбиения интервала интегрирования для достижения требуемой точности.

Вычислительная реализация задачи:

1. Вычислить интеграл, приведенный в таблице 1, точно.
2. Вычислить интеграл по формуле Ньютона – Котеса при $n = 5$.
3. Вычислить интеграл по формулам средних прямоугольников, трапеций и Симпсона при $n = 10$.
4. Сравнить результаты с точным значением интеграла.
5. Определить относительную погрешность вычислений для каждого метода.
6. В отчете ***отразить последовательные вычисления***.

Необязательное задание (до 20 баллов)

1. Установить сходимость рассматриваемых несобственных интегралов 2 рода (2-3 функции). Если интеграл - расходящийся, выводить сообщение: «Интеграл не существует».
2. Если интеграл сходящийся, реализовать в программе вычисление несобственных интегралов 2 рода (заданными численными методами).
3. Рассмотреть случаи, когда подынтегральная функция терпит бесконечный разрыв: 1) в точке a, 2) в точке b, 3) на отрезке интегрирования

1. Метод прямоугольников.

На каждом шаге интегрирования функция аппроксимируется полиномом нулевой степени – отрезком, параллельным оси абсцисс. Площадь криволинейной трапеции приближенно заменяется площадью многоугольника, составленного из n-прямоугольников. Таким образом, вычисление определенного интеграла сводится к нахождению суммы n- элементарных прямоугольников.

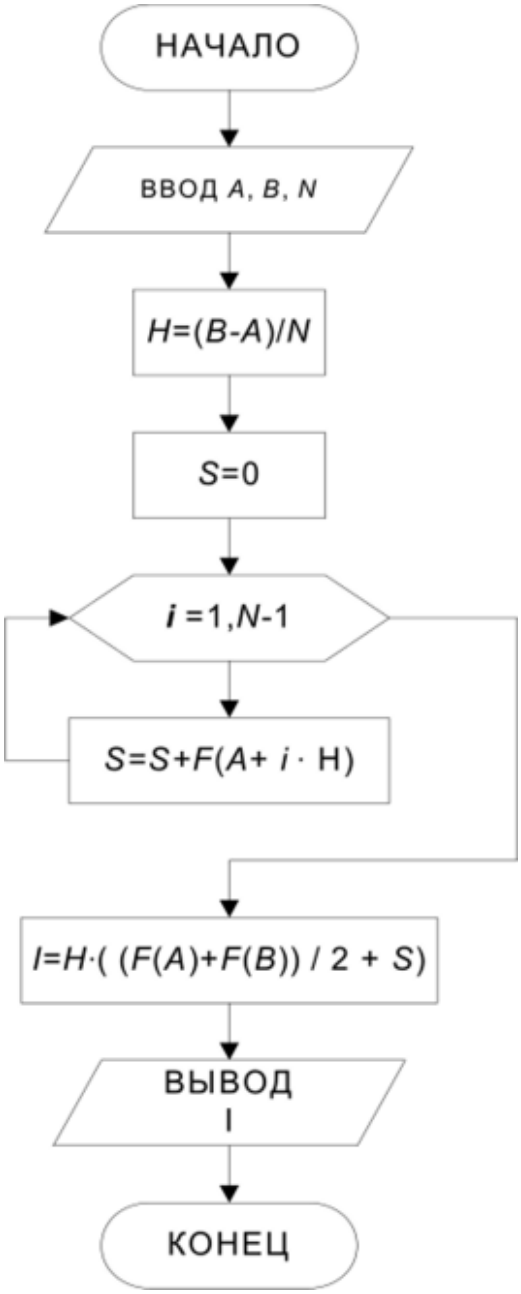
$$\int_a^b f(x)dx = h\sum_{i=1}^{n} y_{i-1}$$

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

```python
def integrate_cell_method(equation, a, b, n, side):
    h = (b - a) / n
    answer = 0

    if side == "center":
        a += h / 2
    elif side == "right":
        a += h

    while a < b and n >= 1:
        answer += function(a, equation)
        a += h
        n -= 1
    answer *= h
    return answer
```



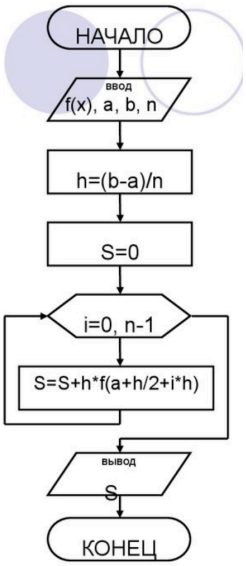
Примеры работы

Enter coefficient of your equation in one line like that: 1 2 3 5 6...
You should type them in standard order: x^(n) x^(n-1) ... x^0
1 0 0
Enter interval as 2 numbers like this: 'a b'
1 2
Enter number of partitions
5
choose: 'left', 'right', 'center'
left
2.04
Runge rule says: I - I_{h/2} = 0.04833333333333378

Enter coefficient of your equation in one line like that: 1 2 3 5 6...
You should type them in standard order: x^(n) x^(n-1) ... x^0
1 0 0
Enter interval as 2 numbers like this: 'a b'
1 2
Enter number of partitions
5
choose: 'left', 'right', 'center'
center
2.3299999999999996
Runge rule says: I - I_{h/2} = 0.0008333333333339077

Enter coefficient of your equation in one line like that: 1 2 3 5 6...
You should type them in standard order: x^(n) x^(n-1) ... x^0
1 0 0
Enter interval as 2 numbers like this: 'a b'
1 2
Enter number of partitions
5
choose: 'left', 'right', 'center'
right
2.64
Runge rule says: I - I_{h/2} = 0.1849999999999996

2. Метод трапеций.



Подынтегральную функцию на каждом отрезке [xi; xi+1] заменяют интерполяционным многочленом первой степени: $fx \approx \varphi i$ (x) =$aix+b$

При $h_i = h = \frac{b-a}{n} = const$ формула трапеций:

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

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

```
def integrate_trapeze_method(equation, a, b, n):
    h = (b - a) / n
    answer = (function(a, equation) + function(b, equation)) / 2
    a += h
    n -= 1

    while a < b and n >= 1:
        tmp = function(a, equation)
        answer += tmp
        a += h
        n -= 1
    answer *= h

    return answer
```

Примеры работы:

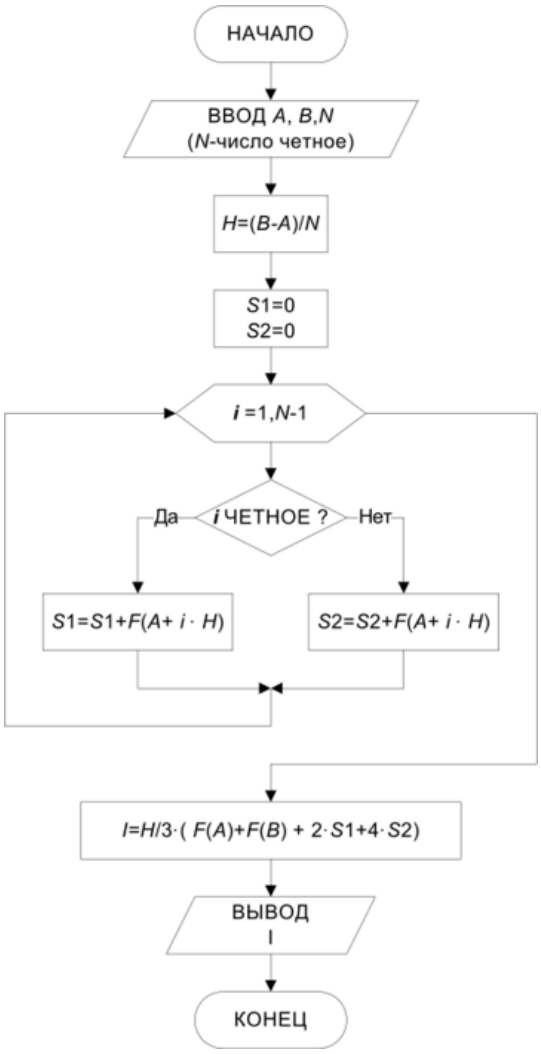
Enter coefficient of your equation in one line like that: 1 2 3 5 6...
You should type them in standard order: x^(n) x^(n-1) ... x^0
1 0 0
Enter interval as 2 numbers like this: 'a b'
1 2
Enter number of partitions
10
2.3350000000000013
Runge rule says: I - I_{h/2} = 0.0004166666666667318

3. Метод Симпсона.

Разобьем отрезок интегрирования [a, b] на четное число n равных частей с шагом h. На каждом отрезке [x0, x2], [x2, x4], ..., [xi-1, xi+1], ... , [xn-2, xn] подынтегральную функцию заменим интерполяционным многочленом второй степени:
$f x \approx \varphi i\ x = aix2+bix+ci,\ xi-1 \le x \le xi+1$

Формула Симпсона

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




```python
def integrate_simpson_method(equation, a, b, n):
    if n % 2:
        n += 1
    h = (b - a) / n

    answer = function(a, equation) + function(b, equation)
    a += h

    for i in range(1, n):
        if not (i % 2):
            tmp = function(a, equation)
            answer += 2 * tmp
        elif i % 2:
            tmp = function(a, equation)
            answer += 4 * tmp
        a += h
    return (answer * h) / 3
```

Примеры работы:

```
Enter coefficient of your equation in one line like that: 1 2 3 5 6...
You should type them in standard order: x^(n) x^(n-1) ... x^0
1 0 0
Enter interval as 2 numbers like this: 'a b'
1 2
Enter number of partitions
5
2.333333333333334
Runge rule says: I - I_{h/2} =  0.0
```

Проверка

Непосредственное использование оценки погрешности неудобно, т.к. требует вычисление производной функции $f\ x$, особенно для подынтегральных функций сложного вида. В вычислительной практике используется правило Рунге.
Правило Рунге - это эмпирический способ оценки погрешности, основанный на сравнении результатов вычислений, проводимых с разными шагами h:

$$I - I_{h/2} = \frac{I_{h/2} - I_h}{2^k - 1}$$

```python
def runge_rule(i_h, i_h_div_2, k):
    d = (abs(i_h_div_2-i_h))/(2**k - 1)
    print("Runge rule says: I - I_{h/2} = ", d)
    return
```

Вычислительная реализация.

Моему варианту соответствует

$$\int_{-3}^{-1} -2x^3 - 4x^2 + 8x - 4 = -\frac{x^4}{2} - \frac{4x^3}{3} + 4x^2 - 4x\,|_{-3}^{-1} = -\frac{104}{3} = -34.667$$

Ньютона-Котеса (n=5)

Для семейства методов Ньютона-Котеса можно записать общее выражение:

$$\int_a^b f(x)dx \approx \sum_{j=1}^{N} c_j f(x_j) = \frac{n \cdot h}{C_n} \sum_{j=1}^{N} \sum_{i=0}^{n} c_{in} f(x_i) \qquad (*)$$

n	C_n	C_{0n}	C_{1n}	C_{2n}	C_{3n}	C_{4n}	C_{5n}
0	1	1					
1	2	1	1				
2	6	1	4	1			
3	8	1	3	3	1		
4	90	7	32	12	32	7	
5	288	19	75	50	50	75	19

$$h = \frac{-1+3}{5} = \frac{2}{5} = 0.4$$

$$C_n = \sum_{i=0}^{n} c_{in}$$

$$\int_a^b f(x)\,dx = \sum_{j=0}^{N} c_j f(x_j)$$

j	x_j	y_j	$c_j \cdot f(x_j)$	h	a	b
0	-3	-10	-190	0,4	-3	-1
1	-2,6	-16,688	-1251,6			
2	-2,2	-19,664	-983,2			
3	-1,8	-19,696	-984,8			
4	-1,4	-17,552	-1316,4			
5	-1	-14	-266			
6	Ответ		-34,6666666666667			

По формулам средних прямоугольников, трапеций и Симпсона при n=10:

По формулам средних прямоугольников

i	x_i	y_i	h
0	-2,9	-12,062	0,2
1	-2,7	-15,394	
2	-2,5	-17,75	
3	-2,3	-19,226	
4	-2,1	-19,918	
5	-1,9	-19,922	
6	-1,7	-19,334	
7	-1,5	-18,25	
8	-1,3	-16,766	
9	-1,1	-14,978	
6	Ответ	-34,72	
	Delta	0,0533333333333	
Относительная	-0,00153846153846058		

По формулам трапеций

i	x_i	y_i	h
0	-3	-10	0,2
1	-2,8	-13,856	
2	-2,6	-16,688	
3	-2,4	-18,592	
4	-2,2	-19,664	
5	-2	-20	
6	-1,8	-19,696	
7	-1,6	-18,848	
8	-1,4	-17,552	
9	-1,2	-15,904	

i	x_i	y_i	h
10	-1	-14	
	Ответ		-34,56
	Delta		-0,1066666666667
Относительная	0,00307692307692404		

По формулам трапеций-1

i	x_i	y_i	h
0	-3	-10	0,2
1	-2,8	-13,856	
2	-2,6	-16,688	
3	-2,4	-18,592	
4	-2,2	-19,664	
5	-2	-20	
6	-1,8	-19,696	
7	-1,6	-18,848	
8	-1,4	-17,552	
9	-1,2	-15,904	
10	-1	-14	
	Ответ		-34,6666666666667
	Delta		0
Относительная	0		

Доп.

В этом случае я использовал библиотеки, к сожалению, тк иначе совсем impossible. Сначала делаю проверку на наличие разрывов 2 рода, после чего считаю интеграл, разбивая на половинки.

```python
import sys
import numpy as np
from sympy import *

from system_util.IO import enter_number_of_partitions

infy = np.inf
n = 1000


def simply_calculate_limit(f, a, b):
    f = sympify(f)
    x = symbols('x')
    return integrate(abs(f), (x, a, b))


def simply_calculate_integral(f, a, b):
    f = sympify(f)
    x = symbols('x')
    return integrate(f, (x, a, b))


def check_integral(f, a, b):
    if simply_calculate_integral(f, a, b) is None:
        return False
    return True


def check_limit(f, a, b):
    if simply_calculate_limit(f, a, b) == infy:
        return False
    return True


def check_integral_for_convergence(f, a, b):
    f = sympify(f)
    if check_integral(f, a, b) and check_limit(f, a, b):
        return
    sys.exit("!!!integral contains convergences!!!")


def merge_calculator(f, a, b):
    check_integral_for_convergence(f, a, b)
    f = lambdify("x", sympify(f))

    def merger(f, a, b):
        # we will divide intervals:
        middle = (a + b) / 2
        if a > b:
            a, b = b, a
        h = (b - a) / (n - 2)
        x = np.linspace(a, b, n)[1:-1]
        y = f(x)

        iterator = zip(x, y)

        if not np.isfinite(y).all():
            i_answer = 0
            for (i, j) in iterator:
                if not np.isfinite(j):
                    i_answer += merger(f, a, x)
                    a = x
            i_answer += merger(f, a, b)
            return i_answer

        left_part = np.sum(y[:len(x) // 2])
        right_part = np.sum(y[len(x) // 2:])

        i_answer = left_part + right_part
        i_answer *= h

        return i_answer

    with np.errstate(divide='ignore'):
        return merger(f, a, b)
```

Выводы:

В ходе лабораторной работы я познакомился с новыми для себя методами вычисления интегралов при помощи языков программирования. Больше всего мне понравился метод Симпсона, так как он является наиболее точным из предложенных (при приблизительно схожей эффективности). Кроме того, попробовал реализацию вычисления несвойственных интегралов с обработкой ситуаций, когда интеграл расходится.