Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Лабораторная работа №3

по дисциплине «Вычислительная математика»

«Численное интегрирование»

Вариант №3

Группа: P3212

Выполнил: Балин А. А.

Проверила: Наумова Н. А.

г. Санкт-Петербург, 2024

Цель работы

Найти приближённое значение определённого интеграла с требуемой точностью различными численными методами.

Вычислительная реализация

Решение интеграла аналитически

$$\int_0^2 (-x^3 - x^2 + x + 3)dx = -\frac{x^4}{4} - \frac{x^3}{3} + \frac{x^2}{2} + 3x|_0^2 = -4 - \frac{8}{3} + 2 + 6 = 1\frac{1}{3} := I_{\text{точн}}$$

Решение интеграла по формуле Ньютона-Котеса при n=6

i	0	1	2	3	4	5	6
x_i	0	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{3}{3}$	$\frac{4}{3}$	$\frac{5}{3}$	$\frac{6}{3}$
$f(x_i)$	3	$3\frac{5}{27}$	$2\frac{25}{27}$	2	$\frac{5}{27}$	$-2\frac{20}{27}$	-7
c_6^i	$\frac{41}{420}$	$\frac{216}{420}$	$\frac{27}{420}$	$\frac{272}{420}$	$\frac{27}{420}$	$\frac{216}{420}$	$\frac{41}{420}$

$$I_{\text{Н-К}} = \sum_{i=o}^{6} \text{c}_6^i f(x_i) = 1\frac{1}{3}$$

Относительная погрешность: $\frac{\frac{4}{3} - \frac{4}{3}}{\frac{4}{3}} * 100\% = 0\%$

Решение интеграла методом средних прямоугольников

i	0	1	2	3	4	5	6	7	8	9
x_i	0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
$f(x_i)$	3.089	3.183	3.125	2.867	2.361	1.559	0.413	-1.125	-3.103	-5.569

$$\text{h} = \frac{(\text{b} - \text{a})}{10} = 0.2$$

$$I_{\text{прямоуг}} = h * \sum_{i=0}^{9} f(x_i) = 1.36$$

$$\varepsilon = \frac{|I_{\text{точн}} - I_{\text{прямоуг}|}}{I_{\text{точн}}} * 100\% = 2\%$$

Решение интеграла методом трапеций

i	0	1	2	3	4	5	6	7	8	9	10
x_i	0	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2
$f(x_i)$	3,000	3,152	3,176	3,024	2,648	2,000	1,032	-0,304	-2,056	-4,272	-7,000

$$I_{\text{трап}} = h * \sum_{i=1}^{10} \frac{f(x_{i-1}) + f(x_i)}{2} = 1.28$$

$$\varepsilon = \frac{|I_{\text{точн}} - I_{\text{трап}}|}{I_{\text{точн}}} * 100\% = 4\%$$

Решение интеграла методом Симпсона

i	0	1	2	3	4	5	6	7	8	9	10
x_i	0,0	0,2	0,4	0,6	0,8	1,0	1,2	1,4	1,6	1,8	2,0
$f(x_i)$	3,000	3,152	3,176	3,024	2,648	2,000	1,032	-0,304	-2,056	-4,272	-7,000

$$I_{\text{C}} = \frac{h}{3}\left[f(x_0) + 4\left(\sum_{i \equiv_{mod(2)} 1} f(x_i)\right) + 2\left(\sum_{i \equiv_{mod(2)} 0} f(x_i)\right)\right] = \frac{4}{3} = I_{\text{точн}}$$

Вывод по вычислительной части

В случае с моим вариантом интеграла, наиболее точный и равный действительному значению интеграла результат дали формулы Ньютона-Котеса и Симпсона, другие методы, однако, так же показали хороший результат.

Программная часть

Реализация численных методов интегрирования

```
from numpy import sin,pi, sqrt, exp, cos, nan,divide
from lab3.Answer import Answer
class Integrals:
#_____
_________
    f1 = lambda x: (x-1)**2
    f2 = lambda x: sin(pi*x)/2
    f3 = lambda x: (sqrt(x))/(exp(x)-1)
    f4 = lambda x: cos(pi*x)-exp(sin(pi*x))+1
    f5 = lambda x: divide(divide(1,(x-1)),(x-2))
    functions = [f1, f2, f3, f4, f5]
#_____
_________
    def check_converging(self,risks):
        miss = 0
        for i in range(len(risks)-1):
            if abs(risks[i])<abs(risks[i+1]):
                miss+=1
            else:
                miss=0
            if miss>3:
                raise ValueError("The integral is not converging")


#_____
_________
    def integrating(self,mode,depth=0):
        previous_int = 10**10
        darbu_sums = 0
        current_partions = self.partions
        risks = []
        while abs(previous_int-darbu_sums) > self.eps/2 or
current_partions==self.partions*2:
            previous_int = darbu_sums
            darbu_sums = 0
            h = (self.b - self.a)/current_partions
            for i in range(current_partions):
                 darbu_sums += mode(h,i)
            darbu_sums *= h
```

```
            current_partions *= 2
                        #checking converging
            if(current_partions>self.partions*2):
                risks.append(darbu_sums-previous_int)
            if len(risks)%10 == 0 and len(risks)!=0:
                self.check_converging(risks)
                risks.clear()
            if current_partions>2**20:
                darbu_sums = nan
                break
        if (str(darbu_sums) == str(nan)) and depth==0:
            self.partions *= 2
            return self.integrating(mode,1)
        elif (str(darbu_sums) == str(nan)) and depth==1:
            self.a = self.a+self.eps**2
            return self.integrating(mode,2)
        elif (str(darbu_sums) == str(nan)) and depth==2:
            self.b = self.b-self.eps**2
            return self.integrating(mode,3)
        elif (str(darbu_sums) == str(nan)) and depth==3 and
current_partions<=2**20 or "inf" in str(darbu_sums) or "inf" in
str(previous_int):
            raise ValueError("The integral is not converging or the
function is not defined in the given interval")
        elif current_partions>2**20 and depth==3:
            raise ValueError("Computation time exceeded, try to use
another method or decrease the interval length")
        else:
            return Answer(darbu_sums,current_partions//2)
#_____
_______
    def rectangle(self,mode):
        return self.integrating(lambda h,step:
self.current_function(self.a + h*mode(step)))
    def rectangle_rights(self):
        return self.rectangle(lambda step: step+1)
    def rectangle_lefts(self):
        return self.rectangle(lambda step: step)
    def rectangle_middles(self):
        return self.rectangle(lambda step: step+0.5)
#_____
_______
```

```
    def trapezoid(self):
        return self.integrating(lambda h,step:
(self.current_function(self.a + h*step)+self.current_function(self.a +
h*(step+1)))/2)
#_____________________________________________________________________
_________
    def simpson(self):
        return self.integrating(lambda h,step:
(self.current_function(self.a + h*step) +
                                                4*self.current_function(self
.a + h*(step+0.5)) +
                                                self.current_function(self.a
+ h*(step+1))) / 6)
#_____________________________________________________________________
_________
    methods = [rectangle_lefts, rectangle_rights, rectangle_middles,
trapezoid, simpson]
#_____________________________________________________________________
_________
    def __init__(self, a, b, equathion, method, eps):
        self.a = a
        self.b = b
        self.current_function = self.functions[equathion-1]
        self.eps = eps
        self.partions = 4 #const starting number of partions
        self.method = self.methods[method-1]

    def solve(self):
        return self.method(self)
```

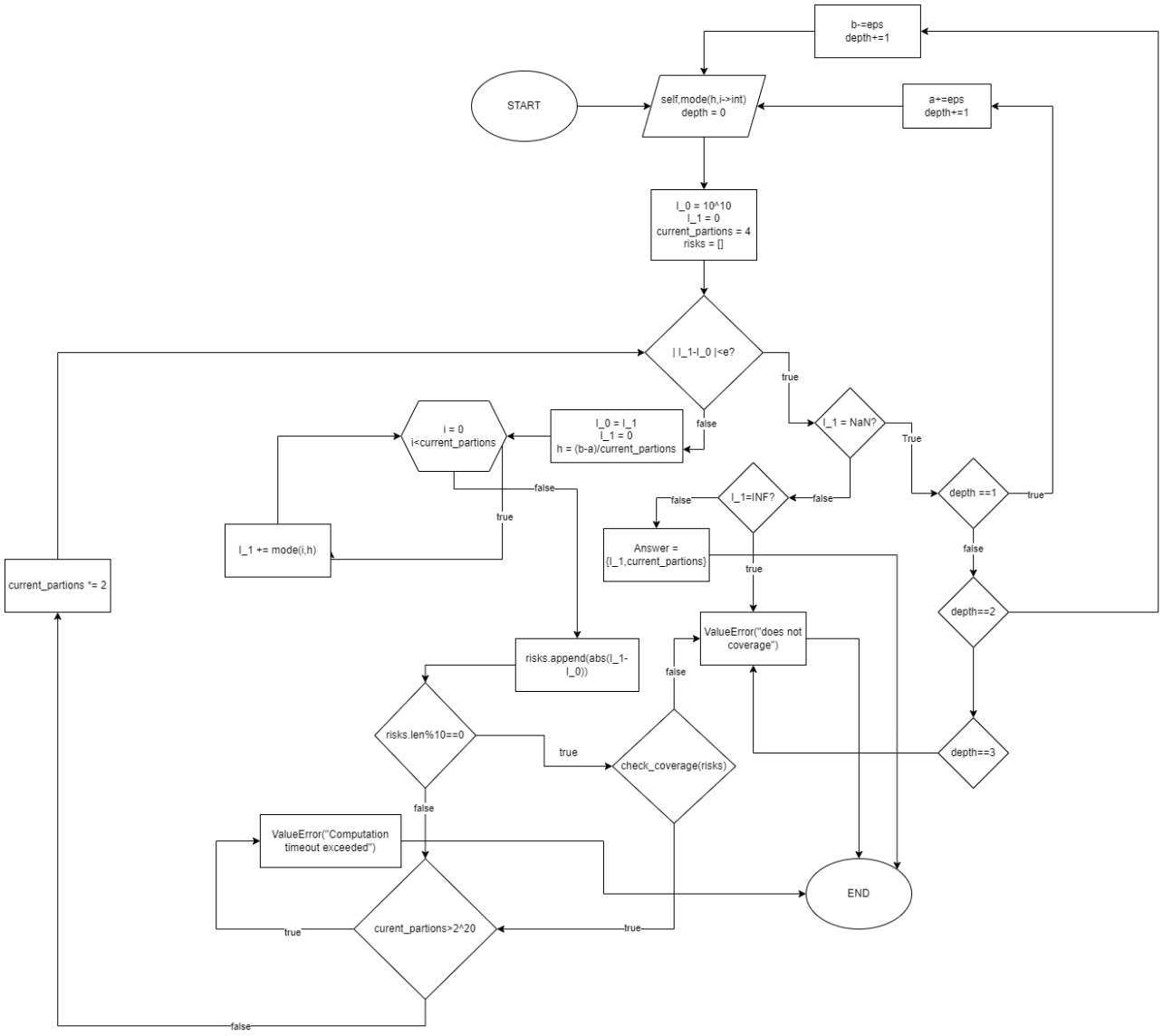

Рисунок 1. Диаграмма

Пример работы программы

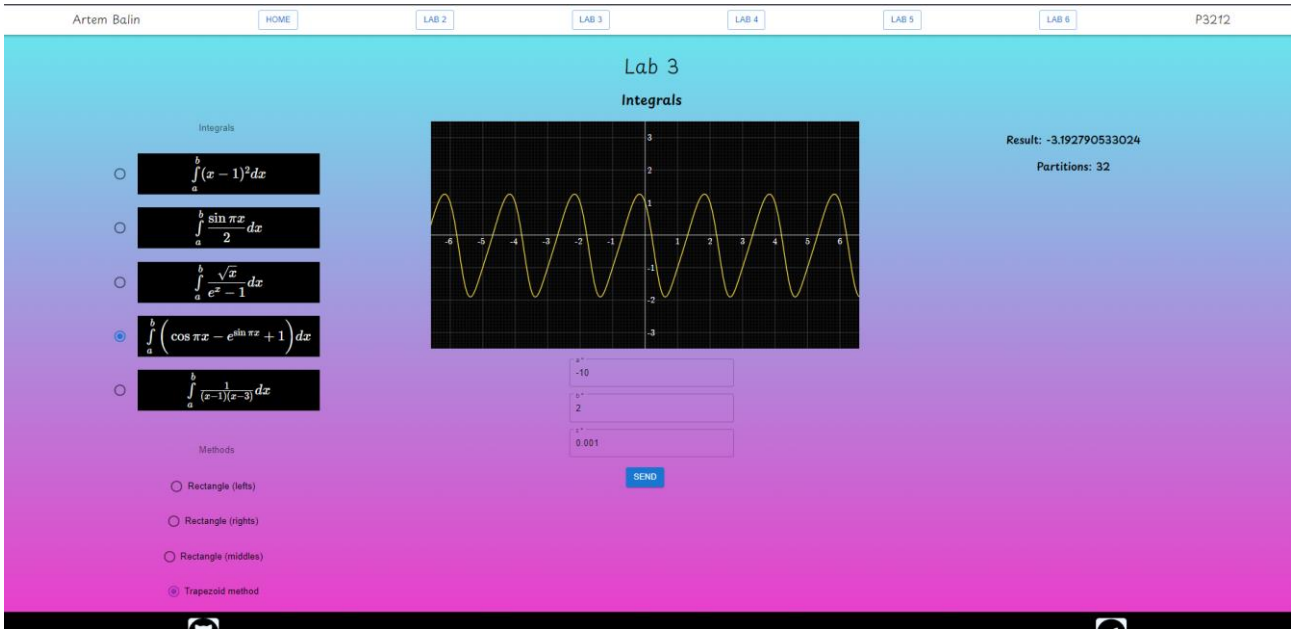

Рисунок 2. Пример работы программы.

Репозиторий с исходниками

https://github.com/ta4ilka69/docs_for_labs/tree/main/Вычмат

Вывод

В ходе реализации данной лабораторной работы я ознакомился с численными методами решения определённых интегралов.