Университет ИТМО МФ КТиУ, Ф ПИиКТ

**Лабораторная работа №3**

**Дисциплина «Вычислительная математика»**

**Численное интегрирование**

**Выполнил** Галлямов Камиль Рустемович

**Преподаватель:**

Машина Екатерина Алексеевна

г. Санкт-Петербург 2024 г.

# Вычислительная реализация задачи

Точное вычисление:

Iточн =

= (- 16 / 4 – 8 / 3 + 4 / 2 + 3 \* 2) – 0 = -4 – 8 / 3 + 2 + 6 = 4 – 8 / 3 = **4 / 3 = 1.3333333**

По формуле Ньютона – Котеса при n = 6:

h = (b – a) / n = (2 – 0) / 6 = 1 / 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| xi | 0 | 1/3 | 2/3 | 1 | 4/3 | 5/3 | 2 |
| yi | 3 | 3.185 | 2.926 | 2 | 0.185 | -2.741 | -7 |
| c6i | 0.098 | 0.514 | 0.064 | 0.648 | 0.064 | 0.514 | 0.098 |

Icotes = = 0.098 \* 3 + 0.514 \* 3.185 + 0.064 \* 2.926 + 0.648 \* 2 + 0.064 \* 0.185 + 0.514 \* (-2.741) + 0.098 \* (-7) = **1.331**

R = | Icotes - Iточн | = 0.002 => 0.15%

По формуле средних прямоугольников при n = 10:

h = (b – a) / n = 0.2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| xi | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1 | 1.2 | 1.4 | 1.6 | 1.8 | 2 |
| (xi + xi - 1) / 2 |  | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 | 1.1 | 1.3 | 1.5 | 1.7 | 1.9 |
| f ((xi + xi - 1) / 2) |  | 3.09 | 3.18 | 3.13 | 2.87 | 2.36 | 1.56 | 0.41 | -1.13 | -3.1 | -5.57 |

Iсред = = 0.2 \* 6.8 = **1.36**

R = | Iсред - Iточн | = 0.03 => 2.25%

По формуле трапеций при n = 10:

h = (b – a) / n = 0.2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| xi | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1 | 1.2 | 1.4 | 1.6 | 1.8 | 2 |
| yi | 3 | 3.15 | 3.18 | 3.02 | 2.65 | 2 | 1.03 | -0.3 | -2.06 | -4.27 | -7 |

Iтрап = = 0.2 \* (-2 + 8.4) = **1.28**

R = | Iтрап - Iточн | = 0.05 => 3.75%

По формуле Симпсона при n = 10:

h = (b – a) / n = 0.2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| xi | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1 | 1.2 | 1.4 | 1.6 | 1.8 | 2 |
| yi | 3 | 3.15 | 3.18 | 3.02 | 2.65 | 2 | 1.03 | -0.3 | -2.06 | -4.27 | -7 |

Iсимп = =

= 0.2 / 3 \* (3 + 4 \* (3.6) + 2 \* (4.8) - 7) = **1.333**

R = | Iтрап - Iсимп | = 0.0 => 0%

# Программная реализация задачи

**from** **math** **import** sqrt

**def** **left\_rectangles\_method**(func, a, b, n):

h = (b - a) / n

xs = [a + i \* h **for** i **in** range(n + **1**)] # [x0, x1, ..., xn]

**return** h \* sum([func(xs[i]) **for** i **in** range(n)])

**def** **right\_rectangles\_method**(func, a, b, n):

h = (b - a) / n

xs = [a + i \* h **for** i **in** range(n + **1**)] # [x0, x1, ..., xn]

**return** h \* sum([func(xs[i]) **for** i **in** range(**1**, n + **1**)])

**def** **middle\_rectangles\_method**(func, a, b, n):

h = (b - a) / n

xs = [a + i \* h **for** i **in** range(n + **1**)] # [x0, x1, ..., xn]

**return** h \* sum([func((xs[i - **1**] + xs[i]) / **2**) **for** i **in** range(**1**, n + **1**)])

**def** **trapezoid\_method**(func, a, b, n):

h = (b - a) / n

xs = [a + i \* h **for** i **in** range(n + **1**)] # [x0, x1, ..., xn]

ys = [func(x) **for** x **in** xs] # [y0, y1, ..., yn]

**return** h \* ((ys[**0**] + ys[n]) / **2** +

sum([ys[i] **for** i **in** range(**1**, n)]))

**def** **simpson\_method**(func, a, b, n):

h = (b - a) / n

xs = [a + i \* h **for** i **in** range(n + **1**)] # [x0, x1, ..., xn]

ys = [func(x) **for** x **in** xs] # [y0, y1, ..., yn]

**return** h / **3** \* (ys[**0**] +

**4** \* sum([ys[i] **for** i **in** range(**1**, n, **2**)]) +

**2** \* sum([ys[i] **for** i **in** range(**2**, n - **1**, **2**)]) +

ys[n])

**def** **find\_breakpoints**(func, a, b, eps):

breakpoints = []

x = a

**while** x <= b:

**try**:

func(x)

**except** **Exception**:

breakpoints.append(x)

x = round(x + eps, **3**)

**return** breakpoints

**def** **try\_to\_compute**(func, x):

**try**:

**return** func(x)

**except** **Exception**:

**return** None

**def** **compute**(func, a, b, eps, method):

n = **4**

i0 = method(func, a, b, n)

i1 = method(func, a, b, n \* **2**)

**while** abs(i1 - i0) > eps:

n \*= **2**

i0 = i1

i1 = method(func, a, b, n \* **2**)

**return** i1, n \* **2**

methods = [left\_rectangles\_method,

right\_rectangles\_method,

middle\_rectangles\_method,

trapezoid\_method,

simpson\_method]

eps = **0.0001**

**print**('f1 = 1 / x')

**print**('f2 = 1 / sqrt(x)')

**print**('f3 = 1 / (1 - x)')

func\_number = input('input function number (1 or 2 or 3): ')

**while** func\_number **not** **in** {'1', '2', '3'}:

func\_number = input('input function number (1 or 2 or 3): ')

**while** **1**:

**try**:

a = float(input('input a (real number, the lower limit of integration): '))

**except** **Exception**:

**continue**

**break**

**while** **1**:

**try**:

b = float(input('input b (real number, the upper limit of integration): '))

**except** **Exception**:

**continue**

**break**

func\_number = int(func\_number)

**if** func\_number == **1**:

f = **lambda** x: **1** / x

**elif** func\_number == **2**:

f = **lambda** x: **1** / sqrt(x)

**else**:

f = **lambda** x: **1** / (**1** - x)

breakpoints = find\_breakpoints(f, a, b, **0.01**)

**print**('breakpoints: ', \*breakpoints)

stop = False

**for** bp **in** breakpoints:

y1 = try\_to\_compute(f, bp - eps)

y2 = try\_to\_compute(f, bp + eps)

**if** y1 **is** **not** None **and** y2 **is** **not** None **and** abs(y1 - y2) > eps:

**print**('integral does not converge!')

stop = True

**if** **not** stop:

**if** a **in** breakpoints:

a += eps

**if** b **in** breakpoints:

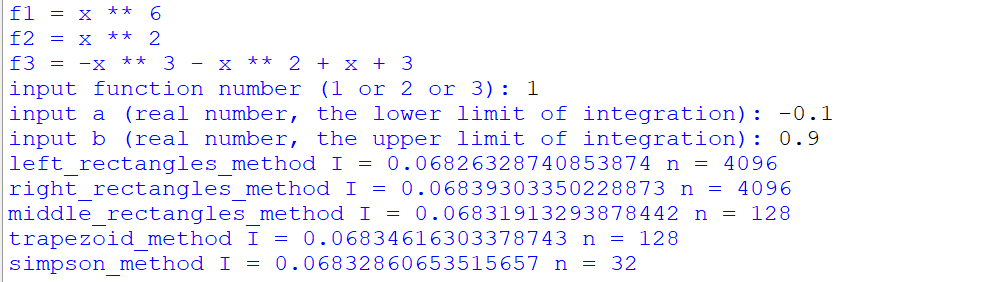
b -= eps

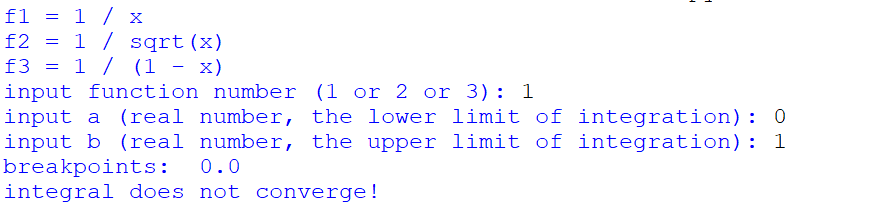
**for** method **in** methods:

res, n = compute(f, a, b, eps, method)

**print**(f'{method.\_\_name\_\_} I = {res} n = {n}')

# Тестовые данные





# Вывод

В ходе лабораторной работы я познакомился с численными методами решения определённых интегралов.

Метод прямоугольников – частный случай метода Ньютона-Котеса. Имеет три модификации (левые, средние, правые прямоугольники). Средние – самая лучшая модификация. Из плюсов – простота в понимании и в реализации, из минусов – не такой точный.

Метод трапеций - частный случай метода Ньютона-Котеса. Тоже простой и не такой точный (но точнее прямоугольников).

Метод Симпсона – частный случай метода Ньютона-Котеса. Более сложный в понимании и имеет более сложную формулу, но зато является довольно точным.

Метод Ньютона-Котеса – общий случай перечисленных выше методов. Использует более общие и сложные формулы, не такой простой в программной реализации.

Квадратурная формула Гаусса – позволяет повысить порядок точности методов за счёт специального выбора узлов интегрирования. Состоит из двух этапов: 1) свести интеграл к интегралу с пределами [-1, 1]; 2) вычислить по специальной формуле (сумма значений подынтегральной функции в специальных точках, умноженных на весовые коэффициенты). Является более сложным в понимании и в программной реализации.