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:

$I_{точн} = \int_0^2 (-x^3 - x^2 + x + 3)dx = \left(-\frac{x^4}{4} - \frac{x^3}{3} + \frac{x^2}{2} + 3x\right)\Big|_0^2 =$
= (- 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
x_i	0	1/3	2/3	1	4/3	5/3	2
y_i	3	3.185	2.926	2	0.185	-2.741	-7
c_6^i	0.098	0.514	0.064	0.648	0.064	0.514	0.098

$I_{cotes} = \sum_{i=0}^{n} c_n^i * f(x_i)$ = 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 = | I_{cotes} - $I_{точн}$ | = 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
x_i	0	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2
$(x_i + x_{i-1}) / 2$		0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
$f((x_i + x_{i-1}) / 2)$		3.09	3.18	3.13	2.87	2.36	1.56	0.41	-1.13	-3.1	-5.57

$I_{сред} = h * \sum_{i=1}^{n} f\left(\frac{(x_{i-1}+x_i)}{2}\right)$ = 0.2 * 6.8 = **1.36**

R = | $I_{сред}$ - $I_{точн}$ | = 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
x_i	0	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2
y_i	3	3.15	3.18	3.02	2.65	2	1.03	-0.3	-2.06	-4.27	-7

$I_{трап} = h * \left(\frac{y_0+y_n}{2} + \sum_{i=1}^{n-1} y_i\right)$ = 0.2 * (-2 + 8.4) = **1.28**

R = | $I_{трап}$ - $I_{точн}$ | = 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
x_i	0	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2

y_i	3	3.15	3.18	3.02	2.65	2	1.03	-0.3	-2.06	-4.27	-7

$$I_{симп} = \frac{h}{3} * (y_0 + 4 * (y_1 + y_3 + \cdots + y_{n-1}) + 2 * (y_2 + y_4 + \cdots + y_{n-2}) + y_n) =$$

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

R = | $I_{трап}$ - $I_{симп}$ | = 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:
```

```python
        b -= eps

    for method in methods:
        res, n = compute(f, a, b, eps, method)
        print(f'{method.__name__} I = {res} n = {n}')
```

Тестовые данные

```
f1 = x ** 6
f2 = x ** 2
f3 = -x ** 3 - x ** 2 + x + 3
input function number (1 or 2 or 3): 1
input a (real number, the lower limit of integration): -0.1
input b (real number, the upper limit of integration): 0.9
left_rectangles_method I = 0.06826328740853874 n = 4096
right_rectangles_method I = 0.06839303350228873 n = 4096
middle_rectangles_method I = 0.06831913293878442 n = 128
trapezoid_method I = 0.06834616303378743 n = 128
simpson_method I = 0.06832860653515657 n = 32
```

```
f1 = 1 / x
f2 = 1 / sqrt(x)
f3 = 1 / (1 - x)
input function number (1 or 2 or 3): 1
input a (real number, the lower limit of integration): 0
input b (real number, the upper limit of integration): 1
breakpoints:  0.0
integral does not converge!
```

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.