Санкт-Петербургский политехнический университет
Институт прикладной математики и механики
Высшая школа теоретической механики

Направление подготовки
"01.03.03 Механика и математическое моделирование"

Отчет по лабораторной работе №4
Тема работы: " Численное интегрирование"
Дисциплина: "Численные методы"

Выполнил студент гр. 3630103/90001 Михеев Евгений Викторович
Преподаватель: Павлова Людмила Владимировна

Санкт-Петербург
2021

Отчет по лаб. работе № 4

"Численное интегрирование"

"Квадратура Лобатто"

№1 Постановка задачи

Произвести численное интегрирование заданной ф-ции методом Лобатто, получить зависимость кол-ва итераций вычисления $\int$ по правилу Рунге от заданной точности

№2) Алгоритм метода

Будем искать $\int_a^b$ как $\int_a^b f(x)dx \approx \sum_{k=1}^{n} A_k f(x_k)$

Дана задача вычислить $\int$ методом Лобатто с 4 узлами.

Рассм:

$$\int_{-1}^{1} f(x)dx = A_1 f(-1) + A_4 f(1) + \sum_{k=2}^{n-1} A_k f(x_k)$$

Представим $f(x) = q(x)\,\tilde{\omega}(x)\,P_{n-3}(x)$

$q = (1-x)(x+1) = (b-x)(x-a) = \tilde{p}(x)$

~~$\tilde{\omega}(x) = \frac{c}{(1-x)(x+1)} \frac{d^2}{dx^2}[$~~

$$\tilde{\omega}(x) = \frac{c}{(1-x)(x+1)} \frac{d^2}{dx^2}\left[(1-x)^3(x+1)^3\right] = \frac{c}{(1-x)(x+1)} \cdot (1-x^2)(5x^2-1)(-6)$$

$$= -6c(5x^2-1)$$

$$\tilde{\omega}(x) = 0 \Leftrightarrow x = \pm\sqrt{\frac{1}{5}} \Rightarrow x_2 = -\sqrt{\frac{1}{5}};\ x_3 = \sqrt{\frac{1}{5}}$$

$A_1 A_4, A_2$ и A_3 находим из сис-мы определяющих уравнений

$$\begin{cases} A_1 + A_2 + A_3 + A_4 = 2 \\ A_1(-1) + A_2\left(-\sqrt{\frac{1}{5}}\right) + A_3\sqrt{\frac{1}{5}} + A_3 = 0 \\ A_4 \cdot 1 + A_2 \cdot \frac{1}{5} + A_3 \cdot \frac{1}{5} + A_3 \cdot 1 = \frac{2}{3} \\ A_1 \cdot (-1) + A_2\left(\frac{-1}{5}\right)^{3/2} + A_3\left(\frac{1}{5}\right)^{3/2} + A_3 \cdot 1 = 0 \end{cases} \Bigg| \begin{array}{l} A_1 = \frac{1}{6} \\ A_2 = \frac{5}{6} \\ A_3 = \frac{5}{6} \\ A_4 = \frac{1}{6} \end{array}$$

Т.о получаем: $\int_{-1}^{1} f(x)\,dx = \frac{1}{6}f(-1) + \frac{5}{6}f\left(-\sqrt{\frac{1}{5}}\right) + \frac{5}{6}f\left(\frac{1}{\sqrt{5}}\right) + \frac{1}{6}f(1)$

кв. ф-ла Лобатто на 4 узлах

$[-1, 1] \to [a, b]$: $t_i = \frac{a+b}{2} + \frac{b-a}{2}x_i$; $\int_a^b \approx \frac{b-a}{2}\sum A_k f\left(\frac{a+b}{2} + \frac{b-a}{2}x_i\right)$

№3) Опр. интеграл $\exists!$ для $\forall$ непр на $[a, b]$ ф-ции

№4) —

№5) Тупой пример

$$\varepsilon_k = \frac{|I_{2n} - I_n|}{2^{2n-2} - 1} = \frac{|I_{2n} - I_n|}{63} \text{ - правило Тупого}$$

$f(x) = \cos(x)$; $[-1, 1]$

$\varepsilon = 0{,}01$

$n = 1$ - число интервалов

$$I = \frac{1}{6}\left[\cos(-1) + \cos(1)\right] + \frac{5}{6}\left[\cos\left(-\frac{1}{\sqrt{5}}\right) + \cos\left(\frac{1}{\sqrt{5}}\right)\right] \approx 1{,}8$$

$n = 2$

$$I = I_{[-1,0]} + I_{[0,1]} = \left[\frac{1}{6}(\cos(-1) + \cos(0)) + \right.$$

$$+ \frac{5}{6}\left[\cos\left(\frac{-1}{2} + \frac{1}{2}\left(-\sqrt{\frac{1}{5}}\right)\right) + \cos\left(\frac{-1}{2} + \frac{1}{2}\sqrt{\frac{1}{5}}\right)\right. + \frac{1}{6}\left(\cos(0) + \cos(1)\right)$$

$$\left. + \frac{5}{6}\left[\cos\left(\frac{1}{2} + \frac{1}{2}\left(-\sqrt{\frac{1}{5}}\right)\right) + \cos\left(\frac{1}{2} + \frac{1}{2}\sqrt{\frac{1}{5}}\right)\right]\right] \cdot \frac{1}{2} \approx 1{,}68$$

$$R_R = \frac{|1{,}68... - 1{,}68...|}{63} \approx 0 < \varepsilon \Rightarrow \int_a^b \cos(x)\,dx \approx 1{,}68...$$

№5) Контрольные тесты

Дано: $f(x) = \cos x$

Ход работы

1) Написать алгоритм, вычисляющий $\int_a^b \cos x\,dx$

2) Получить зависимость кол-ва итераций по правилу Рунге от точности, варьирующейся в пределах $\varepsilon = [10^{-1}; 10^{-15}]$

3) Привести данные исследования на симм. и несимм. промежутках

№7) Структура программы

integral.py ⟷ NMclasses.Integration.Lobatto()

1) integral.py

In: f(x), a, b – границы промежутка

Out: график f(x)

таблица зависимости числа разбиений интервала от точности ε.

Также в таблице содержится информация о времени вычисления ∫ в секундах

2) NMclasses.Integration.Lobatto()

In: f(x), a, b, n – число разбиений [a, b]

Out: значение интеграла $\overline{I}$

№8) Численный анализ. Исследование проводилось на двух уч-ках длиной 10 – симметричном [-5, 5] и несимметричном [-3, 7]. Изучим полученные результаты и сравним их с методом серединных прямоугольников

1) Данную ~~классическое преимущество в скорости сходимости квадратуры Лобатто~~: для достижения погрешности

1) Налицо колоссальное преимущество в скорости сходимости метода квадратуры Лобатто: скажем, для достижения точности в 10^{-10} методу серединных прямоугольников потребуется $\approx 5 \cdot 10^6$ точек разбиения и, соответственно $5 \cdot 10^6$ вычислений значений ф-ции в них, а квадратуре Лобатто потребуется лишь 64 точки разбиения, т.е 192 вычисления значения ф-ции.

2) Квадратура Лобатто спокойно достигает точности ε вплоть до 10^{-15}, в то время как метод серединных прямуг. в случае с данной ф-цией и данным промежутком достиг лишь точности $\approx 10^{-12}$ и на вычисление $\int$ с $\varepsilon = 10^{-12}$ мой программе потребуется целых 8 секунд, в то время как квадратуре Лобатто – $4 \cdot 10^{-2}$ секунды.

3) Симметричность промежутка не оказала заметного влияния на сходимость метода: это видно из таблиц сходимости

1) Квадратура Лобатто, интервал [-3, 7]

Eps	10E-1	10E-2	10E-3	10E-4	10E-5	10E-6	10E-7	10E-8	10E-9	10E-10	10E-11	10E-12	10E-13	10E-14	10E-15
n, точек разб	2	4	4	8	8	16	16	32	32	64	64	128	256	256	512
t, сек	0	0	0	0	0	0,001023	0	0,000996	0,000997	0,000997	0,001996	0,002992	0,006958	0,006981	0,013989

2) Квадратура Лобатто, интервал [-5, 5]

Eps	10E-1	10E-2	10E-3	10E-4	10E-5	10E-6	10E-7	10E-8	10E-9	10E-10	10E-11	10E-12	10E-13	10E-14	10E-15
n, точек разб	2	4	8	8	8	16	16	32	64	64	128	128	256	256	512
t, сек	0	0	0	0,001022	0	0	0,000997	0	0,00197	0,00202	0,002966	0,002992	0,006981	0,006983	0,011991

Для сравнения, метод серединных прямоугольников:

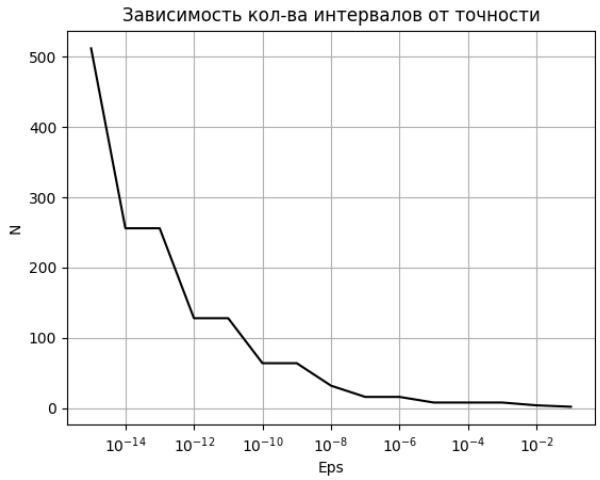
Интервал [-5, 5]

	10E-1	10E-2	10E-3	10E-4	10E-5	10E-6	10E-7	10E-8	10E-9	10E-10	10E-11	10E-12
n	16	32	128	512	1024	4096	16384	32768	131072	524288	1048576	4194304
t	0	0	0	0,001	0,002	0,00898	0,03312	0,0674	0,25648	1,03074	2,05678	8,12976

№9) Вывод: квадратура Лежандра является отличным численным методом расчета определенного интеграла: метод обладает высокой точностью даже при небольшом числе разбиений интервала [a, b], а также способен давать результат с высокой точностью при небольших вычисл. затратах и сходимость не зависит от промежутка интегрирования

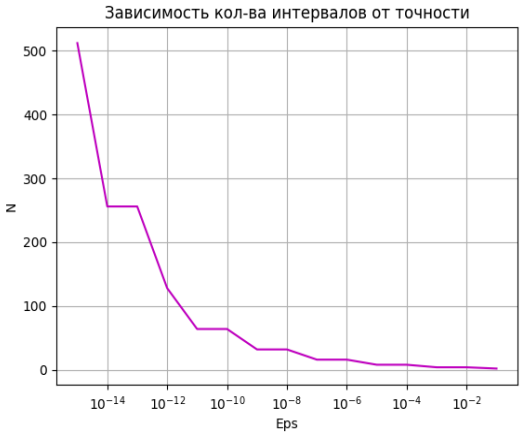
Графики и таблицы

1) Симметричный интервал ([-5, 5])



Eps	10E-1	10E-2	10E-3	10E-4	10E-5	10E-6	10E-7	10E-8	10E-9	10E-10	10E-11	10E-12	10E-13	10E-14	10E-15
n, точек разб	2	4	8	8	8	16	16	32	64	64	128	128	256	256	512
t, сек	0	0	0	0,001022	0	0	0,000997	0	0,00197	0,00202	0,002966	0,002992	0,006981	0,006983	0,011991

2) Несимметричный интервал ([-3, 7])



Eps	10E-1	10E-2	10E-3	10E-4	10E-5	10E-6	10E-7	10E-8	10E-9	10E-10	10E-11	10E-12	10E-13	10E-14	10E-15
n, точек разб	2	4	4	8	8	16	16	32	32	64	64	128	256	256	512
t, сек	0	0	0	0	0	0,001023	0	0,000996	0,000997	0,000997	0,001996	0,002992	0,006958	0,006981	0,013989

Код

Integral.py

```
import numpy as np
import NMclasses as NM
import math
import matplotlib.pyplot as plt
import pandas as pd
import time

f = lambda x: np.cos(x)
a = -3
b = 7
x = np.linspace(a, b, 500)

def integralEps(epsMax):
    eps = 0.1

    K = dict()
    Eps = list()
    N = list()

    for i in range(1, epsMax):
        eps = math.pow(10, -i)
        n, d = 1, 1
        t_in = time.time()
        Ip = NM.Integtration(f, a, b, n).lobattoInt()
        while d > eps:
            In = NM.Integtration(f, a, b, 2*n).lobattoInt()
            d = math.fabs(In - Ip)/63
            Ip = In
            n *= 2
        t_out = time.time()
        ar = [n, t_out - t_in]
        K[f'10E-{i}'] = ar
        N.append(n)
        Eps.append(eps)

    return K, Eps, N

if __name__ == '__main__':

    print(NM.Integtration(f, a, b, 2).lobattoInt())
    K, Eps, N = integralEps(16)

    table = pd.DataFrame(K, index = ['n, точек разб', 't, сек'])
    table.to_excel(r'C:\Users\mchav\OneDrive\Учеба\Предметы\ЧМ\2 сем\3-4\Код\table.xlsx')

    NM.Plots(1, [[x, f(x)]], 'График функции', 'x', 'f(x)').build()
    NM.Plots(2, [[Eps, N]], 'Зависимость кол-
ва интервалов от точности', 'Eps', 'N').build('logX')
    plt.show()
```

NMclasses.py

```python
class Integtration:
    def __init__(self, func, a, b, n=None):
        self.func = func
        self.a = a
        self.b = b
        self.n = n

    def rectangles(self):
        integral = 0.0
        step = (self.b - self.a) / self.n
        X = [self.a+i*step for i in range(self.n)]
        for x in range(len(X)):
            integral += step * self.func(X[x] + step / 2)
        return integral

    def __lobattoAlg(self, start, end):
        integral = 0.0
        t_i = np.array([-1, -math.sqrt(1/5), math.sqrt(1/5), 1])
        weighs = np.array([1/6, 5/6, 5/6, 1/6])
        for i in range(len(weighs)):
            integral += self.func( (end - start)*t_i[i]/2 + (start + end)/2 )*weighs[i]
        integral *= (end - start)/2
        return integral

    def lobattoInt(self):
        integral = 0.0
        step = (self.b - self.a)/self.n
        X = [self.a + i*step for i in range(self.n+1)]
        for i in range(self.n):
            integral += self.__lobattoAlg(X[i], X[i+1])
        return integral
```