# ЛАБОРАТОРНАЯ РАБОТА 5

**Численное интегрирование**

Вариант 20

Студент: Маркаров М.Г

Преподаватель: ст.преп.Крупин

\

**Задача 5.**1. Используя формулы численного интегрирования и численного дифференцирования, вычислить интеграл и производную подинтегральной функции, а также исследовать поведение погрешности при уменьшении шага интегрирования и соответственно дифференцирования.

### ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ

1. Дан интеграл **.** По формуле Ньютона-Лейбница найти егозначение.

2. Реализовать программно серию расчетов вычисления квадратурных сумм по составной формуле численного интегрирования индивидуального варианта, шаги интегрирования взять равными =(b-a)/ , к=1,2,…15. Вычисления следует прекратить на том значении шага, если расчет занимает более 10 минут. Вычислить также величины погрешностей

. Результаты расчетов представить во втором столбце таблицы 1.

3. Вычислить производную подинтегральной функции и найти ее значение в левом конце отрезка . Реализовать программно серию расчетов вычисления приближенного значения производной по формуле численного дифференцирования индивидуального варианта, = , к=1,2,…15. По формуле численного дифференцирования вычислить массив приближенных значений производной в точке *a* и величины погрешности. Результаты расчетов представить в 4 столбце таблицы . Построить график зависимости погрешности от величины шага и указать минимальное значение погрешности и шаг, при котором значение достигнуто.

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| Шаг | и | Шаг | и |
|  |  |  |  |
|  |  |  |  |
| ….. |  | ….. |  |
|  |  |  |  |

5. Оформить отчет по задаче. В отчет включить ответы на вопросы:

a) Можно ли по заполненным таблицам показать, что расчеты подтверждают указанный порядок точности

метода.

b). Можно ли по заполненным таблицам определить, появилась ли в расчетах хорошая или плохая обусловленность метода.

Решение задачи 5.1:

0. N=20

N mod4 + 1 =0+1=1

Имеем формулу левых прямоугольников (и центральную разностную производную)

1.Вычислим интеграл от степенной функции в пределах от N до N+1.

Реализация на языке python:

import numpy as np

N = 20

coeff = [N, 0.1 \* N, -0.5 \* N, -N, 0.5 \* N] #задание коэффицентов полинома

p = np.poly1d(coeff[::-1]) # задание полинома

p\_integ = np.polyint(p) # интегрирование

I = p\_integ(N+1) - p\_integ(N) # ф-ла Ньютона-Лейбница

print(I)

Результат: 1591654.666666668

2. Будем искать интеграл численно по формуле левых прямоугольников

Прим. Длина отрезка интегрирования равна 1 (N+1-N). Используем это находя число разбиений отрезка seg=1/step где step-i-ый шаг.

Реализация на языке python:

def rect\_sum\_left(f, left\_border, step):

   sum=0

   seg=int(1/step)  # сег это число разбиений отрезка интегрирования

   for i in range(seg):sum+=step\*f(left\_border+i\*step)

   return sum

step\_arr=np.ndarray(15) # //массив шагов

for i in range(15):

step\_arr[i]=10\*\*(-(i+1))

N = 20

coeff = [N, 0.1 \* N, -0.5 \* N, -N, 0.5 \* N] #// полином

p = np.poly1d(coeff[::-1])

for i in range(15):

Sk = rect\_sum\_left(p, N, step\_arr[i]) #// 15(в реальности 7-8) сумм с разными шагами

print(Sk)

Результат:

Формат( I, Sk,delta,i)

***1591654.6666666679084301 1575735.5333000002428889 15919.1333666676655412 0***

***1591654.6666666679084301 1590059.1563333300873637 1595.5103333378210664 1***

***1591654.6666666679084301 1591495.0796633353456855 159.5870033325627446 2***

***1591654.6666666679084301 1591638.7076066327281296 15.9590600351803005 3***

***1591654.6666666679084301 1591635.5183714183513075 1.91482952495571226 4***

***1591654.6666666679084301 1591654.7170757011417300 0.1595909667667001 5***

***1591654.6666666679084301 1591654.6507075720001012 0.0159590959083289 6***

***Пересчитано с учетом забытого узла***

Формула левых прямоугольников имеет порядок точности 1. Нетрудно заметить что погрешность убывает согласно формуле.

Вычислим центральную разностную производную в левой точке(N)

Реализация на языке Python:

def central\_div(f, left\_border, right\_border, step):

df = (f(left\_border + step) - f(left\_border - step)) / (2 \* step)

return df

N = 20

coeff = [N, 0.1 \* N, -0.5 \* N, -N, 0.5 \* N]

p = np.poly1d(coeff[::-1])

p\_der = np.polyder(p)

f\_der = p\_der(N) # эталонное значение производной

for i in range(15):

dk = central\_div(p, N, N+1, step\_arr[i])

print('%.16f %.16f %.16f %d' % (f\_der, dk, (abs(f\_der - dk)), i))

Результат работы программы:

Формат(f\_der,dk,delta,i)

295602.0000000000000000 295609.8000000021420419 7.8000000021420419 0

295602.0000000000000000 295602.0780000486411154 0.0780000486411154 1

295602.0000000000000000 295602.0007805200293660 0.0007805200293660 2

295602.0000000000000000 295602.0000064745545387 0.0000064745545387 3

295602.0000000000000000 295601.9999925047159195 0.0000074952840805 4

295602.0000000000000000 295602.0003184676170349 0.0003184676170349 5

295602.0000000000000000 295602.0028796046972275 0.0028796046972275 6

Формула центральной разностной производной имеет порядок точности 2.

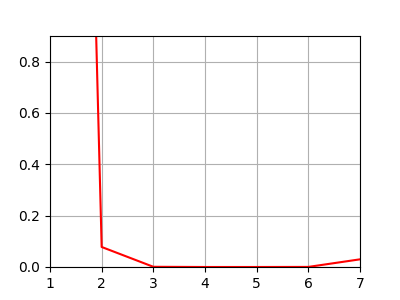
Это проявляется до 4 итерации( i=3). Далее погрешность растет из-за вычислительной погрешности.

Составим таблицу:(поправлена после ошибки)

|  |  |  |  |
| --- | --- | --- | --- |
| Шаг | и | Шаг | и |
|  | *1575735.5333000002428889 15919.1333666676655412* |  | 295609.8000000021420419 7.8000000021420419 |
|  | 1590059.1563333300873637 1595.5103333378210664 |  | 295602.0780000486411154 0.0780000486411154 |
|  | 1591495.0796633353456855 159.5870033325627446 |  | 295602.0007805200293660 0.0007805200293660 |
|  | 1591638.7076066327281296 15.9590600351803005 |  | 295602.0000064745545387 0.0000064745545387 |
|  | 1591635.5183714183513075 1.91482952495571226 |  | 295601.9999925047159195 0.0000074952840805 |
|  | 1591654.7170757011417300 0.1595909667667001 |  | 295602.0003184676170349 0.0003184676170349 |
|  | 1591654.6507075720001012 0.0159590959083289 |  | 295602.0028796046972275 0.0028796046972275 |

Составим графики зависимости погрешностей от шага.

1. Погрешность производной от шага(10^-k)



2.Погрешность интеграла от шага



Вывод по задаче:

1) Данные подтверждают порядок точности формул

В формуле левых прямоугольников погрешность убывает линейно.

В формуле центральной производной погрешность убывает квадратично.

2*) Из-за невозможности провести 15 итераций для формулы прямоугольников делаем вывод о хорошей обусловленности задачи(погрешность стабильно убывает до последней возможной итерации)*

*Для формулы центральной разностной производной для нашей функции делаем вывод о плохой обусловленности задачи,т.к погрешность начинает возрастать уже после 4 итерации(из 7 возможных).*

**Задача 5.2**. Вычислить интеграл из таблицы 5.1, используя встроенные средства Python. Используя теоретическую оценку погрешности для квадратурной формулы индивидуального варианта, найти шаг интегрирования, требуемый для достижения заданной точности . Вычислить значение интеграла при найденном шаге и сравнить полученные результаты.

Решение задачи:

1) Найдем интеграл через средства Python:

import math as m

import numpy as np

N = 20

coeff = [N, 0.1 \* N, -0.5 \* N, -N, 0.5 \* N]

p = np.poly1d(coeff[::-1])

p\_integ = np.polyint(p)

I = p\_integ(N+1) - p\_integ(N)

Результат работы программы:

1591654.666666668

2) Воспользуемся теоретической оценкой погрешности формулы левых прямоугольников:

Запишем :

Отсюда:

Учтем: (b-a)=(N+1-N)=1 и M1=max(abs(df/dx)) на отрезке интегрирования, отсюда M1=df/dx в точке N+1 ввиду роста степенной функции.

Воспользуемся вычислительными средствами языка python:

eps=10\*\*(-8)

p\_der = np.polyder(p)

f\_der = p\_der(N+1)

h0=2\*eps/f\_der

print(h0)

Результат: 5.821365575936804e-14

Это теоретический шаг для достижения такой точности, однако он даст нецелое число разделений отрезка интегрирования.

Найдем целое число разбиений:

Квадратные скобки-оператор выделения целой части числа

Получаем, что n= 1014

Найдем шаг при таком числе разбиений:

h=(b-a)/n=10-14

Вычислим интеграл при таком шаге:

def rect\_sum\_left(f, left\_border, step):

sum=0

seg=int(1/step)

for i in range(seg-1):sum+=step\*f(left\_border+i\*step)

return sum

N = 20

coeff = [N, 0.1 \* N, -0.5 \* N, -N, 0.5 \* N]

p = np.poly1d(coeff[::-1])

Ih=rect\_sum\_left(p,N,10\*\*(-14))

print(Ih)

Результат(спустя много часов): 1591654.666666663

Сравним с эталонным интегралом: 1591654.666666668

Delta=0.000000005<10^-8

Задача выполнена!

**Задача 5.3.** Вычислить интеграл из таблицы 5.3 с заданной точностью .

### ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ

1. Вычислить интеграл  с помощью встроенных средств *Python***.**

2. Составить программу, содержащую следующие разделы:

а) процедуру-функцию, вычисляющую интеграл по составной квадратурной формуле

индивидуального варианта с заданным шагом h.

б) программу, вычисляющую значение интеграла с заданной точностью ; оценку погрешности производить на основе правила Рунге.

в) выполнить уточнение по Рунге для найденного значения интеграла.

г) результатом работы программы должны быть следующие величины:

n- число разбиений отрезка интегрирования, при котором заданная точность достигнута,

-полученное значение интеграла,  - величина погрешности, полученная по правилу Рунге, -уточненное значение интеграла, – величина погрешности уточненного значения интеграла.

3. Вычислить значение интеграла с предельно возможной точностью. Для этого

выполнить вычислительный эксперимент: задавая последовательно значения 

равными, например, , ,…., найти предельную точность вычисления для интеграла . В каждом случае вычислить также реальную погрешность R, используя значение интеграла из п.1.

4. Оформить отчет по задаче.

Решение задачи 5.3:

П.1

Вычислим интеграл , используя средства языка python:

Для этого воспользуемся пакетом scipy:

import numpy as np

import scipy.integrate as integrate

def function(x):

return np.log(x\*\*2 - x)

I = integrate.quad(lambda x: np.log(x\*\*2 -x), 2, 4) # scipy интеграл через lambda где lambda- переменная интегрирования

print(I[0]) # эталонное значение интеграла

Результат: 3.4547199493640006

П.2

Найдем шаг при котором R<=eps согласно правилу Рунге.

import numpy as np

import scipy.integrate as integrate

def function(x):

return np.log(x\*\*2 - x)

def rect\_sum\_right(f, left\_border, right\_border, step): # // правые прямоугольники

seg\_num= int((right\_border - left\_border)/step)

y = [0 for i in range(seg\_num)]

for i in range(seg\_num): y[i] = f(left\_border + step \* i)

rect\_sum = 0

for i in range(seg\_num - 1): rect\_sum += y[i+1] \* step

return rect\_sum

step\_arr=np.ndarray(15) # //массив шагов

for i in range(15):

step\_arr[i]=10\*\*(-(i+1))

R=np.ndarray(15)

p=1

eps=10\*\*(-8)

for i in range(15):

Ih=rect\_sum\_right(lambda x: np.log(x\*\*2 -x),2,4,step\_arr[i])

Ih2=rect\_sum\_right(lambda x: np.log(x\*\*2 -x),2,4,step\_arr[i]/2)

R[i]=abs((Ih-Ih2)/(2\*\*p -1))

if(R[i]<eps):

step\_arr[i+1]=step\_arr[i]/2

break

print(R[i],eps,step\_arr[i+1])

Результат : Прогоняем массив шагов пока погрешность по Рунге не будет меньше eps, тогда берем шаг равный половине шага.

Формат(R,eps,h)

0.08002398396648269 1e-08 0.01

0.0079508637146124 1e-08 0.001

0.0007945707492620357 1e-08 0.0001

7.945191867264612e-05 1e-08 1e-05

2.036962981488699e-05 1e-08 1e-06

7.945135145348559e-07 1e-08 1e-07

7.945088142946588e-08 1e-08 1e-08

2.557312473784989e-09 1e-08 0.5e-08

h=0.5e-08

Посчитаем интеграл с таким шагом методом правых прямоугольников и выполним уточнение по Рунге:

Ih=rect\_sum\_right(lambda x: np.log(x\*\*2 -x),2,4,0.5\*10\*\*(-8))

Ih=3.454719939912535 , abs(I-Ih)= 9.451465832199801e-09

Уточнение по Рунге:

Rh=2.557312473784989e-09

Iуточненное=Ih+Rh(без модуля)= 3.454719939912535+2.557312473784989e-09=

3.4547199424698474

Сравним с I: Rh уточненное= abs(Iуточненное-I)= 6.894153248282464e-09 видно что погрешность уменьшилась(было 9.451465832199801e-09).

На этом П.2 завершен.

П.3

Вычислить значение интеграла с предельно возможной точностью. Для этого

выполнить вычислительный эксперимент: задавая последовательно значения 

равными, например, , ,…., найти предельную точность вычисления для интеграла . В каждом случае вычислить также реальную погрешность R, используя значение интеграла из п.1.

Таким образом мы знаем функцию,пределы интегрирования и погрешность(массив погрешностей от e-01 до e-15). Зная погрешность находим теоретический шаг(1D array),через него число делений+1(тоже 1D arr) и через число делений находим реальный шаг(1D arr):

Прогоняем(пока работает) формулу правых прямоугольников с таким массивом шагов и находим результат с наименьшей погрешностью:

Реализация на python:

import math as m

import numpy as np

import scipy.integrate as integrate

def function(x):

return np.log(x\*\*2 - x)

def rect\_sum\_right(f, left\_border, right\_border, step): # // правые прямоугольники

seg\_num= int((right\_border - left\_border)/step)

y = [0 for i in range(seg\_num)]

for i in range(seg\_num): y[i] = f(left\_border + step \* i)

rect\_sum = 0

for i in range(seg\_num - 1): rect\_sum += y[i+1] \* step

return rect\_sum

eps=np.ndarray(15)

for i in range(15): eps[i]=10\*\*(-(i+1))

#print(eps)

step\_arrT=np.ndarray(15)

for i in range(15): step\_arrT[i]=eps[i]/1.5

n=np.ndarray(15)

for i in range(15):

n[i]=int(3/eps[i])+1

step\_arrR=np.ndarray(15)

for i in range(15):

step\_arrR[i]=2/n[i]

#I = integrate.quad(lambda x: np.log(x\*\*2 -x), 2, 4)

I=3.4547199493640006

Ih=np.ndarray(15)

for i in range(15):

Ih[i]=rect\_sum\_right(lambda x: np.log(x\*\*2 -x),2,4,step\_arrR[i])

print('%.16f %.16f %.16f %d' % (float(I), float(Ih[i]), (abs(I - Ih[i])), i))

Результат:(I,Ih,delta,i)

3.4547199493640006 3.3518841795835712 0.1028357697804294 0

3.4547199493640006 3.4441582584559058 0.0105616909080948 1

3.4547199493640006 3.4536609171586505 0.0010590322053501 2

3.4547199493640006 3.4544483650988473 0.0002715842651533 3

3.4547199493640006 3.4547093558831210 0.0000105934808796 4

3.4547199493640006 3.4547188900128267 0.0000010593511739 5

3.4547199493640006 3.4547196777674811 0.0000002715965195 6

Далее вычисления невозможны ввиду крайне долгого времени ожидания