МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ Федеральное государственное автономное образовательное учреждение

высшего образования

«Национальный исследовательский

Нижегородский государственный университет им. Н.И. Лобачевского»

(ННГУ)

Институт информационных технологий, математики и механики

Направление подготовки: «Фундаментальная информатика и информационные технологии»

ОТЧЕТ

Методы численного интегрирования

Выполнил: студент группы 382006-2

Сухарев Артём Андреевич \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Эгамов Альберт Исмаилович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Нижний Новгород

2022 г.

Содержание

[Введение 3](#_Toc120655505)

[Постановка задачи 4](#_Toc120655506)

[Способ задания функции 5](#_Toc120655507)

[Формулы и методы численного интегрирования 7](#_Toc120655508)

[Формула Симпсона 7](#_Toc120655509)

[Формула «трех восьмых» 8](#_Toc120655510)

[Четырехинтервальная формула Боде 8](#_Toc120655511)

[Производство вычислений 10](#_Toc120655512)

[Представление результатов и используемые технические средства 11](#_Toc120655513)

[Результаты работы 13](#_Toc120655514)

[Литература 14](#_Toc120655515)

[Исходный код 15](#_Toc120655516)

# Введение

Задача численного интегрирования заключается в нахождении значения определенного интеграла, с помощью некоторых методов.

Необходимость применять подобное вычисление интеграла возникает в нескольких случаях: когда подынтегральная функция не задана аналитически, когда для данной функции невозможно найти первообразную для вычисления значения по формуле Ньютона-Лейбница или даже когда вычисление этой первообразной очень сложно и трудозатратно.

Сами же сферы применения настолько разнообразны (от применения в исследованиях и разного рода вычислений и симуляций в физике и химии, до некоторых аспектов экономики и анализа данных), что потребность в данных методах невозможно переоценить.

# Постановка задачи

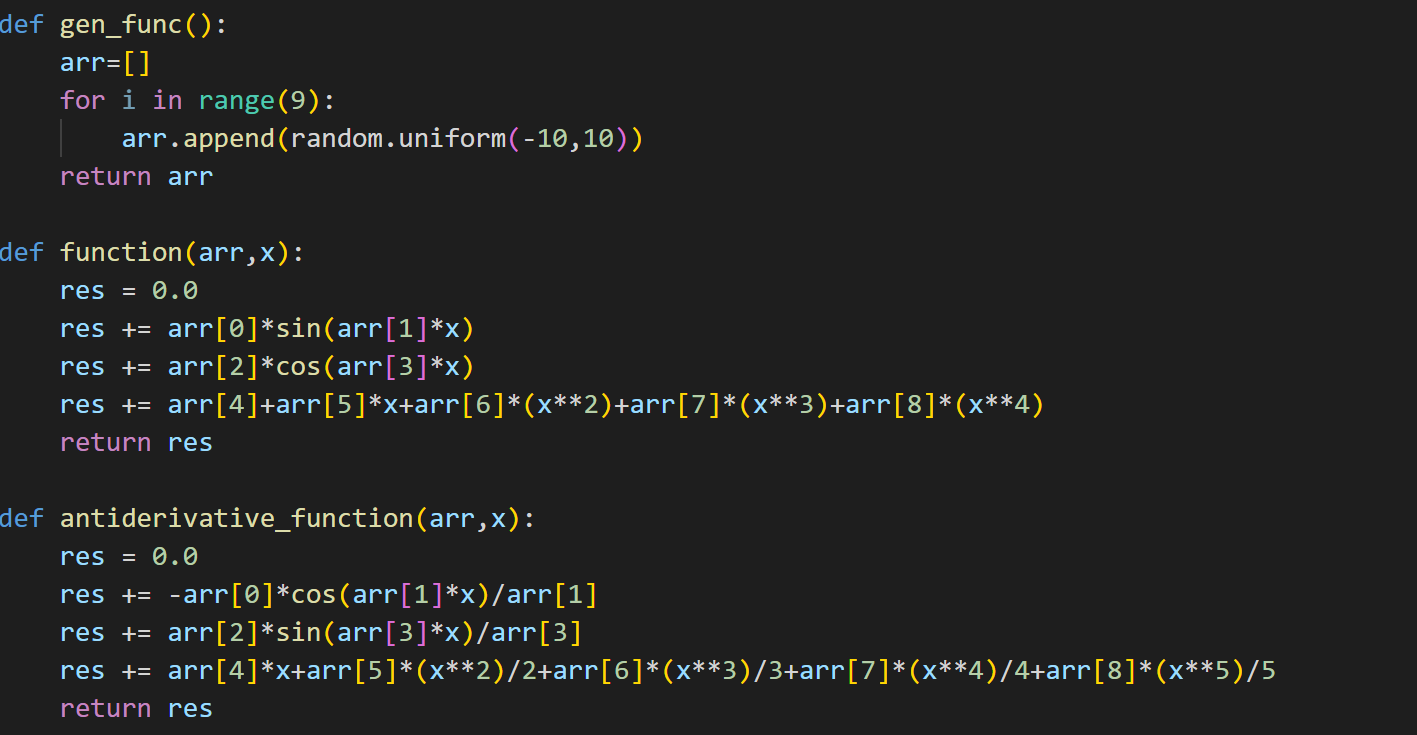
При выполнении данной лабораторной работы я хочу изучить три метода численного интегрирования: формула Симпсона(метод парабол), формула «трех восьмых»(частный случай метода Ньютона-Котеса) и четырехинтервальную формулу Боде - а так же какую точность могут обеспечить данные методы при вычислении ими реальных функций для разных разбиений отрезка интегрирования.

Для этого мне необходимо реализовать программу на языке программирования python, в которой для разного количества отрезков в разбиении я смогу оценить среднюю относительную погрешность на некотором наборе функций.

# Способ задания функции

Во-первых, когда мы определились, что мы будем рассматривать набор случайных функций, нам необходим способ их задания. В силу необязательности рассмотрения всех возможных функций, можно взять какой-нибудь шаблон функции с некоторым количеством параметров. Тем не менее для большей объективности, по возможности нужно скомбинировать сразу несколько. Для этих целей я возьму следующий шаблон:

В данном случае параметрами будут a,b,c,d,e,f,g,h,k (9 штук) и при разных значениях мы будем получать различные функции, и на разных участках x доминировать(задавать поведение) будут разные из них. В коде это можно реализовать следующим образом:

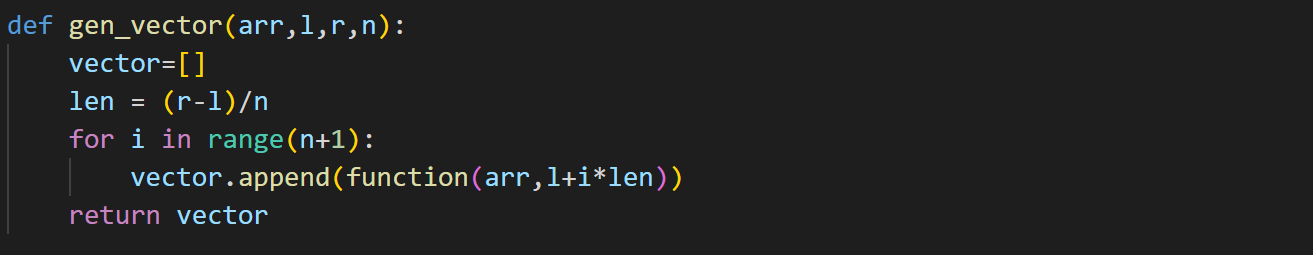


То есть функция будет задаваться набором из 9 чисел, и по данному набору мы легко можем вычислить значение функции и её первообразную в какой-то точке.

Во-вторых, в большинстве случаев нам дается не сама функция, а лишь набор ее точек (абсцисс и ординат), и для большей реалистичности численный интеграл нужно вычислять именно по сформированному набору точек.

Для создания этого набора, нам нужно определиться с количеством интервалов, которые мы будем рассматривать и границы самого отрезка. Не уменьшая общности можно рассматривать отрезок [-40,40]. Но проблема может возникнуть с количеством интервалов, не вдаваясь пока в подробности (это будет представлено далее) можно сказать, что одно вычисление для каждого метода учитывает разное число интервалов (2, 3 или 4), поэтому нам нужно брать только такое количество интервалов, которое кратно всем. Несложно заметить, что для этого нужно брать только кратные 12 количества (12, 24, 36 и т.д.).

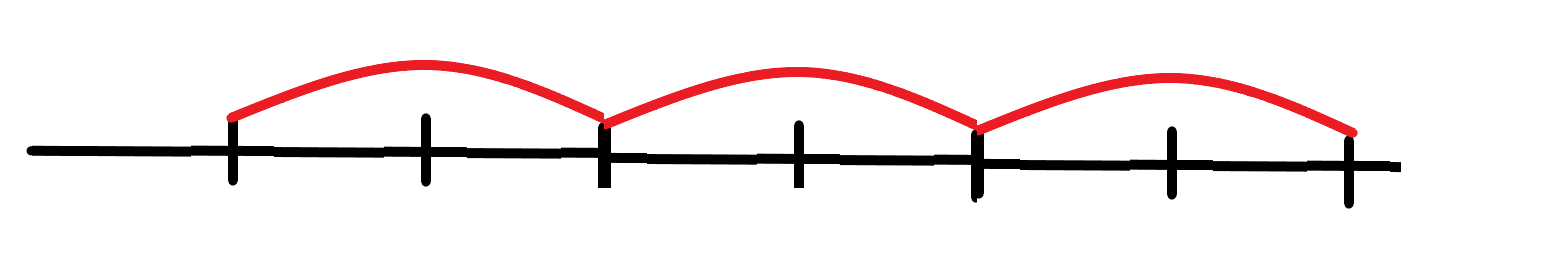
Когда же мы определились со всеми нужными значениями, можно преступить к вычислению, а конкретно если границы отрезка [l,r], а число интервалов n, то точки будут иметь следующие значения абсцисс:



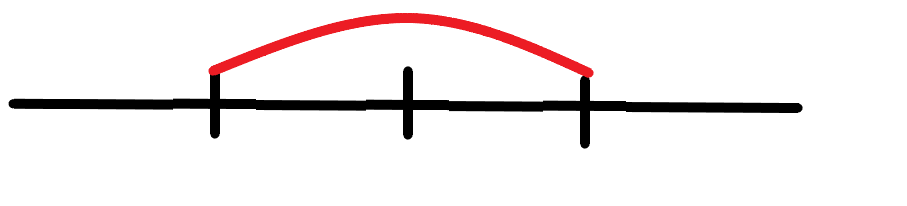
# Формулы и методы численного интегрирования

## Формула Симпсона

Этот метод подсчета является двухинтервальным, то есть для нахождения значения нужно разбить отрезок на два интервала тремя точками. Поскольку нам может быть дано большее число интервалов и точек, то их можно разбить на блоки по нужному нам количеству и считать на каждом блоке по 2 интервала отдельно, а позже просуммировать полученные значения.

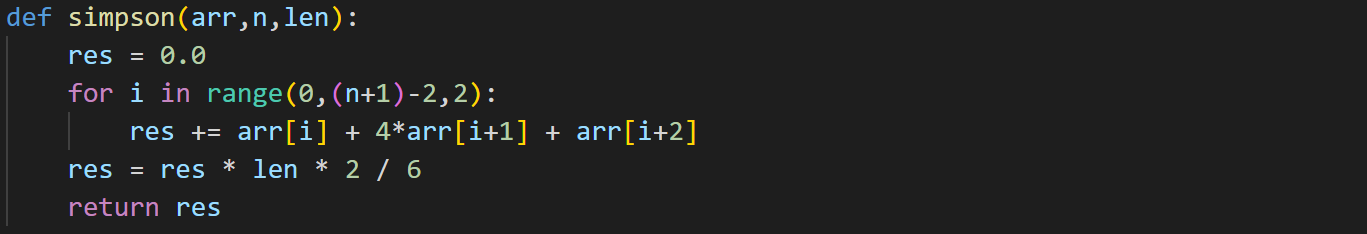


Теперь рассмотрим один блок, состоящий из двух интервалов:



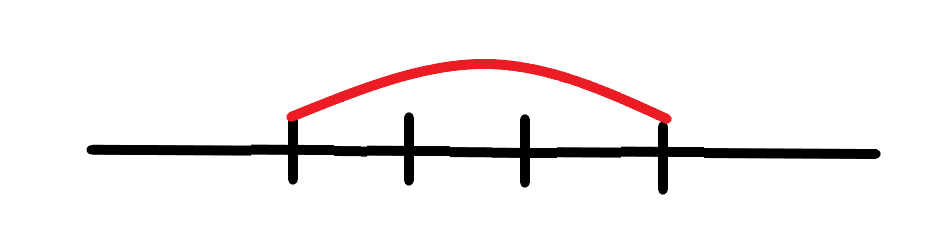
Пусть у нас есть 3 точки и посчитанные в них значения . Тогда интеграл в этом блоке можно посчитать следующим образом:

Сумма по всем блокам и даст нам интеграл, который мы хотим найти:



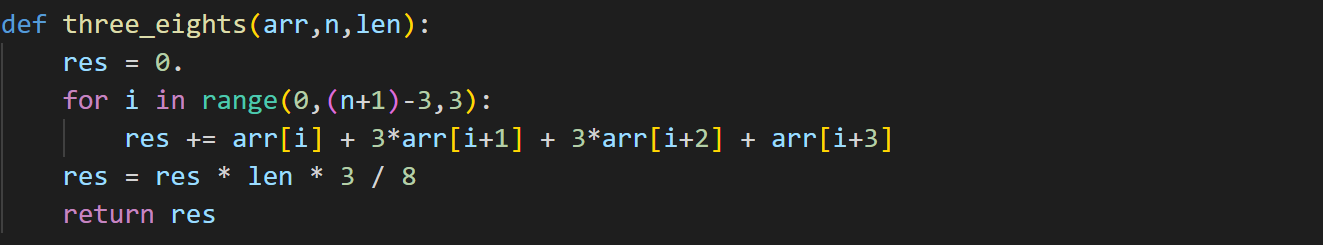
## Формула «трех восьмых»

Это трехинтервальная формула на четырехточечном шаблоне. В данном случае блок будет рассматриваться из трех интервалов.



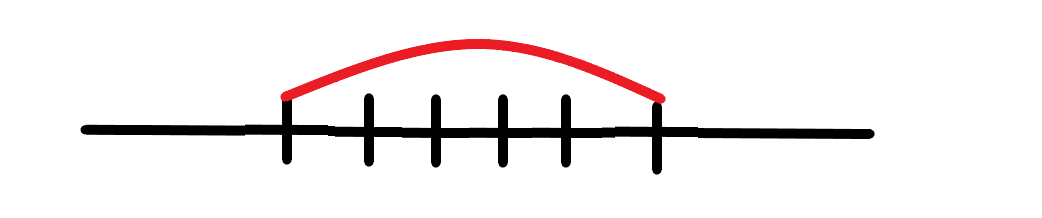
Итак теперь у нас есть 4 точки и посчитанные в них значения . Тогда интеграл в этом блоке можно посчитать следующим образом:

А также поскольку количество блоков изменится, то и сумма тоже:



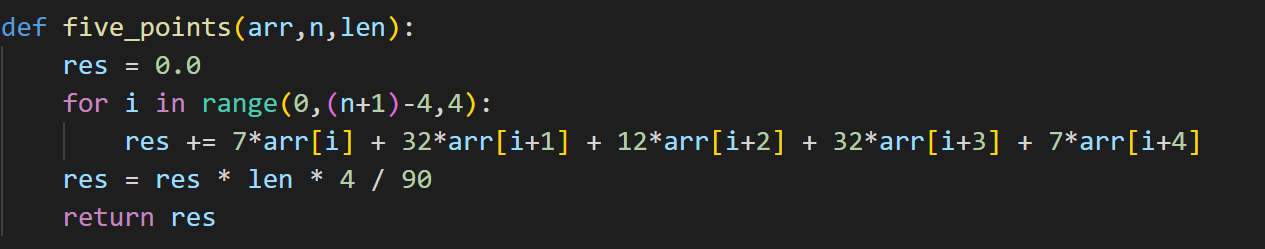
## Четырехинтервальная формула Боде

Это четырехинтервальная формула на пятиточечном шаблоне. То есть в данном случае блок будет рассматриваться из четырех интервалов.



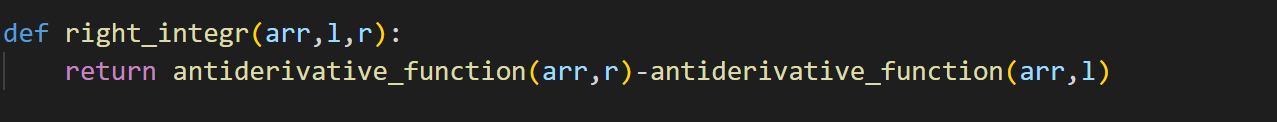
В данной формуле у нас есть 5 точек и посчитанные в них значения . Тогда интеграл в этом блоке можно посчитать следующим образом:

Аналогично изменится и сумма:

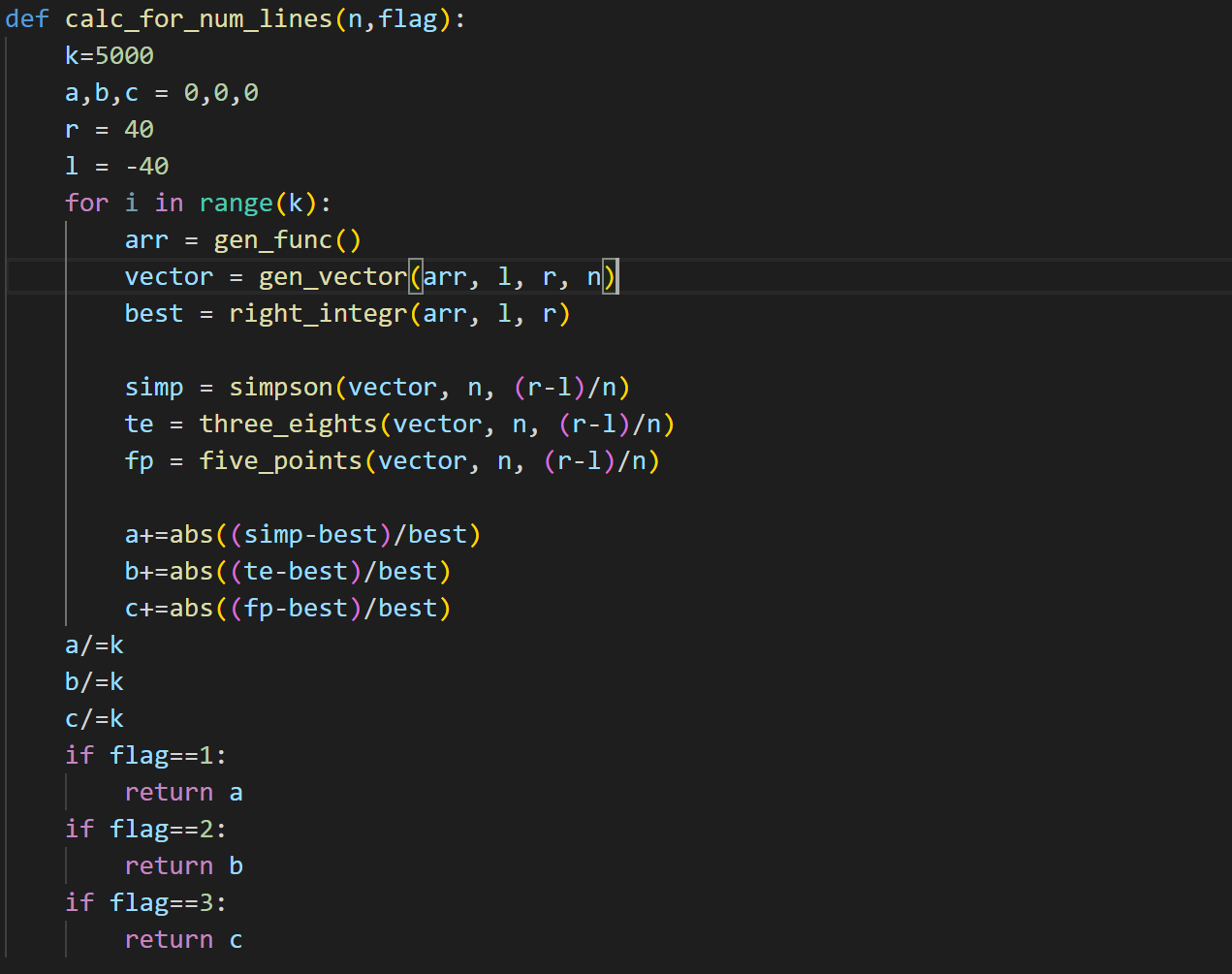


# Производство вычислений

Для проведения сверок с верным значением определенного интеграла нам необходимо его вычислить, а так как у нас определена функция вычисления первообразной в точке, то можно вычислить интеграл с помощью формулы Ньютона-Лейбница.



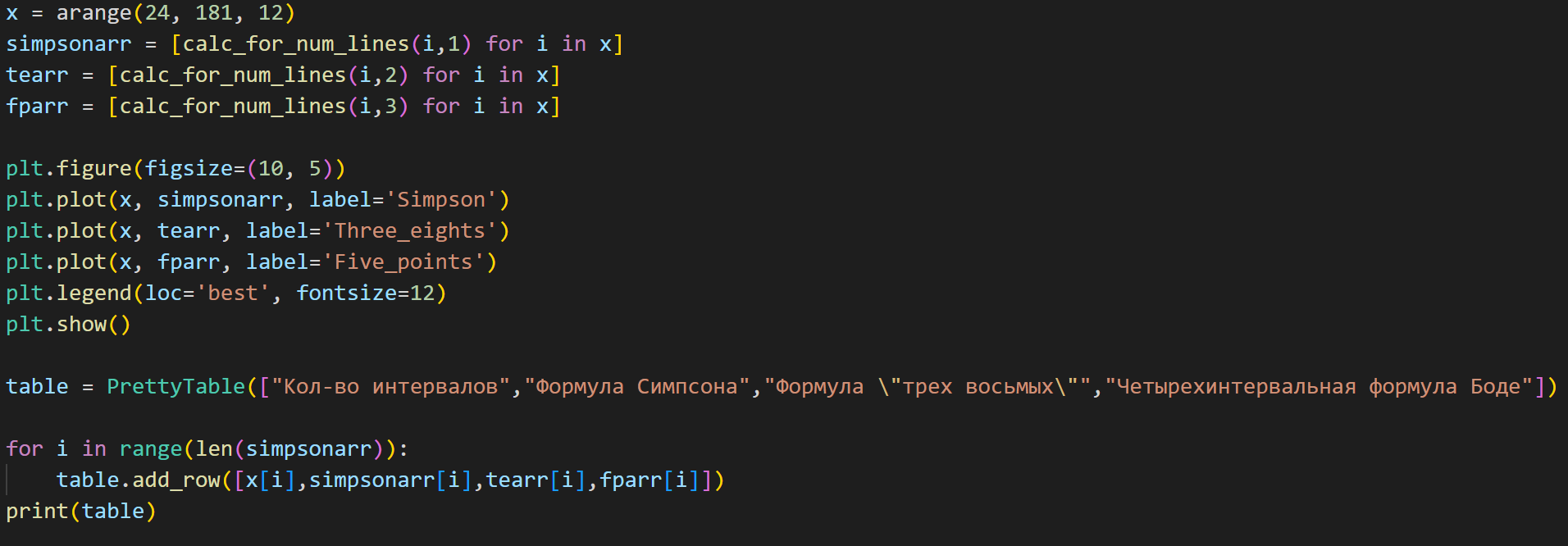
Теперь можем создать функцию, которая проводит серию тестов и вычисляет среднюю относительную погрешность. Создадим выборку из 5000 тестовых случайных функций и посчитаем эту погрешность:



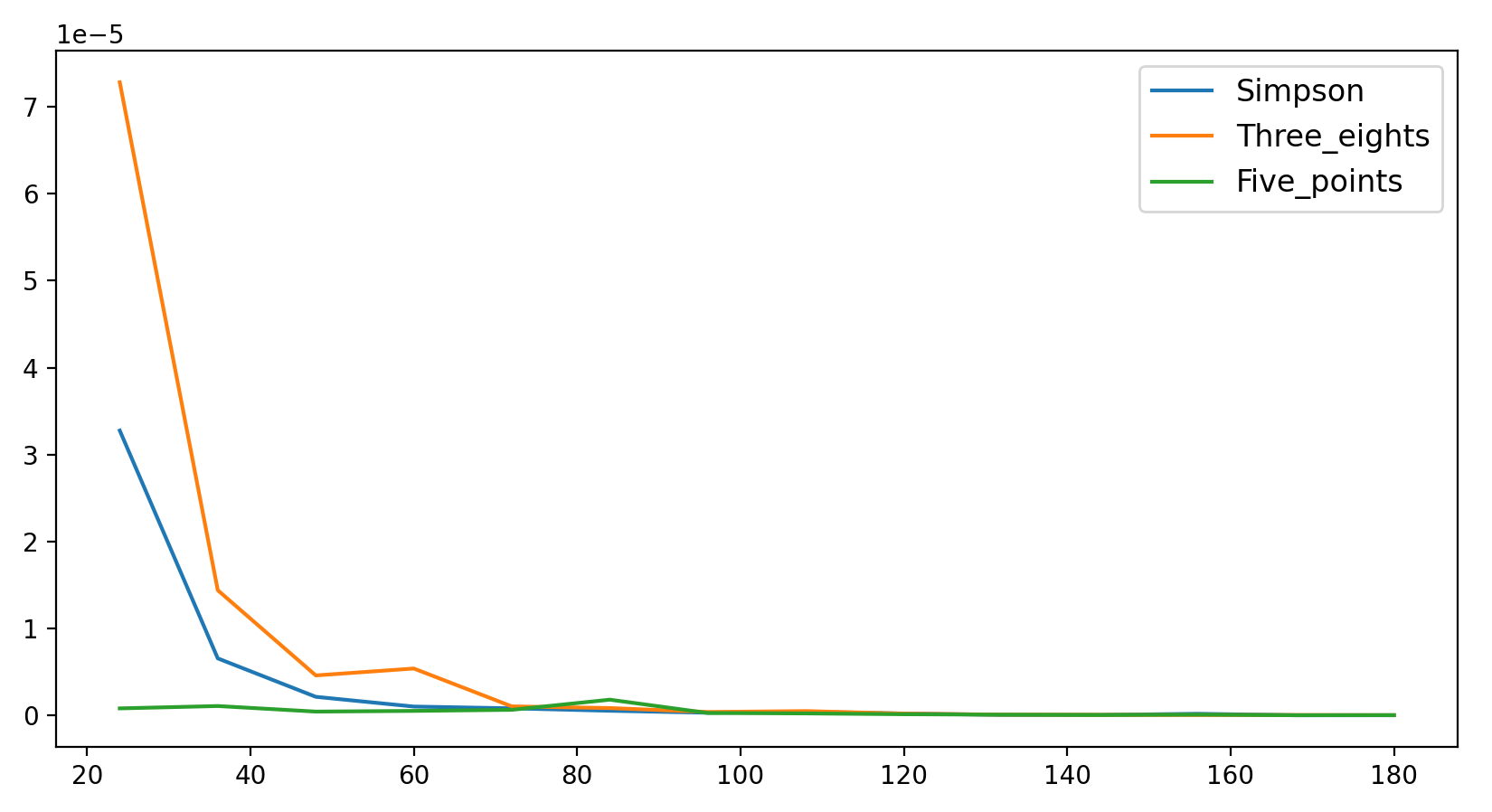
На вход функции так же подадим специальный флаг, который будет показывать какой именно из результатов нам нужен.

# Представление результатов и используемые технические средства

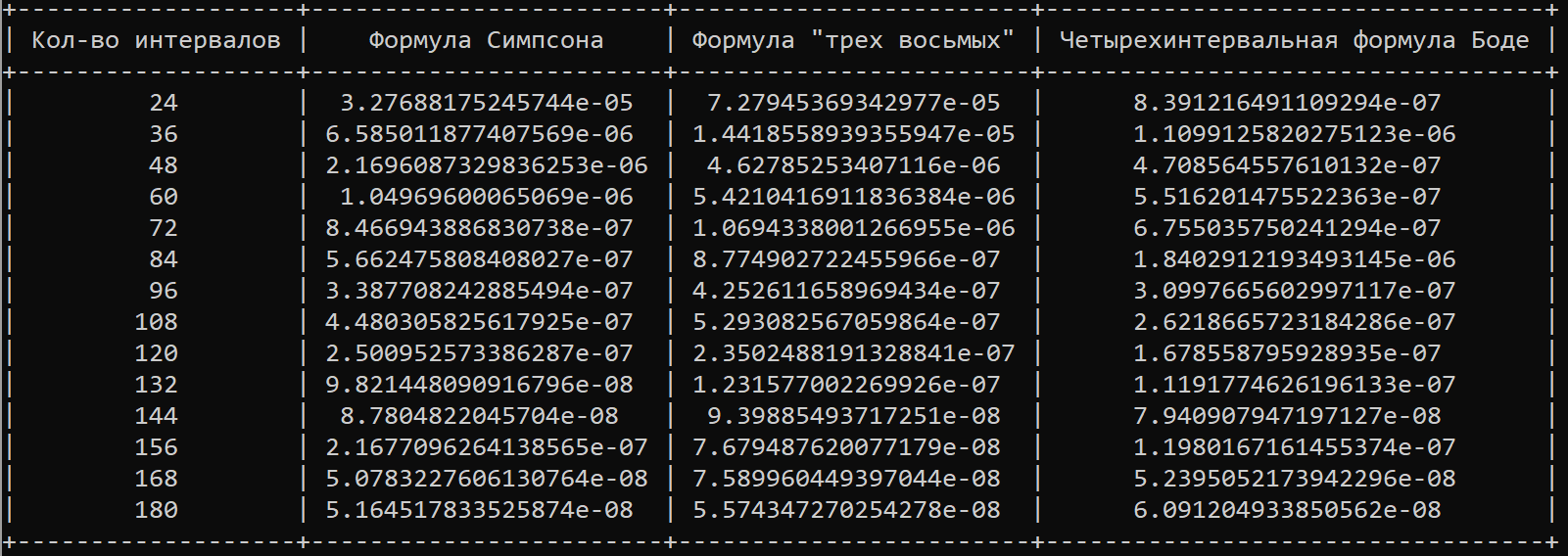
Для визуализации результатов я загнал полученные данные в виде графика и таблицы.



При запуске программы будет показан график с полученными значениями средней относительной погрешности для каждого метода.



Так же данные будут выведены в таблицу, так как график не может в полном масштабе отобразить погрешности при больших количествах интервалов.



Так же стоит заметить, что для получения случайных значений, создания таблицы и графика я использовал дополнительные модули. А именно модули numpy, math, matplotlib и prettytable, поэтому при необходимости запуска стоит убедится, что они у вас установлены.

Так же стоит учесть, что набор функций для тестов большой и для выполнения программы нужно определенное время. На моем компьютере это занимает около минуты(если вам нужна быстрая программа снизьте количество тестов, то есть значение k).

# Результаты работы

По проведенным результатам можно увидеть, что даже при небольших количествах интервалов и точек можно достичь хорошей оценки интеграла.

Так же по данным таблицы и графика можно увидеть, что формула Боде дает меньшую погрешность по сравнению с формулой Симпсона, а формула Симпсона лучшую погрешность по сравнению с методом «трех восьмых». Причем данная тенденция распространяется при всех количествах интервалов.

Отсюда можно сделать вывод, что самый вариант из этих трех это четырехинтервальная формула Боде.

# Литература

1. Лекция. Методы численного интегрирования. <http://dep805.ru/education/portal/4/to/l13_4tochm.pdf>
2. <https://cyclowiki.org/wiki/Формула_трёх_восьмых>
3. <https://ru.wikipedia.org/wiki/Численное_интегрирование>

# Исходный код

from numpy import random,arange

from math import sin,cos

import matplotlib.pyplot as plt

from prettytable import PrettyTable

# function = arr[0] \* sin(arr[1]\*x) + arr[2] \* cos(arr[3]\*x) +

# + arr[4] + arr[5]\*x + arr[6]\*x^2 + arr[7]\*x^3 + arr[8]\*x^4

def gen\_func():

arr=[]

for i in range(9):

arr.append(random.uniform(-10,10))

return arr

def function(arr,x):

res = 0.0

res += arr[0]\*sin(arr[1]\*x)

res += arr[2]\*cos(arr[3]\*x)

res += arr[4]+arr[5]\*x+arr[6]\*(x\*\*2)+arr[7]\*(x\*\*3)+arr[8]\*(x\*\*4)

return res

def antiderivative\_function(arr,x):

res = 0.0

res += -arr[0]\*cos(arr[1]\*x)/arr[1]

res += arr[2]\*sin(arr[3]\*x)/arr[3]

res += arr[4]\*x+arr[5]\*(x\*\*2)/2+arr[6]\*(x\*\*3)/3+arr[7]\*(x\*\*4)/4+arr[8]\*(x\*\*5)/5

return res

def gen\_vector(arr,l,r,n):

vector=[]

len = (r-l)/n

for i in range(n+1):

vector.append(function(arr,l+i\*len))

return vector

def right\_integr(arr,l,r):

return antiderivative\_function(arr,r)-antiderivative\_function(arr,l)

def simpson(arr,n,len):

res = 0.0

for i in range(0,(n+1)-2,2):

res += arr[i] + 4\*arr[i+1] + arr[i+2]

res = res \* len \* 2 / 6

return res

def three\_eights(arr,n,len):

res = 0.

for i in range(0,(n+1)-3,3):

res += arr[i] + 3\*arr[i+1] + 3\*arr[i+2] + arr[i+3]

res = res \* len \* 3 / 8

return res

def five\_points(arr,n,len):

res = 0.0

for i in range(0,(n+1)-4,4):

res += 7\*arr[i] + 32\*arr[i+1] + 12\*arr[i+2] + 32\*arr[i+3] + 7\*arr[i+4]

res = res \* len \* 4 / 90

return res

def calc\_for\_num\_lines(n,flag):

k=5000

a,b,c = 0,0,0

r = 40

l = -40

for i in range(k):

arr = gen\_func()

vector = gen\_vector(arr, l, r, n)

best = right\_integr(arr, l, r)

simp = simpson(vector, n, (r-l)/n)

te = three\_eights(vector, n, (r-l)/n)

fp = five\_points(vector, n, (r-l)/n)

a+=abs((simp-best)/best)

b+=abs((te-best)/best)

c+=abs((fp-best)/best)

a/=k

b/=k

c/=k

if flag==1:

return a

if flag==2:

return b

if flag==3:

return c

x = arange(24, 181, 12)

simpsonarr = [calc\_for\_num\_lines(i,1) for i in x]

tearr = [calc\_for\_num\_lines(i,2) for i in x]

fparr = [calc\_for\_num\_lines(i,3) for i in x]

plt.figure(figsize=(10, 5))

plt.plot(x, simpsonarr, label='Simpson')

plt.plot(x, tearr, label='Three\_eights')

plt.plot(x, fparr, label='Five\_points')

plt.legend(loc='best', fontsize=12)

plt.show()

table = PrettyTable(["Кол-во интервалов","Формула Симпсона","Формула \"трех восьмых\"","Четырехинтервальная формула Боде"])

for i in range(len(simpsonarr)):

table.add\_row([x[i],simpsonarr[i],tearr[i],fparr[i]])

print(table)