Министерство науки и высшего образования Российской Федерации

Федерального государственного бюджетного образовательного учреждения высшего образования

«Казанский национальный исследовательский технологический университет»

(ФГБОУ ВО «КНИТУ»)

Кафедра интеллектуальных систем и управления информационными ресурсами

**ЛАБОРАТОРНАЯ РАБОТА №5**

по курсу  
«**Методы вычисления»**

на тему

**«Интегрирование**»

Выполнил:

студент группы 4321-23

Мвелва К.Ч.

Проверил:

доцент кафедры ИСУИР

Вотякова Л.Р.

Казань, 2024

Contents

[Задания 3](#_Toc183996750)

[Переменные: 3](#_Toc183996751)

[Решение 1: 4](#_Toc183996752)

[Задача 1: 4](#_Toc183996753)

[Метод трапеций 4](#_Toc183996754)

[Метод Симпсона 5](#_Toc183996755)

[Задача 2: 6](#_Toc183996756)

[Метод трапеций 6](#_Toc183996757)

[Метод Симпсона 7](#_Toc183996758)

[Решение 2: 8](#_Toc183996759)

[Задача 1: 8](#_Toc183996760)

[Задача 2: 11](#_Toc183996761)

# Задания

## Переменные:

β - 1 ϑ - 3

γ - 2 μ - 0

ν - 7 α – 4

1. Взяв шаг, равный одной десятой длины интервала интегрирования, вычислить определенный интервал по обобщенным формулам трапеций и Симпсона:

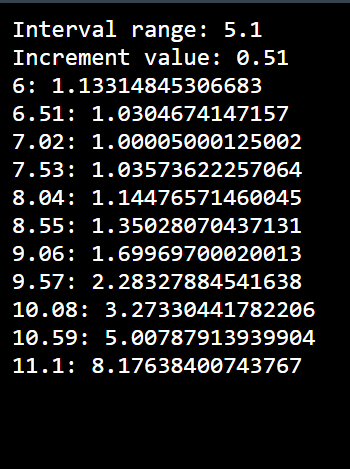
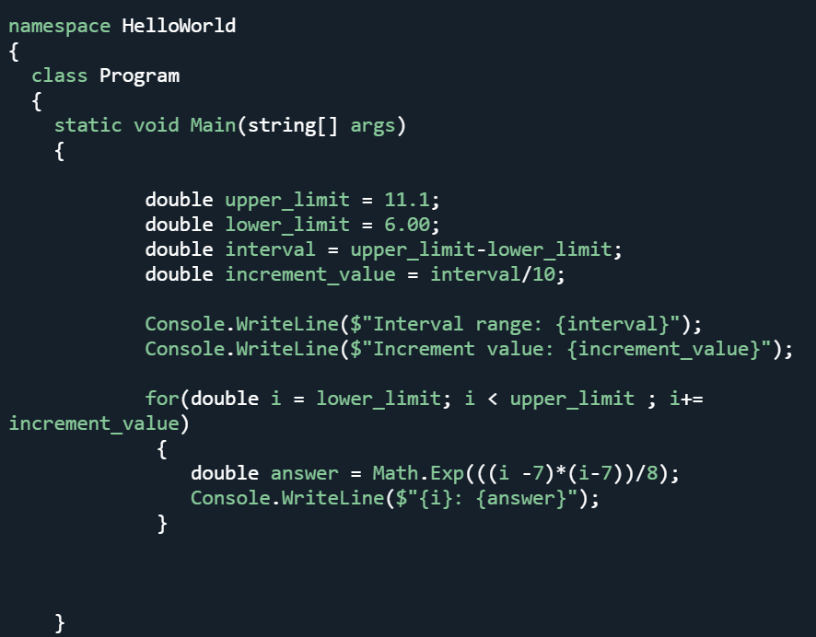
Тот же интеграл вычислить средствами Python с помощью одной из приведенных в пособии специальных функций.

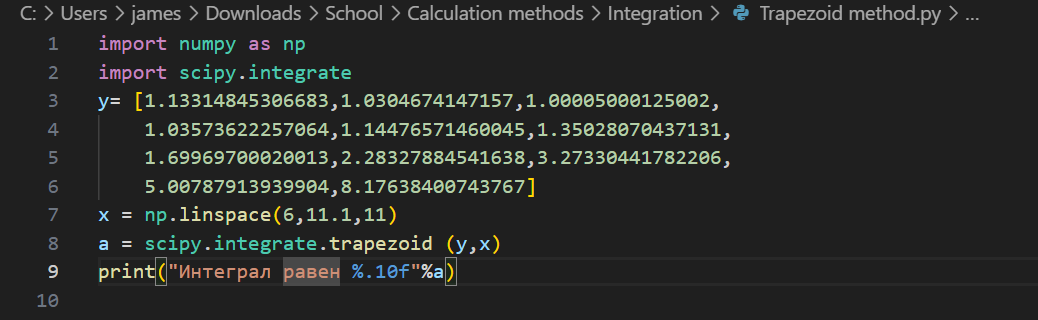
1. Составить программу для вычисления интеграла из задания 1 по обобщенным формулам прямоугольников, трапеций и Симпсона.

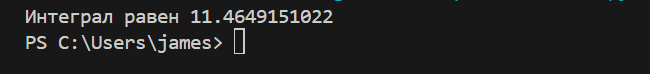
# Решение 1:

## Задача 1:

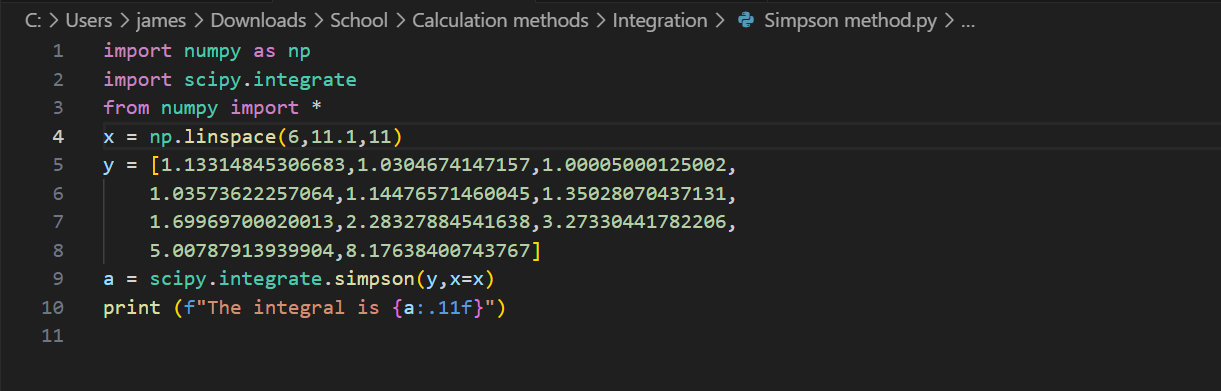
### Метод трапеций

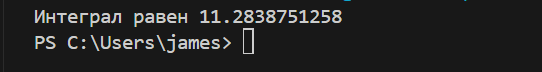






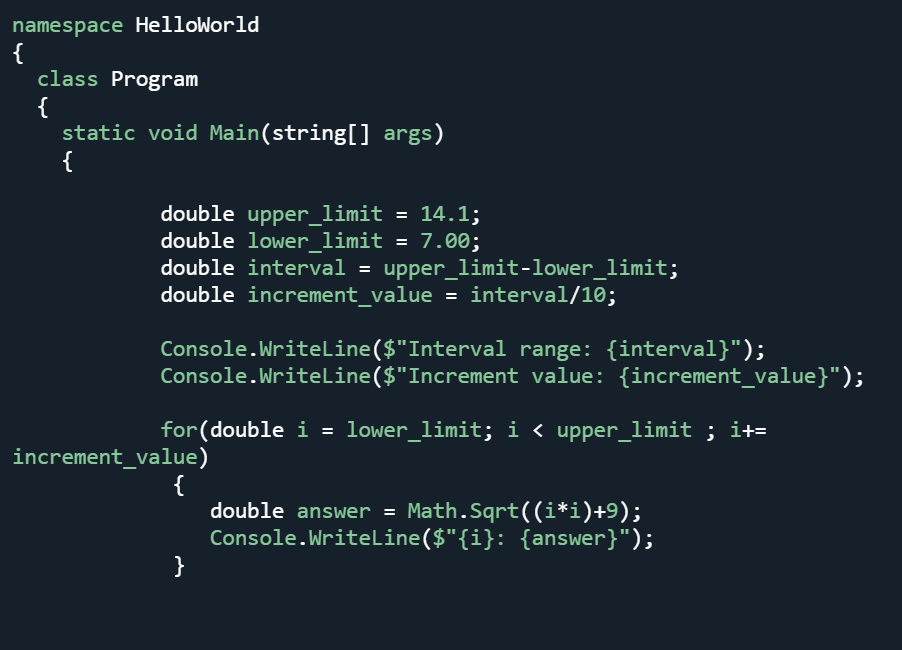
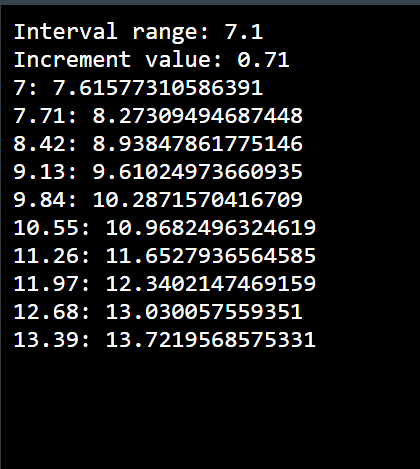
### Метод Симпсона

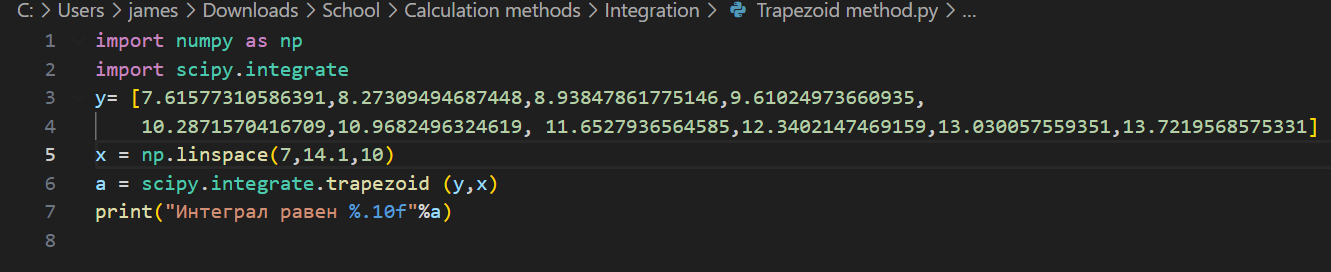


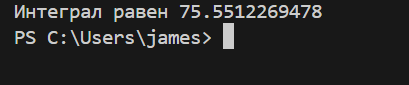


## Задача 2:

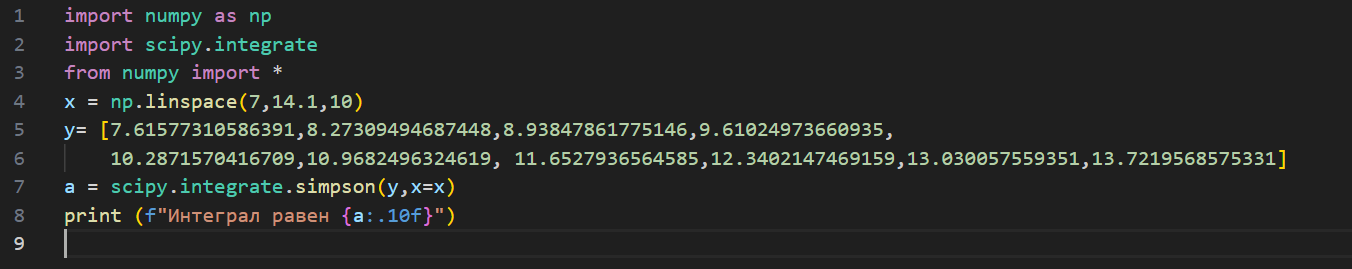
### Метод трапеций





### Метод Симпсона





# Решение 2:

## Задача 1:

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import simpson

#Rectangular method

def rectangular\_rule(f\_values, dx, method = "midpoint"):

    if method == "left":

        return np.sum(f\_values[:-1])\*dx

    elif method == "right":

        return np.sum(f\_values[1:])\*dx

    elif method == "midpoint":

        midpoints = (f\_values[:-1]+f\_values[1:])/2

        return np.sum(midpoints)\*dx

    else:

        raise ValueError("Method must be 'left','right' or 'midpoint'")

#Trapezium rule

def trapezium\_rule(f\_values,dx):

    return (dx/2)\*(f\_values[0]+2\*np.sum(f\_values[1:-1]+f\_values[-1]))

a = 6

b = 11.1

n = 10

x\_values = np.linspace(a,b,n+1)

f\_values = np.exp(((x\_values-7)\*\*2)/8)

dx = (b-a)/n

rect\_left = rectangular\_rule(f\_values,dx,method="left")

rect\_right = rectangular\_rule(f\_values,dx,method="right")

rect\_midpoint = rectangular\_rule(f\_values,dx,method="midpoint")

trap\_result = trapezium\_rule(f\_values, dx)

simp\_result = simpson(f\_values, x=x\_values)

print("Integration Results:")

print(f"Rectangular Rule(Left):{rect\_left:.10f}")

print(f"Rectangular Rule(Right):{rect\_right:.10f}")

print(f"Rectangular Rule(Midpoint):{rect\_midpoint:.10f}")

print(f"Trapezium Rule:{rect\_left:.10f}")

print(f"Simpson's Rule:{rect\_left:.10f}")

fig, ax = plt.subplots(figsize= (10,6))

#Our function

x\_fine = np.linspace(a,b,500)

ax.plot(x\_fine, np.exp(((x\_fine-7)\*\*2)/8),label = "e^((x\_values-7)\*\*2)/8)", color="blue")

#Rectangle function

for i in range(n):

    x\_start = x\_values[i]

    x\_end = x\_values[i+1]

    y\_mid = (f\_values[i] + f\_values[i+1])/2

    ax.fill\_between([x\_start,x\_end],[y\_mid,y\_mid], color = "green", alpha = 0.3, step="mid")

#trapezoid function

for i in range (n):

    ax.plot([x\_values[i], x\_values[i+1]],[f\_values[i],f\_values[i+1]],color = "orange",alpha = 0.8)

#Customize the plot

ax.set\_title("Integration Approximation Methods")

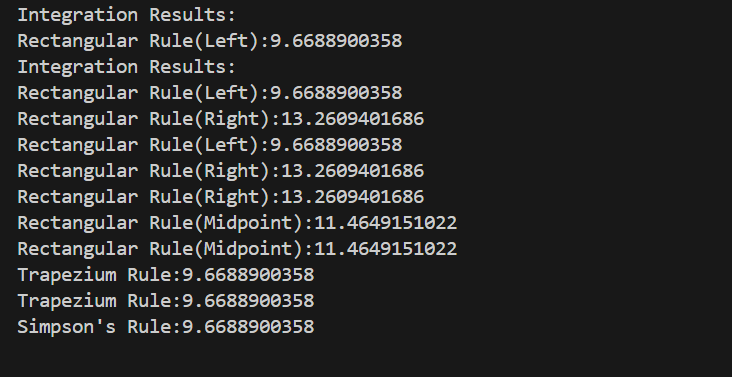
ax.set\_xlabel("X")

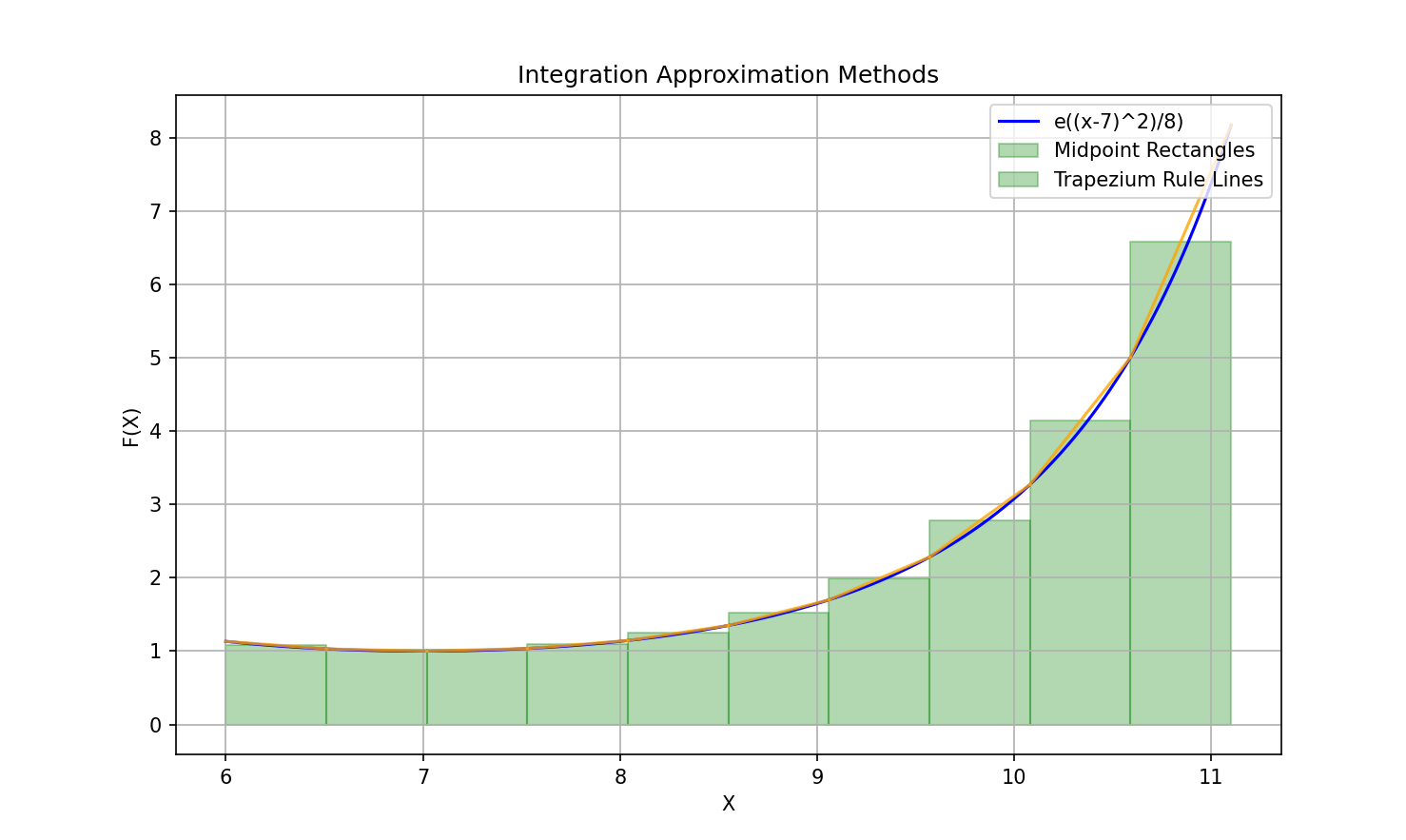
ax.set\_ylabel("F(X)")

ax.legend(["e((x-7)^2)/8)", "Midpoint Rectangles","Trapezium Rule Lines"], loc = "upper right")

plt.grid()

plt.show()





## Задача 2:

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import simpson

#Rectangular method

def rectangular\_rule(f\_values, dx, method = "midpoint"):

    if method == "left":

        return np.sum(f\_values[:-1])\*dx

    elif method == "right":

        return np.sum(f\_values[1:])\*dx

    elif method == "midpoint":

        midpoints = (f\_values[:-1]+f\_values[1:])/2

        return np.sum(midpoints)\*dx

    else:

        raise ValueError("Method must be 'left','right' or 'midpoint'")

#Trapezium rule

def trapezium\_rule(f\_values,dx):

    return (dx/2)\*(f\_values[0]+2\*np.sum(f\_values[1:-1]+f\_values[-1]))

a = 7

b = 14.1

n = 10

x\_values = np.linspace(a,b,n+1)

f\_values = np.sqrt((x\_values)\*\*2+9)

dx = (b-a)/n

rect\_left = rectangular\_rule(f\_values,dx,method="left")

rect\_right = rectangular\_rule(f\_values,dx,method="right")

rect\_midpoint = rectangular\_rule(f\_values,dx,method="midpoint")

trap\_result = trapezium\_rule(f\_values, dx)

simp\_result = simpson(f\_values, x=x\_values)

print("Integration Results:")

print(f"Rectangular Rule(Left):{rect\_left:.10f}")

print(f"Rectangular Rule(Right):{rect\_right:.10f}")

print(f"Rectangular Rule(Midpoint):{rect\_midpoint:.10f}")

print(f"Trapezium Rule:{rect\_left:.10f}")

print(f"Simpson's Rule:{rect\_left:.10f}")

fig, ax = plt.subplots(figsize= (10,6))

#Our function

x\_fine = np.linspace(a,b,11)

ax.plot(x\_fine,np.sqrt((x\_fine)\*\*2 +9),label = "sqrt((x\_values)\*\*2+9)", color="blue")

#Rectangle function

for i in range(n):

    x\_start = x\_values[i]

    x\_end = x\_values[i+1]

    y\_mid = (f\_values[i] + f\_values[i+1])/2

    ax.fill\_between([x\_start,x\_end],[y\_mid,y\_mid], color = "green", alpha = 0.3, step="mid")

#trapezoid function

for i in range (n):

    ax.plot([x\_values[i], x\_values[i+1]],[f\_values[i],f\_values[i+1]], color = "orange",alpha = 0.3)

#Customize the plot

ax.set\_title("Integration Approximation Methods")

ax.set\_xlabel("X")

ax.set\_ylabel("F(X)")

ax.legend(["sqrt((x\_values)^2+9)", "Midpoint Rectangles","Trapezium Rule Lines"], loc = "upper right")

plt.grid()

plt.show()

