Математический анализ

Лабораторная работа №1

Вариант №1

Выполнил:

Беляков Д. В.

Группа P3110

Преподаватель:

Фотин А. Д.

Оглавление

[Аналитическая часть 2](#_Toc100522837)

[Доказательство существования 2](#_Toc100522838)

[Получение интегральной суммы 3](#_Toc100522839)

[Значение интеграла, найденное по формуле Ньютона-Лейбница 4](#_Toc100522840)

[Численный метод интегрирования 4](#_Toc100522841)

[Метод прямоугольников 4](#_Toc100522842)

[Результаты работы программы 4](#_Toc100522843)

[Скриншоты программы 5](#_Toc100522844)

[Код программы 8](#_Toc100522845)

[Метод прямоугольников 9](#_Toc100522846)

[Код 9](#_Toc100522847)

[График 9](#_Toc100522848)

[Полученные результаты 9](#_Toc100522849)

# Аналитическая часть

## Доказательство существования

Докажем существование через критерий Римана в терминах сумм Дарбу

Так как на отрезке возрастающая, то верхняя и нижняя суммы определяются через правые и левые концы отрезков соответственно.

Возьмем равномерное разбиение, значит

## Получение интегральной суммы

Равномерное разбиение

Рассмотрим правые концы отрезков

## Значение интеграла, найденное по формуле Ньютона-Лейбница

Предел интегральной суммы равен значению интеграла, найденного по формуле, значит всё посчитано верно.

# Численный метод интегрирования

## Метод прямоугольников

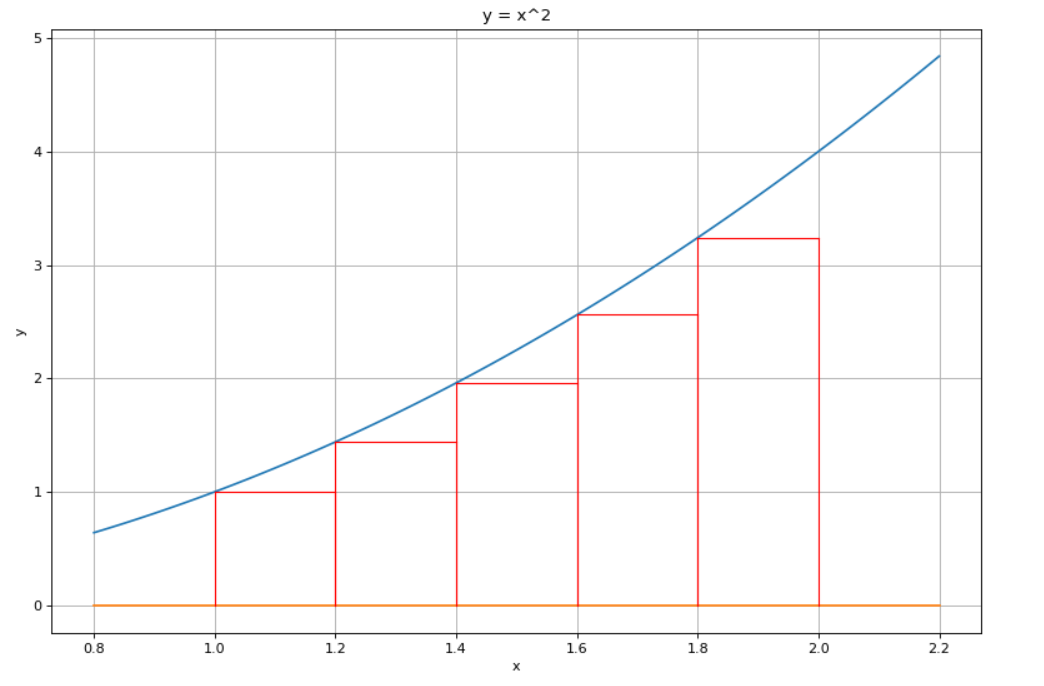
Метод прямоугольников — метод численного интегрирования функции одной переменной, заключающийся в замене подынтегральной функции на многочлен нулевой степени, то есть константу, на каждом элементарном отрезке. Если рассмотреть график подынтегральной функции, то метод будет заключаться в приближённом вычислении площади под графиком суммированием площадей конечного числа прямоугольников, ширина которых будет определяться расстоянием между соответствующими соседними узлами интегрирования, а высота — значением подынтегральной функции в этих узлах.

## Результаты работы программы

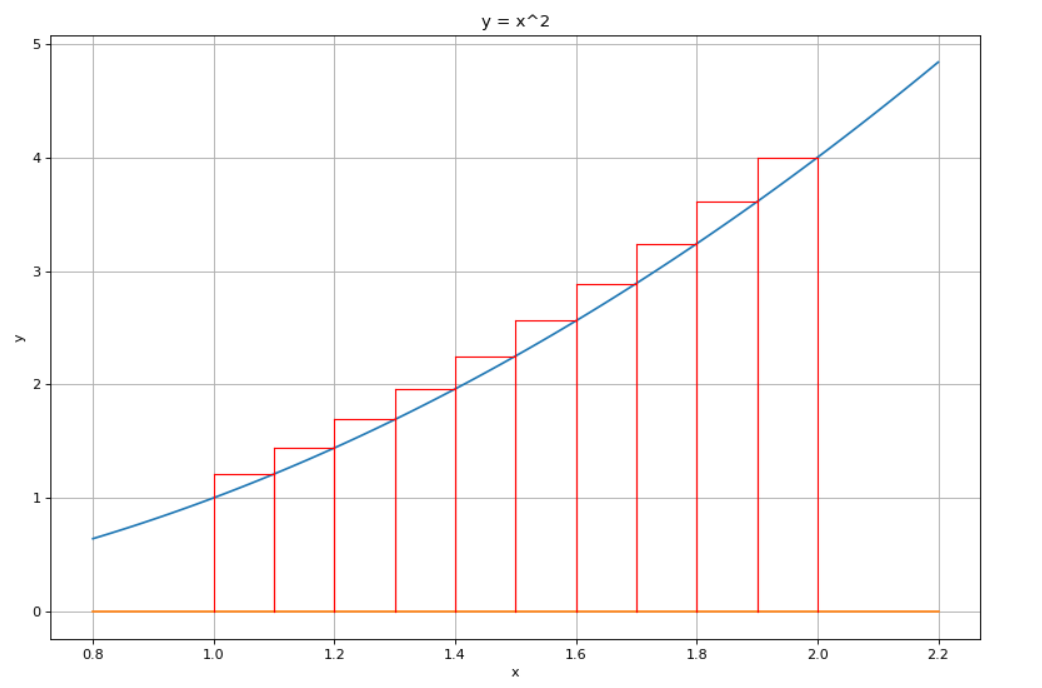
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Шаг интегрирования (h)** | **Численное решение (F)** | **Погрешность** | **Тип оснащения** | **Количество итераций (N)** |
| 0.2 | 2.04 | 0.29(3) | Левые | 5 |
| 2.64 | 0.30(6) | Правые |
| 2.33 | 0.00(3) | Средние |
| 2.25863 | 0.0747031 | Случайные |
| 0.1 | 2.185 | 0.148(3) | Левые | 10 |
| 2.485 | 0.151(6) | Правые |
| 2.3325 | 0.0008(3) | Средние |
| 2.35659 | 0.0232534 | Случайные |
| 0.05 | 2.25875 | 0.07458(3) | Левые | 20 |
| 2.40875 | 0.07541(6) | Правые |
| 0.01 | 2.333325 | 8.(3) \* 10-6 | Средние | 100 |
| 2.333378 | 4.52566 \* 10-5 | Случайные |
| 0.001 | 2.3318335 | 0.0014998(3) | Левые | 1000 |
| 2.3348335 | 0.0015001(6) | Правые |
| 0.0001 | 2.3333333325 | 8.335 \* 10-10 | Средние | 104 |
| 2.3333343058 | 9.725 \* 10-7 | Случайные |
| 0.00001 | 2.3333183 | 1.45 \* 10-5 | Левые | 105 |
| 2.3333483 | 1.5 \* 10-5 | Правые |
| 0.000001 | 2.333333333196 | 1.3716 \* 10-10 | Средние | 106 |
| 2.333333334242 | 9.0823 \* 10-10 | Случайные |

# Скриншоты программы

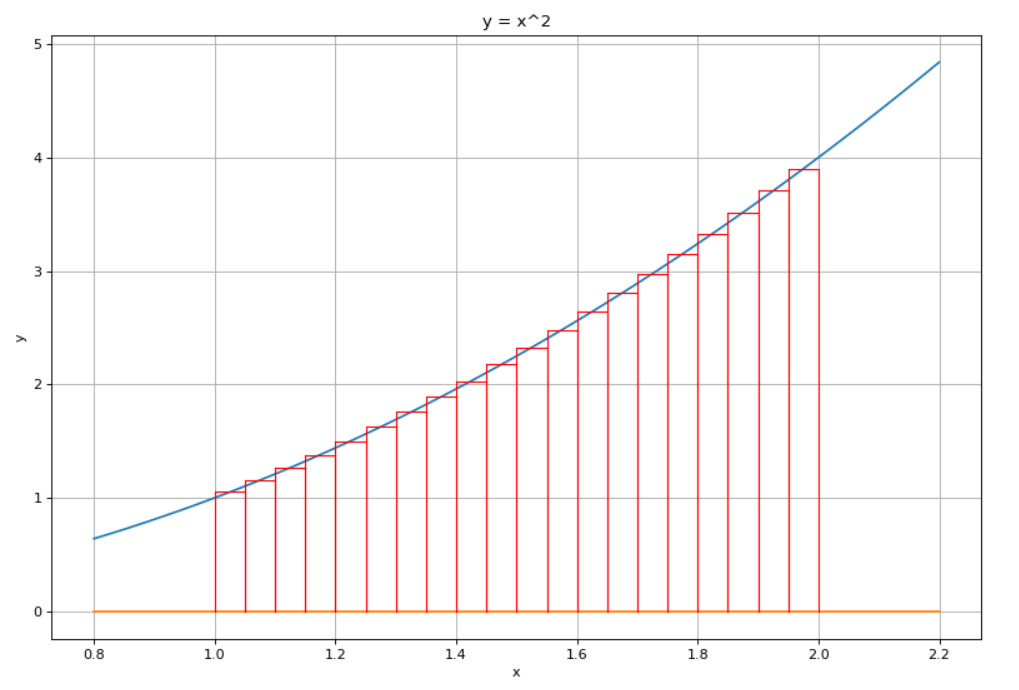
1. График, построенный для n = 5 разбиений с оснащением по левому концу отрезка.



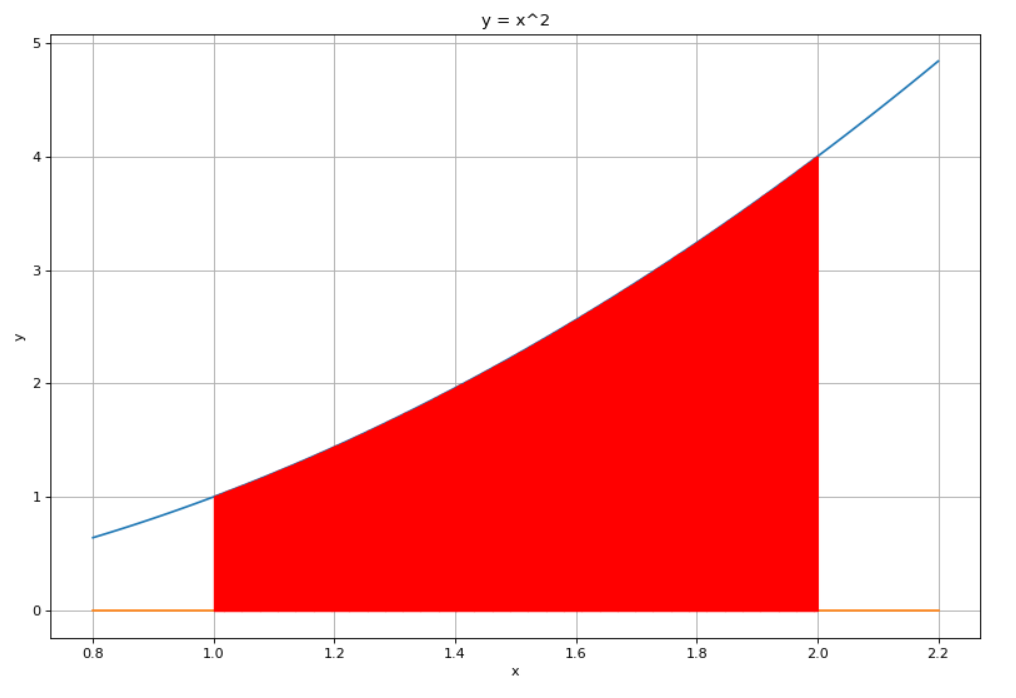
1. График, построенный для n = 10 разбиений с оснащением по правому концу отрезка.



1. График, построенный для n = 20 разбиений с оснащением по середине отрезка.



1. График, построенный для n = 1000 разбиений с оснащением по случайной точке отрезка.



# Код программы

%matplotlib inline

**import** **matplotlib** # импорт библиотек

**import** **matplotlib.pyplot** **as** **plt**

**import** **numpy** **as** **np**

**import** **random**

n = **5** # Установка количества разбиений (n > 0)

type\_of\_furnishing = **0** # Тип оснащения (0 - левые, 1 - правые, 2 - средние, 3 - случайные точки)

upper = **2** # Верхняя граница интеграла

lower = **1** # Нижняя граница интеграла

**def** **fun**(x): # Исследуемая подынтегральная функция

**return** x\*\***2**

x = np.linspace(lower-**0.2**,upper+**0.2**,**100**) # точки для графика функции

y = fun(x)

fraction = np.linspace(lower,upper,n+**1**) # точки отрезков

**if** type\_of\_furnishing == **0**:

furnishing = np.arange(lower,upper,(upper-lower)/n)

**if** type\_of\_furnishing == **1**:

furnishing = np.arange(upper,lower,-(upper-lower)/n)[::-**1**]

**if** type\_of\_furnishing == **2**:

furnishing = np.arange(lower+(upper-lower)/(**2**\*n),upper,(upper-lower)/n)

**if** type\_of\_furnishing == **3**:

furnishing = np.zeros(n)

**for** i **in** range(n):

furnishing[i] = fraction[i]+(upper-lower)/n\*random.random()

furnishing = fun(furnishing) # точки оснащения

# Подсчет сравнение и вывод интегральной суммы

integral\_sum = **0**

**for** i **in** range(n):

integral\_sum += furnishing[i]\*(upper-lower)/n

**print**(f"sum: {integral\_sum}")

**print**(f"delta: {abs(integral\_sum-7/3)}")

# График функции и прямоугольники

plt.figure(figsize=(**12**, **8**), dpi=**80**)

plt.title("y = x^2")

plt.xlabel("x")

plt.ylabel("y")

plt.grid()

plt.plot(x, y)

plt.plot([lower-**0.2**,upper+**0.2**], [**0**,**0**])

**for** i **in** range(n):

# левый отрезок

plt.plot([fraction[i],fraction[i]], [**0**, furnishing[i]], linewidth=**1**, color='red')

# правый отрезок

plt.plot([fraction[i+**1**], fraction[i+**1**]], [**0**, furnishing[i]], linewidth=**1**, color='red')

# перекладина

plt.plot([fraction[i], fraction[i+**1**]], [furnishing[i],furnishing[i]], linewidth=**1**, color='red')

# Метод прямоугольников

## Код

# Вычисление интегральной суммы

furnishing\_trapezoid = fun(fraction)

integral\_sum = **0**

**for** i **in** range(n):

integral\_sum += (furnishing\_trapezoid[i]+furnishing\_trapezoid[i+**1**])\*(upper-lower)/n/**2**

**print**(integral\_sum)

# График функции и трапеций

plt.figure(figsize=(**12**, **8**), dpi=**80**)

plt.title("y = x^2")

plt.xlabel("x")

plt.ylabel("y")

plt.grid()

plt.plot(x, y)

plt.plot([lower-**0.2**,upper+**0.2**], [**0**,**0**]) # ось OX

**for** i **in** range(n):

# левый отрезок

plt.plot([fraction[i],fraction[i]], [**0**, furnishing\_trapezoid[i]], linewidth=**1**, color='red')

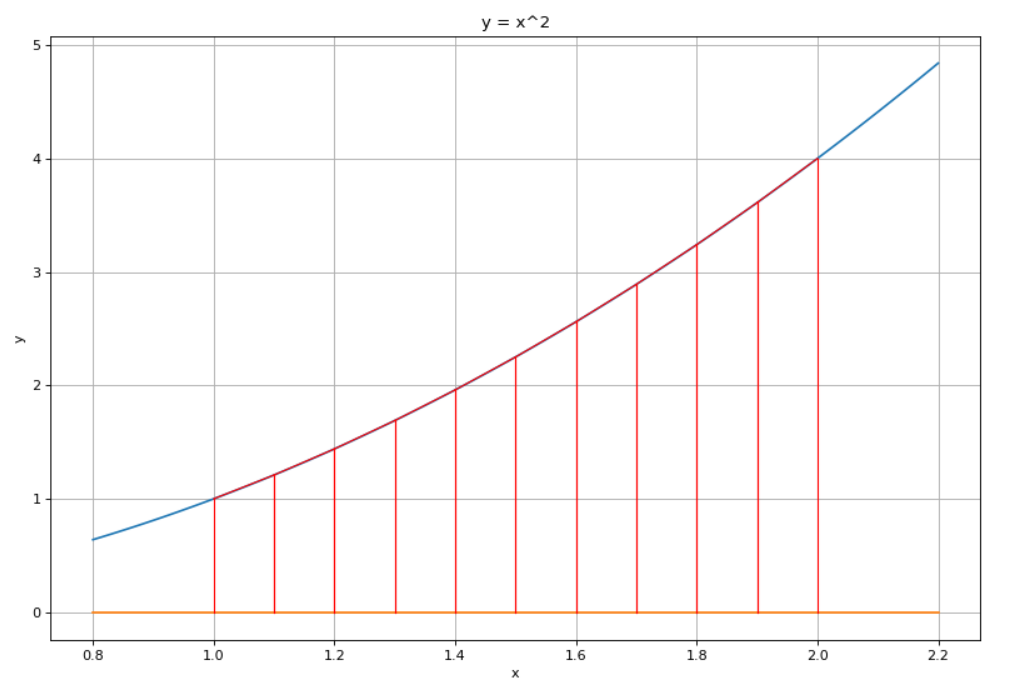
# правый отрезок

plt.plot([fraction[i+**1**], fraction[i+**1**]], [**0**, furnishing\_trapezoid[i+**1**]], linewidth=**1**, color='red')

# перекладина

plt.plot([fraction[i], fraction[i+**1**]], [furnishing\_trapezoid[i],furnishing\_trapezoid[i+**1**]], linewidth=**1**, color='red')

## График



## Полученные результаты

Метод трапеций гораздо точнее вычисляет интегральную сумму, по сравнению с методом прямоугольников, даже на небольшом количестве разбиений.

# Вывод

Было доказано существование данного интеграла, найдено его значение, а также была написана программа, способная вычислять практически любые определенные интегралы с помощью численных методов прямоугольников и трапеций и рисовать графики исследуемых интегральных сумм.