ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

Факультет программной инженерии и компьютерной техники

Дисциплина:
«Вычислительная математика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3
Вариант 10.

Выполнил:
Студент гр. P32151
Понамарев Степан Андреевич

Проверил:
Машина Екатерина Алексеевна

Санкт-Петербург
2023г.

1. **Цель лабораторной работы**: найти приближенное значение определенного интеграла с требуемой точностью различными численными методами.
2. **Задание лабораторной работы**:

Обязательное задание (до 80 баллов)

Исходные данные:

1. Пользователь выбирает функцию, интеграл которой требуется вычислить (3-5 функций), из тех, которые предлагает программа.
2. Пределы интегрирования задаются пользователем.
3. Точность вычисления задается пользователем.
4. Начальное значение числа разбиения интервала интегрирования: n=4.
5. Ввод исходных данных осуществляется с клавиатуры.

Программная реализация задачи:

1. Реализовать в программе методы по выбору пользователя:
   - Метод прямоугольников (3 модификации: левые, правые, средние)
   - Метод трапеций
   - Метод Симпсона
2. Методы должны быть оформлены в виде отдельной(ого) функции/класса.
3. Вычисление значений функции оформить в виде отдельной(ого) функции/класса.
4. Для оценки погрешности и завершения вычислительного процесса использовать правило Рунге.
5. Предусмотреть вывод результатов: значение интеграла, число разбиения интервала интегрирования для достижения требуемой точности.

Вычислительная реализация задачи:

1. Вычислить интеграл, приведенный в таблице 1, точно.
2. Вычислить интеграл по формуле Ньютона – Котеса при $n = 5$.
3. Вычислить интеграл по формулам средних прямоугольников, трапеций и Симпсона при $n = 10$.
4. Сравнить результаты с точным значением интеграла.
5. Определить относительную погрешность вычислений для каждого метода.
6. В отчете ***отразить последовательные вычисления***.

Необязательное задание (до 20 баллов)

1. Установить сходимость рассматриваемых несобственных интегралов 2 рода (2-3 функции). Если интеграл - расходящийся, выводить сообщение: «Интеграл не существует».
2. Если интеграл сходящийся, реализовать в программе вычисление несобственных интегралов 2 рода (заданными численными методами).
3. Рассмотреть случаи, когда подынтегральная функция терпит бесконечный разрыв: 1) в точке a, 2) в точке b, 3) на отрезке интегрирования

3. **Рабочие формулы (вариант 10)**:
   1. **Метод прямоугольников**.
      - Левых: $\int_a^b f(x)dx = h\sum_{i=1}^{n} y_{i-1}$

- Правых: $\int_a^b f(x)dx = h \sum_{i=1}^{n} y_i$
- Средних: $\int_a^b f(x)dx = h \sum_{i=1}^{n} f(x_{i-\frac{1}{2}})$
- Где $h_i = h = \frac{b-a}{n} = const$

2. **Метод трапеций**.
   - $\int_a^b f(x)dx = \frac{h}{2} \cdot (y_0 + y_n + 2 \sum_{i=1}^{n-1} y_i)$
3. **Метод Симпсона**.
   - $\int_a^b f(x) = \frac{h}{3}[(y_0 + 4(y_1 + y_3 + \dots + y_{n-1}) + 2(y_2 + y_4 + \dots + y_{n-2}) + y_n)]$
4. **Формула правила Рунге**
   - $I - I_{\frac{h}{2}} \approx \frac{I_{\frac{h}{2}} - I_h}{2^k - 1}$, где
   - I – точное значение интеграла
   - I_h, $I_{\frac{h}{2}}$ – приближенные значение интеграла, вычисленные с различными шагами h
   - k – порядок точности квадратурной точности ($k = 2$ - для формул средних прямоугольников и трапеций, $k = 4$ - для формулы Симпсона)

Вычислительная реализация задачи:

1. Точное вычисление интеграла.

$$\int_2^4 (x^3 - 3x^2 + 7x - 10)\,dx = \left(\frac{x^4}{4} - x^3 + \frac{7x^2}{2} - 10x\right)\Bigg|_2^4 = 26$$

2. Вычисления с помощью приведённых методов
   a. Вычисление по формуле Ньютона-Котеса при $n = 5$.

$$c_5^0 = c_5^5 = \frac{19(b-a)}{288} = \frac{19(4-2)}{288} = 0{,}13194$$

$$c_5^1 = c_5^4 = \frac{75(b-a)}{288} = \frac{75(4-2)}{288} = 0{,}52083$$

$$c_5^2 = c_5^3 = \frac{50(b-a)}{288} = \frac{50(4-2)}{288} = 0{,}34722$$

$$h = \frac{b-a}{5} = \frac{4-2}{5} = 0{,}4$$

$$\int_2^4 (x^3 - 3x^2 + 7x - 10)\,dx$$

$$= c_5^0 f(a) + c_5^1 f(a+h) + c_5^2 f(a+2h) + c_5^3 f(a+3h) + c_5^4 f(a+4h) + c_5^5 f(b) =$$

$$= 0{,}13194 \cdot (2^3 - 3 \cdot 2^2 + 7 * 2 - 10) +$$

$$+ 0{,}52083 \cdot (2{,}4^3 - 3 \cdot 2{,}4^2 + 7 * 2{,}4 - 10) +$$

$$+ 0{,}34722 \cdot (2{,}8^3 - 3 \cdot 2{,}8^2 + 7 * 2{,}8 - 10) +$$

$$+ 0{,}34722 \cdot (3{,}2^3 - 3 \cdot 3{,}2^2 + 7 * 3{,}2 - 10) +$$

$$+ 0{,}52083 \cdot (3{,}6^3 - 3 \cdot 3{,}6^2 + 7 * 3{,}6 - 10) +$$

$$+0{,}13194 \cdot (4^3 - 3 \cdot 4^2 + 7 * 4 - 10) = 25{,}99971$$

b. Вычисление по формуле средних прямоугольников.

$$h = \frac{b-a}{10} = \frac{4-2}{10} = 0{,}2$$

$$1)\ f(2{,}1) = 2{,}1^3 - 3 \cdot 2{,}1^2 + 7 \cdot 2{,}1 - 10 = 0{,}731$$
$$2)\ f(2{,}3) = 2{,}3^3 - 3 \cdot 2{,}3^2 + 7 \cdot 2{,}3 - 10 = 2{,}397$$
$$3)\ f(2{,}5) = 2{,}5^3 - 3 \cdot 2{,}5^2 + 7 \cdot 2{,}5 - 10 = 4{,}375$$
$$4)\ f(2{,}7) = 2{,}7^3 - 3 \cdot 2{,}7^2 + 7 \cdot 2{,}7 - 10 = 6{,}713$$
$$5)\ f(2{,}9) = 2{,}9^3 - 3 \cdot 2{,}9^2 + 7 \cdot 2{,}9 - 10 = 9{,}459$$
$$6)\ f(3{,}1) = 3{,}1^3 - 3 \cdot 3{,}1^2 + 7 \cdot 3{,}1 - 10 = 12{,}661$$
$$7)\ f(3{,}3) = 3{,}3^3 - 3 \cdot 3{,}3^2 + 7 \cdot 3{,}3 - 10 = 16{,}367$$
$$8)\ f(3{,}5) = 3{,}5^3 - 3 \cdot 3{,}5^2 + 7 \cdot 3{,}5 - 10 = 20{,}625$$
$$9)\ f(3{,}7) = 3{,}7^3 - 3 \cdot 3{,}7^2 + 7 \cdot 3{,}7 - 10 = 25{,}483$$
$$10)\ f(3{,}9) = 3{,}9^3 - 3 \cdot 3{,}9^2 + 7 \cdot 3{,}9 - 10 = 30{,}989$$

$$\int_2^4 (x^3 - 3x^2 + 7x - 10)\,dx = h \sum_{i=1}^{10} f(x_{i-\frac{1}{2}}) = 0{,}2 \cdot 129{,}8 = 25{,}96$$

c. Вычисление по формуле трапеций.

$$h = \frac{b-a}{10} = \frac{4-2}{10} = 0{,}2$$

$$0)\ f(2{,}0) = 2{,}0^3 - 3 \cdot 2{,}0^2 + 7 \cdot 2 - 10 = 0$$
$$1)\ f(2{,}2) = 2{,}2^3 - 3 \cdot 2{,}2^2 + 7 \cdot 2{,}2 - 10 = 1{,}528$$
$$2)\ f(2{,}4) = 2{,}4^3 - 3 \cdot 2{,}4^2 + 7 \cdot 2{,}4 - 10 = 3{,}344$$
$$3)\ f(2{,}6) = 2{,}6^3 - 3 \cdot 2{,}6^2 + 7 \cdot 2{,}6 - 10 = 5{,}496$$
$$4)\ f(2{,}8) = 2{,}8^3 - 3 \cdot 2{,}8^2 + 7 \cdot 2{,}8 - 10 = 8{,}032$$
$$5)\ f(3{,}0) = 3{,}0^3 - 3 \cdot 3{,}0^2 + 7 \cdot 3 - 10 = 11$$
$$6)\ f(3{,}2) = 3{,}2^3 - 3 \cdot 3{,}2^2 + 7 \cdot 3{,}2 - 10 = 14{,}448$$
$$7)\ f(3{,}4) = 3{,}4^3 - 3 \cdot 3{,}4^2 + 7 \cdot 3{,}4 - 10 = 18{,}424$$
$$8)\ f(3{,}6) = 3{,}6^3 - 3 \cdot 3{,}6^2 + 7 \cdot 3{,}6 - 10 = 22{,}976$$
$$9)\ f(3{,}8) = 3{,}8^3 - 3 \cdot 3{,}8^2 + 7 \cdot 3{,}8 - 10 = 28{,}152$$
$$10)\ f(4{,}0) = 4{,}0^3 - 3 \cdot 4{,}0^2 + 7 \cdot 4 - 10 = 34$$

$$\int_2^4 (x^3 - 3x^2 + 7x - 10)\,dx = \frac{h}{2} \cdot \left(y_0 + y_n + 2 \sum_{i=1}^{9} y_i \right)$$
$$= \frac{0{,}2}{2} \cdot (0 + 34 + 2 \cdot 113{,}4) = 26{,}08$$

d. Вычисление по формуле Симпсона при $n = 10$.

Используем значения в узлах, вычисленных в предыдущем пункте.

$$\int_2^4 x^3 - 3x^2 + 7x - 10\,dx$$
$$= \frac{h}{3}[(y_0 + 4(y_1 + y_3 + \cdots + y_9) + 2(y_2 + y_4 + \cdots + y_8) + y_{10})]$$
$$= \frac{0{,}2}{3} \cdot 390 = 26$$

3. Сравнение с точным значением интервала.
   a. Метод Ньютона-Котеса: $|f_{точн} - f_{прибл}| = |26 - 25{,}99971| = 0{,}00029$
   b. Метод средних прямоугольников: $|f_{точн} - f_{прибл}| = |26 - 25{,}96| = 0{,}04$
   c. Метод трапеций: $|f_{точн} - f_{прибл}| = |26 - 26{,}08| = 0{,}08$
   d. Метод Симпсона: $|f_{точн} - f_{прибл}| = |26 - 26| = 0$
4. Относительные погрешности

$$\delta = \frac{|f_{точн} - f_{прибл}|}{f_{точн}} \cdot 100\%$$

   a. Метод Ньютона-Котеса: $\delta = \frac{|f_{точн} - f_{прибл}|}{f_{точн}} \cdot 100\% = \frac{0{,}00029}{26} = 0{,}0001\%$
   b. Метод средних прямоугольников: $\delta = \frac{|f_{точн} - f_{прибл}|}{f_{точн}} \cdot 100\% = \frac{0{,}04}{26} = 0{,}15\%$
   c. Метод трапеций: $\delta = \frac{|f_{точн} - f_{прибл}|}{f_{точн}} \cdot 100\% = \frac{0{,}08}{26} = 0{,}31\%$
   d. Метод Симпсона: $\delta = \frac{|f_{точн} - f_{прибл}|}{f_{точн}} \cdot 100\% = \frac{0}{26} = 0\%$

4. Листинг программы (коды используемых методов):

Main.py:

```
from InputManager import InputManager
from IntegralManager import Integral
from IntegralSolver import Integrate

if __name__ == "__main__":
    variants = ["x^3 - 3x^2 + 7x - 10",
                "2x^3 - 5x^2 - 3x + 21",
                "7x^3 - 2x^2 - 9x + 36",
                "1/x"]
    values = [Integral(lambda x: x ** 3 - 3 * x ** 2 + 7 * x - 10),
              Integral(lambda x: 2 * x ** 3 - 5 * x ** 2 - 3 * x + 21),
              Integral(lambda x: 7 * x ** 3 - 2 * x ** 2 - 9 * x + 36),
              Integral(lambda x: 1/x)]
    integral = InputManager.multiple_choice_input(variants, values, "Выберите
функцию для интегрирования:")

    variants = ["Метод прямоугольника - левый", "Метод прямоугольника -
центр", "Метод прямоугольника - правый",
                "Метод трапеций", "Метод Симпсона"]
    values = [Integrate.left_rectangle, Integrate.center_rectangle,
Integrate.right_rectangle,
              Integrate.trapezoid_method, Integrate.simpson_method]
    method = InputManager.multiple_choice_input(variants, values, "Выберите
метод интегрирования:")

    left = InputManager.float_input("Введите левый предел интегрирования: ")
    right = InputManager.float_input("Введите правый предел интегрирования:
```

```
")
    while not left < right:
        print("Левый предел должен быть меньше правого.")
        left = InputManager.float_input("Введите левый предел интегрирования: ")
        right = InputManager.float_input("Введите правый предел интегрирования: ")

    try:
        result = integral.solve(left, right, method)
        if result is not None:
            print("------------РЕШЕНИЕ------------")
            print("Результат вычисления:", result)
            print("Число разбиений:", integral.steps)
            if InputManager.yes_or_no_input("Нарисовать график функции?"):
                integral.draw_func(left, right)
        else:
            print("На заданном промежутке интеграл не существует.")
    except Exception:
        print("Не удалось найти значение интеграла.\nПопробуйте ввести другие пределы интегрирования.")
```

InputManager.py:

```
import numpy as np
class InputManager:
    @staticmethod
    def string_input(message=""):
        buf = ""
        while buf == "":
            buf = input(message).strip()
        return buf

    @staticmethod
    def _check_number(buf):
        try:
            float(buf.replace(',', '.'))
            return True
        except ValueError:
            return False

    @staticmethod
    def _convert_to_number(num):
        try:
            return float(num.replace(',', '.'))
        except ValueError:
            return None

    @staticmethod
    def float_input(message=""):
        number = None
        while number is None:
            number = InputManager._convert_to_number(InputManager.string_input(message))
        return number

    @staticmethod
    def int_input(message=""):
        return int(InputManager.float_input(message))

    @staticmethod
    def yes_or_no_input(message=""):
```

```
        answer = "0"
        while answer[0].lower() not in ["y", "n", "д", "н"]:
            answer = InputManager.string_input(message + " [y/n]: ")
        return answer[0].lower() in ["y", "д"]

    @staticmethod
    def enum_input(variants_list, message):
        buf = ""
        while buf not in variants_list:
            buf = InputManager.string_input(message)
        return buf

    @staticmethod
    def multiple_choice_input(variant_list, values_list, message=""):
        n = len(variant_list)
        if n == 0 or len(variant_list) != len(values_list):
            raise ValueError

        if message != "":
            print(message)

        for i in range(n):
            s = f"{i + 1}."
            lines = variant_list[i].split('\n')
            print('\t' + s, lines[0])
            for line in lines[1:]:
                print("\t" + ' ' * len(s), line)

        i = int(InputManager.enum_input([str(i) for i in range(1, n + 1)],
f"Введите число от 1 до {n}: ")) - 1
        return values_list[i]

    @staticmethod
    def epsilon_input(message=""):
        e = InputManager.float_input(message)
        while not (0 < e and e <= 1):
            print("Эпсилон должно быть в промежутке от 0 до 1!")
            e = InputManager.float_input(message)
        return e
```

IntegralManager.py:

```
from random import random

import numpy as np
from matplotlib import pyplot as plt

from InputManager import InputManager


class Integral:

    def __init__(self, func):
        self.func = func
        self.steps = 0
        self.left = None
        self.right = None
        self.iterations = 0

    def calculate(self, x):
        return self.func(x)

    def set_steps(self, steps):
```

```
        self.steps = steps

    def solve(self, left, right, method_func):
        self.left = left
        self.right = right
        try:
            steps = 4
            epsilon = InputManager.epsilon_input("Введите epsilon: ")
            while abs(method_func(self, left, right, steps / 2) -
method_func(self, left, right, steps)) > epsilon \
                    and self.iterations < 19:
                steps *= 2
                self.iterations += 1
            self.set_steps(steps)
            return round(method_func(self, left, right, steps), len(str(int(1
/ epsilon))))
        except Exception:
            return None

    def draw_func(self, left=None, right=None):
        if left is None or right is None:
            if self.left is None or self.right is None:
                print("Невозможно нарисовать график. Не определены концы
отрезка")
            else:
                left = self.left
                right = self.right
        x_axis = np.linspace(min(left, right), max(left, right), 100)
        plt.plot(x_axis, self.calculate(x_axis))
        plt.grid(True, which='both')
        plt.axvline(x=0, color='k')
        plt.axhline(y=0, color='k')
        plt.show()
```

IntegralSolver.py:

```
class Integrate:

    @staticmethod
    def left_rectangle(integral, left, right, n):
        result = 0
        step = abs(left - right) / n
        i = left
        while i < right:
            result += integral.calculate(i)
            i += step
        return result * step

    @staticmethod
    def center_rectangle(integral, left, right, n):
        result = 0
        step = abs(left - right) / n
        i = left + step / 2
        while i < right:
            result += integral.calculate(i)
            i += step
        return result * step

    @staticmethod
    def right_rectangle(integral, left, right, n):
        result = 0
        step = abs(left - right) / n
        i = left + step
        while i <= right:
```

```
            result += integral.calculate(i)
            i += step
        return result * step

    @staticmethod
    def trapezoid_method(integral, left, right, n):
        result = 0
        step = abs(left - right) / n
        i = left + step
        while i < right:
            result += integral.calculate(i)
            i += step
        result += (integral.calculate(left) + integral.calculate(right)) / 2
        return result * step

    @staticmethod
    def simpson_method(integral, left, right, n):
        result = 0
        step = abs(left - right) / n
        for i in range(1, n):
            if i % 2:
                result += 2 * integral.calculate(left + step * i)
            else:
                result += 4 * integral.calculate(left + step * i)
        result += integral.calculate(left) + integral.calculate(right)
        return result * step / 3
```

5. Результаты выполнения программы при различных исходных данных.

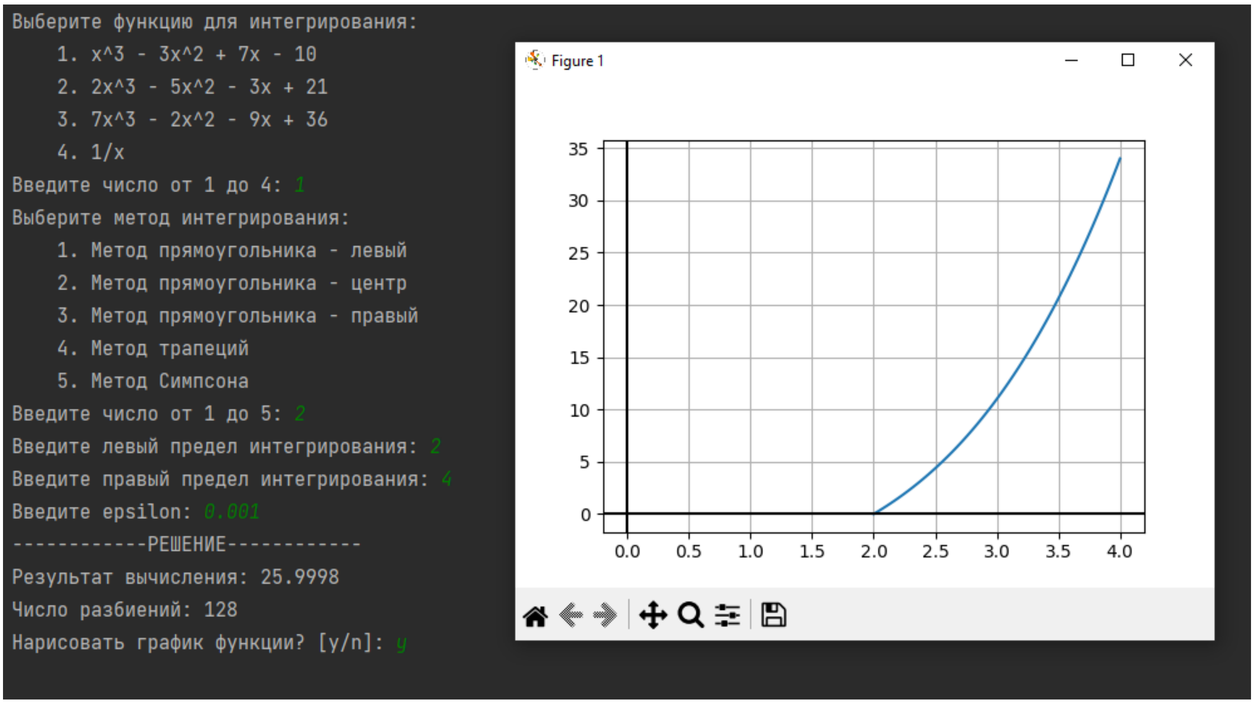

Рис. 1. Пример работы программы 1

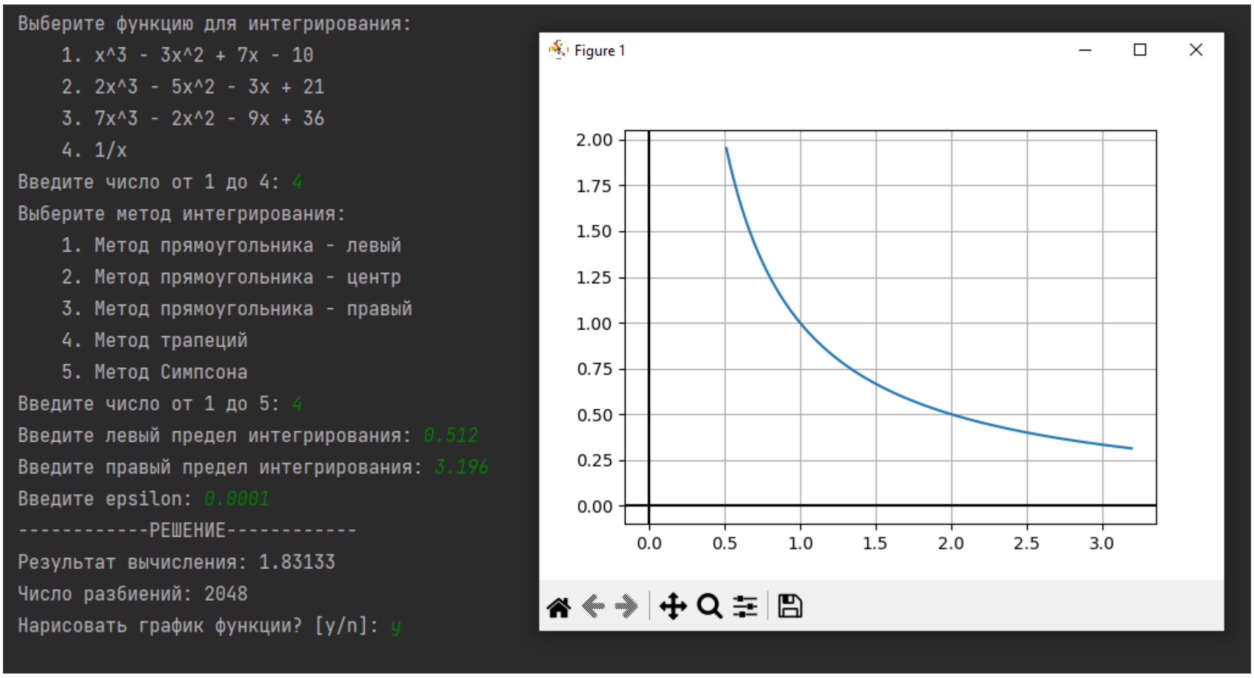

Рис. 3. Пример работы программы 2

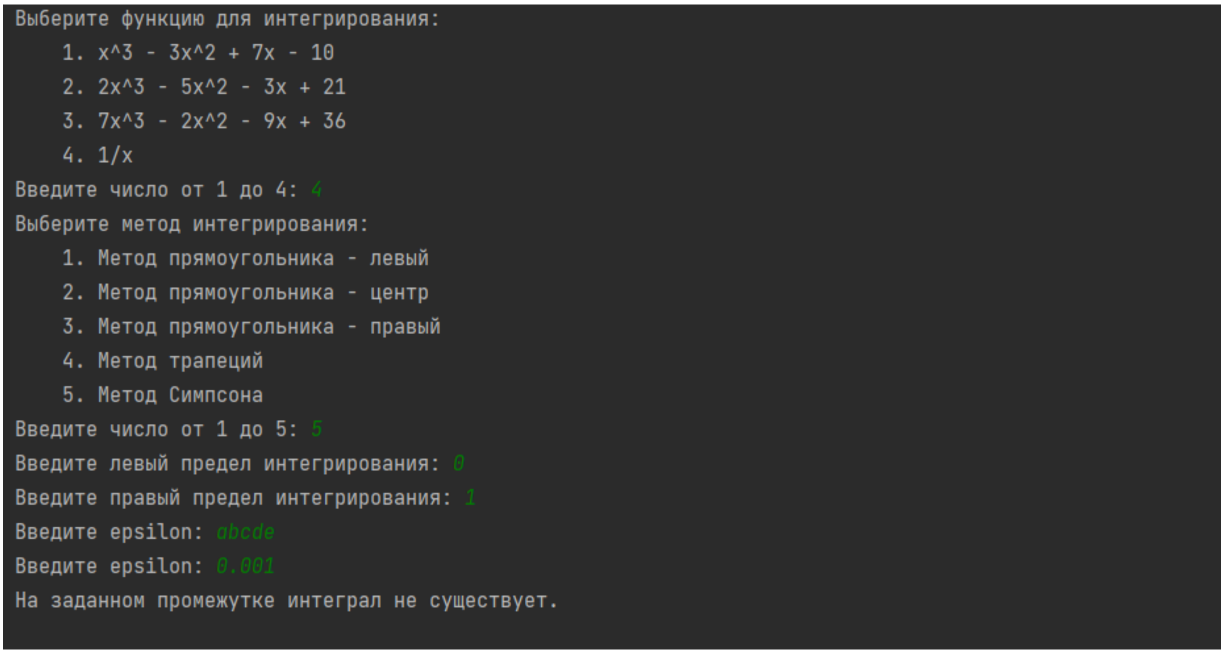

Рис. 2. Пример работы программы 3

6. Выводы

В ходе выполнения лабораторной работы я изучил и реализовал несколько методов вычисления определённых интегралов. Также отточил навыки создания юзер-френдли консольных приложений.