Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования «Санкт-Петербургский
национальный исследовательский университет информационных технологий,
механики и оптики»

Факультет программной инженерии и компьютерной техники

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ 4

ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ

ВАРИАНТ 12

Студент: Пышкин Никита Сергеевич, P3213

Преподаватель:

Санкт Петербург 2025

Содержание

Цель лабораторной работы

Найти функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов.

Порядок выполнения лабораторной работы

Вычислительная реализация задачи:

1. Сформировать таблицу табулирования заданной функции на указанном интервале (см. табл. 1)

2. Построить линейное и квадратичное приближения по 11 точкам заданного интервала;

3. Найти среднеквадратические отклонения для каждой аппроксимирующей функции. Ответы дать с тремя знаками после запятой;

4. Выбрать наилучшее приближение;

5. Построить графики заданной функции, а также полученные линейное и квадратичное приближения;

6. Привести в отчете подробные вычисления.

Программная реализация задачи:

1. Предусмотреть ввод исходных данных из файла/консоли (таблица $y = f(x)$ должна содержать от 8 до 12 точек).

2. Реализовать метод наименьших квадратов, исследуя все указанные функции.

3. Предусмотреть вывод результатов в файл/консоль: коэффициенты аппроксимирующих функций, среднеквадратичное отклонение, массивы значений $x_i, y_i, \varphi(x_i), \varepsilon_i$.

4. Для линейной зависимости вычислить коэффициент корреляции Пирсона.

5. Вычислить коэффициент детерминации, программа должна выводить соответствующее сообщение в зависимости от полученного значения R^2.

6. Программа должна отображать наилучшую аппроксимирующую функцию.

7. Организовать вывод графиков функций, графики должны полностью отображать весь исследуемый интервал (с запасом).

8. Программа должна быть протестирована при различных наборах данных, в том числе и некорректных.

Рабочие формулы

Линейная аппроксимация:

$$\begin{cases} aSXX + bSX = SXY \\ aSX + bn = SY \end{cases}$$

Квадратичная аппроксимация:

$$\begin{cases} na_0 + \left(\sum_{i=1}^{n} x_i\right) a_1 + \left(\sum_{i=1}^{n} {x_i}^2\right) a_2 = \sum_{i=1}^{n} y_i \\ \left(\sum_{i=1}^{n} x_i\right) a_0 + \left(\sum_{i=1}^{n} {x_i}^2\right) a_1 + \left(\sum_{i=1}^{n} {x_i}^3\right) a_2 = \sum_{i=1}^{n} x_i y_i \\ \left(\sum_{i=1}^{n} {x_i}^2\right) a_0 + \left(\sum_{i=1}^{n} {x_i}^3\right) a_1 + \left(\sum_{i=1}^{n} {x_i}^4\right) a_2 = \sum_{i=1}^{n} {x_i}^2 y_i \end{cases}$$

Вычислительная часть задания

Исследуемая функция: $y = \frac{4x}{x^4+12}$

Исследуемый интервал: $x \in [-2; 0], h = 0.2$

График функции:

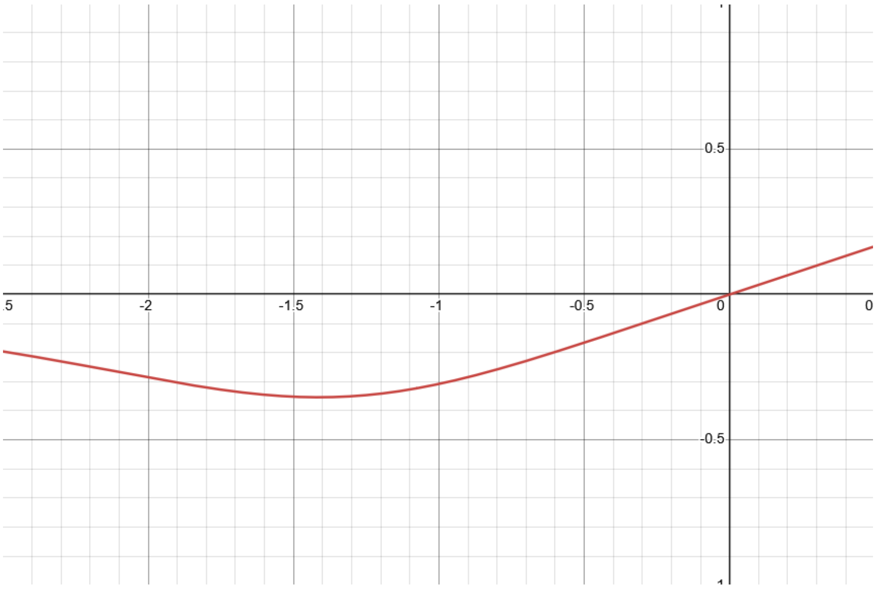


Таблица:

	1	2	3	4	5	6	7	8	9	10	11
x	-2	-1.8	-1.6	-1.4	-1.2	-1	-0.8	-0.6	-0.4	-0.2	0
y	-0.2857	-0.32	-0.3449	-0.3535	-0.3411	-0.3077	-0.2579	-0.1979	-0.133	-0.0667	0

Линейная аппроксимация:

$P_1(x) = ax + b$

Суммы: $SX = -11; SXX = 15.4; SY = -2.608; SXY = 3.303$

$$\begin{cases} 15.4a - 11b = 3.303 \\ -11a + 11b = -2.608 \end{cases}$$

$a = 0.13158; b = -0.1161$

$P_1(x) = 0.13158x - 0.1161$

Рассчитаем $\delta = \sqrt{\frac{\sum_{i=1}^{n}(P_1(x_i)-y_i)^2}{n}} = \sqrt{\frac{0.04293}{10}} = 0.066$

Квадратичная аппроксимация:

$P_2(x) = a_0 + a_1x + a_2x^2$

Суммы: $\sum_{i=1}^{n} x_i = -11$; $\sum_{i=1}^{n} {x_i}^2 = 15.4$; $\sum_{i=1}^{n} {x_i}^3 = -24.2$; $\sum_{i=1}^{n} {x_i}^4 = 40.532$; $\sum_{i=1}^{n} y_i = -2.608$; $\sum_{i=1}^{n} x_iy_i = 3.303$; $\sum_{i=1}^{n} {x_i}^2y_i = -4.815$

$$\begin{cases} 11a_0 - 11a_1 + 15.4a_2 = -2.608 \\ -11a_0 + 15.4a_1 - 24.2a_2 = 3.303 \\ 15.4a_0 - 24.2a_1 + 40.532a_2 = -4.815 \end{cases}$$

$a_0 = 0.0472$; $a_1 = 0.5397$; $a_2 = 0.1855$

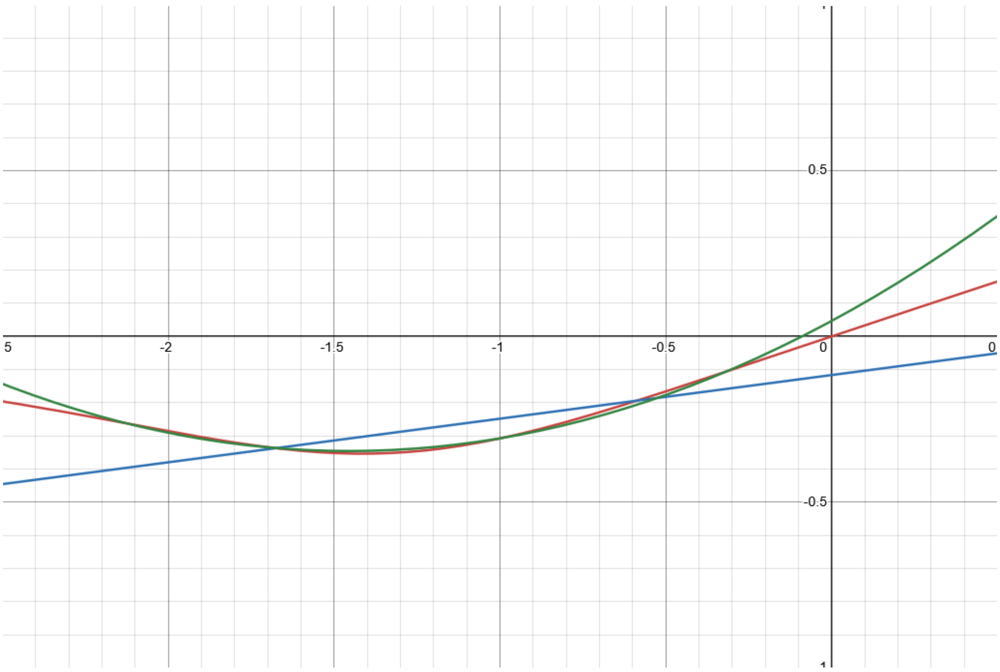
$P_2(x) = 0.0472 + 0.5397x + 0.1855x^2$

Рассчитаем $\delta = \sqrt{\frac{\sum_{i=1}^{n}(P_2(x_i)-y_i)^2}{n}} = \sqrt{\frac{0.002827}{10}} = 0.019$

Итоги:

Наилучшее приближение – квадратичное (т.к. $0.019 < 0.066$)

Графики:



Красная линия – исходная функция

Синяя линия – линейная аппроксимация

Красная линия – квадратичная аппроксимация

Программная часть задания

linear.py:

```
from typing import Callable, Any, List
import sympy as sp


class LinearApproximation:
    def __init__(self, table: bool = False):
        self.table = table


    def solve(self, *args, **kwargs) -> Any:
        if self.table:
            return self._table_solve(*args, **kwargs)


        return self._func_solve(*args, **kwargs)


    def _func_solve(self, f: Callable, left: float, right: float, n:
int, lambdify: bool = True) -> Any:
        sx = sxx = sy = sxy = 0


        h = (right - left) / n
        for i in range(n + 1):
            x = left + h * i
            y = f(x)
            sx += x
            sxx += x**2
            sy += y
            sxy += x*y


        a, b = sp.symbols("a b")
```

```
        root = sp.solve([a * sxx + b * sx - sxy, a * sx + b * n - sy], [a, b], dict=True)[0]

        x = sp.Symbol("x")
        function = root[a] * x + root[b]

        if lambdify:
            return sp.lambdify(x, function)

        return x, function

    def _table_solve(self, table: List[List[float]], lambdify: bool = True) -> Any:
        sx = sxx = sy = sxy = 0

        n = len(table[0])
        for x, y in zip(table[0], table[1]):
            sx += x
            sxx += x**2
            sy += y
            sxy += x*y

        a, b = sp.symbols("a b")
        root = sp.solve([a * sxx + b * sx - sxy, a * sx + b * n - sy], [a, b], dict=True)[0]

        x = sp.Symbol("x")
        function = root[a] * x + root[b]

        if lambdify:
            return sp.lambdify(x, function)

        return x, function
```

quadratic.py:

```
from typing import Callable, Any, List
```

```python
import sympy as sp


class QuadraticApproximation:
    def __init__(self, table: bool = False):
        self.table = table


    def solve(self, *args, **kwargs) -> Any:
        if self.table:
            return self._table_solve(*args, **kwargs)


        return self._func_solve(*args, **kwargs)


    def _func_solve(self, f: Callable, left: float, right: float, n:
int, lambdify: bool = True) -> Any:
        sx = sx2 = sx3 = sx4 = sy = sxy = sx2y = 0


        h = (right - left) / n
        for i in range(n + 1):
            x = left + h * i
            y = f(x)
            sx += x
            sx2 += x**2
            sx3 += x**3
            sx4 += x**4
            sy += y
            sxy += x*y
            sx2y += x**2*y


        a0, a1, a2 = sp.symbols("a0 a1 a2")
        root = sp.solve(
            [
                n * a0 + sx * a1 + sx2 * a2 - sy,
                sx * a0 + sx2 * a1 + sx3 * a2 - sxy,
                sx2 * a0 + sx3 * a1 + sx4 * a2 - sx2y
```

```
            ],
            [a0, a1, a2],
            dict=True
        )[0]

        x = sp.Symbol("x")
        function = root[a0] + root[a1] * x + root[a2] * x ** 2

        if lambdify:
            return sp.lambdify(x, function)

        return x, function

    def _table_solve(self, table: List[List[float]], lambdify: bool =
True) -> Any:
        sx = sx2 = sx3 = sx4 = sy = sxy = sx2y = 0

        n = len(table[0])
        for x, y in zip(table[0], table[1]):
            sx += x
            sx2 += x**2
            sx3 += x**3
            sx4 += x**4
            sy += y
            sxy += x*y
            sx2y += x**2*y

        a0, a1, a2 = sp.symbols("a0 a1 a2")
        root = sp.solve(
            [
                n * a0 + sx * a1 + sx2 * a2 - sy,
                sx * a0 + sx2 * a1 + sx3 * a2 - sxy,
                sx2 * a0 + sx3 * a1 + sx4 * a2 - sx2y
            ],
```

```
        [a0, a1, a2],
        dict=True
    )[0]

    x = sp.Symbol("x")
    function = root[a0] + root[a1] * x + root[a2] * x ** 2

    if lambdify:
        return sp.lambdify(x, function)

    return x, function
```

Заключение

В ходе лабораторной работы я изучил разные численные интегрирования и реализовал их на языке Python.