Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №4

по дисциплине

«Вычислительная математика»

Вариант №13

Выполнил:

Студент группы P3213

Султанов А.Р.

Проверила:

Машина Е.А.

г. Санкт-Петербург

2024г.

Цель работы

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

$y = \frac{31x}{x^4+13}, x \in [0;4], h = 0.4$

Задание

Вычислительная реализация задачи

x_i	0	0,4	0,8	1,2	1,6	2	2,4	2,8	3,2	3,6	4
y_i	0	0,952	1,8494	2,4679	2,5366	2,1379	1,6112	1,1656	0,8417	0,6167	0,461

Линейное приближение:

$\varphi_1(x) = ax + b$

$SX = \sum_{i=1}^{n} x_i = 22,$

$SY = \sum_{i=1}^{n} y_i = 14.64,$

$SXX = \sum_{i=1}^{n} x_i^2 = 61.6,$

$SXY = \sum_{i=1}^{n} x_i y_i = 27.0443$

$\Delta = SXX * n - SX * SX = 61.6 * 11 - 22 * 22 = 193.6$

$\Delta_1 = SXY * n - SX * SY = 27.0443 * 11 - 22 * 14.64 = -24.5927$

$\Delta_2 = SXX * SY - SX * SXY = 61.6 * 14.64 - 22 * 27.0443 = 306.8494$

$a = \frac{\Delta_1}{\Delta} = -0.127$

$b = \frac{\Delta_2}{\Delta} = 1.585$

$\varphi_1(x) = -0.127x + 1.585$

x_i	0	0,4	0,8	1,2	1,6	2	2,4	2,8	3,2	3,6	4
y_i	0	0,952	1,8494	2,4679	2,5366	2,1379	1,6112	1,1656	0,8417	0,6167	0,461
φ_i	1,585	1,5342	1,4834	1,4326	1,3818	1,331	1,2802	1,2294	1,1786	1,1278	1,077
ε_i	1,585	0,5822	-0,366	-1,0353	-1,1548	-0,8069	-0,331	0,0638	0,3369	0,5111	0,616

$$\sigma_1 = \sqrt{\frac{\sum_{i=1}^{n} \varepsilon_i^2}{n}} = 0,7926$$

Квадратичное приближение:

$$\varphi_2(x) = a_0 + a_1 x + a_2 x^2$$

$$SX = \sum_{i=1}^{n} x_i = 22,$$

$$SY = \sum_{i=1}^{n} y_i = 14.64,$$

$$SXX = \sum_{i=1}^{n} x_i^2 = 61.6,$$

$$SXY = \sum_{i=1}^{n} x_i y_i = 27.0443$$

$$SX3 = \sum_{i=1}^{n} x_i^2 = 193.6,$$

$$SX4 = \sum_{i=1}^{n} x_i^4 = 648.5248,$$

$$SX2Y = \sum_{i=1}^{n} x_i^2 y_i = 62.3412,$$

Составляем систему из 3 уравнений:

$$n * a_0 + SX * a_1 + SXX * a_2 = SY$$

$$SX * a_0 + SXX * a_1 + SX3 * a_2 = SXY$$

$$SXX * a_0 + SX3 * a_1 + SX4 * a_2 = SX2Y$$

Подставляем значения:

$$11 * a_0 + 22 * a_1 + 61.6 * a_2 = 14.64$$

$$22 * a_0 + 61.6 * a_1 + 193.6 * a_2 = 27.0443$$

$$61.6 * a_0 + 193.6 * a_1 + 648.5248 * a_2 = 62.3412$$

Решаем систему, находим:

$$a_0 = 0.4158$$

$$a_1 = 1.8215$$

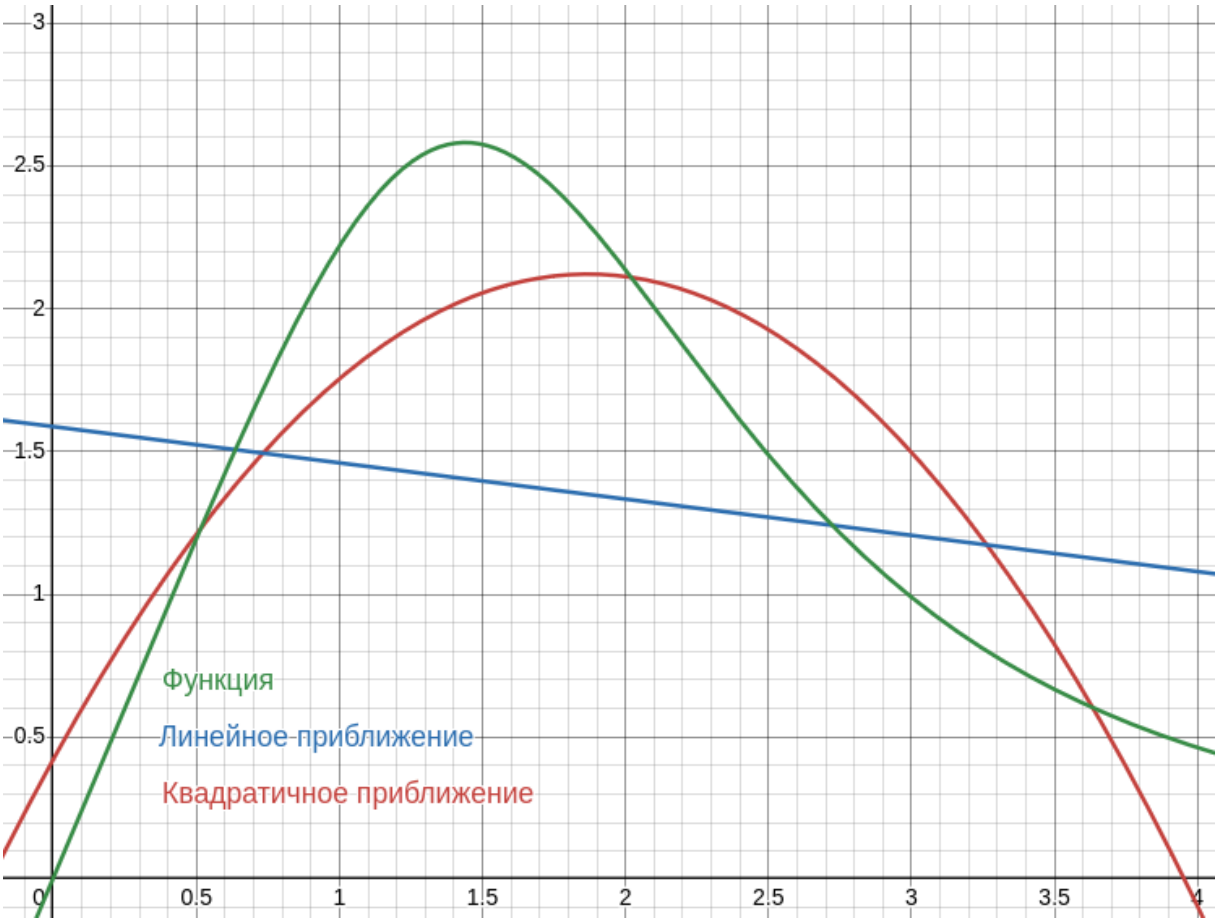
$$a_2 = -0.4871$$

$$\varphi_2(x) = 0.4158 + 1.8215x - 0.4871x^2$$

x_i	0	0,4	0,8	1,2	1,6	2	2,4	2,8	3,2	3,6	4
y_i	0	0,952	1,8494	2,4679	2,5366	2,1379	1,6112	1,1656	0,8417	0,6167	0,461
φ_i	0,4158	1,2393	2,0628	2,8863	3,7098	4,5333	5,3568	6,1803	7,0039	7,8274	8,6509
ε_i	0,4158	0,2873	0,2134	0,4184	1,1732	2,3954	3,7456	5,0147	6,1622	7,2107	8,1899

$$\sigma_2 = \sqrt{\frac{\sum_{i=1}^{n} \varepsilon_i^2}{n}} = 4,3044$$

Тогда наилучшее приближение - линейное, т.к. $\sigma_1 < \sigma_2$



Программная реализация задачи

Листинг программы доступен по ссылке:

https://github.com/MakeCheerfulInstall/Computational-Math-2024/pull/37

Линейное приближение:

```
def get_linear_approximation_factors(dots: Dots) -> tuple[float, float]:
    n = dots.get_n()

    sx, sy, sxx, sxy = dots.sx(), dots.sy(), dots.sxx(), dots.sxy()

    a, b = solve_sys(
        [
            [sxx, sx, sxy],
            [sx, n, sy],
        ],
    )

    return a, b
```

```
def get_linear_approximation_function(dots: Dots) -> Function:
    a, b = get_linear_approximation_factors(dots)

    return lambda x: a * x + b
```

Квадратичное приближение:

```
def get_quadratic_approximation_factors(dots: Dots) -> tuple[float, float,
float]:
    n = dots.get_n()

    sx, sy, sxx, sxy = dots.sx(), dots.sy(), dots.sxx(), dots.sxy()
    sx3, sx4, sx2y = dots.sx3(), dots.sx4(), dots.sx2y()

    a0, a1, a2 = solve_sys(
        [
            [n, sx, sxx, sy],
            [sx, sxx, sx3, sxy],
            [sxx, sx3, sx4, sx2y],
        ],
    )

    return a0, a1, a2


def get_quadratic_approximation_function(dots: Dots) -> Function:
    a0, a1, a2 = get_quadratic_approximation_factors(dots)

    return lambda x: a2 * x ** 2 + a1 * x + a0
```

Кубическое приближение:

```
def get_cubic_approximation_factors(dots: Dots) -> tuple[float, float,
float, float]:
    n = dots.get_n()

    sx, sy, sxx, sxy = dots.sx(), dots.sy(), dots.sxx(), dots.sxy()
    sx3, sx4, sx2y = dots.sx3(), dots.sx4(), dots.sx2y()
    sx5, sx6, sx3y = dots.sx5(), dots.sx6(), dots.sx3y()

    a0, a1, a2, a3 = solve_sys(
```

```
            [
                [n, sx, sxx, sx3, sy],
                [sx, sxx, sx3, sx4, sxy],
                [sxx, sx3, sx4, sx5, sx2y],
                [sx3, sx4, sx5, sx6, sx3y],
            ],
        )

        return a0, a1, a2, a3


def get_cubic_approximation_function(dots: Dots) -> Function:
    a0, a1, a2, a3 = get_cubic_approximation_factors(dots)

    return lambda x: a3 * x ** 3 + a2 * x ** 2 + a1 * x + a0
```

Экспоненциальное приближение:

```
def get_exponential_approximation_factors(dots: Dots) -> tuple[float,
float]:
    A, B = get_linear_approximation_factors(dots)

    a = exp(A)
    b = B

    return a, b


def get_exponential_approximation_function(dots: Dots) -> Function:
    if not dots.all_ys_exp_safe():
        raise ValueError('Есть невалидные Y')

    a, b = get_exponential_approximation_factors(dots)

    return lambda x: a * exp(b * x)
```

Логарифмическое приближение:

```
def get_logarithmic_approximation_factors(dots: Dots) -> tuple[float,
float]:
    A, B = get_linear_approximation_factors(dots)

    a = A
```

```
    b = B

    return a, b


def get_logarithmic_approximation_function(dots: Dots) -> Function:
    if not dots.all_xs_exp_safe():
        raise ValueError('Есть невалидные X')

    a, b = get_logarithmic_approximation_factors(dots)

    return lambda x: a * log(x) + b
```

Степенное приближение:

```
def get_power_approximation_factors(dots: Dots) -> tuple[float, float]:
    A, B = get_linear_approximation_factors(dots)

    a = exp(A)
    b = B

    return a, b


def get_power_approximation_function(dots: Dots) -> Function:
    if not dots.all_xs_exp_safe() or not dots.all_ys_exp_safe():
        raise ValueError('Есть невалидные X или Y')

    transformed_xs = []
    transformed_ys = []

    for x, y in dots.get_paired():
        transformed_xs.append(log(x))
        transformed_ys.append(log(y))

    transformed_dots = Dots(transformed_xs, transformed_ys)

    a, b = get_power_approximation_factors(transformed_dots)

    return lambda x: a * (x ** b)
```