Университет ИТМО

Вычислительная математика

Лабораторная работа №4

ФИО студента: Готовко Алексей Владимирович
Направление подготовки: 09.03.04 (СППО)
Учебная группа: P32101
ФИО преподавателей:
Малышева Т.А.
Рыбаков С.Д.

Санкт-Петербург
2023г.

1 Цель работы

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

2 Вычислительная реализация задачи

Функция по варианту:

$$y = \frac{4x}{x^4+3},\ x \in [-2; 0],\ h = 0.2$$

2.1 Табулирование

x	-2.000	-1.800	-1.600	-1.400	-1.200	-1.000	-0.800	-0.600	-0.400	-0.200	0.000
y	-0.421	-0.533	-0.670	-0.819	-0.946	-1.000	-0.939	-0.767	-0.529	-0.267	0.000

2.2 Линейная аппроксимация

$$\varphi(x) = b + ax$$

$$SX = \sum_{i=1}^{n} x_i = -11.000$$

$$SXX = \sum_{i=1}^{n} x_i^2 = 15.400$$

$$SXY = \sum_{i=1}^{n} x_i \cdot y_i = 7.631$$

$$SY = \sum_{i=1}^{n} y_i = -6.890$$

$$\begin{cases} b \cdot SX + a \cdot SXX = SXY \\ bn + a \cdot SX = SY \end{cases} \Rightarrow \begin{cases} a = 0.178 \\ b = -0.468 \end{cases} \tag{1}$$

x	-2.000	-1.800	-1.600	-1.400	-1.200	-1.000	-0.800	-0.600	-0.400	-0.200	0.000
y	-0.421	-0.533	-0.670	-0.819	-0.946	-1.000	-0.939	-0.767	-0.529	-0.267	0.000
$\varphi(x)$	-0.795	-0.761	-0.727	-0.694	-0.660	-0.626	-0.593	-0.559	-0.525	-0.492	-0.458
ε	0.140	0.052	0.003	0.016	0.082	0.140	0.120	0.043	0.000	0.051	0.210

Среднеквадратичное отклонение: $\delta = 0.279$

2.3 Квадратичная аппроксимация

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

$$SX = \sum_{i=1}^{n} x_i = -11.000$$

$$SXX = \sum_{i=1}^{n} x_i^2 = 15.400$$

$$SXXX = \sum_{i=1}^{n} x_i^3 = -24.200$$

$$SXXXX = \sum_{i=1}^{n} x_i^4 = 40.533$$

$$SXXY = \sum_{i=1}^{n} x_i^2 \cdot y_i = -10.066$$

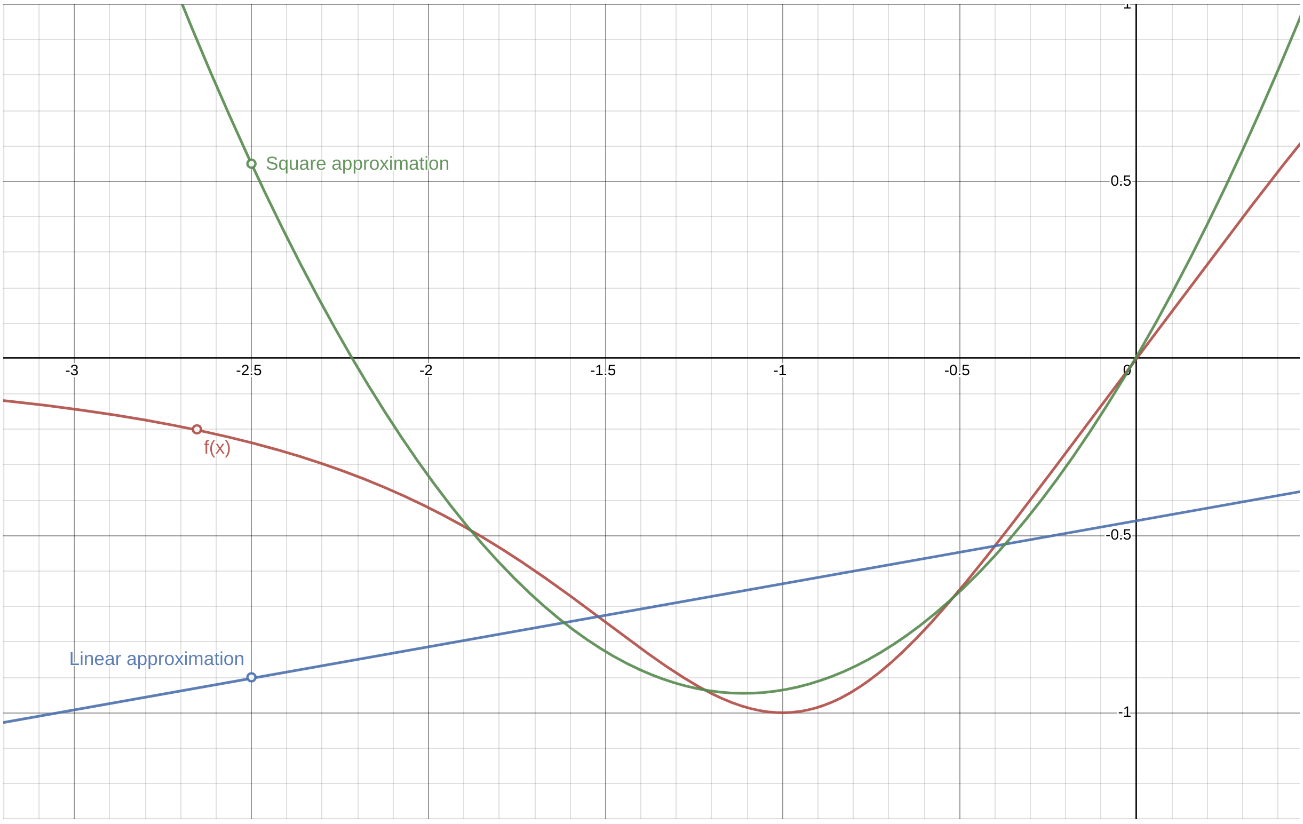
$$SXY = \sum_{i=1}^{n} x_i \cdot y_i = 7.631$$

$$SY = \sum_{i=1}^{n} y_i = -6.890$$

$$\begin{cases} a_0 n + a_1 \cdot SX + a_2 \cdot SXX = SY \\ a_0 \cdot SX + a_1 \cdot SXX + a_2 \cdot SXXX = SXY \\ a_0 \cdot SXX + a_1 \cdot SXXX + a_2 \cdot SXXXX = SXXY \end{cases} \Rightarrow \begin{cases} a_0 = 0.006 \\ a_1 = 1.715 \\ a_2 = 0.773 \end{cases} \tag{2}$$

Среднеквадратичное отклонение: $\delta = 0.055$

2.4 Графики



3 Программная реализация задачи

```
class ApproximationBuilder:
    def __init__(self, x, y):
        self.n = min(len(x), len(y))
        self.x = x[:self.n]
        self.y = y[:self.n]

        self.sx = 0
        self.sx2 = 0
        self.sx3 = 0
        self.sx4 = 0
        self.sx5 = 0
        self.sx6 = 0
        self.sy = 0
        self.syx = 0
        self.syx2 = 0
        self.syx3 = 0

        for i in range(self.n):
            self.sx += x[i]
            self.sx2 += x[i] ** 2
            self.sx3 += x[i] ** 3
            self.sx4 += x[i] ** 4
            self.sx5 += x[i] ** 5
            self.sx6 += x[i] ** 6
            self.sy += y[i]
            self.syx += y[i] * x[i]
            self.syx2 += y[i] * x[i] ** 2
            self.syx3 += y[i] * x[i] ** 3

        self.slnx = 0
        self.slnx2 = 0
        self.sylnx = 0
        self.slny = 0
        self.sxlny = 0
        self.slnxlny = 0
        try:
            for i in range(self.n):
                self.slnx += log(self.x[i])
                self.slnx2 += log(self.x[i]) ** 2
                self.sylnx += log(self.x[i]) * self.y[i]
                self.slny += log(self.y[i])
                self.sxlny += log(self.y[i]) * self.x[i]
                self.slnxlny += log(self.y[i]) * log(self.x[i])
        except ValueError:
            self.slnx = None
            self.slnx2 = None
            self.sylnx = None
            self.slny = None
            self.sxlny = None
            self.slnxlny = None

    def linear(self):
        matrix = Matrix(2,
                        [
                            [self.sx, self.sx2, self.syx],
                            [self.n, self.sx, self.sy]
                        ])
        coeffs = gauss_solve(matrix).data
```

```python
        approx_y = []
        for i in self.x:
            approx_y.append(coeffs[0] + coeffs[1] * i)
        return Results(False, "OK", {
            "coeffs": coeffs, "approx_y": approx_y, "type": ApproximationType.LINEAR
        })

    def poly_square(self):
        matrix = Matrix(3,
                        [
                            [self.n, self.sx, self.sx2, self.sy],
                            [self.sx, self.sx2, self.sx3, self.syx],
                            [self.sx2, self.sx3, self.sx4, self.syx2]
                        ])
        coeffs = gauss_solve(matrix).data
        approx_y = []
        for i in self.x:
            approx_y.append(coeffs[0] + coeffs[1] * i + coeffs[2] * i ** 2)
        return Results(False, "OK", {
            "coeffs": coeffs, "approx_y": approx_y, "type": ApproximationType.POLY_SQUARE
        })

    def poly_cubic(self):
        matrix = Matrix(4,
                        [
                            [self.n, self.sx, self.sx2, self.sx3, self.sy],
                            [self.sx, self.sx2, self.sx3, self.sx4, self.syx],
                            [self.sx2, self.sx3, self.sx4, self.sx5, self.syx2],
                            [self.sx3, self.sx4, self.sx5, self.sx6, self.syx3]
                        ])
        coeffs = gauss_solve(matrix).data
        approx_y = []
        for i in self.x:
            approx_y.append(coeffs[0] + coeffs[1] * i + coeffs[2] * i ** 2 + coeffs[3] * i ** 3)
        return Results(False, "OK", {
            "coeffs": coeffs, "approx_y": approx_y, "type": ApproximationType.POLY_CUBIC
        })

    def exponential(self):
        if None in [self.slny, self.sxlny]:
            return Results(True,
                "Cannot proceed exponential approximation for y < 0",
                {"coeffs": [], "approx_y": []}
            )
        matrix = Matrix(2, [
            [self.n, self.sx, self.slny],
            [self.sx, self.sx2, self.sxlny]
        ])
        coeffs = gauss_solve(matrix).data
        coeffs[0] = e ** coeffs[0]

        approx_y = []
        for i in self.x:
            approx_y.append(coeffs[0] * e ** (coeffs[1] * i))
        return Results(False, "OK", {
            "coeffs": coeffs, "approx_y": approx_y, "type": ApproximationType.EXPONENTIAL
        })

    def logarithmic(self):
        if None in [self.slnx, self.slnx2, self.sylnx]:
            return Results(True,
```

```
                "Cannot proceed logarithmic approximation for x < 0",
                {"coeffs": [], "approx_y": []}
            )

        matrix = Matrix(2, [
            [self.slnx2, self.slnx, self.sylnx],
            [self.slnx, self.n, self.sy]
        ])
        coeffs = gauss_solve(matrix).data

        approx_y = []
        for i in self.x:
            approx_y.append(coeffs[0] * log(i) + coeffs[1])
        return Results(False, "OK", {
            "coeffs": coeffs, "approx_y": approx_y, "type": ApproximationType.LOGARITHMIC
        })

    def power(self):
        if None in [self.slnx, self.slnx2, self.slny, self.slnxlny]:
            return Results(True,
                "Cannot proceed power approximation for x < 0 or y < 0",
                {"coeffs": [], "approx_y": []}
            )

        matrix = Matrix(2, [
            [self.n, self.slnx, self.slny],
            [self.slnx, self.slnx2, self.slnxlny]
        ])
        coeffs = gauss_solve(matrix).data
        coeffs[0] = e ** coeffs[0]
        approx_y = []
        for i in self.x:
            approx_y.append(coeffs[0] * i ** coeffs[1])
        return Results(False, "OK", {
            "coeffs": coeffs,
            "approx_y": approx_y,
            "type": ApproximationType.POWER
        })
```

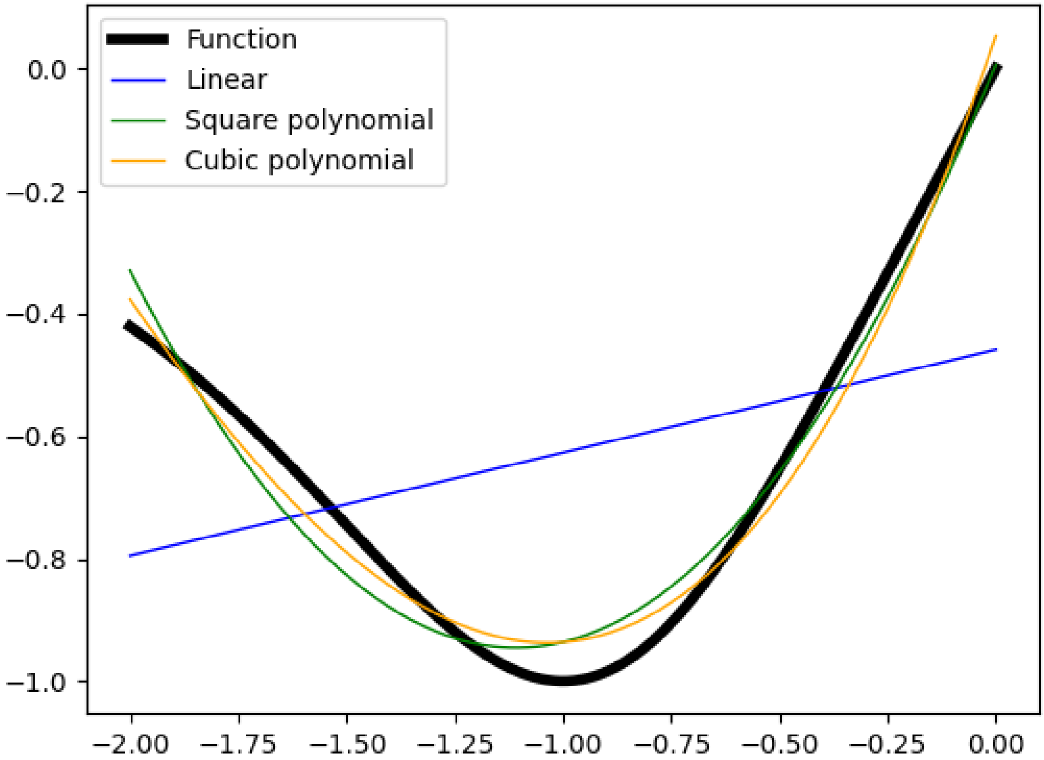
Полный код программы доступен по ссылке.

4 Результат работы программы

```
[1] Keyboard
[2] File

Your choice: 2
Specify file name: 1
Cannot proceed exponential approximation for y < 0
Cannot proceed logarithmic approximation for x < 0
Cannot proceed power approximation for x < 0 or y < 0


Linear approximation
|
| Coefficients:
|
 \
  |        -0.459, 0.168
  |
 /
|
| Standard deviation:
|
 \
  |        2.79e-01
  |
 /
|
| Pearson correlation coefficient:
|
 \
  |        3.57e-01
  |
 /
|
|

-----------------------------------------------------
|     x      |     y      |    f(x)    |    eps     |
|____________|____________|____________|____________|
|  -2.00e+00 |  -4.21e-01 |  -7.95e-01 |  -3.74e-01 |
|  -1.80e+00 |  -5.33e-01 |  -7.61e-01 |  -2.28e-01 |
|  -1.60e+00 |  -6.70e-01 |  -7.28e-01 |  -5.78e-02 |
|  -1.40e+00 |  -8.19e-01 |  -6.94e-01 |   1.25e-01 |
|  -1.20e+00 |  -9.46e-01 |  -6.61e-01 |   2.85e-01 |
|  -1.00e+00 |  -1.00e+00 |  -6.27e-01 |   3.73e-01 |
|  -8.00e-01 |  -9.39e-01 |  -5.93e-01 |   3.46e-01 |
|  -6.00e-01 |  -7.67e-01 |  -5.60e-01 |   2.07e-01 |
|  -4.00e-01 |  -5.29e-01 |  -5.26e-01 |   2.80e-03 |
|  -2.00e-01 |  -2.67e-01 |  -4.93e-01 |  -2.26e-01 |
|   0.00e+00 |   0.00e+00 |  -4.59e-01 |  -4.59e-01 |
|____________|____________|____________|____________|



Square polynomial approximation
|
| Coefficients:
|
 \
```

```
  |        0.006, 1.716, 0.774
  |
 /
|
| Standard deviation:
|
 \
  |        5.48e-02
  |
 /
|
|

-----------------------------------------------------
|     x      |     y      |    f(x)    |    eps     |
|____________|____________|____________|____________|
|  -2.00e+00 |  -4.21e-01 |  -3.30e-01 |   9.10e-02 |
|  -1.80e+00 |  -5.33e-01 |  -5.75e-01 |  -4.20e-02 |
|  -1.60e+00 |  -6.70e-01 |  -7.58e-01 |  -8.82e-02 |
|  -1.40e+00 |  -8.19e-01 |  -8.79e-01 |  -6.04e-02 |
|  -1.20e+00 |  -9.46e-01 |  -9.39e-01 |   7.36e-03 |
|  -1.00e+00 |  -1.00e+00 |  -9.36e-01 |   6.40e-02 |
|  -8.00e-01 |  -9.39e-01 |  -8.71e-01 |   6.76e-02 |
|  -6.00e-01 |  -7.67e-01 |  -7.45e-01 |   2.20e-02 |
|  -4.00e-01 |  -5.29e-01 |  -5.57e-01 |  -2.76e-02 |
|  -2.00e-01 |  -2.67e-01 |  -3.06e-01 |  -3.92e-02 |
|   0.00e+00 |   0.00e+00 |   6.00e-03 |   6.00e-03 |
|____________|____________|____________|____________|



Cubic polynomial approximation
|
| Coefficients:
|
 \
  |        0.053, 2.085, 1.257, 0.161
  |
 /
|
| Standard deviation:
|
 \
  |        4.55e-02
  |
 /
|
|

-----------------------------------------------------
|     x      |     y      |    f(x)    |    eps     |
|____________|____________|____________|____________|
|  -2.00e+00 |  -4.21e-01 |  -3.77e-01 |   4.40e-02 |
|  -1.80e+00 |  -5.33e-01 |  -5.66e-01 |  -3.33e-02 |
|  -1.60e+00 |  -6.70e-01 |  -7.25e-01 |  -5.45e-02 |
|  -1.40e+00 |  -8.19e-01 |  -8.44e-01 |  -2.51e-02 |
|  -1.20e+00 |  -9.46e-01 |  -9.17e-01 |   2.89e-02 |
|  -1.00e+00 |  -1.00e+00 |  -9.36e-01 |   6.40e-02 |
|  -8.00e-01 |  -9.39e-01 |  -8.93e-01 |   4.60e-02 |
|  -6.00e-01 |  -7.67e-01 |  -7.80e-01 |  -1.33e-02 |
|  -4.00e-01 |  -5.29e-01 |  -5.90e-01 |  -6.12e-02 |
```

```
|   -2.00e-01 |   -2.67e-01 |   -3.15e-02 |   -4.80e-02 |
|    0.00e+00 |    0.00e+00 |    5.30e-02 |    5.30e-02 |
|____________|____________|____________|____________|

---------------------------------------
Best approximation: Cubic polynomial
---------------------------------------
```



5 Вывод

Для наиболее оптимального распределения человеческих ресурсов и поддержания количества нервных клеток в головном мозгу, настоятельно рекомендуется использовать готовые библиотеки, содержащие наиболее эффективные реализации алгоритмов, вместо самостоятельной реализации оных.

Иными словами, вообще лучше зачиллиться и не изобретать велосипед.