Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №4

по «Вычислительной математике»

Вариант 4

Выполнил:

Студент группы P3208

Дашкевич Егор Вячеславович

Преподаватели:

Машина Екатерина Алексеевна

Санкт-Петербург

2024

Оглавление

Элементы оглавления не найдены.

Цель работы

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

Текст задания

Вычислительная реализация задачи

Задание:

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

Для исследования использовать:

- линейную функцию,
- полиномиальную функцию 2-й степени,
- полиномиальную функцию 3-й степени,
- экспоненциальную функцию,
- логарифмическую функцию,
- степенную функцию.

Задание:

1. Предусмотреть ввод исходных данных из файла/консоли (таблица ( ) 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 ;

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;

Вычислительная часть

Исследуемая функция:

$$\boldsymbol{y = \frac{15x}{x^4 + 4}, \qquad x \in [-4;\ 0], \qquad h = 0,4}$$

x_i	-4	-3.6	-3.2	-2.8	-2.4	-2	-1.6	-1.2	-0.8	-0.4	0
y_i	-0.231	-0.314	-0.441	-0.642	-0.968	-1.5	-2.274	-2.964	-2.721	-1.49	0

Линейное приближение:

$$\boldsymbol{\varphi(x) = ax + b}$$

$$\boldsymbol{SX = \sum_{i=1}^{n} x_i = -22, \qquad SXX = \sum_{i=1}^{n} x_i^2 = 61.6}$$

$$SY = \sum_{i=1}^{n} y_i = -13.545, \qquad SXY = \sum_{i=1}^{n} y_i x_i \approx 20.553$$

$$\begin{cases} a = \dfrac{SXY * n - SX * SY}{SXX * n - SX * SX} \\ b = \dfrac{SXX * SY - SX * SXY}{SXX * n - SX * SX} \end{cases} = \begin{cases} a \approx -0.371 \\ b \approx -1.974 \end{cases}$$

$$\varphi(x) = -0.371x - 1.974$$

x_i	-4	-3.6	-3.2	-2.8	-2.4	-2	-1.6	-1.2	-0.8	-0.4	0
y_i	-0.231	-0.314	-0.441	-0.642	-0.968	-1.5	-2.274	-2.964	-2.721	-1.49	0
$\varphi(x_i)$	-0.49	-0.638	-0.787	-0.935	-1.084	-1.232	-1.38	-1.529	-1.677	-1.826	-1.974
ε_i	-0.259	-0.324	-0.346	-0.293	-0.116	0.268	0.894	1.435	1.044	-0.337	-1.974

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} \varepsilon_i^2}{n}} \approx 0.875$$

Квадратичное приближение:

$$\varphi(x) = a + bx + cx^2$$

$$SX = \sum_{i=1}^{n} x_i = -22, \qquad SXX = \sum_{i=1}^{n} x_i^2 = 61.6, \qquad S3X = \sum_{i=1}^{n} x_i^3 = -193.6$$

$$S4X = \sum_{i=1}^{n} x_i^4 \approx 648.525, \qquad SY = \sum_{i=1}^{n} y_i = -13.545,$$

$$SXY = \sum_{i=1}^{n} y_i x_i \approx 20.553, \qquad SXXY = \sum_{i=1}^{n} y_i x_i^2 = -40.954$$

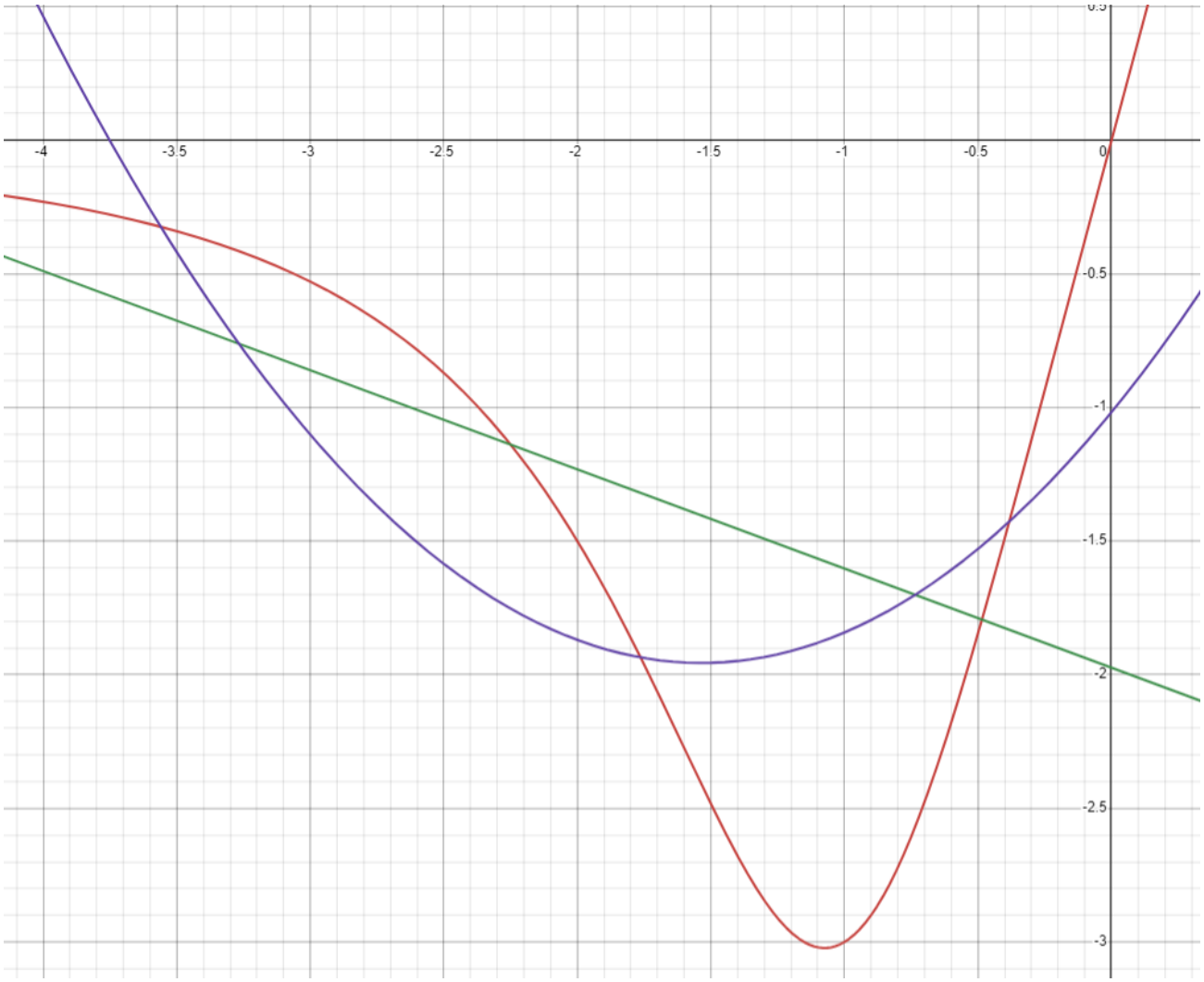
$$\begin{cases} na + SX * b + SXX * c = SY \\ SX * a + SXX * b + S3X * c = SXY \\ SXX * a + S3X * b + S4X * c = SXXY \end{cases} = \begin{cases} a \approx -1.018 \\ b \approx 1.222 \\ c \approx 0.398 \end{cases}$$

$$\varphi(x) = -1.018 + 1.222x + 0.398x^2$$

x_i	-4	-3.6	-3.2	-2.8	-2.4	-2	-1.6	-1.2	-0.8	-0.4	0
y_i	-0.231	-0.314	-0.441	-0.642	-0.968	-1.5	-2.274	-2.964	-2.721	-1.49	0
$\varphi(x_i)$	0.462	-0.259	-0.853	-1.319	-1.658	-1.87	-1.954	-1.911	-1.741	-1.443	-1.018
ε_i	0.693	0.055	-0.412	-0.677	-0.69	-0.37	-0.32	-1.053	-0.98	0.047	-1.018

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} \varepsilon_i^2}{n}} \approx 0,67$$

Итого: квадратичное приближение дало более близкий результат нежели линейное (его среднеквадратичное отклонение меньше)



Листинг программы:

Линейная:

```
def aproximate(points):
    SX, SXX, SY, SXY = 0, 0, 0, 0
    for p in points:
        SX += p[0]
        SXX += p[0] ** 2
        SY += p[1]
        SXY += p[0] * p[1]

    a = (SXY * len(points) - SX*SY) / (SXX * len(points) - SX*SX)
    b = (SXX * SY - SX * SXY) / (SXX * len(points) - SX*SX)
    return Function( koofs: [a, b], FunctionType.linear)
```

Экспоненциальная:

```
def aproximate(points):
    _points = copy.deepcopy(points)
    for i in range(len(_points)):
        _points[i][1] = m.log(_points[i][1], m.e)

    koofs = linear.aproximate(_points).get_koofs()
    koofs.reverse()
    koofs[0] = m.exp(koofs[0])

    return Function(koofs, FunctionType.exponent)
```

Степенная:

```
def aproximate(points):
    _points = copy.deepcopy(points)
    for i in range(len(_points)):
        _points[i][0] = m.log(_points[i][0], m.e)
        _points[i][1] = m.log(_points[i][1], m.e)

    koofs = linear.aproximate(_points).get_koofs()
    koofs.reverse()
    koofs[0] = m.exp(koofs[0])

    return Function(koofs, FunctionType.power)
```

Полиноминальная:

```
def aproximate(points, degree):
    if degree < 2 or degree > 8:
        print("Incorrect degree entered", file=sys.stderr)
        sys.exit(-1)

    A = [[0 for i in range(degree + 1)] for j in range(degree + 1)]
    B = []

    A[0][0] = len(points)
    for i in range(1, degree*2+1):
        _val = 0
        for p in points:
            _val += p[0]**i
        fill_rev_diag(list(A), _val, i)

    for i in range(degree+1):
        _val = 0
        for p in points:
            _val += (p[0] ** i) * p[1]
        B.append(_val)

    _koofs = solve_slau(A, B)

    return Function(_koofs, FunctionType.polynomial)
```

Логарифмическая:

```python
def aproximate(points):
    _points = copy.deepcopy(points)

    for i in range(len(_points)):
        _points[i][0] = m.log(points[i][0], m.e)

    koofs = linear.aproximate(_points).get_koofs()
    return Function(koofs, FunctionType.logarithm)
```

Вывод

В ходе выполнения работы изучил метод наименьших квадратов для различных типов функций, реализовал их на ЭВМ.