Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 4

Тема Построение и программная реализация алгоритма наилучшего среднеквадратичного приближения

Студент Челядинов Илья

Группа ИУ7-43Б

Оценка (баллы) _______________

Преподаватель Градов Владимир Михайлович

Москва.
2020 г.

Тема: Построение и программная реализация алгоритма наилучшего среднеквадратичного приближения.

Цель работы. Получение навыков построения алгоритма метода наименьших квадратов с использованием полинома заданной степени при аппроксимации табличных функций с весами.

Исходные данные.

1. Таблица функции с **весами** ρ_i с количеством узлов N.

x	y	ρ_i

Предусмотреть в интерфейсе удобную возможность изменения пользователем весов в таблице.

2. Степень аппроксимирующего полинома - n.

Результат работы программы.

Графики, построенные по аналогии с рис.1 в тексте Лекции №4: *точки* - заданная табличная функция, *кривые*- найденные полиномы.

Обязательно приводить таблицы, по которым работала программа.

При каких исходных условиях надо представить результаты в отчете?

1. Веса всех точек одинаковы и равны, например, единице. Обязательно построить полиномы при значениях его степени n=1, 2. Можно привести результаты и при других степенях полинома, однако не загромождая сильно при этом рисунок.

2. Веса точек разные. Продемонстрировать, как за счет назначения весов точкам можно изменить положение на плоскости прямой линии (полином первой степени), аппроксимирующей **один и тот же набор точек** (**одну** таблицу y(x)). Например, назначая веса узлам в таблице изменить знак углового коэффициента прямой. На графике в итоге должны быть представлены точки исходной функции и две аппроксимирующие их прямые линии. Одна отвечает значениям ρ_i =1 для всех

узлов, а другая- назначенным разным весам точек. Информацию о том, какие именно веса были использованы в расчете обязательно указать, чтобы можно было проконтролировать работу программы (лучше это сделать в виде таблицы).

Описание алгоритма

Под близостью в среднем исходной и аппроксимирующей функций будем понимать результат оценки суммы

$$I=\sum_{i=1}^{N}\rho_i[y(x_i)-\varphi(x_i)]^2 \qquad (1)$$

Y(x) – исходная функция, ϕ(x) – множество функций, принадлежащих линейному пространству функций, где ρ_i - вес точки. Суммирование выполняется по всем N узлам заданной функции. Наилучшее приближение в данном случае

$$\sum_{i=1}^{N}\rho_i\,[\,y(x_i)-\varphi(x_i)\,]^2=min \qquad (2)$$

Разложим ϕ(x) по системе линейно независимых функий $\phi_k(x)$:

$$\varphi(x)=\sum_{k=0}^{n}a_k\varphi_k(x) \qquad (3)$$

Подставляя (3) в (2) получим:

$$((y-\varphi),(y-\varphi))=(y,y)-2\sum_{k=0}^{n}a_k(y,\varphi_k)+\sum_{k=0}^{n}\sum_{m=0}^{n}a_k a_m(\varphi_k,\varphi_m)=\min \qquad (4)$$

Дифференцируя по a_k получаем:

$$\sum_{m=0}^{n}(x^k,x^m)\;a_m=(y,x^k)\;,\;0\le k\le n$$

Где

$$(x^k,x^m)=\sum_{i=1}^{N}\rho_i x_i^{k+m},\quad (y,x^k)=\sum_{i=1}^{N}\rho_i y_i x_i^k\;.$$

Код программы

```
def f(x_arr, coeff):
    res = np.zeros(len(x_arr))
    for i in range(len(coeff)):
        res += coeff[i] * (x_arr ** i)
    return res


### Считать данные с файла
def read_from_file(filename):
    f = open(filename, "r")
    x, y, ro = [], [], []
    for line in f:
        line = line.split(" ")
        x.append(float(line[0]))
        y.append(float(line[1]))
        ro.append(float(line[2]))
    return x, y, ro
```

```
def print_table(x, y, ro):
    length = len(x)
    print("x       y       ro")
    for i in range(length):
        print("%.4f %.4f %.4f" % (x[i], y[i], ro[i]))
    print()


def print_matr(matr):
    for i in matr:
        print(i)
```

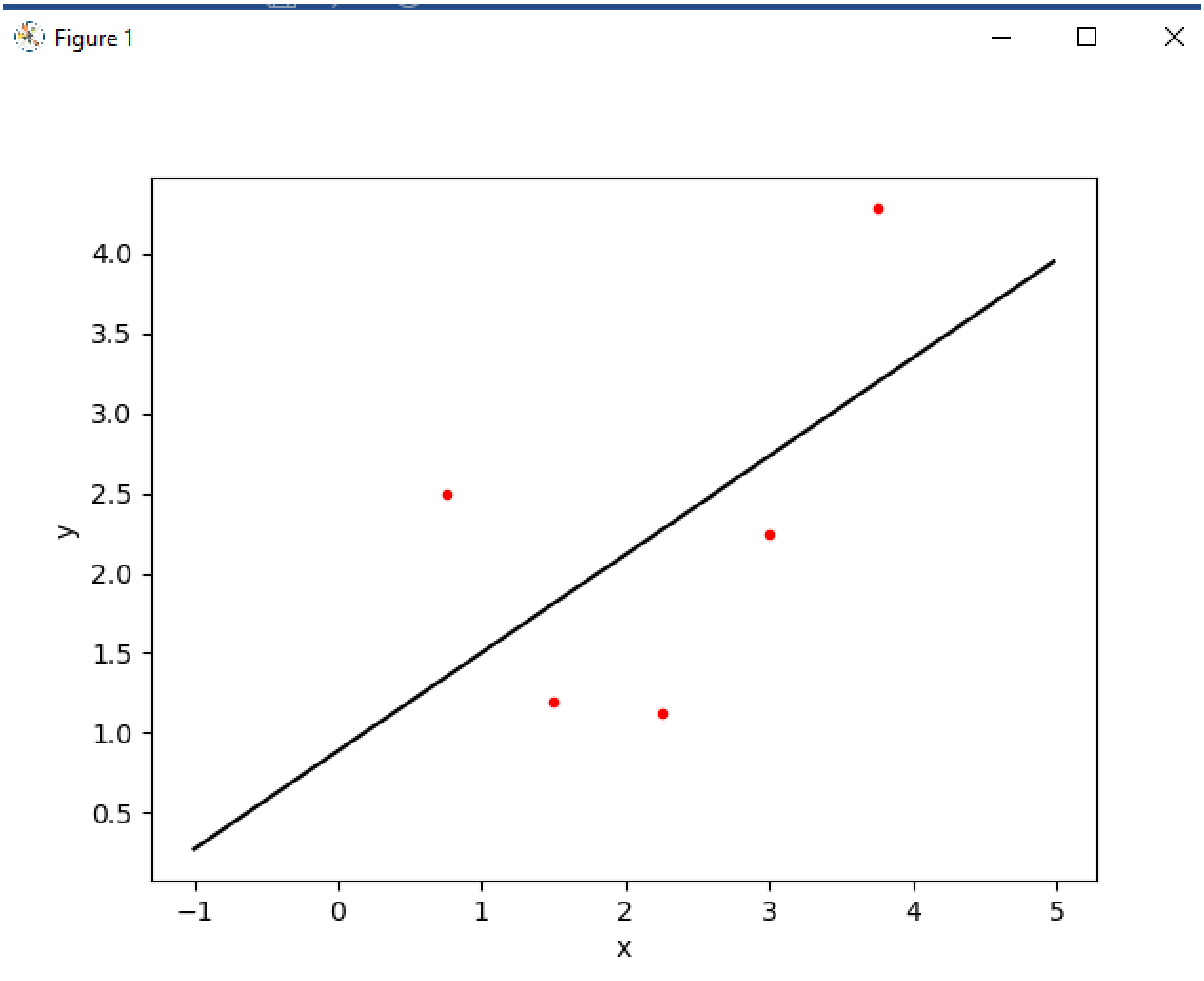
```python
### Вычислить значение
def find_root(x, y, ro, n):  # n - кол-во искомых коэффициентов
    length = len(x)
    sum_x_n = [sum([x[i] ** j * ro[i] for i in range(length)]) for j in range(n * 2 - 1)]
    sum_y_x_n = [sum([x[i] ** j * ro[i] * y[i] for i in range(length)]) for j in range(n)]
    matr = [sum_x_n[i:i + n] for i in range(n)]
    for i in range(n):
        matr[i].append(sum_y_x_n[i])
    print_matr(matr)
    return Gauss(matr)


def Gauss(matr):
    n = len(matr)
    # приводим к треугольному виду
    for k in range(n):
        for i in range(k + 1, n):
            coeff = -(matr[i][k] / matr[k][k])
            for j in range(k, n + 1):
                matr[i][j] += coeff * matr[k][j]
    print("\ntriangled:")
    print_matr(matr)
    # находим неизвестные
    a = [0 for i in range(n)]
    for i in range(n - 1, -1, -1):
        for j in range(n - 1, i, -1):
            matr[i][n] -= a[j] * matr[i][j]
        a[i] = matr[i][n] / matr[i][i]
    return a
### Отобразить результат
def paint_plot(a):
    t = np.arange(-1.0, 5.0, 0.02)
    plt.figure(1)
    plt.ylabel("y")
    plt.xlabel("x")
    plt.plot(t, f(t, a), 'k')
    for i in range(len(x)):
        plt.plot(x[i], y[i], 'ro', markersize=ro[i] + 2)
    plt.show()


x, y, ro = read_from_file("data.txt")
n = 1  # Степень многочлена
print_table(x, y, ro)
a = find_root(x, y, ro, n + 1)
print("\na:", a)
paint_plot(a)
```

Примеры работы программы

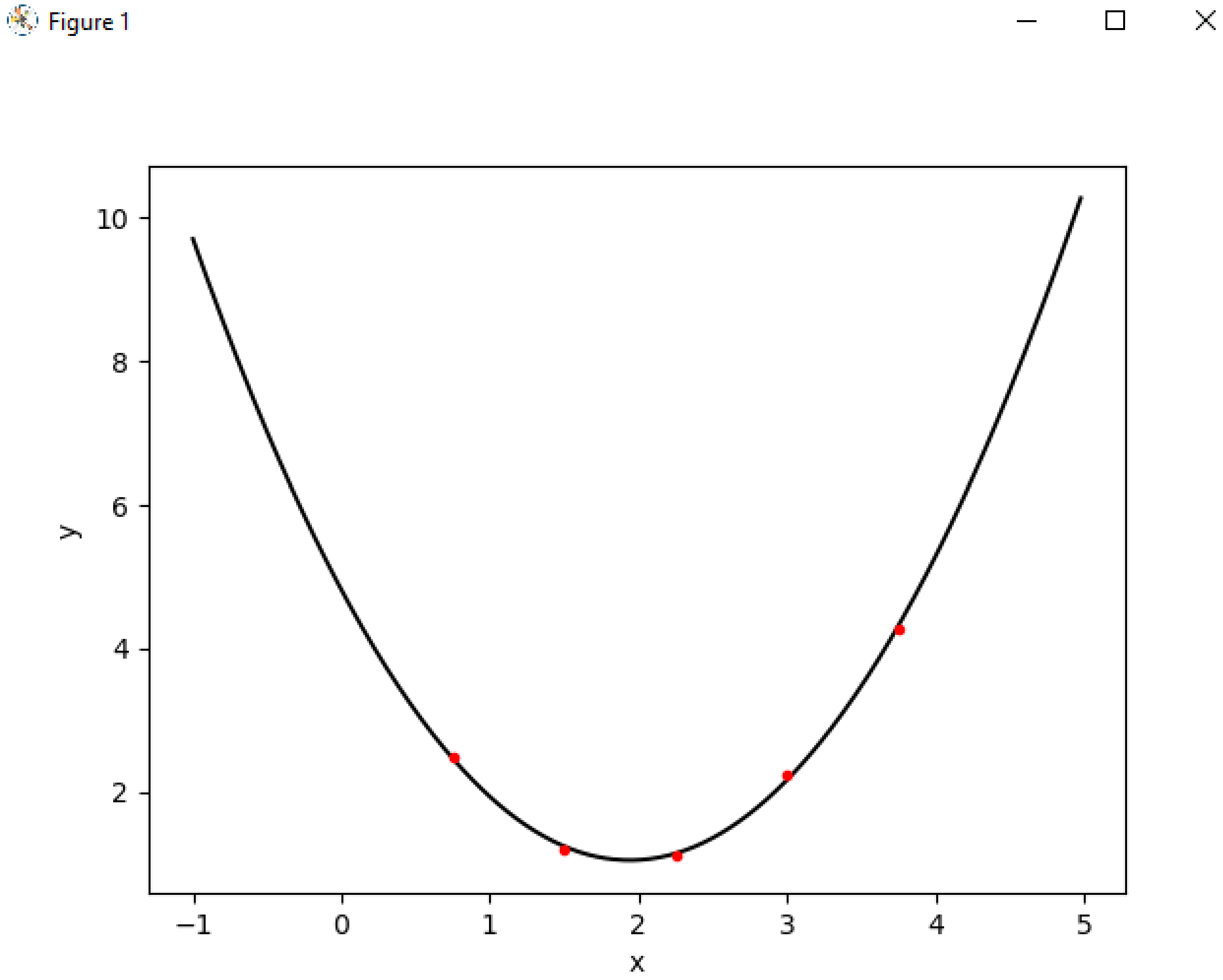
Входные данные:

```
x      y      ro
0.7500 2.5000 1.0000
1.5000 1.2000 1.0000
2.2500 1.1200 1.0000
3.0000 2.2500 1.0000
3.7500 4.2800 1.0000
```

Степень n = 1



n = 2

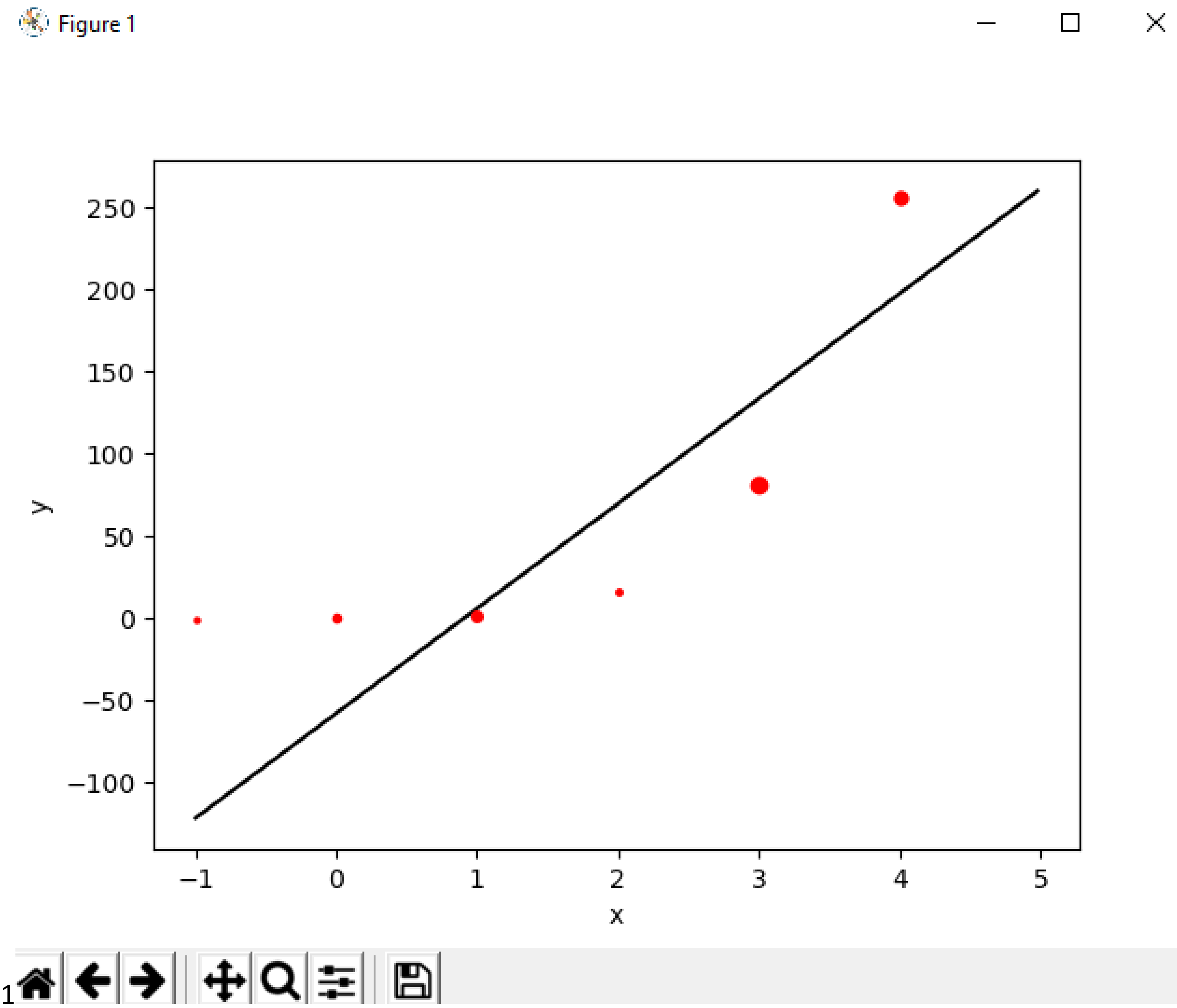


Пример 2.

Входные данные:

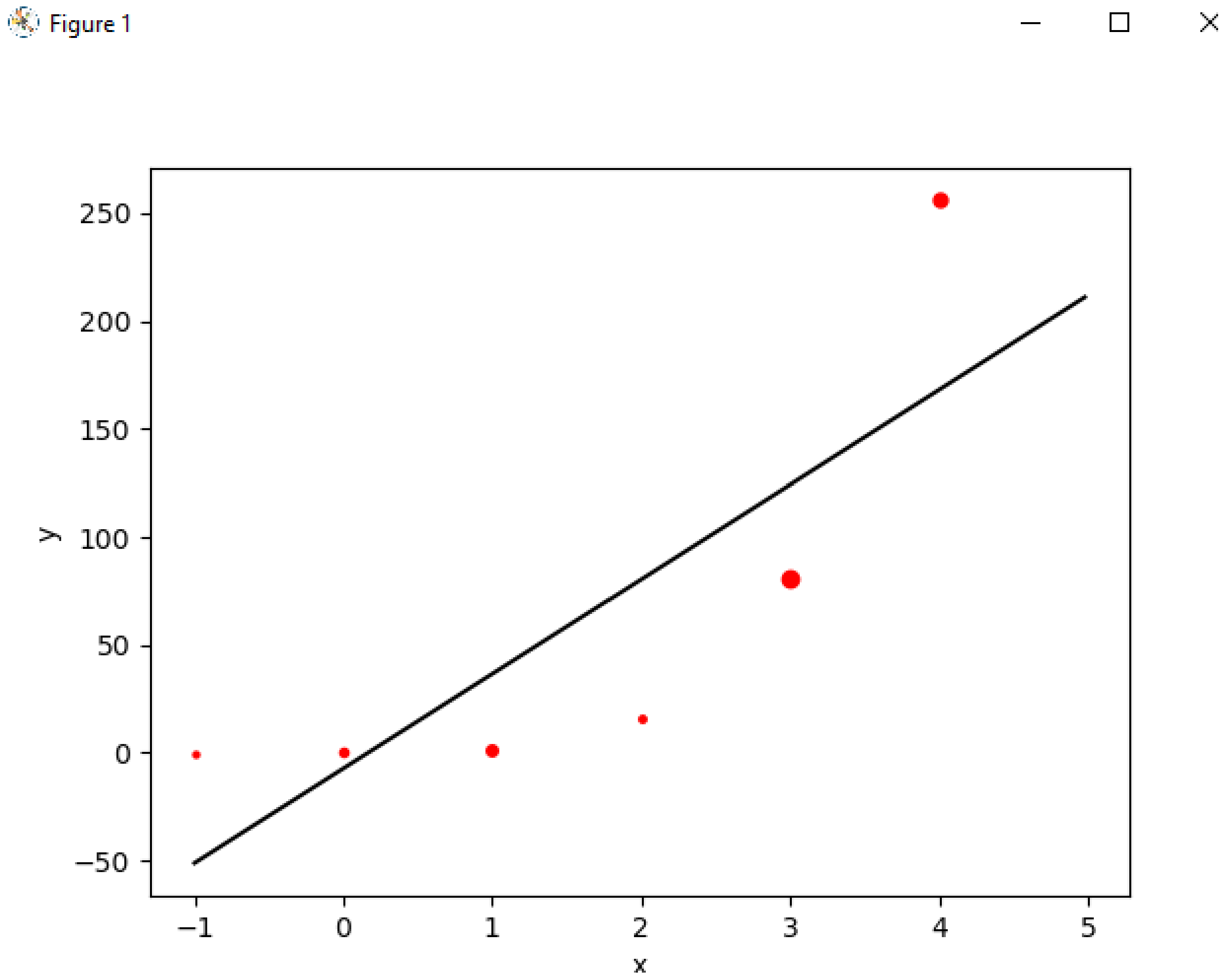
```
x       y       ro
-1.0000 -1.0000 0.1000
0.0000 0.0000 1.0000
1.0000 1.0000 2.0000
2.0000 16.0000 0.5000
3.0000 81.0000 4.0000
4.0000 256.0000 3.0000
```

n = 1



n = 1

После изменения знака углового коэффицента прямой:



Ответы на контрольные вопросы.

1. ***Что произойдет при задании степени полинома n=N-1 (числу узлов таблицы минус 1)?***

   При степени полинома n = N – 1 функция пройдет через все точки, независимо от их веса

2. **Будет ли работать Ваша программа при n >= N? Что именно в алгоритме требует отдельного анализа данного случая и может привести к аварийной остановке?**

   По N точкам нельзя построить полином степени n, так как в данном случае определитель будет равен нулю. Но, программа не завершится аварийно и будет работать из – за погрешности при работе с действительными числами.

3. **Получить формулу для коэффициента полинома a_0 при степени полинома n = 0. Какой смысл имеет величина, которую представляет данный коэффициент?**

   Формула:

$$\frac{\sum_{i=1}^{N} y_i \rho_i}{\sum_{i=1}^{N} \rho_i}$$

   Значение: математичское ожидание

4. ***Записать и вычислить определитель матрицы СЛАУ для нахождения коэффициентов полинома для случая, когда n=N=2. Принять все p_i=1.***

Пусть есть таблица:

x_i	y_i	ρ_i
x_0	y_0	ρ_1
x_1	y_1	ρ_2

Тогда имеем СЛАУ вида:

$$\begin{cases} (\rho_0+\rho_1)a_0 + (\rho_0 x_0 + \rho_1 x_1)a_1 + (\rho_0 x_0^2 + \rho_1 x_1^2)a_2 = \rho_0 y_0 + \rho_1 y_1 \\ (\rho_0 x_0 + \rho_1 x_1)a_0 + (\rho_0 x_0^2 + \rho_1 x_1^2)a_1 + (\rho_0 x_0^3 + \rho_1 x_1^3)a_2 = \rho_0 y_0 x_0 + \rho_1 y_1 x_1 \\ (\rho_0 x_0^2 + \rho_1 x_1^2)a_0 + (\rho_0 x_0^3 + \rho_1 x_1^3)a_1 + (\rho_0 x_0^4 + \rho_1 x_1^4)a_2 = \rho_0 y_0 x_0^2 + \rho_1 y_1 x_0^2 \end{cases}$$

Определитель в данном случае равен нулю, система не имеет решений.

5. ***Построить СЛАУ при выборочном задании степеней аргумента полинома*** $\varphi(x) = a_0 + a_1 x^m + a_2 x^n$ ***причем степени n и m в этой формуле известны***

СЛАУ будет выглядеть следующим образом:

$$\begin{cases} (x^0, x^0)a_0 + (x^0, x^m)a_1 + (x^0, x^n)a_2 = (y, x^0) \\ (x^m, x^0)a_0 + (x^m, x^m)a_1 + (x^m, x^n)a_2 = (y, x^m) \\ (x^n, x^0)a_0 + (x^n, x^m)a_1 + (x^n, x^n)a_2 = (y, x^n) \end{cases}$$