ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | К. А. Кочин |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| «Синтез и реализация алгоритмов оценивания. Метод наименьших квадратов» |
| по курсу: Прикладная теория вероятностей и статистика |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ гр. № | 4332 |  |  |  | А. А. Лютов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2025

**Задание:**

Исходные данные:

Таблица 1. Выборка 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Z1 | Z2 | Z3 | Z4 | Z5 |
| 13 | 0,6 | 1,8 | 1,9 | 2,6 | 4,0 |

Произведено N = 5 наблюдений:  где  – наблюдаемое значение параметра,  – истинное значение параметра,  – значение ошибки (помехи), распределенной по нормальному закону с нулевым средним , дисперсией  и плотностью вероятности .

Вектор наблюдений

|  |  |
| --- | --- |
|  |  |

Установлено, что реализация *x*(*t*) на интервале  путем подбора коэффициентов может быть с малой погрешностью (меньшей, чем дисперсия шума ) представлена в виде конечного ряда: , где *fm* – заданные (известные) функции времени, *m* = 1, 2, …, *М*. Модельные значения оцениваемой величины представляются в виде степенного ряда , где *M* = 1, 2, 3.

Для каждой модели ряда (, , )

выполнить:

1. найти коэффициенты аппроксимирующего полинома по реализации вектора наблюдения ;
2. вычислить оценки значений  на основании полученного полином;
3. построить график с исходными данными  и аппроксимирующим полиномом;
4. вычислить оценку  среднего квадратического отклонения Оценить точность полученных результатов;
5. сделать выводы по работе.

**Ход работы:**

При проведении *N* = 5 измерений получен вектор наблюдений . Найти вектор коэффициентов  и вектор оценок , если модельные значения оцениваемой величины представляются в виде степенного ряда:

|  |  |
| --- | --- |
|  | (1) |

где *M* = 1, 2, 3.

1. M = 1:

Функция квадрата невязок:

Найдём *a*0:

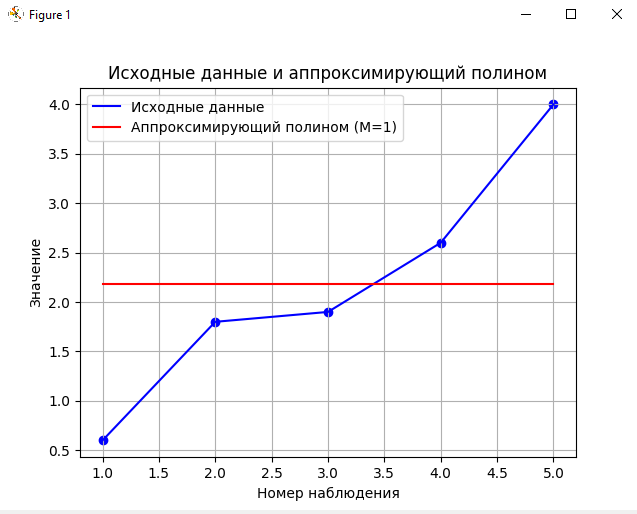
Теперь, используя соотношение (1), можно получить искомые оценки:

.

Оценка дисперсии: .

Оценку среднего квадратического отклонения: .

График:



1. M = 2:

Функция квадрата невязок:



Найдём и :





Получаем систему уравнений:

Решив систему уравнений, находим вектор коэффициентов: .

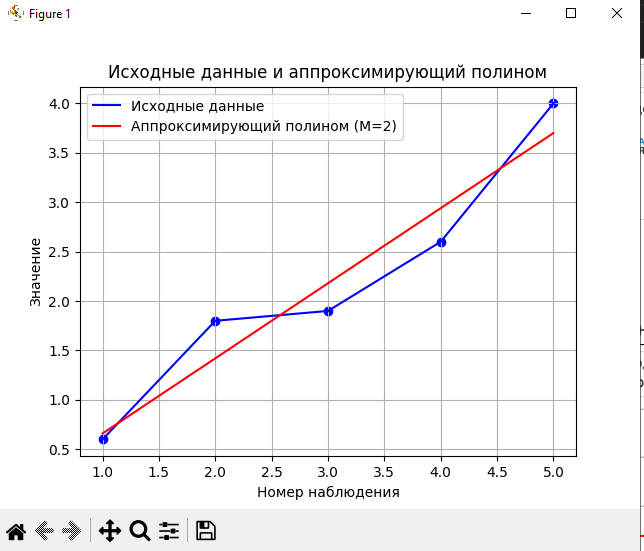
Теперь, используя соотношение (1), можно получить искомые оценки:

=> вектор оценок будет равен: .

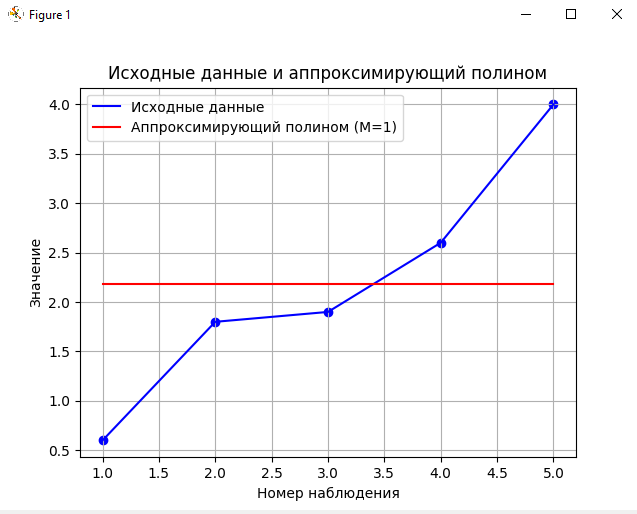
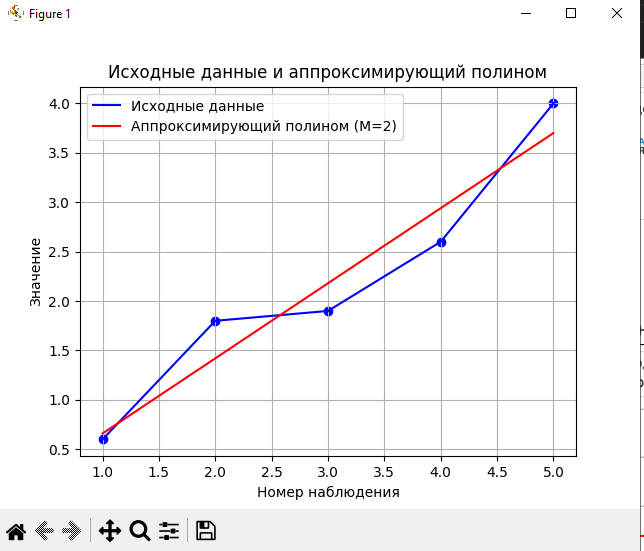
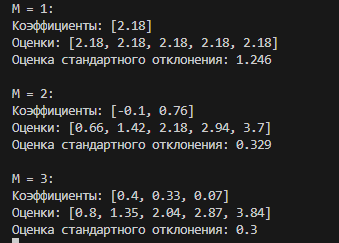
Оценка дисперсии:

Оценку среднего квадратического отклонения:

График:



**Результат:**

**Листинг:**

import numpy as np

import matplotlib.pyplot as plt

z = [0.6, 1.8, 1.9, 2.6, 4.0]

N = len(z)

def calculate\_coefficients(M):

    A = [[0] \* M for \_ in range(M)]

    B = []

    for i in range(M):

        for j in range(M):

            A[i][j] = sum([k \*\* (i + j) for k in range(1, N + 1)])

        B.append(sum(z[j] \* (j + 1) \*\* i for j in range(N)))

    return np.linalg.solve(A, B).tolist()

def calculate\_estimates(coefficients):

    estimates = [0]\*N

    for i in range(N):

        for j in range(len(coefficients)):

            estimates[i] += coefficients[j] \* (i + 1) \*\* j

    return estimates

def calculate\_std\_estimate(estimates):

    N = len(estimates)

    squared\_diff\_sum = sum([(z[i] - estimates[i]) \*\* 2 for i in range(N)])

    std\_estimate = (squared\_diff\_sum / (N - 1)) \*\* 0.5

    return std\_estimate

def mystd(data):

    mean = np.mean(data)

    variance = np.sqrt(sum((x - mean) \*\* 2 for x in data) / (N - 1))

    return variance

def plot\_data\_and\_polynomial(z, estimates, M):

    plt.plot(np.arange(1, N + 1), z, label='Исходные данные',color='blue')

    plt.scatter(np.arange(1, N + 1), z, color='blue')

    plt.plot(np.arange(1, N + 1), estimates, label=f'Аппроксимирующий полином (M={M})', color='red')

    plt.xlabel('Номер наблюдения')

    plt.ylabel('Значение')

    plt.title('Исходные данные и аппроксимирующий полином')

    plt.legend()

    plt.grid(True)

    plt.show()

def solve\_exercise(M\_values):

    for M in M\_values:

        coefficients = calculate\_coefficients(M)

        estimates = calculate\_estimates(coefficients)

        std\_estimate = calculate\_std\_estimate(estimates)

        print(f"M = {M}:")

        print("Коэффициенты:", [round(value, 2) for value in coefficients])

        print("Оценки:", [round(value, 2) for value in estimates])

        print("Оценка стандартного отклонения:", round(std\_estimate, 3))

M\_values = [1, 2, 3]

solve\_exercise(M\_values)

**Вывод:**

В результате выполнения данной лабораторной работы я ознакомился с методом наименьших квадратов, смог програмно построить аппроксимирующий полином, сделал следующие выводы по работе:

* При увеличении размера M дисперсия оценок обычно уменьшается.
* В полиномиальной регрессии увеличение степени полинома (увеличение M) обычно приводит к лучшему соответствию данным.
* Полином высшего порядка может лучше улавливать тонкости данных, уменьшая ошибку между оценочными значениями и фактическими наблюдениями. В результате дисперсия оценок уменьшается, что указывает на близость оценок к истинным значениям.