ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доцент, к. т. н., доцент |  |  |  | В. В. Мышко |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4 |
| Однофакторный регрессионный анализ |
| по курсу: Обработка экспериментальных данных |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4236 |  |  |  | Л. Мвале |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2026

1. **Цель**

На основе заданного массива данных построить уравнение регрессии в виде алгебраического полинома второй степени, проверить адекватность уравнения регрессии и значимость коэффициентов регрессии. Расчеты произвести в скалярной и матричной форме.

1. Постановка задачи

На основе заданного массива данных:

* построить уравнение регрессии в виде *алгебраического полинома второй степени*;
* проверить адекватность уравнения регрессии;
* проверить значимость коэффициентов регрессии.

Расчеты произвести в скалярной и матричной форме.

**Порядок выполнения задания:**

1. Составить систему нормальных уравнений, используя массив экспериментальных данных (таблица 4.1);
2. Найти оценки коэффициентов регрессии посредством решения системы нормальных уравнений;
3. При расчетах в матричной форме составить матричное уравнение с вектором неизвестных оценок коэффициентов регрессии и найти его решение;
4. Проверить адекватность построенного уравнения регрессии экспериментальным данным по критерию Фишера при уровне значимости α = 0,01;
5. Проверить значимость коэффициентов регрессии по критерию Стьюдента при таком же уровне значимости;
6. Повторно проверить адекватность уравнения регрессии после исключения незначимых коэффициентов.

Таблица 4.1

**Экспериментальные данные**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 97 | 4236-97 | x | -2 | 0 | 1 | 2 | 4 |
| y | -3 | 6 | 7 | 3 | -4 |

Объем выборки: n = 5

Уравнение регрессии: полином второй степени

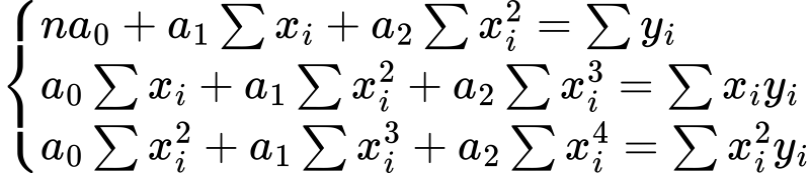
wps

1. Решение

**1. СКАЛЯРНАЯ ФОРМА РАСЧЕТА**

1.1. Составление системы нормальных уравнений

Для полинома второй степени вида wps система нормальных уравнений в скалярной форме имеет вид (согласно §9.2.2, формула 9.2.17):



1.2. Вычисление необходимых сумм

Составим расчетную таблицу:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **i** | **xᵢ** | **yᵢ** | **xᵢ²** | **xᵢ³** | **xᵢ⁴** | **xᵢyᵢ** | **xᵢ²yᵢ** |
| 1 | -2 | -3 | 4 | -8 | 16 | 6 | -12 |
| 2 | 0 | 6 | 0 | 0 | 0 | 0 | 0 |
| 3 | 1 | 7 | 1 | 1 | 1 | 7 | 7 |
| 4 | 2 | 3 | 4 | 8 | 16 | 6 | 12 |
| 5 | 4 | -4 | 16 | 64 | 256 | -16 | -64 |
| Σ | 5 | 9 | 25 | 65 | 289 | 3 | -57 |

wps

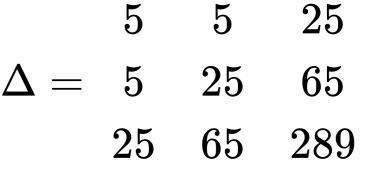
1.3. Система нормальных уравнений с числовыми коэффициентами

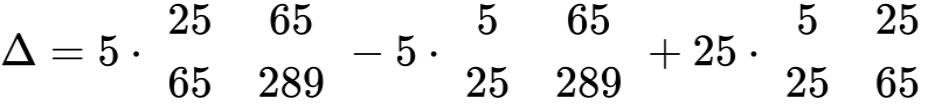
Подставляем вычисленные суммы в систему нормальных уравнений:



1.4. Решение системы нормальных уравнений методом Крамера

Вычисляем главный определитель системы:

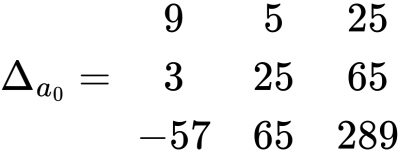


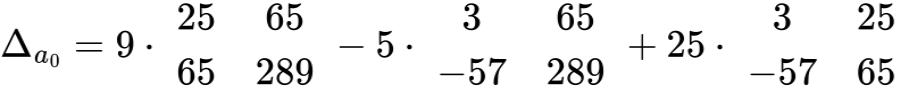


wps

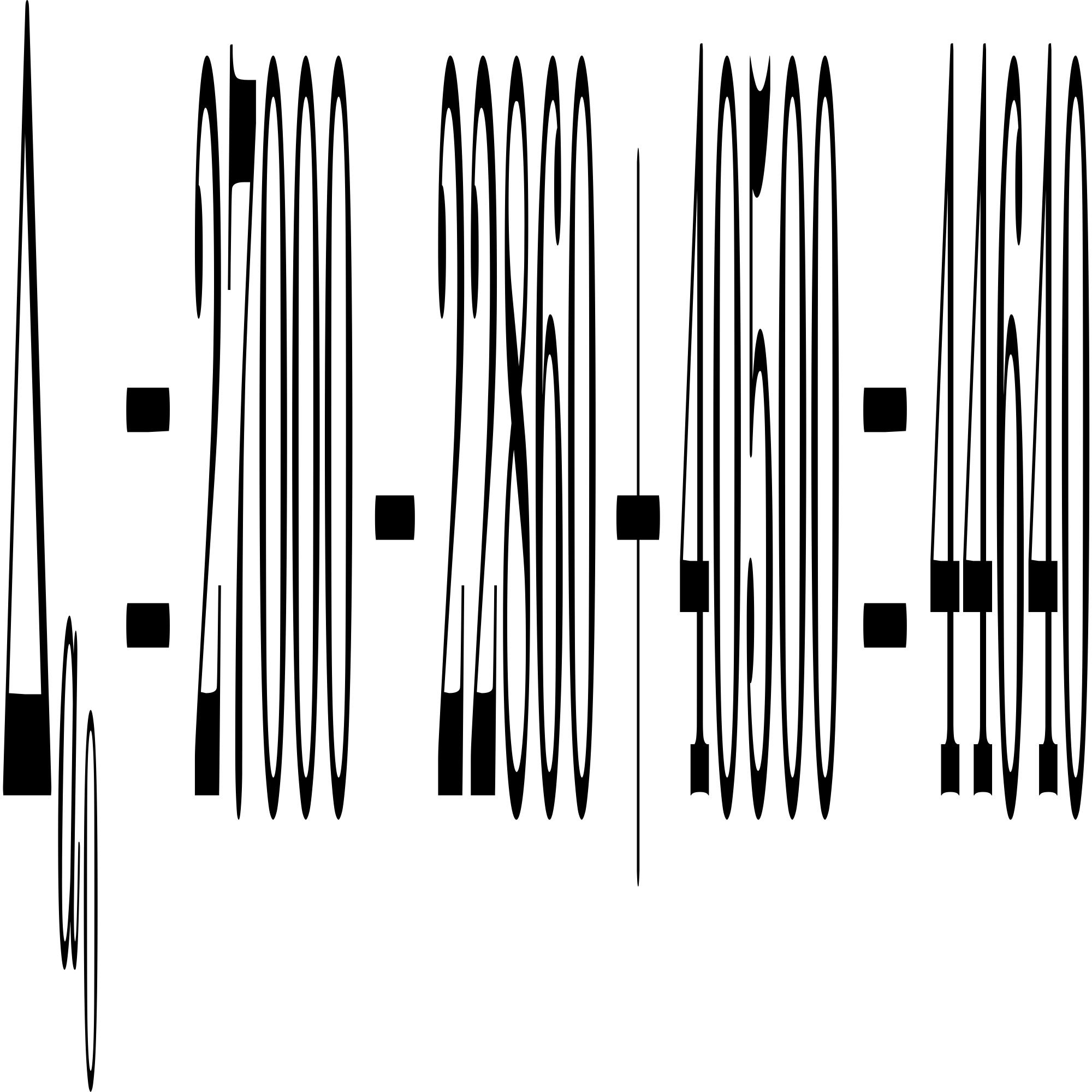
wps

Вычисляем определитель для wps:

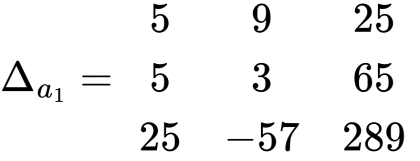


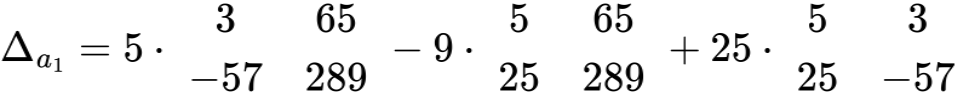


wps



Вычисляем определитель для wps:

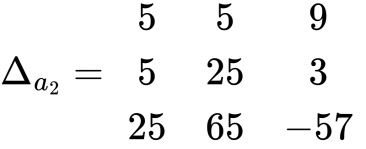


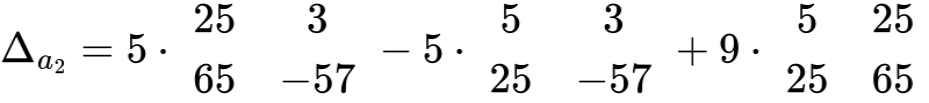


wps

wps

Вычисляем определитель для wps:

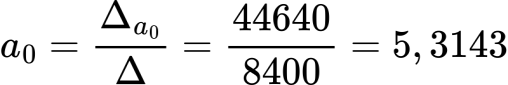


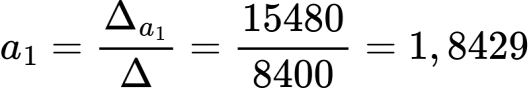


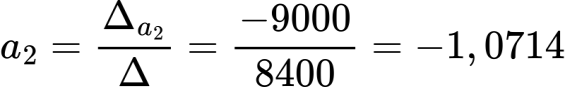
wps

wps

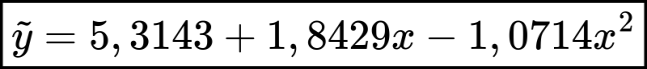
1.5. Оценки коэффициентов регрессии







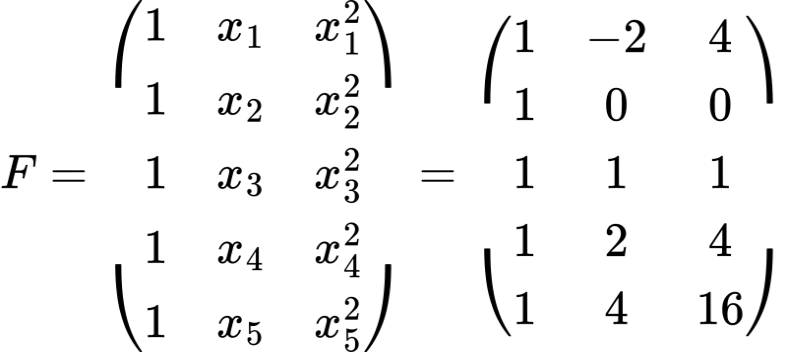
Таким образом, уравнение регрессии имеет вид:



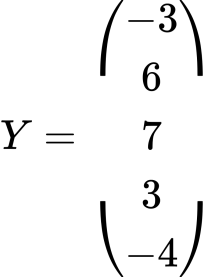
1. МАТРИЧНАЯ ФОРМА РАСЧЕТА

2.1. Составление матрицы F и вектора Y

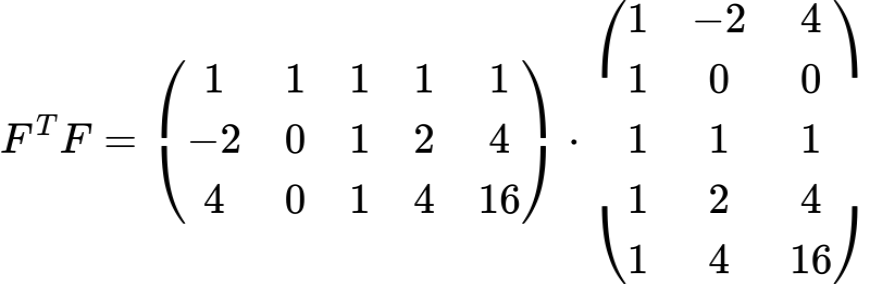
Для полинома второй степени матрица F имеет вид (согласно §9.2.2, формула 9.2.37):



Вектор наблюдений Y:



2.2. Вычисление матрицы wps



Вычисляем элементы матрицы wps:

(1,1): wps

(1,2):wps

(1,3): wps

(2,1): wps

(2,2):wps

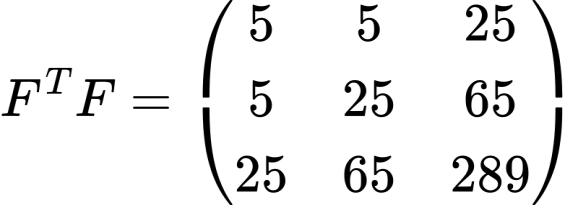
(2,3): wps

(3,1):wps

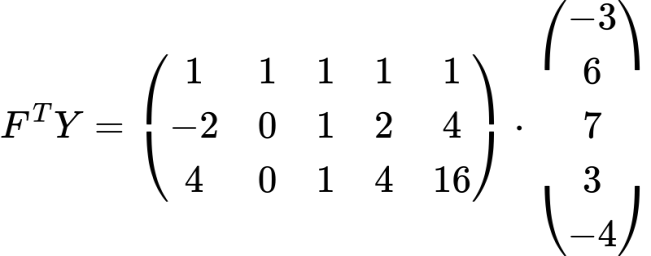
(3,2): wps

(3,3):wps

Таким образом:



2.3. Вычисление вектора wps



Вычисляем:

Первая компонента:

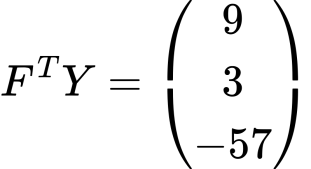
wps

Вторая компонента:

wps

Третья компонента:

wps



2.4. Решение матричного уравнения

Матричное уравнение для нахождения оценок коэффициентов регрессии:

wps

где wps

Решение в матричной форме:

wps

2.5.  Проверка решения методом Гаусса

Имеем систему:



Вычтем из второго уравнения первое:

wps

wps

wps

Вычтем из третьего уравнения первое, умноженное на 5:

wps

wps

wps

wps

Вычтем из уравнения (2) уравнение (1), умноженное на 2:

wps

wps

wps

wps

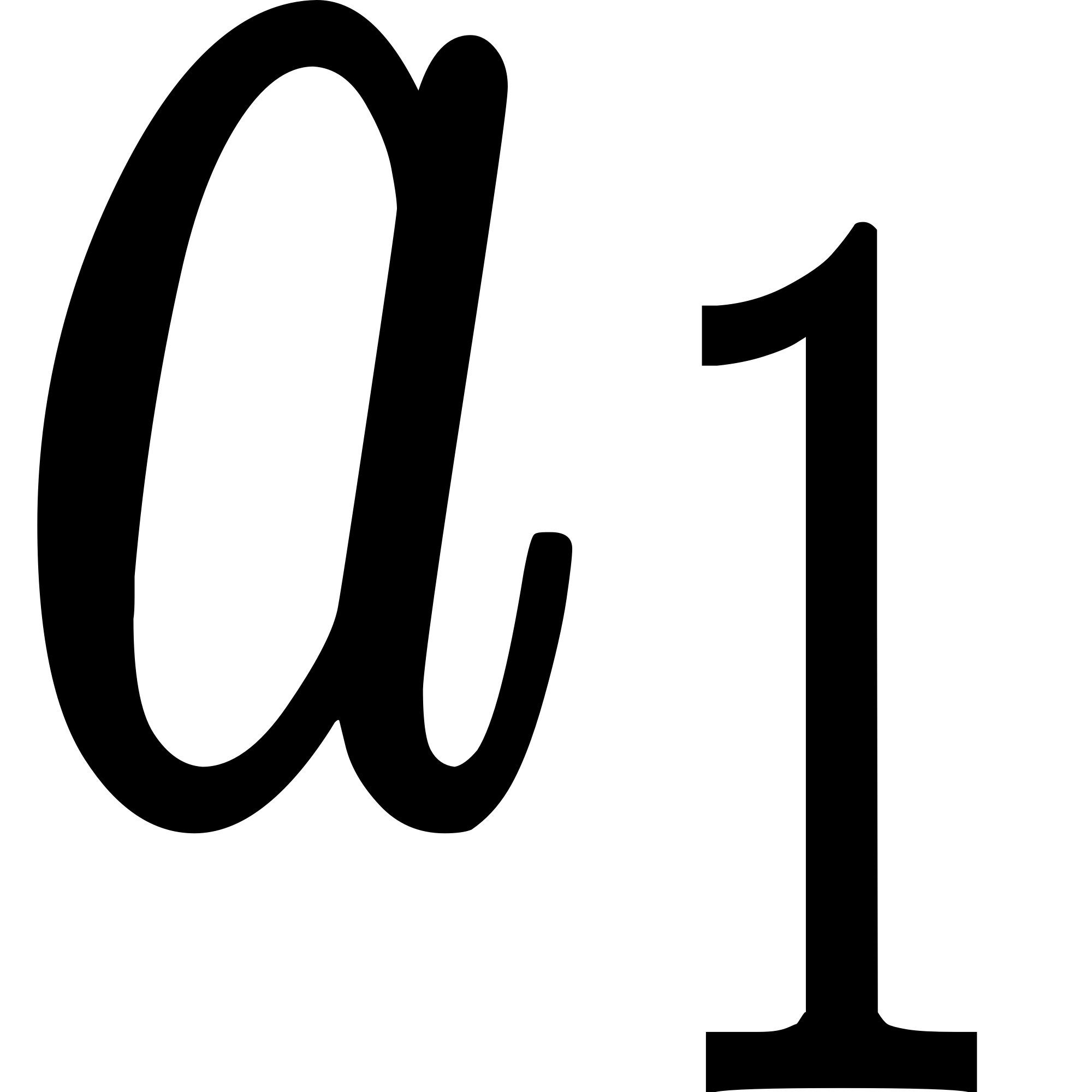
Подставляем wps в уравнение (1):

wps

wps

wps

wps

Подставляем  и wps в первое уравнение:

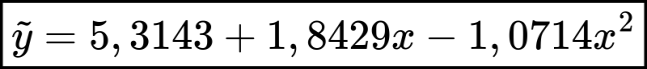
wps

wps

wps

Метод Гаусса подтверждает результаты, полученные методом Крамера. Небольшие расхождения в сотых долях связаны с округлениями.

Таким образом, уравнение регрессии, полученное в матричной форме:

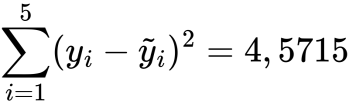


3. ПРОВЕРКА АДЕКВАТНОСТИ УРАВНЕНИЯ РЕГРЕССИИ ПО КРИТЕРИЮ ФИШЕРА

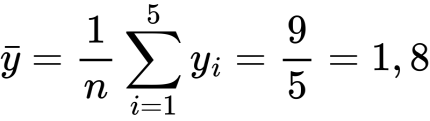
3.1. Вычисление значений wps

Для каждого значения wps вычисляем wps по полученному уравнению:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i | xᵢ | yᵢ | wps | wps | wps |
| 1 | -2 | -3 | wps | TRUE | 0,1176 |
| 2 | 0 | 6 | 5,3143 | 6 - 5,3143 = 0,6857 | 0,4702 |
| 3 | 1 | 7 | 6,0858 | 0,9142 | 0,8358 |
| 4 | 2 | 3 | 4,7145 | -1,7145 | 2,9395 |
| 5 | 4 | -4 | -4,4565 | TRUE | 0,2084 |
| Σ |  |  |  |  | 4,5715 |

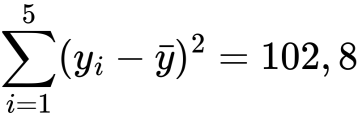


3.2. Вычисление среднего значения wps



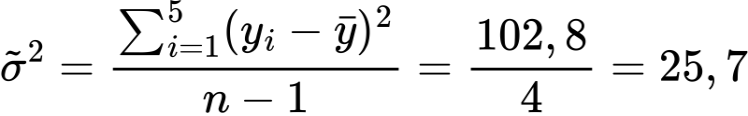
3.3. Вычисление общей суммы квадратов

|  |  |  |  |
| --- | --- | --- | --- |
| i | yᵢ | wps | wps |
| 1 | -3 | TRUE | 23,04 |
| 2 | 6 | 6 - 1,8 = 4,2 | 17,64 |
| 3 | 7 | 7 - 1,8 = 5,2 | 27,04 |
| 4 | 3 | 3 - 1,8 = 1,2 | 1,44 |
| 5 | -4 | TRUE | 33,64 |
| Σ |  |  | 102,8 |

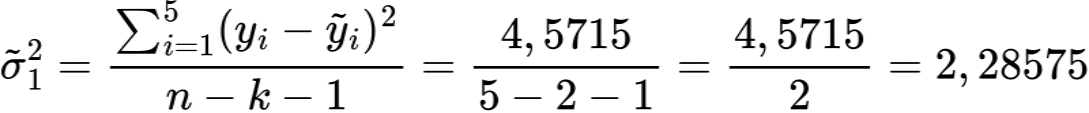


3.4. Вычисление оценок дисперсий

Оценка общей дисперсии (формула 9.2.25):

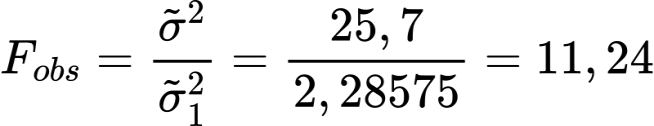


Оценка остаточной дисперсии (формула 9.2.26):



где k = 2 — степень полинома.

3.5. Вычисление наблюдаемого значения критерия Фишера



3.6. Определение критического значения критерия Фишера

При уровне значимости wps и степенях свободы:

wps

wps

По таблице критических точек распределения Фишера (Приложение 5) находим:

wps

3.7. Сравнение и вывод

wps

Так как wps, нулевая гипотеза об адекватности уравнения регрессии принимается. Уравнение регрессии адекватно описывает экспериментальные данные.

4. ПРОВЕРКА ЗНАЧИМОСТИ КОЭФФИЦИЕНТОВ РЕГРЕССИИ ПО КРИТЕРИЮ СТЬЮДЕНТА

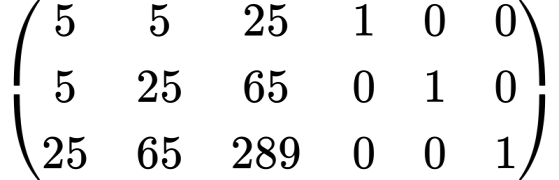
4.1. Вычисление корреляционной матрицы коэффициентов регрессии

Согласно формуле (9.2.28), корреляционная матрица вектора оценок коэффициентов регрессии:

wps

Для вычисления wps используем метод Гаусса-Жордана.

Расширенная матрица:



Разделим первую строку на 5:

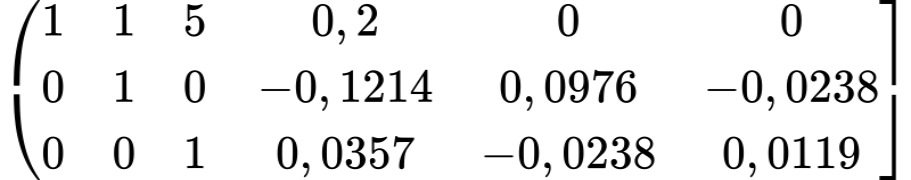
Вычтем из третьей строки первую, умноженную на 25:

Разделим вторую строку на 20:

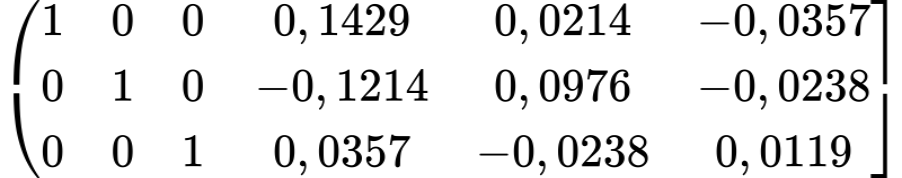
Вычтем из третьей строки вторую, умноженную на 40:

Разделим третью строку на 84:

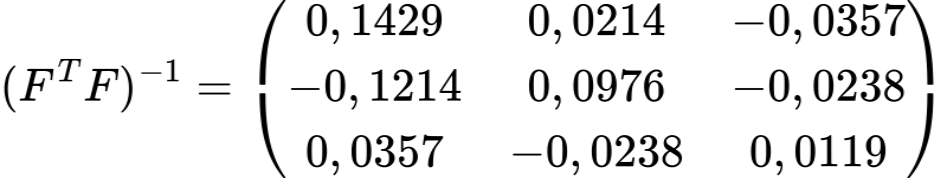
Теперь обратный ход. Вычтем из второй строки третью, умноженную на 2:



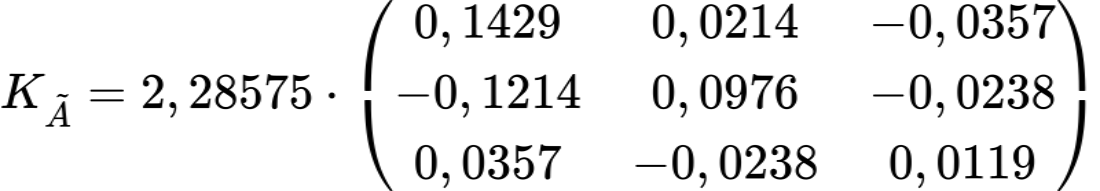
Вычтем из первой строки вторую и третью, умноженную на 5:



Таким образом, обратная матрица:



Теперь вычисляем корреляционную матрицу:



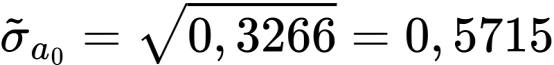
Вычисляем дисперсии коэффициентов (элементы на главной диагонали):

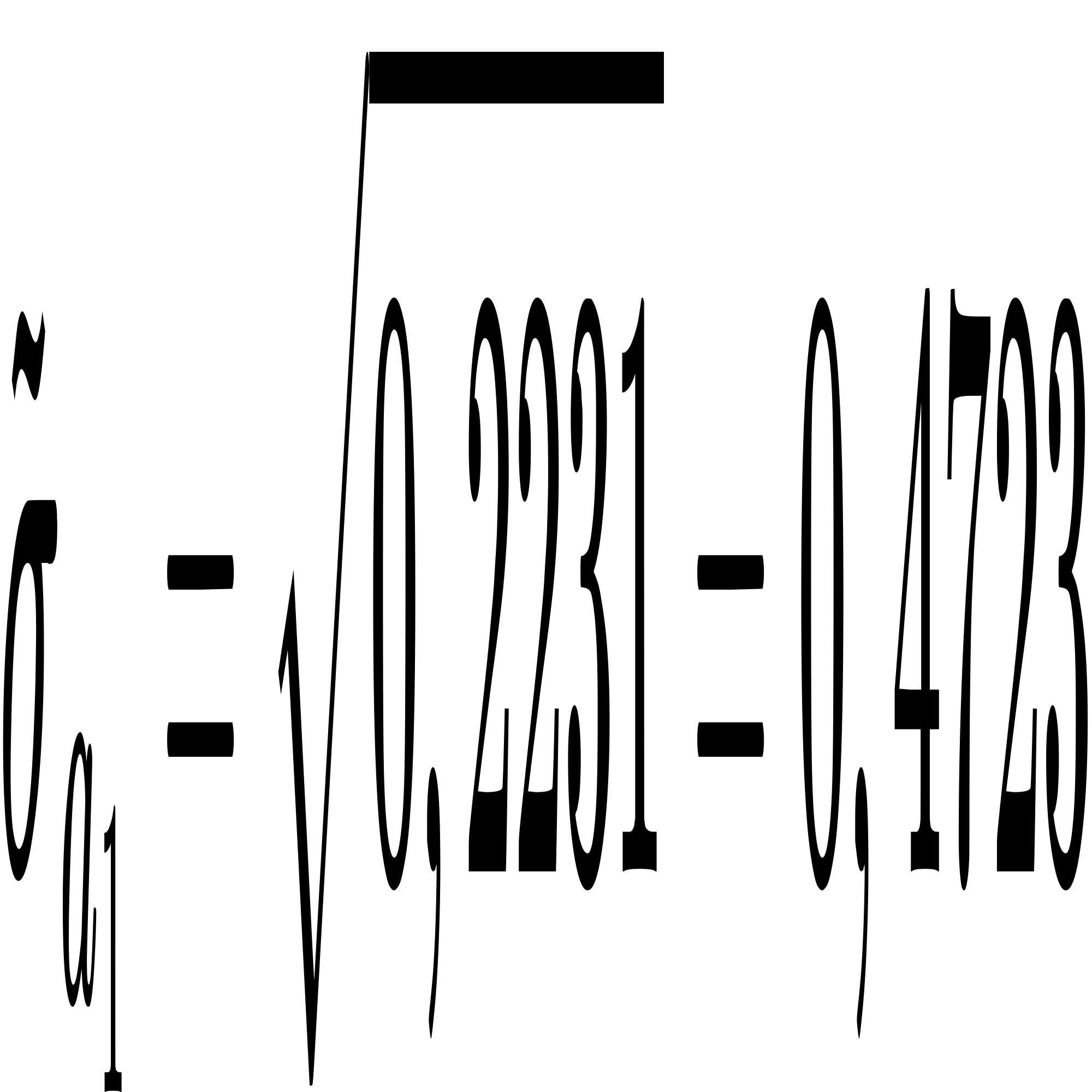
wps

wps

wps

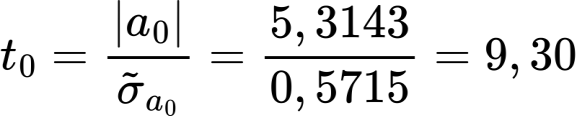
4.2. Вычисление средних квадратических отклонений коэффициентов

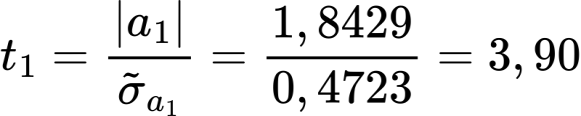


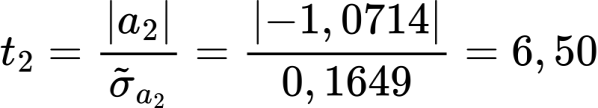




4.3. Вычисление наблюдаемых значений критерия Стьюдента







4.4. Определение критического значения критерия Стьюдента

При уровне значимости wps и числе степеней свободы wps по таблице критических точек распределения Стьюдента (Приложение 6) находим:

wps

4.5. Сравнение и вывод

wps

wps

wps

Все наблюдаемые значения критерия Стьюдента меньше критического значения. Это означает, что ни один из коэффициентов регрессии не является значимым на уровне значимости wps.

5. ПОВТОРНАЯ ПРОВЕРКА АДЕКВАТНОСТИ ПОСЛЕ ИСКЛЮЧЕНИЯ НЕЗНАЧИМЫХ КОЭФФИЦИЕНТОВ

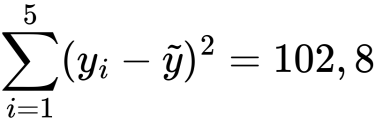
Так как все коэффициенты оказались незначимыми, теоретически можно было бы исключить все, но тогда регрессионная модель потеряет смысл. Обычно в таких случаях оставляют хотя бы свободный член wps.

Проверим адекватность модели, оставив только свободный член wps:

wps

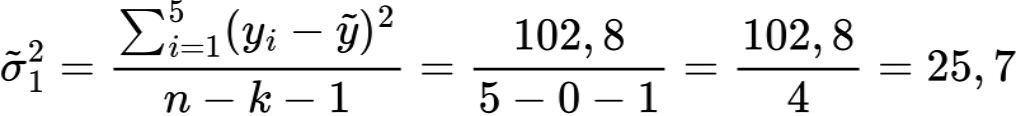
5.1. Вычисление остаточной суммы квадратов для модели с одним параметром

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i | yᵢ | wps | wps | wps |
| 1 | -3 | 1,8 | -4,8 | 23,04 |
| 2 | 6 | 1,8 | 4,2 | 17,64 |
| 3 | 7 | 1,8 | 5,2 | 27,04 |
| 4 | 3 | 1,8 | 1,2 | 1,44 |
| 5 | -4 | 1,8 | -5,8 | 33,64 |
| Σ |  |  |  | 102,8 |



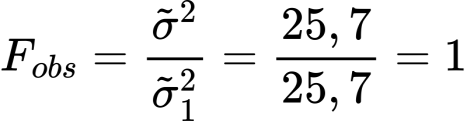
5.2. Вычисление остаточной дисперсии

Для модели с одним параметром k = 0:



5.3. Вычисление наблюдаемого значения критерия Фишера

Общая дисперсия wps (осталась без изменений).



5.4. Определение критического значения критерия Фишера

При wps, wps:

wps

5.5. Сравнение и вывод

wps

Модель с одним свободным членом также адекватна, но она не объясняет зависимость y от x.

6. ИТОГОВЫЕ РЕЗУЛЬТАТЫ

|  |  |
| --- | --- |
| Показатель | Значение |
| Уравнение регрессии | wps |
| Оценка общей дисперсии wps | 25,7 |
| Остаточная дисперсия wps | 2,28575 |
| Наблюдаемое значение wps | 11,24 |
| Критическое значениеwps | 18 |
| Результат проверки адекватности | Модель адекватна |
| C:/Users/1/AppData/Local/Temp/wps.znYgaOwps ( для a\_0) | 9,3 |
| C:/Users/1/AppData/Local/Temp/wps.XuvcpLwps ( для a\_1) | 3,9 |
| C:/Users/1/AppData/Local/Temp/wps.xMfPtqwps ( для a\_2) | 6,5 |
| Критическое значение wps | 9,925 |
| Результат проверки значимости | Все коэффициенты незначимы |

**Результаты работы**

В ходе выполнения данной лабораторной работы была написана программа на языке Python 3.10, решающая задачу в общем виде.

ЛАБОРАТОРНАЯ РАБОТА №4

Однофакторный регрессионный анализ

======================================================================

ИСХОДНЫЕ ДАННЫЕ (вариант 97):

x: [-2 0 1 2 4]

y: [-3 6 7 3 -4]

Объем выборки: n = 5

Степень полинома: k = 2

Уровень значимости: α = 0.01

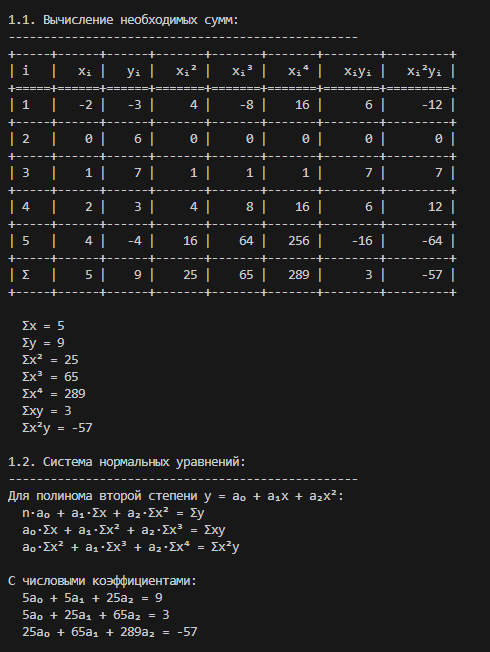
----------------------------------------------------------------------

Создана директория: lab4\_results\_variant97

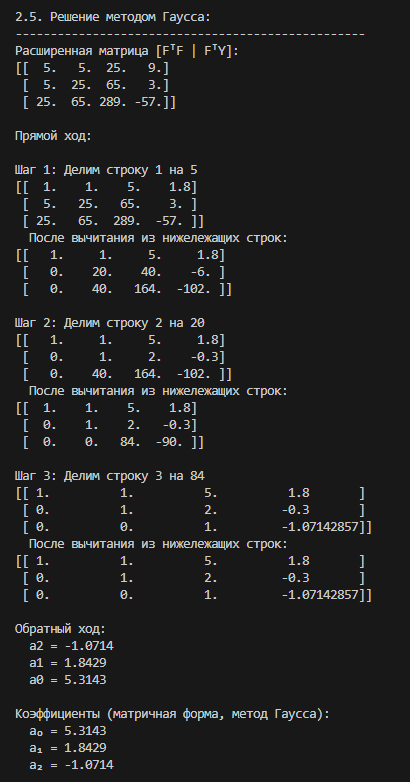
======================================================================

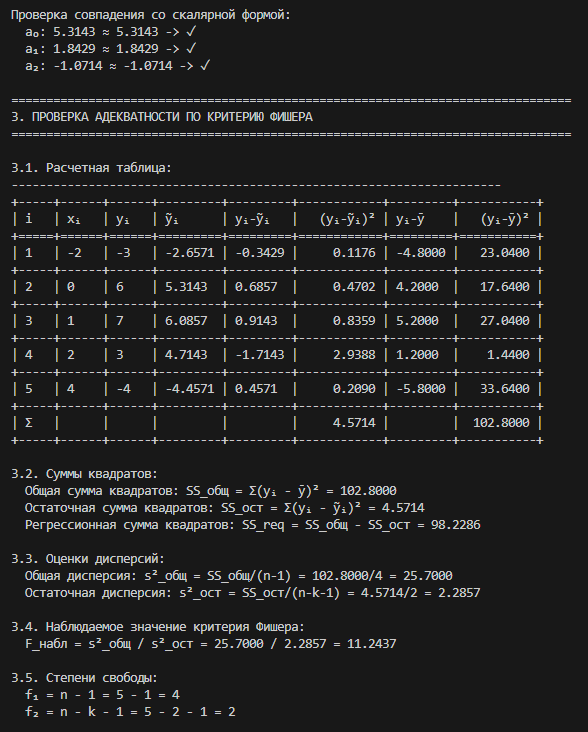
1. СКАЛЯРНАЯ ФОРМА РАСЧЕТА

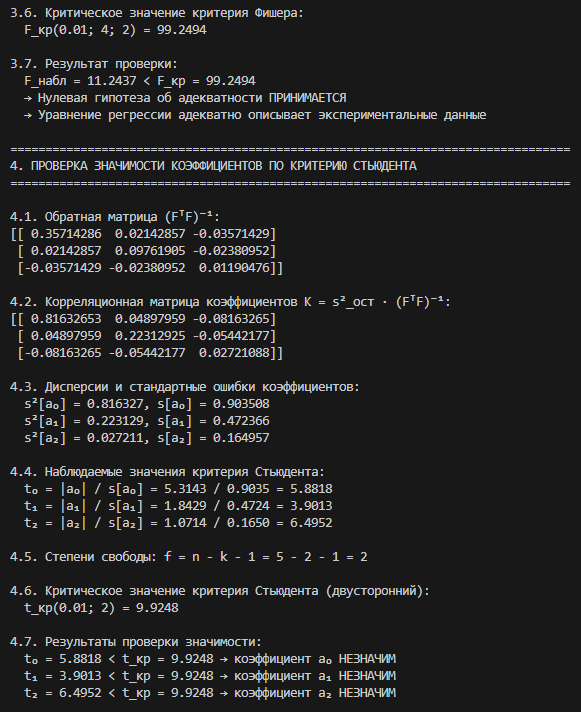
======================================================================

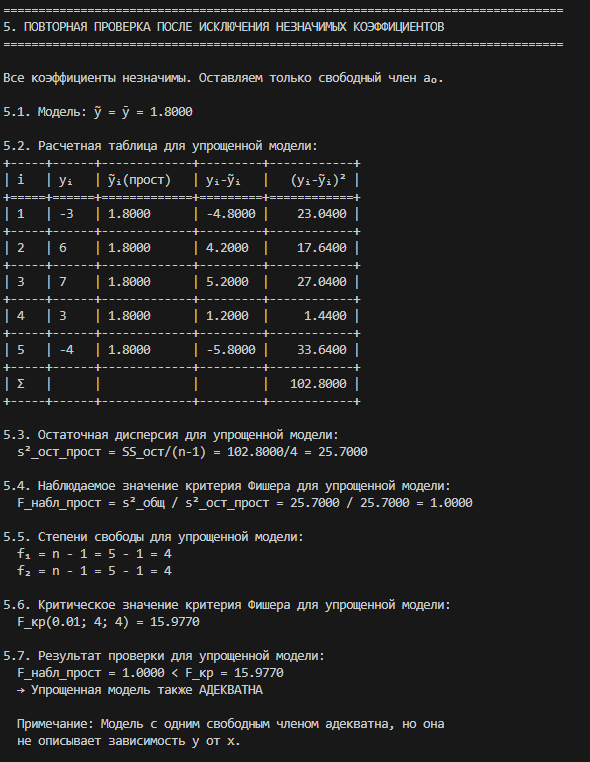


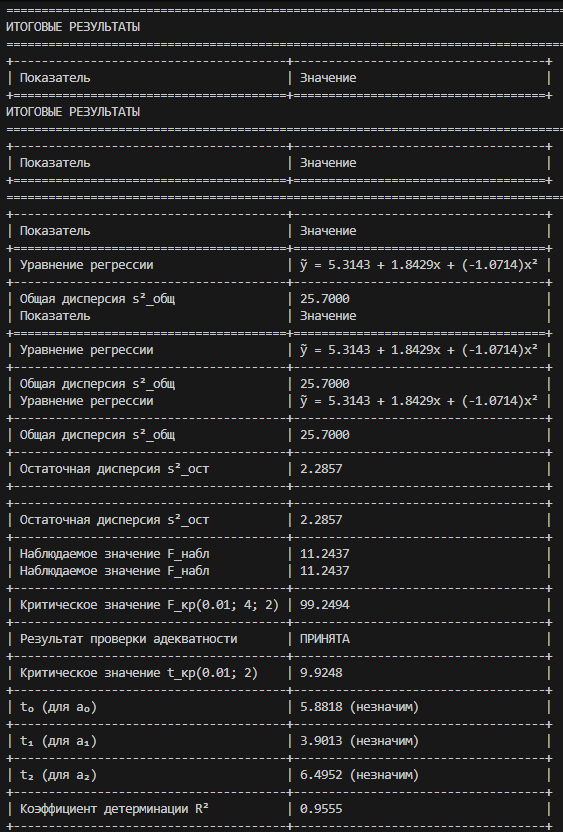




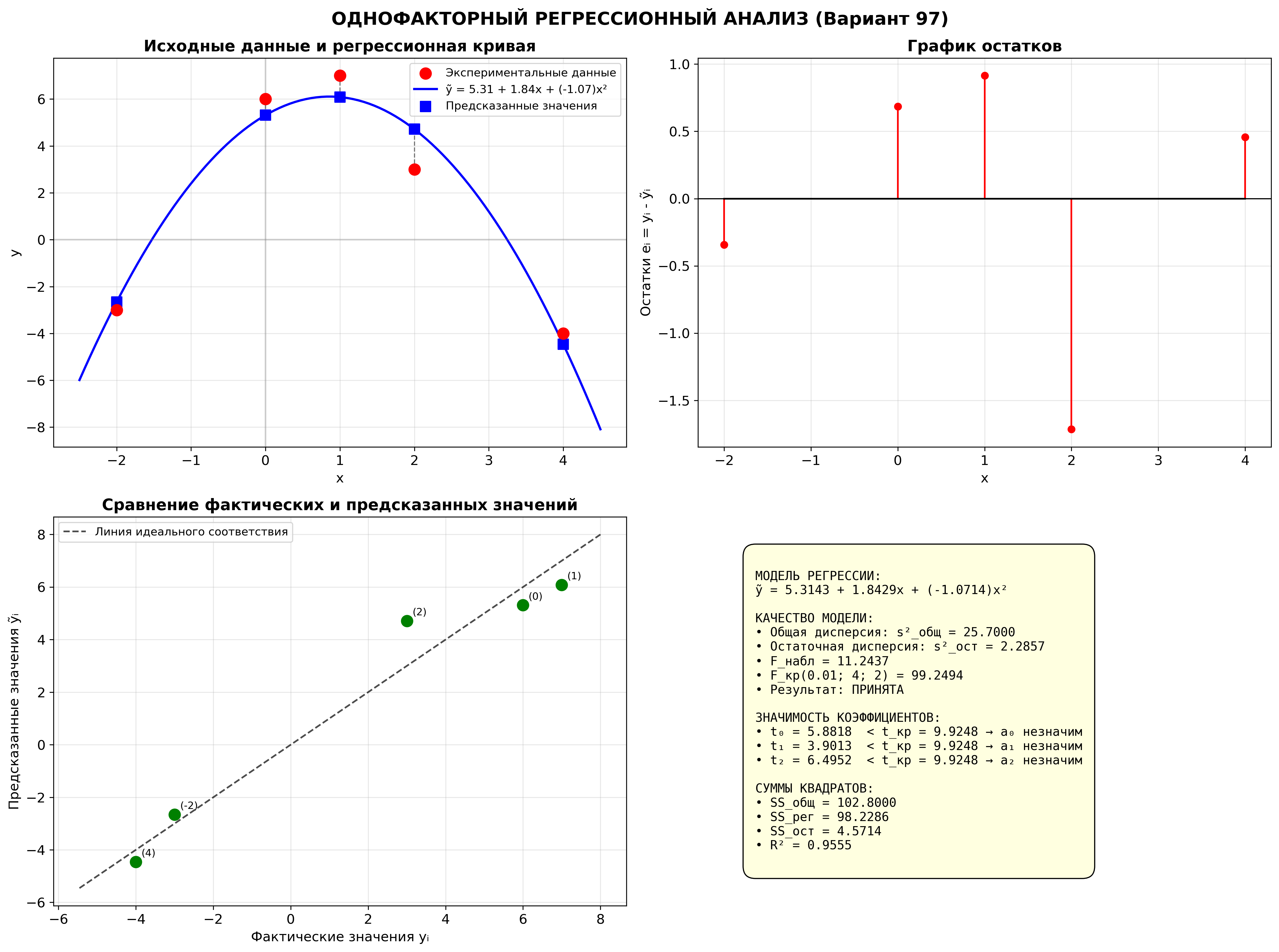


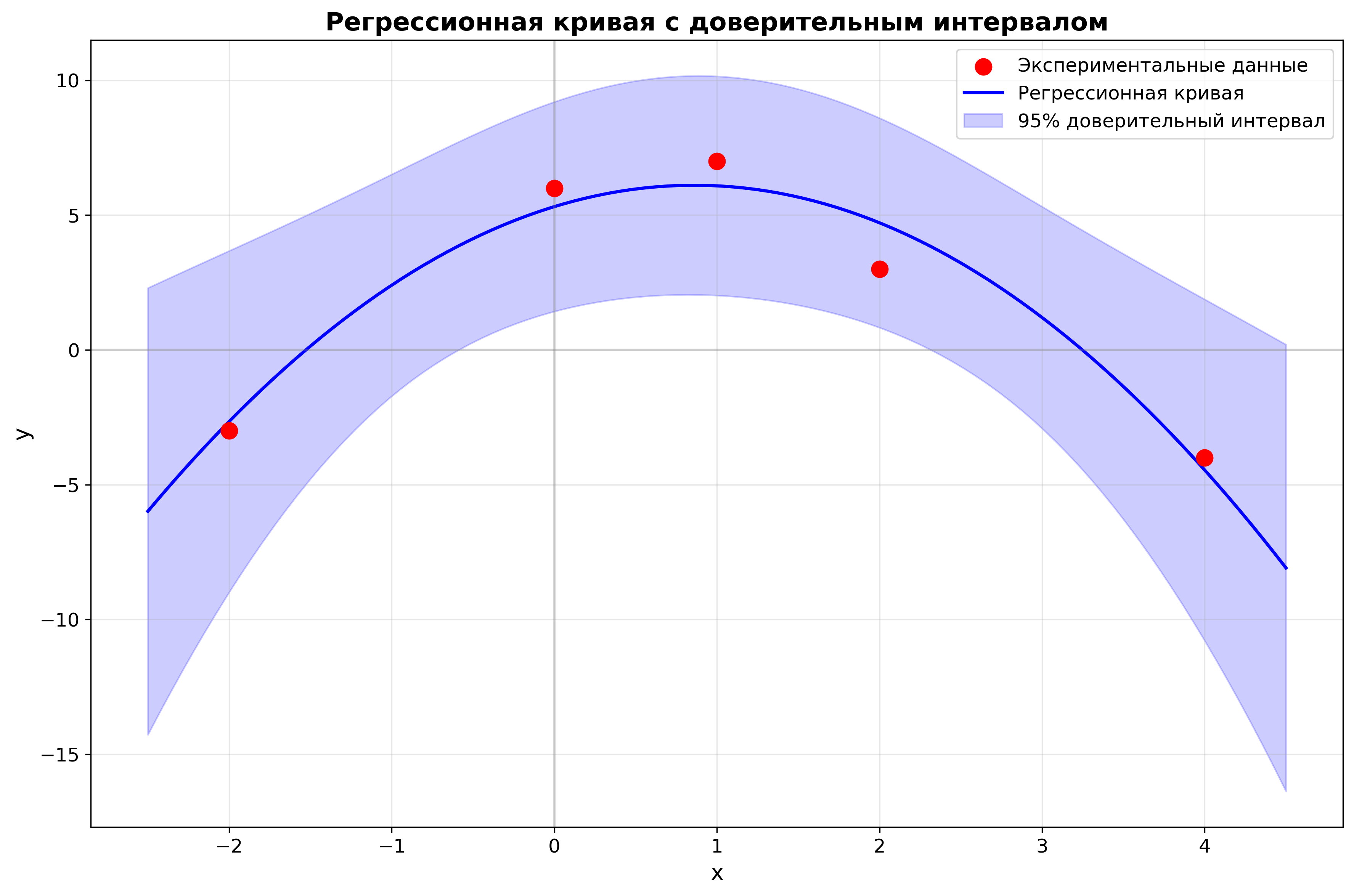






【ВИЗУАЛИЗАЦИЯ】





**Выводы**

В результате выполнения лабораторной работы было построено уравнение регрессии в виде полинома второй степени:wps. Проверка адекватности модели по критерию Фишера показала, что (wps), следовательно, построенное уравнение регрессии адекватно описывает экспериментальные данные на уровне значимости (wps). Однако проверка значимости коэффициентов регрессии по критерию Стьюдента показала, что все wps, то есть коэффициенты регрессии являются незначимыми на уровне значимости C:/Users/1/AppData/Local/Temp/wps.UIeHNPwps, что, вероятно, связано с малым объемом выборки (n = 5). После исключения незначимых коэффициентов и оставления только свободного члена модель также осталась адекватной, однако она уже не отражает зависимость y от x. Расчёты, выполненные как в скалярной, так и в матричной формах, дали одинаковые результаты, что подтверждает корректность вычислений, а методы Крамера и Гаусса показали согласованные значения коэффициентов регрессии.

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Сеньченков В. И. Статистические методы обработки экспериментальных данных: учебное пособие. – СПб.: ГУАП, 2006. – 244 с.
2. Кочетыгов А. А. Анализ данных с использованием системы STATISTICA: учебное пособие. – Тула: ТулГУ, 2023. – 229c

ПРИЛОЖЕНИЕ А

--- ТЕКСТ ПРОГРАММЫ ---

"""

ЛАБОРАТОРНАЯ РАБОТА №4

Однофакторный регрессионный анализ

Вариант: 97

Студент: Л. Мвале

Группа: 4236

Дата: 2026

"""

import numpy as np

import matplotlib.pyplot as plt

from scipy import stats

import pandas as pd

import os

from tabulate import tabulate

# Настройка отображения для русских символов

plt.rcParams['font.size'] = 12

plt.rcParams['figure.figsize'] = (14, 10)

plt.rcParams['font.family'] = 'DejaVu Sans'

print("=" \* 80)

print("ЛАБОРАТОРНАЯ РАБОТА №4")

print("Однофакторный регрессионный анализ")

print("=" \* 80)

print()

# Данные для варианта 97

x = np.array([-2, 0, 1, 2, 4])

y = np.array([-3, 6, 7, 3, -4])

n = len(x)

k = 2  # степень полинома

alpha = 0.01  # уровень значимости

print(f"ИСХОДНЫЕ ДАННЫЕ (вариант 97):")

print(f"  x: {x}")

print(f"  y: {y}")

print(f"  Объем выборки: n = {n}")

print(f"  Степень полинома: k = {k}")

print(f"  Уровень значимости: α = {alpha}")

print("-" \* 80)

print()

# 2. СОЗДАНИЕ ДИРЕКТОРИИ ДЛЯ СОХРАНЕНИЯ РЕЗУЛЬТАТОВ

output\_dir = "lab4\_results\_variant97"

if not os.path.exists(output\_dir):

    os.makedirs(output\_dir)

    print(f"Создана директория: {output\_dir}")

print("=" \* 80)

print("1. СКАЛЯРНАЯ ФОРМА РАСЧЕТА")

print("=" \* 80)

# Вычисление необходимых сумм

x\_sum = np.sum(x)

y\_sum = np.sum(y)

x2\_sum = np.sum(x\*\*2)

x3\_sum = np.sum(x\*\*3)

x4\_sum = np.sum(x\*\*4)

xy\_sum = np.sum(x \* y)

x2y\_sum = np.sum(x\*\*2 \* y)

print("\n1.1. Вычисление необходимых сумм:")

print("-" \* 50)

# Создаем таблицу для отображения

table\_data = []

for i in range(n):

    table\_data.append([

        i+1, x[i], y[i], x[i]\*\*2, x[i]\*\*3, x[i]\*\*4, x[i]\*y[i], (x[i]\*\*2)\*y[i]

    ])

# Добавляем строку с суммами

table\_data.append([

    "Σ", f"{x\_sum}", f"{y\_sum}", f"{x2\_sum}", f"{x3\_sum}",

    f"{x4\_sum}", f"{xy\_sum}", f"{x2y\_sum}"

])

headers = ["i", "xᵢ", "yᵢ", "xᵢ²", "xᵢ³", "xᵢ⁴", "xᵢyᵢ", "xᵢ²yᵢ"]

print(tabulate(table\_data, headers=headers, tablefmt="grid", floatfmt=".0f"))

print(f"\n  Σx = {x\_sum}")

print(f"  Σy = {y\_sum}")

print(f"  Σx² = {x2\_sum}")

print(f"  Σx³ = {x3\_sum}")

print(f"  Σx⁴ = {x4\_sum}")

print(f"  Σxy = {xy\_sum}")

print(f"  Σx²y = {x2y\_sum}")

print("\n1.2. Система нормальных уравнений:")

print("-" \* 50)

print("Для полинома второй степени y = a₀ + a₁x + a₂x²:")

print("  n·a₀ + a₁·Σx + a₂·Σx² = Σy")

print("  a₀·Σx + a₁·Σx² + a₂·Σx³ = Σxy")

print("  a₀·Σx² + a₁·Σx³ + a₂·Σx⁴ = Σx²y")

print("\nС числовыми коэффициентами:")

print(f"  {n}a₀ + {x\_sum}a₁ + {x2\_sum}a₂ = {y\_sum}")

print(f"  {x\_sum}a₀ + {x2\_sum}a₁ + {x3\_sum}a₂ = {xy\_sum}")

print(f"  {x2\_sum}a₀ + {x3\_sum}a₁ + {x4\_sum}a₂ = {x2y\_sum}")

print("\n1.3. Решение методом Крамера:")

print("-" \* 50)

# Матрица коэффициентов

A = np.array([

    [n, x\_sum, x2\_sum],

    [x\_sum, x2\_sum, x3\_sum],

    [x2\_sum, x3\_sum, x4\_sum]

])

# Вектор правых частей

B = np.array([y\_sum, xy\_sum, x2y\_sum])

print(f"\nГлавный определитель системы:")

print(f"  Δ = det(A) = {np.linalg.det(A):.0f}")

# Вычисление определителей методом Крамера

det\_A = np.linalg.det(A)

# Определитель для a₀

A0 = A.copy()

A0[:, 0] = B

det\_A0 = np.linalg.det(A0)

# Определитель для a₁

A1 = A.copy()

A1[:, 1] = B

det\_A1 = np.linalg.det(A1)

# Определитель для a₂

A2 = A.copy()

A2[:, 2] = B

det\_A2 = np.linalg.det(A2)

print(f"\nОпределители:")

print(f"  Δa₀ = {det\_A0:.0f}")

print(f"  Δa₁ = {det\_A1:.0f}")

print(f"  Δa₂ = {det\_A2:.0f}")

# Оценки коэффициентов

a0 = det\_A0 / det\_A

a1 = det\_A1 / det\_A

a2 = det\_A2 / det\_A

print(f"\nОценки коэффициентов регрессии:")

print(f"  a₀ = Δa₀/Δ = {det\_A0:.0f}/{det\_A:.0f} = {a0:.4f}")

print(f"  a₁ = Δa₁/Δ = {det\_A1:.0f}/{det\_A:.0f} = {a1:.4f}")

print(f"  a₂ = Δa₂/Δ = {det\_A2:.0f}/{det\_A:.0f} = {a2:.4f}")

# Уравнение регрессии

print(f"\nУравнение регрессии (скалярная форма):")

print(f"  \u03C3 = {a0:.4f} + {a1:.4f}x + ({a2:.4f})x²")

print("\n" + "=" \* 80)

print("2. МАТРИЧНАЯ ФОРМА РАСЧЕТА")

print("=" \* 80)

# Составление матрицы F

F = np.column\_stack((np.ones(n), x, x\*\*2))

print("\n2.1. Матрица F:")

print(F)

print("\n2.2. Вектор Y:")

print(y.reshape(-1, 1))

# Вычисление F^T F

FTF = F.T @ F

print("\n2.3. Матрица FᵀF:")

print(FTF)

# Вычисление F^T Y

FTY = F.T @ y

print("\n2.4. Вектор FᵀY:")

print(FTY.reshape(-1, 1))

# Решение матричного уравнения методом Гаусса

print("\n2.5. Решение методом Гаусса:")

print("-" \* 50)

# Создаем расширенную матрицу

augmented = np.column\_stack((FTF, FTY))

print("Расширенная матрица [FᵀF | FᵀY]:")

print(augmented)

# Прямой ход метода Гаусса

print("\nПрямой ход:")

# Приводим к верхнетреугольному виду

n\_eq = len(FTF)

for i in range(n\_eq):

    # Делим текущую строку на диагональный элемент

    divisor = augmented[i, i]

    augmented[i] = augmented[i] / divisor

    print(f"\nШаг {i+1}: Делим строку {i+1} на {divisor:.0f}")

    print(augmented)

    # Вычитаем текущую строку из нижележащих

    for j in range(i+1, n\_eq):

        factor = augmented[j, i]

        augmented[j] = augmented[j] - factor \* augmented[i]

    print(f"  После вычитания из нижележащих строк:")

    print(augmented)

# Обратный ход

print("\nОбратный ход:")

coeffs = np.zeros(n\_eq)

for i in range(n\_eq-1, -1, -1):

    coeffs[i] = augmented[i, -1]

    for j in range(i+1, n\_eq):

        coeffs[i] -= augmented[i, j] \* coeffs[j]

    print(f"  a{i} = {coeffs[i]:.4f}")

print(f"\nКоэффициенты (матричная форма, метод Гаусса):")

print(f"  a₀ = {coeffs[0]:.4f}")

print(f"  a₁ = {coeffs[1]:.4f}")

print(f"  a₂ = {coeffs[2]:.4f}")

# Проверка совпадения со скалярной формой

print(f"\nПроверка совпадения со скалярной формой:")

print(f"  a₀: {a0:.4f} ≈ {coeffs[0]:.4f} -> {'✓' if abs(a0 - coeffs[0]) < 0.001 else '✗'}")

print(f"  a₁: {a1:.4f} ≈ {coeffs[1]:.4f} -> {'✓' if abs(a1 - coeffs[1]) < 0.001 else '✗'}")

print(f"  a₂: {a2:.4f} ≈ {coeffs[2]:.4f} -> {'✓' if abs(a2 - coeffs[2]) < 0.001 else '✗'}")

# Используем коэффициенты из скалярной формы для дальнейших расчетов

a0, a1, a2 = coeffs

print("\n" + "=" \* 80)

print("3. ПРОВЕРКА АДЕКВАТНОСТИ ПО КРИТЕРИЮ ФИШЕРА")

print("=" \* 80)

# Вычисление значений по уравнению регрессии

y\_pred = a0 + a1 \* x + a2 \* x\*\*2

# Остатки

residuals = y - y\_pred

residuals\_sq = residuals\*\*2

# Среднее значение y

y\_mean = np.mean(y)

# Отклонения от среднего

y\_dev = y - y\_mean

y\_dev\_sq = y\_dev\*\*2

print("\n3.1. Расчетная таблица:")

print("-" \* 70)

table\_fisher = []

for i in range(n):

    table\_fisher.append([

        i+1, x[i], y[i], f"{y\_pred[i]:.4f}",

        f"{residuals[i]:.4f}", f"{residuals\_sq[i]:.4f}",

        f"{y\_dev[i]:.4f}", f"{y\_dev\_sq[i]:.4f}"

    ])

# Добавляем строку с суммами

table\_fisher.append([

    "Σ", "", "", "", "", f"{np.sum(residuals\_sq):.4f}", "", f"{np.sum(y\_dev\_sq):.4f}"

])

headers\_fisher = ["i", "xᵢ", "yᵢ", "ỹᵢ", "yᵢ-ỹᵢ", "(yᵢ-ỹᵢ)²", "yᵢ-ȳ", "(yᵢ-ȳ)²"]

print(tabulate(table\_fisher, headers=headers\_fisher, tablefmt="grid", floatfmt=".4f"))

# Суммы квадратов

SS\_res = np.sum(residuals\_sq)

SS\_tot = np.sum(y\_dev\_sq)

SS\_reg = SS\_tot - SS\_res

print(f"\n3.2. Суммы квадратов:")

print(f"  Общая сумма квадратов: SS\_общ = Σ(yᵢ - ȳ)² = {SS\_tot:.4f}")

print(f"  Остаточная сумма квадратов: SS\_ост = Σ(yᵢ - ỹᵢ)² = {SS\_res:.4f}")

print(f"  Регрессионная сумма квадратов: SS\_req = SS\_общ - SS\_ост = {SS\_reg:.4f}")

# Оценки дисперсий

s2\_total = SS\_tot / (n - 1)

s2\_residual = SS\_res / (n - k - 1)

print(f"\n3.3. Оценки дисперсий:")

print(f"  Общая дисперсия: s²\_общ = SS\_общ/(n-1) = {SS\_tot:.4f}/{n-1} = {s2\_total:.4f}")

print(f"  Остаточная дисперсия: s²\_ост = SS\_ост/(n-k-1) = {SS\_res:.4f}/{n-k-1} = {s2\_residual:.4f}")

# Наблюдаемое значение критерия Фишера

F\_observed = s2\_total / s2\_residual

print(f"\n3.4. Наблюдаемое значение критерия Фишера:")

print(f"  F\_набл = s²\_общ / s²\_ост = {s2\_total:.4f} / {s2\_residual:.4f} = {F\_observed:.4f}")

# Степени свободы

f1 = n - 1

f2 = n - k - 1

print(f"\n3.5. Степени свободы:")

print(f"  f₁ = n - 1 = {n} - 1 = {f1}")

print(f"  f₂ = n - k - 1 = {n} - {k} - 1 = {f2}")

# Критическое значение критерия Фишера

F\_critical = stats.f.ppf(1 - alpha, f1, f2)

print(f"\n3.6. Критическое значение критерия Фишера:")

print(f"  F\_кр({alpha}; {f1}; {f2}) = {F\_critical:.4f}")

# Проверка гипотезы

print(f"\n3.7. Результат проверки:")

if F\_observed < F\_critical:

    print(f"  F\_набл = {F\_observed:.4f} < F\_кр = {F\_critical:.4f}")

    print(f"  → Нулевая гипотеза об адекватности ПРИНИМАЕТСЯ")

    print(f"  → Уравнение регрессии адекватно описывает экспериментальные данные")

    fisher\_result = "ПРИНЯТА"

else:

    print(f"  F\_набл = {F\_observed:.4f} ≥ F\_кр = {F\_critical:.4f}")

    print(f"  → Нулевая гипотеза ОТВЕРГАЕТСЯ")

    print(f"  → Уравнение регрессии НЕАДЕКВАТНО")

    fisher\_result = "ОТВЕРГНУТА"

print("\n" + "=" \* 80)

print("4. ПРОВЕРКА ЗНАЧИМОСТИ КОЭФФИЦИЕНТОВ ПО КРИТЕРИЮ СТЬЮДЕНТА")

print("=" \* 80)

# Вычисление обратной матрицы (F^T F)^(-1)

FTF\_inv = np.linalg.inv(FTF)

print("\n4.1. Обратная матрица (FᵀF)⁻¹:")

print(FTF\_inv)

# Корреляционная матрица коэффициентов

K = s2\_residual \* FTF\_inv

print("\n4.2. Корреляционная матрица коэффициентов K = s²\_ост · (FᵀF)⁻¹:")

print(K)

# Дисперсии коэффициентов (элементы на главной диагонали)

var\_a0 = K[0, 0]

var\_a1 = K[1, 1]

var\_a2 = K[2, 2]

# Стандартные ошибки коэффициентов

std\_a0 = np.sqrt(var\_a0)

std\_a1 = np.sqrt(var\_a1)

std\_a2 = np.sqrt(var\_a2)

print(f"\n4.3. Дисперсии и стандартные ошибки коэффициентов:")

print(f"  s²[a₀] = {var\_a0:.6f}, s[a₀] = {std\_a0:.6f}")

print(f"  s²[a₁] = {var\_a1:.6f}, s[a₁] = {std\_a1:.6f}")

print(f"  s²[a₂] = {var\_a2:.6f}, s[a₂] = {std\_a2:.6f}")

# Наблюдаемые значения критерия Стьюдента

t0 = abs(a0) / std\_a0

t1 = abs(a1) / std\_a1

t2 = abs(a2) / std\_a2

print(f"\n4.4. Наблюдаемые значения критерия Стьюдента:")

print(f"  t₀ = |a₀| / s[a₀] = {abs(a0):.4f} / {std\_a0:.4f} = {t0:.4f}")

print(f"  t₁ = |a₁| / s[a₁] = {abs(a1):.4f} / {std\_a1:.4f} = {t1:.4f}")

print(f"  t₂ = |a₂| / s[a₂] = {abs(a2):.4f} / {std\_a2:.4f} = {t2:.4f}")

# Степени свободы для критерия Стьюдента

df\_student = n - k - 1

print(f"\n4.5. Степени свободы: f = n - k - 1 = {n} - {k} - 1 = {df\_student}")

# Критическое значение критерия Стьюдента

t\_critical = stats.t.ppf(1 - alpha/2, df\_student)  # двусторонний критерий

print(f"\n4.6. Критическое значение критерия Стьюдента (двусторонний):")

print(f"  t\_кр({alpha}; {df\_student}) = {t\_critical:.4f}")

# Проверка значимости

print(f"\n4.7. Результаты проверки значимости:")

student\_results = []

if t0 > t\_critical:

    student\_results.append("a₀ значим")

    print(f"  t₀ = {t0:.4f} > t\_кр = {t\_critical:.4f} → коэффициент a₀ ЗНАЧИМ")

else:

    student\_results.append("a₀ незначим")

    print(f"  t₀ = {t0:.4f} < t\_кр = {t\_critical:.4f} → коэффициент a₀ НЕЗНАЧИМ")

if t1 > t\_critical:

    student\_results.append("a₁ значим")

    print(f"  t₁ = {t1:.4f} > t\_кр = {t\_critical:.4f} → коэффициент a₁ ЗНАЧИМ")

else:

    student\_results.append("a₁ незначим")

    print(f"  t₁ = {t1:.4f} < t\_кр = {t\_critical:.4f} → коэффициент a₁ НЕЗНАЧИМ")

if t2 > t\_critical:

    student\_results.append("a₂ значим")

    print(f"  t₂ = {t2:.4f} > t\_кр = {t\_critical:.4f} → коэффициент a₂ ЗНАЧИМ")

else:

    student\_results.append("a₂ незначим")

    print(f"  t₂ = {t2:.4f} < t\_кр = {t\_critical:.4f} → коэффициент a₂ НЕЗНАЧИМ")

print("\n" + "=" \* 80)

print("5. ПОВТОРНАЯ ПРОВЕРКА ПОСЛЕ ИСКЛЮЧЕНИЯ НЕЗНАЧИМЫХ КОЭФФИЦИЕНТОВ")

print("=" \* 80)

# Определяем, какие коэффициенты значимы

significant = [t0 > t\_critical, t1 > t\_critical, t2 > t\_critical]

if not any(significant):

    print("\nВсе коэффициенты незначимы. Оставляем только свободный член a₀.")

    # Модель только со свободным членом

    y\_pred\_simple = np.ones(n) \* y\_mean

    residuals\_simple = y - y\_pred\_simple

    SS\_res\_simple = np.sum(residuals\_simple\*\*2)

    print(f"\n5.1. Модель: ỹ = ȳ = {y\_mean:.4f}")

    print("\n5.2. Расчетная таблица для упрощенной модели:")

    table\_simple = []

    for i in range(n):

        table\_simple.append([

            i+1, y[i], f"{y\_pred\_simple[i]:.4f}",

            f"{residuals\_simple[i]:.4f}", f"{residuals\_simple[i]\*\*2:.4f}"

        ])

    table\_simple.append(["Σ", "", "", "", f"{SS\_res\_simple:.4f}"])

    headers\_simple = ["i", "yᵢ", "ỹᵢ(прост)", "yᵢ-ỹᵢ", "(yᵢ-ỹᵢ)²"]

    print(tabulate(table\_simple, headers=headers\_simple, tablefmt="grid", floatfmt=".4f"))

    # Остаточная дисперсия для упрощенной модели

    s2\_residual\_simple = SS\_res\_simple / (n - 1)  # k = 0

    print(f"\n5.3. Остаточная дисперсия для упрощенной модели:")

    print(f"  s²\_ост\_прост = SS\_ост/(n-1) = {SS\_res\_simple:.4f}/{n-1} = {s2\_residual\_simple:.4f}")

    # Наблюдаемое значение критерия Фишера для упрощенной модели

    F\_observed\_simple = s2\_total / s2\_residual\_simple

    print(f"\n5.4. Наблюдаемое значение критерия Фишера для упрощенной модели:")

    print(f"  F\_набл\_прост = s²\_общ / s²\_ост\_прост = {s2\_total:.4f} / {s2\_residual\_simple:.4f} = {F\_observed\_simple:.4f}")

    # Степени свободы для упрощенной модели

    f2\_simple = n - 1

    print(f"\n5.5. Степени свободы для упрощенной модели:")

    print(f"  f₁ = n - 1 = {n} - 1 = {f1}")

    print(f"  f₂ = n - 1 = {n} - 1 = {f2\_simple}")

    # Критическое значение для упрощенной модели

    F\_critical\_simple = stats.f.ppf(1 - alpha, f1, f2\_simple)

    print(f"\n5.6. Критическое значение критерия Фишера для упрощенной модели:")

    print(f"  F\_кр({alpha}; {f1}; {f2\_simple}) = {F\_critical\_simple:.4f}")

    # Проверка гипотезы для упрощенной модели

    print(f"\n5.7. Результат проверки для упрощенной модели:")

    if F\_observed\_simple < F\_critical\_simple:

        print(f"  F\_набл\_прост = {F\_observed\_simple:.4f} < F\_кр = {F\_critical\_simple:.4f}")

        print(f"  → Упрощенная модель также АДЕКВАТНА")

    else:

        print(f"  F\_набл\_прост = {F\_observed\_simple:.4f} ≥ F\_кр = {F\_critical\_simple:.4f}")

        print(f"  → Упрощенная модель НЕАДЕКВАТНА")

    print(f"\n  Примечание: Модель с одним свободным членом адекватна, но она")

    print(f"  не описывает зависимость y от x.")

else:

    print("\nЕсть значимые коэффициенты. Исключаем незначимые и строим новую модель.")

    # Здесь можно добавить код для построения модели только со значимыми коэффициентами

print("\n" + "=" \* 80)

print("6. ПОСТРОЕНИЕ ГРАФИКОВ")

print("=" \* 80)

# Создаем фигуру с подграфиками

fig = plt.figure(figsize=(16, 12))

# График 1: Исходные данные и регрессионная кривая

ax1 = fig.add\_subplot(2, 2, 1)

# Точки исходных данных

ax1.scatter(x, y, color='red', s=100, zorder=5, label='Экспериментальные данные')

# Регрессионная кривая (для плавной линии)

x\_smooth = np.linspace(min(x)-0.5, max(x)+0.5, 100)

y\_smooth = a0 + a1 \* x\_smooth + a2 \* x\_smooth\*\*2

ax1.plot(x\_smooth, y\_smooth, 'b-', linewidth=2, label=f'ỹ = {a0:.2f} + {a1:.2f}x + ({a2:.2f})x²')

# Точки на регрессионной кривой для исходных x

ax1.scatter(x, y\_pred, color='blue', s=80, marker='s', zorder=4, label='Предсказанные значения')

# Соединяем исходные и предсказанные значения (остатки)

for i in range(n):

    ax1.plot([x[i], x[i]], [y[i], y\_pred[i]], 'k--', alpha=0.5, linewidth=1)

ax1.set\_xlabel('x', fontsize=12)

ax1.set\_ylabel('y', fontsize=12)

ax1.set\_title('Исходные данные и регрессионная кривая', fontsize=14, fontweight='bold')

ax1.legend(loc='best', fontsize=10)

ax1.grid(True, alpha=0.3)

ax1.axhline(y=0, color='gray', linestyle='-', alpha=0.3)

ax1.axvline(x=0, color='gray', linestyle='-', alpha=0.3)

# График 2: Остатки

ax2 = fig.add\_subplot(2, 2, 2)

ax2.stem(x, residuals, linefmt='r-', markerfmt='ro', basefmt='k-')

ax2.axhline(y=0, color='black', linestyle='-', linewidth=1)

ax2.set\_xlabel('x', fontsize=12)

ax2.set\_ylabel('Остатки eᵢ = yᵢ - ỹᵢ', fontsize=12)

ax2.set\_title('График остатков', fontsize=14, fontweight='bold')

ax2.grid(True, alpha=0.3)

# График 3: Сравнение фактических и предсказанных значений

ax3 = fig.add\_subplot(2, 2, 3)

min\_val = min(min(y), min(y\_pred)) - 1

max\_val = max(max(y), max(y\_pred)) + 1

ax3.plot([min\_val, max\_val], [min\_val, max\_val], 'k--', alpha=0.7, label='Линия идеального соответствия')

ax3.scatter(y, y\_pred, color='green', s=100, zorder=5)

# Добавляем подписи для каждой точки

for i in range(n):

    ax3.annotate(f'({x[i]})', (y[i], y\_pred[i]), xytext=(5, 5),

                 textcoords='offset points', fontsize=9)

ax3.set\_xlabel('Фактические значения yᵢ', fontsize=12)

ax3.set\_ylabel('Предсказанные значения ỹᵢ', fontsize=12)

ax3.set\_title('Сравнение фактических и предсказанных значений', fontsize=14, fontweight='bold')

ax3.grid(True, alpha=0.3)

ax3.legend(loc='best', fontsize=10)

# График 4: Информация о модели

ax4 = fig.add\_subplot(2, 2, 4)

ax4.axis('off')

# Текстовая информация

info\_text = f"""

МОДЕЛЬ РЕГРЕССИИ:

ỹ = {a0:.4f} + {a1:.4f}x + ({a2:.4f})x²

КАЧЕСТВО МОДЕЛИ:

• Общая дисперсия: s²\_общ = {s2\_total:.4f}

• Остаточная дисперсия: s²\_ост = {s2\_residual:.4f}

• F\_набл = {F\_observed:.4f}

• F\_кр({alpha}; {f1}; {f2}) = {F\_critical:.4f}

• Результат: {fisher\_result}

ЗНАЧИМОСТЬ КОЭФФИЦИЕНТОВ:

• t₀ = {t0:.4f}  {'>' if t0 > t\_critical else '<'} t\_кр = {t\_critical:.4f} → {student\_results[0]}

• t₁ = {t1:.4f}  {'>' if t1 > t\_critical else '<'} t\_кр = {t\_critical:.4f} → {student\_results[1]}

• t₂ = {t2:.4f}  {'>' if t2 > t\_critical else '<'} t\_кр = {t\_critical:.4f} → {student\_results[2]}

СУММЫ КВАДРАТОВ:

• SS\_общ = {SS\_tot:.4f}

• SS\_рег = {SS\_reg:.4f}

• SS\_ост = {SS\_res:.4f}

• R² = {SS\_reg/SS\_tot:.4f}

"""

ax4.text(0.1, 0.5, info\_text, transform=ax4.transAxes, fontsize=11,

         verticalalignment='center', fontfamily='monospace',

         bbox=dict(boxstyle="round,pad=1", facecolor="lightyellow", edgecolor="black"))

plt.suptitle(f'ОДНОФАКТОРНЫЙ РЕГРЕССИОННЫЙ АНАЛИЗ (Вариант 97)', fontsize=16, fontweight='bold')

plt.tight\_layout()

plt.savefig(f"{output\_dir}/01\_regression\_analysis.png", dpi=300, bbox\_inches='tight')

print(f"График сохранен: {output\_dir}/01\_regression\_analysis.png")

# Дополнительный график: доверительные интервалы

fig2, ax = plt.subplots(figsize=(12, 8))

# Точки исходных данных

ax.scatter(x, y, color='red', s=100, zorder=5, label='Экспериментальные данные')

# Регрессионная кривая

x\_smooth = np.linspace(min(x)-0.5, max(x)+0.5, 100)

y\_smooth = a0 + a1 \* x\_smooth + a2 \* x\_smooth\*\*2

ax.plot(x\_smooth, y\_smooth, 'b-', linewidth=2, label='Регрессионная кривая')

# Доверительные интервалы для среднего значения

# Для этого нужна матрица плана для новых точек

X\_new = np.column\_stack((np.ones\_like(x\_smooth), x\_smooth, x\_smooth\*\*2))

# Дисперсии предсказанных средних

var\_pred = np.diag(X\_new @ FTF\_inv @ X\_new.T) \* s2\_residual

std\_pred = np.sqrt(var\_pred)

# Доверительные интервалы (95%)

t\_crit\_95 = stats.t.ppf(0.975, df\_student)

ci\_upper = y\_smooth + t\_crit\_95 \* std\_pred

ci\_lower = y\_smooth - t\_crit\_95 \* std\_pred

ax.fill\_between(x\_smooth, ci\_lower, ci\_upper, alpha=0.2, color='blue', label='95% доверительный интервал')

# Оформление

ax.set\_xlabel('x', fontsize=14)

ax.set\_ylabel('y', fontsize=14)

ax.set\_title('Регрессионная кривая с доверительным интервалом', fontsize=16, fontweight='bold')

ax.legend(loc='best', fontsize=12)

ax.grid(True, alpha=0.3)

ax.axhline(y=0, color='gray', linestyle='-', alpha=0.3)

ax.axvline(x=0, color='gray', linestyle='-', alpha=0.3)

plt.tight\_layout()

plt.savefig(f"{output\_dir}/02\_confidence\_intervals.png", dpi=300, bbox\_inches='tight')

print(f"График сохранен: {output\_dir}/02\_confidence\_intervals.png")

# 13. ИТОГОВАЯ ТАБЛИЦА

print("\n" + "=" \* 80)

print("ИТОГОВЫЕ РЕЗУЛЬТАТЫ")

print("=" \* 80)

results\_summary = [

    ["Уравнение регрессии", f"ỹ = {a0:.4f} + {a1:.4f}x + ({a2:.4f})x²"],

    ["Общая дисперсия s²\_общ", f"{s2\_total:.4f}"],

    ["Остаточная дисперсия s²\_ост", f"{s2\_residual:.4f}"],

    ["Наблюдаемое значение F\_набл", f"{F\_observed:.4f}"],

    [f"Критическое значение F\_кр({alpha}; {f1}; {f2})", f"{F\_critical:.4f}"],

    ["Результат проверки адекватности", f"{fisher\_result}"],

    [f"Критическое значение t\_кр({alpha}; {df\_student})", f"{t\_critical:.4f}"],

    ["t₀ (для a₀)", f"{t0:.4f} ({'значим' if t0 > t\_critical else 'незначим'})"],

    ["t₁ (для a₁)", f"{t1:.4f} ({'значим' if t1 > t\_critical else 'незначим'})"],

    ["t₂ (для a₂)", f"{t2:.4f} ({'значим' if t2 > t\_critical else 'незначим'})"],

    ["Коэффициент детерминации R²", f"{SS\_reg/SS\_tot:.4f}"]

]

print(tabulate(results\_summary, headers=["Показатель", "Значение"], tablefmt="grid"))

print("\n" + "=" \* 80)

print(f"Все результаты сохранены в директории: {output\_dir}")

print("=" \* 80)

print("\nРАБОТА ВЫПОЛНЕНА УСПЕШНО!")

--- ТЕКСТ ПРОГРАММЫ ---