## Gerb-BMSTU_01Министерство 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** Климов И.С.

**Группа** ИУ7-42Б

## Оценка (баллы)

**Преподаватель** Градов В.М.

Москва. 2021 г

**Цель работы**: Получение навыков владения метода наименьших квадратов с использованием полинома заданной степени при аппроксимации табличных функций с весами.

1. **Исходные данные**
2. Таблица функции с **весами** pi с количеством узлов N. Сформировать таблицу самостоятельно со случайным разбросом точек.
3. Степень аппроксимирующего полинома.
4. **Код программы**

**Листинг 1. table.py**

**def** **print\_table**(table):

**def** **separator**():

**print**('|', '-' \* **9**, '|', '-' \* **9**, '|', '-' \* **9**, '|', sep='')

**print**('Исходная таблица:')

separator()

**print**('| x | y | Вес |')

separator()

**for** line **in** table:

**print**(f'|{str(line[0]):^9s}|{str(line[1]):^9s}|{str(line[2]):^9s}|')

separator()

**print**()

**def** **create\_table**(filename):

**with** open(f'data/{filename}') **as** f:

table = [tuple(map(float, line.split())) **for** line **in** f **if** line]

**return** table

**Листинг 2. find.py**

**def** **get\_system**(table, degree):

degree += **1**

n = len(table)

zeros = [**0** **for** \_ **in** range(degree)]

system = [zeros[:] **for** \_ **in** range(degree)]

free\_column = zeros[:]

**for** i **in** range(degree):

**for** j **in** range(n):

tmp = table[j][**2**] \* (table[j][**0**] \*\* i)

**for** z **in** range(degree):

system[i][z] += tmp \* (table[j][**0**] \*\* z)

free\_column[i] += tmp \* table[j][**1**]

**return** system, free\_column

**def** **get\_inverse\_matrix**(matrix):

n = len(matrix)

zeros = [**0** **for** \_ **in** range(n)]

inverse\_matrix = [zeros[:] **for** \_ **in** range(n)]

**for** i **in** range(n):

column = convert\_column(matrix, i)

**for** j **in** range(n):

inverse\_matrix[j][i] = column[j]

**return** inverse\_matrix

**def** **convert\_column**(matrix, index):

n = len(matrix)

extended\_matrix = [matrix[i][:] **for** i **in** range(n)]

new\_column = [**0** **for** \_ **in** range(n)]

**for** i **in** range(n):

extended\_matrix[i] += [**1.0**] **if** i == index **else** [**0.0**]

**for** i **in** range(n):

**if** **not** extended\_matrix[i][i]:

**for** j **in** range(i + **1**, n):

**if** extended\_matrix[j][j]:

extended\_matrix[i], extended\_matrix[j] = extended\_matrix[j], extended\_matrix[i]

**for** j **in** range(i + **1**, n):

d = -extended\_matrix[j][i] / extended\_matrix[i][i]

**for** z **in** range(n + **1**):

extended\_matrix[j][z] += d \* extended\_matrix[i][z]

**for** i **in** range(n - **1**, -**1**, -**1**):

result = **0**

**for** j **in** range(n):

result += extended\_matrix[i][j] \* new\_column[j]

new\_column[i] = (extended\_matrix[i][n] - result) / extended\_matrix[i][i]

**return** new\_column

**def** **multiply**(inverse\_matrix, free\_column):

n = len(free\_column)

result = [**0** **for** \_ **in** range(n)]

**for** i **in** range(n):

**for** j **in** range(n):

result[i] += free\_column[j] \* inverse\_matrix[i][j]

**return** result

**def** **find\_coefficients**(table, degree):

system, free\_column = get\_system(table, degree)

inverse\_matrix = get\_inverse\_matrix(system)

coefficients = multiply(inverse\_matrix, free\_column)

**return** coefficients

**Листинг 3. show.py**

**import** **numpy** **as** **np**

**import** **matplotlib**

**import** **matplotlib.pyplot** **as** **plt**

**def** **show\_result**(table, result, degree):

matplotlib.use('TkAgg')

**if** len(table) > **1**:

dx = table[**1**][**0**] - table[**0**][**0**]

**else**:

dx = **10**

x = np.linspace(table[**0**][**0**] - dx, table[-**1**][**0**] + dx, **100**)

y = []

**for** xi **in** x:

tmp = **0**

**for** i **in** range(degree + **1**):

tmp += xi \*\* i \* result[i]

y.append(tmp)

plt.plot(x, y)

y\_min, y\_max = min(y), max(y)

x = [point[**0**] **for** point **in** table]

y = [point[**1**] **for** point **in** table]

plt.plot(x, y, 'o', color='red', label='Исходные точки')

plt.legend(loc='best')

y\_min, y\_max = min(y\_min, min(y)), max(y\_max, max(y))

dy = (y\_max - y\_min) \* **0.03**

plt.axis([table[**0**][**0**] - dx, table[-**1**][**0**] + dx, y\_min - dy, y\_max + dy])

plt.show()

**Листинг 4. main.py**

**from** **find** **import** find\_coefficients

**from** **show** **import** show\_result

**from** **table** **import** create\_table, print\_table

**def** **main**():

filename = '1.txt'

**print**(f'Результат на основе данных файла data/{filename}**\n**')

table = create\_table(filename)

print\_table(table)

**try**:

degree = int(input('Введите степень полинома: '))

**except** **ValueError**:

**return** **print**('Ошибка! Вы должны были ввести число')

result = find\_coefficients(table, degree)

**print**('**\n**Результат можно увидеть на появившемся графике')

show\_result(table, result, degree)

**if** \_\_name\_\_ == '\_\_main\_\_':

main()

1. **Результаты работы**
2. Веса всех точек одинаковые

Результат на основе данных файла data/1.txt

Исходная таблица:

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

| x | y | Вес |

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

| 1.0 | 1.0 | 1.0 |

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

| 2.0 | 4.0 | 1.0 |

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

| 5.0 | 3.0 | 1.0 |

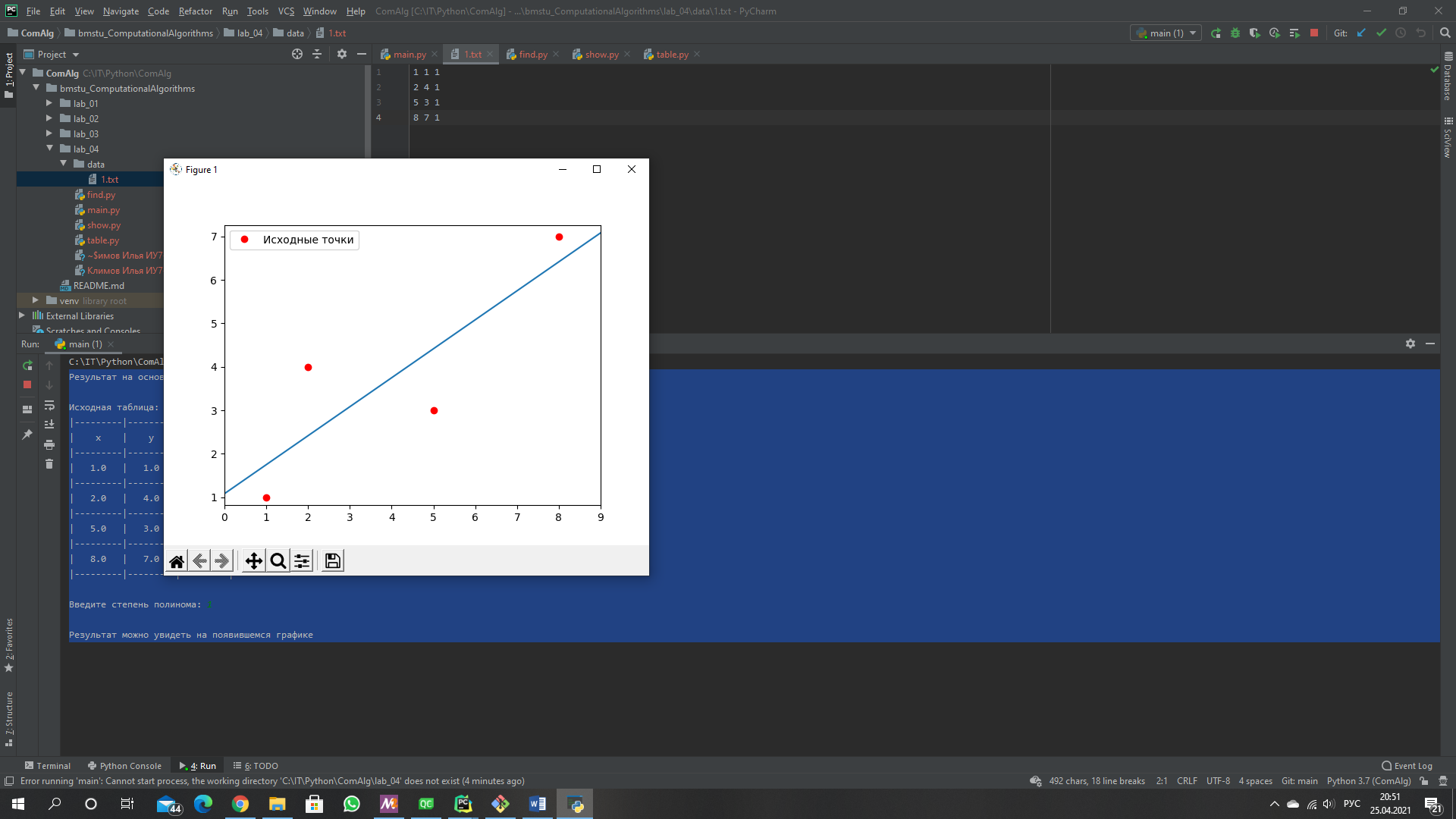
|---------|---------|---------|

| 8.0 | 7.0 | 1.0 |

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

Введите степень полинома: 1

Результат можно увидеть на появившемся графике



Результат на основе данных файла data/1.txt

Исходная таблица:

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

| x | y | Вес |

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

| 1.0 | 1.0 | 1.0 |

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

| 2.0 | 4.0 | 1.0 |

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

| 5.0 | 3.0 | 1.0 |

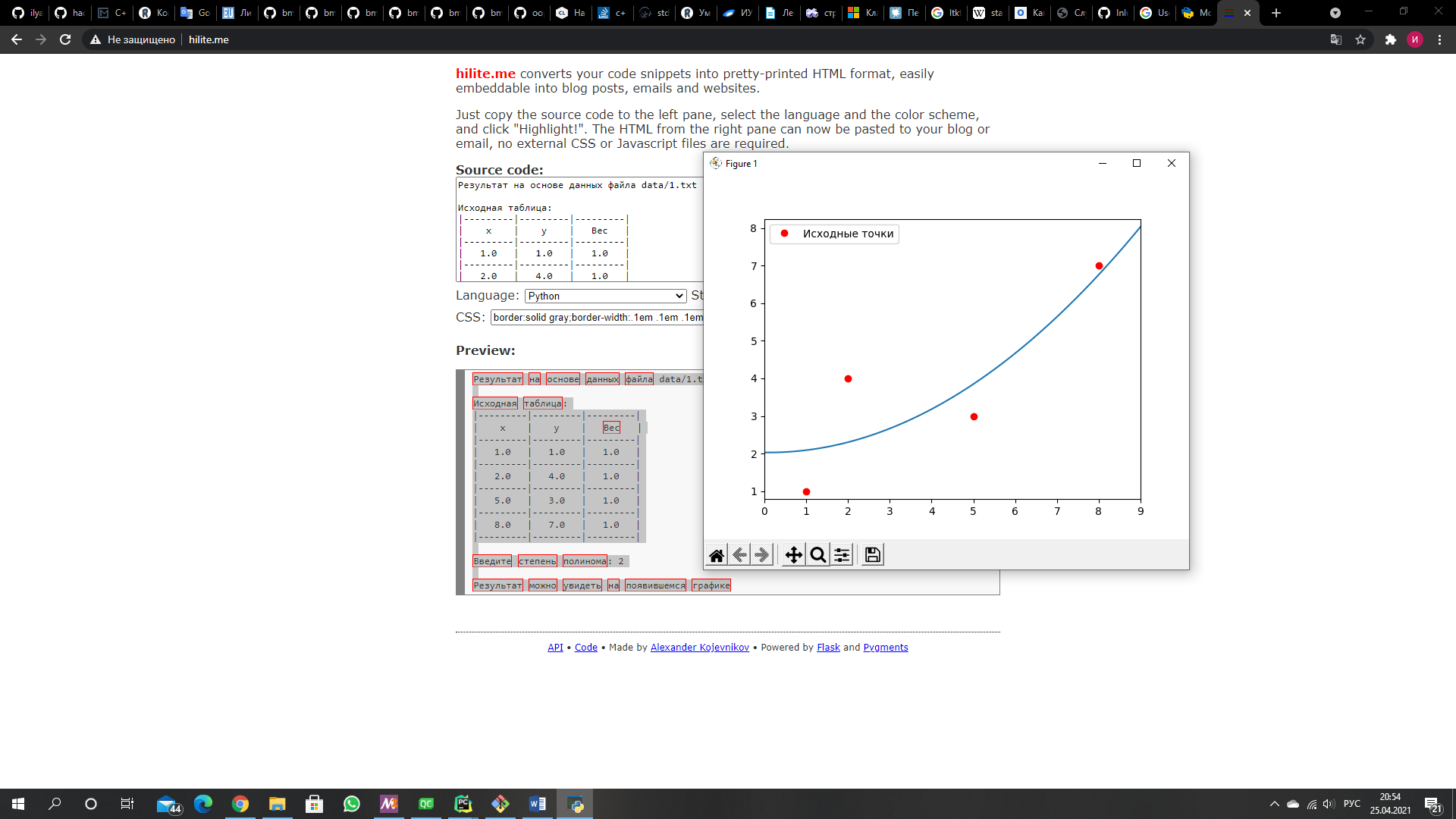
|---------|---------|---------|

| 8.0 | 7.0 | 1.0 |

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

Введите степень полинома: 2

Результат можно увидеть на появившемся графике



Результат на основе данных файла data/1.txt

Исходная таблица:

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

| x | y | Вес |

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

| 1.0 | 1.0 | 1.0 |

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

| 2.0 | 4.0 | 1.0 |

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

| 5.0 | 3.0 | 1.0 |

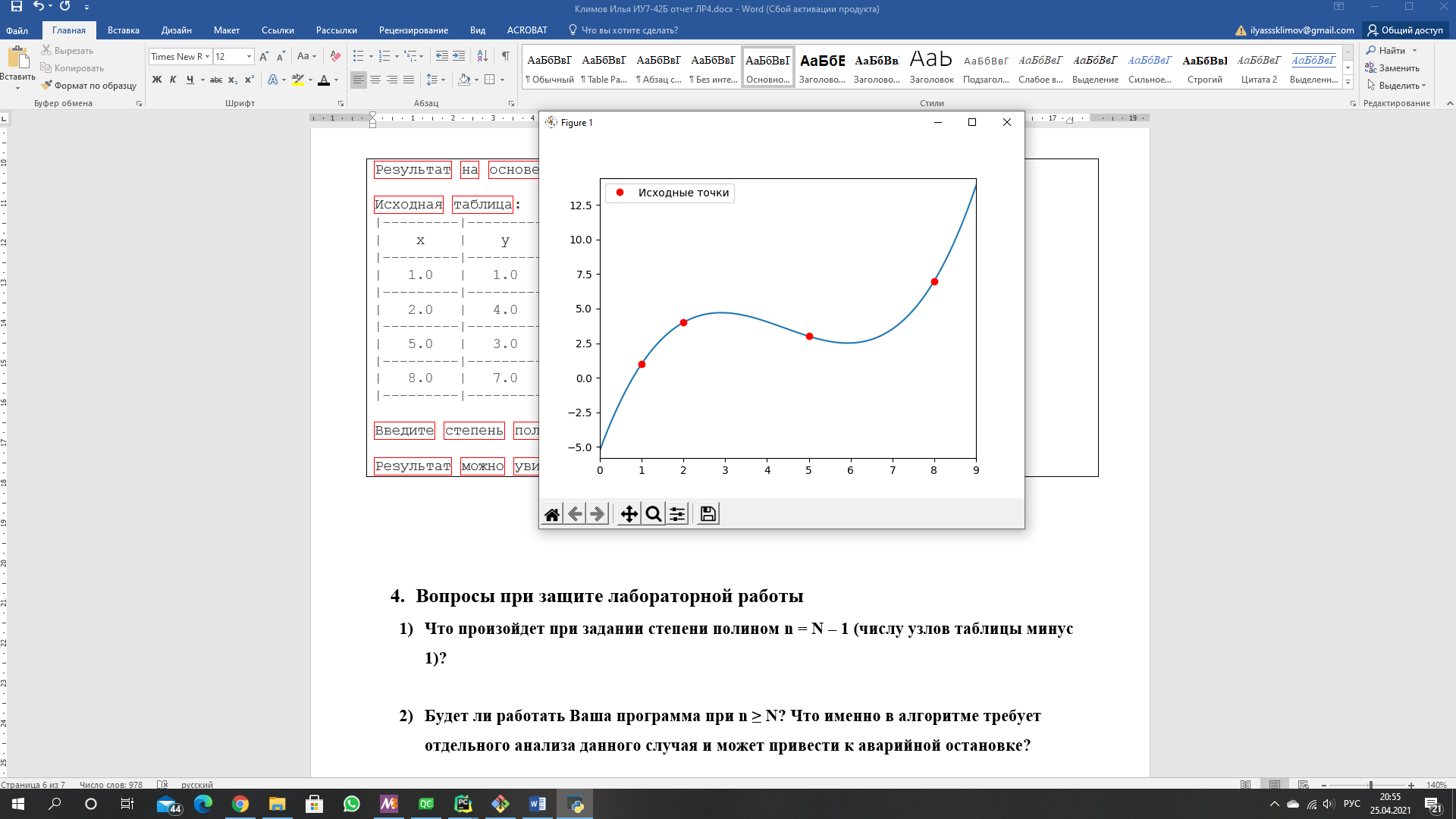
|---------|---------|---------|

| 8.0 | 7.0 | 1.0 |

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

Введите степень полинома: 3

Результат можно увидеть на появившемся графике



1. Веса одних и тех же точек одинаковые и разные (сравнение)

Результат на основе данных файла data/2.txt

Исходная таблица:

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

| x | y | Вес |

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

| 3.0 | 3.0 | 1.0 |

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

| 5.0 | 4.0 | 1.0 |

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

| 4.0 | 5.0 | 1.0 |

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

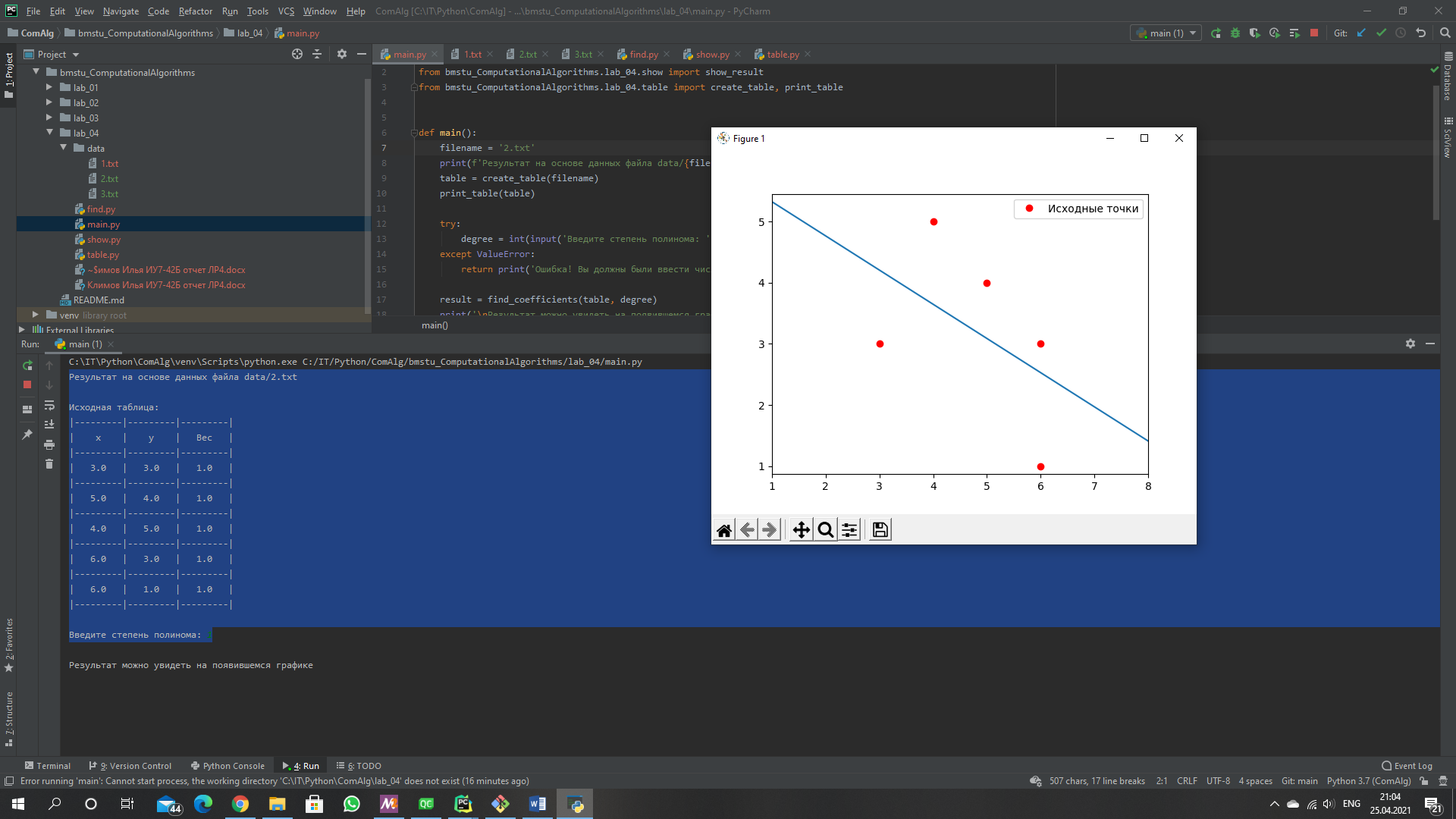
| 6.0 | 3.0 | 1.0 |

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

| 6.0 | 1.0 | 1.0 |

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

Введите степень полинома: 1



Результат на основе данных файла data/3.txt

Исходная таблица:

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

| x | y | Вес |

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

| 3.0 | 3.0 | 9.0 |

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

| 5.0 | 4.0 | 0.1 |

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

| 4.0 | 5.0 | 10.0 |

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

| 6.0 | 3.0 | 0.3 |

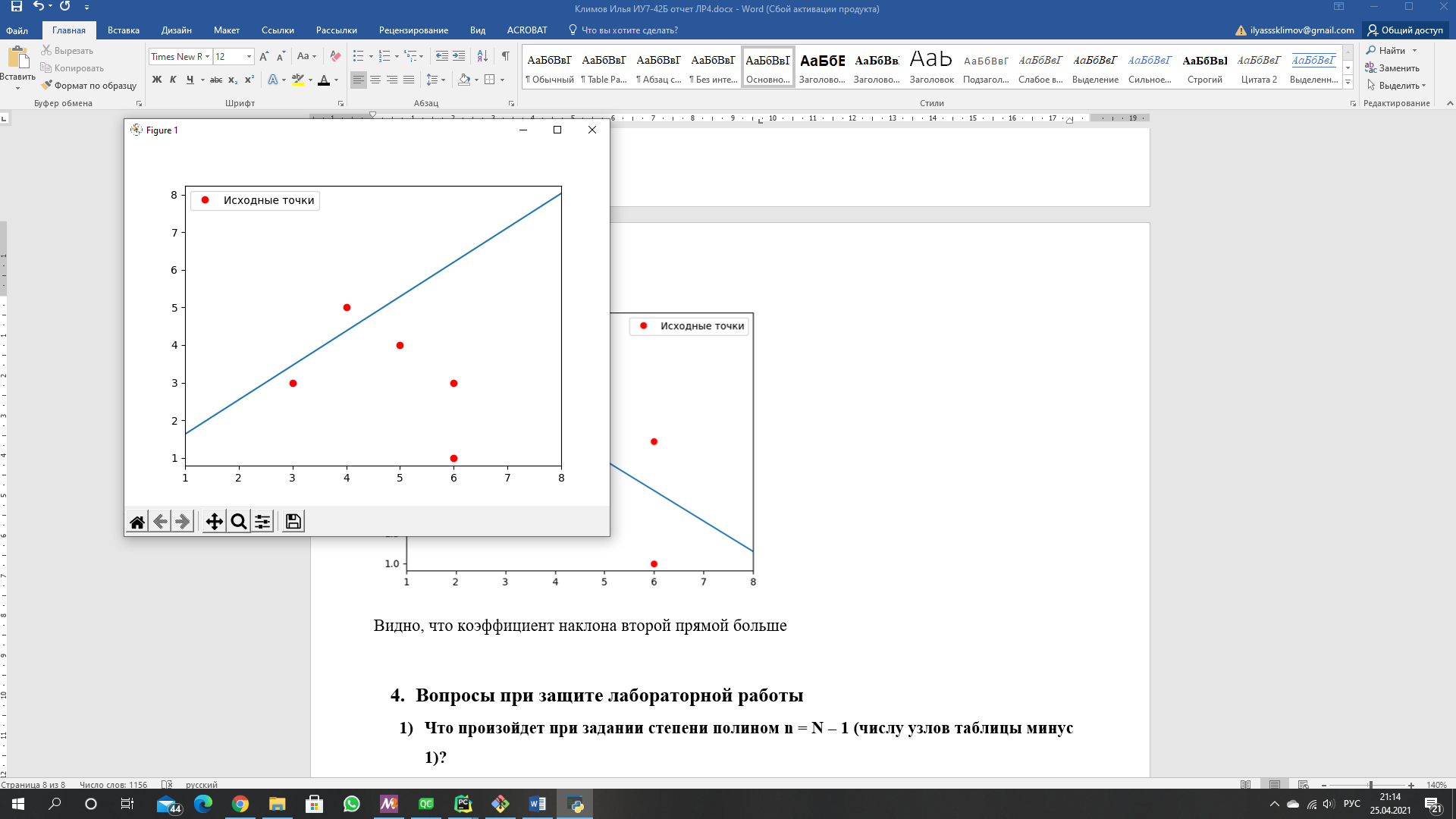
|---------|---------|---------|

| 6.0 | 1.0 | 0.2 |

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

Введите степень полинома: 1

Результат можно увидеть на появившемся графике



Благодаря изменению весов точек был изменен знак углового коэффициента прямой

1. **Вопросы при защите лабораторной работы**
2. **Что произойдет при задании степени полинома n = N – 1 (числу узлов таблицы минус 1)?**

При n = N – 1 получим полином, прошедший через все узлы таблицы, то есть график будет проходить через все табличные точки.

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

В моем случае программа работать будет, но не так, как ожидается. При таких входных данных СЛАУ не будет иметь решения, так как определитель матрицы будет равен 0. Поэтому в программу следует добавить дополнительную проверку.

1. **Получить формулу для коэффициента полинома a0 при степени полинома n = 0. Какой смысл имеет величина, которую представляет данный коэффициент?**

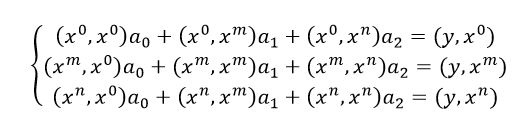
Данная величина будет иметь смысл математического ожидания, где – вероятность «встретить»

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

Данные входные данные подходят под случай, описанный в пункте 2, то есть определитель будет равен 0.

1. **Построить СЛАУ при выборочном задании степеней аргумента полинома , причем степени n и m в этой формуле известны.**

Получим следующую СЛАУ (обнулив остальные коэффициенты):

****

1. **Предложить схему алгоритма решения задачи из вопроса 5, если степени n и m подлежат определению наравне с коэффициентами ak, т.е. количество неизвестных равно 5.**

Предварительно можно найти степени n и m, а затем использовать предыдущую формулу для составления СЛАУ.