| Gerb-BMSTU_01 | **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 (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

**ОТЧЕТ**

| **по лабораторной работе №** | 4 |
| --- | --- |



Построение и программная реализация алгоритма наилучшего среднеквадратичного приближения

**Дисциплина:** Вычислительные алгоритмы

| Студент | ИУ7И - 46Б |  |  | Андрич К. |
| --- | --- | --- | --- | --- |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | В.М. Градов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2021

Цель работы

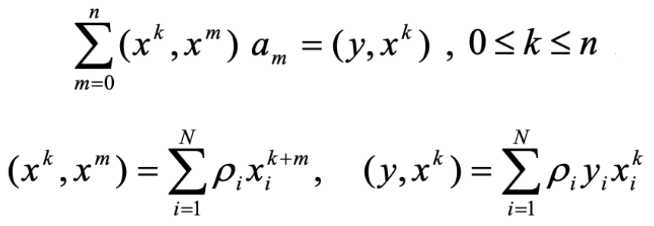
Получение навыков построения алгоритма метода наименьших квадратов с использованием полинома заданной степени при аппроксимации табличных функций с весами

Исходные данные

1. Таблица функции с весами i с количеством узлов N. Сформировать таблицу самостоятельно со случайным разбросом точек. Предусмотреть в интерфейсе удобную возможность изменения пользователем весов в таблице
2. Степень аппроксимирующего полинома - n

Описание алгоритма

1. Возьмем степень полинома *n < N*
2. Составим СЛАУ вида



1. В решении СЛАУ находится коэффициенты полинома

Код программы

В программе есть 3 файлов: main.py, functions.py и result.py

main.py

| from functions import \*  def main():  tbl = make\_table()  option = 2  while (option != 0):  choice(option, tbl)  menu()  try:  option = int(input("\nInput choice: "))  except:  print("Invalid input")  continue  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| --- |

functions.py

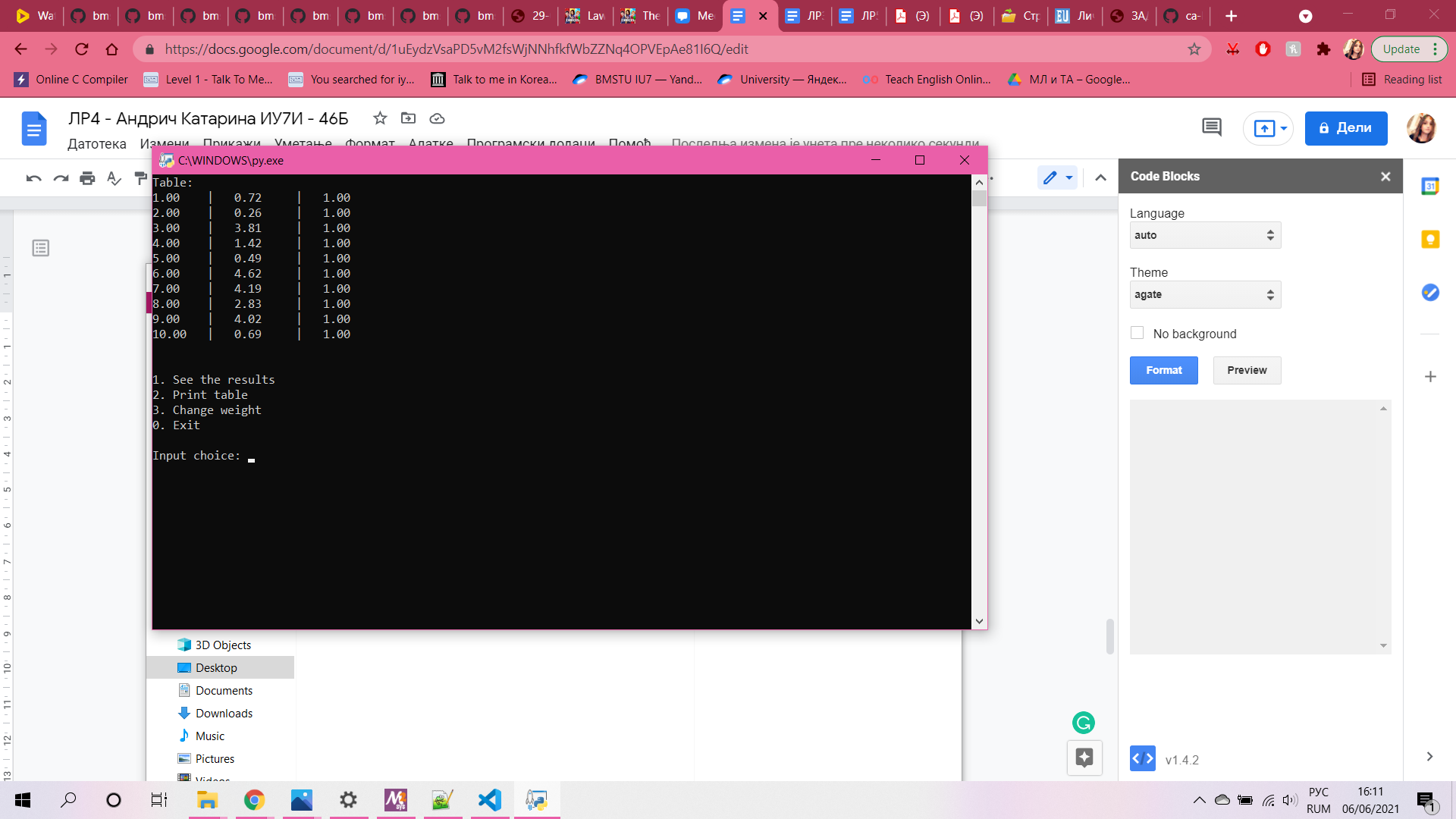
| from random import random from result import \*   def make\_table():  tbl = []  for i in range(SIZE):  tbl.append([i + 1, random() \* 5, 1])  return tbl   def table\_print(tbl):  print("Table:")  for i in range(SIZE):  print("%-5.2f | %-5.2f | %-5.2f " %(tbl[i][0], tbl[i][1], tbl[i][2]))  print("\n")  def change\_weight(tbl):  try:  indx = int(input("\nInput row number: "))  except:  print("Invalid input")  return tbl  if (indx < 1 or indx > SIZE):  print("The row you chose doesn't exist")  return tbl  try:  weight = int(input("\nInput weight: "))  except:  print("Invalid input")  return tbl  tbl[indx - 1][2] = weight  return tbl  def choice(argument, tbl):  if argument == 1:  result(tbl)  elif argument == 2:  table\_print(tbl)  if argument == 3:  change\_weight(tbl)  def menu():  print("1. See the results")  print("2. Print table")  print("3. Change weight")  print("0. Exit") |
| --- |

result.py

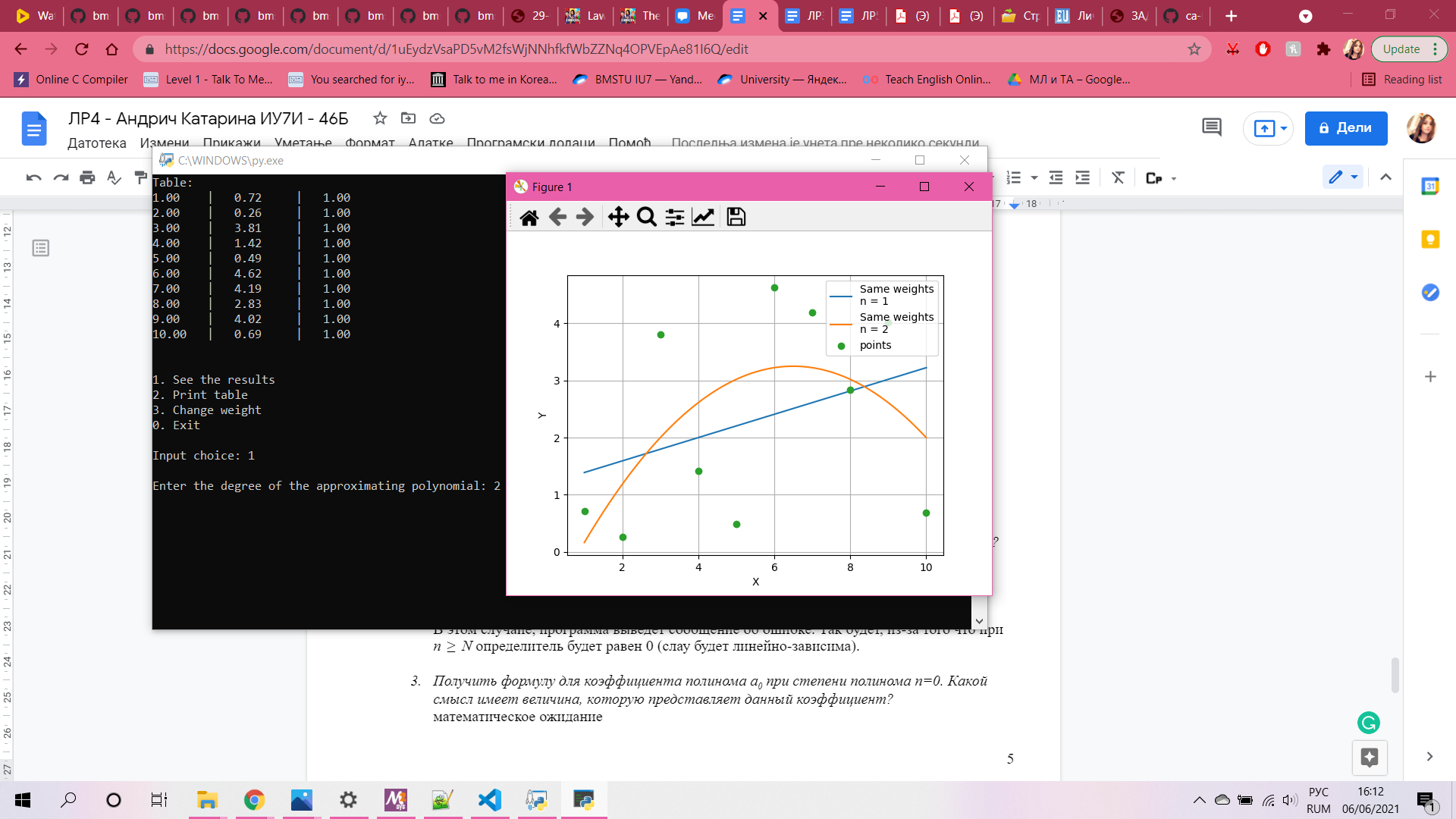
| import matplotlib.pyplot as plt  EPS = 0.001 SIZE = 10  def check\_for\_changes(tbl):  for i in range(SIZE):  if not (tbl[i][2] == 1):  return True   return False  def mtx\_model(i):  mtx = []  for j in range(i + 1):  row = []  for k in range(i + 2):  row.append(0)  mtx.append(row)  return mtx  def slae\_mtx(tbl, i):  mtx = mtx\_model(i)  for j in range(i + 1):  for k in range(i + 1):  mtx[j][k] = 0  mtx[j][i + 1] = 0  for l in range(SIZE):  weight = tbl[l][2]  x, y = tbl[l][0], tbl[l][1]  mtx[j][k] += weight \* pow(x, (j + k))  mtx[j][i + 1] += weight \* y \* pow(x, j)  return mtx  def gauss(mtx):  res = []  mxlen = len(mtx)  for i in range(SIZE):  for j in range(i + 1, mxlen):  if (i == j):  continue  k = mtx[j][i] / mtx[i][i]  for l in range(i, mxlen + 1):  mtx[j][l] -= k \* mtx[i][l]  for i in range(mxlen):  res.append(0)  k = -1  for i in range(mxlen - 1, k, -1):  for j in range(mxlen - 1, i, -1):  mtx[i][mxlen] -= res[j] \* mtx[i][j]  res[i] = mtx[i][mxlen] / mtx[i][i]  return res  def get\_points(tbl, i):  mtx = slae\_mtx(tbl, i)  res = gauss(mtx)  X, Y = [], []  x = tbl[0][0] - EPS  while (x <= tbl[SIZE - 1][0] + EPS):  y = 0  for j in range(0, i + 1):  y += res[j] \* pow(x, j)  X.append(x)  Y.append(y)  x += EPS  return X, Y  def plot(tbl, n, points):  for i in range(1, n + 1):  if (i > 2 and i < n):  continue  X, Y = get\_points(tbl, i)  plt.plot(X, Y, "-", label = "%s\nn = %d" %(points, i))  def get\_original(tbl):  new = []  for i in range(SIZE):  new.append([tbl[i][0], tbl[i][1], 1])  return new  def draw\_points(tbl):  X, Y = [], []  for i in range(SIZE):  X.append(tbl[i][0])  Y.append(tbl[i][1])  return X, Y  def result(tbl):  try:  n = int(input("\nEnter the degree of the approximating polynomial: "))  except:  print("Invalid input")  return tbl  if n <= 0 or n >= SIZE:  print("Invalid degree input \n")  return tbl  change\_indx = check\_for\_changes(tbl)  if change\_indx:  points = "Different weights"  else:  points = "Same weights"  plot(tbl, n, points)  if change\_indx:  points = "Same weights"  same = get\_original(tbl)  plot(same, n, points)  PX, PY = draw\_points(tbl)  plt.plot(PX, PY, 'o', label = "points")  plt.legend()  plt.grid()  plt.xlabel("X")  plt.ylabel("Y")  plt.show() |
| --- |

Результаты работы

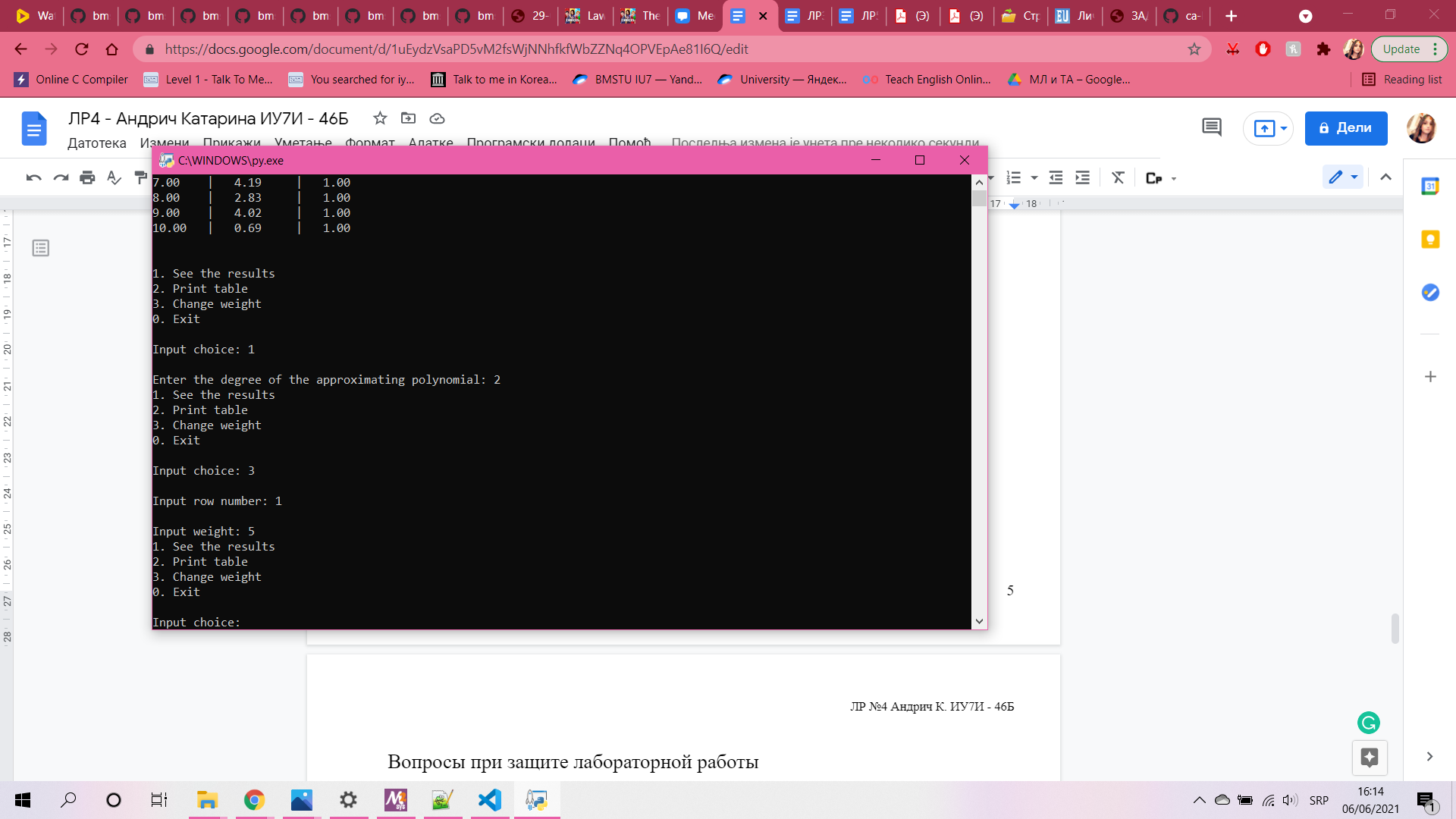
Интерфейс:

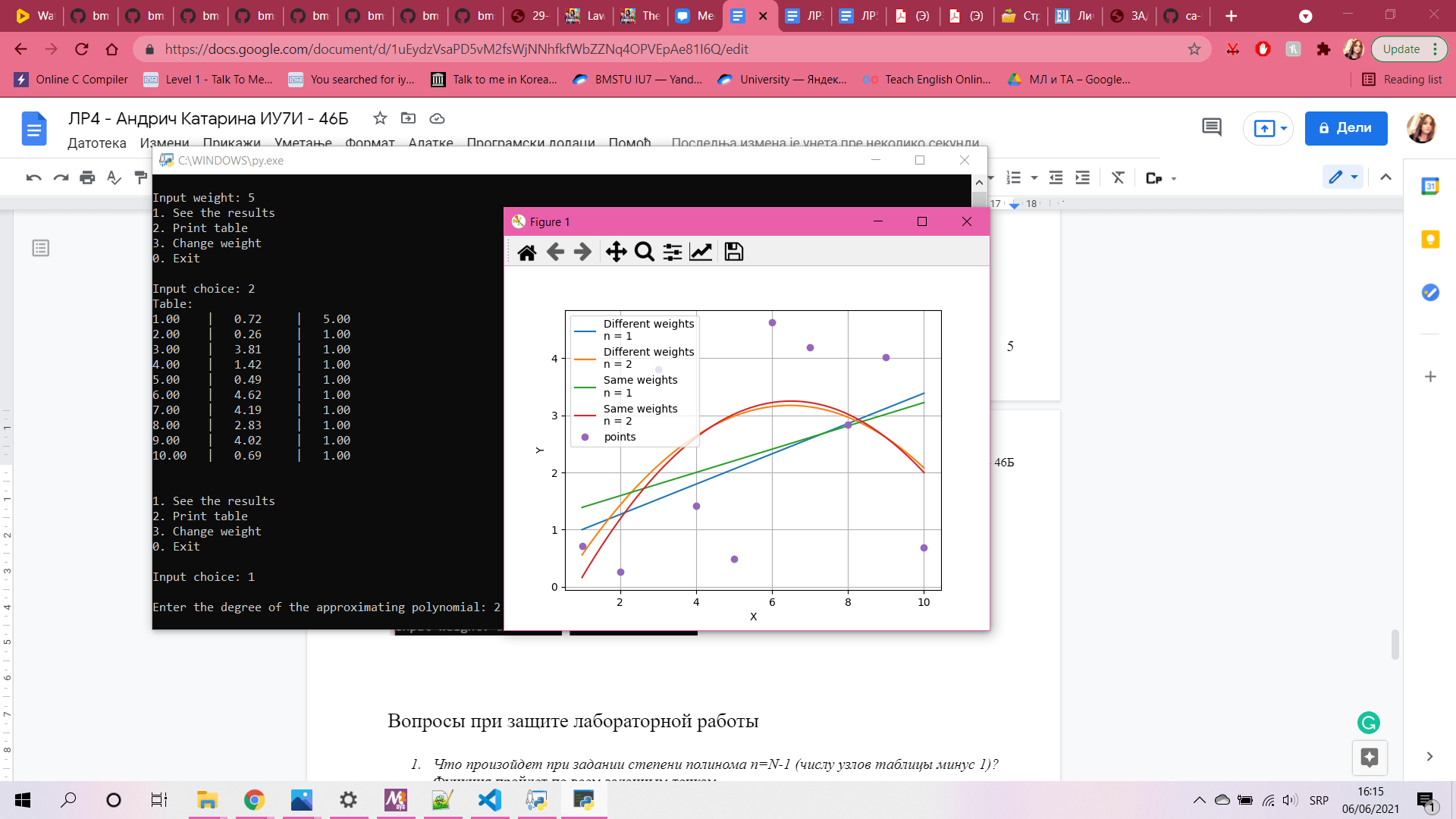


Если не изменаем вес (степень аппроксимирующего полинома 2):



Если изменаем вес (степень аппроксимирующего полинома 2):

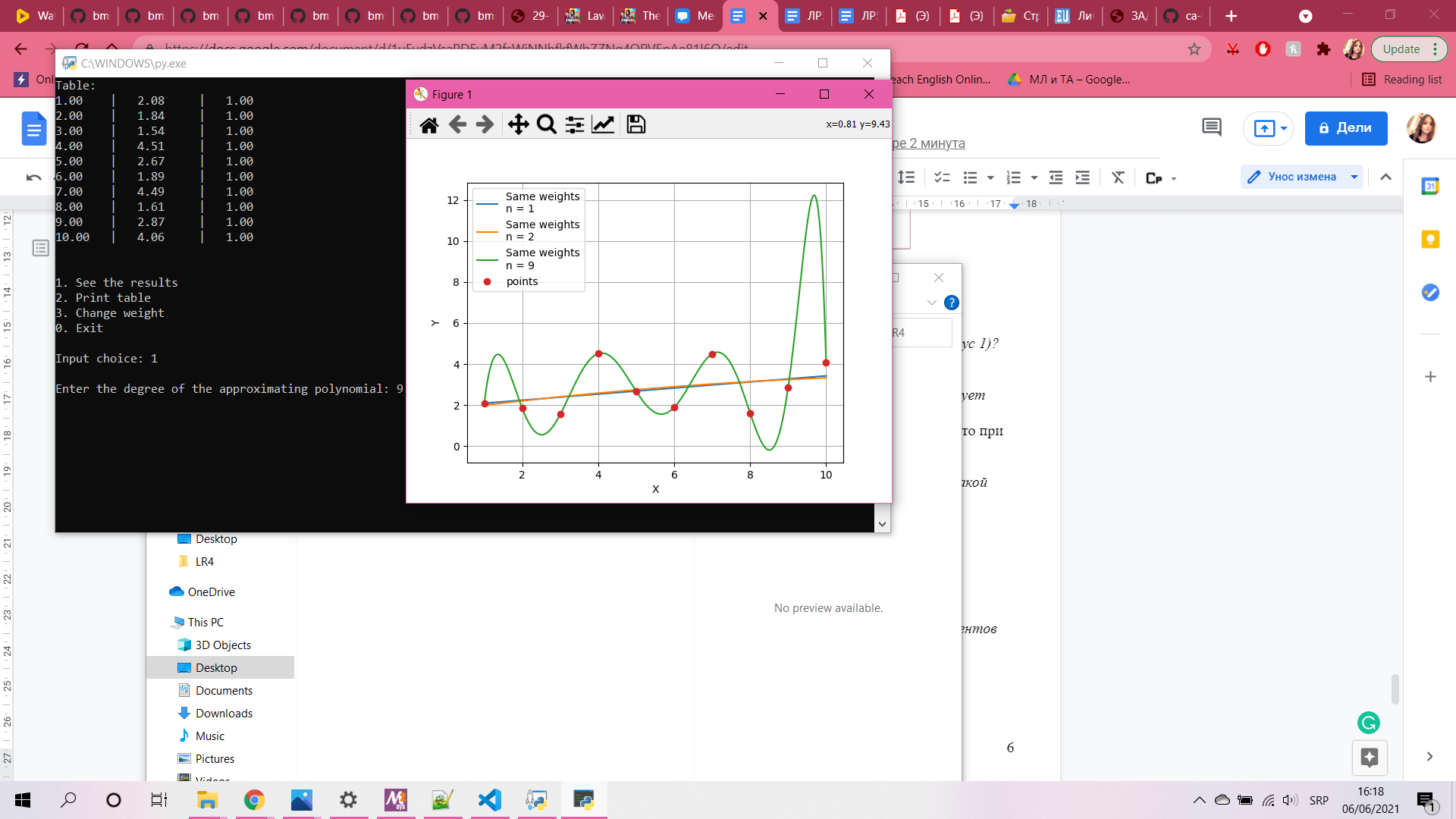




Вопросы при защите лабораторной работы

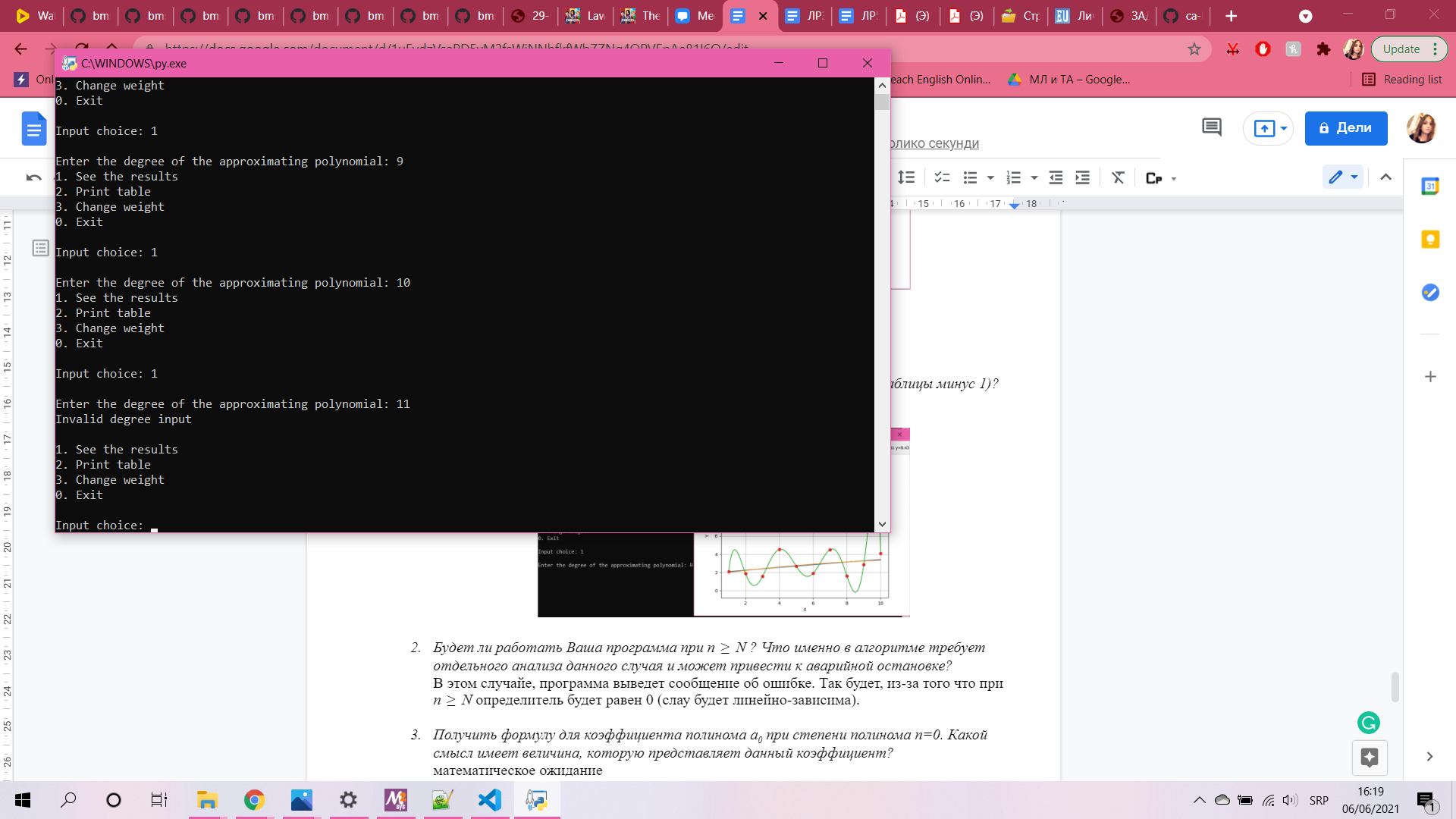
1. *Что произойдет при задании степени полинома n=N-1 (числу узлов таблицы минус 1)?*

Функцияпройдет по всем заданным точкам



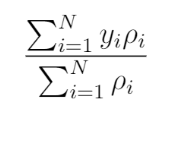
1. *Будет ли работать Ваша программа при n N ? Что именно в алгоритме требует отдельного анализа данного случая и может привести к аварийной остановке?*

В этом случайе, программа выведет сообщение об ошибке. Так будет, из-за того что при *n N* определитель будет равен 0 (слау будет линейно-зависима).

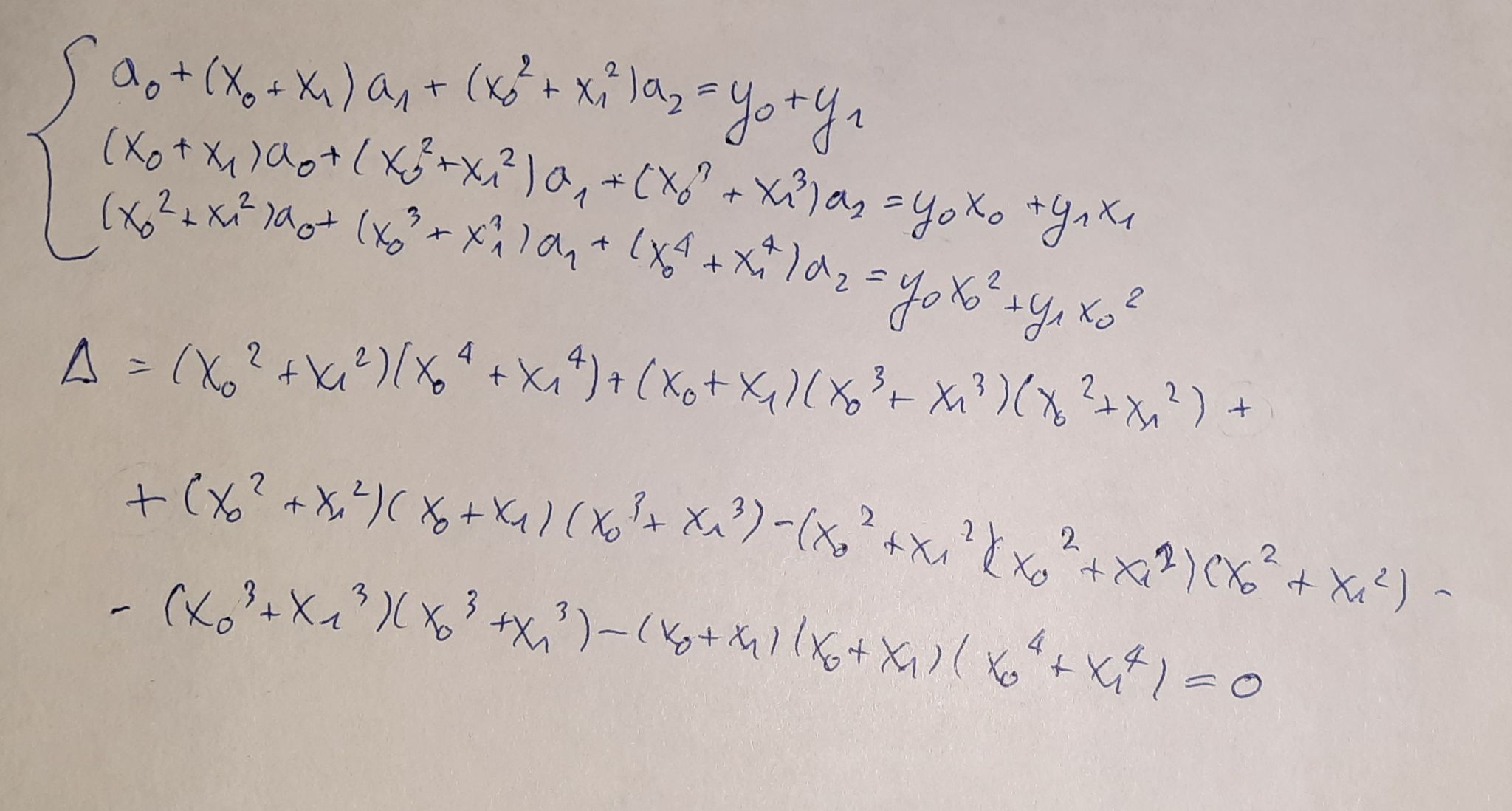


1. *Получить формулу для коэффициента полинома a0 при степени полинома n=0. Какой смысл имеет величина, которую представляет данный коэффициент?*

математическое ожидание

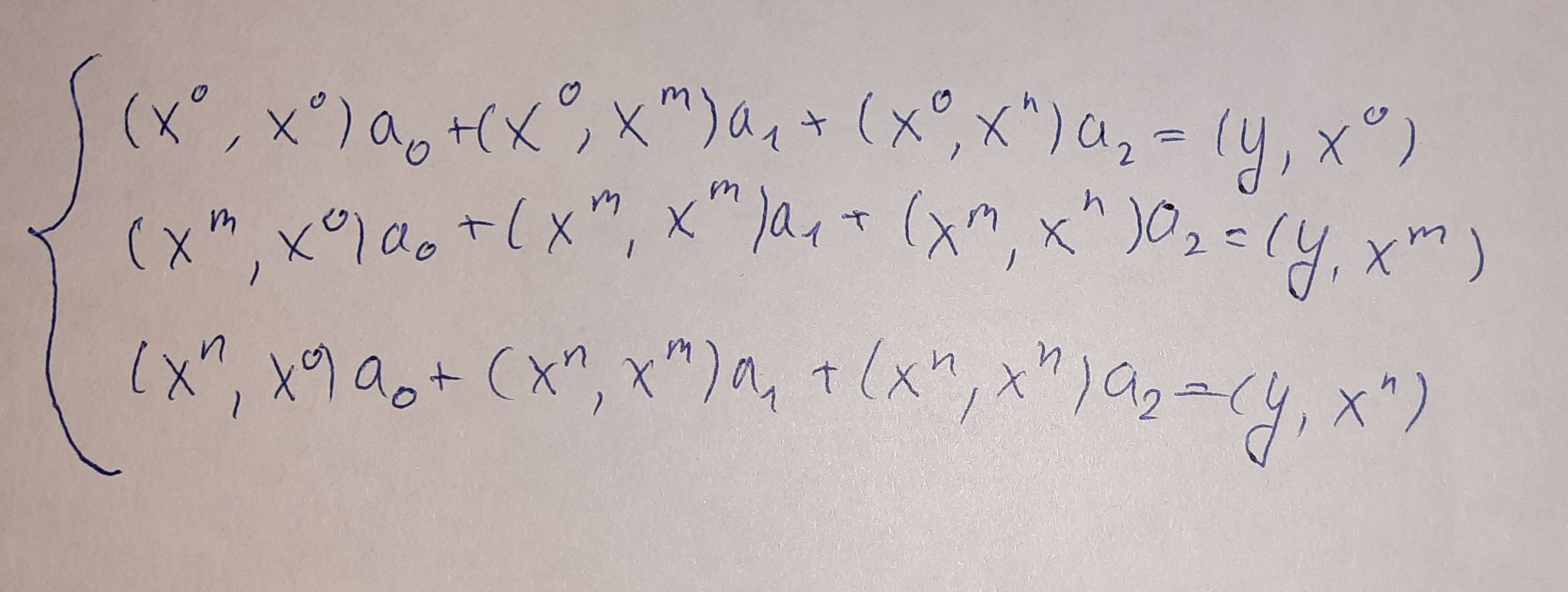


1. *Записать и вычислить определитель матрицы СЛАУ для нахождения коэффициентов полинома для случая, когда n=N=2. Принять все =1.*



Из-за того что , ситема не имеет решение

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



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*

Сначала сделаем перебор всех степеней n и m которые возможны. Затем, для каждый пар степени найдем коэффициент *а*. Наконец, выберем пар, у которого значение найболе блиско эталонному.