Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования   
«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчет защищен с оценкой \_\_\_\_\_

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Отчет

по лабораторной работе № 5

"Интерполяционный многочлен Лагранжа"

по дисциплине

«Вычислительные алгоритмы»

Студенты гр. ПИ-92 Шинтяпин И.И.

Шульпов В.М.

Преподаватель, к.т.н. Проскурин А.В.

Барнаул 2022

**Задание**

Интерполяционный многочлен Лагранжа

1. Придумайте функцию и составьте таблицу ее значений.

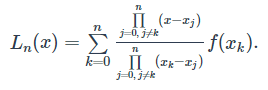
2. Напишите программу, которая интерполирует значения функции в произвольных точках по ее таблице при помощи интерполяционного многочлена Лагранжа.

3. Постройте сравнительные графики: исходной функции, множества точек интерполяции, многочлена Лагранжа по этим точкам.

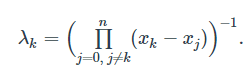
4. Исследуйте зависимость уклонения интерполяционного многочлена от количества и расположения узлов интерполяции.

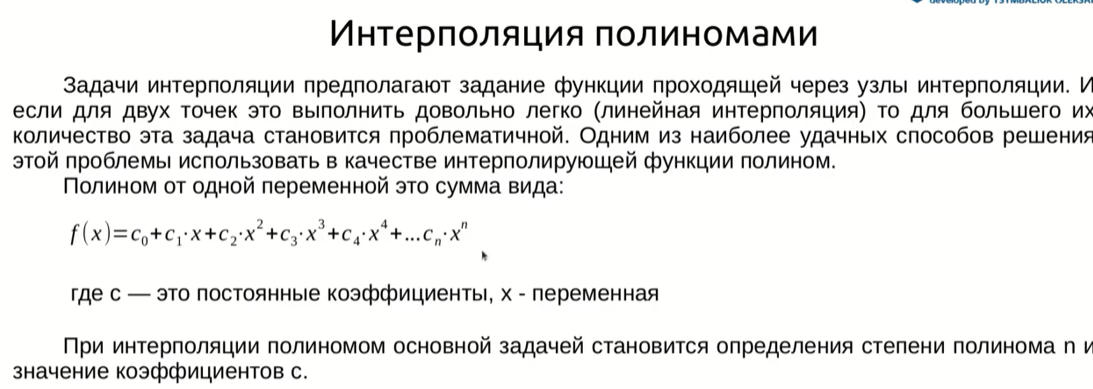
5. Повторите пункты 3,4 для функций: y=x^2 , y=|x| , y=e −x^2 .

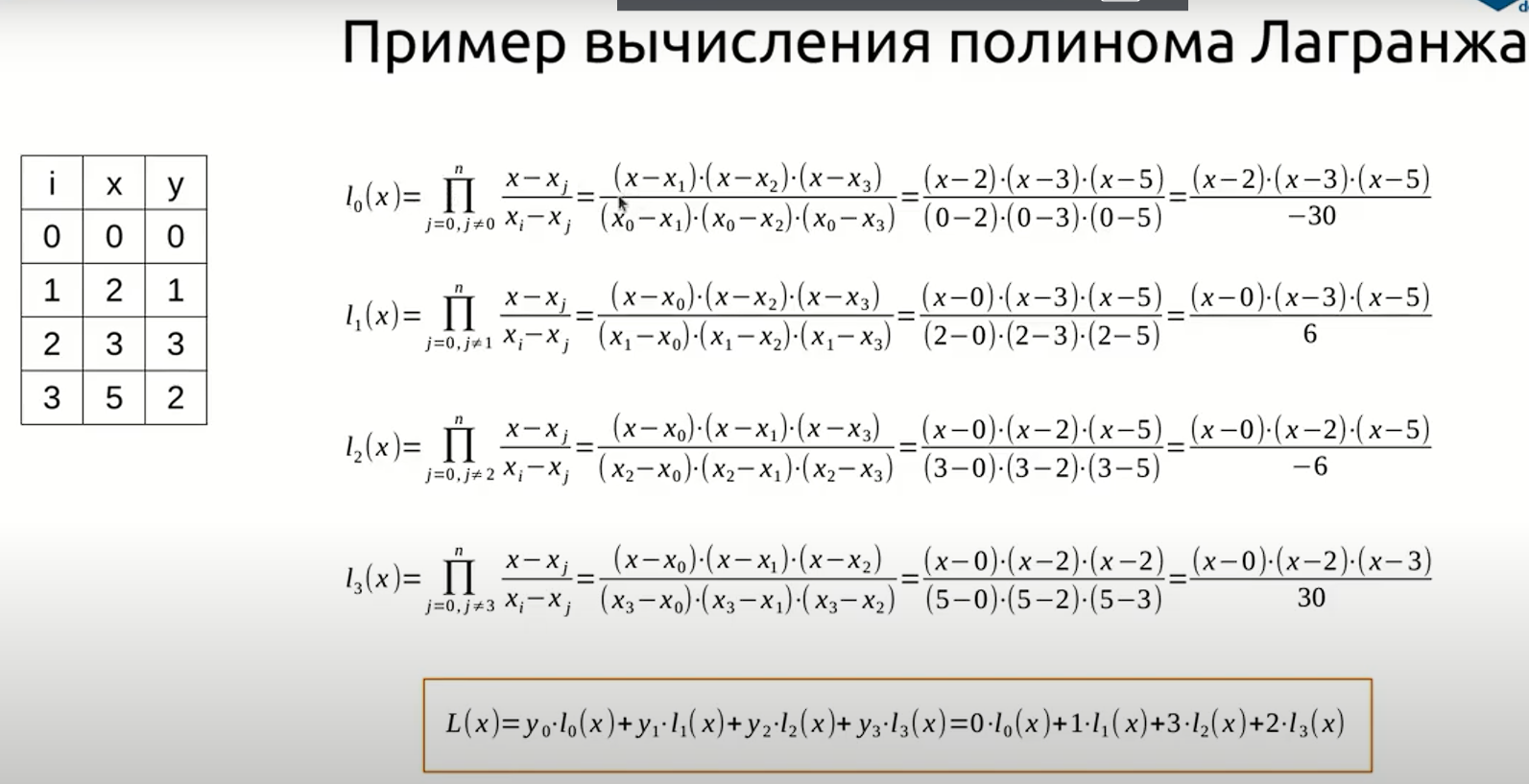










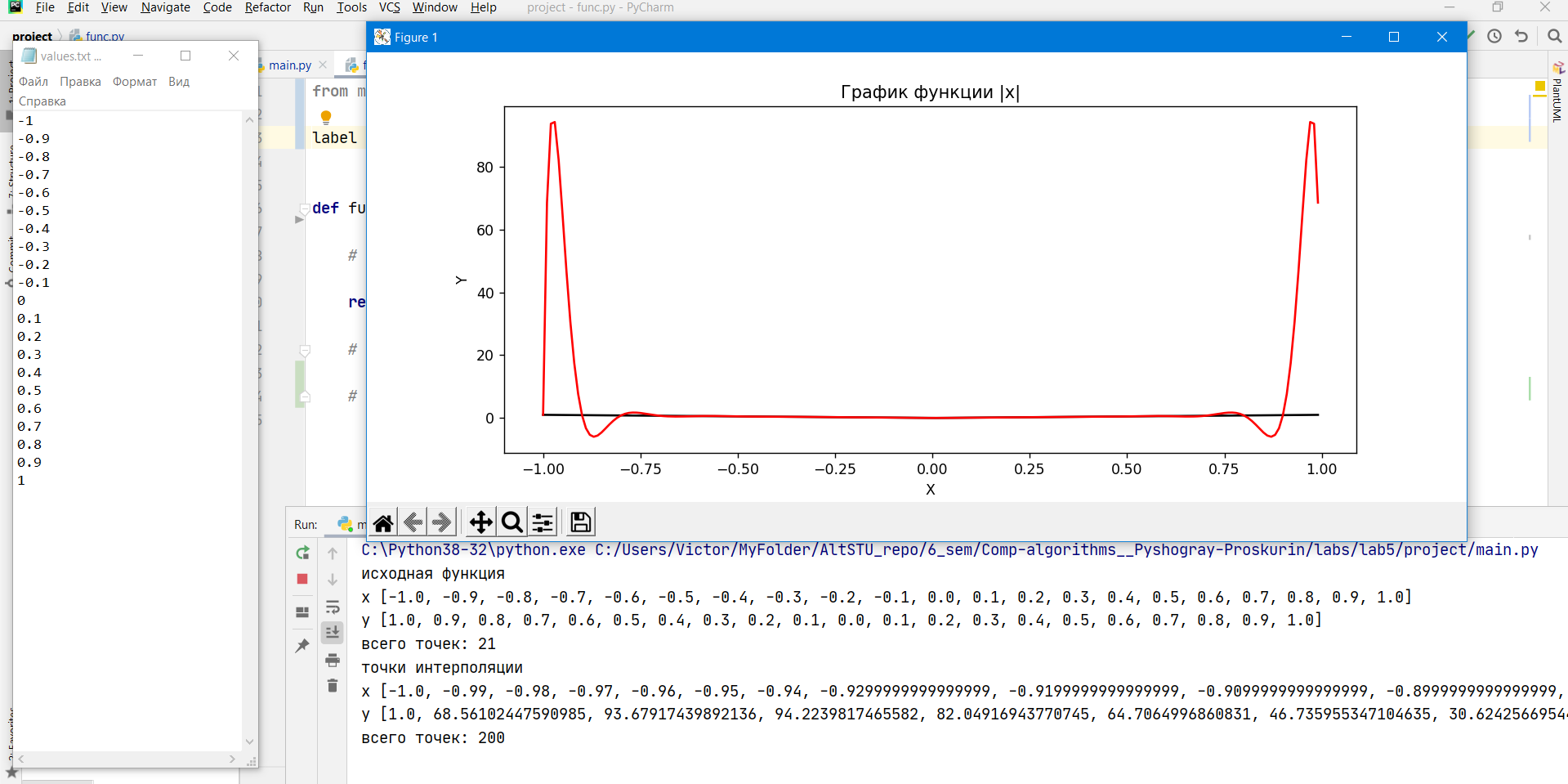


 среди значений    нет одинаковых. Точки  называют узлами интерполяции.

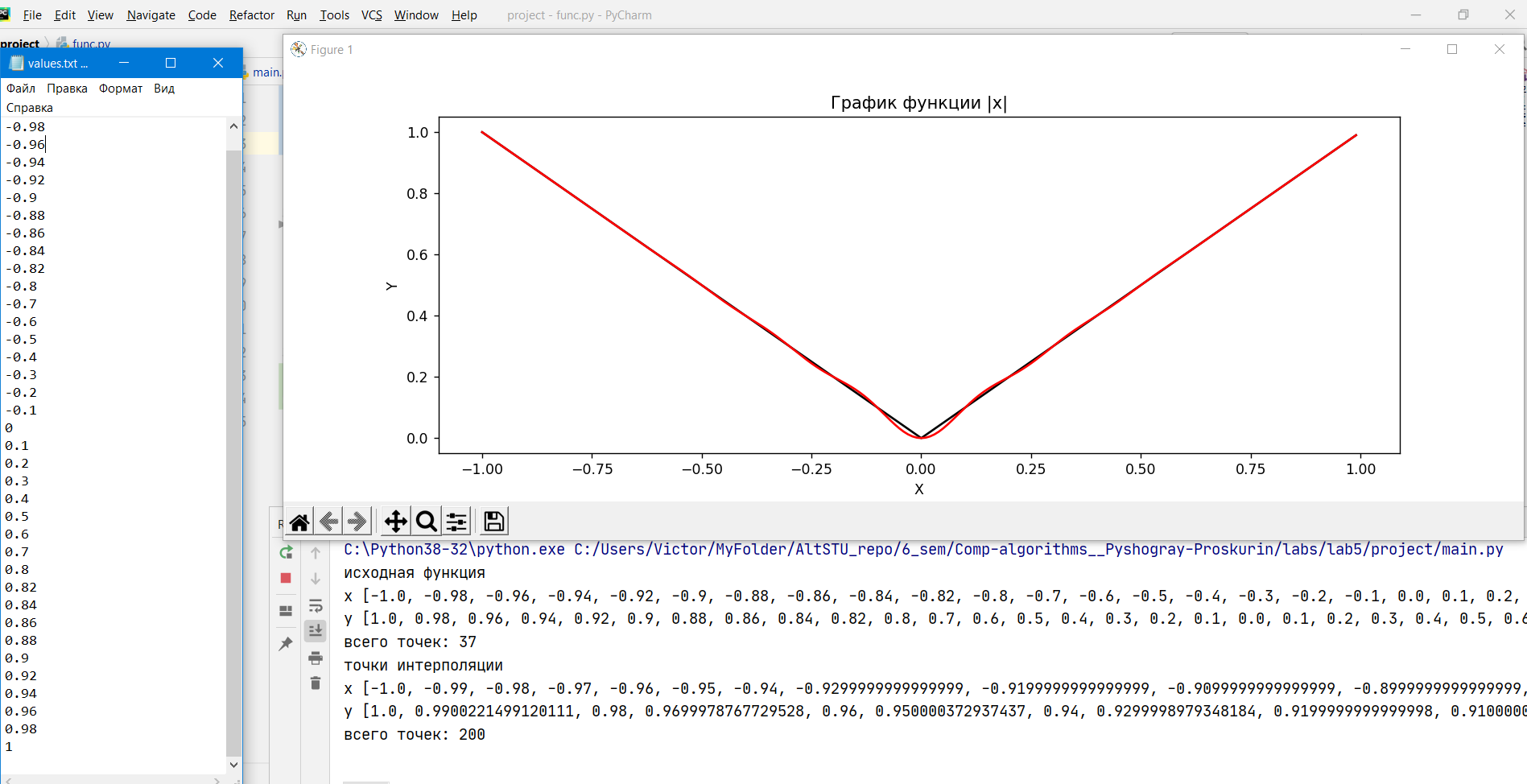
Если  ищется только на отрезке  – то это задача **интерполяции**, а если за пределами первоначального отрезка, то это задача **экстраполяции**.

**Тесты**

**1) y = |x|**

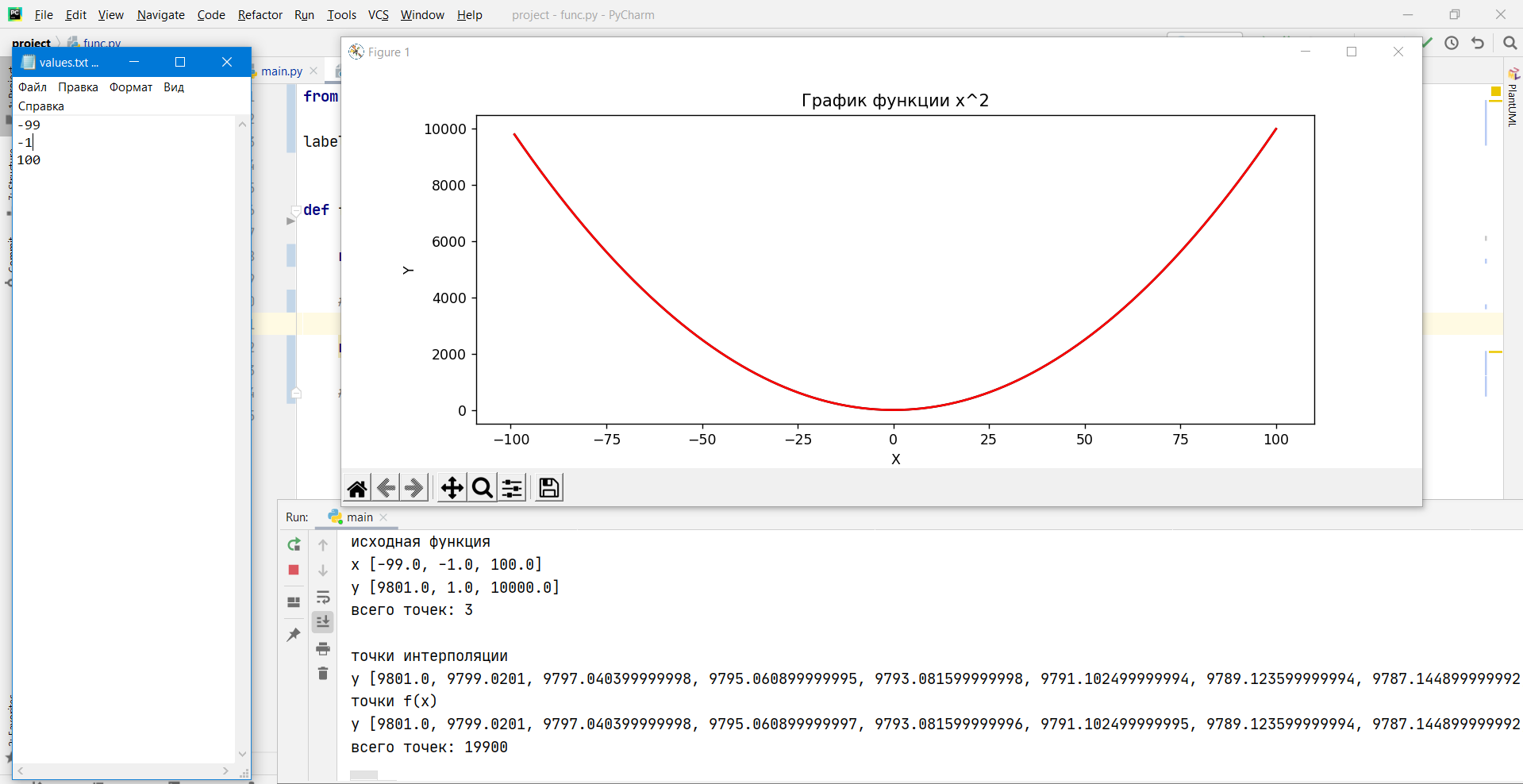


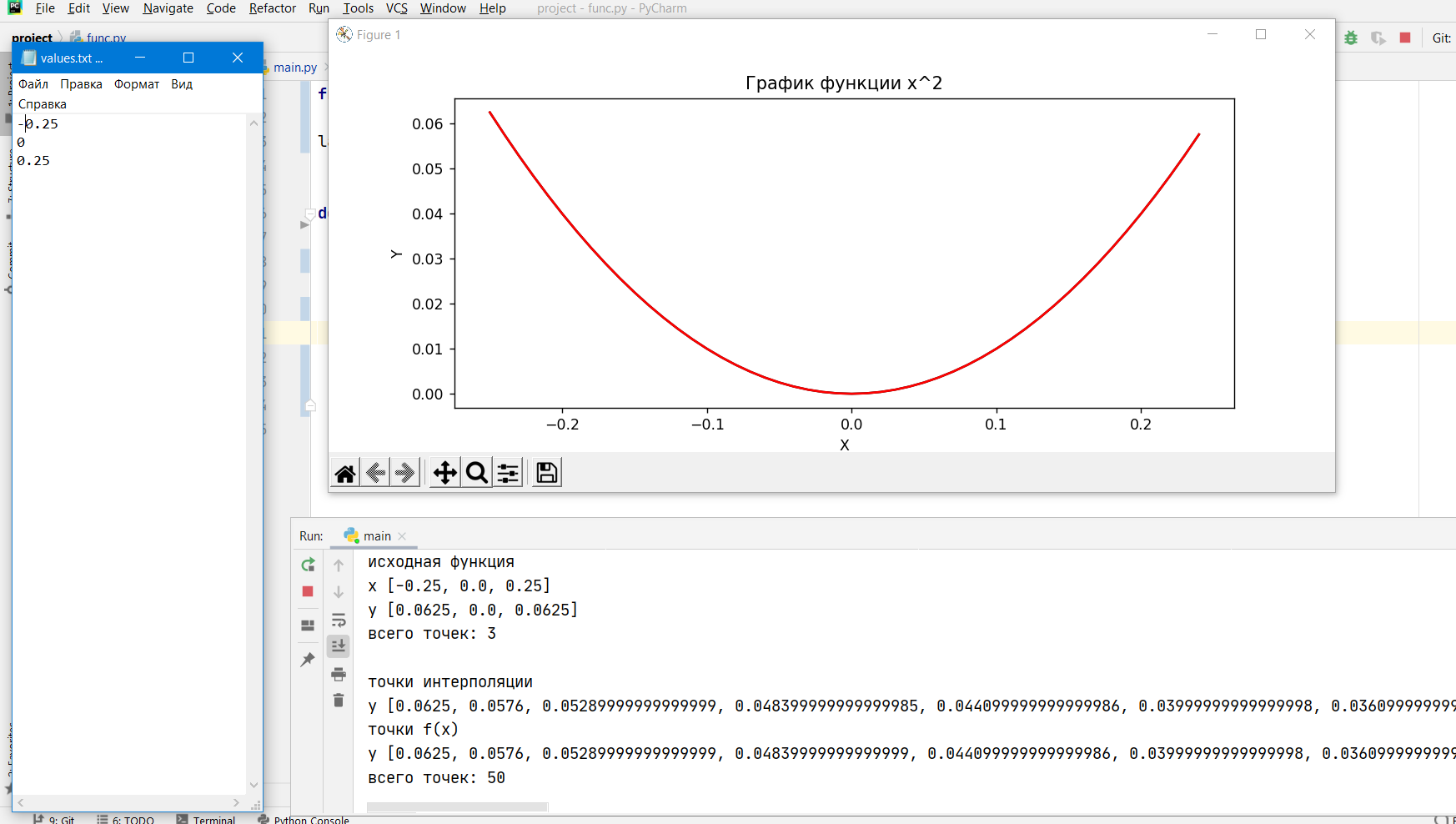
уменьшим расстояние между крайними узлами:



Функция модуля не является непрерывно дифференцируемой, поэтому не подходит для интерполирования. Хотя погрешность интерполирования может существенно снижаться с ростом узлов интерполяции, точность интерполирования остается невысокой.

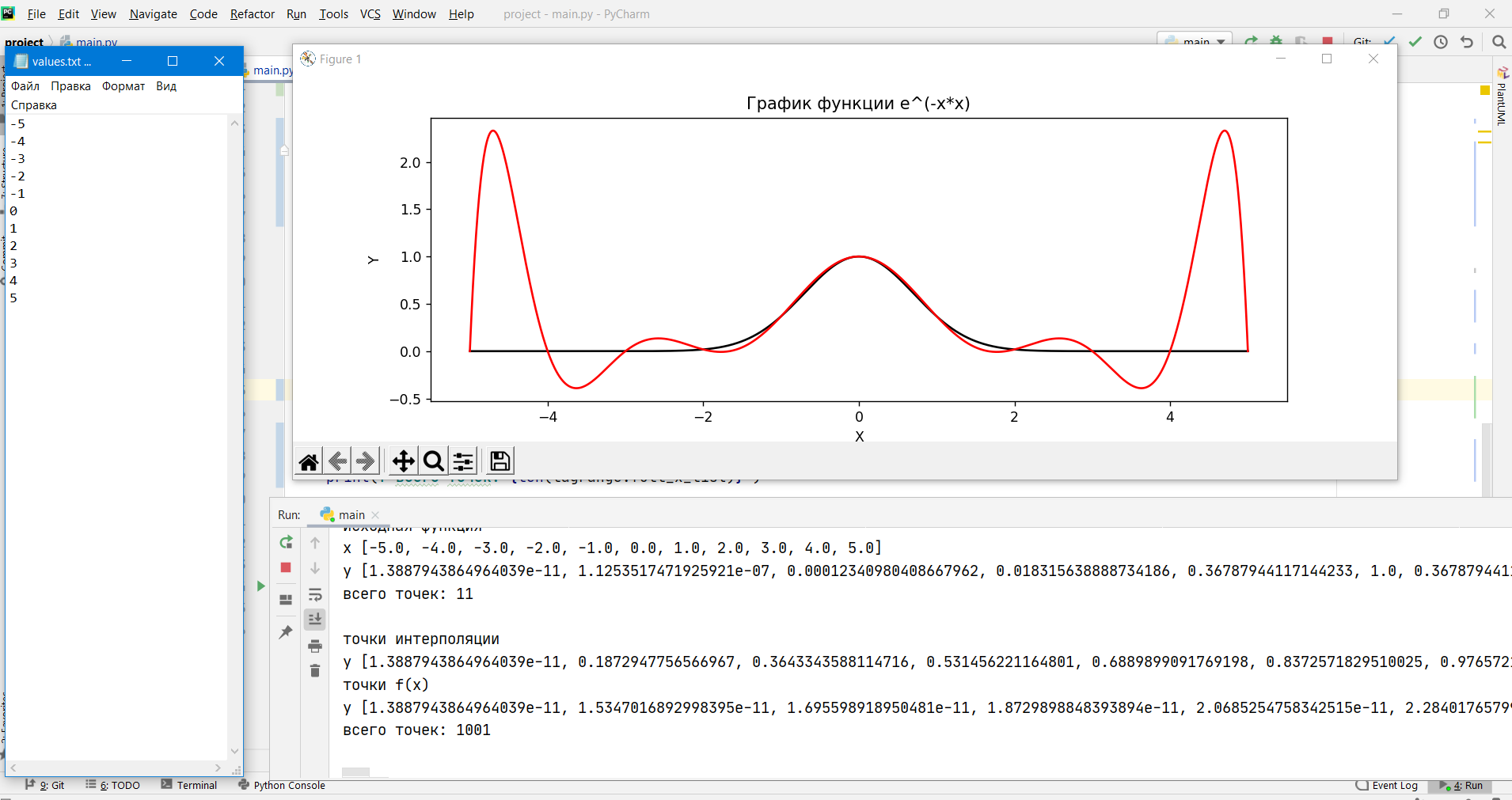
**2) y = x^2**

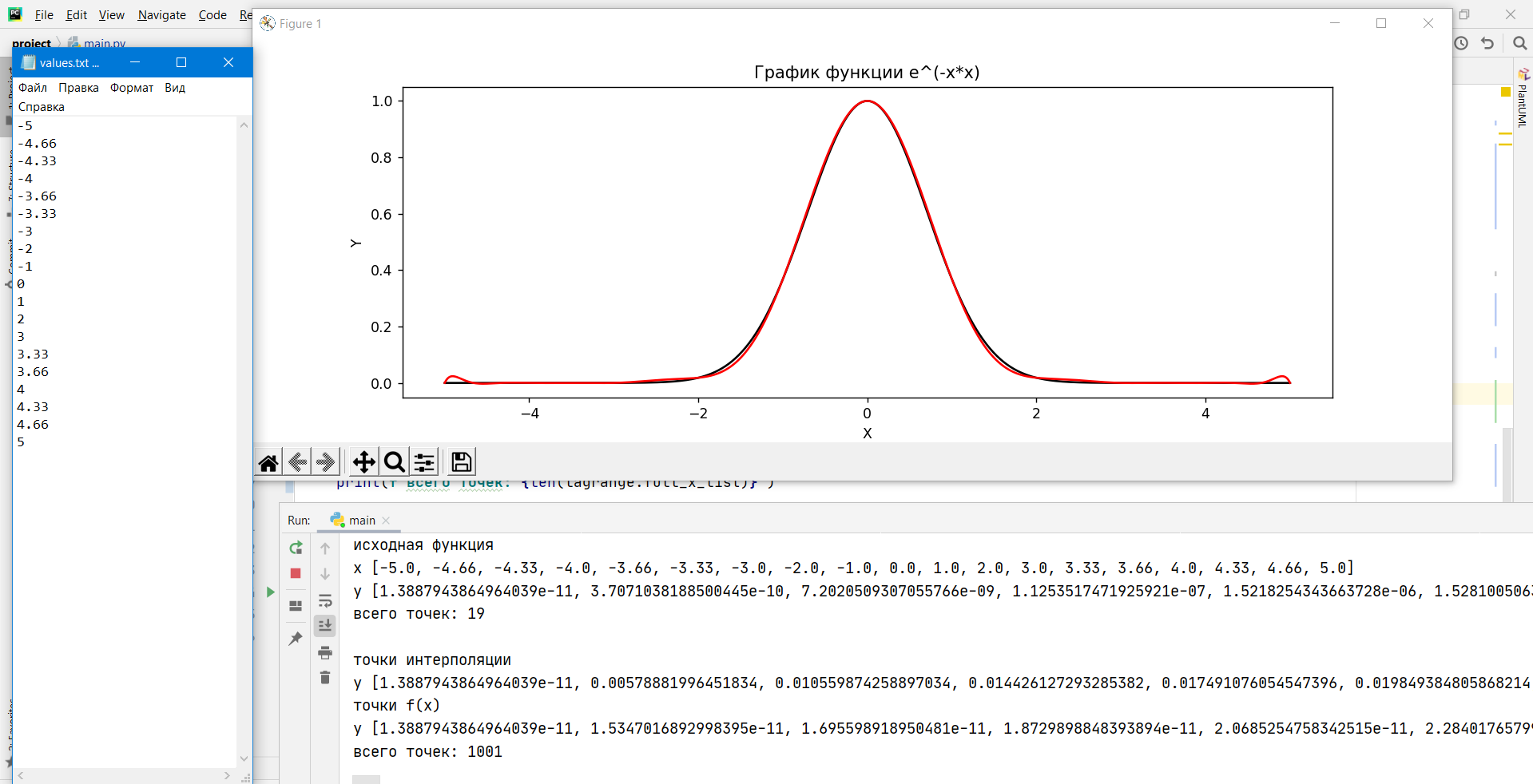




бесконечно непрерывно дифференцируемая (гладкая) функция хорошо интерполируется

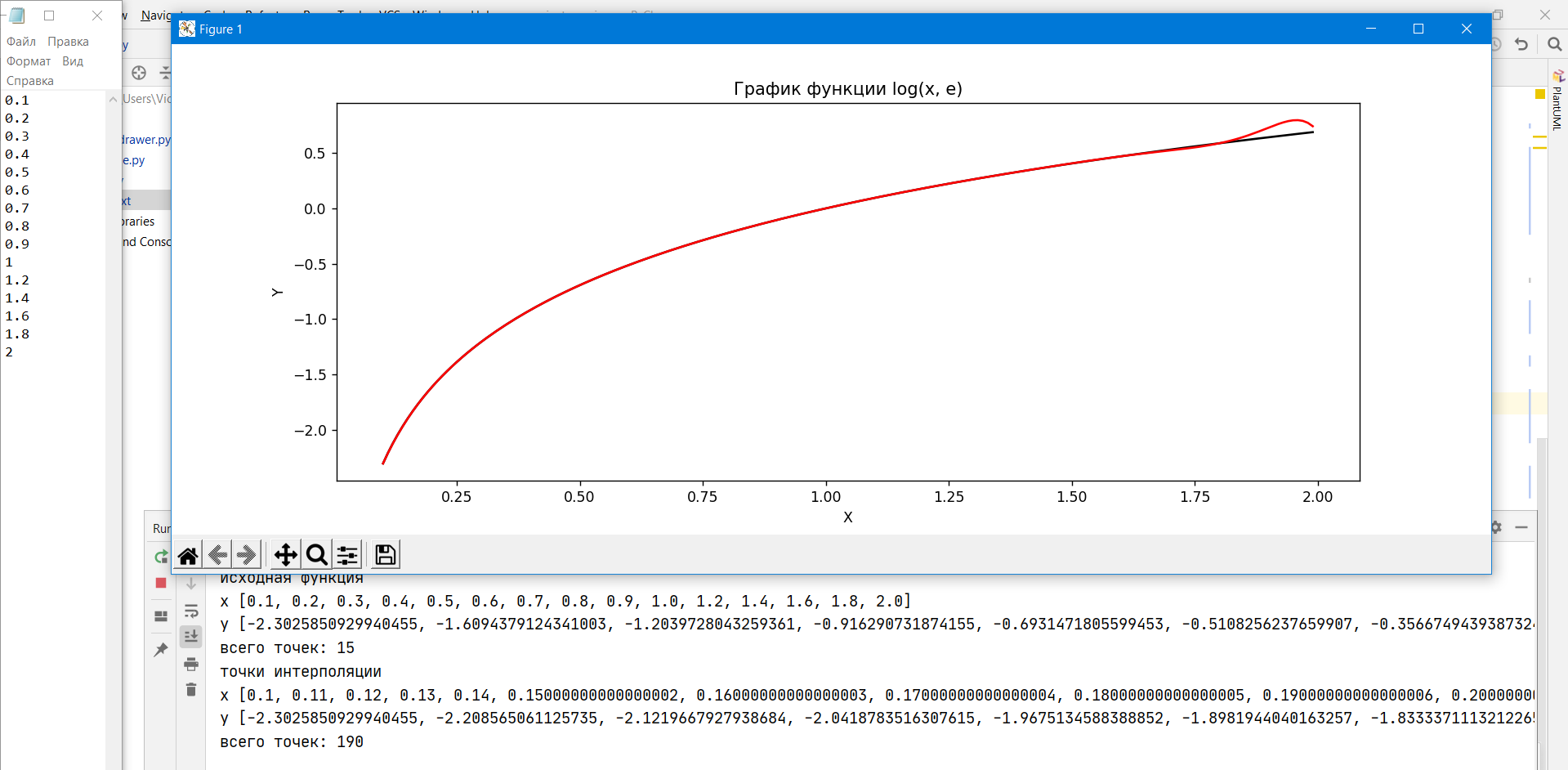
**3) y = e^(-x\*x)**

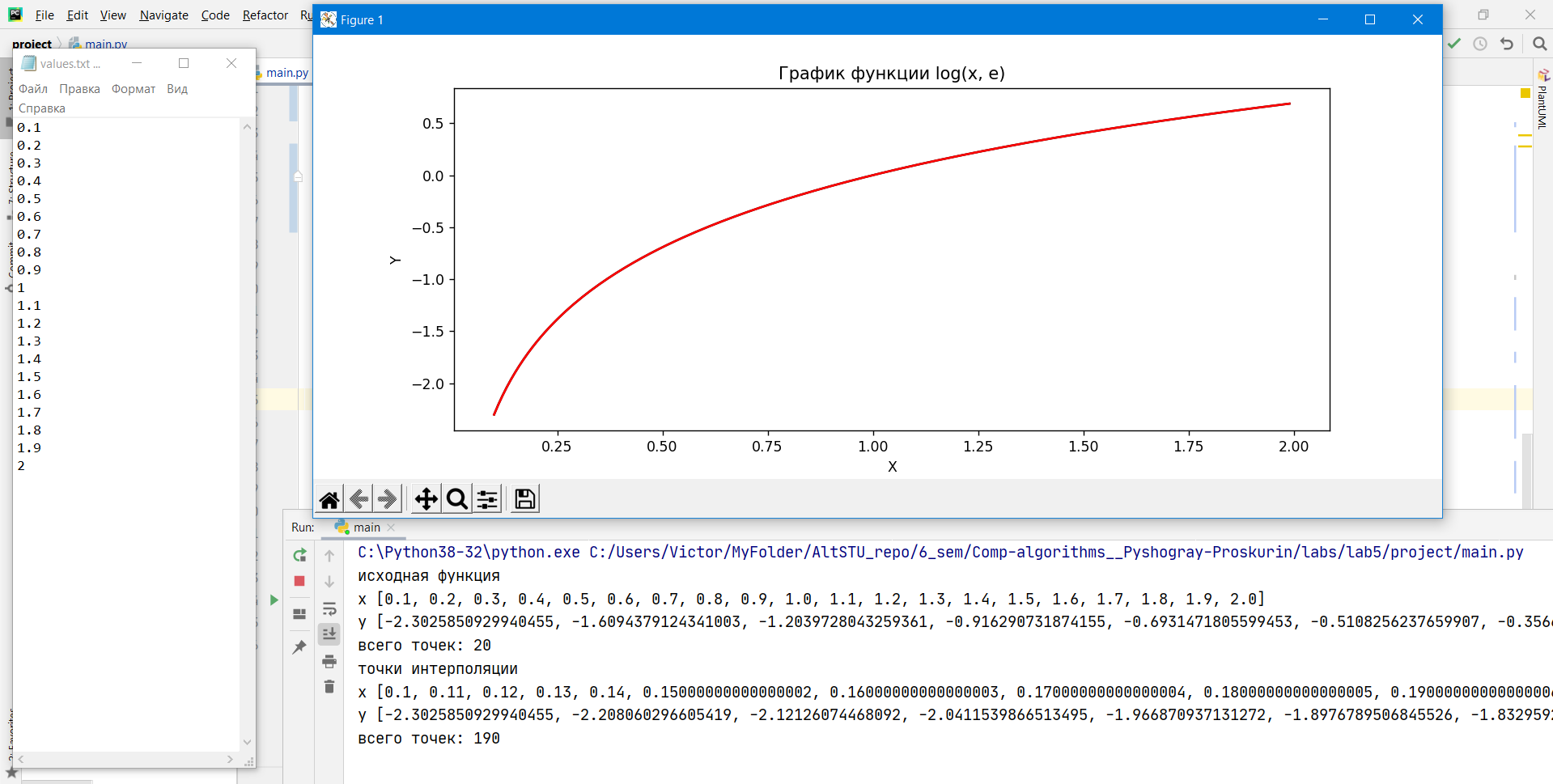


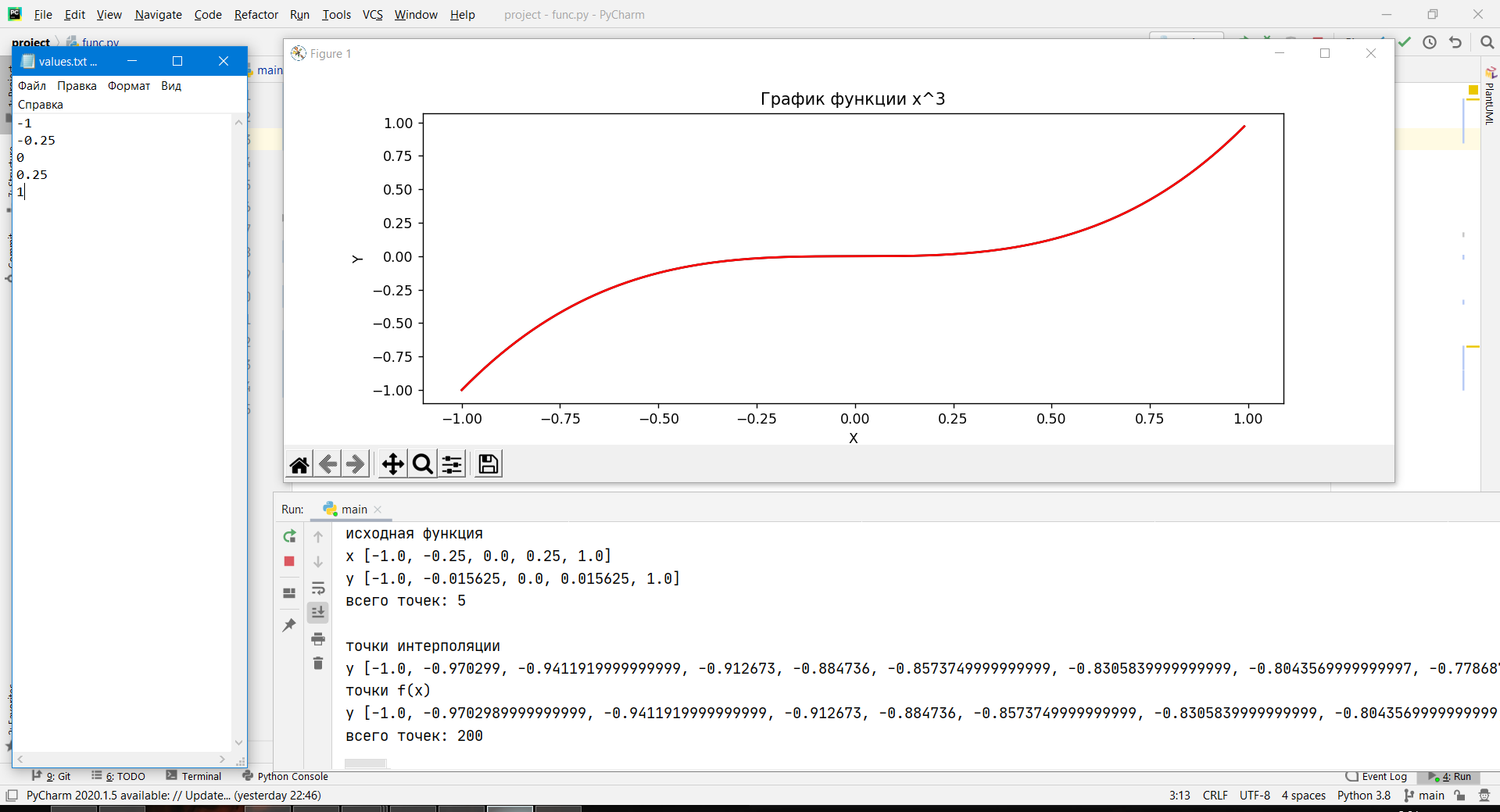


Для того, чтобы получить наиболее точное значение функции с помощью интерполяционного многочлена, нужно выбирать отрезок так, чтобы искомое значение располагалось ближе к середине.

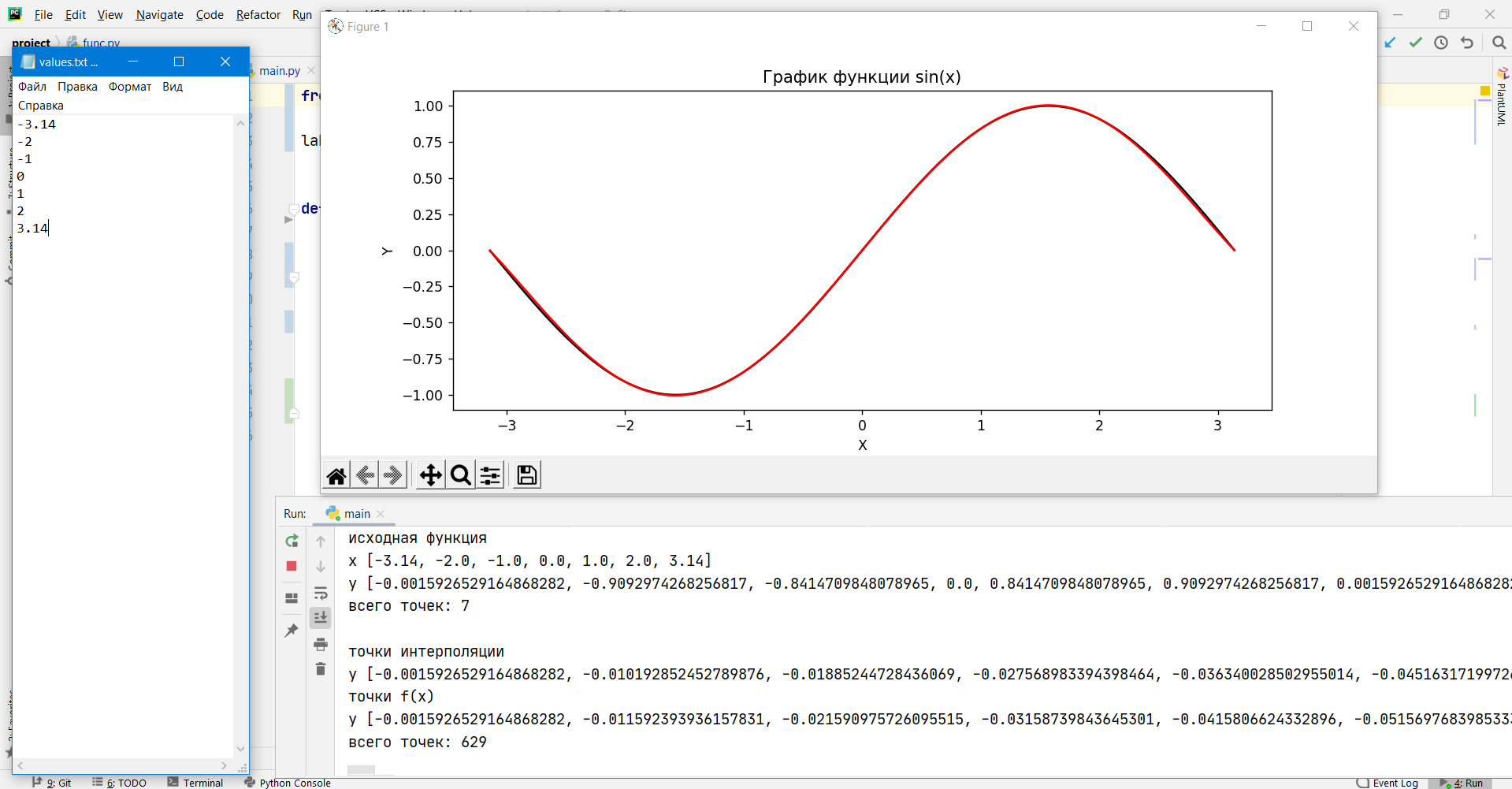
**4)**

Увеличим количество узлов интерполяции для правой части значений:



**5)**

**6)**



**Вывод:**

Как видно из графиков, что отклонение уменьшается с увеличением количества узлов интерполяции, а также большое значение имеет расположение этих узлов, чем ближе друг к другу эти узлы, тем точнее будут значения многочлена на этом интервале. Кроме того, большое значение имеет дифференцируемость функции, если функция не дифференцируема, то точность интерполирования будет низкой.

**Код:**

func.py -----------------------------------------------------------------------------------------------------------

**from** math **import** e, pow, log, sin  
  
label = **'sin(x)'** *# 'log(x,e)' #'|x|' # 'e^(-x\*x)'*  
  
  
**def** func(x):  
 **return** sin(x)  
 *# return x\*x\*x*  
 *# return abs(x)*  
 *# return pow(e, -x\*x)*  
 *# return log(x, e)*

lagrange.py ----------------------------------------------------------------------------------------------------

**class** Lagrange:  
 *""" интерполирует значения функции в произвольных точках по ее таблице при помощи интерполяционного многочлена*  
 *Лагранжа. """*  
  
**def** \_\_init\_\_(self):  
 self.x\_list = [] *# для использования в формуле Лагранжа*  
self.y\_list = []  
 self.full\_x\_list = [] *# значения больше, чем в x\_list (их график выводим)*  
self.full\_y\_list = []  
 self.computed\_y\_list = [] *# вычисленные по формуле Лагранжа (их график выводим)*  
  
**def** create\_basic\_polynomial(self, i):  
 **def** basic\_polynomial(x):  
 divider = 1  
 result = 1  
 **for** j **in** range(len(self.x\_list)):  
 **if** j != i:  
 result \*= (x - self.x\_list[j])  
 divider \*= (self.x\_list[i] - self.x\_list[j])  
 **return** result / divider  
  
 **return** basic\_polynomial  
  
 **def** create\_Lagrange\_polynomial(self):  
 basic\_polynomials = []  
 **for** i **in** range(len(self.x\_list)):  
 basic\_polynomials.append(self.create\_basic\_polynomial(i))  
  
 **def** lagrange\_polynomial(x):  
 result = 0  
 **for** i **in** range(len(self.y\_list)):  
 result += self.y\_list[i] \* basic\_polynomials[i](x)  
 **return** result  
  
 **return** lagrange\_polynomial

graph\_drawer.py ----------------------------------------------------------------------------------------------

**import** matplotlib.pyplot **as** plt  
  
**def** draw\_graphs(x\_list, y\_list, lagrange\_y\_list, label):  
 fig = plt.figure()  
 ax = fig.add\_subplot(111)  
 ax.set(title=**'График функции '** + label,  
 xlabel=**'X'**,  
 ylabel=**'Y'**)  
 ax.plot(x\_list, y\_list, color=**'black'**)  
 ax.plot(x\_list, lagrange\_y\_list, color=**'red'**)  
 plt.show()

main.py -------------------------------------------------------------------------------------------------------------

**from** lagrange **import** Lagrange  
**from** graph\_drawer **import** draw\_graphs  
**from** func **import** func, label  
**from** random **import** randint  
**import** math  
  
**def** frange(start, stop, step):  
 *"""range для float"""*  
i = start  
 **while** i < stop:  
 **yield** i  
 i += step  
  
**def** main():  
 lagrange = Lagrange()  
  
 **with** open(**"values.txt"**, **"r"**) **as** f:  
 **for** x **in** f:  
 lagrange.x\_list.append(float(x))  
 lagrange.y\_list.append(func(float(x)))  
  
 min\_x\_list = min(lagrange.x\_list)  
 max\_x\_list = max(lagrange.x\_list)  
 **for** x **in** frange(min\_x\_list, max\_x\_list, 0.01):  
 lagrange.full\_x\_list.append(x)  
 lagrange.full\_y\_list.append(func(x))  
 f = lagrange.create\_Lagrange\_polynomial()  
 lagrange.computed\_y\_list.append(f(x))  
  
 draw\_graphs(lagrange.full\_x\_list, lagrange.full\_y\_list, lagrange.computed\_y\_list, label)  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()