

МИНОБРНАУКИ РОССИИ САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
КАФЕДРА САПР

ОТЧЁТЫ ПО ЛАБОРАТОРНЫМ РАБОТАМ
по дисциплине «Информатика»

Выполнил студент:

Коняев Александр Евгеньевич
группа: 2302

Проверил:

ассистент каф. САПР,
Копец Екатерина Евгеньевна

Санкт-Петербург
2022

Содержание

Лабораторная работа №1	4
Основная часть.....	4
Теоретическая часть.....	4
Полученные результаты и их анализ.....	4
Выводы	5
Лабораторная работа №2	6
Основная часть.....	6
Теоретическая часть.....	6
Полученные результаты и их анализ.....	7
Выводы	18
Лабораторная работа №3	19
Основная часть.....	19
Теоретическая часть.....	19
Полученные результаты и их анализ.....	19
Выводы	22
Лабораторная работа №4	23
Основная часть.....	23
Теоретическая часть.....	23
Полученные результаты и их анализ.....	23
Выводы	32
Лабораторная работа №5	33
Основная часть.....	33
Теоретическая часть.....	33
Полученные результаты и их анализ.....	33
Выводы	38
Лабораторная работа №6	39
Основная часть.....	39
Теоретическая часть.....	39
Полученные результаты и их анализ.....	39
Выводы	44
Лабораторная работа №7	45
Основная часть.....	45
Теоретическая часть.....	45
Полученные результаты и их анализ.....	45
Выводы	48

Лабораторная работа №8	49
Основная часть.....	49
Теоретическая часть.....	49
Полученные результаты и их анализ.....	50
Выводы	58
Лабораторная работа №9	59
Основная часть.....	59
Теоретическая часть.....	59
Полученные результаты и их анализ.....	60
Выводы	71

1 Лабораторная работа №1

1.1 Основная часть

1.1.1 Теоретическая часть

В L^AT_EX для вставки изображения используется шаблон:

Листинг 1: Шаблон для вставки изображения

```
1 \begin{figure}[ht!]
2   \centering
3   \includegraphics[scale=]{}
4   \caption{}
5   \label{fig:1}
6 \end{figure}
```

А для вставки простой таблицы:

Листинг 2: Шаблон для вставки таблицы

```
1 \begin{center}
2   \begin{table}[h!]
3     \centering
4     \begin{tabular}{ c c c }
5       cell1 & cell2 & cell3 \\
6       cell4 & cell5 & cell6 \\
7       cell7 & cell8 & cell9 \\
8     \end{tabular}
9     \caption{}
10    \label{tbl:1}
11  \end{table}
12 \end{center}
```

1.1.2 Полученные результаты и их анализ

$$\begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix}$$

Таблица 1: Главная и побочная диагональ матрицы из цифр 1

Номер	альфа	альфа ср.	дельта альфа
1	20		
2	22		
3	23	22.8	0.0534072
4	23		
5	26		

Таблица 2: Расчёт значений для выборки из 5 элементов

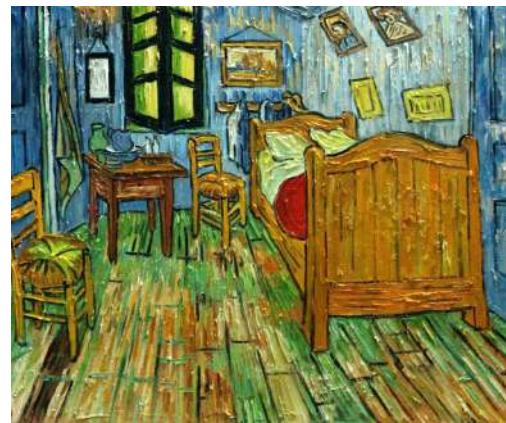


Рис. 1: Винсент Ван Гог "Спальня в Арле"



Рис. 2: Картина нейросети Midjourney "Театр пространственной оперы"

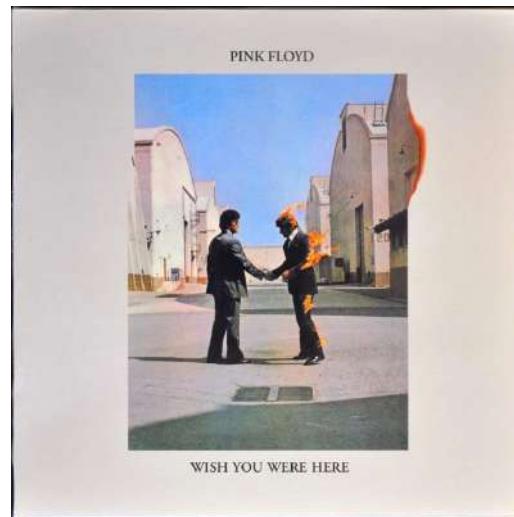


Рис. 3: Pink Floyd обложка альбома "Wish You Were Here"

1.2 Выводы

Я научился вставлять изображения и таблицы в проект \LaTeX Overleaf. Было интересно поработать с новой для меня системой. Мне кажется, что у \LaTeX есть множество преимуществ по сравнению с тем же Microsoft Word, я был бы рад освоить эту систему и оформлять документы в ней.

1 Лабораторная работа №2

1.1 Основная часть

1.1.1 Теоретическая часть

Для работы с графиками в этой работе используется математическая библиотека SumPy. Сначала нам необходимо подключить эту библиотеку и модуль plot:

```
1 from sympy import *
2 from sympy.plotting import plot
```

А также написать строку кода, которая позволит выводить красивые формулы:

```
1 x=Symbol('x')
```

Для вывода графиков, поиска нулей функции и промежутков знакопостоянства используются следующие функции:

```
1 plot(f)
2 solve(f)
3 solve_univariate_inequality(f>0,x)
```

Дополнительно используются функции для вычисления экспоненты, натурального логарифма и квадратного корня из числа:

```
1 exp(x)
2 ln(x)
3 sqrt(x)
```

1.1.2 Полученные результаты и их анализ

Понятие функции. Построение графиков функций с помощью SymPy.

(1) Импортируйте библиотеку и модуль `plot`, вызовите команду для красивой отрисовки формул. Затем, создав переменную, создайте функцию $f(x) = -3x + 5$ и постройте ее график. Найдите значения этой функции для значений $x = \frac{5}{3}$ и $x = 2$.

Проделайте тоже самое для следующих функций и их точек, с некоторыми из них мы встретимся в будущем и разберем их более подробно, но уже сейчас можно посмотреть на их графики:

- (2) $f(x) = 3x^2 + 3x - 6$. Найдите значения функции при $x = 0.5$ и $x = 7$.
- (3) $f(x) = 2x^4 + x^3 - 6x^2 + 7x + 1$. Найдите значения функции при $x = 3$ и $x = -5$.
- (4) $f(x) = (x^2)^x$. Найдите значения функции при $x = 0$, $x = 10$.

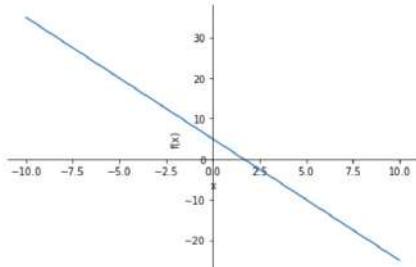
```
Ввод [1]: from sympy import *
from sympy.plotting import plot

Ввод [2]: x=Symbol('x')

Ввод [3]: f=-3*x+5

Ввод [4]: f
Out[4]: 5 - 3*x

Ввод [7]: plot(f)
```



```
Ввод [7]: plot(f)
Out[7]: <sympy.plotting.plot.Plot at 0x1c2f6288340>

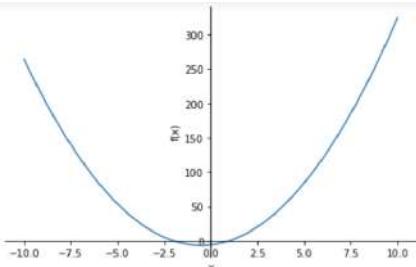
Ввод [5]: f.subs(x,(5/3))
Out[5]: 0
```

```
Ввод [5]: f.subs(x,(5/3))
Out[5]: 0

Ввод [6]: f.subs(x,2)
Out[6]: -1

Ввод [9]: f=3*x**2+3*x-6

Ввод [10]: plot(f)
```



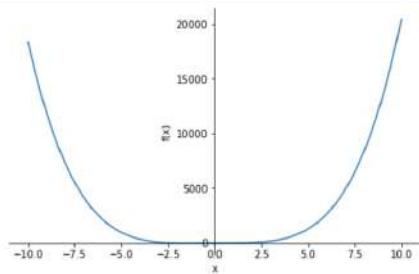
```
Out[10]: <sympy.plotting.plot.Plot at 0x1c2faa44ac0>

Ввод [11]: f.subs(x,0.5)
Out[11]: -3.75

Ввод [12]: f.subs(x,7)
Out[12]: 162
```

Ввод [13]: $f=2*x^{**4}+x^{**3}-6*x^{**2}+7*x+1$

Ввод [14]: plot(f)



Out[14]: <sympy.plotting.plot.Plot at 0x1c2fab58520>

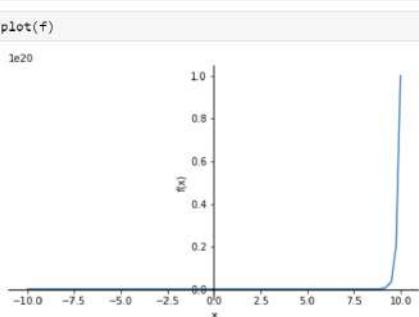
Ввод [15]: f.subs(x, 3)

Out[15]: 157

Ввод [16]: f.subs(x, -5)

Out[16]: 941

Ввод [17]: $f(x) = x^{x^2}$



Ввод [19]: `f.subs(x, e)`

Out[19]: 1

Ввод [20]: f.subs(x, 10)

Out[20]: 10000000000000000000000000

Ввод []:

Исследование параболы с помощью SymPy
Первая часть

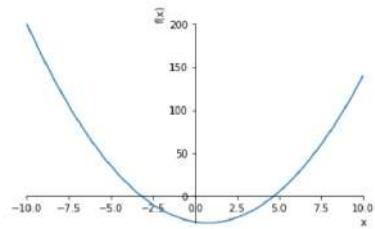


Рис. 1: $a>0$, $b<0$, $c<0$

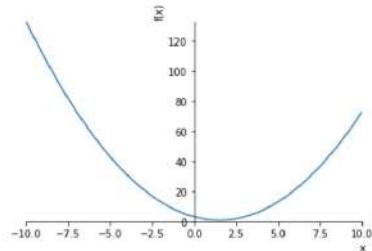


Рис. 2: $a>0$, $b<0$, $c>0$

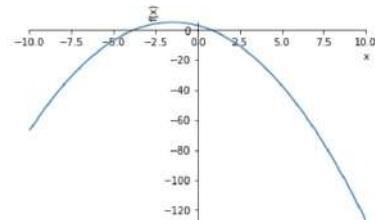


Рис. 3: $a<0$, $b>0$, $c>0$

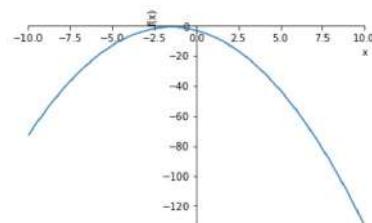


Рис. 4: $a<0$, $b>0$, $c<0$

Вторая часть

Импортируйте библиотеку и модуль `plot`, вызовите команду для красивой отрисовки формул. Постройте график каждой функции, проверьте, является ли она четной, нечетной или ни той, ни другой по виду графика и с помощью подстановки значений. Проверьте свойства коэффициентов, которые были упомянуты в уроке про элементарные функции, покрутите линейную функцию, варьируя угловой коэффициент; и подвигайте графики линейной функции и параболу вверх и вниз, меняя свободный член. Графики показательной функции и корня также можно двигать вверх и вниз, добавляя к ним положительные и отрицательные числа, например $f(x) = e^x + 1$ – это график функции $f(x) = e^x$, сдвинутый вверх на единицу.

Для параболы проверьте свойство коэффициента при x^2 в квадрате. Будьте внимательны с промежутками для переменной (областью определения), не сломайте `sympy`:

(1) В качестве примера показательной функции используйте экспоненциальную функцию e^x . В `sympy` ее можно объявить так:

```
Ввод [ ]: f = exp(x)
```

(2) В качестве примера логарифмической функции используйте натуральный логарифм. В `sympy` его можно объявить так:

```
Ввод [ ]: f = ln(x)
```

(3) В качестве примера одной квадратичной функции используйте следующую функцию: $f(x) = 2x^2 + 3x - 5$.

(4) В качестве примера другой квадратичной функции используйте следующую функцию: $f(x) = 5x^2 - 8$.

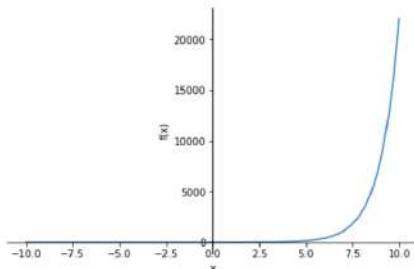
(5) В качестве функции корня используйте просто корень: $f(x) = \sqrt{x}$.

```
Ввод [1]: from sympy import *
from sympy.plotting import plot
```

```
Ввод [2]: x=Symbol('x')
```

```
Ввод [3]: f=exp(x)
```

```
Ввод [4]: plot(f)
```



```
Out[4]: <sympy.plotting.plot.Plot at 0x214d08202e0>
```

```
Ввод [7]: f.subs(x,1)
```

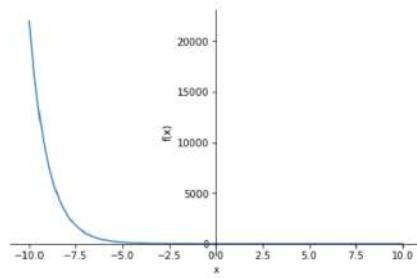
```
Out[7]: e
```

```
Ввод [8]: f.subs(x,-1)
```

```
Out[8]: e^-1
```

```
Ввод [9]: f=exp(-x)
```

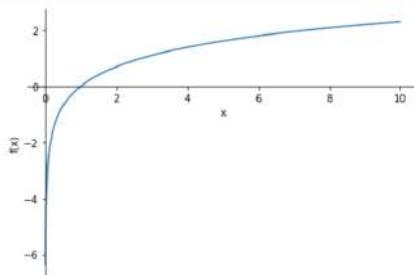
```
Ввод [11]: plot(f)
```



```
Out[11]: <sympy.plotting.plot.Plot at 0x214d0813f40>
```

```
Ввод [12]: f=ln(x)
```

```
Ввод [14]: plot(f)
```



```
Out[14]: <sympy.plotting.plot.Plot at 0x214d5e943d0>
```

```
Ввод [15]: f.subs(x,1)
```

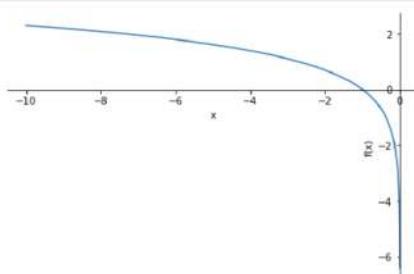
```
Out[15]: 0
```

```
Ввод [16]: f.subs(x,-1)
```

```
Out[16]: iπ
```

```
Ввод [17]: f=ln(-x)
```

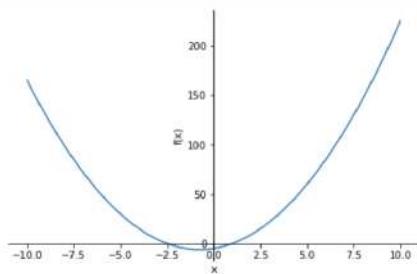
```
Ввод [18]: plot(f)
```



```
Out[18]: <sympy.plotting.plot.Plot at 0x214d5e70820>
```

```
Ввод [20]: f=2*x**2+3*x-5
```

```
Ввод [21]: plot(f)
```



```
Out[21]: <sympy.plotting.plot.Plot at 0x214d5ec08e0>
```

```
Ввод [23]: f.subs(x,4)
```

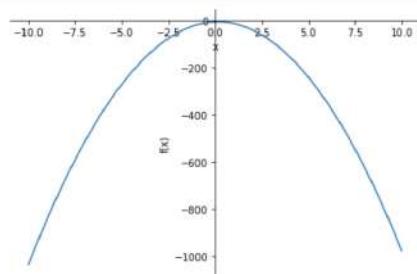
```
Out[23]: 39
```

```
Ввод [27]: f.subs(x,-5.5)
```

```
Out[27]: 39.0
```

```
Ввод [28]: f=-10*x**2+3*x-5
```

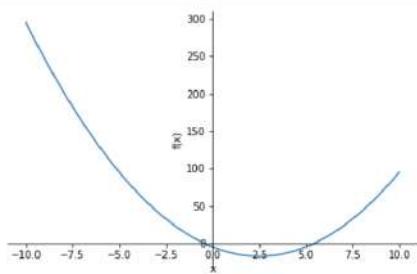
```
Ввод [29]: plot(f)
```



```
Out[29]: <sympy.plotting.plot.Plot at 0x214d0820760>
```

```
Ввод [30]: f=2*x**2-10*x-5
```

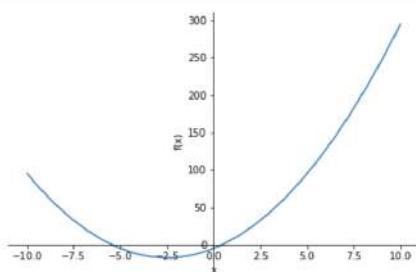
```
Ввод [31]: plot(f)
```



```
Out[31]: <sympy.plotting.plot.Plot at 0x214d08190a0>
```

```
Ввод [32]: f=2*x**2+18*x-5
```

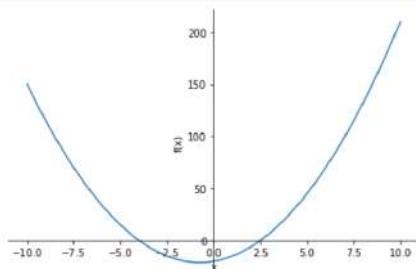
```
Ввод [33]: plot(f)
```



```
Out[33]: <sympy.plotting.plot.Plot at 0x214d63b2b80>
```

```
Ввод [34]: f=2*x**2+3*x-20
```

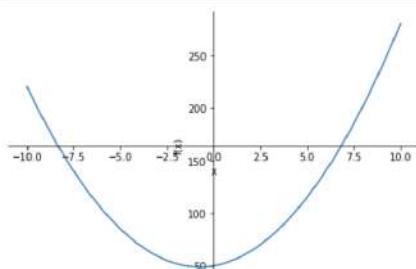
```
Ввод [35]: plot(f)
```



```
Out[35]: <sympy.plotting.plot.Plot at 0x214d6300430>
```

```
Ввод [36]: f=2*x**2+3*x+50
```

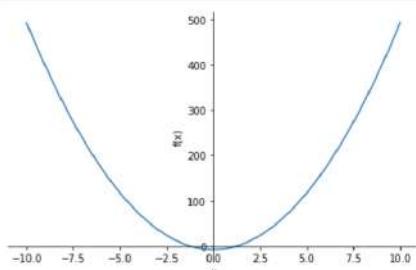
```
Ввод [37]: plot(f)
```



```
Out[37]: <sympy.plotting.plot.Plot at 0x214d080d4c0>
```

```
Ввод [38]: f=5*x**2-8
```

```
Ввод [39]: plot(f)
```



```
Out[39]: <sympy.plotting.plot.Plot at 0x214d4dadfa0>
```

```
Ввод [40]: f.subs(x,2)
```

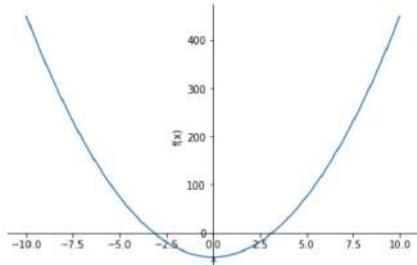
```
Out[40]: 12
```

```
Ввод [41]: f.subs(x,-2)
```

```
Out[41]: 12
```

```
Ввод [42]: f=5*x**2-50
```

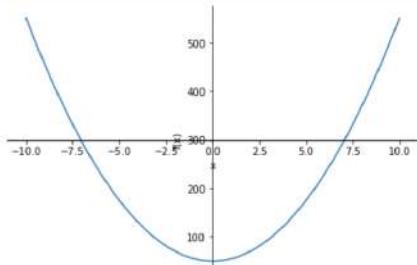
```
Ввод [43]: plot(f)
```



```
Out[43]: <sympy.plotting.plot.Plot at 0x214d596fc0>
```

```
Ввод [46]: f=5*x**2+50
```

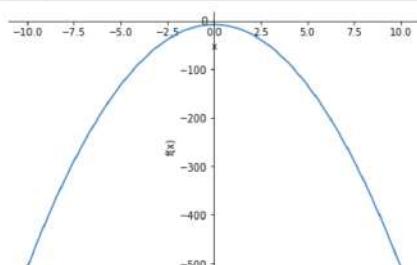
```
Ввод [47]: plot(f)
```



```
Out[47]: <sympy.plotting.plot.Plot at 0x214d65f1970>
```

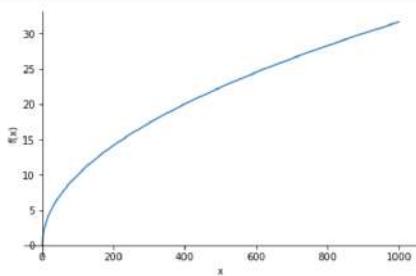
```
Ввод [52]: f=-5*x**2-8
```

```
Ввод [53]: plot(f)
```



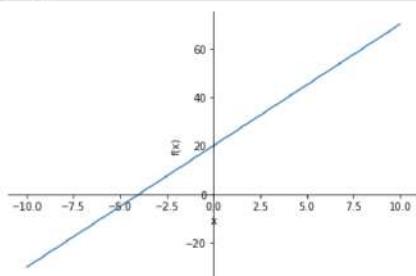
```
Out[53]: <sympy.plotting.plot.Plot at 0x214d64e5570>
```

```
Ввод [54]: f=x**0.5  
Ввод [56]: plot(f, (x, 0, 1000))
```



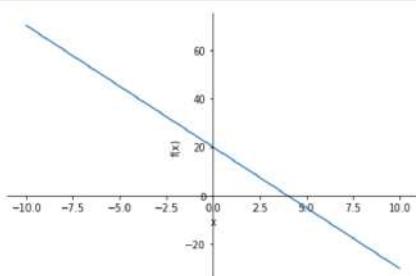
```
Out[56]: <sympy.plotting.plot.Plot at 0x214d76f6c40>
```

```
Ввод [57]: f=5*x+20  
Ввод [58]: plot(f)
```



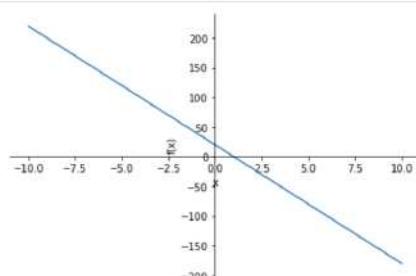
```
Out[58]: <sympy.plotting.plot.Plot at 0x214d76380a0>
```

```
Ввод [59]: f=-5*x+20  
Ввод [60]: plot(f)
```

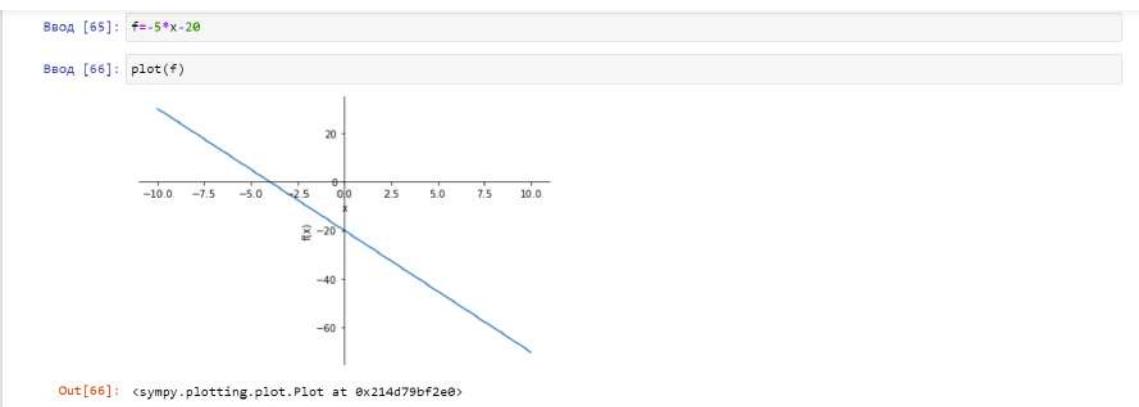
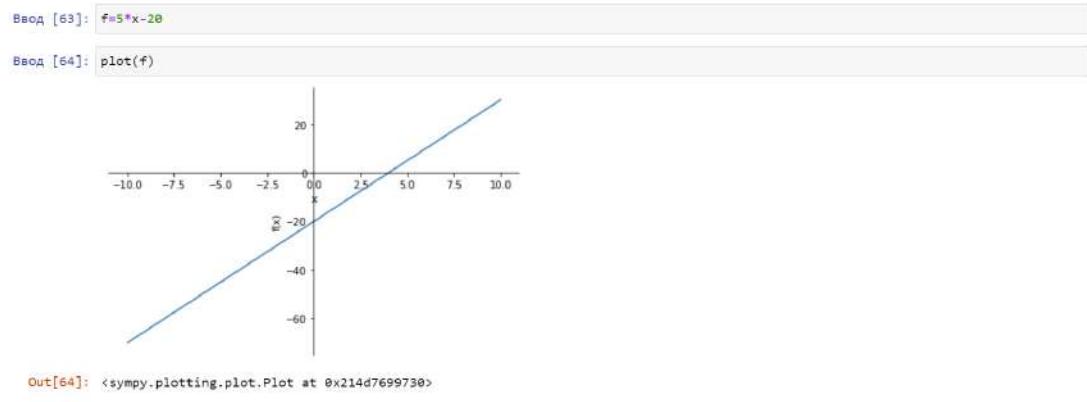


```
Out[60]: <sympy.plotting.plot.Plot at 0x214d783a160>
```

```
Ввод [61]: f=-20*x+20  
Ввод [62]: plot(f)
```



```
Out[62]: <sympy.plotting.plot.Plot at 0x214d77dcbb20>
```



Дополнительный функционал SymPy для исследования функций

В качестве практики найдите нули и промежутки знакопостоянства функций, с которыми вы работали в рамках предыдущей практики:

1. $f(x) = e^x$
2. $f(x) = \ln(x)$
3. $f(x) = 2x^2 + 3x - 5$
4. $f(x) = \sqrt{x}$

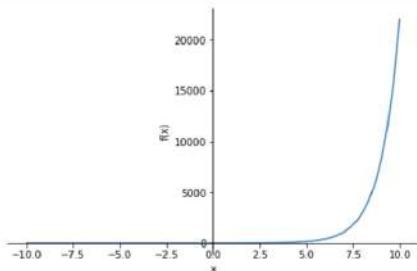
Проверьте, согласуются ли найденные вами значения с поведением графика.

```
Ввод [1]: from sympy import *
from sympy.plotting import plot

Ввод [2]: x=Symbol('x')
```

```
Ввод [3]: f=exp(x)
```

```
Ввод [5]: plot(f)
```



```
Out[5]: <sympy.plotting.plot.Plot at 0x292b254a0a0>
```

```
Ввод [4]: solve(f)
```

```
Out[4]: []
```

```
Ввод [6]: solve_univariate_inequality(f>0,x)
```

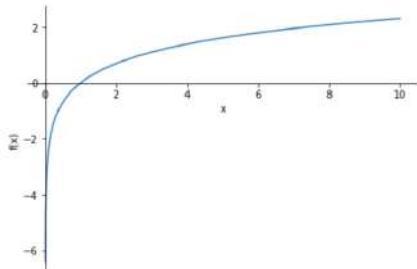
```
Out[6]: -∞ < x ∧ x < ∞
```

```
Ввод [7]: solve_univariate_inequality(f<0,x)
```

```
Out[7]: False
```

```
Ввод [8]: f=ln(x)
```

```
Ввод [9]: plot(f)
```



```
Out[9]: <sympy.plotting.plot.Plot at 0x292b254a880>
```

```
Ввод [10]: solve(f)
```

```
Out[10]: [1]
```

```
Ввод [11]: solve_univariate_inequality(f>0,x)
```

```
Out[11]: 1 < x ∧ x < ∞
```

```
Ввод [12]: solve_univariate_inequality(f<0,x)
```

```
Out[12]: 0 < x ∧ x < 1
```

```

Ввод [13]: f=2*x**2+3*x-5
Ввод [14]: plot(f)


Out[14]: <sympy.plotting.plot.Plot at 0x292b2544160>

```

```

Ввод [15]: solve(f)
Out[15]: [-5/2, 1]

Ввод [16]: solve_univariate_inequality(f>0,x)
Out[16]: (-∞ < x ∧ x < -5/2) ∨ (1 < x ∧ x < ∞)

Ввод [17]: solve_univariate_inequality(f<0,x)
Out[17]: -5/2 < x ∧ x < 1

```

```

Ввод [27]: f=sqrt(x)
Ввод [30]: plot(f, (x, 0, 100))


Out[30]: <sympy.plotting.plot.Plot at 0x292b8aa99a0>

```

```

Ввод [29]: solve(f)
Out[29]: [0]

Ввод [28]: solve_univariate_inequality(f>0,x)
Out[28]: 0 < x ∧ x < ∞

Ввод [31]: solve_univariate_inequality(f<0,x)
Out[31]: False

```

1.2 Выводы

В ходе выполнения лабораторной работы я лучше узнал о том, что такое функция, как коэффициенты влияют на график функции, а главное познакомился с библиотекой SymPy на языке программирования Python, которая позволяет строить и анализировать графики.

1 Лабораторная работа №3

1.1 Основная часть

1.1.1 Теоретическая часть

Для нахождения коэффициентов функции по заданным точкам используется метод интерполяции. Сначала необходимо записать систему уравнений с N неизвестными(где N -количество точек) для данных x и y и записать решения в виде:

```
1 a2 , a1 , a0 = symbols('a2 , a1 , a0 ')
2 eq1=6.25*a2 -2.5*a1+a0-15
3 eq2=4*a2 -2*a1+a0-10
4 eq3=2.25*a2 -1.5*a1+a0-7
```

Далее можно решить эту систему вручную или использовать функцию для решения нелинейной системы уравнений, что гораздо эффективнее при большом количестве точек:

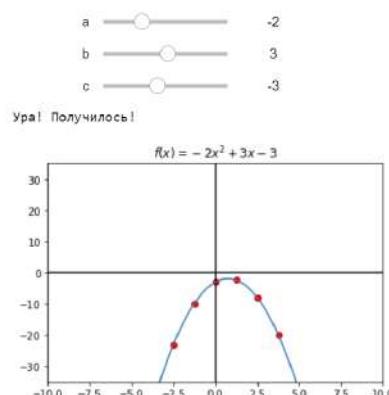
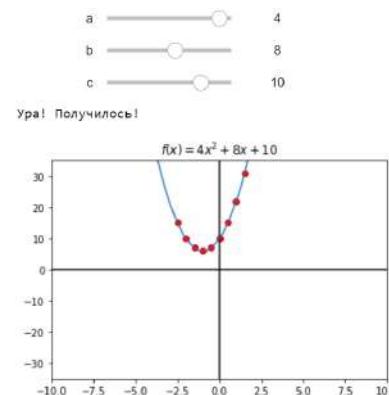
```
1 nonlinsolve([eq1 , eq2 , eq3 ],[a2 , a1 , a0 ])
```

1.1.2 Полученные результаты и их анализ

Свойства коэффициентов квадратичной функции. Практика

В качестве практики подберите, варьируя коэффициенты, квадратичную функцию для набора точек. Другими словами, вам нужно решить задачу интерполяции полиномом второй степени. Ниже вы найдете код, который создает ползунки, с помощью которых можно варьировать коэффициенты a , b и c , подбирая нужный вид параболы, а также рисует набор точек.

Программа будет печатать над графиком актуальный вид функции и сама скажет вам, когда вы достигнете цели.



Нахождение коэффициентов полиномов. Практика

С помощью SymPy найдите полиномы, описывающие наборы точек, данные ниже. Затем проведите полное исследование каждого полученного полинома: проверьте чётность/нечётность, найдите нули, промежутки знакопостоянства. Постройте графики полиномов. С помощью изменения промежутка для x добейтесь того, чтобы все нули были отображены на графике функции.

1. $(-2.5, 15.0), (-2, 10), (-1.5, 7.0)$
2. $(-0.5, -15.0), (-1, -31), (0, -7), (0.5, -4.0)$
3. $(-3, 33), (-2, 31), (-1, 18), (1, -18), (2, -31), (3, -33)$

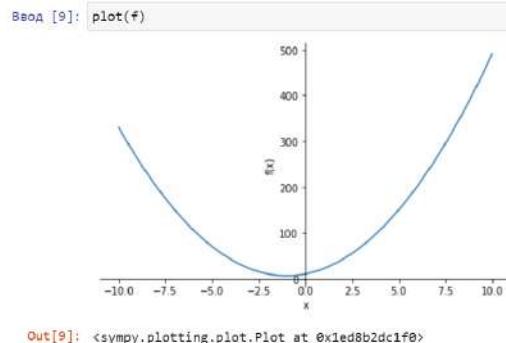
```
Ввод [1]: from sympy import*
from sympy.plotting import plot
from sympy.solvers.inequalities import solve_univariate_inequality
a2,a1,a0 = symbols('a2,a1,a0')

Ввод [4]: eq1=6.25*a2-2.5*a1+a0-15
eq2=4*a2-2*a1+a0-10
eq3=2.25*a2-1.5*a1+a0-7

Ввод [5]: nonlinsolve([eq1,eq2,eq3],[a2,a1,a0])
Out[5]: {(4.0, 8.000000000000002, 10.0)}

Ввод [7]: x=Symbol('x')

Ввод [8]: f=4*x**2+8*x+10
```



Полином не имеет вещественных решений. $f(-2)=10$ и $f(0)=10$, следовательно, функция чётная.

```
Ввод [10]: solve(f)
Out[10]: [-1 - sqrt(6)*I/2, -1 + sqrt(6)*I/2]

Ввод [11]: f.subs(x,-2)
Out[11]: 10

Ввод [14]: f.subs(x,0)
Out[14]: 10

Ввод [15]: solve_univariate_inequality(f>0,x)
Out[15]: -infinity < x ∧ x < infinity

Ввод [16]: solve_univariate_inequality(f<0,x)
Out[16]: False
```

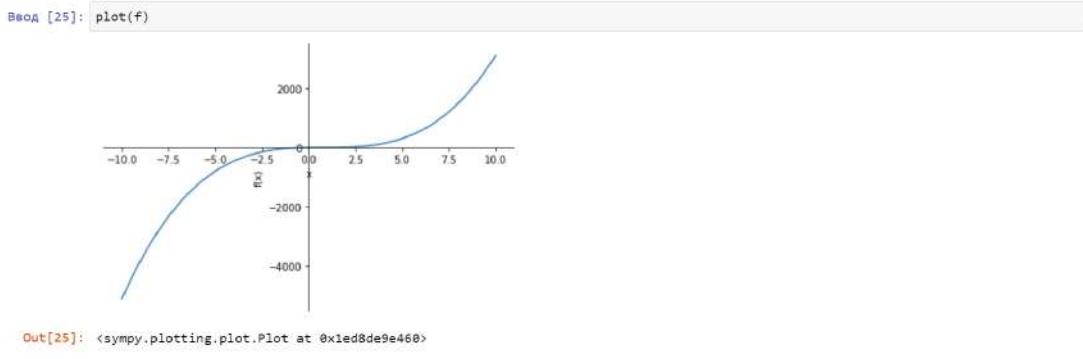
```
Ввод [17]: a3,a2,a1,a0 = symbols('a3,a2,a1,a0')

Ввод [22]: eq1=-0.125*a3+0.25*a2-0.5*a1+a0+15
eq2=-1*a3+1*a2-1*a1+a0+31
eq3=-7
eq4=0.125*a3+0.25*a2+0.5*a1+a0+4

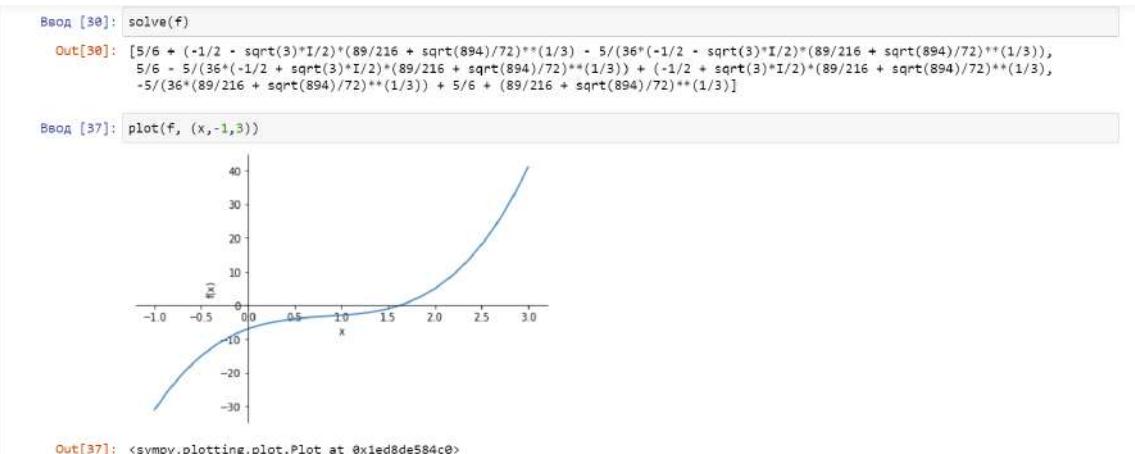
Ввод [23]: nonlinsolve([eq1,eq2,eq3,eq4],[a3,a2,a1,a0])
Out[23]: {(a3, 1.0*a3 - 14.0, 11.0 - 0.25*a3, - 0.25*a3 - 6.0)}

Ввод [24]: f=4*x**3-10*x**2+10*x-7
```

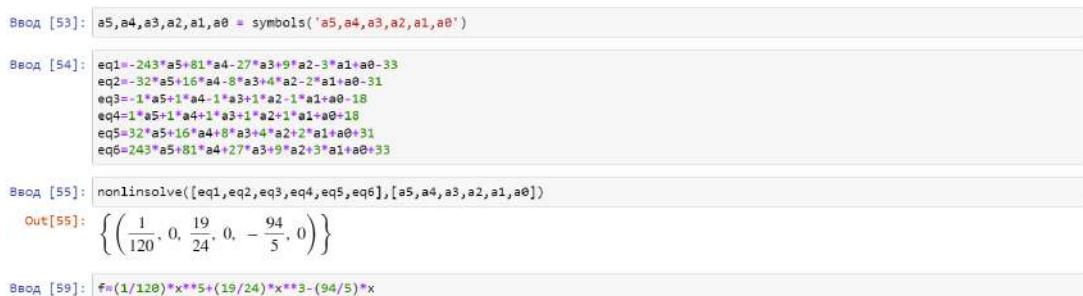
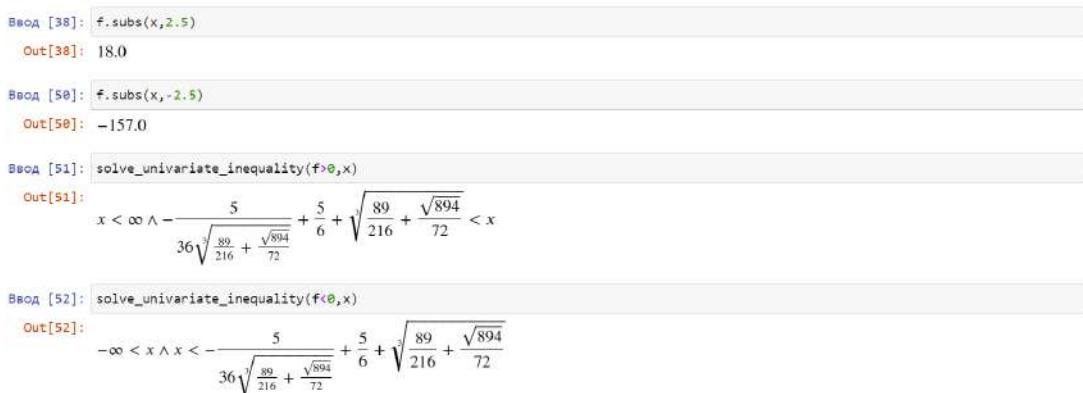
a_3 - любое число. Возьмём $a_3 = 4$, тогда $f(x) = 4 * x^3 - 10 * x^2 + 10 * x - 7$

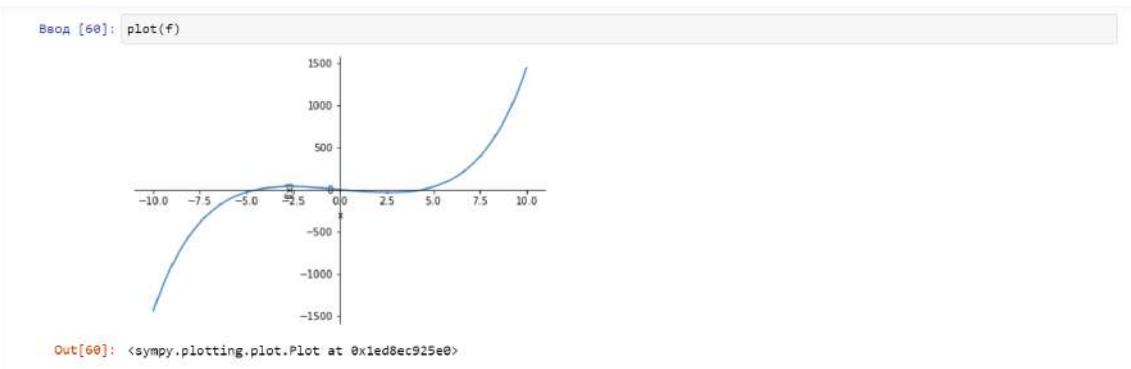


Полином имеет только одно вещественное решение.



$f(-2.5) = -157$ и $f(2.5) = 18$, функция гиперболическая, следовательно, функция нечётная.

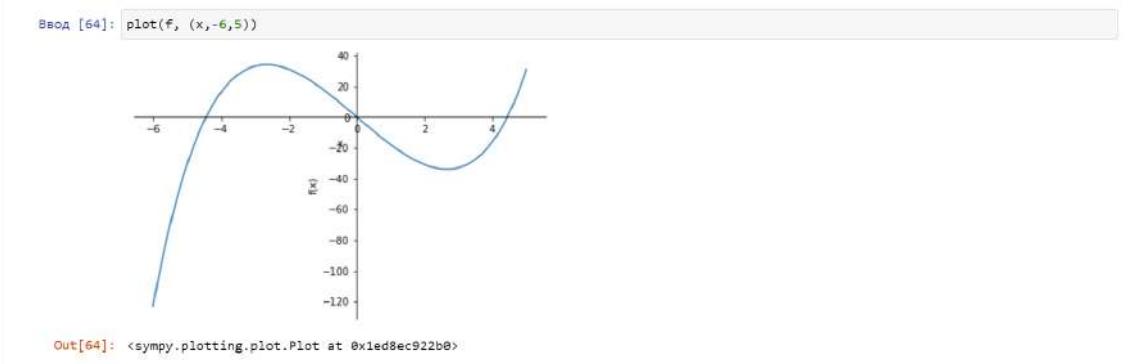




Полином имеет 3 вещественных решения. $f(-2.5)=-f(2.5)$, следовательно, функция нечётная.

Ввод [63]: `solve(f)`

Out[63]: [-4.43545751638198, 0.0, 4.43545751638198, -10.7085612189327*I, 10.7085612189327*I]



Ввод [65]: `f.subs(x, -2.5)`
Out[65]: 33.81640625

Ввод [66]: `f.subs(x, 2.5)`
Out[66]: -33.81640625

Ввод [67]: `solve_univariate_inequality(f>0,x)`
Out[67]: (-4.43545751638198 < x ∧ x < 0) ∨ (4.43545751638198 < x ∧ x < ∞)

Ввод [68]: `solve_univariate_inequality(f<0,x)`
Out[68]: (-∞ < x ∧ x < -4.43545751638198) ∨ (0 < x ∧ x < 4.43545751638198)

1.2 Выводы

В ходе выполнения лабораторной работы я научился методу интерполяции и узнал, как решать нелинейные системы уравнений с помощью библиотеки `sympy`.

1 Лабораторная работа №4

1.1 Основная часть

1.1.1 Теоретическая часть

При аппроксимации обычно возникает небольшое расхождение значений полученной в результате функции и аппроксимируемых точек. Это расхождение называется среднеквадратическая ошибка(MSE), она находится по формуле:

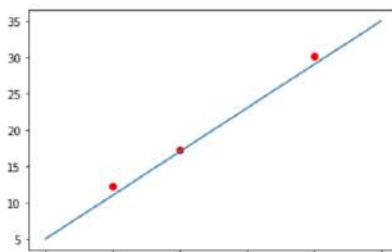
$$MSE = \frac{1}{N} \sum_{i=0}^N (\hat{y}_i - y_i)^2$$

При решении задач с использованием метода аппроксимации нам необходима сделать значение MSE как можно меньше.

1.1.2 Полученные результаты и их анализ

Задание 1

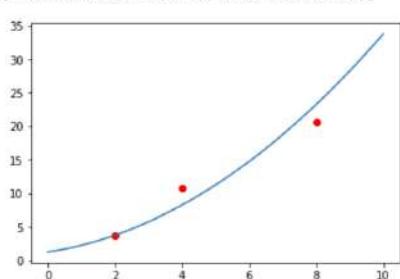
```
Ввод [9]: import matplotlib.pyplot as plt
import numpy as np
from sympy import *
Ввод [12]: plt.plot([2,4,8],[12.258,17.24,30.151],'ro')
x = np.linspace(0, 10, 100)
plt.plot(x, 3*x+5)
Out[12]: []
```



```
Ввод [24]: for i in [2,4,8]:
    print(3*i+5)
11
17
29
Ввод [25]: mse=((12.258-11)**2+(17.24-17)**2+(30.151-29)**2)/3
mse
Out[25]: 0.9883216666666655
Ввод [27]: rmse=sqrt(mse)
rmse
Out[27]: 0.994143685121354
```

```
Ввод [11]: plt.plot([2,4,8],[3.688,10.791,20.705], 'ro')
x = np.linspace(0, 10, 100)
plt.plot(x, 0.25*x**2+0.75*x+1.25)

Out[11]: [<matplotlib.lines.Line2D at 0x229688ded60>]
```



```
Ввод [23]: for i in [2,4,8]:
    print(0.25*i**2+0.75*i+1.25)

3.75
8.25
23.25
```

```
Ввод [13]: mse=((3.688-3.75)**2+(10.791-8.25)**2+(20.705-23.25)**2)/3
mse
```

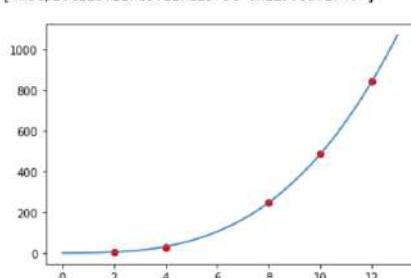
```
Out[13]: 4.312516666666667
```

```
Ввод [14]: rmse=sqrt(mse)
rmse
```

```
Out[14]: 2.07665997858741
```

```
Ввод [18]: plt.plot([2,4,8,10,12],[4.872,29.707,246.971,485.727,840.658], 'ro')
x = np.linspace(0, 13, 100)
plt.plot(x, 0.5*x**3-0.25*x**2+0.75*x+1.25)

Out[18]: [<matplotlib.lines.Line2D at 0x22968af1f40>]
```



```
Ввод [20]: for i in [2,4,8,10,12]:
    print(0.5*i**3-0.25*i**2+0.75*i+1.25)

5.75
32.25
247.25
483.75
838.25
```

```
Ввод [21]: mse=((4.872-5.75)**2+(29.707-32.25)**2+(246.971-247.25)**2+(485.727-483.75)**2+(840.658-838.25)**2)/5
mse
```

```
Out[21]: 3.40451339999994
```

```
Ввод [22]: rmse=sqrt(mse)
rmse
```

```
Out[22]: 1.84513235297634
```

Задание 2

```
Ввод [1]: from sympy import *
from sympy.plotting import plot
init_printing(use_unicode=False, wrap_line=False, no_global=True)

Ввод [2]: import matplotlib.pyplot as plt
import numpy as np

Ввод [3]: x = Symbol('x')
```

Получите значение MSE меньшее 110.

```
Ввод [4]: import matplotlib.pyplot as plt
import numpy as np

def print_points_and_function1(sympy_function):
    def function(x_): return float(sympy_function.subs(x, x_))

    points_X = np.array([-3, -2, -1, 1, 2, 3])
    points_Y = np.array([60, 30, 10, -10, -30, -20])
    plt.xlim(-15, 15)
    plt.ylim(-40, 80)

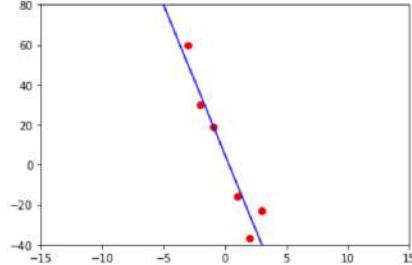
    plt.scatter(points_X, points_Y, c='r')
    x_range = np.linspace(plt.xlim()[0], plt.xlim()[1], num=100)
    function_Y = [function(x_) for x_ in x_range]
    plt.plot(x_range, function_Y, 'b')
    plt.show()

    MSE = sum([(points_Y[i] - function(points_X[i]))**2 for i in range(len(points_Y))]) / len(points_Y)
    print(f'MSE = {MSE}'')
```

```
Ввод [20]: x = Symbol('x')
f1 = - 15 * x + 5
f1
```

Out[20]: $5 - 15x$

```
Ввод [21]: print_points_and_function1(f1)
```



MSE = 99.16666666666667

2.

Получите значение MSE меньшее 150.

```
Ввод [203]: def print_points_and_function2(sympy_function):
    def function(x_): return float(sympy_function.subs(x, x_))

    points_X = np.array([-3, -2, -1, 1, 2, 3])
    points_Y = np.array([-55, -40, 7, 5, 38, 53])
    plt.xlim(-5, 25)
    plt.ylim(-70, 70)

    plt.scatter(points_X, points_Y, c='r')
    x_range = np.linspace(plt.xlim()[0], plt.xlim()[1], num=100)
    function_Y = [function(x_) for x_ in x_range]
    plt.plot(x_range, function_Y, 'b')
    plt.show()

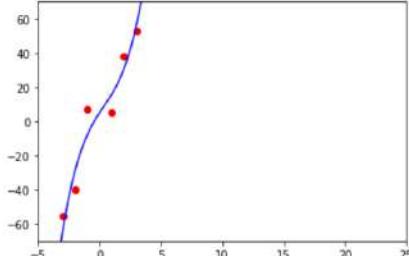
    MSE = sum([(points_Y[i] - function(points_X[i]))**2 for i in range(len(points_Y))]) / len(points_Y)
    print(f'MSE = {MSE}'')
```

```
Ввод [204]: f2 = 1 * x**3 - 1 * x**2 + 11 * x + 5
```

```
f2
```

```
x3 - x2 + 11x + 5
```

```
Ввод [205]: print_points_and_function2(f2)
```



```
MSE = 101.0
```

Задание 3

```
Ввод [1]: from sympy import *
from sympy.plotting import plot
init_printing(use_unicode=False, wrap_line=False, no_global=True)
```

```
Ввод [2]: import matplotlib.pyplot as plt
import numpy as np
```

```
Ввод [3]: x = Symbol('x')
```

1.

Получите значение MSE меньшее 50.

```
Ввод [4]: def print_points_and_function1(sympy_function):
    def function(x_): return float(sympy_function.subs(x, x_))

    points_X = np.array([-2, -1, 0, 1, 2, 3, 3.5, 4, 4.5, 5])
    points_Y = np.array([-15, -1, 4, -9, -2, -5, -8, 4, 13, 21])
    plt.xlim(-3, 6)
    plt.ylim(-20, 20)

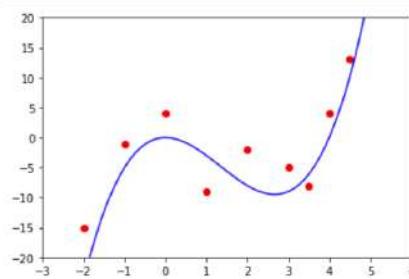
    plt.scatter(points_X, points_Y, c='r')
    x_range = np.linspace(plt.xlim()[0], plt.xlim()[1], num=100)
    function_Y = [function(x_) for x_ in x_range]
    plt.plot(x_range, function_Y, 'b')
    plt.show()

    MSE = sum([(points_Y[i] - function(points_X[i]))**2 for i in range(len(points_Y))]) / len(points_Y)
    print(f'MSE = {MSE}'')
```

```
Ввод [30]: f1 = 1 * x ** 3 - 4 * x**2
f1
```

```
x3 - 4x2
```

```
Ввод [31]: print_points_and_function1(f1)
```



```
MSE = 24.478125
```

2.

Получите значение MSE меньшее 150.

```
Ввод [32]: def print_points_and_function2(sympy_function):
    def function(x_):
        return float(sympy_function.subs(x, x_))

    points_X = np.array([-3, -2, -1, 1, 2, 3])
    points_Y = np.array([-55, -40, 7, 5, 38, 53])
    plt.xlim(-10, 10)
    plt.ylim(-70, 70)

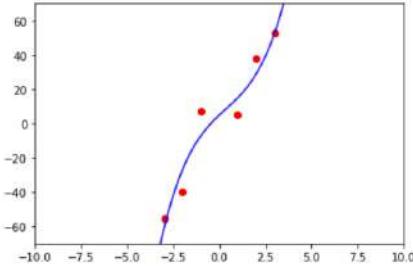
    plt.scatter(points_X, points_Y, c='r')
    x_range = np.linspace(plt.xlim()[0], plt.xlim()[1], num=100)
    function_Y = [function(x_) for x_ in x_range]
    plt.plot(x_range, function_Y, 'b')
    plt.show()

    MSE = sum([(points_Y[i] - function(points_X[i]))**2 for i in range(len(points_Y))]) / len(points_Y)
    print(f'MSE = {MSE}')
```

```
Ввод [45]: f2 = 1 * x**3 - 1 * x**2 + 10 * x + 5
f2
```

```
Out[45]:  $x^3 - x^2 + 10x + 5$ 
```

```
Ввод [46]: print_points_and_function2(f2)
```



```
MSE = 97.0
```

Задание 4

Блок 1.

С помощью symPy найдите полиномы, описывающие данные наборы точек. Затем проведите полное исследование каждого полученного полинома: проверьте четность/нечетность, найдите нули, промежутки знакопостоянства. Постройте их графики. С помощью изменения промежутка для x добейтесь того, чтобы все нули были отражены на графике функции.

a) $(-4; -4268), (-3; -1227), (-1; -17), (1; 17), (3; 1227), (4; 4268)$

б) $(-4; -16729), (-3; -3999), (-1; 5), (1; 1), (3; 4005), (4; 16735)$

```
Ввод [3]: from sympy import *
from sympy.plotting import plot
from sympy.solvers.inequalities import solve_univariate_inequality

a0, a1, a2, a3, a4, a5 = symbols('a0, a1, a2, a3, a4, a5')

Ввод [4]: eq1=-1024*a5+256*a4-64*a3+16*a2-4*a1+a0+4268
eq2=-243*a5+81*a4-27*a3+9*a2-3*a1+a0+1227
eq3=-1*a5+1*a4-1*a3+1*a2-1*a1+a0+17
eq4=1*a5+1*a4+1*a3+1*a2+1*a1+a0-17
eq5=243*a5+81*a4+27*a3+9*a2+3*a1+a0-1227
eq6=1024*a5+256*a4+64*a3+16*a2+4*a1+a0-4268

Ввод [9]: nonlinsolve([eq1,eq2,eq3,eq4,eq5,eq6],[a5,a4,a3,a2,a1,a0])
out[9]: {(3, 0, 19, 0, -5, 0)}
```

```

Ввод [10]: x=Symbol('x')
a5 = 3; a4 = 0; a3 = 19; a2 = 0; a1 = -5; a0 = 0;
f=a5 * x**5 + a4 * x**4 + a3 * x**3 + a2 * x**2 + a1 * x + a0
f

Out[10]: 3x5 + 19x3 - 5x

Ввод [15]: def print_points_and_function(sympy_function):
    def function(x_):
        return float(sympy_function.subs(x, x_))

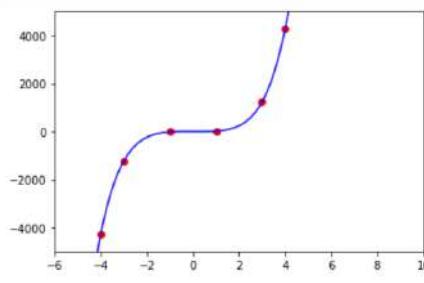
    points_X = np.array([-4, -3, -1, 1, 3, 4])
    points_Y = np.array([-4268, -1227, -17, 17, 1227, 4268])
    plt.xlim(-6, 10)
    plt.ylim(-5000, 5000)

    plt.scatter(points_X, points_Y, c='r')
    x_range = np.linspace(plt.xlim()[0], plt.xlim()[1], num=100)
    function_Y = [function(x_) for x_ in x_range]
    plt.plot(x_range, function_Y, 'b')
    plt.show()

    MSE = sum([(points_Y[i] - function(points_X[i]))**2 for i in range(len(points_Y))]) / len(points_Y)
    print(f'MSE = {MSE}')

```

```
Ввод [16]: print_points_and_function(f)
```



```
MSE = 0.0
```

```
Ввод [8]: f.subs(x,4)
```

```
Out[8]: 4268
```

```
Ввод [9]: f.subs(x,-4)
```

```
Out[9]: -4268
```

```
Ввод [10]: solve(f)
```

```
Out[10]: [0, -i*sqrt(19/6 + sqrt(421)/6), i*sqrt(19/6 + sqrt(421)/6), -sqrt(-19/6 + sqrt(421)/6), sqrt(-19/6 + sqrt(421)/6)]
```

```
Ввод [11]: solve_univariate_inequality(f>0,x)
```

```
Out[11]: (x < 0 ∧ -sqrt(-19/6 + sqrt(421)/6) < x) ∨ (x < ∞ ∧ sqrt(-19/6 + sqrt(421)/6) < x)
```

```
Ввод [12]: solve_univariate_inequality(f<0,x)
```

```
Out[12]: (-∞ < x ∧ x < -sqrt(-19/6 + sqrt(421)/6)) ∨ (0 < x ∧ x < sqrt(-19/6 + sqrt(421)/6))
```

При изменении знака аргумента меняется знак значения функции, следовательно, функция нечётная. Существует 3 действительных решения.

```
Ввод [19]: a0, a1, a2, a3, a4, a5 = symbols('a0, a1, a2, a3, a4, a5')
eq1=-1024*a5+256*a4-64*a3+16*a2-4*a1+a0+16729
eq2=-243*a5+81*a4-27*a3+9*a2-3*a1+a0+3999
eq3=-1*a5+1*a4-1*a3+1*a2-1*a1+a0-5
eq4=1*a5+1*a4+1*a3+1*a2+1*a1+a0-1
eq5=243*a5+81*a4+27*a3+5*a2+3*a1+a0-4005
eq6=1024*a5+256*a4+64*a3+16*a2+4*a1+a0-16735
```

```
Ввод [20]: nonlinsolve([eq1,eq2,eq3,eq4,eq5,eq6],[a5,a4,a3,a2,a1,a0])
```

```
Out[20]: {(16, 0, 7, 0, -25, 3)}
```

```
Ввод [23]: x=Symbol('x')
a5 = 16; a4 = 0; a3 = 7; a2 = 0; a1 = -25; a0 = 3;
f=a5 * x**5 + a4 * x**4 + a3 * x**3 + a2 * x**2 + a1 * x + a0
f
```

```
Out[23]: 16x5 + 7x3 - 25x + 3
```

```
Ввод [24]: def print_points_and_function(sympy_function):
    def function(x_): return float(sympy_function.subs(x, x_))

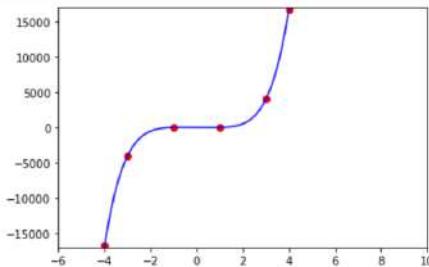
    points_X = np.array([-4, -3, -1, 1, 3, 4])
    points_Y = np.array([-16729, -3999, 5, 1, 4005, 16735])
    plt.xlim(-6, 10)
    plt.ylim(-17000, 17000)

    plt.scatter(points_X, points_Y, c='r')
    x_range = np.linspace(plt.xlim()[0], plt.xlim()[1], num=100)
    function_Y = [function(x_) for x_ in x_range]
    plt.plot(x_range, function_Y, 'b')
    plt.show()

    MSE = sum([(points_Y[i] - function(points_X[i]))**2 for i in range(len(points_Y))]) / len(points_Y)
    print(f'MSE = {MSE}')


print_points_and_function(f)
```

```
Ввод [25]: print_points_and_function(f)
```



MSE = 0.0

```
Ввод [20]: f.subs(x,4)
```

```
Out[20]: 16735
```

```
Ввод [21]: f.subs(x,-4)
```

```
Out[21]: -16729
```

```
Ввод [28]: solve(f)
```

```
Out[28]: [CRootOf(16x5 + 7x3 - 25x + 3, 0), CRootOf(16x5 + 7x3 - 25x + 3, 1), CRootOf(16x5 + 7x3 - 25x + 3, 2), CRootOf(16x5 + 7x3 - 25x + 3, 3), CRootOf(16x5 + 7x3 - 25x + 3, 4)]
```

```
Ввод [23]: solve_univariate_inequality(f>0,x)
```

```
Out[23]: (x < infinity ∧ CRootOf(16x5 + 7x3 - 25x + 3, 2) < x) ∨ (x < CRootOf(16x5 + 7x3 - 25x + 3, 1) ∧ CRootOf(16x5 + 7x3 - 25x + 3, 0) < x)
```

```
Ввод [24]: solve_univariate_inequality(f<0,x)
```

```
Out[24]: (-infinity < x ∧ x < CRootOf(16x5 + 7x3 - 25x + 3, 0)) ∨ (x < CRootOf(16x5 + 7x3 - 25x + 3, 2) ∧ CRootOf(16x5 + 7x3 - 25x + 3, 1) < x)
```

При изменении знака аргумента меняется знак значения функции, следовательно, функция нечётная. Корни не выводятся в явном виде, но Wolfram Alpha показывает, что существует 3 действительных решения:

Real solutions



$$x \approx -1.05749$$

$$x \approx 0.120506$$

$$x \approx 0.986408$$

Exact forms

More digits

Complex solutions

$$x \approx -0.02471 - 1.22107i$$

$$x \approx -0.02471 + 1.22107i$$

More digits

Exact forms

Блок 2.

```
Ввод [13]: from sympy import *
from sympy.plotting import plot
init_printing(use_unicode=False, wrap_line=False, no_global=True)
```

```
Ввод [14]: import matplotlib.pyplot as plt
import numpy as np
```

```
Ввод [3]: x = Symbol('x')
```

1.

Получите значение MSE меньшее 5.

```
Ввод [25]: def print_points_and_function1(sympy_function):
    def function(x_):
        return float(sympy_function.subs(x, x_))

    points_X = np.array([-2, -1, 0, 1, 2, 3, 3.5, 4, 4.5, 5])
    points_Y = np.array([2, -4, 1, 8, 21, 40, 47, 65, 75, 92])
    plt.xlim(-6, 10)
    plt.ylim(-1, 100)

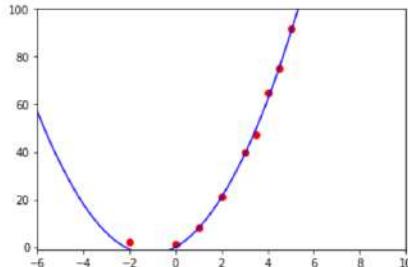
    plt.scatter(points_X, points_Y, c='r')
    x_range = np.linspace(plt.xlim()[0], plt.xlim()[1], num=100)
    function_Y = [function(x_) for x_ in x_range]
    plt.plot(x_range, function_Y, 'b')
    plt.show()

    MSE = sum([(points_Y[i] - function(points_X[i]))**2 for i in range(len(points_Y))]) / len(points_Y)
    print(f'MSE = {MSE}')
```

```
Ввод [41]: f1 = 2.5 * x**2 + 5.5 * x
f1
```

```
Out[41]: 2.5x2 + 5.5x
```

```
Ввод [42]: print_points_and_function1(f1)
```



MSE = 3.340625

2.

Получите значение MSE меньшее 35.

```
Ввод [45]: def print_points_and_function2(sympy_function):
    def function(x_): return float(sympy_function.subs(x, x_))

    points_X = np.array([-2, -1, 0, 1, 2, 3, 3.5, 4, 4.5, 5])
    points_Y = np.array([-31, -9, 4, -1, 9, 24, 47, 92, 120, 170])
    plt.xlim(-3, 6)
    plt.ylim(-35, 200)

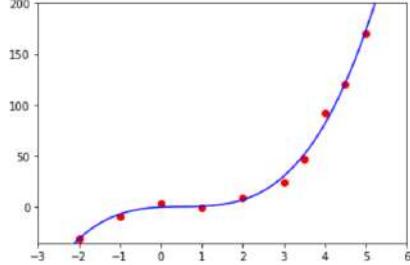
    plt.scatter(points_X, points_Y, c='r')
    x_range = np.linspace(plt.xlim()[0], plt.xlim()[1], num=100)
    function_Y = [function(x_) for x_ in x_range]
    plt.plot(x_range, function_Y, 'b')
    plt.show()

    MSE = sum([(points_Y[i] - function(points_X[i]))**2 for i in range(len(points_Y))]) / len(points_Y)
    print(f'MSE = {MSE}')
```

```
Ввод [52]: f2 = 1.9 * x**3 - 3 * x**2 + 2 * x
f2
```

```
Out[52]: 1.9x3 - 3x2 + 2x
```

```
Ввод [53]: print_points_and_function2(f2)
```



```
MSE = 20.553281249999994
```

3.

Получите значение MSE меньшее 3300.

```
Ввод [56]: def print_points_and_function3(sympy_function):
    def function(x_): return float(sympy_function.subs(x, x_))

    points_X = np.array([-2, -1, 0, 1, 2, 3, 3.5, 4, 4.5, 5])
    points_Y = np.array([60, 25, 4, -8, -57, -195, -295, -540, -700, -760])
    plt.xlim(-10, 6)
    plt.ylim(-850, 100)

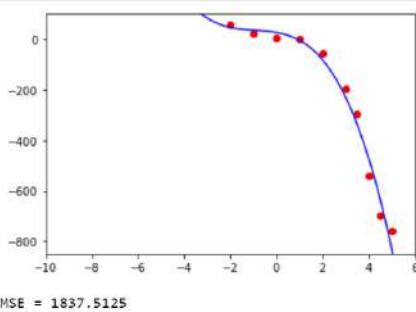
    plt.scatter(points_X, points_Y, c='r')
    x_range = np.linspace(plt.xlim()[0], plt.xlim()[1], num=100)
    function_Y = [function(x_) for x_ in x_range]
    plt.plot(x_range, function_Y, 'b')
    plt.show()

    MSE = sum([(points_Y[i] - function(points_X[i]))**2 for i in range(len(points_Y))]) / len(points_Y)
    print(f'MSE = {MSE}')
```

```
Ввод [54]: f3 = -4 * x**3 - 11 * x**2 - 16 * x + 27
f3
```

```
Out[54]: -4x3 - 11x2 - 16x + 27
```

```
Ввод [57]: print_points_and_function3(f3)
```



4.

Получите значение MSE меньшее 25.

```
Ввод [60]: def print_points_and_function4(sympy_function):
    def function(x_): return float(sympy_function.subs(x, x_))

    points_X = np.array([-2, -1, 0, 1, 2, 3, 3.5, 4, 4.5, 5])
    points_Y = np.array([-42, -37, -23, -36, -45, -80, -83, -110, -131, -155])
    plt.xlim(-4, 20)
    plt.ylim(-160, -10)

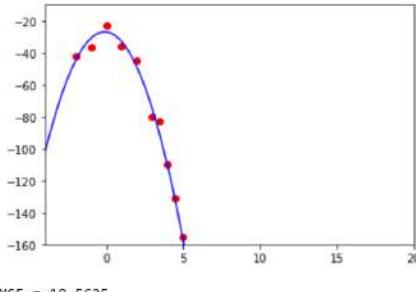
    plt.scatter(points_X, points_Y, c='r')
    x_range = np.linspace(plt.xlim()[0], plt.xlim()[1], num=100)
    function_Y = [function(x_) for x_ in x_range]
    plt.plot(x_range, function_Y, 'b')
    plt.show()

    MSE = sum([(points_Y[i] - function(points_X[i]))**2 for i in range(len(points_Y))]) / len(points_Y)
    print(f'MSE = {MSE}')
```

```
Ввод [58]: f4 = -5 * x**2 - 1 * x - 27
f4
```

```
Out[58]: -5x2 - x - 27
```

```
Ввод [61]: print_points_and_function4(f4)
```



1.2 Выводы

В ходе выполнения лабораторной работы я научился методу аппроксимации, нахождению среднеквадратической ошибки и построению графиков с помощью библиотеки matplotlib.

1 Лабораторная работа №5

1.1 Основная часть

1.1.1 Теоретическая часть

При использовании метода аппроксимации необходимо найти функцию с минимальным значением функции потерь(MSE). Для этого удобно работать с производной функции потерь(MSE), находя с её помощью точку минимума.

Для нахождения производной используются следующие базовые правила:

$$(cf(x))' = c f'(x)$$

$$(u(b) + v(b))' = u'(b) + v'(b)$$

$$(x^n)' = nx^{n-1}$$

$$(uv)' = u'v + v'u$$

$$c' = 0$$

$$\left(\frac{u}{v}\right)' = \frac{u'v - v'u}{v^2}$$

$$(f(g(x)))' = f'(g(x))g'(x)$$

1.1.2 Полученные результаты и их анализ

Задание 1

```
Ввод [20]: from sympy import *
x=Symbol('x')

Ввод [3]: f=2*x+3
f

Out[3]: 2x + 3

Ввод [4]: solve_univariate_inequality(f<0,x)

Out[4]: -∞ < x ∧ x < - $\frac{3}{2}$ 

Ввод [5]: solve_univariate_inequality(f>0,x)

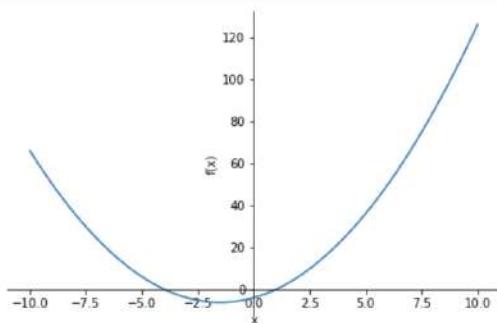
Out[5]: - $\frac{3}{2}$  < x ∧ x < ∞

Ввод [6]: solve(f)

Out[6]: [-3/2]
```

```
Ввод [8]: f=x**2+3*x-4
```

```
Ввод [9]: plot(f)
```



```
Out[9]: <sympy.plotting.plot.Plot at 0x246ddd4db20>
```

На интервале $(-\infty; -\frac{3}{2})$ функция убывает, на интервале $(-\frac{3}{2}; \infty)$ функция возрастает, $x = -\frac{3}{2}$ - точка минимума функции.

```
Ввод [11]: f=3*x**2-4*x+1  
f
```

```
Out[11]: 3x2 - 4x + 1
```

```
Ввод [12]: solve_univariate_inequality(f<0,x)
```

```
Out[12]: 1/3 < x ∧ x < 1
```

```
Ввод [13]: solve_univariate_inequality(f>0,x)
```

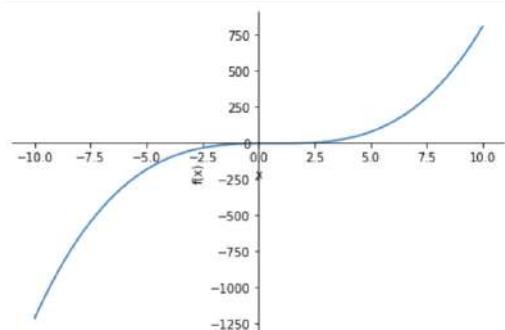
```
Out[13]: (-∞ < x ∧ x < 1/3) ∨ (1 < x ∧ x < ∞)
```

```
Ввод [14]: solve(f)
```

```
Out[14]: [1/3, 1]
```

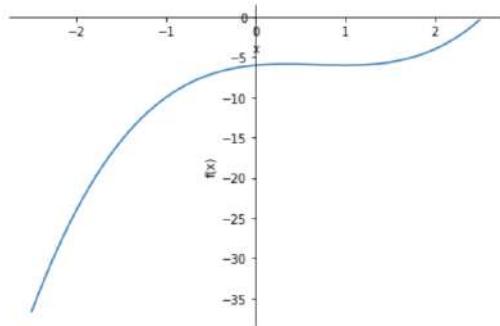
```
Ввод [15]: f=x**3-2*x**2+x-6
```

```
Ввод [16]: plot(f)
```



```
Out[16]: <sympy.plotting.plot.Plot at 0x246ddd70be0>
```

```
Ввод [18]: plot(f,(x, -2.5, 2.5))
```



```
Out[18]: <sympy.plotting.plot at 0x246ddfe7f0>
```

На интервале $(\frac{1}{3}; 1)$ функция убывает, на интервале $(-\infty; \frac{1}{3}) \cup (1; +\infty)$ функция возрастает, $x = \frac{1}{3}$ точка максимума функции, $x = 1$ - точка минимума функции.

```
Ввод [28]: f=(2-2*x)*e**(2*x-x**2)
```

```
Out[28]: 2.71828182845905 $^{-x^2+2x}$  (2 - 2x)
```

```
Ввод [29]: solve_univariate_inequality(f<0,x)
```

```
Out[29]: 1 < x ∧ x < 0
```

```
Ввод [30]: solve_univariate_inequality(f>0,x)
```

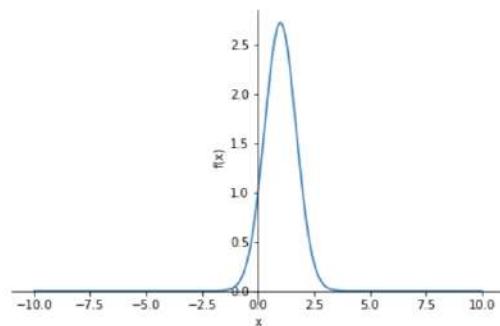
```
Out[30]: -∞ < x ∧ x < 1
```

```
Ввод [33]: solve(f)
```

```
Out[33]: [1.00000000000000]
```

```
Ввод [34]: f=e**(2*x-x**2)
```

```
Ввод [40]: plot(f)
```



```
Out[40]: <sympy.plotting.plot at 0x246e118e280>
```

На интервале $(-\infty; 1)$ функция возрастает, на интервале $(1; +\infty)$ функция убывает, $x = 1$ - точка максимума функции.

Все значения найденные с помощью производной совпали с графиком функции.

Задание 2

$$1. f(x) = 5x^2 \quad (x^2)' = 2x$$

$$(5x^2)' = 5(x^2)' = 5 \cdot 2x = 10x$$

$$2. f(x) = 3\sin x \quad (\sin x)' = \cos x$$

$$(3\sin x)' = 3(\sin x)' = 3\cos x$$

$$3. f(x) = \frac{2}{3}e^x \quad (e^x)' = e^x$$

$$\left(\frac{2}{3}e^x\right)' = \frac{2}{3}(e^x)' = \frac{2}{3}e^x$$

$$4. f(x) = 5x^2 + 3\sin x$$

$$(5x^2 + 3\sin x)' = (5x^2)' + (3\sin x)' = 10x + 3\cos x$$

$$5. f(x) = 7x^3 + 3x - 4, \quad (x^3)' = 3x^2$$

$$(x)' = 1 \quad (-4)' = 0$$

$$(7x^3 + 3x - 4)' = 7(x^3)' + 3(x)' + (-4)' =$$

$$= 7 \cdot 3x^2 + 3 = 21x^2 + 3$$

$$6. f(x) = 13e^x + 3x^2 - 4\sin x$$

$$(13e^x + 3x^2 - 4\sin x)' =$$

$$= 13(e^x)' + 3(x^2)' - 4(\sin x)' =$$

$$= 13e^x + 6x - 4\cos x$$

Задание 3

$$1. f(x) = 5x^5 + 7x, \quad (x^n)' = n \cdot x^{n-1}$$

$$f'(x) = 5 \cdot 5x^4 + 7 = 25x^4 + 7$$

$$2. f(x) = 6x^4 + 12x^2 + 5$$

$$\begin{aligned} f'(x) &= 6 \cdot 4x^3 + 12 \cdot 2x = 24x^3 + 24x = \\ &= 24(x^3 + x) \end{aligned}$$

$$3. f(x) = 6x^3 + 3x^4 - 4$$

$$f'(x) = 6 \cdot 3x^2 + 3 \cdot 4x^3 = 18x^2 + 12x^3$$

$$4. f(x) = (2x+3)(6x+2) \quad (f(x)g(x))' =$$

$$\begin{aligned} f'(x) &= (2x+3)'(6x+2) + \quad = f'(x)g(x) + \\ &\quad + (2x+3)(6x+2)' = 2(6x+2) + 6(2x+3) = \\ &= 12x+4 + 12x+18 = 24x+22 \end{aligned}$$

$$5. f(x) = (3x^2 + e^x) \sin x$$

$$\begin{aligned}f'(x) &= (3x^2 + e^x)' \sin x + (3x^2 + e^x)(\sin x)' = \\&= (6x + e^x) \sin x + (3x^2 + e^x) \cos x\end{aligned}$$

$$6. f(x) = 3 \sin^2 x \quad f'(g(x)) = f(x) g'(x)$$

$$f'(x) = (\beta \sin^2 x)' = \beta (\sin^2 x)' =$$

~~$\sin x \cdot 2 \sin x \cos x$~~

$$= 3 \cdot 2 \sin x \cdot \cos x = 3 \sin 2x$$

$$7. f(x) = 13e^{2x} + 5x^7$$

$$f'(x) = 13 \cdot 2e^{2x} + 5 \cdot 7x^6 =$$

$$= 26e^{2x} + 35x^6$$

1.2 Выводы

В ходе выполнения лабораторной работы я больше узнал о том, как оптимально находить аппроксимирующую функцию с помощью минимизации значения функции потерь, а также попрактиковался в нахождении производных элементарных функций.

1 Лабораторная работа №6

1.1 Основная часть

1.1.1 Теоретическая часть

Часто возникает необходимость в применении функций, которые используют сразу несколько переменных. Область определения таких функций задаётся как декартово произведение областей определения всех переменных. Например:

$$A = 18, 19, \dots, 80$$

$$S = [0, 1000000]$$

$$X = 0, 1, \dots, 62$$

$$Y = [0, 1]$$

$$f(a, s, x) : A \times S \times X \longrightarrow Y$$

Размерность гиперплоскости, которую задаёт функция нескольких переменных, равна количеству переменных и на 1 меньше, чем пространство, в котором находится. Гиперплоскость размером \mathbb{R} - прямая, \mathbb{R}^2 - плоскость.

Для построения трёхмерных графиков в sympy используется функция plot3d().

1.1.2 Полученные результаты и их анализ

Задание 1

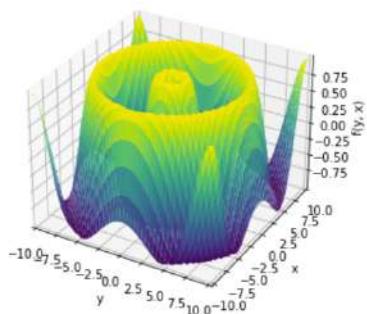
```
Ввод [1]: from sympy import *
from sympy.plotting import plot3d
init_printing(use_unicode=False, wrap_line=False, no_global=True)

Ввод [2]: x, y = symbols('x, y')

Ввод [3]: f = sin(sqrt(x**2+y**2))
f
```

Out[3]: $\sin(\sqrt{x^2 + y^2})$

```
Ввод [4]: plot3d(f)
```



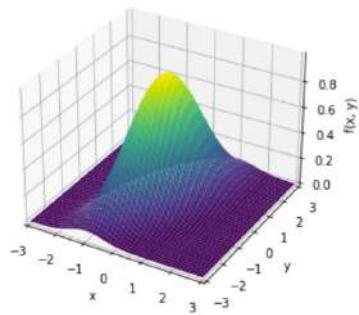
Out[4]: <sympy.plotting.plot.Plot at 0x245c6ca4af0>

```
Ввод [5]: f = exp(-0.9*x**2-0.45*(x-y)**2)
```

```
f
```

```
e-0.9x2-0.45(x-y)2}
```

```
Ввод [7]: plot3d(f, (x, -3, 3), (y, -3, 3))
```



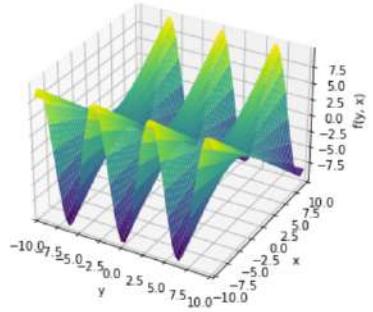
```
Out[7]: <sympy.plotting.plot.Plot at 0x245c49fb040>
```

```
Ввод [8]: f = x*cos(y)
```

```
f
```

```
x cos(y)
```

```
Ввод [9]: plot3d(f)
```



```
Out[9]: <sympy.plotting.plot.Plot at 0x245c9909b20>
```

Задание 2

$$1. f(e_1, e_2, e_3) : E_1 \times E_2 \times E_3 \rightarrow Y$$

$$E_1 = E_2 = E_3 = \{1, 2, 3, 4, 5\}$$

$$Y = [0, 1]$$

$$2. f(r, t, e) : R \times T \times E \rightarrow Y$$

$$R = [0, 1]$$

$$T = [-30, 35]$$

$$E = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$Y = \{0, 1\}$$

Задание 3

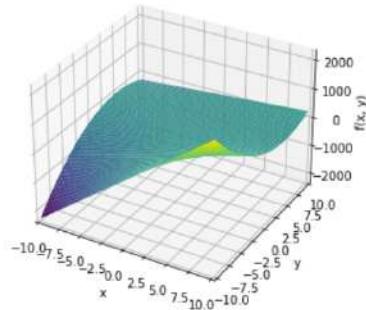
```
Ввод [1]: from sympy import *
from sympy.plotting import plot3d
init_printing(use_unicode=False, wrap_line=False, no_global=True)

Ввод [2]: x, y = symbols('x, y')

Ввод [4]: f = x*(y-5)**2
f

Out[4]: x(y - 5)2
```

```
Ввод [5]: plot3d(f)
```



```
Out[5]: <sympy.plotting.plot.Plot at 0x256a696be20>
```

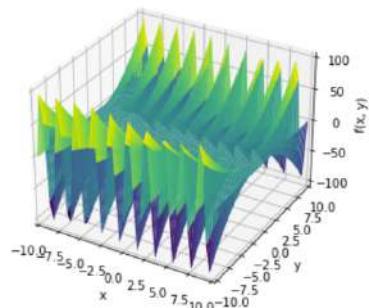
```
Ввод [6]: solve(f)
```

```
Out[6]: [{x: 0}, {y: 5}]
```

```
Ввод [10]: f=(y-exp(1)/20)**2*sin(pi*x)
```

$$f = \left(y - \frac{e}{20}\right)^2 \sin(\pi x)$$

```
Ввод [11]: plot3d(f)
```



```
Out[11]: <sympy.plotting.plot.Plot at 0x256a9769700>
```

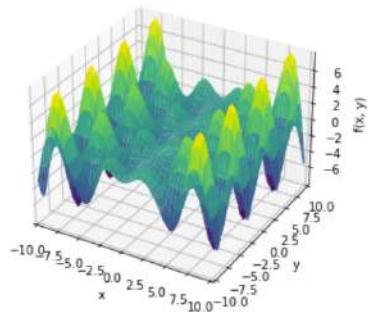
```
Ввод [12]: solve(f)
```

$$\text{Out[12]: } \left[\{x: 0\}, \{x: 1\}, \left\{y: \frac{e}{20}\right\} \right]$$

```
Ввод [13]: f=x*sin(x)*sin(y-2)
```

$$\text{Out[13]: } x \sin(x) \sin(y - 2)$$

Ввод [14]: `plot3d(f)`



Out[14]: <sympy.plotting.plot.Plot at 0x256aae4c5e0>

Ввод [15]: `solve(f)`

Out[15]: $\{x: 0\}, \{x: \pi\}, \{y: 2\}, \{y: 2 + \pi\}$

$$1. f(x_1, x_2, x_3, x_4): X_1 \times X_2 \times X_3 \times X_4 \rightarrow Y$$

$$X_1 = \{0, 1, 2, 3, \dots, +\infty\}$$

$$X_2 = \{0, 1\}$$

$$X_3 = \{0, 1\}$$

$$X_4 = \{1, 2, 3, \dots, +\infty\}$$

$$Y = \{0, 1\}$$

2. \mathbb{R}^4

$$1. f(x_1, x_2, x_3): X_1 \times X_2 \times X_3 \rightarrow Y$$

$$X_1 = \{0, 1, 2, 3, \dots, +\infty\}$$

$$X_2 = [0, +\infty]$$

$$X_3 = \{0, 1\}$$

$$Y = [0, 1]$$

2. \mathbb{R}^3

1. $f(x_1, x_2, x_3, x_4) : X_1 \times X_2 \times X_3 \times X_4 \rightarrow Y$

$$X_1 = \{0, 1\}$$

$$X_2 = [35, 40]$$

$$X_3 = \{0, 1\}$$

$$X_4 = \{0, 1\}$$

$$Y = \{0, 1\}$$

2. \mathbb{R}^4

1.2 Выводы

В ходе выполнения лабораторной работы я узнал об определении функций нескольких переменных, пространствах и гиперплоскостях, а также о том, как строить графики трёхмерных функций в SymPy.

1 Лабораторная работа №7

1.1 Основная часть

1.1.1 Теоретическая часть

Для нахождения аппроксимирующей функции, зависящей от нескольких переменных, так же используется функция потерь MSE. Для определения функции с минимальным значением MSE находится частная производная от каждой неизвестной в функции MSE, а потом решается система, в которой записаны эти производные приравненные к нулю, так находятся точки экстремума.

Главное правило при взятии частной производной: брать ту переменную, по которой берётся производная, как обычную производную, а остальные переменные принять за константу.

$$f'_{x_1}(x_1, x_2) = (2x_1 + 3x_3)'_{x_1} = (2x_1)'_{x_1} + (3x_3)'_{x_1} = 2 + 0 = 2$$

$$f'_{x_2}(x_1, x_2) = (2x_1 + 3x_3)'_{x_2} = (2x_1)'_{x_2} + (3x_3)'_{x_2} = 0 + 3 = 3$$

В sympy для нахождения производной используется функция diff(), в которую записывается функция и, при необходимости найти частную производную, после запятой указывается переменная, по которой находится производная. Например:

```
1 f=10*x1-5*x2
2 diff(f, x1)
```

1.1.2 Полученные результаты и их анализ

Задание 1

```
Ввод [1]: from sympy import *
Ввод [5]: MSE1=1/3*(((-2 * 10 + 30 - 7) - 7)**2 + ((-2 * (-5) + 15 - 7) - 20)**2 + ((-2 * 16 + 31 - 7) + 4)**2)
MSE2=1/3*((20 * 10 + 3 * 30 - 4) - 7)**2 + ((20 * (-5) + 3 * 15 - 4) - 20)**2 + ((20 * 16 + 3 * 31 - 4) + 4)**2)
print(MSE1, MSE2)
12.0 84883.66666666666
Ввод [8]: MSE1=1/3*(((-2 * 16 - 17 + 60) - 13)**2 + ((-2 * (-3) - 28 + 60) - 42)**2 + ((-2 * 14 - 85 + 60) + 39)**2)
MSE2=1/3*((2 * 16 + 17 * 17 - 9) - 13)**2 + ((2 * (-3) + 17 * 28 - 9) - 42)**2 + ((2 * 14 + 17 * 85 - 9) + 39)**2)
print(MSE1, MSE2)
72.0 841323.6666666666
Ввод [7]: MSE1=1/3*(((-4 * 7 + 7 * 39 - 11) + 60)**2 + ((-4 * 12 + 7 * 48 - 11) - 17)**2 + ((-4 * 3 + 7 * 55 - 11) - 83)**2)
MSE2=1/3*((-0.5 * 7 + 9 * 39 - 400) + 60)**2 + ((-0.5 * 12 + 9 * 48 - 400) - 17)**2 + ((-0.5 * 3 + 9 * 55 - 400) - 83)**2)
print(MSE1, MSE2)
77292.33333333333 82.5
```

1. Первая функция имеет значение MSE равное 12.0, а вторая 84883.(6), следовательно, значение MSE меньше у первой функции.

2. Первая функция имеет значение MSE равное 72.0, а вторая 841323.(6), следовательно, значение MSE меньше у первой функции.

3. Первая функция имеет значение MSE равное 77292.(3), а вторая 82.5, следовательно, значение MSE меньше у второй функции.

Задание 2

```
Ввод [1]: from sympy import *
Ввод [2]: x1, x2, x3 = symbols('x1, x2, x3')
Ввод [3]: f=10*x1-5*x2
f
Out[3]: 10x1 - 5x2
Ввод [4]: diff(f, x1)
Out[4]: 10
Ввод [5]: diff(f, x2)
Out[5]: -5
```

```
Ввод [6]: f=3*x1+4*x2+7
f
Out[6]: 3x1 + 4x2 + 7
Ввод [7]: diff(f, x1)
Out[7]: 3
Ввод [8]: diff(f, x2)
Out[8]: 4
```

```
Ввод [9]: f=x1**2
f
Out[9]: x12
Ввод [10]: diff(f, x1)
Out[10]: 2x1
Ввод [11]: diff(f, x2)
Out[11]: 0
```

```
Ввод [12]: f=x1+5*x2-6*x3+3
f
Out[12]: x1 + 5x2 - 6x3 + 3
Ввод [13]: diff(f, x1)
Out[13]: 1
Ввод [14]: diff(f, x2)
Out[14]: 5
Ввод [15]: diff(f, x3)
Out[15]: -6
```

```
Ввод [17]: f=10*x1-x1**2+4*x1**3
```

```
f
```

```
Out[17]: 4x13 - x12 + 10x1
```

```
Ввод [18]: diff(f, x1)
```

```
Out[18]: 12x12 - 2x1 + 10
```

```
Ввод [19]: diff(f, x2)
```

```
Out[19]: 0
```

```
Ввод [20]: diff(f, x3)
```

```
Out[20]: 0
```

```
Ввод [21]: f=x1**2+12*x1*x2+4*x2**3+x3
```

```
f
```

```
Out[21]: x12 + 12x1x2 + 4x23 + x3
```

```
Ввод [22]: diff(f, x1)
```

```
Out[22]: 2x1 + 12x2
```

```
Ввод [23]: diff(f, x2)
```

```
Out[23]: 12x1 + 12x22
```

```
Ввод [24]: diff(f, x3)
```

```
Out[24]: 1
```

Задание 3

```
Ввод [1]: from sympy import *
```

```
Ввод [2]: a2, a1, a0 = symbols('a2, a1, a0')
```

```
f=1/3*((2*a2+200*a1+a0-200)**2+(a2+450*a1+a0-300)**2+(3*a2+550*a1+a0-600)**2)
```

```
f
```

```
Out[2]: 0.33333333333333(a0 + 200a1 + 2a2 - 200)2 + 0.33333333333333(a0 + 450a1 + a2 - 300)2 + 0.33333333333333(a0 + 550a1 + 3a2 - 600)2
```

```
Ввод [3]: diff(f, a1)
```

```
Out[3]: 800.0a0 + 363333.33333333a1 + 1666.6666666667a2 - 336666.666666667
```

```
Ввод [4]: diff(f, a0)
```

```
Out[4]: 2.0a0 + 800.0a1 + 4.0a2 - 733.33333333333
```

Задание 4

```
Ввод [1]: from sympy import *
from sympy.plotting import plot3d
init_printing(use_unicode=False, wrap_line=False, no_global=True)
```

```
Ввод [2]: a2, a1, a0 = symbols('a2, a1, a0')
```

```
Ввод [3]: MSE = 1/3*((a2 * 2 + a1 * 200 + a0) - 200)**2 + ((a2 + a1 * 300 + a0) - 450)**2 + ((a2 * 3 + a1 * 600 + a0) - 550)**2
```

```
MSE
```

```
Out[3]: 0.33333333333333(a0 + 200a1 + 2a2 - 200)2 + 0.33333333333333(a0 + 300a1 + a2 - 450)2 + 0.33333333333333(a0 + 600a1 + 3a2 - 550)2
```

```
Ввод [6]: MSEa2=diff(MSE, a2)
MSEa1=diff(MSE, a1)
MSEa0=diff(MSE, a0)
```

```
Ввод [7]: nonlinsolve([MSEa2, MSEa1, MSEa0], [a2, a1, a0])
```

```
Out[7]: {(-130.0, 1.2, 220.0)}
```

```
Ввод [10]: MSE.subs({a2:-130, a1:1.2, a0:220})
```

```
Out[10]: 0
```

```
Ввод [11]: f = -130 * 4 + 1.2 * 666 + 220
```

```
f
```

```
Out[11]: 499.2
```

Значение MSE получилось равным нулю. Значение площади дома, которое было известно на уроке, составляло 500 м^2 , а полученное в результате вычислений составило 499.2 м^2 , что меньше всего лишь на 0.8 м^2 .

1.2 Выводы

В ходе выполнения лабораторной работы я научился находить значение MSE для функции нескольких переменных, попрактиковался в нахождении частной производной и узнал о том, как находить производную с помощью SymPy.

1 Лабораторная работа №8

1.1 Основная часть

1.1.1 Теоретическая часть

Функцию $f(x_1, x_2, \dots, x_n) = a_n x_n + a_{n-1} x_{n-1} + \dots + a_1 x_1$ можно записать как произведение вектор-строки из коэффициентов на вектор-строку из переменных.

Тогда получим $f(\vec{x}) = \vec{a} \cdot \vec{x}$.

Запишем некоторые операции над вектор-строками(столбцами).

Сложение:

$$\vec{c} = \vec{a} + \vec{b} = (a_1 + b_1 \ a_2 + b_2 \ a_3 + b_3 \ \dots \ a_n + b_n)$$

Умножение на число:

$$\lambda(a_1, a_2, \dots, a_n) = (\lambda a_1, \lambda a_2, \dots, \lambda a_n)$$

Скалярное произведение:

$$\vec{x} \cdot \vec{a} = \sum_{i=1}^n a_i \cdot x_i = a_1 x_1 + a_2 x_2 + \dots + a_n x_n$$

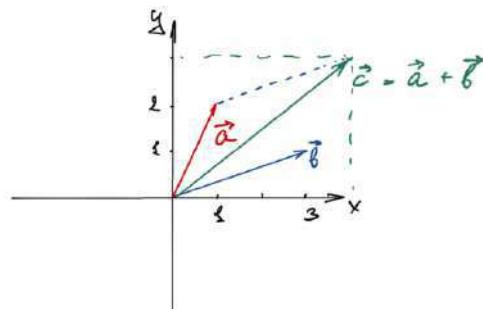
Несколько вектор-строк(столбцов) образуют матрицу. Она имеет такой общий вид:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & \dots & \dots & a_{2m} \\ \vdots & & & \vdots \\ a_{n1} & \dots & \dots & a_{nm} \end{pmatrix}_{n \times m}$$

Операция замены вектор-строк на вектор-столбцы и наоборот называется транспонированием:

$$\begin{pmatrix} 200 & 2 & 200 \\ 450 & 1 & 300 \\ 550 & 3 & 600 \end{pmatrix}^T = \begin{pmatrix} 200 & 450 & 550 \\ 2 & 1 & 3 \\ 200 & 300 & 600 \end{pmatrix}$$

Двумерные и трёхмерные вектор-строки имеют геометрическое представление, это вектор с началом в точке отсчёта и концом в точке с соответствующими координатами. Их можно складывать по правилу параллелограмма:



Градиентом функции потерь называется вектор-строка из частных производных всех её переменных:

$$\nabla MSE = \left(\frac{\partial MSE}{\partial x_1} \quad \frac{\partial MSE}{\partial x_2} \right)$$

Градиент всегда указывает направление скорейшего роста функции, а антиградиент ($-\nabla MSE$) - убывания. Так можно искать точки минимума функции потерь.

1.1.2 Полученные результаты и их анализ

Задание 1

1. $(1 \ 2 \ 3 \ 4) + (5 \ 6 \ 7 \ 8 \ 9) -$
операции не имеют
смысла

$$2. (1 \ 2 \ 3 \ 4) + (10 \ 11 \ 12 \ 13) =$$

$$= (1+10 \ 2+11 \ 3+12 \ 4+13) =$$

$$= (11 \ 13 \ 15 \ 17)$$

3. $\begin{pmatrix} 15 \\ 16 \\ 17 \\ 18 \\ 19 \end{pmatrix} + \begin{pmatrix} 3 \\ 3 \\ 3 \\ 3 \\ 3 \end{pmatrix} = \begin{pmatrix} 15+3 \\ 16+3 \\ 17+3 \\ 18+3 \\ 19+3 \end{pmatrix} = \begin{pmatrix} 18 \\ 19 \\ 20 \\ 21 \\ 22 \end{pmatrix}$

4. $3. (1 \ 2 \ 3 \ 4) + \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} -$ операции
не имеют
смысла

5. $\begin{pmatrix} 15 \\ 16 \\ 17 \\ 18 \\ 19 \\ 20 \\ 6+1 \end{pmatrix} + \begin{pmatrix} 3 \\ 3 \\ 3 \\ 3 \\ 3 \end{pmatrix} -$ операции
не имеют
смысла

$$1. 5 \cdot (1 \ 2 \ 3 \ 4 \ 5 \ 6) - 3 \cdot (5 \ 6 \ 7 \ 8 \ 9 \ 10) = \\ = (5 \ 10 \ 15 \ 20 \ 25 \ 30) - (15 \ 18 \ 21 \ 24 \ 27 \ 30) = \\ = (-10 \ -8 \ -6 \ -4 \ -2 \ 0)$$

$$2. 12 \cdot (7 \ 12 \ 18 \ 14 \ 9 \ 16 \ 21) - \\ - 3 \cdot 2 \cdot (13 \ 6 \ 1 \ 24 \ 76 \ 1 \ 38) = \\ = (84 \ 144 \ 132 \ 168 \ 108 \ 192 \ 252) - \\ - (41,6 \ 195,2 \ 76,8 \ 243,2 \ 3,2 \ 9,6 \ 25,6) \\ = (42,4 \ -51,2 \ 55,2 \ -75,2 \ 104,8 \ 182,4 \ 26,4)$$

$$3. 7 \cdot \begin{pmatrix} 15 \\ 16 \\ 14 \\ 18 \\ 19 \end{pmatrix} + \frac{1}{3} \cdot \begin{pmatrix} 3 \\ 3 \\ 3 \\ 3 \\ 3 \end{pmatrix} = \begin{pmatrix} 105 \\ 112 \\ 119 \\ 126 \\ 133 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \\ = \begin{pmatrix} 106 \\ 113 \\ 120 \\ 127 \\ 134 \end{pmatrix}$$

$$4. 7 \cdot \begin{pmatrix} 11 \\ 21 \\ 78 \\ 32 \\ 2 \end{pmatrix} - \frac{2}{5} \cdot \begin{pmatrix} 5 \\ 6 \\ 7 \\ 9 \\ 3 \end{pmatrix} = \\ = \begin{pmatrix} 77 \\ 147 \\ 546 \\ 224 \\ 14 \end{pmatrix} - \begin{pmatrix} 2 \\ 2,4 \\ 2,8 \\ 3,6 \\ 1,2 \end{pmatrix} = \begin{pmatrix} 75 \\ 144,6 \\ 543,2 \\ 220,4 \\ 12,8 \end{pmatrix}$$

Задание 2

$$\begin{aligned}
 1. & (1 \ 2 \ 3 \ 4 \ 5) \times (5 \ 6 \ 7 \ 8 \ 9) = \\
 & = 1 \cdot 5 + 2 \cdot 6 + 3 \cdot 7 + 4 \cdot 8 + 5 \cdot 9 = \\
 & = 5 + 12 + 21 + 32 + 45 = \underline{\underline{115}} \\
 2. & (4 \ 3 \ 8 \ 12 \ 1) \times (3 \ 2 \ 13 \ 8 \ 5) = \\
 & = 4 \cdot 3 + 3 \cdot 2 + 8 \cdot 13 + 12 \cdot 8 + 1 \cdot 5 = \\
 & = 12 + 6 + 104 + 96 + 5 = \underline{\underline{223}}
 \end{aligned}$$

3. $(1 \ 2 \ 3 \ 4) + (5 \ 6 \ 7 \ 8 \ 9)$ - one row and
 $\begin{matrix} & \\ 1 & 4 & & & \\ & & & & \end{matrix}$ $\begin{matrix} & \\ & & & & \\ & & & & \end{matrix}$ - one column
 $\begin{matrix} & \\ & & & & \\ & & & & \end{matrix}$ called

1. $(5 \ 6 \ 7 \ 8 \ 9)^T = \begin{pmatrix} 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{pmatrix}$

2. $\begin{pmatrix} 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{pmatrix}^T = (5 \ 6 \ 7 \ 8 \ 9)$

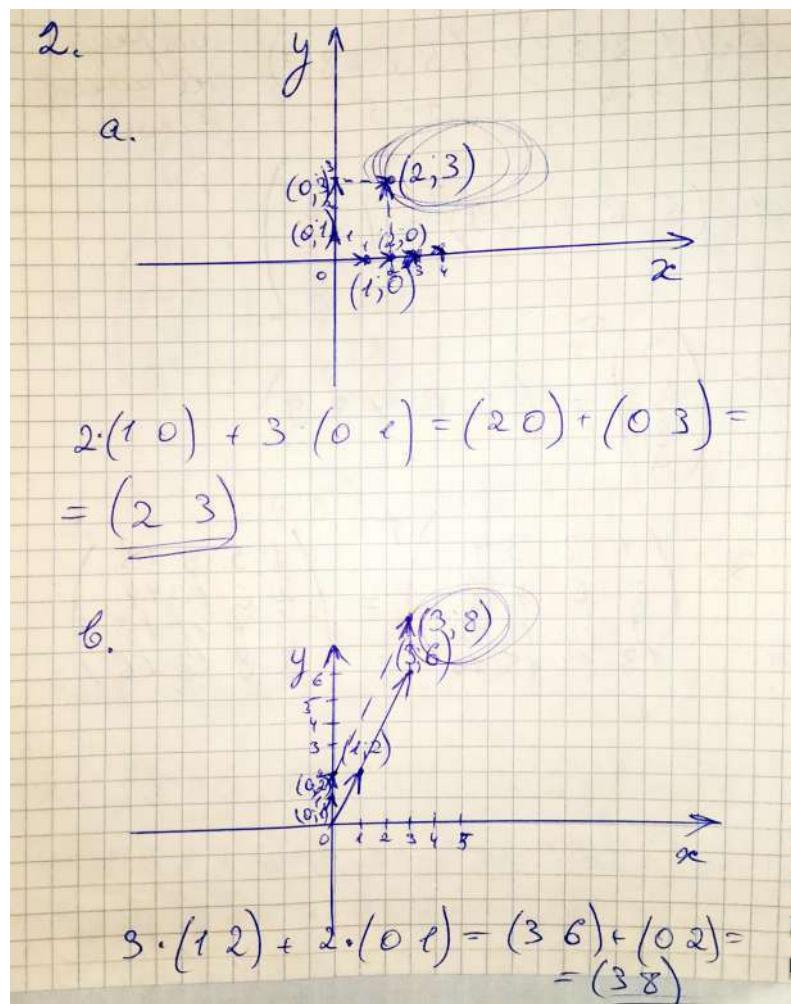
3. $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}^T = \begin{pmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 4 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{pmatrix}$

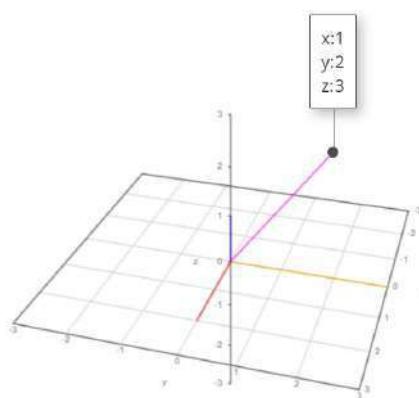
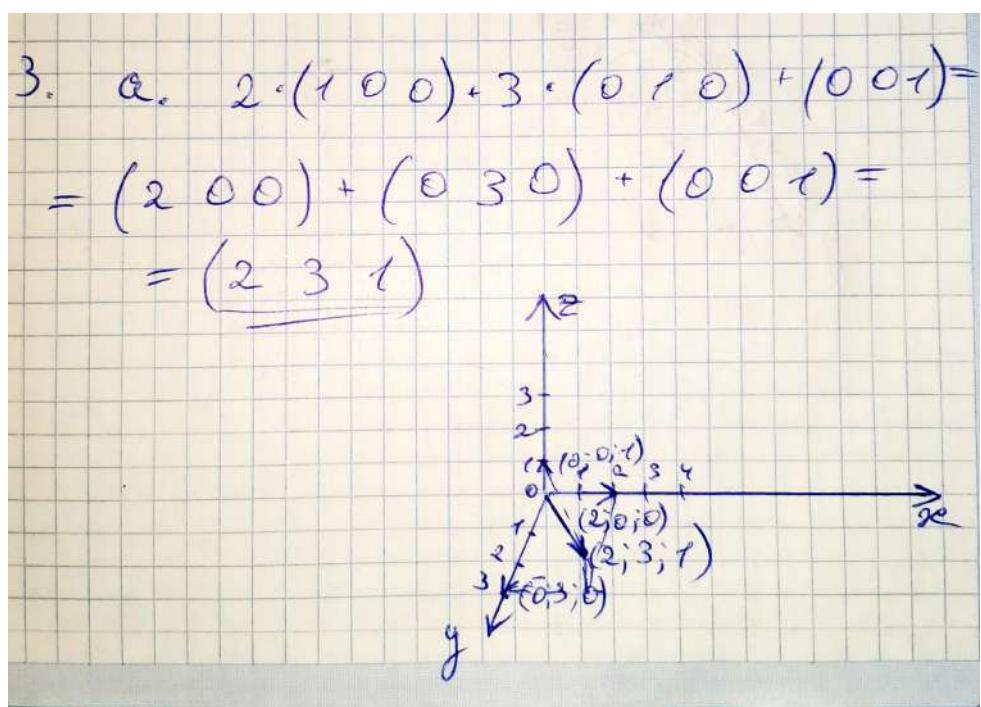
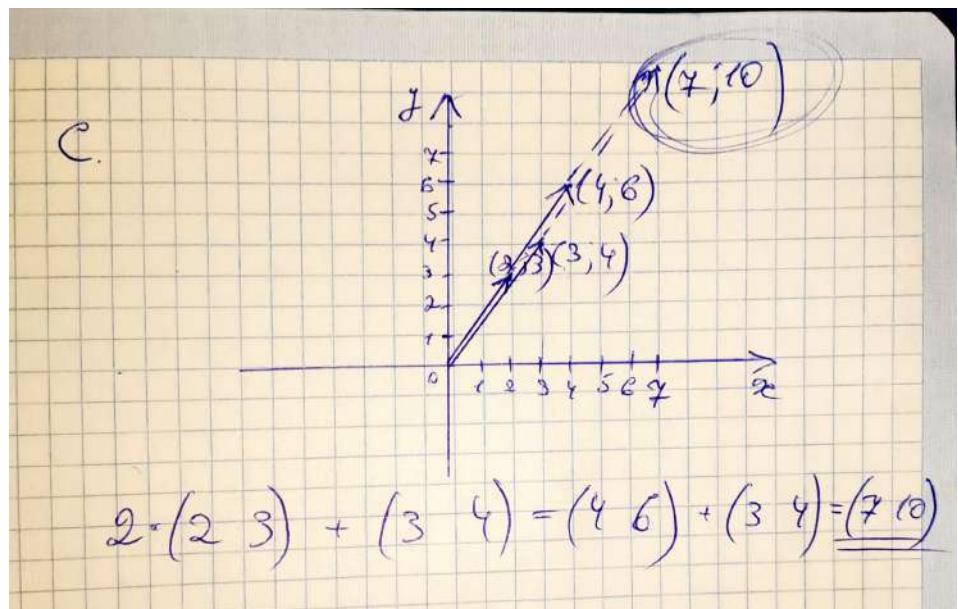
Задание 3

```
Ввод [5]: A = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]]
for i in range(len(A)):
    for j in range(len(A[0])):
        print(A[i][j], ' '* (3 - len(str(A[i][j]))), end='')
    print('\n')
1   2   3   4
5   6   7   8
9   10  11  12
13  14  15  16
```

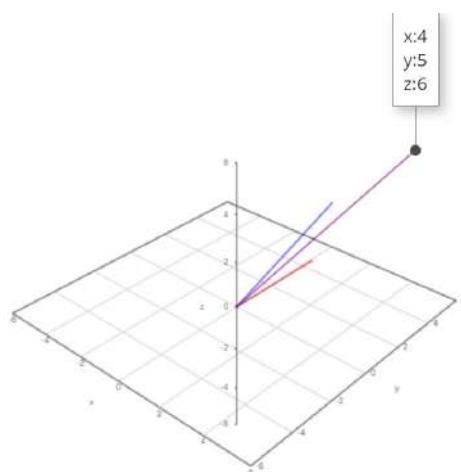
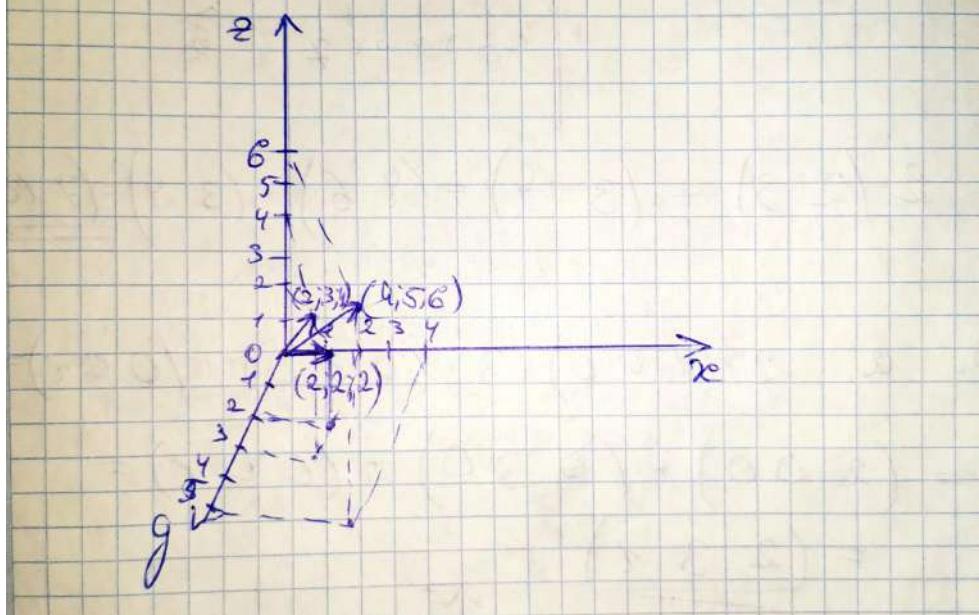
```
Ввод [8]: A = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]]
for i in range(len(A[0])):
    for j in range(len(A)):
        print(A[j][i], ' '* (3 - len(str(A[j][i]))), end='')
    print('\n')
1   5   9   13
2   6   10  14
3   7   11  15
4   8   12  16
```

Также можно вместо вывода матрицы производить запись в новый массив(что реализовано в функции в 4ом задании). Двумерные массивы аналогичны матрицам, где каждый вложенный массив - строка.





$$\begin{aligned}
 6. \quad & (2 \ 3 \ 4) + 2 \cdot (1 \ 1 \ 1) = \\
 & = (2 \ 3 \ 4) + (2 \ 2 \ 2) = (4 \ 5 \ 6)
 \end{aligned}$$



Задание 4

$$\begin{aligned}
 & 1. 3 \times (25068106757) \cdot \\
 & 2 \times (8679662323) = \\
 & = (615018243018211521) \cdot
 \end{aligned}$$

$$\begin{aligned}
 & (1612141812124949) = \\
 & = 6 \cdot 16 + 15 \cdot 12 + 0 \cdot 14 + 18 \cdot 18 + 24 \cdot 12 + \\
 & + 30 \cdot 12 + 18 \cdot 4 + 21 \cdot 9 + 15 \cdot 4 + 21 \cdot 9 = \\
 & = \underline{\underline{1758}} \\
 & 2. 6 \times (9677016812) \cdot \\
 & 5 \times (0226788318) = \\
 & = (54364242063648612) \cdot \\
 & (010103035404015540) = \\
 & = 54 \cdot 0 + 36 \cdot 10 + 42 \cdot 10 + 42 \cdot 30 + 0 \cdot 35 + \\
 & + 6 \cdot 40 + 36 \cdot 40 + 48 \cdot 15 + 6 \cdot 5 + 12 \cdot 40 = \\
 & = \underline{\underline{4950}}
 \end{aligned}$$

$$\begin{aligned}
 & 3. (2 \ 5 \ 0 \ 6 \ 8 \ 10 \ 6 \ 7 \ 5 \ 2) \cdot \\
 & (8 \ 6 \ 7 \ 9 \ 6 \ 6 \ 23 \ 23) = \\
 & = 2 \cdot 8 + 5 \cdot 6 + 0 \cdot 7 + 6 \cdot 9 + 8 \cdot 6 + 10 \cdot 6 + \\
 & + 6 \cdot 2 + 7 \cdot 3 + 5 \cdot 2 + 7 \cdot 3 = \underline{\underline{272}}
 \end{aligned}$$

$$\begin{aligned}
 & 4. (9 \ 6 \ 7 \ 7 \ 0 \ 1 \ 6 \ 8 \ 12) \cdot \\
 & (0 \ 2 \ 2 \ 6 \ 7 \ 8 \ 8 \ 3 \ 18) = \\
 & = 9 \cdot 0 + 6 \cdot 2 + 7 \cdot 2 + 7 \cdot 6 + 0 \cdot 7 + \\
 & + 1 \cdot 8 + 6 \cdot 8 + 8 \cdot 3 + 1 \cdot 1 + 2 \cdot 8 = \\
 & = \underline{\underline{165}}
 \end{aligned}$$

Листинг 1: Текст функции

```

1 def T(A):
2     B = []
3     for i in range(len(A[0])):
4         B.append([])
5         for j in range(len(A)):
6             B[-1].append(A[j][i])
7     return B

```

```

Ввод [8]: def T(A):
    B = []
    for i in range(len(A[0])):
        B.append([])
        for j in range(len(A)):
            B[-1].append(A[j][i])
    return B

Ввод [9]: A1=[[2, 1, 7, 4], [5, 6, 7, 3], [9, 8, 2, 12], [11, 14, 15, 15]]
print(T(A1))
[[2, 5, 9, 11], [1, 6, 8, 14], [7, 7, 2, 15], [4, 3, 12, 15]]

Ввод [10]: A2=[[3, 7, 8, 3, 6], [2, 5, 9, 4, 13]]
print(T(A2))
[[3, 2], [7, 5], [8, 9], [3, 4], [6, 13]]

```

```

Ввод [16]: a1=0.395
           a2=0.62
           count=1
           MSE=1/4*((a1 + 2*a2 - 5)**2 + (5*a1 + 3*a2 - 6)**2 + (2*a1 + 4*a2 - 10)**2 + (3*a1 + 7*a2 - 8)**2)
           print(count, ' ', a1, ' ', a2, ' ', MSE, sep='')
           while(MSE>6.36):
               delta_a1=-0.01*(19.5*a1 + 23*a2 - 39.5)
               delta_a2=-0.01*(23*a1 + 39*a2 - 62)
               a1+=delta_a1
               a2+=delta_a2
               count+=1
               MSE=1/4*((a1 + 2*a2 - 5)**2 + (5*a1 + 3*a2 - 6)**2 + (2*a1 + 4*a2 - 10)**2 + (3*a1 + 7*a2 - 8)**2)
               print(count, ' ', a1, ' ', a2, ' ', MSE, sep='')

1. 0.395 0.62 16.85724375
2. 0.570375000000001 0.90735 8.593654828593749
3. 0.645461375000001 1.042297249999998 6.851914496792685
4. 0.674868039375001 1.107345206249998 6.477251488262853
5. 0.6835793742593751 1.1492699267562498 6.3897803764667005
6. 0.6830213831248595 1.158335909241656 6.363220554275257
7. 0.6784149542899309 1.1694899865186925 6.350108490117965

```

Всего потребовалось 7 шагов градиентного спуска, чтобы добиться значения MSE меньше 6.36.

1.2 Выводы

В ходе выполнения лабораторной работы я научился умножать матрицу на вектор, находить градиент функции потерь и искать с помощью него минимальное значение MSE.

1 Лабораторная работа №9

1.1 Основная часть

1.1.1 Теоретическая часть

Описать задачу аппроксимации или интерполяции можно с помощью системы линейных уравнений(СЛАУ). А сами СЛАУ удобно записывать как матричное уравнение такого вида:

$$X\vec{a} = \vec{b}$$

Вектор весов
Матрица объектов-признаков
Вектор ответов

, где $X\vec{a}$ имеет вид:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & \dots & \dots & a_{2m} \\ \vdots & & & \vdots \\ a_{n1} & \dots & \dots & a_{nm} \end{pmatrix}_{n \times m} \times \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}_{m \times 1} = \begin{pmatrix} a_{11}b_1 & a_{12}b_2 & \dots & a_{1m}b_m \\ a_{21}b_1 & \dots & \dots & a_{2m}b_m \\ \vdots & & & \vdots \\ a_{n1}b_1 & \dots & \dots & a_{nm}b_m \end{pmatrix}$$

Для записи матриц в symPy используется объект `Matrix()`, в который записывается двумерный массив. А для решения СЛАУ можно использовать функцию `linsolve()`, для этого необходимо сообщить ей расширенную матрицу(она дополнительно содержит в себе вектор-столбец ответов).

Может возникнуть ситуация, когда уравнений в системе больше, чем неизвестных. Такая система называется переопределённой. В такой СЛАУ нельзя подобрать такие значения переменных, чтобы левые части были равны правым во всех уравнениях. Поэтому мы можем добиться только максимально близких значений, минимизировав значение MSE, а сделать это можно с помощью градиента уже известным способом.

1.1.2 Полученные результаты и их анализ

Задание 1

$$1. \begin{cases} 2x + 5y = 11, \\ 7x - 3y = -23; \end{cases}$$

$$\begin{cases} x = \frac{11 - 5y}{2}, \\ 7x - 3y = -23 \end{cases}$$

$$\frac{7y - 35y}{2} - 3y = -23 \mid \cdot 2$$

$$7y - 35y - 6y = -46$$

$$-44y = -123$$

$$y = 3$$

$$x = \frac{11 - 15}{2} = -2$$

$$\text{Ответ: } x = -2, y = 3$$

$$2. \begin{cases} -3x + y = -2, \\ + \begin{cases} 3x + 5y = 8; \end{cases} \end{cases}$$

$$6y = 6$$

$$y = 1$$

$$-3x + 1 = -2$$

$$-3x = -3$$

$$x = 1$$

Ortsvektor: $x = 1, y = 1$

$$3. \begin{cases} 2x + 3y = 12, \\ 3x - y = 7; \end{cases}$$

$$\begin{cases} x = \frac{y+7}{3}, \\ 2x + 3y = 12; \end{cases}$$

$$\frac{14+2y}{3} + 3y = 12 \mid \cdot 3$$

$$14+2y+9y=36$$

$$11y=22$$

$$y=2$$

$$x = \frac{y+2}{3} = 3$$

Onsatz: $x=3, y=2$

$$4 \cdot + \begin{cases} x+y+2z=-1 \\ 2x-y+2z=-4 \\ 4x+y+4z=-2 \end{cases}$$

$$- \begin{cases} 3x+4z=-5 \\ 4x+y+4z=-2 \end{cases}$$

$$x+y=3$$

$$x=3-y$$

$$12 - 4y + y + 4z = -2$$

$$-3y + 4z = -14$$

$$z = \frac{-14 + 3y}{4}$$

$$3 - y + y + 2 \cdot \frac{-14 + 3y}{4} = -1$$

$$3 + \frac{-14 + 3y}{2} = -11.2$$

$$6 - 14 + 3y = -2$$

$$3y = 6$$

$$y = 2$$

$$x = 3 - 2 = 1$$

$$z = \frac{-14 + 6}{4} = -2$$

Umkehr: $x = 1, y = 2, z = -2$

Задание 2

$$\begin{aligned}
 1. & \begin{cases} 2x + 5y = 11, \\ 7x - 3y = -23; \end{cases} \Rightarrow \begin{pmatrix} 2 & 5 \\ 7 & -3 \end{pmatrix} \times \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 11 \\ -23 \end{pmatrix} \\
 2. & \begin{cases} -3x + y = -2, \\ 3x + 5y = 8; \end{cases} \Rightarrow \begin{pmatrix} -3 & 1 \\ 3 & 5 \end{pmatrix} \times \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -2 \\ 8 \end{pmatrix} \\
 3. & \begin{cases} 2a + 3b = 12, \\ 3a - b = 7; \end{cases} \Rightarrow \begin{pmatrix} 2 & 3 \\ 3 & -1 \end{pmatrix} \times \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 12 \\ 7 \end{pmatrix} \\
 4. & \begin{cases} x_1 + x_2 + 2x_3 = -1, \\ 2x_1 - x_2 + 2x_3 = -4, \\ 4x_1 + x_2 + 4x_3 = -2; \end{cases} \Rightarrow \begin{pmatrix} 1 & 1 & 2 \\ 2 & -1 & 2 \\ 4 & 1 & 4 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -1 \\ -4 \\ -2 \end{pmatrix} \\
 1. & \begin{pmatrix} 2 & 5 \\ 7 & -3 \end{pmatrix} \times \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 2 \cdot 2 & 5 \cdot 3 \\ 7 \cdot 2 & -3 \cdot 3 \end{pmatrix} = \begin{pmatrix} 4 & 15 \\ 14 & -9 \end{pmatrix} \\
 2. & \begin{pmatrix} -3 & 1 \\ 3 & 5 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} -3 \cdot 1 & 1 \cdot 3 \\ 3 \cdot 1 & 5 \cdot 3 \end{pmatrix} = \begin{pmatrix} -3 & 3 \\ 3 & 15 \end{pmatrix}
 \end{aligned}$$

$$\begin{aligned}
 3. & \begin{pmatrix} 2 & 3 \\ 3 & -1 \end{pmatrix} \times \begin{pmatrix} 4 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \cdot 4 & 3 \cdot 2 \\ 3 \cdot 4 & -1 \cdot 2 \end{pmatrix} = \begin{pmatrix} 18 & 6 \\ 12 & -2 \end{pmatrix} \\
 4. & \begin{pmatrix} 1 & 1 & 2 \\ 2 & -1 & 2 \\ 4 & 1 & 4 \end{pmatrix} \times \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix} = \begin{pmatrix} 1 \cdot 2 & 1 \cdot 3 & 2 \cdot 5 \\ 2 \cdot 2 & -1 \cdot 3 & 2 \cdot 5 \\ 4 \cdot 2 & 1 \cdot 3 & 4 \cdot 5 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 10 \\ 4 & -3 & 10 \\ 8 & 3 & 20 \end{pmatrix}
 \end{aligned}$$

Задание 3

$$1. \begin{cases} -x + 7y = -34, \\ 8x + 8y = -48; \end{cases}$$

$$\begin{cases} x = 34 + 7y, \\ 8x + 8y = -48 \end{cases}$$

$$272 + 56y + 8y = -48$$

$$64y = -320$$

$$y = -5 \quad x = 34 - 35 = -1$$

Ответ: $x = -1, y = -5$

$$2. \quad - \begin{cases} 4x - 7y = -4, \\ 3x - 4y = -3; \end{cases}$$

$$x \cancel{+} 9y + -3y = -1$$

$$x = -1 + 3y$$

$$-3 + 9y - 4y = -3$$

$$y = 0 \quad x = -1$$

Ortsvektor: $x = -1, y = 0$

$$3. \quad \begin{cases} 8a - 4b = 64, \\ + \quad -3a + 3b = -21; \end{cases}$$

$$5a - b = 43$$

$$b = -43 + 5a$$

$$-3a - 129 + 15a = -21$$

$$12a = 108$$

$$\alpha = 9 \quad b = -43 + 45 = 2$$

OnGem: $\alpha = 9, b = 2$

$$4. \begin{cases} 5x_1 + 7x_2 - 5x_3 = -47, \\ -2x_2 + 2x_3 = 10, \textcircled{1} \\ -4x_1 - 8x_2 - 7x_3 = 63; \end{cases}$$

$$\textcircled{1} \quad x_3 = \frac{10 + 2x_2}{2} = 5 + x_2$$

$$\begin{cases} 5x_1 + 7x_2 - 25 - 5x_2 = -47, \\ -4x_1 - 8x_2 - 35 - 7x_2 = 63; \end{cases}$$

$$\begin{cases} x_2 = \frac{-22 - 5x_1}{2}, \\ -4x_1 - 15x_2 = 98; \end{cases}$$

$$-4x_1 - \frac{-330 - 75x_1}{2} = 98 / \cdot 2$$

$$-8x_1 + 330 + 75x_1 = 196$$

$$67x_1 = -139$$

$$x_1 = -2$$

$$x_2 = \frac{-22 + 10}{2} = -6$$

$$x_3 = \frac{10 - 12}{2} = -1$$

Ответ: $x_1 = -2, x_2 = -6, x_3 = -1$

```
Ввод [1]: from sympy import *
```

```
Ввод [2]: X_aug = Matrix([[-1, 7, -34], [8, 8, -48]])
X_aug
```

```
Out[2]:  $\begin{bmatrix} -1 & 7 & -34 \\ 8 & 8 & -48 \end{bmatrix}$ 
```

```
Ввод [4]: x, y, z, a, b = symbols('x, y, z, a, b')
linsolve(X_aug, [x, y])
```

```
Out[4]: {(-1, -5)}
```

```
Ввод [5]: X_aug = Matrix([[4, -7, -4], [3, -4, -3]])
X_aug
```

```
Out[5]:  $\begin{bmatrix} 4 & -7 & -4 \\ 3 & -4 & -3 \end{bmatrix}$ 
```

```
Ввод [6]: linsolve(X_aug, [x, y])
```

```
Out[6]: {(-1, 0)}
```

```
Ввод [7]: X_aug = Matrix([[8, -4, 64], [-3, 3, -21]])
X_aug
```

$$\text{Out[7]: } \begin{bmatrix} 8 & -4 & 64 \\ -3 & 3 & -21 \end{bmatrix}$$

```
Ввод [8]: linsolve(X_aug, [a, b])
```

$$\text{Out[8]: } \{(9, 2)\}$$

```
Ввод [9]: X_aug = Matrix([[5, 7, -5, -47], [0, -2, 2, 10], [-4, -8, -7, 63]])
X_aug
```

$$\text{Out[9]: } \begin{bmatrix} 5 & 7 & -5 & -47 \\ 0 & -2 & 2 & 10 \\ -4 & -8 & -7 & 63 \end{bmatrix}$$

```
Ввод [10]: linsolve(X_aug, [x, y, z])
```

$$\text{Out[10]: } \{(-2, -6, -1)\}$$

$$1. \begin{pmatrix} -1 & 7 \\ 8 & 8 \end{pmatrix} \times \begin{pmatrix} -1 \\ -5 \end{pmatrix} = \begin{pmatrix} -1 \cdot (-1) & 7 \cdot (-5) \\ 8 \cdot (-1) & 8 \cdot (-5) \end{pmatrix} = \begin{pmatrix} 1 & -35 \\ -8 & -40 \end{pmatrix}$$

$$2. \begin{pmatrix} 4 & -7 \\ 3 & -4 \end{pmatrix} \times \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{pmatrix} 4 \cdot (-1) & -7 \cdot 0 \\ 3 \cdot (-1) & -4 \cdot 0 \end{pmatrix} = \begin{pmatrix} -4 & 0 \\ -3 & 0 \end{pmatrix}$$

$$3. \begin{pmatrix} 8 & -4 \\ -3 & 3 \end{pmatrix} \times \begin{pmatrix} 9 \\ 2 \end{pmatrix} = \begin{pmatrix} 8 \cdot 9 & -4 \cdot 2 \\ -3 \cdot 9 & 3 \cdot 2 \end{pmatrix} = \begin{pmatrix} 72 & -8 \\ -27 & 6 \end{pmatrix}$$

$$4. \begin{pmatrix} 5 & 7 & -5 \\ 0 & -2 & 2 \\ -4 & -8 & -7 \end{pmatrix} \times \begin{pmatrix} -2 \\ -6 \\ -1 \end{pmatrix} = \begin{pmatrix} 5 \cdot (-2) & 7 \cdot (-6) & -5 \cdot (-1) \\ 0 \cdot (-2) & -2 \cdot (-6) & 2 \cdot (-1) \\ -4 \cdot (-2) & -8 \cdot (-6) & -7 \cdot (-1) \end{pmatrix}$$

$$= \begin{pmatrix} -10 & -42 & 5 \\ 0 & 12 & -2 \\ 8 & 48 & 7 \end{pmatrix}$$

```
Ввод [12]: X=Matrix([[-1, 7], [8, 8]])
```

```
X
```

```
Out[12]:
```

$$\begin{bmatrix} -1 & 7 \\ 8 & 8 \end{bmatrix}$$

```
Ввод [13]: V=Matrix([-1, -5])
```

```
V
```

```
Out[13]:
```

$$\begin{bmatrix} -1 \\ -5 \end{bmatrix}$$

```
Ввод [14]: X*V
```

```
X
```

```
Out[14]:
```

$$\begin{bmatrix} -34 \\ -48 \end{bmatrix}$$

```
Ввод [15]: X=Matrix([[4, -7], [3, -4]])
```

```
X
```

```
Out[15]:
```

$$\begin{bmatrix} 4 & -7 \\ 3 & -4 \end{bmatrix}$$

```
Ввод [16]: V=Matrix([-1, 0])
```

```
V
```

```
Out[16]:
```

$$\begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

```
Ввод [17]: X*V
```

```
X
```

```
Out[17]:
```

$$\begin{bmatrix} -4 \\ -3 \end{bmatrix}$$

```
Ввод [18]: X=Matrix([[8, -4], [-3, 3]])
```

```
X
```

```
Out[18]:
```

$$\begin{bmatrix} 8 & -4 \\ -3 & 3 \end{bmatrix}$$

```
Ввод [19]: V=Matrix([9, 2])
```

```
V
```

```
Out[19]:
```

$$\begin{bmatrix} 9 \\ 2 \end{bmatrix}$$

```
Ввод [20]: X*V
```

```
X
```

```
Out[20]:
```

$$\begin{bmatrix} 64 \\ -21 \end{bmatrix}$$

```
Ввод [22]: X=Matrix([[5, 7, -5], [0, -2, 2], [-4, -8, -7]])
```

```
X
```

```
Out[22]:
```

$$\begin{bmatrix} 5 & 7 & -5 \\ 0 & -2 & 2 \\ -4 & -8 & -7 \end{bmatrix}$$

```
Ввод [23]: V=Matrix([-2, -6, -1])
```

```
V
```

```
Out[23]:
```

$$\begin{bmatrix} -2 \\ -6 \\ -1 \end{bmatrix}$$

```
Ввод [24]: X*V
```

```
Out[24]:
```

$$\begin{bmatrix} -47 \\ 10 \\ 63 \end{bmatrix}$$

В результатах, полученных с помощью SymPy, ответом является вектор-столбец из сумм элементов строк, полученных ручным способом.

```
Ввод [26]: x, y, z = symbols('x, y, z')
MSE=1/6*((5*x + 7*y - 5*z + 47)**2 + (0*x - 2*y + 2*z - 10)**2 + (-4*x - 8*y - 7*z - 63)**2 + (x + y + 2*z + 1)**2 + (2*x - y + z)**2)
dx, dy, dz = diff(MSE, x), diff(MSE, y), diff(MSE, z)
print(dx, dy, dz, sep='\n')
```

```
20.666666666666667*x + 23.33333333333333*y + 8.33333333333333*z + 168.0
23.333333333333*x + 40.0*y + 6.99999999999999*z + 284.0
8.333333333333*x + 6.99999999999999*y + 34.0*z + 68.0
```

```
Ввод [29]: x, y, z = 0, 0, 0
count=1
MSE=1/6*((5*x + 7*y - 5*z + 47)**2 + (0*x - 2*y + 2*z - 10)**2 + (-4*x - 8*y - 7*z - 63)**2 + (x + y + 2*z + 1)**2 + (2*x - y + z)**2)
print(count, '. x:', x, ' y:', y, ' z:', z, ' MSE=', MSE, sep=' ')
while(MSE>55):
    delta_x=-0.01*(20.666666666666667*x + 23.33333333333333*y + 8.33333333333333*z + 168.0)
    delta_y=-0.01*(23.333333333333*x + 40.0*y + 6.99999999999999*z + 284.0)
    delta_z=-0.01*(8.333333333333*x + 6.99999999999999*y + 34.0*z + 68.0)
    x+=delta_x
    y+=delta_y
    z+=delta_z
    count+=1
    MSE=1/6*((5*x + 7*y - 5*z + 47)**2 + (0*x - 2*y + 2*z - 10)**2 + (-4*x - 8*y - 7*z - 63)**2 + (x + y + 2*z + 1)**2 + (2*x - y + z)**2)
    print(count, '. x:', x, ' y:', y, ' z:', z, ' MSE=', MSE, sep=' ')
1. x:0 y:0 z:0 MSE=1049.833333333333
2. x:-1.68 y:-2.84 z:-0.68 MSE=247.497333333333
3. x:-2.293466666666667 y:-4.1044 z:-0.790000000000004 MSE=104.48600507259258
4. x:-2.4759568888888897 y:-4.712197777777777 z:-0.7229697777777785 MSE=73.77443515037976
5. x:-2.4844988355555566 y:-5.038987508148149 z:-0.620976468148149 MSE=64.18571208511977
6. x:-2.423523951960495 y:-5.246207757155557 z:-0.538073773777788 MSE=59.55918818381417
7. x:-2.3557743774042157 y:-5.381530568004775 z:-0.4610738183624044 MSE=56.568654549593
8. x:-2.238934388676031 y:-5.49162915212318 z:-0.4129537155751685 MSE=54.31880450271997
```

Получились следующие значения: $x \approx -2.24$, $y \approx -5.49$, $z \approx -0.41$

1.2 Выводы

В ходе выполнения лабораторной работы я научился записывать СЛАУ в виде матричного уравнения и матрицы в SymPy, умножать матрицу на вектор и решать переопределенные СЛАУ.