Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Воронежский государственный лесотехнический университет

имени Г.Ф. Морозова»

Факультет компьютерных наук и технологий

Кафедра Информационных технологий

(название кафедры)

ОТЧЕТ

о прохождении \_\_\_ \_ознакомительной\_\_\_\_\_ практики

(учебной, производственной, научно-исследовательской, преддипломной)

(на)\_\_\_кафедре Информационных технологий, ФКНиТ, ВГЛТУ\_\_\_\_\_\_\_\_\_\_

(полное наименование предприятия, учреждения организации, их структурного подразделения –

места прохождения практики)

Выполнил студент Сапожков Вадим Андреевич ИС2-242-ЗБ

Руководитель от ВГЛТУ\_ Анциферова В.И.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Сроки прохождения практики с « 27.06» 2025 по « 10.07» 2025

Отчет защищен «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члены комиссии: к.т.н, доцент, Анциферова В.И.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ученая степень, ученое звание, ФИО)

\_\_\_к.ф.-м.н., Ягодкин А.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ученая степень, ученое звание, ФИО)

Воронеж 2025

**План - График**

**Прохождения ознакомительной практики**

Студента группы **ИС2-242-ЗБ**

Сапожкова Вадима Андреевича

Место прохождения практики - кафедра Информационных технологий

|  |  |  |
| --- | --- | --- |
| № | Наименование мероприятий | Дата |
| 1 | Вводный инструктаж, инструктаж по технике безопасности на рабочем месте. Ознакомительная лекция по прохождению практики | 27.06.2025 |
| 2 | Получение задания по практике | 24.06.2025 |
| 3 | Изучения литературы по программированию | 25.06.2025 |
| 4 | Разработка алгоритма решение задачи | 26.06.2025 |
| 5 | Разработка алгоритма решение задачи | 27.06.2025 |
| 6 | Разработка отдельных модулей программы | 28.06.2025 |
| 7 | Разработка отдельных модулей программы | 29.06.2025 |
| 8 | Разработка общего программного приложения | 01.07.2025 |
| 9 | Отладка программного продукта | 02.07.2025 |
| 10 | Тестирование и отладка программного продукта | 03.07.2025 |
| 11 | Анализ результатов работы программы | 04.07.2025 |
| 12 | Оформления отчета прохождения ознакомительной практики | 10.07.2025 |
| 13 | Сдача отчета по практике | 10.07.2025 |

Реализовать следующую задачу:

1. Обеспечить в программу ввод данных с клавиатуры или из файла.

2. Реализовать вывод результатов в файл и на монитор.

3. Реализовать защиту данных и справочную информацию (по возможности).

4. Определить показатели качества созданного программного обеспечения.

5. Вычислительная часть задачи представлена вариантами.

# Описание создания программы

**1. Ввод данных**

Данные могут вводиться с клавиатуры и из файла, который может быть на жестком диске и на внешнем запоминающем устройстве.

При вводе данных с клавиатуры необходимо создать файл с начальными значениями. При этом самостоятельно выбрать директорию, где записать данный файл. Если необходимо, то создать новую директорию.

При вводе данных с диска нужно самостоятельно выбрать директорию, где находится данный файл, значения х от 1 до 15, значения по у, соответственно функции по варианту задания.

**2.Вывод результатов**

Результаты должны быть представлены в виде: а) графика, б) числовых значений. Они также должны быть записаны в виде файла. При этом самостоятельно выбрать директорию, где должен храниться данный файл.

Для данной программы обеспечить вывод графики в виде изменения функции y=f(x) с осями Х и Y. График помещается в отдельном окне.

**3. Обеспечить диспетчеризацию решения задачи**

Это управление всем процессом решения задачи.

Предусмотреть вывод информации.

**4. Защита данных и справочная информация**

Защита данных обеспечивается вводом пароля.

Справочная информация включает некоторые сведения о программе и об авторе разработки. В данную программу можно поместить свою фотографии, фамилию, имя и отчество. Можно указать год разработки.

**5. Показатели качества созданного программного обеспечения.**

Определить быстродействие и затраты памяти в зависимости от степени загрузки

**6. Вычислительная часть задачи.**

***Вариант 2***

Вводятся значения Х и соответствующие им значения Y. Необходимо построить функцию - степенную аппроксимацию данных значений по методу наименьших квадратов и отдельно вычислить среднеквадратичное отклонение от нее.

Y=B\*XK

**Задание выдал:**

**руководитель практики**

**к.т.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Анциферова В.И.**

(дата)

**Задание получил:**

**практикант, группы ИС2-242-ЗБ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сапожков В. А.** (дата)

АЛГОРИТМ

1. Авторизация:

Пользователь вводит пароль в отведенное для этого поле. Программа проверяет, совпадает ли введенный пароль с заданным значением.

* Если данные верны:

Окно авторизации закрывается, открывается основная форма программы.

* Если данные неверны: отображается сообщение об ошибке «Пароль инвалид».
* Если закрыть окно входа, программа также завершится

1. Ввод данных:

У пользователя есть возможность вводить значения X;Y в отведенные для этого текстовые поля.

После нажатия кнопки "Добавить" происходит: проверка корректности введенных данных.

* Если данные корректны:

Пара значений добавляется в список данных.

* Если данные некорректны: отображается сообщение об ошибке.

Альтернативно: Пользователю также предоставлена возможность загружать данные координат при помощи файла, нажав на копку «Открыть файл»: открывается диалоговое окно выбора файла.

* Если файл выбран успешно:

Программа считывает из файла значения, пары значений добавляются в список данных

Если возникла ошибка при открытии файла: отображается сообщение об ошибке «Ошибка» с текстом самой ошибки.

1. Расчет и отображение результатов:

При нажатии кнопки "Рассчитать" происходит: проверка наличия достаточного количества данных.

* Если данных достаточно:

вычисляются суммы, рассчитываются коэффициенты K и B линейной аппроксимации методом наименьших квадратов. Рассчитывается среднеквадратичное отклонение. Рассчитываются значения Y = B \* X^K для каждой точки. Результаты выводятся в отведенные текстовые поля.

Строится график: определяются минимальные и максимальные значения X и Y для установки пределов осей, создаются серии данных для отображения точек и аппроксимирующей прямой. Добавляются точки и прямая на график.

1. Очистка данных:

При нажатии кнопки "Очистить" происходит: очистка списка данных, очистка графика, очистка текстовых полей с результатами и полями ввода X и Y.

1. Разработчик

При наведении на текст «Разработчик» появляется подсказка о создателе данного проекта.

1. Сохранение результатов:

При нажатии кнопки "Сохранить результаты" происходит: сохранение результатов (K, B, среднеквадратичное отклонение, доверительный интервал, значения Y = B \* X^K) в текстовый файл (.txt), cохранение графика (.png), отображение сообщений об успешном сохранении «Результаты удачно сохранились в файл».

Выход из программы: при нажатии кнопки "Завершить" программа завершает работу.

ХОД РАБОТЫ

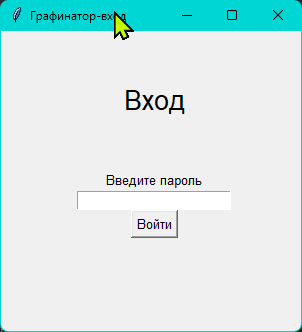
Для решения поставленной задачи была использована среда разработки PyCharm 2024.1.3 (Community Edition)

Рис 1. Запуск программы

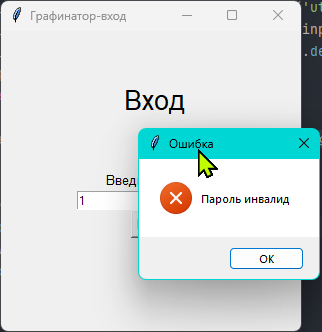
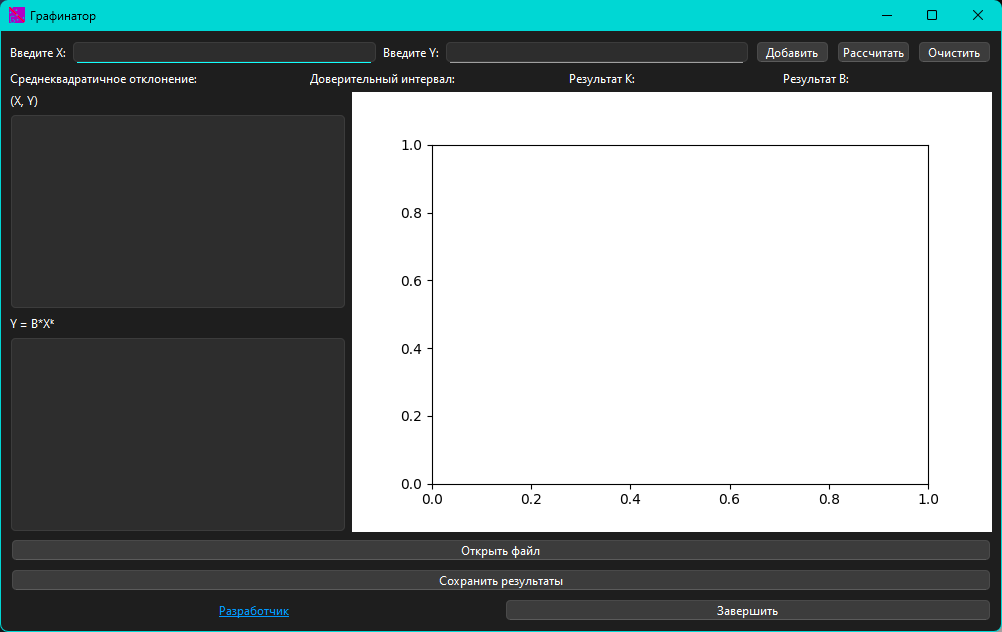
При запуске программы пользователю требуется ввести пароль. Пароль был реализован для защиты программы. Чтобы перейти к следующей форме нужно ввести мой номер студента из списка «50» и нажать кнопку «Войти». Если пароль введен неправильно, пользователь будет уведомлен об этом.

Рис 2. Сообщение о неправильно введенном логине и пароле.

Рис 3. Основное окно программы

Данные можно ввести как вручную, так и из файла. Так же пользователь может сохранить в файл подсчитанные значения. Для этого программа создает файл с параметрами, заданными пользователем, и записывает в него данные. Ниже на рисунках будут изображены результаты работы на основной форме с разными значениями, добавленными из файла.

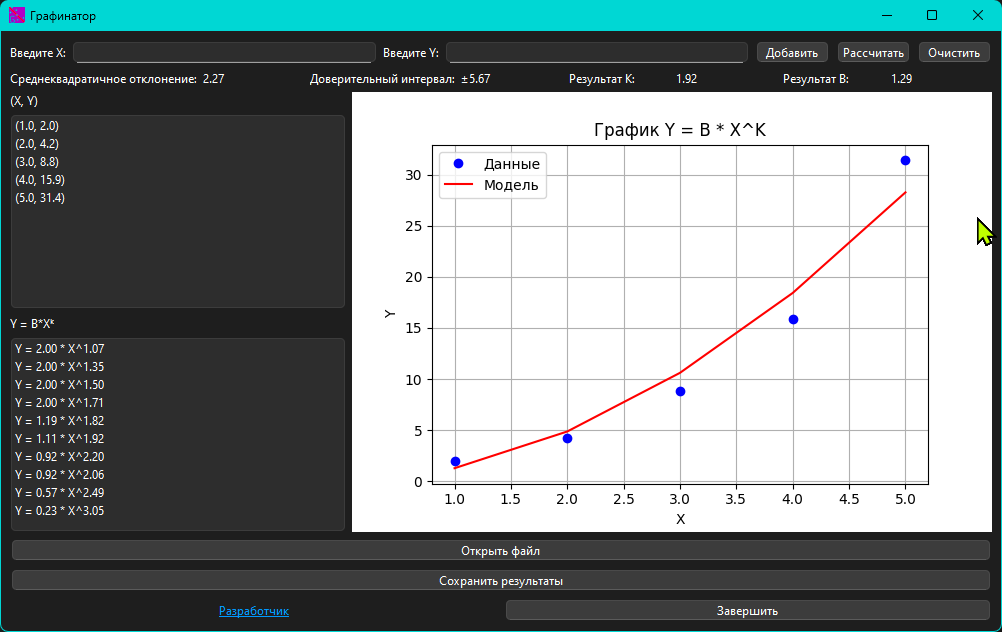
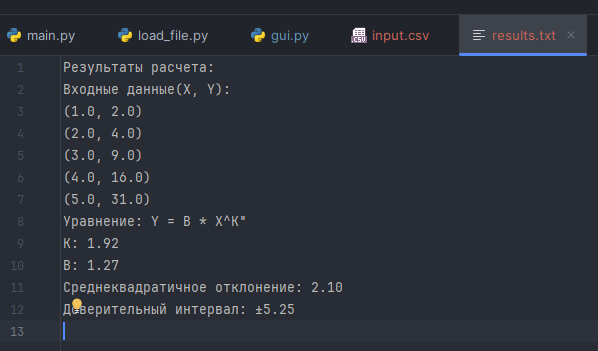


Рис 4. Результат работы приложения на основной форме



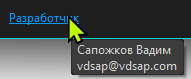
Рис 5. Результаты, сохраненные в файл

Рис 6. Информация о создателе

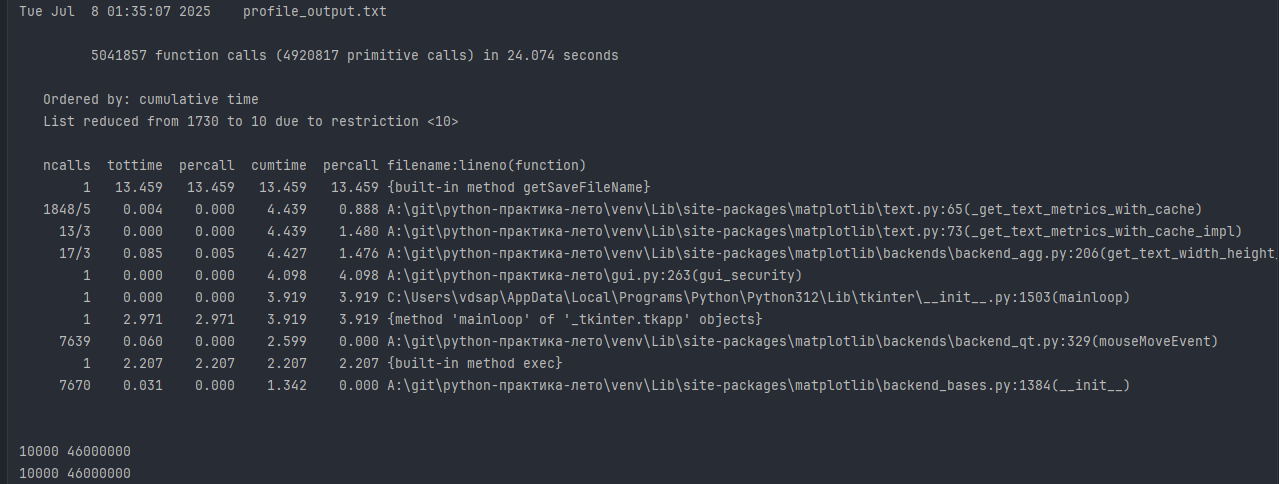


Рис 7. Затраченная память

ЛИСТИНГ ПРОГРАММЫ

**main.py**

*import* sys  
*import* cProfile  
*import* pstats  
*from* gui *import* security\_window, MainWindow  
*from* PyQt6.QtWidgets *import* QApplication  
  
  
*def* main():  
 security = security\_window()  
 security.gui\_security()  
 app = QApplication([])  
 app\_window = MainWindow()  
 app\_window.show()  
 sys.exit(app.exec())  
  
  
*if* \_\_name\_\_ == "\_\_main\_\_":  
 cProfile.run("main()", 'profile\_output.txt')  
 p = pstats.Stats('profile\_output.txt')  
 p.sort\_stats('cumulative').print\_stats(10)

**utils.py**

*import* numpy *as* np  
*from* scipy *import* stats  
*import* math  
  
  
*def* fit\_line(x: np.ndarray, y: np.ndarray) -> tuple[float, float, np.ndarray]:  
 *if* np.any(x <= 0) *or* np.any(y <= 0):  
 *raise* ValueError("Все значения X и Y должны быть положительными")  
 log\_x = np.log(x)  
 log\_y = np.log(y)  
 k, log\_b = np.polyfit(log\_x, log\_y, 1)  
 b = np.exp(log\_b)  
 y\_pred = b \* x \*\* k  
 *return* k, b, y\_pred  
  
  
*def* calculate\_k(x, y, n):  
 sum\_ln\_x = sum(math.log(x) *for* x *in* x)  
 sum\_ln\_y = sum(math.log(y) *for* y *in* y)  
 sum\_ln\_xln\_y = sum(math.log(x[i]) \* math.log(y[i]) *for* i *in* range(n))  
 sum\_ln\_x\_squared = sum(math.log(x) \*\* 2 *for* x *in* x)  
 numerator = sum\_ln\_x \* sum\_ln\_y - n \* sum\_ln\_xln\_y  
 denominator = sum\_ln\_x \*\* 2 - n \* sum\_ln\_x\_squared  
  
 *if* denominator == 0:  
 *return* float('inf')  
  
 k = numerator / denominator  
 *return* k  
  
  
*def* calculate\_sigma(y, yi):  
 n = len(yi)  
 sum\_squared\_diff = sum([(y[i] - yi[i]) \*\* 2 *for* i *in* range(n)])  
 sigma = math.sqrt(sum\_squared\_diff / (n - 1))  
 *return* sigma  
  
  
*def* calculate\_stats(x: np.ndarray, y: np.ndarray, y\_pred: np.ndarray) -> tuple[float, float]:  
 sigma = calculate\_sigma(y, y\_pred)  
 n = len(x)  
 mean\_log\_x = np.mean(np.log(x))  
 se\_k = sigma / np.sqrt(np.sum((np.log(x) - mean\_log\_x) \*\* 2))  
 t = stats.t.ppf(0.975, df=n - 2)  
 ci = t \* se\_k  
 *return* sigma, ci

**load\_file.py**

import csv

def load\_data\_from\_file(file\_path):

x\_data = []

y\_data = []

try:

with open(file\_path, newline='', encoding='utf-8') as csvfile:

reader = csv.reader(csvfile)

for row in reader:

if len(row) >= 2:

try:

x = float(row[0])

y = float(row[1])

x\_data.append(x)

y\_data.append(y)

except ValueError:

continue

except Exception as e:

raise RuntimeError(f"Ошибка чтения файла: {e}")

return x\_data, y\_data

**passwd\_base64.py**

*def* get\_pass():  
 *return* "NTA="

**gui.py**

*import* sys  
*from* tkinter *import* \*  
*from* tkinter *import* messagebox  
*import* base64  
*from* passwd\_base64 *import* get\_pass  
*import* numpy *as* np  
*from* PyQt6.QtWidgets *import* (  
 QMainWindow, QWidget, QPushButton, QLineEdit, QLabel,  
 QVBoxLayout, QHBoxLayout, QListWidget, QFileDialog, QMessageBox  
)  
*from* PyQt6.QtGui *import* QIcon  
*from* PyQt6.QtCore *import* pyqtSlot, Qt  
*from* matplotlib.backends.backend\_qt5agg *import* FigureCanvasQTAgg *as* FigureCanvas  
*import* matplotlib.pyplot *as* plt  
*from* load\_file *import* load\_data\_from\_file  
*from* utils *import* calculate\_stats  
  
  
*class* MainWindow(QMainWindow):  
 *def* \_\_init\_\_(*self*):  
 super().\_\_init\_\_()  
 *self*.setWindowTitle("Графинатор")  
 *self*.setGeometry(100, 100, 1000, 600)  
 *self*.setWindowIcon(QIcon("графинатор-ico.ico"))  
 *self*.central\_widget = QWidget()  
 *self*.setCentralWidget(*self*.central\_widget)  
 *self*.x\_input = QLineEdit()  
 *self*.y\_input = QLineEdit()  
 *self*.k\_output = QLabel(" ")  
 *self*.b\_output = QLabel(" ")  
 *self*.std\_dev = QLabel(" ")  
 *self*.confidence = QLabel(" ")  
 *self*.add\_btn = QPushButton("Добавить")  
 *self*.calc\_btn = QPushButton("Рассчитать")  
 *self*.clear\_btn = QPushButton("Очистить")  
 *self*.load\_btn = QPushButton("Открыть файл")  
 *self*.save\_btn = QPushButton("Сохранить результаты")  
 *self*.exit\_btn = QPushButton("Завершить")  
 *self*.dev\_text = QLabel("Разработчик")  
 *self*.dev\_text.setStyleSheet("QLabel {text-decoration: underline; color: #0099ff;}")  
 *self*.dev\_text.setToolTip("Сапожков Вадим\nvdsap@vdsap.com")  
 *self*.dev\_text.setToolTipDuration(5000)  
 *self*.dev\_text.setAlignment(Qt.AlignmentFlag.AlignCenter)  
 *self*.xy\_list = QListWidget()  
 *self*.eq\_list = QListWidget()  
 *self*.figure, *self*.ax = plt.subplots()  
 *self*.canvas = FigureCanvas(*self*.figure)  
 *self*.x\_data = []  
 *self*.y\_data = []  
 *self*.init\_ui()  
  
 *def* init\_ui(*self*):  
 layout = QVBoxLayout()  
 input\_layout = QHBoxLayout()  
 input\_layout.addWidget(QLabel("Введите X:"))  
 input\_layout.addWidget(*self*.x\_input)  
 input\_layout.addWidget(QLabel("Введите Y:"))  
 input\_layout.addWidget(*self*.y\_input)  
 input\_layout.addWidget(*self*.add\_btn)  
 input\_layout.addWidget(*self*.calc\_btn)  
 input\_layout.addWidget(*self*.clear\_btn)  
 result\_layout = QHBoxLayout()  
 result\_layout.addWidget(QLabel("Среднеквадратичное отклонение:"))  
 result\_layout.addWidget(*self*.std\_dev)  
 result\_layout.addWidget(QLabel("Доверительный интервал:"))  
 result\_layout.addWidget(*self*.confidence)  
 result\_layout.addWidget(QLabel("Результат K:"))  
 result\_layout.addWidget(*self*.k\_output)  
 result\_layout.addWidget(QLabel("Результат B:"))  
 result\_layout.addWidget(*self*.b\_output)  
 list\_layout = QVBoxLayout()  
 list\_layout.addWidget(QLabel("(X, Y)"))  
 list\_layout.addWidget(*self*.xy\_list)  
 list\_layout.addWidget(QLabel("Y = B\*Xᵏ"))  
 list\_layout.addWidget(*self*.eq\_list)  
 plot\_and\_lists = QHBoxLayout()  
 plot\_and\_lists.addLayout(list\_layout)  
 plot\_and\_lists.addWidget(*self*.canvas)  
 file\_layout = QVBoxLayout()  
 file\_layout.addWidget(*self*.load\_btn)  
 file\_layout.addWidget(*self*.save\_btn)  
 bottom\_layout = QHBoxLayout()  
 bottom\_layout.addWidget(*self*.dev\_text)  
 bottom\_layout.addWidget(*self*.exit\_btn)  
 layout.addLayout(input\_layout)  
 layout.addLayout(result\_layout)  
 layout.addLayout(plot\_and\_lists)  
 layout.addLayout(file\_layout)  
 layout.addLayout(bottom\_layout)  
 *self*.central\_widget.setLayout(layout)  
 *self*.add\_btn.clicked.connect(*self*.add\_data)  
 *self*.calc\_btn.clicked.connect(*self*.calculate)  
 *self*.clear\_btn.clicked.connect(*self*.clear\_all)  
 *self*.exit\_btn.clicked.connect(*self*.close)  
 *self*.load\_btn.clicked.connect(*self*.load\_from\_file)  
 *self*.save\_btn.clicked.connect(*self*.save\_results)  
  
 *def* add\_data(*self*):  
 *try*:  
 x = float(*self*.x\_input.text())  
 y = float(*self*.y\_input.text())  
 *self*.x\_data.append(x)  
 *self*.y\_data.append(y)  
 *self*.xy\_list.addItem(f"({x}, {y})")  
 *self*.x\_input.clear()  
 *self*.y\_input.clear()  
 *except* ValueError:  
 QMessageBox.warning(*self*, "Ошибка", "Введите числовые значения X и Y")  
  
 *def* calculate(*self*):  
 *if* len(*self*.x\_data) < 2:  
 QMessageBox.warning(*self*, "Ошибка", "Введите минимум два значения")  
 *return* x = np.array(*self*.x\_data)  
 y = np.array(*self*.y\_data)  
 *try*:  
 k\_list = []  
 b\_list = []  
 y\_fits = []  
 *for* i *in* range(len(x)):  
 *for* j *in* range(i + 1, len(x)):  
 x1, x2 = x[i], x[j]  
 y1, y2 = y[i], y[j]  
 *if* x1 <= 0 *or* x2 <= 0 *or* y1 <= 0 *or* y2 <= 0:  
 *continue  
 try*:  
 k\_ij = (np.log(y2) - np.log(y1)) / (np.log(x2) - np.log(x1))  
 b\_ij = np.exp(np.log(y1) - k\_ij \* np.log(x1))  
 k\_list.append(k\_ij)  
 b\_list.append(b\_ij)  
 y\_fits.append((k\_ij, b\_ij))  
 *except* Exception:  
 *continue  
 if not* k\_list *or not* b\_list:  
 *raise* ValueError("Недостаточно данных для расчета")  
 k = sum(k\_list) / len(k\_list)  
 b = sum(b\_list) / len(b\_list)  
 y\_pred = b \* x \*\* k  
 *except* ValueError:  
 QMessageBox.critical(*self*, "Ошибка", "Недостаточно данных для расчета")  
 *return* std\_dev, ci = calculate\_stats(x, y, y\_pred)  
 *self*.k\_output.setText(f"{k:.2f}")  
 *self*.b\_output.setText(f"{b:.2f}")  
 *self*.std\_dev.setText(f"{std\_dev:.2f}")  
 *self*.confidence.setText(f"±{ci:.2f}")  
 *self*.eq\_list.clear()  
 *for* k\_val, b\_val *in* y\_fits:  
 *self*.eq\_list.addItem(f"Y = {b\_val:.2f} \* X^{k\_val:.2f}")  
 *self*.ax.clear()  
 *self*.ax.plot(x, y, 'bo', label='Данные')  
 *self*.ax.plot(x, y\_pred, 'r-', label='Модель')  
 *self*.ax.set\_title("График Y = B \* X^K")  
 *self*.ax.set\_xlabel("X")  
 *self*.ax.set\_ylabel("Y")  
 *self*.ax.grid(*True*)  
 *self*.ax.legend()  
 *self*.canvas.draw()  
  
 *def* clear\_all(*self*):  
 *self*.x\_data.clear()  
 *self*.y\_data.clear()  
 *self*.xy\_list.clear()  
 *self*.eq\_list.clear()  
 *self*.k\_output.setText("-")  
 *self*.b\_output.setText("-")  
 *self*.std\_dev.setText("-")  
 *self*.confidence.setText("-")  
 *self*.ax.clear()  
 *self*.canvas.draw()  
 *self*.x\_input.clear()  
 *self*.y\_input.clear()  
  
 *def* save\_results(*self*):  
 *if not self*.x\_data *or not self*.y\_data:  
 QMessageBox.warning(*self*, "Предупреждение", "Нет данных для сохранения.")  
 *return* text\_path, \_ = QFileDialog.getSaveFileName(*self*, "Сохранить результаты", "results.txt", "Text Files (\*.txt)")  
 *if not* text\_path:  
 *return* plot\_path = text\_path.rsplit('.', 1)[0] + "\_plot.png"  
 *try*:  
 *with* open(text\_path, 'w', encoding='utf-8') *as* f:  
 f.write("""Результаты расчета:  
Входные данные(X, Y):  
""")  
 *for* x, y *in* zip(*self*.x\_data, *self*.y\_data):  
 f.write(f"""({x}, {y})  
""")  
 f.write(f"""Уравнение: Y = B \* X^K"  
K: {*self*.k\_output.text()}  
B: {*self*.b\_output.text()}  
Среднеквадратичное отклонение: {*self*.std\_dev.text()}  
Доверительный интервал: {*self*.confidence.text()}  
""")  
 *self*.figure.savefig(plot\_path)  
 QMessageBox.information(*self*, "Успех", "Результаты успешно сохранены.")  
 *except* Exception *as* e:  
 QMessageBox.critical(*self*, "Ошибка", f"Ошибка при сохранении: {str(e)}")  
  
 @pyqtSlot()  
 *def* load\_from\_file(*self*) -> *None*:  
 file\_dialog = QFileDialog()  
 file\_dialog.setNameFilter("CSV файлы (\*.csv);;Все файлы (\*)")  
 *if* file\_dialog.exec():  
 selected = file\_dialog.selectedFiles()  
 *if* selected:  
 file\_path = selected[0]  
 *try*:  
 x\_data, y\_data = load\_data\_from\_file(file\_path)  
 *self*.x\_data = x\_data  
 *self*.y\_data = y\_data  
 *self*.xy\_list.clear()  
 *for* x, y *in* zip(x\_data, y\_data):  
 *self*.xy\_list.addItem(f"({x}, {y})")  
 *self*.eq\_list.clear()  
 *self*.k\_output.clear()  
 *self*.b\_output.clear()  
 *self*.std\_dev.clear()  
 *self*.confidence.clear()  
 *self*.ax.clear()  
 *self*.canvas.draw()  
 *except* RuntimeError *as* e:  
 QMessageBox.critical(*self*, "Ошибка", str(e))  
  
  
*class* security\_window:  
 pass\_field = *None* window = *None  
  
 def* gui\_security(*self*):  
 *self*.window = Tk()  
 *self*.window.title("Графинатор-вход")  
 *self*.window.geometry('300x300')  
 title\_text = Label(*self*.window, text="Вход", font=("Arial", 20))  
 title\_text.grid(column=0, row=0, padx=120, pady=50)  
 pass\_text = Label(*self*.window, text="Введите пароль", font=("Arial", 10))  
 pass\_text.grid(column=0, row=1)  
 *self*.pass\_field = Entry(*self*.window, width=25)  
 *self*.pass\_field.grid(column=0, row=2)  
 *self*.pass\_field.focus()  
 enter\_btn = Button(*self*.window, text="Войти", font=("Arial", 10), command=*self*.gui\_security\_enter)  
 enter\_btn.grid(column=0, row=3)  
 *self*.window.protocol("WM\_DELETE\_WINDOW", *self*.on\_close)  
 *self*.window.mainloop()  
  
 *def* gui\_security\_enter(*self*):  
 input\_passwrd = *self*.pass\_field.get()  
 input\_passwrd\_bytes = input\_passwrd.encode('utf-8')  
 input\_passwrd\_b64\_bytes = base64.b64encode(input\_passwrd\_bytes)  
 input\_passwrd\_b64 = input\_passwrd\_b64\_bytes.decode("utf-8")  
 *if* input\_passwrd\_b64 == get\_pass():  
 *self*.window.destroy()  
 *else*:  
 messagebox.showerror('Ошибка', 'Пароль инвалид')  
 *# self.window.destroy()  
  
 def* on\_close(*self*):  
 *self*.window.destroy()  
 sys.exit()

**Заключение**

В ходе практики была разработана программа на языке Python в среде PyCharm, реализующая степенную аппроксимацию данных значений по методу наименьших квадратов

Программа обладает следующим функционалом:

1. Удобный интерфейс:

* Гибкий ввод данных: можно вводить данные или загружать из файла.
* Наглядное представление результатов: Программа отображает коэффициенты аппроксимирующей функции, среднеквадратичное отклонение и строит график для визуализации соответствия модели исходным данным.

2. Этапы работы программы:

* Авторизация: Базовый уровень безопасности обеспечивается путём проверки пароля.
* Ввод данных: Пользователь выбирает способ ввода данных - вручную или из файла.
* Расчёт параметров аппроксимации: методом наименьших квадратов вычисляются коэффициенты K и B степенной функции Y=B\*XK, наилучшим образом описывающей данные. Рассчитывается среднеквадратичное отклонение (СО), для оценки точности аппроксимации.
* Визуализация результатов: строится график, демонстрирующий соответствие полученной функции исходным данным. Отображаются значения коэффициентов K и B, а также среднеквадратичное отклонение.
* Очистка данных: Программа позволяет очистить все введённые данные и результаты для проведения новых расчётов.
* Сохранение результатов: Результаты расчёта и график можно сохранить для дальнейшего анализа и использования.

Разработанная программа является инструментом для линейной аппроксимации данных и может быть полезна в различных областях науки и техники, где требуется анализировать и моделировать зависимости между переменными.

Библиографический список

1. <https://docs.python.org/3/library> - Общая база знаний по Python
2. <https://habr.com/ru/companies/skillfactory/articles/599599/> - PyQt6 руководство
3. <https://pythonru.com/> - База знаний Python
4. <https://www.pythonguis.com/pyqt6-tutorial/> - Обучение PyQt6
5. <https://github.com/vdsap/summer-graphinator/> - Репозиторий проекта