Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский  
политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчёт по лабораторной работе № 5  
тема «Метод наименьших квадратов»  
по дисциплине «Информатика»

Выполнил: студент группы ПМ-23-2б Воронин М.А.

Проверил: ст. пр. каф. ВММБ Ильиных Г.В.

Пермь, 2024

**Задание 1**

***1.1 Постановка задачи***

1. Исходя из данных задачи подобрать функцию аппроксимации исходных

точек.

1. Построить систему линейных алгебраических уравнений для определения коэффициентов выбранной функции.
2. С помощью электронных таблиц определить коэффициенты аппроксимирующей исходные данные функции методом наименьших квадратов.
3. Построить точечную диаграмму исходных данных (маркеры без линии). Добавить кривую полученной функции (линия без маркеров). Диаграмма должна включать подписи осей, легенду, сетку. В диаграмме не должно быть названия, внешней границы фигуры.
4. Повторить шаги 1-4 для функции другого вида. Разместить новое решение необходимо на новом листe.
5. Найти квадратичную невязку каждой функции по формуле   
    для каждой точки (). Найти максимум квадратичной невязки. При помощи функции ЕСЛИ сделать вывод о том, какая функция лучше описывает исходные данные.
6. На основе лучшей по итогу 5 шага функции сделать прогноз для одной точки за пределами исходного диапазона. Добавить точку на диаграмму.

***1.2 Алгоритм выполнения***

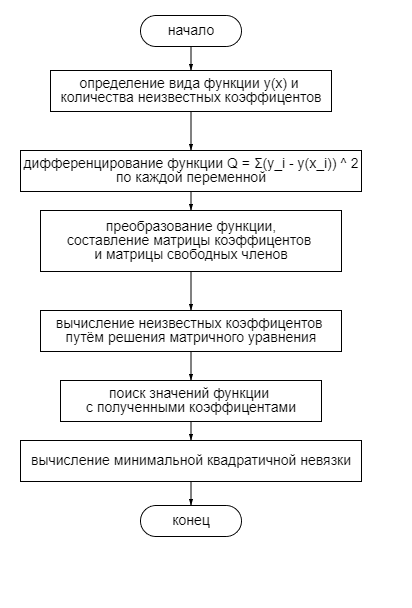
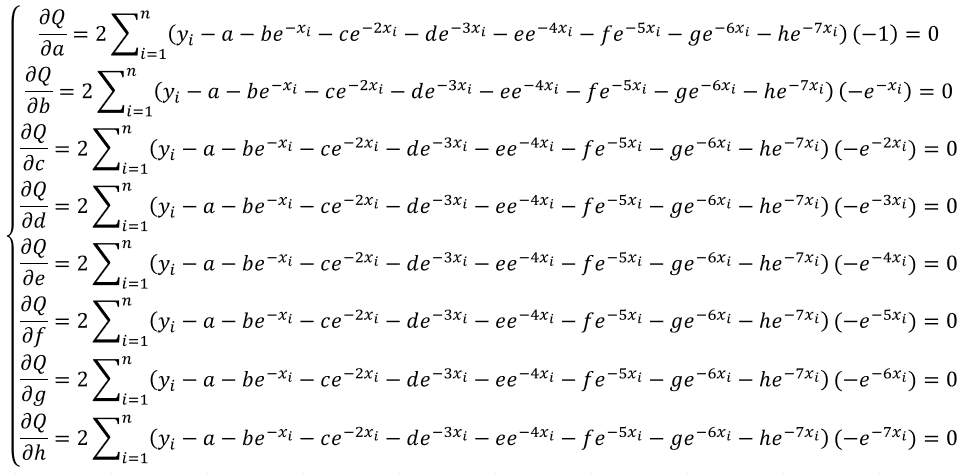


Рис. 1 – Блок-схема выполнения задания №1

***1.3 Решение задачи 1 в Excel***

Для решения задачи была составлена система уравнений из частных производных функции . Далее система была преобразована и решена с помощью матриц в Excel.



Шаги решения в Excel

1. Составление таблицы из степеней известных значений функции

2. Вычисление их суммы с помощью СУММ ()

3. Составление матрицы коэффициентов и матрицы свободных членов

4.Решение СЛАУ с помощью МУМНОЖ(МОБР(X-матрица); B-матрица)

5.Вычисление значений функции с найденными коэффициентами и поиск минимальной квадратичной невязки

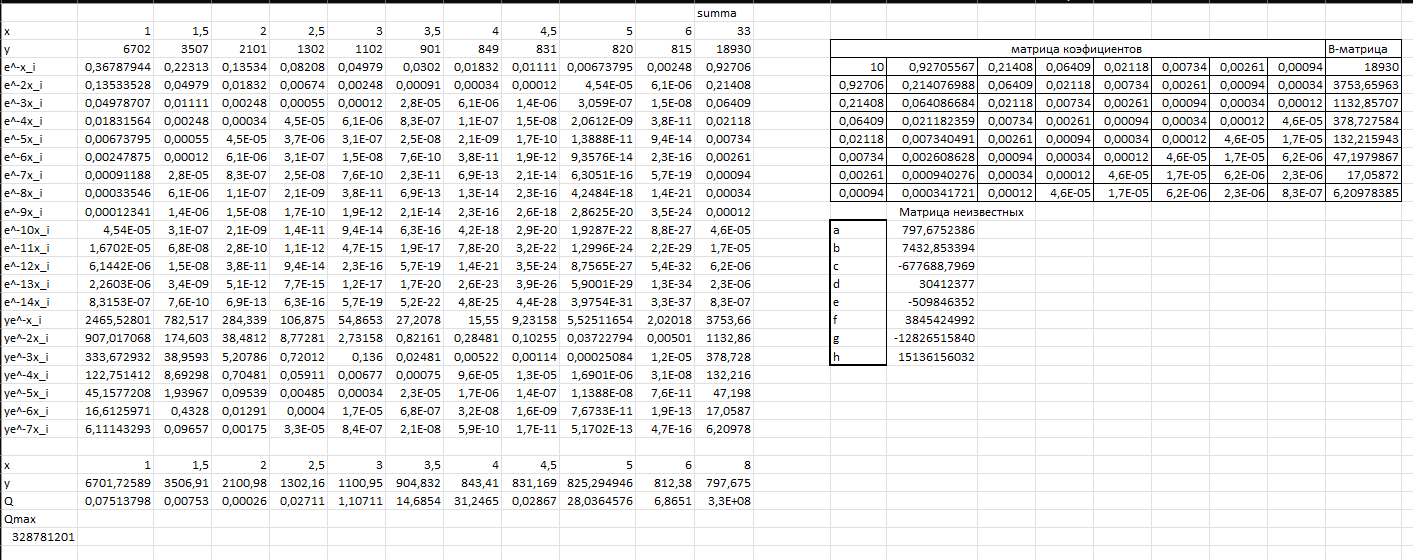


Рис.2 – Решение задачи 1

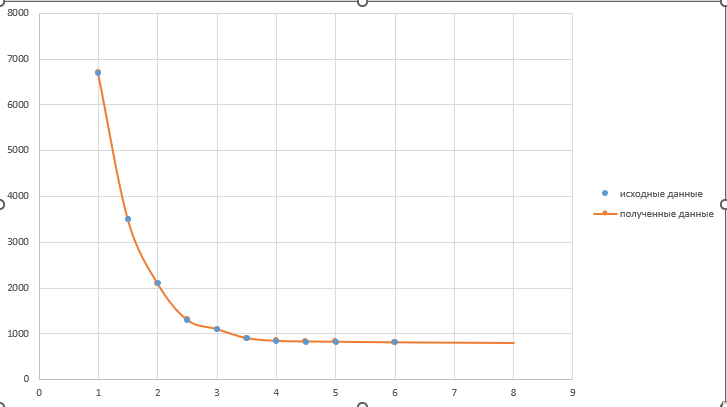
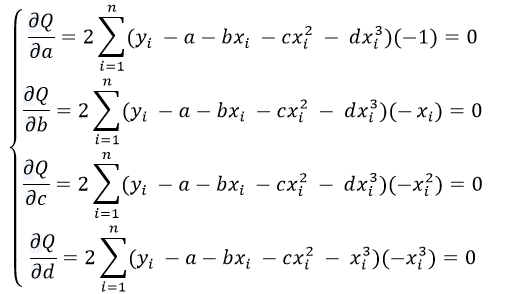


Рис.3 – График задачи 1

Значение функции при x = 8 равно **797,6752**

***1.4 Решение задачи 2 в Excel***

Для решения задачи была составлена система уравнений из частных производных функции . Далее система была преобразована и решена с помощью матриц в Excel.



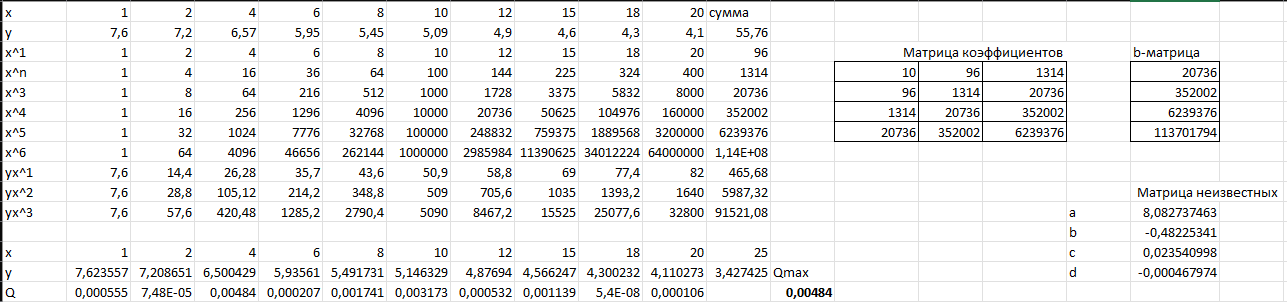
****

Рис.4 – Решение задачи 2

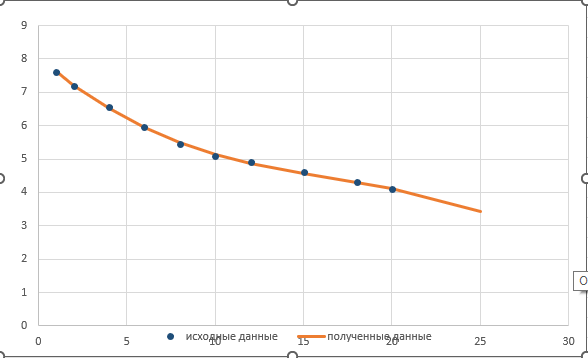


Рис.5 – График задачи 2 Шаги решения в Excel

1. Составление таблицы из степеней известных значений функции

2. Вычисление их суммы с помощью СУММ()

3. Составление матрицы коэффициентов и матрицы свободных членов

4.Решение СЛАУ с помощью МУМНОЖ(МОБР(X-матрица);B-матрица)

5.Вычисление значений функции с найденными коэффициентами и поиск минимальной квадратичной невязки

**Задание 2**

**2.1 Постановка задачи**

Разработать программу, определяющую коэффициенты аппроксимирующей исходные данные функции методом наименьших квадратов. С помощью программы определить коэффициенты функций, выбранных в Задании 1 и сравнить результаты.

**2.2 Алгоритм решения**

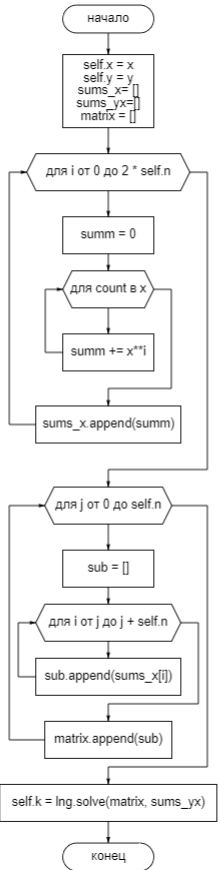


Рис.6 – Блок-схема метода approximate (x, y)

**2.3 Код программы**

import numpy as np  
import numpy.linalg as lng  
  
  
class Function:  
 def \_\_init\_\_(self, n=3, k=None):  
 if k is None:  
 k = []  
 self.k = k  
 self.n = n  
 self.x = []  
 self.y = []  
  
 def \_\_del\_\_(self):  
 del self.k  
 del self.n  
 del self.x  
 del self.y  
  
 def \_\_repr\_\_(self):  
 string = str(self.k[0])  
 for element in self.k[1:]:  
 string += f', {element}'  
 return self.n, string  
  
 def \_\_len\_\_(self):  
 return self.n + 1  
  
 def residual(self):  
 get\_array = [self.\_\_call\_\_(value) for value in self.x]  
 residual\_array = []  
 for i in range(len(self.y)):  
 residual\_array.append((get\_array[i] - self.y[i])\*\*2)  
 return max(residual\_array)  
  
  
class PowerFunction(Function):  
 count = 0  
  
 def \_\_init\_\_(self, n=3, k=None):  
 super().\_\_init\_\_(n, k)  
 PowerFunction.count += 1  
  
 def approximate(self, x, y):  
 self.x = x  
 self.y = y  
 sums\_x = []  
 sums\_yx = []  
 matrix = []  
 for i in range(2 \* self.n + 1):  
 summ = 0  
 for count in x:  
 summ += count \*\* i  
 sums\_x.append(summ)  
 for j in range(self.n + 1):  
 sub = []  
 for i in range(j, j + self.n + 1):  
 sub.append(sums\_x[i])  
 matrix.append(sub)  
 for i in range(self.n + 1):  
 summ = 0  
 for j in range(len(x)):  
 summ += y[j] \* x[j] \*\* i  
 sums\_yx.append(summ)  
 self.k = lng.solve(matrix, sums\_yx)  
  
 def \_\_str\_\_(self):  
 EPS = 10\*\*-4  
 string = str(round(self.k[0], 4)) if self.k[0] > EPS else f"{self.k[0]}"  
 for i in range(1, len(self.k)):  
 string += " + " if self.k[i] >= 0 else " "  
 string += f"{round(self.k[i], 4)} \* x\*\*{i}" if self.k[i] > EPS else f"{round(self.k[i], 10)} \* x\*\*{i}"  
 return string  
  
 def \_\_call\_\_(self, x):  
 i = 0  
 y = 0  
 for count in self.k:  
 y += count \* x \*\* i  
 i += 1  
 return round(y, 5)  
  
 def \_\_len\_\_(self):  
 return super().\_\_len\_\_()  
  
 def \_\_del\_\_(self):  
 super().\_\_del\_\_()  
 PowerFunction.count -= 1  
  
 def \_\_repr\_\_(self):  
 n, string = super().\_\_repr\_\_()  
 return f'PowerFunction({n},[{string}])'  
  
 def residual(self):  
 return super().residual()  
  
  
class ExponentialFunction(Function):  
 count = 0  
  
 def \_\_init\_\_(self, n=3, k=None):  
 super().\_\_init\_\_(n, k)  
 ExponentialFunction.count += 1  
  
 def approximate(self, x, y):  
 self.x = x  
 self.y = y  
 sums\_x = []  
 sums\_yx = []  
 matrix = []  
 for i in range(2 \* self.n + 1):  
 summ = 0  
 for count in x:  
 summ += np.e \*\* (-1\*(count \* i))  
 sums\_x.append(summ)  
 for j in range(self.n + 1):  
 sub = []  
 for i in range(j, j + self.n + 1):  
 sub.append(sums\_x[i])  
 matrix.append(sub)  
 for i in range(self.n + 1):  
 summ = 0  
 for j in range(len(x)):  
 summ += y[j] \* np.e \*\* (-1\*(x[j] \* i))  
 sums\_yx.append(summ)  
 self.k = lng.solve(matrix, sums\_yx)  
  
 def \_\_str\_\_(self):  
 string = str(round(self.k[0], 4))  
 for i in range(1, len(self.k)):  
 string += " + " if self.k[i] >= 0 else " "  
 string += f"{round(self.k[i],4)} \* e^-1\*{i}x"  
 return string  
  
 def \_\_call\_\_(self, x):  
 i = 0  
 y = 0  
 for count in self.k:  
 y += count \* np.e\*\*(-1\*(x \* i))  
 i += 1  
 return round(y, 5)  
  
 def \_\_len\_\_(self):  
 return super().\_\_len\_\_()  
  
 def \_\_del\_\_(self):  
 super().\_\_del\_\_()  
 ExponentialFunction.count -= 1  
  
 def \_\_repr\_\_(self):  
 n, string = super().\_\_repr\_\_()  
 return f'ExponentialFunction({n}, [{string}])'  
  
 def residual(self):  
 return super().residual()

**2.4 Тестирование работы программы и проверка решений**

В качестве исходных данных использовались значения x и y из задачи 1.

Функция имеет вид

Таблица 1 – Проверка решений с значениями из задачи 1.

|  |  |  |
| --- | --- | --- |
|  | Решение Python | Решение Excel |
| Значение функции при x = 8(решение задачи 1) |  |  |

**Задание 3**

**3.1 Постановка задачи**

Разработать графический пользовательский интерфейс программы, реализующей аппроксимацию данных методом наименьших квадратов

**3.2 Алгоритм решения**

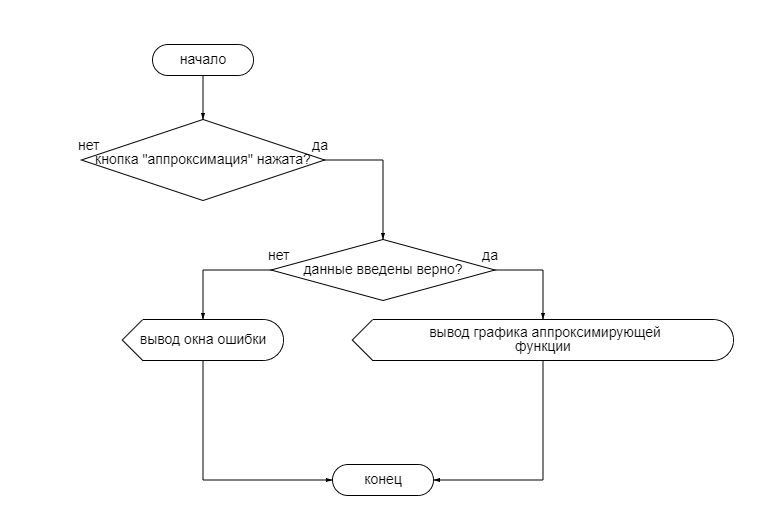


Рис.8 – Блок-схема программы №3

**3.3 Код программы**

from PyQt5.QtWidgets import (QApplication, QLabel, QLineEdit, QMainWindow, QPushButton, QFormLayout, QWidget, QComboBox, QMessageBox, QTableWidget, QTableWidgetItem)  
import numpy as np  
import matplotlib.pyplot as plt  
from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg as FigureCanvas  
  
class MainWindow(QMainWindow):  
 def \_\_init\_\_(self, parent=None):  
 super(MainWindow, self).\_\_init\_\_(parent)  
  
 self.setWindowTitle('График')  
 self.fig = plt.figure()  
 self.canvas = FigureCanvas(self.fig)  
 self.setGeometry(100, 100, 1100, 700)  
 cental\_widget = QWidget()  
 layout = QFormLayout()  
 cental\_widget.setLayout(layout)  
 layout.addWidget(self.canvas)  
 plt.grid(True)  
 self.setCentralWidget(cental\_widget)  
  
 self.input\_table = QTableWidget()  
 self.input\_table.setColumnCount(10)  
 self.input\_table.setRowCount(2)  
 self.input\_table.setVerticalHeaderLabels(['x', 'y'])  
  
 self.functions = QComboBox()  
 self.functions.addItems(['степенная', 'Экспонента'])  
  
 self.power\_text = QLabel()  
 self.power\_text.setText('Введите степень функции')  
 self.power = QLineEdit()  
 self.power.setFixedSize(50, 25)  
  
 self.function\_out = QLabel()  
 self.function\_out.setText('Функция: ')  
 self.function\_output = QLabel()  
  
 self.error\_message1 = QMessageBox()  
 self.error\_message1.setText("Точки функции заданы неверно!")  
 self.error\_message1.setWindowTitle('Ошибка!')  
 self.error\_message1.setIcon(QMessageBox.Warning)  
  
 self.error\_message2 = QMessageBox()  
 self.error\_message2.setText('Степень указана неверно!')  
 self.error\_message2.setWindowTitle('Ошибка!')  
 self.error\_message2.setIcon(QMessageBox.Warning)  
  
 self.approximate\_button = QPushButton('Аппроксимация')  
 self.approximate\_button.clicked.connect(self.plot\_approximate)  
  
 self.sort\_buttom = QPushButton('Отсортировать x')  
 self.sort\_buttom.clicked.connect(self.sorted\_arrays)  
  
 layout.addRow(self.power\_text)  
 layout.addRow(self.power, self.functions)  
 layout.addWidget(self.input\_table)  
 layout.addWidget(self.approximate\_button)  
 layout.addWidget(self.function\_out)  
 layout.addWidget(self.function\_output)  
 layout.addWidget(self.sort\_buttom)  
  
 def plot\_approximate(self):  
 if not self.power.text().isdigit():  
 self.error\_message2.show()  
 return 0  
 if self.functions.currentText() == 'степенная':  
 user\_function = PowerFunction(int(self.power.text()))  
 elif self.functions.currentText() == 'Экспонента':  
 user\_function = ExponentialFunction(int(self.power.text()))  
 x\_vector = [self.input\_table.item(0, i).text() for i in range(10) if self.input\_table.item(0, i) is not None]  
 y\_vector = [self.input\_table.item(1, i).text() for i in range(10) if self.input\_table.item(1, i) is not None]  
 try:  
 x\_vector = [float(x) for x in x\_vector]  
 y\_vector = [float(y) for y in y\_vector]  
 except ValueError:  
 self.error\_message1.show()  
  
 if len(x\_vector) == len(y\_vector) and len(x\_vector) >= 3 and len(y\_vector) >= 3:  
 user\_function.approximate(x\_vector, y\_vector)  
 plt.plot(x\_vector, y\_vector, marker='.', linewidth=0)  
 plot\_x = np.linspace(x\_vector[0], x\_vector[-1], 100)  
 plot\_y = [user\_function(x) for x in plot\_x]  
 plt.plot(plot\_x, plot\_y)  
 self.canvas.draw()  
 self.function\_output.setText(str(user\_function))  
 else:  
 self.error\_message1.show()  
  
 def sorted\_arrays(self):  
 x\_vector = [self.input\_table.item(0, i).text() for i in range(10) if self.input\_table.item(0, i) is not None]  
 y\_vector = [self.input\_table.item(1, i).text() for i in range(10) if self.input\_table.item(1, i) is not None]  
 x\_vector, y\_vector = sortirovka(x\_vector, y\_vector)  
 for i in range(len(x\_vector)):  
 self.input\_table.setItem(0, i, QTableWidgetItem(str(x\_vector[i])))  
 self.input\_table.setItem(1, i, QTableWidgetItem(str(y\_vector[i])))  
  
def sortirovka(x, y):  
 for \_ in x:  
 for i in range(1, len(x)):  
 if x[i - 1] > x[i]:  
 x[i], x[i - 1] = x[i - 1], x[i]  
 y[i], y[i - 1] = y[i - 1], y[i]  
 return x, y  
  
app = QApplication([])  
main = MainWindow()  
main.show()  
app.exec()

**3.4 Тестирование работы программы с проверкой**

Проверка №1 – аппроксимирующая функция из задачи 1

