Университет ИТМО

ФПИиКТ

Лабораторная работа №4  
по Вычислительной математике

Выполнил: Балтабаев Дамир  
Группа: P3210  
Вариант: 3

Преподаватель: Малышева Татьяна Алексеевна

Санкт-Петербург  
2022

**Цель лабораторной работы:**

Найти функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов

**Порядок выполнение работы:**

1. **Методика проведения исследования:**
2. Вычислить меру отклонения: для всех исследуемых функций.
3. Уточнить значения коэффициентов эмпирических функций, минимизируя функцию S.
4. Сформировать массивы предполагаемых эмпирических зависимостей (.
5. Определить среднеквадратичное отклонение для каждой аппроксимирующей функции. Выбрать наименьшее значение и, следовательно, наилучшее приближение.
6. Построить графики полученных эмпирических функций.
7. **Вычислительная реализация задачи:**

а) Для заданной функции (см. таблицу 1) построить наилучшие линейное и квадратичное приближения по 11 точкам указанного интервала.

b) Найти среднеквадратические отклонения. Ответы дать с тремя знаками после запятой.

c) Построить графики линейного и квадратичного приближений и заданной функции.

d) ***Привести в отчете подробные вычисления***.

1. **Программная реализация задачи:**
   1. Предусмотреть ввод исходных данных из файла/консоли (таблица *y=f(x)* должна содержать 10 - 12 точек).
   2. Реализовать метод наименьших квадратов, исследуя все функции п.1.
   3. Предусмотреть вывод результатов в файл/консоль.
   4. Для линейной зависимости вычислить коэффициент корреляции Пирсона.
   5. Программа должна отображать наилучшую аппроксимирующую функцию.
   6. Организовать вывод графиков функций, графики должны полностью отображать весь исследуемый интервал (с запасом).

**Рабочие формулы используемых методов:**

**Изображение выглядит как текст

Автоматически созданное описание**

**Изображение выглядит как текст

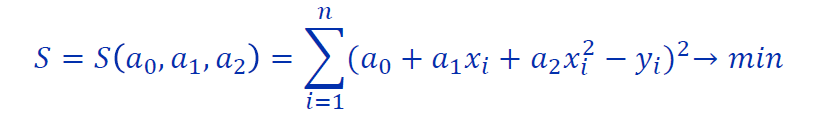
Автоматически созданное описание**

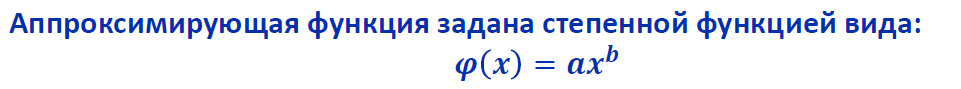
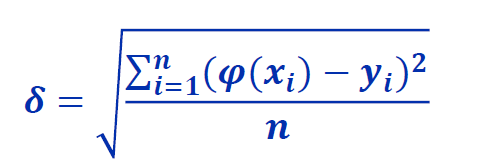
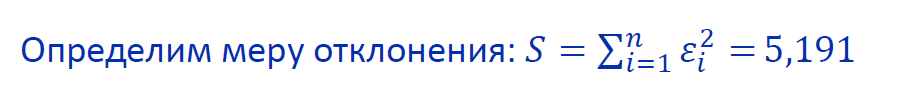
**Изображение выглядит как текст

Автоматически созданное описание**

**Изображение выглядит как текст

Автоматически созданное описание**

****

**Изображение выглядит как текст

Автоматически созданное описание**

**Изображение выглядит как текст

Автоматически созданное описание**

**Листинг программы**

**Линейная функция:**

**@Override**

**public void approximate() {**

**double SX = 0;**

**double SXX = 0;**

**double SY = 0;**

**double SXY = 0;**

**double xAverage = 0;**

**double yAverage = 0;**

**for (int i = 0; i < getN(); i++) {**

**SX += getX()[i];**

**SXX += getX()[i] \* getX()[i];**

**SY += getY()[i];**

**SXY += getX()[i] \* getY()[i];**

**}**

**xAverage = SX / getN();**

**yAverage = SY / getN();**

**a = (SXY \* getN() - SX \* SY) / (SXX \* getN() - SX \* SX);**

**b = (SXX \* SY - SX \* SXY) / (SXX \* getN() - SX \* SX);**

**double numerator = 0;**

**double firstPartDenominator = 0;**

**double secondPartDenominator = 0;**

**for (int i = 0; i < getN(); i++) {**

**S += Math.pow((linearXArgument(getX()[i]) - getY()[i]), 2);**

**numerator += (getX()[i] - xAverage) \* (getY()[i] - yAverage);**

**firstPartDenominator += Math.pow((getX()[i] - xAverage), 2);**

**secondPartDenominator += Math.pow((getY()[i] - yAverage), 2);**

**}**

**r = numerator / Math.sqrt(firstPartDenominator \* secondPartDenominator);**

**sigma = Math.sqrt(S / getN());**

**}**

**@Override**

**public double linearXArgument(double x) {**

**return a \* x + b;**

**}**

**Полиномиальная функция 2-й степени:**

**@Override**

**public void approximate() {**

**double X4S = 0;**

**double X3S = 0;**

**double X2S = 0;**

**double X2YS = 0;**

**double XS = 0;**

**double XYS = 0;**

**double YS = 0;**

**for (int i = 0; i < getN(); i++) {**

**X4S += Math.pow(getX()[i], 4);**

**X3S += Math.pow(getX()[i], 3);**

**X2S += Math.pow(getX()[i], 2);**

**X2YS += Math.pow(getX()[i], 2) \* getY()[i];**

**XS += getX()[i];**

**XYS += getX()[i] \* getY()[i];**

**YS += getY()[i];**

**}**

**double[][] firstMatrix = {{getN(), XS, X2S},**

**{XS, X2S, X3S},**

**{X2S, X3S, X4S}};**

**double[][] secondsMatrix = {{YS, XS, X2S},**

**{XYS, X2S, X3S},**

**{X2YS, X3S, X4S}};**

**double[][] thirdMatrix = {{getN(), YS, X2S},**

**{XS, XYS, X3S},**

**{X2S, X2YS, X4S}};**

**double[][] fourthMatrix = {{getN(), XS, YS},**

**{XS, X2S, XYS},**

**{X2S, X3S, X2YS}};**

**double delta = det(firstMatrix);**

**double delta1 = det(secondsMatrix);**

**double delta2 = det(thirdMatrix);**

**double delta3 = det(fourthMatrix);**

**a = delta3 / delta;**

**b = delta2 / delta;**

**c = delta1 / delta;**

**for (int i = 0; i < getN(); i++) {**

**S += Math.pow((linearXArgument(getX()[i]) - getY()[i]), 2);**

**}**

**sigma = Math.sqrt(S / getN());**

**}**

**@Override**

**public double linearXArgument(double x) {**

**return a \* Math.pow(x, 2) + b \* x + c;**

**}**

**Полиномиальная функция 3-й степени:**

**@Override**

**public void approximate() {**

**double XS = 0;**

**double X2S = 0;**

**double X3S = 0;**

**double X4S = 0;**

**double X5S = 0;**

**double X6S = 0;**

**double YS = 0;**

**double YXS = 0;**

**double YX2S = 0;**

**double YX3S = 0;**

**for (int i = 0; i < getN(); i++) {**

**XS += getX()[i];**

**X2S += Math.pow(getX()[i], 2);**

**X3S += Math.pow(getX()[i], 3);**

**X4S += Math.pow(getX()[i], 4);**

**X5S += Math.pow(getX()[i], 5);**

**X6S += Math.pow(getX()[i], 6);**

**YS += getY()[i];**

**YXS += getY()[i] \* getX()[i];**

**YX2S += getY()[i] \* Math.pow(getX()[i], 2);**

**YX3S += getY()[i] \* Math.pow(getX()[i], 3);**

**}**

**double[][] firstMatrix = {{getN(), XS, X2S, X3S},**

**{XS, X2S, X3S, X4S},**

**{X2S, X3S, X4S, X5S},**

**{X3S, X4S, X5S, X6S}};**

**double[][] secondsMatrix = {{YS, XS, X2S, X3S},**

**{YXS, X2S, X3S, X4S},**

**{YX2S, X3S, X4S, X5S},**

**{YX3S, X4S, X5S, X6S}};**

**double[][] thirdMatrix = {{getN(), YS, X2S, X3S},**

**{XS, YXS, X3S, X4S},**

**{X2S, YX2S, X4S, X5S},**

**{X3S, YX3S, X5S, X6S}};**

**double[][] fourthMatrix = {{getN(), XS, YS, X3S},**

**{XS, X2S, YXS, X4S},**

**{X2S, X3S, YX2S, X5S},**

**{X3S, X4S, YX3S, X6S}};**

**double[][] fifthMatrix = {{getN(), XS, X2S, YS},**

**{XS, X2S, X3S, YXS},**

**{X2S, X3S, X4S, YX2S},**

**{X3S, X4S, X5S, YX3S}};**

**double delta = det(firstMatrix);**

**double delta1 = det(secondsMatrix);**

**double delta2 = det(thirdMatrix);**

**double delta3 = det(fourthMatrix);**

**double delta4 = det(fifthMatrix);**

**a = delta4 / delta;**

**b = delta3 / delta;**

**c = delta2 / delta;**

**d = delta1 / delta;**

**for (int i = 0; i < getN(); i++) {**

**S += Math.pow((linearXArgument(getX()[i]) - getY()[i]), 2);**

**}**

**sigma = Math.sqrt(S / getN());**

**}**

**@Override**

**public double linearXArgument(double x) {**

**return a \* Math.pow(x, 3) + b \* Math.pow(x, 2) + c \* x + d;**

**}**

**Экспоненциальная функция:**

**@Override**

**public void approximate() {**

**double SX = 0;**

**double SXX = 0;**

**double SY = 0;**

**double SXY = 0;**

**for (int i = 0; i < getN(); i++) {**

**SX += getX()[i];**

**SXX += getX()[i] \* getX()[i];**

**SY += Math.log(getY()[i]);**

**SXY += getX()[i] \* Math.log(getY()[i]);**

**}**

**a = (SXX \* SY - SX \* SXY) / (SXX \* getN() - SX \* SX);**

**b = (SXY \* getN() - SX \* SY) / (SXX \* getN() - SX \* SX);**

**a = Math.pow(Math.E, a);**

**for (int i = 0; i < getN(); i++) {**

**S += Math.pow((linearXArgument(getX()[i]) - getY()[i]), 2);**

**}**

**sigma = Math.sqrt(S / getN());**

**}**

**@Override**

**public double linearXArgument(double x) {**

**return a \* Math.pow(Math.E, b \* x);**

**}**

**Логарифмическая функция:**

**@Override**

**public void approximate() {**

**double SX = 0;**

**double SXX = 0;**

**double SY = 0;**

**double SXY = 0;**

**for (int i = 0; i < getN(); i++) {**

**SX += Math.log(getX()[i]);**

**SXX += Math.log(getX()[i]) \* Math.log(getX()[i]);**

**SY += getY()[i];**

**SXY += Math.log(getX()[i]) \* getY()[i];**

**}**

**a = (SXY \* getN() - SX \* SY) / (SXX \* getN() - SX \* SX);**

**b = (SXX \* SY - SX \* SXY) / (SXX \* getN() - SX \* SX);**

**for (int i = 0; i < getN(); i++) {**

**S += Math.pow((linearXArgument(getX()[i]) - getY()[i]), 2);**

**}**

**sigma = Math.sqrt(S / getN());**

**}**

**@Override**

**public double linearXArgument(double x) {**

**return a \* Math.log(x) + b;**

**}**

**Степенная функция:**

**@Override**

**public void approximate() {**

**double SX = 0;**

**double SXX = 0;**

**double SY = 0;**

**double SXY = 0;**

**for (int i = 0; i < getN(); i++) {**

**SX += Math.log(getX()[i]);**

**SXX += Math.log(getX()[i]) \* Math.log(getX()[i]);**

**SY += Math.log(getY()[i]);**

**SXY += Math.log(getX()[i]) \* Math.log(getY()[i]);**

**}**

**a = (SXX \* SY - SX \* SXY) / (SXX \* getN() - SX \* SX);**

**b = (SXY \* getN() - SX \* SY) / (SXX \* getN() - SX \* SX);**

**a = Math.pow(Math.E, a);**

**for (int i = 0; i < getN(); i++) {**

**S += Math.pow((linearXArgument(getX()[i]) - getY()[i]), 2);**

**}**

**sigma = Math.sqrt(S / getN());**

**}**

**@Override**

**public double linearXArgument(double x) {**

**return a \* Math.pow(x, b);**

**}**

**Результаты выполнения программы**

**Изображение выглядит как текст

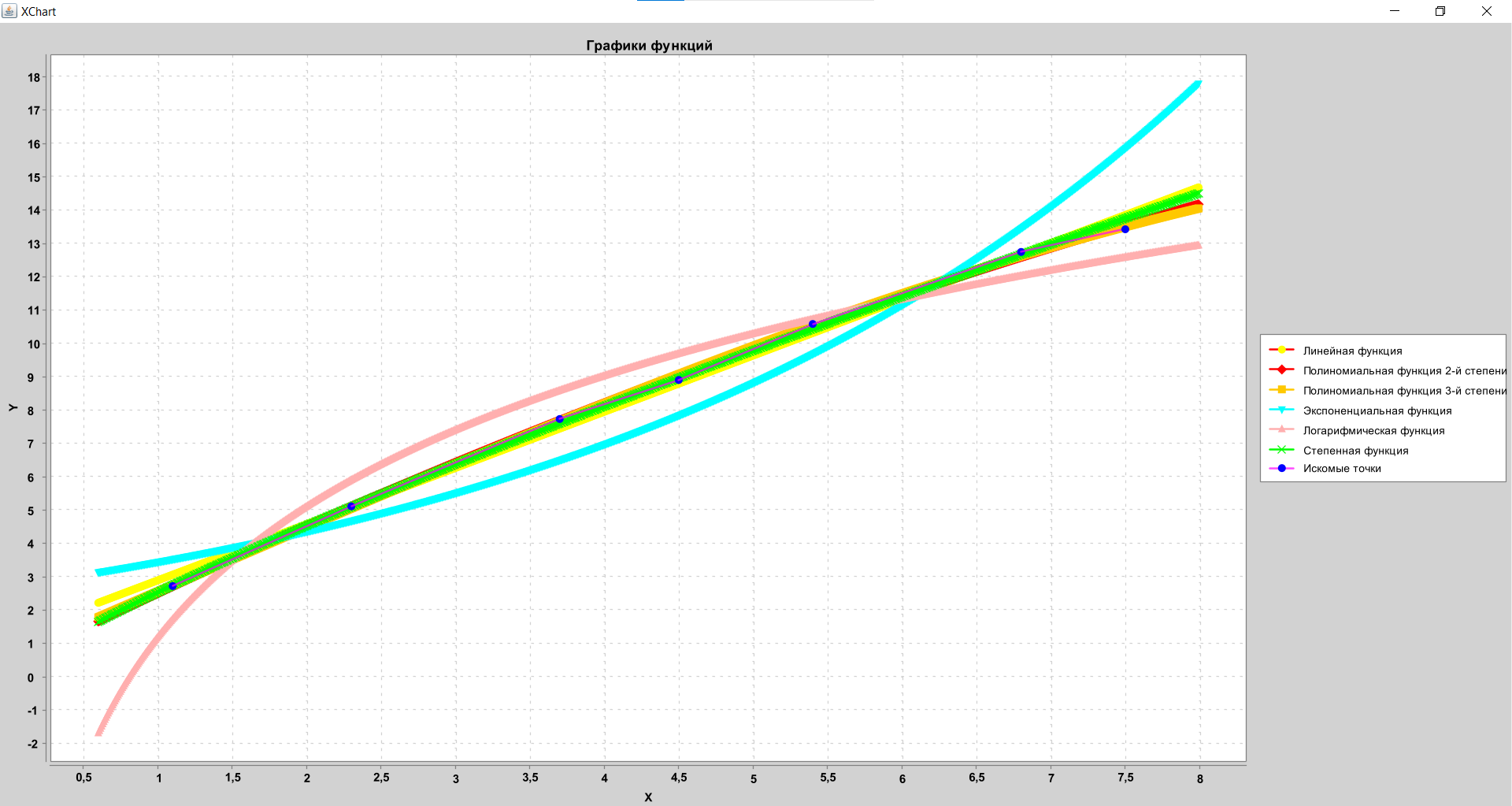
Автоматически созданное описание**

**Изображение выглядит как текст

Автоматически созданное описание**

**Изображение выглядит как текст

Автоматически созданное описание**



**Вычислительная реализация задачи**

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

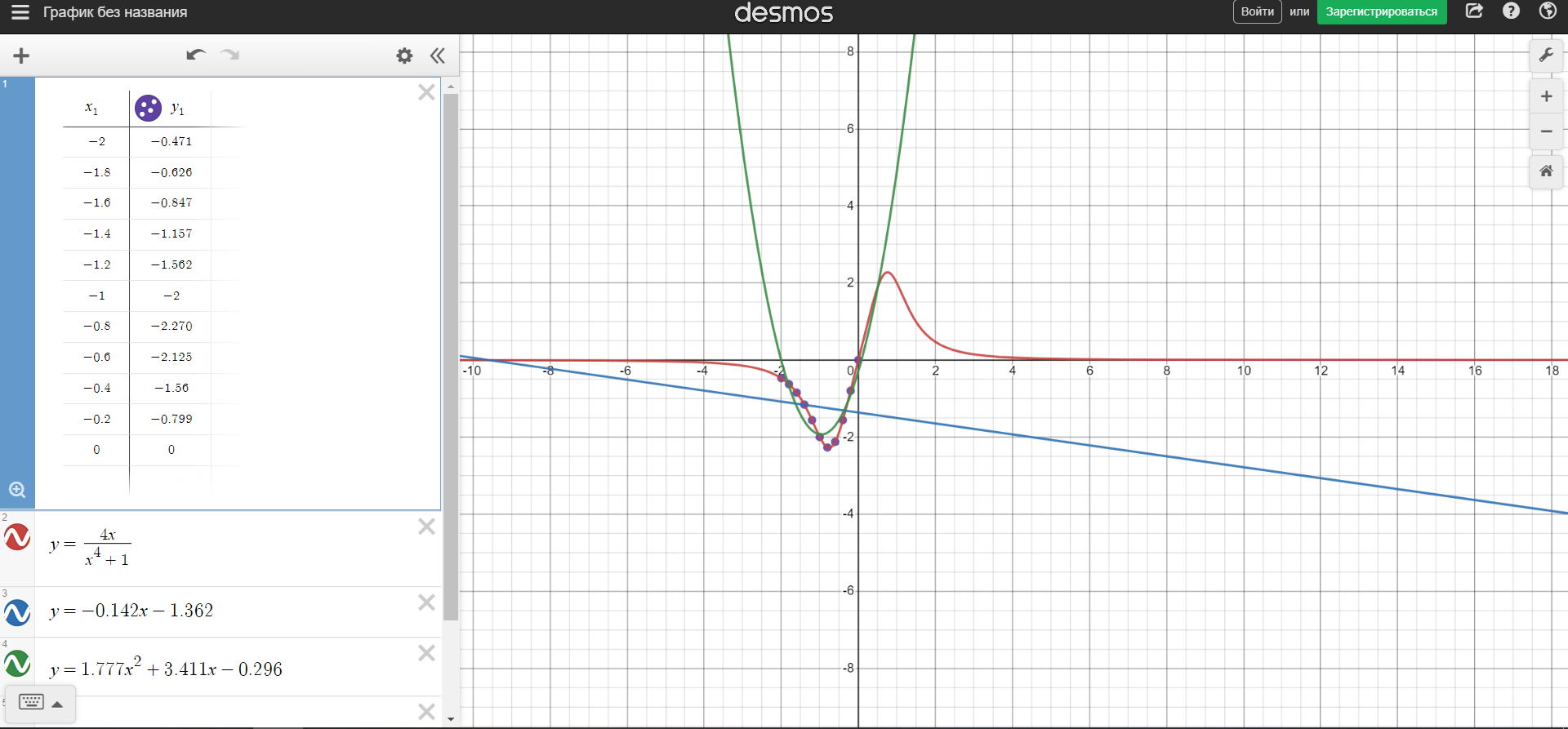
Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

****

**Выводы**

В ходе выполнения данной лабораторной работы я узнал что такое аппроксимация функций и поработал с аппроксимацией функций методом наименьших квадратов. В ходе исследования использовал такие функции, как: линейную функцию, полиномиальную функцию 2-й степени, полиномиальную функцию 3-й степени, экспоненциальную функцию, логарифмическую функцию, степенную функцию. Вычислил меру отклонения, среднеквадратическое отклонение, коэффициент Пирсона и научился определять наилучшую аппроксимирующую функцию.