Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика, искусственный интеллект и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

ОТЧЕТ

по лабораторной работе № 3

по курсу «Численные методы»

на тему: «Аппроксимация методом наименьших квадратов. Двупараметрические модели»

Вариант № 4

Студент ИУ9-61Б (Группа) ____________ (Подпись, дата) Афанасьев И. (И. О. Фамилия)

Преподаватель ____________ (Подпись, дата) Домрачева А. Б. (И. О. Фамилия)

2024 г.

1 Постановка задачи

- Построить графики таблично заданной функции и функции $z(x)$.
- Найти зачения x_a, x_g, x_h, y_a, y_g, y_h, $z(x_a)$, $z(x_g)$, $z(x_h)$, $\delta_1, \ldots, \delta_0$, $\delta_k = \min \delta_i$, $i = 1, \ldots, 9$.
- Составить систему уравнений для определения a и b; решить систему.
- Найти среднеквадратичное отклонение Δ.

2 Основные теоретические сведения

Пусть значения приближаемой функции $y = f(x)$ заданы лишь в узлах (x_i, y_i), $i = 1, \ldots, n$. Рассмотрим задачу *аппроксимации*: найдём гладкую аналитически заданную функцию $z(x)$, придающую наименьшее значение величине

$$\text{СКУ} = \sqrt{\sum_{k=1}^{n} (z(x_k) - y_k)^2}.$$

Эту величину называют *среднеквадратичным уклонением* (СКУ) функции $z(x)$ от системы узлов (x_i, y_i), $i = 0, \ldots, n$, а описанный подход к решению задачи приближения функции — *методом наименьших квадратов* (МНК). Абсолютной погрешностью аппроксимации служит среднеквадратичное отклонение (СКО):

$$\Delta = \frac{\text{СКУ}}{\sqrt{n}}$$

Существует формальный подход для выбора вида аппроксимирующей функции $z(x)$, зависящей от небольшого числа параметров (в данном случае двух — a и b). Пусть $x_a = \frac{x_0 + x_n}{2}$ — среднее арифметическое, $x_g = \sqrt{x_0 x_n}$ — среднее геометрическое и $x_h = \frac{2}{\frac{1}{x_0} + \frac{1}{x_n}}$ — среднее гармоническое чисел x_0 и x_n. Тогда

$$z_1(x) = ax + b \iff z(x_a) = z_a,$$

$$z_2(x) = a + x^b \iff z(x_g) = z_g,$$

$$z_3(x) = ae^{bx} \iff z(x_a) = z_g,$$

$$z_4(x) = a \ln x + b \iff z(x_g) = z_a,$$

$$z_5(x) = \frac{a}{x} + b \iff z(x_h) = z_a,$$

$$z_6(x) = \frac{1}{ax + b} \iff z(x_a) = z_h,$$

$$z_7(x) = \frac{x}{ax + b} \iff z(x_h) = z_h,$$

$$z_8(x) = ae^{\frac{b}{x}} \iff z(x_h) = z_g,$$

$$z_9(x) = \frac{1}{a \ln x + b} \iff z(x_g) = z_h,$$

где z_a, z_g, z_h — среднее арифметическое, среднее геометрическое и среднее гармоническое значения функции $z(x)$ в точках x_0 и x_n. Для выбора функции из рассмотренного семейства необходимо

1. Нанести на график заданные точки (x_i, y_i), $i = 0, \ldots, n$, и провести гладкую монотонную кривую, аппроксимирующую эту зависимость.

2. Вычислить значения величин x_a, x_g, x_h и y_a, y_g, y_h относительно значений x_0, x_n и y_0, y_n, а также определить по графику функции $z(x)$ значения $z(x_a)$, $z(x_g)$ и $z(x_h)$.

3. Вычислить значения следующих величин:

$$\delta_1 = |z(x_a) - y_a|, \quad \delta_2 = |z(x_g) - y_g|, \quad \delta_3 = |z(x_a) - y_g|,$$

$$\delta_4 = |z(x_g) - y_a|, \quad \delta_5 = |z(x_h) - y_a|, \quad \delta_6 = |z(x_a) - y_h|,$$

$$\delta_7 = |z(x_h) - y_h|, \quad \delta_8 = |z(x_h) - y_g|, \quad \delta_9 = |z(x_g) - y_h|.$$

   Номер k наименьшей величины δ_k, $i = 1, \ldots, 9$, определяет выбираемую функцию.

Аппроксимирующая функция индивидуального варианта есть $z_2 = ax^b$. Прологарифмировав функцию получим $\ln z_2(x) = a^* + b \ln x$, где $a^* = \ln a$. СКУ есть функция $F(a^*, b)$ двух переменных:

$$F(a^*, b) = \sum_{i=1}^{n} (a^* + b \ln x_i - \ln y_i)^2.$$

Задача сводится к нахождению минимума функции $F(a^*, b)$ решением системы уравнений

$$\begin{cases} \frac{\partial F}{\partial a^*} = 0, \\ \frac{\partial F}{\partial b} = 0. \end{cases} \tag{2.1}$$

Вычислим частные производные $\frac{\partial F}{\partial a^*}$ и $\frac{\partial F}{\partial b}$ функции:

$$\frac{\partial F}{\partial a^*} = 2 \sum_{i=1}^{n} (a^* + b \ln x_i - \ln y_i),$$

$$\frac{\partial F}{\partial b} = 2 \sum_{i=1}^{n} (a^* \ln x_i + b \ln^2 x_i - \ln y_i \ln x_i).$$

Запишем систему 2.1 в матричной форме:

$$\begin{pmatrix} \sum\limits_{i=1}^{n} \ln^2 x_i & \sum\limits_{i=1}^{n} \ln x_i \\ \sum\limits_{i=1}^{n} \ln x_i & n+1 \end{pmatrix} \begin{pmatrix} b \\ a^* \end{pmatrix} = \begin{pmatrix} \sum\limits_{i=1}^{n} \ln y_i \ln x_i \\ \sum\limits_{i=1}^{n} \ln y_i \end{pmatrix}$$

Найдём решение путём вычисления обратной матрицы системы методом алгебраических дополнений:

$$\begin{pmatrix} b \\ a^* \end{pmatrix} = \frac{1}{n \sum\limits_{i=1}^{n} \ln^2 x_i - (\sum\limits_{i=1}^{n} \ln x_i)^2} \begin{pmatrix} n \sum\limits_{i=1}^{n} \ln y_i \ln x_i - \sum\limits_{i=1}^{n} \ln x_i \sum\limits_{i=1}^{n} \ln y_i \\ - \sum\limits_{i=1}^{n} \ln x_i \sum\limits_{i=1}^{n} \ln y_i \ln x_i + \sum\limits_{i=1}^{n} \ln^2 x_i \sum\limits_{i=1}^{n} \ln y_i \end{pmatrix}.$$

3 Реализация

В листинге 3.1 представлен исходный код программы на языке C++.

Листинг 3.1 – Исходный код программы на языке C++

```cpp
// clang-format off
#include <sprout/math/exp.hpp>
#include <sprout/math/log.hpp>
#include <sprout/math/pow.hpp>
#include <sprout/math/sqrt.hpp>
// clang-format on

#include <array>
#include <cmath>
#include <iomanip>
#include <limits>

namespace {

using Deltas = std::array<double, 9>;

struct Parameters {
  double x_a, x_g, x_h;
  double y_a, y_g, y_h;
  double z_x_a, z_x_g, z_x_h;
  Deltas deltas;
};

struct Matrix {
  double a11, a12, a21, a22;
};

struct Vector {
  double b1, b2;
};

struct AugmentedMatrix {
  Matrix m;
  Vector v;
};

constexpr double ArithmeticMean(const double a, const double b) {
```

```
  return (a + b) / 2;
}

constexpr double GeometricMean(const double a, const double b) {
  return sprout::sqrt(a * b);
}

constexpr double HarmonicMean(const double a, const double b) {
  return 2 / (1 / a + 1 / b);
}

template <std::size_t N>
constexpr AugmentedMatrix BuildAugmentedMatrix(
    const std::array<double, N>& xs, const std::array<double,
        N>& ys) {
  Matrix m{};
  Vector v{};

  double ln_x = 0, ln_y = 0;
  for (std::size_t i = 0; i < N; ++i) {
    ln_x = sprout::log(xs[i]);
    ln_y = sprout::log(ys[i]);

    m.a11 += ln_x * ln_x;
    m.a12 += ln_x;
    m.a21 += ln_x;

    v.b1 += ln_y * ln_x;
    v.b2 += ln_y;
  }

  m.a22 = N;

  return {m, v};
}

constexpr Vector CalculateSolution(const AugmentedMatrix& am) {
  const auto det = am.m.a11 * am.m.a22 - am.m.a21 * am.m.a12;

  // Transposed matrix of algebraic complements
  const Matrix com{am.m.a22, -am.m.a12, -am.m.a21, am.m.a11};
```

```

  return {
      (com.a11 * am.v.b1 + com.a12 * am.v.b2) / det,
      (com.a21 * am.v.b1 + com.a22 * am.v.b2) / det,
  };
}

template <std::size_t N>
constexpr double CalculateStandartDeviation(const double b, const double ln_a,
                                            const std::array<double, N>& xs,
                                            const std::array<double, N>& ys) {
  double sum = 0;
  for (std::size_t i = 0; i < N; ++i) {
    sum += sprout::pow(ln_a + b * sprout::log(xs[i]) - sprout::log(ys[i]), 2);
  }

  return sprout::sqrt(sum / N);
}

template <std::size_t N>
constexpr Parameters CalculateParameters(const std::array<double, N>& xs,
                                         const std::array<double, N>& ys) {
  const auto x_a = ArithmeticMean(xs.front(), xs.back());
  const auto x_g = GeometricMean(xs.front(), xs.back());
  const auto x_h = HarmonicMean(xs.front(), xs.back());

  const auto y_a = ArithmeticMean(ys.front(), ys.back());
  const auto y_g = GeometricMean(ys.front(), ys.back());
  const auto y_h = HarmonicMean(ys.front(), ys.back());

  // Set according to the graph
  const auto kZXA = 4.2;
```

```
  const auto kZXG = 1.9;
  const auto kZXH = 0.9;

  const Deltas deltas{
      std::abs(kZXA - y_a), std::abs(kZXG - y_g), std::abs(kZXA - y_g),
      std::abs(kZXG - y_a), std::abs(kZXH - y_a), std::abs(kZXA - y_h),
      std::abs(kZXH - y_h), std::abs(kZXH - y_g), std::abs(kZXG - y_h),
  };

  return {x_a, x_g, x_h, y_a, y_g, y_h, kZXA, kZXG, kZXH, deltas};
}

template <std::size_t N>
void PrintCoordinates(const std::array<double, N>& xs,
                      const std::array<double, N>& ys) {
  static constexpr std::size_t kWidth = 6;

  std::cout << "Coordinates:\n\n";

  std::cout << std::setw(kWidth) << "i";
  for (std::size_t i = 0; i < N; ++i) {
    std::cout << std::setw(kWidth) << i;
  }
  std::cout << '\n';

  std::cout << std::setw(kWidth) << "x_i";
  for (std::size_t i = 0; i < N; ++i) {
    std::cout << std::setw(kWidth) << xs[i];
  }
  std::cout << '\n';

  std::cout << std::setw(kWidth) << "y_i";
  for (std::size_t i = 0; i < N; ++i) {
    std::cout << std::setw(kWidth) << ys[i];
  }
  std::cout << "\n\n";
}
```

```cpp

void PrintParameters(const Parameters& ps) {
  std::cout << "Parameters:\n\n";

  std::cout << "x_a = " << ps.x_a << '\n';
  std::cout << "x_g = " << ps.x_g << '\n';
  std::cout << "x_h = " << ps.x_h << "\n\n";

  std::cout << "y_a = " << ps.y_a << '\n';
  std::cout << "y_g = " << ps.y_g << '\n';
  std::cout << "y_h = " << ps.y_h << "\n\n";

  std::cout << "z(x_a) = " << ps.z_x_a << '\n';
  std::cout << "z(x_g) = " << ps.z_x_g << '\n';
  std::cout << "z(x_h) = " << ps.z_x_h << "\n\n";

  for (std::size_t i = 0, end = ps.deltas.size(); i < end; ++i) {
    std::cout << "delta_" << i + 1 << " = " << ps.deltas[i] << '\n';
  }
  std::cout << '\n';
}

void PrintAugmentedMatrix(const AugmentedMatrix& am) {
  std::cout << "Augmented matrix of the equations system:\n\n";

  std::cout << "a11 = " << am.m.a11 << '\n';
  std::cout << "a21 = " << am.m.a21 << '\n';
  std::cout << "a12 = " << am.m.a12 << '\n';
  std::cout << "a22 = " << am.m.a22 << "\n\n";

  std::cout << "b1 = " << am.v.b1 << '\n';
  std::cout << "b2 = " << am.v.b2 << "\n\n";
}

void PrintSolution(const Vector& v) {
  std::cout << "Solution:\n\n";

  std::cout << "ln(a) = " << v.b2 << ", a = " << sprout::exp(v.b2) << '\n';
  std::cout << "b = " << v.b1 << "\n\n";
```

```cpp
}

void PrintDeviation(const double d) {
  std::cout << "Standart deviation:\n\n";

  std::cout << " = " << d << '\n';
}

}  // namespace

int main() {
  constexpr std::size_t kN = 9;
  constexpr std::array<double, kN> kXs{
      1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0,
  };
  constexpr std::array<double, kN> kYs{
      0.16, 0.68, 1.96, 2.79, 3.80, 6.81, 9.50, 15.60, 24.86,
  };
  PrintCoordinates(kXs, kYs);

  constexpr auto parameters = CalculateParameters(kXs, kYs);
  PrintParameters(parameters);

  constexpr auto augmented_matrix = BuildAugmentedMatrix(kXs,
     kYs);
  PrintAugmentedMatrix(augmented_matrix);

  constexpr auto solution = CalculateSolution(augmented_matrix);
  PrintSolution(solution);

  constexpr auto deviation =
      CalculateStandartDeviation(solution.b1, solution.b2, kXs,
         kYs);
  PrintDeviation(deviation);
}
```

4 Результаты

В листинге 4.1 представлены результаты работы программы.

Листинг 4.1 – Результаты работы программы

```
Coordinates:

     i      0      1      2      3      4      5      6      7      8
   x_i      1    1.5      2    2.5      3    3.5      4    4.5      5
   y_i   0.16   0.68   1.96   2.79    3.8   6.81    9.5   15.6  24.86

Parameters:

x_a = 3
x_g = 2.23607
x_h = 1.66667

y_a = 12.51
y_g = 1.99439
y_h = 0.317954

z(x_a) = 4.2
z(x_g) = 1.9
z(x_h) = 0.9

delta_1 = 8.31
delta_2 = 0.0943921
delta_3 = 2.20561
delta_4 = 10.61
delta_5 = 11.61
delta_6 = 3.88205
delta_7 = 0.582046
delta_8 = 1.09439
delta_9 = 1.58205

Augmented matrix of the equations system:

a11 = 11.0352
a21 = 8.86609
a12 = 8.86609
a22 = 9
```

```
b1 = 17.5448
b2 = 10.946

Solution:

ln(a) = -1.67865, a = 0.186625
b = 2.9386

Standart deviation:

 = 0.161075
```