Федеральное государственное автономное образовательное учреждение высшего образования
"Национальный Исследовательский Университет ИТМО"
Мегафакультет Компьютерных Технологий и Управления
Факультет Программной Инженерии и Компьютерной Техники

Вариант №15
Лабораторная работа №4
по дисциплине
Вычислительная математика

Выполнил Студент группы P32101
Лапин Алексей Александрович
Преподаватель:
Рыбаков Степан Дмитриевич

г. Санкт-Петербург
2023г.

Содержание

1 Цель работы:

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

2 Порядок выполнения работы:

2.1 Методика проведения исследования:

1. Вычислить меру отклонения: $S = \sum_{i=1}^{n} [\varphi(x_i) - y_i]^2$ для всех исследуемых функций;
2. Уточнить значения коэффициентов эмпирических функций, минимизируя функцию S;
3. Сформировать массивы предполагаемых эмпирических зависимостей $\phi(x_i)$,;
4. Определить среднеквадратичное отклонение для каждой аппроксимирующей функции. Выбрать наименьшее значение и, следовательно, наилучшее приближение;
5. Построить графики полученных эмпирических функций.

2.2 Программная реализация задачи:

1. Предусмотреть ввод исходных данных из файла/консоли (таблица $y = f(x)$должна содержать от 8 до 12 точек);
2. Реализовать метод наименьших квадратов, исследуя все указанные функции;
3. Предусмотреть вывод результатов в файл/консоль: коэффициенты аппроксимирующих функций, среднеквадратичное отклонение, массивы значений $x_i, y_i, \varphi(x_i), \varepsilon_i$;
4. Для линейной зависимости вычислить коэффициент корреляции Пирсона;
5. Программа должна отображать наилучшую аппроксимирующую функцию;
6. Организовать вывод графиков функций, графики д полностью отображать весь исследуемый интервал (с запасом);
7. Программа должна быть протестирована при различных наборах данных, в том числе и некорректных;

2.3 Вычислительная реализация задачи:

1. Сформировать таблицу табулирования заданной функции на указанном интервале (см. табл. 1)
2. Построить линейное и квадратичное приближения по 11 точкам заданного интервала;
3. Найти среднеквадратические отклонения для каждой аппроксимирующей функции. Ответы дать с тремя знаками после запятой;
4. Выбрать наилучшее приближение;

5. Построить графики заданной функции, а также полученные линейное и квадратичное приближения;
6. Привести в отчете подробные вычисления.

3 Рабочие формулы используемых методов.

Линейная функция:

$$\varphi(x, a, b) = ax + b$$

Квадратичная функция:

$$\varphi(x, a_0, a_1, a_2) = a_0 + a_1 x + a_2 x^2$$

Степенная функция:

$$\varphi(x) = ax^b$$

Экспоненциальная функция:

$$\varphi(x) = ae^{bx}$$

Логарифмическая функция:

$$\varphi(x) = aln(x) + b$$

Коэффициент корреляции Пирсона:

$$r = \frac{\sum\limits_{i=1}^{n}(x_i - \overline{x})(y_i - y)}{\sqrt{\sum\limits_{i=1}^{n}(x_i - \overline{x})^2 \sum\limits_{i=1}^{n}(y_i - \overline{y})^2}}$$

Среднеквадратичное отклонение:

$$\delta = \sqrt{\frac{\sum\limits_{i=1}^{n}(\varphi(x_i) - y_i)^2}{n}}$$

4 Вычислительная реализация задачи:

$$y = \frac{4x}{x^4 + 15}$$

4.1 Линейная аппроксимация

X	-2	-1.8	-1.6	-1.4	-1.2	-1	-0.8	-0.6	-0.4	-0.2	0
Y	-0.258	-0.282	-0.297	-0.297	-0.281	-0.250	-0.208	-0.159	-0.106	-0.053	0.000

$SX = -11$
$SXX = 15.4$
$SY = -2.192$
$SXY = 2.818$
$\Delta = 48.4$
$\Delta_1 = 6.883$

$\Delta_2 = -2.761$
$a = \dfrac{\Delta_1}{\Delta} = \dfrac{6.883}{48.4} \approx 0.142$
$b = \dfrac{\Delta_2}{\Delta} = \dfrac{-2.761}{48.4} \approx -0.057$
$P(x) = 0.142x - 0.057$

X	-2	-1.8	-1.6	-1.4	-1.2	-1	-0.8	-0.6	-0.4	-0.2	0
Y	-0.258	-0.282	-0.297	-0.297	-0.281	-0.250	-0.208	-0.159	-0.106	-0.053	0.000
ax+b	-0.341	-0.313	-0.285	-0.256	-0.228	-0.199	-0.171	-0.142	-0.114	-0.085	-0.057
ε	-0.083	-0.031	0.012	0.041	0.053	0.051	0.037	0.016	-0.007	-0.032	-0.057
S	0.007	0.001	0.000	0.002	0.003	0.003	0.001	0.000	0.000	0.001	0.003
δ	0.044										

4.2 Квадратичная аппроксимация

X	-2	-1.8	-1.6	-1.4	-1.2	-1	-0.8	-0.6	-0.4	-0.2	0
Y	-0.258	-0.282	-0.297	-0.297	-0.281	-0.250	-0.208	-0.159	-0.106	-0.053	0.000

n	11.000
SX	-11.000
S2X	15.400
S3X	-24.200
S4X	40.533
SY	-2.192
SXY	2.818
S2XY	-4.154

$$\begin{cases} a_0 n + a_1 \sum_{i=1}^{n} x_i + a_2 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i \\ a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + a_2 \sum_{i=1}^{n} x_i^3 = \sum_{i=1}^{n} x_i y_i \\ a_0 \sum_{i=1}^{n} x_i^2 + a_1 \sum_{i=1}^{n} x_i^3 + a_2 \sum_{i=1}^{n} x_i^4 = \sum_{i=1}^{n} x_i^2 y_i \end{cases} = \begin{cases} a_0(11) + a_1(-11) + a_2(15.4) = (-2.192) \\ a_0(-11) + a_1(15.4) + a_2(-24.2) = (2.818) \\ a_0(15.4) + a_1(-24.2) + a_2(40.533) = (-4.154) \end{cases}$$

$$\begin{aligned} a_0 &= 0.016 \\ a_1 &= 0.384 \\ a_2 &= 0.121 \end{aligned}$$

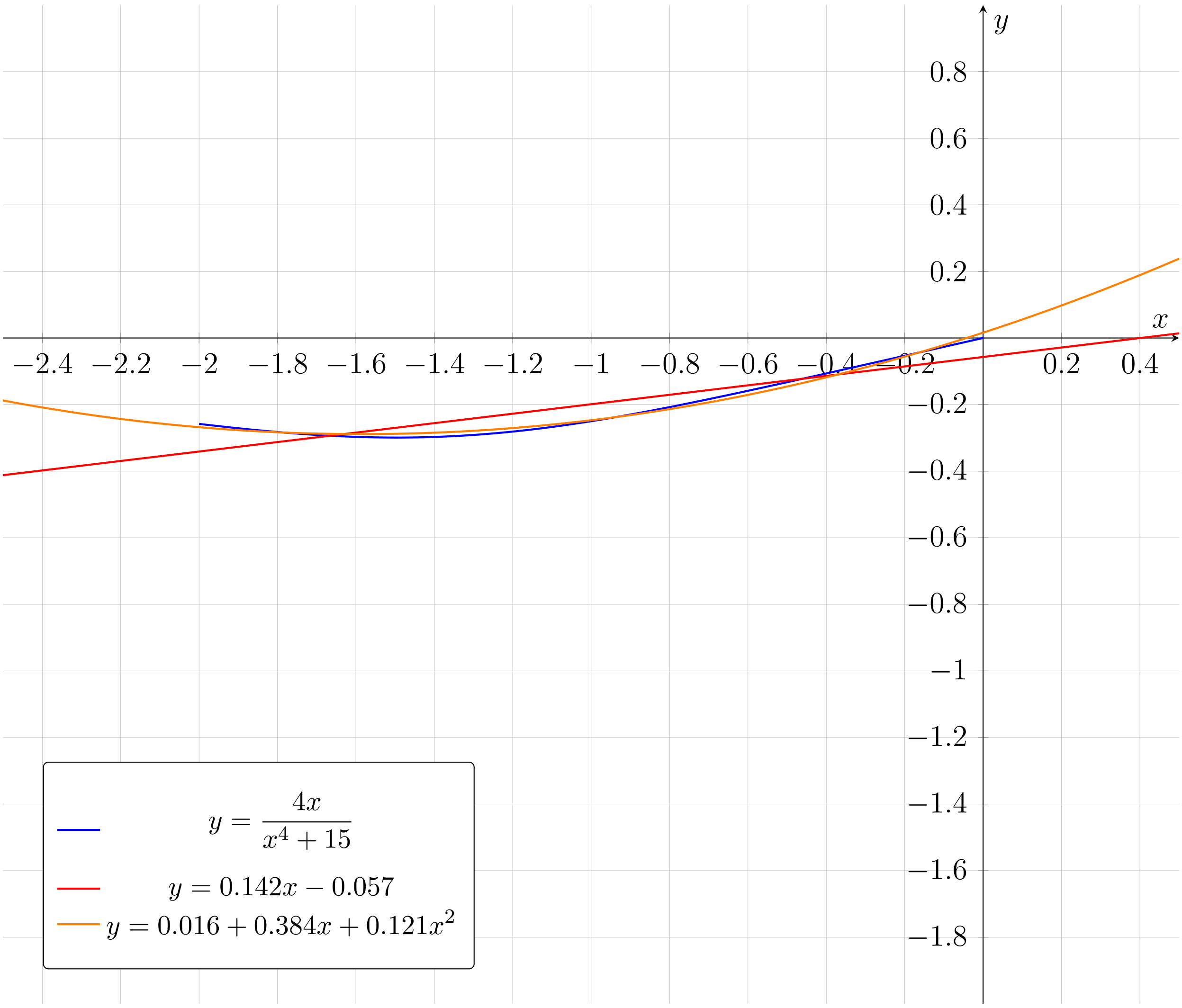
$P(x) = 0.016 + 0.384x + 0.121x^2$

X	-2	-1.8	-1.6	-1.4	-1.2	-1	-0.8	-0.6	-0.4	-0.2	0
Y	-0.258	-0.282	-0.297	-0.297	-0.281	-0.250	-0.208	-0.159	-0.106	-0.053	0.000
$a_0 + a_1 x + a_2 x^2$	-0.269	-0.284	-0.289	-0.285	-0.271	-0.248	-0.214	-0.171	-0.119	-0.056	0.016
ε	-0.011	-0.002	0.008	0.012	0.010	0.002	-0.007	-0.013	-0.012	-0.003	0.016
S	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
δ	0.010										

4.3 Наилучшее приближение

Квадратичная аппроксимация: $\delta = 0.01$

4.4 Графики функций



5 Листинг программы, по крайней мере, коды используемых методов.

```cpp
#include "Linear.h"

void approx::Linear::fit() {
    auto n = static_cast<double>(_points.size());
    double SX = 0, SY = 0, SXX = 0, SXY = 0;
    for(auto point : _points) {
        SX+=point.first;
        SXX+=point.first*point.first;
        SY+=point.second;
        SXY+=point.first*point.second;
    }
    std::vector<std::vector<double>> A = {{SX, SXX}, {n, SX}};
    std::vector<double> b = {SXY, SY};
    _params = std::move(GaussianElimination::solve(A, b));
    set_s();
    set_standard_deviation();
    set_pearson_correlation_coefficient();
}


void approx::Linear::set_pearson_correlation_coefficient() {
    double X = 0, Y = 0;
    for(auto point : _points) {
       X+=point.first;
       Y+=point.second;
    }
    double Sx = X / _points.size();
    double Sy = Y / _points.size();
    double Sxy = 0, Sxx = 0, Syy = 0;
    for(auto point : _points) {
        Sxy += (point.first - Sx)*(point.second - Sy);
        Sxx += (point.first - Sx)*(point.first - Sx);
        Syy += (point.second - Sy)*(point.second - Sy);
    }
    _extras["Pearson Correlation Coefficient"] = Sxy / sqrt(Sxx*Syy);
}

#include "Quadratic.h"

void approx::Quadratic::fit() {
    auto n = static_cast<double>(_points.size());
    double SX = 0, S2X = 0, S3X=0, S4X = 0,  SY = 0, SXY = 0, S2XY = 0;
    for(auto point : _points) {
        SX+=point.first;
        S2X+=point.first*point.first;
        S3X+=point.first*point.first*point.first;
        S4X+=point.first*point.first*point.first*point.first;
        SY+=point.second;
        SXY+=point.first*point.second;
```

```
        S2XY+=point.first*point.first*point.second;
    }
    std::vector<std::vector<double>> A = {{n, SX, S2X},
                                          {SX, S2X, S3X}, {S2X, S3X, S4X}};
    std::vector<double> b = {SY, SXY, S2XY};
    _params = std::move(GaussianElimination::solve(A, b));
    set_s();
    set_standard_deviation();
}

#include "Cubic.h"

void approx::Cubic::fit() {
    auto n = static_cast<double>(_points.size());
    std::vector<std::vector<double>> A(4);
    for(int i = 0; i < A.size(); i++) {
        A[i].push_back(pair_sum(_points, i));
        A[i].push_back(pair_sum(_points, i+1));
        A[i].push_back(pair_sum(_points, i+2));
        A[i].push_back(pair_sum(_points, i+3));
    }
    std::vector<double> b(4);
    for(int i = 0; i < b.size(); i++) {
        b[i] = two_pair_sum(_points, i, 1);
    }
    _params = std::move(GaussianElimination::solve(A, b));
    set_s();
    set_standard_deviation();

}

#include "Exponential.h"
#include "Linear.h"

void approx::Exponential::fit() {
    auto new_points = _points;
    for(auto & point : new_points) {
        point.second = log(point.second);
    }
    Linear linear(new_points);
    linear.fit();
    auto params = linear.getParams();
    _params.push_back(exp(params[0]));
    _params.push_back(params[1]);
    set_s();
    set_standard_deviation();
}

#include "Logarithmic.h"
```

```cpp
#include "Linear.h"

void approx::Logarithmic::fit() {
    auto new_points = _points;
    for(auto & point : new_points) {
        point.first = log(point.first);
    }
    Linear linear(new_points);
    linear.fit();
    auto params = linear.getParams();
    _params.push_back(params[0]);
    _params.push_back(params[1]);
    set_s();
    set_standard_deviation();
}
```

```cpp
#include "Power.h"
#include "Linear.h"

void approx::Power::fit() {
    auto new_points = _points;
    for(auto & point : new_points) {
        point.first = log(point.first);
        point.second = log(point.second);
    }
    Linear linear(new_points);
    linear.fit();
    auto params = linear.getParams();
    _params.push_back(exp(params[0]));
    _params.push_back(params[1]);
    set_s();
    set_standard_deviation();

}
```

```cpp
#include "AbstractApproximation.h"

approx::AbstractApproximation::AbstractApproximation(const
std::function<double(double, std::vector<double>)> &f,
const std::vector<std::pair<double,double>> &points) : _f(f),
                             _points(points) {}
void approx::AbstractApproximation::set_s() {
    double S = 0;
    if(_params.empty()){
        return;
    }
    for(auto point : _points) {
        double test = _f(point.first, _params);
```

```
        S+=pow(_f(point.first, _params) - point.second, 2);
    }
    _extras["Minimization criterion"] = S;
}

void approx::AbstractApproximation::set_standard_deviation() {
    if(_extras.find("Minimization criterion") == _extras.end()){
        set_s();
    }
    _extras["Standard Deviation"] =
    sqrt(_extras["Minimization criterion"] / (_points.size()));
}

std::pair<std::vector<std::vector<std::string>>,
std::vector<std::string>>
 approx::AbstractApproximation::get_info() {
    std::vector<std::vector<std::string>> res(_points.size());
    for(int i = 0; i < _points.size(); ++i) {
        res[i].push_back(std::to_string(static_cast<int>(i + 1)));
        res[i].push_back(std::to_string(_points[i].first));
        res[i].push_back(std::to_string(_points[i].second));
        res[i].push_back(std::to_string(_f(_points[i].first, _params)));
        res[i].push_back(std::to_string(_f(_points[i].first, _params) -
        _points[i].second));
    }
    return std::make_pair<std::vector<std::vector<std::string>> &,
    std::vector<std::string>>(res, {"N p.p.", "xi", "yi", "fi", "ei"});
}

double approx::AbstractApproximation::
get_extra_info(std::string key) {
    return _extras[key];
}

std::pair<std::vector<std::vector<std::string>>,
std::vector<std::string>>
    approx::AbstractApproximation::get_extras_info() {
    std::vector<std::vector<std::string>> res(_extras.size());
    int i = 0;
    for(auto extra : _extras) {
        res[i].push_back(extra.first);
        res[i].push_back(std::to_string(extra.second));
        ++i;
    }
    for(int i = 0; i < _params.size(); ++i) {
        res.push_back({"a" + std::to_string(i),
        std::to_string(_params[i])});
    }
```

```
    return std::make_pair<std::vector<std::vector<std::string>> &,
    std::vector<std::string>>(res,  {"Key", "Value"});
}

const std::function<double(double, std::vector<double>)> &

approx::AbstractApproximation::getF() const {
    return _f;
}

const std::vector<double>
&approx::AbstractApproximation::getParams() const {
    return _params;
}

double approx::AbstractApproximation::pair_sum(const

std::vector<std::pair<double, double>> array, int powNum, bool choice)

const {
    double sum = 0;
    for(auto& el : array) {
        if(choice) {
            sum += pow(el.first, powNum);
        } else {
            sum += pow(el.second, powNum);
        }
    }
    return sum;
}

double approx::AbstractApproximation::two_pair_sum(const
std::vector<std::pair<double, double>> array, int powNum1,
                                int powNum2) const {
    double sum = 0;
    for(auto& el : array) {
        sum += pow(el.first, powNum1) * pow(el.second, powNum2);
    }
    return sum;
}
```

6 Результаты выполнения программы при различных исходных данных.

```
Do you want to load points from a file? (y/n): y
Enter _filename: test1
Do you want to output to a file? (y/n): n
```

----------POWER----------

N p.p.	xi	yi	fi	ei
1	1.100000	2.730000	2.753465	0.023465
2	2.300000	5.120000	5.108953	-0.011047
3	3.700000	7.740000	7.609647	-0.130353
4	4.500000	8.910000	8.966163	0.056163
5	5.400000	10.590000	10.446321	-0.143679
6	6.800000	12.750000	12.672535	-0.077465
7	7.500000	13.430000	13.757005	0.327005

Key	Value
Standard Deviation	0.148514
Minimization criterion	0.154396
a0	2.542090
a1	0.838036

----------LOGARITHMIC----------

N p.p.	xi	yi	fi	ei
1	1.100000	2.730000	1.737381	-0.992619
2	2.300000	5.120000	5.904843	0.784843
3	3.700000	7.740000	8.591004	0.851004
4	4.500000	8.910000	9.696968	0.786968
5	5.400000	10.590000	10.727092	0.137092
6	6.800000	12.750000	12.029559	-0.720441
7	7.500000	13.430000	12.583152	-0.846848

```
+-------------------------+-----------+
```

Key	Value
Standard Deviation	0.774576
Minimization criterion	4.199778
a0	1.198875
a1	5.650037

----------EXPONENTIAL----------

N p.p.	xi	yi	fi	ei
1	1.100000	2.730000	3.534788	0.804788
2	2.300000	5.120000	4.683819	-0.436181
3	3.700000	7.740000	6.504437	-1.235563
4	4.500000	8.910000	7.846950	-1.063050
5	5.400000	10.590000	9.691220	-0.898780
6	6.800000	12.750000	13.458233	0.708233
7	7.500000	13.430000	15.859622	2.429622

Key	Value
Standard Deviation	1.236764
Minimization criterion	10.707090
a0	2.730945
a1	0.234550

----------CUBIC----------

N p.p.	xi	yi	fi	ei
1	1.100000	2.730000	2.757914	0.027914
2	2.300000	5.120000	5.064653	-0.055347
3	3.700000	7.740000	7.669250	-0.070750

4	4.500000	8.910000	9.079564	0.169564
5	5.400000	10.590000	10.569680	-0.020320
6	6.800000	12.750000	12.624264	-0.125736
7	7.500000	13.430000	13.504675	0.074675

Key	Value
Standard Deviation	0.092117
Minimization criterion	0.059399
a0	0.639772
a1	1.911877
a2	0.019107
a3	-0.006042

----------QUADRATIC----------

N p.p.	xi	yi	fi	ei
1	1.100000	2.730000	2.720172	-0.009828
2	2.300000	5.120000	5.116916	-0.003084
3	3.700000	7.740000	7.698891	-0.041109
4	4.500000	8.910000	9.070725	0.160725
5	5.400000	10.590000	10.523993	-0.066007
6	6.800000	12.750000	12.595125	-0.154875
7	7.500000	13.430000	13.544178	0.114178

Key	Value
Standard Deviation	0.099289
Minimization criterion	0.069008

a0	0.374260
a1	2.197386
a2	-0.058853

----------LINEAR----------

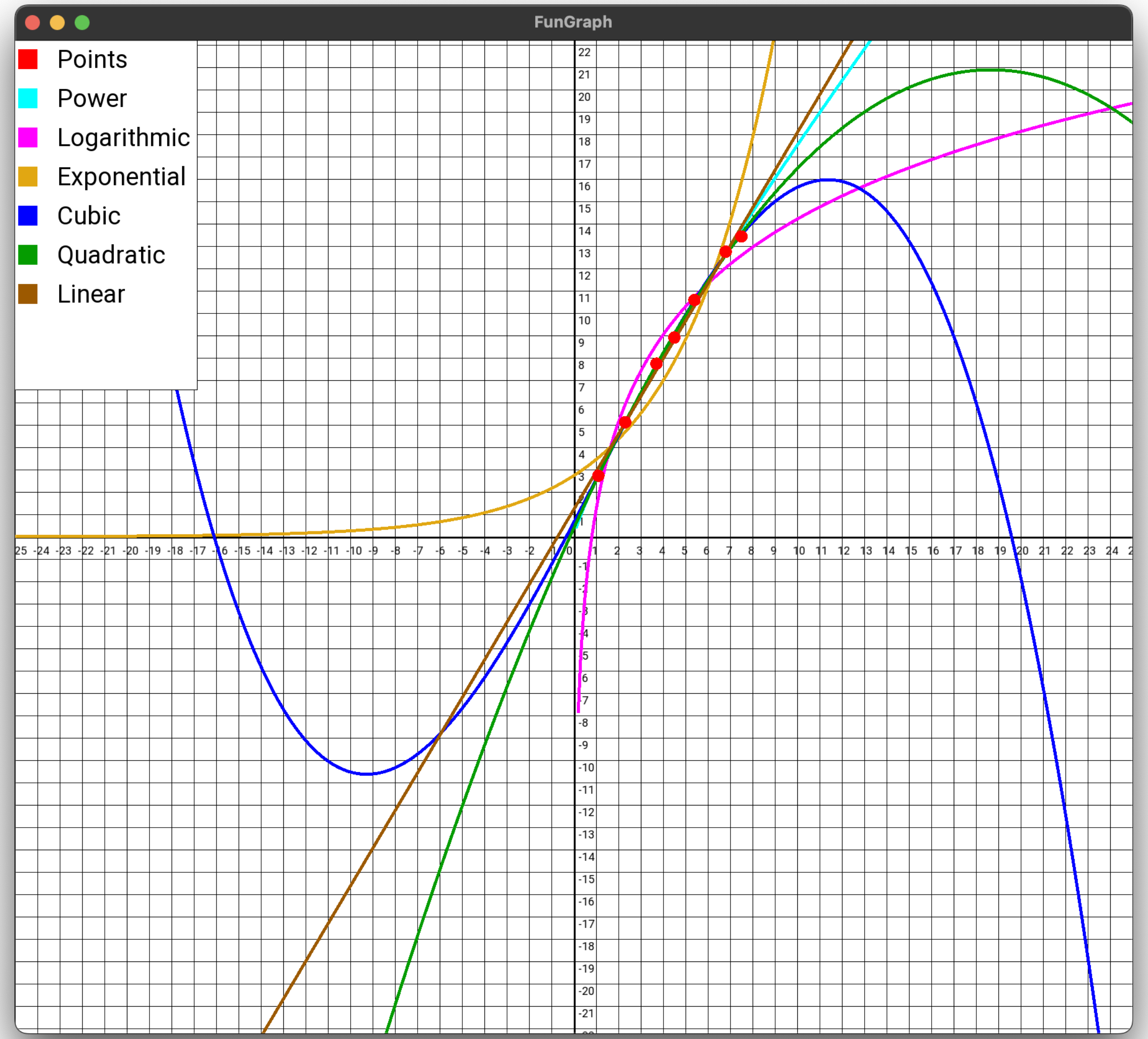
N p.p.	xi	yi	fi	ei
1	1.100000	2.730000	3.070710	0.340710
2	2.300000	5.120000	5.093169	-0.026831
3	3.700000	7.740000	7.452705	-0.287295
4	4.500000	8.910000	8.801011	-0.108989
5	5.400000	10.590000	10.317855	-0.272145
6	6.800000	12.750000	12.677391	-0.072609
7	7.500000	13.430000	13.857159	0.427159

Key	Value
Pearson Correlation Coefficient	0.997419
Standard Deviation	0.259950
Minimization criterion	0.473020
a0	1.216788
a1	1.685383

----------BEST APPROXIMATION----------

Best approximation: Cubic

Standard Deviation: 0.0921169




```
Do you want to load points from a file? (y/n): y
Enter _filename: test2
Do you want to output to a file? (y/n): n
----------POWER----------
+---------+------------+------------+------+------+
| N p.p.  | xi         | yi         | fi   | ei   |
+---------+------------+------------+------+------+
| 1       | -2.000000  | -0.258000  | nan  | nan  |
+---------+------------+------------+------+------+
| 2       | -1.800000  | -0.282000  | nan  | nan  |
+---------+------------+------------+------+------+
| 3       | -1.600000  | -0.297000  | nan  | nan  |
+---------+------------+------------+------+------+
| 4       | -1.400000  | -0.297000  | nan  | nan  |
+---------+------------+------------+------+------+
```

5	-1.200000	-0.281000	nan	nan
6	-1.000000	-0.250000	nan	nan
7	-0.800000	-0.208000	nan	nan
8	-0.600000	-0.159000	nan	nan
9	-0.400000	-0.106000	nan	nan
10	-0.200000	-0.053000	nan	nan
11	0.000000	0.000000	nan	nan

Key	Value
Standard Deviation	nan
Minimization criterion	nan
a0	nan
a1	nan

----------LOGARITHMIC----------

N p.p.	xi	yi	fi	ei
1	-2.000000	-0.258000	nan	nan
2	-1.800000	-0.282000	nan	nan
3	-1.600000	-0.297000	nan	nan
4	-1.400000	-0.297000	nan	nan
5	-1.200000	-0.281000	nan	nan
6	-1.000000	-0.250000	nan	nan
7	-0.800000	-0.208000	nan	nan
8	-0.600000	-0.159000	nan	nan
9	-0.400000	-0.106000	nan	nan
10	-0.200000	-0.053000	nan	nan

11	0.000000	0.000000	nan	nan

Key	Value
Standard Deviation	nan
Minimization criterion	nan
a0	nan
a1	nan

----------EXPONENTIAL----------

N p.p.	xi	yi	fi	ei
1	-2.000000	-0.258000	nan	nan
2	-1.800000	-0.282000	nan	nan
3	-1.600000	-0.297000	nan	nan
4	-1.400000	-0.297000	nan	nan
5	-1.200000	-0.281000	nan	nan
6	-1.000000	-0.250000	nan	nan
7	-0.800000	-0.208000	nan	nan
8	-0.600000	-0.159000	nan	nan
9	-0.400000	-0.106000	nan	nan
10	-0.200000	-0.053000	nan	nan
11	0.000000	0.000000	nan	nan

Key	Value
Standard Deviation	nan
Minimization criterion	nan
a0	nan

a1	nan

----------CUBIC----------

N p.p.	xi	yi	fi	ei
1	-2.000000	-0.258000	-0.254427	0.003573
2	-1.800000	-0.282000	-0.286678	-0.004678
3	-1.600000	-0.297000	-0.299760	-0.002760
4	-1.400000	-0.297000	-0.296040	0.000960
5	-1.200000	-0.281000	-0.277886	0.003114
6	-1.000000	-0.250000	-0.247667	0.002333
7	-0.800000	-0.208000	-0.207751	0.000249
8	-0.600000	-0.159000	-0.160506	-0.001506
9	-0.400000	-0.106000	-0.108301	-0.002301
10	-0.200000	-0.053000	-0.053503	-0.000503
11	0.000000	0.000000	0.001517	0.001517

Key	Value
Standard Deviation	0.002496
Minimization criterion	0.000069
a0	0.001517
a1	0.271717
a2	-0.026807
a3	-0.049340

----------QUADRATIC----------

N p.p.	xi	yi	fi	ei

1	-2.000000	-0.258000	-0.268636	-0.010636
2	-1.800000	-0.282000	-0.283836	-0.001836
3	-1.600000	-0.297000	-0.289339	0.007661
4	-1.400000	-0.297000	-0.285145	0.011855
5	-1.200000	-0.281000	-0.271255	0.009745
6	-1.000000	-0.250000	-0.247667	0.002333
7	-0.800000	-0.208000	-0.214382	-0.006382
8	-0.600000	-0.159000	-0.171400	-0.012400
9	-0.400000	-0.106000	-0.118721	-0.012721
10	-0.200000	-0.053000	-0.056345	-0.003345
11	0.000000	0.000000	0.015727	0.015727

Key	Value
Standard Deviation	0.009681
Minimization criterion	0.001031
a0	0.015727
a1	0.384606
a2	0.121212

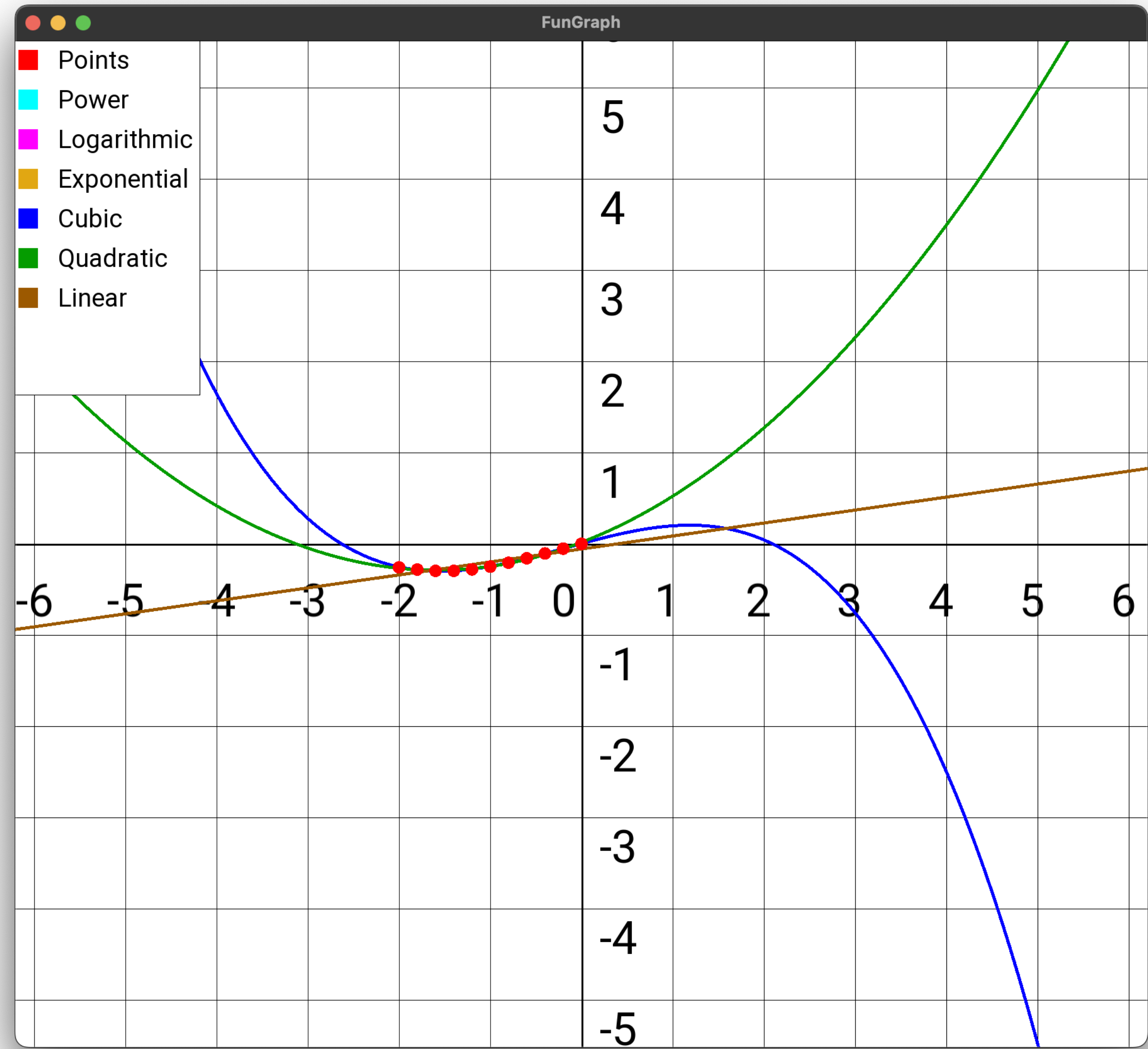
----------LINEAR----------

N p.p.	xi	yi	fi	ei
1	-2.000000	-0.258000	-0.341364	-0.083364
2	-1.800000	-0.282000	-0.312927	-0.030927
3	-1.600000	-0.297000	-0.284491	0.012509
4	-1.400000	-0.297000	-0.256055	0.040945
5	-1.200000	-0.281000	-0.227618	0.053382

6	-1.000000	-0.250000	-0.199182	0.050818
7	-0.800000	-0.208000	-0.170745	0.037255
8	-0.600000	-0.159000	-0.142309	0.016691
9	-0.400000	-0.106000	-0.113873	-0.007873
10	-0.200000	-0.053000	-0.085436	-0.032436
11	0.000000	0.000000	-0.057000	-0.057000

Key	Value
Pearson Correlation Coefficient	0.898626
Standard Deviation	0.043901
Minimization criterion	0.021201
a0	-0.057000
a1	0.142182

```
----------BEST APPROXIMATION----------
Best approximation: Cubic
Standard Deviation: 0.00249577
```




```
Do you want to load points from a file? (y/n): n
Enter number of points: 11
Enter x and y for point 1: 0 0
Enter x and y for point 2: 0.2 2.396
Enter x and y for point 3: 0.4 4.680
Enter x and y for point 4: 0.6 6.374
Enter x and y for point 5: 0.8 6.810
Enter x and y for point 6: 1 6
Enter x and y for point 7: 1.2 4.685
Enter x and y for point 8: 1.4 3.470
Enter x and y for point 9: 1.6 2.542
Enter x and y for point 10: 1.8 1.879
Enter x and y for point 11: 2 1.412
Do you want to output to a file? (y/n): n
----------POWER----------
```

N p.p.	xi	yi	fi	ei
1	0.000000	0.000000	nan	nan
2	0.200000	2.396000	nan	nan
3	0.400000	4.680000	nan	nan
4	0.600000	6.374000	nan	nan
5	0.800000	6.810000	nan	nan
6	1.000000	6.000000	nan	nan
7	1.200000	4.685000	nan	nan
8	1.400000	3.470000	nan	nan
9	1.600000	2.542000	nan	nan
10	1.800000	1.879000	nan	nan
11	2.000000	1.412000	nan	nan

Key	Value
Standard Deviation	nan
Minimization criterion	nan
a0	nan
a1	nan

----------LOGARITHMIC----------

N p.p.	xi	yi	fi	ei
1	0.000000	0.000000	nan	nan
2	0.200000	2.396000	nan	nan
3	0.400000	4.680000	nan	nan
4	0.600000	6.374000	nan	nan

5	0.800000	6.810000	nan	nan
6	1.000000	6.000000	nan	nan
7	1.200000	4.685000	nan	nan
8	1.400000	3.470000	nan	nan
9	1.600000	2.542000	nan	nan
10	1.800000	1.879000	nan	nan
11	2.000000	1.412000	nan	nan

Key	Value
Standard Deviation	nan
Minimization criterion	nan
a0	nan
a1	nan

----------EXPONENTIAL----------

N p.p.	xi	yi	fi	ei
1	0.000000	0.000000	nan	nan
2	0.200000	2.396000	nan	nan
3	0.400000	4.680000	nan	nan
4	0.600000	6.374000	nan	nan
5	0.800000	6.810000	nan	nan
6	1.000000	6.000000	nan	nan
7	1.200000	4.685000	nan	nan
8	1.400000	3.470000	nan	nan
9	1.600000	2.542000	nan	nan
10	1.800000	1.879000	nan	nan

11	2.000000	1.412000	nan	nan

Key	Value
Standard Deviation	nan
Minimization criterion	nan
a0	nan
a1	nan

----------CUBIC----------

N p.p.	xi	yi	fi	ei
1	0.000000	0.000000	-0.435566	-0.435566
2	0.200000	2.396000	2.984385	0.588385
3	0.400000	4.680000	5.096664	0.416664
4	0.600000	6.374000	6.121599	-0.252401
5	0.800000	6.810000	6.279515	-0.530485
6	1.000000	6.000000	5.790739	-0.209261
7	1.200000	4.685000	4.875597	0.190597
8	1.400000	3.470000	3.754415	0.284415
9	1.600000	2.542000	2.647520	0.105520
10	1.800000	1.879000	1.775238	-0.103762
11	2.000000	1.412000	1.357895	-0.054105

Key	Value
Standard Deviation	0.335814
Minimization criterion	1.240482
a0	-0.435566

a1	20.736144
a2	-19.099971
a3	4.590132

----------QUADRATIC----------

N p.p.	xi	yi	fi	ei
1	0.000000	0.000000	0.886392	0.886392
2	0.200000	2.396000	2.719993	0.323993
3	0.400000	4.680000	4.127228	-0.552772
4	0.600000	6.374000	5.108098	-1.265902
5	0.800000	6.810000	5.662601	-1.147399
6	1.000000	6.000000	5.790739	-0.209261
7	1.200000	4.685000	5.492510	0.807510
8	1.400000	3.470000	4.767916	1.297916
9	1.600000	2.542000	3.616956	1.074956
10	1.800000	1.879000	2.039629	0.160629
11	2.000000	1.412000	0.035937	-1.376063

Key	Value
Standard Deviation	0.932766
Minimization criterion	9.570580
a0	0.886392
a1	10.233922
a2	-5.329575

----------LINEAR----------

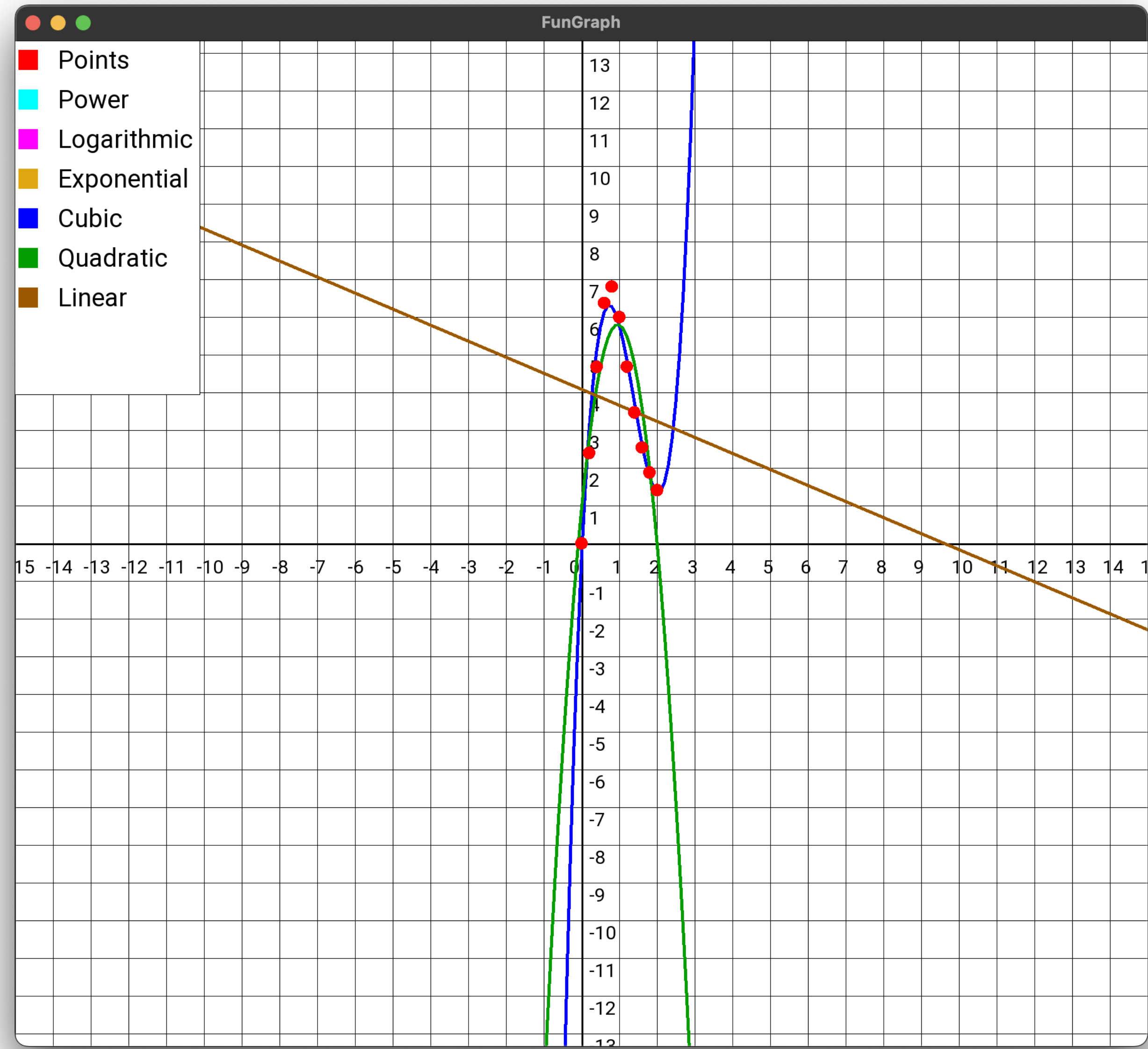
N p.p.	xi	yi	fi	ei
1	0.000000	0.000000	4.084136	4.084136
2	0.200000	2.396000	3.999091	1.603091
3	0.400000	4.680000	3.914045	-0.765955
4	0.600000	6.374000	3.829000	-2.545000
5	0.800000	6.810000	3.743955	-3.066045
6	1.000000	6.000000	3.658909	-2.341091
7	1.200000	4.685000	3.573864	-1.111136
8	1.400000	3.470000	3.488818	0.018818
9	1.600000	2.542000	3.403773	0.861773
10	1.800000	1.879000	3.318727	1.439727
11	2.000000	1.412000	3.233682	1.821682

Key	Value
Pearson Correlation Coefficient	-0.126958
Standard Deviation	2.101171
Minimization criterion	48.564093
a0	4.084136
a1	-0.425227

----------BEST APPROXIMATION----------

Best approximation: Cubic

Standard Deviation: 0.335814



7 Выводы

В этой лабораторной работе я изучил методы нахождения функции являющейся наилучшим приближением заданной табличной функции по методу наименьших квадратов, выполнил программную реализацию методов.