**Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»**

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №4**

По Вычислительной математике

Вариант №10

Выполнил:

Таджеддинов Рамиль Эмильевич

Группа № P3108

Поток № 1.3

Преподаватель:

Санкт-Петербург 2025

**Содержание**

[**1 Цель работы** 2](#_Toc27722)

[**2 Порядок выполнения работы** 2](#_Toc27723)

[2.1 Вычислительная реализация задачи 2](#_Toc27724)

[2.1.1 Линейная аппроксимация 2](#_Toc27725)

[2.1.2 Квадратичная аппроксимация 5](#_Toc27726)

[2.1.3 Выбор лучшего приближения 7](#_Toc27727)

[2.1.4 Графики функций 7](#_Toc27728)

[2.2 Программная реализация задачи 8](#_Toc27729)

[2.2.1 Листинг программы 8](#_Toc27730)

[2.2.2 Результаты выполнения программы 11](#_Toc27731)

[**3 Вывод** 13](#_Toc27732)

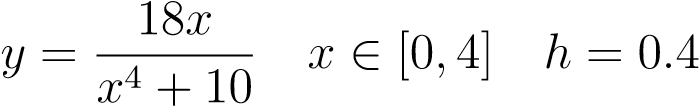
# **Цель работы**

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

# **Порядок выполнения работы**

## Вычислительная реализация задачи

Функция в соответствии с вариантом



Сформируем таблицу табулирования функции

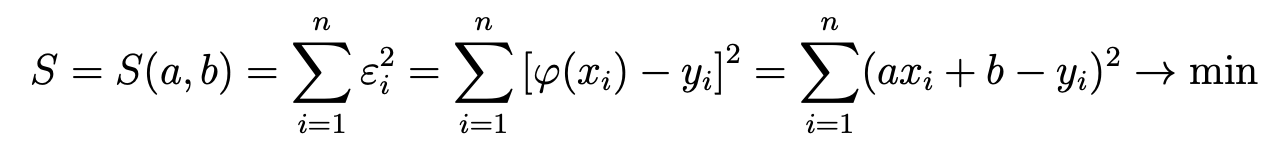
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *i* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| *xi* | 0 | 0.4 | 0.8 | 1.2 | 1.6 | 2.0 | 2.4 | 2.8 | 3.2 | 3.6 | 4.0 |
| *yi* | 0.000 | 0.718 | 1.383 | 1.789 | 1.740 | 1.385 | 1.001 | 0.705 | 0.501 | 0.364 | 0.271 |

### Линейная аппроксимация

Найдём линейное приближение данной функции

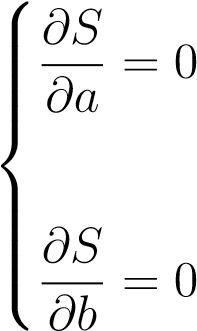
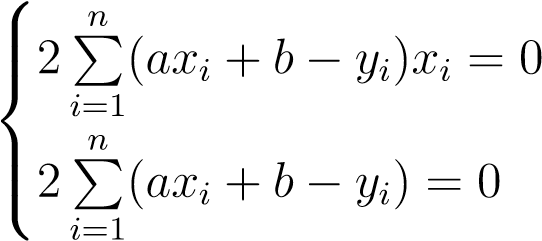
*ϕ*(*x*) = *ax* + *b*

Сумма квадратов отклонений запишется следующим образом:

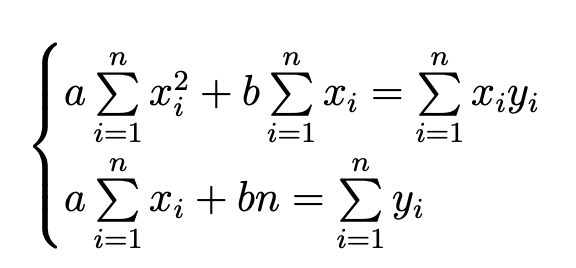
**

Для нахождения *a* и *b* необходимо найти минимум функции *S*(*a,b*).

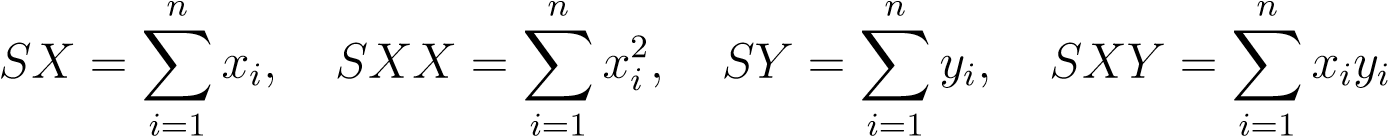
Необходимое условие существования минимума для функции *S*:

 или 

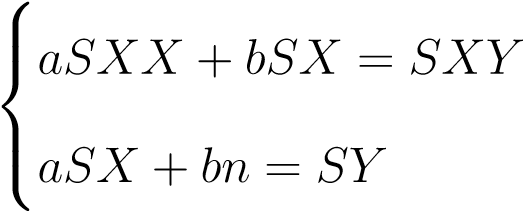
Упростим полученную систему:



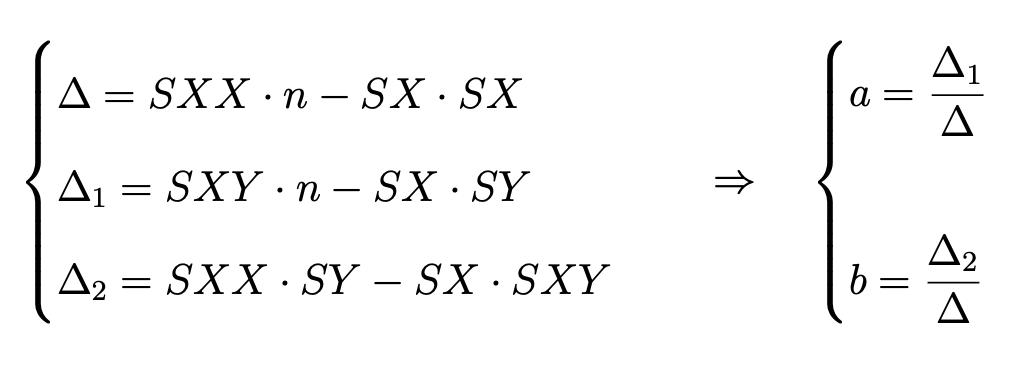
Введем обозначения:

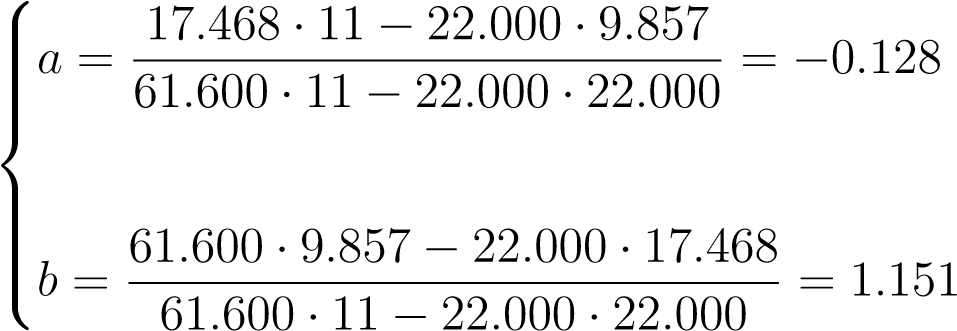


Получим систему уравнений для нахождения параметров *a* и *b*:



из которой по правилу Крамера получаем:

Поставив соответствующие значения получаем:

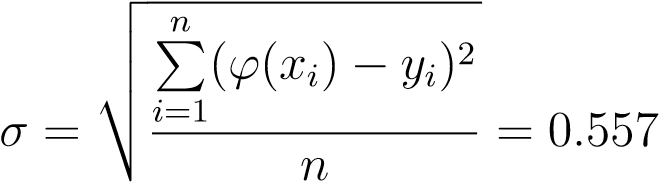


Таким образом аппроксимирующая линейная функция имеет вид:

*ϕ*(*x*) = −0*.*128*x* + 1*.*151

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *i* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| *xi* | 0.0 | 0.4 | 0.8 | 1.2 | 1.6 | 2.0 | 2.4 | 2.8 | 3.2 | 3.6 | 4.0 |
| *yi* | 0.000 | 0.718 | 1.383 | 1.789 | 1.740 | 1.385 | 1.001 | 0.705 | 0.501 | 0.364 | 0.271 |
| *φ*(*xi*) | 1.151 | 1.100 | 1.049 | 0.998 | 0.947 | 0.896 | 0.845 | 0.794 | 0.743 | 0.692 | 0.641 |
| (*φ*(*xi*) − *yi*)2 | 1.326 | 0.146 | 0.112 | 0.625 | 0.628 | 0.239 | 0.024 | 0.008 | 0.058 | 0.107 | 0.137 |

Вычислим среднеквадратичное отклонение для полученной аппроксимации

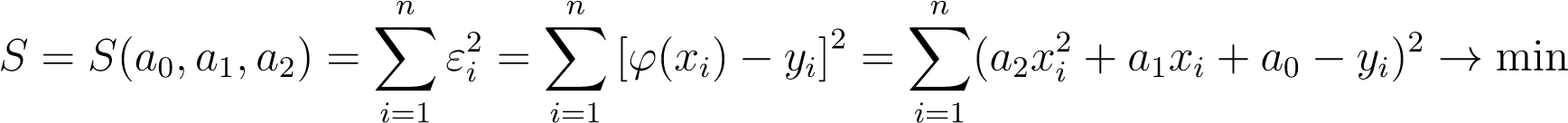


### Квадратичная аппроксимация

Найдём квадратичное приближение данной функции

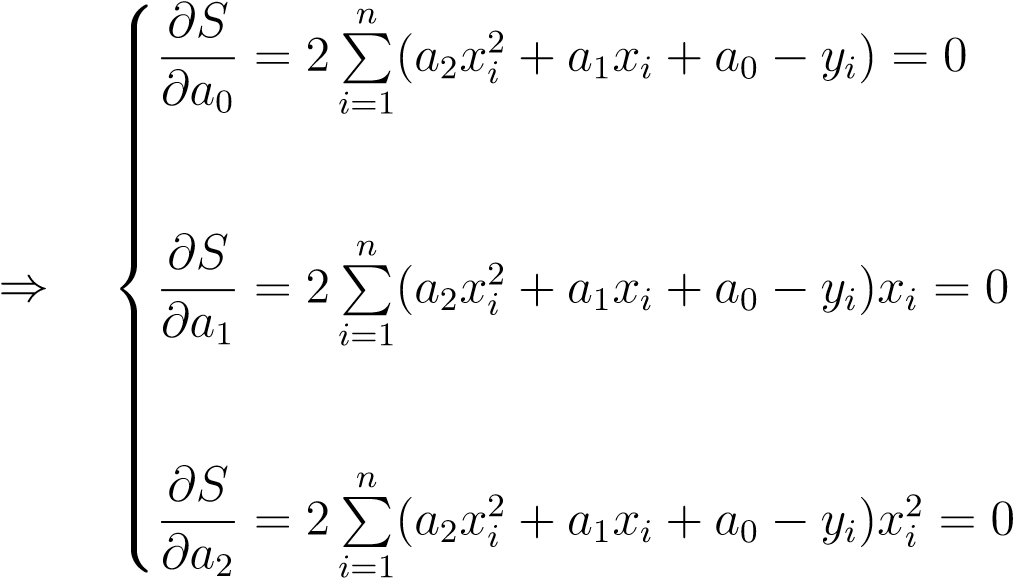
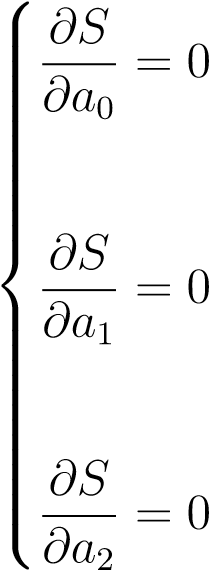
*ϕ*(*x*) = *a*2*x*2 + *a*1*x* + *a*0

Сумма квадратов отклонений запишется следующим образом:

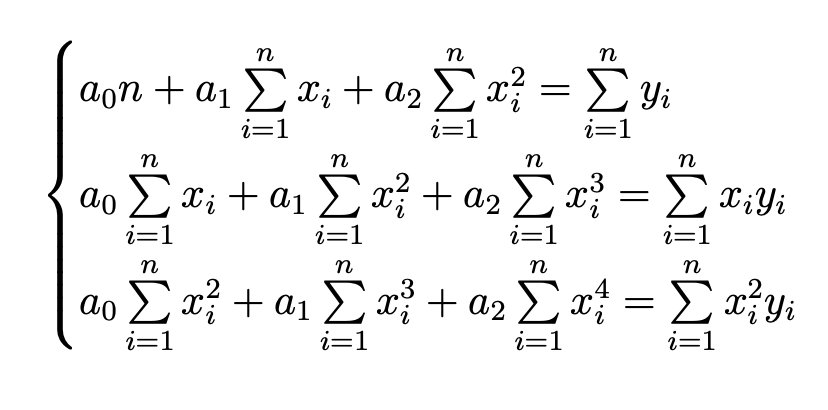


Для нахождения *a*0*,a*1 и *a*2 необходимо найти минимум функции *S*(*a*0*,a*1*,a*2).

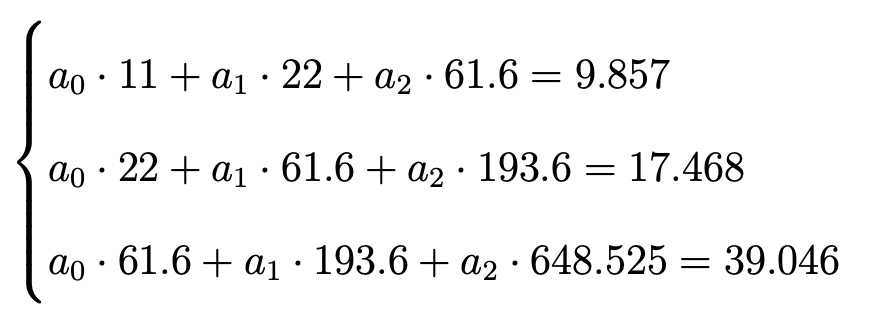
Необходимое условие существования минимума для функции *S*:



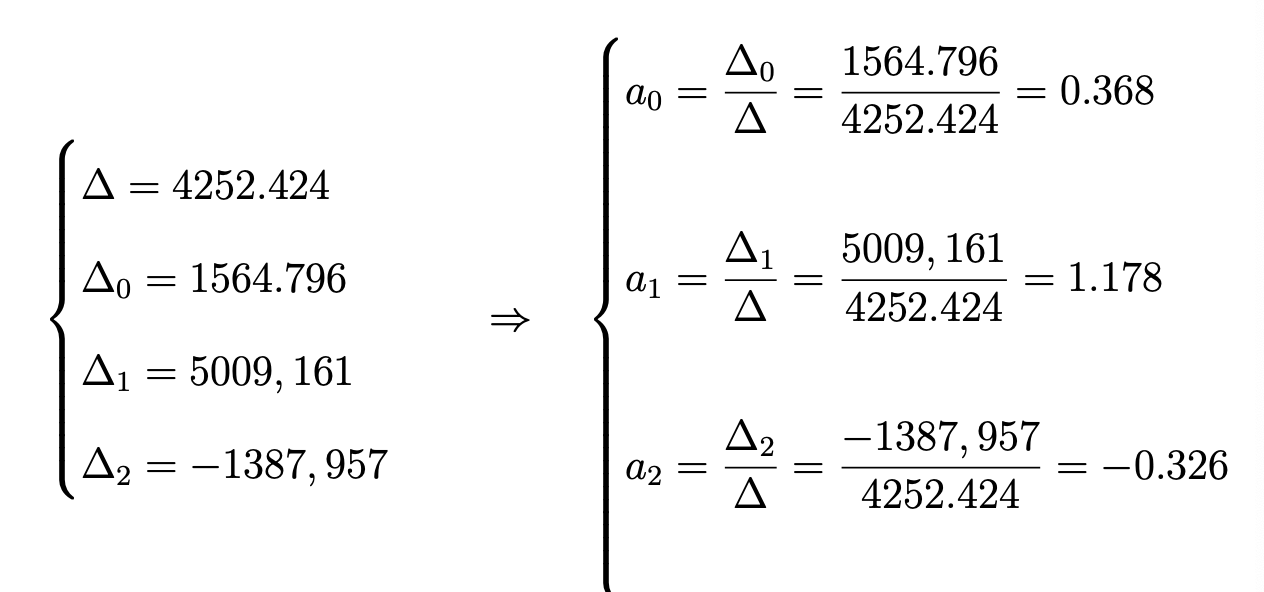
Упростим полученную систему:



Поставив соответствующие значения получаем систему:



из которой по правилу Крамера получаем:

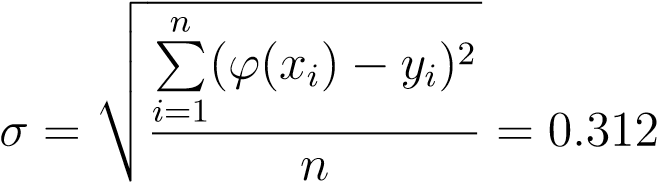


Таким образом аппроксимирующая квадратичная функция имеет вид:

*ϕ*(*x*) = −0*.*326*x*2 + 1*.*178*x* + 0*.*368

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *i* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| *xi* | 0.0 | 0.4 | 0.8 | 1.2 | 1.6 | 2.0 | 2.4 | 2.8 | 3.2 | 3.6 | 4.0 |
| *yi* | 0.000 | 0.718 | 1.383 | 1.789 | 1.740 | 1.385 | 1.001 | 0.705 | 0.501 | 0.364 | 0.271 |
| *ϕ*(*xi*) | 0.368 | 0.787 | 1.101 | 1.312 | 1.417 | 1.418 | 1.315 | 1.107 | 0.795 | 0.379 | -0.142 |
| (*ϕ*(*xi*) − *yi*)2 | 0.135 | 0.005 | 0.079 | 0.228 | 0.104 | 0.001 | 0.099 | 0.162 | 0.086 | 0.000 | 0.171 |

Вычислим среднеквадратичное отклонение для полученной аппроксимации



### Выбор лучшего приближения

Сравнив среднеквадратичные отклонения для линейного и квадратичного приближений становится понятно, что для квадратичной аппроксимации оно меньше, а значит эта аппроксимация является более точной.

### Графики функций



## Программная реализация задачи

### Листинг программы

|  |
| --- |
| def linear\_approximation(x, y, n): if n < 2:  raise Exception('Должно быть минимум 2 точки')  sx = sum(x)  sxx = sum(xi \*\* 2 for xi in x) sy = sum(y) sxy = sum(xi \* yi for xi, yi in zip(x, y))  try:  a, b = np.linalg.solve( [  [sxx, sx],  [sx, n]  ],  [sxy, sy]  ) except np.linalg.LinAlgError: raise Exception('Не удалось подобрать коэффициенты')  phi = lambda x\_: a \* x\_ + b return phi, (a, b)  def square\_approximation(x, y, n): |

if n < 3:

raise Exception('Должно быть минимум 3 точки')

sx = sum(x) sxx = sum(xi \*\* 2 for xi in x) sxxx = sum(xi \*\* 3 for xi in x) sxxxx = sum(xi \*\* 4 for xi in x) sy = sum(y)

sxy = sum(xi \* yi for xi, yi in zip(x, y)) sxxy = sum(xi \* xi \* yi for xi, yi in zip(x, y))

try:

a0, a1, a2 = np.linalg.solve( [

[n, sx, sxx],

[sx, sxx, sxxx],

[sxx, sxxx, sxxxx]

],

[sy, sxy, sxxy]

) except np.linalg.LinAlgError: raise Exception('Не удалось подобрать коэффициенты')

phi = lambda x\_: a2 \* x\_ \*\* 2 + a1 \* x\_ + a0

return phi, (a0, a1, a2)

def cubic\_approximation(xs, ys, n): if n < 4:

raise Exception('Должно быть минимум 4 точки')

sx = sum(xs) sxx = sum(xi \*\* 2 for xi in xs) sxxx = sum(xi \*\* 3 for xi in xs) sxxxx = sum(xi \*\* 4 for xi in xs) sxxxxx = sum(xi \*\* 5 for xi in xs) sxxxxxx = sum(xi \*\* 6 for xi in xs) sy = sum(ys)

sxy = sum(xi \* yi for xi, yi in zip(xs, ys)) sxxy = sum(xi \* xi \* yi for xi, yi in zip(xs, ys)) sxxxy = sum(xi \* xi \* xi \* yi for xi, yi in zip(xs, ys))

try:

a0, a1, a2, a3 = np.linalg.solve(

[

[n, sx, sxx, sxxx],

[sx, sxx, sxxx, sxxxx],

[sxx, sxxx, sxxxx, sxxxxx],

[sxxx, sxxxx, sxxxxx, sxxxxxx]

],

[sy, sxy, sxxy, sxxxy]

) except np.linalg.LinAlgError: raise Exception('Не удалось подобрать коэффициенты')

phi = lambda x\_: a3 \* x\_ \*\* 3 + a2 \* x\_ \*\* 2 + a1 \* x\_ + a0

return phi, (a0, a1, a2, a3)

def exponential\_approximation(x, y, n): if n < 2:

raise Exception('Должно быть минимум 2 точки')

if min(y) <= 0:

raise ValueError('Аппроксимация возможна только для наборов точек

*,*→ у которых y > 0')

\_, (a\_, b\_) = linear\_approximation(x, np.log(y), n)

1. = a\_
2. = np.exp(b\_) phi = lambda x\_: b \* np.exp(a \* x\_) return phi, (a, b)

def logarithmic\_approximation(x, y, n): if n < 2:

raise Exception('Должно быть минимум 2 точки')

if min(x) <= 0:

raise ValueError('Аппроксимация возможна только для наборов точек

*,*→ у которых x > 0')

\_, (a\_, b\_) = linear\_approximation(np.log(x), y, n)

a = a\_ b = b\_

phi = lambda x\_: a \* np.log(np.clip(x\_, 1e-10, None)) + b return phi, (a, b)

def power\_approximation(x, y, n): if n < 2:

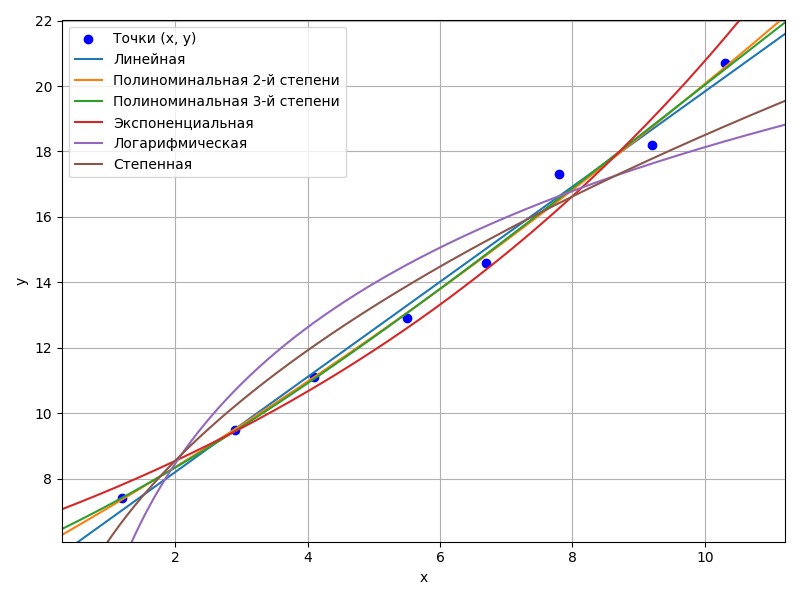
raise Exception('Должно быть минимум 2 точки') if min(x) <= 0 or min(y) <= 0:

|  |
| --- |
| raise ValueError('Аппроксимация возможна только для наборов точек  *,*→ у которых x > 0 и y > 0')  \_, (b\_, a\_) = linear\_approximation(np.log(x), np.log(y), n)  a = np.exp(a\_) b = b\_  def phi(x\_): if b < 0:  x\_ = np.where(x\_ != 0, x\_, 1e-10)  if abs(b) < 1:  x\_ = np.clip(x\_, 1e-10, None) return a \* np.power(x\_, b) return phi, (a, b) |

### Результаты выполнения программы

|  |
| --- |
| Выберите способ ввода:   1. -> Консоль 2. -> Файл   1  Вводите точки, по одной в строке. По окончании ввода введите q  1.2 7.4  2.9 9.5  4.1 11.1  5.5 12.9  6.7 14.6  7.8 17.3  9.2 18.2 10.3 20.7 q  Выберите способ вывода ответа:   1. -> Консоль 2. -> Файл   1  ==================================================  Аппроксимирующая функция: Линейная  Функция: phi(x) = 1.454x + 5.291  Среднеквадратичное отклонение: sigma = 0.410  Коэффициент детерминации: R² = 0.991  Мера отклонения: S = 1.346  Коэффициент кореляции Пирсона: r = 0.995  ================================================== |

|  |
| --- |
| Аппроксимирующая функция: Полиноминальная 2-й степени  Функция: phi(x) = 0.026x^2 + 1.153x + 5.943  Среднеквадратичное отклонение: sigma = 0.356  Коэффициент детерминации: R² = 0.993  Мера отклонения: S = 1.016  ==================================================  Аппроксимирующая функция: Полиноминальная 3-й степени  Функция: phi(x) = -0.002x^3 + 0.067x^2 + 0.955x + 6.178  Среднеквадратичное отклонение: sigma = 0.353  Коэффициент детерминации: R² = 0.993  Мера отклонения: S = 1.000  ==================================================  Аппроксимирующая функция: Экспоненциальная  Функция: phi(x) = 6.840 \* e^0.111x  Среднеквадратичное отклонение: sigma = 0.583  Коэффициент детерминации: R² = 0.982  Мера отклонения: S = 2.719  ==================================================  Аппроксимирующая функция: Логарифмическая  Функция: phi(x) = 6.009 \* ln(x) + 4.296  Среднеквадратичное отклонение: sigma = 1.529  Коэффициент детерминации: R² = 0.873  Мера отклонения: S = 18.693  ==================================================  Аппроксимирующая функция: Степенная  Функция: phi(x) = 6.129 \* x^0.480  Среднеквадратичное отклонение: sigma = 1.003  Коэффициент детерминации: R² = 0.945  Мера отклонения: S = 8.046  ==================================================  Лучшая аппроксимирующая функция: Полиноминальная 3-й степени |



# **Вывод**

В ходе выполнения данной лабораторной работы я научился выполнять аппроксимацию таблично заданных функций, используй линейное, квадратичное, кубическое, логарифмическое, экспоненциальное и показательное приближения. Используя эти знания, я написал программу на языке python, который определяет коэффициенты приближений из описанного выше списка функций. Также для каждой из функций аппроксимации вычисляются среднеквадратичное отклонение, коэффициент детерминации и мера отклонения. Для линейной зависимости также вычисляется коэффициент корреляции Пирсона. Когда все функции определены программа строит график, на котором изображены введённые пользователем точки и графики аппроксимирующих функций.