Федеральное государственное автономное образовательное учреждение высшего образования Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа №4

Вариант № 1

Выполнил: студент группы P3208, Васильев Н. А.

Преподаватель: Машина Е.А.

Санкт-Петербург 2025

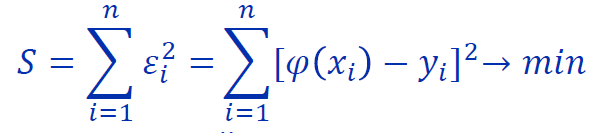
# Текст задания

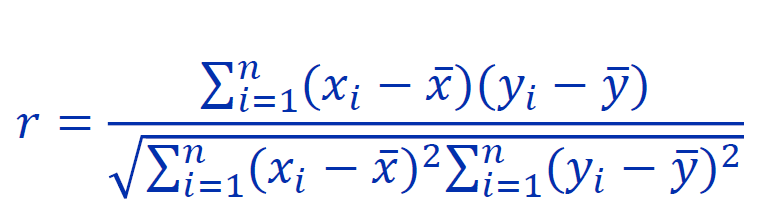
Аппроксимация функции методом наименьших квадратов.

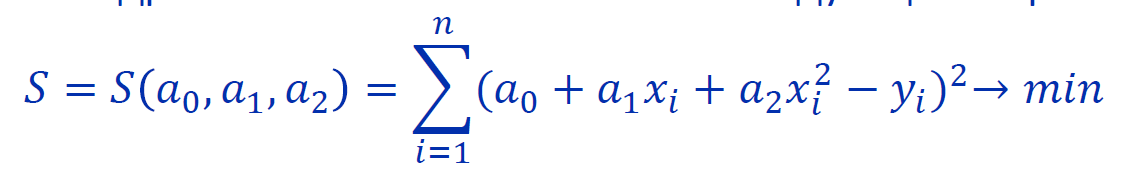
Цель работы

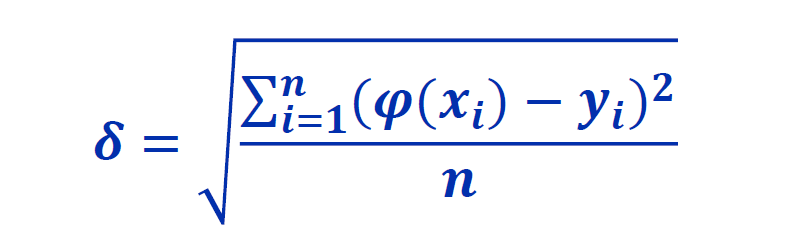
Найти функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов.

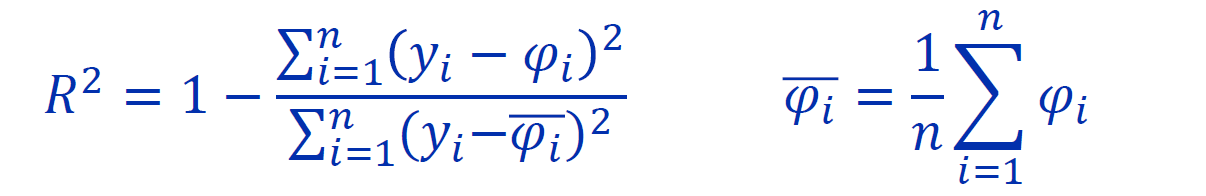
Описание метода, расчётные формулы











Вычислительная реализация задачи:

Функция:

Исследуемый интервал:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

Линейное приближение:

Введем обозначения:

Получаем:

Тогда:

Вычислим значения аппроксимирующей функции:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

Квадратичное приближение:

Введем обозначения:

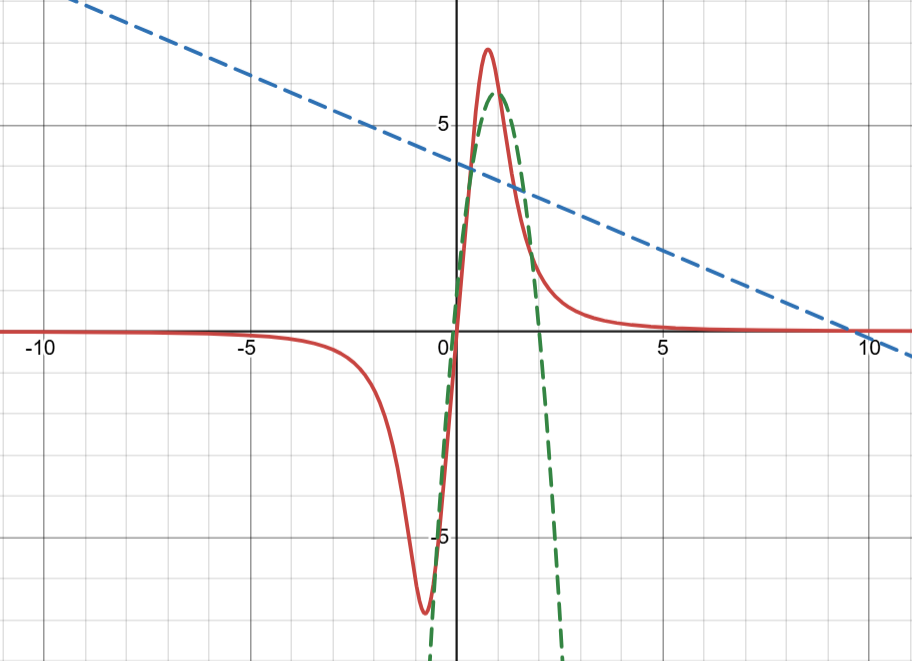
Получаем:

Тогда:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

Сравним полученные результаты:

Следовательно, квадратичное приближение лучше.



Листинг программы

import numpy as np  
  
  
def linear\_approximation(x, y, n):  
 x = np.array(x)  
 y = np.array(y)  
  
 sx = np.sum(x)  
 sxx = np.sum(x \*\* 2)  
 sy = np.sum(y)  
 sxy = np.sum(x \* y)  
  
 a, b = np.linalg.solve(  
 [  
 [n, sx],  
 [sx, sxx]  
 ],  
 [sy, sxy])  
 return lambda xi: a + b \* xi, a.item(), b.item()  
  
  
def quadratic\_approximation(x, y, n):  
 x = np.array(x)  
 y = np.array(y)  
  
 sx = np.sum(x)  
 sxx = np.sum(x \*\* 2)  
 sxxx = np.sum(x \*\* 3)  
 sxxxx = np.sum(x \*\* 4)  
 sy = np.sum(y)  
 sxy = np.sum(x \* y)  
 sxxy = np.sum(x \*\* 2 \* y)  
  
 a, b, c = np.linalg.solve(  
 [  
 [n, sx, sxx],  
 [sx, sxx, sxxx],  
 [sxx, sxxx, sxxxx]  
 ],  
 [sy, sxy, sxxy]  
 )  
 return lambda xi: a + b \* xi + c \* xi \*\* 2, a.item(), b.item(), c.item()  
  
  
def cubic\_approximation(x, y, n):  
 x = np.array(x)  
 y = np.array(y)  
  
 sx = np.sum(x)  
 sy = np.sum(y)  
 sxy = np.sum(x \* y)  
 sxx = np.sum(x \*\* 2)  
 sxxx = np.sum(x \*\* 3)  
 sxxxx = np.sum(x \*\* 4)  
 sxxxxx = np.sum(x \*\* 5)  
 sxxxxxx = np.sum(x \*\* 6)  
 sxxy = np.sum(x \*\* 2 \* y)  
 sxxxy = np.sum(x \*\* 3 \* y)  
  
 a, b, c, d = np.linalg.solve(  
 [  
 [n, sx, sxx, sxxx],  
 [sx, sxx, sxxx, sxxxx],  
 [sxx, sxxx, sxxxx, sxxxxx],  
 [sxxx, sxxxx, sxxxxx, sxxxxxx]  
 ],  
 [sy, sxy, sxxy, sxxxy]  
 )  
 return lambda xi: a + b \* xi + c \* xi \*\* 2 + d \* xi \*\* 3, a.item(), b.item(), c.item(), d.item()  
  
def exponential\_approximation(x, y, n):  
 y\_log = np.log(y)  
 \_, a\_log, b\_log = linear\_approximation(x, y\_log, n)  
  
 a = np.exp(a\_log)  
 b = b\_log  
  
 return lambda xi: a \* np.exp(b \* xi), a.item(), b  
  
  
def logarithmic\_approximation(x, y, n):  
 x\_log = np.log(x)  
 \_, a, b = linear\_approximation(x\_log, y, n)  
  
 return lambda xi: a + b \* np.log(xi), a, b  
  
  
def power\_approximation(x, y, n):  
 x\_log = np.log(x)  
 y\_log = np.log(y)  
  
 \_, a\_log, b\_log = linear\_approximation(x\_log, y\_log, n)  
  
 a = np.exp(a\_log)  
 b = b\_log  
  
 return lambda xi: a \* xi \*\* b, a.item(), b

import numpy as np  
  
  
def compute\_pearson\_correlation(x, y):  
 av\_x = np.mean(x)  
 av\_y = np.mean(y)  
  
 numerator = np.sum((x - av\_x) \* (y - av\_y))  
 denominator = np.sqrt(np.sum((x - av\_x) \*\* 2) \* np.sum((y - av\_y) \*\* 2))  
  
 return numerator / denominator  
  
  
def compute\_mean\_squared\_error(x, y, phi):  
 x = np.array(x)  
 y = np.array(y)  
  
 errors = phi(x) - y  
 mse = np.mean(errors \*\* 2)  
 return np.sqrt(mse)  
  
  
def compute\_measure\_of\_deviation(x, y, phi):  
 x = np.array(x)  
 y = np.array(y)  
  
 deviations = phi(x) - y  
 return np.sum(deviations \*\* 2)  
  
  
def compute\_coefficient\_of\_determination(x, y, phi, n):  
 x = np.array(x)  
 y = np.array(y)  
  
 av\_phi = np.sum(phi(x)) / n  
 total\_variation = np.sum((y - av\_phi) \*\* 2)  
 unexplained\_variation = np.sum((y - phi(x)) \*\* 2)  
  
 return 1 - (unexplained\_variation / total\_variation)

Примеры и результаты работы программы

**Пример 1:** Точки: **(**0.2; 2.396), (0.4; 4.68), (0.6; 6.374), (0.8; 6.810), (1; 6), (1.2; 4.685), (1.4; 3.47), (1.6; 2.542), (1.8; 1.879), (2; 1.412):

Линейная аппроксимация:

Коэффициенты: [5.9901, -1.7866]

Среднеквадратичное отклонение: 1.55241

Коэффициент детерминации: 0.30414

Мера отклонения: S = 24.09984

Коэффициент корреляции Пирсона: r = -0.5514913913498515

Полиноминальная 2-й степени аппроксимация:

Коэффициенты: [2.1126, 7.9071, -4.4063]

Среднеквадратичное отклонение: 0.87738

Коэффициент детерминации: 0.77773

Мера отклонения: S = 7.69802

Полиноминальная 3-й степени аппроксимация:

Коэффициенты: [-2.0762, 26.4832, -24.5446, 6.1025]

Среднеквадратичное отклонение: 0.18335

Коэффициент детерминации: 0.99029

Мера отклонения: S = 0.33616

Экспоненциальная аппроксимация:

Коэффициенты: [6.4761, -0.5459]

Среднеквадратичное отклонение: 1.72108

Коэффициент детерминации: 0.16569

Мера отклонения: S = 29.62102

Логарифмическая аппроксимация:

Коэффициенты: [3.9504, -0.7519]

Среднеквадратичное отклонение: 1.78604

Коэффициент детерминации: 0.07894

Мера отклонения: S = 31.89952

Степенная аппроксимация:

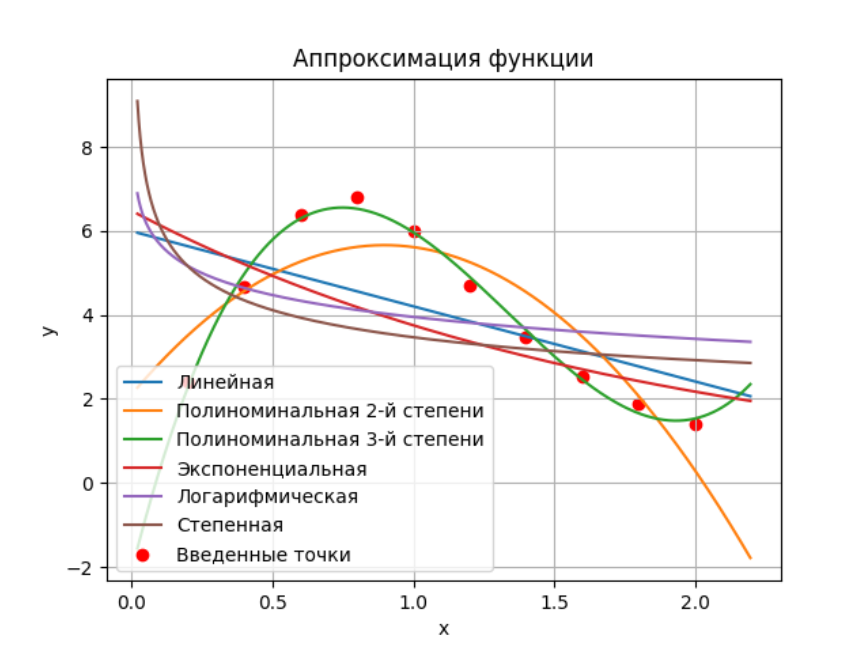
Коэффициенты: [3.467, -0.2463]

Среднеквадратичное отклонение: 1.89193

Коэффициент детерминации: 0.01593

Мера отклонения: S = 35.79388

Лучшая функция приближения: Полиноминальная 3-й степени



Вывод

В ходе выполнения лабораторной работы была реализована аппроксимация таблично заданной функции с использованием метода наименьших квадратов. Были подобраны коэффициенты приближающей функции заданного вида (линейной, квадратичной), минимизирующей сумму квадратов отклонений между значениями аппроксимирующей функции и исходными данными.

Результатом работы стала функция, наилучшим образом аппроксимирующая исходные данные в смысле наименьших квадратов. Полученные значения коэффициентов и графическое сравнение аппроксимирующей функции с табличными данными подтвердили адекватность выбранной модели.

Таким образом, была достигнута поставленная цель — построена аппроксимирующая функция, максимально приближённая к заданным данным по критерию наименьших квадратов.

Была реализована программа, позволяющая вычислять аппроксимацию для функции по заданным точкам.