«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет программной инженерии и компьютерной техники  
Направление подготовки 09.03.04 Программная инженерия  
Дисциплина «Вычислительная математика»

Отчет

По лабораторной работе №4

Вариант 1

Студент:

Алхимовици А.

Р3210

Преподаватель:

Наумова Н. А.

Санкт-Петербург, 2025 г.

**Цель лабораторной работы**:

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

# **1 Вычислительная реализация задачи:**

Линейная аппроксимация:

y =  **;** n = 11 ; x [0; 2] ; h = 0.2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| xi | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 1.2 | 1.4 | 1.6 | 1.8 | 2.0 |
| yi | 0 | 2.396 | 4.68 | 6.374 | 6.81 | 6 | 4.685 | 3.47 | 2.542 | 1.879 | 1.412 |

φ(x) = a + bx

Вычисляем суммы: sx = 11, sxx = 15.4, sy = 40.25, sxy = 38.38

φ(x) =

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| xi | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 1.2 | 1.4 | 1.6 | 1.8 | 2.0 |
| yi | 0 | 2.396 | 4.68 | 6.374 | 6.81 | 6 | 4.685 | 3.47 | 2.542 | 1.879 | 1.412 |
| φ(xi) | 4.084 | 3.999 | 3,914 | 3,829 | 3,744 | 3,659 | 3,574 | 3,489 | 3.404 | 3.319 | 3.234 |
| (φ(xi)- yi)^2 | 16.679 | 2.569 | 0.587 | 6,477 | 9,403 | 5,480 | 1,234 | 0 | 0,743 | 2.075 | 3.321 |

**σ = = 2.101**

Квадратичная аппроксимация:

y =  **;** n = 11 ; x [0; 2] ; h = 0.2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| xi | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 1.2 | 1.4 | 1.6 | 1.8 | 2.0 |
| yi | 0 | 2.396 | 4.68 | 6.374 | 6.81 | 6 | 4.685 | 3.47 | 2.542 | 1.879 | 1.412 |

φ(x) = a + bx + cx2

Вычисляем суммы:

sx = 11, sxx = 15.4, sxxx = 24,2; sxxxx = 40,53; sy = 40,25; sxy = 38,38; sxxy = 45,29

φ(x)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| xi | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 1.2 | 1.4 | 1.6 | 1.8 | 2.0 |
| yi | 0 | 2.396 | 4.68 | 6.374 | 6.81 | 6 | 4.685 | 3.47 | 2.542 | 1.879 | 1.412 |
| φ(xi) | 0.878 | 2.716 | 4.128 | 5.111 | 5.667 | 5.796 | 5.497 | 4.771 | 3.618 | 2.036 | 0.028 |
| (φ(xi)- yi)^2 | 0.771 | 0.103 | 0.305 | 1.595 | 1.307 | 0.042 | 0.660 | 1.693 | 1.157 | 0.025 | 1.915 |

**σ = = 0.933**

0,933 < 2,101 у квадратичной аппроксимации среднеквадратичное отклонение меньше, поэтому это приближение лучше.

A graph on a grid

AI-generated content may be incorrect.

A graph with a line and a blue line

AI-generated content may be incorrect.

A graph with a blue line and black dots

AI-generated content may be incorrect.

# **2. Программная реализация**

# **Блок схемы**

Линейная фукнция

A diagram of a algorithm

AI-generated content may be incorrect.Полиномиальная функция 2-й степени

A diagram of a program

AI-generated content may be incorrect.

Полиномиальная функция 3-й степени

A diagram of a algorithm

AI-generated content may be incorrect.

Экспоненциальная функция: y = a \* exp(b\*x)

A diagram of a algorithm

AI-generated content may be incorrect.

Логарифмическая функция

A diagram of a function

AI-generated content may be incorrect.

Степенная функция

A diagram of a program

AI-generated content may be incorrect.

# **Листинг программы**

<https://github.com/senya-2011/Vu4Math/tree/main/lab4>

|  |
| --- |
| def calc\_measure\_of\_deviation(errors):  squared = [error \*\* 2 for error in errors]  return sum(squared) / len(squared)  # линейная аппроксимация: y = a\*x + b def approx\_linear(x\_list, y\_list):  design = [[x, 1.0] for x in x\_list]  return logic.solve\_normal\_equations(design, y\_list)   # квадратичная аппроксимация: y = a\*x^2 + b\*x + c def approx\_quadratic(x\_list, y\_list):  design = [[x \*\* 2, x, 1.0] for x in x\_list]  return logic.solve\_normal\_equations(design, y\_list)   # кубическая аппроксимация: y = a\*x^3 + b\*x^2 + c\*x + d def approx\_cubic(x\_list, y\_list):  design = [[x \*\* 3, x \*\* 2, x, 1.0] for x in x\_list]  return logic.solve\_normal\_equations(design, y\_list)   # экспоненциальная аппроксимация: y = a \* exp(b\*x) def approx\_exponential(x\_list, y\_list):  # логарифмируем y, чтобы получить линейную зависимость ln(y) = ln(a) + b\*x  design = [[1.0, x] for x in x\_list]  y\_log = [math.log(y\_i) for y\_i in y\_list]  ln\_coeffs = logic.solve\_normal\_equations(design, y\_log)  a = math.exp(ln\_coeffs[0]) # восстановление a  b = ln\_coeffs[1]  return [a, b]   # логарифмическая аппроксимация: y = a + b\*ln(x) def approx\_logarithmic(x\_list, y\_list):  design = [[1.0, math.log(x)] for x in x\_list]  return logic.solve\_normal\_equations(design, y\_list)   # степенная аппроксимация: y = a \* x^b def approx\_power(x\_list, y\_list):  # берем логарифмы: ln(y) = ln(a) + b\*ln(x)  design = [[1.0, math.log(x)] for x in x\_list]  y\_log = [math.log(y\_i) for y\_i in y\_list]  ln\_coeffs = logic.solve\_normal\_equations(design, y\_log)  a = math.exp(ln\_coeffs[0])  b = ln\_coeffs[1]  return [a, b]  def solve\_normal\_equations(design\_matrix, y\_vector):  Xt = transpose\_matrix(design\_matrix)  XtX = mat\_mat\_mul(Xt, design\_matrix)  XtY = mat\_vec\_mul(Xt, y\_vector)  return solve\_gauss(XtX, XtY)  def solve\_gauss(A, b, epsilon=1e-10):  n = len(A)  # Формируем расширенную матрицу  M = [row[:] + [b\_i] for row, b\_i in zip(A, b)]  # Прямой ход  for i in range(n):  # Поиск ведущего элемента  max\_row, max\_val = i, abs(M[i][i])  for r in range(i+1, n):  if abs(M[r][i]) > max\_val:  max\_row, max\_val = r, abs(M[r][i])  M[i], M[max\_row] = M[max\_row], M[i]  pivot = M[i][i]  for j in range(i, n+1): M[i][j] /= pivot  for k in range(i+1, n):  factor = M[k][i]  for j in range(i, n+1):  M[k][j] -= factor \* M[i][j]  # Обратный ход  x = [0.0]\*n  for i in range(n-1, -1, -1):  x[i] = M[i][n]  for j in range(i+1, n):  x[i] -= M[i][j]\*x[j]  x[i] /= M[i][i]  return x  def calc\_r2\_and\_pearson(x\_list, y\_list):  predictions = generate\_all\_predictions(x\_list, y\_list)  mean\_y = sum(y\_list) / len(y\_list)  stats = []  for name, coeffs, y\_pred in predictions:  ss\_res = sum((y - yp) \*\* 2 for y, yp in zip(y\_list, y\_pred))  ss\_tot = sum((y - mean\_y) \*\* 2 for y in y\_list)  r2 = 1 - ss\_res / ss\_tot if ss\_tot != 0 else None  pearson\_r = None  if name == 'Linear':  mean\_x = sum(x\_list) / len(x\_list)  cov = var\_x = var\_y = 0.0  for x, y in zip(x\_list, y\_list):  dx = x - mean\_x  dy = y - (sum(y\_list) / len(y\_list))  cov += dx \* dy  var\_x += dx \*\* 2  var\_y += dy \*\* 2  pearson\_r = cov / math.sqrt(var\_x \* var\_y) if var\_x and var\_y else None  if r2 is None:  reliability = 'Невозможно вычислить R²'  elif r2 >= 0.95:  reliability = 'Высокая точность аппроксимации (модель хорошо описывает явление)'  elif r2 >= 0.75:  reliability = 'Удовлетворительная аппроксимация (модель в целом адекватно описывает явление)'  elif r2 >= 0.5:  reliability = 'Слабая аппроксимация (модель слабо описывает явление)'  else:  reliability = 'Точность аппроксимации недостаточна и модель требует изменения'  stats.append({  'name': name,  'r2': r2,  'pearson\_r': pearson\_r,  'reliability': reliability  })  return stats |

## 

## **Примеры и результаты работы программы**

## A screenshot of a computer AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a graph

AI-generated content may be incorrect.

# **Выводы**:

В ходе работы мы реализовали шесть методов аппроксимации (линейный, квадратичный, кубический, экспоненциальный, логарифмический и степенной), вычислили для каждой модели MSE, а для линейной — ещё и коэффициент Пирсона. По итогам тестов степенная функция наиболее точная и универсальная, но для каждой из функций можно подобрать тест, где она покажет себя лучше всего. В итоге получен удобный инструмент для быстрого подбора и визуализации оптимальной модели приближения данных.