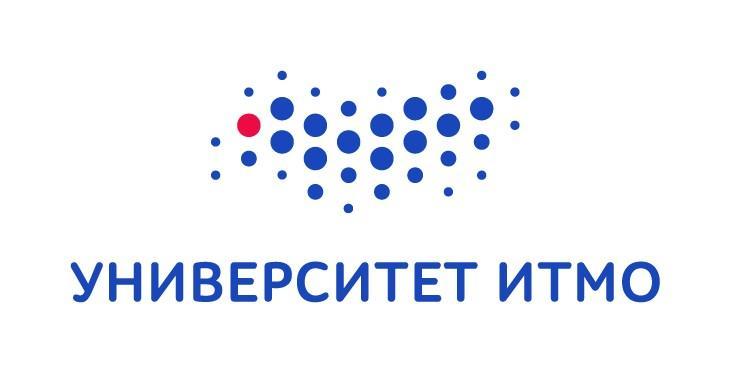
**НИУ ИТМО**

**Факультет ПиИКТ**

****

**Вычислительная математика**

**Лабораторная работа № 4**

Аппроксимация функции методом наименьших квадратов

Вариант № 7

Работу выполнил: Иванов Евгений

Группа: Р3213

Преподаватель: Малышева Татьяна Алексеевна

**Санкт-Петербург**

**2022 г.**

**Цель лабораторной работы**: найти функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов.

***Вычислительная реализация:***

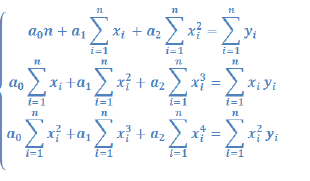
|  |  |
| --- | --- |
|  |  |

Таблица значений:

|  |  |
| --- | --- |
| X | Y |
| -2 | -0.3158 |
| -1.8 | -0.4001 |
| -1.6 | -0.5024 |
| -1.4 | -0.6139 |
| -1.2 | -0.7096 |
| -1.0 | -0.75 |
| -0.8 | -0.7039 |
| -0.6 | -0.5752 |
| -0.4 | -0.3966 |
| -0.2 | -0.1999 |
| 0 | 0 |

**Квадратичная аппроксимация**

Решим следующую систему:



Полученные результаты:

a0 = -0.048

a1 = 2.286

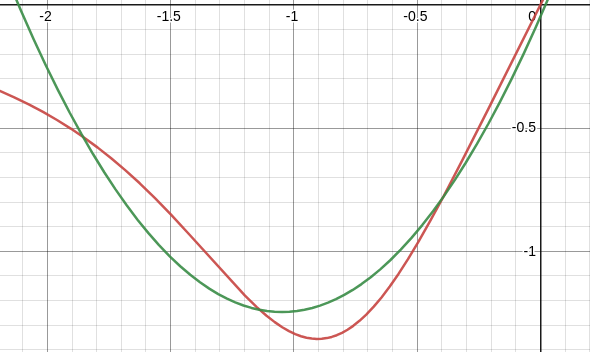
a2 = 1.090

**Полученная аппроксимация: 1.090x^2 + 2.286x + -0.048**

|  |  |  |
| --- | --- | --- |
| X | Y | eps |
| -2 | -0.4444 | 0.03450033475671183 |
| -1.8 | -0.5761 | 0.002918870636216922 |
| -1.6 | -0.7482 | 0.027611903382615835 |
| -1.4 | -0.9586 | 0.02334249764829586 |
| -1.2 | -1.1783 | 0.0018377709467455909 |
| -1.0 | -1.3333 | 0.008023064678196707 |
| -0.8 | -1.328 | 0.022183081938481237 |
| -0.6 | -1.127 | 0.009967159873343625 |
| -0.4 | -0.7899 | 3.4557856673305434E-6 |
| -0.2 | -0.3997 | 0.0038427861098339205 |
| 0 | 0 | 0.002314753777690516 |

Мера отклонения: 0.136546

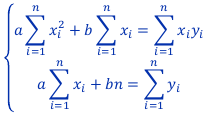
График



**Линейная аппроксимация**

**-**11

Решим следующую систему:



Полученные результаты:

a = 0.1053

b = -0.7023

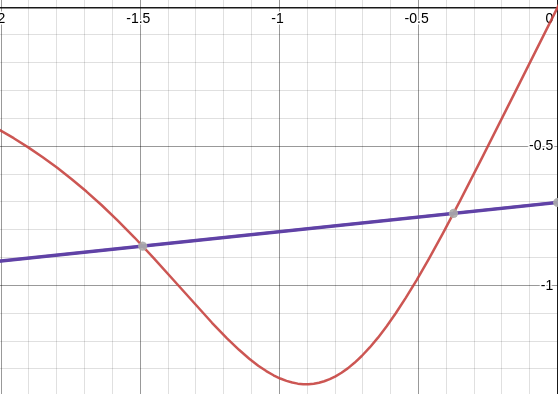
**Полученная аппроксимация: 0.1053x - 0.7023**

|  |  |  |
| --- | --- | --- |
| X | Y | eps |
| -2 | -0.4444 | 0.21945817859504158 |
| -1.8 | -0.5761 | 0.09967223008264478 |
| -1.6 | -0.7482 | 0.01501961661157028 |
| -1.4 | -0.9586 | 0.011859209999999999 |
| -1.2 | -1.1783 | 0.12225830115702478 |
| -1.0 | -1.3333 | 0.27637004826446293 |
| -0.8 | -1.328 | 0.2931828695041326 |
| -0.6 | -1.127 | 0.13069539578512424 |
| -0.4 | -0.7899 | 0.0020677689256198804 |
| -0.2 | -0.3997 | 0.10476403438016493 |
| 0 | 0 | 0.49325082851239577 |

Мера отклонения: 0.17686

Коэффициент корреляции: 0.1638

График



**Программная реализация**

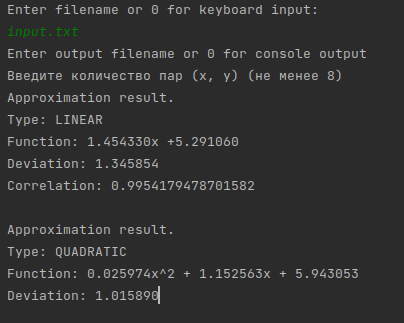
|  |
| --- |
| **public ApproximationResult squareApproximation(double[][] functionTable) {  double x\_sum = 0, x2\_sum = 0, x3\_sum = 0, x4\_sum = 0,  y\_sum = 0, xy\_sum = 0, x2y\_sum = 0;   for (int i = 0; i < functionTable.length; i++) {  x\_sum += functionTable[i][0];  x2\_sum += Math.pow(functionTable[i][0], 2);  x3\_sum += Math.pow(functionTable[i][0], 3);  x4\_sum += Math.pow(functionTable[i][0], 4);  y\_sum += functionTable[i][1];  xy\_sum += functionTable[i][0] \* functionTable[i][1];  x2y\_sum += Math.pow(functionTable[i][0], 2) \* functionTable[i][1];  }   double[][] matrix = new double[][] {  {functionTable.length, x\_sum, x2\_sum},  {x\_sum, x2\_sum, x3\_sum},  {x2\_sum, x3\_sum, x4\_sum}  };   double[] constants = new double[] {y\_sum, xy\_sum, x2y\_sum};  System.out.println(Arrays.deepToString(matrix));  System.out.println(Arrays.toString(constants));  double[] solution = solveLinearSystem(matrix, constants);  reverseArray(solution);  Function<Double, Double> function = coefficientsToSquareFunction(solution);  double deviation = deviationMeasure(functionTable, function);  return new ApproximationResult(ApproximationType.QUADRATIC, solution, function, deviation); }** |

|  |
| --- |
| **public ApproximationResult exponentialApproximation(double[][] functionTable) {  double[][] modifiedFunctionTable = Arrays.stream(functionTable).map(double[]::clone).toArray(double[][]::new);  for (double[] xy: modifiedFunctionTable) {  if (xy[1] <= 0) continue;  xy[1] = Math.log(xy[1]);  }  ApproximationResult linear = linearApproximation(modifiedFunctionTable);  double[] coefficients = linear.getCoefficients();  coefficients[1] = Math.exp(coefficients[1]);  Function<Double, Double> f = coefficientsToExpFunction(coefficients);  return new ApproximationResult(ApproximationType.EXPONENTIAL, coefficients, f, deviationMeasure(functionTable, f)); }  public ApproximationResult logarithmicApproximation(double[][] functionTable) {  double[][] modifiedFunctionTable = Arrays.stream(functionTable).map(double[]::clone).toArray(double[][]::new);  for (double[] xy: modifiedFunctionTable) {  xy[0] = Math.log(xy[0]);  }  ApproximationResult linear = linearApproximation(modifiedFunctionTable);  double[] coefficients = linear.getCoefficients();  Function<Double, Double> f = coefficientsToLogFunction(coefficients);  return new ApproximationResult(ApproximationType.LOGARITHMIC, coefficients, f, deviationMeasure(functionTable, f)); }  public ApproximationResult powerApproximation(double[][] functionTable) {  double[][] modifiedFunctionTable = Arrays.stream(functionTable).map(double[]::clone).toArray(double[][]::new);  for (double[] xy: modifiedFunctionTable) {  xy[0] = Math.log(xy[0]);  xy[1] = Math.log(xy[1]);  }  ApproximationResult linear = linearApproximation(modifiedFunctionTable);  double[] coefficients = linear.getCoefficients();  coefficients[1] = Math.exp(coefficients[1]);  Function<Double, Double> f = coefficientsToPowerFunction(coefficients);  return new ApproximationResult(ApproximationType.POWER, coefficients, f, deviationMeasure(functionTable, f)); }  public ApproximationResult cubicApproximation(double[][] functionTable) {  double x\_sum = 0, x2\_sum = 0, x3\_sum = 0, x4\_sum = 0, x5\_sum = 0, x6\_sum = 0,  y\_sum = 0, xy\_sum = 0, x2y\_sum = 0, x3y\_sum = 0;  for (int i = 0; i < functionTable.length; i++) {  x\_sum += functionTable[i][0];  x2\_sum += Math.pow(functionTable[i][0], 2);  x3\_sum += Math.pow(functionTable[i][0], 3);  x4\_sum += Math.pow(functionTable[i][0], 4);  x5\_sum += Math.pow(functionTable[i][0], 5);  x6\_sum += Math.pow(functionTable[i][0], 6);  y\_sum += functionTable[i][1];  xy\_sum += functionTable[i][0] \* functionTable[i][1];  x2y\_sum += Math.pow(functionTable[i][0], 2) \* functionTable[i][1];  x3y\_sum += Math.pow(functionTable[i][0], 3) \* functionTable[i][1];  }   double[][] matrix = new double[][] {  {functionTable.length, x\_sum, x2\_sum, x3\_sum},  {x\_sum, x2\_sum, x3\_sum, x4\_sum},  {x2\_sum, x3\_sum, x4\_sum, x5\_sum},  {x3\_sum, x4\_sum, x5\_sum, x6\_sum}  };   double[] constants = new double[] {y\_sum, xy\_sum, x2y\_sum, x3y\_sum};  double[] solution = solveLinearSystem(matrix, constants);  reverseArray(solution);  Function<Double, Double> function = coefficientsToCubicFunction(solution);  double deviation = deviationMeasure(functionTable, function);  return new ApproximationResult(ApproximationType.CUBIC, solution, function, deviation); }** |

**input.txt:**

|  |
| --- |
| **0 8 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** |

**Вывод программы:**

****

**Вывод**: в ходе данной л.р я познакомился с различными видами аппроксимации функции через метод наименьших квадратов. Данный метод довольно просто программируем.