Университет ИТМО

Мегафакультет компьютерных технологий и управления

Факультет программной инженерии и компьютерной техники

Изображение выглядит как Шрифт, текст, белый, Графика

Автоматически созданное описание

ЛАБОРАТОРНАЯ РАБОТА №4

АППРОКСИМАЦИЯ ФУНКЦИИ МЕТОДОМ НАИМЕНЬШИХ КВАДРАТОВ

Вариант №10

Группа: P3211

Студент: Орчиков Даниил Валерьевич

Преподаватель: Малышева Татьяна Алексеевна

г. Санкт-Петербург

2024

Оглавление

[Цель работы 2](#_Toc168266142)

[Вычислительная реализация задачи: 2](#_Toc168266143)

[Рабочие формулы 2](#_Toc168266144)

[Решение 2](#_Toc168266145)

[Линейное приближение 2](#_Toc168266146)

[Квадратичное приближение 3](#_Toc168266147)

[Программная реализация задачи 4](#_Toc168266148)

[Листинг программы 4](#_Toc168266149)

[Примеры и результаты работы программы 7](#_Toc168266150)

[Пример 1 7](#_Toc168266151)

[Пример 2 8](#_Toc168266152)

[Пример 3 10](#_Toc168266153)

[Вывод 12](#_Toc168266154)

Цель работы

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

# Вычислительная реализация задачи:

## Рабочие формулы

## Решение

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| X | 0 | 0.4 | 0.8 | 1.2 | 1.6 | 2 | 2.4 | 2.8 | 3.2 | 3.6 | 4 |
| Y | 0 | 0.718 | 1.383 | 1.789 | 1.74 | 1.385 | 1.001 | 0.705 | 0.501 | 0.364 | 0.271 |

### Линейное приближение

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| X | 0 | 0.4 | 0.8 | 1.2 | 1.6 | 2 | 2.4 | 2.8 | 3.2 | 3.6 | 4 |
| Y | 0 | 0.718 | 1.383 | 1.789 | 1.74 | 1.385 | 1.001 | 0.705 | 0.501 | 0.364 | 0.271 |
|  | 1.151 | 1.1 | 1.049 | 0.997 | 0.946 | 0.895 | 0.844 | 0.793 | 0.741 | 0.69 | 0.639 |
|  | 1.151 | 0.382 | -0.335 | -0.792 | -0.794 | -0.49 | -0.157 | 0.087 | 0.24 | 0.326 | 0.368 |

### Квадратичное приближение

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| X | 0 | 0.4 | 0.8 | 1.2 | 1.6 | 2 | 2.4 | 2.8 | 3.2 | 3.6 | 4 |
| Y | 0 | 0.718 | 1.383 | 1.789 | 1.74 | 1.385 | 1.001 | 0.705 | 0.501 | 0.364 | 0.271 |
|  | 0.368 | 0.787 | 1.102 | 1.312 | 1.418 | 1.42 | 1.317 | 1.111 | 0.799 | 0.384 | -0.136 |
|  | 0.368 | 0.069 | -0.282 | -0.477 | -0.322 | 0.035 | 0.317 | 0.405 | 0.298 | 0.02 | -0.407 |

Изображение выглядит как линия, График, диаграмма, Параллельный

Автоматически созданное описание

# Программная реализация задачи

## Листинг программы

Представлен только код, непосредственно выполняющий вычисления

Весь код можно посмотреть [тут (GitHub)](https://github.com/DaniilOrchikov/Computational_mathematics/tree/master/laba4)

function polynomial(m) {  
 m += 1  
 let A = create2DArray(m, m)  
 for (let i = 0; i < m; i++) {  
 for (let j = 0; j < m; j++) {  
 let s = 0  
 *XY*.forEach(x => s += x[0] \*\* (i + j))  
 A[i][j] = s  
 }  
 }  
 A[0][0] = *n* let B = *Array*(m)  
 for (let i = 0; i < m; i++) {  
 let s = 0  
 *XY*.forEach(x => s += x[0] \*\* i \* x[1])  
 B[i] = s  
 }  
 let coeffs = solveLinearSystem(A, B);  
 let w = getting\_rid\_of\_scientific\_notation(coeffs)  
  
 let latex\_fun = []  
 for (let i = 0; i < m; i++) {  
 latex\_fun.push(w[i] + "\*x^" + i)  
 }  
 latex\_fun = latex\_fun.join("+")  
 *calculator*.setExpression({id: 'graph' + m, latex: `f\_{${m - 1}}=` + latex\_fun})  
 let S = 0  
 latex\_fun = latex\_fun.replaceAll("^", "\*\*").replaceAll("{", "(").replaceAll("}", ")")  
 *XY*.forEach(x => S += (eval(latex\_fun.replaceAll("x", `(${x[0]})`)) - x[1]) \*\* 2)  
 *document*.getElementById("res").innerHTML += `S\_${m - 1} = ${*Math*.round(S \* 10000) / 10000}<br>`  
 draw\_table(m - 1, latex\_fun)  
 return [coeffs, S]  
}  
  
function exponential() {  
 let sx = 0, sxx = 0, sy = 0, sxy = 0  
 *XY*.forEach(x => {  
 sx += x[0]  
 sxx += x[0] \*\* 2  
 sy += *Math*.log(x[1])  
 sxy += x[0] \* *Math*.log(x[1])  
 })  
 let A = [  
 [*n*, sx],  
 [sx, sxx]  
 ]  
 let B = [sy, sxy]  
 let coeffs = solveLinearSystem(A, B)  
 coeffs[0] = *Math*.exp(coeffs[0])  
 let w = getting\_rid\_of\_scientific\_notation(coeffs)  
 *calculator*.setExpression({id: 'graph6', latex: `f\_e=${w[0]}\*e^{${w[1]}x}`})  
 let latex\_fun = `${w[0]}\*Math.E \*\* (${w[1]}\*x)`.replaceAll("{", "(").replaceAll("}", ")")  
 let S = 0  
 *XY*.forEach(x => S += (eval(latex\_fun.replaceAll("x", `(${x[0]})`)) - x[1]) \*\* 2)  
 if (isNaN(S))  
 *document*.getElementById("res").innerHTML += `S\_e - невозможно аппроксимировать с помощью экспоненциальной функции<br>`  
 else  
 *document*.getElementById("res").innerHTML += `S\_e = ${*Math*.round(S \* 10000) / 10000}<br>`  
 draw\_table("e", latex\_fun)  
 return [coeffs, S]  
}  
  
function logarithmic() {  
 let sx = 0, sxx = 0, sy = 0, sxy = 0  
 *XY*.forEach(x => {  
 sx += *Math*.log(x[0])  
 sxx += *Math*.log(x[0]) \*\* 2  
 sy += x[1]  
 sxy += *Math*.log(x[0]) \* x[1]  
 })  
 let A = [  
 [*n*, sx],  
 [sx, sxx]  
 ]  
 let B = [sy, sxy]  
 let coeffs = solveLinearSystem(A, B)  
 let w = getting\_rid\_of\_scientific\_notation(coeffs)  
 *calculator*.setExpression({id: 'graph7', latex: `f\_{log}=${w[1]}\*\\ln{x}+${w[0]}`})  
 let latex\_fun = `${w[1]}\*Math.log(x) + ${w[0]}`.replaceAll("{", "(").replaceAll("}", ")")  
 let S = 0  
 *XY*.forEach(x => S += (eval(latex\_fun.replaceAll("x", `(${x[0]})`)) - x[1]) \*\* 2)  
 if (isNaN(S))  
 *document*.getElementById("res").innerHTML += `S\_log - невозможно аппроксимировать с помощью логарифмической функции<br>`  
 else  
 *document*.getElementById("res").innerHTML += `S\_log = ${*Math*.round(S \* 10000) / 10000}<br>`  
 draw\_table("log", latex\_fun)  
 return [coeffs, S]  
}  
  
function power() {  
 let sx = 0, sxx = 0, sy = 0, sxy = 0  
 *XY*.forEach(x => {  
 sx += *Math*.log(x[0])  
 sxx += *Math*.log(x[0]) \*\* 2  
 sy += *Math*.log(x[1])  
 sxy += *Math*.log(x[0]) \* *Math*.log(x[1])  
 })  
 let A = [  
 [*n*, sx],  
 [sx, sxx]  
 ]  
 let B = [sy, sxy]  
 let coeffs = solveLinearSystem(A, B)  
 coeffs[0] = *Math*.exp(coeffs[0])  
 let w = getting\_rid\_of\_scientific\_notation(coeffs)  
 *calculator*.setExpression({id: 'graph8', latex: `f\_{power}=${w[0]}\*x^{${w[1]}}`})  
 let latex\_fun = `${w[0]}\*x\*\*(${w[1]})`.replaceAll("{", "(").replaceAll("}", ")")  
 let S = 0  
 *XY*.forEach(x => S += (eval(latex\_fun.replaceAll("x", `(${x[0]})`)) - x[1]) \*\* 2)  
 if (isNaN(S))  
 *document*.getElementById("res").innerHTML += `S\_power - невозможно аппроксимировать с помощью степенной функции<br>`  
 else  
 *document*.getElementById("res").innerHTML += `S\_power = ${*Math*.round(S \* 10000) / 10000}<br>`  
 draw\_table("power", latex\_fun)  
 return [coeffs, S]  
}  
  
  
// Решение системы линейных уравнений Ax = B методом Гаусса  
function solveLinearSystem(A, B) {  
 let n = A.length;  
  
 for (let i = 0; i < n; i++) {  
 // Поиск главного элемента  
 let maxEl = *Math*.abs(A[i][i]);  
 let maxRow = i  
 for (let k = i + 1; k < n; k++)  
 if (*Math*.abs(A[k][i]) > maxEl) {  
 maxEl = *Math*.abs(A[k][i])  
 maxRow = k  
 }  
  
 // Меняем местами строки  
 for (let k = i; k < n; k++) {  
 let tmp = A[maxRow][k]  
 A[maxRow][k] = A[i][k]  
 A[i][k] = tmp  
 }  
 let tmp = B[maxRow]  
 B[maxRow] = B[i]  
 B[i] = tmp  
  
 // Приводим к нижнетреугольной форме  
 for (let k = i + 1; k < n; k++) {  
 let c = -A[k][i] / A[i][i];  
 for (let j = i; j < n; j++) {  
 if (i === j)  
 A[k][j] = 0  
 else  
 A[k][j] += c \* A[i][j]  
 }  
 B[k] += c \* B[i]  
 }  
 }  
  
 // Обратная подстановка  
 let x = new *Array*(n).fill(0);  
 for (let i = n - 1; i >= 0; i--) {  
 x[i] = B[i] / A[i][i]  
 for (let k = i - 1; k >= 0; k--) {  
 B[k] -= A[k][i] \* x[i]  
 }  
 }  
  
 return x;  
}  
  
  
function create2DArray(n, m) {  
 let array = new *Array*(n);  
  
 for (let i = 0; i < n; i++)  
 array[i] = new *Array*(m).fill(0);  
  
 return array;  
}

# Примеры и результаты работы программы

## Пример 1

Ввод:  
1 2.7

2 7.38

3 20.08

4 54.59

5 148.41

6 403.29

-3 0.049

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описаниеИзображение выглядит как текст, диаграмма, снимок экрана, Шрифт

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

## Пример 2

Ввод

1 0.1

2 0.69

3 1.09

4 1.38

5 1.6

6 1.79

7 1.94

8 2.08

0.5 -0.69

0.3 -1.2

0.002 -6.214

0.0000002 -15.42

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описаниеИзображение выглядит как текст, диаграмма, черно-белый, Шрифт

Автоматически созданное описаниеИзображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

## Пример 3

Ввод

1 1

2 0.5

3 0.3333

4 0.25

5 0.2

6 0.16667

7 0.142

8 0.125

9 0.111111

10 0.1Изображение выглядит как текст, диаграмма, линия, График

Автоматически созданное описаниеИзображение выглядит как текст, диаграмма, снимок экрана, Шрифт

Автоматически созданное описаниеИзображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

# Вывод

Во время выполнения данной лабораторной работы я познакомился с методом наименьших квадратов для аппроксимации табличной функции с помощью функций и реализовал это на языке JavaScript.