МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Поволжский государственный университет телекоммуникаций и информатики»

Факультет информатики и вычислительной техники

Кафедра ПрИ

# ОТЧЁТ

## ПО ЛАБОРАТОРНОЙ РАБОТЕ №7

**Дисциплина: Численные методы**

Выполнил: студент
ПрИ-21 Морзюков
М.А.
Проверил(а):
Осанов В.А.

Самара 2024

**Вариант №11**

**Цель работы:** изучить приближение функций: интерполяционный полином Лагранжа, приближение полиномами Ньютона, интерполирование функций с помощью кубического сплайна и аппроксимация функций методом наименьших квадратов многочленом второй степени.

| Вариант 11 |
| --- |
| 2,15 |
| 2,41 |
| 2,58 |
| 2,84 |
| 3,28 |
| 3,46 |
| 4,02 |
| 4,11 |
| 4,61 |
| 5,03 |
| 5,34 |
| 5,86 |
| 6,33 |
| 6,81 |
| 7,21 |
| 7,67 |
| 8,23 |
| 8,68 |
| 9,35 |
| 9,93 |

**Интерполяционный полином Лагранжа — это** способ интерполирования функции, проходящий через заданные точки. Формула Лагранжа позволяет построить многочлен, который точно совпадает со значениями функции в этих точках.

```java
public class LagrangeInterpolation {
    public static double interpolate(double[] x, double[] y, double q) {
        int n = x.length;
        double result = 0.0;

        for (int i = 0; i < n; i++) {
            double term = y[i];
            for (int j = 0; j < n; j++) {
                if (j != i) {
                    term *= (q - x[j]) / (x[i] - x[j]);
                }
            }
            result += term;
        }

        return result;
    }
}
```

**Интерполирование функций с помощью кубического сплайна — это** метод сглаживания данных, при котором функция аппроксимируется кусочно-кубическими полиномами, обеспечивающими непрерывность самой функции и её первой и второй производных в точках стыковки.

```java
public class CubicSplineInterpolation {

    private double[] x;
    private double[] y;
    private double[] a, b, c, d;

    public CubicSplineInterpolation(double[] x, double[] y) {
        int n = x.length;
        this.x = x;
        this.y = y;
        this.a = new double[n];
        this.b = new double[n - 1];
        this.c = new double[n];
        this.d = new double[n - 1];
        computeSplineCoefficients();
    }

    private void computeSplineCoefficients() {
        int n = x.length - 1;
        double[] h = new double[n];
        double[] alpha = new double[n];

        for (int i = 0; i < n; i++) {
            h[i] = x[i + 1] - x[i];
            if (i > 0) {
                alpha[i] = (3 / h[i]) * (y[i + 1] - y[i]) - (3 / h[i - 1]) * (y[i] - y[i - 1]);
            }
        }

        double[] l = new double[n + 1];
        double[] mu = new double[n];
        double[] z = new double[n + 1];

        l[0] = 1;
        mu[0] = 0;
        z[0] = 0;
```

```
        for (int i = 1; i < n; i++) {
            l[i] = 2 * (x[i + 1] - x[i - 1]) - h[i - 1] * mu[i - 1];
            mu[i] = h[i] / l[i];
            z[i] = (alpha[i] - h[i - 1] * z[i - 1]) / l[i];
        }

        l[n] = 1;
        z[n] = 0;
        c[n] = 0;

        for (int j = n - 1; j >= 0; j--) {
            c[j] = z[j] - mu[j] * c[j + 1];
            b[j] = (y[j + 1] - y[j]) / h[j] - h[j] * (c[j + 1] + 2 * c[j]) / 3;
            d[j] = (c[j + 1] - c[j]) / (3 * h[j]);
            a[j] = y[j];
        }
    }

    public double interpolate(double q) {
        int i = findSegment(q);
        double dx = q - x[i];
        return a[i] + b[i] * dx + c[i] * dx * dx + d[i] * dx * dx * dx;
    }

    private int findSegment(double q) {
        int i = 0, j = x.length - 1;
        while (i < j) {
            int mid = (i + j) / 2;
            if (q < x[mid]) j = mid;
            else i = mid + 1;
        }
        return Math.max(i - 1, 0);
    }
}
```

**Приближение полиномами Ньютона — это** метод численной аппроксимации функций, использующий разложения в ряды Тейлора или Маклорена для нахождения приближенного значения функции в точке.

```
public class NewtonPolynomial {
    private double[] x;
    private double[] y;
    private double[][] dividedDifferences;

    public NewtonPolynomial(double[] x, double[] y) {
        this.x = x;
        this.y = y;
        this.dividedDifferences = computeDividedDifferences();
    }

    private double[][] computeDividedDifferences() {
        int n = x.length;
        double[][] dd = new double[n][n];

        for (int i = 0; i < n; i++) {
            dd[i][0] = y[i];
        }

        for (int j = 1; j < n; j++) {
            for (int i = 0; i < n - j; i++) {
                dd[i][j] = (dd[i + 1][j - 1] - dd[i][j - 1]) / (x[i + j] - x[i]);
            }
        }

        return dd;
    }

    public double interpolate(double q) {
        double result = dividedDifferences[0][0];
        double term = 1;

        for (int i = 1; i < x.length; i++) {
            term *= (q - x[i - 1]);
            result += dividedDifferences[0][i] * term;
        }
```

**Аппроксимация функций методом наименьших квадратов многочленом второй степени — это** статистический подход, используемый для подбора параболической кривой (многочлена второй степени) к набору данных таким образом, чтобы минимизировать сумму квадратов отклонений точек данных от этой кривой. Этот метод помогает найти наилучшее приближенное соответствие между данными и моделью, представляющей зависимость второго порядка.

```java
public class LeastSquaresApproximation {
    private double a;
    private double b;
    private double c;

    public LeastSquaresApproximation(double[] x, double[] y) {
        calculateCoefficients(x, y);
    }

    private void calculateCoefficients(double[] x, double[] y) {
        int n = x.length;

        double sumX = 0, sumY = 0, sumX2 = 0, sumX3 = 0, sumX4 = 0;
        double sumXY = 0, sumX2Y = 0;

        for (int i = 0; i < n; i++) {
            sumX += x[i];
            sumY += y[i];
            sumX2 += x[i] * x[i];
            sumX3 += x[i] * x[i] * x[i];
            sumX4 += x[i] * x[i] * x[i] * x[i];
            sumXY += x[i] * y[i];
            sumX2Y += x[i] * x[i] * y[i];
        }

        double[][] matrix = {
                {n, sumX, sumX2},
                {sumX, sumX2, sumX3},
                {sumX2, sumX3, sumX4}
        };

        double[] constants = {sumY, sumXY, sumX2Y};

        double[] solution = solveSystem(matrix, constants);
        this.c = solution[0];
        this.b = solution[1];
        this.a = solution[2];
```

```java
private double[] solveSystem(double[][] matrix, double[] constants) {
    double d = determinant(matrix);
    double[] solution = new double[3];

    for (int i = 0; i < 3; i++) {
        double[][] tempMatrix = new double[3][3];
        for (int j = 0; j < 3; j++) {
            for (int k = 0; k < 3; k++) {
                tempMatrix[j][k] = (k == i) ? constants[j] : matrix[j][k];
            }
        }
        solution[i] = determinant(tempMatrix) / d;
    }

    return solution;
}

private double determinant(double[][] matrix) {
    return matrix[0][0] * (matrix[1][1] * matrix[2][2] - matrix[1][2] * matrix[2][1])
            - matrix[0][1] * (matrix[1][0] * matrix[2][2] - matrix[1][2] * matrix[2][0])
            + matrix[0][2] * (matrix[1][0] * matrix[2][1] - matrix[1][1] * matrix[2][0]);
}

public double approximate(double q) {
    return a * q * q + b * q + c;
}
```

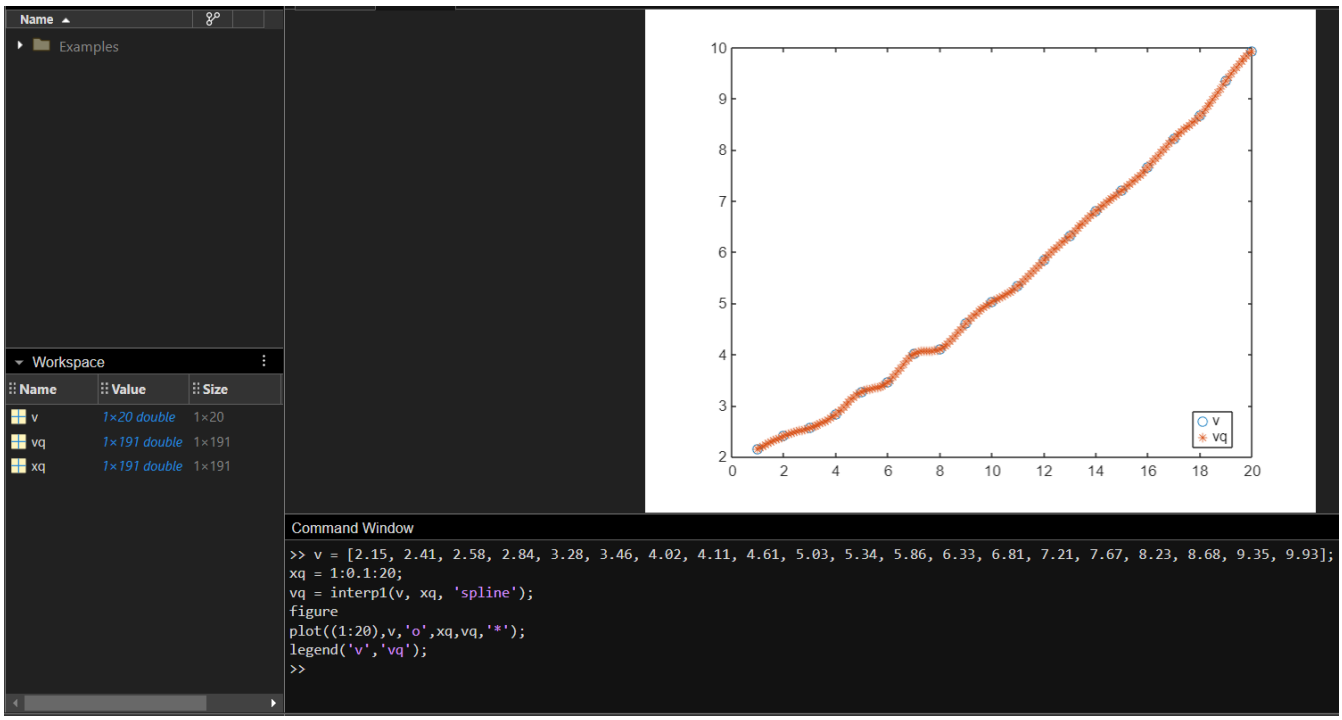
**Результат выполнения программы:**

| Интерполяционный полином Лагранжа: | Интерполирование функций с помощью кубического сплайна: |
|---|---|
| f(1,00) = 2,1500 | f(1,00) = 2,1500 |
| f(1,50) = -54,2175 | f(1,50) = 2,2894 |
| f(2,00) = 2,4100 | f(2,00) = 2,4100 |
| f(2,50) = 8,6265 | f(2,50) = 2,5006 |
| f(3,00) = 2,5800 | f(3,00) = 2,5800 |
| f(3,50) = 1,4244 | f(3,50) = 2,6783 |
| f(4,00) = 2,8400 | f(4,00) = 2,8400 |
| f(4,50) = 3,4891 | f(4,50) = 3,0801 |
| f(5,00) = 3,2800 | f(5,00) = 3,2800 |
| f(5,50) = 3,1704 | f(5,50) = 3,3513 |
| f(6,00) = 3,4600 | f(6,00) = 3,4600 |
| f(6,50) = 3,8370 | f(6,50) = 3,7497 |
| f(7,00) = 4,0200 | f(7,00) = 4,0200 |
| f(7,50) = 4,0454 | f(7,50) = 4,0788 |
| f(8,00) = 4,1100 | f(8,00) = 4,1100 |
| f(8,50) = 4,3186 | f(8,50) = 4,3176 |
| f(9,00) = 4,6100 | f(9,00) = 4,6100 |
| f(9,50) = 4,8633 | f(9,50) = 4,8519 |
| f(10,00) = 5,0300 | f(10,00) = 5,0300 |
| f(10,50) = 5,1623 | f(10,50) = 5,1709 |
| f(11,00) = 5,3400 | f(11,00) = 5,3400 |
| f(11,50) = 5,5868 | f(11,50) = 5,5870 |
| f(12,00) = 5,8600 | f(12,00) = 5,8600 |
| f(12,50) = 6,1086 | f(12,50) = 6,1011 |
| f(13,00) = 6,3300 | f(13,00) = 6,3300 |
| f(13,50) = 6,5598 | f(13,50) = 6,5737 |
| f(14,00) = 6,8100 | f(14,00) = 6,8100 |
| f(14,50) = 7,0388 | f(14,50) = 7,0155 |
| f(15,00) = 7,2100 | f(15,00) = 7,2100 |
| f(15,50) = 7,3814 | f(15,50) = 7,4217 |
| f(16,00) = 7,6700 | f(16,00) = 7,6700 |
| f(16,50) = 8,0459 | f(16,50) = 7,9577 |
| f(17,00) = 8,2300 | f(17,00) = 8,2300 |
| f(17,50) = 8,1459 | f(17,50) = 8,4463 |
| f(18,00) = 8,6800 | f(18,00) = 8,6800 |
| f(18,50) = 10,6443 | f(18,50) = 9,0010 |
| f(19,00) = 9,3500 | f(19,00) = 9,3500 |
| f(19,50) = -6,3330 | f(19,50) = 9,6559 |
| f(20,00) = 9,9300 | f(20,00) = 9,9300 |

```
Приближение полиномами Ньютона:
f(1,00) = 2,1500
f(1,50) = -54,2175
f(2,00) = 2,4100
f(2,50) = 8,6265
f(3,00) = 2,5800
f(3,50) = 1,4244
f(4,00) = 2,8400
f(4,50) = 3,4891
f(5,00) = 3,2800
f(5,50) = 3,1704
f(6,00) = 3,4600
f(6,50) = 3,8370
f(7,00) = 4,0200
f(7,50) = 4,0454
f(8,00) = 4,1100
f(8,50) = 4,3186
f(9,00) = 4,6100
f(9,50) = 4,8633
f(10,00) = 5,0300
f(10,50) = 5,1623
f(11,00) = 5,3400
f(11,50) = 5,5868
f(12,00) = 5,8600
f(12,50) = 6,1086
f(13,00) = 6,3300
f(13,50) = 6,5598
f(14,00) = 6,8100
f(14,50) = 7,0388
f(15,00) = 7,2100
f(15,50) = 7,3814
f(16,00) = 7,6700
f(16,50) = 8,0459
f(17,00) = 8,2300
f(17,50) = 8,1459
f(18,00) = 8,6800
f(18,50) = 10,6443
f(19,00) = 9,3500
f(19,50) = -6,3330
f(20,00) = 9,9300
```

```
Аппроксимация функций методом наименьших квадратов многочленом второй степени:
f(1,00) = 2,1221
f(1,50) = 2,2433
f(2,00) = 2,3690
f(2,50) = 2,4992
f(3,00) = 2,6340
f(3,50) = 2,7732
f(4,00) = 2,9169
f(4,50) = 3,0652
f(5,00) = 3,2179
f(5,50) = 3,3752
f(6,00) = 3,5369
f(6,50) = 3,7032
f(7,00) = 3,8740
f(7,50) = 4,0492
f(8,00) = 4,2290
f(8,50) = 4,4133
f(9,00) = 4,6021
f(9,50) = 4,7954
f(10,00) = 4,9932
f(10,50) = 5,1955
f(11,00) = 5,4023
f(11,50) = 5,6136
f(12,00) = 5,8294
f(12,50) = 6,0497
f(13,00) = 6,2745
f(13,50) = 6,5038
f(14,00) = 6,7377
f(14,50) = 6,9760
f(15,00) = 7,2188
f(15,50) = 7,4662
f(16,00) = 7,7180
f(16,50) = 7,9744
f(17,00) = 8,2352
f(17,50) = 8,5006
f(18,00) = 8,7704
f(18,50) = 9,0448
f(19,00) = 9,3237
f(19,50) = 9,6071
f(20,00) = 9,8949
```

## Проверка:


Name
Examples
Workspace
Name
Value
Size
v
1×20 double
1×20
vq
1×191 double
1×191
xq
1×191 double
1×191
10
9
8
7
6
5
4
3
2
0
2
4
6
8
10
12
14
16
18
20
v
vq
Command Window
>> v = [2.15, 2.41, 2.58, 2.84, 3.28, 3.46, 4.02, 4.11, 4.61, 5.03, 5.34, 5.86, 6.33, 6.81, 7.21, 7.67, 8.23, 8.68, 9.35, 9.93];
xq = 1:0.1:20;
vq = interp1(v, xq, 'spline');
figure
plot((1:20),v,'o',xq,vq,'*');
legend('v','vq');
>>