Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 2

Тема: Многомерная интерполяция на регулярной сетке

Студент: Чаушев Александър

Группа: ИУ7-46Б

Оценка (баллы) _____

Преподаватель : Градов В. М.

Москва.
2020 г.

Цель работы: Изучить применение одномерной интерполяции полиномом Ньютона к многомерной интерполяции.

Входные данные: таблица значений функции двух переменных, аргументы, степень полинома Ньютона для каждой переменной.

Результат работы программы: Значение функции от двух переменных f(x,y).

Алгоритм:
Сначало проверяем что есть две точки в таблице. Потом при помощи цикла находим два узла между которьми находиться точка которую ищем. Если не нашли выводим сообщение что точка находиться вне таблице. Потом проверяем не является ли один из этих двух узлов которые нашли ищемая точка, если да возвращаем его значение.
Чтобы найти нужное значение, нам нужно сначало найти a, b, U, d, t. Мы находим a ,t сразу потому что есть необходимые значения в таблице.
Чтобы найти U нам нужно установить начальные значения U[1] = 0, U[n] = 0, но этого не достаточно, нам нужно еще найти psi и nyu. Чтобы их найти мы установливаем начальные значения :

```
psi[3] = D[2] / B[2];

nyu[3] = F[2] / B[2];
```

По формуле видно что мы еще не нашли B, D, F. Еще чтобы найти все значения psi и nyu, нехватает A.

```
psi[i + 1] = D[i] / (B[i] - A[i] * psi[i]);

nyu[i + 1] = (A[i] * nyu[i] + F[i]) / (B[i] - A[i] * psi[i]);
```

Чтобы найти A, B, D, F нам нужен h его можем найти сразу по формуле

```
h[i] = table.get(i).getKey() - table.get(i - 1).getKey();
```

Отсюда мы находим A,B,D,F по формулам

```
A[i] = h[i - 1];

B[i] = -2 * (h[i - 1] + h[i]);

D[i] = h[i];

F[i] = -3 * ((table.get(i).getValue() - table.get(i - 1).getValue()) / h[i] -
        (table.get(i - 1).getValue() - table.get(i - 2).getValue()) / h[i - 1]);
```

Потом дальше находим psi и nyu (см. выше). Отсюда находим U по формулу:

```
U[i] = psi[i + 1] * U[i + 1] + nyu[i + 1];
```

Осталось найти b и d. Все необходимые значения уже известны поэтому мы их находим по формуле:

```
d[i] = (U[i + 1] - U[i]) / (3 * h[i]);
```

b[i] = ((table.get(i).getValue() - table.get(i-1).getValue()) /

h[i]) - (h[i] * (U[i+1] + 2 * U[i])) / 3;

Возвращем необходимое значение:

a[xi] + b[xi] * t + U[xi] * t * t + d[xi] * t * t * t;

Полный Код:

```
public final class SplineInterpolation {

   public static Double getActualValue(Double x) {
      // return Math.cos(x) - x;
      return x * x;
   }

   public static double calculate(ArrayList<Map.Entry<Double, Double>> table, Double x)
         throws IllegalArgumentException {

      double result;
      int n = table.size();

      /* Check arguments */
      if (n < 2) {
         throw new IllegalArgumentException("Table size is less than 2!");
      }

      /* Find x0 */
      int xi = 0; // First number in table.key less than x
      boolean finded = false;
      for (int i = 0; i < table.size() - 1; i++) {
         if (table.get(i).getKey() <= x && table.get(i + 1).getKey() >= x ||
               table.get(i).getKey() >= x && table.get(i + 1).getKey() <= x ) {
            xi = i;
            finded = true;
            break;
         }
      }

      if (!finded)
         throw new IllegalArgumentException("x is outside the table!");

      // If x is in table
      if (Math.abs(x - table.get(xi).getKey()) <= 1e-5)
         return table.get(xi).getValue();
      if (Math.abs(x - table.get(xi + 1).getKey()) <= 1e-5)
         return table.get(xi + 1).getValue();

      xi++;

      // Algorithm
      double[] a = new double[n]; // OK
      double[] b = new double[n];
      // double[] c = new double[n];
      double[] d = new double[n];
      double[] h = new double[n]; // OK

      for (int i = 1; i < n; i++) {
```

```
            a[i] = table.get(i - 1).getValue();
            h[i] = table.get(i).getKey() - table.get(i - 1).getKey();
        }

        // c[1] = 0;

        double[] A = new double[n]; // OK
        double[] B = new double[n]; // OK
        double[] D = new double[n]; // OK
        double[] F = new double[n]; // OK
        for (int i = 2; i < n; i++) {
            A[i] = h[i - 1];
            B[i] = -2 * (h[i - 1] + h[i]);
            D[i] = h[i];
            F[i] = -3 * ((table.get(i).getValue() - table.get(i - 1).getValue()) / h[i] -
                    (table.get(i - 1).getValue() - table.get(i - 2).getValue()) / h[i - 1]);
        }

        double[] psi = new double[n + 1];
        double[] nyu = new double[n + 1];
        psi[3] = D[2] / B[2];
        nyu[3] = F[2] / B[2];
        // for (int i = 3; i <= n; i++) {
        for (int i = 3; i < n; i++) {
            psi[i + 1] = D[i] / (B[i] - A[i] * psi[i]);
            nyu[i + 1] = (A[i] * nyu[i] + F[i]) / (B[i] - A[i] * psi[i]);
        }

        double[] U = new double[n + 1];

        U[1] = 0;
        U[n] = 0;
        for (int i = n - 1; i >= 2; i--) {
            U[i] = psi[i + 1] * U[i + 1] + nyu[i + 1];
        }

        for(int i = 1; i < n; i++) {
            d[i] = (U[i + 1] - U[i]) / (3 * h[i]);
            b[i] = ((table.get(i).getValue() - table.get(i-1).getValue()) /
                    h[i]) - (h[i] * (U[i+1] + 2 * U[i])) / 3;
        }


        final double t = x - table.get(xi - 1).getKey();

        return a[xi] +
                b[xi] * t +
                U[xi] * t * t +
                d[xi] * t * t * t;
    }
}
```