Университет ИТМО

Факультет программной инженерии и компьютерных технологий

Лабораторная работа №4 по Вычислительной Математике

Выполнил: Богатов Александр Сергеевич

Группа: P3233

Вариант: метод кубических сплайнов

Преподаватель: Перл Ольга Вячеславовна

Санкт-Петербург

2022

Описание математического метода:

В методе интерполяции кубическими сплайнами между любыми двумя соседними узлами функция интерполируется кубическим полиномом – сплайном – коэффициенты которого на каждом интервале определяются из условий сопряжения в узлах:

$$f_i = y_i$$

$$f'(x_i - 0) = f'(x_i + 0)$$

$$f''(x_i - 0) = f''(x_i + 0)$$

, где i = 1,2,…,n-1.

Также ставятся условия, что вторая производная в точках x0 и xn равняется нулю.

Полином ищется в виде $F(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3$

Для нахождения коэффициентов решается система линейных уравнений $\begin{cases} c0 = 0 \\ C_{i-1}h_i + 2(h_i + h_{i+1})C_i + C_{i+1}h_{i+1} = 6\left(\frac{y_{i+1}-y_i}{h_i} - \frac{y_i-y_{i-1}}{h_{i-1}}\right) \\ cn = 0 \end{cases}$, которую можно представить в виде трехдиагональной матрицы:

$$A = \begin{pmatrix} C_1 & B_1 & 0 & 0 & \cdots & 0 & 0 \\ A_2 & C_2 & B_2 & 0 & \cdots & 0 & 0 \\ 0 & A_3 & C_3 & B_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & B_{n-1} & \\ 0 & 0 & 0 & 0 & \cdots & A_n & C_n \end{pmatrix},$$

, элементы которой находятся по формулам: $h_i = x_{i+1} - x_i, A_i = h_i, B_i = h_{i+1}, c_i = 2(h_i + h_{i+1})$. Данные формулы выводятся из условий сопряжения сплайна.

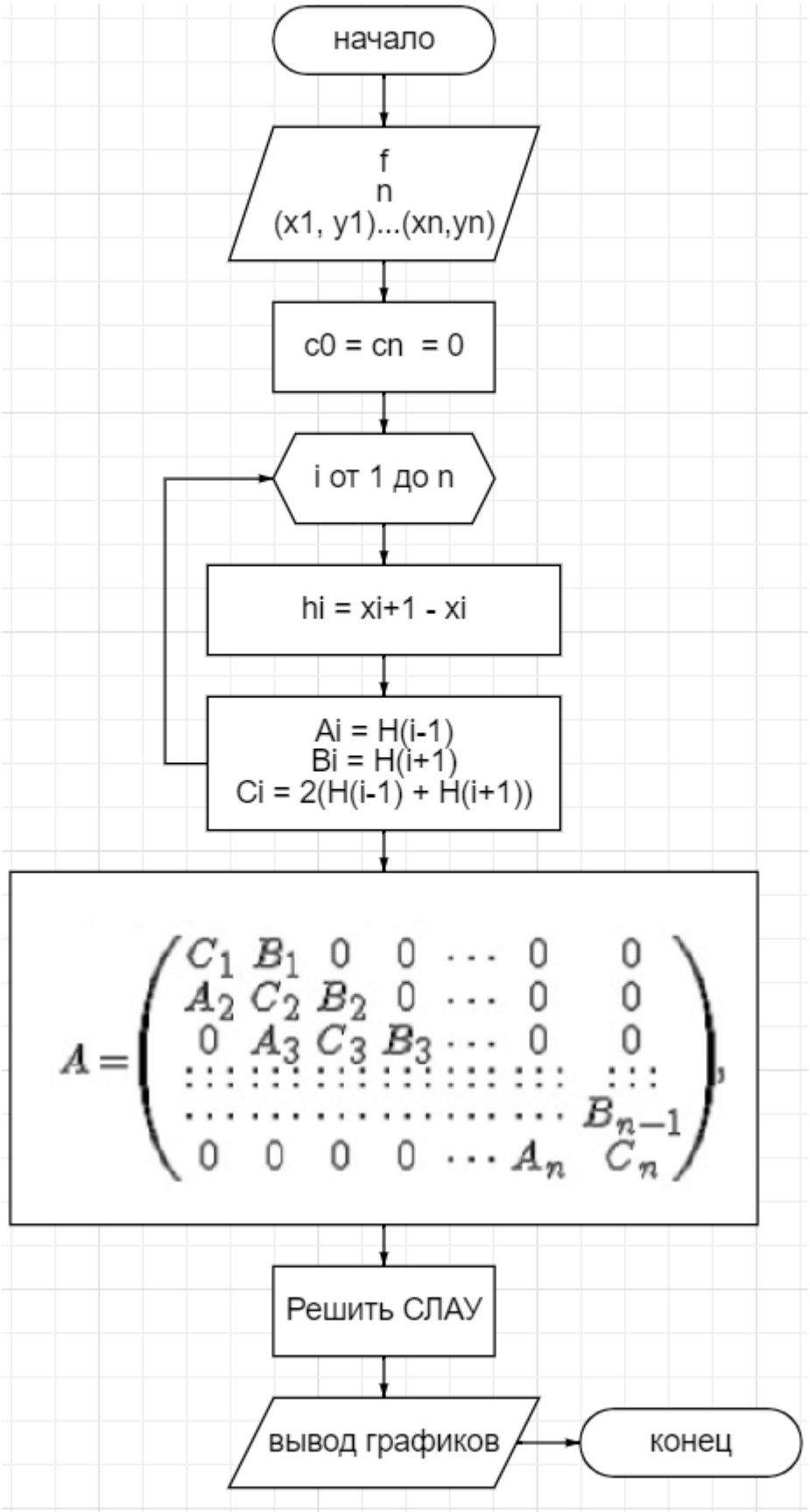
После решения СЛАУ ищем оставшиеся коэффициенты полиномов:

$$a_i = f_i$$

$$b_i = \frac{f_i - f_{i-1}}{h_i} + c_i\frac{h_i}{3} + c_{i-1}\frac{h_j}{6}$$

$$d_i = \frac{c_i - c_{i-1}}{h_i}$$

Блок-схема математического метода:

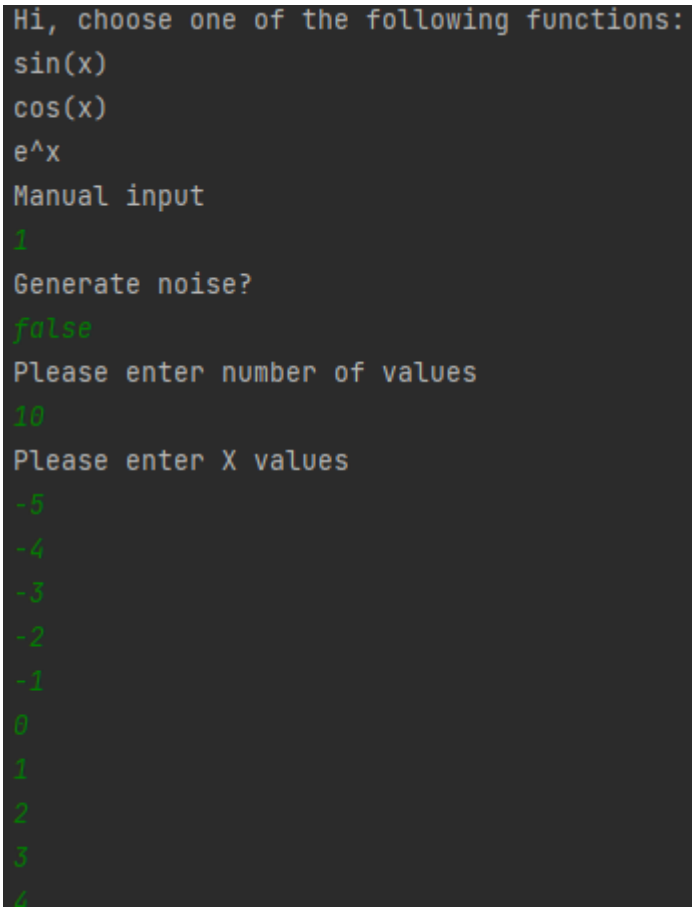


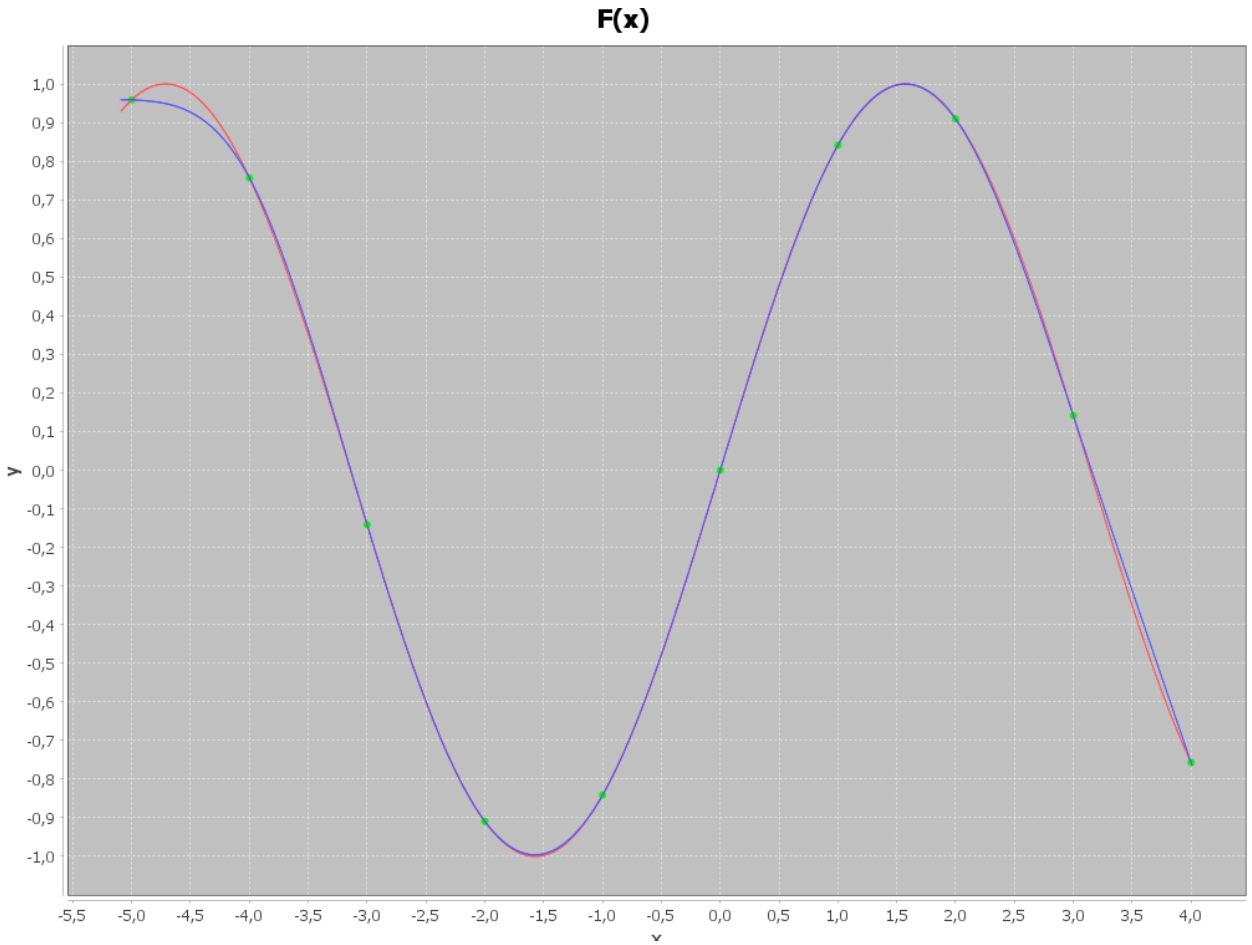
Листинг программы:

```
public Spline makeSpline() {
        double[] hs = getHArray(x);
        double[][] cs = getSplineMatrix(hs, y);
        Matrix matrix = new Matrix(y.length, cs);
        SystemSolver systemSolver = new SystemSolver();
        Matrix solvedMatrix =
systemSolver.findTriangularMatrix(matrix);
        double[] result = new double[n];
        for (int i = 0; i < n; i++) {
            result[n - 1 - i] =
calculateResult(solvedMatrix.getElements(), n, result, n - 1 - i);
        }
        double[] bs = getBArray(hs, result);
        double[] ds = getDArray(hs, result);
        return new Spline(this.y, bs, result, ds, this.x);
    }

    private double calculateResult(double[][] elements, int size,
double[] result, int i) {
        double r = elements[i][size];
        for (int j = 0; j < size; j++)
            if (j != i)
                r -= elements[i][j]*result[j];
        r /= elements[i][i];
        return r;
    }

    private double[] getHArray(double[] x) {
        double[] result = new double[x.length - 1];
        for (int i = 0; i < result.length; i++) {
            result[i] = x[i + 1] - x[i];
        }
        return result;
    }

    private double[] getBArray(double[] hs, double[] cs) {
        double[] bs = new double[y.length];
        bs[0] = 0;
        for (int i = 1; i < bs.length; i++)
            bs[i] = (y[i] - y[i - 1]) / hs[i - 1] + hs[i - 1] * (cs[i]
* 2 / 3 + cs[i - 1] * 2 / 6);
        return bs;
    }


    private double[][] getSplineMatrix(double[] hs, double[] y) {
        double[][] result = new double[y.length][y.length + 1];

        for (int i = 0; i < y.length; i++) {
            for (int j = 0; j < y.length; j++)
                result[i][j] = 0;
        }
        for (int i = 1; i < y.length - 1; i++) {
            result[i][i - 1] = hs[i - 1];
            result[i][i] = 2 * (hs[i] + hs[i - 1]);
            result[i][i + 1] = hs[i];
            result[i][hs.length + 1] = 3 * ((y[i + 1] - y[i]) / hs[i]
- (y[i] - y[i - 1]) / hs[i - 1]);
```

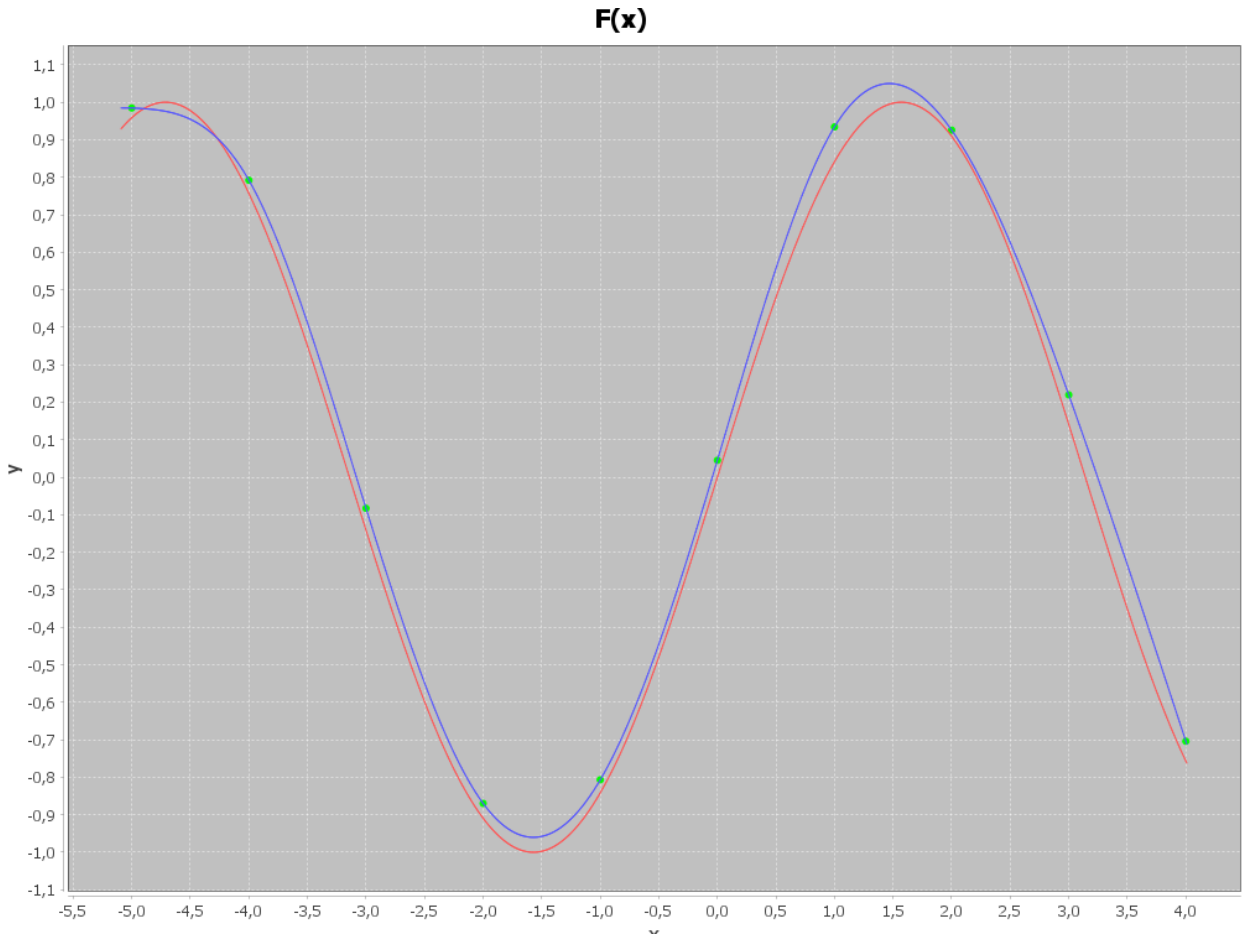
```
        }
        result[0][0] = 1;
        result[hs.length][hs.length] = 1;
        result[0][hs.length + 1] = 0;
        result[hs.length][hs.length + 1] = 0;
        return result;
    }

    private double[] getDArray(double[] hs, double[] cs) {
        double[] ds = new double[y.length];
        ds[0] = 0;
        for (int i = 1; i < ds.length; i++)
            ds[i] = (cs[i] - cs[i - 1]) / (3 * hs[i - 1]);
        return ds;
    }
```

Примеры работы программы:






```
Hi, choose one of the following functions:
sin(x)
cos(x)
e^x
Manual input
1
Generate noise?
true
Please enter number of values
10
Please enter X values
-5
-4
-3
-2
-1
0
1
2
3
4
```



Вывод:

Преимущество метода кубических сплайнов в локальной интерполяции – в отличие от методов Ньютона и Лагранжа, где вычисляется единственный многочлен для всего отрезка интерполирования, в методе сплайнов для каждого промежутка свой полином. Из-за этого результаты применения метода сплайнов более устойчивы к мелким отклонениям значений функций (шуму). Также, в методе сплайнов степень многочлена не зависит от количества узлов интерполяции и всегда равна трем.