Университет ИТМО

Факультет ПИиКТ

Вычислительная математика

Лабораторная работа №3

Интерполирование кубическими сплайнами

Выполнила: Наумова Надежда
Группа P3201
Преподаватель: Перл Ольга Вячеславовна

Санкт-Петербург
2020

Описание метода

Некоторая функция $f(x)$ задана на отрезке $[a, b]$, разбитом на части $[x_{i1}, x_i]$, таком, что $a = x_0 < x_1 < ... < x_n = b$. Кубическим сплайном называется функция $S(x)$, которая
– на каждом отрезке $[x_{i1}, x_i]$ является многочленом степени не выше третьей;
– имеет непрерывные первую и вторую производные на всём отрезке $[a, b]$;
– в точках x_i выполняется равенство $S(x_i) = f(x_i)$, т. е. сплайн $S(x)$ интерполирует функцию f в точках x_i.
Интерполяция кубическими сплайнами - частный случай кусочно-полиноминальной интерполяции. В данном случае между любыми двумя соседними узлами функция интерполируется кубическим полиномом. Его коэффициенты на каждом интервале определяются из условий сопряжения в узлах:

$$f_i = y_i, f^{'}(x_i - 0) = f^{'}(x_i + 0), f^{''}(x_i - 0) = f^{''}(x_i + 0), i = 1, 2, ..., n - 1.$$

Кроме того, на границе при $x = x_0$ и $x = x_n$ ставятся условия:

$$f''(x_0) = 0, f''(x_n) = 0. \tag{2}$$

Будем искать кубический полином в виде

$$f(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3, x_{i-1} \leq \xi \leq \xi_i. \tag{3}$$

Из условия $f_i = y_i$ имеем

$$f(x_{i-1}) = a_i = y_{i-1}, f(x_i) = a_i+b_ih_i+c_ih_i^2+d_ih_i^3 = y_i, h_i = x_i-x_{i-1}, i = 1, 2, \cdots, n-1. \tag{4}$$

Вычислим производные:

$$f'(x) = b_i + 2c_i(x - x_{i-1}) + 3d_i(x - x_{i-1})^2, f''(x) = 2c_i + 6d_i(x - x_{i-1}), x_{i-1} \leq \xi \leq \xi_i,$$

и потребуем их непрерывности при $x = x_i$:

$$b_{i+1} = b_i + 2c_ih_i + 3d_ih_i^2, c_{i+1} = c_i + 3d_ih_i, i = 1, 2, \cdots, n - 1. \tag{5}$$

Общее число неизвестных коэффициентов, очевидно, равно $4n$, число уравнений (4) и (5) равно $4n - 2$. Недостающие два уравнения получаем из условия (2) при $x = x_0$ и $x = x_n$:

$$c_1 = 0, c_n + 3d_nh_n = 0.$$

Выражение из (5) $d_i = \frac{c_{i+1}-c_i}{3h_i}$, подставляя это выражение в (4) и исключая $a_i = y_{i-1}$, получим

$$b_i = \frac{y_i - y_{i-1}}{h_i} - \frac{1}{3}h_i(c_{i+1} + 2c_i), i = 1, 2, \cdots, n - 1, b_n = \frac{y_n - y_{n-1}}{h_n} - \frac{2}{3}h_nc_n.$$

Подставив теперь выражения для b_i, b_{i+1} и d_i в первую формулу (5), после несложных преобразований получаем для определения c_i разностное уравнение второго порядка

$$h_ic_i + 2(h_i + h_{i+1})c_{i+1} + h_{i+1}c_{i+2} = 3\left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}\right), i = 1, 2, \cdots, n-1. \quad (6)$$

С краевыми условиями

$$c_1 = 0, c_{n+1} = 0 \quad (7)$$

Условие $c_{n+1} = 0$ эквивалентно условию $c_n + 3d_nh_n = 0$ и уравнению $c_{i+1} = c_i + d_ih_i$. Разностное уравнение (6) с условиями (7) можно решить методом прогонки, представив в виде системы линейных алгебраических уравнений вида $A * x = F$, где вектор x соответствует вектору $\{c_i\}$, вектор F поэлементно равен правой части уравнения (6), а матрица A имеет следующий вид:

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

где $A_i = h_i, i = 2, \cdots, n, B_i = h_{i+1}, i = 1, \cdots, n-1$
и $C_i = 2(h_i + h_{i+1}), i = 1, \cdots, n$.

Метод прогонки

Метод прогонки, основан на предположении, что искомые неизвестные связаны рекуррентным соотношением:

$$x_i = \alpha_{i+1}x_{i+1} + \beta_{i+1}\, i = 1, \cdots, n-1 \quad (8)$$

Используя это соотношение, выразим x_{i-1} и x_i через x_{i+1} и подставим в i-е уравнение:

$$(A_i\alpha_i\alpha_{i+1} + C_i\alpha_{i+1} + B_i)\, x_{i+1} + A_i\alpha_i\beta_{i+1} + A_i\beta_i + C_i\beta_{i+1} - F_i = 0$$

, где F_i - правая часть i-го уравнения. Это соотношение будет выполняться независимо от решения, если потребовать

$$A_i\alpha_i\alpha_{i+1} + C_i\alpha_{i+1} + B_i = 0$$

$$A_i\alpha_i\beta_{i+1} + A_i\beta_i + C_i\beta_{i+1} - F_i = 0$$

Отсюда следует:

$$\alpha_{i+1} = \frac{-B_i}{A_i\alpha_i + C_i}\ \beta_{i+1} = \frac{F_i - A_i\beta_i}{A_i\alpha_i + C_i}$$

Из первого уравнения получим:

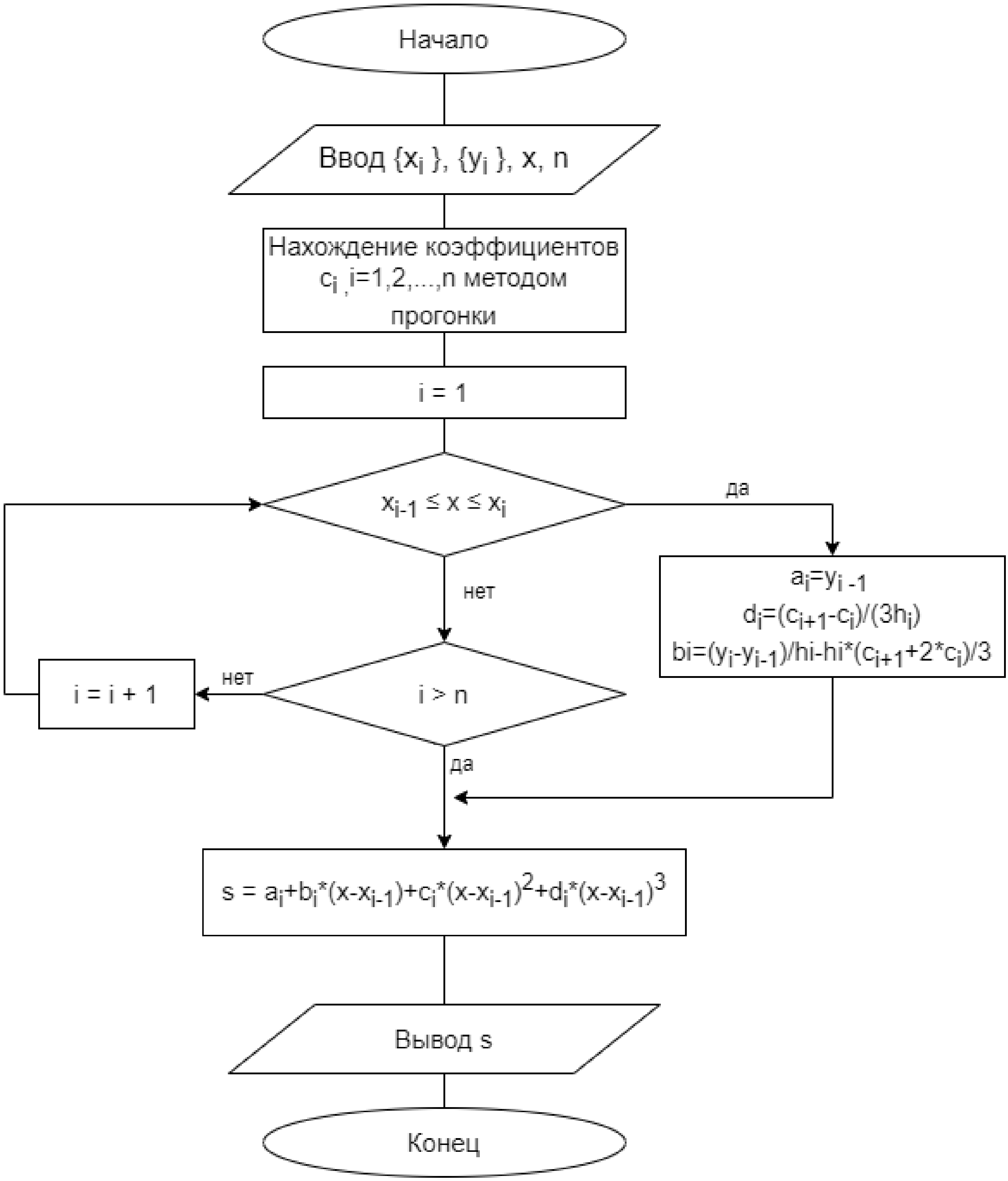
$$\alpha_2 = \frac{-B_1}{C_1}\beta_2 = \frac{F_1}{C_1}$$

После нахождения прогоночных коэффициентов α и β, используя уравнение (1), получим решение системы. При этом,

$$x_n = \frac{F_n - A_n \beta_n}{C_n + A_n \alpha_n}$$

.

Блок-схема



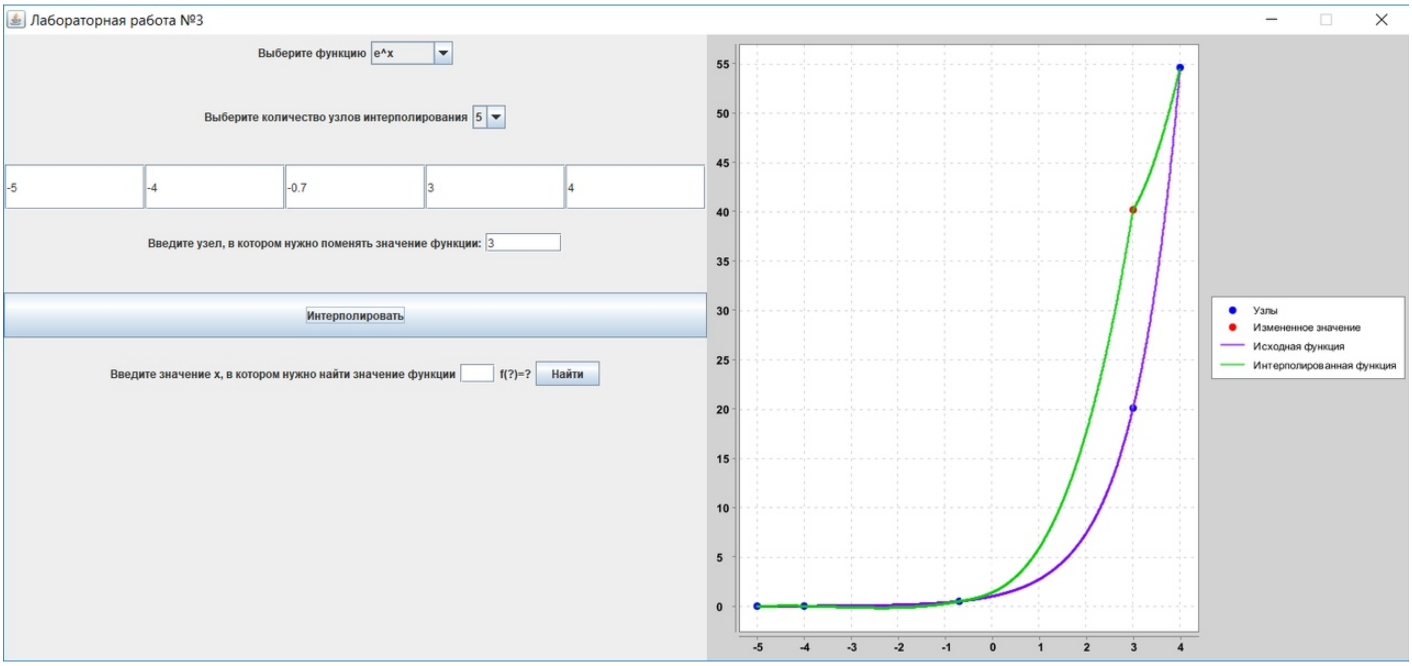
Начало
Ввод {xi }, {yi }, x, n
Нахождение коэффициентов
ci ,i=1,2,...,n методом
прогонки
i = 1
xi-1 ≤ x ≤ xi
да
ai=yi -1
di=(ci+1-ci)/(3hi)
bi=(yi-yi-1)/hi-hi*(ci+1+2*ci)/3
нет
i > n
нет
i = i + 1
да
s = ai+bi*(x-xi-1)+ci*(x-xi-1)2+di*(x-xi-1)3
Вывод s
Конец

Листинг численного метода

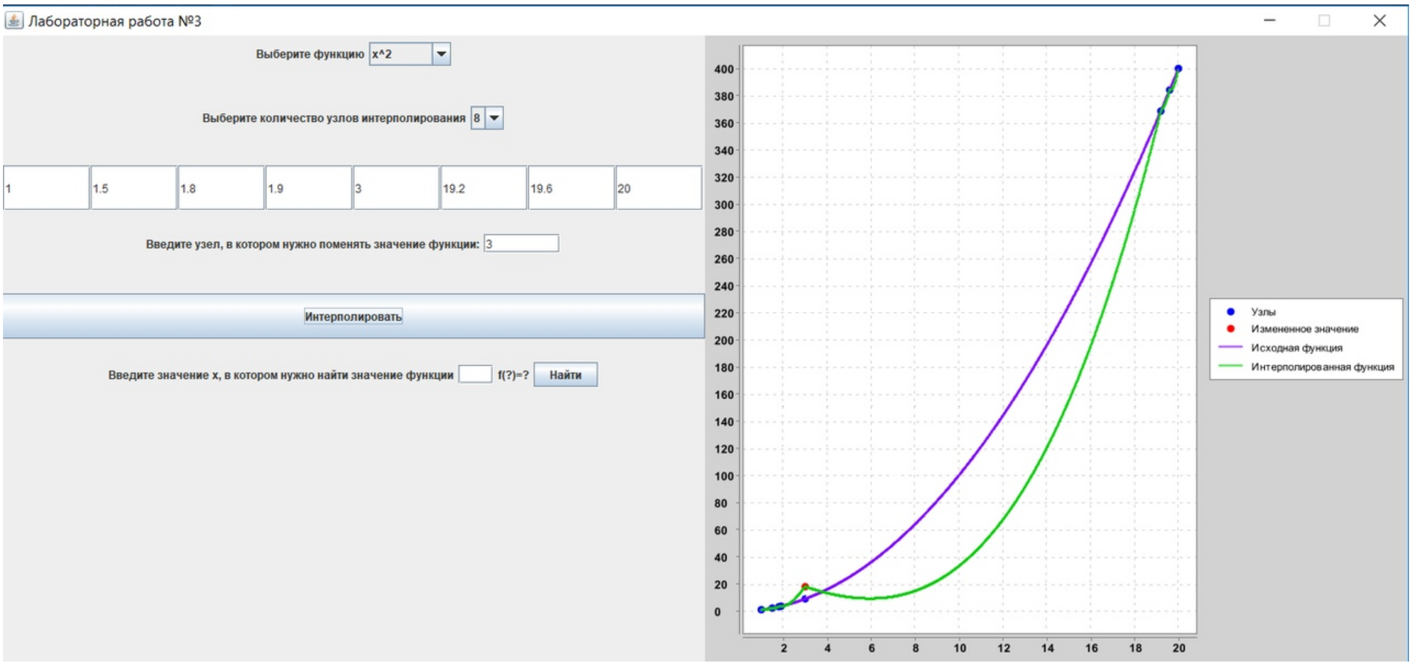
```
public void initSplines(double[] x, double[] y) {
     int n = x.length;
    splines = new Spline[n];
    for (int i = 0; i < n; i ++) {
        splines[i] = new Spline();
        splines[i].setX(x[i]);
        splines[i].setA(y[i]);
    }
    splines[0].setC(0d);
    solveByTridiagonalMatrixAlgorithm(x, y, n);
}

private void solveByTridiagonalMatrixAlgorithm(double[] x, double[] y,
int n) {
    double[] alpha = new double[n - 1];
    double[] beta = new double[n - 1];
    alpha[0] = beta[0] = 0;

    double hi, hi_inc, A = 0, B, C = 0, F = 0, t;

    for (int i = 1; i < n - 1; i ++) {
        hi = x[i] - x[i - 1];
        A = hi;
        hi_inc = x[i + 1] - x[i];
        B = hi_inc;
        C = (hi_inc + hi) * 2;
        F = 6 * ((y[i + 1] - y[i])/hi_inc + (y[i] - y[i - 1])/hi);
        t = (A * alpha[i - 1] + C);

        alpha[i] = -B / t;
        beta[i] = (F - A * beta[i - 1]) / t;
    }
    splines[n - 1].setC((F - A * beta[n - 2])/(C + A * alpha[n - 2]));

    for (int i = n - 2; i > 0; i --)
        splines[i].setC(alpha[i] * splines[i + 1].getC() + beta[i]);

    for (int i = n - 1; i > 0; i --) {
        hi = x[i] - x[i -1];
        splines[i].setD((splines[i].getC() - splines[i - 1].getC())
        / hi);
        splines[i].setB((hi * (2 * splines[i].getC() +
        splines[i - 1].getC()) / 6) +
```

```
            (y[i] - y[i - 1]) / hi);
        }
    }

    public Function interpolate () {
        return this::getInterpolatedY;
    }

private double getInterpolatedY(double x) {
        Spline spline;
        if (x >= splines[splines.length - 1].getX())
            spline = splines[splines.length - 1];
        else if (x <= splines[0].getX())
            spline = splines[0];
        else {
            int k, left = 0, right = splines.length - 1;
            while (right > left + 1) {
                k = left + (right - left) / 2;
                if (x <= splines[k].getX())
                    right = k;
                else
                    left = k;
            }
            spline = splines[right];
        }
        double dx = x - spline.getX();
        double value = spline.getA() + spline.getB() * dx
                + spline.getC() * dx * dx/2 +
                spline.getD()* dx * dx * dx/6;
        return value;
    }
```

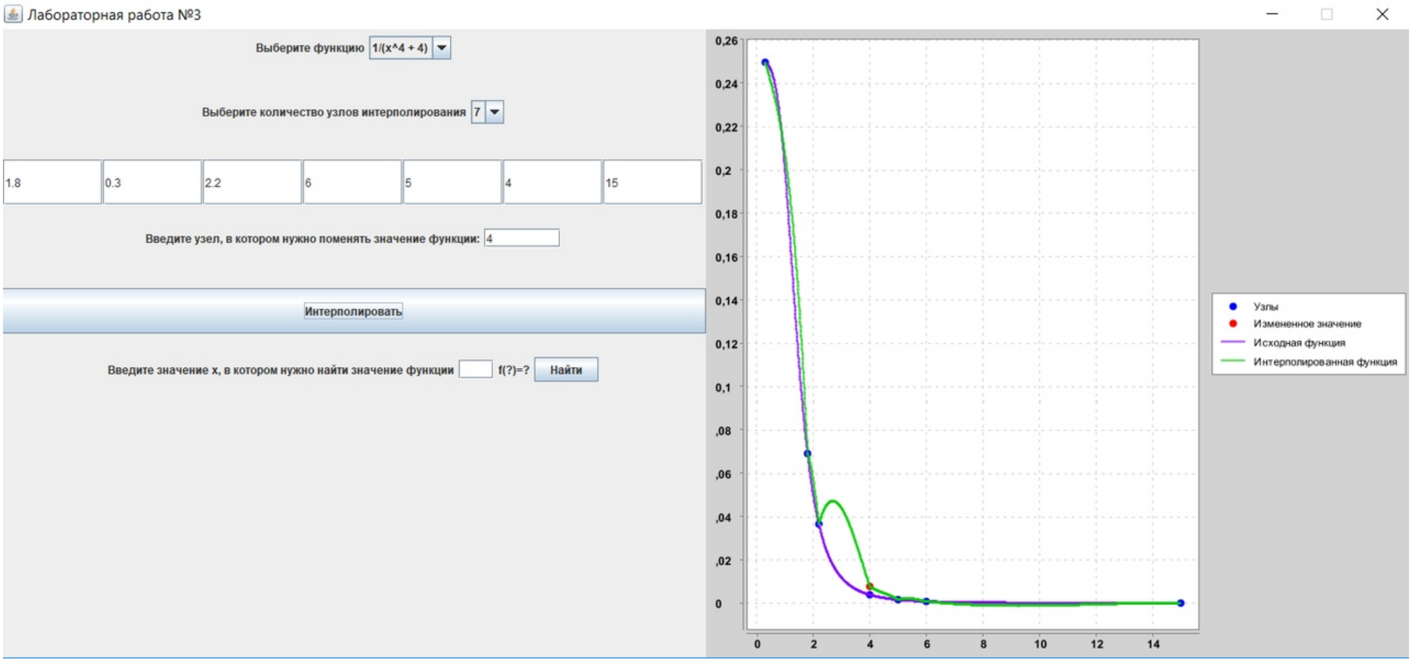
Примеры



Лабораторная работа №3
Выберите функцию
e^x
Выберите количество узлов интерполирования
5
-5
-4
-0.7
3
4
Введите узел, в котором нужно поменять значение функции:
3
Интерполировать
Введите значение x, в котором нужно найти значение функции
f(?)=?
Найти
Узлы
Измененное значение
Исходная функция
Интерполированная функция



Лабораторная работа №3
Выберите функцию
x^2
Выберите количество узлов интерполирования
8
1
1.5
1.8
1.9
3
19.2
19.6
20
Введите узел, в котором нужно поменять значение функции:
3
Интерполировать
Введите значение x, в котором нужно найти значение функции
f(?)=?
Найти
Узлы
Измененное значение
Исходная функция
Интерполированная функция


Лабораторная работа №3
Выберите функцию 1/(x^4 + 4)
Выберите количество узлов интерполирования 7
1.8
0.3
2.2
6
5
4
15
Введите узел, в котором нужно поменять значение функции: 4
Интерполировать
Введите значение x, в котором нужно найти значение функции
f(?)=?
Найти
Узлы
Измененное значение
Исходная функция
Интерполированная функция

Вывод: интерполирование кубическими сплайнами - один из способов кусочно-полиноминальной интерполяции, когда весь отрезок разбивают на частичные отрезки и на каждом из частичных отрезков приближенно заменяют исходную функцию многочленом невысокой (в данном случае, третьей степени), в отличие от формул Ньютона и Лагранжа, где отрезок не рабивается. Интерполяцию кубическими сплайнами рационально применять, если f(x) - периодическая или тригонометрическая функция. Что касается других методов интерполяции, а именно формул Ньютона и Лагранжа, то формулу Лагранжа можно применять для таблиц с различными расстояниями между узлами, а формулы Ньютона – только для таблиц с равноотстоящими узлами. Формулы Ньютона имеют следующее преимущество перед формулой Лагранжа: добавление в таблицу узлов интерполяции при использовании формулы Лагранжа ведет к необходимости пересчета каждого коэффициента заново, тогда как при использовании формулы Ньютона достаточно добавить к уже существующему многочлену только одно слагаемое. Кроме того, по сравнению с этими методами большую точность интерполяции можно получить применением методов сплайн–интерполяции. Что касается сравнения с методом аппроксимации, то следует обратить внимание на разницу в постановке задач аппроксимации и интерполяции: интерполянт должен принадлежать к определенному классу и в точках $x_i (i = 0, 1, ..., n)$ принимать те же значения, что и исходная функция, для аппроксиманта это требование обязательным не является, но должен выполняться критерий наилучшего приближения. В методе наименьших квадратов поле выбора класса аппроксимирующей функции $f(x_i, A, B, C, ...)$ строится сумма вида $Q = \sum\limits_{i=1}^{n} [f(x_i, A, B, C, ...) - y_i]^2$. Исходными значениями параметров $A, B, C, ...$ полагаются числа, которые обеспечивают минимум суммы Q.