Университет ИТМО
Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа №4
Вариант – метод наименьших квадратов

Выполнила: Екатерина Машина

Группа P3210

Преподаватель: Ольга Вячеславовна Перл

Санкт-Петербург
2020 г.

Описание использованного метода

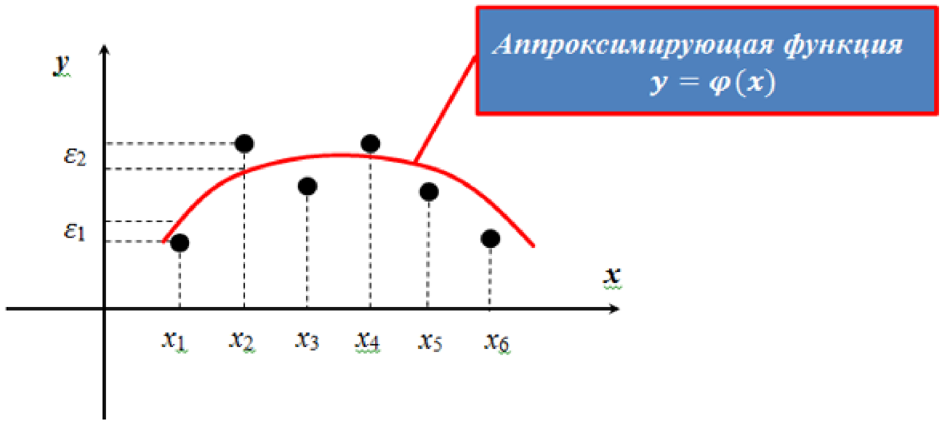
Представим, что вид аппроксимирующей функции уже известен:

$y = \varphi(x,\ a_0,\ a_1, a_2, \dots,\ a_m),$

где φ – известная функция, $a_0,\ a_1, a_2, \dots,\ a_m$ – неизвестные параметры.

Требуется определить такие параметры, при которых значения аппроксимирующей функции приблизительно совпадали со значениями исследуемой функции в точках x_i, т.е. $y_i \approx \varphi(x_i)$. Разность между этими значениями (отклонения) обозначим через ε_i.

Тогда $\varepsilon_i = \varphi(x,\ a_0,\ a_1, a_2, \dots,\ a_m) - y_i,\ i = 1,2,\ \dots n$



Мерой отклонения многочлена $\varphi(x)$ от заданной функции *f(x)* на множестве точек $((x_i,\ y_i)$ является величина S (критерий минимизации), равная сумме квадратов разности между значениями многочлена и функции для всех точек $x_0,\ x_1, \dots,\ x_n$:

$$S = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} [\varphi(x_i) - y_i]^2 \to min$$

Задача нахождения наилучших значений параметров $a_0,\ a_1, \dots,\ a_m$ сводятся к некоторой минимизации отклонений ε_i.

Параметры $a_0,\ a_1, \dots,\ a_m$ эмпирической формулы находятся из условия минимума функции $S = S(a_0,\ a_1, a_2, \dots,\ a_m)$,.

Так как здесь параметры выступают в роли независимых переменных функции S, то её минимум найдем, приравнивая к нулю частные производные по этим переменным (m – степень многочлена, n - число точек) :

$$\frac{\partial S}{\partial a_0} = 2\sum_{i=1}^{n} a_0 + a_1 x_i + \dots + a_{m-1} x_i^{m-1} + a_m x_i^m - y_i = 0$$

$$\frac{\partial S}{\partial a_1} = 2\sum_{i=1}^{n} (a_0 + a_1 x_i + \dots + a_{m-1} x_i^{m-1} + a_m x_i^m - y_i) x_i = 0$$

$$\dots \dots \dots$$

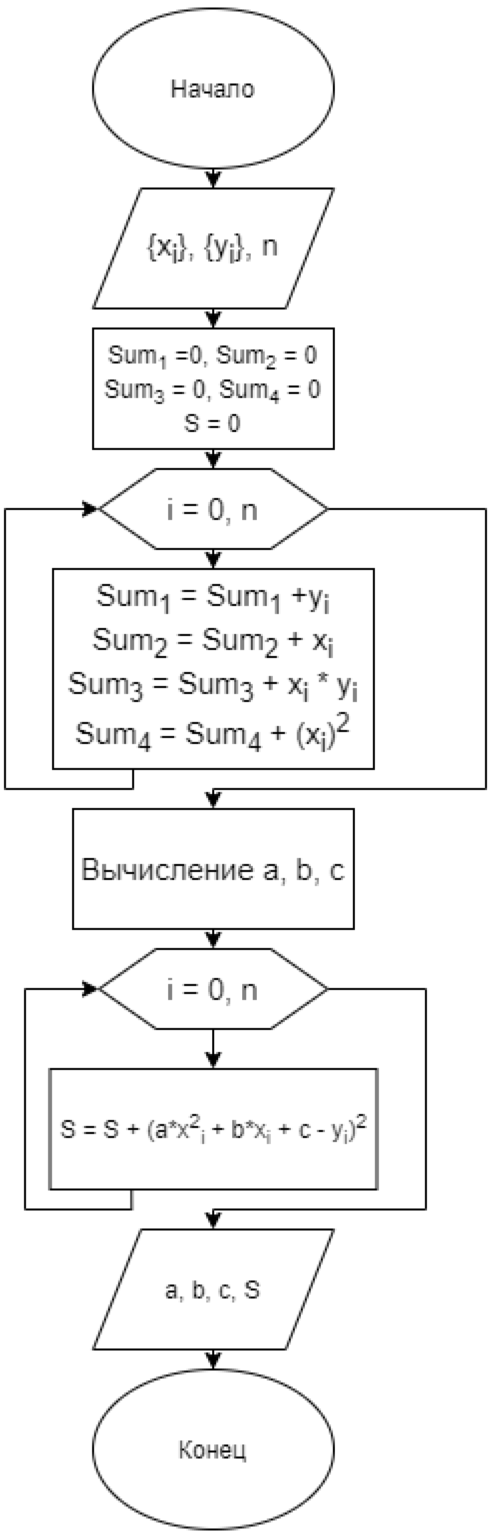
$$\frac{\partial S}{\partial a_m} = 2\sum_{i=1}^{n} (a_0 + a_1 x_i + \dots + a_{m-1} x_i^{m-1} + a_m x_i^m - y_i) x_i^m = 0$$

Преобразуем полученную линейную систему уравнений: раскроем скобки и перенесем свободные слагаемые в правую часть выражения.

в матричном виде:

$$\begin{vmatrix} n & \sum_{i=1}^{n} x_i & \dots & \sum_{i=1}^{n} x_i^m \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \dots & \sum_{i=1}^{n} x_i^{m+1} \\ \dots & \dots & \dots & \\ \sum_{i=1}^{n} x_i^m & \sum_{i=1}^{n} x_i^{m+1} & \dots & \sum_{i=1}^{n} x_i^{2m} \end{vmatrix} \cdot \begin{vmatrix} a_0 \\ a_1 \\ \dots \\ \dots \\ \dots \\ \dots \\ a_m \end{vmatrix} = \begin{vmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \dots \\ \sum_{i=1}^{n} x_i^m y_i \end{vmatrix}$$

Блок-схема



Начало
{xi}, {yi}, n
Sum1 =0, Sum2 = 0
Sum3 = 0, Sum4 = 0
S = 0
i = 0, n
Sum1 = Sum1 +yi
Sum2 = Sum2 + xi
Sum3 = Sum3 + xi * yi
Sum4 = Sum4 + (xi)2
Вычисление a, b, c
i = 0, n
S = S + (a*x2i + b*xi + c - yi)2
a, b, c, S
Конец

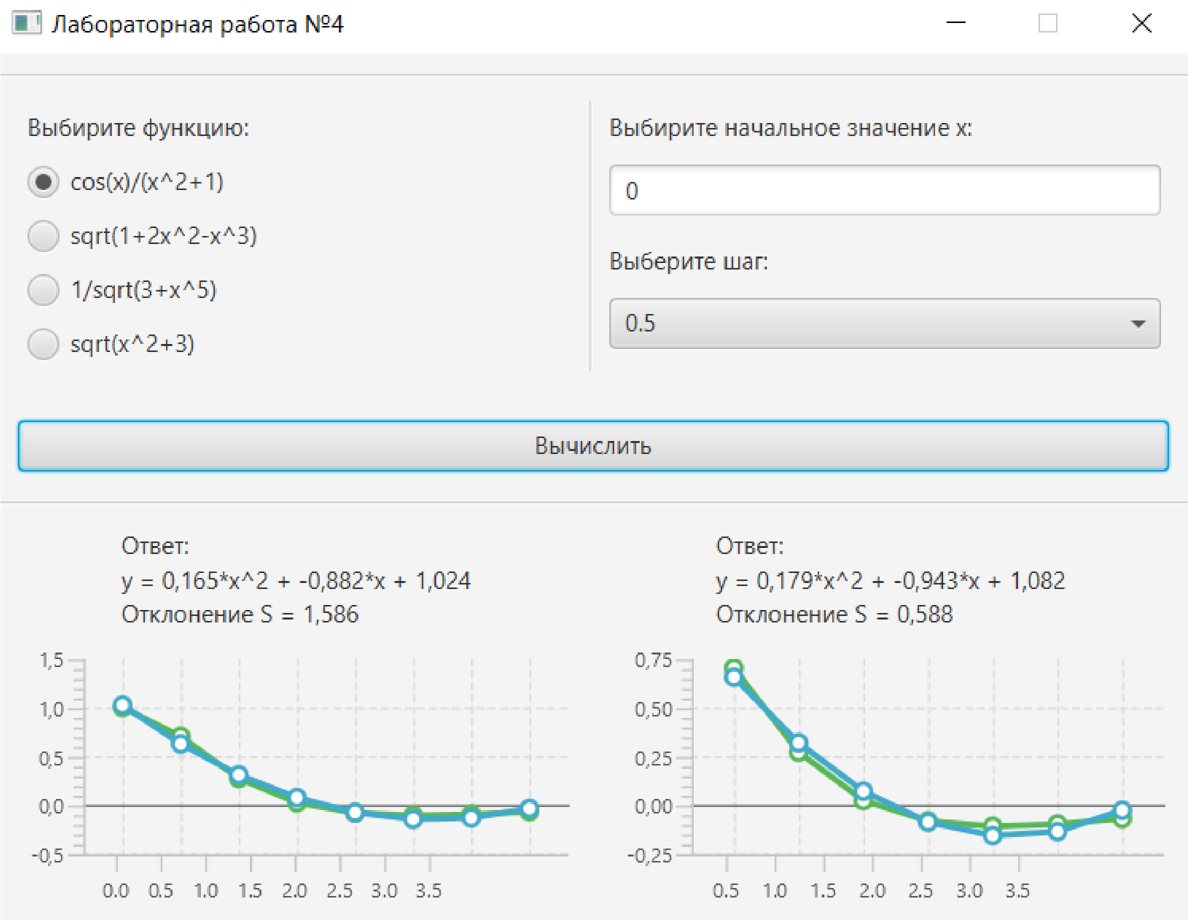
Листинг численного метода

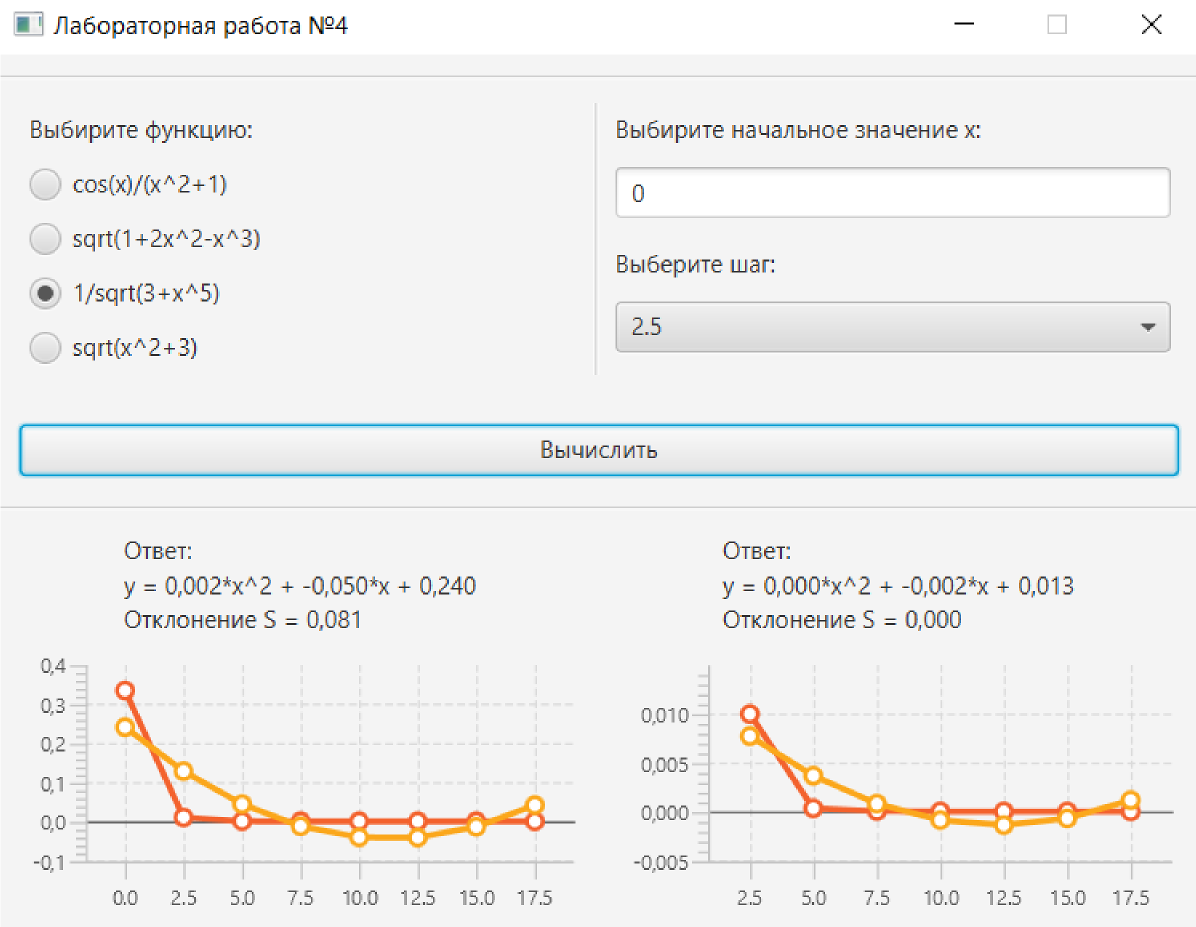
```java
 public void calculateNewOrdinaryLeastSquare() {
        yiDb = new ArrayList<>();
        d = new ArrayList<>();
        dPow2 = new ArrayList<>();
        matrix = new double[][]{
                {xiPow4.get(n)-xiPow4.get(worst_ind), xiPow3.get(n)-
xiPow3.get(worst_ind), xiPow2.get(n)-xiPow2.get(worst_ind),
xiPow2ToYi.get(n)-xiPow2ToYi.get(worst_ind)},
                {xiPow3.get(n)-xiPow3.get(worst_ind), xiPow2.get(n)-
xiPow2.get(worst_ind), xi.get(n)-xi.get(worst_ind), xiyi.get(n)-
xiyi.get(worst_ind)},
                {xiPow2.get(n)-xiPow2.get(worst_ind), xi.get(n)-
xi.get(worst_ind), n-1, yi.get(n)-yi.get(worst_ind)}
        };
        inversionOfMatrix(matrix, 3);
        result = matrixMultiplication();
        a = result[0][0];
        b = result[1][0];
        c = result[2][0];
        findYiDb();
        findD();
        int s = findDpow2();
    }
```

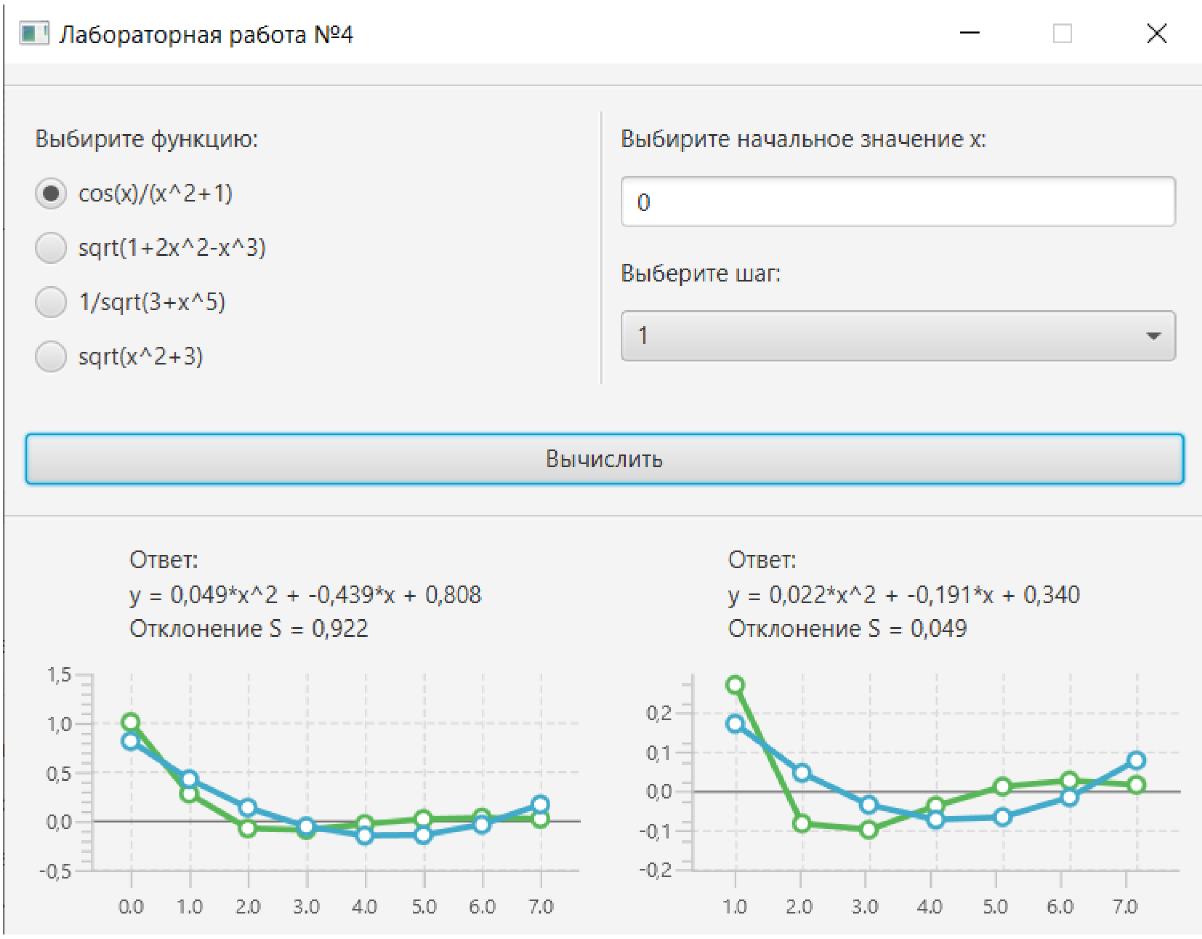
```java
private void findYiDb () {
        double next, sum = 0;
        for (int i = 0; i < n; i++) {
            next = a*xiPow2.get(i) + b*xi.get(i) + c;
            yiDb.add(next);
            sum += next;
        }
        yiDb.add(sum);
    }
```

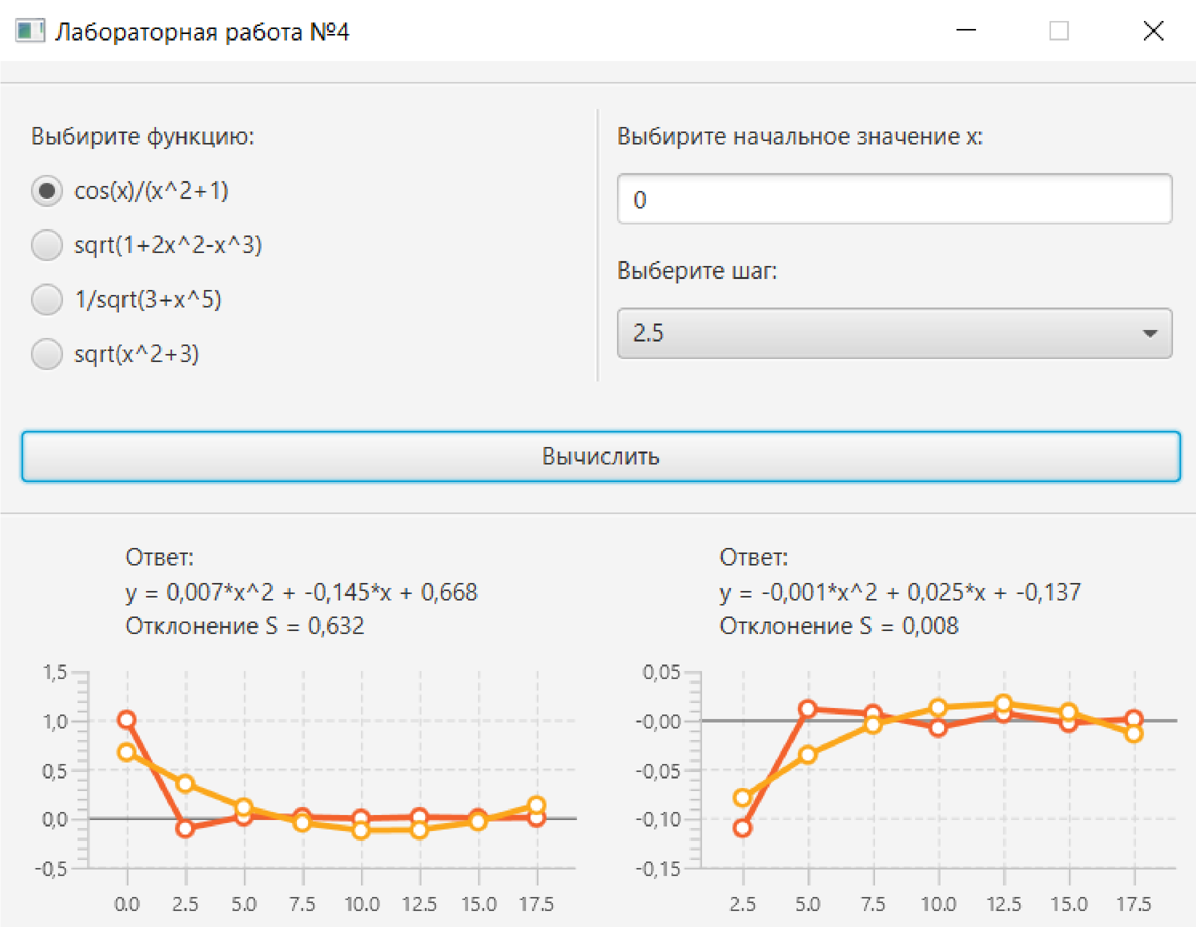
```java
    private void findD () {
        double next, sum = 0;
        for (int i = 0; i < n; i++) {
            next = yi.get(i) - yiDb.get(i);
            d.add(next);
            sum += next;
        }
        d.add(sum);
    }
```

Примеры


Лабораторная работа №4
Выбирите функцию:
cos(x)/(x^2+1)
sqrt(1+2x^2-x^3)
1/sqrt(3+x^5)
sqrt(x^2+3)
Выбирите начальное значение x:
0
Выберите шаг:
0.5
Вычислить
Ответ:
y = 0,165*x^2 + -0,882*x + 1,024
Отклонение S = 1,586
1,5 1,0 0,5 0,0 -0,5
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5
Ответ:
y = 0,179*x^2 + -0,943*x + 1,082
Отклонение S = 0,588
0,75 0,50 0,25 0,00 -0,25
0.5 1.0 1.5 2.0 2.5 3.0 3.5


Лабораторная работа №4
Выбирите функцию:
cos(x)/(x^2+1)
sqrt(1+2x^2-x^3)
1/sqrt(3+x^5)
sqrt(x^2+3)
Выбирите начальное значение x:
0
Выберите шаг:
2.5
Вычислить
Ответ:
y = 0,002*x^2 + -0,050*x + 0,240
Отклонение S = 0,081
0,4 0,3 0,2 0,1 0,0 -0,1
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5
Ответ:
y = 0,000*x^2 + -0,002*x + 0,013
Отклонение S = 0,000
0,010 0,005 0,000 -0,005
2.5 5.0 7.5 10.0 12.5 15.0 17.5


Лабораторная работа №4
Выбирите функцию:
cos(x)/(x^2+1)
sqrt(1+2x^2-x^3)
1/sqrt(3+x^5)
sqrt(x^2+3)
Выбирите начальное значение x:
0
Выберите шаг:
1
Вычислить
Ответ:
y = 0,049*x^2 + -0,439*x + 0,808
Отклонение S = 0,922
Ответ:
y = 0,022*x^2 + -0,191*x + 0,340
Отклонение S = 0,049


Лабораторная работа №4
Выбирите функцию:
cos(x)/(x^2+1)
sqrt(1+2x^2-x^3)
1/sqrt(3+x^5)
sqrt(x^2+3)
Выбирите начальное значение x:
0
Выберите шаг:
2.5
Вычислить
Ответ:
y = 0,007*x^2 + -0,145*x + 0,668
Отклонение S = 0,632
Ответ:
y = -0,001*x^2 + 0,025*x + -0,137
Отклонение S = 0,008

Выводы

Выполнив данную лабораторную работу я сделала следующие выводы:

Во первых необходимо обратить внимание на разницу в постановке задач аппроксимации и интерполяции: интерполянт должен принадлежать к определенному классу и в точках $x_i(i = 0, 1, \ldots, n)$ принимать те же значения, что и исходная функция, для аппроксиманта это требование обязательным не является, но должен выполняться критерий наилучшего приближения.

В большинстве практических случаев вычислений нам требуется *установить определенный вид функциональной зависимости* между характеристиками изучаемого явления. Этой цели и служит задача о приближении функции.

Т.е. задача о приближении (аппроксимации) функции состоит в том, чтобы *данную функцию f(x) приближенно заменить (аппроксимировать) некоторой функцией φ(x), значения которой в заданной области мало отличались от опытных данных (f(x) ≈ φ(x))*.

<u>Построение эмпирической формулы состоит из 2 этапов:</u>

1. *Подбор общего вида формулы.*

Иногда он известен из физических соображений.

Если характер зависимости неизвестен, то первоначально его выбирают геометрически: экспериментальные точки наносятся на график, и примерно угадывается общий вид зависимости путем сравнения полученной кривой с графиками известных функций (многочлена, логарифмической, показательной функций и т.п.).

Выбор вида эмпирической зависимости – наиболее сложная часть решения задачи, так как класс известных аналитических зависимостей необъятен. Практика, однако, показывает, что при выборе аналитической зависимости достаточно ограничиться довольно узким кругом функций: линейные, степенные и показательные.

2. *Определение значений параметров аппроксимирующей функции.*

(это и было реализовано в самой лабораторной работе, а алгоритм описан в теоретическом блоке выше)

Что касается задачи интерполирования, то существует 3 метода:

1. Интерполирование кубическими сплайнами – это 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 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.
2. Формула Лагранжа.
   Основным отличием этого метода является то, что его можно применять для таблиц с различными расстояниями между узлами
3. Формула Ньютона
   Можно применять только для таблиц с равноудаленными 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 всего вышесказанного следует отметить, что большую точность интерполяции можно получить применением методов сплайн–интерполяции.