Лабораторная работа №4

«Исследование адаптивного линейного элемента»

Цель работы:

Углубление теоретических знаний в области разработки нейронных сетей с линейной активационной функцией, исследование свойств квадратичной целевой функции и LMS-алгоритма обучения, приобретение практических навыков обучения однослойной сети линейных адаптивных элементов при решении задачи классификации и адаптивной фильтрации.

Ход работы:

Вариант задания:

	[2; 2]	[-2; 2]	[-2; -1]	[2; -1]	
7	[1; 0], [2; 1]	[-1; 0], [-2; 1]	[-1; -1], [-2; -2]	[1; -1], [2; -2]	

Задача 3.2 (часть 1)

Даны четыре класса, каждый из которых представлен 2-мя примерами (столбцами матрицы P), указанными в таблице 3.2 к лабораторной работе №3 (используйте свой вариант). Необходимо:

- разработать структурную схему классификатора на основе АЛЭ, распознающего эти 4 класса;

- выполнить предварительный анализ задачи, изобразив точки четырех классов и построив графически возможные границы решений;

- задать ту же целевую матрицу T, которая использовалась при выполнении задания 3.3 в лабораторной работе №3, заменив все нули на -1;

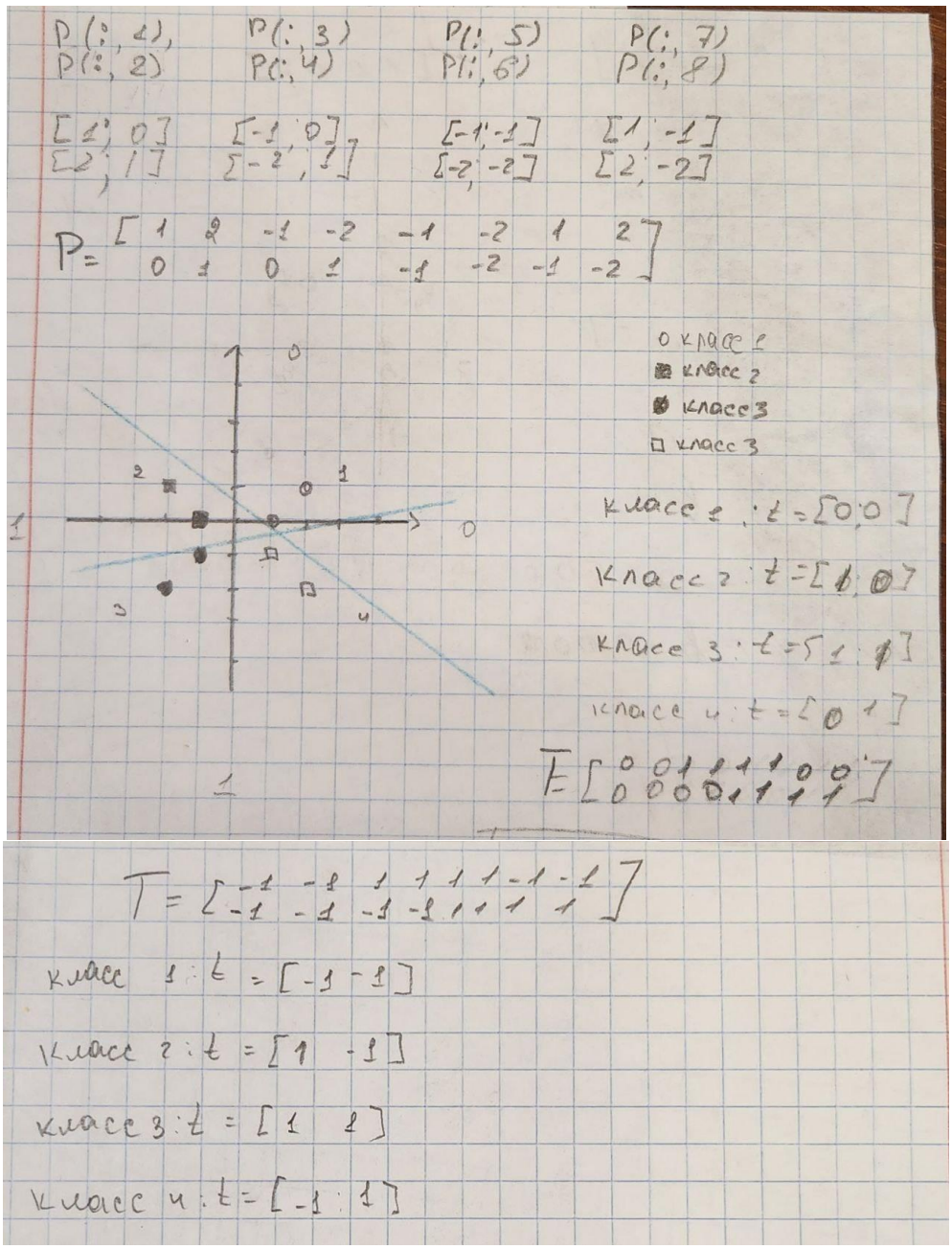

Рисунок 1 – Нахождение матрицы T

- полагая, что все входные векторы p равновероятны, написать программу, вычисляющую корреляционную матрицу C, собственные числа гессиана целевой функции A=2C и максимальное устойчивое значение параметра αmax LMS-алгоритма;

Код программы:

```
P = [1 0;
     2 1;
     -1 0;
     -2 1;
     -1 -1;
```

```
      -2 -2;
      1 -1;
      2 -2];

Q = size(P, 1);
R = size(P, 2) + 1;

Cor = zeros(R, R);

for q = 1:Q
    z = [P(q, :) 1];
    Cor = Cor + z' * z;
end

Cor = 1 / Q * Cor;

evals = spec(2 * Cor);
alpha_max = 1 / max(evals);

disp("Корреляционная матрица C:");
disp(Cor);
disp("Собственные числа гессиана A = 2C:");
disp(evals);
disp("Максимальное устойчивое значение параметра alpha_max LMS-алгоритма:");
disp(alpha_max);
```

```
-->
 "Корреляционная матрица C:"

-->
   2.5    0.     0.
   0.     1.5   -0.5
   0.    -0.5    1.

-->
 "Собственные числа гессиана A = 2C:"

-->
   1.3819660
   3.6180340
   5.

-->
 "Максимальное устойчивое значение параметра alpha_max LMS-алгоритма:"

-->
   0.2
```

Рисунок 2 – нахождение корреляционной матрицы C

Задача 3.2 (часть 2)

- изучить встроенные функции ann_ADALINE и ann_ADALINE_online пакета NeuralNetworks 3.0, реализующие блочный и последовательный варианты LMS-алгоритма;

- используя указанные функции, написать программу, которая:

- отображает диаграмму размещения входных точек (примеров) из P на плоскости с координатами (p1, p2);

- обучает АЛЭ правильному распознаванию входных классов с использованием 2-х указанных функций обучения при разных значениях параметра α;

- строит кривые обучения - зависимости СКО от номера эпохи для 2-х указанных функций обучения (для этого необходимо модифицировать встроенные функции ann_ADALINE и ann_ADALINE_online);

- накладывает на диаграмму входных точек данных границы решения после обучения слоя АЛЭ;

- выполняет тестирование полученного решения для всех заданных входных данных;

- сравнить получаемые границы решения слоя АЛЭ с границами решения персептрона, полученными в лабораторной работе №3, обратив внимание на равноудаленность границ от точек соседних классов для случая слоя АЛЭ;

- сравнить кривые обучения слоя АЛЭ двумя модифицированными функциями ann_ADALINE и ann_ADALINE_online для случая, когда параметр α LMS-алгоритма значительно меньше αmax и когда он близок к αmax .

Код программы:

```
// Функция обучения ADALINE
function [w, b, mse]=ann_ADALINE(P, T, alpha, itermax)
    [m, n] = size(P);
    w = rand(m, 1);
    b = rand();
    mse = zeros(itermax, 1);
```

```
    for epoch = 1:itermax
        for i = 1:n
            x = P(:, i);
            d = T(i);
            y = w' * x + b;
            e = d - y;

            w = w + alpha * e * x;
            b = b + alpha * e;
        end

        predictions = sign(w' * P + b);
        mse(epoch) = mean((predictions - T).^2);
    end
end

// Функция обучения ADALINE Online
function [w, b, mse]=ann_ADALINE_online(P, T, alpha, itermax)
    [m, n] = size(P);
    w = rand(m, 1);
    b = rand();
    mse = zeros(itermax, 1);

    for epoch = 1:itermax
        for i = 1:n
            x = P(:, i);
            d = T(i);
            y = w' * x + b;
            e = d - y;

            w = w + alpha * e * x;
            b = b + alpha * e;

            predictions = sign(w' * x + b);
            mse(epoch) = mse(epoch) + (predictions - d)^2;
        end
    end

    mse = mse / n;
end

// Функция для отображения границы решения
function plot_decision_boundary(w, b)
    x = linspace(-3, 3, 100);
    y = (-w(1) * x - b) / w(2);
    plot(x, y, 'k-', 'LineWidth', 1.5);
end

// Входные точки и классы
P = [1 0 2 1 -1 0 -2 1;
     -1 -1 -2 -2 1 -1 2 -2];

T = [-1 -1 -1 -1 1 1 1 1];

// Отображение диаграммы размещения входных точек
clf; //Очищаем текущий график

subplot(2, 2, 1);
for i = 1:size(P, 2)
    if T(i) == 0
        plot(P(1, i), P(2, i), 'bo', 'MarkerSize', 8, 'MarkerFaceColor',
'b');
    else
```

```
        plot(P(1, i), P(2, i), 'ro', 'MarkerSize', 8, 'MarkerFaceColor',
'r');
    end
end

xlabel('p1');
ylabel('p2');
title('Размещение входных точек');

// Обучение и построение кривых обучения для ADALINE с одной скоростью
обучения
alpha1 = 0.01;
itermax1 = 100;
[w1, b1, mse1] = ann_ADALINE(P, T, alpha1, itermax1);

subplot(2, 2, 2);
epoch1 = 1:itermax1;
plot(epoch1, mse1, 'b-', 'LineWidth', 1.5);
xlabel('Номер эпохи');
ylabel('СКО');
title('Скорость обучения 1');

// Обучение и построение кривых обучения для ADALINE Online с одной скоростью
обучения
alpha2 = 0.1;
itermax2 = 200;
[w2, b2, mse2] = ann_ADALINE_online(P, T, alpha2, itermax2);

subplot(2, 2, 3);
epoch2 = 1:itermax2;
plot(epoch2, mse2, 'r--', 'LineWidth', 1.5);
xlabel('Номер эпохи');
ylabel('СКО');
title('Скорость обучения 1');

// Обучение и построение кривых обучения для ADALINE с другой скоростью
обучения
alpha3 = 0.05;
itermax3 = 150;
[w3, b3, mse3] = ann_ADALINE(P, T, alpha3, itermax3);

subplot(2, 2, 4);
epoch3 = 1:itermax3;
plot(epoch3, mse3, 'g-', 'LineWidth', 1.5);
xlabel('Номер эпохи');
ylabel('СКО');
title('Скорость обучения 2');

// Обучение и построение кривых обучения для ADALINE Online с другой
скоростью обучения
alpha4 = 0.2;
itermax4 = 250;
[w4, b4, mse4] = ann_ADALINE_online(P, T, alpha4, itermax4);

subplot(2, 2, 4);
epoch4 = 1:itermax4;
plot(epoch4, mse4, 'm--', 'LineWidth', 1.5);
xlabel('Номер эпохи');
ylabel('СКО');
title('Скорость обучения 2');

// Тестирование решения для всех входных данных
predictions = sign(w1' * P + b1);
disp('Тестирование решения:');
```

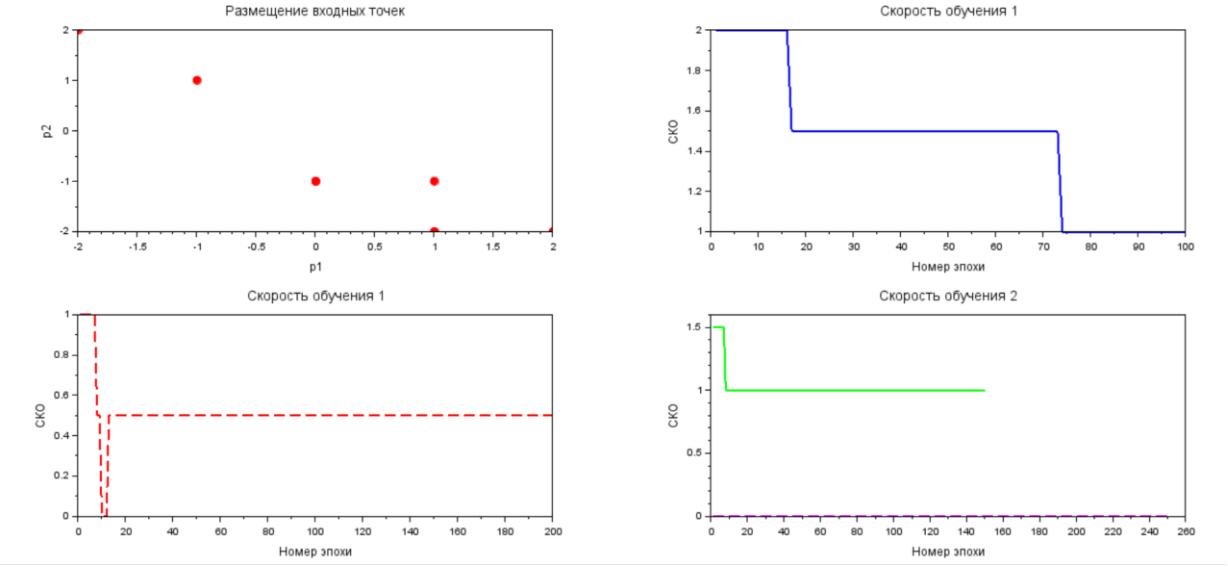

Рисунок 3 – Кривые обучения

Задача 3.3 (Часть 1)

- сгенерируйте входной y(k) и желаемый t(k)=y(k) сигналы адаптивного предсказателя. При этом параметры генератора выбирайте в соответствии с вариантом из таблицы 4.1.

- запишите развернутые выражения для всех элементов (C,h,c) целевой функции предсказателя, заданной в виде СКО (4.12);

- вычислите конкретные значения матрицы C, вектора h и константы c для сгенерированного входного сигнала y(k) ;

- вычислите собственные значения и собственные векторы матрицы Гессе целевой функции предсказателя, точку минимума целевой функции, постройте линии контуров равных уровней целевой функции;

- вычислите максимальное устойчивое значение скорости обучения αmax для LMS-алгоритма;

Вариант задания:

Вариант	Число составляющих L	Основная частота F	Частота среза Fc
7	10	0.02	F*6

Код программы:

```
L = 10;
F = 0.02;
Fc = F * 6;

// Генератор полигармонического входного сигнала
td = 1 / (20 * F);
t = 0:td:2 / F;
fi = (2 * %pi * F) .* t;
X = 0;
for i = 1:L
    X = X + (Fc) / (Fc + 2 * %pi * F * i) * sin(fi * i);
end

Y=X(2:$) // Y задержен на такт - вход предсказателя
T=X(1:$-1); // желаемый выход предсказателя

D = 2;  // кол-во элементов задержки предсказателя
P = [];
for cnt = 1:D  // для каждого выхода линии задержки
    // формируем строки матрицы P из отсчетов Y
   // очередная строка P - сдвинутая на один отсчет копия предыдущей строки
 P = [P; Y(cnt:$-D+cnt-1)];
end

//формируем вектор целевых значений с длиной, равной длине строки из P
T1 = T(1:$-D+1);

// Построение графика формирования матрицы входных данных
figure;
plot(P');
title('Формирование матрицы входных данных');
xlabel('Отсчеты');
ylabel('Значения');
legend('p1', 'p2');
```

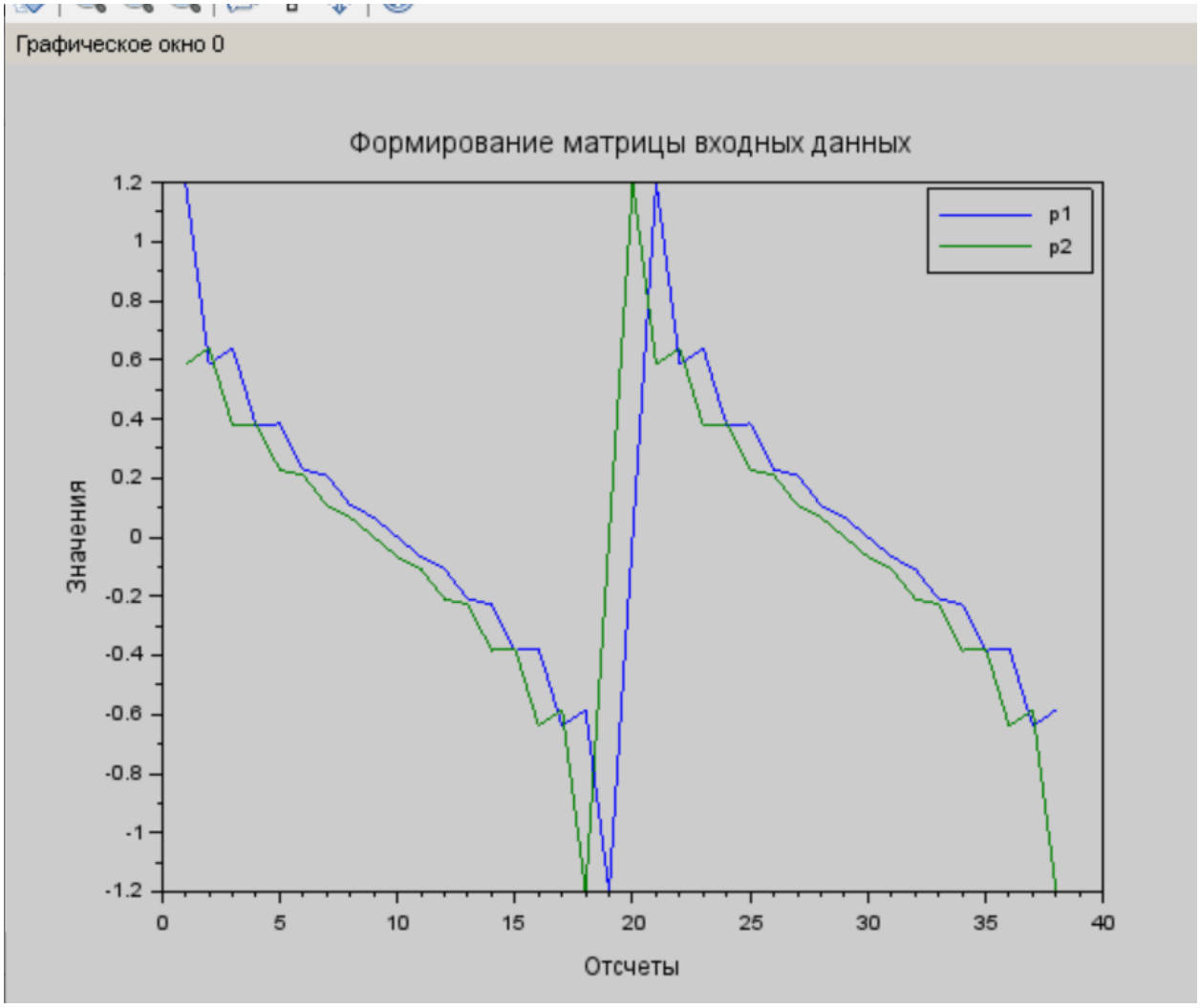

Рисунок 4 – Формирование матрицы входных данных

```
--> Y =

        column 1 to 13

  1.199909   0.5847072   0.6389186   0.376535   0.383262   0.2274078   0.2093546   0.1076964   0.0668999   7.743D-17  -0.0668999  -0.1076964  -0.2093546

        column 14 to 26

 -0.2274078  -0.383262  -0.376535  -0.6389186  -0.5847072  -1.199909   5.125D-15   1.199909   0.5847072   0.6389186   0.376535   0.383262   0.2274078

        column 27 to 39

  0.2093546   0.1076964   0.0668999  -2.263D-16  -0.0668999  -0.1076964  -0.2093546  -0.2274078  -0.383262  -0.376535  -0.6389186  -0.5847072  -1.199909

        column 40

  1.025D-14
```

Рисунок 5 – Выход предсказателя

Задача 3.3 (Часть 2)

- напишите программу, обучающую предсказатель с использованием встроенной функции ann_ADALINE_predict пакета NeuralNetworks 3.0;

- модифицируйте функцию ann_ADALINE_predict таким образом, чтобы по результатам её работы можно было построить кривую обучения предсказателя и траекторию движения вектора параметров предсказателя на диаграмме контуров равных уровней;

- постройте в одном графическом окне графики входного процесса и его предсказанных значений, кривую обучения, траекторию движения вектора параметров на диаграмме контуров;

- убедитесь, что алгоритм обучения сходится, если ααmax ;

- убедитесь, что при малых α траектория движения вектора параметров при использовании LMS алгоритма аппроксимирует в среднем траекторию движения вектора параметров алгоритма наискорейшего спуска

Код основной программы:

```
exec('C:\Users\Мария\Desktop\Универ\3 курс\ДПО\ОСН\Лаба
4\ann_ADALINE1_predict.sci')

D = 2; // кол-во элементов задержки предсказателя
P = [];
for cnt = 1:D // для каждого выхода линии задержки
    //формируем строки матрицы P из отсчетов Y
    // очередная строка P - сдвинутая на один отсчет копия предыдущей строки
    P = [P; Y(cnt:$-D+cnt-1)];
end

//формируем вектор целевых значений с длиной, равной длине строки из P
T1 = T(1:$-D+1);

Q = size(P, 2); // кол-во обучающих примеров
R = size(P, 1); // кол-во входов АЛЭ
Cor = zeros(R, R); // корреляционная матрица
c = 0;
h = 0;
for q = 1:Q
    p = P(:, q); // q-ый входной вектор
    Cor = Cor + (p * p').*1 / Q;
    c = c + T1(q)^2 * 1 / Q;
    h = h + T1(q) * p * 1 / Q;
end

[evals, diagevals] = spec(2 * Cor);
xstar = inv(Cor) * h; //решение Винера

function z=f(w1, w2, c, h, Cor)
    x = [w1; w2];
    z = c - 2 * x' * h + x' * Cor * x;
endfunction

x = linspace(-20, 20, 100); // диапазон по оси x
```

```
y = linspace(-20, 20, 100); // диапазон по оси y
z = feval(x, y, f); //вычисляем значения высот целевой функции f на сетке x,y

clf(1);
figure(1);
subplot(1, 2, 1);
surf(x, y, z'); //строим 3D поверхность целевой функции
subplot(1, 2, 2);
contour2d(x, y, z, 30); //отображаем линии контуров равных уровней
xset("fpf", " ") //подавляет отображение чисел над линиями контуров
xtitle("Контуры равных уровней квадратичной функции", "w1", "w2");
xgrid;
plot(xstar(1), xstar(2), '*b'); //отображаем точку решения минимума СКО (4.15

// Создание и обучение предсказателя по значениям P и T
// Используем модифицированную функцию предсказателя
[w, b, y, ee, mse, W] = ann_ADALINE1_predict(X, T, 0.1 * alpha_max, 40, D,
'ones');
plot2d(W(:, 1), W(:, 2), 5); //отображение траектории движения весо
```

Код программы ann_ADALINE1_predict:

```
function [w, b, y, ee, mse, W]=ann_ADALINE1_predict(Y, T, alpha, itermax, D,
initfunc)
    mse = zeros(1, itermax); // создаем вектор СКО
    w = initfunc(D, 1); // инициализируем переменную w
    W = [w]; // матрица, каждая строка которой – вектор весов на очередном
шаге
    itercnt = 0; // Счетчик итераций - эпох

    while itercnt < itermax
        for cnt = 1:size(Y, 2) // Цикл по всем обучающим примерам из Y (1
эпоха)
            e = T(cnt) - y; // Вычисляем ошибку предсказания
            w = (w + 2 * alpha * e * Y(:, cnt)');
            // b = b + 2 * alpha * e; предсказатель не использует смещение
            // Вычисляем и запоминаем квадраты текущих ошибок
            e_all(cnt) = e.^2;
            W = [W; w];
        end

        itercnt = itercnt + 1;
        mse(itercnt) = mean(e_all); // вычисляем СКО на шаге и запоминаем

        //disp('Epoch: ...
    end
endfunction
```

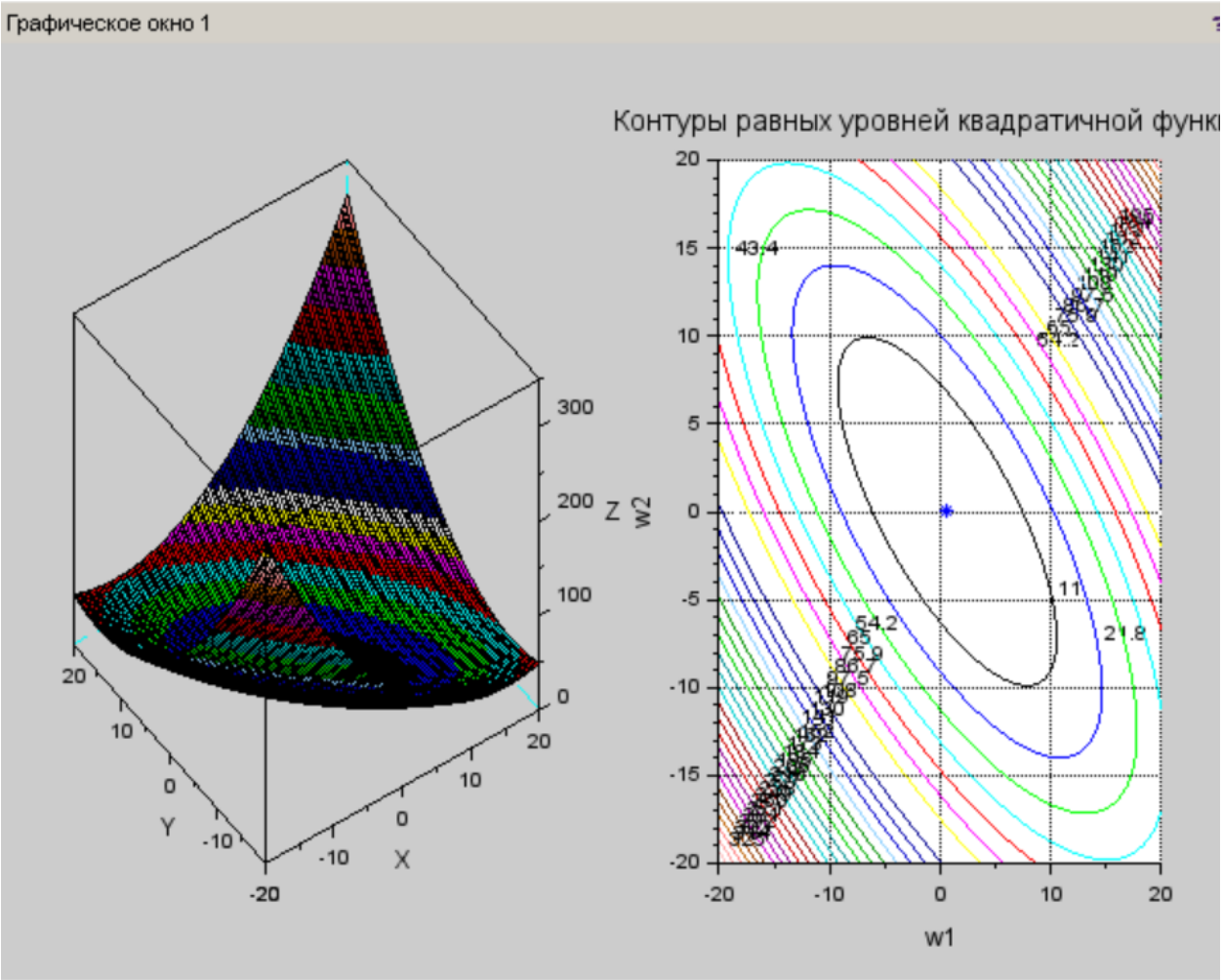

Рисунок 5 - Поверхность целевой функции, линии контуров и траектория движения вектора параметров для LMS-алгоритма W[-1.199909; 1.025D-14]

Вывод:

В ходе лабораторной работы были углублены теоретические знания в области разработки нейронных сетей с линейной активационной функцией, исследование свойств квадратичной целевой функции и LMS-алгоритма обучения. Были приобретены практические навыки обучения однослойной сети линейных адаптивных элементов при решении задачи классификации и адаптивной фильтрации.

Был сделан вывод, что при использовании блочного алгоритма функция ведёт себя более «спокойно». Т.е. более устойчива, чем при использовании online алгоритма.

Были построены график формирования входных данных. На котором видно, что обучение смещается из-за задержки, но график почти не изменяется.

Так же были построены поверхность целевой функции, линии контуров и траектория движения вектора параметров для LMS-алгоритма и вычислено значение W[-1.199909; 1.025D-14]. Для наглядного представления работы алгоритма.