МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)
Кафедра вычислительной математики и программирования

Лабораторная работа №7
по спецкурсу «Нейроинформатика»

Автоассоциативные сети с узким горлом

Выполнил: Днепров И.С.
Группа: М8О-407Б, вариант 10
Преподаватели: Тюменцев Ю.В.

Москва, 2020

Цель работы

Целью работы является исследование свойств автоассоциативных сетей с узким горлом, алгоритмов обучения, а также применение сетей для выполнения линейного и нелинейного анализа главных компонент набора данных.

Основные этапы работы:

1. Использовать автоассоциативную сеть с узким горлом для отображения набора данных, выделяя первую главную компоненту данных.

2. Использовать автоассоциативную сеть с узким горлом для аппроксимации кривой на плос- кости, выделяя первую нелинейную главную компоненту данных.

3. Применить автоассоциативную сеть с узким горлом для аппроксимации пространственной кривой, выделяя старшие нелинейные главные компоненты данных.

Оборудование

Процессор: 2,4 GHz Intel Core 2 Duo

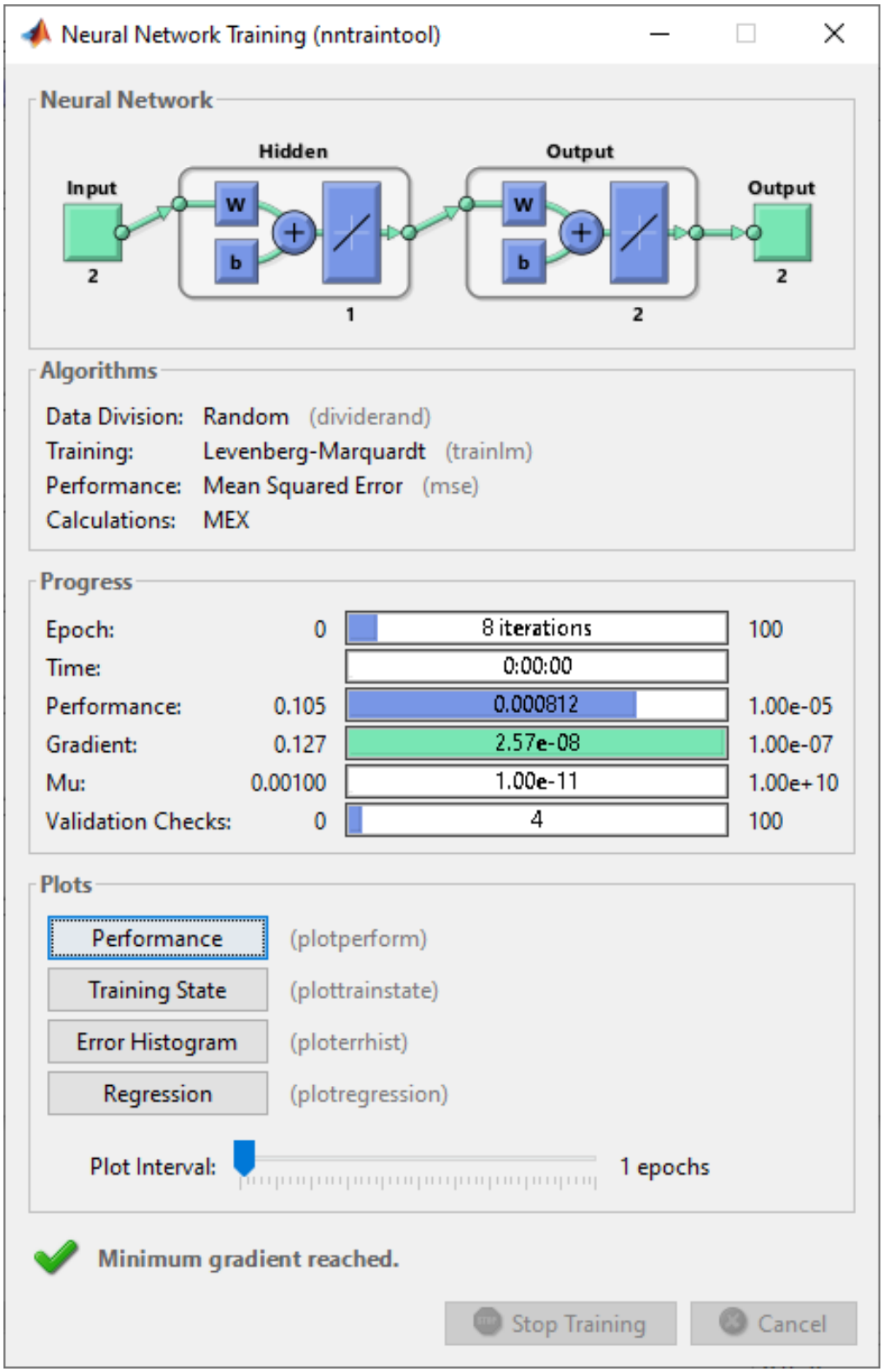
Оперативная память: 8 ГБ 1067 MHz DDR3

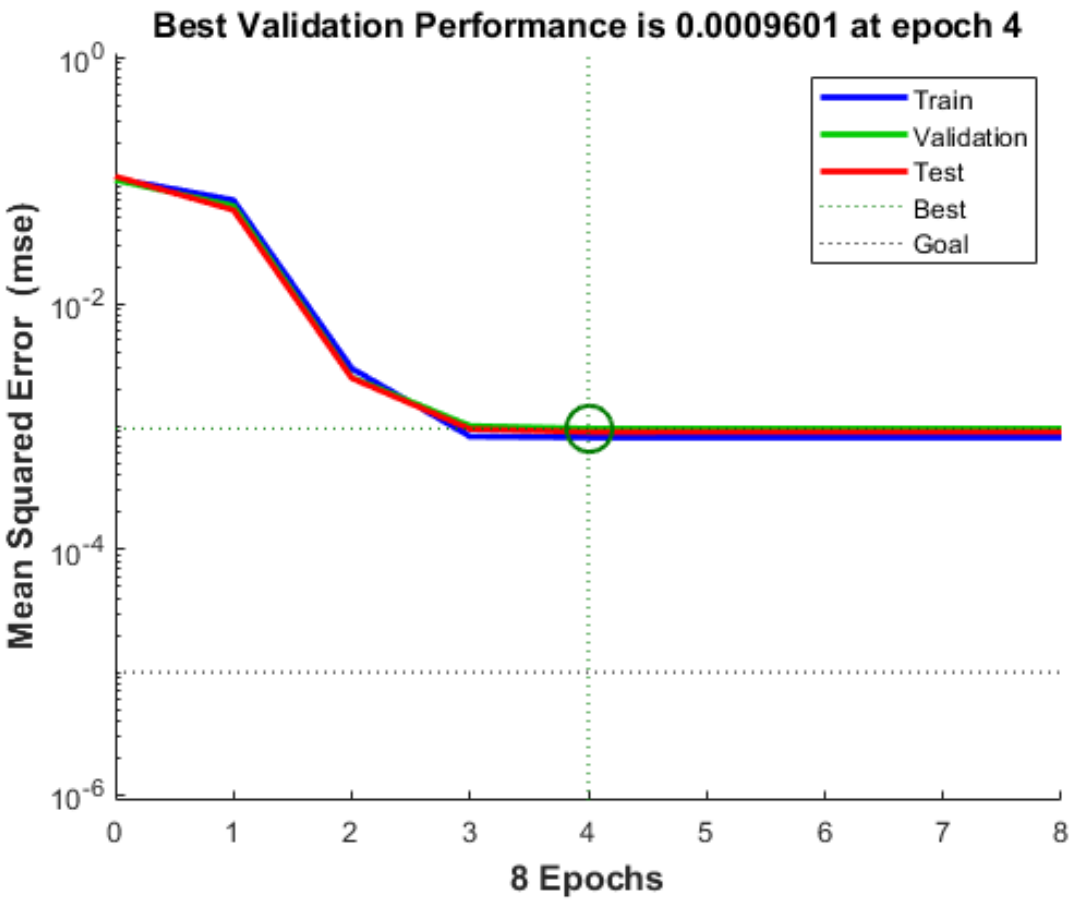
Программное обеспечение

Matlab R2020b, 64-bit.

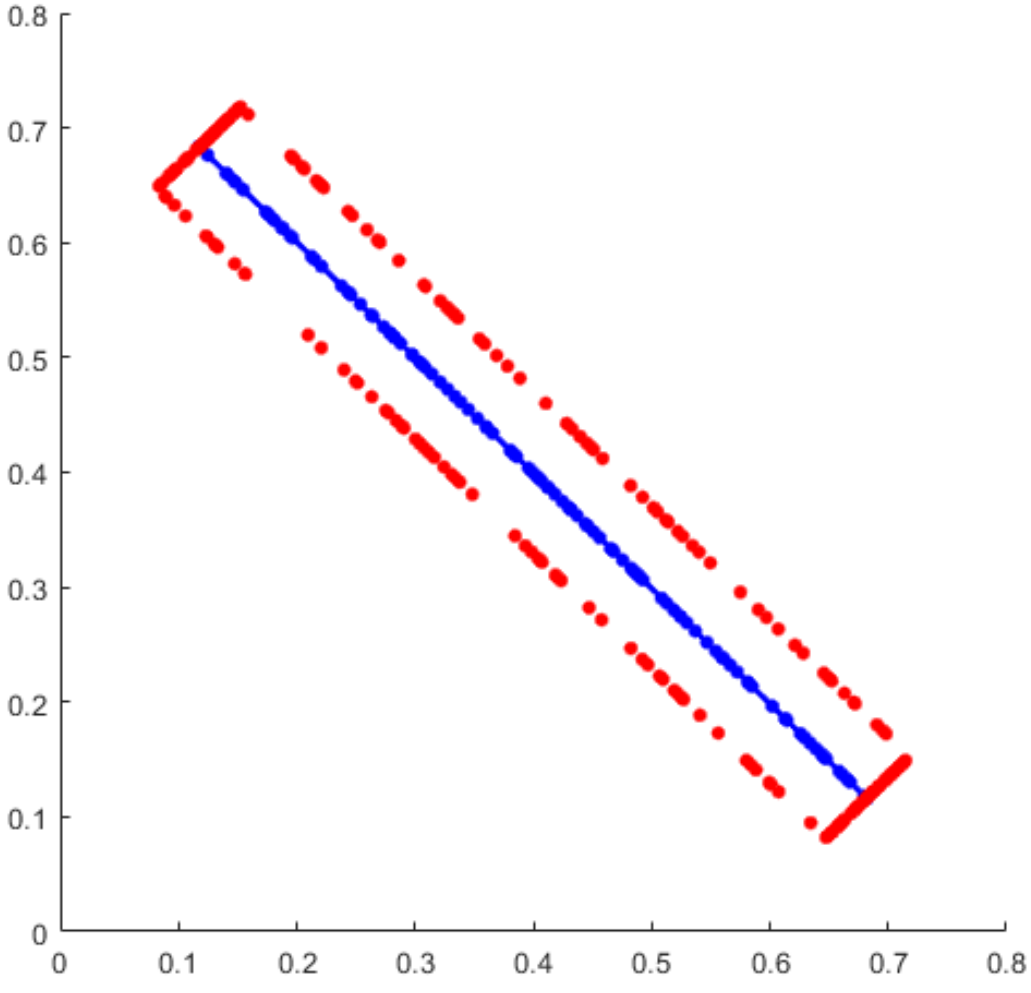
Задание 1.

Обучение сети:



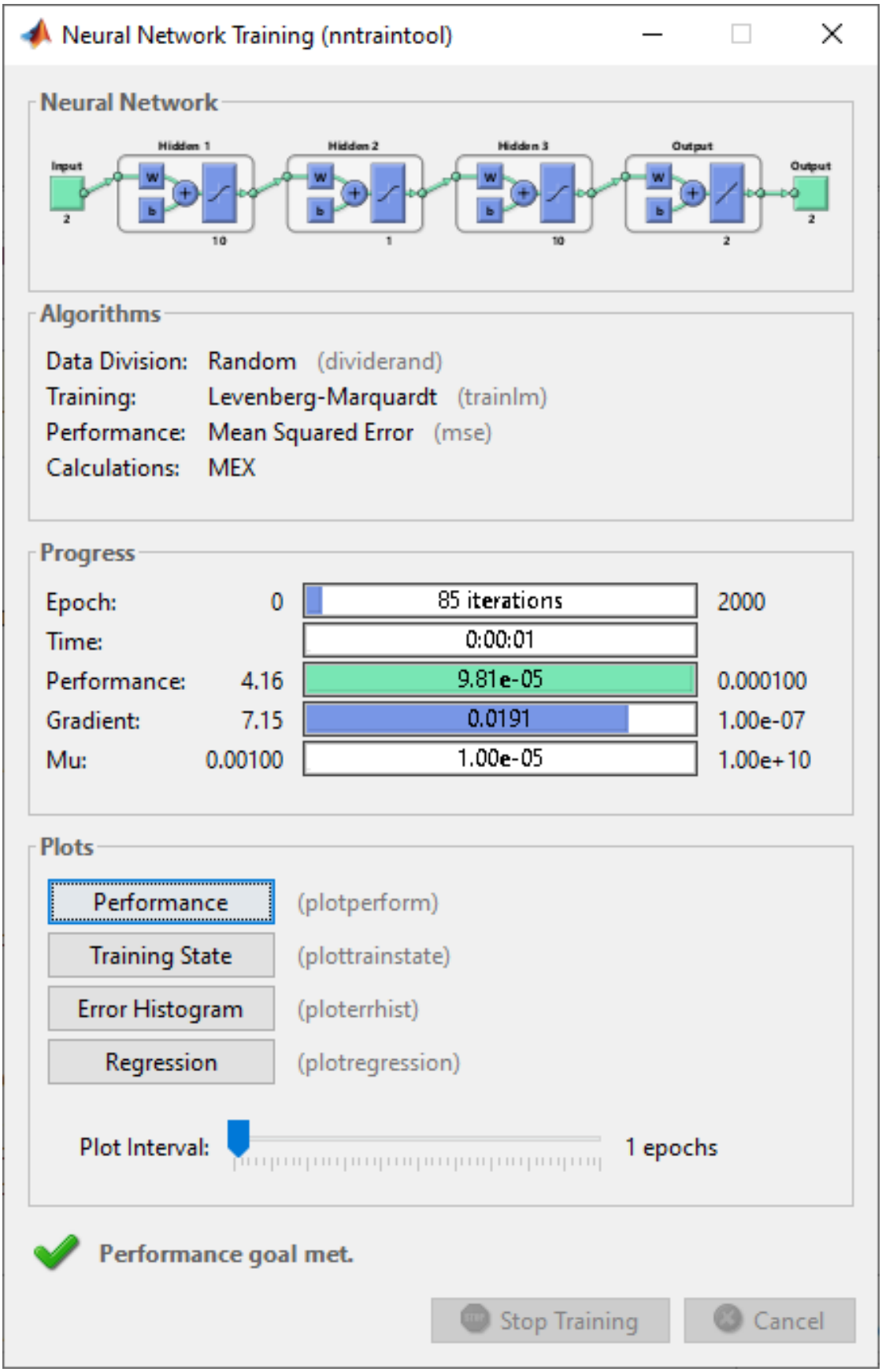


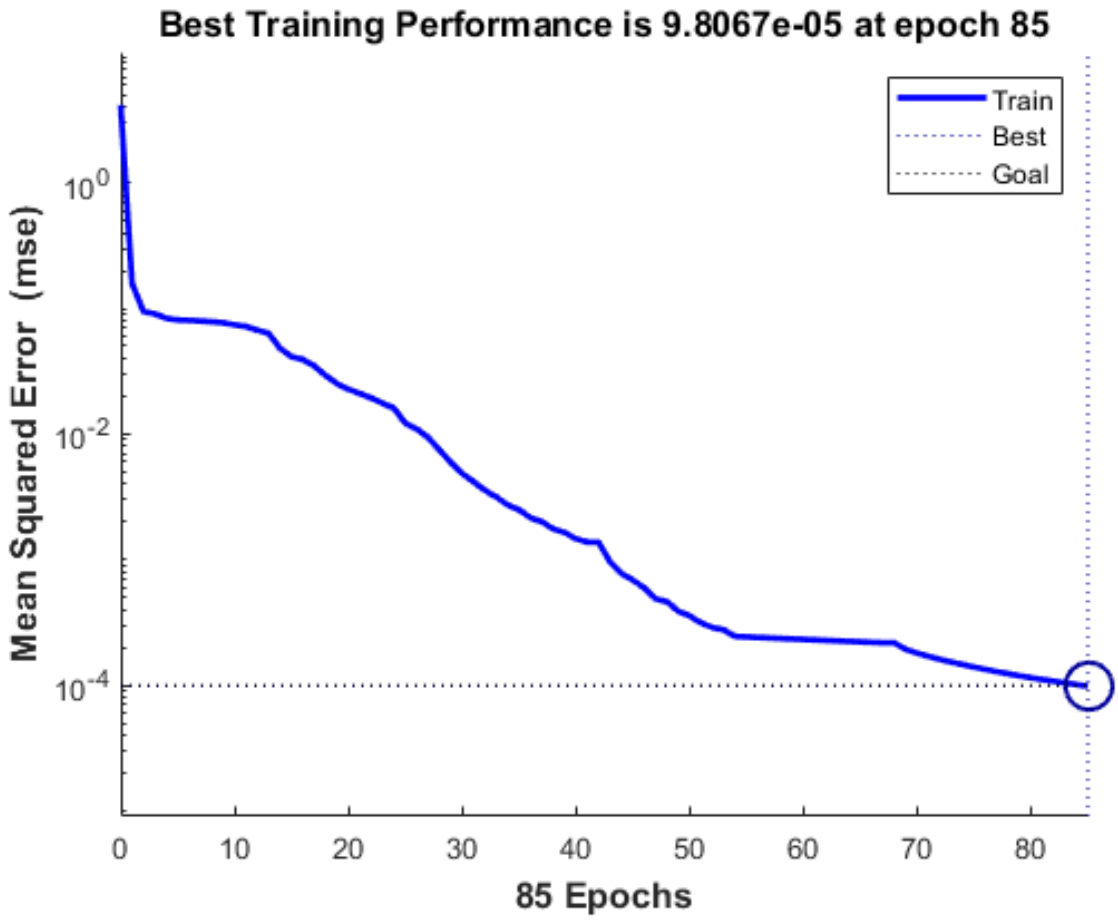
Результат обучения:



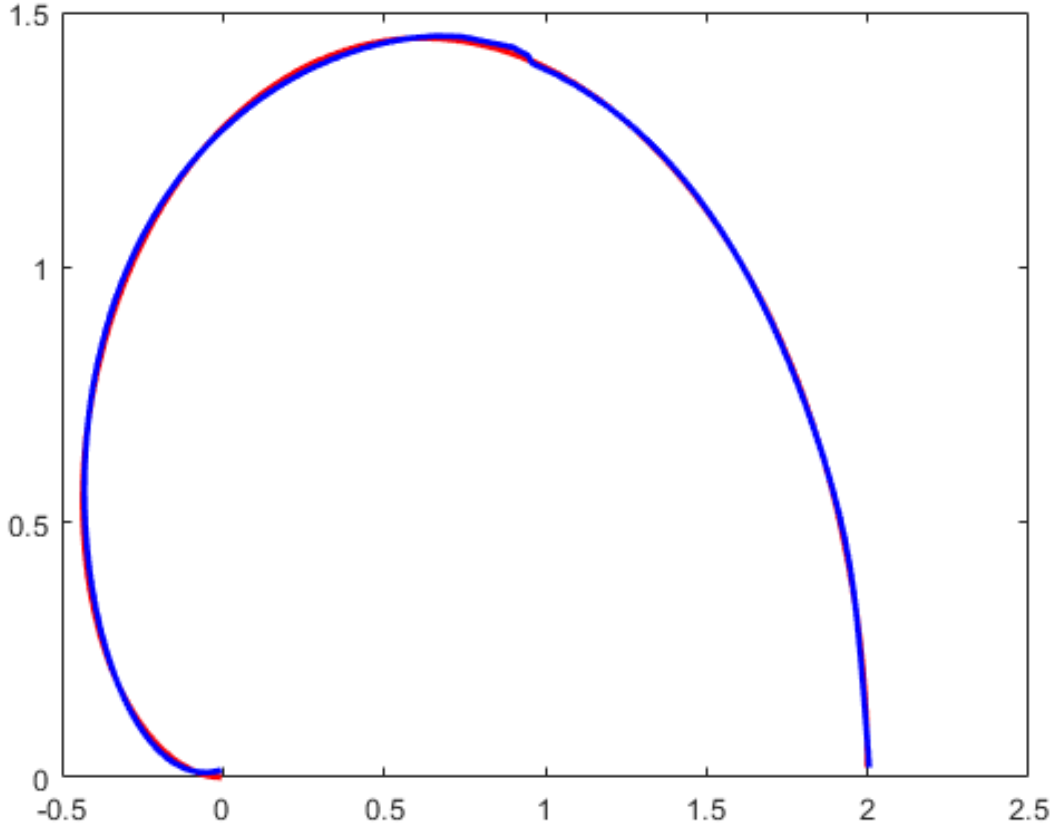
Задание 2.

Обучение сети:



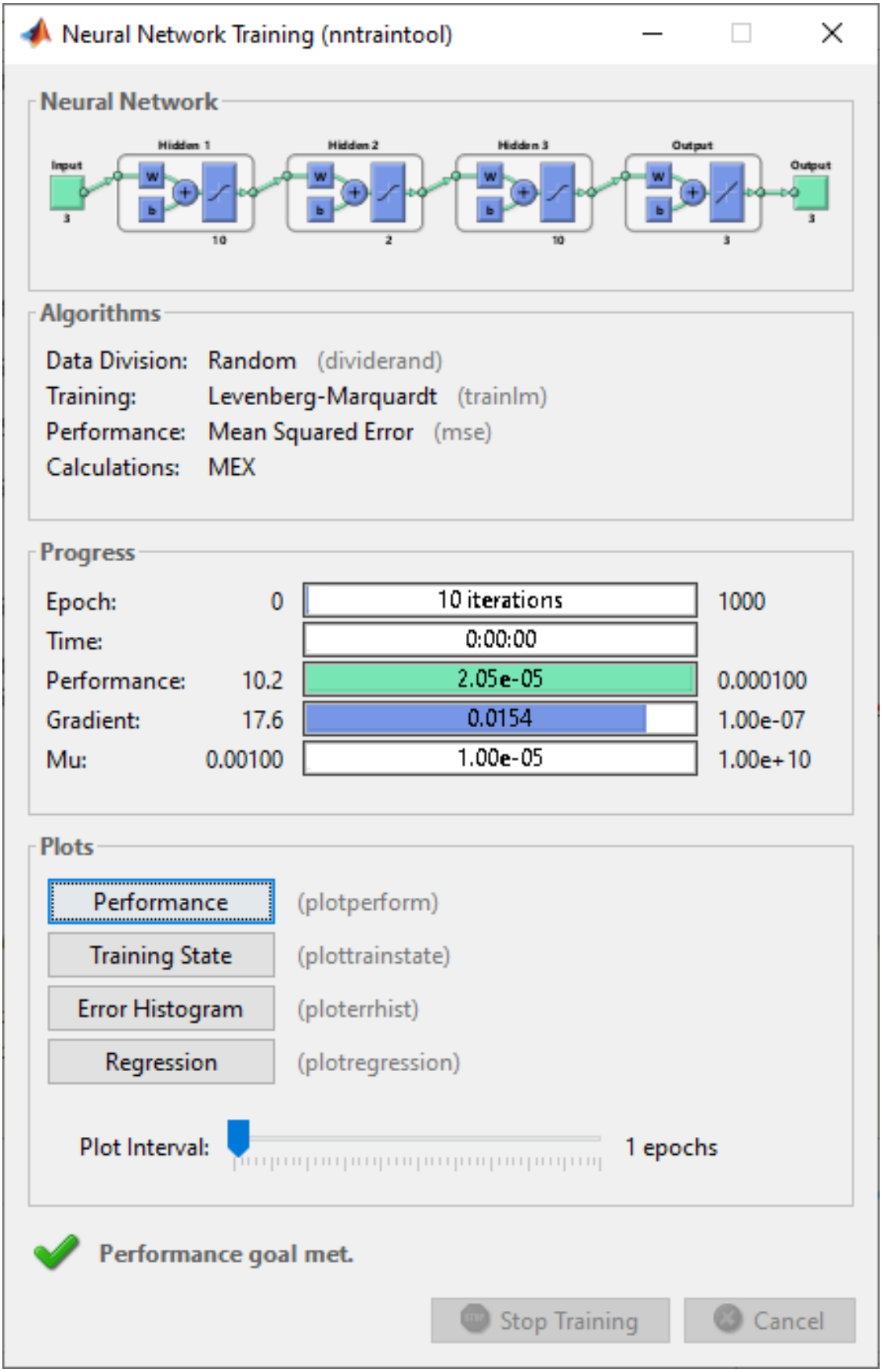


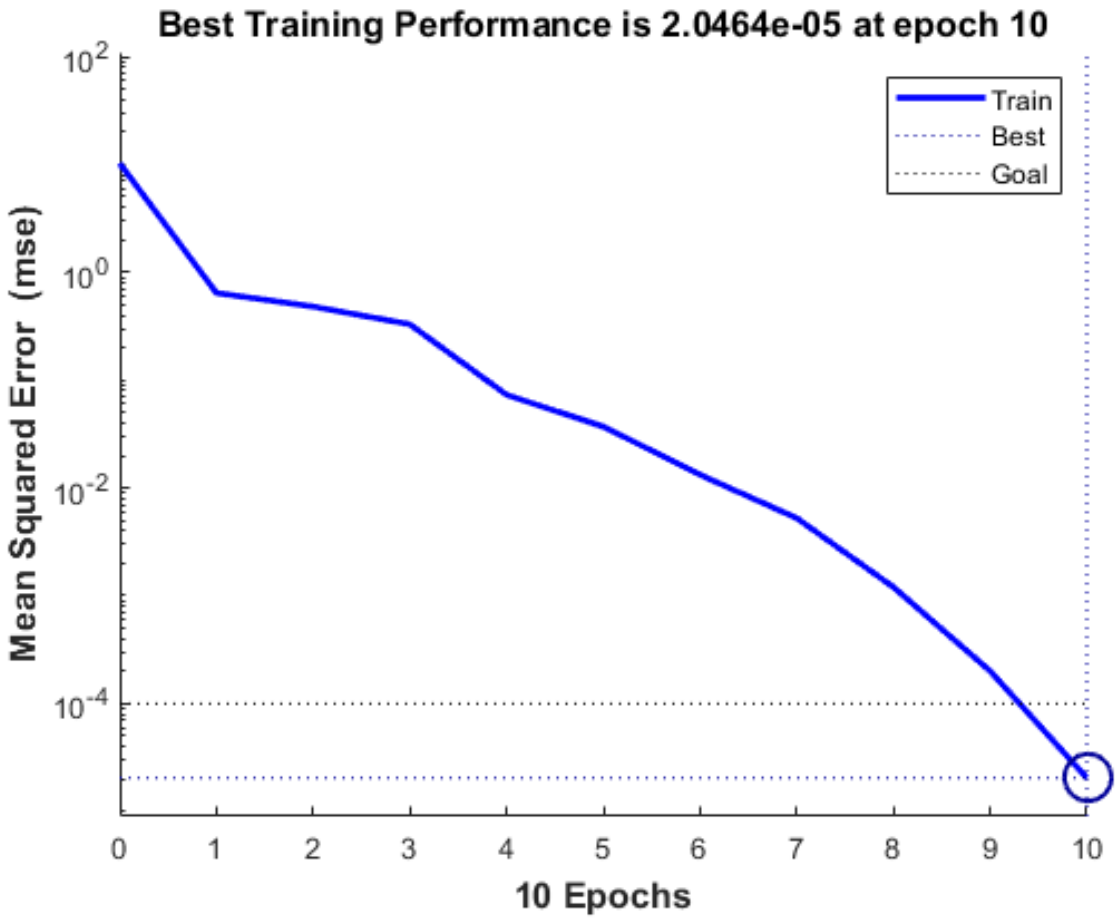
Результат обучения:



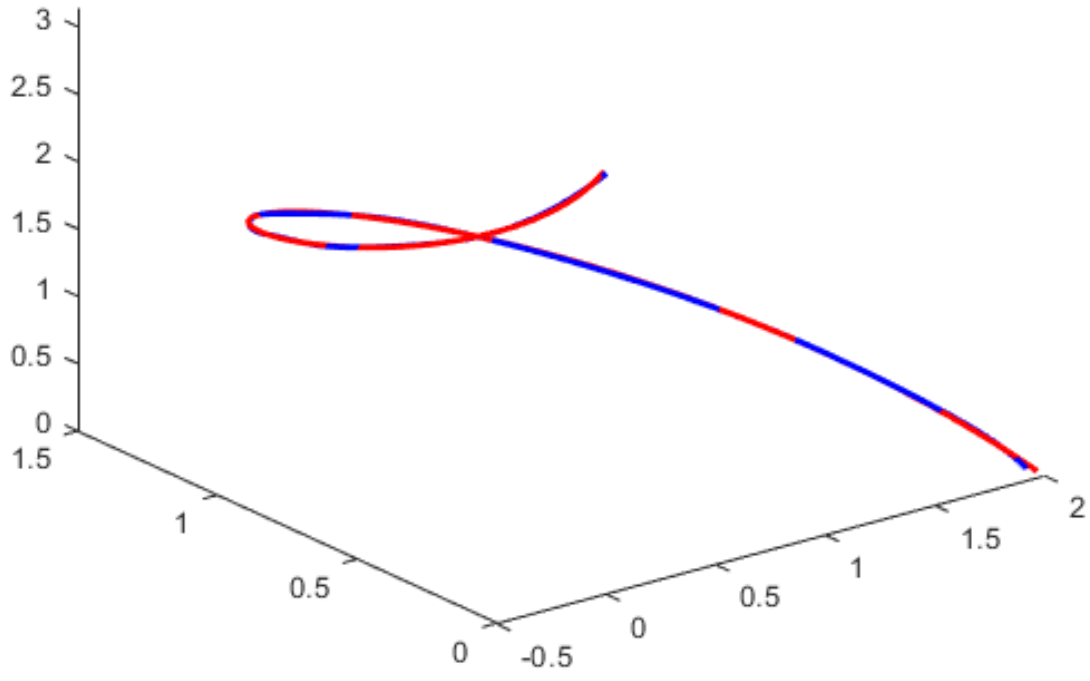
Задание 3.

Обучение сети:





Результат обучения:



Код программы

Lab7.m

```
% Задание 1
% Генерируем обучающее множесво
t = 0:0.025:2*pi;
d1 = 0.1;
d2 = 0.8;
alpha = pi / 4;
x0 = 0.4;
y0 = 0.4;

% Получаем количество точек для каждой стороны
angle = t - pi /4;
firstSideLength = length(angle(angle < pi / 4));
secondSideLength = length(angle(angle > pi / 4 & angle < 3 * pi / 4));
thirdSideLength = length(angle(angle > 3 * pi / 4 & angle < 5 * pi / 4));
fourthSideLength = length(angle(angle > 5 * pi /4));

% Генерируем точки для каждой стороны
probabilityDistribution = makedist('Uniform',-d2/2, d2/2);
firstSideDots = [repmat(d1/2, 1, firstSideLength); random(probabilityDistribution, 1,
firstSideLength)];
probabilityDistribution = makedist('Uniform',-d1/2, d1/2);
secondSideDots = [random(probabilityDistribution, 1, secondSideLength) ;
repmat(d2/2, 1, secondSideLength)];
probabilityDistribution = makedist('Uniform',-d2/2, d2/2);
thirdSideDots = [repmat(-d1/2, 1, thirdSideLength); random(probabilityDistribution,
1, thirdSideLength)];
probabilityDistribution = makedist('Uniform',-d1/2, d1/2);
fourthSideDots = [random(probabilityDistribution, 1, fourthSideLength) ; repmat(-
d2/2, 1, fourthSideLength)];

% Домножеме на матрицу поворота
transform = [cos(alpha), -sin(alpha); sin(alpha), cos(alpha)];
rectandleDots = [firstSideDots,secondSideDots,thirdSideDots,fourthSideDots];
points = transform * rectandleDots + [x0; y0];

% Обучаем нейросеть
network = feedforwardnet(1);
network = configure(network, points, points);
network.layers{1}.transferFcn = 'purelin';
network.layers{2}.transferFcn = 'purelin';
network.trainParam.epochs = 100;
network.trainParam.goal = 1e-5;
network.trainParam.max_fail = 100;
network = train(network, points, points);

%% Результат обучения
y = sim(network, points);
```

```
hold on
plot(y(1,:),y(2,:),'marker','.','markersize',15, 'color', 'b')
plot(points(1,:),points(2,:),'linestyle', 'none','marker','.','markersize',
15,'color','r')
hold off

%% Задание 2
% Генерируем обучающее множесво
phi = 0.01 : 0.025 : pi;
r = 2 * sin(phi) ./ phi;
x = [ r .* cos(phi); r .* sin(phi)];
xPrepared = con2seq(x);

% Создаем сеть
network = feedforwardnet([10, 1, 10], 'trainlm');
network.layers{1}.transferFcn = 'tansig';
network.layers{2}.transferFcn = 'tansig';
network.layers{3}.transferFcn = 'tansig';
network.layers{4}.transferFcn = 'purelin';
network = configure(network, xPrepared, xPrepared);
network = init(network);

% Обучаем ее
network.trainParam.epochs = 2000;
network.trainParam.goal = 10e-5;
network = train(network, xPrepared, xPrepared);
yPrepared = sim(network, xPrepared);
y = cell2mat(yPrepared);

%% Результат обучения
plot(x(1, :), x(2, :), '-r', y(1, :), y(2, :), '-b', 'LineWidth', 2);

%% Задание 3
% Генерируем обучающее множесво
phi = 0.01 : 0.025 : pi;
r = 2 * sin(phi) ./ phi;
x = [ r .* cos(phi); r .* sin(phi); phi];
xPrepared = con2seq(x);

% Создаем сеть
network = feedforwardnet([10, 2, 10], 'trainlm');
network.layers{1}.transferFcn = 'tansig';
network.layers{2}.transferFcn = 'tansig';
network.layers{3}.transferFcn = 'tansig';
network.layers{4}.transferFcn = 'purelin';
network = configure(network, xPrepared, xPrepared);
network = init(network);

% Обучаем ее
network.trainParam.epochs = 1000;
network.trainParam.goal = 10e-5;
network = train(network, xPrepared, xPrepared);
yPrepared = sim(network, xPrepared);
y = cell2mat(yPrepared);
```

```
%% Результат обучения
plot3(x(1, :), x(2, :), x(3, :), '-r', y(1, :), y(2, :), y(3, :), '-b', 'LineWidth',
2);
```

Вывод

В этой лабораторной я научился использовать автоассоциативные сети с узким горлом для аппроксимации функций и отображения данных путем выделения их линейных и нелинейных компонент.

Данные сети применимы в тех задачах, где обучение должно происходить в режиме реального времени, адаптируясь к текущему потоку данных. Также они применимы при нелинейном сжатии информации. Просто меняем линейные нейроны на нелинейные и теперь нам не нужно владеть явным решением, чтобы найти оптимальное сжатие.