

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

**Лабораторная работа №4**  
**по спецкурсу «Нейроинформатика»**

**Сети с радиальными базисными элементами**

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

Москва, 2020

### **Цель работы**

Исследование свойств некоторых видов сетей с радиальными базисными элементами, алгоритмов обучения, а также применение сетей в задачах классификации и аппроксимации функции.

### **Основные этапы работы**

1. Использовать вероятностную нейронную сеть для классификации точек в случае, когда классы не являются линейно разделимыми.
2. Использовать сеть с радиальными базисными элементами (*RBF*) для классификации точек в случае, когда классы не являются линейно разделимыми.
3. Использовать обобщенно-регрессионную нейронную сеть для аппроксимации функции. Проверить работу сети с рыхлыми данными.

### **Оборудование**

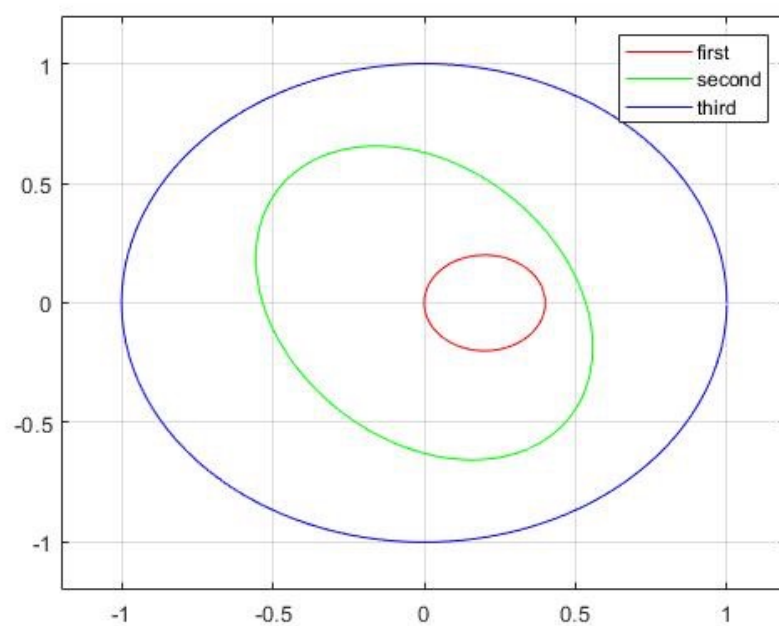
Процессор: 2,4 GHz Intel Core 2 Duo

Оперативная память: 8 ГБ 1067 MHz DDR3

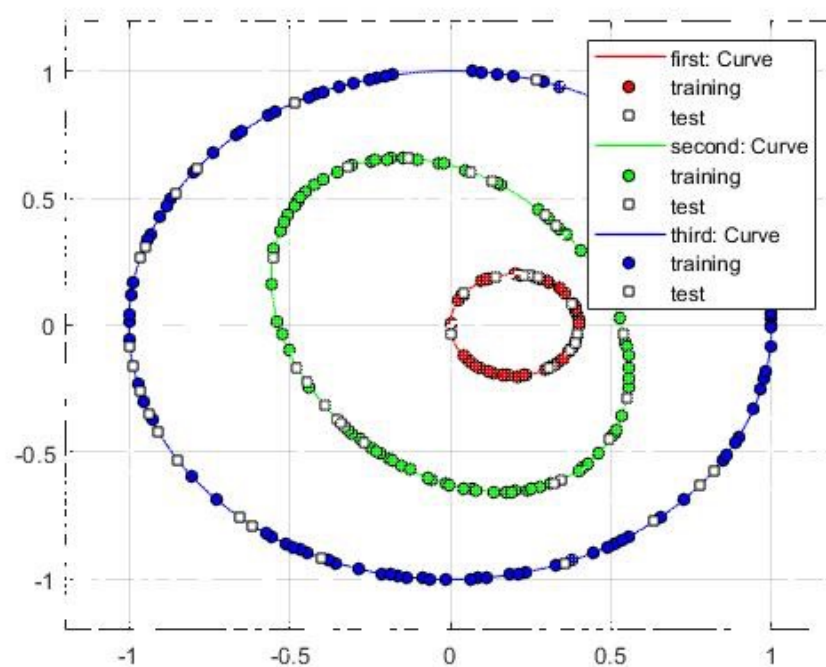
### **Программное обеспечение**

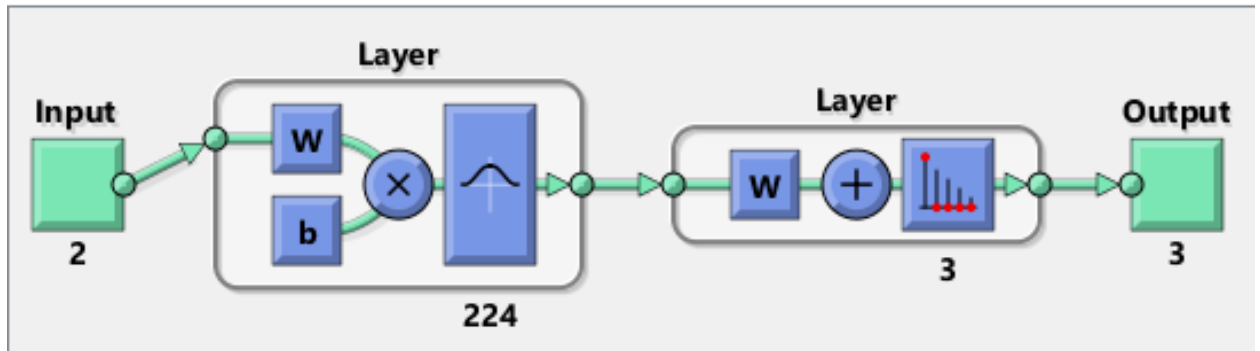
*Matlab R2020b, 64-bit.*

Алгебраические линии:

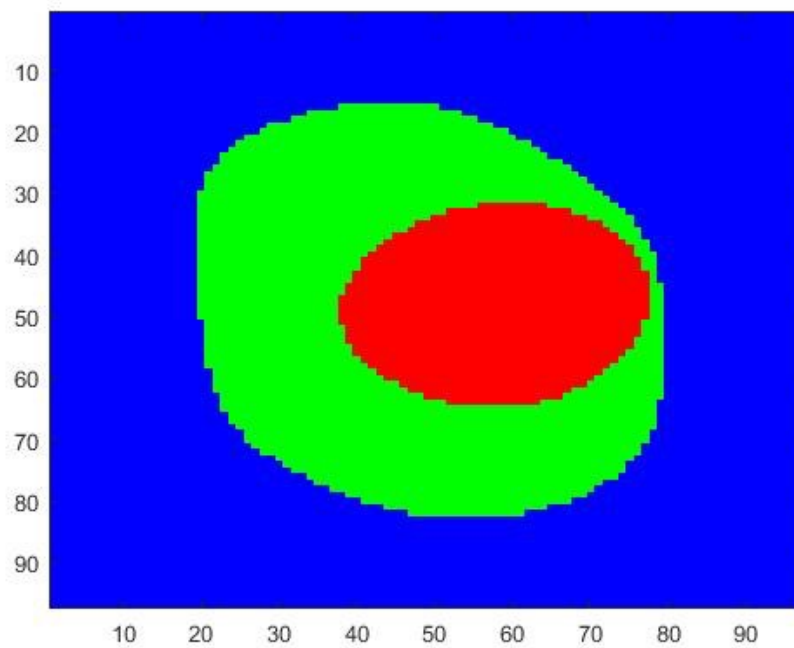


Множество точек:

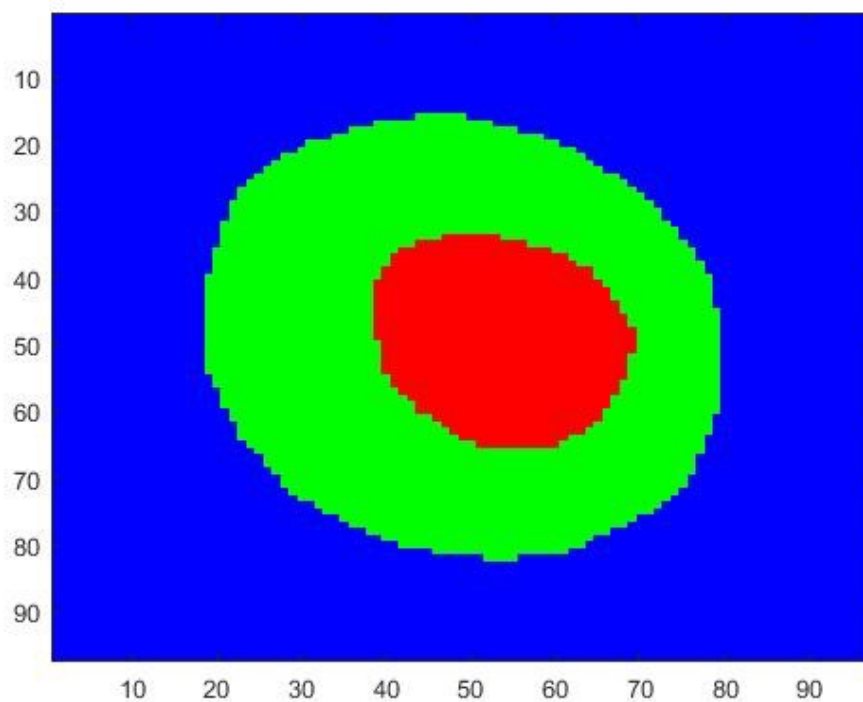




Классификация ( $SPREAD = 0.3$ ):



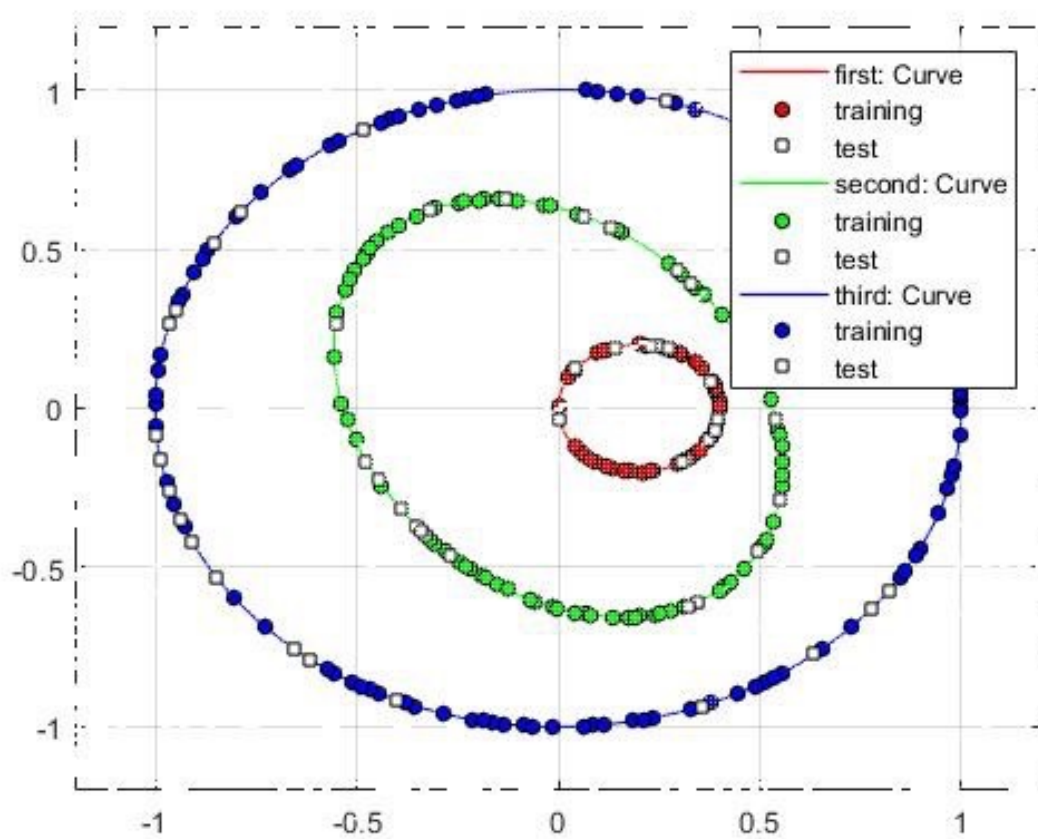
Классификация ( $SPREAD = 0.1$ ):



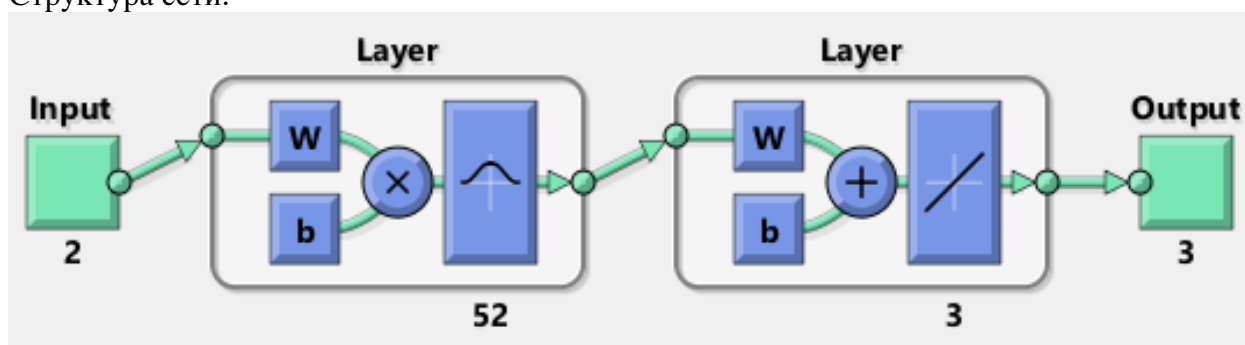
Обучающее множество: 224:224

Тестовое множество: 56:56

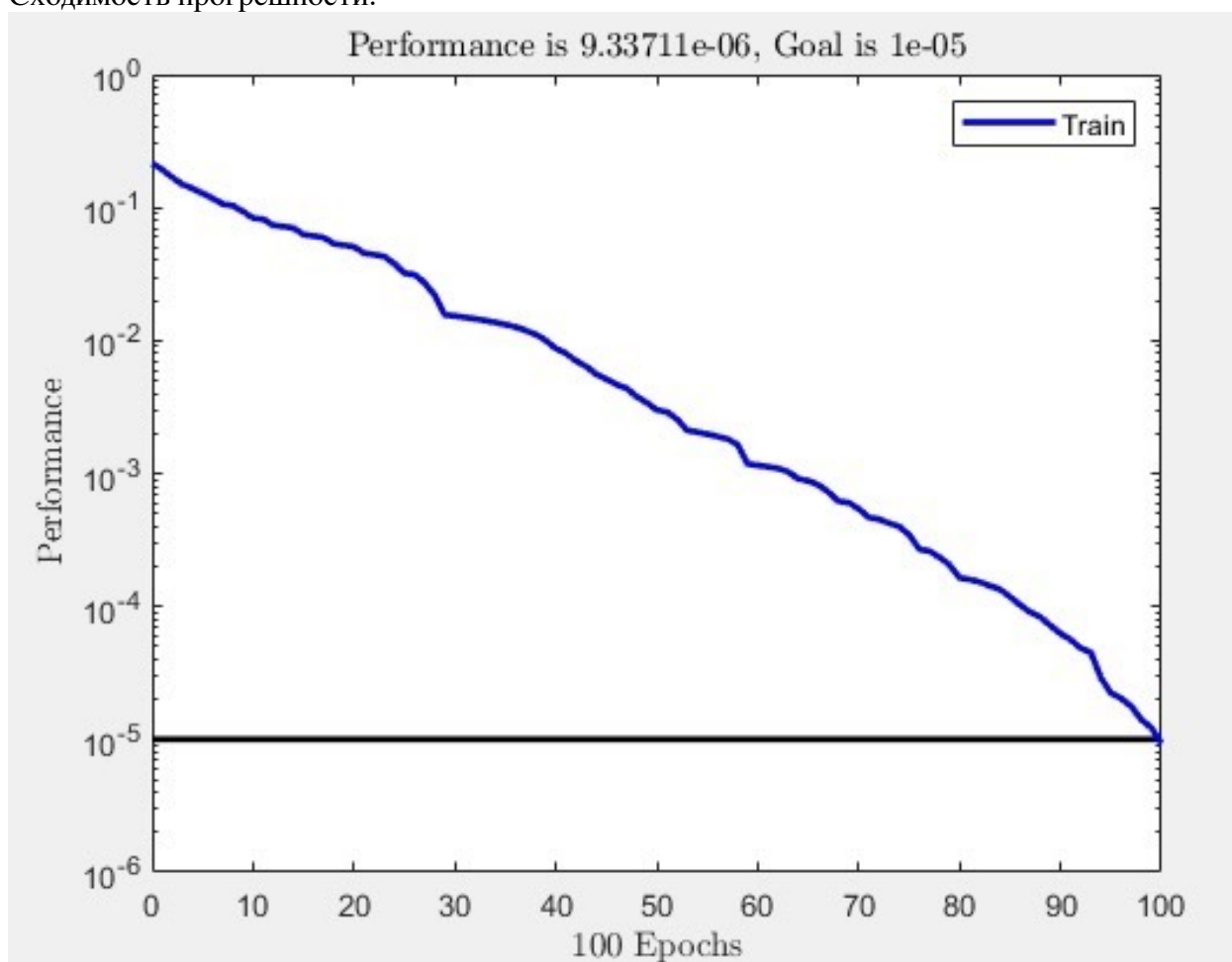
Множество точек:



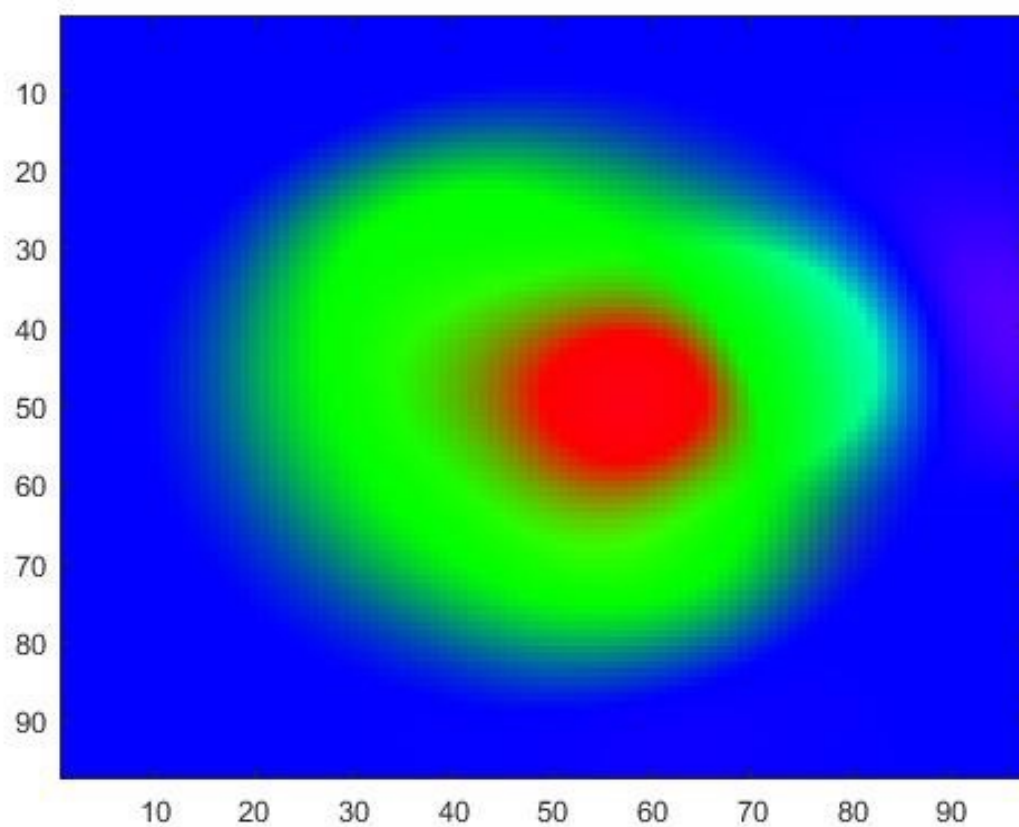
Структура сети:



Сходимость прогрешности:

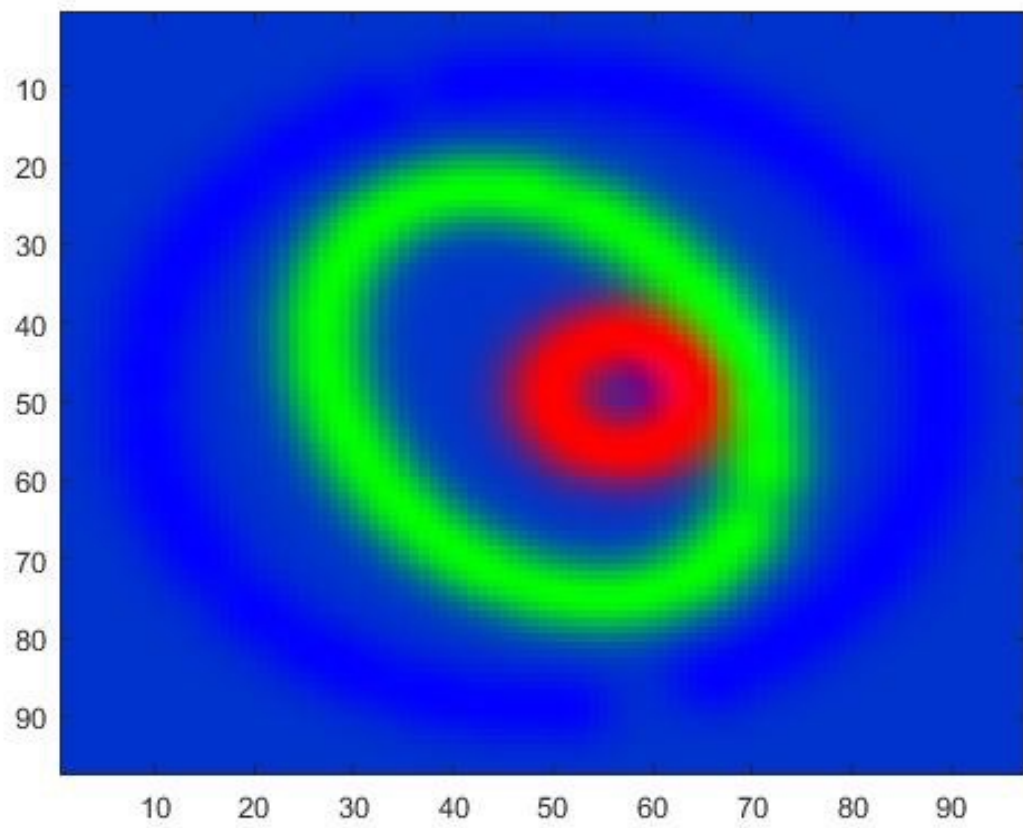


Классификация ( $SPREAD = 0.3$ ):

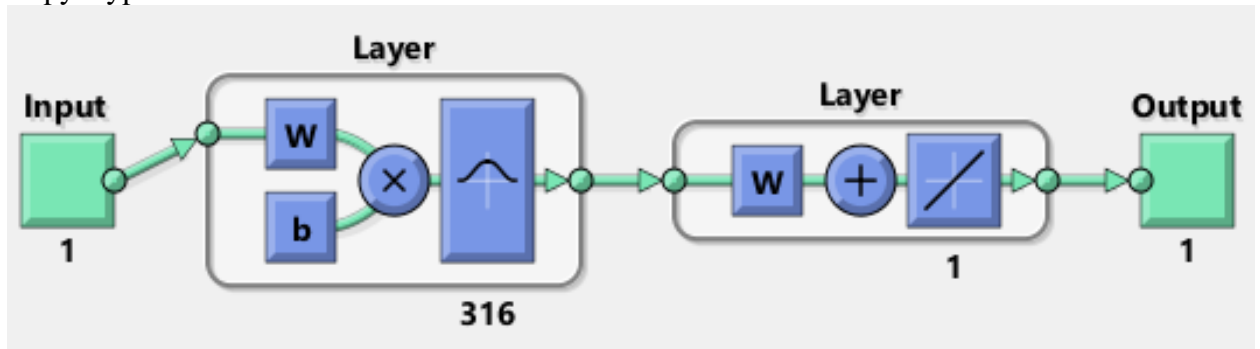




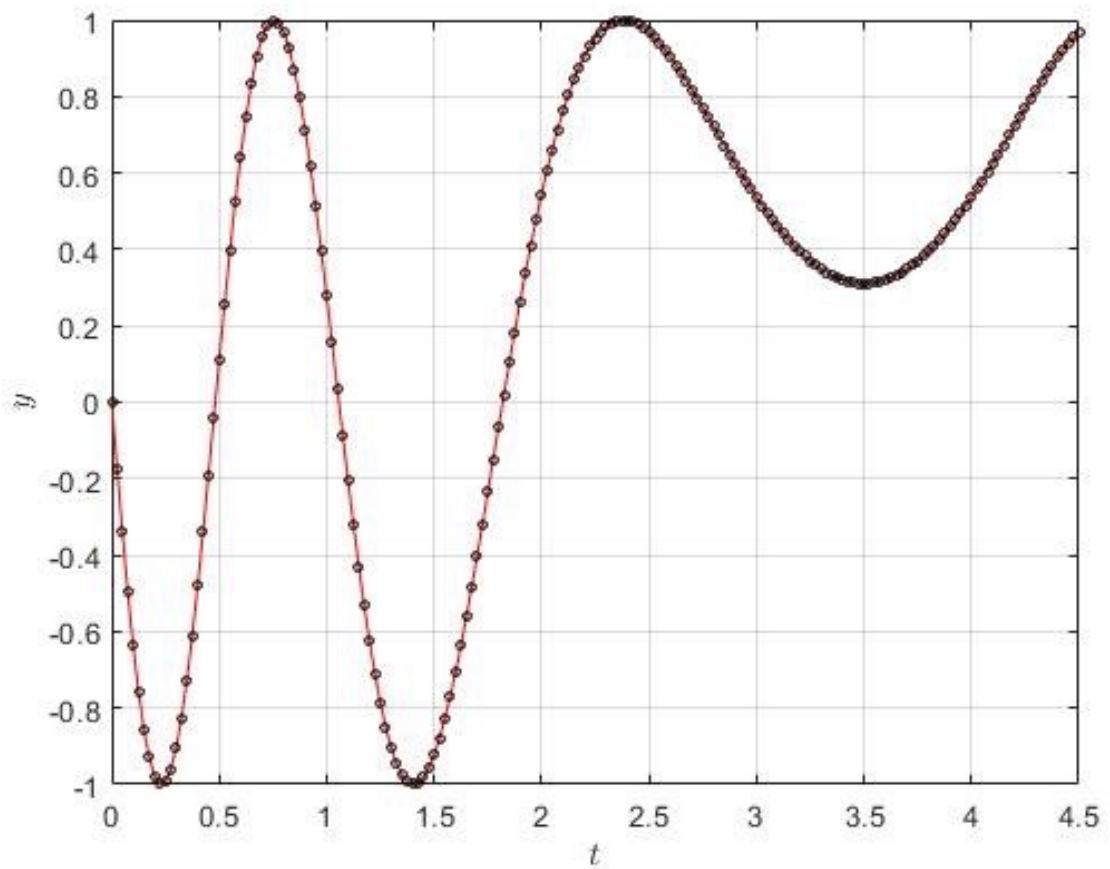
Классификация ( $SPREAD = 0.1$ ):



Структура сети:



Обучающее множество:



$R^2$ : 1.000000

MSE: 0.000000

RMSE: 0.000200 Относительная СК0: 0.010020% MAE: 0.000082

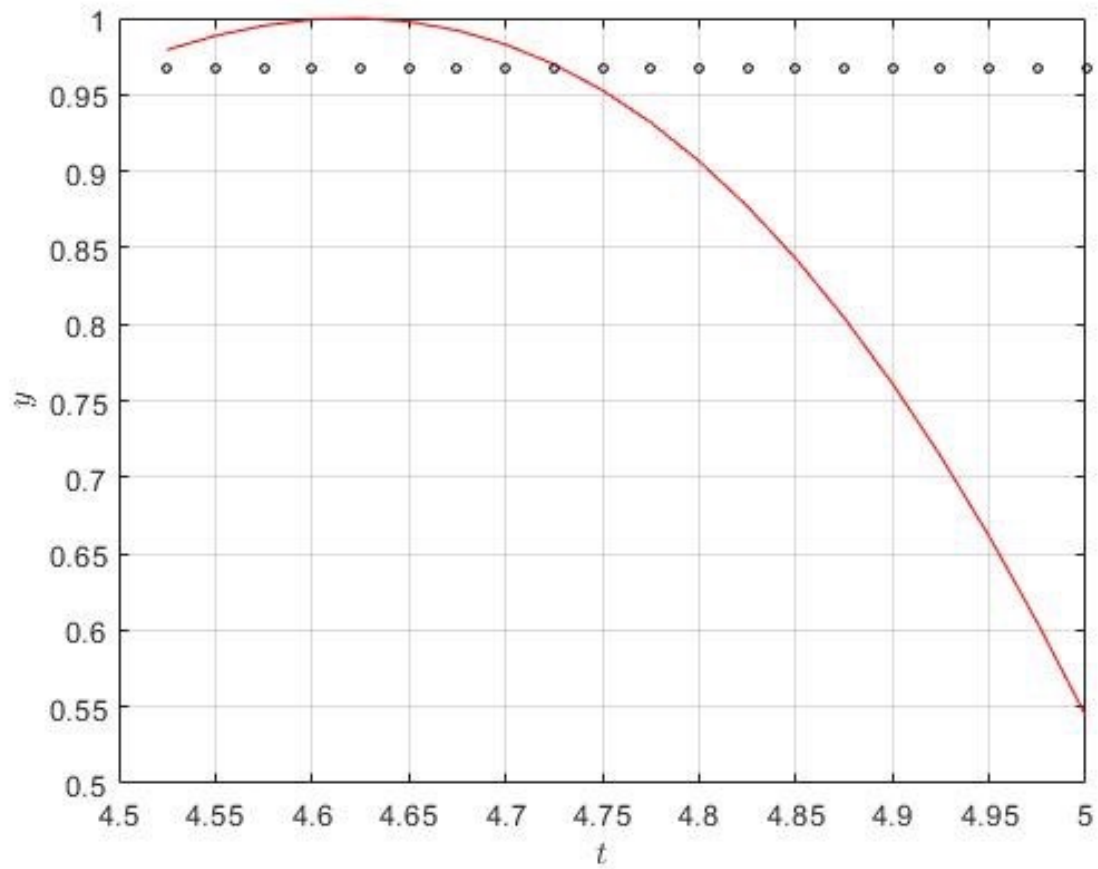
min abs err: 0.000000

max abs err: 0.002250

MAPE: 0.564164

Доля с ошибкой менее 5%: 99.447514% Доля с ошибкой от 5% до 10%: 0.000000% Доля с ошибкой от 10% до 20%: 0.000000% Доля с ошибкой от 20% до 30%: 0.000000% Доля с ошибкой более 30%: 0.552486%

Тестовое множество:



$R^2$ : -634059056116673.375000

MSE: 0.028666

RMSE: 0.169311

Относительная СК0: 548791933.847422% MAE: 0.111950

min abs err: 0.002038

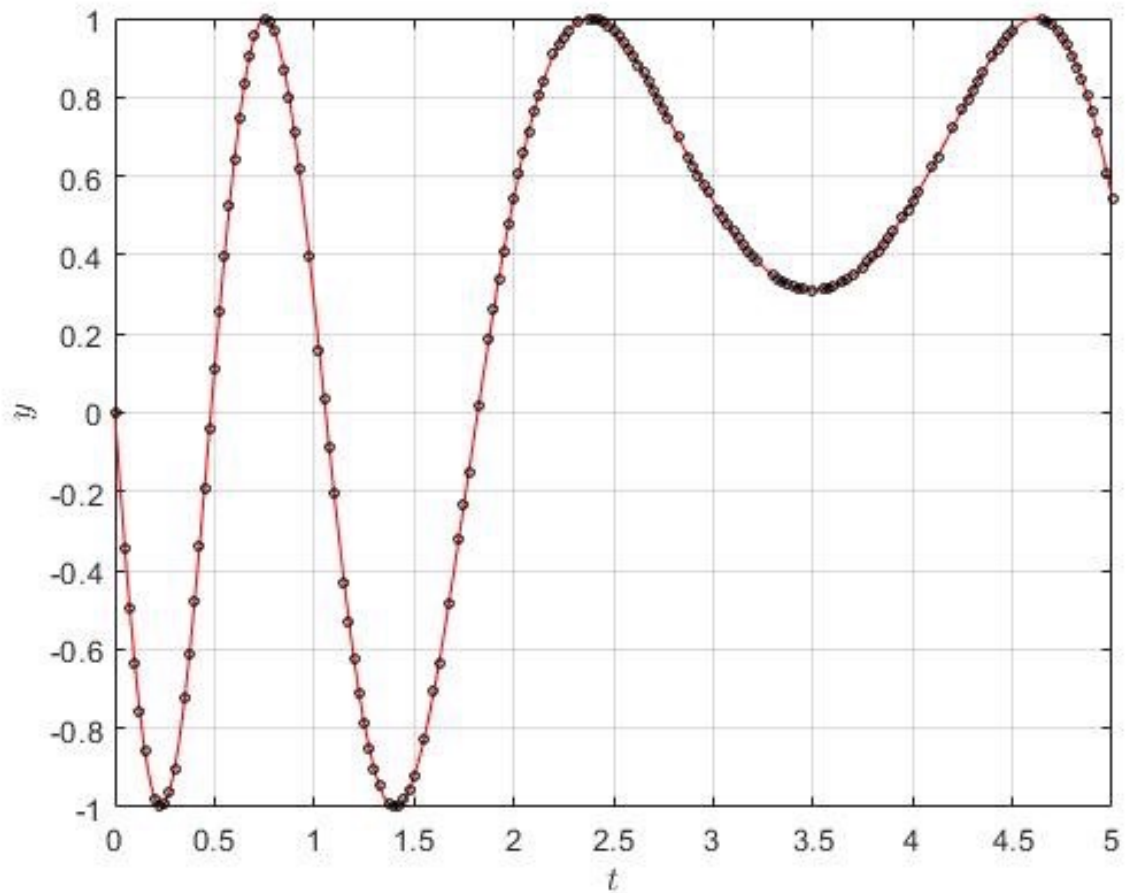
max abs err: 0.423787

MAPE: 11.567386

Доля с ошибкой менее 5%: 55.000000%

Доля с ошибкой от 5% до 10%: 10.000000% Доля с ошибкой от 10% до 20%: 10.000000% Доля с ошибкой от 20% до 30%: 10.000000% Доля с ошибкой более 30%: 15.000000%

Обучающее множество:



$R^2$ : 0.999999

MSE: 0.000000

RMSE: 0.000455 Относительная СК0: 0.022743% MAE: 0.000236

Доля с ошибкой

Доля с ошибкой

Доля с ошибкой

Доля с ошибкой

min abs err: 0.000000 max abs err: 0.002250 MAPE: 0.683478

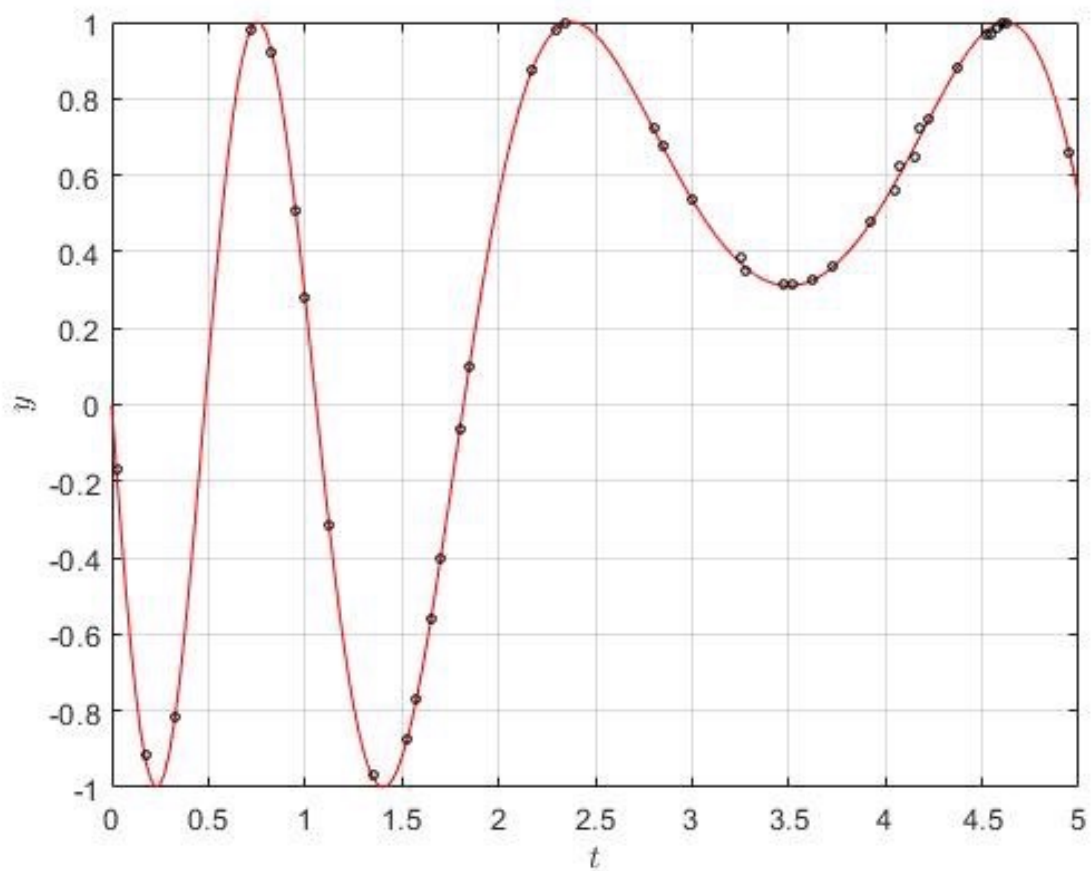
Доля с ошибкой

менее 5%: 99.378882%

от 5% до 10%: 0.000000% от 10% до 20%: 0.000000% от 20% до 30%:

0.000000% более 30%: 0.621118%

Тестовое множество:



$R^2$ : 0.999147

MSE: 0.000248

RMSE: 0.015763 Относительная СК0: 0.855423% MAE: 0.010220

min abs err: 0.000059

max abs err: 0.060845

MAPE: 1.574866

Доля с ошибкой менее 5%: 97.500000% Доля с ошибкой от 5% до 10%:

0.000000% Доля с ошибкой от 10% до 20%: 2.500000%

## Код программы

### Lab4.m

```
set(0, 'DefaultTextInterpreter', 'latex');

% Этап 1
% Задаем три эллипса
t = 0:0.025:2*pi;

a = 0.2;
b = 0.2;
alpha = 0;
x0 = 0.2;
y0 = 0;
firstCurve = [cos(alpha), -sin(alpha); sin(alpha), cos(alpha)] * [a * cos(t); b * sin(t)] + [x0 * ones(1, length(t)); y0 * ones(1, length(t))];

a = 0.7;
b = 0.5;
alpha = -pi/3;
x0 = 0;
y0 = 0;
secondCurve = [cos(alpha), -sin(alpha); sin(alpha), cos(alpha)] * [a * cos(t); b * sin(t)] + [x0 * ones(1, length(t)); y0 * ones(1, length(t))];

a = 1;
b = 1;
alpha = 0;
x0 = 0;
y0 = 0;
thirdCurve = [cos(alpha), -sin(alpha); sin(alpha), cos(alpha)] * [a * cos(t); b * sin(t)] + [x0 * ones(1, length(t)); y0 * ones(1, length(t))];

% Показываем эти эллипсы
plot(firstCurve(1, :), firstCurve(2, :), 'r', secondCurve(1, :), secondCurve(2, :), 'g', thirdCurve(1, :), thirdCurve(2, :), 'b');
legend('First', 'Second', 'Third');
axis([-1.2 1.2 -1.2 1.2]);
grid on;

%% Создаем множество точек и разделяем его
% Генерируем точки, принадлежащие эллипсам
firstDots = firstCurve(:, randperm(end, 60));
secondDots = secondCurve(:, randperm(end, 100));
thirdDots = thirdCurve(:, randperm(end, 120));

% Делим точки
[firstTraining, firstControl, firstTest] = dividerand(firstDots, 0.8, 0.0, 0.2);
[secondTraining, secondControl, secondTest] = dividerand(secondDots, 0.8, 0.0, 0.2);
[thirdTraining, thirdControl, thirdTest] = dividerand(thirdDots, 0.8, 0.0, 0.2);

trainingSetSize = length(firstTraining) + length(secondTraining) + length(thirdTraining);
n_val = length(firstControl) + length(secondControl) + length(thirdControl);
```

```

testSetSize = length(firstTest) + length(secondTest) + length(thirdTest);

% Демонстрация

p = plot(firstCurve(1, :), firstCurve(2, :), '-r', firstTraining(1, :), firstTraining(2, :), 'or', firstControl(1, :), firstControl(2, :), 'rV', firstTest(1, :), firstTest(2, :), 'rs', ...
        secondCurve(1, :), secondCurve(2, :), '-g', secondTraining(1, :), secondTraining(2, :), 'og', secondControl(1, :), secondControl(2, :), 'gV', secondTest(1, :), secondTest(2, :), 'gs', ...
        thirdCurve(1, :), thirdCurve(2, :), '-b', thirdTraining(1, :), thirdTraining(2, :), 'ob', thirdControl(1, :), thirdControl(2, :), 'bV', thirdTest(1, :), thirdTest(2, :), 'bs');

mSize = 5;
lWidth = 1;
edgeColor = 'black';
faceColor = 'white';

p(1).LineWidth = lWidth;
p(2).MarkerEdgeColor = edgeColor;
p(2).MarkerFaceColor = 'r';
p(2).MarkerSize = mSize;
p(3).MarkerEdgeColor = edgeColor;
p(3).MarkerFaceColor = faceColor;
p(3).MarkerSize = mSize;

p(4).LineWidth = lWidth;
p(5).MarkerEdgeColor = edgeColor;
p(5).MarkerFaceColor = 'g';
p(5).MarkerSize = mSize;
p(6).MarkerEdgeColor = edgeColor;
p(6).MarkerFaceColor = faceColor;
p(6).MarkerSize = mSize;

p(7).LineWidth = lWidth;
p(8).MarkerEdgeColor = edgeColor;
p(8).MarkerFaceColor = 'b';
p(8).MarkerSize = mSize;
p(9).MarkerEdgeColor = edgeColor;
p(9).MarkerFaceColor = faceColor;
p(9).MarkerSize = mSize;

axis([-1.2 1.2 -1.2 1.2]);
legend('first: Curve', 'training', 'test', ...
      'second: Curve', 'training', 'test', ...
      'third: Curve', 'training', 'test');
grid on;

%% Этапы 2 и 3
% Готовим данные для обучения
xTrainig = [firstTraining secondTraining thirdTraining];
yTrainig = [1 * ones(1, length(firstTraining)) 2 * ones(1, length(secondTraining)) 3 * ones(1, length(thirdTraining))];

```

```

xTest = [firstTest secondTest thirdTest];
yTest = [1 * ones(1, length(firstTest)) 2 * ones(1, length(secondTest)) 3 * ones(1,
length(thirdTest))];

% Создаем и обучаем сеть
SPREAD = 0.3;
network = newpnn(xTrainig, ind2vec(yTrainig), SPREAD);

% Смотрим насколько хорошо сеть обучилась
correctTrainig = sum(vec2ind(sim(network, xTrainig)) == yTrainig);
correctTest = sum(vec2ind(sim(network, xTest)) == yTest);
fprintf('Обучающее множество: %d:%d\nТестовое множество: %d:%d\n', correctTrainig,
trainingSetSize, correctTest, testSetSize);

% Разделение по классам на картине
h = 0.025;
n = int32((1.2 + 1.2) / h) + 1;
x = zeros(2, n * n);

for i = 1:n
    for j = 1:n
        x(:, (i-1)*n + j) = [-1.2 + (double(i)-1)*h; 1.2 - (double(j)-1)*h];
    end
end

image(permute(reshape(sim(network, x), [3 n n]), [2 3 1]));

%% Создаем и обучаем сеть
SPREAD = 0.1;
network = newpnn(xTrainig, ind2vec(yTrainig), SPREAD);

% Смотрим насколько хорошо сеть обучилась
correctTrainig = sum(vec2ind(sim(network, xTrainig)) == yTrainig);
correctTest = sum(vec2ind(sim(network, xTest)) == yTest);
fprintf('Обучающее множество: %d:%d\nТестовое множество: %d:%d\n', correctTrainig,
trainingSetSize, correctTest, testSetSize);

% Разделение по классам на картине
h = 0.025;
n = int32((1.2 + 1.2) / h) + 1;
x = zeros(2, n * n);

for i = 1:n
    for j = 1:n
        x(:, (i-1)*n + j) = [-1.2 + (double(i)-1)*h; 1.2 - (double(j)-1)*h];
    end
end

image(permute(reshape(sim(network, x), [3 n n]), [2 3 1]));

%% Создаем и обучаем сеть
SPREAD = 0.3;
network = newrb(xTrainig, ind2vec(yTrainig), 1e-5, SPREAD);

```



```

% Смотрим насколько хорошо сеть обучилась
correctTrainig = sum(vec2ind(sim(network, xTrainig)) == yTrainig);
correctTest = sum(vec2ind(sim(network, xTest)) == yTest);
fprintf('Обучающее множество: %d:%d\nТестовое множество: %d:%d\n', correctTrainig,
trainingSetSize, correctTest, testSetSize);

% Разделение по классам на картине
h = 0.025;
n = int32((1.2 + 1.2) / h) + 1;
x = zeros(2, n * n);

for i = 1:n
    for j = 1:n
        x(:, (i-1)*n + j) = [-1.2 + (double(i)-1)*h; 1.2 - (double(j)-1)*h];
    end
end

image(permute(reshape(sim(network, x), [3 n n]), [2 3 1]));

%% Создаем и обучаем сеть
SPREAD = 0.1;
network = newrb(xTrainig, ind2vec(yTrainig), 1e-5, SPREAD);

%% Смотрим насколько хорошо сеть обучилась
correctTrainig = sum(vec2ind(sim(network, xTrainig)) == yTrainig);
correctTest = sum(vec2ind(sim(network, xTest)) == yTest);
fprintf('Обучающее множество: %d:%d\nТестовое множество: %d:%d\n', correctTrainig,
trainingSetSize, correctTest, testSetSize);

% Разделение по классам на картине
h = 0.025;
n = int32((1.2 + 1.2) / h) + 1;
x = zeros(2, n * n);

for i = 1:n
    for j = 1:n
        x(:, (i-1)*n + j) = [-1.2 + (double(i)-1)*h; 1.2 - (double(j)-1)*h];
    end
end

image(permute(reshape(sim(network, x), [3 n n]), [2 3 1]));

%% Этап 2 и 3
% Аппроксимация функции
f = a(t) sin(t.^2 - 7 * t);
t = 0:0.025:5;

X = t;
y = f(t);

% Оставляем с конца 10%
trainingSetSize = ceil(length(X) * 0.9);
xTrainig = X(1:trainingSetSize);
yTrainig = y(1:trainingSetSize);

```

```

xTest = X(trainingSetSize+1:end);
yTest = y(trainingSetSize+1:end);

% Создаем и обучаем нейросеть
network = newgrnn(xTrainig, yTrainig, 0.01);

%% Результаты обучения на обучающем подмножестве
disp(accuracy(sim(network, xTrainig), yTrainig));
p = plot(xTrainig, yTrainig, xTrainig, sim(network, xTrainig), 'o');
p(1).Color = [1 0 0];
p(2).MarkerSize = 3;
p(2).Color = [0 0 0];
xlabel('$t$');
ylabel('$y$');
grid on;

%% Результаты обучения на тестовом подмножестве
disp(accuracy(sim(network, xTest), yTest));
p = plot(xTest, yTest, xTest, sim(network, xTest), 'o');
p(1).Color = [1 0 0];
p(2).MarkerSize = 3;
p(2).Color = [0 0 0];
xlabel('$t$');
ylabel('$y$');
grid on;

%% Делим подмножества в заданном соотношении
[tariningSplit, x, testSplit] = dividerand(1:length(X), 0.8, 0.0, 0.2);
trainingSetSize = length(tariningSplit);
testSetSize = length(testSplit);
xTrainig = X(tariningSplit);
yTrainig = y(tariningSplit);
xTest = X(testSplit);
yTest = y(testSplit);
network = newgrnn(xTrainig, yTrainig, 0.01);

% Результаты обучения на обучающем подмножестве
disp(accuracy(sim(network, xTrainig), yTrainig));
p = plot(X, y, xTrainig, sim(network, xTrainig), 'o');
p(1).Color = [1 0 0];
p(2).MarkerSize = 3;
p(2).Color = [0 0 0];
xlabel('$t$');
ylabel('$y$');
grid on;

%% Результаты обучения на тестовом подмножестве
disp(accuracy(sim(network, xTest), yTest));
p = plot(X, y, xTest, sim(network, xTest), 'o');
p(1).Color = [1 0 0];
p(2).MarkerSize = 3;
p(2).Color = [0 0 0];
xlabel('$t$');
ylabel('$y$');

```

grid on;

### dataForTable.m

```
function res = dataForTable(y, yp)
    R2 = 1 - sum((y - yp) .^ 2)/sum((y - mean(y)) .^ 2);
    MSE = mse(y - yp);
    RMSE = sqrt(MSE);
    CK0 = RMSE / (max(y) - min(y)) * 100;
    MAE = mae(y - yp);
    MinAbsErr = min(abs(y - yp));
    MaxAbsErr = max(abs(y - yp));
    MAPE = mean(abs((y - yp) ./ y)) * 100;
    errors = abs((y - yp) ./ y) * 100;
    Under5PercentPortion = sum(errors < 5) / length(y) * 100;
    Under10PercentPortion = sum(5 <= errors & errors < 10) / length(y) * 100;
    Under20PercentPortion = sum(10 <= errors & errors < 20) / length(y) * 100;
    Under30PercentPortion = sum(20 <= errors & errors < 30) / length(y) * 100;
    Over30PercentPortion = sum(errors >= 30) / length(y) * 100;

    res = sprintf(['R^2: %f\n' ...
        'MSE: %f\n' ...
        'RMSE: %f\n' ...
        'Относительная CK0: %f%%\n' ...
        'MAE: %f\n' ...
        'min abs err: %f\n' ...
        'max abs err: %f\n' ...
        'MAPE: %f\n' ...
        'Доля с ошибкой менее 5%%: %f%%\n' ...
        'Доля с ошибкой от 5%% до 10%%: %f%%\n' ...
        'Доля с ошибкой от 10%% до 20%%: %f%%\n' ...
        'Доля с ошибкой от 20%% до 30%%: %f%%\n' ...
        'Доля с ошибкой более 30%%: %f%%\n'], ...
        R2, MSE, RMSE, CK0, MAE, MinAbsErr, MaxAbsErr, MAPE,
        Under5PercentPortion, Under10PercentPortion, Under20PercentPortion,
        Under30PercentPortion, Over30PercentPortion);

end
```

### **Вывод**

В данной работе было проведено исследование свойств некоторых видов сетей с радиальными базисными элементами. Благодаря визуализации обученной сети и вычислению количества верно классифицированных точек я имел возможность наглядно сравнить работу этих сетей и алгоритмов обучения. Также я имел возможность посмотреть на то, как ведут себя сети с радиальными базисными элементами при аппроксимации функции.

RBF (локально-рецептивные) сети в отличие от MLP сетей для любого входного вектора влиять на выходной результат сети будет ограниченное число нейронов. Таким образом эти сети быстрее переобучаются (обучение происходит не полное, а частичное). Таким образом MLP лучше чем RBF показывает себя на обучающей выборке, а на тестовой все происходит наоборот.