**Задание 1.** **Радиально-базисная сеть с нулевой ошибкой**

Создать радиальную базисную сеть с нулевой ошибкой для обучающей последовательности P = 0:3 и T = [0.0 2.0 4.1 5.9], проанализировать структурную схему построенной сети и значения параметров ее вычислительной модели, выполнив следующие действия:

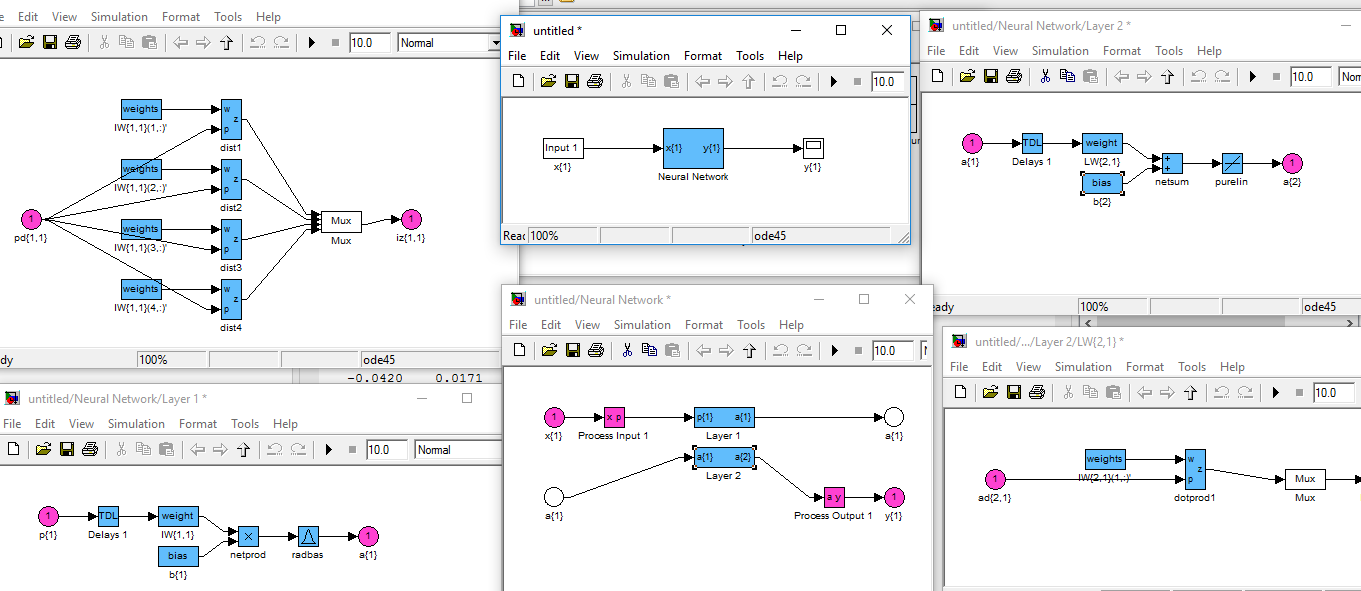
1. Создать радиальную базисную сеть с нулевой ошибкой:

P = 0:3;

T = [0.0 2.0 4.1 5.9];

net = newrbe(P, T);

2. Проанализировать структурную схему построенной сети:

gensim(net);

Видно, что входных нейронов в первом слое 4, функции активации и структура и функцию активации второго слоя.

1. Проанализировать параметры вычислительной модели сети и объяснить полученные результаты:

net.layers{1}.size % – число нейронов в первом слое;

ans =

4

net.layers{2}.size % – число нейронов во втором слое;

ans =

1

net.layers{1}.netInputFcn % – входная функция первого слоя **n = |p-w|\*b**

ans

netprod

net.layers{1}.transferFcn % – функция активации первого слоя

ans =

radbas

net.layers{2}. transferFcn % – функция активации второго слоя

ans =

purelin

net.layers{2}.netInputFcn % – входная функция второго слоя

ans =

netsum

net.inputWeights{1, 1}.weightFcn % – функции настройки начальных весов первого слоя

ans =

dist

net.IW{1, 1}, net.b{1} % – значение весов и смещения первого слоя

ans =

0

1

2

3

ans =

0.8326

0.8326

0.8326

0.8326

net.LW{2, 1}, net.b{2} % – значения весов и смещения второго слоя

ans =

0 2.4432 0.8654 6.6032

ans =

-1.2886

net.inputWeights{1, 1}, net.biases{1} % – общие характеристики первого слоя

ans =

delays: 0

initFcn: ''

learn: 1

learnFcn: ''

learnParam: ''

size: [4 1]

userdata: [1x1 struct]

weightFcn: 'dist'

weightParam: [1x1 struct]

ans =

initFcn: ''

learn: 1

learnFcn: ''

learnParam: ''

size: 4

userdata: [1x1 struct]

net.inputWeights{2, 1}, net.biases{2} % – общие характеристики второго слоя

ans =

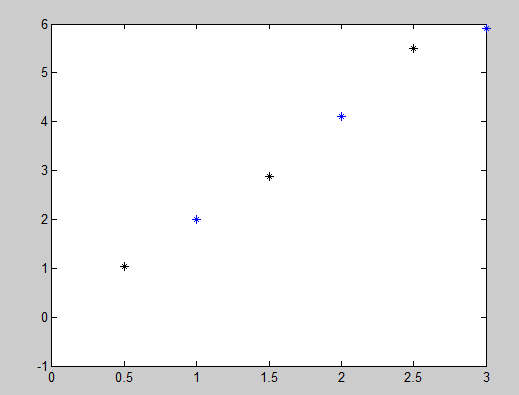
initFcn: ''

learn: 1

learnFcn: ''

learnParam: ''

size: 1

 userdata: [1x1 struct]

1. Выполнить моделирование сети и построить графики:

V=sim(net, P); % Моделирование на входных значениях

P1=0.5:2.5;

Y=sim(net, P1); % Моделирование на других значениях

plot(P,T,'+r', P,V, '\*b',P1,Y, '\*k') % Построить график

Видим, что на других значениях получились достаточно близкие результаты

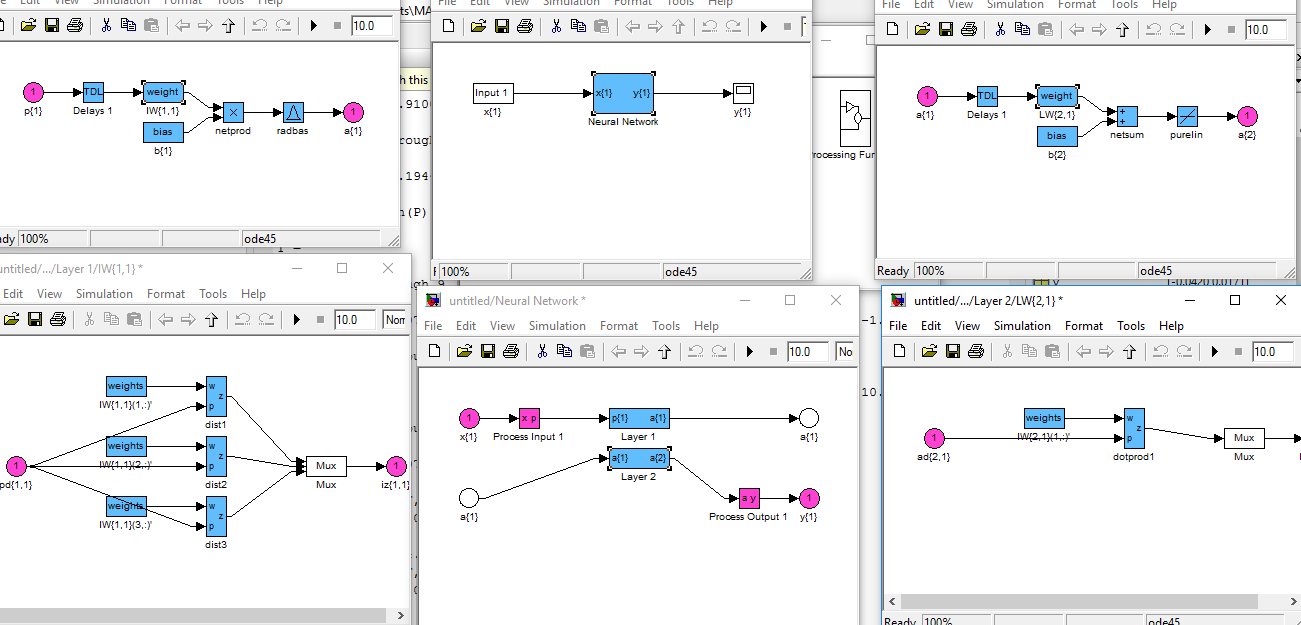
**Радиально-базисная сеть с ненулевой ошибкой**

Создание радиально-базисной сети с ненулевой ошибкой проводится аналогично, но для создания сети используется функция newrb(P, T, goal, spread), значение goal и spread выбираем, например так goal=0.01 spread = 1 (всегда меньше или равно шагу Х)

net=newrb(P, T, 0.1, 0.5);

NEWRB, neurons = 0, MSE = 4.905

gensim(net);



>> net.layers{1}.size

ans =

3

>> net.layers{2}.size

ans =

1

net.layers{1}.netInputFcn

ans =

netprod

>> net.layers{1}.transferFcn

ans =

radbas

>> net.layers{2}. transferFcn

ans =

purelin

>> net.layers{2}.netInputFcn

ans =

netsum

>> net.inputWeights{1, 1}.weightFcn

ans =

dist

>> net.IW{1, 1}, net.b{1}

ans =

3

2

1

ans =

1.6651

1.6651

1.6651

>> net.LW{2, 1}, net.b{2}

ans =

5.7841 3.7385 1.8841

ans =

-0.1178

>> net.inputWeights{1, 1}, net.biases{1}

ans =

delays: 0

initFcn: ''

learn: 1

learnFcn: ''

learnParam: ''

size: [3 1]

userdata: [1x1 struct]

weightFcn: 'dist'

weightParam: [1x1 struct]

ans =

initFcn: ''

learn: 1

learnFcn: ''

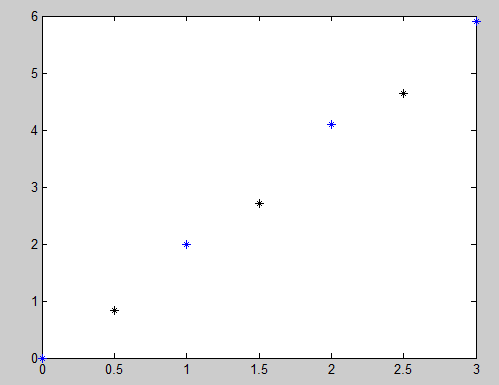
learnParam: ''

size: 3

userdata: [1x1 struct]

>> net.inputWeights{2, 1}, net.biases{2}

ans =

 initFcn: ''

learn: 1

learnFcn: ''

learnParam: ''

size: 1

userdata: [1x1 struct]

>> V=sim(net, P);

>> P1=0.5:2.5;

>> Y=sim(net, P1);

>> plot(P,T,'+r', P,V, '\*b',P1,Y, '\*k')

**Исследование радиально-базисной нейронной сети** **GRNN**

1. Аппроксимация функции y=sin(x)/x2 на промежутке [3.1, 10]

Введем значения аргумента с промежутком 0.5

x=[3.5 4 4.5 5 5.5 6 6.5 7 7.5 8 8.5 9 9.5 10];

>> y=sin(x)./(x.\*x)

Построим точки на графике

plot(x,y,'+r')

Теперь построим радиально-базисную сеть GRNN

spread = 0.7;

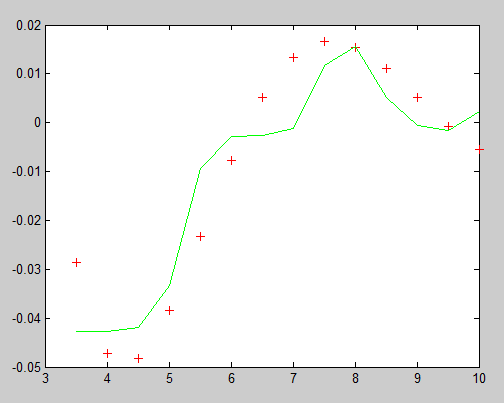
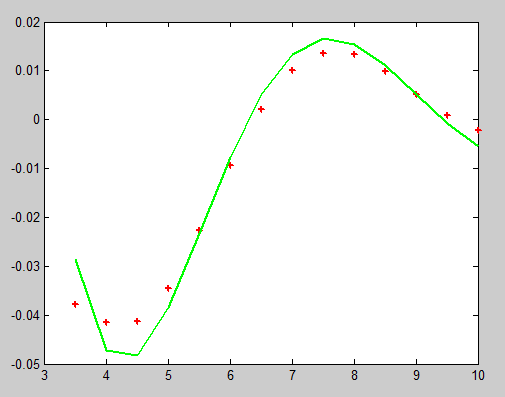
net = newgrnn(x,y,spread);

net.layers{1}.size

ans =

14

plot(x, A, '+r ',x ,y,'-g','MarkerSize',5,'LineWidth',2)

Видим, что аппроксимация при помощи многослойного персептрона хуже, сравнивая графики

Аппроксимация при помощи Аппроксимация при помощи

GRNN нейронной сети многослойного персептрона

с параметром Spread =0.7 newff([3.5, 10] ,[5,1],{'tansig' , 'purelin' },'trainrp');

newgrnn(x,y,spread)

1. Провести аппроксимацию функции двух аргументов: y= exp(x12)\*cos(-x23) на промежутке x1=[-1;1] , x2=[-1.5; 1.5]

x1 = -1.0; x2 = +1.0; y1 = -1.5; y2 = +1.5; %пределы изменения функции и аргументов

nx = 30; % Количество точек разбиения

ny =30 ;

step\_x = (x2-x1)/(nx-1); % Шаг для аргументов

step\_y = (y2-y1)/(ny-1);

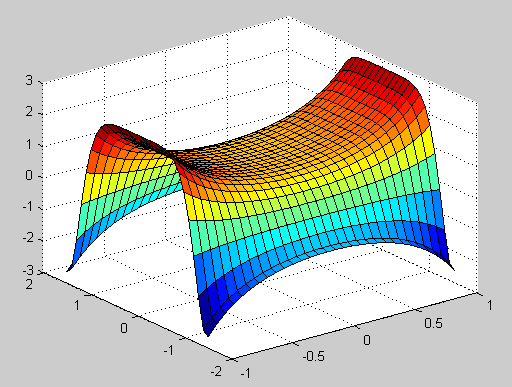
step\_min = min(step\_x,step\_y); % минимальный из шагов аргументов

[x,y] = meshgrid([x1:step\_x:x2], [y1:step\_y:y2]); % построение сетки графика

z = exp(x.^2).\*cos(-y.^3); % Вычисление значения функции

surf(x,y,z) % Построение графика функции

pause;



xx = reshape(x,1,nx\*ny); % Создание одномерного массива значения аргумента

yy = reshape(y,1,nx\*ny);

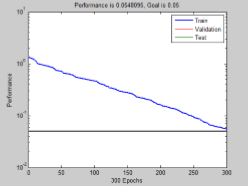
zz = exp(xx.^2).\*cos(-yy.^3); %Вычисление значений функции

p = [xx; yy]; % Создание массива входных значений

t = zz; % Создание массива целей

goal = 0.05; % Создание НС GRNN

spread = 1.0\*step\_min;

net = newrb(p,t, goal,spread);

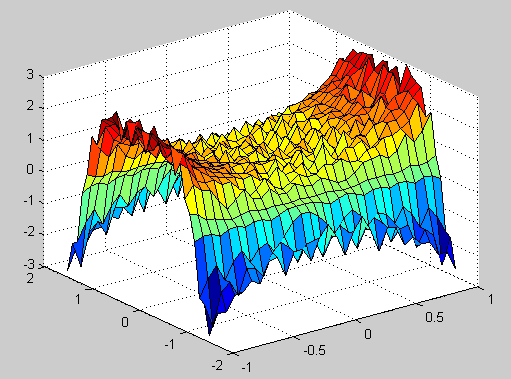
Идет процесс обучения сети с визуализацией

net.layers{1}.size % Количество нейронов входного слоя (3111)

b = sim(net,p); %Моделирование работы сети на входных значениях

[zz' b'] % Создание транспонированной матрицы значений функции и

% результата моделирования

c = reshape(b,ny,nx); % из одномерного массива

% в трехмерный для построения графика

surf(x,y,c) % построение графика

результата аппроксимации

**Исследование радиально-базисной нейронной сети PNN**

Разбить на 3 класса точки пространства так, чтобы точка [1, 3] принадлежала 2 классу, [0,1] принадлежала 1 классу и [5,2] принадлежала 3 классу.

Возьмем по нескольку точек, которые будут находиться в окрестности данных точек и, следовательно принадлежать к выделенным классам, например так:

[0 0;1 1;0 3;1 4;3 1;4 1;4 3]

На 3 класса: [1 1 2 2 3 3 3]

P = [0 0;1 1;0 3;1 4;3 1;4 1;4 3]'; % Транспонируем матрицу входных векторов

Tc = [1 1 2 2 3 3 3]; % Создадим матрицу выходов

T = ind2vec(Tc) % Создадим матрицу связности

T =

(1,1) 1

(1,2) 1

(2,3) 1

(2,4) 1

(3,5) 1

(3,6) 1

(3,7) 1

>> T = full(T) % Полная матрица связности

T =

1 1 0 0 0 0 0

0 0 1 1 0 0 0

0 0 0 0 1 1 1

>> net = newpnn(P,T); % Создаем НС типа PNN, на входе - координаты, на выходе матрица

% связности

>> net.layers{1}.size % Количество нейронов в первом слое

ans =

7

>> Y = sim(net,P); % Моделируем работу сети

Yc = vec2ind(Y) % Из матрицы связности формируем массив выходных значений

Yc =

1 1 2 2 3 3 3 % Вот и разбили по классам

>> p = [1 3; 0 1; 5 2]'; % Берем еще 3 точки и подаем на вход сети

>> a = sim(net,p);

ac = vec2ind(a)

ac =

2 1 3 % Сеть классифицировала точки

Для визуализации классификации воспользуемся таким кодом

clf reset, drawnow

drawnow

p1 = 0:.05:5; % Разбиваем плоскость на точки с шагом 0.05

p2 = p1;

[P1,P2]=meshgrid(p1,p2); % Наносим сетку

pp = [P1(:) P2(:)]; % Формируем массив из этих координат

aa = sim(net,pp'); % Подаем на вход сети в транспонированном виде

aa = full(aa); % Формируем матрицу связности

m = mesh(P1,P2,reshape(aa(1,:),length(p1),length(p2))); % Строим область для первой строки

% матрицы связности

set(m,'facecolor',[0.75 0.75 0.75],'linestyle','none'); % Закрашиваем область

hold on

view(3) % Выводим на график

m = mesh(P1,P2,reshape(aa(2,:),length(p1),length(p2))); % Для второй строки

set(m,'facecolor',[0 1 0.5],'linestyle','none');,

m = mesh(P1,P2,reshape(aa(3,:),length(p1),length(p2))); % Для третьей

set(m,'facecolor',[0 1 1],'linestyle','none');

plot3(P(1,:),P(2,:),ones(size(P,2))+0.1,'.','markersize',30) % Наносим точки из обучающей

% последовательности

plot3(p(1,:),p(2,:),1.1\*ones(size(p,2)),'\*','markersize',20, 'color',[1 0 0]) % И дополнительные точки

view(2)

