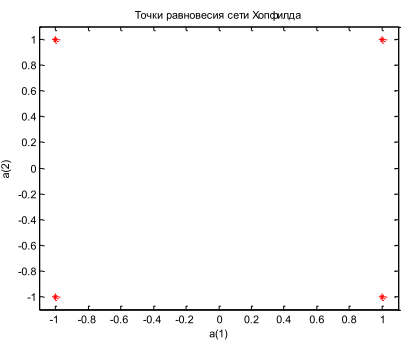
Примеры и методические указания:

**Задание 1.** Создание нейронной сети Хопфилда и исследование точек равновесия

1. Создадим нейронную сеть Хопфилда с четырьмя нейронами и определим 4 точки равновесия в виде следующего массива целевых векторов [4]

T = [1 -1; -1 1; 1 1; -1 -1]';

На рисунке показаны эти 4 точки равновесия на плоскости состояний сети Хопфилда

plot(T(1,:), T(2,:),'\*r')

axis([-1.1 1.1 -1.1 1.1])

title('Точки равновесия сети Хопфилда')

xlabel('a(1)'), ylabel('a(2)')

1. Рассчитаем веса и смещения модифицированной сети Хопфилда, используя М-функцию **newhop**

**net = newhop(T);**

**W = net.LW{1,1}**

**b = net.b{1,1}**

**W = B =**

**1.1618 0 1.0e-016 \* 0**

**0 1.1618 -0.1797**

1. Проверим, принадлежат ли вершины квадрата сети Хопфилда:

**Ai = T;**

**[Y,Pf,Af] = sim(net,4,[],Ai)**

**Y =**

**1 -1 1 -1**

**-1 1 1 -1**

**Pf = 258**

**[]**

**Af =**

**1 -1 1 -1**

**-1 1 1 -1**

Как и следовало ожидать, выходы сети равны целевым векторам.

*Синтаксис:*

[net,tr,Y,E,Pf,Af] = train(NET,P,T,Pi,Ai,VV,TV)

*Описание:*

Тренирует сеть NET в соответствии с NET.trainFcn и NET.trainParam.

TRAIN(NET,P,T,Pi,Ai) в качестве входных параметров использует:

* NET – сеть;
* P – входы сети;
* T – целевые значения, по умолчанию – нули;
* Pi – начальные входные задержки, по умолчанию – нули;
* Ai – начальные задержки слоев, по умолчанию – нули;
* VV – структура векторов верификации, по умолчанию = [];
* TV – структура тестовых векторов, по умолчанию = [];
* и возвращает:
* NET – новая сеть (тренированная);
* TR – результат тренировки (количество эпох и функция выполнения);
* Y – выходы сети;
* E – ошибки сети;
* Pf – окончательные входные задержки;
* Af – окончательные задержки слоев.

1. Теперь проверим поведение сети при случайных начальных условиях.

**сolor = 'rgbmy';**

**plot(T(1,:), T(2,:),'\*r', 0,0,'rh'), hold on,**

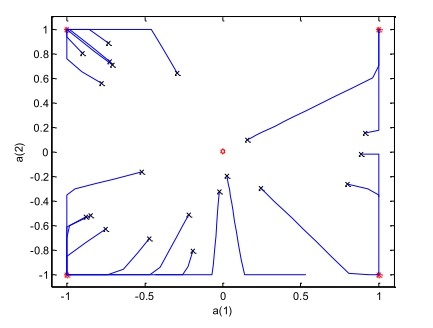
**axis([-1.1 1.1 -1.1 1.1])**

**xlabel('a(1)'), ylabel('a(2)')**

**new = newhop(T);**

**[Y,Pf,Af] = sim(net,4,[],T);**

**for i = 1:20**

**a = {rands(2,1)};**

**[Y, Pf, Af] = sim(net,{1,20},{},a);**

**record = [cell2mat(a) cell2mat(Y)];**

**start = cell2mat(a);**

**plot(start(1,1), start(2,1), 'kx', record(1,:), record(2,:) ,color(rem(i,5)+1))**

**end**

Результат представлен на рисунке

Переходный процесс заканчивается в одном из локальных устойчивых положений.

Сеть Хопфилда обеспечила переход к устойчивому положению равновесия, ближайшему к заданному входу. Сети Хопфилда разработаны для того, чтобы приводить случайные входные векторы к точкам равновесия, которые соответствуют определенным, заданным пользователем, целям.

**Задание 2.** Создать и обучить сеть Хопфилда распознавать зашумленные символы ла-

тинского алфавита, исследовать влияние величины искажений на качество рас-

познавания.

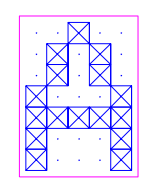
***Пример для 1-го символа А:***

**[alphabet, targets] = prprob;**

**i = 1;**

**ti = alphabet(:, i);**

**letter{i} = reshape(ti, 5, 7)';**

**letter{i}**

**ans =**

**0 0 1 0 0**

**0 1 0 1 0**

**0 1 0 1 0**

**1 0 0 0 1**

**1 1 1 1 1**

**1 0 0 0 1**

**1 0 0 0 1**

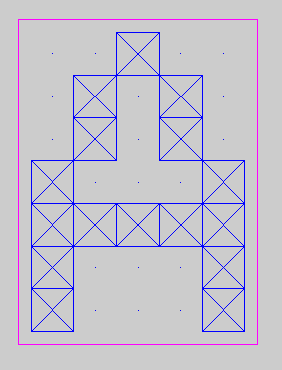
**plotchar(reshape(letter{i}', 1, 35)');**

**net = newhop(letter{i});**

**[Y,Pf,Af] = sim(net,5,[], letter{i});**

**Y**

**Y =**

** -0.0459 -0.0459 1.0000 -0.0459 -0.0459**

**-0.0229 1.0000 -0.0229 1.0000 -0.0229**

**-0.0229 1.0000 -0.0229 1.0000 -0.0229**

**1.0000 -0.0153 -0.0153 -0.0153 1.0000**

**1.0000 1.0000 1.0000 1.0000 1.0000**

**1.0000 -0.0153 -0.0153 -0.0153 1.0000**

**1.0000 -0.0153 -0.0153 -0.0153 1.0000**

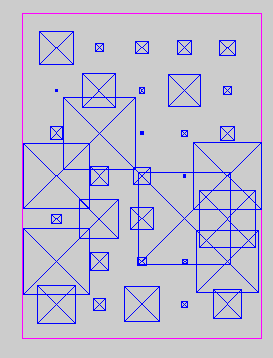
**Зашуметь букву:**

**noisyA = alphabet(:,1) + randn(35,1) \*0.4;**

**reshape(noisyA, 5, 7)'**

**ans =**

**-0.2462 0.2992 0.9230 0.3554 -0.3059**

** -0.5609 0.4310 0.1953 0.9290 -0.0784**

**0.5677 1.1166 0.0791 1.6351 -0.3218**

**1.2786 0.3340 -0.0975 0.0863 0.5337**

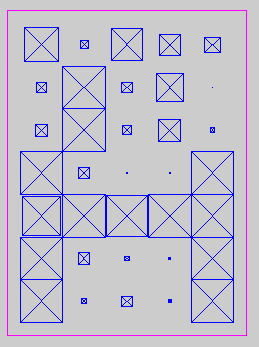
**0.5408 1.0419 1.2889 2.0342 0.7332**

**1.0749 -0.0330 -0.7732 -0.1756 0.2821**

**1.3362 -0.3552 0.0400 -0.2178 1.1214**

**plotchar(noisyA);**

**Подаем на сеть:**

** [Y,Pf,Af] = sim(net,5,[],reshape(noisyA, 5, 7)');**

**Y =**

**0.0702 0.0554 1.0000 -0.2144 0.1257**

**-0.2520 1.0000 -0.0512 1.0000 0.0029**

**-0.0377 0.9957 0.1371 1.0000 -0.0525**

**1.0000 -0.1042 -0.0992 0.0253 1.0000**

**1.0000 1.0000 1.0000 1.0000 1.0000**

**1.0000 -0.1767 0.0126 0.0507 1.0000**

**1.0000 -0.0614 -0.1562 0.0088 1.0000**

**plotchar(reshape(Y', 1, 35)');**

**Задание 3.** Теперь если не зашумлять букву А , а вызвать похожую букву и подать на сеть (например букву Н, у нее тоже есть черточка посередине) , она опознает ее как зашумленную букву А.