Министерство образования и науки Украины
Донбасская государственная машиностроительная академия

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к компьютерному практикуму

по дисциплине

«КОМПЬЮТЕРНЫЕ СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА»

(для студентов специальности 123 “Компьютерная инженерия»)

Утверждено
на заседании
кафедры АПП
Протокол № 3 от 06.11.2017

Краматорск 2017

УДК 681.142.2

Методические указания к компьютерному практикуму по дисциплине ”Компьютерные системы искусственного интеллекта” (для студентов специальности 123 “Компьютерная инженерия») / Сост. А. А. Сердюк. – Краматорск: ДГМА, 2017 – 56 с.

Содержатся сведения о методах и средствах создания, настройки, адаптации и обучения нейронных сетей для решения задач распознавания образов, аппроксимации функций, фильтрации сигналов. Приведена методика применения пакета прикладных программ Neural Network Toolbox в среде MATLAB для реализации нейронных сетей.

Составитель А. А. Сердюк, доц.,

Отв. за выпуск С. П. Сус, доц.

СОДЕРЖАНИЕ

1 СОЗДАНИЕ И ОБУЧЕНИЕ НЕЙРОННОЙ СЕТИ С ПОМОЩЬЮ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ GUI

Цель работы: освоить методику разработки и обучения нейронной сети с применением графического интерфейса пользователя (GUI) системы MATLAB.

1.1 GUI-интерфейс для пакета прикладных программ NNT

Графический интерфейс пользователя (***Graphical User Interface***) позволяет, не обращаясь к командному окну системы MATLAB, выполнить создание, обучение и моделирование, а также экспорт и импорт нейронных сетей.

Вызов GUI-интерфейса пакета прикладных программ ***Neural Network Toolbox*** возможен либо командой ***nntool*** из командной строки, либо из окна запуска приложений ***Launch Pad*** с помощью опции ***NNTool.*** После вызова появляется окно (рис. 1.1) ***Network/Data Manager*** (Управление сетью/данными).

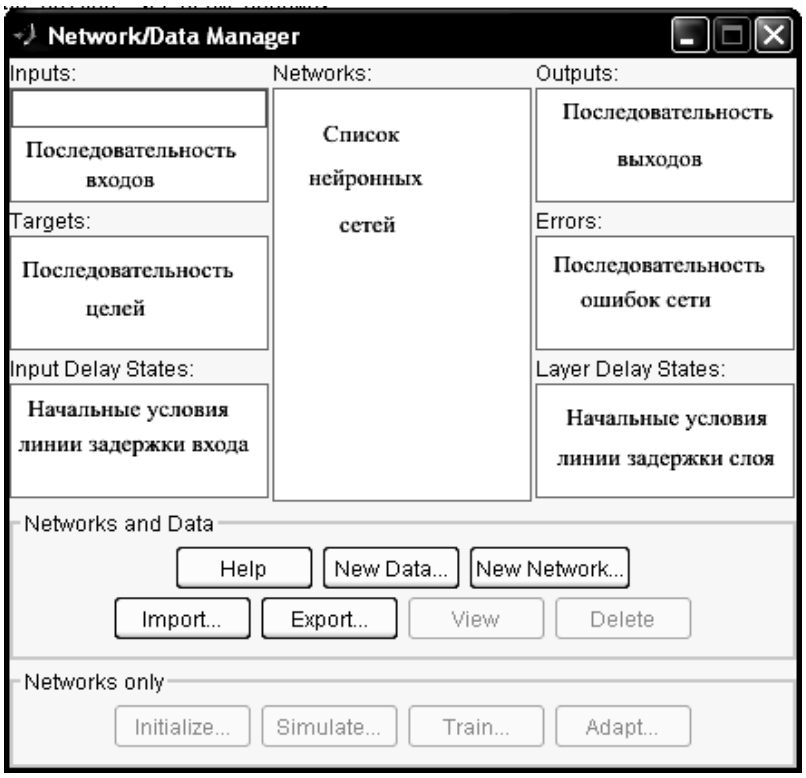


Рисунок 1.1 – Окно управления сетью и данными

Чтобы создать нейронную сеть, нужно выполнить следующие операции:

- кнопкой ***New Data*** вызвать окно форматирования данных ***Create New Data*** (рис. 1.2) и сформировать последовательность входов и целей;

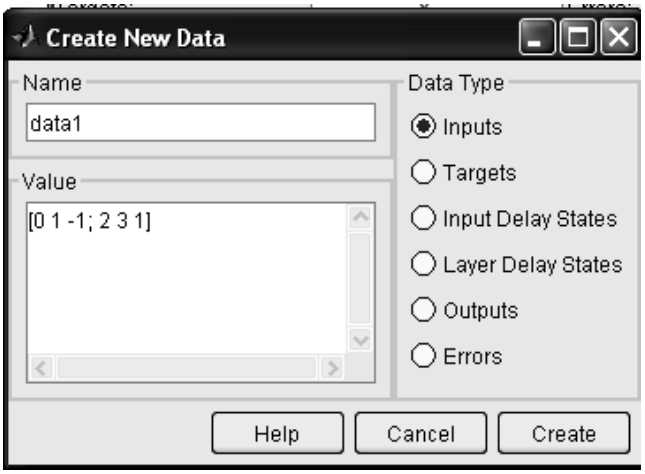


Рисунок 1.2 – Окно ввода данных

- в окне ***Create New Network*** **(**рис. 1.3), которое вызывается кнопкой ***New Network*****,** создать новую сеть;

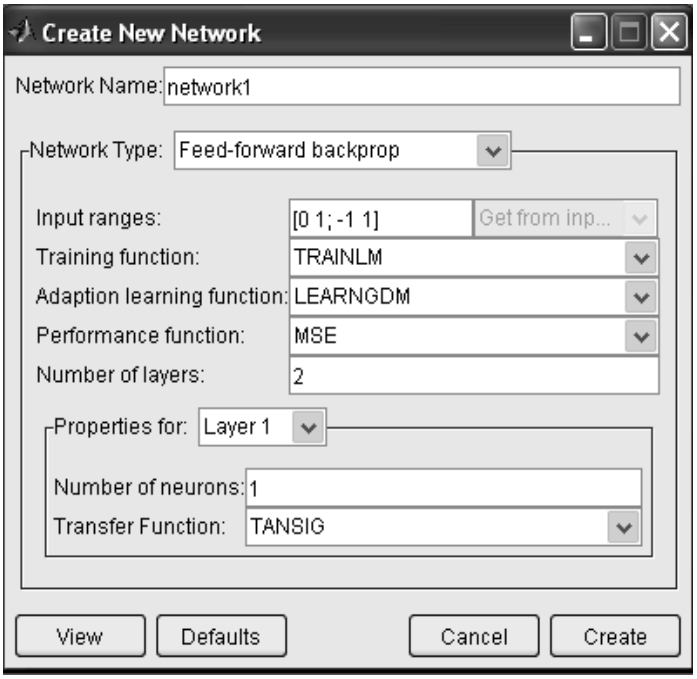


Рисунок 1.3 – Окно создания сети ***Create New Network***

- задать параметры инициализации, адаптации, обучения и моделирования сети, вызывая соответствующие окна кнопками ***Initialize, Adapt, Train, Simulate*** в окне ***Network/Data Manager***.

В окне ввода данных ***Create New Data*** можно указать один из шести типов данных:

- ***Inputs*** (входы) – последовательность значений входов;
- ***Targets*** (цели) – последовательность значений цели;
- ***Input Delay States*** (состояния линии задержки входа) – начальные условия линии задержки на входе;

- ***Layer Delay States*** (состояния линии задержки слоя) – начальные условия линии задержки в слое;
- ***Outputs*** (выходы) – последовательность значений выхода сети;
- ***Errors*** (ошибка) – разность значений целей и выходов.

Обычно пользователь задает только последовательности входа и цели.

Окно ***Create New Network*** (рис. 1.3) включает поля для задания параметров создаваемой сети. Количество и названия полей изменяются в зависимости от типа сети.

Поле **Network Type** предназначено для выбора типа сети. Типы сетей, доступных для работы с интерфейсом NNTool, приведены в таблице 1.1.

Таблица 1.1 – Список типов сетей, доступных для работы с интерфейсом NNTool

№ п/п	Тип сети	Название сети	Число слоев	Обучаемые параметры
1	Competitive	Конкурирующая сеть	1	IW[1,1], b[1]
2	Cascade forward backprop	Каскадная сеть	2	Просмотр не обеспечивается
3	Elman backprop	Сеть Элмана	2	Просмотр не обеспечивается
4	Feed-forward backprop	Сеть с прямым распространением сигнала	2	IW[1,1], b[1] LW[2,1], b[2]
5	Time delay backprop	Сеть с запаздыванием	2	IW[1,1], b[1] LW[2,1], b[2]
6	Generalized regression	Обобщенная регрессионная сеть	2	IW[1,1], b[1] LW[2,1]
7	Hopfield	Сеть Хопфилда	1	Просмотр не обеспечивается
8	Linear layer (design)	Линейный слой (создание)	1	IW[1,1], b[1]
9	Linear layer (train)	Линейный слой (обучение)	1	IW[1,1], b[1]
10	LVQ	Сеть классификации векторов	2	IW[1,1], LW[2,1]
11	Perceptron	Персептрон	1	IW[1,1], b[1]
12	Probabalistic	Вероятностная сеть	2	IW[1,1], b[1] LW[2,1]
13	Radial basis (exact fit)	Радиальная базисная сеть с нулевой ошибкой	2	IW[1,1], b[1] LW[2,1]
14	Radial basis (fewer neurons)	Радиальная базисная сеть с минимальным числом нейронов	2	IW[1,1], b[1] LW[2,1], b[2]
15	Self-organizing map	Самоорганизующаяся карта Кохонена	1	IW[1,1]

Следует учесть, что этот интерфейс позволяет создавать нейронные сети только с одним или двумя слоями. Обучаемые параметры представляют собой весовые матрицы входа **IW** (Input Weight) и выхода **LW** (Layer Weight), а также смещение **b**. Сети с двумя слоями обычно имеют последовательную структуру, когда выход первого слоя служит входом второго слоя.

Следующее поле **Input ranges** (диапазоны входа) позволяет задать допустимые границы входов.

Поле **Training function** (функция обучения) позволяет задать алгоритм обучения сети.

Остальные поля предназначены для задания следующих функций:

- *Adaption learning function* – выводит список функций настроек режима адаптации;

- *Performance function* – выводит список функций оценки качества обучения;

- *Number of layers* –задание количества слоев нейронной сети;

- *Properties for* – служит для задания свойств слоя;

- *Number of neurons* – задает количество нейронов в слое;

- *Transfer function* – задает функцию активации слоя.

Кнопками ***Import***, ***Export*** окна управления сетью ***Network/Data Manager*** можно открыть окна ***Import or Load to Network/Data Manager*** или ***Export or Save from Network/Data Manager***. Эти окна позволяют загрузить данные из рабочей области MATLAB в рабочую область GUI-интерфейса, или, наоборот, передать данные в рабочую область MATLAB.

1.2 Методика выполнения работы

Пусть, например, необходимо создать нейронную сеть в виде персептрона, для разделения векторов входа на два класса, обозначенных как 0 и 1.

Процесс создания сети будет включать следующие операции.

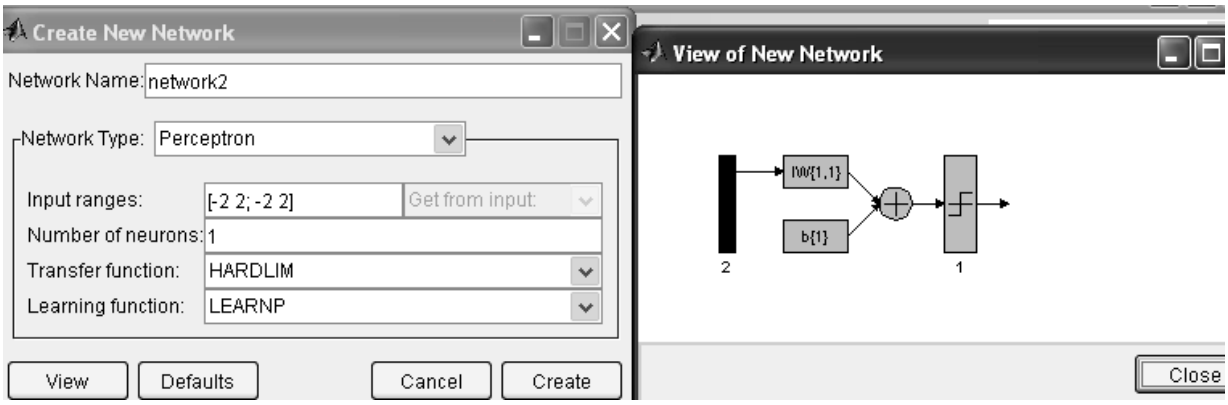
1 Зададим обучающую последовательность в виде двух массивов ячеек: массива входов P = {[2; 2] [1; 2] [-2; 2] [-1; 1] [1; -2]} и массива целей T = {0 0 1 1 1}. Массив целей задает принадлежность каждого вектора входа к определенному классу. Выполним эту операцию в рабочей области системы MATLAB.

```
>> nntool
>> P={[2;2] [1;2] [-2;2] [-1;1] [1;-2]}
P =
 Columns 1 through 4
 [2x1 double] [2x1 double] [2x1 double] [2x1 double]
  Column 5
    [2x1 double]
```

```
>> T={0 0 1 1 1}
T =
    [0]    [0]    [1]    [1]    [1]
```

2 Зададим персептрон с одним нейроном на два входа с диапазоном [-2; 2], применим функцию активации HARDLIM и правило настройки LEARNP. (рис. 1.4,а). После нажатия кнопки ***Create*** в поле ***Networks*** (окно ***Network/Data Manager***) установится наименование сети ***network1***. При выборе этого названия активизируются кнопки ***View, Delete, Initialize, Simulate, Train, Adapt***.

Кнопкой ***View*** можно вызвать структуру сети (рис. 1.4,б).



а б

а – для задания параметров сети;
б – структура сети в окне ***View of New Network***

Рисунок 1.4 – Окно ***Create New Network***

3 Импортируем из рабочей области MATLAB массив входов и массив целей (кнопка ***Import)***. В окне ***Import or Load to Network/Data Manager*** (рис. 1.5) должен быть установлен флажок ***Import from MATLAB***.

В поле ***Select a Variable*** необходимо выбрать наименование массива данных ***P***, в поле ***Import As*** установить тип массива (флажок ***Input***) и нажать ***Import***. Эту операцию следует повторить с массивом целей ***T*** (флажок ***Targets***).

4 Выполним инициализацию сети (кнопка ***Initialize*** окна ***Network/Data Manager***). При этом откроется диалоговая панель ***Network: network1*** (рис. 1.6). В поле ***Input Ranges*** необходимо установить диапазоны двух входов сети [-2 2; -2 2], затем нажать кнопку ***Set Ranges***. Далее, нажав кнопку ***Weights***, необходимо таким же образом установить веса входов ***p1*** и ***p2***.

5 Выполним адаптацию и обучение сети, вызывая окна диалога кнопками ***Adapt*** (рис. 1.7) и ***Train*** (рис. 1.8). В открывающихся окнах указываются имена входа и цели, а при выборе команды ***Adaptation Parameters*** в поле ***passes*** задается количество циклов адаптации. В данном примере достаточно трёх-пяти циклов адаптации.

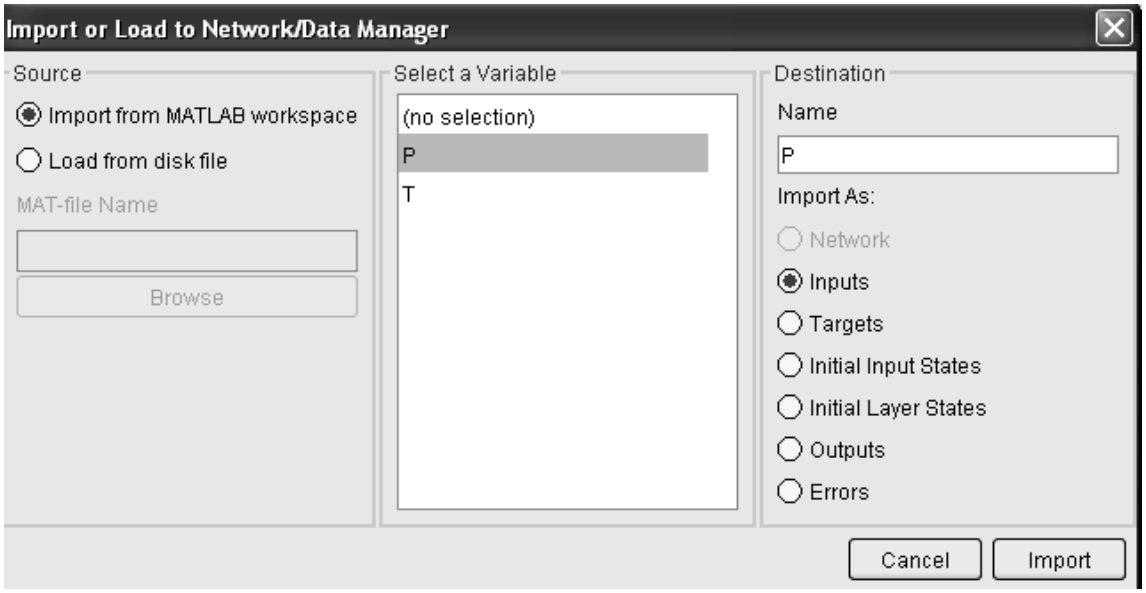


Рисунок 1.5 – Окно импорта данных из рабочей области MATLAB

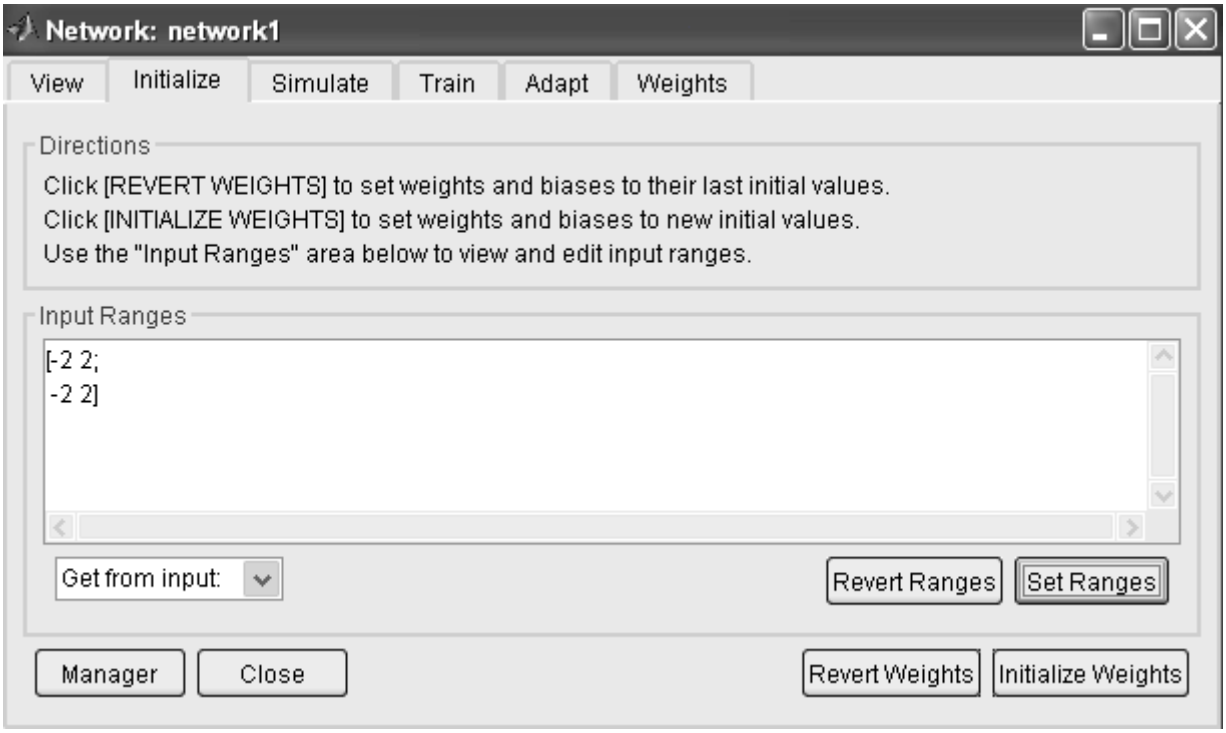


Рисунок 1.6 – Окно инициализации создаваемой сети network1

Результаты обучения можно увидеть, выбрав команду ***Weights***, по которой открывается окно с полем ***Select the weight or bias to view*** для выбора установившихся весов входов ***IW*** и смещения **b** (рис. 1.9).

Как видно из рисунка 1.9, начальная установка весов ***IW* = [1, 1]** в результате обучения изменилась: веса входов равны **[-2, -1]**.

Раскрыв закладку ***Select the weight or bias to view***, можно увидеть значение смещения: **b = [1]**.

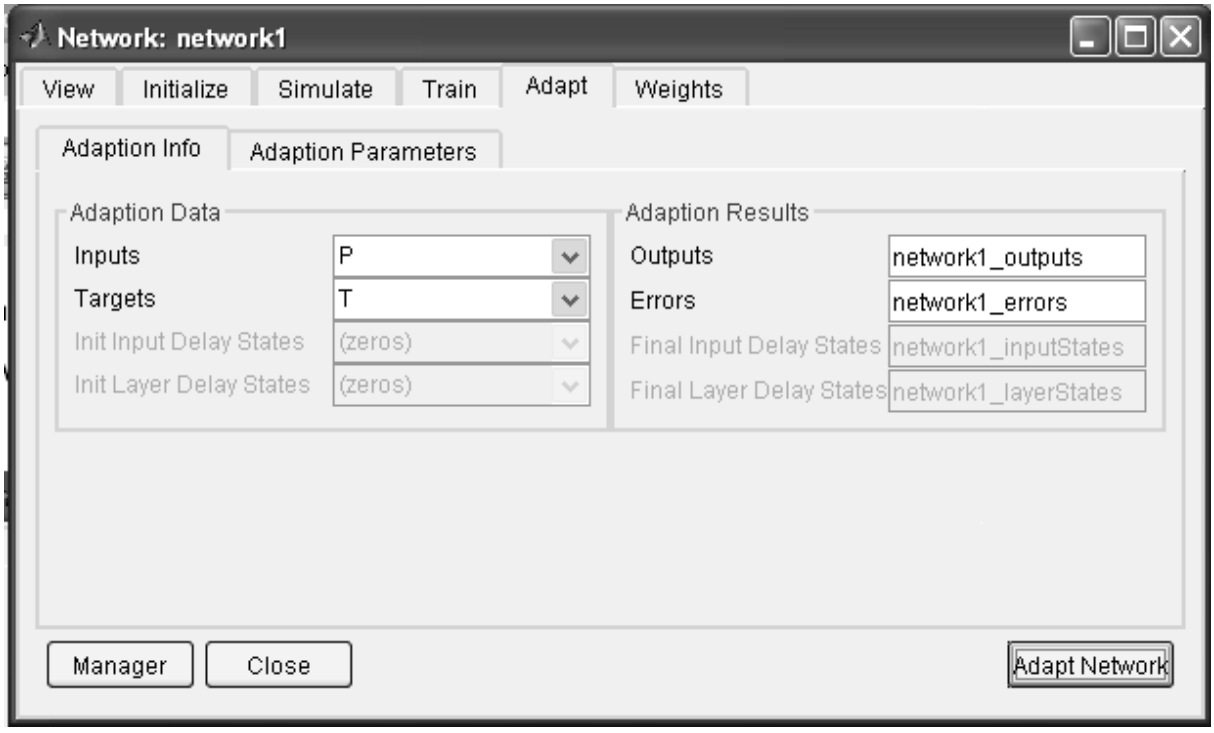


Рисунок 1.7 – Окно настройки параметров адаптации сети

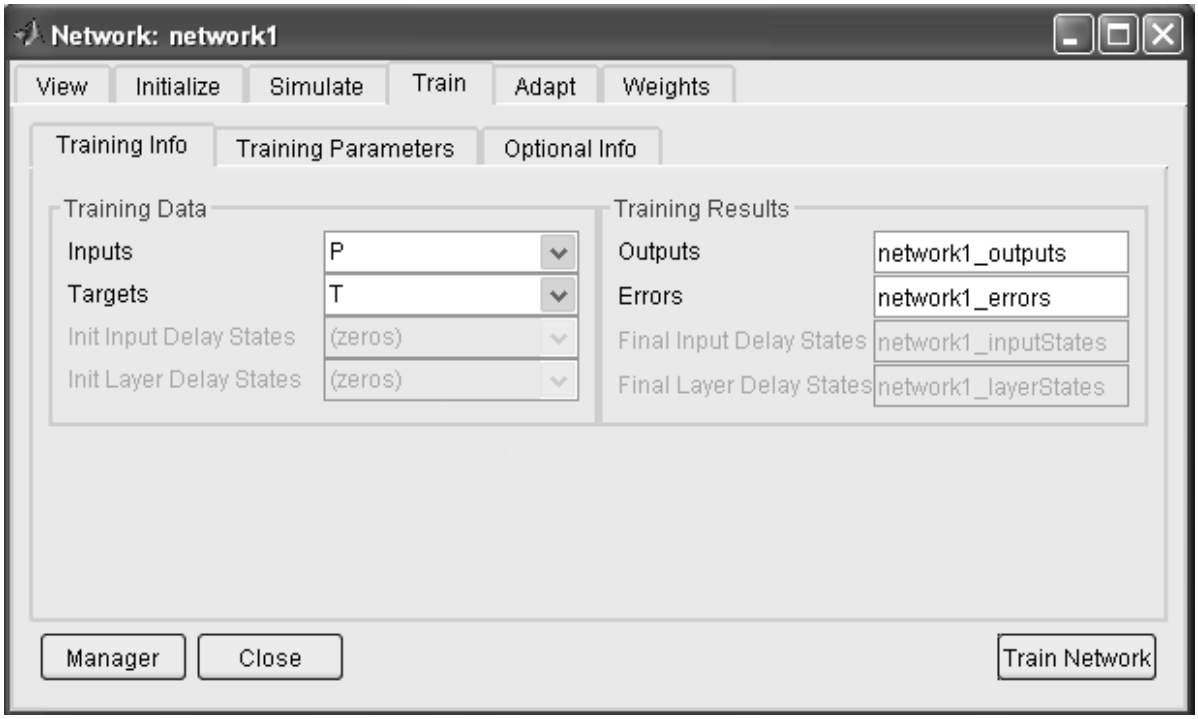


Рисунок 1.8 – Окно задания параметров обучения сети

Таким образом, линия переключения, разделяющая классы объектов в плоскости $(p_1; p_2)$, записывается следующим образом:

$$L: -2p_1 - p_2 + 1 = 0.$$

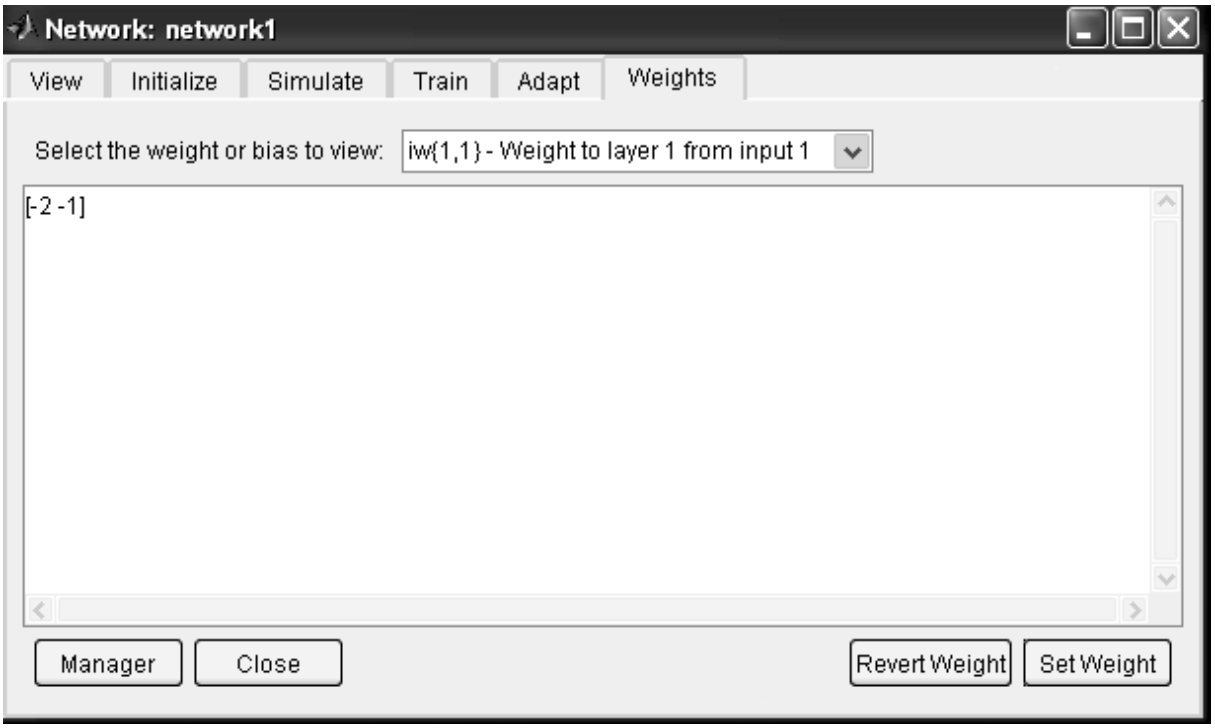


Рисунок 1.9 – Результат обучения сети `IW{1,1}=[-2 -1]`

Перейдя в окно ***Network/Data Manager***, можно просмотреть значения сигналов на выходе и ошибку сети. Для этого в поле ***Outputs*** выделим сигнал на выходе сети `network1_outputs` и выберем команду ***View***, по которой откроется окно ***Data: network1_outputs*** (рис. 1.10). Аналогичную процедуру выполним и для вывода ошибки `network1_errors`.

Для вывода графика линии переключения нужно в окне команд MATLAB исходные данные ввести в другом формате. После этого произвести настройку и адаптацию сети. Программа будет иметь следующий вид:

```
>> P=[2 1 -2 -1 1;2 2 2 1 -2];
>> T=[0 0 1 1 1];
>> net=newp([-2 2;-2 2],1);
>> net.adaptParam.passes=3;
>> net=adapt(net,P,T);
>> hold on;
>> plotpv(P,T);
>> plotpc(net.IW{1},net.b{1});
>> hold off
```

В результате будет выведен график, на котором в координатах p(1) – p(2) нанесена линия, разделяющая классы входных векторов (рис. 1.11).

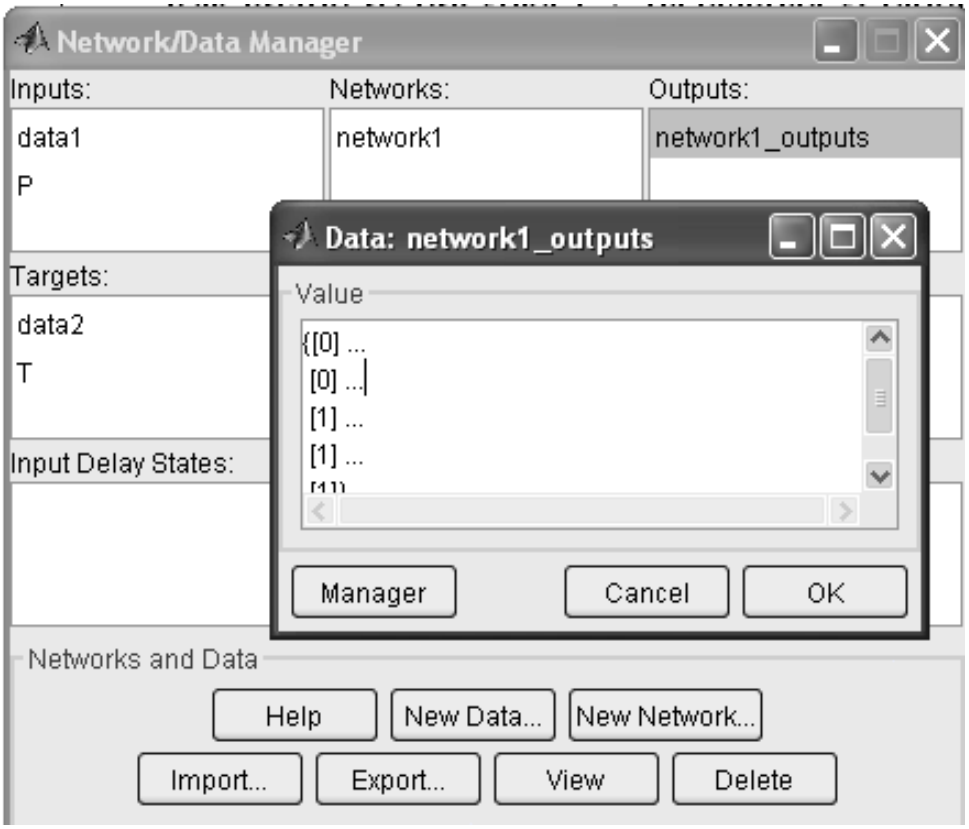


Рисунок 1.10 – Окно для просмотра выходных сигналов сети

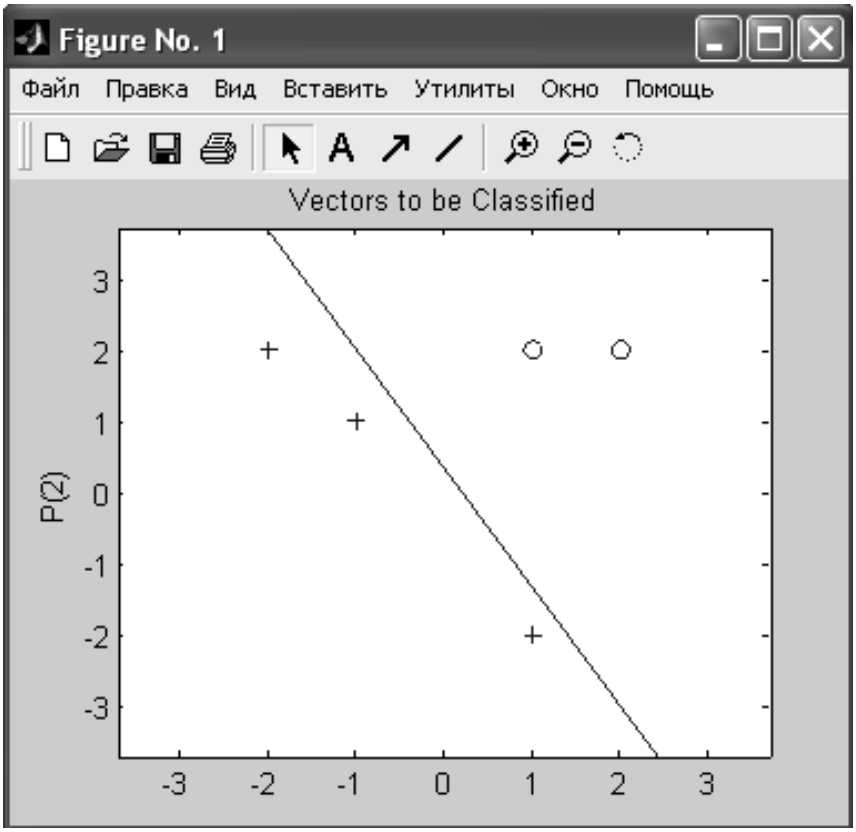


Рисунок 1.11 – График разделения классов объектов

Таблица 1.2 – Варианты индивидуальных заданий

Вар.	Входная последовательность	Диапазон	Последовательность целей
1	p_1 = [–2 –3 +5 +1 –6 +3 –5] p_2 = [0 –4 +2 +3 +1 +6 –3]	–6...+6	[0 0 1 1 0 1 0]
2	p_1 = [+2 –3 +5 +1 –6 +3 –5] p_2 = [0 –4 +2 +3 +1 –6 –3]	–5...+5	[0 1 0 0 0 1 1]
3	p_1 = [–1 +5 +4 –5 +2 –3] p_2 = [–3.5 +1.5 –5 +2.5 +1 –2]	–5...+5	[0 1 0 1 1 0]
4	p_1 = [–10 +8 +4 +12 –7 –5] p_2 = [+2 +3.5 –2 –10 +7 –6]	–12...+12	[0 1 1 1 0 0]
5	p_1 = [–1 +2 +1.5 –0.5 +1.5 +2] p_2 = [+1 +1.5 +0.5 +2 –1 +0.5]	–2...+2	[0 1 0 0 1 1]
6	p_1 = [+1 –2 +0.5 +1.5 0 –2.5] p_2 = [–3 –1 +1 –0.5 –2 +0.5]	–3...3	[0 1 1 0 0 1]
7	p_1 = [5 4 5 3 4 2] p_2 = [1 3.5 1 2 3 0.5]	0...5	[0 1 1 0 1 0]
8	p_1 = [0.5 1.5 2 1 0 1] p_2 = [1 0.5 1 2 1.5 2]	0...2	[1 0 0 1 1 0]
9	p_1 =[–0.5 –0.5 +0.3 –0.1 +0.2 –0.7] p_2 =[–0.5 +0.5 –0.3 +1.0 +0.8]	–1...+1	[1 1 0 0 0 1]
10	p_1 = [+2 +1.5 +3 +0.5 –0.5 –1] p_2 = [–1 +0.5 +2 +0.5 +2 +1]	–1...+3	[0 1 1 0 1 0]
11	p_1 = [+2 +4.5 +3 +2.5 –0.5 –1] p_2 = [–3 +0.5 +2 +1.5 +2 –4]	–4...+5	[0 1 1 0 0 0]
12	p_1 = [+2 –3 +5 +1 –6 +3] p_2 = [–1 +3.5 –4 +2 +3 –0.5]	–6...+5	[0 1 0 1 1 0]
13	p_1 = [+1 –2 +0.5 +1.5 –2.5 –0.5] p_2 = [–0.5 +0.5 –0.3 +1.0 +0.8 –2]	–3...+2	[1 0 1 1 0 0]
14	p_1 = [–10 +8 +4 +2 –7 –5] p_2 = [+2 +3.5 –2 –10 +7 –6]	–10...+10	[0 1 1 1 0 0]
15	p_1 = [0.5 1.5 2 1 0 1] p_2 = [1 0.5 1 2 1.5 2]	0...2	[0 1 1 0 0 1]
16	p_1 = [–2 –3 +5 +1 –6 +3 –5] p_2 = [0 –4 +2 +3 +1 +6 –3]	–6...+6	[1 1 0 0 1 0 1]
17	p_1 = [+3 +4 +1 +4 –1 –0.5] p_2 = [+3 +1 +0.5 –2 +2 –2]	–3...+5	[1 0 1 0 1 0]
18	p_1 = [+2 –3 +5 +1 –6 +3] p_2 = [–1 +3.5 –4 +2 +3 –0.5]	–6...+6	[1 0 1 0 0 1]
19	p_1 = [0.5 1.5 2 1 0 1] p_2 = [1 0.5 1 0.2 1.5 2]	0...2	[0 1 0 0 0 1]
20	p_1 = [+1 –2 +0.5 +1.5 –2.5 –0.5] p_2 = [–0.5 +0.5 –0.3 +1.0 +0.8 –2]	–3...+2	[0 1 0 0 1 1]

1.3 Содержание и защита отчета

Для выполнения работы студенты получают индивидуальные задания, варианты которых приведены в таблице 1.2.

Отчет по работе должен содержать задание, структуру сети, параметры настройки и результаты обучения сети.

При защите отчета студент должен ответить на вопросы, касающиеся методики работы с графическим интерфейсом пользователя GUI, а также особенностей настройки нейронных сетей.

1.4 Контрольные вопросы

1 Назначение и функциональные возможности GUI-интерфейса.
2 Правила инициализации, адаптации и обучения сети.
3 Оценка качества работы сети.

2 СОЗДАНИЕ, АДАПТАЦИЯ И ОБУЧЕНИЕ ЛИНЕЙНОЙ НЕЙРОННОЙ СЕТИ В КОМАНДНОМ ОКНЕ MATLAB

Цель работы: освоить методику и приобрести навыки создания, адаптации и обучения линейной нейронной сети с применение пакета прикладных программ (ППП) `Neural Network Toolbox` системы программирования MATLAB.

2.1 Методика создания, адаптации и обучения сети в ППП *NEURAL NETWORK TOOLBOX*

Формирование и инициализация сети. Нейронная сеть создается оператором **`net`**, а линейный слой – оператором **`newlin`**. Например, линейная сеть с двухэлементным вектором входа, значения которого находятся в диапазоне **`[-1 1]`**, одним слоем, без линии задержки на входе (тип 0) и параметром скорости настройки 0.2 формируется следующей командой:

```
net=newlin([-1 1;-1 1],1,0,0.2);
```

Такая сеть функционирует в соответствии с линейной зависимостью вида:

$$a = purelin(\mathbf{W}\mathbf{p} + b),$$

где a – выход сети, $purelin$ – линейная функция активации нейрона, $\mathbf{W}$ – весовая матрица входов, $\mathbf{p}$ – вектор входов, b – смещение нейрона.

Модель этой нейронной сети показана на рисунке 2.1.

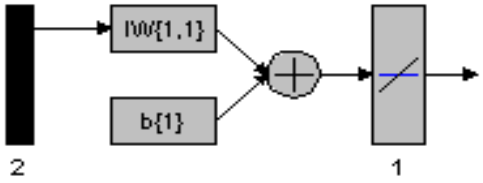


Рисунок 2.1 – Модель линейной нейронной сети с двухэлементным вектором входа

Пусть, например, весовая матрица должна быть задана вектором $\mathbf{W} = [\mathbf{1}\ \mathbf{2}]$, а смещение $b = 0$. Тогда эти параметры сети в описании её структуры будут представлены следующим образом:

```
net.IW{1,1}=[1,2];
net.b{1}=0;
```

При формировании сети оператором `newlin` инициализация сети производится по умолчанию, поэтому применять метод `net=init(net)` не следует.

Адаптация (настройка) нейронной сети. Процедура адаптации осуществляется при введении обучающей выборки последовательным или групповым способом. Для адаптации сети необходимо задать векторы входа **P** и цели **T**, а также установить начальные значения весов **IW** и смещения b.

Процесс адаптации рассмотрим на примере формирования линейной зависимости вида:

$$t = 2p_1 + p_2.$$

При ***последовательном способе*** представления обучающей последовательности векторы задаются в виде массивов ячеек формата `cell`**:**

```
>> net=newlin([-1 1;-1 1],1,0,0);
>> P={[-1;1] [-1/3;1/4] [1/2;0] [1/6;2/3]};
>> T={-1 -5/12 1 1};
>> P1=[P{:}], T1=[T{:}]
P1 =
   -1.0000   -0.3333    0.5000    0.1667
    1.0000    0.2500         0    0.6667
```

```
T1 =
   -1.0000   -0.4167    1.0000    1.0000
```

Вначале зададим сеть с нулевыми значениями начальных весов и смещений:

```
>> net.IW{1}=[0 0];
>> net.b{1}=0;
```

Чтобы обеспечить возможность выбирать произвольные функции для настройки весов и смещений, воспользуемся М-функцией `adapt`. Для линейных сетей, создаваемых с помощью метода `newlin`, по умолчанию устанавливается функция `adaptwb` настройки режима и функция `learnwh` настройки параметров сети по алгоритму WH (правило Уидроу-Хоффа).

Learnwh вычисляет величину изменения веса входа данного нейрона **dw** от вклада нейрона `pn'`, ошибки `e` и скорости настройки `lr`, согласно `Widrow-Hoff` правилу:

$$\mathbf{dw} = lr * e * pn'.$$

Функции настройки весов и смещений задаются в формате
`net.inputWeighst{i,j}.learnFcn` и
`net.biases{i,j}.learnFcn`, соответственно.

Выполним первый цикл адаптации с нулевым параметром скорости настройки и определим значения выходов `(a)` и ошибки `(e)`:

```
>> [net1,a,e]=adapt(net,P,T);
>> net1.IW{1,1},a,e
ans =
     0     0
a =
    [0]    [0]    [0]    [0]
e =
    [-1]    [-0.4167]    [1]    [1]
```

Как видно из приведенного фрагмента, веса не модифицируются, выходы сети остаются равными нулю и адаптация не происходит, потому что параметр скорости настройки не задан.

Зададим начальные значения весов входов и смещения, а также установим функции и параметр скорости настройки. При этом полагаем, что в искомой функции выхода не должно быть постоянной составляющей, а значит, для параметра скорости настройки смещения устанавливаем нулевое значение:

```
>> net.IW{1}=[0 0];
>> net.b{1}=0;
>> net.inputWeights{1,1}.learnParam.lr=0.2;
>> net.biases{1,1}.learnParam.lr=0;
```

```
>> [net1,a,e]=adapt(net,P,T);
>> net1.IW{1,1},a,e
ans =
    0.3454   -0.0694
a =
    [0]    [-0.1167]    [0.1100]    [-0.0918]
e =
    [-1]    [-0.3000]    [0.8900]    [1.0918]
```

Отсюда видно, что процесс адаптации начался, но для достижения требуемой точности, например 0,015, одного цикла недостаточно.

Выполним последовательную адаптацию сети в течение 30 циклов. Для вычисления среднеквадратической ошибки используем оператор `mse`, а для конвертирования массива ячеек `cell` в массив чисел удвоенной точности `double` применим функцию `cell2mat`:

```
>> net=newlin([-1 1;-1 1],1, 0, 0);
>> net.IW{1}=[0 0];
>> net.b{1}=0;
>> net.inputWeights{1,1}.learnParam.lr=0.2;
>> net.biases{1,1}.learnParam.lr=0;
>> P={[-1;1] [-1/3; 1/4] [1/2; 0] [1/6; 2/3]};
>> T={-1 -5/12 1 1};
>> for i=1:30,
[net, a{i}, e{i}]=adapt(net, P, T);
W(i,:)=net.IW{1,1};
end
>> mse(cell2mat(e{30}))
ans =
    0.0017
>> W(30,:)
ans =
    1.9199    0.9250
>> cell2mat(e{30})
ans =
   -0.0056   -0.0081    0.0434    0.0699
```

После 30 циклов адаптации сети по четырем обучающим векторам входа и цели произошло изменение значений весов входов и ошибок.

Построим графики этих изменений в функции числа циклов (итераций) процесса адаптации. Расположим их в одной фигуре друг над другом, применяя для этого функцию `subplot`:

```
>> subplot(2,1,1)
>> for i=1:30, E(i)=mse(e{i}); end
>> semilogy(1:30, E, '+k')
>> xlabel('Циклы'), ylabel('Ошибка'), grid
```

```
>> subplot(2,1,2)
>> plot(0:30,[[0 0]; W], 'k');
>>    xlabel('Циклы'),    ylabel('Веса    входов
w(i)'),grid
```

Графики функций представлены на рисунке 2.2.

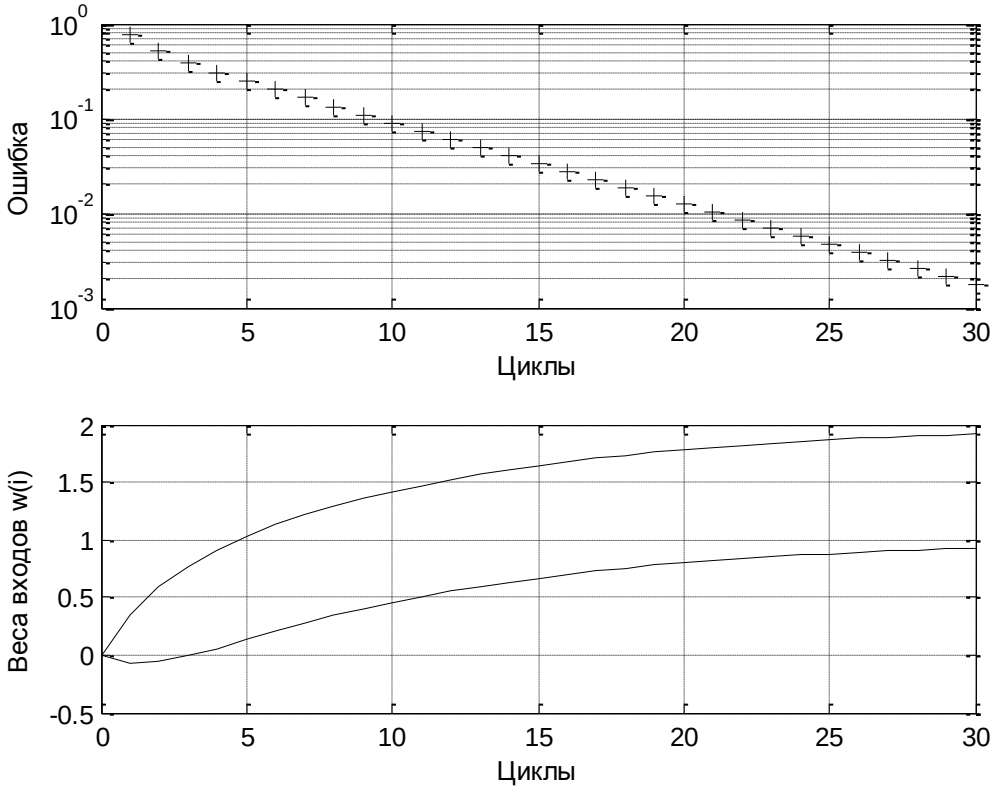


Рисунок 2.2 – Графики изменения значений ошибки и весов входов в процессе адаптации

Не трудно убедиться, что процесс адаптации потребовал 12 шагов – ошибка обучения достигла на этом этапе значения 1,489e–3. Это свидетельствует о том, что обучающая выборка была представительной и сеть соответствует решаемой задаче.

При ***групповом способе*** представления обучающей последовательности векторы входа и цели задаются массивами в формате `double`.

Основной цикл адаптации сети организуется следующим образом:

```
net=newlin([-1 1;-1 1],1,0,0);
P=[-1 -1/3 1/2 1/6; 1 1/4 0 2/3];
T=[-1 -5/12 1 1];
net=newlin([-1 1;-1 1],1,0,0.2);
net.IW{1}=[0 0];
net.b{1}=0;
net.inputWeights{1,1}.learnParam.lr=0.2;
```

```
P=[-1 -1/3 1/2 1/6; 1 1/4 0 2/3];
T=[-1 -5/12 1 1];
EE=10; i=1;
while EE>0.0017176
    [net,a{i}, e{i}, pf]=adapt(net,P,T);
    W(i,:)=net.IW{1,1};
    EE=mse(e{i});
    ee(i)=EE;
    i=i+1;
end
```

Результатом адаптации являются следующие значения коэффициентов линейной зависимости (весов входов), значений выходов сети и среднеквадратичной погрешности адаптации:

```
W(63,:)
ans =
    1.9114    0.8477
cell2mat(a(63))
ans =
   -1.0030   -0.3624    1.0172    0.9426
EE=mse(e{63})
EE =
    0.0016
```

Представим процедуру адаптации нейронной сети в динамике на графиках. Объединим в одну фигуру графики функций выходов, весов входов и ошибки:

```
subplot(3,1,1)
plot(0:63,[zeros(1,4); cell2mat(a')],'k')
xlabel(''),ylabel('Выходы a(i)'),grid
subplot(3,1,2)
plot(0:63,[[0 0];W],'k')
xlabel(''),ylabel('Веса входов w(i)'),grid
subplot (3,1,3)
semilogy(1:63, ee,'+k')
xlabel('Циклы'),ylabel('Ошибка'),grid
```

Графики функций показаны на рисунке 2.3.

Как следует из анализа графиков, для достижения требуемой точности адаптации здесь также потребовалось 12 шагов.

Следует учесть, что в линейных *динамических* сетях естественным является только последовательный способ, так как эти сети имеют линии задержки сигналов.

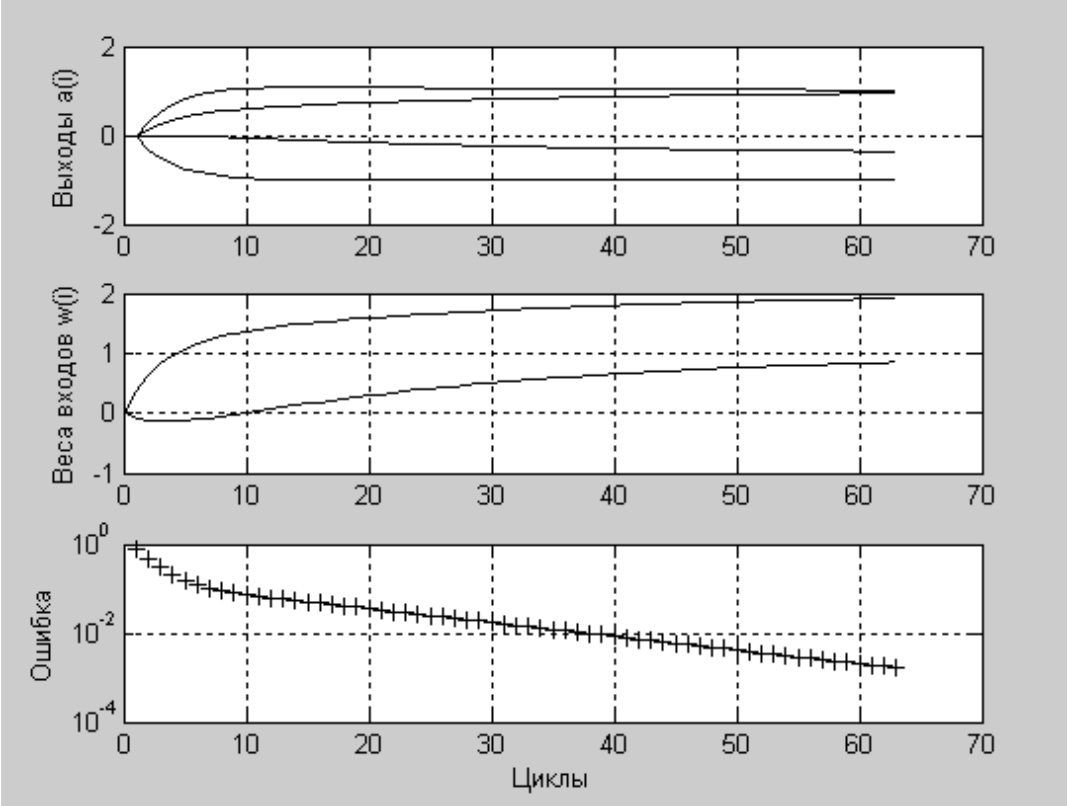


Рисунок 2.3 – Графики выходных сигналов сети, весов входов и ошибки при адаптации с групповым представлением обучающей выборки

Обучение сети. Для обучения и настройки параметров сети используются функции **`trainwb`** и **`learnwh`**, соответственно.

После формирования сети, задания обучающей выборки (последовательным способом), ввода параметров скорости настройки и количества циклов обучения (эпох) получим значения весов входов и график зависимости величины ошибки от числа циклов обучения, который представлен на рисунке 2.4.

```
>> net=newlin([-1 1;-1 1],1, 0, 0);
net.IW{1}=[0 0];
net.b{1}=0;
P={[-1;1] [-1/3; 1/4] [1/2; 0] [1/6; 2/3]};
T={-1 -5/12 1 1};
net.inputWeights{1,1}.learnParam.lr=0.2;
net.biases{1,1}.learnParam.lr=0;
net.trainParam.epochs=30;
net1=train(net,P,T);
W=net1.IW{1}
TRAINB, Epoch 0/30, MSE 0.793403/0.
TRAINB, Epoch 25/30, MSE 0.00373997/0.
TRAINB, Epoch 30/30, MSE 0.00138167/0.
```

```
TRAINB, Maximum epoch reached.
W =
    1.9214    0.9260
```

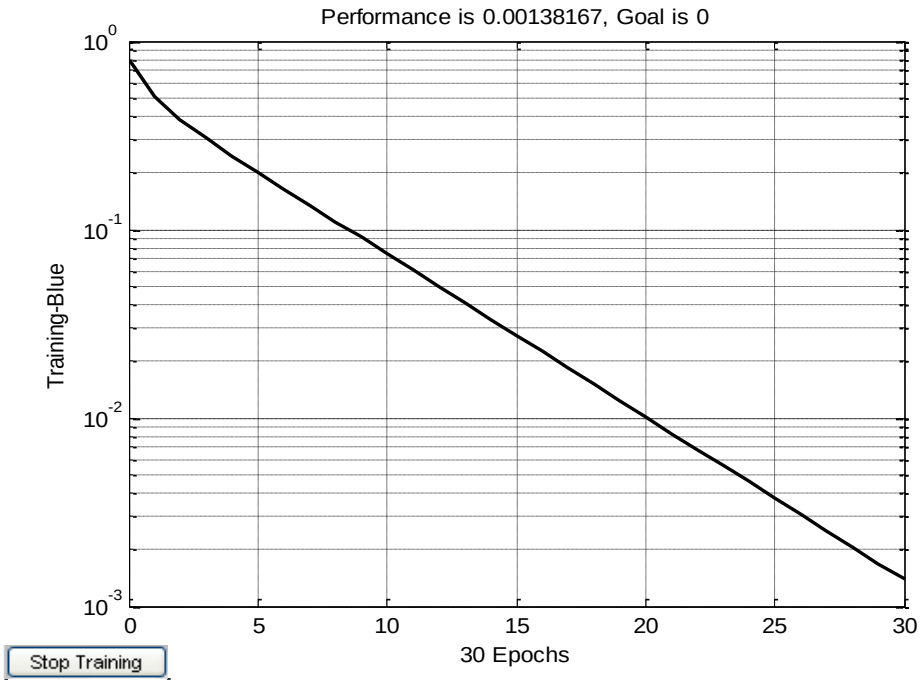


Рисунок 2.4 – График зависимости ошибки обучения от числа циклов

2.2 Методика выполнения работы

Для выполнения работы студенты получают индивидуальные задания, варианты которых приведены в таблице 2.1.

В процессе создания сети необходимо выполнить следующее.

1 Выбрать структуру сети и произвести её инициализацию (задать начальные значения весов и смещение).

2 Выбрать критерий качества и функцию настройки сети.

3 Задать количество циклов адаптации, произвести адаптацию сети.

4 Построить графики функций выхода сети, весов коэффициентов и критерия качества адаптации.

2.3 Содержание и защита отчета

Отчет по работе должен содержать задание, структуру сети, программу и результаты адаптации сети.

При защите отчета студент должен ответить на вопросы, касающиеся методики создания и адаптации линейной нейронной сети в пакете прикладных программ.

Таблица 2.1 – Варианты индивидуальных заданий

Вариант	Вход	**Значения входных сигналов обучающей последовательности** $P1/P2$ **(по порядку)**						**Значения функции цели** T					
		1	2	3	4	5	6	1	2	3	4	5	6
1	1	1/2	-1/2	1	-1/4	1/4	-1/2	0,9	-0,5	1,5	-1	1	-1
	2	0	-1/4	1/4	1/2	-3/4	1/6						
2	1	-1/2	-1	3/4	1/2	-1	-3/4	-0,5	-1,8	1	0,8	1	0
	2	1/5	3/4	-2/5	-1/4	-1	-1/4						
3	1	1/4	1	1/2	-1	-1/2	0	0,3	0,8	0	-0,7	0,2	0,5
	2	-1/4	-1/2	3/4	1/4	0	1						
4	1	1/2	-1/2	-1	1	3/4	-3/4	1,5	1,5	0	-3	-2	0,8
	2	1	1/2	-1/2	-1	-2/3	0						
5	1	-3/4	1/2	-1	0	1	3/4	1,5	0,7	1,5	-1	-2	-2,2
	2	0	1/4	-1/2	-1	0	-3/4						
6	1	3/4	-1	0	1/2	-1/4	-3/4	2	-1	1,5	1,2	-1	-2,3
	2	-3/4	0	-1	-1/2	1/2	1						
7	1	-3/4	1	1/4	1/2	-1	0	-1,6	1,8	-0,1	1	-1,5	-0,2
	2	1	-1/2	1	-1	0	1/2						
8	1	1	1/2	-2/3	3/4	-1	3/4	-0,3	2,4	0	0,2	-1,3	2,3
	2	1	-1	-1/2	1/2	0	-3/4						
9	1	3/4	-1	1/4	-1/4	1	-1/5	1,5	-2	-0,4	1	1	-1
	2	-3/4	1	1/2	-1	-1/4	3/4						
10	1	1/2	-1/2	-1	1	1/2	0	-0,3	0,7	-2,8	3,5	2	-1
	2	3/4	-1	1/2	-1	-2/3	2/3						

2.4 Контрольные вопросы

1 В каком формате записывается оператор создания линейной сети?

2 Какой зависимостью описывается линейная нейронная сеть?

3 Как представляется структура модели однослойной линейной сети с R входами?

4 Как производится инициализация сети?

5 Как осуществляется адаптация сети?

6 В каких форматах вводятся обучающие значения?

7 Какой критерий обычно применяется для настройки линейной сети?

8 Какую значимость имеет параметр скорости настройки сети?

9 Чем отличаются последовательный и групповой способы задания обучающих выборок?

3 РАЗРАБОТКА РАДИАЛЬНОЙ БАЗИСНОЙ НЕЙРОННОЙ СЕТИ ДЛЯ АППРОКСИМАЦИИ ФУНКЦИЙ

Цель работы: приобрести навыки создания, настройки и моделирования радиальных базисных сетей для аппроксимации функций.

3.1 Методика формирования, обучения и моделирования радиальной базисной сети

Принцип функционирования сети. Сети радиальных базисных функций (RBF) отличаются от линейных нейронных сетей тем, что содержат слой радиально-симметричных скрытых нейронов (шаблонный слой). Для обеспечения радиально-симметричных свойств необходимо:

- наличие центра, представленного вектором;
- наличие способа измерения расстояния от центра до входного вектора;
- наличие специальной функции, отображающей функцию расстояния.

В радиальных базисных сетях все эти условия обеспечиваются применением радиального базисного нейрона с функцией активации вида:

$$a = radbas(n) = e^{-n^2},$$

где n – вход функции активации, определяемый как модуль разности вектора весов и вектора входа, умноженный на смещение: $n = \|\mathbf{p} - \mathbf{w}\| b$.

Практически это означает, что выходной сигнал шаблонного нейрона – это функция расстояния между входным вектором и сохраненным центром. Чем ближе входной вектор к центру, тем больше выходной сигнал нейрона (рис. 3.1). При этом радиальный базисный нейрон действует как индикатор, который формирует значение 1, когда вход $\mathbf{p}$ идентичен вектору весов $\mathbf{IW}$. Смещение b позволяет корректировать чувствительность радиального базисного нейрона.

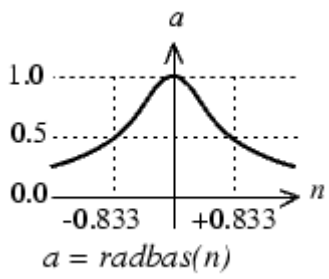
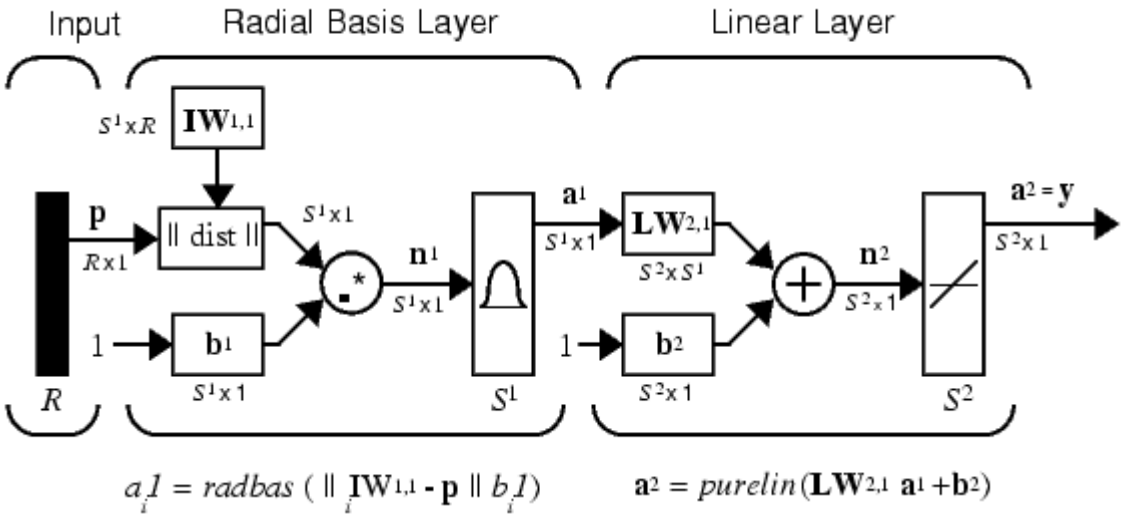
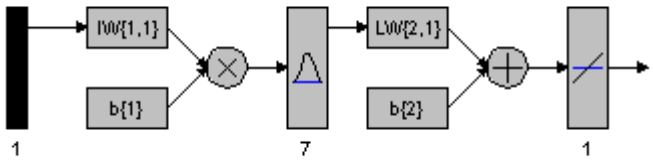


Рисунок 3.1 – График функции активации

Структура сети показана на рисунке 3.2. Сеть содержит радиальный базисный слой и линейный слой.



а



б

Рисунок 3.2 – Полная (а) и упрощенная (б) структуры модели радиальной базисной сети

Скрытый радиальный базисный слой имеет S^1 нейронов, а выходной слой – S^2 нейронов. Радиальный базисный слой содержит специальный блок `dist`, на вход которого подаются входной вектор **p** и матрица весов **IW** . Выходом блока является вектор, состоящий из S^1 элементов, которые определяются расстояниями между i-м вектором входа и i-ой вектор-строкой **IW** матрицы весов. Выход блока `dist` умножается поэлементно на вектор смещения **b** и формирует вход функции активации. В качестве функции активации используется обычно функция Гаусса.

Создание сети. Для создания радиальной базисной сети предназначены М-функции `newrbe` и `newrb`. Первая позволяет построить сеть с нулевой ошибкой, вторая позволяет управлять количеством нейронов скрытого слоя. Недостаток функции `newrbe` заключается в том, что она формирует сеть с числом нейронов в скрытом слое, равным числу элементов обучающего множества. Если таких элементов много, что

характерно для реальных случаев, то приемлемое решение достигается с трудом.

Создадим сеть с нулевой ошибкой:

```
net=newrbe(P,T,SPREAD);
```

Входными аргументами функции `newrbe` являются массивы входных векторов **p** и векторов цели **T**, а также параметр влияния `SPREAD`. Функция устанавливает веса первого слоя равными **p**. Смещения устанавливаются равными 0,8326 / `SPREAD`. Это означает, что уровень перекрытия радиальных базисных функций равен 0,5 и все входы в диапазоне ±SPREAD считаются значимыми.

Веса второго слоя могут быть найдены путем моделирования выходов первого слоя (A {1}) и последующего решения системы линейных алгебраических уравнений:

```
[W{2,1} b{2}] * [A{1}; ones]=T
```

Так как входы второго слоя A {1} и цели T известны, а функция активации линейная, то для вычисления весов и смещений второго слоя достаточно решить систему уравнений:

```
Wb = T/[P; ones(1, size(P,2))]
```

Рассмотрим процедуру создания и моделирования радиальной базисной сети для произвольной функции. Значения входов изменяются в диапазоне от -1 до 1. Обучающая выборка составлена для 21 точки функции через 0,1. График функции представлен на рисунке 3.3.

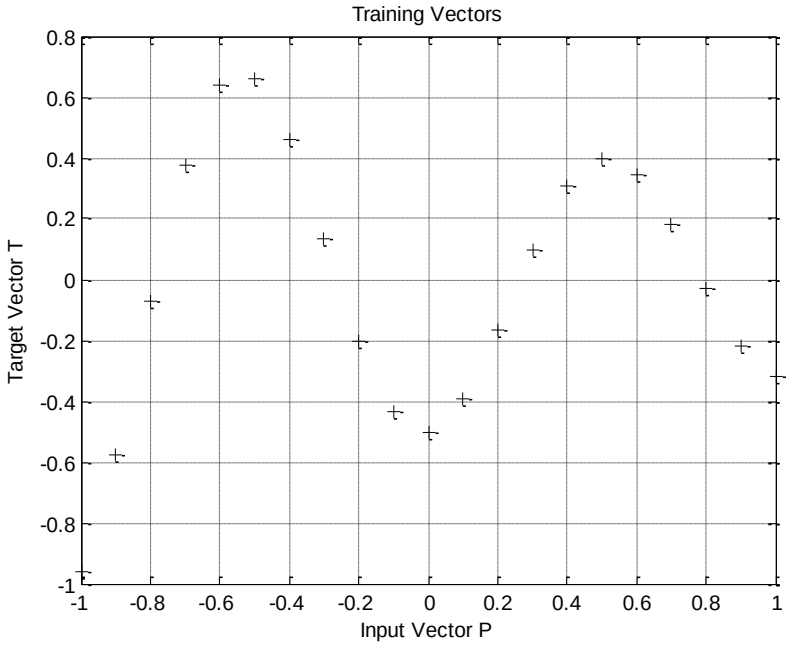


Рисунок 3.3 – График обучающей функции (вектор **T***)*

```
P = -1:.1:1;
```

```
T = [-.9602 -.5770 -.0729  .3771  .6405  .6600  .4609
   .1336 -.2013 -.4344 -.5000 -.3930 -.1647  .0988
   .3072  .3960  .3449  .1816 -.0312 -.2189 -.3201];
plot(P,T,'+');
title('Training Vectors');
xlabel('Input Vector P');
ylabel('Target Vector T');grid
```

Сохраним текущий график и сформируем сеть:

```
hold on
net = newrbe(P,T);
net.layers{1}.size
Warning: Rank deficient, rank = 13, tol =
2.2386e-014.
> In newrbe>designrbe at 120
  In newrbe at 103
ans =
   21          % Количество нейронов скрытого слоя равно 21
```

Произведем моделирование сети:

```
V = sim(net,P);
plot (P,V,'ob','MarkerSize',5,'LineWidth',2)
p = [-0.75 -0.25 0.25 0.75];
v = sim(net,p);
plot(p,v,'+k','MarkerSize',10,'LineWidth',2)
```

Результаты моделирования приведены на рисунке 3.4.

Как видно из рисунка 3.4, вектор **T** для новых значений входа определен с высокой точностью.

Создадим сеть с меньшим числом нейронов, применив функцию `newrb`. Новую сеть обучим прежней последовательностью целей:

```
P = -1:.1:1;
T = [-.9602 -.5770 -.0729  .3771  .6405  .6600  .4609
     .1336 -.2013 -.4344 -.5000 -.3930 -.1647  .0988
     .3072  .3960  .3449  .1816 -.0312 -.2189 -.3201];
plot(P,T,'*r','MarkerSize',4,'LineWidth',2)
hold on
GOAL=0.01; %Допустимое значение ошибки
net=newrb(P,T,GOAL);
net.layers{1}.size
NEWRB, neurons = 0, SSE = 3.69051
ans =
```

```
6   % Количество нейронов скрытого слоя равно 6
v=sim(net,p);
p=[-0.75 -0.25 0.25 0.75];
v=sim(net,p);
plot(p,v,'+k','MarkerSize',10,'LineWidth',2)
```

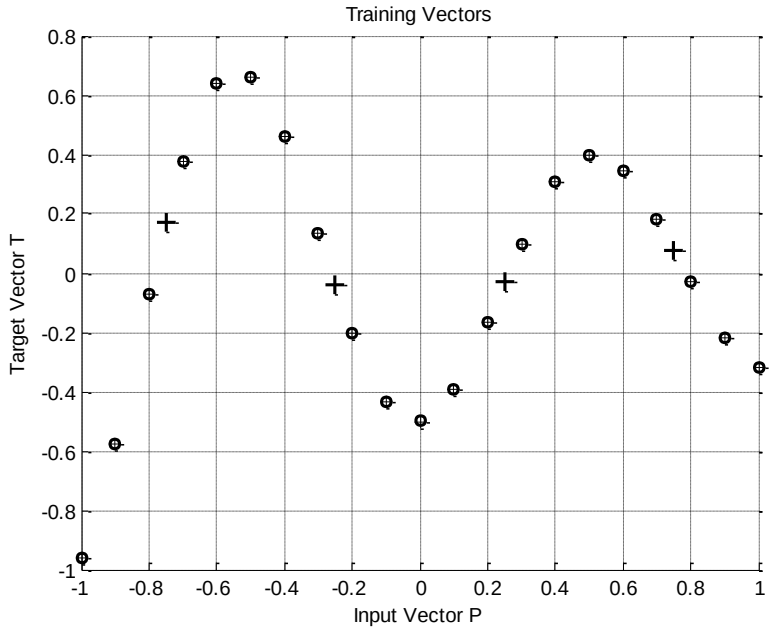


Рисунок 3.4 – Результаты моделирования сети при подаче на вход четырех новых значений

Как видно из М-файла, число нейронов первого слоя равно 6. При этом обеспечивается среднеквадратическая ошибка менее 0,01. Результаты моделирования приведены на рисунке 3.5.

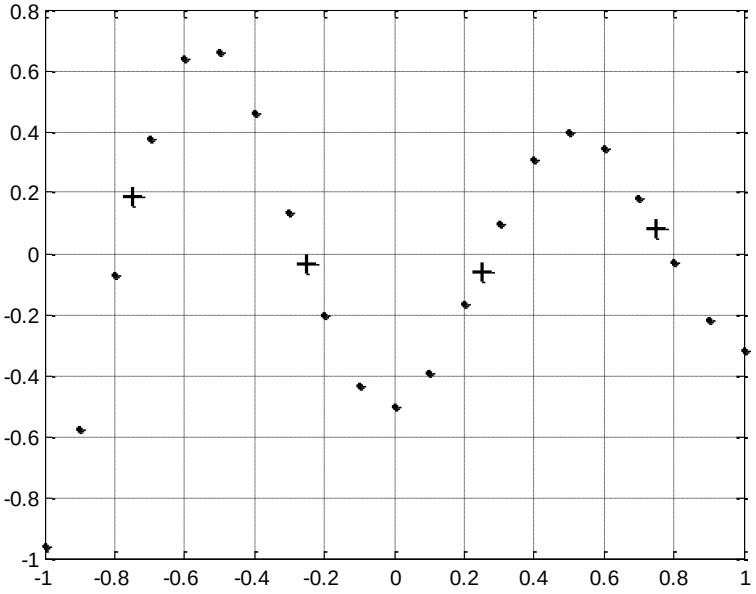


Рисунок 3.5 – Результаты моделирования сети с шестью нейронами

Аппроксимация функций. При применении радиальных базисных сетей для аппроксимации функций используется идея разложения произвольной функции на ряд радиальных базисных функций следующим образом:

$$f(x) = \sum_{i=1}^{N} a_i \varphi_i(x),$$

где $\varphi_i(x)$ – радиальная базисная функция.

Рассмотрим процесс аппроксимации трех радиальных базисных функций, заданных на интервале [-3 3]:

```
p = -3:.1:3;
a1 = radbas(p);
a2 = radbas(p-1.5);
a3 = radbas(p+2)*0.5;
a4 = a1 + a2 + a3;
plot(p,a1,'b-',p,a2,'b--',p,a3,'b--',p,a4,'m-')
title('Weighted Sum of Radial Basis Transfer Functions');
xlabel('Input p');
ylabel('Output a');
```

Результаты аппроксимации представлены на рисунке 3.6.

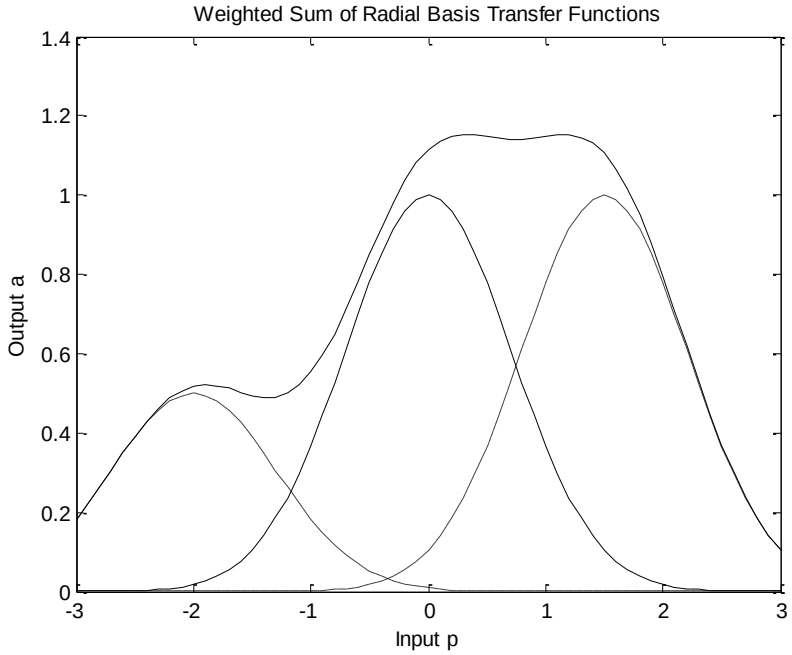


Рисунок 3.6 – Графики базисных функций и результат их аппроксимации

Из рисунка видно, что при разложении произвольной функции радиальными базисными функциями обеспечивается необходимая гладкость.

Степень гладкости и точность аппроксимации связаны с параметром влияния SPREAD. Как известно, этот параметр определяет диапазон значимых входов (см. выше).

Рассмотрим, в чем проявляется это влияние.

Используя предыдущий пример обучающей выборки, будем задавать значения SPREAD, равными 0,01; 1 и 20.

Введем М-файл со значением `SPREAD = 0,01` и получим результат (рис. 3.7).

```
P = -1:.1:1;
T = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609
   .1336 -.2013 -.4344 -.5000 -.3930 -.1647 .0988
   .3072 .3960 .3449 .1816 -.0312 -.2189 -.3201];
  GOAL=0.01;
  SPREAD=0.01;
  net=newrb(P,T,GOAL,SPREAD);
  net.layers{1}.size
NEWRB, neurons = 0, SSE = 2.758
ans =
    19        % Количество нейронов скрытого слоя равно 19
X=-1:.01:1;
Y=sim(net,X);
plot(X,Y);
title('Training Vectors');
xlabel('Input Vector P');
ylabel('Target Vector T');
```

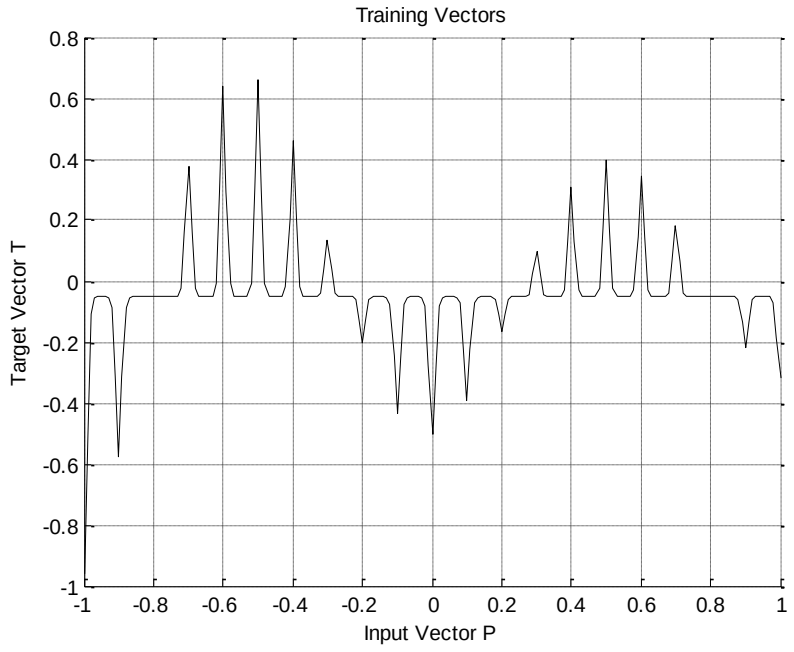


Рисунок 3.7 – Результат аппроксимации при `SPREAD=0,01`

Как видно из результатов моделирования, сеть состоит из 19 нейронов и не обеспечивает гладкости функции аппроксимации.

Рассмотрим процесс аппроксимации при `SPREAD = 1`.

```
P = -1:.1:1;
T = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609
   .1336 -.2013 -.4344 -.5000 -.3930 -.1647 .0988
   .3072 .3960 .3449 .1816 -.0312 -.2189 -.3201];
  GOAL=0.01;
>> SPREAD=1;
>> net=newrb(P,T,GOAL,SPREAD);
  net.layers{1}.size
NEWRB, neurons = 0, SSE = 3.69051
ans =
     6
>> X=-1:.01:1;
Y=sim(net,X);
hold on;
plot(X,Y);
title('Training Vectors');
xlabel('Input Vector P');
>> grid
```

Результаты моделирования сети представлены на рисунке 3.8.

Эта сеть состоит из 6 нейронов скрытого слоя и достаточно хорошо аппроксимирует нелинейную зависимость, заданную множеством из 21 точки.

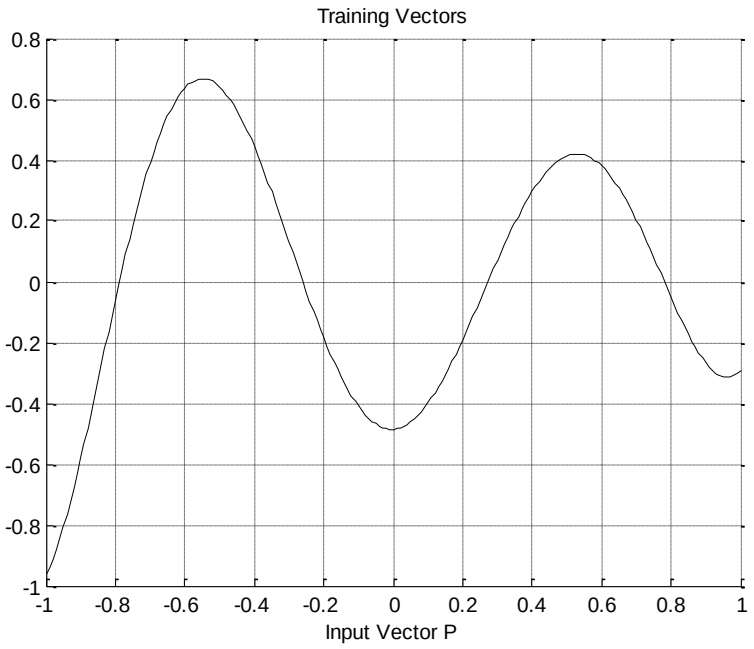


Рисунок 3.8 –График функции аппроксимации при значении `SPREAD=1`

Установим параметр влияния SPREAD=20.

```
P = -1:.1:1;
T = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609
   .1336 -.2013 -.4344 -.5000 -.3930 -.1647 .0988
   .3072 .3960 .3449 .1816 -.0312 -.2189 -.3201];
  GOAL=0.01;
SPREAD=20;
net=newrb(P,T,GOAL,SPREAD);
  net.layers{1}.size
NEWRB, neurons = 0, SSE = 3.69972
ans =
    21            % Количество нейронов скрытого слоя в сети
X=-1:.01:1;
Y=sim(net,X);
hold on;
plot(X,Y);
title('Training Vectors');
xlabel('Input Vector P'); grid
```

Результат моделирования приведен на рисунке 3.9.

Из рисунка 3.9 видно, что при большом значении параметра влияния SPREAD и большом количестве нейронов скрытого слоя (21 нейрон) функции активации перекрываются, и каждый базисный нейрон для всех значений входа генерирует выходные значения, близкие к 1. Это приводит к тому, что сеть не может обеспечить требуемую точность из-за вычислительных проблем.

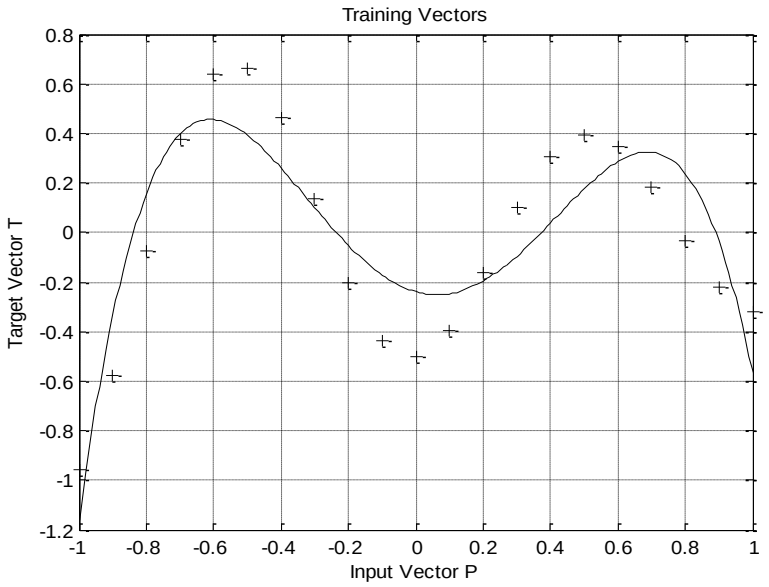


Рисунок 3.9 – График функции аппроксимации при значении SPREAD=20

Таким образом, по результатам выполненных исследований можно сделать вывод о том, что параметр влияния SPREAD следует брать большим, чем шаг разбиения функции обучающей последовательности, но меньшим самого интервала её разбиения. Для рассмотренного примера этот диапазон составляет от 0,1 до 2.

3.2 Методика выполнения работы

Для выполнения работы студенты получают индивидуальные задания, варианты которых приведены в таблице 3.1.

Таблица 3.1 – Варианты заданий

Вариант	Диапазон входного вектора P		Диапазон цели T	
	от	до	от	до
1	-1	+1	0	+1
2	-2	+2	0	+2
3	-3	+3	-1	+1
4	-4	+4	-2	+2
5	-5	+5	-3	+3

В процессе выполнения работы необходимо выполнить следующее.

1 С применением функции newrbe создать структуру сети с нулевой ошибкой и произвести её обучение.

2 С применением функции newrb создать структуру сети и произвести её обучение при различных значениях параметра влияния SPREAD.

3 Построить графики функций выхода сети и сделать вывод о качестве аппроксимации.

3.3 Содержание и защита отчета

Отчет по работе должен содержать задание, структуру сети, программы и результаты работы сети.

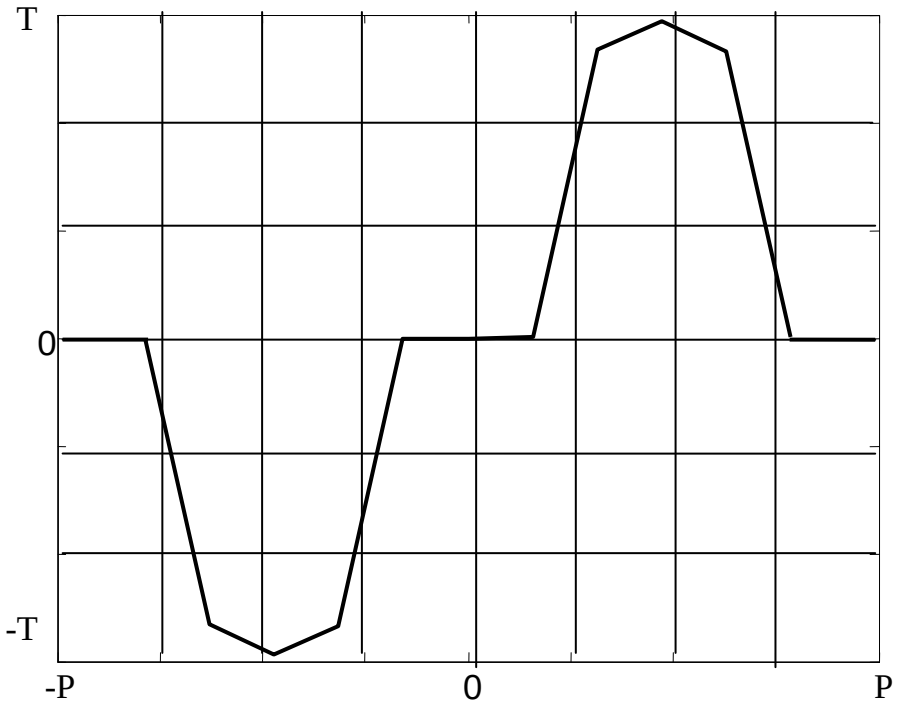
При защите отчета студент должен ответить на вопросы, касающиеся методики создания и применения радиальной базисной нейронной сети.

3.4 Контрольные вопросы

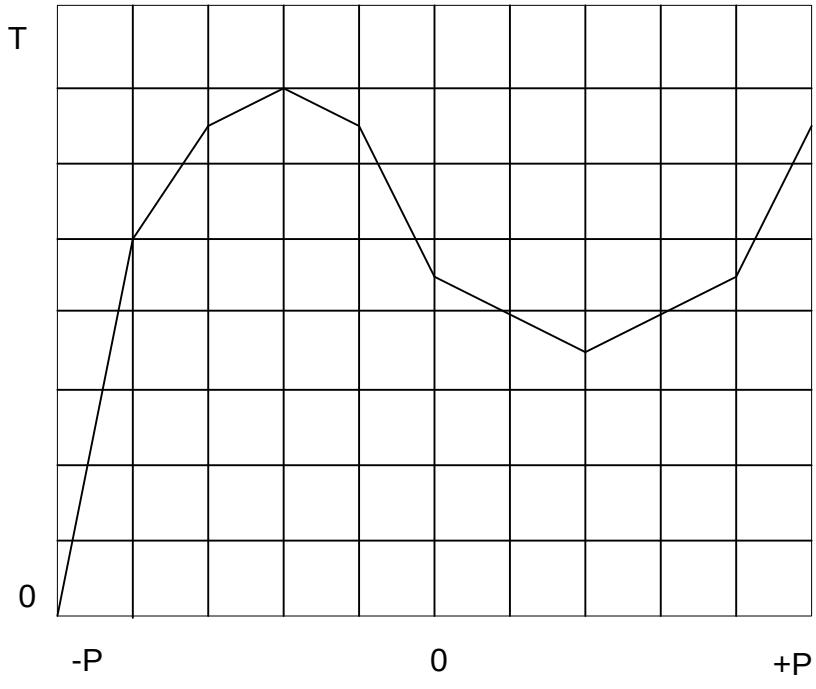
1 Чем отличается структура радиальных базисных сетей от структуры линейных нейронных сетей?

2 Какими свойствами должна обладать радиальная базисная сеть?

3 Какие функции активации применяются в радиальных базисных сетях?

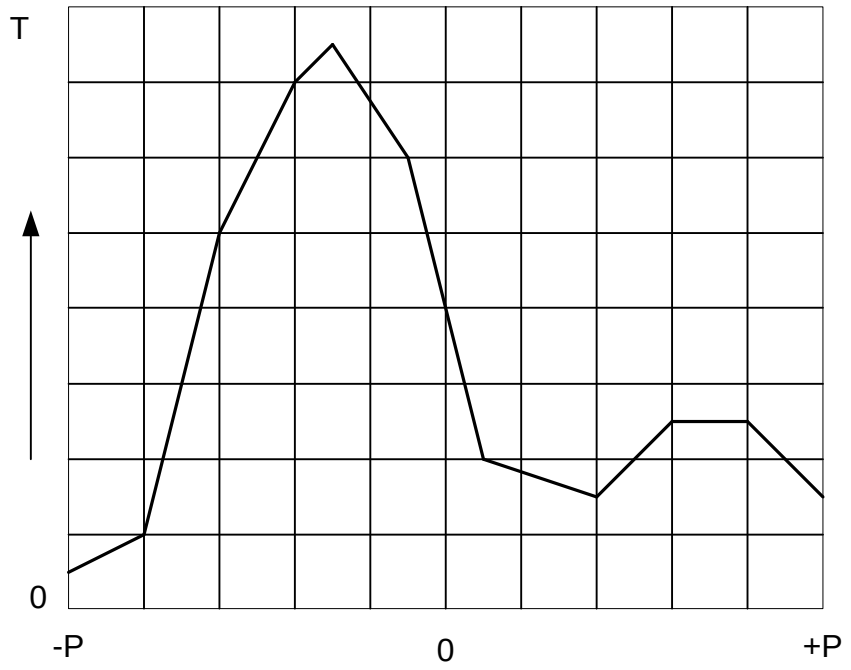


а

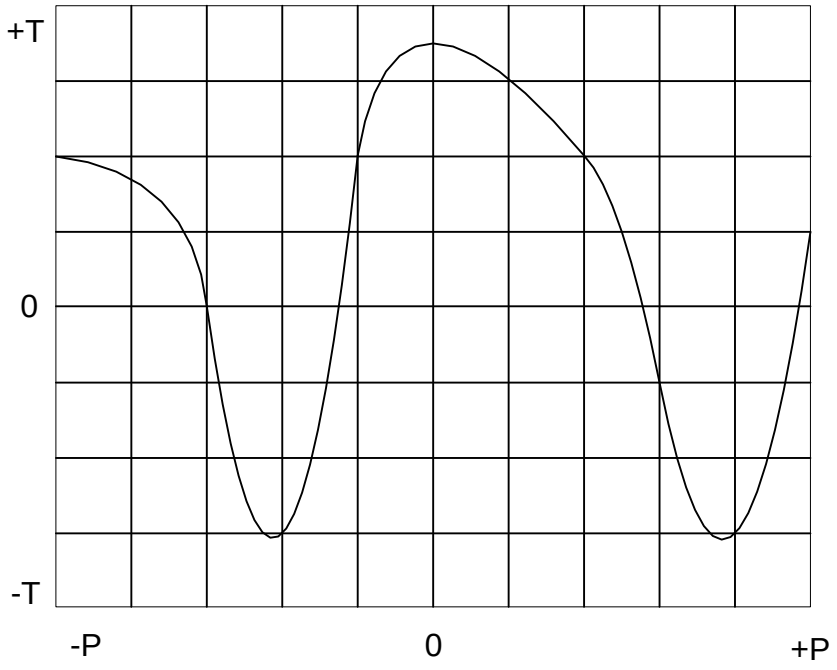


б

Рисунок 3.10, лист 1



в



г

Рисунок 3.10, лист 2

4 Для чего предназначен скрытый шаблонный слой?

5 Какие М-функции применяются для создания радиальных базисных сетей и в чем их различие?

6 Как задается обучающая выборка для аппроксимации функций?

7 Какая идея используется для аппроксимации функций радиальной базисной сетью?

8 От чего зависит количество нейронов скрытого слоя?

9 Какое влияние оказывает параметр **SPREAD** на точность и гладкость функции аппроксимации. В каком диапазоне он должен находиться?

3.5 Индивидуальные задания

Вариант индивидуального задания определяется номером рисунка 3.10 и строкой таблицы. Например, вариант "а-2" означает, что исходные данные для индивидуального задания приведены на рис. 3.10,а и в строке 2 таблицы 3.1.

4 РАЗРАБОТКА НЕЙРОННОЙ СЕТИ ДЛЯ МОДЕЛИРОВАНИЯ СТАЦИОНАРНЫХ СИГНАЛОВ

Цель работы: приобрести умения и навыки в создании и обучении нейронных сетей для идентификации параметров стационарных сигналов.

4.1 Постановка задачи и синтез сети для моделирования стационарных сигналов

Постановка задачи предсказания стационарного сигнала. В цифровых системах управления непрерывные сигналы представляются дискретными сигналами путем применения операции квантования. Большое значение имеет возможность предсказания значения сигнала на несколько шагов вперед. Время шага определяется периодом дискретности цифрового сигнала.

Пусть, например, требуется предсказать значение сигнала y_k на выходе сети в момент времени t_k, используя для этого 5 последних значений сигналов T. В математическом смысле – это задача экстраполяции функции, которая может быть сформулирована следующим образом:

$$\begin{cases} p_{k-1} = T_{k-1}, \ \ i = 1,...5; \\ y_k = \sum_{i=1}^{5} w_i p_{k-1}, \end{cases}$$

где w_i – весовые коэффициенты сети.

Зададим гармонический сигнал y_k, диапазон наблюдения 0…5 с и период дискретности (такт квантования) $h = 0{,}025$ с:

$$y_k = \sin(4\pi t_k) = \sin(4\pi kn),\ t_k = t_0 : h : t_f = 0 : 0.025 : 5.$$

С позиции теории нейронных сетей задача экстраполяции превращается в задачу настройки параметров и обучения сети.

Синтез нейронной сети для предсказания значений сигнала. Поскольку сигнал стационарный и соотношения между прошлыми и будущими значениями неизменны, можно воспользоваться линейной моделью нейронной сети. Для анализа 5 сигналов сеть должна состоять из одного нейрона с 5 входами и содержать линию задержки с пятью блоками запаздывания. Структура сети приведена на рисунке 4.1.

Входную последовательность выберем на интервале от 0 до 1с, а контрольное подмножество сформируем из значений сигналов в интервале от 3 до 5с.

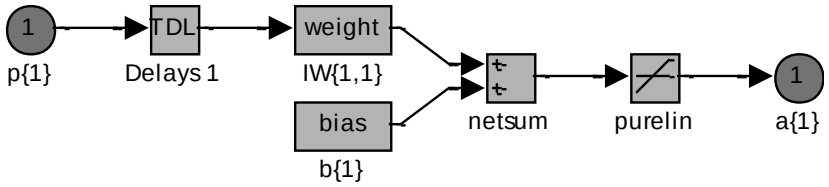


Рисунок 4.1 – Структура сети для экстраполяции стационарного сигнала

Входная последовательность будет иметь длину $Q1 = 1/h$, каждый вектор входа должен быть разложен на компоненты, которые будут соответствовать запаздывающим значениям сигнала.

Сформируем сигнал, зададим входной и целевой векторы, а также зададим структуру сети:

```
clf;      % Очистка графического окна
figure(gcf)
echo on
time = 0:0.025:5;    % от 0 to 5 секунд
T = sin(time*4*pi);% Задание сигнала
Q = length(T);     % Задание длины вектора
P = zeros(5,Q);
P(1,2:Q) = T(1,1:(Q-1));
P(2,3:Q) = T(1,1:(Q-2));
P(3,4:Q) = T(1,1:(Q-3));
P(4,5:Q) = T(1,1:(Q-4));
P(5,6:Q) = T(1,1:(Q-5));
```

```
plot(time,T)     % Построение графика
xlabel('Time');
ylabel('Target Signal');
title('Signal to be Predicted');
>> net = newlind(P,T);     % Задание структуры сети
```

График входного сигнала для задачи предсказания показан на рисунке 4.2.

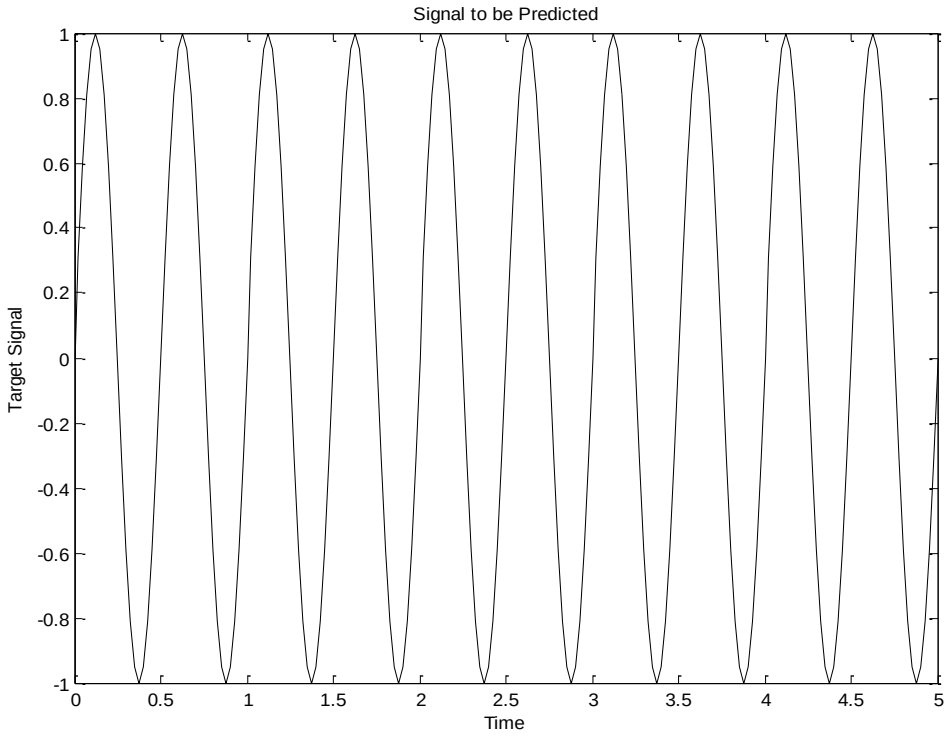


Рисунок 4.2 – График стационарного дискретного сигнала

Выполним проверку сети, используя два множества – обучающее и контрольное.

```
h=0.025;
Q1=1/h;      % Определение длины обучающего множества (Q1=40)
P1=zeros(5,Q1);    % Обнуление матрицы входного вектора
P1(1,1:Q1)=T(1,1:Q1);  % Сопоставление векторов входа и цели
                                  для 5 тактов
P1(2,2:Q1)=T(1,1:(Q1-1));
P1(3,3:Q1)=T(1,1:(Q1-2));
P1(4,4:Q1)=T(1,1:(Q1-3));
P1(5,5:Q1)=T(1,1:(Q1-4));
T1=T(1,6:(Q1+5));   % Формирование вектора цели с 6 по 45
                                  такты
T2=T(1,3/h:Q);
net = newlind(P1,T1);
```

```
a=sim(net,P1(:,1:Q1));     ;% Моделирование сети
t1=6:Q1+5;    %
plot(time(t1),a,'*r',time(1:Q1+5),T(1,1:Q1+5))
xlabel('Time,c');
```

Результаты моделирования представлены на графике рисунка 4.3.

Из рисунка 4.3 видно, что нейронная сеть обеспечивает достаточно высокую точность отслеживания сигнала.

Теперь проверим работу сети, используя контрольное множество.

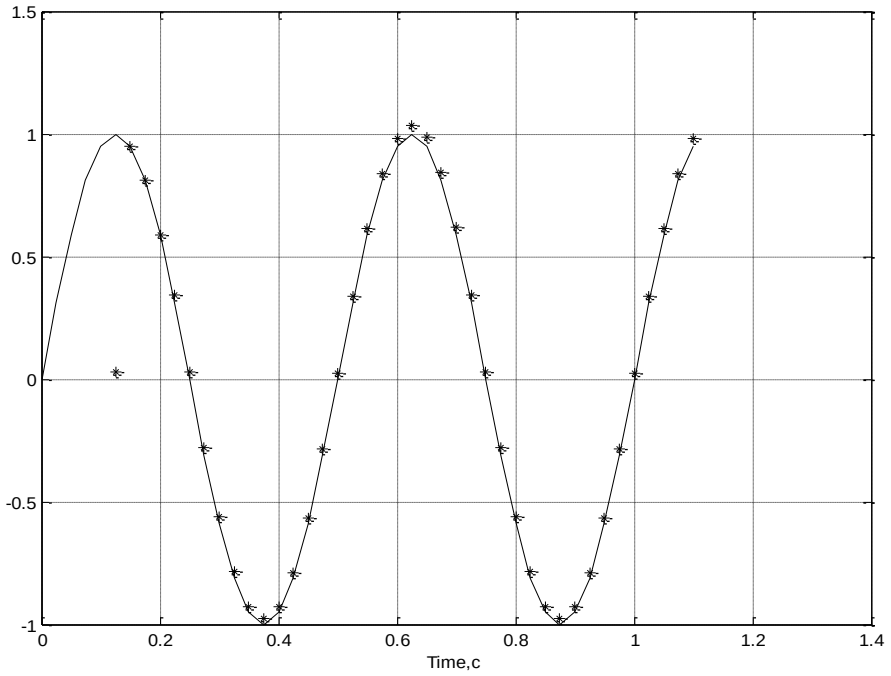


Рисунок 4.3 – Графики изменения моделированного (звездочки) и фактического сигналов

Зададим длину входной последовательности $N1$.

```
N1=20;
Tt=T2(1,1:N1);
P2(1,:)=Tt(1,:);
P2(2,2:end)=Tt(1,1:end-1);
P2(3,3:end)=Tt(1,1:end-2);
P2(4,4:end)=Tt(1,1:end-3);
P2(5,5:end)=Tt(1,1:end-4);
a=sim(net,P2);    % Моделирование сети
figure(3),clf
h1=plot(time(1:size(P2, 2)-5), a(1:end-5),'*');
hold on
h2=plot(time(1:size(P2, 2)-5), Tt(6:end), 'r');
>> grid
```

Используя свойства графического объекта, вычислим погрешности экстраполяции в функции длины обучающей выборки:

```
y1=get(h1,'YData'); y2=get(h2,'YData'
minlength=min(length(y1), length(y2));
e=y1(1:minlength)-y2(1:minlength); % Расчет ошибки
nre=sqrt(mse(e));     % Вычисление квадратного корня
plot(time(1:size(P2, 2)-5),e);
```

Результаты моделирования сети и вычисления погрешности экстраполяции представлены на рисунке 4.4.

Из анализа рисунка 4.4 можно сделать вывод о том, что нейронная сеть обеспечивает достаточную точность экстраполяции – ошибка равна $e = 0.0220$.

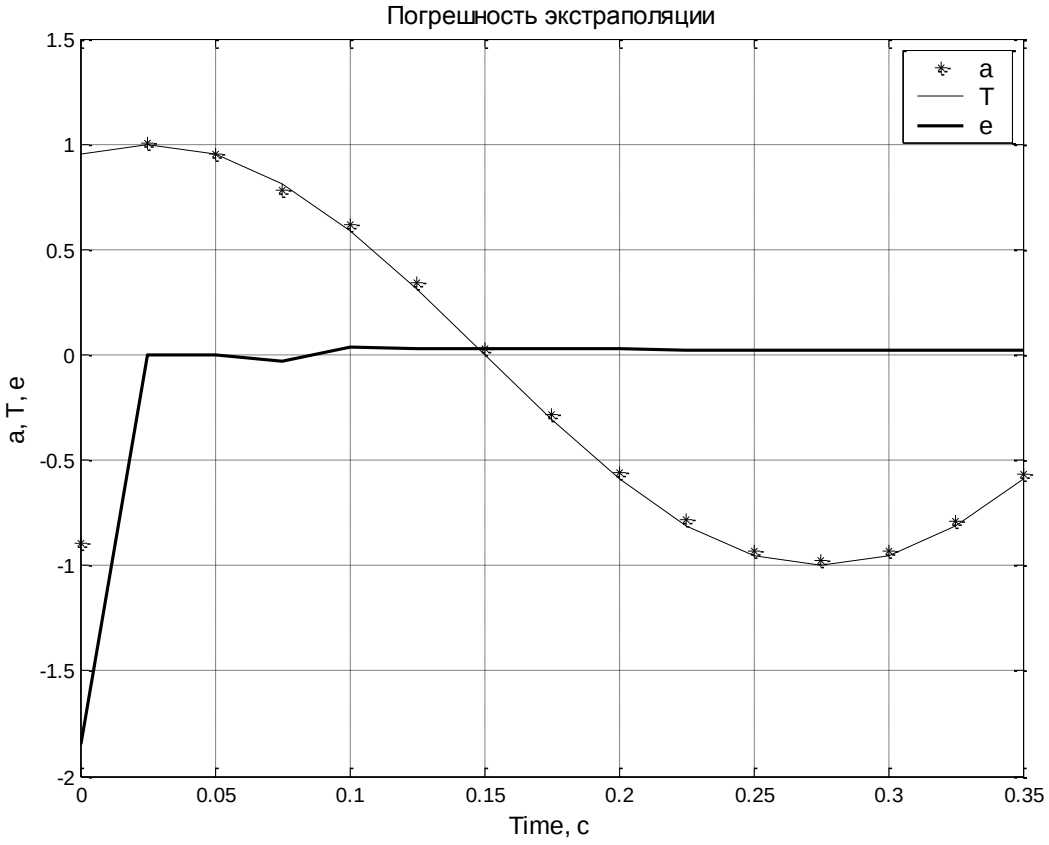


Рисунок 4.4 – График реакции сети на контрольное множество

4.2 Методика выполнения работы

Для выполнения работы студенты получают индивидуальные задания, варианты которых приведены в таблице 4.1.

В процессе выполнения работы необходимо выполнить следующее.

1 Определить необходимую длину входного вектора.

2 Сформировать обучающую последовательность (вектор цели).

3 Произвести синтез нейронной сети и её обучение.

4 Построить графики функций выхода сети и сделать вывод о качестве экстраполяции сигнала.

4.3 Содержание и защита отчета

Отчет по работе должен содержать задание, структуру сети, программы и результаты работы сети.

При защите отчета студент должен ответить на вопросы, касающиеся методики создания и применения нейронной сети для экстраполяции функций.

Таблица 4.1 – Варианты индивидуальных заданий для аппроксимации функции

Вар.	Функция	t_f	h	Вар.	Функция	t_f	h
1	$y = \sin(2\pi t_k)$	10	0,05	11	$y = \sin(2\pi t_k)$	8	0,04
2	$y = \sin(2{,}5\pi t_k)$	7,5	0,05	12	$y = \sin(2{,}5\pi t_k)$	5	0,04
3	$y = \sin(3\pi t_k)$	6	0,02	13	$y = \sin(3\pi t_k)$	7,5	0,05
4	$y = \sin(3{,}5\pi t_k)$	5	0,02	14	$y = \sin(3{,}5\pi t_k)$	6	0,05
5	$y = \sin(4\pi t_k)$	4,5	0,02	15	$y = \sin(4\pi t_k)$	4	0,05
6	$y = \sin(4{,}5\pi t_k)$	4,25	0,01	16	$y = \sin(4{,}5\pi t_k)$	3	0,02
7	$y = \sin(5\pi t_k)$	4	0,01	17	$y = \sin(5\pi t_k)$	3	0,02
8	$y = \sin(5{,}5\pi t_k)$	3	0,01	18	$y = \sin(5{,}5\pi t_k)$	2,5	0,02
9	$y = \sin(6\pi t_k)$	2	0,005	19	$y = \sin(6\pi t_k)$	1,5	0,01
10	$y = \sin(1{,}5\pi t_k)$	8	0,04	20	$y = \sin(1{,}5\pi t_k)$	10	0,05

4.4 Контрольные вопросы

1 В чем заключается сущность задачи экстраполяции функций?

2 Что включает в себя структура сети для экстраполяции сигналов?

3 Как задается длина вектора обучающей последовательности при решении задачи экстраполяции сигнала?

4 Как проверяется качество работы нейронной сети при предсказании значений стационарного сигнала?

5 РАЗРАБОТКА НЕЙРОННОЙ СЕТИ ДЛЯ МОДЕЛИРОВАНИЯ СТАЦИОНАРНОГО ФИЛЬТРА

Цель работы: приобрести умения и навыки в создании и обучении нейронных сетей для моделирования стационарных фильтров.

5.1 Постановка задачи и синтез сети для моделирования стационарного фильтра

Задача заключается в создании сети, которая в процессе обучения определяет параметры фильтра, а затем в качестве модели фильтра используется для анализа выходных сигналов при произвольных значениях входных сигналов. Первая часть этой задачи известна как задача идентификации.

Рассмотрим линейный стационарный фильтр, описываемый рекуррентным выражением:

$$y[n] = -0.5y[n-1] + 1.5y[n-2] + r[n].$$

Этот цифровой фильтр второго порядка может быть представлен дискретной передаточной функцией:

$$y[n] = \frac{1}{1 + 0.5z^{-1} - 1.5z^{-2}} \cdot r[n].$$

В системе MATLAB такой фильтр описывается М-функцией:

```
Y=filter([1 0.5 -1.5],1,R;
```

Допустим, что на вход фильтра подается сигнал:

$$r(t) = \sin(10 * \sin(t) * t).$$

Зададим сигнал массивом значений с периодом дискретности 0,025 с на интервале 5 с. Входную последовательность **P** зададим на основе текущего и двух предшествующих значений входа **X**.

```
>> Q=size(X,2);
P = zeros(3,Q);
P(1,1:Q) = X(1,1:Q);
P(2,2:Q) = X(1,1:(Q-1));
P(3,3:Q) = X(1,1:(Q-2));
T = filter([1 0.5 -1.5],1,X);
```

Выведем графики входного и выходного сигналов фильтра (рис. 5.1).

```
subplot(2,1,1)
plot(time,X)
axis([0 5 -2 2]);
```

```
xlabel('Time');
ylabel('Input Signal');
title('Input Signal to the System');
subplot(2,1,2)
plot(time,T)
axis([0 5 -2 2]);
xlabel('Time');
ylabel('Output Signal');
title('Output Signal of the System');
```

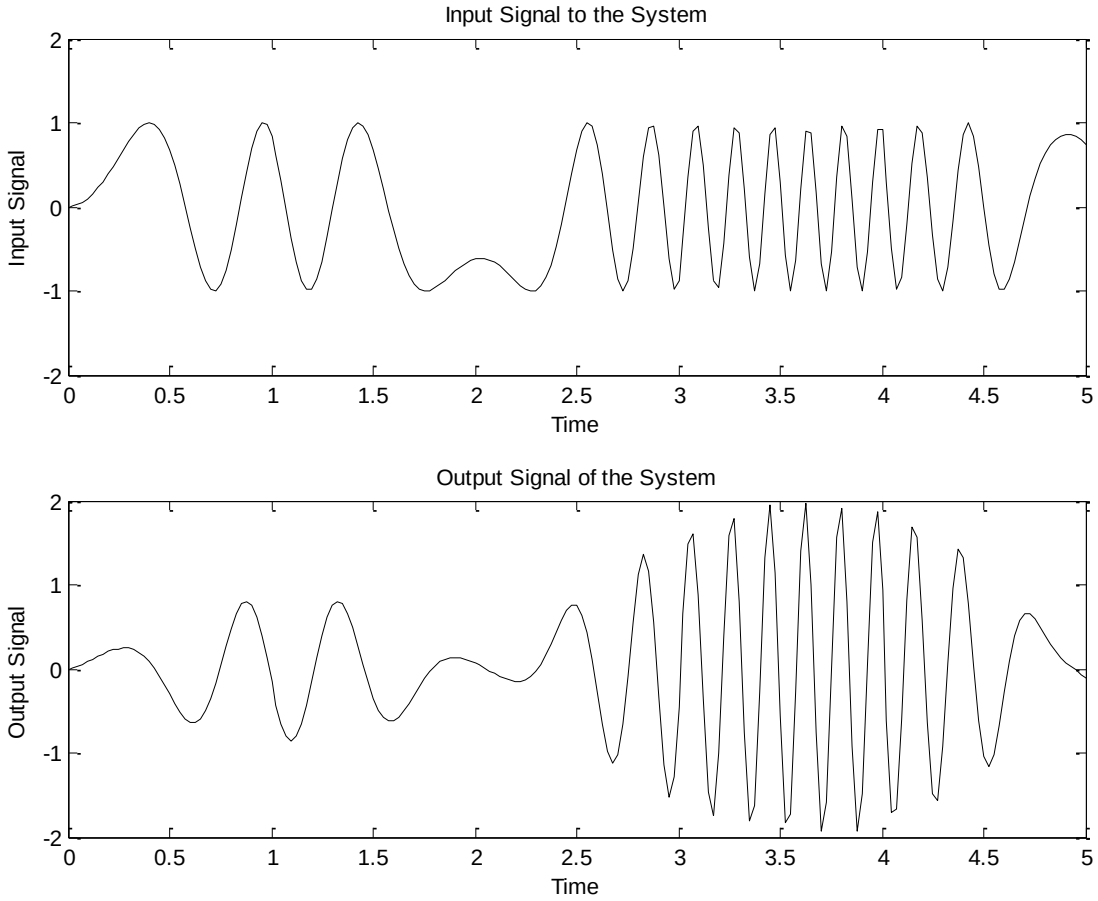


Рисунок 5.1 – Графики входного и выходного сигналов фильтра при моделировании фильтра средствами MATLAB

Задача нейронной сети – определить в процессе обучения параметры фильтра, а затем использовать их для моделирования при произвольных значениях входа.

Так как фильтр имеет один выход, то нейронная сеть должна состоять из одного нейрона. Чтобы получить и обработать одно текущее и два запаздывающих значения, нейрон должен иметь три входа.

В качестве целевого вектора $\mathbf{T}$ принимаем массив выхода $\mathbf{Y}$, а входной последовательности $\mathbf{P}$ сопоставим текущее значение и два

последующих значения входов **R**. Чтобы получить два запаздывающих значения, введем М-функцию задержки `newlind`.

Структура сети для моделирования фильтра представлена на рисунке 5.2.

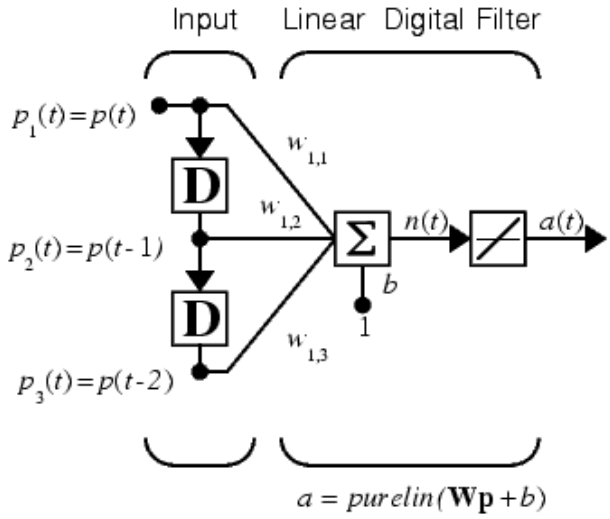


Рисунок 5.2 – Структура сети для моделирования фильтра второго порядка

Введем сигналы обучения и произведем настройку сети:

```
T=Y;
Q=size(R,2);
P=zeros(3,Q);
P(1,1:Q)=R(1,1:Q);
P(2,2:Q)=R(1,1:(Q-1));
P(3,3:Q)=R(1,1:(Q-2));
net=newlind(P,T);
net.IW{1}, net.b
ans =
    1.0000    0.5000   -1.5000
ans =
    [-3.5239e-017]
```

Для проверки функционирования сети сравним выход **a** с целевой последовательностью **T**:

```
a=sim(net,P);
figure(2),clf
subplot(2,1,1)
plot(time,T,'ok'),hold on
plot(time,a,'k'), grid on
axis([0 5 -2 2]);
```

Здесь же вычислим и построим график погрешности моделирования:

```
subplot(2,1,2)
e=T-a;
plot(time,e)
xlabel('Time, c')
grid
```

Результаты моделирования представлены на рисунке 5.3.

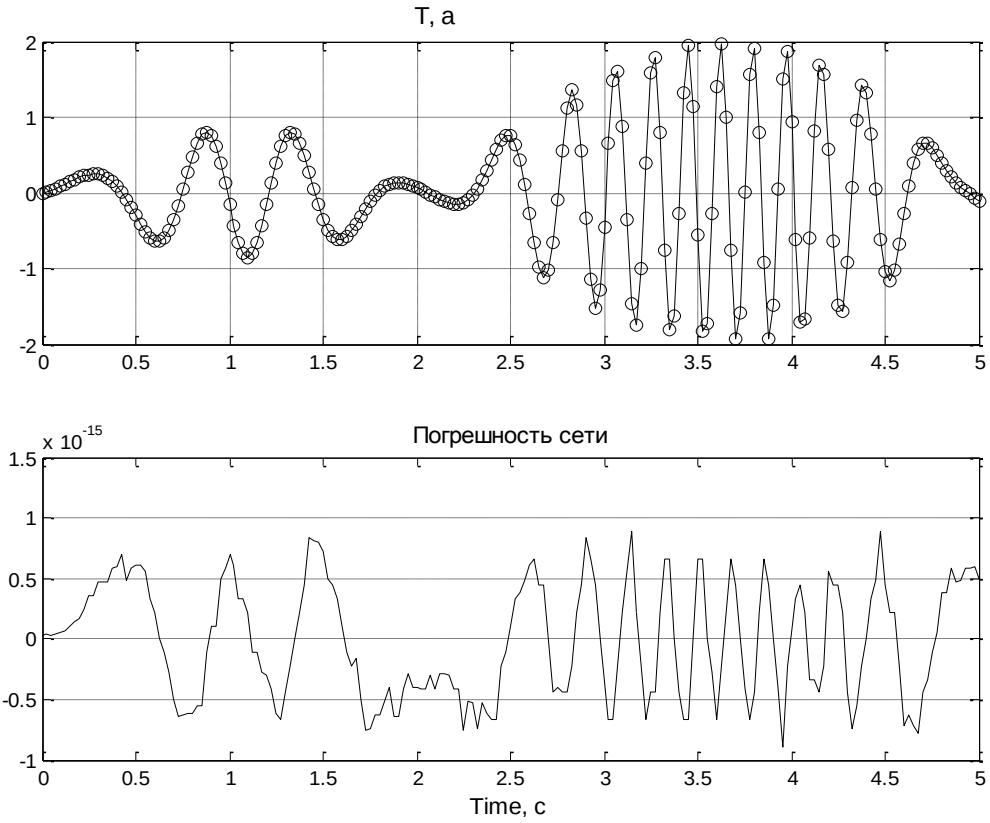


Рисунок 5.3 – Результаты моделирования фильтра сетью

Результаты моделирования показывают, что сеть обеспечивает решение поставленной задачи с высокой точностью. Погрешность моделирования находится в пределах точности вычислений (10^{-15}).

5.2 Методика выполнения работы

Для выполнения работы студенты получают индивидуальные задания, варианты которых приведены в таблице 5.1.

При решении задачи моделирования стационарного фильтра необходимо выполнить следующее:

1 Произвести моделирование фильтра средствами MATLAB.

2 Произвести синтез нейронной сети для моделирования фильтра.

3 Произвести обучение и моделирование сети.

4 Вывести графики результатов моделирования и оценить качество работы сети.

Таблица 5.1 – Варианты заданий для моделирования фильтра

Вар.	Функция входного сигнала	Вар.	Функция входного сигнала
1	$r(t)=\sin(2*\sin(t)*t)$	11	$r(t)=\sin(2*\sin(t)*t)$
2	$r(t)=\sin(3*\sin(t)*t)$	12	$r(t)=\sin(3*\sin(t)*t)$
3	$r(t)=\sin(4*\sin(t)*t)$	13	$r(t)=\sin(4*\sin(t)*t)$
4	$r(t)=\sin(5*\sin(t)*t)$	14	$r(t)=\sin(5*\sin(t)*t)$
5	$r(t)=\sin(6*\sin(t)*t)$	15	$r(t)=\sin(6*\sin(t)*t)$
6	$r(t)=\sin(7*\sin(t)*t)$	16	$r(t)=\sin(7*\sin(t)*t)$
7	$r(t)=\sin(8*\sin(t)*t)$	17	$r(t)=\sin(8*\sin(t)*t)$
8	$r(t)=\sin(9*\sin(t)*t)$	18	$r(t)=\sin(9*\sin(t)*t)$
9	$r(t)=\sin(10*\sin(t)*t)$	19	$r(t)=\sin(10*\sin(t)*t)$
10	$r(t)=\sin(12*\sin(t)*t)$	20	$r(t)=\sin(12*\sin(t)*t)$
$y[n]=\frac{1}{1+0.5z^{-1}-1.5z^{-2}}\cdot r[n]$		$y[n]=\frac{1}{1+1.5z^{-1}-3.5z^{-2}}\cdot r[n]$	

5.3 Содержание и защита отчета

Отчет по работе должен содержать задание, структуру сети, программы и результаты работы сети.

При защите отчета студент должен ответить на вопросы, касающиеся методики создания и применения нейронной сети для моделирования стационарного фильтра.

5.4 Контрольные вопросы

1. В чем заключается суть задачи моделирования стационарного фильтра?
2. Как представляются параметры цифрового фильтра?
3. Что включает в себя структура нейронной сети для моделирования стационарного фильтра?

ЛИТЕРАТУРА

1 Медведев В. С. Нейронные сети. MATLAB 6 / В. С. Медведев, В. Г. Потёмкин; Под общ. ред. к.т.н. В. Г. Потёмкина. – М.; ДИАЛОГ – МИФИ, 2002. – 496 с.

2 Комашинский В. И.Нейронные сети и их применение в системах управления и связи / В. И. Комашинский, Д. А. Смирнов. – М.: Горячая линия – Телеком, 2003. –94 с.

3 Разработка систем искусственного интеллекта: Конспект лекций (для студентов специальности “Автоматизированное управление технологическими процессами») / Сост. А. А. Сердюк. – Краматорск: ДГМА, 2006 – 60 с.

ПРИЛОЖЕНИЕ А

Демонстрационные примеры ППП NNT

Имя функции	Содержание примера
Персептороны	
`demop1`	Персептрон с двумя входами
`demop4`	Формирование входных векторов
`demop5`	Обучение с нормированной функцией настройки
`demop6`	Пример линейно неразделимых векторов
Линейные сети	
`demolin1`	Пример функционирования линейной сети
`demolin2`	Обучение линейного нейрона
`demolin4`	Задача линейной аппроксимации
`demolin5`	Задача с неполными данными
`demolin6`	Задача с линейно зависимыми данными
`demolin7`	Оценка влияния параметров скорости настройки
`demolin8`	Адаптируемый линейный слой
Радиальные базисные сети	
`demorb1`	Пример радиальной базисной сети
Рекуррентная сеть Хопфилда	
`demohop1`	Пример двумерной сети Хопфилда
Применение нейронных сетей	
`applin2`	Предсказание нестационарного сигнала
`appcr1`	Распознавание символов
`predcstr`	Управление каталитическим реактором
`narmamaglev`	Управление магнитной подушкой
`mrefrobotarm`	Управление звеном робота

Перечень всех демонстрационных примеров, включенных в пакет Neural Network Toolbox, можно получить по команде `help nndemos`.

МЕТОДИЧНІ ВКАЗІВКИ

до комп’ютерного практикуму

з дисципліни

«КОМП’ЮТЕРНІ СИСТЕМи ШТУЧНОГО ІНТЕЛЕКТУ»

для студентів спеціальності 123 “Комп'ютерна інженерія»

(Російською мовою)

Укладач Олександр Олександрович Сердюк

Редактор О. М. Болкова

Комп’ютерна верстка О. П. Ордіна

Підп. до друку . Формат 60 х 84/16.
Папір офсетний. Ум. друк. арк. 3,00. Обл.-вид. арк. 2,18.
Тираж прим. Зам. №

Видавець і виготівник
«Донбаська державна машинобудівна академія»
84313, м. Краматорськ, вул. Шкадінова, 72.
Свідоцтво про внесення до Державного реєстру
серія ДК №1633 від 24.12.03.