МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Тульский государственный университет»

Институт Высокоточных систем им. В.П. Грязева
Кафедра "Системы автоматического управления"

Утверждено на заседании кафедры
«Системы автоматического управления»
«09» декабря 2022 г., протокол № 5

Заведующий кафедрой

О.В. Горячев

Методические указания по выполнению лабораторных работ по дисциплине «Методы искусственного интеллекта в мехатронике »

основной профессиональной образовательной программы
высшего образования – программы магистратуры

по направлению подготовки
15.04.06 Мехатроника и робототехника

с направленностью (профилем
Высокоточные мехатронные и электронные системы

Форма обучения: очная

Идентификационный номер образовательной программы: 150406-03-23

Тула 2022 г.

ЛИСТ СОГЛАСОВАНИЯ

рабочей программы дисциплины (модуля)

Разработчик:

Горячев Олег Владимирович, зав. каф. САУ, д.т.н., проф.
(ФИО, должность, ученая степень, ученое звание)

Аннотация

Искусственные нейронные сети основаны на весьма простой биологической модели нервной системы, состоящей из огромного числа (10^{11}) нейронов, каждый из которых принимает взвешенную сумму входных сигналов и при определенных условиях передает сигнал другим нейронам. Количество связей нейронов в системе достигает 10^{15}.

Теория нейронных сетей возникла из исследований в области искусственного интеллекта и связана с попытками воспроизведения способности нервных биологических систем обучаться и исправлять ошибки, моделируя низкоуровневую структуру мозга. Исследования по созданию таких систем на основе высокоуровневого (символьного) моделирования процесса мышления не принесли желаемых результатов. Эта теория развивалась в течение последних пяти десятилетий и за последние пятнадцать лет нашла широкое практическое применение: в космонавтике и аэронавтике – для имитации траекторий полета и построения систем автоматического пилотирования; в военном деле – для управления оружием и слежением за целями; в электронике – для разработки систем машинного зрения и синтеза речи; в медицине – для диагностики заболеваний и конструирования протезов; в производстве – для управления технологическими процессами, роботами и т. д. Такой успех нейронных сетей объясняется тем, что была создана необходимая элементная баз для реализации нейронных сетей, а также разработаны мощные инструментальные средства для их моделирования в виде пакетов прикладных программ. К числу подобных пакетов относится пакет Neural Networks Toolbox (NNT) системы математического моделирования MATLAB 6 фирмы Math Works.

Пакет прикладных программ NNT содержит средства для построения нейронных сетей, базирующихся на поведении математического аналога нейрона. Пакет обеспечивает эффективную поддержку проектирования, обучения, анализа и моделирования множества известных типов сетей – от базовых моделей персептрона до самых современных ассоциативных и самоорганизующихся сетей. В пакете имеется более 15 таких типов. Для каждого типа архитектуры и обучающих правил имеются M-функции инициализации, обучения, адаптации, создания, моделирования, отображения, оценки и демонстрации, а также примеры применения. Обеспечена возможность генерации переносимого кода с помощью пакета Real Time Workshop, также входящего в систему MATLAB 6.

Лабораторные работы, описания которых содержатся в данном практикуме, разработаны с целью научить студентов эффективно использовать мощный инструментальный пакет NNT системы MATLAB 6 для проектирования, анализа и моделирования нейронных сетей. Лабораторные работы дают возможность познакомиться со многими типами нейронных сетей, научиться создавать, обучать и исследовать такие сети. Описательная часть лабораторной работы содержит необходимый минимум сведений по теме, практическая часть включает достаточное число заданий для приобретения навыков в использовании пакета NNT.

СОДЕРЖАНИЕ

ПАКЕТ NNT СИСТЕМЫ MATLAB И РЕШЕНИЕ ПРИКЛАДНЫХ ЗАДАЧ С ПРИМЕНЕНИЕМ НЕЙРОННЫХ СЕТЕЙ

Пакет прикладных программ NNT (The Neural Network Toolbox) расширяет возможности системы MATLAB.

Авторами нейроимитатора NNT являются профессор Г. Демус, специалист по нейросетевым технологиям, и М. Билл, проектирующий аппаратные реализации искусственных нейронных сетей.

Любой нейрон в пакете NNT характеризуется векторами весов, смещений и функцией активации. Нейроны, принимающие одинаковые входы и использующие одинаковые функции преобразования, группируются в слои.

Возможные нейросетевые парадигмы имитатора, определяемые выбранной структурой сети и используемым алгоритмом обучения, можно разделить на наблюдаемые и ненаблюдаемые.

Наблюдаемые сети обучаются производить требуемые выходы на входы примеров. Они могут моделировать и исследовать динамические системы, классифицировать зашумленные данные и решать многие другие задачи. Возможные обучающие правила и методы проектирования сетей прямого распространения: персептроны, линейные сети (Adaline), сети с обратным распространением ошибки по Левенбергу – Марквардту, радиальные базисные сети. Алгоритм обучения Левенберга – Марквардта относится к нелинейным методам оптимизации, является самым быстрым из алгоритмов обратного распространения ошибки и связан с использованием доверительных областей. При продвижении к минимуму среднеквадратичной функции ошибок в этом алгоритме анализируются промежуточные точки, которые получились бы по обычному методу градиентного спуска, и в случае улучшения модели осуществляется переход в эти точки, а в противном случае уменьшается шаг продвижения и переход не осуществляется. Этот алгоритм требует памяти порядка l^2, если l – количество весовых коэффициентов сети [8].

Рекуррентные сети используют обратное распространение ошибки для пространственных и временных образов и включают сети Элмана и Хопфилда. Сети Элмана характеризуются наличием линии задержки во входном слое и обратной связью выходного слоя со скрытым. Обучаемый вектор квантования (LVQ) – наиболее мощный метод для классификации линейно неотделимых векторов, позволяющий точно определить границы классов и зернистость классификации.

Ненаблюдаемые нейронные сети могут определять схемы классификации, адаптивные к изменениям входных данных. Поддерживаются обучающие ассоциативные правила Хебба, Кохонена, правила входящей звезды (instar) и исходящей звезды (outstar). Самоорганизующиеся сети могут обучаться определению регулярности и корреляции входных данных и адаптировать их будущие образы этим входам. Самоорганизующиеся нейронные сети включают сети встречного распространения, самоорганизующиеся сети и разделяющие карты.

Встроенные функции активации, такие как гиперболический тангенс (tansig), симметричная линейная (satlins), радиальная базисная (radbas), логистическая (logsig), конкурирующая (compet) функции и другие, могут быть дополнены самим пользователем.

Другие встроенные функции пакета NNT системы MATLAB позволяют осуществлять нормализацию данных, визуализацию весовых коэффициентов и смещений и представлять результаты анализа ошибок на диаграммах.

ВЫЧИСЛИТЕЛЬНАЯ МОДЕЛЬ НЕЙРОННОЙ СЕТИ

Пакет NNT использует специальный класс объектов **network.object**. Эти объекты представлены в пакете в виде массива записей, поля которых определяют их свойства, характеристики и параметры. Массивы записей позволяют задать вычислительную модель нейронной сети, для которой используется стандартное имя **net**, являющееся также и именем массива записей.

Архитектура нейронной сети характеризуется количеством входов, слоев, выходов, целей, смещений, а также топологией их соединения. Описание полей массива сети net приведено в табл. 1.

Таблица 1

net	Имя поля	Размер	Тип
.numInputs	Количество входов	1×1	integer
.numLayers	Количество слоев	1×1	integer
.biasConnect	Матрица связности для смещений	numLayers×1	Boolean array
.inputConnect	Матрица связности для входов	numLayers× numInputs	Boolean array
.layerConnect	Матрица связности для слоев	numLayers× numLayers	Boolean array
.outputConnect	Матрица связности для выходов	1×numLayers	Boolean array
.targetConnect	Матрица связности для целей	1×numLayers	Boolean array
.numOutputs	Количество выходов	1×1	integer
.numTargets	Количество целей	1×1	integer
.numInputDelays	Максимальное значение задержки для входов	1×1	integer
.numLayerDelays	Максимальное значение задержки для слоев	1×1	integer

Количество векторов входа сети **numInputs** следует отличать от числа элементов вектора входа. Количество входов задается целым положительным числом и указывает, как много векторов входа подано на сеть. В свою очередь количество элементов каждого входного вектора задается свойством inputs{i}.size. Любое изменение поля numInputs будет влиять на размеры матрицы связности inputConnect и массивов ячеек inputs{i}.

Количество слоев **numLayers** задается целым положительным числом. Любое его изменение будет влиять на размеры матриц связности biasConnect, inputConnect, layerConnect, outputConnect, targetConnect, а также размеры массивов весов и смещений IW, LW, b.

Одномерная матрица связности для смещений **biasConnect** является булевой матрицей размера $N_l \times 1$, где N_l – количество слоев, определяемых свойством numLayers. Наличие или отсутствие смещений в слое *i* отмечается в элементе вектора biasConnect(i) как 1 или 0 соответственно. Наличие смещения означает, что в массивах ячеек biases{i} и b{i} будут созданы структуры, задающие все характеристики смещения.

Матрица связности для входов **inputConnect** является булевой матрицей размера $N_l \times N_i$, где N_l – количество слоев, определяемых свойством numLayers, и N_i – количество входов, определяемых свойством numInputs. Наличие или отсутствие веса при связывании слоя *i* со слоем *j* отмечается в элементе матрицы inputConnect(i,j) как 1 или 0, соответсвенно. Наличие веса означает, что в массивах ячеек inputWeights{i} и IW{i} будут созданы структуры, задающие характеристики весов входа.

Матрица связности для слоев **layerConnect** является булевой матрицей размера $N_l \times N_l$, где N_l – количество слоев, определяемых свойством numLayers. Наличие или отсутствие веса в слое *i* по входу *j* отмечается в элементе матрицы layerConnect(i,j) как 1 или 0 соответственно. Наличие веса означает, что в массивах ячеек layerWeights{i} и LW{i} будут созданы структуры, задающие характеристики весов слоя.

Матрица связности для выходов **outputConnect –** одномерная булева матрица размера $1 \times N_l$, где N_l – количество слоев, определяемых свойством numLayers. Наличие или отсутствие выхода в слое *i* отмечается в элементе вектора outputConnect(i) как 1 или 0 соответственно. Наличие выхода изменяет значение свойства numOutputs и означает, что в массиве ячеек outputs{i} будет сформирована структура, задающая характеристики выхода.

Матрица связностей для целей **targetConnect –** одномерная булева матрица размера $1 \times N_l$, где N_l – количество слоев, определяемых свойством numLayers. Наличие или отсутствие целевого выхода в слое *i* отмечается в элементе вектора targetConnect(i) как 1 или 0 соответственно. Наличие цели изменяет значение свойства numTargets и означает, что в массиве ячеек targets{i} будет сформирована структура, задающая характеристики целевого выхода.

Число выходов **numOutputs,** определенное только для чтения, задается количеством единиц в матрице связности для выходов.

Число целей **numTargets,** определенное только для чтения, задается количеством единиц в матрице связности для целей.

Максимальное значение задержки для входов **numInputDelays** (только для чтения) определяет максимальное значение задержки для входных последовательностей.

Максимальное значение задержки для слоев **numLayerDelays** определяет максимальное значение задержки для всех слоев сети.

Перечень функций, используемых для инициализации, адаптации и обучения нейронной сети net, приводится в табл. 2.

Таблица 2

net	Имя поля	Состав	Тип
.initFcn	Функции инициализации	initcon |initiay initnw | initnwb |	char

		initzero	
.initParam	Параметры функции инициализации		
.adaptFcn	Функции адаптации	adaptwb | trains	char
.adaptParam	Параметры функции адаптации		
.trainFcn	Функции обучения	trainbr | trainc | traincgb | traincgf| traingda | traingdm| traingdx | trainlm| trainoss | trainr | trainrp | trainscg	char
.trainParam	Параметры функции обучения		
.performFcn	Функции оценки качества обучения	mae | mse | sse | msereg	char
.performParam	Параметры функции оценки качества обучения		

Параметры всех функций этой таблицы определяют набор параметров для используемой функции. Узнать набор таких параметров можно, применив оператор help, например:

Help(net.trainFcn).

Функции инициализации **initFcn** определяют, какая из функций инициализации будет использована для задания начальных матриц весов и векторов смещений при вызове метода init для всей сети. При изменении этого свойства параметры функции инициализации initParam будут использовать значения, соответствующие новой функции.

Функции адаптации **adaptFcn** определяют, какая из функций адаптации будет использована при вызове метода adapt.

Функции обучения **trainFcn** определяют, какая из функций обучения будет использована при вызове метода train.

Функции оценки качества обучения **performFcn** определяют, какая из функций оценки качества обучения будет использована при вызове метода train.

Пользователь может дополнить список используемых функций инициализации, адаптации, обучения и оценки качества обучения.

Элементы сети задаются с помощью массивов ячеек, которые включают структуры для описания входов, слоев, выходов, целей, смещений и весов входа. Все эти структуры определяются однотипно, поэтому в пособии рассмотрим только описание слоев.

Описание полей структуры, используемой для определения каждого слоя нейронной сети net.layers{i}, приводится в табл. 3.

Таблица 3

net	Имя поля	Тип	Размер, состав
.layers{i}	Описатель i-го слоя сети	Cell array	numLayers×1
.dimensions	Распределение нейронов по размерностям слоя	Double array	1×numdim
.distanceFcn	Функции вычисления расстояния между нейронами	Char	boxdist | dist | linkdist | mandist
.distances	Расстояния между нейронами	Double array	
.initFcn	Функции инициализации	Char	initw | initwb
.netInputFcn	Функции накопления	Char	netprod| netsum
.positions	Положения нейронов	Array	
.size	Количество нейронов	Integer	1×1
.topologyFcn	Функции топологии	Char	gridtop | hextop | randtop
.transferFcn	Функции активации	Char	compet| logsig | poslin |purelin | radbas | satlin | satlins | tansig | tribas | hardlims
.userdata	Информация пользователя	Struct	1×1
.note	текст	Char	1×var

Описатель слоев нейронной сети **layers{i}** является массивом ячеек размера $N_l \times 1$, где N_l – число слоев сети, равное numLayers, состоящим из ячеек layers{i}, каждая из которых является массивом записей для описания *i*-го слоя сети.

Вектор распределения по размерностям слоя **dimensions** позволяет описывать многомерные слои нейронов реальных геометрических размерностей: одномерные, двумерные, трехмерные. Многомерный слой размерности numdim может быть задан вектор-строкой, элементы которой указывают число нейронов по каждой размерности, тогда их произведение будет определять общее количество нейронов в многомерном слое layers{i}.**size**. Это свойство позволяет определить положение нейронов layers{i}.positions, если известна функция топологии слоя layers{i}.**topologyFcn**. При изменении этого вектора будут автоматически изменяться layers{i}.size, положение нейронов layers{i}.**positions** и расстояния между ними layers{i}.**distances**.

Функция оценки расстояния между нейронами **distanceFcn** задает функцию, используемую для вычисления расстояния между нейронами в слое *i*. При замене функции будут автоматически пересчитаны значения расстояний между нейронами слоя layers{i}. distances. Список применяемых функций оценки расстояния может быть расширен самим пользователем.

Расстояние между нейронами в *i*-ом слое определяет свойство layers{i}.**distances**.

Функция инициализации слоя **initFcn** определяет, какая функция инициализации используется для слоя *i*.

Функция накопления **netInputFcn** определяет, какая функция накопления применяется для слоя *i*.

Для построения графика расположения нейронов в многомерном слое для слоя *i* можно использовать функцию plotsom с аргументом net.layers{i}.positions.

Функция активации слоя **transferFcn** определяет функцию активации, используемую для задания нейрона в слое *i*.

Поле для записи информации пользователя для слоя *i* предусмотрено в массиве записей **userdata.** По умолчанию поле inputs{i}.userdata.**note**, предусмотренное для записи текста, содержит строку «Put your custom input information here», что означает – «Информацию разместите здесь».

Матрицы весов и векторы смещений описываются функциями, представленными в табл. 4.

Таблица 4

net	Имя поля	Тип	Размер
.IW	Массив ячеек для матриц весов входа	Cell array	numLayers× numInputs
.LW	Массив ячеек для матриц весов слоя	Cell array	numLayers× numLayers
.b	Массив ячеек для векторов смещений	Cell array	numLayers×1

Массив ячеек **IW** имеет размеры $N_l \times N_i$, где N_l – число слоев numLayers и N_i – число входов numInputs сети net, каждый элемент которого является матрицей весов, связывающе слой *i* со входом *j* сети. Структура этого массива согласована с матрицей связности inputConnect(i,j). Каждая матрица весов должна иметь число строк, равное параметру layers{i}.size, а число столбцов соответствовать параметру inputWeights{i,j}.size.

Массив ячеек **LW** имеет размеры $N_l \times N_l$, где N_l – число слоев numLayers сети net, каждый элемент которого является матрицей весов, связывающий слой *i* со слоем *j* сети. Структура этого массива согласована с матрицей связности layerConnect(i,j). Каждая матрица весов должна иметь число строк, равное параметру layers{i}.size , а число столбцов соответствовать параметру layerWeights{i,j}.size.

Вектор смещений **b** является массивом ячеек размера $N_l \times 1$, где N_l – число слоев numLayers объекта net, каждый элемент которого является вектором смещений для слоя *i* сети. Структура этого вектора согласована с вектором связности biasConnect(i,j). Длина вектора смещений для слоя *i* должна соответствовать параметру biases{i}.size. Следует отметить, что для получения информации о структуре полей инициированного объекта network можно использовать функцию fieldnames(<имя_сети>), которая отражает текущее состояние нейронной сети.

Лабораторная работа №1.

МОДЕЛИ ИСКУССТВЕННОГО НЕЙРОНА. РЕШЕНИЕ ПРОСТЫХ ЗАДАЧ В НЕЙРОСЕТЕВОМ БАЗИСЕ

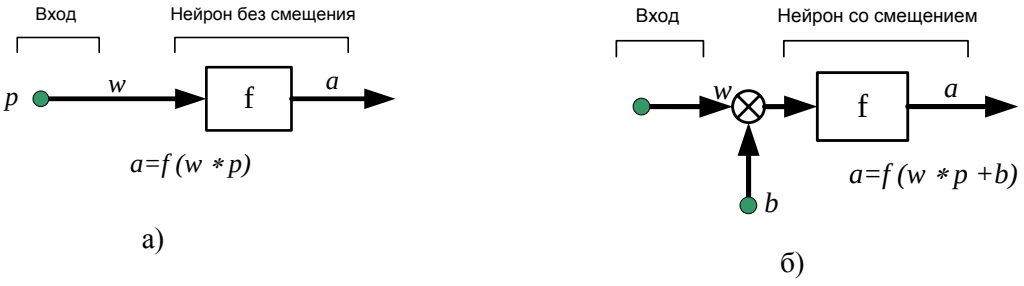
I. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Изучение основных моделей искусственного нейрона, их математического описания, а также функционального и структурного графических представлений, исследование функций активации и рассмотренных моделей нейронов с помощью инструментального пакета имитационного моделирования Simulink системы MATLAB.

II. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Простой нейрон

Элементарной ячейкой нейронной сети является нейрон. Структура нейрона с единственным скалярным входом показана на рис.1.1.



а) б)

Рис.1.1.

Скалярный входной сигнал p умножается на скалярный весовой коэффициент w, и результирующий взвешенный вход w*p является аргументом функции активации нейрона, которая порождает скалярный выход a. Нейрон, показанный на рис. 1.1,б, дополнен скалярным смещением b. Смещение суммируется со взвешенным входом w*p и приводит к сдвигу аргумента функции на величину b.

Действие смещения можно свести к схеме взвешивания, если представить что нейрон имеет второй входной сигнал со значением, равным 1. Вход n функции активации нейрона по-прежнему остается скалярным и равным сумме взвешенного входа и смещения b. Эта сумма является аргументом функции активации f; выходом функции активации является сигнал a. Константы w и b являются скалярными параметрами нейрона.

Основной принцип работы нейронной сети состоит в настройке параметров нейрона с тем, чтобы функционирование сети соответствовало некоторому желаемому поведению. Регулируя веса или параметры смещения, можно “научить” сеть выполнять конкретную работу; возможно также, что сеть сама будет корректировать свои параметры, чтобы достичь требуемого результата.

Уравнение нейрона со смещением имеет вид a = f(w*p+b*l). Как уже отмечалось, смещение b – настраиваемый скалярный параметр нейрона, который не является входом, а константа 1, которая управляет смещением, рассматривается как вход и может быть учтена в виде линейной комбинации векторов входа

$$a = \begin{bmatrix} w & \mathrm{b} \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix}.$$

Функция активации

Функции активации (передаточные функции) нейрона могут иметь самый разнообразный вид. Функция активации f, как правило, принадлежит классу сигмоидальных функций с аргументом n и выходом a.

Ниже рассмотрены три наиболее распространенные функции активации.

Единичная функция активации с жестким ограничениям hardlim

Эта функция описывается соотношением a = hardlim(n) = 1(n) и показана на рис. 1.2. Она равна 0, если n < 0, и 1,если n >= 0.

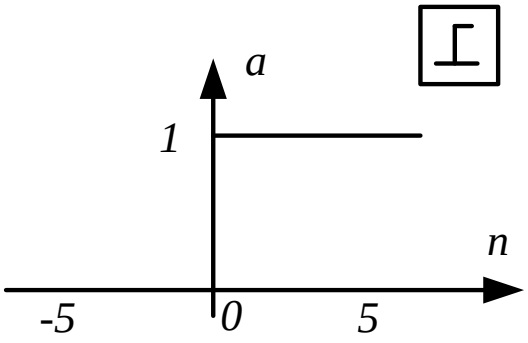


Рис.1.2.

В состав пакета ППП Neural Network Toolbox входит M-функция hardlim, реализующая функцию активации с жесткими ограничениями.

Линейная функция активации purelin.

Эта функция описывается соотношением a = purelin(n) = n и показана на рис. 1.3.

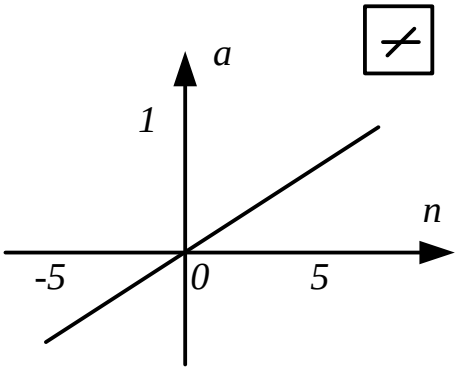


Рис.1.3.

Логистическая функция активации logsig.

Эта функция описывается соотношением a = logsig(n) = 1/(1 + exp(-n)) и показана на рис. 1.4. Она принадлежит к классу сигмоидальных функций, и ее аргумент может принимать любое значение в диапазоне от $\infty -$ до $\infty +$, а выход изменяется в диапазоне от 0 до 1. В пакете ППП Neural Network Toolbox она представлена M-функцией logsig. Благодаря свойству дифференцируемости эта функция часто используется в сетях с обучением на основе метода обратного распространения ошибки.

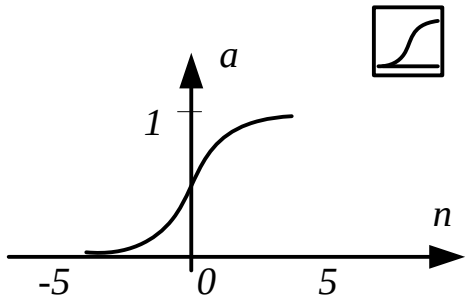


Рис.1.4.

Символ в квадрате в правом верхнем углу графика характеризует функцию активации. Это изображение используется на структурных схемах нейронных сетей. В пакет ППП Neural Network Toolbox включены и другие функции активации. Используя язык MATLAB, пользователь может создавать и свои собственные уникальные функции.

Нейрон с векторным входом

Нейрон с одним вектором входа p с R элементами $p_1, p_2, \ldots, p_R$ показан на рис.1.5. Здесь каждый элемент входа умножается на веса $w_{11}, w_{12}, \ldots, w_{1R}$ соответственно и взвешенные значения передаются на сумматор. Их сумма равна скалярному произведению вектора- строки W на вектор входа p.

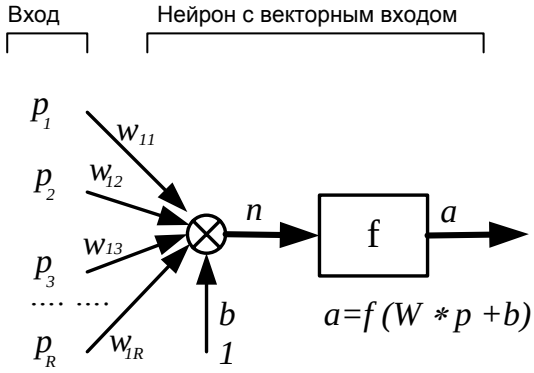


Рис.1.5.

Нейрон имеет смещение b, которое суммируется со взвешенной суммой входов. Результирующая сумма n определяется как

$$n = w_{11}p_1 + w_{12}p_2 + ... + w_{1R}p_R + b$$

и служит аргументом функции активации f. В нотации языка MATLAB это выражение записывается так:

$$n = W * \mathrm{p} + b.$$

Структура нейрона, показанная на рис. 1.5, содержит много лишних деталей. При рассмотрении сетей, состоящих из большого числа нейронов, будет использоваться укрупненная структурная схема нейрона (рис. 1.6). Вход нейрона изображается в виде темной вертикальной черты, под которой указывается количество элементов входа R. Размер вектора входа p указывается ниже символа p и равен Rxl. Вектор входа умножается на вектор-строку W длины R. Как и прежде, константа 1 рассматривается как вход, который умножается на скалярное смещение b. Входом n функции активации нейрона служит сумма смещения b и произведения W*p. Эта сумма преобразуется функцией активации f, на выходе которой получается выходная величина нейрона a, которая в данном случае является скалярной величиной. Структурная схема, приведенная на рис. 3.6, называется слоем сети. Слой характеризуется матрицей весов W, смещением b, операциями умножения W*p, суммирования и функцией активации f. Вектор входов p обычно не включается в характеристики слоя.

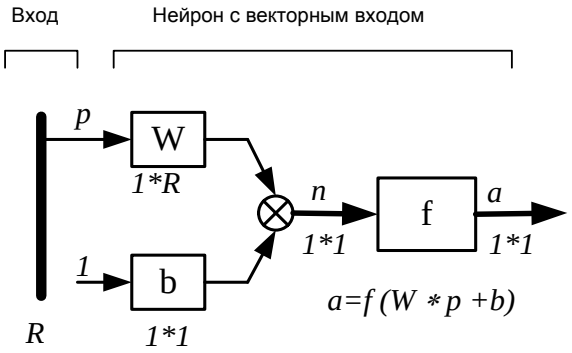


Рис.1.6.

Каждый раз, когда используется сокращенное обозначение сети, размерность матриц указывается под именами векторно-матричных переменны1. Эта система обозначений поясняет строение сети и связанную с ней матричную математику. На укрупненной структурной схеме для обозначения типа функции активации применяются специальные графические символы; некоторые из них приведены на рис. 1.7, где а – ступенчатая; б – линейная; в – логистическая функция.

hardlim
а)

purelin
б)

logsig
в)

Рис.1.6.

3. ОБОРУДОВАНИЕ

В лабораторной работе используются персональные компьютеры (не ниже Pentium 4) с установленным пакетом моделирования MatLab (версии не ниже 6.5).

4. ЗАДАНИЕ НА ВЫПОЛНЕНИЕ

Задание 1.

Для функции активации (***hardlim, hardlims, purelin, poslin, satlin, satlins, radbas, tribas, logsig, tansig***) и её производной определяемыми следующими соотношениями:

$$\mathrm{hardlim}(n) = \begin{cases} 0, \mathrm{n} < 0; \\ 1, \mathrm{n} \geq 0; \end{cases}$$

$$\mathrm{dhardlim}(n) = \begin{cases} 0, \mathrm{n} < 0; \\ 0, \mathrm{n} \geq 0, \end{cases}$$

выполнить следующие действия:

1. Вывести на экран информацию о функции с помощью следующих команд (на примере функции ***radbas*** и ***dradbas***):

```
name = radbas('name') % – полное название функции;
dname = radbas('deriv') % – название производной;
inrange = radbas('active') % – диапазон входа;
outrange = radbas('output') % – диапазон выхода;
```

2. Построить графики функции и производной:

```
n = -5:0.1:5;
a = radbas(n);
da = dradbas(n, a);
plot(n,a,'r') % – график функции активации – красный;
hold on
plot(n,da,'b') % – график производной – синий;
```

3. Рассчитать векторы выхода ***A*** и производной ***dA_dN*** для слоя из трёх нейронов с вектором входа ***N***, состоящим из трёх компонентов:

```
N = [-0.7; 0.1; 0.8];
A = radbas(N) % – вектор выхода функции актива;
dA_dN = dradbas(N,A) % – вектор выхода производной.
```

4. Оформить рассмотренную последовательность команд в виде скрипта в m-файле.

Задание 2.

Для симметричной функции активации с жёсткими ограничениями hardlims и её производной dhardlims, определяемыми соотношениями

$$\text{hardlims}(n) = \begin{cases} -1, \text{n} < 0; \\ 1, \text{n} \geq 0; \end{cases}$$

$$\text{dhardlims}(n) = \begin{cases} 0, \text{n} < 0; \\ 0, \text{n} \geq 0, \end{cases}$$

выдать на экран информацию об этих функциях, построить их графики и рассчитать векторы выхода, воспользовавшись скриптом из M-файла hardlimfile. Новый скрипт записать в файл под именем hardlimsfile.

Задание 3.

Для линейной функции активации purelin и ее производной, dpurelin, определяемыми соотношениями

purelin = n; dpurelin = 1,

выдать на экран информацию об этих функциях, построить их гра- фики и рассчитать векторы выхода, воспользовавшись скриптом из M-файла hardlimfile. Новый скрипт записать в файл под именем purelinfile.

Задание 4 .

Для положительной линейной функции активации poslin и ее производной dposlin, определяемыми соотношениями

$$poslin(n) = \begin{cases} 0, \text{n} < 0; \\ n, \text{n} \geq 0; \end{cases}$$

$$\text{d}poslin(n) = \begin{cases} 0, \text{n} < 0; \\ 1, \text{n} \geq 0, \end{cases}$$

выдать на экран информацию об этих функциях, построить их графики и рассчитать векторы выхода, воспользовавшись скриптом из M-файла hardlimfile. Новый скрипт записать в файл под именем poslinfile.

Задание 5 .

Для линейной функции активации с ограничениями satlins и ее производной dsatlins, определяемыми соотношениями

$$\text{satlins}(n) = \begin{cases} 0, \text{n} < 0; \\ \text{n}, \ 0 \leq \text{n} \leq 1; \\ \text{n}, \ \text{n} > 1, \end{cases}$$

$$\text{dsatlins}(n) = \begin{cases} 0, \text{n} < 0; \\ 1, \ 0 \le \text{n} \le 1; \\ 0, \ \text{n} > 1, \end{cases}$$

выдать на экран информацию об этих функциях, построить их графики и рассчитать векторы выхода, воспользовавшись скриптом из М-файла hardlimfile. Новый скрипт записать в файл под именем satlinfile.

Задание 6.

Для треугольной функции активации tribas и ее производной dtribas, определяемыми соотношениями

$$\text{tribas}(n) = \begin{cases} 0, \text{n} < -1; \\ 1 - \text{abs(n)}, \ -1 \le \text{n} \le 1; \\ 0, \ \text{n} > 1, \end{cases}$$

$$\text{dtribas}(n) = \begin{cases} 0, \text{n} < -1; \\ 1, \ -1 \le \text{n} \le 0; \\ -1, \ 0 < \text{n} \le 1; \\ 0, \ \text{n} > 1, \end{cases}$$

выдать на экран информацию об этих функциях, построить их графики и рассчитать векторы выхода, воспользовавшись скриптом из М-файла hardlimfile. Новый скрипт записать в файл под именем tribasfile.

Задание 7.

Для логистической функции активации logsig и ее про- изводной dlogsig определяемыми соотношениями

$$\text{logsig(n)} = 1/\left(1+\text{e}^{-\text{n}}\right);$$
$$\text{dlogsig(n)} = \text{e}^{-\text{n}}/\left(1+\text{e}^{-\text{n}}\right)^2,$$

выдать на экран информацию об этих функциях, построить их графики и рассчитать векторы выхода воспользовавшись скриптом из М-файла hardlimfile. Новый скрипт записать в файл под именем logsigfile.

Задание 8.

Для простого нейрона с одним входом **p** и функции активации **hardlim** подобрать весовой коэффициент **w** и смещение **b** таким образом, чтобы обеспечить инвертирование входного сигнала, т. е. замену нуля единицей, а единицы нулем.

Уравнение нейрона со смещением имеет вид
a = f(w*p+b)
в данном случае:

a = hardlim(w*p+b)

Для инвертирования входного сигнала значение **w** и **b** должны быть **w=-1, b=-0,5:**

```
w = -1;
b = 0;
p = 0;
a = hardlim(w*p+b);
p = 1;
a = hardlim(w*p+b);
```

Задание 9.

Для нейрона с двумя двоичными входами $\mathbf{p}_1$ и $\mathbf{p}_2$ и функцией активации **hardlim** подобрать весовые коэффициенты и смещение таким образом, чтобы нейрон выполнял функции логического сложения и логического умножения.

Логическое сложение:

P1	P2	Выход
0	0	0
0	1	1
1	0	1
1	1	1

Логическое умножение:

P1	P2	Выход
0	0	0
0	1	0
1	0	0
1	1	1

Решение:

```
% логическое сложение
w = [1, 1];
b = -0.5;

p = [0; 0]
a = hardlim(w*p + b)
p = [0; 1]
a = hardlim(w*p + b)
p = [1; 0]
a = hardlim(w*p + b)
p = [1; 1]
a = hardlim(w*p + b)

% логическое умножение
w = [1, 1];
b = -1.5;

p = [0; 0]
```

a = hardlim(w*p + b)
p = [0; 1]
a = hardlim(w*p + b)
p = [1; 0]
a = hardlim(w*p + b)
p = [1; 1]
a = hardlim(w*p + b)

Задание 10.

Для нейрона с двумя двоичными входами $\mathbf{p_1}$ и $\mathbf{p_2}$ и функцией активации **hardlim** подобрать весовые коэффициенты w11 w12 и смещение b таким образом, чтобы нейрон мог классифицировать входные двоичные наборы на два класса – нулевой и первый:

а) {00,01}- нулевой класс, {10,11}- первый класс;

б) {11}- нулевой класс, {00,01,10}- первый класс;

в) {00,10}- нулевой класс, {01,11}- первый класс;

г) {10}- первый класс, {00,01,11}- нулевой класс;

д) {00,11}- нулевой класс, {10,01}- первый класс.

Задание 11.

Для нейрона с двумя непрерывными входами p_1 и p_2 и функции активации hardlim построить график разделяющей линии, определяемой уравнением

$$\mathbf{W_{11}\, p_1 + W_{12}\, p_2 + b = 0,}$$

считая, что значения весовых коэффициентов W_{11} W_{12} и смещения b заданы. Убедиться, что наборы входов p_1 и p_2 по разную сторону от разделяющей линии принадлежат разным классам и что не всякое множество наборов значений входов можно разделить на два класса, используя нейрон рассмотренного типа.

Задание 12.

Используя блоки имитационного моделирования инструментального пакета Simulink системы MATLAB (блоки источников и регистраторов сигналов, математические и нелинейные блоки) построить рассмотренные в заданиях 1 – 12 модели нейронов, провести исследования моделей нейронов и оформить отчет.

5. ОБОРУДОВАНИЕ

Лабораторная работа выполняется на персональных компьютерах с применением пакетов моделирования SimuLink, NNT в среде MatLab

6.ОТЧЕТ

Отчет должен содержать краткое описание пакет NNT MatLab, описание математических моделей искусственных нейронов, описание активационных функций и решение заданий, представленных

Лабораторная работа №2

ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

Цель работы: изучение архитектуры искусственных нейронных сетей, способов их графического изображения в виде функциональных и структурных схем и программного представления в виде объектов специального класса network, включающих массив структур с атрибутами сети и набор необходимых методов для создания, инициализации, обучения, моделирования и визуализации сети, а также приобретение навыков построения сетей различной архитектуры с помощью инструментального программного пакета Neural Network Toolbox системы MATLAB.

Теоретические сведения

Хотя отдельные нейроны и способны после некоторой процедуры обучения решать ряд задач искусственного интеллекта, все же для эффективного решения сложных задач по распознаванию образов, идентификации и классификации объектов, распознаванию и синтезу речи, оптимальному управлению применяют достаточно большие группы нейронов, образуя из них искусственные нейронные сети в виде связанных между собой слоёв, напоминающие биологические нейронные (нервные) сети человека и животных.

Существует множество способов организации искусственных нейронных сетей, которые могут содержать различное число слоёв нейронов. Нейроны могут быть связаны между собой как внутри отдельных слоёв, так и между слоями. В зависимости от направления передачи сигнала эти связи могут быть прямыми или обратными.

Слой нейронов, непосредственно принимающий информацию из внешней среды, называется входным слоем, а слой, передающий информацию во внешнюю среду, – выходным слоем. Остальные слои, если они имеются в сети, называются промежуточными, или скрытыми. В ряде случаев такого функционального распределения слоёв сети не производится, так что входы и выходы могут присоединяться к любым слоям и иметь произвольное число компонент.

Структура, или архитектура сети искусственных нейронов зависит от той конкретной задачи, которую должна решать сеть. Она может быть однослойной без обратных связей или с обратными связями, двухслойной с прямыми связями, трехслойной с обратными связями и т. д. Сети с обратными связями называют часто рекуррентными.

Описание архитектуры искусственной нейронной сети помимо указания числа слоёв и связей между ними должно включать сведения о количестве нейронов в каждом слое, виде функций активации в каждом слое, наличии смещений для каждого слоя, наличии

компонент входных, выходных и целевых векторов, а в ряде случаев и характеристики топологии слоёв. Например, для аппроксимации любой функции с конечным числом точек разрыва широко используется сеть с прямой передачей сигналов. В этой сети имеется несколько слоёв с сигмоидальными функциями активации. Выходной слой содержит нейроны с линейными функциями активации. Данная сеть не имеет обратных связей, поэтому её называют сетью с прямой передачей сигналов (FF-net).

Графически искусственная нейронная сеть изображается в виде функциональной или структурой схемы. На функциональной схеме сети с помощью геометрических фигур изображаются её функциональные блоки, а стрелками показываются входы, выходы и связи. В блоках и на стрелках указываются необходимые обозначения и параметры.

Структурная схема сети изображается с помощью типового набора блоков, соединительных элементов и обозначений, принятых в инструментальном программном пакете Neural Network Toolbox системы MATLAB и пакете имитационного моделирования Simulink той же системы. Структурная схема сети может быть укрупнённой или

детальной, причём степень детализации определяется пользователем. Системы обозначений блоков, элементов и параметров сети является векторно-матричной, принятой в системе MATLAB. Если в обозначении используется два индекса, то, как правило, первый индекс (индекс строки) указывает адресата, или пункт назначения, а второй индекс (индекс столбца) – источник структурной схемы сети. Структурные схемы создаются системой автоматически с помощью команды gensim. Если элементом вектора или матрицы на структурной схеме является сложный объект, то используются соответственно ячейка и массив ячеек.

Программным представлением, или вычислительной моделью искусственной нейронный сети, является объект специального класса network, который включает массив структур с атрибутами сети и набор методов, необходимых для создания сети, а также для её инициализации, обучения, моделирования и визуализации. Класс Network имеет два общих конструктора, один из которых не имеет параметров и обеспечивает создание массива структур с нулевыми значениями полей, а второй – имеет минимальный набор параметров для создания модели нейронной сети, достраиваемой затем до нужной конфигурации с помощью операторов присваивания. Для создания нейронных сетей определённого вида используются специальные конструкторы.

Практические задания

Задание 1. Создать вычислительную модель нейронной сети с двумя входами, тремя слоями и одним выходом, используя общий конструктор сети с параметрами

net = network (numInputs, numLayers, biasConnect, imputConnect, layerConnect, outputConnect, tartegtConnect).

Связи между слоями должны быть только прямыми, входы необходимо соединить с первым слоем, а выход – с последним. Первый слой должен иметь смещения.
Смысл и значения параметров конструктора для создания модели сети заданной архитектуры:

numImputs =2 – количество входов сети;
numLayers =3 – количество слоёв в сети;
biasConnect =[1; 0; 0] – матрица связности для смещений размера numLayers * 1;
inputConnect =[1 1; 0 0; 0 0] – матрица связности для входов размера numLayers * numImputs;
layerConnect=[0 0 0; 1 0 0; 0 1 0] – матрица связности для слоёв размера numLayers * numLayers;
outputConnect=[0 0 1] – матрица связности для выходов размера 1* numLayers;
targetConnect=[0 0 1] – матрица связности для целей размера 1 * numLayers.

Порядок выполнения заданий следующий:

1. Создать шаблон сети, выполнив команду
net = network (2, 3, [1; 0; 0], [1 1; 0 0 ; 0 0], …. , [0 0 0 ; 1 0 0 ; 0 1 0], [0 0 1])

2. Проверить значения полей вычислительной модели нейронной сети net и их соответствие заданным значениям в списке параметров.

3. Проверить значения вычисляемых полей модели, которые дополняют описание архитектуры сети

Пример:
```
>> Net.numInputs
ans =
    2

>> Net.InputConnect
ans =
    1     1
    0     0
    0     0
```

Пример: (количество выходов сети, максимальное значение задержки для входов сети, максимальное значение задержки для слоев сети):
```
>> Net.numOutputs
ans =
    1

>> Net.numInputDelays
ans =
    0

>> Net.numLayerDelays
ans =
    0
```

numOutputs = 1 – количество выходов сети;
numInputDelays = 0 – максимальное значение задержки для входов сети.
numLayersDelays = 0 – максимальное значение задержки для слоёв сети.

Заметим, что каждый выход присоединяется к одному или нескольким слоям при этом количество компонент выхода равно количеству нейронов в соответствующем слое. Для увеличения возможности модели в сеть включают линии задержки либо на её входах, либо между слоями. Каждая линия задерживает сигнал на один такт.
Параметры numInputDelays и NumLayerDelays определяют максимальное число линий для какого-либо входа или слоя соответственно.

4. Проанализировать структурную схему построенной сети, выполнив команду **gensim(net)** и детализируя блоки с помощью двойного щелчка левой клавиши мыши по рассматриваемому блоку.

На структурных схемах искусственных нейронных сетей в пакете NNT используются следующие обозначения:

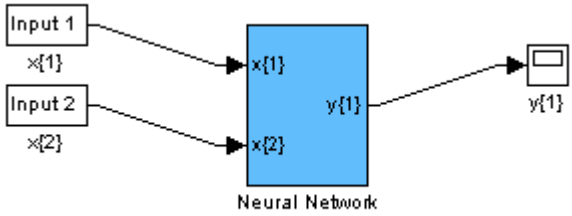


Рис.2.1. Свернутая нейросеть.

а) Neural Network – искусственная нейронная сеть с обозначениями входов p{1}, p{2}, … и выхода y{1};
б) входы Input1 , или p{1} и Input2 , или p{2};
в) дисплей y{1};

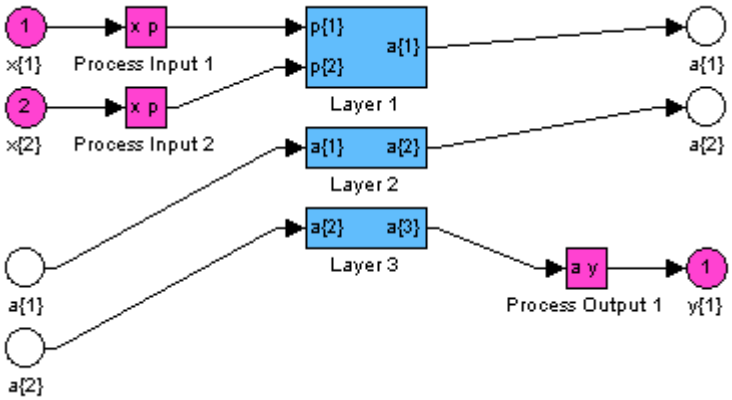


Рис.2.2. Архитектура нейросети.

г) Layer 1, Layer 2, Layer 3, … слои нейронов с обозначениями входов p{1}, p{2], a{1}, a{2}, … и выходов a{1}, a{2}, a{3}, … , y{1};

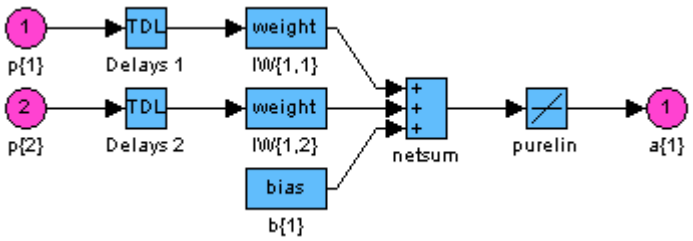


Рис.2.3. Архитектура первого слоя нейросети.

д) TDL – линии задержки (Time Delay) с именами Delays1, Delays2, ..., которые обеспечивают задержку входных сигналов или сигналов между слоями нейронов на 1, 2, 3, … такта;
е) Weights – весовая матрица для входных сигналов или сигналов между слоями нейронов; размер матрицы весов для каждого вектора входа S×R, где S – число нейронов входного слоя, а R – число компонент вектора входа, умноженное на число задержек; размер матрицы для сигналов от слоя j к слою i равен S×R, где S – число нейронов в слое i, а R – число нейронов в слое j, умноженное на число задержек;

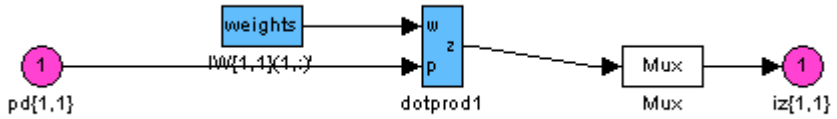


Рис.2.4.Задание весовых коэффициентов синаптических связей.

ж) dotprod – блок взвешивания входных сигналов и сигналов между слоями, на выходе которого получается сумма взвешенных, т. е. умноженных на соответствующие веса компонент сигнала;
з) mux – концентратор входных сигналов и сигналов между слоями, преобразует набор скалярных сигналов в вектор, а набор векторов в один вектор суммарной длины;
и) netsum – блок суммирования компонент для каждого нейрона слоя: компонент от нескольких векторов входа с учётом задержек, смещения и т. п.;
к) hardlim, purelin и т. д. – блоки функций активации;
л) pd{1, 1}, pd{1, 2}, ad{2, 1}, ... – сигналы после линий задержки (d - delay);
м) iz{1, 1}, iz{1, 2}, lz{2, 1}, lz{3, 2} – вектор-сигналы с выхода концентратора;
н) bias – блок весов смещений для слоя нейронов;
о) IW – массив ячеек с матрицами весов входов: IW{i, j} – матрицы для слоя i от входного вектора j;

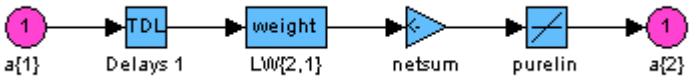


Рис.2.5. Установка параметров связей между слоями.

п) LW – массив ячеек с матрицами весов для слоёв: LW{i, j} – матрицы для слоя i от слоя j.
5. Проанализировать все параметры каждого блока структурной схемы рассматриваемой нейронной сети и в случае необходимости обратиться к справочной системе пакета NNT.
6. Задать нулевые последовательности сигналов для входов
P = [0 0 ; 0 0]
и произвести моделирование сети
A = sim(net, P).

7. Задать диапазоны входных сигналов и весовые матрицы с помощью следующих присваиваний:
net.inputs{1}.range = [0 1];
net.inputs{2}.range = [0 1];
net.b{1} = - ¼;
net.IW{1, 1} = [0.5];
net.IW{1, 2} = [0.5];

net.LW{2, 1} = [0.5];
net.LW{3, 2} = [0.5].
Исполнить команду **gensim(net)** и проверить параметры блока.

8. Вывести на экран поля вычислительной модели и их содержимое, используя функцию **celldisp.** Убедиться в правильности значений полей модели.
Пример:
celldisp(Net.IW)
celldisp(Net.LW)
celldisp(Net.b)

9. Промоделировать созданную статическую сеть, т. е. сеть без линий задержки, используя групповое и последовательное представление входных сигналов

PG = [0.5 1 ; 1 0.5];
PS = {[0.5 1] [1 0.5]};
AG = sim(net, PG);
AS = sim(net, PS).

Убедиться, что для статической сети групповое и последователь-
ное представления входных сигналов дают один и тот же результат.

10. Дополнить архитектуру созданной нейронной сети линиями задержки для входных сигналов и для сигналов между 2-м и 3-м слоями, превратив таким образом статическую сеть в динамическую:

net.inputWeights{1, 1}.delays = [0 1];
net.inputWeights{1, 2}.delays = [0 1];
net.layerWeights{3, 2}.delays = [0 1 2].

Построить структурную схему динамической сети и выяснить смысл используемых операторов присваивания.

11. Скорректировать весовые матрицы:
net.IW{1, 1} = [0.5 0.5];
net.IW{1, 2} = [0.5 0.25];
net.LW{3, 2} = [0.5 0.25 1].

12. Промоделировать динамическую сеть, используя групповое и последовательное представление входных сигналов:

AG = sim(net, PG);
AS = sim(net, PS).

Убедиться, что групповое представление входных сигналов искажает результат, так как в этом случае работа одной сети заменяется параллельной работой двух (по числу последовательностей) одинаковых сетей с нулевыми начальными значениями сигналов на выходах линий задержки.

13. Вывести на печать поля вычислительной модели и их содержимое, используя функцию celldisp.

14. Сохранить содержимое командного окна в М-файле для последующего использования.

Задание 2.

Создать точно такую же динамическую сеть **asgne**t, используя конструктор класса **network** без параметров и задавая значения соответствующих полей вычислительной модели с помощью операторов присваивания.

asgnet.numImputs =2 – количество входов сети;
asgnet.numLayers =3 – количество слоёв в сети;
asgnet.biasConnect =[1; 0; 0] – матрица связности для смещений размера numLayers * 1;
asgnet.inputConnect =[1 1; 0 0; 0 0] – матрица связности для входов размера numLayers * numImputs;
asgnet.layerConnect =[0 0 0; 1 0 0; 0 1 0] – матрица связности для слоёв размера numLayers * numLayers;
asgnet.outputConnect=[0 0 1] – матрица связности для выходов размера 1* numLayers;
asgnet.targetConnect=[0 0 1] – матрица связности для целей размера 1 * numLayers.

Убедиться в идентичности сетей net и asgnet.
Сравнить результаты работы полученных сетей.

Задание 3.

Используя конструктор класса **network** с параметрами и операторы присваивания для полей и ячеек объектов этого класса, построить, выдать на экран и промоделировать искусственные нейронные сети следующей архитектуры:

а) однослойная сеть с тремя нейронами, тремя входами и одним выходом;

б) двухслойная сеть с двумя нейронами в каждом слое, двумя входами и одним выходом.

Для выполнения задания 3(б) предлагается следующая последовательность действий:

1. Строится базовая двухслойная сеть средствами конструктора класса **network**, например с помощью следующего m-файла:

```
numInputs = 2              % количество входов сети;
numLayers = 2              % количество слоёв в сети;
biasConnect = [1; 1]       % матрица связности для смещений размера
                               numLayers * 1;
inputConnect=[1 1; 0 0]    % матрица связности для входов размера
                               numLayers * numImputs;
layerConnect=[0 0; 1 0]    % матрица связности для слоёв размера
                               numLayers * numLayers;
outputConnect=[0 1]        % матрица связности для выходов размера
                               1* numLayers;

net = network(numInputs, numLayers, biasConnect,inputConnect,layerConnect,
        outputConnect)
```

```
net.layers{1}.transferFcn = 'hardlim';
net.layers{2}.transferFcn = 'hardlim';

gensim(net);
```

2. На основе базовой структурной нейросети, выполняя последовательно операции копирования нейронов в слоях и, при необходимости, корректируя связи, формируется нейросеть требуемой структуры:

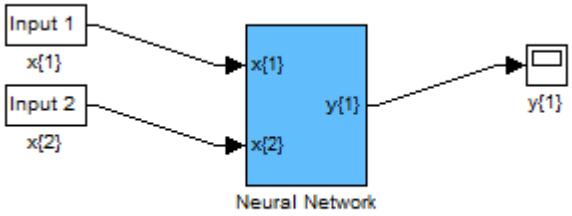


Рис. 2.6. Свернутая базовая нейросеть

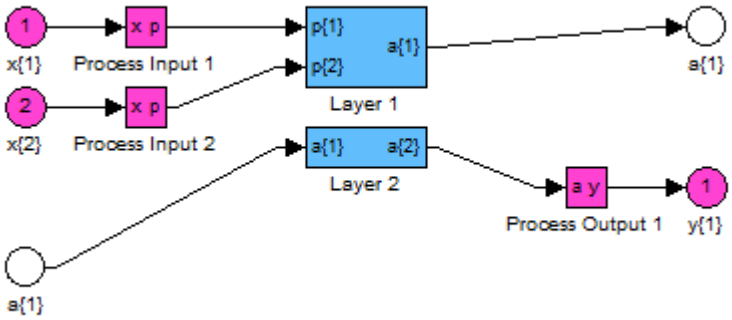


Рис. 2.7. Связи между слоями базовой нейросети

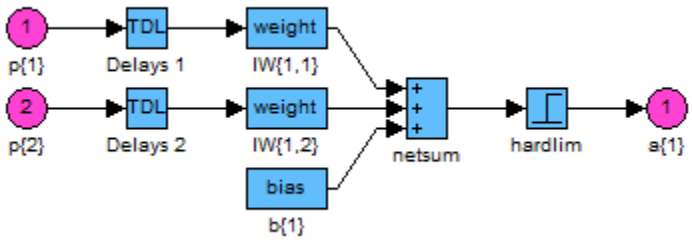


Рис. 2.8.Архитектура первого слоя базовой нейросети.

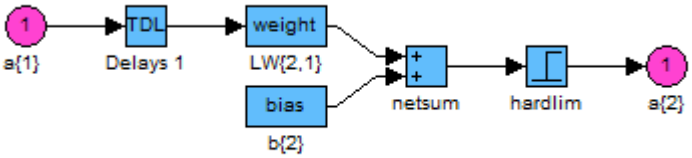


Рис. 2.9. Архитектура второго слоя базовой нейросети.

1. На основе базовой структурной нейросети, выполняя последовательно операции копирования нейронов в слоях и, при необходимости, корректируя связи, формируется нейросеть требуемой структуры:

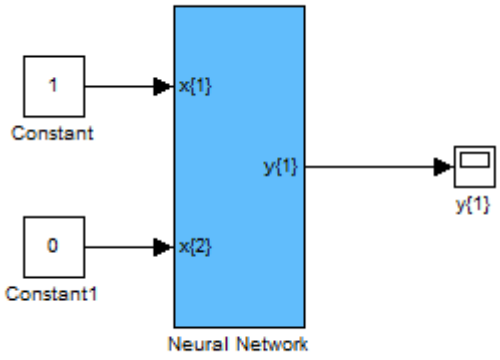


Рис. 2.10. Свернутая нейросеть, выполняющая логическую операцию XOR.

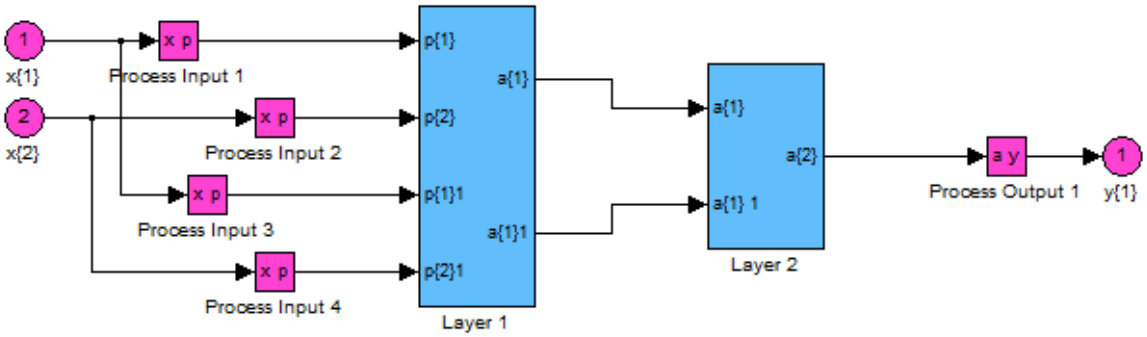


Рис. 2.11. Связи между слоями нейросети.

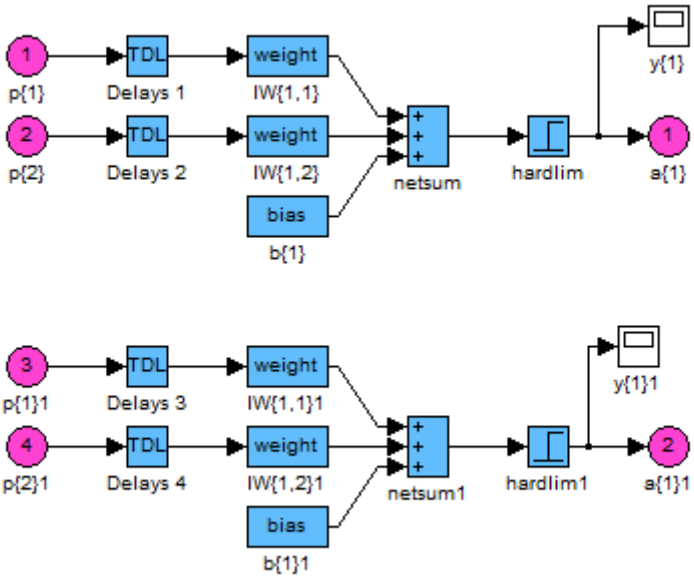


Рис. 2.12. Архитектура первого слоя нейросети

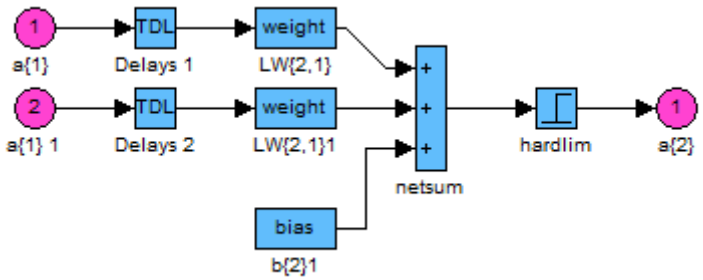


Рис. 2.13. Архитектура второго слоя нейросети.

в) настроить нейросеть на выполнение логической операции XOR для двоичных переменных x1 и x2.

г) трёхслойная сеть с прямой передачей сигналов и с тремя нейронами в каждом слое; количество входов – три с двумя, пятью и тремя компонентами; для всех слоёв имеется смещение; выход – один.

4. ОБОРУДОВАНИЕ

Лабораторная работа выполняется на персональных компьютерах с применением пакетов моделирования SimuLink, NNT в среде MatLab

5.ОТЧЕТ

Отчет должен содержать краткое описание пакет NNT MatLab, описание математических моделей искусственных нейронов, описание активационных функций и решение заданий, представленных в разделе 4 настоящего описания.

Лабораторная работа №3

ФОРМИРОВАНИЕ И ОБУЧЕНИЕ НЕЙРОСЕТИ С ПОМОЩЬЮ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ MATLAB

I. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

ознакомиться на практике с нейросетями, особенностями их применения и использованием в среде Matlab.

II. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

АЛГОРИТМЫ ОБУЧЕНИЯ НЕЙРОННЫХ СЕТЕЙ

Адаптация и самоорганизация искусственных нейронных сетей достигается в процессе их обучения. Целью *обучения* нейронных сетей является такая подстройка их весов, которая обеспечивала бы для некоторого множества входов требуемое множество выходов.

При решении прикладных задач с помощью нейронных сетей необходимо собрать представительный объем данных, для того чтобы обучить нейронную сеть решению таких задач. Обучающий набор данных – это набор наблюдений, содержащих признаки изучаемого объекта. Первоначальный выбор признаков осуществляется на основе имеющегося опыта, с учетом мнения экспертов. Вопрос о том, сколько необходимо иметь наблюдений для обучения сети, часто оказывается непростым. Известен ряд эвристических правил, которые устанавливают связь между количеством используемых для обучения наблюдений и размерами сети. Простейшее из них гласит, что количество наблюдений должно быть в 10 раз больше числа связей в сети. Процесс обучения – это процесс определения параметров модели процесса или явления, реализуемого нейронной сетью. Ошибка обучения для конкретной конфигурации сети определяется после прогона через сеть всех имеющихся наблюдений и сравнения выходных значений с целевыми значениями в случае обучения с учителем. Соответствующие разности позволяют сформировать так называемую функцию ошибок. Если ошибка сети, выходной слой которой имеет n нейронов, $e_i = y_i - t_i$ есть разность между реальным и желаемым сигналами на выходе i-го нейрона, то в качестве функций ошибок могут быть использованы следующие функции:

- сумма квадратов ошибок sse = $\sum_{i-1}^{n} e_i^2$,
- средняя квадратичная ошибка mse = $\frac{1}{n} \sum_{i-1}^{n} e_i^2$,

- регулируемая или комбинированная ошибка

  msereg = $\frac{\gamma}{n}\sum_{i=1}^{n} e_i^2 + \frac{1-\gamma}{n}\sum_{i=1}^{n} e_i^2$, где γ - параметр регуляции,

- средняя абсолютная ошибка mae = $\frac{1}{n}\sum_{i=1}^{n}|e_i|$.

При моделировании нейронных сетей с линейными функциями активации нейронов можно построить алгоритм, гарантирующий достижение абсолютного минимума ошибки обучения. Для нейронных сетей с нелинейными функциями активации в общем случае нельзя гарантировать достижение глобального минимума функции ошибки. Поверхность функции ошибок определяется как совокупность точек-значений ошибок в N+1-мерном пространстве всевозможных сочетаний весов и смещений с общим числом N. Цель обучения при геометрическом анализе или изучении поверхности ошибок состоит в том, чтобы найти на ней глобальный минимум. В случае линейной модели сети и минимизации суммы квадратов ошибок поверхность ошибок представляет собой параболоид, имеющий единственную точку минимума. В случае нелинейной модели поверхность ошибок имеет более сложное строение и может иметь локальные минимумы, плоские участки, седловые точки и длинные узкие овраги. Как правило, при обучении такой сети вычисляется градиент функции ошибок в случайно выбранной точке поверхности, а затем эта информация используется для продвижения вниз по склону. Алгоритм продвижения завершается в точке минимума, локального или глобального. По существу алгоритмы обучения нейронных сетей аналогичны алгоритмам поиска глобального экстремума функции многих переменных.

Сети с большим количеством весов позволяют воспроизводить сложные функции, но в этом случае возможно так называемое «переобучение» сети, когда ошибки обучения малы, но полученная модель имеет слабое отношение к истинной зависимости. Нейронная сеть же с небольшим количеством весов может оказаться недостаточно гибкой, чтобы смоделировать имеющуюся зависимость. Для преодоления эффекта переобучения используется механизм контрольной проверки. Часть обучающих наблюдений резервируется как контрольные наблюдения и не используется при обучении сети. Если контрольная ошибка перестает убывать или начинает расти, то это означает, что сеть слишком близко следует исходным данным и обучение следует остановить. В этом случае следует уменьшить количество нейронов или слоев, так как сеть является слишком мощной для решаемой задачи. Если же сеть имеет недостаточную мощность для воспроизведения имеющейся зависимости, явление переобучения, скорее всего, наблюдаться не будет и обе ошибки, обучения и проверки, не достигнут достаточно малого значения.

Способность сети, обученной на некотором множестве данных выдавать правильные результаты для достаточно широкого класса новых данных, в том числе и не представленных при обучении, называется *свойством обобщения* нейронной сети.

Для настройки параметров нейронных сетей широко используется также процедура *адаптации,* когда подбираются веса и смещения с использованием произвольных функций их настройки, обеспечивающие соответствие между входами и желаемыми значениями на выходе.

Методы определения экстремума функции нескольких переменных делятся на три категории – методы нулевого, первого и второго порядка:

- методы *нулевого порядка*, в которых для нахождения экстремума используется только информация о значениях функции в заданных точках;
- методы *первого порядка*, где для нахождения экстремума используется градиент функционала ошибки по настраиваемым параметрам;
- методы *второго порядка*, вычисляющие матрицу вторых производных функционала ошибки (матрицу Гессе).

Кроме перечисленных методов можно выделить также *стохастические алгоритмы* оптимизации и алгоритмы *глобальной оптимизации*, перебирающие значения аргументов функции ошибки.

В пакете NNT системы MATLAB реализованы два способа адаптации и обучения, последовательный и групповой, в зависимости от того, применяется ли последовательное или групповое представление входов и целевого вектора (массив ячеек cell и массив формата double соответственно).

Названия процедур обучения и адаптации нейроимитатора NNT системы MATLAB приводятся по мере рассмотрения соответствующих правил обучения нейронных сетей.

1. ПРАВИЛА ОБУЧЕНИЯ ХЕББА

Правила обучения Хебба относятся к ассоциативным обучающим правилам. Согласно правилу Хебба обучение сети происходит в результате усиления силы связи между одновременно активными элементами, его можно определить так:

$$w_{ij}(t+1) = w_{ij}(t) + x_i \cdot y_j,$$

где t – время, x_i, y_j – соответственно выходные значения i-го и j-го нейронов. В начальный момент предполагается, что $w_{ij}(0)=0$ для всех i и j. Правило Хебба можно использовать как при обучении с учителем, так и без него. Если в качестве выходных значений сети используются целевые значения, тогда реализуется обучение с учителем. При использовании в качестве Y реальных значений, которые получаются при подаче входных образов на вход сети, правило Хебба соответствует обучению без учителя. В этом случае коэффициенты сети в начальный момент времени инициализируются случайным образом. Обучение заканчивается после подачи всех имеющихся

входных образов на нейронную сеть. В общем случае это правило не гарантирует сходимости процедуры обучения сети.

2. ПРАВИЛА ОБУЧЕНИЯ ПЕРСЕПТРОНА

Персептрон – нейронная сеть прямой передачи сигнала с бинарными входами и бинарной пороговой функцией активации.

Правило обучения Розенблатта в общем случае является вариантом правил обучения Хебба, формирующих симметричную матрицу связей, и в тех же обозначениях имеет вид:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \cdot (t_i - y_j) \cdot x_i,$$

где η – коэффициент обучения, $0<\eta<1$, t_j – эталонные или целевые значения. Правило обучения персептрона с нормализацией используется тогда, когда имеется большой разброс в значениях компонент входного вектора. Тогда каждая компонента делится на длину вектора:

$$x_i' = \frac{x_i}{\sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}}, \qquad i=1,2,\ldots,n.$$

В нейроимитаторе NNT обучающие процедуры настройки персептрона, обычная и с нормализацией, соответственно, learnp и learnpn.

Алгоритм обучения Розенблатта включает такие шаги:

1. Весовые коэффициенты сети инициализируются случайным образом или устанавливаются нулевыми.
2. На вход сети поочередно подаются входные образы из обучающей выборки, которые затем преобразуются в выходные сигналы Y.
3. Если реакция нейронной сети y_j совпадает с эталонным значением t_j, то весовой коэффициент w_{ij} не изменяется.
4. Если выходное значение не совпадает с эталонным значением, то есть y_j не равно t_j, то производится модификация весовых коэффициентов w_{ij} по соответствующей формуле.
5. Алгоритм повторяется до тех пор, пока для всех входных образов не будет достигнуто совпадение с целевыми значениями или не перестанут изменяться весовые коэффициенты.

Линейная разделяющая поверхность, формируемая персептроном, позволяет решать ограниченный круг задач.

3. ПРАВИЛО ОБУЧЕНИЯ ВИДРОУ – ХОФФА

Правило Видроу – Хоффа используется для обучения сети, состоящей из слоя распределительных нейронов и одного выходного нейрона с линейной функцией активации. Такая сеть называется адаптивным нейронным элементом (Adaptive Linear Element) или «ADALINE». Выходное значение такой сети определяется по формуле

$$y = \sum_{i=1}^{n} w_{i1} \cdot x_i - S_0 \ ,$$

где n – число нейронов распределительного слоя, S_0 – смещение.

Правило обучения Видроу – Хоффа известно под названием дельта-правила (delta-rule). Оно предполагает минимизацию среднеквадратичной ошибки нейронной сети, которая для L входных образов определяется следующим образом:

$$E = \frac{1}{2} \cdot \sum_{k=1}^{L} \left(y^{(k)} - t^{(k)} \right)^2 ,$$

где $y^{(k)}$ и $t^{(k)}$ – выходное и целевое значения сети соответственно для k-го образа. Правило Видроу – Хоффа базируется на методе градиентного спуска в пространстве весовых коэффициентов и смещений нейронной сети. По этому методу веса и смещения изменяются с течением времени следующим образом:

$$w_{i1}(t+1) = w_{i1}(t) - \alpha \cdot \frac{\partial E(k)}{\partial w_{i1}(t)}, \quad S_0(t+1) = S_0(t) - \alpha \cdot \frac{\partial E(k)}{\partial S_0(t)}, \quad i=1,2,\ldots,n,$$

где α – скорость обучения, $0<\alpha<1$.

Вычислим частные производные:

$$\frac{\partial E(k)}{\partial w_{i1}(t)} = \frac{\partial E(k)}{\partial y^{(k)}} \cdot \frac{\partial y^{(k)}}{\partial w_{i1}(t)} = (y^{(k)} - t^{(k)}) \cdot x_i^{(k)},$$

$$\frac{\partial E(k)}{\partial \mathrm{T}(t)} = \frac{\partial E(k)}{\partial y^{(k)}} \cdot \frac{\partial y^{(k)}}{\partial S_0(t)} = -(y^{(k)} - t^{(k)}),$$

где $x_i^{(k)}$ – i-я компонента k-го образа. По дельта-правилу

$$w_{i1}(t+1) = w_{i1}(t) - \alpha \cdot (y^{(k)} - t^{(k)}) \cdot x_i^{(k)},$$

$$S_0(t+1) = S_0(t) + \alpha \cdot (y^{(k)} - t^{(k)}), \qquad i = 1,2,\ldots,n.$$

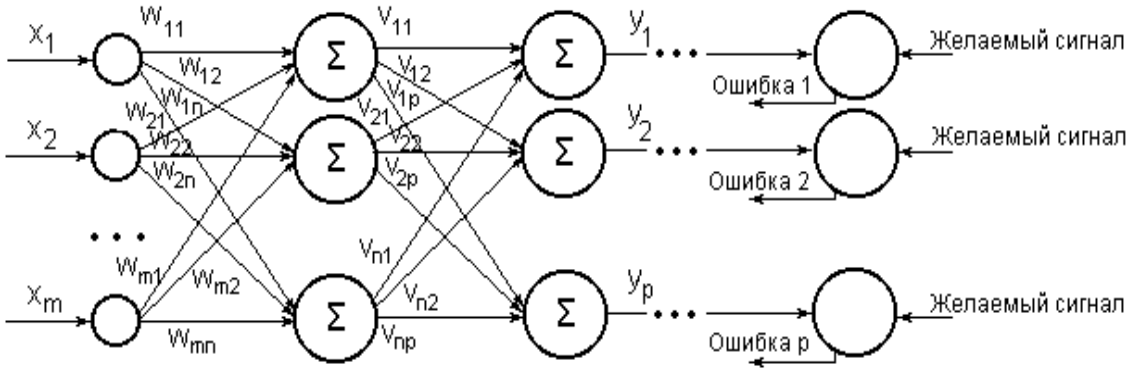
Выражение для пересчета весов сети эквивалентно правилу обучения Розенблатта, если на выходе линейного адаптивного элемента использовать пороговый элемент. Видроу и Хофф доказали, что минимизация среднеквадратичной ошибки сети производится независимо от начальных значений весовых коэффициентов. Суммарная среднеквадратичная ошибка сети E должна достигнуть заданного минимального значения E_{min}. Для алгоритма Видроу – Хоффа рекомендуется выбирать значение скорости обучения α по следующему правилу: $\alpha(l) = \frac{1}{l}$, где l – номер итерации в алгоритме обучения. Адаптивный шаг обучения для линейной нейронной сети позволяет повысить скорость обучения и определяется по формуле [2]:

$$\alpha(t) = \frac{1}{1 + \sum_{i=1}^{n} x_i^2(t)}.$$

В пакете NNT процедура настройки для обучения по правилу Видроу – Хоффа – learnwh.

4. МЕТОД ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ

Метод обратного распространения ошибки предложен несколькими авторами независимо в 1986 г. для многослойных сетей с прямой передачей сигнала. Многослойная нейронная сеть способна осуществлять любое отображение входных векторов в выходные (рис. 1).



Р и с . 1. Многослойная сеть со скрытым слоем из *n* нейронов.

Входной слой нейронной сети выполняет распределительные функции. Выходной слой нейронов служит для обработки информации от предыдущих слоев и выдачи результатов. Слои нейронных элементов, расположенные между входным и выходным слоями, называются скрытыми (hidden). Как и выходной слой, скрытые слои являются обрабатывающими. Выход каждого нейрона предыдущего слоя сети соединен синаптическими связями со всеми входами нейронных элементов следующего слоя. В качестве функций активации в многослойных сетях чаще других используются логистическая функция и гиперболический тангенс. Метод обратного распространения ошибки минимизирует среднеквадратичную ошибку нейронной сети, при этом используется метод градиентного спуска в пространстве весовых коэффициентов и смещений нейронных сетей. Согласно методу градиентного спуска по поверхности ошибок изменение весовых коэффициентов и смещений сети осуществляется по формулам:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \cdot \frac{\partial E(k)}{\partial w_{ij}(t)},$$

$$S_{0j}(t+1) = S_{0j}(t) - \alpha \cdot \frac{\partial E(k)}{\partial S_{0j}(t)}, \qquad i=1,2,\ldots,n; \quad j=1,2,\ldots,p,$$

где E – среднеквадратичная ошибка сети для одного из k образов, определяемая по формуле

$$E = \frac{1}{2}\sum_{j=1}^{p}(y_j - t_j)^2,$$

здесь t_j – желаемое или целевое выходное значение j-го нейрона.

Ошибка j-го нейрона выходного слоя равна: $\gamma_j = y_j - t_j$.

Для любого скрытого слоя ошибка i-го нейронного элемента определяется рекурсивно через ошибки нейронов следующего слоя j:

$$\gamma_i = \sum_{j=1}^{m} \gamma_j \cdot F'(S_j) \cdot w_{ij} ,$$

где m – количество нейронов следующего слоя по отношению к слою i, w_{ij} – вес или синаптическая связь между i-ым и j-ым нейронами разных слоев, S_j – взвешенная сумма j-го нейрона.

Весовые коэффициенты и смещения нейронных элементов с течением времени изменяются следующим образом:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \cdot \gamma_j \cdot F'(S_j) \cdot y_i ,$$

$$S_{0j}(t+1) = S_{0j}(t) + \alpha \cdot \gamma_j \cdot F'(S_j), \qquad i=1,2,\ldots,n, \;\; j=1,2,\ldots,p,$$

где α – скорость обучения. Это правило называется обобщенным дельта-правилом. Для логистической функции активации:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \cdot \gamma_j \cdot y_j(1-y_j) \cdot y_i$$

$$S_{0j}(t+1) = S_{0j}(t) + \alpha \cdot \gamma_j \cdot y_j(1-y_j), \;\; i=1,2,\ldots,n, \;\; j=1,2,\ldots,p,$$

ошибка j-го нейрона выходного слоя определяется как

$$\gamma_j = y_j - t_j,$$

а j-го нейрона скрытого слоя

$$\gamma_j = \sum_{i=1}^{m} \gamma_j \cdot y_i \cdot (1-y_i) \cdot w_{ij} , \text{ где}$$

m – количество нейронов следующего слоя по отношению к слою j.

К недостаткам метода обратного распространения ошибки относят следующие:

- медленную сходимость градиентного метода при постоянном шаге обучения;
- возможное смешение точек локального и глобального минимумов;
- влияние случайной инициализации весовых коэффициентов на скорость поиска минимума.

Для их преодоления предложено несколько модификаций алгоритма обратного распространения ошибки:

1) с импульсом, позволяющим учесть текущий и предыдущий градиенты (метод тяжелого шарика), изменение веса тогда:

$$\Delta w_{ij}(t+1) = \alpha \cdot \gamma_j \cdot F'(S_j) \cdot y_i + \eta \cdot \Delta w_{ij}(t) ,$$

где α – коэффициент скорости обучения, η – импульс или момент, обычно $0<\alpha<1$, $\eta \approx 0{,}9$;

2) с адаптивным шагом обучения, изменяющимся по формуле

$$\alpha(t) = \frac{1}{1 + \sum x_i^2(t)};$$

3) с модификацией по Розенбергу, предложенной им для решения задачи преобразования английского печатного текста в качественную речь:

$$\Delta w_{ij}(t+1) = (1-\alpha)\gamma_j \cdot F'(S_j) \cdot y_i + \alpha \cdot \Delta w_{ij}(t),$$

$$w_{ij}(t+1) = w_{ij}(t) + \eta \cdot \Delta w_{ij}(t+1).$$

Использование производных второго порядка для коррекции весов алгоритма обратного распространения ошибки, как показала практика, не дало значительного улучшения решений прикладных задач.

Процедуры, обеспечивающие настройку и обучение методом обратного распространения ошибки с импульсом и адаптивным шагом обучения, в пакете NNT названы соответственно learngdx и traingdx.

Для многослойных сетей прямой передачи сигнала с логистической функцией активации рекомендуется случайные начальные значения весов инициализировать по правилу (Р. Палмер)

$$w_{ij} \approx \frac{1}{\sqrt{n(i)}},$$

где $n(i)$ – количество нейронных элементов в слое i.

Начальные весовые коэффициенты рекомендуется выбирать в диапазоне [-0.05;0.05] или [-0.1;0.1]. При этом смещение S_0 принимает единичные значения в начальный момент времени. Кроме этого, количество нейронных элементов скрытых слоев должно быть меньше тренировочных образов. Для обеспечения требуемой обобщающей способности сети можно использовать сеть с несколькими скрытыми слоями, размерность которых меньше, чем для сети с одним скрытым слоем. Однако нейронные сети, имеющие несколько скрытых слоев, обучаются значительно медленнее.

3. ЗАДАНИЕ НА ВЫПОЛНЕНИЕ

сформировать и обучить нейросеть для вычисления заданной функции.

Порядок выполнения работы:

1. Запустить Matlab.

2. Ознакомиться с функциями ***newff***, ***train***, ***sim***. Использовать для этого команду help (например, “help newff”) и doc (“doc newff”).

3. Создать нейросеть командой **newff .**

Например,

net = newff ([0 25], [5 7 1], {'tansig' 'tansig' 'purelin'});

Здесь 0 и 25 - минимальное и максимальное значения входа нейросети (он здесь один), нейросеть имеет 5 нейронов в первом слое, 7 нейронов во втором слое и 1 нейрон на выходе, в качестве передаточной функции для обоих промежуточных слоев используется сигмоид, а на выходе стоит нейрон с линейной передаточной функцией.

4. Для заданной функции создать обучающие массивы P и T, где P - массив входных значений (аргумент функции), а T - массив выходных значений (результат функции).

5. Задать количество итераций обучения, например, так: net.trainParam.epochs = 150;

Здесь 150 - количество итераций (эпох) обучения нейросети.

6. Обучить нейросеть командой: net = train(net,P,T);

Посмотреть, что получилось и объяснить полученные результаты.

7. Если на шаге 6 выполнено все заданное число итераций, попробовать повторить обучение. Объяснить полученные результаты.

8. Несколько раз повторять две команды - создание сети и ее обучение. Объяснить полученные результаты.

9. Создать массив проверочных входных значений функции P1, отличающийся от P и вычислить для него значения функции (массив T1). Достаточно взять десять разных значений P1.

10. С помощью полученной нейросети предсказать, какие должны были получиться значения функции: Y = sim(net,P1);

Найти ситуации, в которых нейросеть ошибается. Пояснить, почему.

11. Сравнить реальные и предсказанные значения функции: plot(P1,T1,P1,Y,'o')

12. Сделать тоже самое для обучающих массивов: Y = sim(net,P); plot(P,T,P,Y,’o’)

13. Занести полученные графики в отчет. Объяснить увиденное.

Листинг программы, реализующей перечисленные выше шаги, приведен на рис.3.1.

```
net = newff([0 25],[5 7 1],{'tansig' 'tansig' 'purelin'});
% формирование обучающих данных
P=zeros(1, 25);
T=zeros(1, 25);
for i=0:24
    P(1, i+1) = i;
    T(1, i+1) = 2*i^1.5;
end;
plot(T)
% число итераций обучения
net.trainParam.epochs = 250;
% обучение нейросети.
net = train(net,P,T);
% формирование проверочных данных
P1=zeros(1, 25);
T1=zeros(1, 25);
for i=0:24
    P1(1, i+1) = i + 0.5;
    T1(1, i+1) = 2*(i+ 0.5)^1.5;
end;
Y = sim(net,P1);
plot(P1,T1,P1,Y,'o');
Y = sim(net,P);
figure;
plot(P,T,P,Y,'o');
```

Рис.3.1. Листинг script- функции.

В результате вызова функции **`train`** на экран выводится окно обучения нейросети (педставлено на рис.3.2) и происходит процесс настройки весов в соответствии с выбранным алгоритмом.

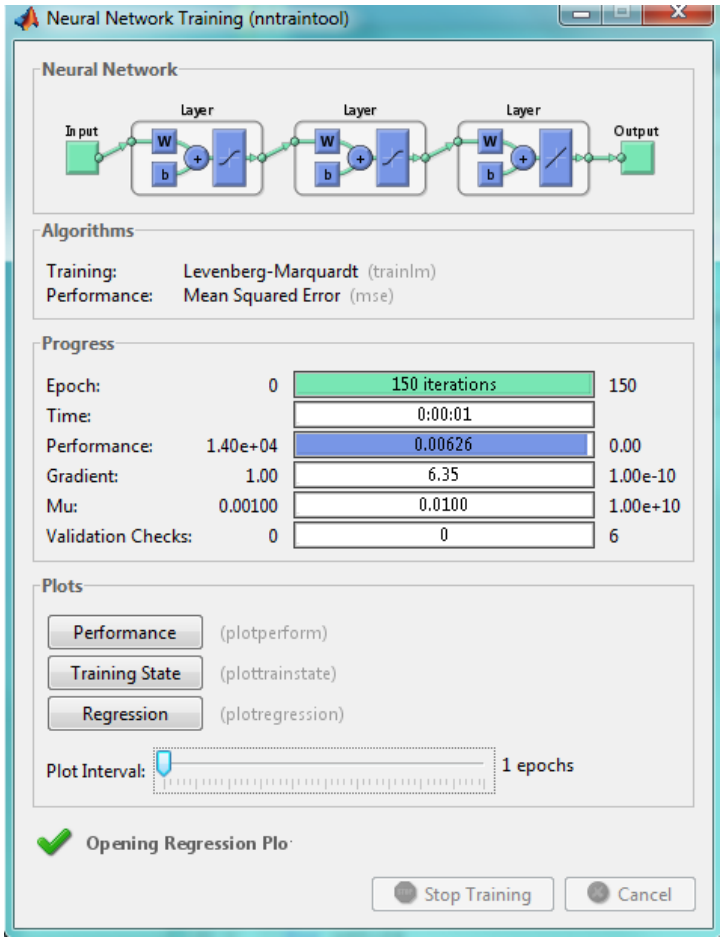


Рис.3.2.

Изменение функции качества в процессе обучения представлено на рис.3.3.

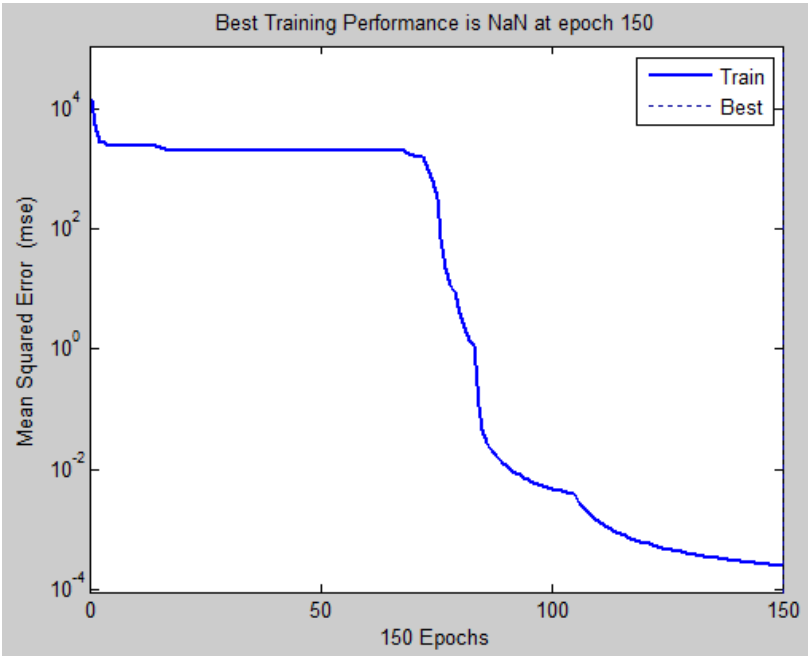


Рис.3.3.

На рис.3.4. представлены результаты расчета функции по аналитической зависимости и с помощью аппроксимации «обученной» нейросети.

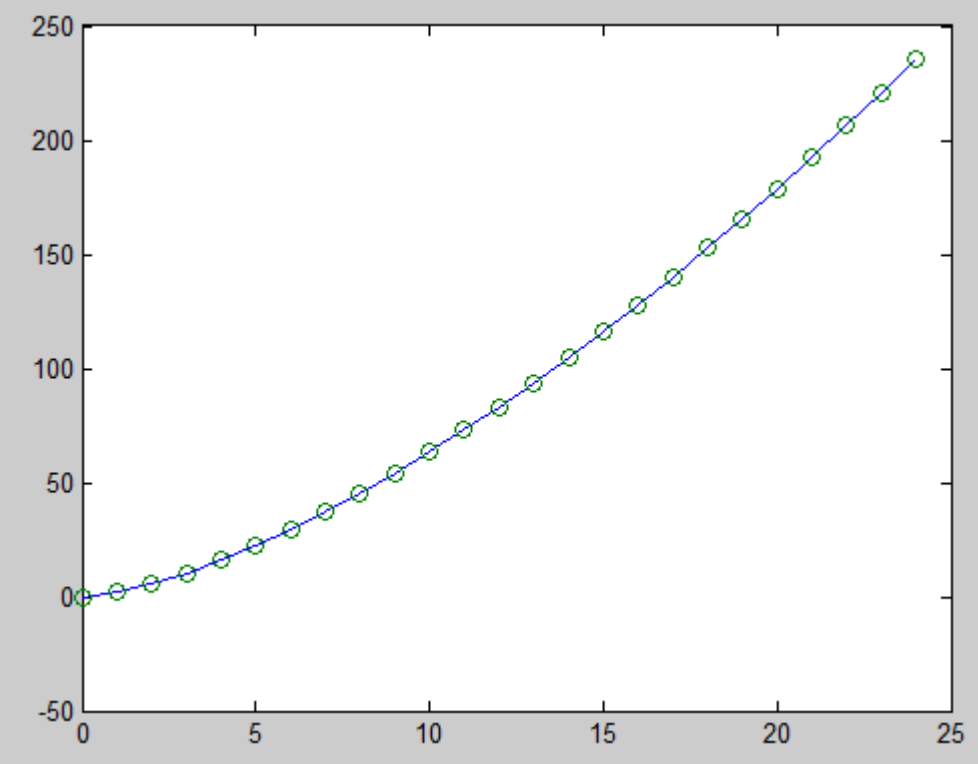


Рис.3.4.

1. Варианты аппроксимируемых функций:

1. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.4x 1.3 ; f(x) = 5, если x<4 и 8, если x>=4

2. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 2x 1.5 ; f(x) = 5, если x<3 и 10, если x>=3

3. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 3.2x 1.25 ; f(x) = 4.5, если x<3 и 8, если x>=3

4. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.72x 1.6 ; f(x) = 50, если x<7 и 78, если x>=7

5. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.5x 1.4 ; f(x) = 5.2, если x<3.45 и 9, если x>=3.45

6. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.6x 1.32 ; f(x) = 3, если x<3 и 8, если x>=3

7. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.22x 1.43 ; f(x) = 6, если x<4.2 и 2, если x>=4.2

8. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.8x 1.11 ; f(x) = 5, если x<4.1 и 3, если x>=4.1

9. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.23x 1.37 ; f(x) = 7.5, если x<9 и 1, если x>=9

10. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.45x 1.35 ; f(x) = 0.5, если x<42 и 80, если x>=42

2. Варианты функций активации:

tansig – гиперболический тангенс;

tribas – треугольная функция активации;

radbas – радиальная базисная функция активации;

logsig – сигмоидальная функция активации.

3. Количество итераций обучения:

100, 200, 300

Порядок выполнения работы.

1. Написать script – файл, реализующий процесс формирования и обучения нейросети по представленному выше алгоритму.
2. Провести процедуру обучения нейросети для трех различных функций, заданных преподавателем.
3. Вывести на экран структуру сформированной нейросети с помощью команды gensim(net). Проанализировать структуру нейросети.
4. Провести анализ влияния количества итераций обучения на качество аппроксимации функции.
5. Рассмотреть п.3 для различных функций активации.

Составить отчет. В отчете необходимо сделать выводы относительно полученных результатов.

5. ОБОРУДОВАНИЕ

Лабораторная работа выполняется на персональных компьютерах с применением пакетов моделирования SimuLink, NNT в среде MatLab

6.ОТЧЕТ

Отчет должен содержать краткое описание пакет NNT MatLab, описание математических моделей искусственных нейронов, описание активационных функций и решение заданий, представленных в разделе 4 настоящего описания.

Лабораторная работа №4

СОЗДАНИЯ И ОБУЧЕНИЯ НЕЙРОСЕТИ ЗАДАННОЙ СТРУКТУРЫ С ПОМОЩЬЮ ИНСТРУМЕНТА NETWORK/DATA MANAGER

1. ЦЕЛЬ РАБОТЫ

Знакомство с решением практической задачи создания и обучения нейросети заданной структуры с помощью инструмента Network/Data Manager.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

См.лекции.

3. ЗАДАНИЕ НА ВЫПОЛНЕНИЕ.

Требуется сформировать нейросеть, имеющую два слоя (при использовании этого инструмента имеется ограничение на большее число слоев), 10 нейронов в первом слое с тангенциальными функциями активации (передаточными функциями). Нейросеть должна реализовывать нелинейную функцию:

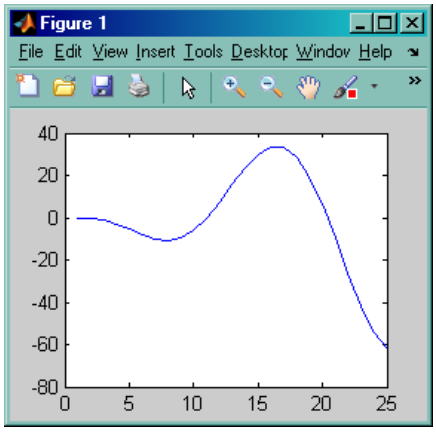
$F(x) = \mathrm{Sin}((x-10)/3)*x^{1.3}$ в диапазоне x=0..25.

Алгоритм решения задачи

1 шаг. Формируем обучающие и проверочные данные для заданной функции. С этой целью можно создать и выполнить следующий script – файл (рис.4.1):

```
P=zeros(1, 25);
T=zeros(1, 25);
for i=0:24
   P(1, i+1) = i;
   T(1, i+1) = sin((i-10)/3)*i^1.3;
end;

P1=zeros(1, 25);
T1=zeros(1, 25);
for i=0:24
   P1(1, i+1) = i + 0.5;
   T1(1, i+1) = sin(((i+0.5)-10)/3)*(i+0.5)^1.3;
```



end;

Рис.4.1.

Рис.4.2.

График функции, построенный для диапазона изменения аргумента 0..24 представлен на рис.4.2.

По окончании данного шага (по результатам работы script – функции) в рабочей области MatLab будут сформированы массивы обучающих и проверочных входов и целей (выходов):

P, P1, T, T1.

2 шаг. Вызвать инструмент Network/Data Manager, набрав в командной строке матлаба nntool.

Появится окно (рис.4.3):

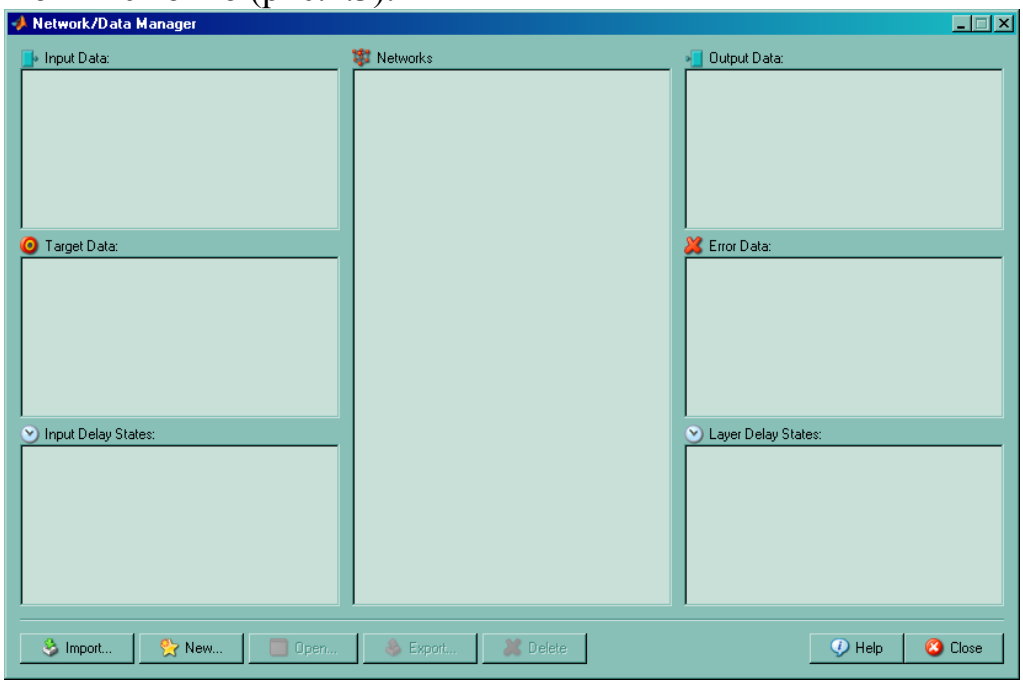


Рис.4.3.

2. добавить обучающие и проверочные данные в окна Input Data и Target Data с помощью кнопки **Import**, импортировав их из переменных MatLab (рис.4.4.).

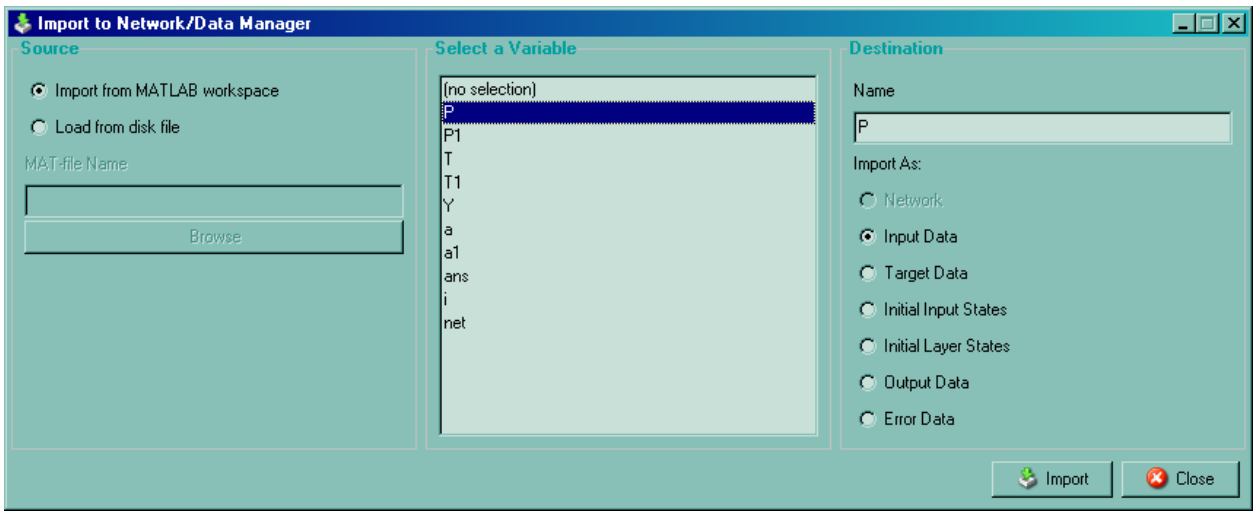


Рис.4.4.

После этого массивы обучающих и проверочных входов и целей (выходов) (P, P1, T, T1) должны оказаться в окнах Input Data и Target Data (рис.4.5):

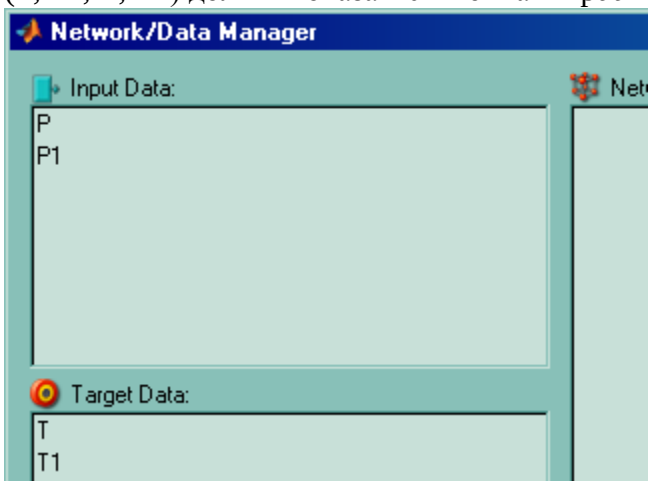


Рис.4.5.

3 шаг. Создание нейросети заданной структуры. С этой целью необходимо «нажать» кнопку **New** (рис.4.6).

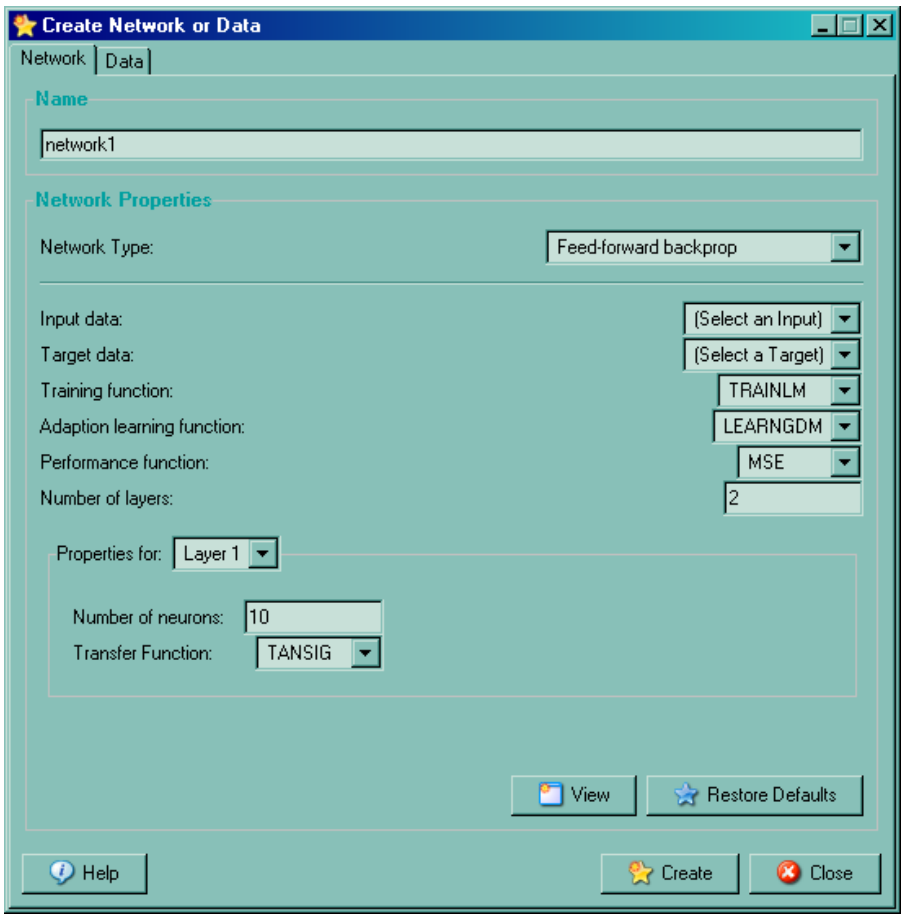


Рис.4.6.

Здесь необходимо указать параметры сети.

Network Type – тип используемой сети.

Input Data, Target Data – обучающие данные.

Training Function – обучающий алгоритм

Perfomance Function – Функция определяющая качество обучения (отклонение выхода нейросети от выходных данных обучающей выборки)

Number of layers – число слоев

Proprties for, Number Neurons, Transfer Function – параметры для каждого слоя, число нейронов и передаточная функция.

В соответствии с заданием, выберем 2 слоя, 10 нейронов в первом слое и тангенциальные передаточные функции.

Необходимо задать имена соответствующих переменных в окнах “Input data” и “Target data”.

4 шаг. Контроль структуры сформированной нейросети.

Нажав кнопку **View** можно посмотреть структурную схему создаваемой сети (рис.4.7):

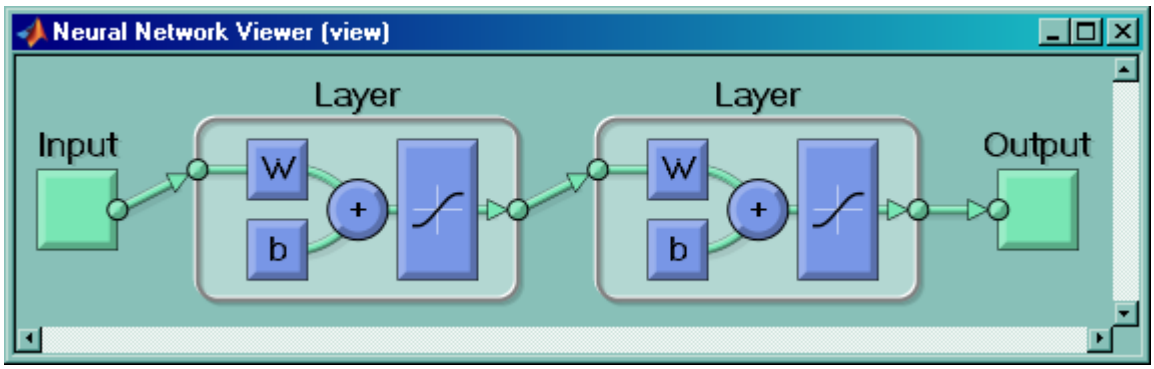


Рис.4.7.

При нажатии кнопки **Create** сеть создается и добавляется в окно Networks (рис.4.8):

Рис.4.8.

Двойной щелчек левой клавишей мыши по созданной сети вызывает окно работы с сетью, позволяющей просматривать ее структуру, обучать, симулировать, адаптировать и редактировать веса нейронов (рис.4.9):

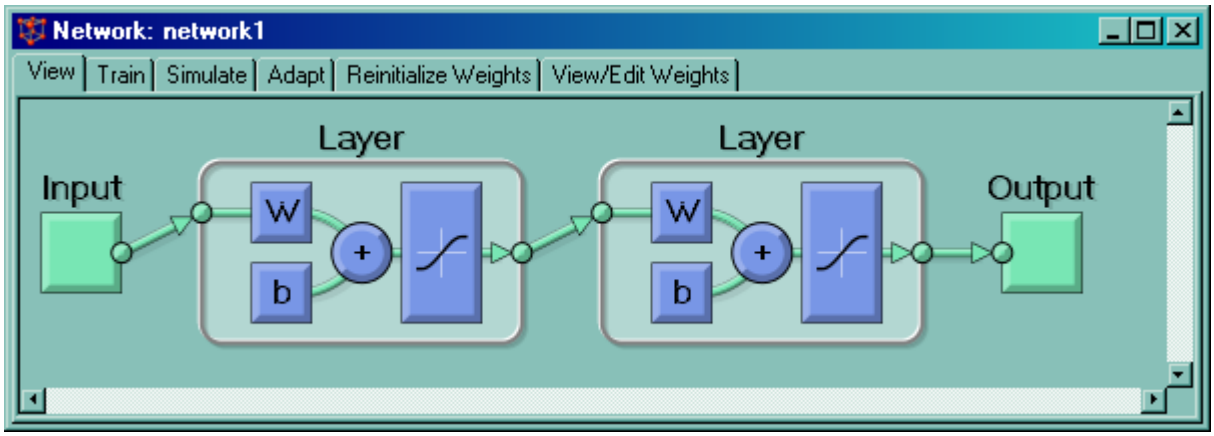


Рис.4.9.

5 шаг. Обучение нейросети.

Перейдем на закладку **Train** для обучения нейросети. В качестве обучающих данных укажем массивы P и T (рис.4.10):

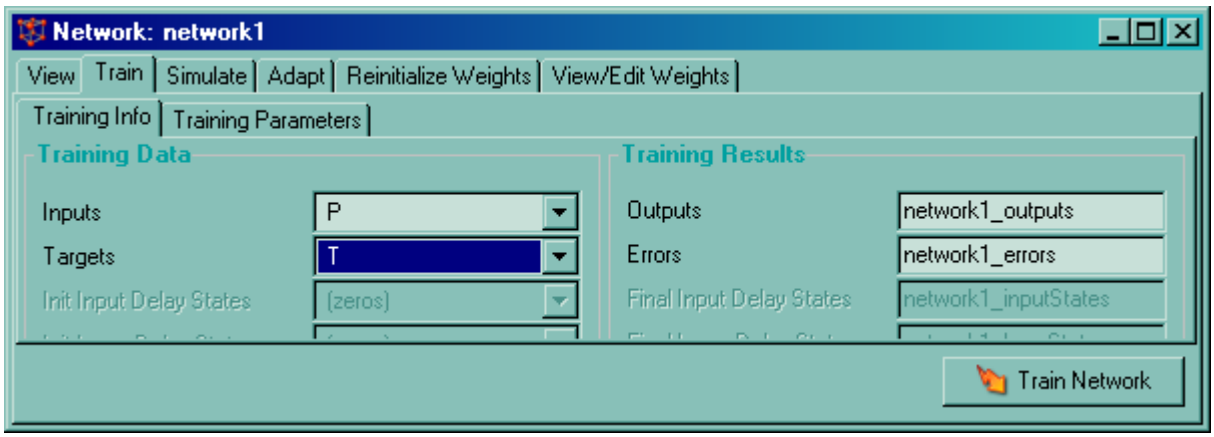


Рис.4.10.

На закладке Train Parameters можно редактировать параметры обучения.
Увеличим параметр max_fail до 100 (рис.4.11).

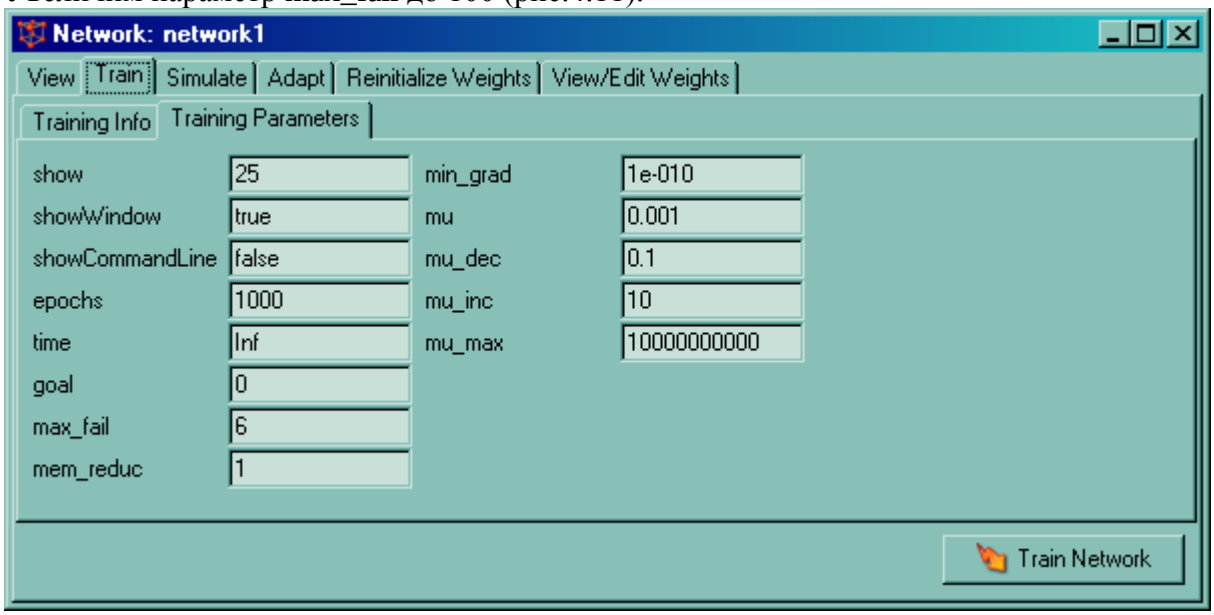


Рис.4.11.

6. При нажатии кнопки **Train Network** открывается окно обучения нейросети и происходит процесс настройки весов в соответствии с выбранным алгоритмом (рис.4.12):

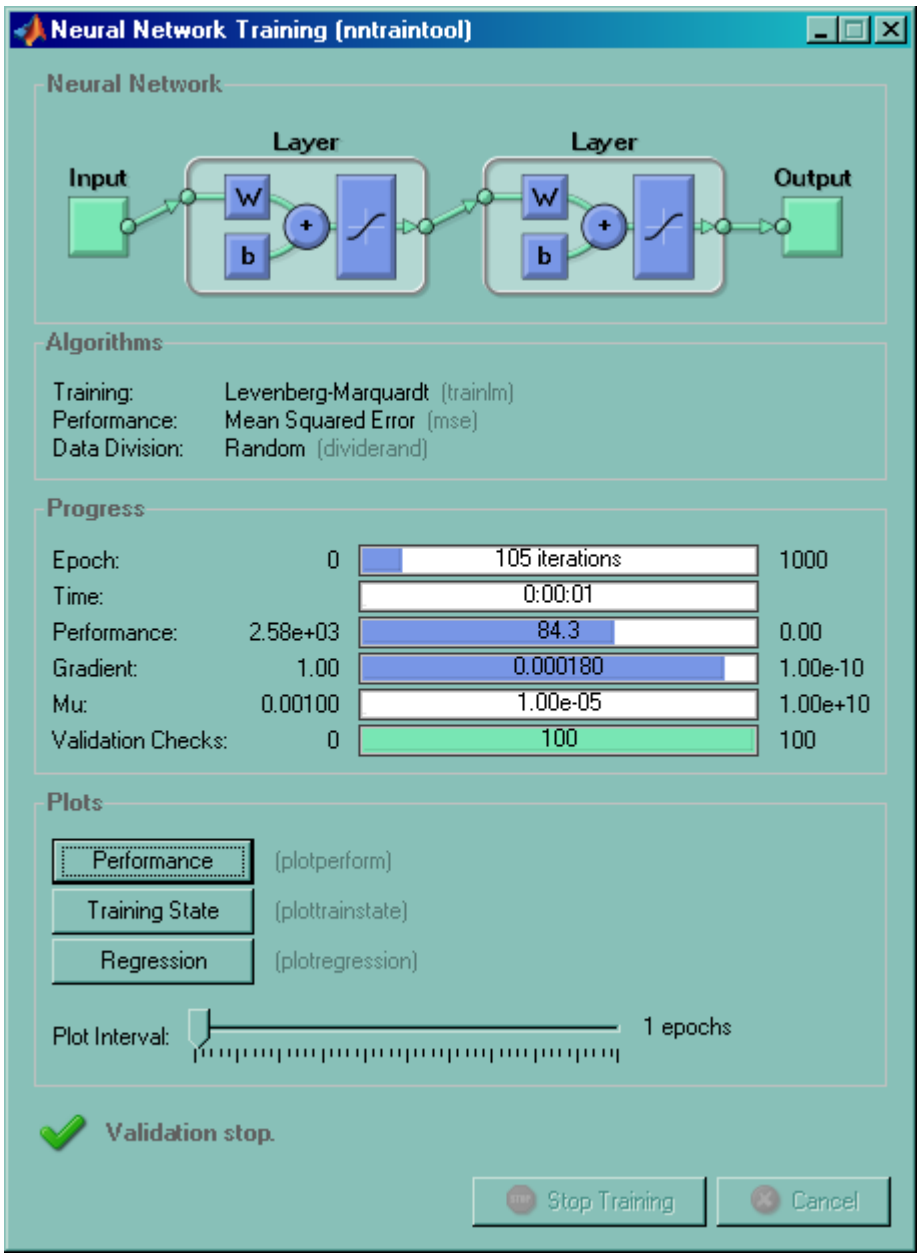


Рис.4.12.

При нажатии кнопки **Perfomance** можно посмотреть изменение функции качества обучения (рис.4.13):

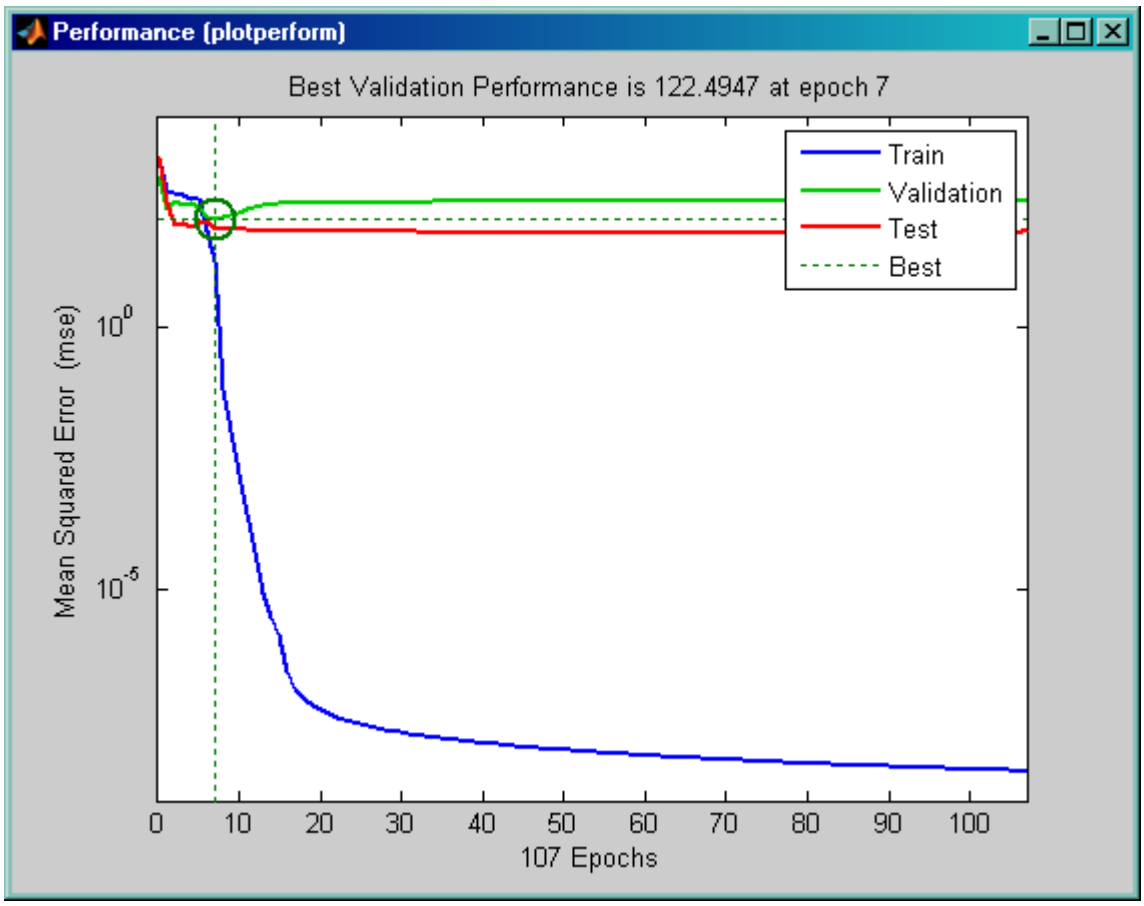


Рис.4.13.

7. После обучения проведем моделирование работы сети и получим выходы Y и Y1 сети для обучающей и проверочной выборок (рис.4.14):

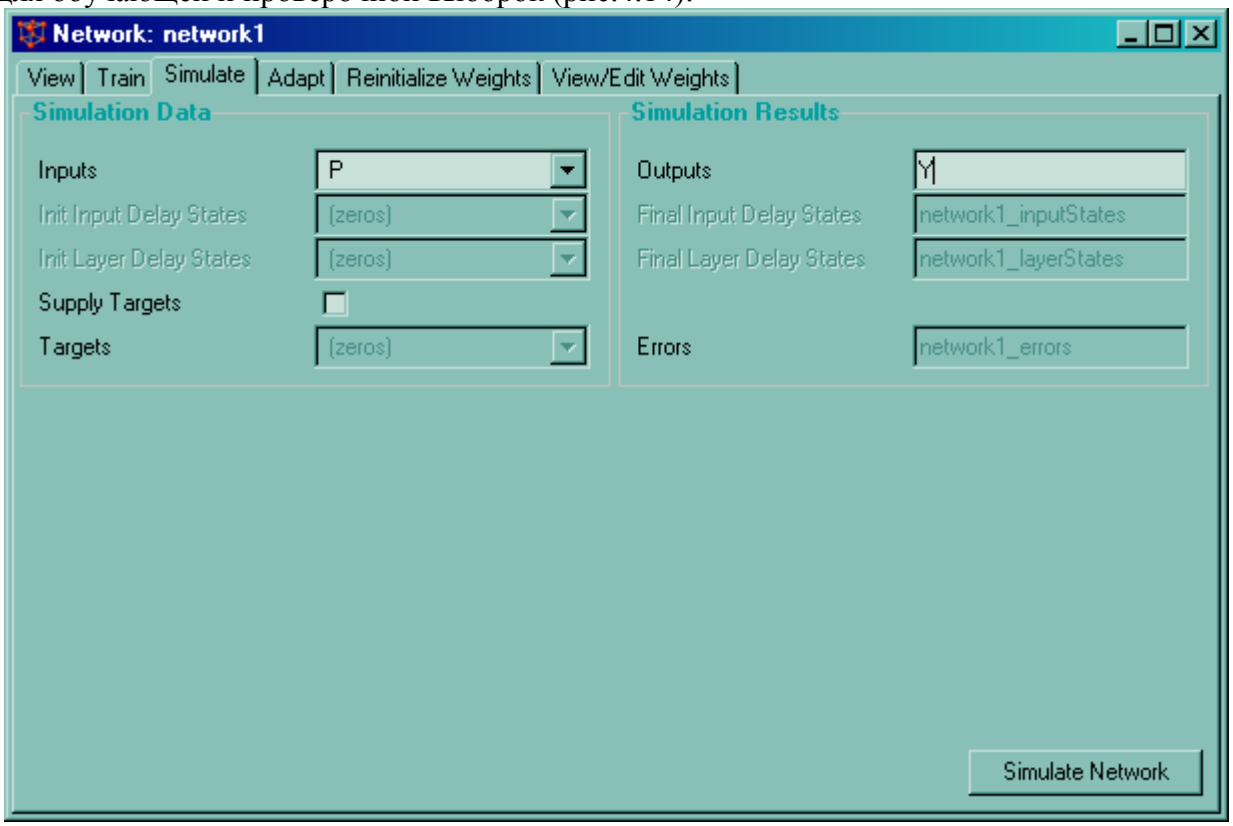


Рис.4.14.

Кнопка Simulate запускает моделирование сети.

После этого сформированные в результате моделирования выходы сети Y и Y1 окажутся в окне Output Data (рис.4.15).

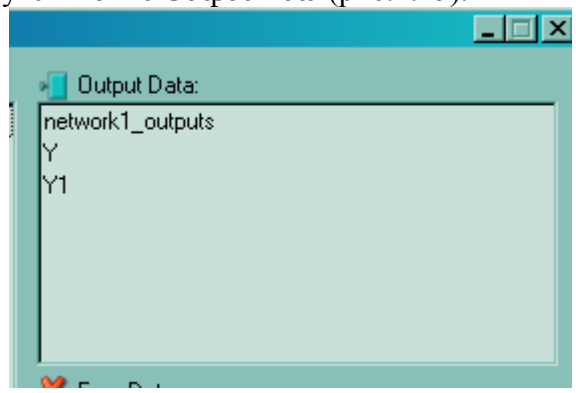


Рис.4.15.

8. Экспортируем данные Y и Y1 в Workspace матлаба нажатием кнопки **Export** (рис.4.16**):**

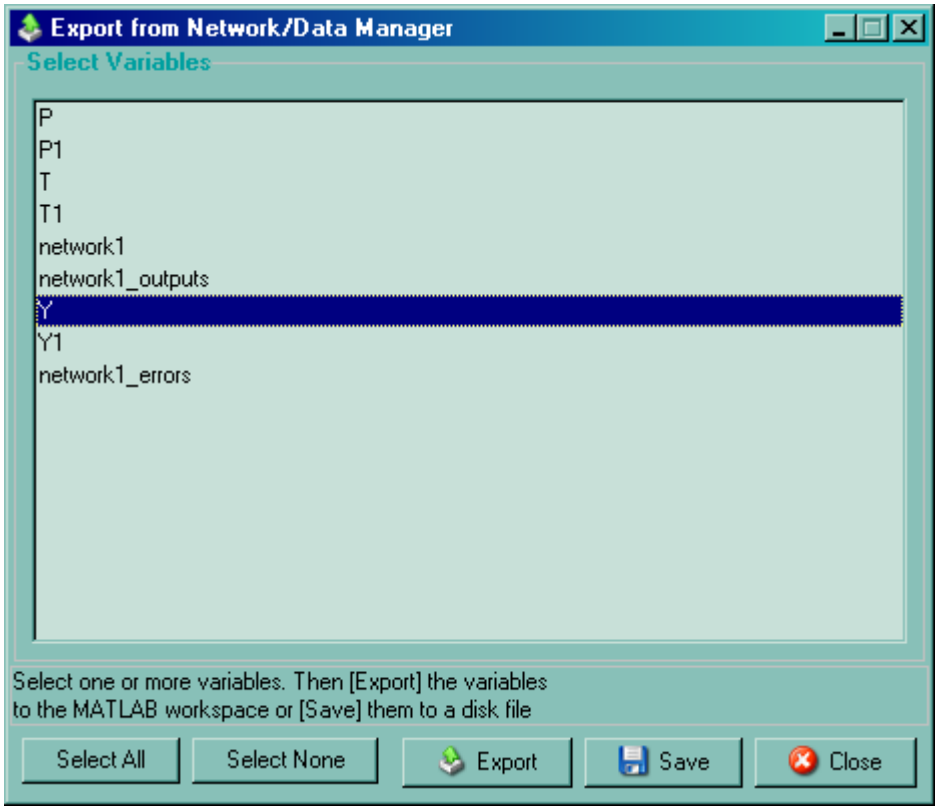


Рис.4.16.

9. С помощью команд

>> plot(P,T,P,Y,'o');
>> plot(P1,T1,P1,Y1,'o');

Можно построить графики работы нейросети с обучающими и проверочными данными (рис.4.17 а,б соответственно):

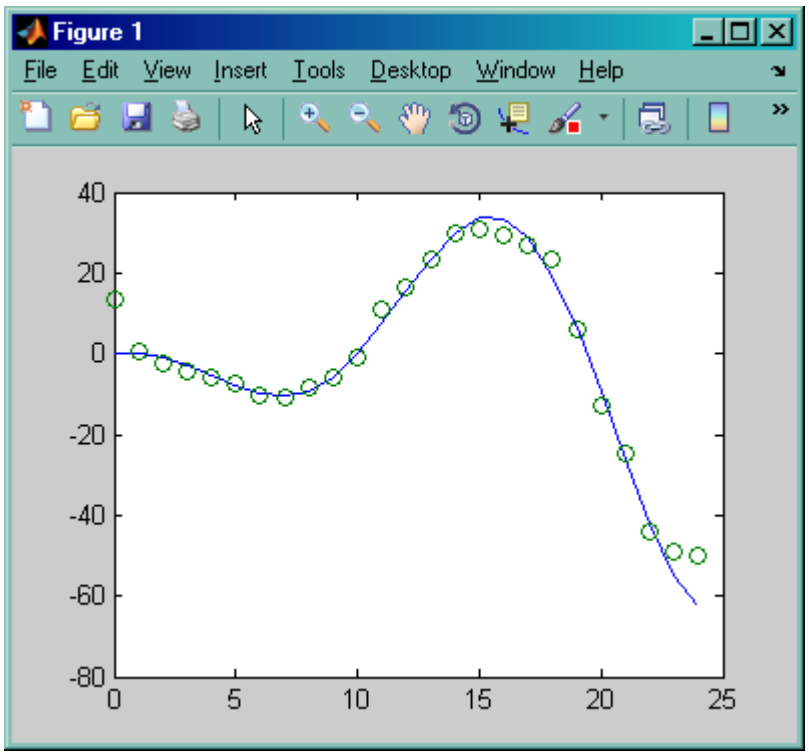


Рис.4.17.а

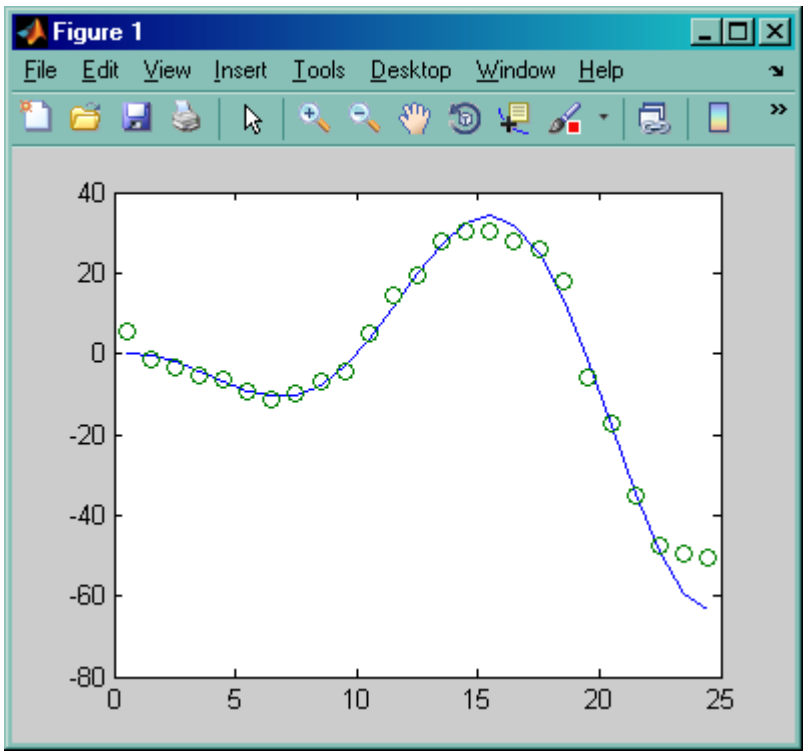


Рис.4.17.б

10. С помощью команды Export экспортируем в Workspase и саму обученную нейросет network1. Далее построим модель этой сети в SimuLink:

>> gensim(network1, -1)

(второй параметр здесь – sample time или такт квантования генерируемой модели сети.

-1 – для непрерывной модели.)

Структура модели сети в **Simulink** (рис.4.18, 4.19):

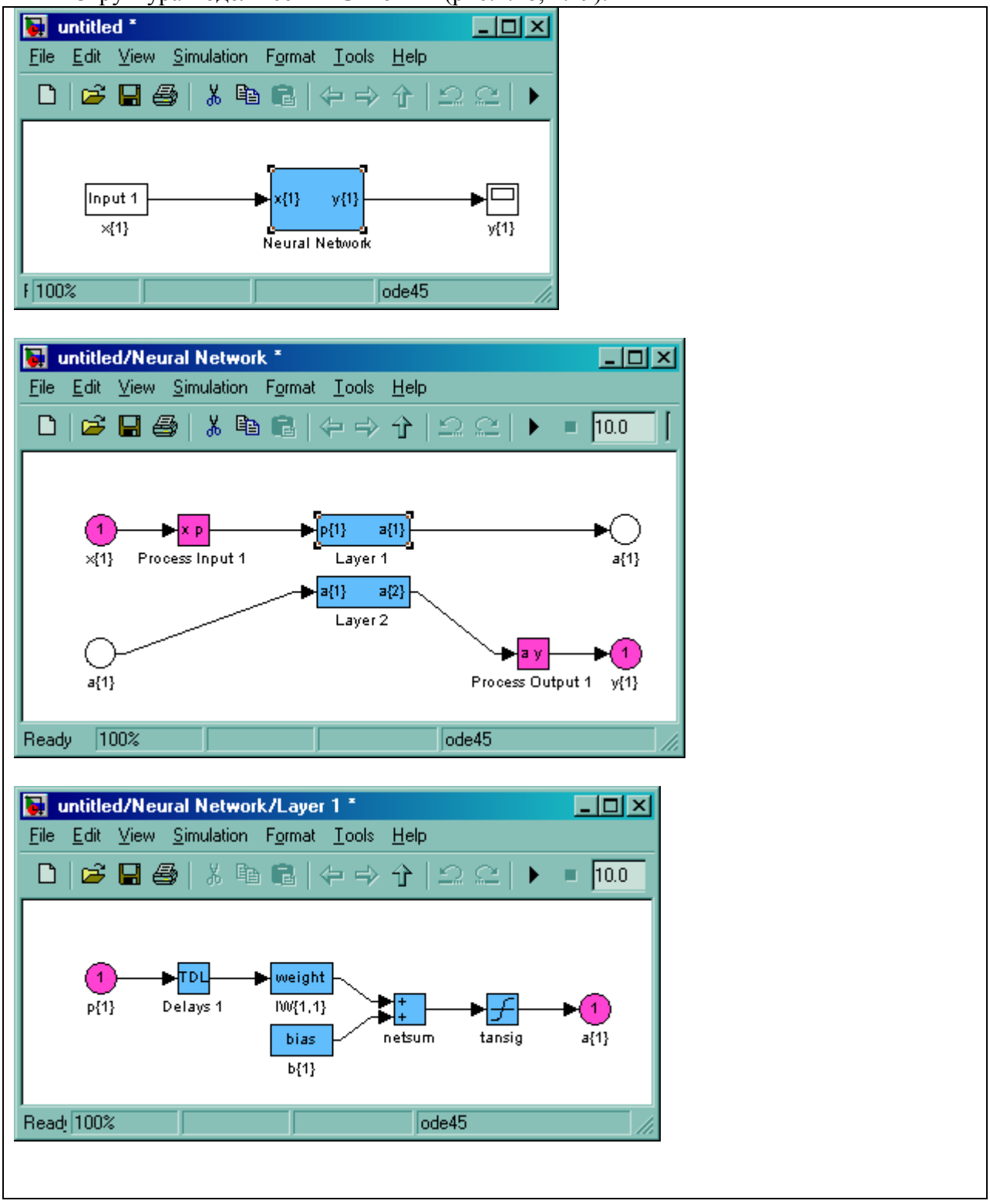


Рис.4.18.

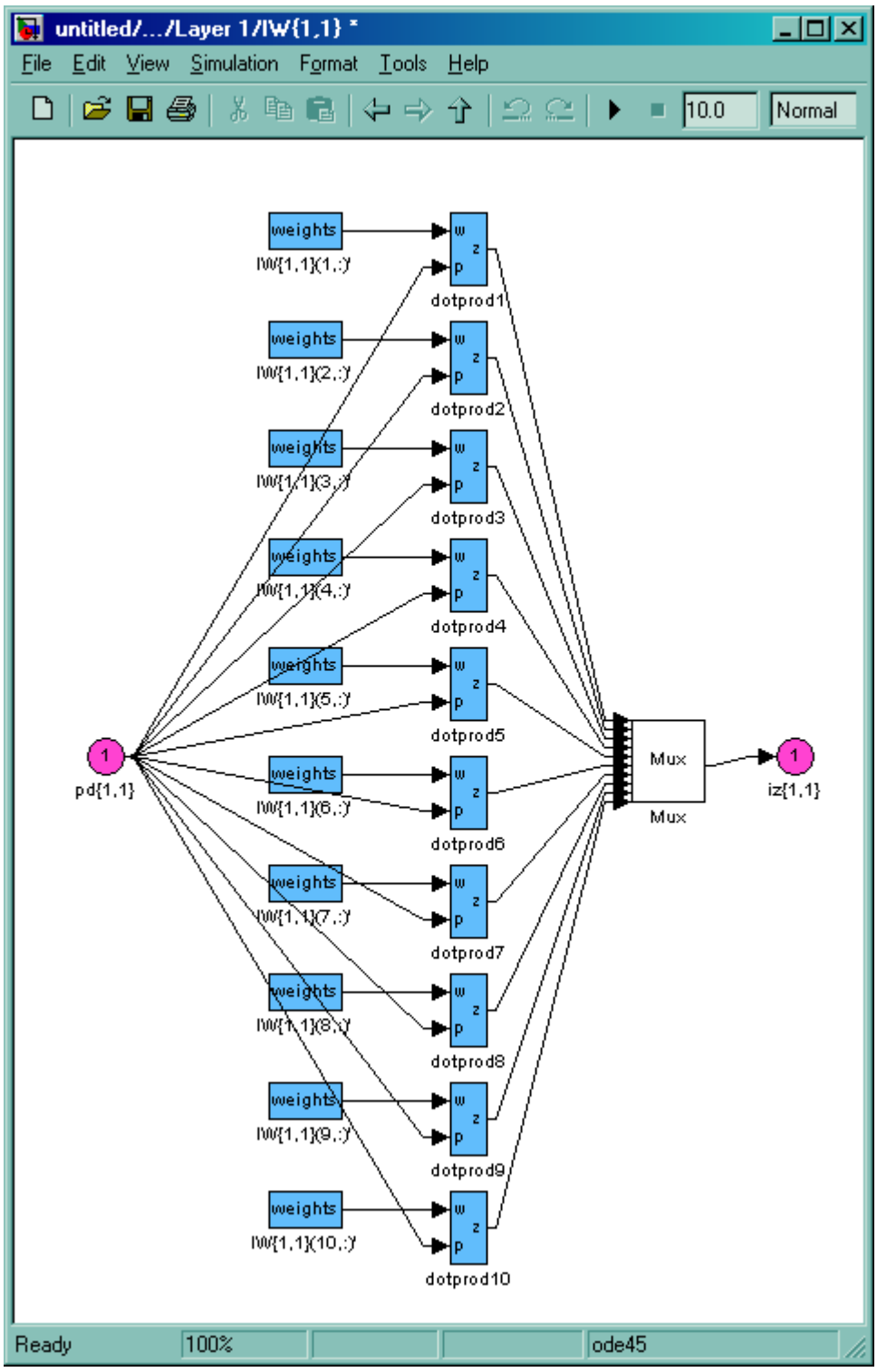

untitled/.../Layer 1/IW{1,1} *
File Edit View Simulation Format Tools Help
10.0
Normal
weights
IW{1,1}(1,:)
dotprod1
IW{1,1}(2,:)
dotprod2
IW{1,1}(3,:)
dotprod3
IW{1,1}(4,:)
dotprod4
IW{1,1}(5,:)
dotprod5
IW{1,1}(6,:)
dotprod6
IW{1,1}(7,:)
dotprod7
IW{1,1}(8,:)
dotprod8
IW{1,1}(9,:)
dotprod9
IW{1,1}(10,:)
dotprod10
w
z
p
1
pd{1,1}
Mux
Mux
iz{1,1}
Ready
100%
ode45

Рис.4.20.

11. Проверим работу сети по аппроксимации исходной функции в Simulink (рис.4.20):

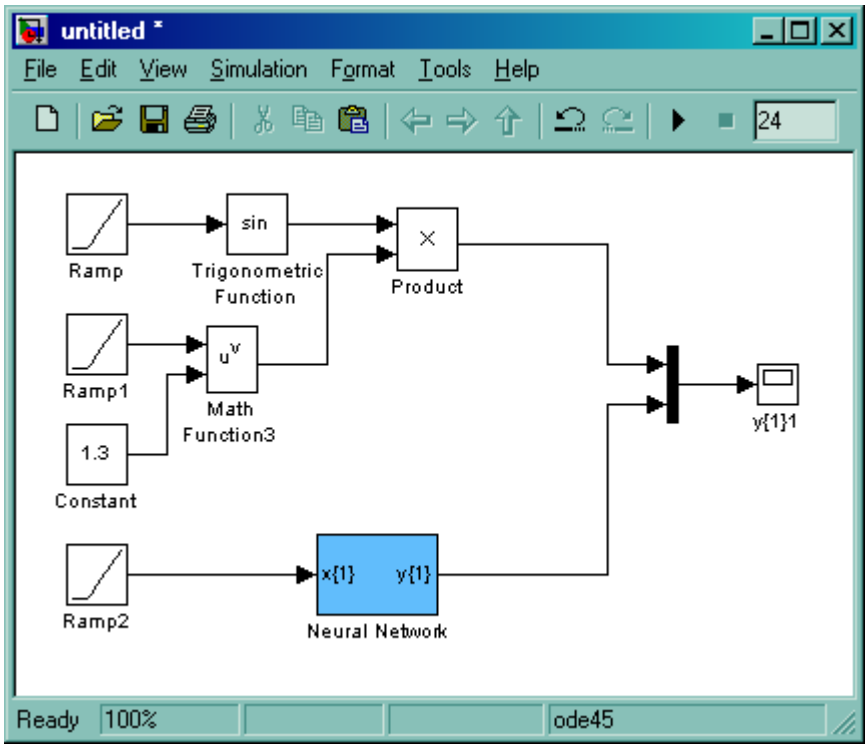


Рис.4.20.

Графики исходной функции и аппроксимирующей ее нейросети (рис.4.21):

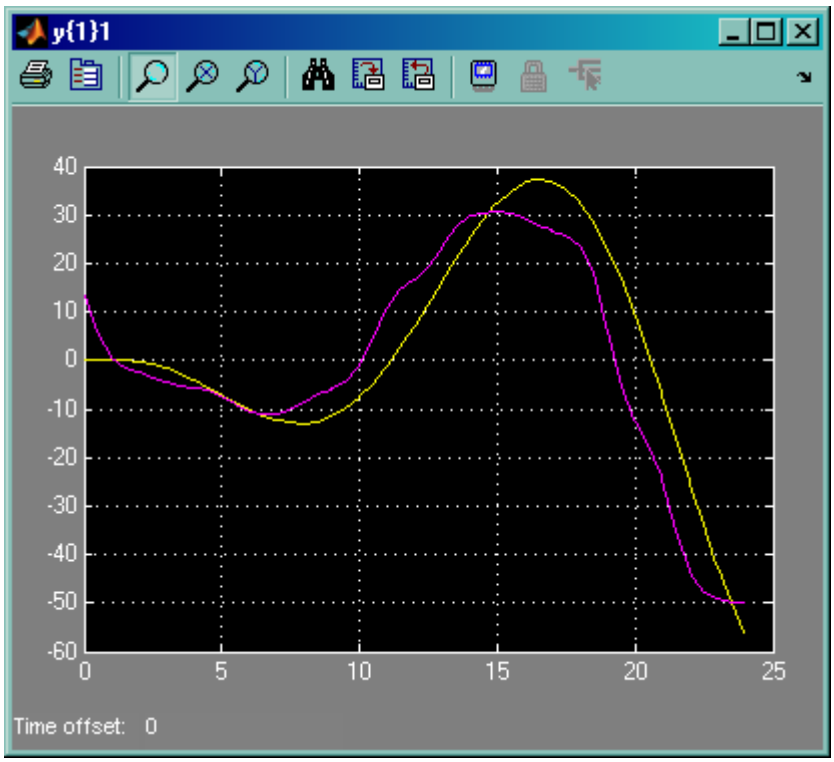


Рис.4.21.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Реализовать в среде MatLab процесс формирования и обучения нейросети по представленному выше алгоритму.
2. Провести процедуру обучения нейросети одной из функций, заданных преподавателем.
3. Рассмотреть п. для различных методов обучения.
4. Составить отчет. В отчете необходимо сделать выводы относительно полученных результатов.

1. Варианты аппроксимируемых функций:

1. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.4x 1.3 ; f(x) = 5, если x<4 и 8, если x>=4

2. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 2x 1.5 ; f(x) = 5, если x<3 и 10, если x>=3

3. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 3.2x 1.25 ; f(x) = 4.5, если x<3 и 8, если x>=3

4. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.72x 1.6 ; f(x) = 50, если x<7 и 78, если x>=7

5. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.5x 1.4 ; f(x) = 5.2, если x<3.45 и 9, если x>=3.45

6. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.6x 1.32 ; f(x) = 3, если x<3 и 8, если x>=3

7. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.22x 1.43 ; f(x) = 6, если x<4.2 и 2, если x>=4.2

8. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.8x 1.11 ; f(x) = 5, если x<4.1 и 3, если x>=4.1

9. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.23x 1.37 ; f(x) = 7.5, если x<9 и 1, если x>=9

10. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.45x 1.35 ; f(x) = 0.5, если x<42 и 80, если x>=42

5. ОБОРУДОВАНИЕ

Лабораторная работа выполняется на персональных компьютерах с применением пакетов моделирования SimuLink, NNT в среде MatLab

6.ОТЧЕТ

Отчет должен содержать краткое описание порядка выполнения работы и результаты обучения нейросети.

Лабораторная работа №5

СИНТЕЗ НЕЙРОННОЙ СЕТИ ДЛЯ РЕШЕНИЯ СИСТЕМ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

1. ЦЕЛЬ РАБОТЫ

С использованием нейросетевых технологий синтезировать алгоритм интегрирования и сформировать модель в среде MatLab для решения систем дифференциальных уравнений

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Рассмотрим задачу решения систем обыкновенных дифференциальных уравнений (ОДУ) в нейросетевом базисе. Нейросетевой базис представляет собой операцию нелинейного преобразования взвешенной суммы, т.е. операцию взвешенного суммирования нескольких входных сигналов с последующим преобразованием этой суммы посредством функции активации формального нейрона, которая может быть как линейной, так и нелинейной.

Пусть задана система n дифференциальных уравнений 1-го порядка

$$\dot{Y}(x) = A\,Y(x), \quad (5.1)$$

где A - квадратная матрица постоянных коэффициентов размера n×n; Y - n - мерный вектор искомой функции аргумента x.

Для метода Рунге-Кутты 1-го порядка (метод прямоугольников) решение системы (5.1) можно представить в виде:

$$Y_{t+1} = Y_t + hAY_t = (E + hA)Y_t = B_t\,Y_t, \quad (5.2)$$

где E - единичная матрица размером n×n.

Элементы матрицы B есть первые два члена разложения матричной экспоненты в степенной ряд.

Для системы двух уравнений выражение (5.2) принимает вид:

$y_{1t+1} = b_{11}y_{1t} + b_{12}y_{2t} = (1 + ha_{11})y_{1t} + ha_{12}y_{2t}$,

$y_{2t+1} = b_{21}y_{1t} + b_{22}y_{2t} = ha_{21}y_{1t} + (1 + ha_{22})y_{2t}$.

Схема соединения нейронов, реализующая решение системы показана на рис.5.1.

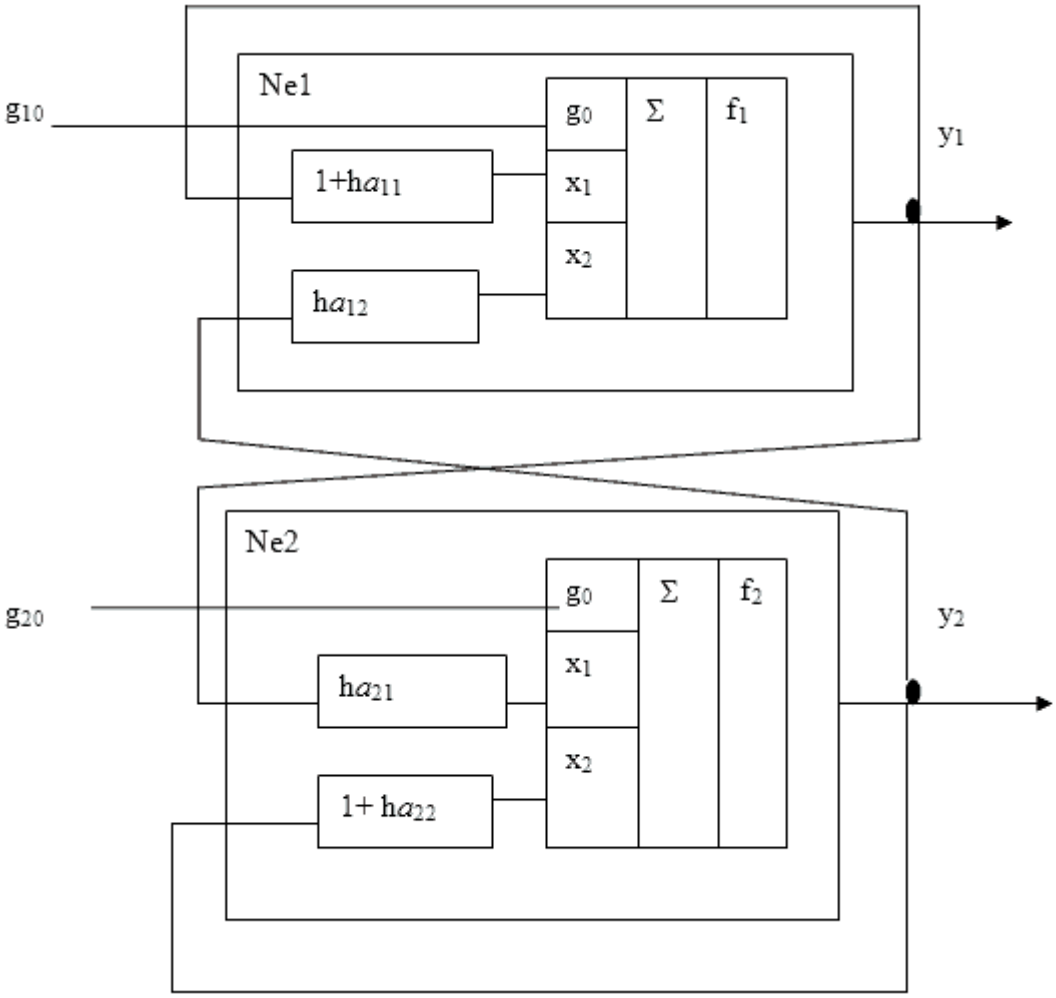


Рис. 5.1. Схема соединения нейронов, реализующая решение системы ОДУ в нейросетевом базисе методом Рунге-Кутты 1-го порядка.

Здесь Ne1 и Ne2 - нейроны, участвующие в операции интегрирования. Выходом сети являются сигналы y_1 и y_2. Входные сигналы g_{10} и g_{20} вводят в нейроны начальное возбуждение, эквивалентное начальным условиям решения системы уравнений. Функция активации у обоих нейронов - симметричная линейная функция.

3. Пример решения задачи разработки ИНС для решения ОДУ в системе Simulink

Разработаем ИНС для решения системы уравнений (5.3)

$$y'1 = y2, \qquad (5.3)$$
$$y'2 = -y1$$

при начальных условиях: y1(0) = -1, y2(0) = 0, t = 0…2π.

Блок-схема решения системы (5.3) в нейросетевом базисе, созданная в среде

Simulink, показана на рис.5.2.

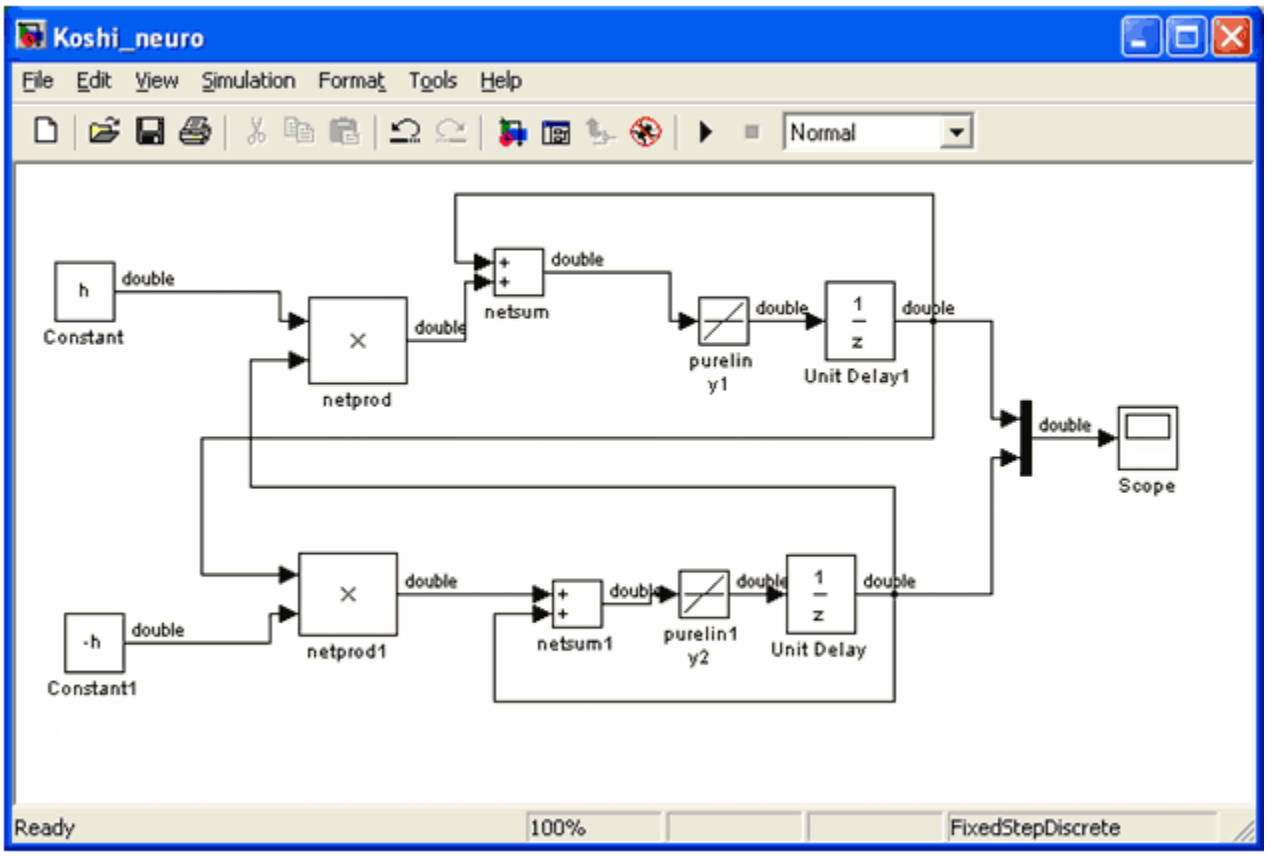


Рис. 5.2. Блок-схема решения системы (5.3) в нейросетевом базисе.

При построении схемы используются блоки из меню Neural Network Blockset: netprod, netsum (Net Input Functions), purelin (Transfer Functions). Для визуализации результатов используется блок Scope (Sinks), для задания шага интегрирования - блок Constant (Sources), для установки начальных значений - блок Unit Delay (Discrete). Окно задания параметров блока Unit Delay показано на рис5.3. Для показа двух графиков в блоке Scope использован блок Mux (Signals&Systems).

Окно браузера библиотеки Simulink показано на рис.5.4.

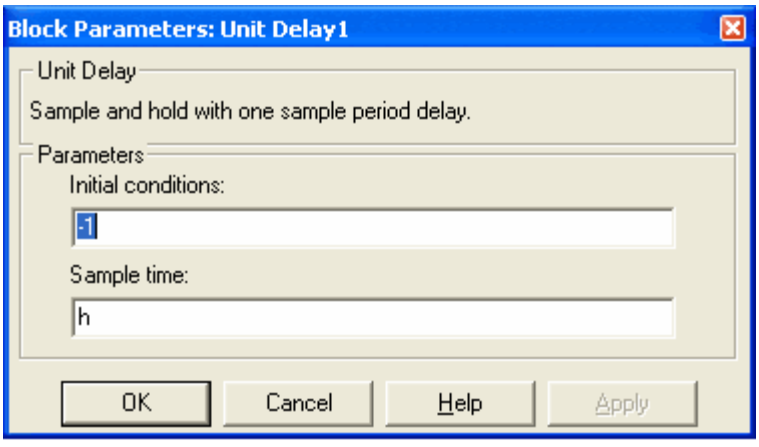


Рис. 5.3. Окно задания параметров блока Unit Delay.

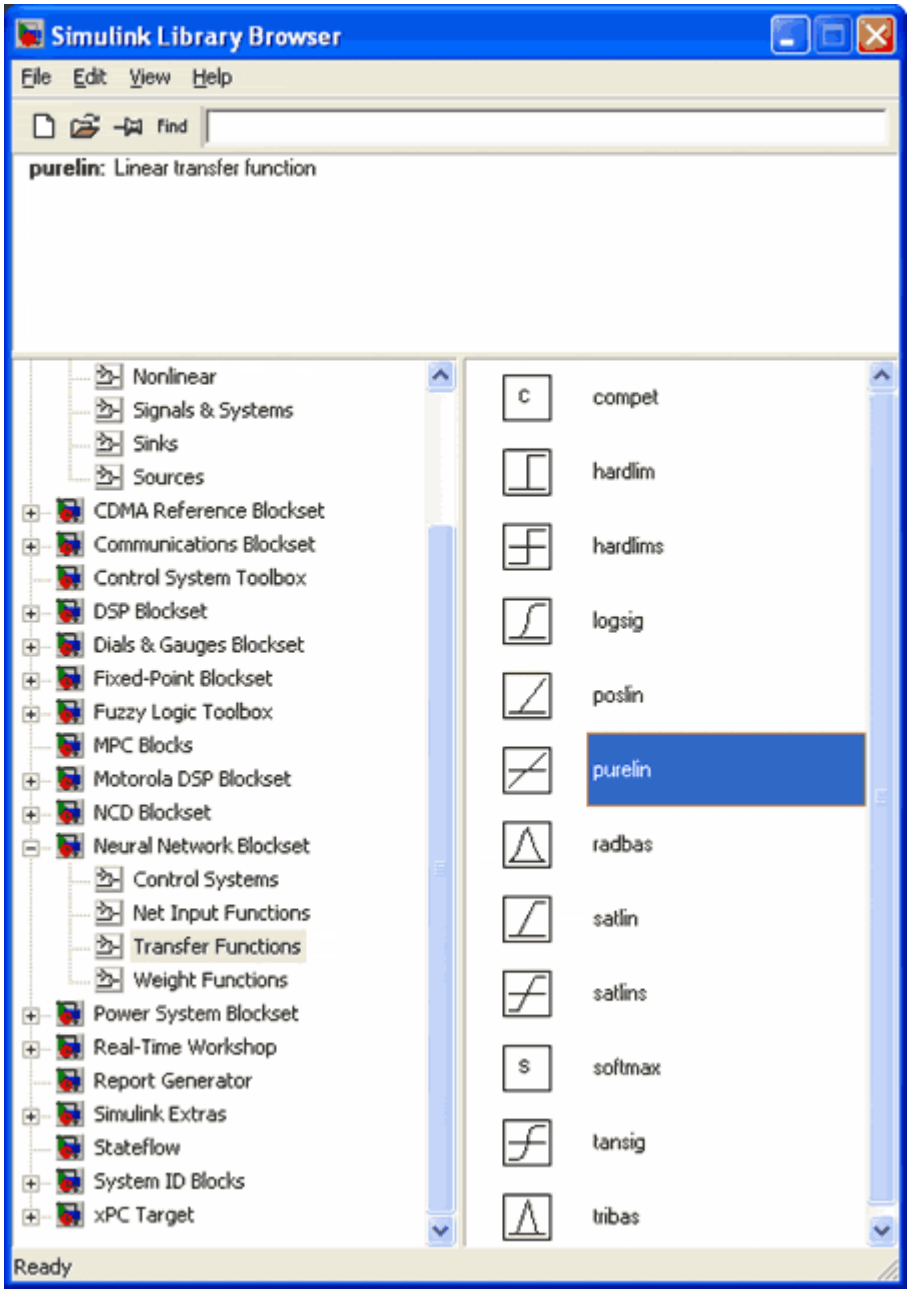


Рис. 5.4. Окно библиотеки блоков системы Simulink.

Для запуска схемы из командной строки MatLab следует набрать:

```
>> N=100;
>> h=2*pi/N;
>> sim('Koshi_neuro')
```

После открытия блока Scope можно увидеть графики решения задачи (рис.5.5).

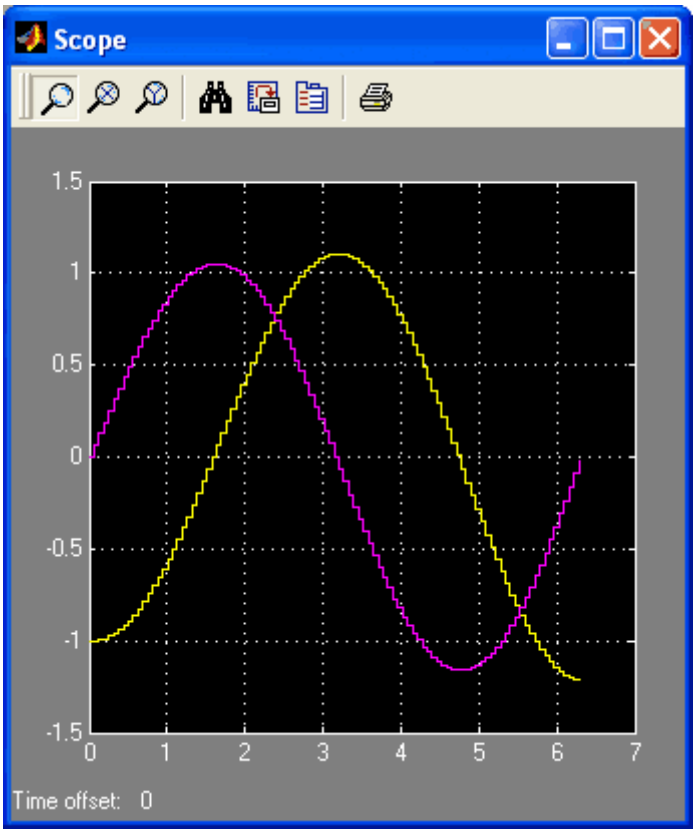


Рис. 5.5. Графики решения задачи в Simulink.

Сравним полученные графики с аналогичными графиками решения задачи в MatLab. Для этого создадим М-файл (рис.5.6), задающий систему уравнений (3):

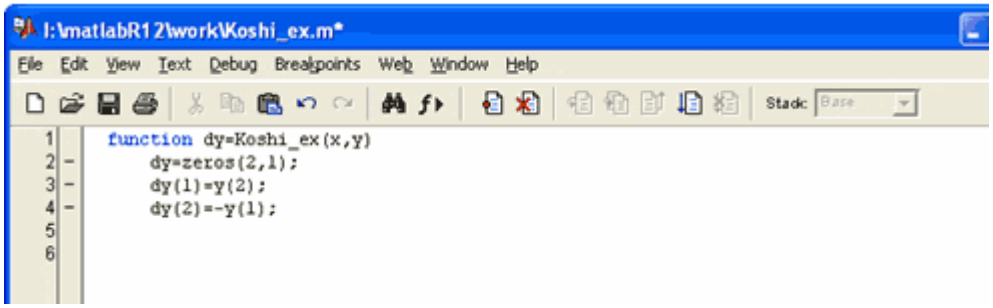


Рис.5.6.

После сохранения М-файла с именем Koshi_ex используем для решения рассматриваемой системы ОДУ функцию ode45, набрав в командном окне MatLab:

```
>> [T,Y] = ode45('Koshi_ex',[0 2*pi],[-1 0]);
>> plot(T,Y(:,1),'-',T,Y(:,2),'-.')
```

Получаем графики решения, идентичные графикам на рис.5.7 (рис.5.7).

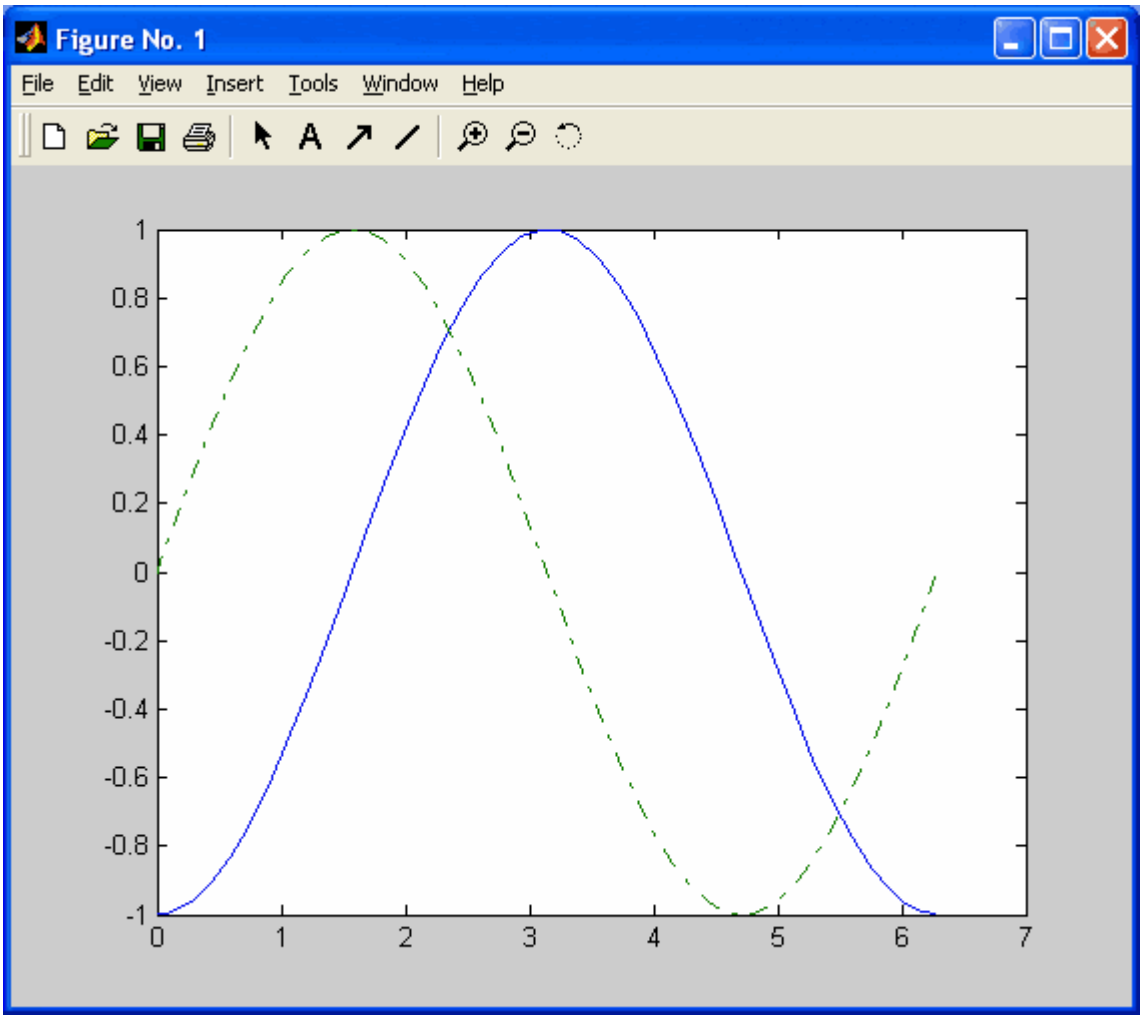


Рис. 5.7. Графики решения задачи с помощью функции ode45.

4. ЗАДАНИЕ НА ВЫПОЛНЕНИЕ

Разработать в среде Simulink ИНС для решения системы ОДУ:

$$\frac{dy_1}{dt} = -y_2 + y_3,$$
$$\frac{dy_2}{dt} = y_1 - y_3,$$
$$\frac{dy_{31}}{dt} = -y_1 + y_2$$

при начальных условиях: $y_1(0) = 0$, $y_2(0) = 0$, $y_3(0) = 1$, $t = 0…15$.

5. ОБОРУДОВАНИЕ

Лабораторная работа выполняется на персональных компьютерах с применением пакетов моделирования SimuLink, NNT в среде MatLab

6.ОТЧЕТ

Отчет должен содержать краткое описание решения поставленных задач.

Лабораторная работа № 6

ПРИМЕНЕНИЕ НЕЙРОННЫХ СЕТЕЙ ДЛЯ ПРОЕКТИРОВАНИЯ СИСТЕМ УПРАВЛЕНИЯ ДИНАМИЧЕСКИМИ ПРОЦЕССАМИ

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Познакомиться с основными архитектурами нейронных сетей,которые реализованы в пакете прикладных программ (ППП)NeuralNetworkToolbox в виде специализированных контроллеров. Освоить основные приемы работы с архитектурой NARMA-L2 Controller, реализованной в среде компьютерной математики MATLAB.
Учебно-методическое обеспечение – методическая разработка, класс ПЭВМ.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В практике проектирования систем управления динамическими процессами все большее применение находят нейронные сети. Это связано с одной стороны со все возрастающими требованиями к характеристикам систем управления и усложняющимися объектами и процессами, которыми необходимо управлять, а с другой с интенсивным развитием специализированной вычислительной техники, например, нейропроцессоров. Универсальные возможности аппроксимации с помощью многослойного персептрона делают их полезным инструментом для решения задач идентификации, проектирования и моделирования нелинейных регуляторов.

В настоящем методическом пособии кратко представлены три архитектуры нейронных сетей, которые реализованы в пакете прикладных программ (ППП)NeuralNetworkToolbox в виде следующих контроллеров:

- контроллера с предсказанием (NN PredictiveController);
- контроллера на основе модели авторегрессии со скользящим средним (NARMA-L2 Controller);
- контроллера на основе эталонной модели (ModelReferenceController).

Применение нейронных сетей для решения задач управления позволяетпредполагает два этапа проектирования:

- этап идентификации управляемого процесса;
- этап синтеза закона управления.

На первом этапе (этапе идентификации) формируется модель управляемого процесса в виде нейронной сети, которая на втором этапе (этапе синтеза) используется для синтеза регулятора. Для каждой из трех архитектур используется

одна и та же процедура идентификации, однако этапы синтеза существенно различаются.

При управлении с предсказанием модель управляемого процесса используется для того чтобы предсказать его будущее поведение, а алгоритм оптимизации применяется для расчета такого управления, которое минимизирует разность между желаемыми и действительными изменениями выхода модели.

При управлении на основе модели авторегрессии со скользящим средним регулятор представляет собой достаточно простую реконструкцию модели управляемого процесса.

При управлении на основе эталонной модели регулятор – это нейронная сеть, которая обучена управлять процессом так, чтобы он отслеживал поведение эталонного процесса. При этом модель управляемого процесса активно используется при настройке параметров самого регулятора.

В последующих разделах обсуждаются все три структуры систем управления и архитектуры соответствующих нейросетевых контроллеров. Каждый раздел включает краткое изложение принципа управления динамическим процессом и сопровождается описанием сценария функционирования проектируемой системы, который реализован в виде комбинации GUI-интерфейса и динамической модели регулятора в системе Simulik.

Динамические модели систем управления с нейросетевыми регуляторами размещены в разделе **ControlSystems**библиотеки блоковдля моделирования нейронных сетей **NeuralNetworkBlockset**, доступ к которым обеспечивается браузером **LibraryBrowser**пакета**Simulink**или командой **neural**(рисунок 6.1) .

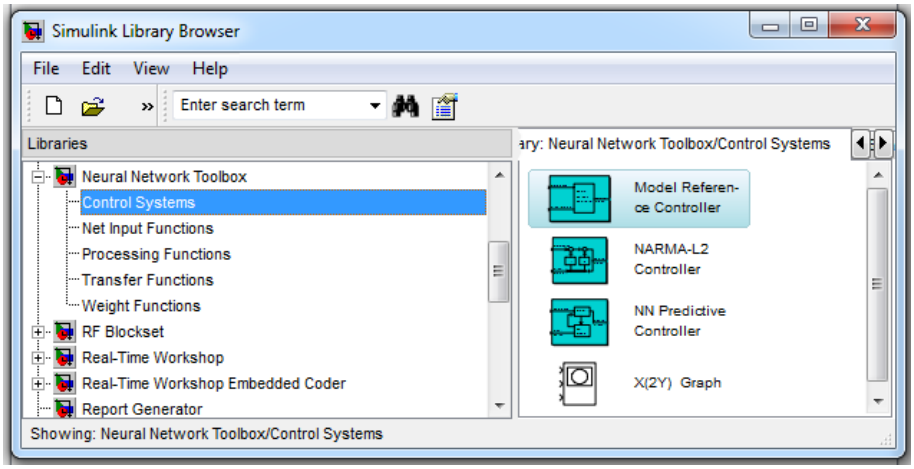


Рис.6.1.

Поскольку ни один конкретный регулятор не является универсальным, то описаны функциональные возможности всех трех типов регуляторов, каждый из которых имеет свои преимущества и недостатки.

Регулятор с предсказанием - NeuralNetPredictiveController– регулятор с предсказанием будущих реакций процесса на случайные сигналы управления. Этот регулятор использует модель управляемого процесса в виде нейронной сети. Алгоритм оптимизации вычисляет управляющие сигналы, которые минимизируют разность между желаемыми и действительными изменениями сигнала на выходе модели и таким образом оптимизируютуправляемый процесс. Построение модели управляемого процессавыполняется автономно с использованием нейронной сети, котораяобучается в групповом режиме с использованием одного из алгоритмов обучения. Реализация такого регулятора требует значительногообъема вычислений, поскольку расчеты по оптимизации выполняются на каждом такте управления.

Регулятор NARMA-L2 (NonlinearAutoregressive – MovingAverage) – регулятор на основе модели авторегрессии со скользящим средним. Данный регулятор представляет собой модифицированную нейросетевую модель управляемого процесса, полученную на этапе автономной идентификации.Изо всех архитектур этот регулятор требует наименьшего объем вычислений. Данный регулятор – это просто некоторая реконструкция нейросетевой модели управляемого процесса, полученной на этапе автономной идентификации. Вычисления в реальном времени связаны только с реализацией нейронной сети. Недостаток метода состоит в том, что модель процесса должна быть задана в канонической форме пространства состояния, которой соответствует сопровождающая матрица, что может приводить к вычислительным погрешностям.

Регулятор на основе эталонной модели - ModelReferenceController– регулятор на основе эталонной модели. Требуемый объем вычислений для этого регулятора сравним с предыдущим. Однако архитектура регулятора с эталонной моделью требует обучения нейронной сети управляемого процесса и нейронной сети регулятора. При этом обучение регулятора оказывается достаточно сложным, поскольку обучение основано на динамическом варианте метода обратного распространения ошибки, так как нейронная сеть использует линии задержки. Достоинством регуляторов на основе эталонной модели является то, что они применимы к различным классам управляемых процессов.

Для каждой из трёх архитектур регуляторов используется одна ита же процедура идентификации управляемого процесса. Нейроннаямодель во всех случаях представляет

собой двухслойную сеть с прямой передачей сигнала и с линиями задержки на каждом слое. Входной, или скрытый (**hiden**) слой может иметь произвольное числонейронов. Выходной слой имеет только один нейрон. Для входногослоя функции взвешивания, накопления и активизации являются соответственно **dotprod**, **netsum**и **logsig**. Выходной слой имеет такиеже функции взвешивания и накопления, а функцией активизации для

него является линейная функция **purelin**. Известно, что сети с такойархитектурой могут воспроизводить весьма сложные нелинейныезависимости между входом и выходом сети.

Настройка параметров нейронной сети, являющейся модельюобъекта, выполняется автономно методом последовательного обучения с использованием данных, полученных при испытаниях реального объекта. Для обучения сети может быть использован любой из обучающих алгоритмов для нейронных сетей. Использование контрольного и тестового множеств обучающих данных позволяет избежать явления переобучения сети. Изменяя число нейронов в первом слое, количество линий задержки на входе и выходе сети, а также интервал квантования, или дискретности, можно обеспечить требуемую точность моделирования управляемого процесса.

Диалоговая панель для идентификации управляемого процесса**PlantIdentification**входит в состав всех трёх регуляторов раздела**ControlSystems**библиотеки нейронных блоков системы **Simulink**,является универсальным средством и может быть использована дляпостроения нейросетевых моделей любых динамических объектов,которые могут быть представлены блоками этой системы.

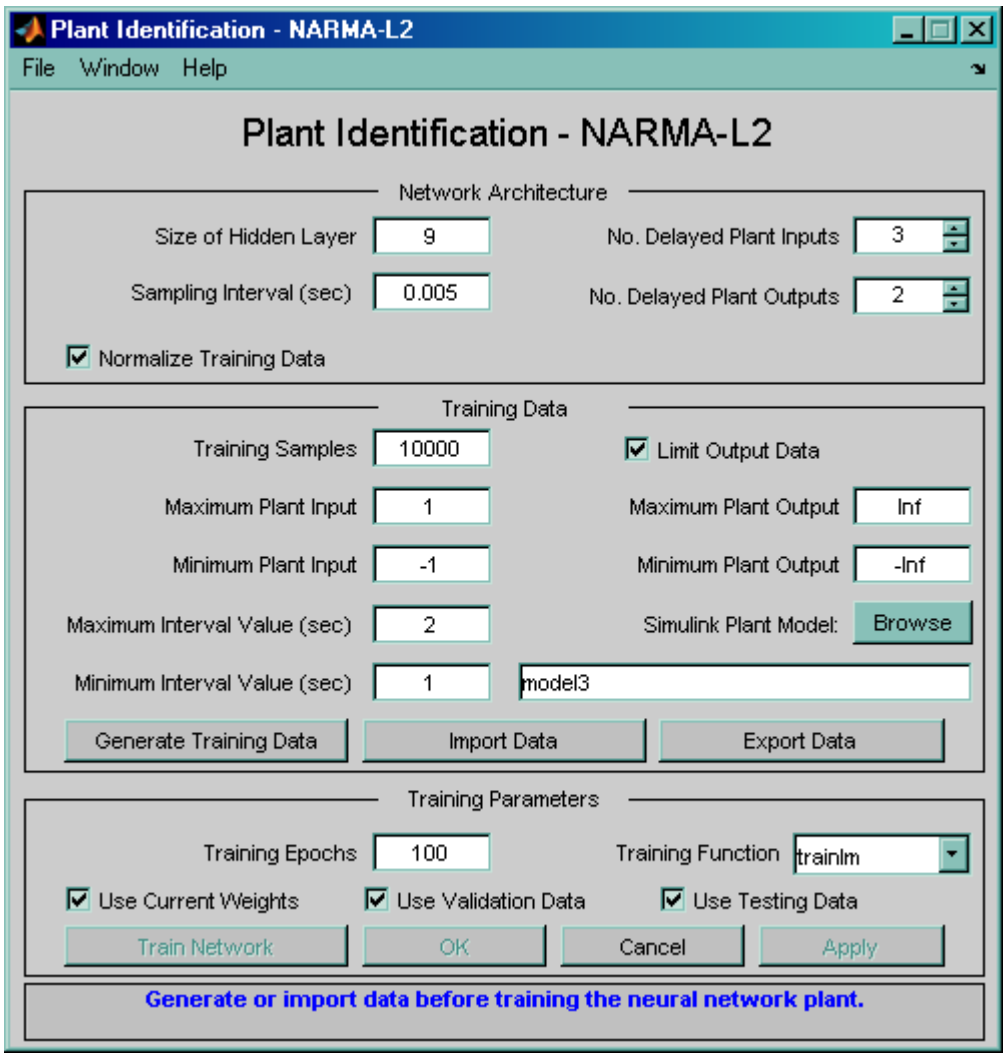


Рис. 6.2

С помощью управляющих элементов панели **PlantIdentification**можно задать архитектуру нейронной сети, параметры обучающейпоследовательности и параметры обучения, а также управлять процессом идентификации и оценивать качество этого процесса.

Набор управляющих элементов для задания архитектурных параметров нейронной сети следующий:

1. **SizeoftheHidenLayer**– количество нейронов на входном илискрытом слое;

2. **No. DelayedPlantInputs**– число линий задержки для входногослоя;

3. **No. DelayedPlantOutputs**– число линий задержки для выходного слоя;

4. **SamlingInterval**– интервал квантования или шаг дискретности, в секундах, между двумя последовательными моментами отсчёта данных;

5. **NotmalizeTrainingData**– переключатель нормирования дляпреобразования обучающих данных к диапазону **[0 1]**.

Набор управляющих элементов для задания характеристик обучающей последовательности таков:

1. **TrainingSamples**– число точек отсчёта для получения обучающей последовательности в виде пар значений вход-выход дляуправляемого процесса, определяемого моделью **Simulink**. В качестве обучающей последовательности генерируется случайный прямоугольный сигнал;

2. **MaximumPlantInput**– максимальное значение случайного входного сигнала;

3. **MinimumPlantInput**– минимальное значение случайного входного сигнала;

4. **MaximumIntervalValue (sec)** – максимальный интервал периода в секундах случайного сигнала;

5. **MinimumIntervalValue (sec)** – минимальный интервал периода в секундах случайного сигнала;

6. **LimitOutputData**– переключатель для ограничения значений выходного сигнала;

7. **MaximumPlantOutput**– максимальное значение выходногосигнала, задаваемое при включённом переключателе **LimitOutputData**;

8. **MinimumPlantOutput**– минимальное значение выходногосигнала, задаваемое при включённом переключателе **LimitOutputData**;

9. **SimulinkPlantModel**– для задания модели управляемого процесса, реализованной с помощью блоков **Simulink**, имеющий портывхода и выхода и сохранённой в файле ***.mdl**; выбор модели производится с помощью кнопки **Browse;**

Параметры обучения задаются следующим образом:

1. **TrainingEpochs**– количество циклов обучения;

2. **TrainingFunction**– задание обучающей функции;

3. **UseCurrentWeights**– переключатель для использования текущих весов нейронной сети;

4. **UseValidationData**– переключатель для использования контрольного множества в объёме **25 %** от обучающего множества;

5. **UseTestingData**– переключатель для использования тестового множества в объёме **25%** от обучающего множества.

РЕГУЛЯТОР NARMA-L2

Нейросетевой регулятор, описанный в этом разделе, использует в качестве модели управляемого процесса модель нелинейной авторегрессии со скользящим средним (NonlinearAutoregressive-MovingAverage - NARMA-L2)

Рассмотрим пример обучения нейросетевого регулятора на основе модели авторегрессии со скользящимсреднийм: **Narmal2(NonlinearAutoregressive – MovingAverage).** В качестве объекта управления выберем модель двигателя постоянного тока:

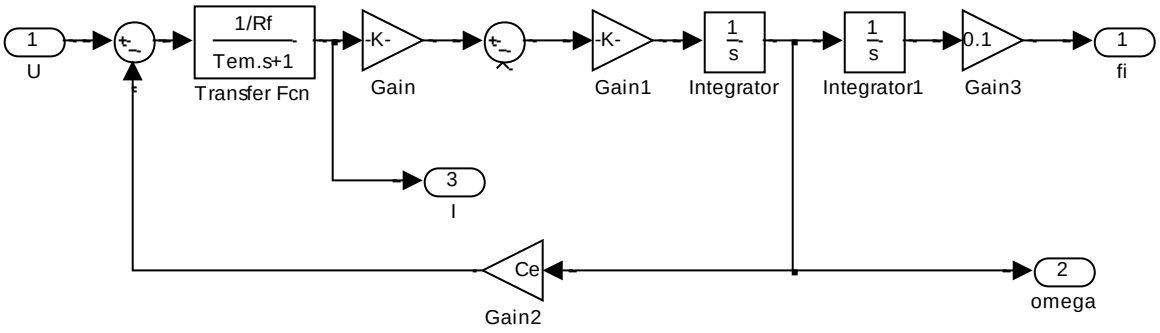


Рис. 6.3.

Модель системы управления с нейроконтроллером выглядит следующим образом:

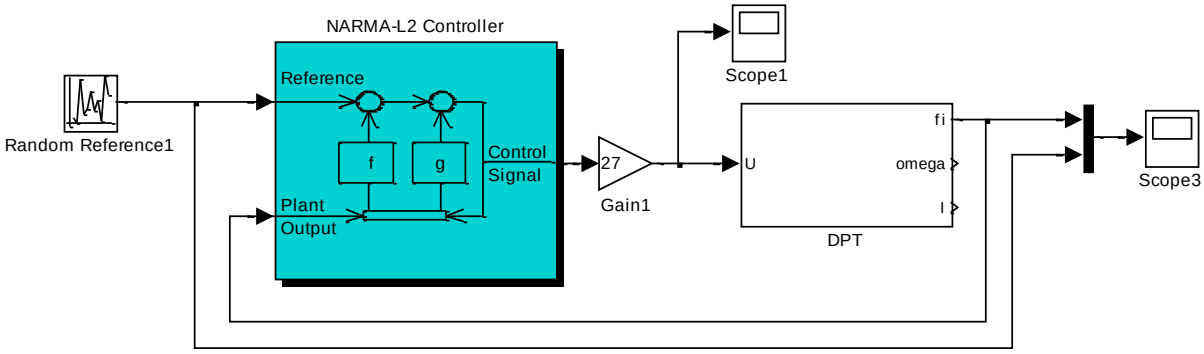


Рис. 6.4.

Для идентификации управляемого процесса необходимо в диалоговой панели **PlantIdentification** выполнить следующие действия:

3. ОБОРУДОВАНИЕ

В лабораторной работе используются персональные компьютеры (не ниже Pentium 4) с установленным пакетом моделирования MatLab (версии не ниже 6.5).

4. ЗАДАНИЕ НА ВЫПОЛНЕНИЕ

1. Задать архитектуру нейронной сети, которая будет моделью управляемого процесса (задание количества нейронов и элементов задержек).

Рекомендации: для линейных объектов до 3/4 порядков количество нейронов должно быть не менее 10 из соображений обеспечения требуемой точности моделирования процессов; количество задержек определяется порядком моделируемого объекта.

2. Задать параметры обучения.

3. Выбрать модель **Simulink**для управляемого процесса.

4. Сгенерировать обучающую последовательность заданногообъёма, запустив модель **Simulink**с помощью кнопки **GenerateTrainingData**. Генерация обучающей последовательности производится с помощью воздействия ряда ступенчатых сигналов на модельуправляемого процесса и снятия значений на выходе и входе моделичерез каждый шаг квантования. Графики входного и выходного сигнала отображаются в окне **PlantInput-OutputData**.

Иллюстрация рисунками 6.5-6.15.

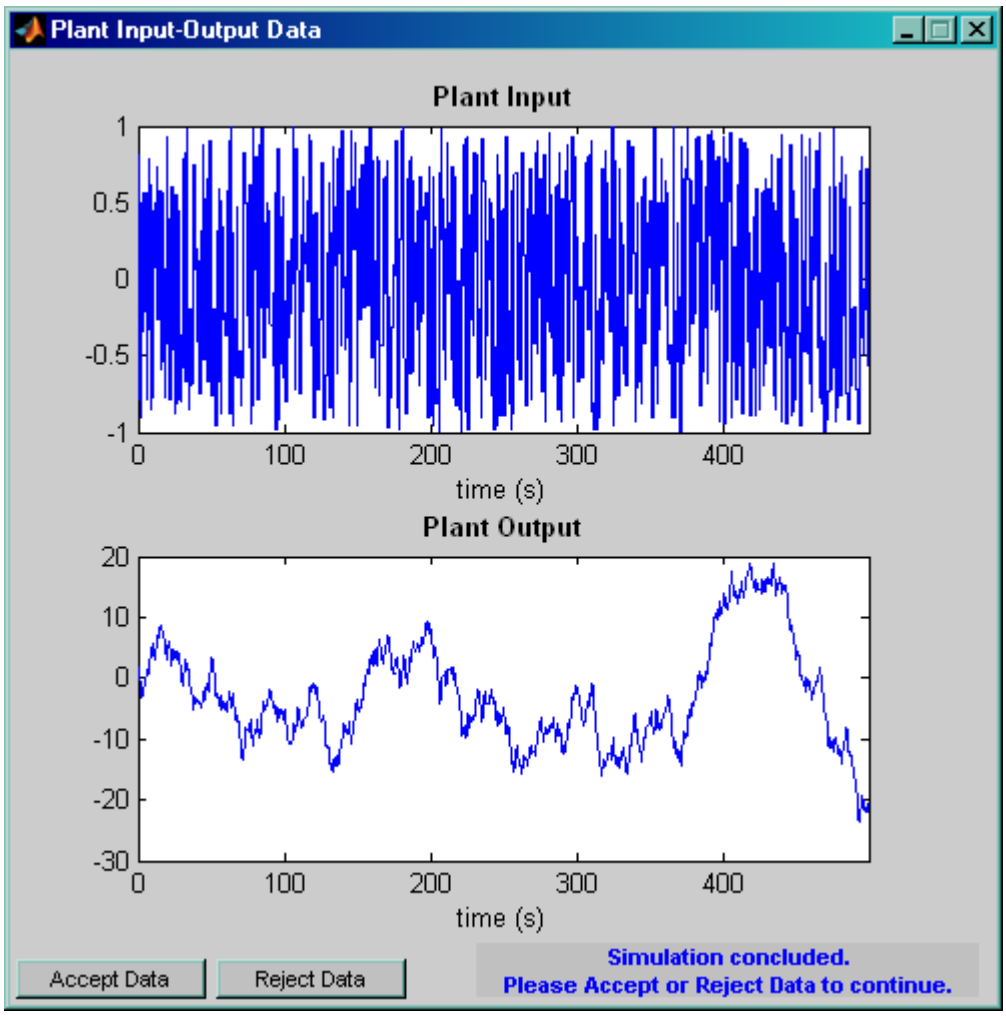


Рис.6.5 Пример формирования входной и выходной последовательности для обучения нейросети

5. По завершении генерации обучающей последовательности необходимо либо принять эти данные, нажав на кнопку **AcceptData**, итогда они будут использованы для обучения нейронной сети, либоотвергнуть их, нажав кнопку **RejectData**, и повторить процесс иден

тификации управляемого процесса, представленного моделью **Simulink.**

6. После получения обучающей последовательности необходимоустановить требуемые параметры обучения и с помощью кнопки**TrainNetwork**запустить процесс обучения нейронной сети.

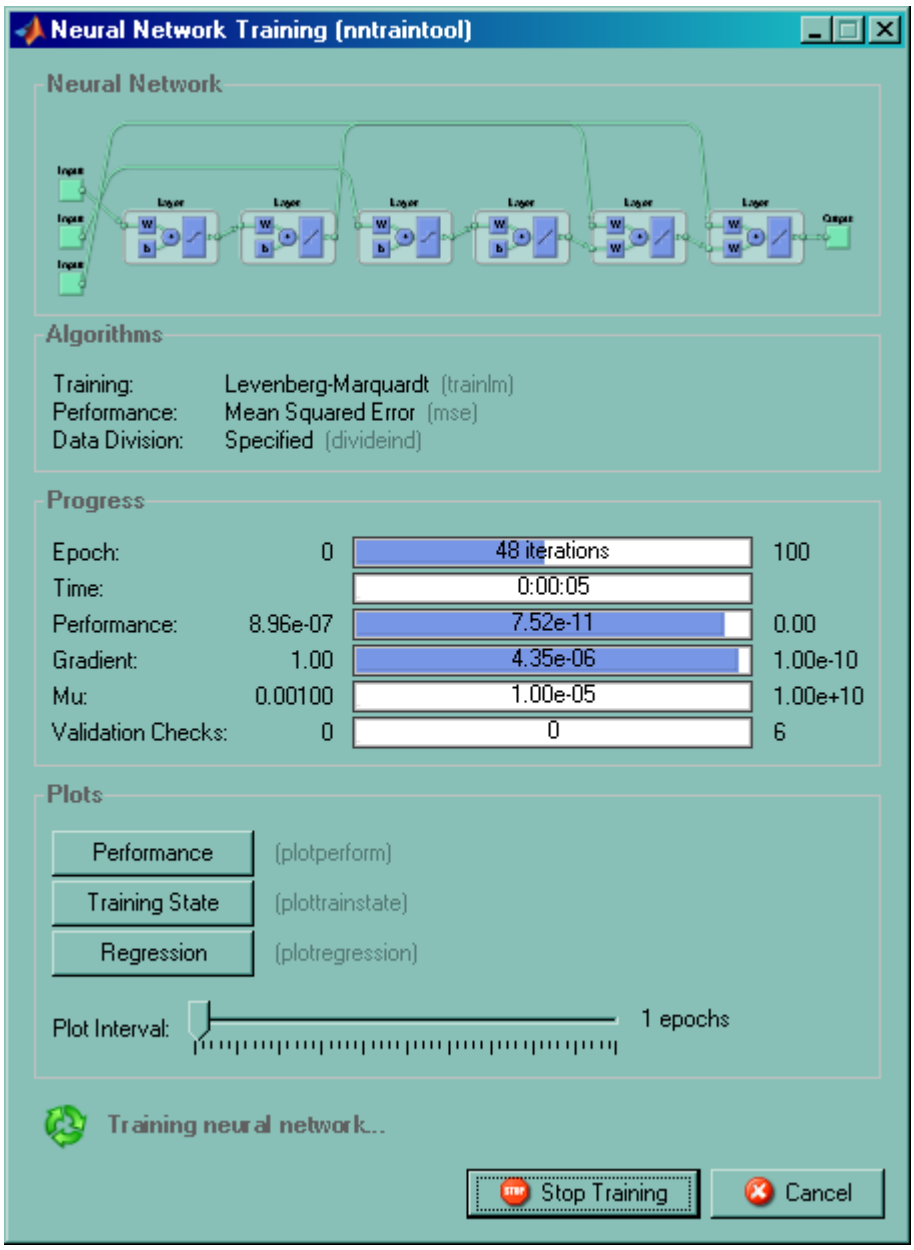


Рис.6.6

7. После завершения обучения его результаты отображаются награфиках изменения ошибки сети для обучающей, контрольной итестирующей последовательностей, а также выходных значений модели и сети при подаче на вход указанных последовательностей.

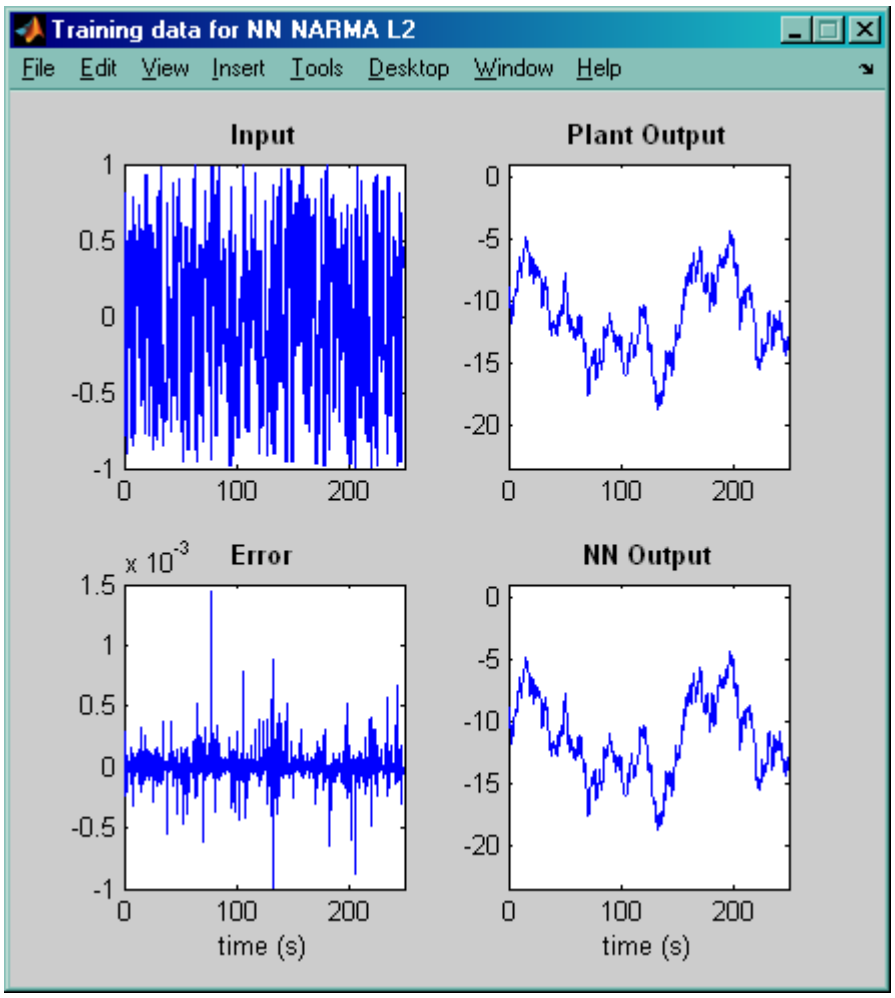


Рис.6.7 Анализ качества обучения

8. Если результаты обучения приемлемы, то надо сохранить параметры нейросетевой модели управляемого процесса и приступитьк синтезу регулятора того или иного класса, нажав кнопки **Apply**и **Ok.**

9. Если результаты обучения неприемлемы, то следует нажатькнопку **Cancel**и повторить процесс идентификации сначала, изменяяархитектуру сети и параметры обучающей последовательности.

10. Обучающую последовательность можно импортировать израбочей области или из файла, нажав на кнопку **ImportData**. Еслинеобходимо обучающую последовательность сохранить в рабочейобласти или в файле для подбора параметров архитектуры нейронной сети, то следует после получения данных нажать на кнопку**ExportData.**

11. Удалить только что сгенерированные данные при необходимости можно с помощью кнопки **EraseGeneratedData**.

Таким образом, диалоговая панель **PlantIdentification**позволяетидентифицировать управляемый процесс, представленный в виде имитационной модели **Simulink**, построить

двухслойную нейроннуюсеть прямой передачи сигнала с необходимым числом нейронов илиний задержки, обучить эту сеть для получения нейронной моделиуправляемого процесса, оценить качество обучения и работу нейронной сети.

Для регулятора на основе авторегрессии со скользящимсредним, этап его синтеза отсутствует, так как такой регулятор представляет собой полученную нейросетевую модель управляемогопроцесса с предсказанием.Для регуляторов на основе эталонной модели с предсказанием, этап синтеза необходим.

Работа системы управления с обученной нейросетью при отработке ступенчатых сигналов (Рис. 6.8).

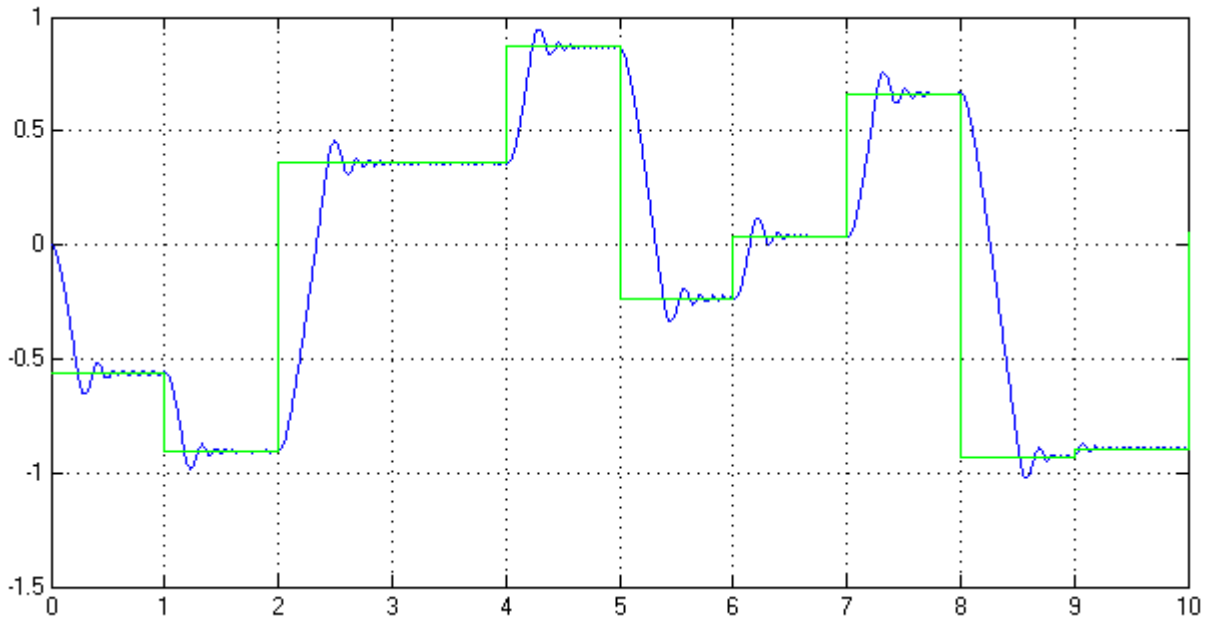


Рис. 6.8.

Управление (Рис. 6.9.):

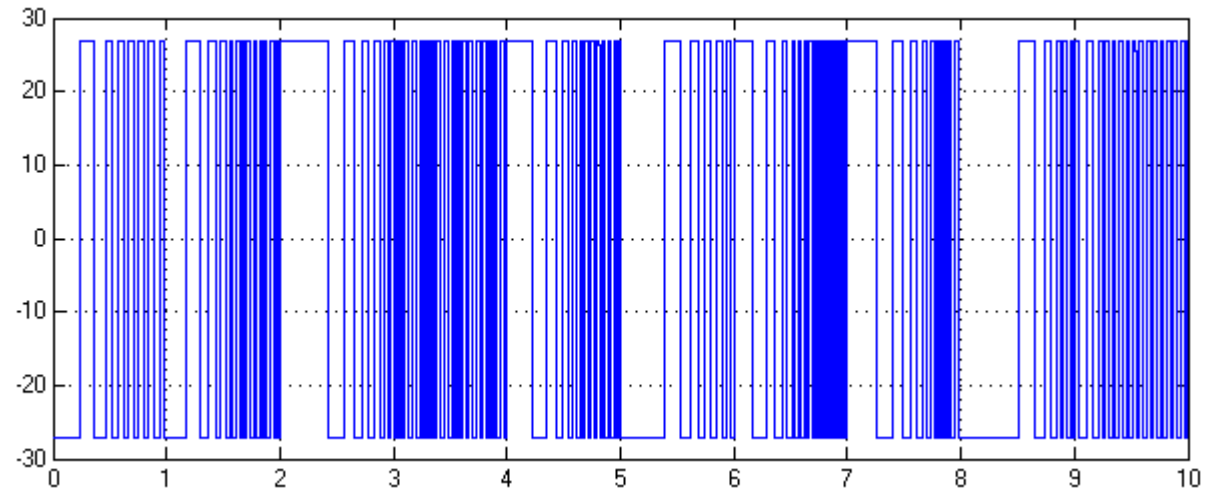


Рис. 6.9.

Из анализа полученных данных следует, что реакция системы на ступенчатые воздействия со случайной амплитудой вполне удовлетворительна, имеет колебательный характер с достаточно быстрым затуханием, на интервале 5 с все установки эффективно отрабатываются. Таким образом, регулятор NARMA-L2, реализованный в виде нейронной сети, можно использовать для управления исполнительным двигателем постоянного тока.

5.ОТЧЕТ

Отчет должен содержать краткое описание нейросетевых регуляторов и решение заданий, представленных в разделе 4 настоящего описания.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие архитектуры нейронных сетей были рассмотрены?
2. Что собой представляет регулятор с предсказанием?
3. Что собой представляет регулятор на основе модели авторегрессии со скользящим средним?
4. Что собой представляет регулятор на основе эталонной модели?
5. Что входит в набор управляющих элементов для задания характеристик обучающей последовательности?

Лабораторная работа №7

ФОРМИРОВАНИЕ И ОБУЧЕНИЕ НЕЙРОНЕЧЕТКОЙ СЕТИ ДЛЯ ЦЕЛЕЙ УПРАВЛЕНИЯ ДИНАМИЧЕСКИМИ ОБЪЕКТАМИ С ПОМОЩЬЮ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ MATLAB

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Освоение на практике основных приемов работы с редактором anfisedit, позволяющих формировать нейронечеткие модели динамических объектов и нейронечеткие регуляторы с использованием среды Matlab.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В системе MatLabреализован ANFIS-редактор, который позволяет с помощью нейронечеткого описания автоматически синтезировать из экспериментальных данных нечеткие правила. Сеть ANFIS описывает систему нечеткого логического вывода типа Сугено. Параметры сети после обучения настраиваются так, чтобы минимизировать отклонения между результатами моделирования и экспериментальными данными.

Загрузка ANFIS-редактора осуществляется по команде

>>anfisedit

Графическое окно AnfisEditorпоказано на рис.7.1.

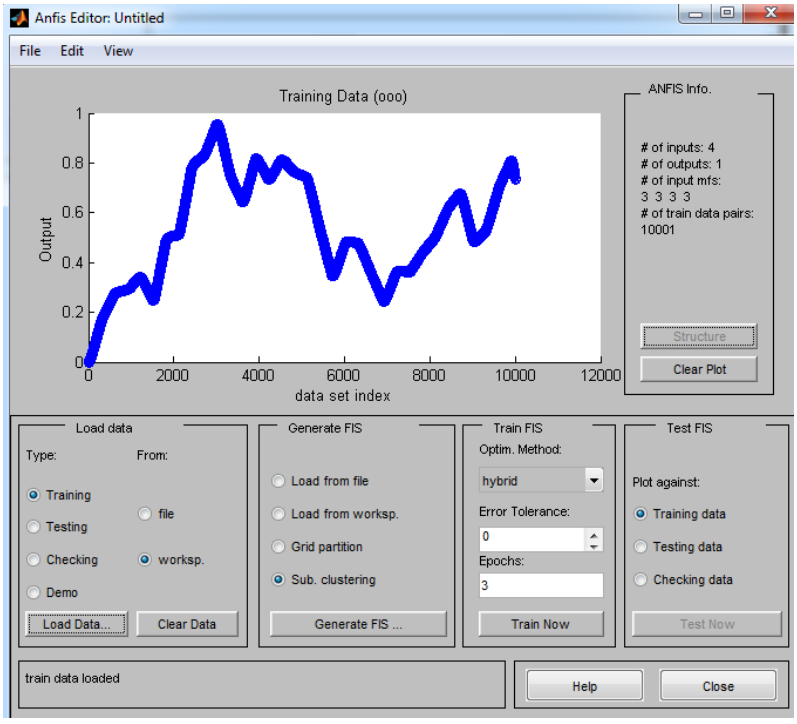


Рис.7.1.

В центре окна изображена область визуализации. В этой области выводится два типа информации:

- при обучении системы – график зависимости ошибки обучения от порядкового номера итерации;

- при загрузке данных и тестировании системы – экспериментальные данные и результаты моделирования. При этом по оси абсцисс откладывается порядковый номер строчки данных в выборке (обучающей, тестирующей или контрольной), а по оси ординат – значение выходной переменной для данной строчки выборки. Используются следующие символы: голубая точка (.) – тесирующая выборка; голубая окружность (o) – обучающая выборка; голубой плюс (+) – контрольная выборка; красная звездочка (*) – результаты моделирования.

Правее области визуализации находится область свойств ANFISinfo.

В области свойств выводится информация о количестве входных и выходных переменных, о количестве функций принадлежностей для для каждой входной переменной, а также о количестве строчек в выборках.

В этой области расположены две кнопки: Structureи ClearPlot.

Нажатие кнопки Structureоткрывает новое графическое окно, в котором система нечеткого логического вывода представляется в виде нейро-нечеткой сети. Количество входов этой сети зависит от числа столбцов обучающей выборки. Количество нейронов – 2-4 слоя соответствует количеству правил.

На рис.7.2. показан пример сети ANFIS, в которой две входные переменные (для описания каждой использовано по три терма) и восемь правил. Количество правил зависит от выбора параметров в области GenerateFIS.

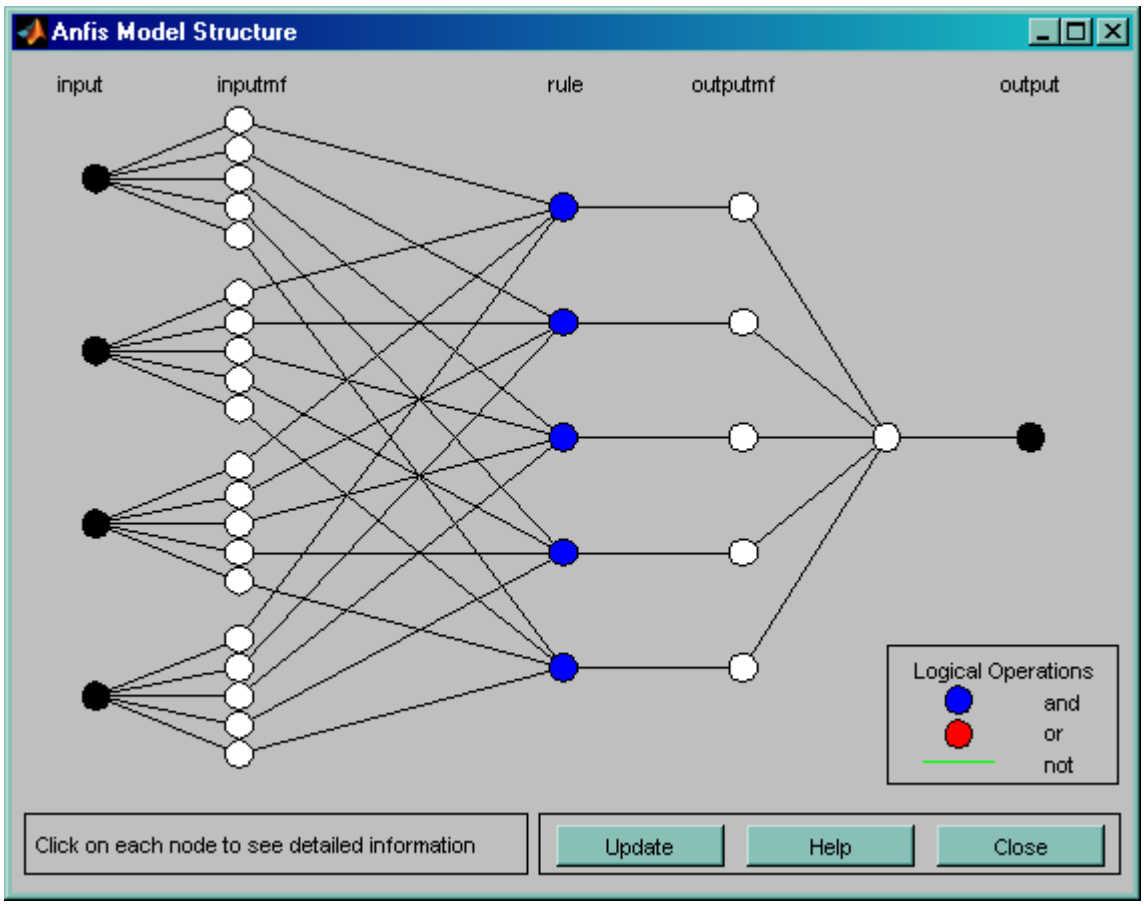


Рис.7.2.

Нажатие кнопки ClearPlot позволяет очистить область визуализации.

В области загрузки данных LoadDataрасположены:

- Type - меню выбора типа данных: Traning- обучающая выборка; Testing - тестирующая выборка; Checking- контрольная выборка; Demo- демонстрационный пример;

- From - меню выбора источника данных: Disk–диск; Worksp. –рабочая область MatLab;

- LoadData – кнопка загрузки данных;

-ClearData - кнопка очистки данных.

В течение одного сеанса работы ANFIS - редактора следует загружать данные одного формата, т.е. количество входных переменных в выборках должно быть одинаковым.

GenerateFIS – меню создания исходной системы нечеткого логического вывода, содержащее альтернативы:

- Loadfromdisk – загрузка существующей системы с диска;

- Loadfromworksp. – загрузка системы из рабочей области MatLab;

-– генерирование описания входных переменных по методу решетки (без кластеризации);

- Sub. Clustering – генерирование по методу субкластеризации.

В области также расположена кнопка Generate, по нажатию котрой генерируется исходная система нечеткого логического вывода.

При выборе Gridpartition появляется окно ввода параметров метода решетки (рис.7.3), в котором нужно указать количество термов для каждой входной переменной и тип функции принадлежности для входных и выходной переменных.

Таким образом, при выборе метода решетки выполняется нечеткое разбиение базовых шкал входных переменных. Настройка параметров термов не требуется, а количество правил равно произведению мощносте терм-множеств лингвистических переменных, описывающих посылки (см. рис.7.3).

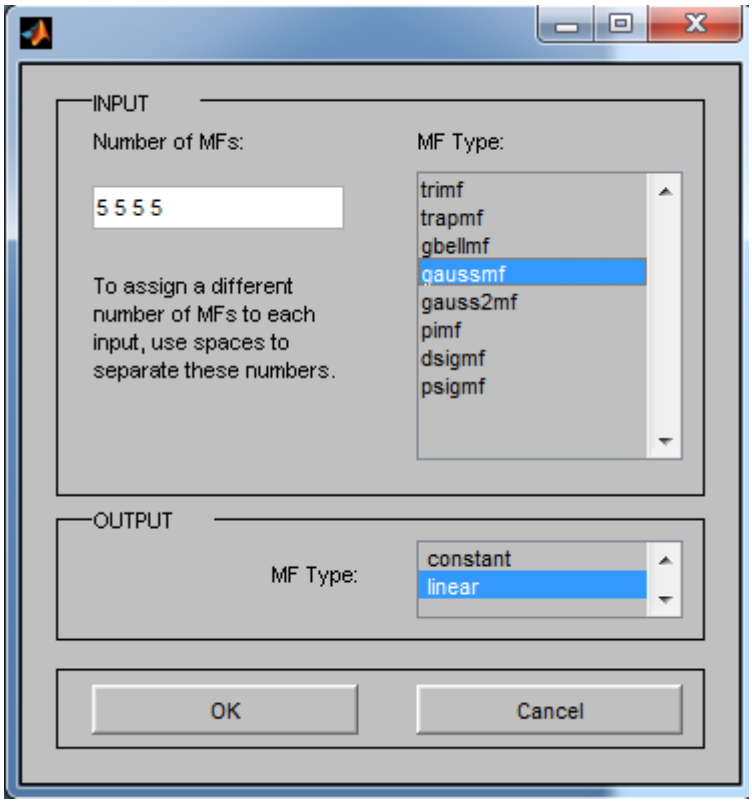


Рис.7.3.

При выборе Sub.clustering появляется окно ввода следующих параметров метода субкластеризации (рис.7.4).

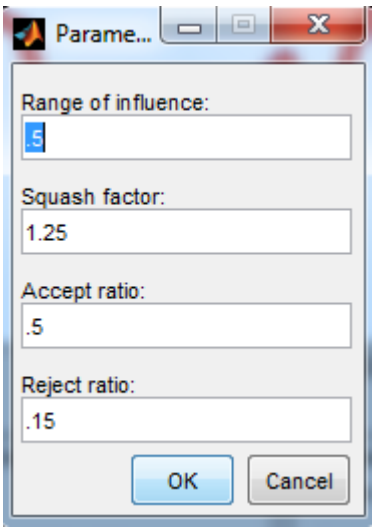


Рис.7.4.

Нарис.7.4обозначены:

Rangeofinfluence– уровни влияния входных переменных;

Squashfactor– коэффициентподавления;

Acceptratio – коэффициент, устанавливающий во сколько раз потенциал данной точки должен быть выше потенциала центра первого кластера для того, чтобы центром одного из кластеров была назначена рассматриваемая точка;

Rejectratio – коэффициент, устанавливающий во сколько раз потенциал данной точки должен быть ниже потенциала центра первого кластера, чтобы рассматриваемая точка была исключена из возможных центров кластеров.

Область обучения (TrainFIS) содержит меню выбора метода оптимизации (Optim.method), поле задания требуемой точности обучения (Errortolerance), поле задания количества итераций обучения (Epochs) и кнопка TrainNow, нажатие которой запускает режим обучения. Промежуточные результаты обучения выводятся в область визуализации и в рабочую область MatLab. В ANFIS - редакторе реализованы два метода обучения:

-Backpropa - метод обратного распространения ошибки, основанный на идеях метода наискорейшего спуска;

- Hybrid - гибридный метод, объединяющий метод обратного распространения ошибки с методом наименьших квадратов.

В области тестирования (TestFIS) расположены меню выбора выборки и кнопка TestNow, при нажатии которой происходит тестирование нечеткой системы с выводом результатов в область визуализации.

3. ОБОРУДОВАНИЕ

В лабораторной работе используются персональные компьютеры (не ниже Pentium 4) с установленным пакетом моделирования MatLab (версии не ниже 6.5).

4.ЗАДАНИЕ НА ВЫПОЛНЕНИЕ

Рассмотреть решение задачи построения нейронечеткой модели динамического объекта, заданного своей передаточной функцией. Тип и параметры передаточной функции задает преподаватель. Нейронечеткая модель в последующем может быть использована для реализации нейронечеткого управления, например, для построения инверсного нейронечеткого регулятора.

5.ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:

1. Запустить Matlab.

2. Для передаточной функции, заданной преподавателем, сформировать SimuLink-модель, которая в последующем будет использоваться для обучения нейронечеткой сети.

В качестве примера рассмотрим динамический объект, передаточная характеристика которого имеет вид (рис.7.6):

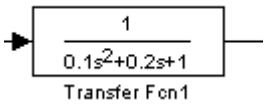


Рис.7.6.

3. Сформировать SimuLink- модель для расчета динамических характеристик объекта (рис.7.7)

Объект:

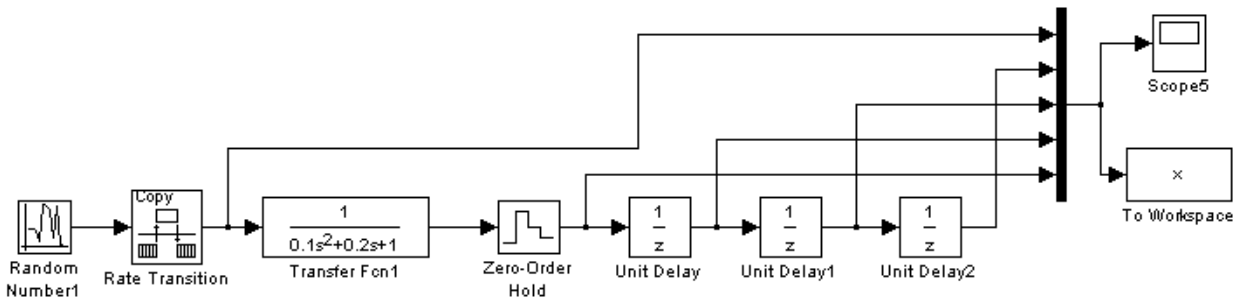


Рис.7.7.

Подача на объект случайного сигнала и снятие тестовых характеристик для последующего обучения нейроэмулятора (рис.7.8).

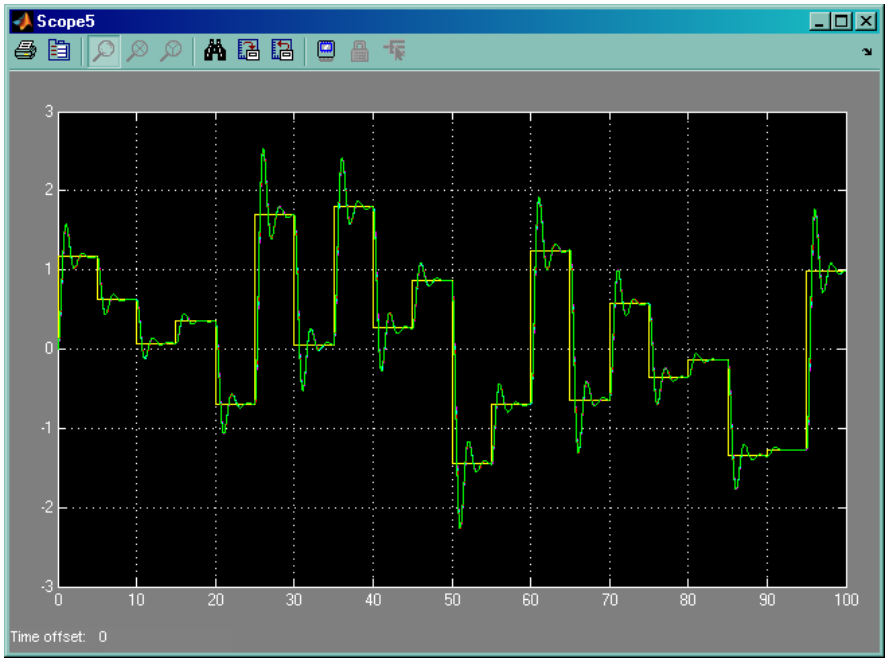


Рис.7.8.

Загрузка полученных данных в редактор Anfis (вызывается командой anfiseditв командной строке Matlab) (рис.7.9):

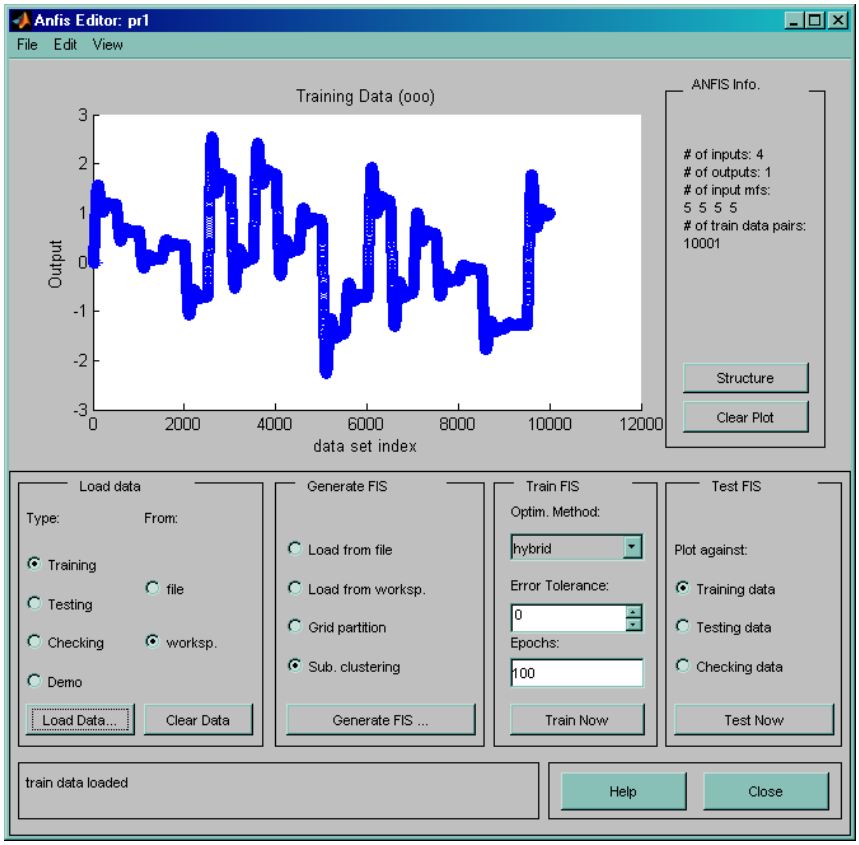


Рис.7.9.

Генерация нечеткого регулятора (GenerateFIS) (рис.7.10):

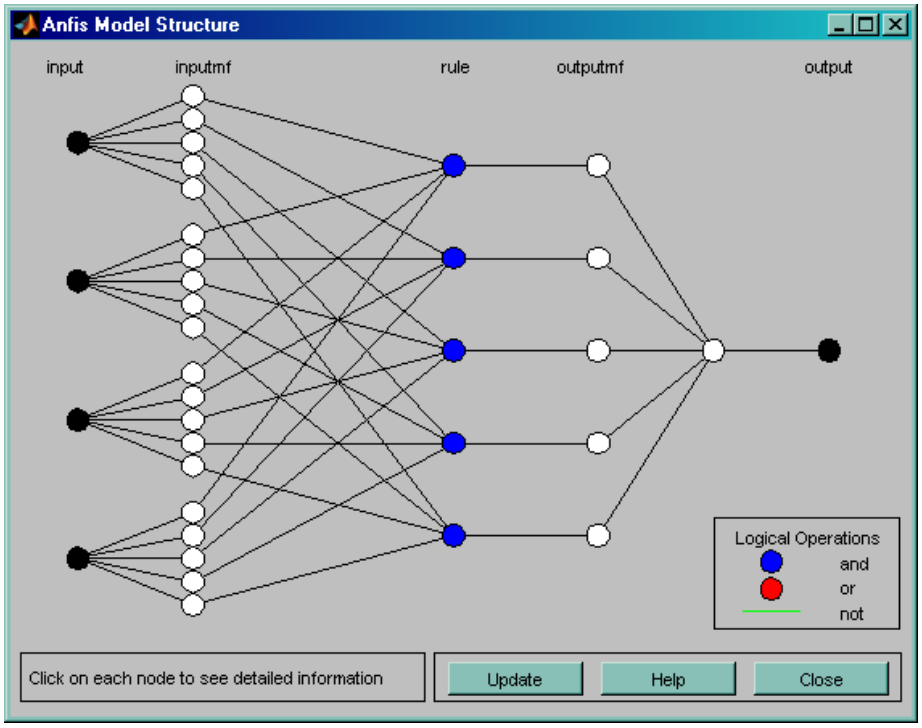


Рис.7.10.

Обучение контроллера (Trainnow), тестирование (TestFIS) и экспорт обученного контроллера в WorkSpace (рис.7.11).

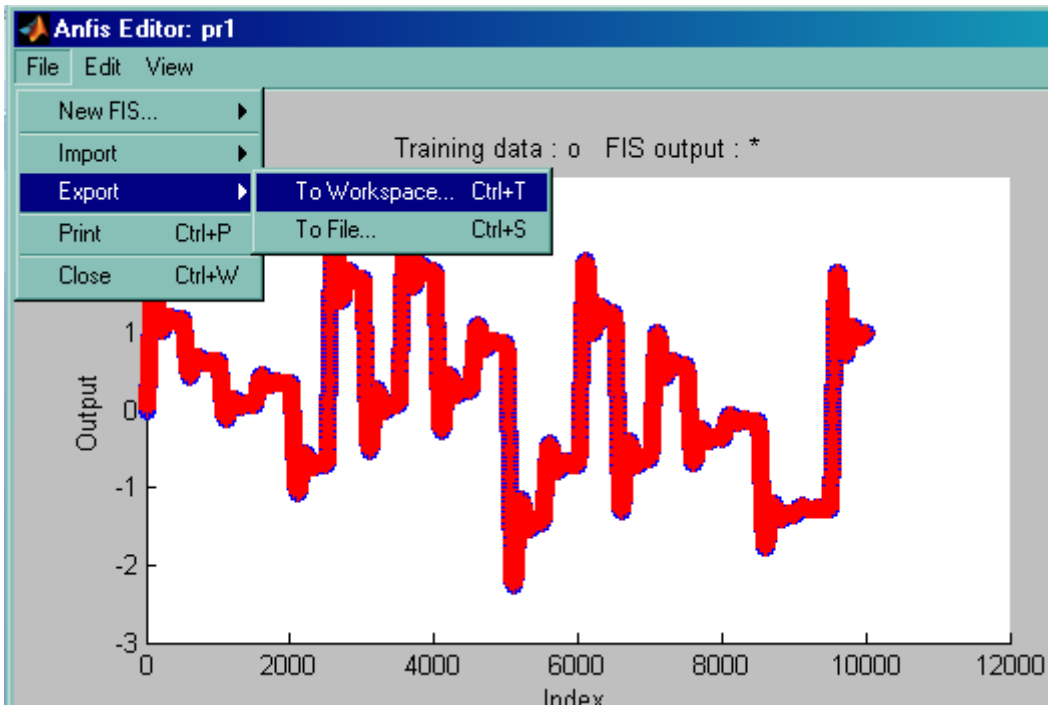


Рис.7.11.

Проверка работы обученногонейроконтроллера (рис.7.12- 7.13).

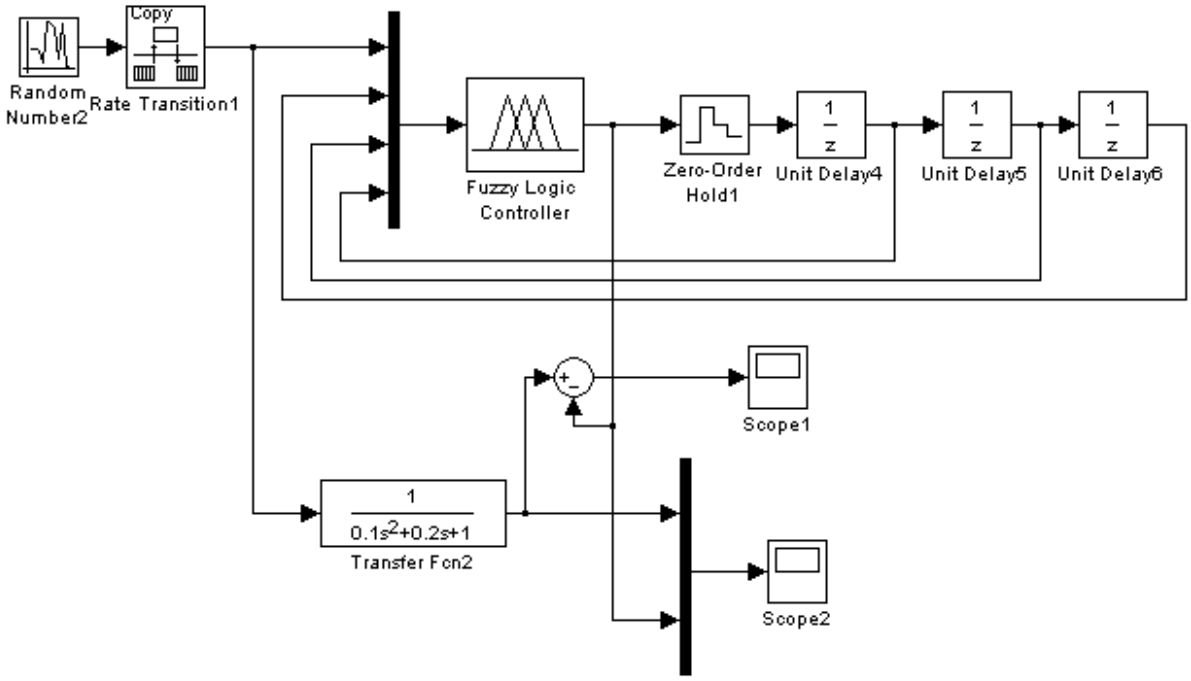


Рис.7.12.

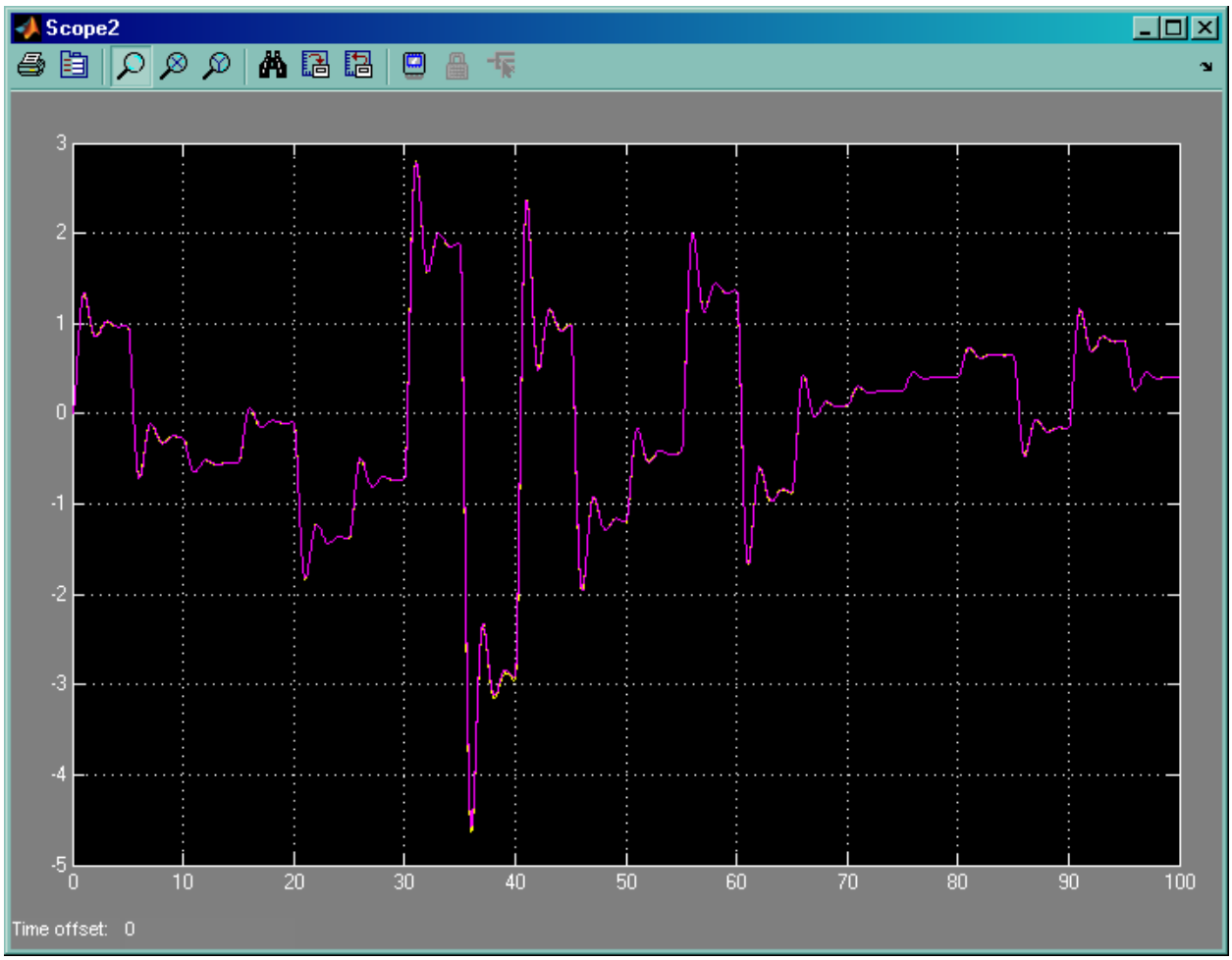


Рис.7.13.

Ошибка (рис.7.14).

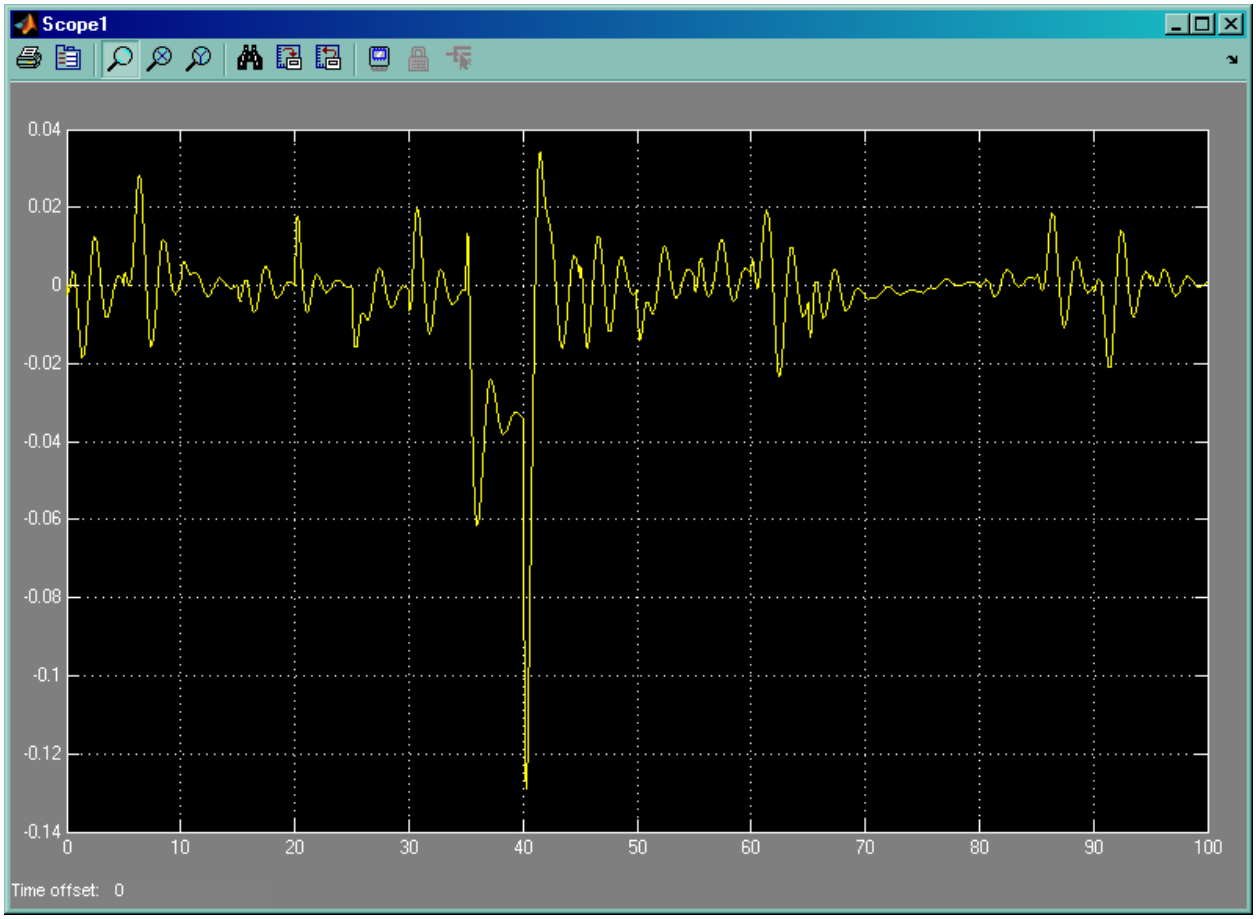


Рис.7.14.

Инверсное управление (рис.7.15).

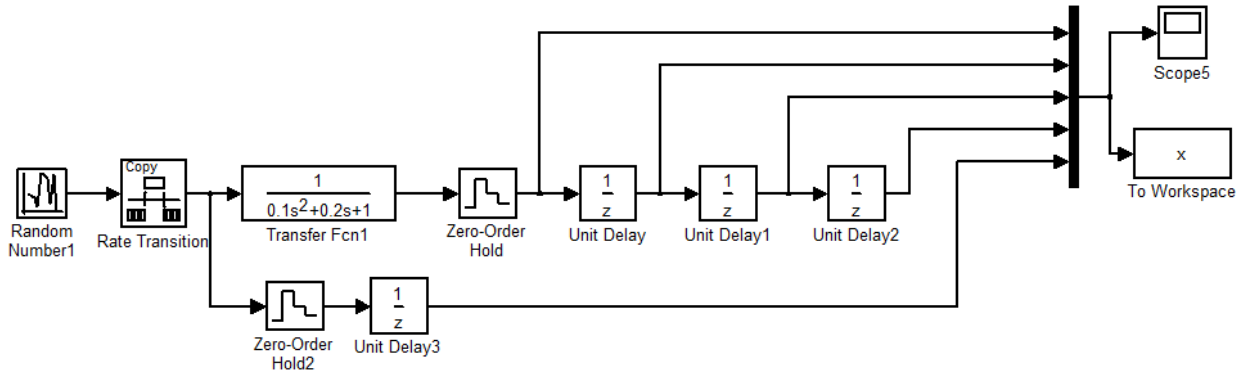


Рис.7.15.

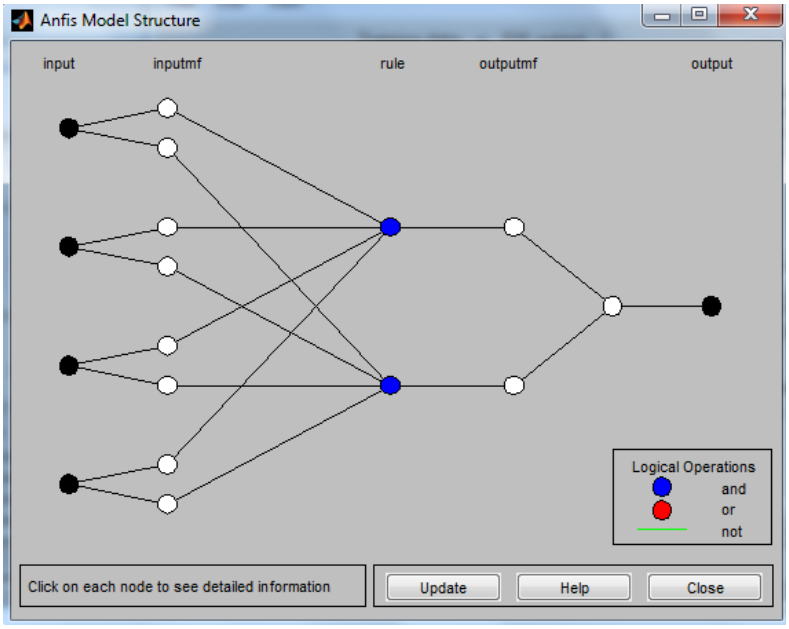


Рис.7.16.

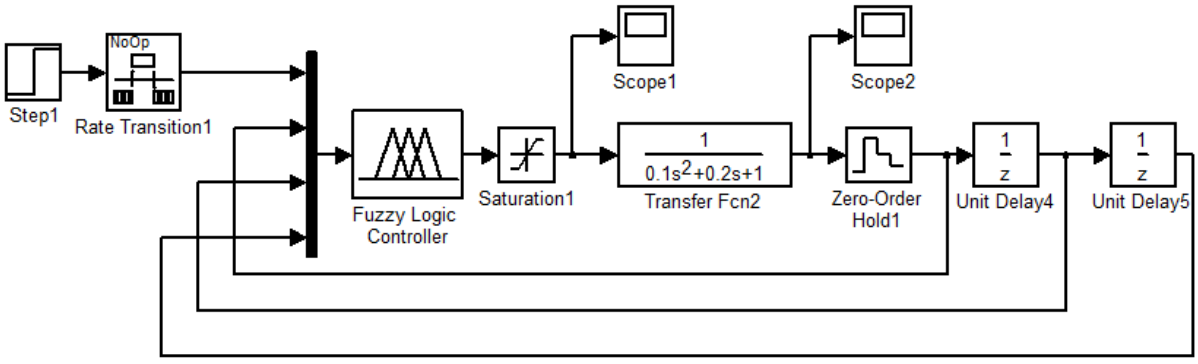


Рис.7.17.

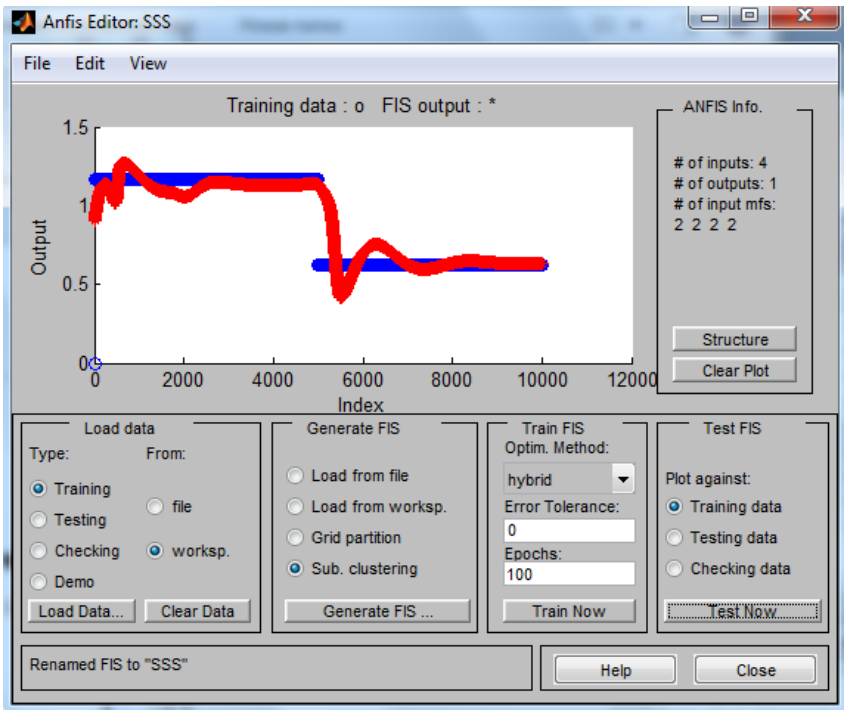


Рис.7.18.

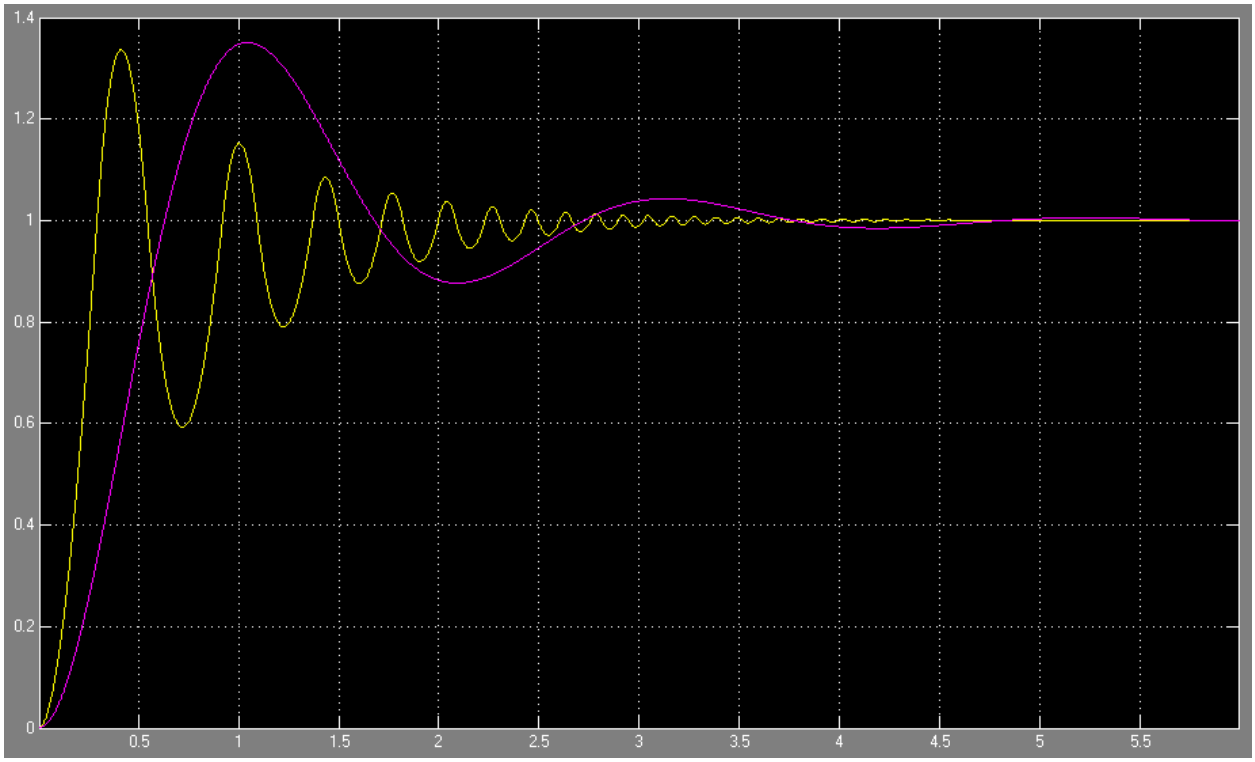


Рис.7.19.

Инверсное управление динамическим объектом (рис.7.20):

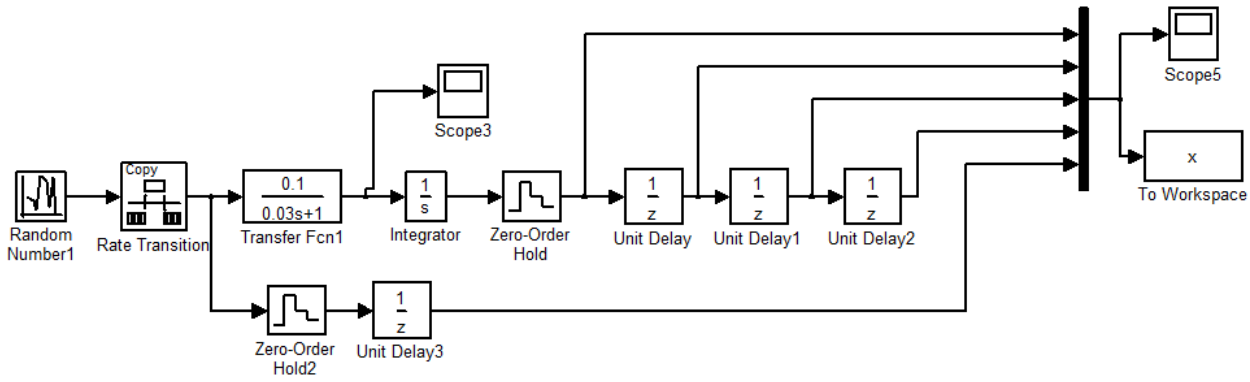


Рис.7.20.

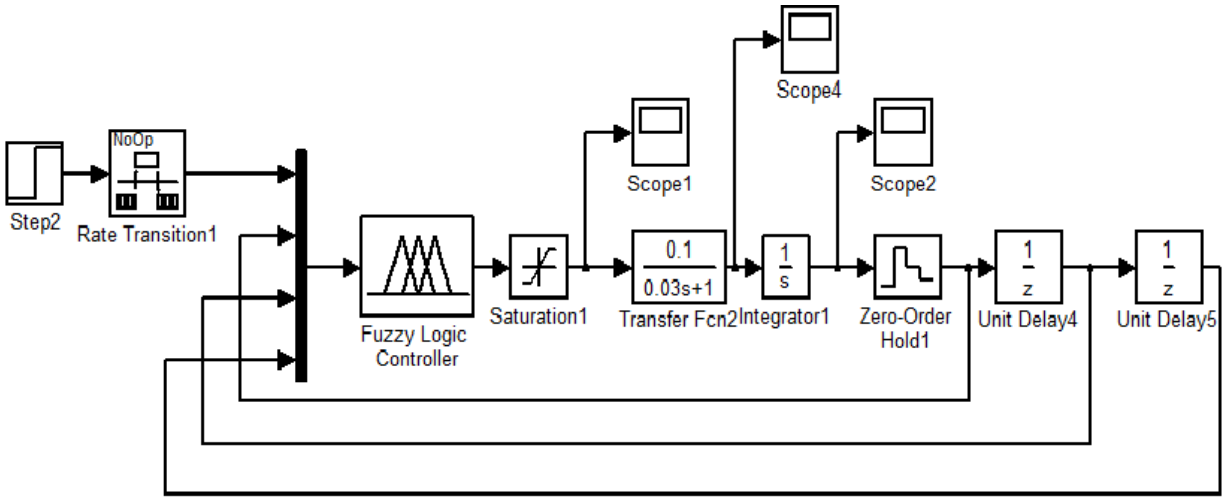


Рис.7.21.

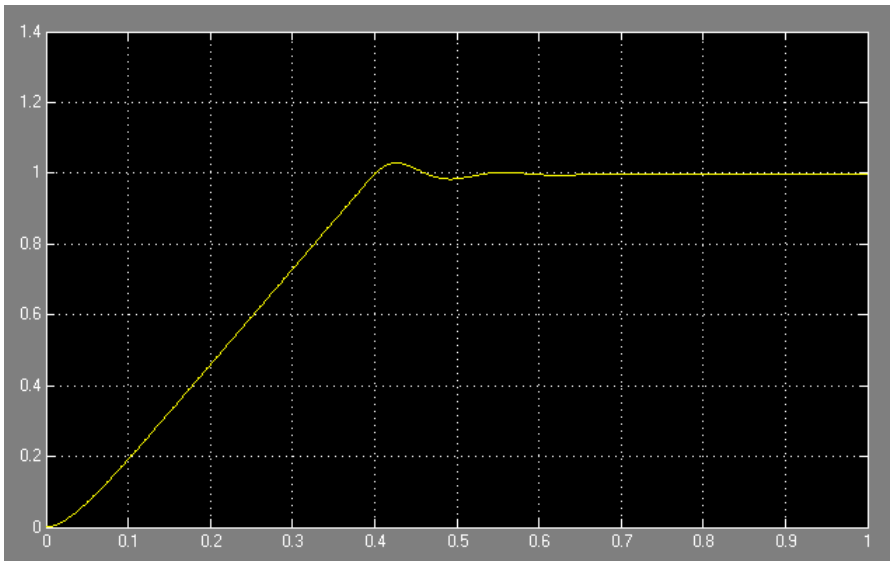


Рис.7.22.

6.ОТЧЕТ

Отчет должен содержать краткое описание ANFIS-редактора и решение заданий, представленных в разделе 4 настоящего описания.

7.КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что собой представляет сеть ANFIS?
2. Как функционирует первый слой ANFIS?
3. Какие методы обучения использует ANFIS?
4. К какому классу нейронных сетей относится ANFIS?
5. Как функционирует второй слой ANFIS?
6. Опишите механизм функционирования третьего слоя ANFIS.
7. Опишите механизм функционирования четвертого и пятого слоев ANFIS.
8. Опишите назначение слоев искусственной нейронной сети ANFIS.

Лабораторная работа №8

СОЗДАНИЕ И ОБУЧЕНИЕ НЕЙРОНЕЧЕТКОГО РЕГУЛЯТОРА, РЕАЛИЗУЮЩЕГО СТАНДАРТНЫЕ АЛГОРИТМЫ (П-, ПИ-, ПД-, ПИД- РЕГУЛЯТОРЫ)

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Знакомство с решением практической задачи создания и обучения нейросети заданной структуры с помощью инструмента Network/DataManager.

2. ОБОРУДОВАНИЕ

В лабораторной работе используются персональные компьютеры (не ниже Pentium 4) с установленным пакетом моделирования MatLab (версии не ниже 6.5).

3. ЗАДАНИЕ НА ВЫПОЛНЕНИЕ

Требуется сформировать нейросеть,имеющую два слоя (при использовании этого инструмента имеется ограничение на большее число слоев), 10 нейронов в первом слое с тангенциальными функциями активации (передаточными функциями). Нейросеть должна реализовывать нелинейную функцию:

4.ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ.

1. По заданным параметрам необходимо сформировать SimuLink - модель силовой системы ЭСП и оформить ее субмоделью (рис.8.1).

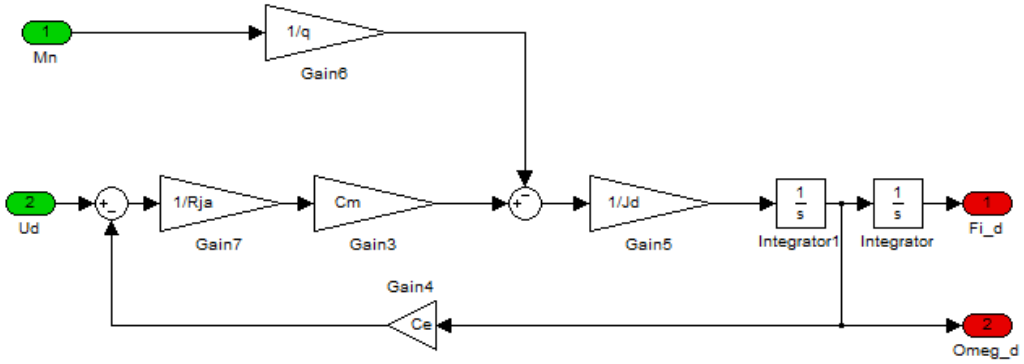


Рис8.1.

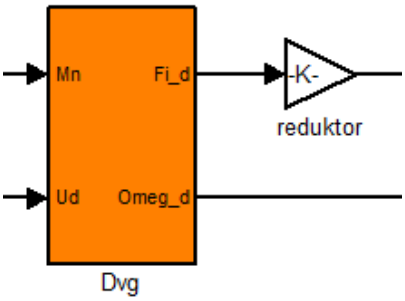


Рис.8.2.

Параметры силовой системы должны определяться m- функцией (рис.8.3)

```
% программа расчета динамических характеристик привода с ЭМУ

Ce = 0.056;                  % коэффициент противоЭДС двигателя
Cm = 0.056;
Rja = 0.14;
Mn = 1;
Jd = 0.0001;

Kmost = 1;
q = 1;
Tmost = 0.0002;
kred = 459;

Kd = 1/Ce;
Km = Rja/(Ce*Cm);
Td = Jd*Rja/(Ce*Cm);
gbx = 2;
Ke = 1000*20;
Ti = 1;
Td= 180*0;
T0 = 0.001;

q0 = Ke+Td/T0;
q1 = -(Ke +Td/T0 - T0/Ti);
q2 = Td/T0;


privod_1;
```

Рис.8.3.

Сформировать SimuLink - модель ЭСП с ПИД - регулятором. (При выполнении задания реализуется регулятор, заданный преподавателем). Схема SimuLink - модель ЭСП с ПИД - регулятором представлена на рис.8.4.

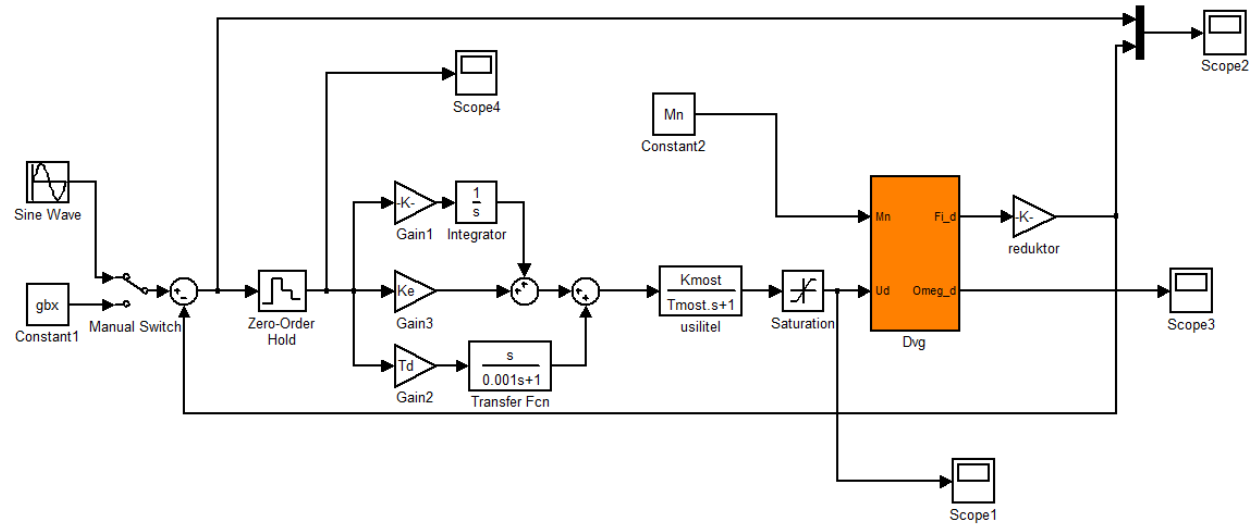


Рис.8.4.

С целью обучения нейронечеткого регулятора, приведенная выше схема дополняется датчиками информации о величине ошибки, определяемой в реальном времени и со смещением. При управлении динамическим объектом рекомендуется количество элементов задержки устанавливать равным порядку объекта управления. Информация о реализации процесса управления записывается в рабочее пространство MatLab (рис.8.5).

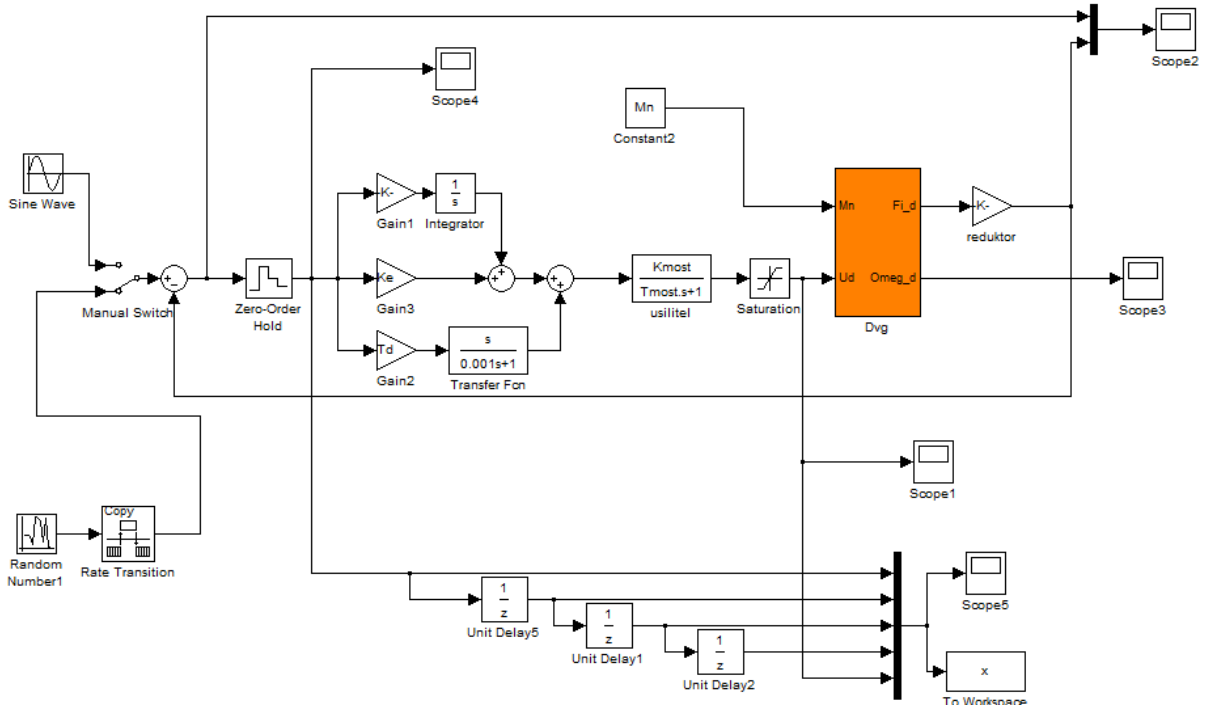


Рис.8.5.

Процедура обучения выполняется аналогично процедурам, описанным в

лабораторной работе №7.

Схема моделирования ЭСП с нейронечетким регулятором имитирующим ПИД-алгоритм управления, представлена на рис.8.6.

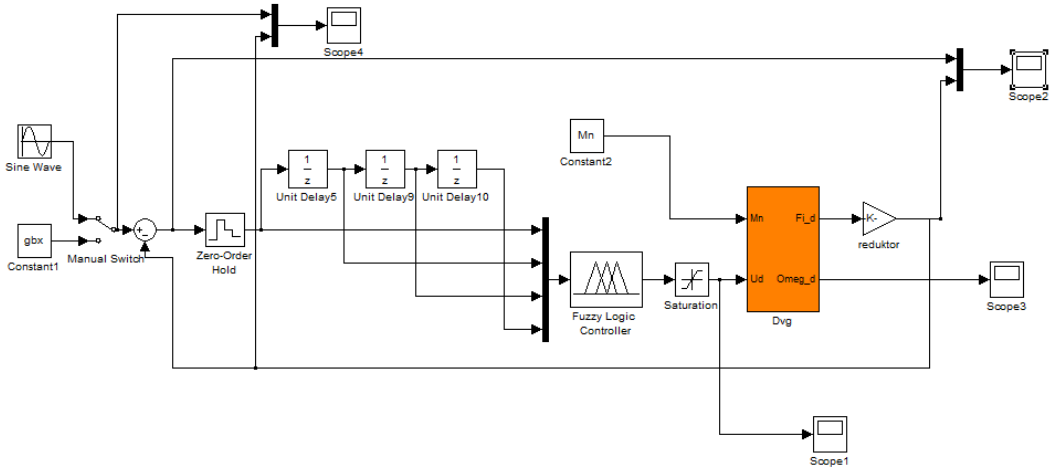


Рис.8.6.

Необходимо отметить, что для отработки типовых входных воздействий нейронечеткий регулятор должен настраиваться на свой тип сигнала. В противном случае необходимо формировать специальные тестовые сигналы.

Для отработки синусоидального входного воздействия настройка нейронечеткого регулятора проводилась на основе гармонического сигнала.

Поверхность управления, сформированная в результате обучения нейронечеткой сети для этого случая представлена на рис.8.7.

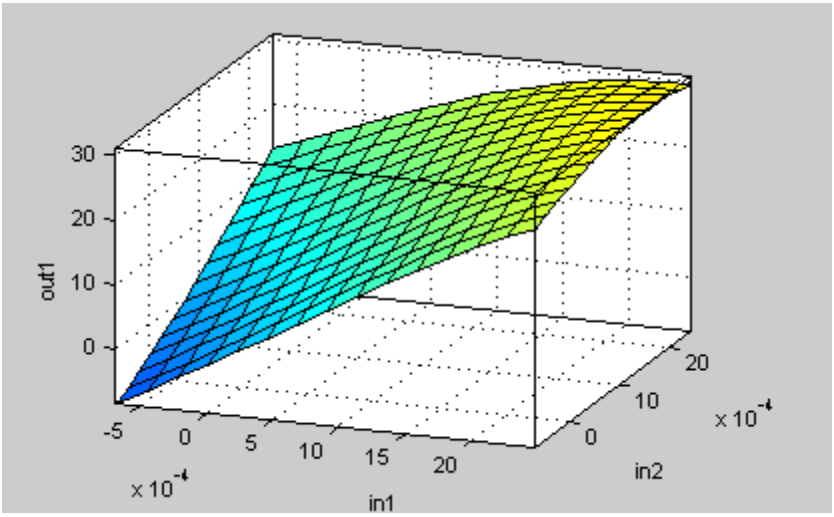


Рис.8.7.

Результаты отработки гармонического входного сигнала и ошибка отработки с применением аналогового и нейронечеткого регулятора представлены на рис.8.8а,б,в (соответственно).

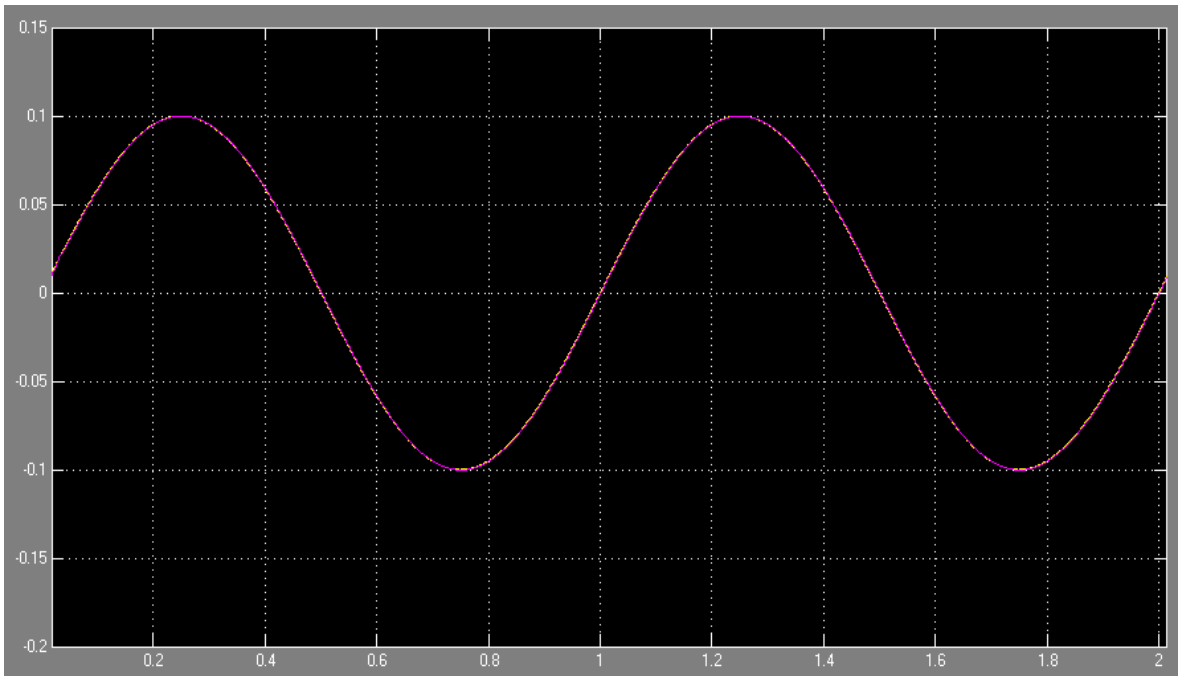


Рис.8.8.а

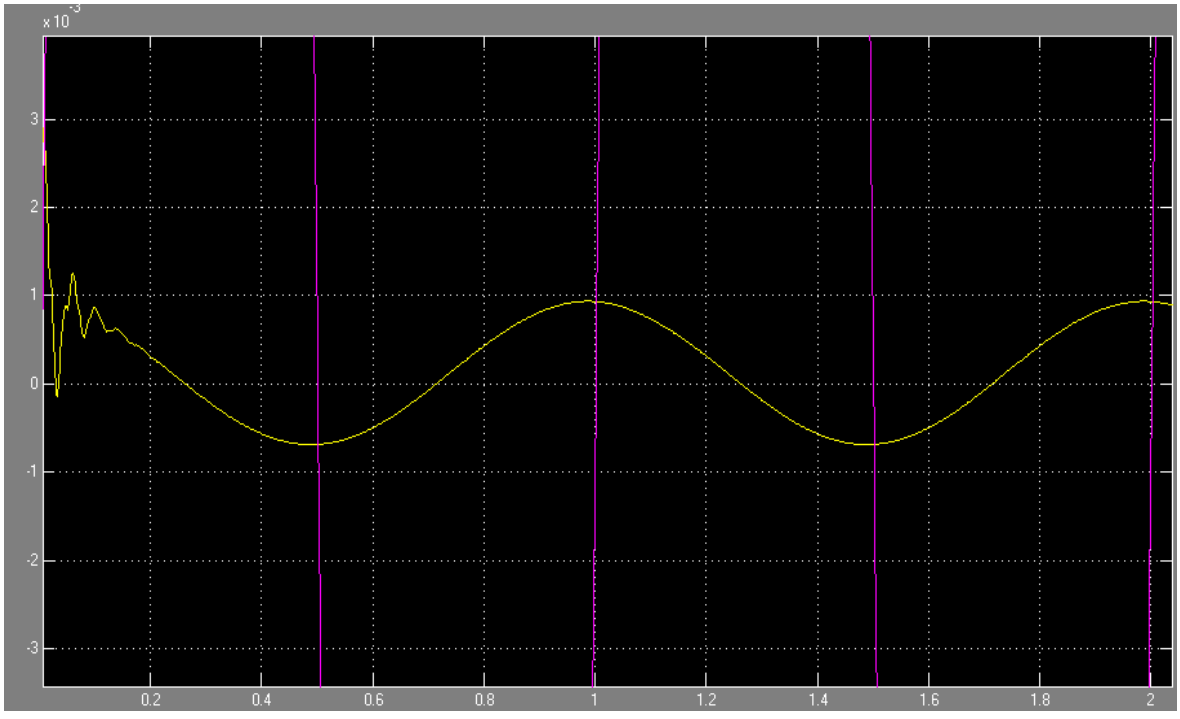


Рис.8.8.б.

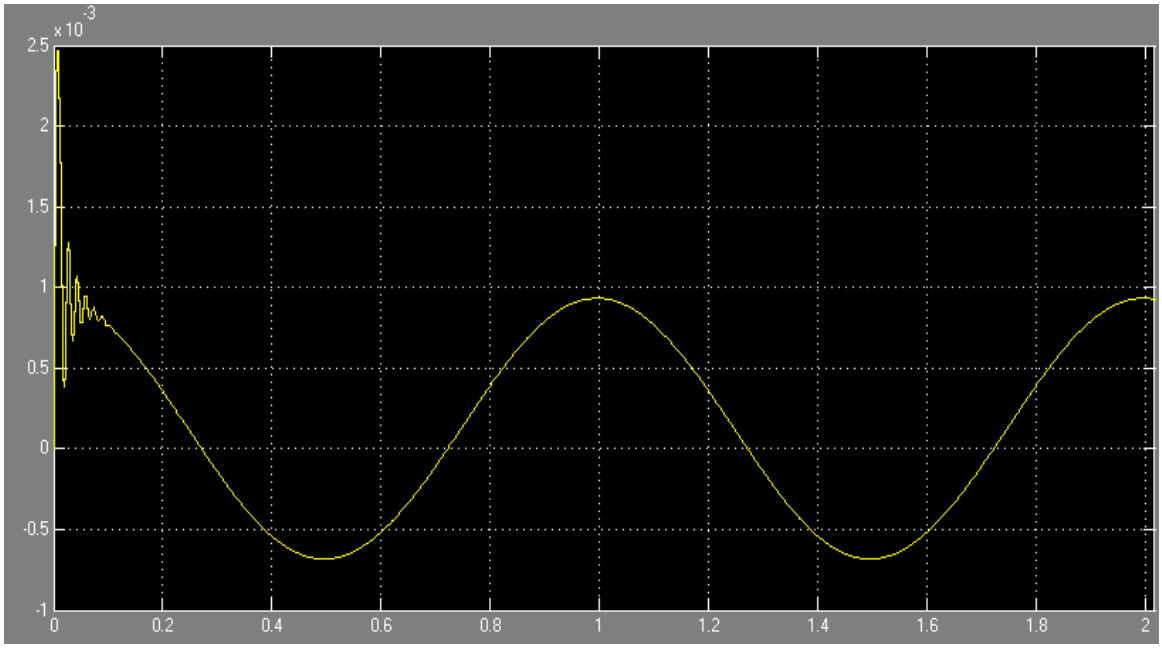


Рис.8.8.в

Для отработки ступенчатых входных воздействий нейронечеткий регулятор обучался на примере отклика на случайный входной сигнал. Сформированная поверхность управления для этого случая имеет вид, представленный на рис.8.9.

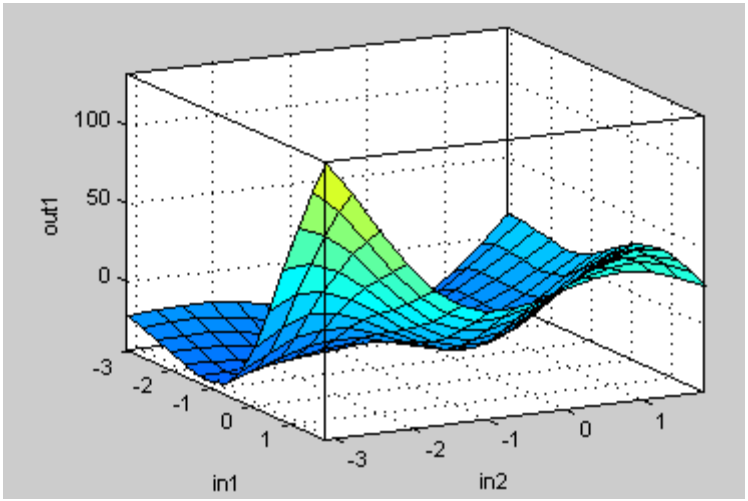


Рис.8.9.

Результаты расчета переходных процессов в ЭСП с нейронечетким регулятором представлены на рис.8.10.а и б.

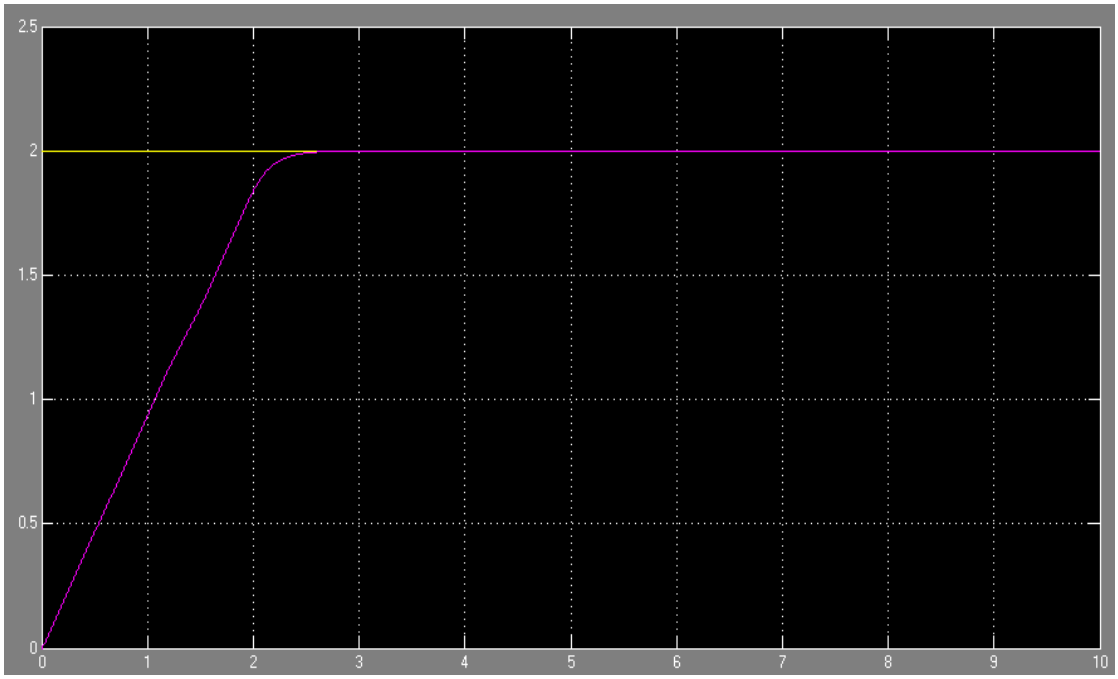


Рис.8.10.а.

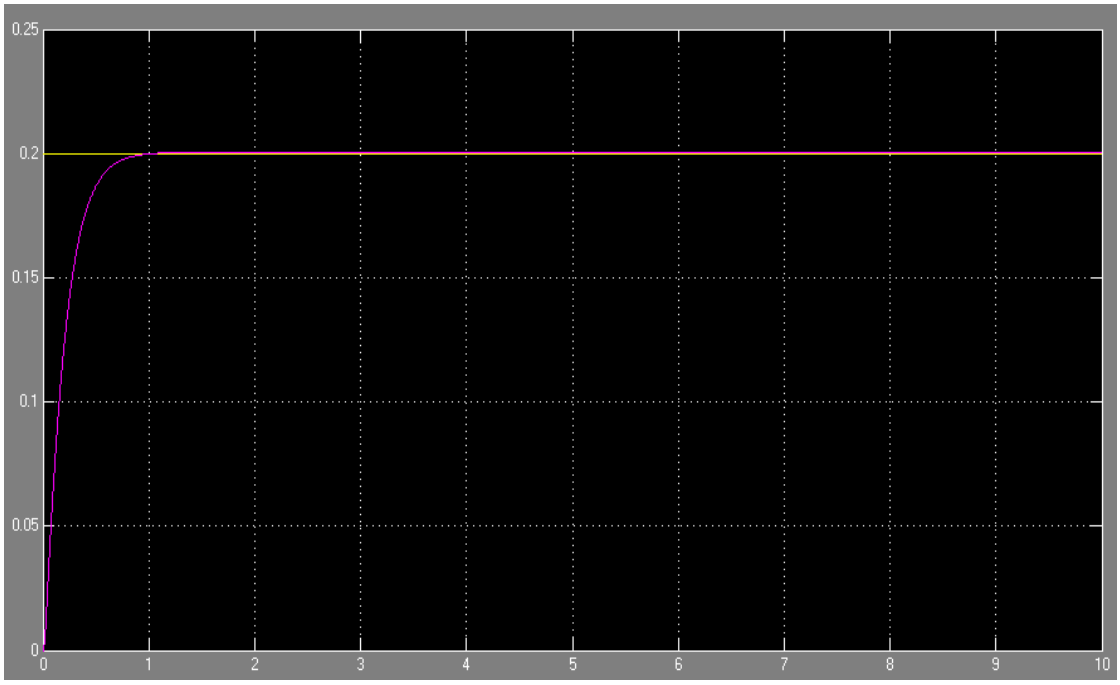


Рис.8.10.б.

.

5.ОТЧЕТ

Отчет должен содержать решение заданий, представленных в разделе 3 настоящего описания.

6.КОНТРОЛЬНЫЕ ВОПРОСЫ

1. На каком примере нейронечеткий регулятор обучается для отработки гармонического сигнала?
2. На каком примере нейронечеткий регулятор обучается для отработки ступенчатого входного сигнала?
3. Какое влияние на переходный процесс оказывают различные коэффициенты ПИД регулятора?
4. Что такое нечеткий супервизор?
5. Что такое скользящий режим нечеткого регулятора?
6. В чем заключается проблема использования обычных регуляторов П-типа?

СПИСОК ЛИТЕРАТУРЫ:

1. Методы классической и современной теории автоматического управления : учебник для вузов : в 5 т.. Т. 5. Методы современной теории автоматического управления / К. А. Пупков [и др.] ; под ред. К. А. Пупкова, Н. Д. Егупова. 2-е изд., перераб. и доп. М. : МГТУ им.Баумана, 2004. 784 с. : ил. (Методы теории автоматического управления/под общ.ред.К.А.Пупкова) . ISBN 5-7038-2190-8 (Т. 5) ((в пер.)) . ISBN 5-7038-2194-0. [*12 экземпляров ТулГУ*]
2. Гаскаров, Д.В. Интеллектуальные информационные системы : Учебник для вузов / Д.В.Гаскаров. М. : Высш.шк., 2003. 431с. ISBN 5-06-004611-7 /в пер./ : 111.00. [*10 экземпляров ТулГУ*]
3. Ясницкий, Л.Н. Введение в искусственный интеллект : учеб. пособие для вузов / Л.Н. Ясницкий. М. : Академия, 2005. 176с. : ил. (Высшее профессиональное образование: Информатика и вычислительная техника) . ISBN 5-7695-1958-4 : 139.00. [*5 экземпляров ТулГУ*]
4. Усков, А.А. Интеллектуальные технологии управления. Искусственные нейронные сети и нечеткая логика / А.А.Усков, А.В.Кузьмин. М. : Горячая линия-Телеком, 2004. 143с. : ил. ISBN 5-93517-181-3 : 123.00. [*3 экземпляра ТулГУ*]
5. Смолин, Д.В. Введение в искусственный интеллект: конспект лекций / Д.В.Смолин. М. : ФИЗМАТЛИТ, 2004. 208с. : ил. ISBN 5-9221-0513-2 /в пер./ : 236.00. [*3 экземпляра ТулГУ*]
6. Павлов, С. Н. Системы искусственного интеллекта. Часть 1 : учебное пособие / С. Н. Павлов. Системы искусственного интеллекта. Часть 1, Весь срок охраны авторского права. Томск : Томский государственный университет систем управления и радиоэлектроники, Эль Контент, 2011. 176 c. ISBN 978-5-4332-0013-5. [*ЭБС "IPRbooks"*].
7. Пальмов, С. В. Интеллектуальные системы и технологии : учебное пособие / С. В. Пальмов. Интеллектуальные системы и технологии, Весь срок охраны авторского права. Самара : Поволжский государственный университет телекоммуникаций и информатики, 2017. 195 c. ISBN 2227-8397. [*ЭБС "IPRbooks"*].
8. Назаров, Дмитрий Михайлович. Интеллектуальные системы: основы теории нечетких множеств : Учебное пособие для вузов / Назаров Д. М., Конышева Л. К. 3-е изд., испр. и доп. Москва: Юрайт, 2020. 186 с. (Высшее образование) . ISBN 978-5-534-07496-3 : 519.00. [*ЭБС "Юрайт"*]
9. Васильев, В.И.Уфимский авиацион. техн. ин-т. Интеллектуальные системы управления с использованием нечеткой логики : учеб.пособие / В.И.Васильев, Б.Г.Ильясов; Уфимский авиац.техн.ун-т. Уфа, 1995. 100с. : ил. : 6.00. [*1 экземпляр ТулГУ*]
10. Системы искусственного интеллекта. Практический курс : учеб. пособие для вузов / В. А. Чулюков [и др.]. М. : БИНОМ. Лаборатория знаний: ФИЗМАТЛИТ, 2008. 293 с : ил. (Адаптивные и интеллектуальные системы) . ISBN 978-5-94774-731-7 ((в пер.)) . [*1 экземпляр ТулГУ*]
11. Аверкин, А.Н. Нечеткие множества в моделях управления и искусственного интелекта / А.Н.Аверкин [и др.]; под ред. Д.А.Поспелова. М.: Наука, 1986. 312с.: ил. (Проблемы искусственного интеллекта; 8) . ISBN /В пер./ : 2.40. [*1 экземпляр ТулГУ*]
12. Батыршин, И.З. Нечеткие гибридные системы.Теория и практика / Батыршин И.З.[и др.];под ред.Н.Г.Ярушкиной. М. : Физматлит, 2007. 208с. (Информационные и компьютерные технологии) . ISBN 978-5-9221-0786-0 /в пер./ : 130.00. [*1 экземпляр ТулГУ*]
13. Пенькова, , Т. Г. Модели и методы искусственного интеллекта : учебное пособие /

Т. Г. Пенькова, Ю. В. Вайнштейн. Модели и методы искусственного интеллекта, 2025-10-09. Красноярск : Сибирский федеральный университет, 2019. 116 c. ISBN 978-5-7638-4043-8. [*ЭБС "IPRbooks"*].

14. Трофимов, В. Б. Экспертные системы в АСУ ТП: учебник / В. Б. Трофимов, И. О. Темкин. Экспертные системы в АСУ ТП, 2025-08-03. Москва, Вологда : Инфра-Инженерия, 2020. 284 c. ISBN 978-5-9729-0480-8 [*ЭБС "IPRbooks"*].
15. Лубенцова, Е. В. Системы управления с динамическим выбором структуры, нечеткой логикой и нейросетевыми моделями: монография / Е. В. Лубенцова. Системы управления с динамическим выбором структуры, нечеткой логикой и нейросетевыми моделями, Весь срок охраны авторского права. Ставрополь : Северо-Кавказский федеральный университет, 2014. 248 c. ISBN 978-5-88648-902-6. [*ЭБС "IPRbooks"*].
16. Сырецкий, Г. А. Моделирование систем. Часть 2. Интеллектуальные системы : учебное пособие / Г. А. Сырецкий. Моделирование систем. Часть 2. Интеллектуальные системы, 2025-02-05. Новосибирск : Новосибирский государственный технический университет, 2010. 80 c. ISBN 978-5-7782-1341-8. [*ЭБС "IPRbooks"*].