Министерство образования и науки РФ

Санкт-Петербургский государственный
электротехнический университет "ЛЭТИ"

ИССЛЕДОВАНИЕ НЕЧЕТКИХ И НЕЙРОСЕТЕВЫХ СИСТЕМ

Методические указания к лабораторным работам по дисциплине
«Нечеткие и нейросетевые системы управления»

Санкт-Петербург
Издательство СПбГЭТУ "ЛЭТИ"
2013

УДК 62.50
ББК З 813я7
П 54

Исследование нечетких и нейросетевых систем: Методические указания к лабораторным работам по дисциплине «Нечеткие и нейросетевые системы управления» / Сост.: Н. Д. Поляхов, И. А. Приходько, А. А. Карачев, А. Д. Стоцкая, СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2013. 28 с.

Содержат программы и методики выполнения лабораторных работ, посвященных аппроксимации функций с помощью нейронных и нейронечетких сетей, построению и исследованию нечетких и нейросетевых систем управления. Описание лабораторных работ № 1–5 ориентировано на пакет MATLAB 6.5, работы № 6 – на пакет MATLAB 7.10.

Предназначены для студентов, обучающихся по направлению "Управление в технических системах".

Утверждено
редакционно-издательским советом университета
в качестве методического указания

ISBN 5-7629-0541-1

Лабораторная работа № 1

ПОСТРОЕНИЕ И ИССЛЕДОВАНИЕ НЕЧЕТКОГО РЕГУЛЯТОРА НА ОСНОВЕ АЛГОРИТМА ЗАДЕ–МАМДАНИ

Цель работы – ознакомление с пакетом *Fuzzy Logic Toolbox* (MATLAB), построение нечеткого регулятора (НР – *Fuzzy Logic Controller*) на основе алгоритма Заде–Мамдани в пакете *Fuzzy Logic Toolbox*. Эффективность управления системы с нечетким регулятором проверяется по результатам моделирования в *Toolbox Simulink*.

1.1. Основные сведения

Нечеткими называют регуляторы, ориентированные на обработку нечетких правил с целью поиска решения задачи управления. *Нечеткий регулятор* включает: фаззификатор, блок правил, блок выработки решения и дефаззификатор (рис. 1.1). Иногда блок правил и блок выработки решения объединяют в один блок [1], [2] .

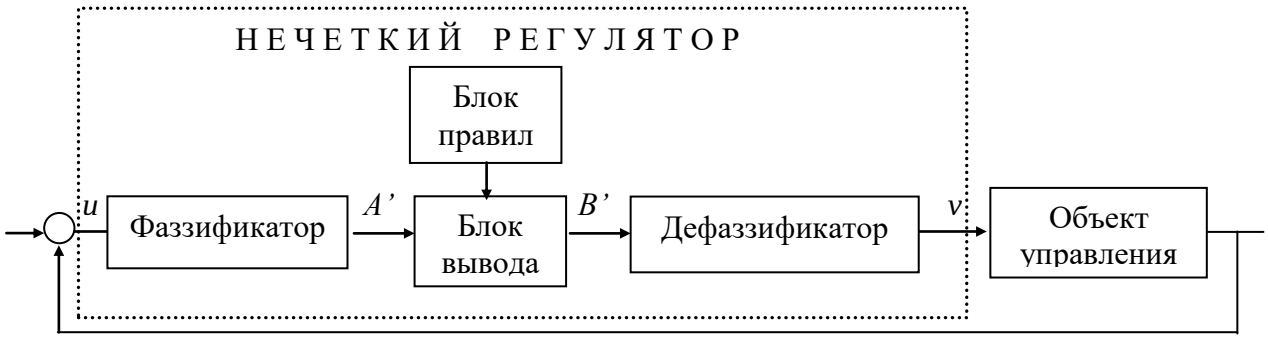


Рис. 1.1

Фаззификатор. Система управления с нечеткой логикой оперирует нечеткими множествами. Поэтому реальные (четкие) значения входных и выходных переменных регулятора (u, v) подлежат операции *фаззификации*, в результате которой реальным значениям переменных ставятся в соответствие нечеткие множества, т.е. *фаззификация* – это процедура определения степени принадлежности $\mu_A(u_0)$ элемента u_0 нечеткому множеству A. В задачах управления часто применяется операция фаззификации типа *синглтон* (singleton): $\mu_{A'_j}(u_{j0}) = 1,\ \mu_{A'_j}(u_j \neq u_{j0}) = 0,\ j = \overline{1,n}$.

Блок правил состоит из множества нечетких условных высказываний (нечетких правил)

R_i: если u_1 есть A_{1i} и u_2 есть A_{2i} и….. u_n есть A_{ni}, то v есть B_i,

где $i=\overline{1, N}$ – число нечетких правил; n – число входных переменных нечеткого регулятора; u_i, v – входные и выходная переменные; A_{1i}, A_{2i} … A_{ni} – нечеткие подмножества универсальных множеств U_1, U_2,..., U_n; B_i, – нечеткие подмножества универсального множества V. Нечеткие правила включают нечеткую логическую операцию "и" (t-норму) и нечеткую операцию импликации "если…, то". Импликация R_i = (A_{1i} и A_{2i} и … A_{ni}) $\rightarrow B_i$ состоит из двух частей: *посылки*, которая содержит набор условий: «если u_1 есть A_{1i} и u_2 есть A_{2i} и….. u_n есть A_{ni}» и *заключения*, которое задает вывод: «то v есть B_i». Нечеткое правило также можно интерпретировать как нечеткое отношение $R \subset U \times V$.

В блоке вывода определяется нечеткое управляющее воздействия по заранее сформулированным в блоке правил нечетким правилам посредством композиционного правила вывода.

Схема нечеткого логического вывода:

посылка 1 (импликация):

R_i: если u_1 есть A_{1i} и u_2 есть A_{2i} и….. u_n есть A_{ni}, то v есть B_i;

посылка 2 (условие): u_1 есть $A_1^{'}$ и u_2 есть $A_2^{'}$ и….. u_n есть $A_n^{'}$;

следствие: v есть B'.

Композиционное правило нечеткого вывода

$$B' = (A_1^{'}, A_2^{'}, \dots, A_n^{'}) \circ \bigcup_{i=1}^{k} R_i = \bigcup_{i=1}^{k} (A_1^{'}, A_2^{'}, \dots, A_n^{'}) \circ R_i = \bigcup_{i=1}^{k} B_i^{'}, \qquad (1.1)$$

где $B_i^{'} = A' \circ R_i = A' \circ (A_{1i}$ и A_{2i} и … $A_{ni}) \rightarrow B_i$, $A^{'} = A_1^{'}, A_2^{'}, \dots, A_n^{'}$.

Функция принадлежности (ФП) результата нечеткого вывода

$$\mu_{B'}(v) = \max_i \mu_{B_i^{'}}(v).$$

Если в посылке 2 входные нечеткие множества синглтоны, а в качестве импликации используется импликация Мамдани, то ФП результата нечеткого вывода i-го нечеткого правила $\mu_{B_i^{'}}(v) = \min(\alpha_i, \mu_{B_i}(v))$, $i = \overline{1,n}$, где

$$\alpha_i = \min(\mu_{A_{1i}}(u_{10}), \mu_{A_{2i}}(u_{20}), \dots, \mu_{A_{ni}}(u_{n0})). \qquad (1.2)$$

Если применяется импликация Ларсена, то ФП – $\mu_{B_i'}(v) = \alpha_i \ \mu_{B_i}(v)$.

В дефаззификаторе вычисляется четкое (детерминированное) управляющее воздействие на основе результата нечеткого вывода (1.1).

1.2. Программа работы

Для открытия *Fuzzy Toolbox* необходимо набрать в командной строке основного рабочего поля команду *fuzzy*, что вызовет открытие окна *FIS Editor: Untitled, c*одержащго разделы: *File* (команды работы с файлами и опции настройки системы), *Edit* (команды редактирования информации, отображенной в рабочем поле), *View* (взгляд – команды, предназначенные для выбора рабочего поля).

1. Построение блок-схемы НР: в выпадающем меню раздела *Edit* указать *Add Variable…Input* (добавить переменную... вход), при этом схема изображенного НР будет содержать 2 входа. Активизировать блок *input1*, в поле настройки *Current Variable* (текущая переменная) в графе *Name* указать *e* (ошибка управления), в блоке *input2* записать *de* (производная ошибки), в блоке *output – vf* (выход НР).

2. Формирование графиков ФП нечетких множеств (фаззификация): в выпадающем меню раздела *Edit* выбрать команду *Membership Function* – Ctrl+2 (редактирование функции принадлежности). При этом откроется окно *Membership function Editor* (редактор функции принадлежности), включающее (*Fis Variables*) переменные (*e, de, vf*), график ФП (*Membership functions plots*) и два поля настройки: переменных (*Current Variable*) и ФП (*Current Membership Functions*)). Построить графики ФП последовательно для первого входа (*e*), второго входа (*de*) и выхода (*vf*), т.е. последовательно активизировать блоки *e, de, vf* и выполнить следующую последовательность действий: а) в окне *Membership function Editor* указать границы изменения входных переменных, для этого в поле настройки C*urrent variable* изменить границы в графе *Range* (диапазон) и *Display Range*: *e , de, vf* – [– 5 5]; б) в выпадающем меню раздела *Edit* выбрать команду *Remove All Mfs* (удалить все ФП); в выпадающем меню раздела *Edit* выбрать команду *Add MFs* (добавить ФП). При этом откроется окно *Membership functions.* В графе *MF type* указать *trimf* (треугольная ФП), в графе *Number of MFs* указать 5 (число ФП). ОК;

в) в окне *Membership Function Editor:* на графике ФП активизировать одну из ФП, в поле настройки C*urrent Membership Function* указать: *Name* (имя ФП) – ОБ, ОМ – отрацительные большое и малое соответственно; Н – нулевое; ПБ, ПМ – положительные большое и малое соответственно; *Type* (тип): ОМ, Н, ПМ – *trimf* (треугольная ФП); ОБ – *zmf* (Z-типа); ПБ – *smf* (S-типа); *Params* (параметры) ФП задаются в соответствии графиком на рис. 1.2

3. Формирование нечетких правил блока правил: в выпадающем меню раздела *Edit* окна *FIS Editor: Untitled* выбрать команду *Rules* – Ctrl+3; в окне *Rule Editor* составить правила в соответствии с таблицей. Например, *If e* is ОБ and *de is* ОБ Then *vf is* ОБ. Использовать кнопку *Add Rule* (добавить правило). Другие кнопки: *Change Rule* (изменить правило), *Delete Rule* (удалить правило); графе Connection установить *"And*" для всех правил; в графе *Weight* (вес) – 1.

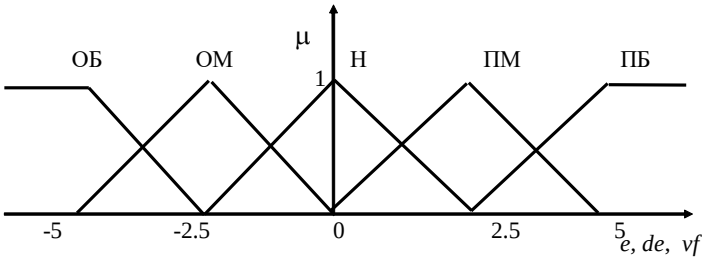


Рис. 1.2

Таблица

de / e	ОБ	ОМ	Н	ПМ	ПБ
ПБ	Н	ПМ	ПМ	ПМ	ПБ
ПМ	ОМ	Н	ПМ	ПМ	ПМ
Н	ОМ	ОМ	Н	ПМ	ПМ
ОМ	ОМ	ОМ	ОМ	Н	ПМ
ОБ	ОБ	ОМ	ОМ	ОМ	Н

4. Задание правила нечеткого вывода (блока вывода). Для выполнения в выпадающем меню раздела *Edit* выбрать команду *FIS Properties*- Ctrl+1. При этом откроется окно *FIS Editor.*

4.1. Задание логических операций "и" и импликации: в выпадающем меню графы *And methode* установить *min*; в графе *Implication – min.*

4.2. Задание способа определения выходного нечеткого множества (B' по B'_i – агрегирование): в выпадающем меню графы *Aggregation* установить *max.*

5. Задание способа дефаззификации (способ определения реального управляющего воздействия). В окне *FIS Editor* в графе *Defazzification* установить способ дефаззификации: *centroid* (метод центра тяжести). Другие способы: *bisector* (центр площади, биссектриса площади); *mean of max-mom* (метод среднего максимума); *largest of max-lom* (метод наибольшего максимума); *smallest of max-som* (метод наименьшего максимума); Custom (способ дефаззификации, созданный пользователем).

6. Сохранение полученного файла с расширением .fis на диске. Для этого в выпадающем меню раздела *File* окна *FIS Editor* выбрать команду *Export To Disk* (Ctrl+S). Для сохранения .fis файла в рабочем пространстве использовать команду *Export To Workspace* (Ctrl+T).

7. Построить схему моделирования в Toolbox Simulink.

В качестве примера здесь приводится сравнение пропорционально-дифференциального (ПД) и нечеткого регуляторов для объекта управления третьего порядка. Структурная схема системы управления представлена на рис. 1.3. Оценивается эффективность регуляторов в условиях изменения параметров объекта управления, при воздействии внешнего возмущения и введении нелинейности – звена с насыщением ±0.15.

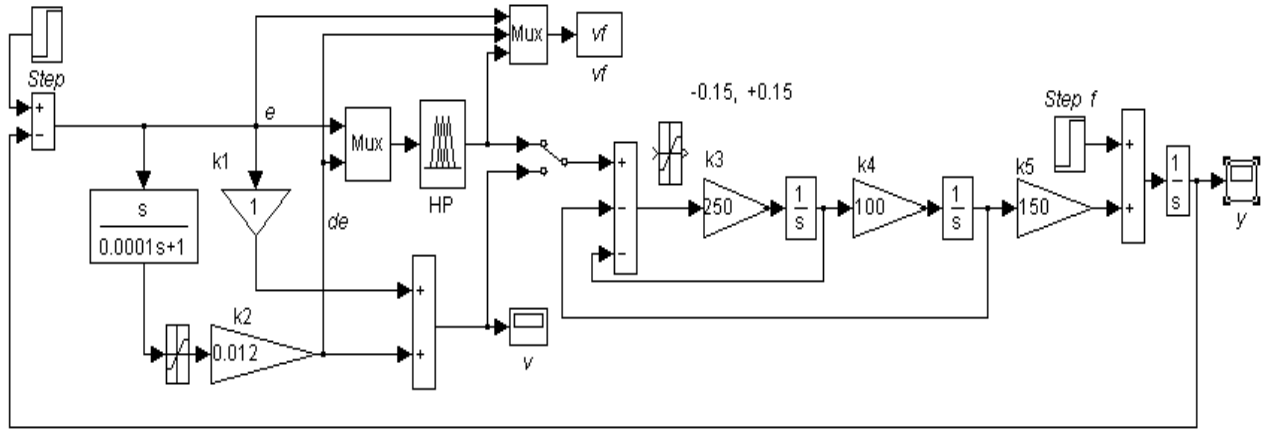


Рис. 1.3

Параметры объекта управления в номинальном режиме: $k_3 = 250$, $k_4 = 100$, $k_5 = 150$. Закон управления ПД-регулятора: $u = k_1 e + k_2 de$, где $k_1 = 1$, $k_2 = 0.012$. НР (*Fuzzy Logic Controller*) взять из библиотеки *Library Simulink* раздела *Fuzzy Logic Toolbox*. Для открытия блока параметров НР (*Block Parameters: Fuzzy Logic Controller*) активизировать блок *Fuzzy Logic Controller*. В блоке параметров указать имя используемого .fis файла НР.

8. Эффективность управления проверить по результатам моделирования в *Simulink*. Сравнить качество переходных процессов в САУ с линейным и нечетким регуляторами: а) в номинальном режиме; б) при наличии ограниченных изменений параметров (k3 =125; k4=200; k5=500); в) при введении звена насыщением ± 0.15 для номинальных параметров; с) при воздействии внешних возмущений.

9. Оценить влияние на качество переходных процессов системы управления изменения параметров НР: а) границ входных переменных НР; б) типа функции принадлежности; в) границ ФП; г) блока правил; д) способа реализации логической операции "и"; е) способа реализации логической операции импликации; ж) способа агрегирования; з) способа дефаззификации.

10. Формирование файла данных. Файл данных с расширением .dat в лабораторной работе включает три числовых столбца данных: 2 входа и выход. Для формирования файла данных с расширением .dat используются числовые значения входов: *e, de* и желаемого выхода. Схема моделирования должна включать блок сохранения данных в рабочей области - *To Workspase* (блок из раздела *"Sinks"* (Получатели) библиотеки *Library: Simulink*). В блоке сохранения (*To Workspase*) должно быть записано имя переменной (в лабораторной работе – *vf*) вместо *simout*, в графе *Save format* (сохранить формат) вместо *Structure* (структура) указать *Array* (область). Числовые значения данных можно получить, если в командной строке основного рабочего поля записать имя переменной – *vf*. Полученные значения сохранить с расширением .dat.

1.3. Содержание отчета

Отчет должен включать описание нечеткого регулятора (графики ФП, блок правил, способ дефаззификации), результаты моделирования, приведенные в таблице (показатели качества управления), выводы по работе.

1.4. Контрольные вопросы

1. Как задается логико-лингвистическое описание систем управления?
2. Какие способы задания нечеткой логической операции «и» и нечеткой операции импликации используются в *Fuzzy Logic Toolbox?*
3. Какие способы агрегирования применяются в *Fuzzy Logic Toolbox*?
4. Какие способы дефаззификации используются в *Fuzzy Logic Toolbox*?
5. Объяснить способ составления таблицы набора правил.

Лабораторная работа № 2

ПОСТРОЕНИЕ И ИССЛЕДОВАНИЕ НЕЧЕТКОГО РЕГУЛЯТОРА НА ОСНОВЕ АЛГОРИТМА ТАКАГИ-СУГЕНО

Цель работы – построение нечеткого регулятора на основе правила нечеткого логического вывода Такаги–Сугено в пакете *Fuzzy Logic Toolbox*. Эффективность управления проверяется по результатам моделирования в *Toolbox Simulink*.

2.1. Основные сведения

Нечеткие правила составляются на основе условных высказываний вида [1], [3]: если u_1 есть A_1 и u_2 есть A_2 и….. u_n есть A_n, то $v = f(u_1, u_2, ..., u_n)$, где все обозначения такие же, как в (1.1), $v = f(u_1, u_2, ..., u_n)$ – четкая функция. В качестве заключений нечетких правил (заключений операции импликации v) обычно используют уравнения первого и нулевого порядков.

Схема нечеткого логического вывода:

посылка 1 (импликация):

R_i: если u_1 есть A_{1i} и u_2 есть A_{2i} и….. u_n есть A_{ni}, то

$v_i = b_{1i}u_1 + b_{2i}u_2 ... + b_{ni}u_n + b_{0i}$;

посылка 2: u_1 есть u_{10} и u_2 есть u_{20} …и u_n есть u_{n0};

следствие: v есть v'.

Нечеткие правила для алгоритма Такаги–Сугено нулевого порядка задаются на основе условных высказываний вида

если u_1 есть A_1 и u_2 есть A_2 и….. u_n есть A_n, то $v = b_0$.

Четкое (детерминированное) управляющее воздействие определяется по формуле средневзвешенного значения $v' = \frac{\sum_i \alpha_i v_i}{\sum_i \alpha_i}$, где α_i как в (1.2).

2.2. Программа работы

1. Построение блок-схемы НР регулятора Такаги–Сугено: в *Fuzzy Toolbox* открыть *New FIS Sugeno* раздела меню *File*; построить блок-схему регулятора аналогично рассмотренному в лабораторной 1.

2. Редактирование адаптивного нейронечеткого регулятора (*Anfis*): в выпадающем меню раздела *Edit* указать *Anfis*; в окне *ANFIS Editor: Untitled 2* (редактор адаптивного нейронечеткого регулятора) в колонке *Load data* (загрузка данных) в графе *Type* отметить *training* (настройка); в графе *From* указать: *disk*; нажать кнопку *Load Data*. Загрузить файл *dan.dat*, который получен в лабораторной 1 (или задается преподавателем).

В окне *ANFIS Editor: Untitled 2* появится график *Training data*; в колонке *Generate FIS* (формирование НР) указать *Grid partition* (формирование сети); нажать кнопку *Generate FIS*; в поле настройки в окне *input* указать число ФП (*Number of Mfs*) для первого и второго входов (3 3 – для вариантов 1, 4; 3 4 – для варианта 2; 2 2 – для варианта 3; 2 3 для варианта 5); в графе тип ФП (*MF Type*) указать *gbellmf* (1, 5 варианты); *gaussmf* (варианты 2, 3), *trimf* (вариант 4); в поле настройки *output* указать тип НР Сугено (*MF Type*) – linear (1-го порядка); в колонке *ANFIS Info.* (данные о сети) появится информация: *#of input: 2* (число входов); *# of outputs: 1* (число выходов); *# of input mfs:* 3 3 (число ФП входов); нажать кнопку *Structure*; полученную структуру необходимо привести в отчете; в колонке *Train Fis* (настройка НР) в графе *optim. Method* (метод оптимизации) в выпадающем меню выбрать *hybrid*; в графе *Error Tolerance* (допуск на ошибку) ввести 0; в графе *Epochs*: (число итераций) ввести 25; нажать кнопку *Train Now* (Начать настройку); в колонке *Test Fis* (проверка настройки) в графе *Plot against* (исходный график) указать *Training data*; нажать кнопку *Test Now* (проверка); сохранить полученный файл на диске и в рабочей области.

3. Эффективность регулятора на основе алгоритма Такаги–Сугено оценить по результатам моделирования в *Toolbox Simulink* системы управления, структурная схема которой изображена на рис. 1.3. Сравнить качество переходных процессов в системе управления с линейным и нечетким регуляторами: a) в номинальном режиме; b) при наличии ограниченных изменений параметров (k3 =200; k4=200; k5=500; c) при введении звена с насыщением ± 0.15; d) при наличии возмущающего воздействия.

4. Оценить влияние на качество переходных процессов системы управления изменения параметров НР: а) типа ФП; б) числа ФП; в) типа заключения нечетких правил (нулевого порядка); г) число итераций при обучении.

2.3. Содержание отчета

Отчет должен включать описание НР, результаты моделирования, приведенные в таблице (показатели качества управления), структуру адаптивной сети; выводы по работе.

2.4. Контрольные вопросы

1. Как с помощью адаптивной нейронечеткой сети настраиваются параметры НР Такаги–Сугено?
2. В чем отличие нечетких правил Такаги–Сугено?
3. В чем преимущества и недостатки НР Такаги–Сугено в сравнении с регулятором на основе алгоритма Заде–Мамдани?

Лабораторная работа № 3

РЕШЕНИЕ ЗАДАЧ АППРОКСИМАЦИИ НА ОСНОВЕ НЕЙРОНЕЧЕТКОГО ПОДХОДА

Цель работы: исследование возможности работы в пакете *Fuzzy Logic Toolbox* в режиме командной строки; оценка эффективности нейронечеткого подхода для формирования заданных функций.

3.1. Основные сведения

Теоретические основы доказательства аппроксимирующий свойств нечетких моделей представлены в работах L. Wang (1992 г.), B. Kosko (1992 г.).

Множество нечетких моделей являются *универсальными аппроксиматорами*, т.е. для любого заданного оператора $G(u)$ $(G: R^n \to R)$ и для любого заданного $\varepsilon > 0$ существует нечеткая модель $F_f(u)$ $(F_f: R^n \to R)$ такая, что $\max \{ | G(u) - F_f(u) |, u \in R^n \} \le \varepsilon$.

Описание используемых функций Fuzzy Logic Toolbox [4]

1. Функцией *genfis1* генерируется структура системы нечеткого вывода типа Сугено, являющейся исходной для последующего формирования и обучения гибридной системы с помощью *anfis*.

Синтаксис: имя=genfis1 (trnData, numMFs, inmftype, outmftype)

Аргументы функции: *trnData* – матрица данных для обучения сети (обучающая выборка); последний столбец соответствует единственной выходной пе-

ременной, остальные столбцы – входным переменным, число строк равно числу наборов экспериментальных данных (образцов); *numMFs* – вектор, элементы которого определяют число ФП, задаваемых для каждого входа; если для всех входов нужно указать одно и то же число таких функций, данный аргумент задается как скаляр; *inmftype* – строковой массив, элементы которого – типы ФП, задаваемые для входных переменных; *outmftype* – строковыя переменная, определяющая тип выходной переменной («*linear*» или «*constant*»).

2. Функция *anfis* предназначена для создания и/или обучения гибридных сетей с архитектурой *anfis*.

Синтаксис: имя = anfis (trnData, имя, trnOpt, dispOpt)

Значения аргументов функции: *trnData* – матрица данных для обучения сети (обучающая выборка); последний столбец соответствует единственной выходной переменной, остальные столбцы – входным переменным; *имя* – идентификатор создаваемой гибридной сети; если структура системы с таким идентификатором уже создана, то она будет использована для настройки числовых параметров, в противном случае структура будет создана при выполнении функции с опциями, по умолчанию соответствующими выполнению функции genfis1; *trnOpt* – вектор опций обучения, элементы которого имеют следующий смысл: *trnOpt*(1) – количество циклов обучения (по умолчанию 10); *trnOpt(*2) – целевой уровень ошибки обучения (по умолчанию 0); *trnOpt*(3) – начальный шаг алгоритма обучения (по умолчанию 0.01); *trnOpt*(*4*) – коэффициент уменьшения шага (по умолчанию 0.9); *trnOpt*(5) – коэффициент увеличения шага (по умолчанию 1.1); *dispOpt* – вектор опций вида выводимой инфорации (по умолчанию все элементы единичные, что означает вывод всех видов возможной информации в процессе выполнения функции) со следующими элементами: *dispOpt*(1) – anfis-информация; *dispOpt*(2) – ошибка; *dispOpt*(3) – шаг обновления (корректировки) по каждому параметру; *dispOpt*(*4*) – конечные результаты.

3. Функция *evalfis* выполняет нечеткий вывод.

Синтаксис: out = evalfis(input, out_fismat)

Аргументы: *input* – число или матрица, которая определяет входные переменные; *out_fismat* – имя файла, для которого выполняется нечеткий вывод.

Функция *evalfis* возвращает *out* – матрицу выходов.

4. Функция *genfis2* генерирует структуру системы нечеткого вывода (типа Сугено) с использованием алгоритма кластеризации данных:
Синтаксис: имя = genfis2(Xin, Xout, radii)
Аргументами функции являются: *Xin* – матрица входных данных обучающей выборки, столбцы которой ассоциированы с входными переменными, а каждая строка с отдельным опытом; *Xout* – матрица выходных переменных обучающей выборки, столбцы которой представляют значения данных переменных, а число строк равно числу строк матрицы *Xin*; *radii* (радиусы) – вектор, определяющий «области влияния» центров кластеров по каждой входной переменной.
5. Функция *ruleview* вызывает графический интерфейс программы просмотра правил.
Синтаксис: *ruleview*(имя файла)
6. Функция *ruleedit* вызывает графический интерфейс редактора правил системы нечеткого вывода.
Синтаксис: *ruleedit*(имя файла).

3.2. Порядок выполнения работы

1. Сформировать заданные функции с точностью до 1%, используя функции *anfis* и *genfis2*.
2. Оценить эффективность изменения: а) числа данных обучающей выборки; б) числа функций принадлежности; в) типа функций принадлежности (*gbellmf, dsigm, gauss2mf, gaussmf, pimf, smf, trapmf, trimf, zmf*); г) типа алгоритма Сугено (*constant, linear*); д) числа циклов обучения; е) величины "области влияния" (*genfis2*).
3. Оценить допустимость наиболее простой реализации: алгоритм Такаги–Сугено нулевого порядка; треугольная ФП, наименьшее число ФП.
4. Сравнить результаты графика, функций *evalfis* и *ruleview*.
Листинг программы с функцией *anfis*

```
x = (0:0.1:10)';
y = sin(2*x)./exp(x/5);
trnData = [x y];
numMFs = 5 ;
inmfType = 'gbellmf';
outmfType = 'linear';
epoch_n = 20;
```

```
in_fismat = genfis1(trnData,numMFs,inmfType, outmfType);
out_fismat = anfis(trnData,in_fismat,epoch_n);
plot(x,y,x,evalfis(x,out_fismat));
legend('Training Data','ANFIS Output');
out = evalfis(2,out_fismat)
ruleedit(out_fismat)
ruleview(out_fismat)
```

Листинг программы с функцией *genfis2*

```
x = (0:0.1:10)';
y = atan(0.5*x);
trnData = [x y];
Xin = x;
Xout = y;
fismat = genfis2(Xin,Xout,0.1);
z = evalfis(x, fismat)
plot(x,y,x,z);
legend('Training Data','ANFIS Output');
out = evalfis(1,fismat)
ruleedit(fismat)
ruleview(fismat)
```

3.3. Содержание отчета

Отчет должен содержать краткое описание используемых функций, результаты исследования, выводы по работе.

3.4. Контрольные вопросы

1. В чем важность наличия аппроксимирующих свойств у нечетких моделей?
2. Как связана величина «области влияния» с числом нечетких правил?
3. Какой алгоритм обучения используется в функциях *anfis* и *genfis2*?

Лабораторная работа №4

РЕАЛИЗАЦИЯ ЛОГИЧЕСКИХ ФУНКЦИЙ «И», «ИЛИ», «НЕ» С ПОМОЩЬЮ ИСКУССТВЕННОГО НЕЙРОНА

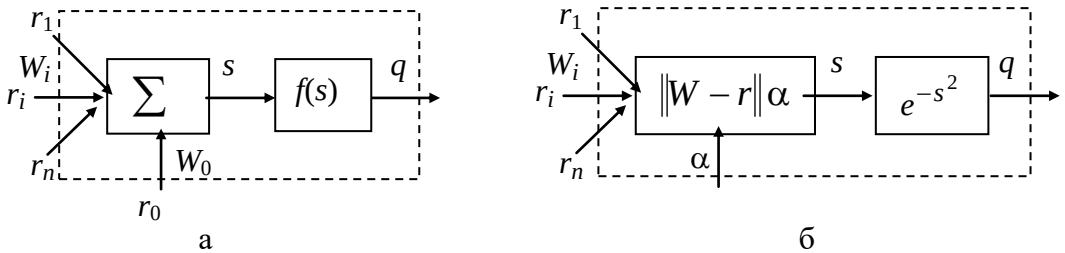
Цель работы: исследование искусственных нейронов (ИН) типа перцептрона, обучение ИН выполнению логических функций «не», «и», «или»; решение задачи классификации с помощью ИН в пакете *Neural Networks Toolbox*, моделирование функций в *Toolbox Simulink.*

4.1. Основные сведения

Искусственный нейрон – элементарный преобразовательный элемент, содержащий n вектор входов $\boldsymbol{r}$, суммирующий блок, блок преобразования сигнала с помощью функции активации, скалярный выход q (рис. 4.1, а). В суммирующем блоке вычисляется взвешенная сумма n входных сигналов r_i $s = \sum_i W_i r_i + W_0 r_0$, где W_i – весовой коэффициент r_i входа. Вход r_0 и коэффициент W_0 вводят специально для смещения нейронов сети, обычно $r_0 = 1$. В модели ИН типа *перцептрон* (модель МакКаллока–Питса) в качестве функции активации $f(s)$ используется пороговая функция, в нейроне *сигмоидального* типа – униполярная (логистическая) или биполярная (гиперболический тангенс) сигмоидальные функции, в нейроне типа *адалина* – линейная функция [1].

Радиальный базисный нейрон (рис. 4.1, б) включает n вектор входов $\boldsymbol{r}$, блок, в котором вычисляется расстояние между вектором входа $\boldsymbol{r}$ и вектором весовых коэффициентов $\boldsymbol{W}$, блок преобразования с помощью функции активации, в качестве которой используется *радиальная базисная функция*. Полученное в первом блоке расстояние умножается на фиксированный порог α, который позволяет управлять чувствительностью ИН. Радиальная базисная функция (*RBF*) имеет максимум равный единице, когда вход равен нулю, т.е. единица на выходе, когда входной вектор равен вектору весовых коэффициентов.

С помощью ИН можно моделировать ряд функций, например [5], как показано на рис. 4.2, а, б, в, логические функции «и», «или», «не». Функция «и» имеет значение 1, если оба входа равны, в противном случае – 0. Поэтому, если заданы оба входа (вектор $\boldsymbol{r}$ =(1, 1)), то, используя пороговую функцию



а б

Рис. 4.1

активации $f(s) = \begin{cases} 1, s > 0; \\ 0, s \le 0 \end{cases}$, получим следующие результаты

$s = \sum_i W_i r_i + W_0 r_0 = (1 \times 1) + (1 \times 1) - 1 = 1$, $q = f(s) = 1$. Если вектор: $\boldsymbol{r}$ =(1, 0); (0, 1); (0, 0), то $q = f(s) = 0$.

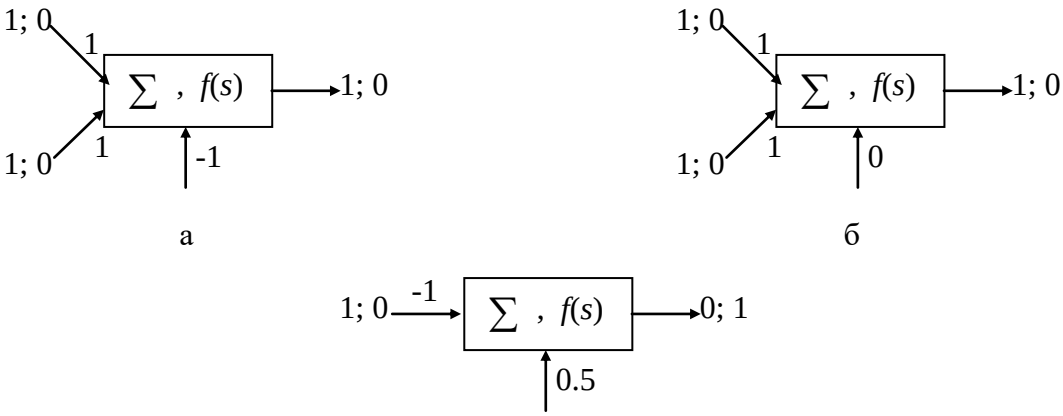


Рис. 4.2

Описание функции создания перцептрона, реализующего логические функции, *newp*: имя = newp (PR, S, TF, LF). Аргументы функции: PR – R×2 – матрица минимальных и максимальных значений для R входных элементов; S – число нейронов; TF – функция активации, по умолчанию «*hardlim*» (пороговая); LF – функция обучения, по умолчанию «*learnp*» (алгоритм обучения перцептрона). Задание входных векторов функций «и», «или»: P = [0 1 0 1; 0 0 1 1]; задание выходов нейрона: для функции «и» – T=[0 0 0 1]; для функции «или» – T=[0 1 1 1]. Вход функции «не» – P = [0 1]; выход – T=[1 0].

Описание функции формирования нейронной сети (НС) в Toolbox Simulink: *gensim(net)*, где *net* – имя сети.

4.2. Программа работы

1. Реализация логических функций «и», «или», «не».

Листинг программы

```
P = [0 1 0 1; 0 0 1 1]; % задание входных векторов;
T=[0 0 0 1]; % задание выходов нейрона;
```

```
plotpv(P,T); % графическое представление исходных векторов;
net1=newp([0 1; 0 1], 1); % создание перцептрона с 1 нейроном;
E=1;% присвоение начального значения ошибки;
net1=init(net1); % инициализация перцептрона;
while(sse(E)) % организация цикла обучения перцептрона, классификация;
   [net1,Y,E]=adapt(net1,P,T); % обучение нейрона на выборке [P,T];
% Получение управляющей структуры linehandle для изображения разделяющей
% линии в координатах весов (IW) и порога срабатывания нейрона (b)
   linehandle = plotpc(net1.IW{1},net1.b{1});
   drawnow;
end;
```

2. Формирование нейросетевой модели в *Toolbox Simulink*.

Задание функции *gensim*(*net1*) приводит к открытию блок-диаграммы, включающей блок Input1, являющийся стандартным блоком задания константы, блок НС (*Neural Network*) и регистрирующий блок (*Scope* – осциллограф). Активизируя блок НС, а затем блок сеть1 (layer1), можно получить структуру созданного нейрона. Вместо функции активации *hardlim*, используемой по умолчанию, включить в структуру блок *sign* из раздела Library Simulink Math Operations и блок *saturation* из раздела Discontinuities.

Весовые коэффициенты (*weight*) для моделирования функций «и», «или» определить согласно рис. 4.2 а, б: W = [1 1], для «не» в соответствии с рис. 4.2 в – W = [–1] коэффициент смещения (*buas*) W_0 = -1 для «и», W_0 = 0 – для «или», W_0 = 0.5 – для «не». Выполнить моделирование функций, задавая входные вектора в блоке Input1.

3. Оценить возможность реализации функции «исключающее или» с помощью перцептрона. Вход функции: P = [0 1 0 1; 0 0 1 1]; выход: T=[0 1 1 0].

4.3. Содержание отчета

Отчет должен содержать краткое описание используемых функций, результаты исследования, схему перцептрона, выводы по работе.

4.4. Контрольные вопросы

1. Как формируются логические функции с помощью перцептрона?

2. Возможна ли реализация функции «исключающее или» с помощью однослойного перцептрона? Почему?

Лабораторная работа № 5

РЕШЕНИЕ ЗАДАЧ АППРОКСИМАЦИИ СРЕДСТВАМИ НЕЙРОСЕТЕВЫХ ТЕХНОЛОГИЙ

Цель работы: ознакомление с пакетом *Neural Networks Toolbox* системы (MATLAB); рассмотрение способов формирования и обучения сетей прямого распространения и RBF-сетей; исследование эффективности нейросетевого подхода для формирования заданных функций.

5.1. Основные сведения

Группы искусственных нейронов в сети образуют *слои*. Однослойные сети позволяют моделировать ряд функций, например, логические функции «и», «или», «не». Нереализуемые однослойной сетью функции (например, «исключающее или») называются *линейно неразделимыми*. Для моделирования подобных функций используют *многослойные сети*.

На рис. 5.1 многослойная сеть *прямого распространения* состоит из входного, промежуточного (скрытого) и выходного слоев. Обозначено: $\boldsymbol{W}^1$, $\boldsymbol{W}^2$ – матрицы весовых коэффициентов скрытого и выходного слоев соответственно, q_i^j – выход *i*-го ИН *j*-го слоя. ИН входного слоя служат для распределения сигналов между ИН скрытого слоя и не осуществляют преобразование входных сигналов. ИН каждого слоя не связаны между собой, выходы ИН *l*-го слоя поступают только на входы ИН *l*+1-го слоя. Функция активации принимается одинаковой для всех ИН скрытых слоев сети. Выходной слой, как правило, состоит из ИН типа адалина и называется в этом случае *мадалина*. *Радиальная базисная сеть* состоит из двух слоев: скрытый слой из радиальных базисных нейронов (описание дано в лабораторной работе 4), выходной слой – мадалина.

Теоретические основы доказательства аппроксимирующий свойств нейронных сетей заложены в работах А. Н. Колмогорова В. И. Арнольда, Fu-

nahashi, R. Hecht-Nielsen [1]. *Теорема А. Н. Колмогорова* (*1958 г.*): любая непрерывная функция *n* переменных на замкнутом ограниченном множестве представима с помощью операций сложения, умножения и суперпозиции непрерывных функций одной переменной. *Теорема* (*Funahashi 1989 г.)*: бесконечно большая сеть с единственным скрытым слоем способна аппроксимировать любую непрерывную функцию. *Теорема* (*R. Hecht-Nielsen 1990 г.)*: реализация функции *n* переменных возможна с использованием сети прямого распространения с *n* искусственными нейронами в первом скрытом слое, и *2n* +1 – во втором скрытом слое.

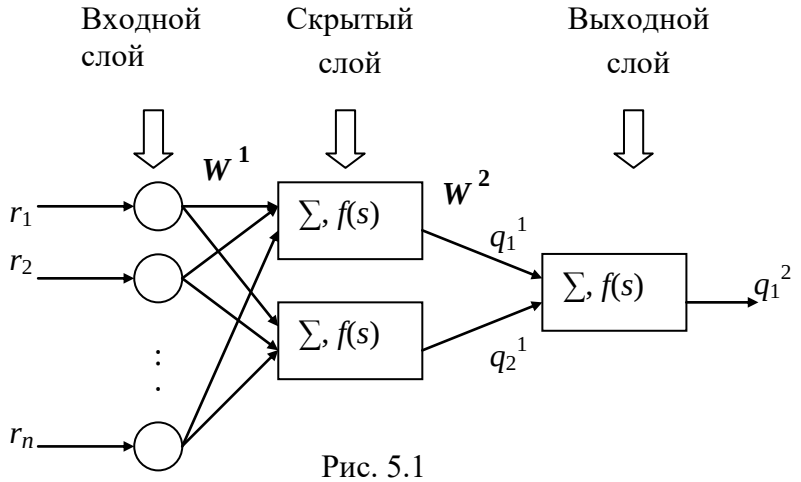


Рис. 5.1

Описание функции создания многослойной нейронной сети прямого распространения *newff* :

имя = newff (PR, [S_1 S_2… S_N], {TF_1 TF_2… TF_N}, BTF, BLF, PF).

Аргументы функции: PR – R×2 – матрица минимальных и максимальных значений для R входных элементов; S_i – число нейронов i-го слоя; TF_i – функция активации i-го слоя, по умолчанию «*tansig*» (гиперболический тангенс); BTF – функция обучения, по умолчанию «*traingd*» (алгоритм обучения обратного распространения ошибки), BLF – функция настройки весов и смещений, по умолчанию «*learngdm*» (градиентный алгоритм оптимизации с инерционной составляющей), PF – функция ошибки, по умолчанию «*mse*» (среднеквадратичная ошибка).

Описание функции создания *RBF*-сети:

имя = newrb (P, T, goal, spread).
Аргументы функции: P – матрица входных векторов; T – матрица целевых векторов; goal – заданная среднеквадратичная ошибка; spread – разброс функции, по умолчанию 1.0.

5.2. Порядок выполнения работы

1. Формирование заданной функции с точностью до 0.1% с помощью сети прямого распространения, изменяя: а) число данных обучающей выборки;
б) число слоев сети; в) число ИН в скрытом слое; г) тип функция активации скрытого слоя: (*tansig* (гиперболический тангенс); *hardlim* (пороговая); logsig (сигмоидальная); *hardlims* (знаковая); *satlin* (полулинейная с насыщением); *tribas* (треугольная)); *purelin* (линейная); д) число циклов обучения (net.trainParam.epochs); е) алгоритм обучения: *traingd* (алгоритм обучения обратного распространения ошибки); *trainbr* (байесовская регуляризация, алгоритм Левенберга-Марквардта); *trainlm* (алгоритм оптимизации Левенберга-Марквардта).
Листинг программы

```
% Аппроксимация функции
% humps(x) = 1./((x– 0.3)*0.2+0.01)+1./((x-0.9)*0.2+0.04) – 6
x = –1:.05:2;  % задание входного вектора;
y = humps(x);% задание выходного вектора;
P = x; T = y;
% Создание и тестирование нейронной сети
net = newff([-1 2],[20, 1],{'tansig','purelin'},'trainlm');
% newff– сеть прямого распространения(feedforward);
% [–1 2] – матрица минимальных и максимальных значений входных элементов;
% [5, 1] – матрица, указывающая на число ИН в слоях: 5 – число ИН в первом скрытом слое, 1– число ИН в выходном слое;
%{'tansig','purelin'}
% 'tansig'– функция активации скрытого слоя(гиперболический тангенс).
% Варианты: tansig;  hardlim; logsig; hardlims; satlin; tribas;
%'purelin'– линейная функция активации выходного слоя;
%'traingd'– алгоритм обучения обратного распространения ошибки;
% Варианты: 'trainbr'; 'trainlm';
net.trainParam.show = 400;    % результаты выводить через 400 итераций;
net.trainParam.lr = 0.05;     % скорость обучения;
net.trainParam.epochs = 1000; % количество циклов обучения;
```

```
net.trainParam.goal = 1e-3;   % заданная ошибка обучения;
% Обучение сети
net1 = train(net, P, T);
% Тестирование сети
a=sim(net1,P);
% Создание графиков исходного и аппроксимированного сигнала
plot(P,a,'k-'); grid; hold;
xlabel('time(s)'); ylabel('output'); title('humps function')
plot(P,T,'k*')
gensim(net1)
```

2. Реализация логических функции «и», «или», «исключающее или» с помощью RBF-сети.

Листинг программы

```
% Реализация логических функций функции
P1 = [0  0  1  1;  0  1  0  1];        T1 = [0  1  1  0];
% Создание и тестирование нейронной сети с радиальными базисными элементами;
goal = 0; %  среднеквадратичная ошибка;
spread=0.5; %  разброс параметров;
net=newrb(P1,T1,goal,spread);
% Тестирование сети;
a=sim(net,P1)
gensim(net) % формирование нейросетевой модели в Toolbox Simulink;
```

3. Формирование заданной функции с помощью RBF-сети, изменяя: число данных обучающей выборки, величину ошибки, разброс параметров.

Листинг программы

```
% Аппроксимация функции
x=-3:.05:3; %  задание входного вектора;
y=humps(x);% задание выходного вектора;
P=x; T=y;
% Создание и тестирование нейронной сети с радиальными базисными элементами
goal = 0.02;  % среднеквадратичная ошибка;
spread=0.1; %  разброс параметров;
net = newrb(P,T,goal,spread);
%Тестирование сети
a=sim(net,P);
% Создание графиков исходного и аппроксимированного сигнала
plot(P,a,'k-'); grid; hold;
plot(P,T,'k*'); xlabel('Time(s)'); ylabel('Output of network and error');
title ('Humps function approximation – radial basis function')
gensim(net) % создание модели сети в Toolbox Simulink
% SSE  Sum squared error performance function (суммарная квадратичная ошибка).
```

5.3. Содержание отчета

Отчет должен содержать краткое описание используемых функций, результаты исследования, схему нейронной сети прямого распространения для 2-х ИН в скрытом слое, схему RBF-сети, выводы по работе.

5.4. Контрольные вопросы

1. В чем отличие сетей прямого распространения от нейронечетких сетей?
2. Почему для аппроксимации функций с помощью RBF-сети достаточно двух слоев?

Лабораторная работа № 6
ПОСТРОЕНИЕ СИСТЕМЫ УПРАВЛЕНИЯ С НЕЙРОСЕТЕВЫМ РЕГУЛЯТОРОМ

Цель работы – ознакомление с пакетом *Neural Networks Toolbox* (MATLAB), построение нейросетевого регулятора. Эффективность управления системы с нейросетевым регулятором проверяется по результатам моделирования в *Toolbox Simulink.*

6.1. Порядок выполнения работы

В состав пакета нейронных сетей (*Neural Networks Toolbox*) включено инструментальное средство организации диалога с пользователем *NNTool*, которое является графическим интерфейсом пользователя GUI (*Graphic User Interface*) системы Matlab.

1. Вызов GUI-интерфейса *NNTool* осуществляется командой *nntool* из командной строки. После вызова появляется окно *Network/Data Manager* (Управление сетью/данными), которое содержит области:

- *Inputs* - последовательность входов (заданные входы);
- *Targets* - последовательность целей (заданные выходы);
- *Input Delay States* -начальные условия линии задержки входа;
- *Networks* - (список нейронных сетей);

- *Outputs* - последовательность выходов (действительные выходы);
- *Errors* - последовательность ошибок обучения;
- *Layer Delay States* - начальные условия линии задержки слоя.

В поле Network and Data содержатся кнопки:

- *Help* - вызов окна подсказки;
- *New Data* - вызов окна формирования данных;
- *New Network* - вызов окна создания новой нейронной сети;
- *Impor*t - вызов окна для извлечения или загрузки данных;
- *Export* - вызов окна для передачи или загрузки данных в файл.

Кнопки *View, Delete* становятся активными только после создания и активизации данных входа и выхода. Кнопка *View* позволяет посмотреть, а кнопка *Delete* удалить активизированные данные.

Кнопки *View, Delete, Initialize*…, *Simulate*…, *Train*…, *Adapt*… становятся активными после создания и активизации нейронной сети, позволяют просмотреть, удалить, инициализировать, промоделировать, обучить или адаптировать нейронную сеть.

2. Для формирования последовательностей входа и целей нажать кнопку *New Data.* В поле *Name* окна *Create New Data* указать имя переменной *ede*, затем в области значений *Value* ввести вектор *ede'* (вектор-строка), приведенный в файле *dat.m* и, используя кнопку *Inputs* (в правой части окна), указать тип переменной (*Inputs*- входы). Ввод завершается нажатием кнопки *Create* (создать). Аналогичная операция выполняется для задания последовательности целей *vn.* Ввести вектор *vn'* (вектор-строка), приведенный в файле *dat.m*. Указывается тип переменной: *Targets* – цели.

3. Для создания нейронной сети в окне *Network/Data Manager* нажать кнопку *New Network.*

В окне *Create New Network* выбрать нейронную сеть прямого распространения *feed-forward*. При создании сети сохранить имя, приведенное по умолчанию (*Network*1). Диапазон изменения входов задается с помощью опции *Get from input*. Указать число слоев (*Number of layers*) – 3. В поле свойства слоев (*Properties for Layer*) задать количество нейронов (*Number of neurons*) и функцию активации (*Transfer Function*).

Варианты: № 1: 1-й слой - 5; *tansig*; 2-й слой - 5; *tansig*; 3-й слой -1 *purelin*; № 2: 1-й слой - 3; *tansig*; 2-й слой - 3; *tansig*; 3-й слой -1 *purelin*; № 3:

1-й слой - 5; *tansig*; 2-й слой - 3; *tansig*; 3-й слой -1 *purelin*; № 4: 1-й слой - 3; *tansig*; 2-й слой - 6; *tansig*; 3-й слой -1 *purelin*; № 5: 1-й слой - 5; *tansig*; 2-й слой - 2; *tansig*; 3-й слой -1 *purelin*.

Остальные установки оставить заданными по умолчанию. Создание сети завершить нажатием кнопки *Create.*

4. В окне Network/Data Manager, в области *Networks* появится имя новой созданной сети *Network*1. Выделение имени с помощью мышки приведет к активизации всех кнопок окна.

Для инициализации сети открыть с помощью кнопки *Initialize* диалоговую панель Network: *Network*1, указать закладку Initialize. Для ввода установленных диапазонов переменных и инициализации весов отметить кнопки *Set Ranges* (установить диапазоны) и *Initialize Weights* (инициализировать веса).

5. Для обучения сети в окне Network: *Network*1 выбрать закладку Train. Выбранная диалоговая панель имеет три закладки:

- *Training Info* - информация об обучающих последовательностях;
- *Training Parameters* -параметры обучения;
- *Optional Info* - дополнительная информация.

Последняя закладка применяется, когда в процессе обучения используется контрольная и тестовая последовательности.

Применяя закладки, можно установить имена последовательностей входа *ede* и цели *vn,* а также на закладке *Training Parameters* значения параметров процедуры обучения. Число циклов обучения (epochs) установить 500. Остальные параметры процедуры обучения оставить, как задано по умолчанию. Обучение сети выполняется после нажатия кнопки *Train Network.*

6. Результаты обучения можно посмотреть в окне *Network/Data Manager*. Активизируя имена последовательностей выходов network1_outputs или ошибок network1_errors, можно получить результаты, используя кнопку *View.* Для просмотра структурной схемы сети необходимо, выбрав имя сети (network1), отметить кнопку *View.*

7. Для передачи созданной сети в рабочую область системы *Matlab* нужно отметить кнопку *Export* и в открывшемся окне *Export* or *Save from Network/Data Manager* выбрать *Select All* (выбрать все) и *Export.* Для построения модели нейронной сети в пакете Simulink использовать функцию

gensim(network1, -1)

Полученную сеть подставить в файл *if.mdl* вместо нечеткого регулятора (рис. 1.3).

8. Сравнить качество переходных процессов в системе управления с линейным и нейросетевым регуляторами.

6.2. Содержание отчета

В отчете привести структуру нейронной сети и результаты моделирования системы управления с нейросетевым регулятором.

6.3. Контрольные вопросы

1. В чем состоит основное преимущество нейросетевых систем управления?
2.?

Список литераторы

1. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы /Пер. с польск. И. Д. Рудинского. – М.: Горячая линия – Телеком, 2004. – 452 с.
2. Алиев Р. А., Церковный А. Э., Мамедова Г. А. Управление производством при нечеткой исходной информации. – М.: Энергоатомиздат, 1991.–236 с.
3. Поляхов Н. Д., Приходько И. А. Нечеткие системы управления: Учебн. пособие. СПб.: Изд СПбГЭТУ «ЛЭТИ», 2003. – 48с.
4. Дьяконов В., Круглов В. Математические пакеты расширения MATLAB: Спец. справ. – СПб.: Питер, 2001.– 480 с.
5. Джонс М. Т. Программирование искусственного интеллекта в приложениях /Пер. с англ. Осипов А. И. – М.: ДМК Пресс, 2004. – 312 с.

Содержание

Редактор
ЛР № от

Подписано в печать . . Формат 60 × 84 1/16. Бумага офсетная.
Печать офсетная. °Усл. печ. л. 1,62. Уч.-изд. л. 1,75 . Тираж 100 экз. Заказ.

Издательство СПбГЭТУ "ЛЭТИ"
197376, С.-Петербург, ул. Проф. Попова, 5
° Гарнитура "Times New Roman"