ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ЦИФРОВЫХ ОБРАЗОВАТЕЛЬНЫХ ТЕХНОЛОГИЙ

Кафедра «Робототехника и мехатроника»

Учебное пособие

по дисциплинам

«Искусственный интеллект» «Искусственный интеллект в мехатронике и робототехнике» «Компьютерное управление»

Авторы

Тугенгольд А. К.,

Лукьянов Е. А.,

Юсупов А. Р.,

Ивацевич Ю. Б.

Ростов-на-Дону, 2020

Аннотация

Переход на обучение по ФГОС3, ФГОС3+, ФГОС++ предъявляет повышенные требования к формированию профессиональных компетенций, причем в учебных планах значительно увеличено время на самостоятельную работу студентов. В связи с этим актуальной задачей является разработка методического обеспечения для самостоятельной работы студентов, как индивидуальной, так и под руководством преподавателя. Предлагаемый сборник является основой для самостоятельной работы студентов, обучающихся по направлениям: Мехатроника и робототехника, Вычислительная техника и информатика при изучении таких дисциплин как «Искусственный интеллект в мехатронике и робототехнике», «Искусственный интеллект», «Компьютерное управление». Значительный объем практических и лабораторных занятий позволяет преподавателю разнообразить варианты заданий и перейти от группового выполнения заданий к индивидуальному. Перед началом занятий необходимо ознакомиться с теоретической частью методических указаний. Кроме того, по каждой теме преподавателем подробно разбирается пример выполнения задания и выдаются номера вариантов для самостоятельного решения. Отчет необходимо сдавать в распечатанном виде. В отдельных случаях вариант задания может быть выдан на подгруппу студентов.

Авторы

д.т.н., профессор кафедры «Робототехника и мехатроника» Тугенгольд А.К.,
к.т.н., доцент, зав. кафедрой «Робототехника и мехатроника» Лукьянов Е.А.,
ст. преподаватель «Робототехника и мехатроника» Юсупов А.Р.,
к.т.н., доцент кафедры «Робототехника и мехатроника» Ивацевич Ю.Б.

Оглавление

ЧАСТЬ 1. КЛАССИЧЕСКИЕ МОДЕЛИ ЗНАНИЙ ПРАКТИЧЕСКИЕ ЗАНЯТИЯ.

Продукционная модель представления знаний

Продукционные модели знаний занимают особое положение, т. к. они являются наиболее декларативным способом представления знаний. Продукционная модель представления знаний – это набор правил вила: ЕСЛИ , ТО , где - образец для поиска в базе данных (утверждение о состоянии базы данных), - действие, выполняемое при успешном исходе поиска в базе (процедуры, которые могут изменять состояние базы данных). Действия могут быть промежуточными, выступающие далее как условия и целевыми, завершающими работу системы. В продукционных моделях используются некоторые элементы логических моделей, что позволяет организовывать на них эффективные процедуры вывода, а с другой стороны, более наглядно отражают знания, чем классические логические модели. Правила вывода в этих моделях называются продукциями. Системы продукций – это набор правил, используемый как базы знаний, поэтому его называют еще базой правил. Продукции соответствуют навыкам в долгосрочной памяти человека. Подобно навыкам в долгосрочной памяти эти продукции не изменяются при работе системы. Они вызываются «по образцу» для решения данной проблемы. Рабочая память продукционной системы соответствует краткосрочной памяти, или текущей области внимания человека. Содержание рабочей области после решения задачи не сохраняется. Достоинствами продукционной модели являются: наглядность, высокая модульность, легкость внесения дополнений и изменений, простота логического вывода, простота интерпретации. Основной недостаток: при накоплении большого числа продукций они начинают противоречить друг другу, возникают трудности при добавлении правил, зависящих от уже имеющихся в базе знаний, отсутствует целостный образ знаний, неясна взаимосвязь между правилами, возможны конфликты между правилами.

Пример составления продукций:

Имеется шахматная доска 3х3. Найти последовательность ходов конем, чтобы он ставился на каждую клетку только один раз. Начало движения с поля 1. Возможные состояния:

Возможные состояния:

P1: Если(конь в поле 1, То (ход конем в поле 8;

P2: Если(конь в поле 1, То (ход конем в поле 6;
P3: Если(конь в поле 2, То (ход конем в поле 9;
P4: Если(конь в поле 2, То (ход конем в поле 7;
P5: Если(конь в поле 3, То (ход конем в поле 4;
P6: Если(конь в поле 3, То (ход конем в поле 8;
P7: Если(конь в поле 4, То (ход конем в поле 9;
P8: Если(конь в поле 4, То (ход конем в поле 3;
P9: Если(конь в поле 6, То (ход конем в поле 1;
P10: Если(конь в поле 6, То (ход конем в поле 7;
P11: Если(конь в поле 7, То (ход конем в поле 2;
P12: Если(конь в поле 7, То (ход конем в поле 6;
P13: Если(конь в поле 8, То (ход конем в поле 3;
P14: Если(конь в поле 8, То (ход конем в поле 1;
P15: Если(конь в поле 9, То (ход конем в поле 2;
P16: Если(конь в поле 9), То (ход конем в поле

Допустим, что необходимо переместить коня в поле 2 из поля 1. Вышеуказанное условие можно представить в виде операторов:

Move (1,8) Move (6,1)
Move (1,6) Move (6,7)
Move (2,9) Move (7,2)
Move (2,7) Move (7,6)
Move (3,4) Move (8,3)
Move (3,8) Move (8,1)
Move (4,9) Move (9,2)
Move (4,3) Move (9,4)

Это база знаний (база факторов).

Итерации для задачи:

№ итерации	Текущее поле	Целевое поле	Конфликтное множество	Активация правила
1	1	2	1,2	1
2	8	2	13,14	13
3	3	2	5,6	5
4	4	2	7,8	7
5	9	2	15,16	15
6	2	2		Выход

В итерациях 3 и 5 возникает конфликтное множество. Наиболее простой способ разрешения – выбор первого соответствующего перемещения, которое ведет в еще не посещаемое состояние.. Конфликтное множество – это простейшая база це-

лей. Таким образом, если конь имеет систему движений, то последовательность движений 1-8-3-4-9-2.

Конфликтное множество – это список всех правил, имеющих удовлетворенные условия при некотором, текущем состоянии фактов и объектов и которые еще не выполнены.

Задачи:

1. Построить продукционную модель представления знаний в предметной области «Аэропорт» (диспетчерская).
2. Построить продукционную модель представления знаний в предметной области «Железная дорога» (продажа билетов).
3. Построить продукционную модель представления знаний в предметной области «Торговый центр» (организация)
4. Построить продукционную модель представления знаний в предметной области «Автозаправка» (обслуживание клиентов).
5. Построить продукционную модель представления знаний в предметной области «Автопарк» (пассажирские перевозки).
6. Построить продукционную модель представления знаний в предметной области «Компьютерные сети» (организация).
7. Построить продукционную модель представления знаний в предметной области «Университет» (учебный процесс).
8. Построить продукционную модель представления знаний в предметной области «Компьютерная безопасность» (средства и обеспечения).
9. Построить продукционную модель представления знаний в предметной области «Компьютерная безопасность» (угрозы).
10. Построить продукционную модель представления знаний в предметной области «Интернет-кафе» (организация и обслуживание). Управление дистанционного обучения и повышения квалификации Искусственный интеллект 6
11. Построить продукционную модель представления знаний в предметной области «Разработка информационных систем» (ведение информационного проекта). 12.Построить продукционную модель представления знаний в предметной области «Туристическое агентство» (работа с клиентами)

Семантическая модель представления знаний

В бытовом понимании семантика означает смысл слова, действия, художественного произведения и т.п. Семантическая сеть – это ориентированный граф, вершинам которого сопоставляются понятия (объекты, процессы, явления), дуги графа – это отношения между вершинами. Любой фрагмент сети, например,

одна вершина, две вершины и соединяющие их дуги, называют подсетью. Логический вывод (поиск решения) на семантической сети заключается в том, чтобы найти или сконструировать подсеть, удовлетворяющую некоторым условиям. Метки вершин имеют ссылочный характер и представляют собой некоторые имена. В роли имен могут выступать, например, слова естественного языка. Метки дуг обозначают элементы множества отношений. Достоинства семантических сетей: наглядность, соответствует представлениям об организации долговременной памяти человека, позволяет снизить объем хранимых данных. Недостатки: сети представляют собой пассивные структуры, для обработки которых необходим специальный аппарат формального вывода и планирования, произвольная структура и различные типы вершин и связей усложняют процедуру обработки информации, модель не дает ясного представления о структуре предметной области. По типу знания выделяют экстенсиональные сети, которые описывают конкретные отношения данной ситуации и интенсиональные, описывающие имена классов объектов, а не их индивидуальные имена, связи отражают те отношения, которые всегда присущи объектам данного класса. По типу ограничений на дуги и вершины сети подразделяются на простые, в которых вершины не обладают внутренней структурой и иерархические, когда вершины обладают внутренней структурой. В таких сетях возможно сеть разделять на подсети и устанавливать отношения между подсетями. Можно выделить также динамические сети (сценарии), т.е. сети с событиями. Однородные сети обладают одним типом отношений, неоднородные – предполагают количество отношений больше двух. Управление дистанционного обучения и повышения квалификации Искусственный интеллект 7 Кроме того, можно выделить бинарные и N- арные сети,. в последних существуют отношения ,связывающие больше двух объектов. Возможные типы отношений: являться наследником (a-kind-of) – задает иерархические связи между классами; являться экземпляром (is-a, например) – описывает конкретные объект, понятие; это (are,есть) – используется в отношениях, подразумевающих равенство или эквивалентность; являться частью (has-part) – описывает части или целые объекты; функциональные – отражают различные отношения; количественные – отображают количественные отношения между вершинами; временные – описывают временные связи между вершинами; логические – описывают логические связи между вершинами; атрибутивные – описывают свойства объектов, понятий.

Пример составления семантической сети.

Предложение: Если станок закончил обработку, робот грузит кассету с деталями на робокар, который перевозит их на склад, где штабелер помещает кассету в ячейку.

Выделяем факты (факт – конкретизация отношений между объектами).

F1 – станок закончил обработку;

F2 – робот грузит;

F3 – робокар перевозит;

F4 – кассета содержит;

F5 – штабелер размещает.

Обозначаем факты кружками, а связанные с ними события – прямоугольниками. Дуги помечаем наименованиями отношений, которые они выражают рис.1).

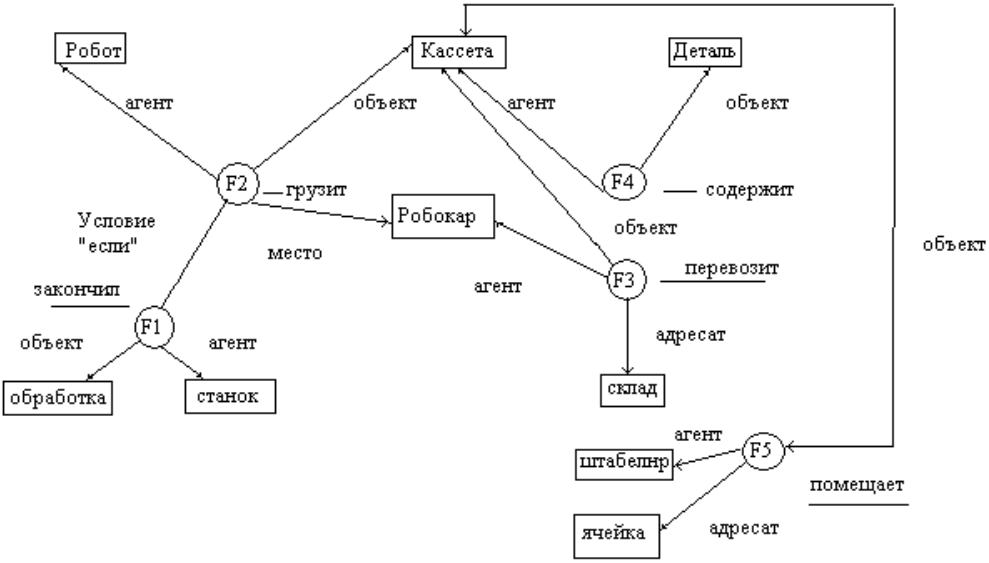


Рис.1. Пример семантической сети.

Запрос можно представить графом, в котором вершины, соответствующие некоторым переменным, не определены.

Запрос: Оператор сообщил, что робокар что-то перевозит. Определить, что и как перевозит робокар (рис.2).

Факт сообщил

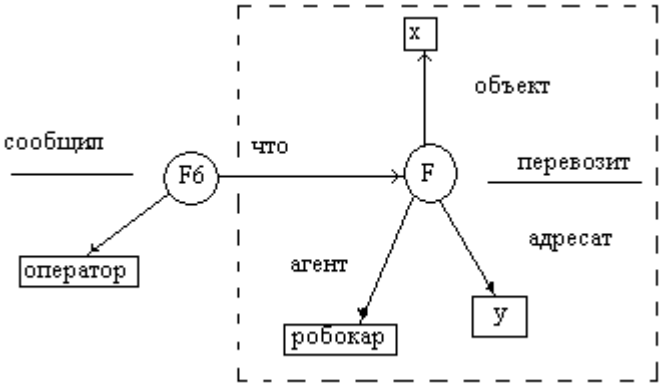


Рис.2. Пример подграфа.

Вложив этот подграф в семантическую сеть (совместив с F3) получим конкретные ответы: x – кассета, y – склад. Ответ будет выглядеть так: робокар перевозит кассету на склад.

Задания

1.Построить семантическую модель (сеть) представления знаний в предметной области «Аэропорт» (диспетчерская)

2. Построить семантическую модель (сеть) представления знаний в предметной области «Железная дорога» (продажа билетов).

3. Построить семантическую модель (сеть) представления знаний в предметной области «Торговый центр» (организация).

4. Построить семантическую модель (сеть) представления знаний в предметной области «Автозаправка» (обслуживание клиентов).

5. Построить семантическую модель (сеть) представления знаний в предметной области «Автопарк» (пассажирские перевозки).

6. Построить семантическую модель (сеть) представления знаний в предметной области «Компьютерные сети» (организация).

7. Построить семантическую модель (сеть) представления знаний в предметной области «Университет» (учебный процесс).

8. Построить семантическую модель (сеть) представления знаний в предметной области «Компьютерная безопасность» (средства и способы ее обеспечения).

9. Построить семантическую модель (сеть) представления знаний в предметной области «Компьютерная безопасность»

(угрозы).

10.Построить семантическую модель (сеть) представления знаний в предметной области «Интернет-кафе» (организация и обслуживание).

11.Построить семантическую модель (сеть) представления знаний в предметной области «Разработка информационных систем» (ведение информационного проекта). 12.Построить семантическую модель (сеть) представления знаний в предметной области «Туристическое агентство» (работа с клиентами).

Фреймовая модель представления знаний

Все типы моделей знаний перед их применением в конкретной системе необходимо заполнить информацией, уточняющей используемые общие символы и понятия. Модель без наполнения информацией до уровня соответствия единичной реальной системе называется абстрактной. В обычном диалоге значительная часть информации не выражается собеседником определенно и ясно (умалчивается). Предполагается, что оба собеседника хорошо знают тему разговора и нет смысла лишний раз описывать очевидные детали, которые являются стандартными для данной ситуации. Термин фрейм (frame – рамка, остов, каркас) предложен в 1975 г. Марвеллом Минским. Фрейм – это единица представления знаний, заполненная в прошлом, детали которой могут быть изменены согласно текущей ситуации, т.е. это минимальное описание, которое еще сохраняет сущность описываемого явления и такое, что дальнейшее ее сокращение приводит к потере сущности. Получается, что фрейм – это абстрактный образ, объект или ситуация. Достоинства: гибкость, наглядность, удобный способ включения процедурных знаний, модульность, сводимость к другим моделям. Недостатки: отсутствие универсальной процедуры управления выводом кроме механизма наследования. Фрейм отражает основные свойства объекта или явления. Информация в фреймах записывается в виде списка свойств, называемых во фрейме слотами (slot – паз, щель), таким образом, слот является основной структурной единицей фрейма. Слоты – это некоторые незаполненные подструктуры фрейма, заполнение которых приводит к тому, что данный фрейм ставится в соответствие некоторой ситуации, явлению или объекту. Слот представляет собой пару: имя слота и его значение. В качестве значения слота могут выступать константы (факты), выражения с переменными, ссылки на другие слоты и т.п. Слот может иметь структуру, элементы которой сами являются слотами. Фрейм состоит из конечного зна-

чения слотов. Слотам фреймов могут быть приписаны по умолчанию некоторые стандартные значения. Значения, присвоенные по умолчанию, могут быть заменены значениями, подходящими для обрабатываемой ситуации. Различают фреймы-образцы или прототипы, хранящиеся в базе знаний, и фреймы-экземпляры, которые создаются для отображения реальных фактических ситуаций на основе поступающих данных. Фрейм-прототип – это интенсиональное описание некоторого множества фреймов-примеров. Пример фрейма-прототипа: ДАТА ()()()(){}()() ()(){} () В слоте () на месте значения записано ИМЯ, т.е. значением слота может быть любое буквенное выражение. Значением слота ДЕНЬ являются целые числа, причем перечень их приводится в слоте. В качестве функции могут быть использованы любые функ- Уп 11 ции языка LISP. Так, в слоте ГОД с использованием языка LISP могут быть организованы следующие процедуры. Если во входном предложении указан ГОД, то он вносится в поле значения фрейма-примера; если год не указан, то активизируется процедура, которая заполняет значение текущим годом. Такого рода функция называется «по умолчанию». В слоте «день недели» можно организовать процедуры, которые при обработке входного сообщения будут вызываться автоматически, для проверки на непротиворечивость значения дня недели, указанного пользователем, либо вычисления этого значения. Конкретный пример может выглядеть следующим образом: ДАТА>()()()() Метка ISA обозначает, что данный слот является фреймом примером. Формально фрейм – это тип данных вида: – имя объекта; – множество слотов, содержащих факты, определяющие декларативную семантику фрейма; – множество слотов, обеспечивающих связи с другими фреймами (каузальные, семантические и т. д.); – множество слотов, обеспечивающих преобразования, определяющие процедурную семантику фрейма. Модель фрейма является достаточно универсальной, т.к. существуют не только фреймы для обозначения объектов и понятий, но и другие типы: - фреймы-сценарии, используемые для обозначения объектов и понятий (лекция, собрание, заем); - фреймы-роли (отец, мать, менеджер, кассир, клиент); - фреймы-сценарии (собрание акционеров, празднование дня рождения); - фреймы-ситуации (тревога, авария, рабочий режим работы устройства) и другие.

Формально как модуль для отображения образа структура фрейма может быть представлена следующим образом:

(имя фрейма)

(имя 1-го слота); (значение 1-го слота)

(имя 2-го слота); (значение 2-го слота)
.....
.....
(имя N-го слота); (значение N-го слота)

Ту же запись можно представить в виде таблицы

Имя фрейма			
Имя слота	Значение слота	Способ получения слота	Присоединенная процедура (демон)

Иногда применяют другой вариант:

Имя фрейма			
Имя слота	Указатель типа данных	значение слота	Присоединенная процедура (демон)

Значения столбцов этой таблицы:

1. Имя фрейма – идентификатор, присваиваемый фрейму; это имя – единственное в данной системе, т. е. уникальное имя.

2. Имя слота – идентификатор, присваиваемый слоту; это уникальное имя во фрейме, к которому он принадлежит. Обычно имя слота не несет никакой смысловой нагрузки, но в ряде случаев может иметь специфический смысл. В их число входят слоты IS-A или A KIND OF (орел), показывающие фрейм-родитель данного фрейма (АКО-связи), слот указателей дочерних фреймов, дата изменения фрейма, имен пользователей, текста комментариев и др. Такие слоты называются системными и используются при редактировании БЗ и управлении выводом.

3.Указатель типа данных (атрибутов слота), показывает, что слот имеет численное значение либо служит указателем другого фрейма. Возможные типы значений: INTEGER – целый, REAL – действительный, BOOL – булев, текст, список, таблица, указатель на другой фрейм, LISP – вызываемая процедура.

4. Значение слота – должно совпадать с указанным типом данных этого слота, кроме того, должно выполняться условие наследования.

5. Демон – процедура, автоматически запускаемая при выполнении некоторого условия. Условия бывают следующих типов: IF-NEEDED – если в момент обращения к слоту его значение не будет установлено, IF-ADDED – при подстановке в слот значения, IF-REMOVED – при стирании значения слота.

Дополнительные столбцы предназначены для описания способа получения слотом его значения и возможного присоединения к тому или иному слоту специальных процедур, которые

выполняются, когда информация в слотах (значения атрибутов) меняется. С каждым слотом можно связать любое число процедур.

Процедуры должны решать следующие задачи:

1.поместить новую информацию в слот;

2.удалить информацию из слота;

3.обработать обращение к информации пока не заполненного слота.

Конкретные процедуры, включаемые в слот, делят на два типа: - процедуры-демоны – активизируются автоматически каждый раз, когда данные попадают в соответствующий фрейм-пример или удаляются из него. Демон, в основном, имеет структуру ЕСЛИ-ТОГДА. Эта процедура выполняется каждый раз, когда атрибут в условной части изменяет свое значение. С помощью процедур этого типа автоматически выполняются все рутинные операции, связанные с ведением баз данных и знаний (обновление). - процедуры-слуги – активизируются только по запросу. Например, если пользователь не указал год, то активизируется процедура-слуга. Фреймы и слоты описывают ситуацию в семантических форматах. С каждым слотом фрейма связаны описания условий, которые должны быть соблюдены, чтобы могло произойти означивание слота. В более сложных случаях условия могут касаться отношения между значениями, выбираемыми сразу для нескольких слотов. В качестве значения слота может выступать имя другого фрейма, так образуются сети фреймов. Существуют несколько способов получения слотом значений во фрейме-экземпляре: - по умолчанию от фрейма-образца (Delauf – значение); - через наследование свойств от фрейма, указанного в слоте АКО; - по формуле, указанной в слоте; - через присоединенную процедуру; - явно из диалога с пользователем; - из базы данных. Слотам фрейма могут быть заранее приписаны по умолчанию некоторые стандартные значения и факты. Это позволяет с помощью фреймов анализировать ситуации, в которых отсутствует упоминание о ряде деталей. Стандартные значения, присвоенные по умолчанию, могут быть заменены значениями, подходящими для обрабатываемой ситуации. Фрейм с незаполненными слотами называют протофреймом. Фрейм с заполненными слотами называют фреймом-экземпляром.

Пример представления знаний фреймами.

Пусть есть список сотрудников:

Иванов 1965 слесарь

Петров 1975 токарь.
Сидоров 1970 токарь. Попов 1968 наладчикю
Протофрейм будет иметь вид:
Сотрудник:
Фамилия
Год рождении
Специальность
Стаж

Важнейшим свойством теории фреймов является заимствование из теории семантических сетей наследование свойств. Такое наследование происходит по АКО-связям (A-King-Of – это). Слот АКО указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются, т.е. переносятся, значения аналогичных слотов. Пример (рис.3): .

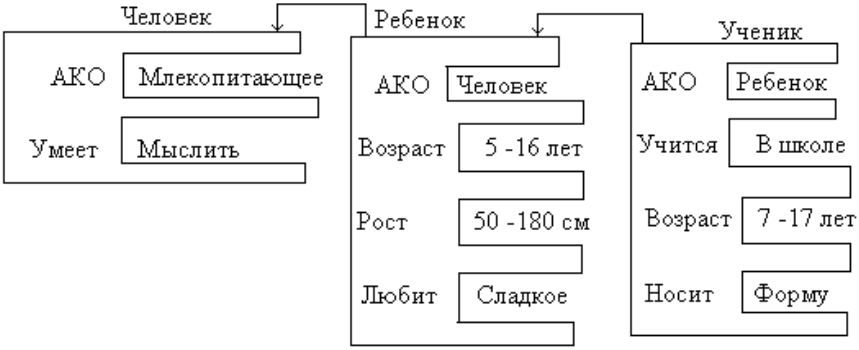


Рис.3. Пример наследования свойств.

Понятие «ученик» наследует свойства фреймов «ребенок» и «человек», которые находятся на более высоком уровне иерархии, На вопрос: «любят ли ученики сладкое?» следует ответ - «да», т.к. этим свойством обладают все дети, указанные во фрейме «ребенок».

Задания:

Построить фреймовую модель представления знаний в предметной области «Аэропорт» (диспетчерская).

Построить фреймовую модель представления знаний в предметной области «Железная дорога» (продажа билетов)

3. Построить фреймовую модель представления знаний в предметной области «Торговый центр» (организация).

4. Построить фреймовую модель представления знаний в предметной области «Автозаправка» (обслуживание клиентов).

5. Построить фреймовую модель представления знаний в

предметной области «Автопарк» (пассажирские перевозки).

6. Построить фреймовую модель представления знаний в предметной области «Компьютерные сети» (организация).

7. Построить фреймовую модель представления знаний в предметной области «Университет» (учебный процесс).

8. Построить фреймовую модель представления знаний в предметной области «Компьютерная безопасность» (средства и способы ее обеспечения).

9.Построить фреймовую модель представления знаний в предметной области «Компьютерная безопасность» (угрозы).

10 Построить фреймовую модель представления знаний в предметной области «Интернет-кафе» (организация и обслуживание).

11.Построить фреймовую модель представления знаний в предметной области «Разработка информационных систем» (ведение информационного проекта).

12.Построить фреймовую модель представления знаний в предметной области «Туристическое агентство» (работа с клиентами)

ЧАСТЬ 2. ОСНОВЫ РАБОТЫ В MATLAB

Методические указания по выполнению лабораторных работ

Первая часть - знакомство с основами работы с пакетом Matlab (работа №1). Вторая часть – выполнение заданий по лабораторной работе №2. Номер задания для выполнения совпадает с порядковым номером студента в списке группы. Работу следует выполнять в вычислительном классе кафедры. Отчет о выполнении необходимо сдать преподавателю в распечатанном виде.

Лабораторная работа №1 «Основы MATLAB. Часть 1»

Цель лабораторной работы

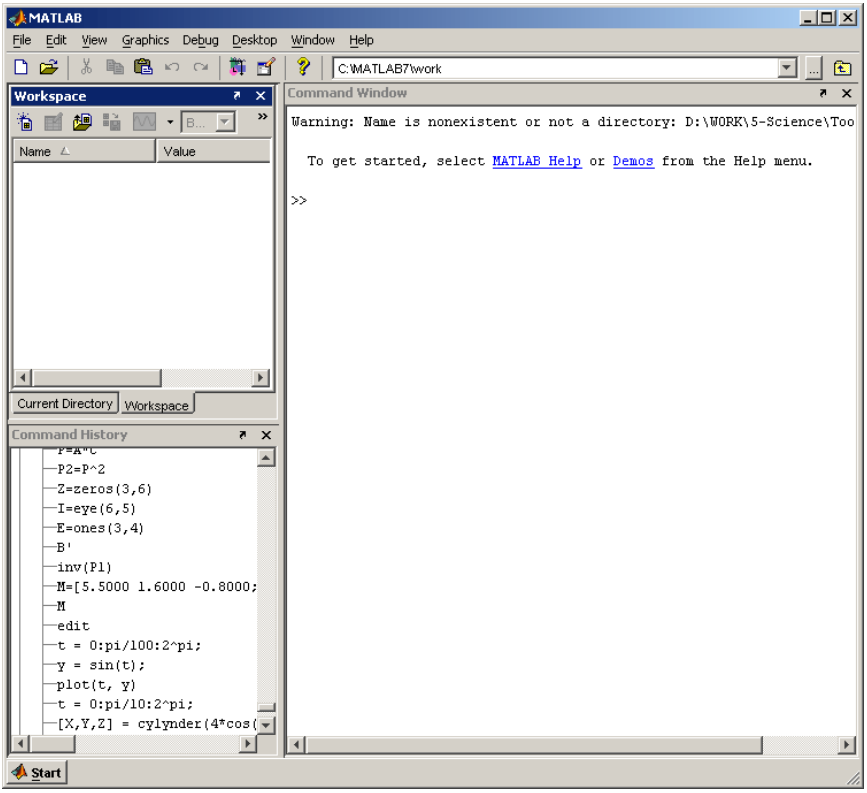
Целью работы является изучение основных возможностей пакета *MatLab*, а также получение практических навыков работы с матрицами и средствами графической визуализации вычислений.

1. Базовые сведения

1.1. Рабочая среда MatLab

Чтобы запустить программу дважды щелкните на иконку

MATLAB 6.5. Перед Вами откроется рабочая среда, изображенная на рисунке.



Команды следует набирать в командном окне. Символ », обозначающий приглашение к вводу командной строки, набирать не нужно. Для просмотра рабочей области удобно использовать полосы скроллинга или клавиши **Home**, **End**, для перемещения влево или вправо, и **PageUp**, **PageDown** для перемещения вверх или вниз. Если вдруг после перемещения по рабочей области командного окна пропала командная строка с мигающим Рабочая среда MatLab 6.x немного отличается от рабочей среды предыдущих версий, она имеет более удобный интерфейс для доступа ко многим вспомогательным элементам.

Рабочая среда MatLab 6.x содержит следующие элементы:

• панель инструментов с кнопками и раскрывающимся списком;

• окно с вкладками **Launch Pad** и **Workspace**, из которого можно получить доступ к различным модулям ToolBox и к содержимому рабочей среды;

• окно с вкладками **Command History** и **Current Directory**, предназначенное для просмотра и повторного вызова ранее введенных команд, а также для установки текущего каталога;

• командное окно, в котором находится приглашение к вводу » и мигающий вертикальный курсор;

• строку состояния.

Если в рабочей среде MatLab 6.x отсутствуют некоторые окна, приведенные на рисунке, то следует в меню **View** выбрать соответствующие пункты: **Command Window**, **Command History**, **Current Directory**, **Workspase**, **Launch Pad**.

курсором, просто нажмите **Enter**.

Важно помнить, что набор любой команды или выражения должен заканчиваться нажатием на **Enter**, для того, чтобы программа MatLab выполнила эту команду или вычислила выражение.

1.2. Простейшие вычисления

Наберите в командной строке 1+2 и нажмите **Enter**. В результате в командном окне MatLab отображается следующее:

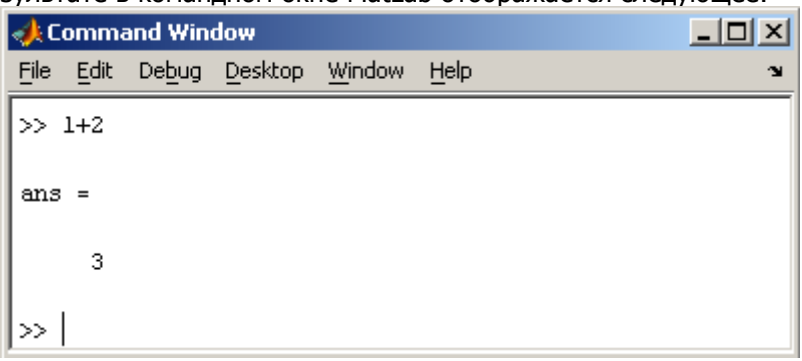


Рис. 2 Графическое представление метода главных компонент

Что сделала программа MatLab? Сначала она вычислила сумму 1+2, затем записала результат в специальную переменную ans и вывела ее значение, равное 3, в командное окно. Ниже ответа расположена командная строка с мигающим курсором, обозначающая, что MatLab готов к дальнейшим вычислениям. Можно набирать в командной строке новые выражения и находить их значения. Если требуется продолжить работу с предыдущим выражением, например, вычислить (1+2)/4.5, то проще всего воспользоваться уже имеющимся результатом, который хранится в

переменной ans. Наберите ans/4.5 (при вводе десятичных дробей используется точка) и нажмите **Enter**, получается

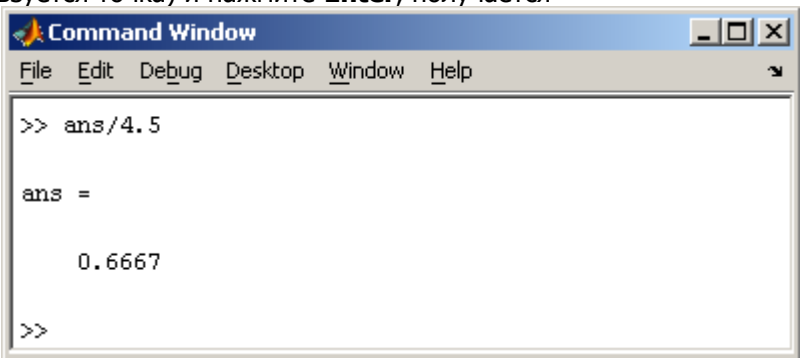


Рис. 3 Графическое представление метода главных компонент

1.3. Эхо команд

Выполнение каждой команды в MatLab сопровождается эхом. В приведенном выше примере — это ответ ans = 0.6667. Часто эхо затрудняет восприятие работы программы и тогда его можно отключить. Для этого команда должна завершаться символом точка с запятой. Например

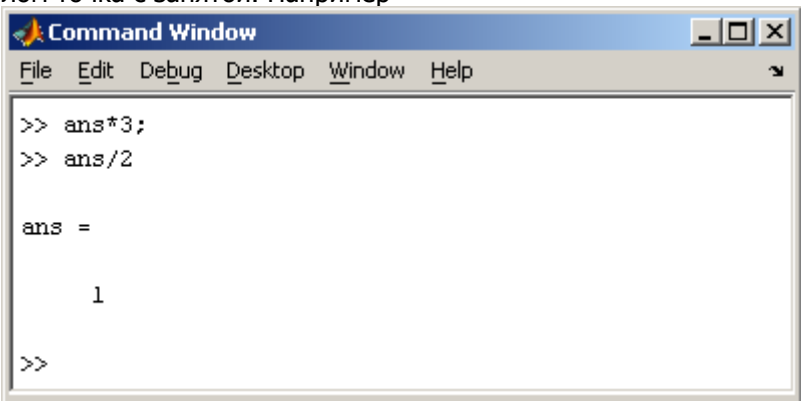


Рис. 4 Пример ввода функции ScoresPCA.

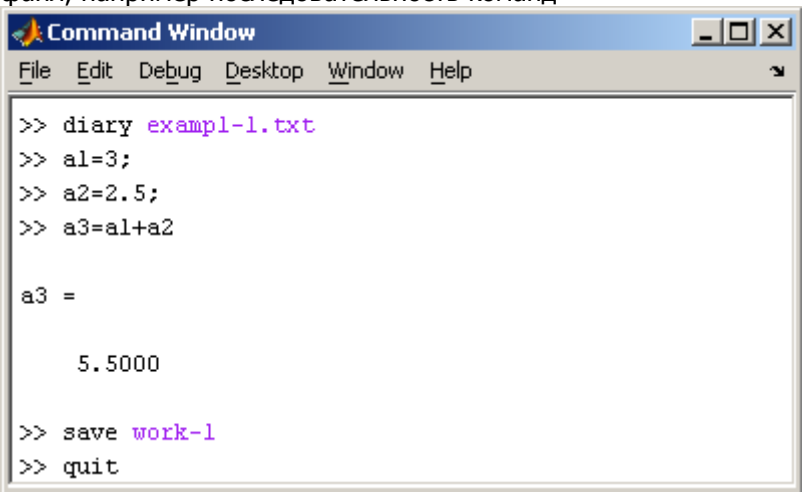
1.4. Сохранение рабочей среды. MAT файлы

Самый простой способ сохранить все значения переменных — использовать в меню File пункт Save Workspase As. При этом появляется диалоговое окно Save Workspase Variables, в котором следует указать каталог и имя файла. По умолчанию предлагается сохранить файл в подкаталоге work основного каталога MatLab. Программа сохранит результаты работы в файле с расши-

рением mat. Теперь можно закрыть MatLab. В следующем сеансе работы для восстановления значений переменных следует открыть этот сохраненный файл при помощи подпункта Open меню File. Теперь все переменные, определенные в прошлом сеансе, опять стали доступными. Их можно использовать во вновь вводимых командах.

1.5. Журнал

В MatLab имеется возможность записывать исполняемые команды и результаты в текстовый файл (вести журнал работы), который потом можно прочитать или распечатать из текстового редактора. Для начала ведения журнала служит команда diary. В качестве аргумента команды diary следует задать имя файла, в котором будет храниться журнал работы. Набираемые далее команды и результаты их исполнения будут записываться я в этот файл, например последовательность команд



производит следующие действия:

- открывает журнал в файле exampl-1.txt;
- производит вычисления;
- сохраняет все переменные в MAT файле work-1.mat;
- сохраняет журнал в файле exampl-1.txt в подкаталоге work корневого каталога MatLab и закрывает MatLab;

Посмотрите содержимое файла exampl-1.txt в каком-нибудь текстовом редакторе. В файле окажется следующий текст:

```
      a1=3;
a2=2.5;
a3=a1+a2

a3                                    =

                                   5.5000

save                               work-1
quit
```

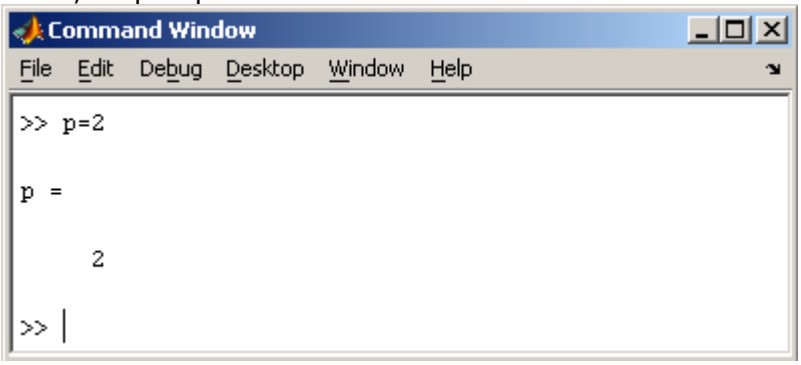
1.6. Система помощи

Окно справки MatLab появляется после выбора опции Help Window в меню Help или нажатием кнопки вопроса на панели инструментов. Эта же операция может быть выполнена при наборе команды helpwin. Для вывода окна справки по отдельным разделам, наберите helpwin topic. Окно справки предоставляет Вам такую же информацию, как и команда help, но оконный интерфейс обеспечивает более удобную связь с другими разделами справки.

2. Матрицы

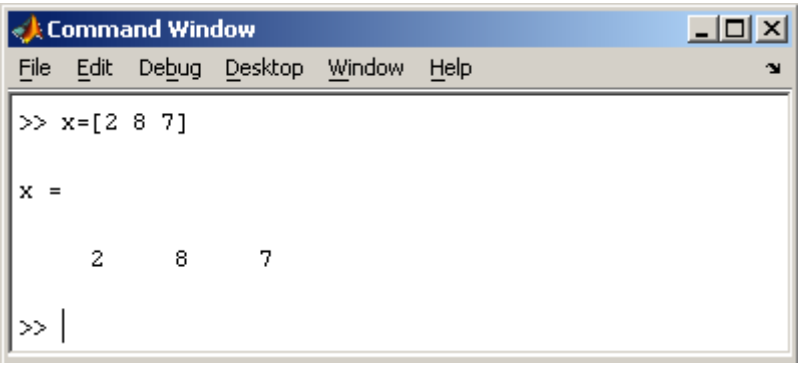
2.1. Скаляры, векторы и матрицы

В MatLab можно использовать скаляры, векторы и матрицы. Для ввода скаляра достаточно приписать его значение какой-то переменной, например

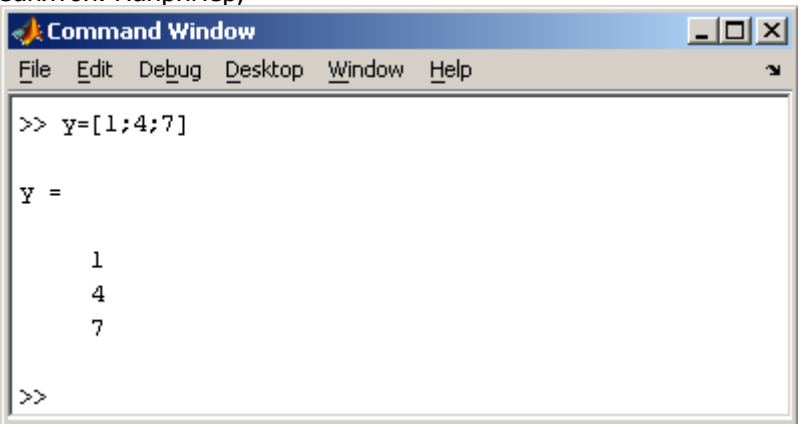


MatLab различает заглавные и прописные буквы, так что p и P — это разные переменные. Для ввода массивов (векторов или матриц) их элементы заключают в квадратные скобки. Так для ввода вектора-строки размером 1×3, используется следующая

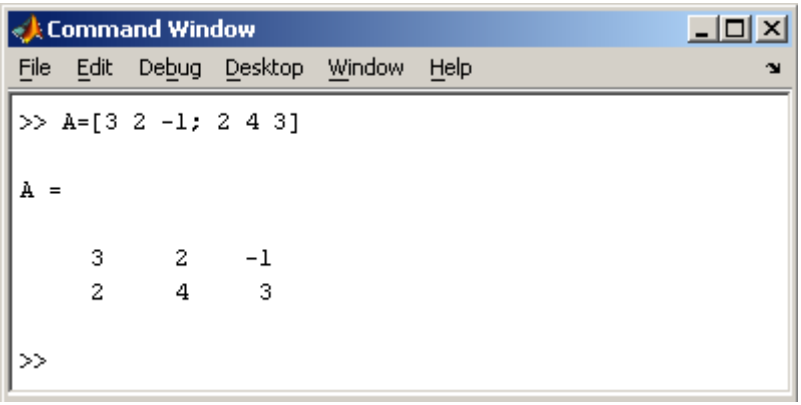
команда, в которой элементы строки отделяются пробелами или запятыми.



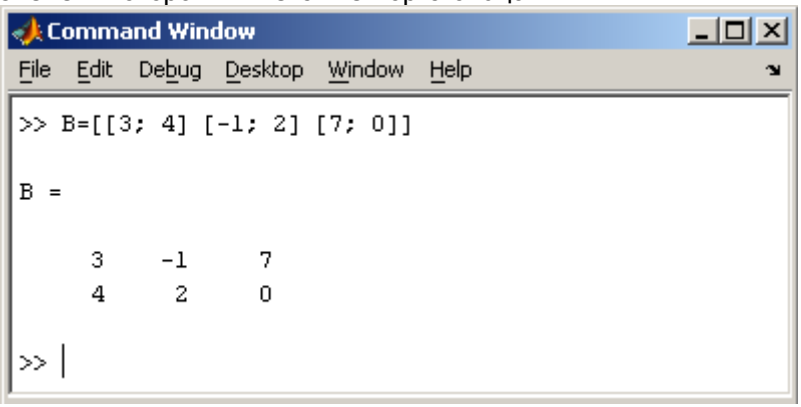
При вводе вектора-столбца элементы разделяют точкой с запятой. Например,



Вводить небольшие по размеру матрицы удобно прямо из командной строки. При вводе матрицу можно рассматривать как вектор-столбец, каждый элемент которого является вектор-строкой.

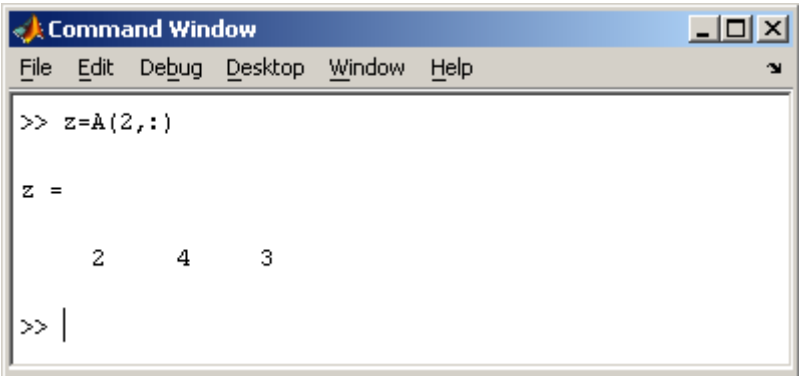


или матрицу можно трактовать как вектор строку, каждый элемент которой является вектор-столбцом.

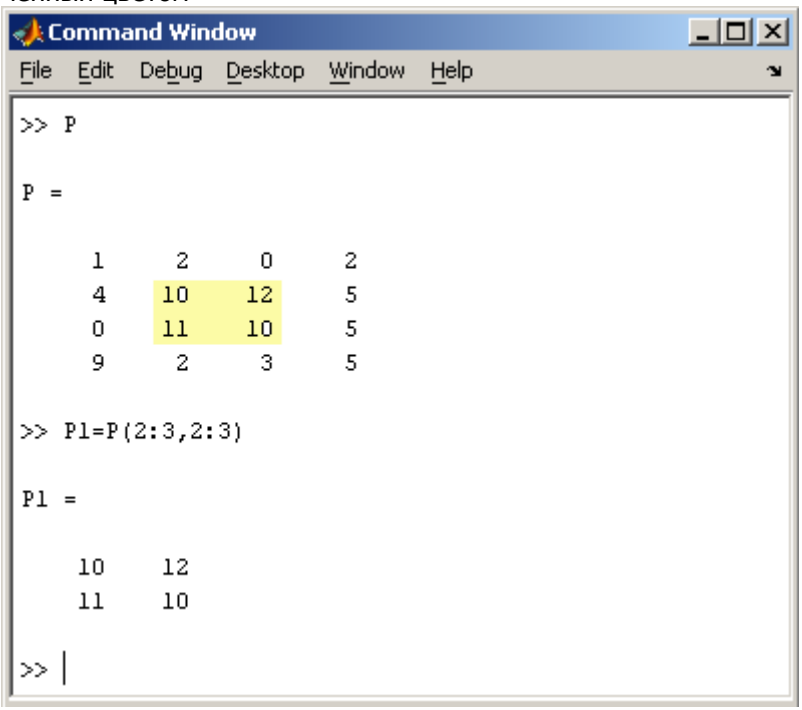


2.2. Доступ к элементам

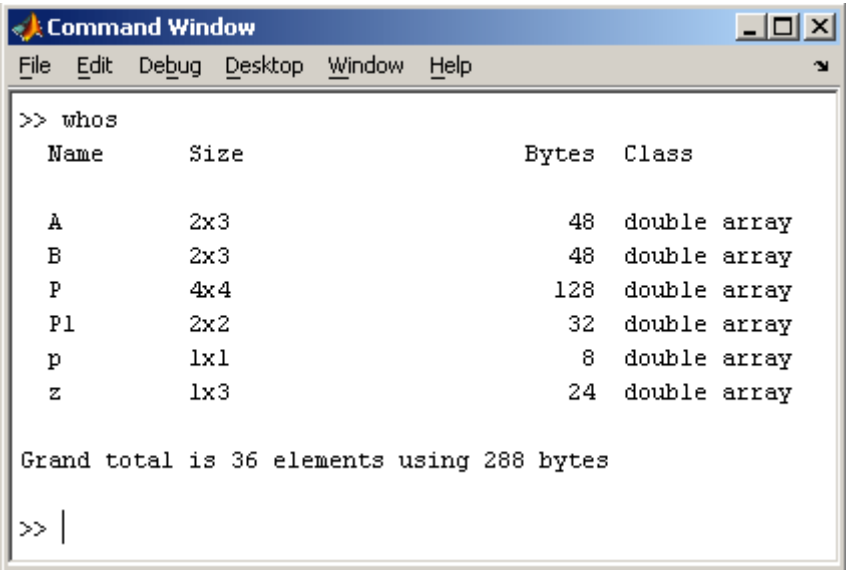
Доступ к элементам матриц осуществляется при помощи двух индексов — номеров строки и столбца, заключенных в круглые скобки, например команда B(2,3) выдаст элемент второй строки и третьего столбца матрицы B. Для выделения из матрицы столбца или строки следует в качестве одного из индексов использовать номер столбца или строки матрицы, а другой индекс заменить двоеточием. Например, запишем вторую строку матрицы A в вектор z



Также можно осуществлять выделение блоков матриц при помощи двоеточия. Например, выделим из матрицы P блок отмеченный цветом



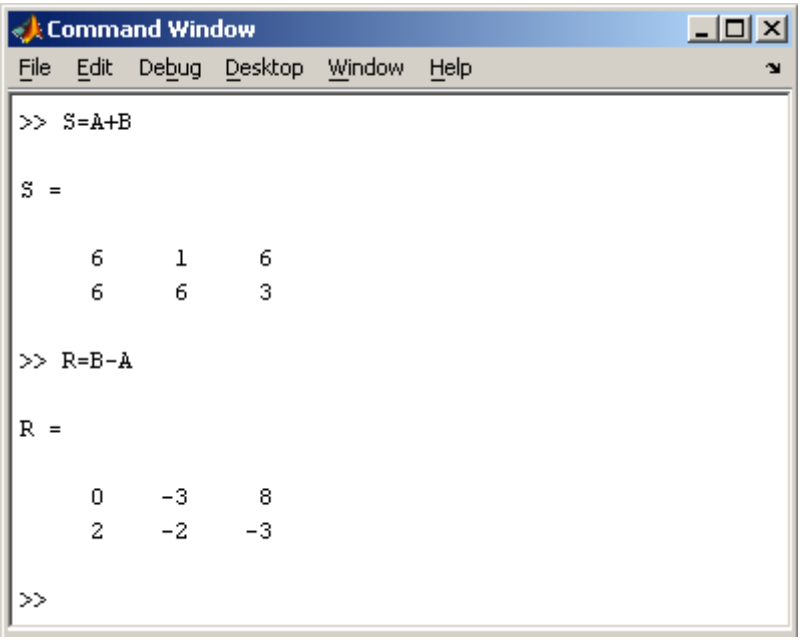
Если необходимо посмотреть переменные рабочей среды, в командной строке необходимо набрать команду whos.



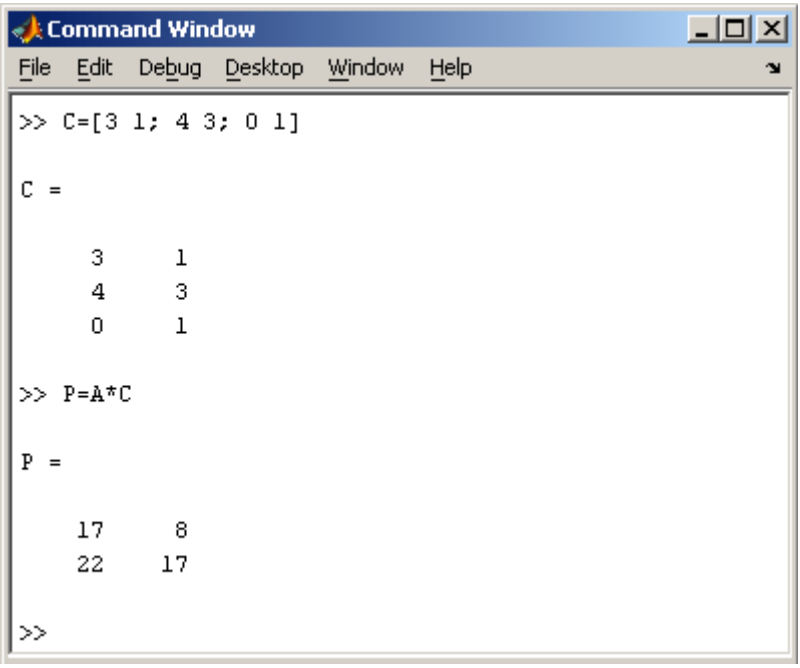
Видно, что в рабочей среде содержатся один скаляр (p), четыре матрицы (A, B, P, P1) и вектор-строка (z).

2.3. Основные матричные операции

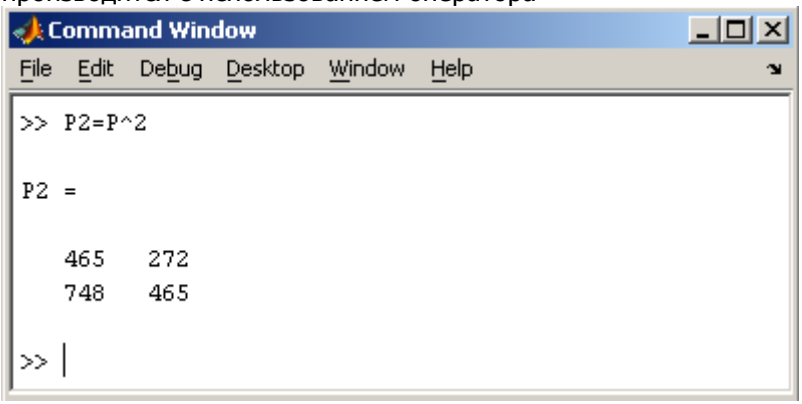
При использовании матричных операций следует помнить, что для сложения или вычитания матрицы должны быть одного размера, а при перемножении число столбцов первой матрицы обязано равняться числу строк второй матрицы. Сложение и вычитание матриц, так же как чисел и векторов, осуществляется при помощи знаков плюс и минус



а умножение — знаком звездочка *. Введем матрицу размером 3×2



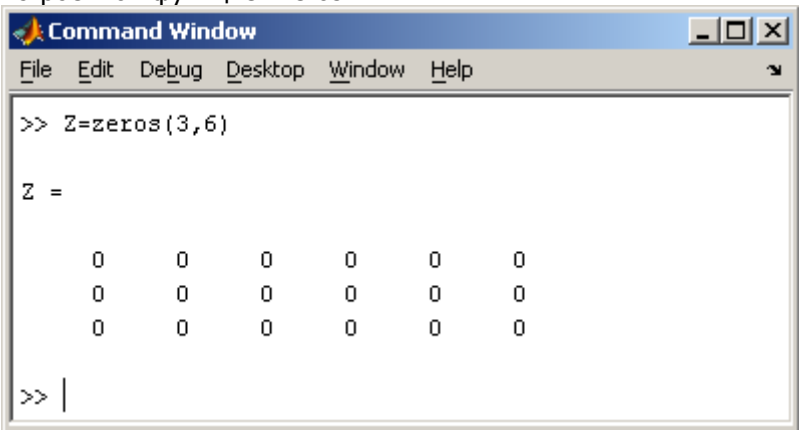
Умножение матрицы на число тоже осуществляется при помощи звездочки, причем умножать на число можно как справа, так и слева. Возведение квадратной матрицы в целую степень производится с использованием оператора ^



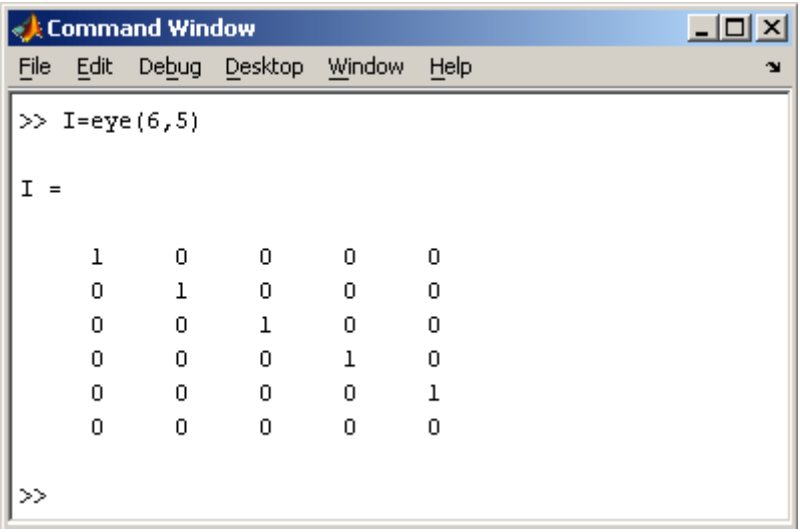
Проверьте полученный результат, умножив матрицу P саму на себя.

2.4. Создание матриц специального вида

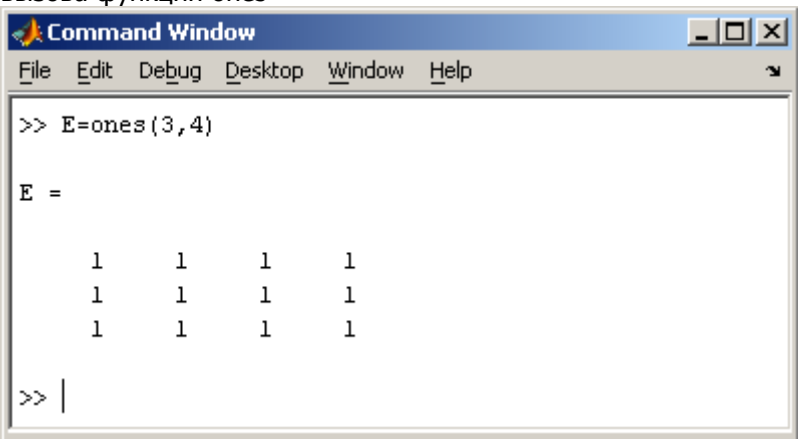
Заполнение прямоугольной матрицы нулями производится встроенной функцией zeros



Единичная матрица создается при помощи функции eye



Матрица, состоящая из единиц, образуется в результате вызова функции ones

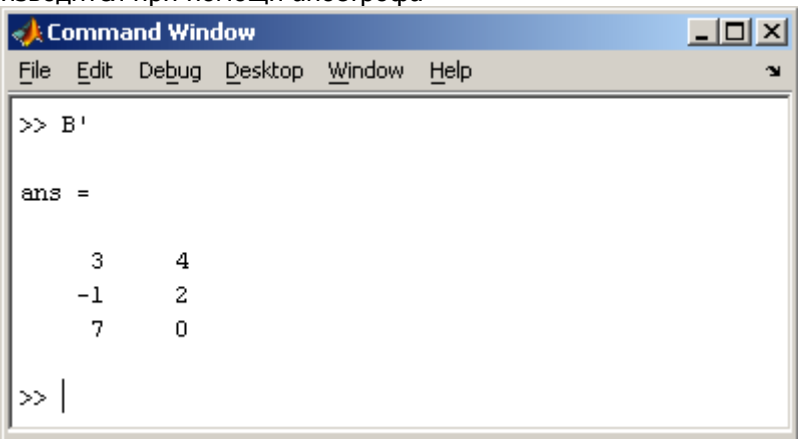


MatLab предоставляет возможность заполнения матриц случайными числами. Результатом функции rand является матрица чисел, равномерно распределенных между нулем и единицей, а функции randn — матрица чисел, распределенных по нормальному закону с нулевым средним и единичной дисперсией.

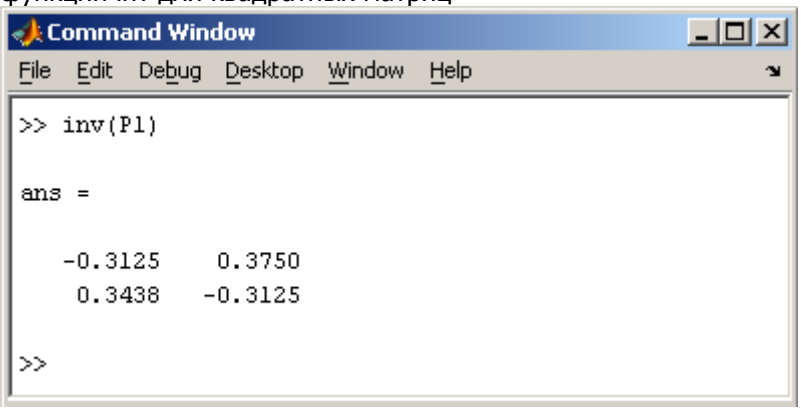
Функция diag формирует диагональную матрицу из вектора, располагая элементы по диагонали.

2.5. Матричные вычисления

MatLab содержит множество различных функций для работы с матрицами. Так, например, транспонирование матрицы производится при помощи апострофа '



Нахождение обратной матрицы проводится с помощью функции inv для квадратных матриц



Псевдообратную матрицу можно найти с помощью функции pinv.

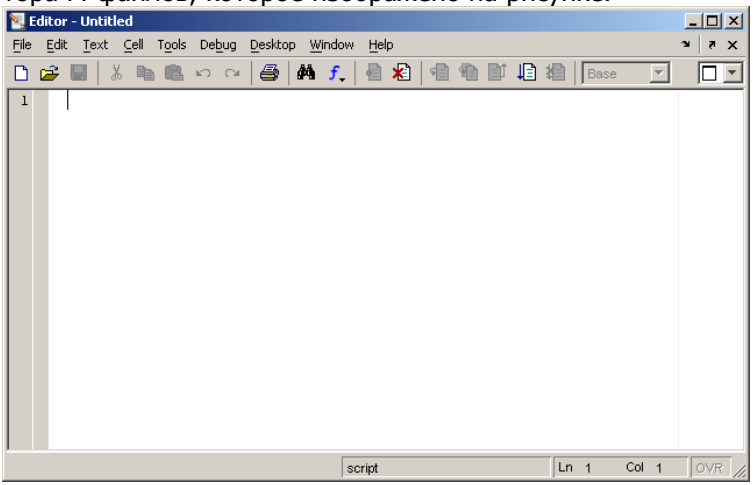
Более подробно про обработку матричных данных можно узнать, если вывести список всех встроенных функций обработки данных командой help datafun, а затем посмотреть информацию о нужной функции, например help max.

3. Программирование

3.1. М-файлы

Работа из командной строки MatLab затрудняется, если требуется вводить много команд и часто их изменять. Ведение дневника при помощи команды diary и сохранение рабочей среды незначительно облегчают работу. Самым удобным способом выполнения групп команд MatLab является использование М-файлов, в которых можно набирать команды, выполнять их все сразу или частями, сохранять в файле и использовать в дальнейшем. Для работы с М-файлами предназначен редактор М-файлов. С его помощью можно создавать собственные функции и вызывать их, в том числе и из командного окна.

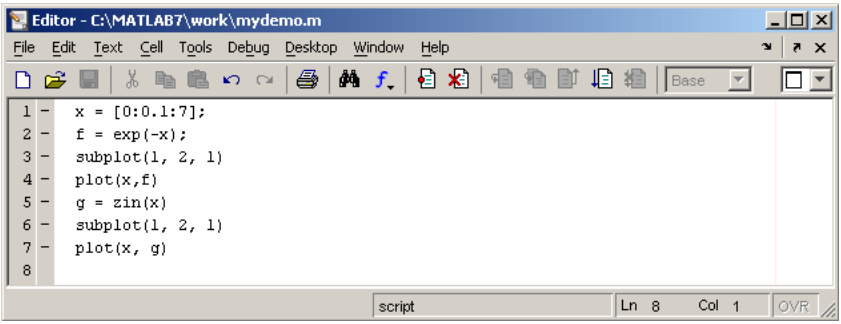
Раскройте меню File основного окна MatLab и в пункте New выберите подпункт M-file. Новый файл открывается в окне редактора М-файлов, которое изображено на рисунке.



М-файлы в MatLab бывают двух типов: файл-программы (*Script M-Files*), содержащие последовательность команд, и файл-функции, (*Function M-Files*), в которых описываются функции, определяемые пользователем.

3.2. Файл-программа

Наберите в редакторе команды, приводящие к построению двух графиков на одном графическом окне

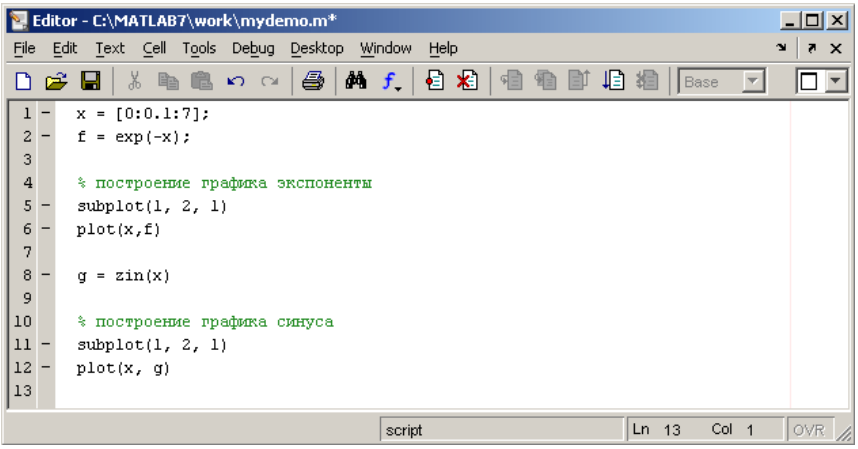


Сохраните теперь файл с именем mydemo.m в подкаталоге work основного каталога MatLab, выбрав пункт **Save as** меню **File** редактора. Для запуска на выполнение всех команд, содержащихся в файле, следует выбрать пункт **Run** в меню **Debug**. На экране появится графическое окно *Figure 1*, содержащее графики функций.

Команды файл-программы осуществляют вывод в командное окно. Для подавления вывода следует завершать команды точкой с запятой. Если при наборе сделана ошибка и MatLab не может распознать команду, то происходит выполнение команд до неправильно введенной, после чего выводится сообщение об ошибки в командное окно.

Очень удобной возможностью, предоставляемой редактором М-файлов, является выполнение части команд. Закройте графическое окно *Figure 1*. Выделите при помощи мыши, удерживая левую кнопку, или клавишами со стрелками при нажатой клавише **Shift**, первые четыре команды и выполните их из пункта **Text**. Обратите внимание, что в графическое окно вывелся только один график, соответствующий выполненным: командам. Запомните, что для выполнения части команд их следует выделить и нажать клавишу **F9**.

Отдельные блоки М-файла можно снабжать комментариями, которые пропускаются при выполнении, но удобны при работе с М-файлом. Комментарии начинаются со знака процента и автоматически выделяются зеленым цветом, например:

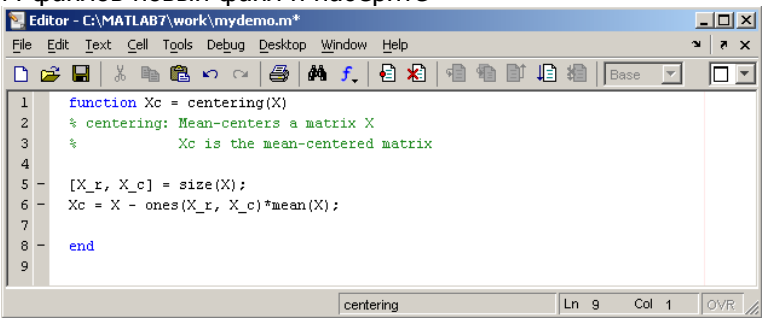


Открытие существующего М-файла производится при помощи пункта **Open** меню **File** рабочей среды, либо редактора М-файлов.

3.3. Файл-функция

Рассмотренная выше файл-программа является только последовательностью команд MatLab, она не имеет входных и выходных аргументов. Для использования численных методов и при программировании собственных приложений в MatLab необходимо уметь составлять файл-функции, которые производят необходимые действия с входными аргументами и возвращают результат действия в выходных аргументах. Разберем несколько простых примеров, позволяющих понять работу с файл-функциями.

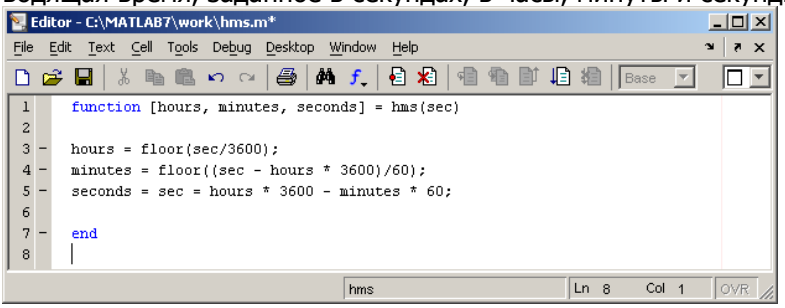
Проводя предобработку данных многомерного анализа хемометрики часто применяет центрирование. Имеет смысл один раз написать файл-функцию, а потом вызывать его всюду, где необходимо производить центрирование. Откройте в редакторе М-файлов новый файл и наберите



Слово function в первой строке определяет, что данный файл содержит файл-функцию. Первая строка является заголовком функции, в которой размещается имя функции и списка входных и выходных аргументов. В примере имя функции centering, один входной аргумент X и один выходной — Xc. После заголовка следуют комментарии, а затем — тело функции (оно в данном примере состоит из двух строк), где и вычисляется ее значение. Важно, что вычисленное значение записывается в Xc. Не забудьте поставить точку с запятой для предотвращения вывода лишней информации на экран. Теперь сохраните файл в рабочем каталоге. Обратите внимание, что выбор пункта **Save** или **Save as** меню **File** приводит к появлению диалогового окна сохранения файла, в поле **File name** которого уже содержится название centering. Не изменяйте его, сохраните файл функцию в файле с предложенным именем!

Теперь созданную функцию можно использовать так же, как и встроенные sin, cos и другие. Вызов собственных функций может осуществляться из файл-программы и из другой файл-функции. Попробуйте сами написать файл-функцию, которая будет шкалировать матрицы, т.е. делить каждый столбец на величину среднеквадратичного отклонения по этому столбцу.

Можно написать файл-функции с несколькими входными аргументами, которые размещаются в списке через запятую. Можно также создавать и функции, возвращающие несколько значений. Для этого выходные аргументы добавляются через запятую в список выходных аргументов, а сам список заключается в квадратные скобки. Хорошим примером является функция, переводящая время, заданное в секундах, в часы, минуты и секунды.

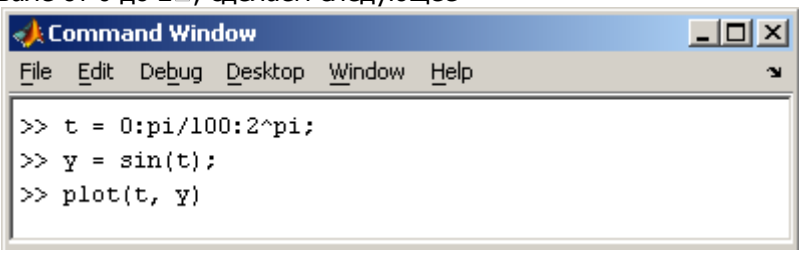


При вызове файл-функций с несколькими выходными аргументами результат следует записывать в вектор соответствующей длины.

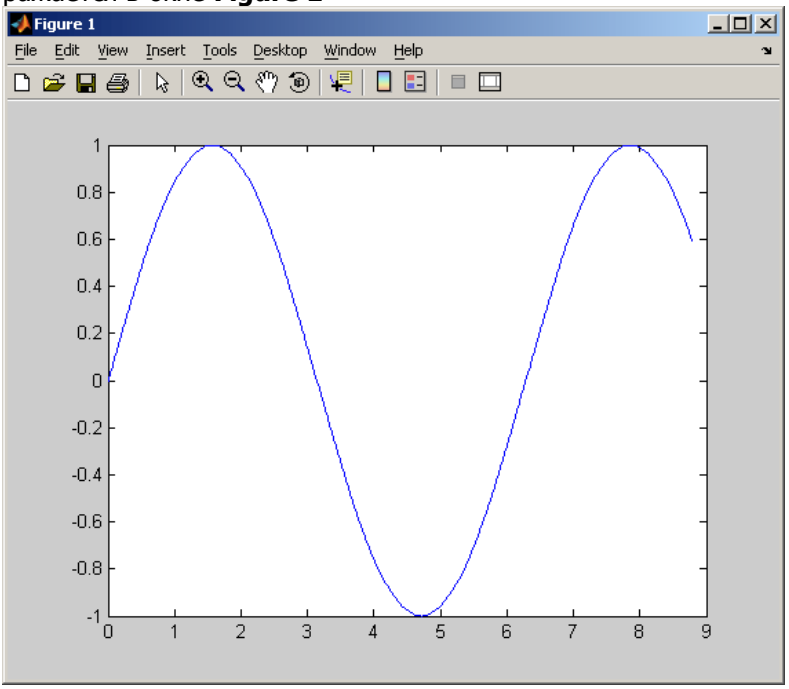
3.4 Создание графика

MatLab имеет широкие возможности для графического изображения векторов и матриц, а также для создания комментариев и печати графиков. Дадим описание несколько важных графических функций.

Функция plot имеет различные формы, связанные с входными параметрами, например plot(y) создает кусочно-линейный график зависимости элементов y от их индексов. Если в качестве аргументов заданы два вектора, то plot(x,y) создаст график зависимости y от x. Например, для построения графика функции sin в интервале от 0 до 2□, сделаем следующее

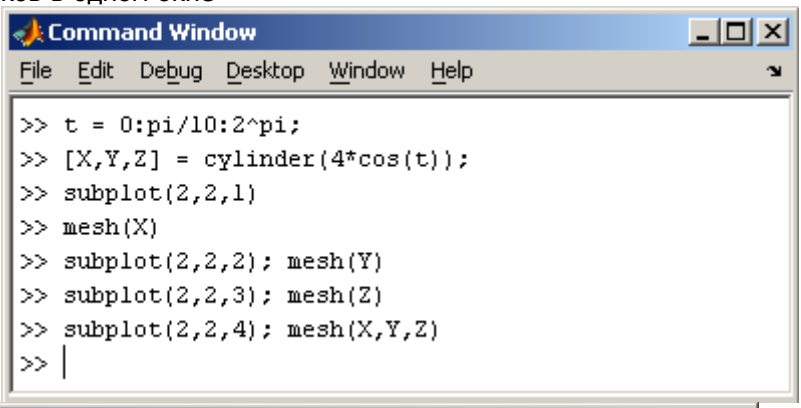


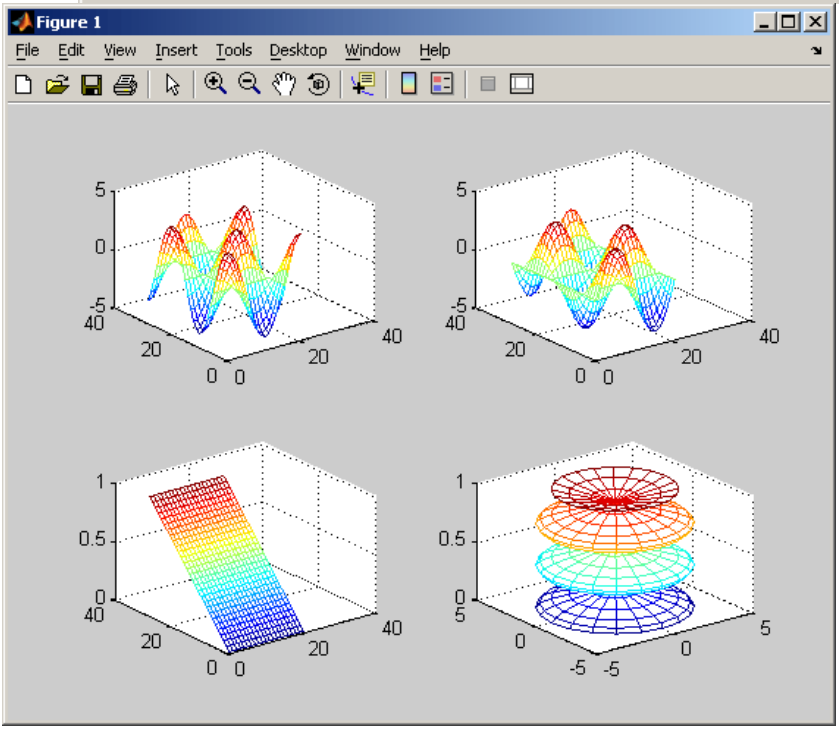
Программа построила график зависимости, который отображается в окне **Figure 1**



MatLab автоматически присваивает каждому графику свой цвет (исключая случаи, когда это делает пользователь), что позволяет различать наборы данных.

Команда hold on позволяет добавлять кривые на существующий график. Функция subplot позволяет выводить множество графиков в одном окне





3.5 Печать графиков

Пункт Print в меню File и команда print печатают графику MatLab. Меню Print вызывает диалоговое окно, которое позволяет выбирать общие стандартные варианты печати. Команда print обеспечивает большую гибкость при выводе выходных данных и позволяет контролировать печать из M-файлов. Результат может быть послан прямо на принтер, выбранный по умолчанию, или сохранен в заданном файле.

4. Примеры программ

В этом разделе приведены наиболее употребительные алгоритмы, используемые при анализе многомерных данных. Рассмотрены как простейшие методы преобразования данных центрирование и шкалирование, так и алгоритмы для анализа данных — PCA, PLS.

4.1. Центрирование и шкалирование

Часто при анализе требуется преобразовать исходные данные. Наиболее используемыми методами преобразования данных выступают центрирование и шкалирование каждой переменной на стандартное отклонение. В разделе 4.3 приводился код функции для центрирования матрицы. Поэтому ниже показан только код функции, которая шкалирует данные. Обратите внимание, что исходная матрица должна быть центрирована

```
function Xs = scaling(X)
% scaling: the output matrix is Xs
% matrix X must be centered

Xs = X * inv(diag(std(X)));

%end of scaling
```

4.2. SVD/PCA

Наиболее популярным способом сжатия данных в многомерном анализе является метод главных компонент (PCA). С математической точки зрения PCA — это декомпозиция исходной матрицы X, т.е. представление ее в виде произведения двух матриц T и P

$X = TP^t + E$

Матрица T называется матрицей счетов (scores) , матрица P — матрицей нагрузок (loadings), а E — матрицей остатков.

Простейший способ найти матрицы T и P — использовать SVD разложение через стандартную функцию MatLab, называемую svd.

```
function [T, P] = pcasvd(X)
% pcasvd: calculates PCA components.
% The output matrices are T and P.
% T contains scores
% P contains loadings
[U,D,V] = svd(X);
T = U * D;
P = V;
%end of pcasvd
```

4.3 PCA/NIPALS

Для построения PCA счетов и нагрузок, используется рекурентный алгоритм NIPALS, который на каждом шагу вычисляет одну компоненту. Сначала исходная матрица X преобразуется (как минимум – центрируется; см. раздел 4.3) и превращается в матрицу E_0, a=0. Далее применяют следующий алгоритм.

1. Выбрать начальный вектор t
2. $p^t = t^t E_a / t^t t$
3. $p = p / (p^t p)^{½}$
4. $t = E_a p / p^t p$
5. Проверить сходимость, если нет, то идти на 2

После вычисления очередной (a-ой) компоненты, полагаем t_a=t и p_a=p. Для получения следующей компоненты надо вычислить остатки $E_{a+1} = E_a – t\ p^t$ и применить к ним тот же алгоритм, заменив индекс a на a+1.

Код алгоритма NIPALS может быть написан и самими читателями, в данном же пособии авторы приводят свой вариант. При расчете PCA, можно вводить число главных компонент (переменная numberPC). Если же не известно, сколько необходимо компонент, следует написать в командной строке [P,T] = pcanipals (X) и тогда программа задаст число компонент равным наименьшему из показателей размерности исходной матрицы X.

```
function [T, P] = pcanipals(X, numberPC)
% pcanipals: calculates PCA components.
% The output matrices are T and P.
% T contains scores
% P contains loadings
```

```
% calculation of number of components
[X_r, X_c] = size(X); P=[]; T=[];

if lenfth(numberPC) > 0
      pc = numberPC{1};
elseif (length(numberPC) == 0) & X_r < X_c
      pc = X_r;
else
      pc = X_c;
end;

% calculation of scores and loadings for each
component
for k = 1:pc
      P1 = rand(X_c, 1); T1 = X * P1; d0 =
T1'*T1;
      P1 = (T1' * X/(T1' * T1))'; P1 =
P1/norm(P1); T1 = X * P1; d = T1' * T1;

      while d - d0 > 0.0001;
            P1 = (T1' * X/(T1' * T1)); P1 =
P1/norm(P1); T1 = X * P1; d0 = T1'*T1;
            P1 = (T1' * X/(T1' * T1)); P1 =
P1/norm(P1); T1 = X * P1; d = T1'*T1;
      end

      X = X - T1 * P1; P = cat(1, P, P1'); T =
[T,T1];
end
```

4.4 PLS1

Самым популярным способом для многомерной калибровки является метод проекции на латентные структуры (PLS). В этом методе проводится одновременная декомпозиция матрицы предикторов X и матрицы откликов Y:

$X=TP^t+E$ $Y=UQ^t+F$ $T=XW(P^tW)^{-1}$

Проекция строится согласованно – так, чтобы максимизировать корреляцию между соответствующими векторами X-счетов t_a и Y-счетов u_a. Если блок данных Y включает несколько откликов (т.е. K>1), можно построить две проекции исходных данных – PLS1 и PLS2. В первом случае для каждого из откликов y_k строится

свое проекционное подпространство. При этом и счета T (U) и нагрузки P (W, Q) , зависят от того, какой отклик используется. Этот подход называется PLS1. Для метода PLS2 строится только одно проекционное пространство, которое является общим для всех откликов.

Детальное описание метода PLS приведено в этой книге Для построения PLS1 счетов и нагрузок, используется рекуррентный алгоритм. Сначала исходные матрицы X и Y центрируют

```
[E0, mX] = mc(X);
[F0, mY] = mc(Y);
```

и они превращаются в матрицу E_0 и вектор f_0, a=0. Далее к ним применяет следующий алгоритм

1. $w^t = f_a^t E_a$
2. $w = w / (w^t w)^{½}$
3. $t = E_a w$
4. $q = t^t f_a / t^t t$
5. $u = q f_a / q^2$
6. $p^t = t^t E_a / t^t t$

После вычисления очередной (a-ой) компоненты, полагаем t_a=t и p_a=p. Для получения следующей компоненты надо вычислить остатки $E_{a+1} = E_a - t\ p^t$ и применить к ним тот же алгоритм, заменив индекс a на a+1.

Код этого алгоритма

```
function [w, t, u, q, p] = pls(x, y)
%PLS: calculates a PLS component.
%The output vectors are w, t, u, q and p.
%
% Choose a vector from y as starting vector u.

   u = y(:, 1);

% The convergence criterion is set very high.
   kri = 100;

% The commands from here to end are repeat-
ed until convergence.
   while (kri > 1e - 10)

% Each starting vector u is saved as uold.
      uold = u; w = (u' * x)'; w = w/norm(w);
```

```
    t = x * w; q = (t' * y)'/(t' * t);
    u = y * q/(q' * q);

% The convergence criterion is the norm of u-
uold divided by the norm of u.
    kri = norm(uold - u)/norm(u);
  end;

% After convergence, calculate p.
  p = (t' * x)'/(t' * t);
% End of pls
```

4.5 PLS2

Для PLS2 алгоритм выглядит следующим образом. Сначала исходные матрицы X и Y преобразуют (как минимум – центрируют), и они превращаются в матрицы E_0 и F_0, a=0. Далее к ним применяет следующий алгоритм.

1. Выбрать начальный вектор u
2. $w^t = u^t E_a$
3. $w = w / (w^t w)^{1/2}$
4. $t = E_a w$
5. $q^t = t^t F_a / t^t t$
6. $u = F_a q / q^t q$
7. Проверить сходимость, если нет, то идти на 2
8. $p^t = t^t E_a / t^t t$

После вычисления очередной (a-ой) PLS2 компоненты надо положить: $t_a=t$, $p_a=p$, $w_a=w$, $u_a=u$ и $q_a=q$. Для получения следующей компоненты надо вычислить остатки $E_{a+1} = E_a – t\ p^t$ и $F_{a+1} = F_a – tq^t$ и применить к ним тот же алгоритм, заменив индекс a на a+1.

Код

```
function [W, T, U, Q, P, B, SS] = plsr(x, y, a)
% PLS: calculates a PLS component.
% The output matrices are W, T, U, Q and P.
% B contains the regression coefficients and SS the
sums of
% squares for the residuals.
% a is the numbers of components.
%
% For a components: use all commands to end.

  for i=1:a
% Calculate the sum of squares. Use the function ss.
    sx = [sx; ss(x)];
    sy = [sy; ss(y)];

% Use the function pls to calculate one component.
    [w, t, u, q, p] = pls(x, y);

% Calculate the residuals.
    x = x - t * p';
    y = y - t * q';

% Save the vectors in matrices.
    W = [W w];
    T = [T t];
    U = [U u];
    Q = [Q q];
    P = [P p];
  end;

% Calculate the regression coefficients after the
loop.
  B=W*inv(P'*W)*Q';

% Add the final residual SS to the sum of squares
vectors.
  sx=[sx; ss(x)];
  sy=[sy; ss(y)];

% Make a matrix of the ss vectors for X and Y.
  SS = [sx sy];
```

```
%Calculate the fraction of SS used.
   [a, b] = size(SS);
   tt = (SS * diag(SS(1,:).^(-1)) - ones(a, b)) * (-1)

%End of plsr
function [ss] = ss(x)
%SS: calculates the sum of squares of a matrix X.
%
   ss=sum(sum(x. * x));
%End of ss
```

Лабораторная работа №2
«Основы MATLAB. Часть 2»

Цель лабораторной работы

Целью работы является получение практических навыков работы с матрицами и средствами графической визуализации вычислений.

Описание системы MATLAB

Краткая характеристика MATLAB

MATLAB - это высокопроизводительный инструмент для выполнения технических расчетов. Он включает в себя вычисления, визуализацию и программирование в удобной среде, где задачи и решения выражаются в форме, близкой к математической. Типичное использование *MATLAB* - это:

- математические вычисления
- создание алгоритмов
- моделирование
- анализ данных, исследования и визуализация
- научная и инженерная графика
- разработка приложений, включая создание графического интерфейса

Рабочая среда *MATLAB* содержит следующие элементы:

- меню;
- панель инструментов с кнопками и раскрывающимся списком;
- окно с вкладками *Launch Pad* и *Workspace*, из которого можно получить простой доступ к различным дополнительным модулям *Toolbox* и к содержимому рабочей среды;
- окно с вкладками *Command History* и *Current Directory*, предназначенное для просмотра и повторного вызова ранее введенных команд, а также для установки текущего каталога;
- командное окно *Command Window*;
- строку состояния.

Настройка окон рабочей среды производится с помощью меню *View и View -> Desktop Layout*

Дополнительные возможности MATLAB

В MATLAB важная роль отводится специализированным группам программ, называемым *toolboxes. Toolboxes* – это всесторонняя коллекция функций *MATLAB*, которые позволяют решать частные технические задачи. *Toolboxes* применяются для обра-

ботки сигналов, анализа изображений, моделирования систем управления и т.д.

Для удобства работы в состав *MATLAB* входит программа *Simulink*, которая позволяет выполнять моделирование систем в графическом виде. *Simulink* содержит библиотеку элементов (*blocksets*) для построения систем из отдельных блоков и позволяет соединять эти блоки друг с другом с помощью мыши.

MATLAB в режиме прямых вычислений

Работа с системой в режиме прямых вычислений носит диалоговый характер и происходит по правилу «задал вопрос – получил ответ». Пользователь набирает на клавиатуре вычисляемое выражение, редактирует его (если нужно) в командном окне и завершает ввод нажатием клавиши *ENTER*.

Примеры:[1]

>>1+2

ans=3

>>*ans*/10

ans=0.08415

>>4*5; %(для блокировки вывода результата вычислений добавьте символ ";" (без кавычек) %в конец выражения)

>>sin(1)

ans=0.8415

Замечание:

✓ Когда выходная переменная не определена, *MATLAB* использует переменную *ans,* коротко от answer - ответ, для хранения результатов вычисления.

✓ Если вводимое математическое выражение окажется настолько длинным, что на него не хватит одной строки, то часть выражения можно перенести на **новую строку с помощью знака многоточия «...»** (3 или более точек).

✓ Текстовый комментарий к выполняемым действиям в *MATLAB* можно ввести после знака %.

Понятие о математическом выражении

Центральным понятием всех математических систем является математическое выражение. Оно задает то, что должно быть вычислено в численном (реже символьном) виде. Но в отличие от других систем, эти выражения в *MATLAB* включают матрицы. Ма-

---

[1] Для усвоения материала данного пособия рекомендуется переносить примеры в рабочую область *MATLAB*, запускать их и анализировать полученные результаты

тематические выражения строятся на основе чисел, констант, переменных, операторов, функций и разных спецзнаков.

Вот примеры простых математических выражений:

2.301*sin(x)
4+exp(3)/5
sqrt(*y*)/2
sin(*pi*/2)

Действительные и комплексные числа

Число – простейший объект языка *MATLAB*, представляющий количественные данные. **Числа можно считать константами, имена которых совпадают с их значениями:**

2
-3

Возможно представление чисел в научном формате с указанием мантиссы и порядка чисел:

2.301
123.456*e*-24
-234.456*e*10

В мантиссе чисел целая часть отделяется от дробной не запятой, а точкой. **Для отделения порядка чисел от мантиссы используется символ *e*. Пробелы между символами в числах не допускаются.**

Числа могут быть комплексными: *z=Re(x) + Im(x)*i*.

3*i*
2*j*
2+3*i*
-3.14*i*
-123.456-3*i*

Константы и системные переменные

Основные системные переменные (задаются системой при ее загрузке и могут переопределяться):

i или *j* – мнимая единица
pi – число "пи"
eps – погрешность операций над числами с плавающей точкой (2^{-52})
realmin – наименьшее число с плавающей точкой (2^{-1022})
realmax – наибольшее число с плавающей точкой (2^{1023})
inf – значение машинной бесконечности
ans – переменная, хранящая результат последней операции
NaN – указание на нечисловой характер данных (*Not-a-Number*)

Бесконечность появляется при делении на нуль или при выполнении математического выражения, приводящего к переполнению, т.е. к превышению *realmax.* Не число *(NaN)* генерируется при вычислении выражений типа 0/0 или *Inf- Inf,* которые не имеют определенного математического значения.

Пример:

>>2**pi*

>>*eps*

>>1/0

Inf

>>0/0

NaN

Переменные и присваивание им значений

{Имя_переменной} = {выражение}

В *MATLAB* нет необходимости в определении типа переменных или размерности. Когда *MATLAB* встречает новое имя переменной, он автоматически создает переменную и выделяет соответствующий объем памяти. Если переменная уже существует, *MATLAB* изменяет ее состав и если это необходимо выделяет дополнительную память. Например, если мы назначим:

a = 25,

система создает матрицу 1x1 с именем *a* и сохраняет значение 25 в ее единственном элементе. Имя переменной может содержать сколько угодно символов, но запоминается и идентифицируется только 31 начальный символ. Имя должно начинаться с буквы, может содержать буквы, цифры и символ подчеркивания. Недопустимо включать в имена переменных пробелы и спец. знаки +, - и т.д.

Уничтожение определений переменных:

who и *whos*– вывести список имеющихся переменных;

clear – уничтожение определений всех переменных;

clear x – уничтожить только *x*;

clear a, b, c – только *a, b* и *c*;

Операторы

Простейшие алгебраические:

+ сложение

- вычитание

* умножение

/, \ деление

^ степень

' комплексно сопряженное транспонирование

() определение порядка вычисления

Отношения:
< - меньше
> - больше
<= - меньше или равно
>= - больше или равно
== - равно
~= - не равно

Функции

MATLAB предоставляет большое количество элементарных математических функций, таких как abs, sqrt, exp, sin. Вычисление квадратного корня или логарифма отрицательного числа не является ошибкой: в этом случае результатом является соответствующее комплексное число. *MATLAB* также предоставляет и более сложные функции, включая Гамма функцию и функции Бесселя. Большинство из этих функций имеют комплексные аргументы. Чтобы вывести список всех элементарных математических функций, наберите:

help elfun

Для вывода более сложных математических и матричных функций, наберите

help specfun

help elmat

Некоторые функции, такие как sqrt и sin, - встроенные. Они являются частью *MATLAB*, поэтому они очень эффективны, но их вычислительные детали трудно доступны. В то время как другие функции, такие как *gamma* и *sink,* реализованы в **виде программного кода в *M*-файлах.** Поэтому вы можете легко увидеть их код и, в случае необходимости, даже модифицировать его.

Форматы вывода результата вычислений

По умолчанию *MATLAB* выдает числовые результаты в нормализованной форме с четырьмя цифрами после десятичной точки и одной до нее. Однко это не всегда удобно, поэтому в *MATLAB* предусмотрена возможность задавать различные форматы представления чисел.

Для установки формата используется команда:

>>*format name*, где *name* – имя формата

short – короткое представление в фиксированном формате (5 знаков)

short e – короткое представление в экспоненциальном формате (5 знаков мантиссы и 3 знака порядка)

long – длинное представление в фиксированном формате (15 знаков)

long e – (15 знаков мантиссы и 3 знака порядка)

hex – представление чисел в шестнадцатеричной форме

bank – представление для денежных единиц

Задание формата сказывается только на форме вывода чисел. Вычисления все равно происходят в формате двойной точности, а ввод возможен в любом удобном для пользователя виде. Альтернативный вариант задания формата представления числовых данных осуществляется через *File -> Preferences -> Command Window -> Numeric Format*.

Основные операции над матрицами

Задание векторов и матриц

Вы можете вводить матрицы в *MATLAB* несколькими способами:

а) вводить полный список элементов

>>*V*=[1 2 3] – задает вектор, имеющий три элемента;

>>*M*=[1 2 3; 4 5 6; 7 8 9]; - задает матрицу 3x3;

>>*V*=[2+2/(3+4) exp(5) sqrt(10)]; - задает значение выражения в качестве элементов матрицы.

```
>> A(1:5,1:5)=0
A =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

```
>> A(end,3:end)=-1
A =
     1     1     1     1     1
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0    -1    -1    -1
```

```
>> A(1,:)=1
A =
     1     1     1     1     1
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

б) генерировать матрицы, используя встроенные функции

Имеется ряд специальных функций для задания векторов и матриц. Например, функция *magic*(*n*) задает магическую матрицу размера nxn, у которой сумма всех столбцов, всех строк и даже диагоналей равна одному и тому же числу:

>>*M*=*magic*(4), а также:

zeros все нули

ones все единицы
rand равномерное распределение случайных элементов
randn нормальное распределение случайных элементов
eye единичная матрица (в диагонали единицы)

>>*Z* = *zeros*(2,4)
>>*F* = 5**ones*(3,3)
>>*N* = 10**rand*(1,10)

в) загружать матрицы из внешних файлов

Команда *load* считывает двоичные файлы, содержащие матрицы, созданные в *MATLAB* ранее, или текстовые файлы, содержащие численные данные. Текстовые файлы должны быть сформированы в виде прямоугольной таблицы чисел, отделенных пробелами, с равным количеством элементов в каждой строке.

Например, создадим вне *MATLAB* текстовой файл, содержащий 4 строки:

16.0	3.0	2.0	13.0
5.0	10.0	11.0	8.0
9.0	6.0	7.0	12.0
4.0	15.0	14.0	1.0

Сохраним этот файл под именем *magik.dat*. Тогда команда *load magik.dat* прочитает этот файл и создаст переменную *magik*, содержащую нашу матрицу.

г) загружать матрицы из *M*-файлов

Вы можете создавать свои собственные матрицы, используя ***M*-файлы, которые представляют собой текстовые файлы, содержащие код *MATLAB***. Просто создайте файл с выражением, которое вы хотите написать в командной строке *MATLAB*. Сохраните его в файле с расширением *.m*.[2]

Например, создадим файл, содержащий строку:

A=[16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1];

Сохраним его под именем *magik.m*. Тогда выражение *magik* прочитает файл и создаст переменную A, содержащую исходную матрицу.

Замечание: Очень часто необходимо произвести формирование упорядоченных числовых последовательностей (число-

[2] Вообще говоря, любую программу, написанную на языке *MATLAB,* удобнее **создавать и сохранять в виде m-файла**. (О работе с m-файлами см. раздел 6.2).

вых массивов - **векторов**). Первоначальное создание таких массивов производится с помощью оператора ":" (без кавычек)

{Начальное_значение}:{Шаг}:{Конечное_значение}

Пример:

>>1:5

>>i=0:2:10

>>x=1:-.2:0

>>x=0:5

Обращение к элементам матрицы

Если мы ввели какую-либо матрицу, то она автоматически запоминается средой *MATLAB* и мы можем к ней легко обратиться, указав имя переменной, которой она присвоена.

Для выбора отдельного элемента, расположенного в строке *i* и столбце *j* матрицы можно использовать индексы *A*(i,j)

>>*A*(2,2)

>>*A*(2,2)=10

Если вы пытаетесь использовать значение элемента вне матрицы, *MATLAB* выдаст ошибку: *t=A*(4,5)

??? *Index exceeds matrix dimensions*.

Найдем сумму элементов в четвертом столбце матрицы A, набрав:

A(1,4) + *A*(2,4) + *A*(3,4) + *A*(4,4) получим

ans = 34

То же самое можно сделать, набрав:

sum(*A*(:,4))

Суммирование элементов, транспонирование и диагонализация матрицы

Подсчитаем сумму элементов всех столбцов матрицы A:

```
sum (A)

  ans  =

34          34    34    34
```

MATLAB предпочитает работать со столбцами матрицы. Для того чтобы получить сумму в строках, необходимо транспонировать матрицу, подсчитать сумму в столбцах, а потом транспонировать результат. Операция транспонирования обозначается апострофом или одинарной кавычкой. Она зеркально отображает матрицу относительно главной диагонали и меняет строки на столбцы.

>>*sum*(*A*')'

Сумму элементов на главной диагонали можно получить с помощью функции *diag,* которая выбирает эту диагональ.

diag(*A*)

ans = 16 10 7 1

И суммированием этих элементов:

sum (*diag* (A))

ans = 34

Замечание: Для доступа к последней строке или столбцу матрицы удобно пользоваться оператором *end*. Так:

sum (*A* (:, *end*)),

вычисляет сумму элементов в последнем столбце матрицы *A*.

Объединение малых матриц в большую

Объединение - это процесс соединения маленьких матриц для создания больших. Пара квадратных скобок - это оператор объединения. Например, начнем с матрицы *A* и сформируем

B = [*A A*+32; *A*+48 *A*+16]1

Результатом будет матрица 8x8, получаемая соединением четырех подматриц

=

16	2	3	13	48	34	35	45
5	11	10	8	37	43	42	40
9	7	6	12	41	39	38	44
4	14	15	1	36	46	47	33
64	50	51	61	32	18	19	29
53	59	58	56	21	27	26	24
57	55	54	60	25	23	22	28
52	62	63	49	20	30	31	17

Удаление столбцов и строк матриц

Вы можете удалять строки и столбцы матрицы, используя просто пару квадратных скобок.

Удалим второй столбец матрицы *A*.

>>*A*(:,2)=[]

Эта операция изменит *A* следующим образом

=

16	**3**	**13**
5	**10**	**8**
9	**6**	**12**

4 15 1

Удалим вторую строку
>>*A*(2,:)=[]

=

16	**3**	**13**
9	**6**	**12**
4	**15**	**1**

Если вы удаляете один элемент матрицы, то результат уже не будет матрицей. Так использование одного индекса удаляет отдельный элемент или последовательность элементов и преобразует оставшиеся элементы в вектор-строку:

A(2:2:10) = []

выдаст результат

A= 16 9 3 6 13 12 1

Матричные и поэлементные вычисления

При работе с массивами и матрицами для выполнения поэлементных операций перед символом операции вводится точка "." (без кавычек) .

Пример:

>> *a*=[1 2 3] % Задаем массив из трех элементов

>>*a***a* % Пытаемся умножить массив сам на себя

Ответ системы – сообщение об ошибке:

??? *Error using ==> mtimes*

Inner matrix dimensions must agree.

Причина ошибки в том, что по умолчанию *MATLAB* пытается произвести матричное умножение.

Ответом на

>>*a*.**a*

Будет

ans = 1 4 9

Выполните задание к разделу 3: (задание 1 на стр. 25).

Графическая визуализация вычислений

MATLAB имеет широкие возможности для графического изображения векторов и матриц, а также для создания комментариев и печати графики. Особое внимание в системе уделено трехмерной графике с функциональной окраской отображаемых фигур, имитацией различных световых эффектов и анимацией.

Построение диаграмм и гистограмм

Наглядным способом представления векторных и матричных данных являются диаграммы и гистограммы. Значение элемента вектора пропорционально высоте столбика диаграммы (в случае столбчатой диаграммы) или площади сектора диаграммы (для круговой). Гистограммы используются для получения информации о распределении данных по заданным интервалам.

Функция ***bar*** отображает вектор в виде столбчатой диаграммы.

bar {*X*,*Y*,*W*}

>> *data* = [1.2 1.7 2.2 2.4 2.5 1.3 1.1 0.5 0.4 0.1];

>> *bar*(*data*)

по горизонтальной оси откладывается номер элемента вектора, а по вертикальной – его значение. Аргументом функции ***bar*** может быть как вектор-строка, так и вектор-столбец.

Для отображения значений элементов векторов в зависимости не от номера, а, например, от времени, если в вектор записаны результаты измерений в некоторые моменты времени, функция ***bar*** вызывается с двумя аргументами:

>> *time* = [0.0 0.1 0.2 0.4 0.5 0.8 1.1 1.3];

>> *data* = [2.85 2.93 2.99 3.26 3.01 2.25 2.09 1.79];

>> *bar*(*time*, *data*) – важно, чтобы размеры data и time совпадали

Выбор ширины столбцов осуществляется заданием третьего аргумента (по умолчанию она равна 0.8). Если ширина равна 1, то диаграмма получается без промежутков между столбцами.

Если требуется оценить вклад каждого из элементов вектора в общую сумму его элементов, то удобно построить круговую диаграмму – ***pie***.

>> *data* = [19.5 13.4 42.6 7.9];

>> *pie*(*data*)

Часто необходимо отодвинуть от круга диаграммы сектор, соответствующий некоторому элементу.

>> *data* = [19.5 13.4 42.6 7.9];

>> *parts* = [0 1 0 0];

>> *pie*(*data*, *parts*) - важно, чтобы размеры *data* и *parts* совпадали

Обработка данных включает вопрос о том, сколько данных попало в тот или иной интервал. Для получения наглядного представления о распределении данных служит функция ***hist***.

```
>> data = randn(100000,1);
>> hist(data)
```

Построение графиков функций

MATLAB позволяет строить графики функций в линейном, логарифмическом и полулогарифмическом масштабах. Общий порядок построения графиков функций:

• Задать аргумент в формате *x*={нач. значение}:{шаг}:{кон. значение}.

• Вычислить функцию, например, *y=f(x)*.

• Построить график функции при помощи функции *plot(x,y,s)*

Построение графиков функций одной переменной в линейном масштабе осуществляется при помощи функции *plot*. В зависимости от входных аргументов функция *plot* позволяет строить один или несколько графиков, изменять цвет и стиль линий и добавлять маркеры на каждый график.

Функция *plot* имеет различные формы, связанные с входными параметрами, например *plot(y)* создает график зависимости элементов ***y* от их индексов**. Если вы задаете два вектора в качестве аргументов, *plot(x,y)* **создаст график зависимости *y* от *x*.** Например, для построения графика значений функции sin от нуля до 2□, сделаем следующее

```
t =  0:pi/100:2*pi;
y =  sin(t) ;
plot(t,y)
```

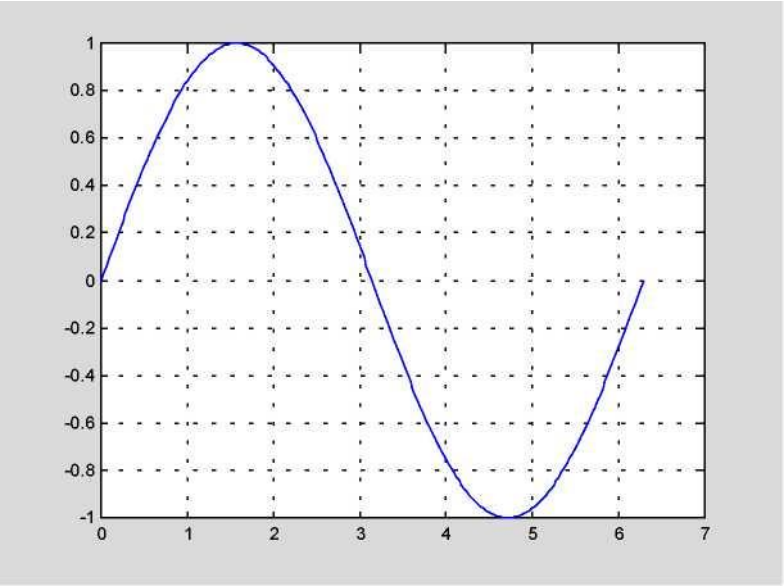


Рис. 1.1. График функции *y*=sin(*x*)

Возможно задание цвета, стиля линий и маркеров графиков при создании графика, с помощью параметра команды *plot*:

plot(*x*, *y*, *'s'*),

где *s* – строковая константа, задающая параметры линии графика:

Таблица 1.1

Цвет линии		Тип точки		Тип линии	
y	Желтый	.	Точка	-	сплошная
m	Фиолетовый	*o*	Кружок	:	двойной пунктир
C	Голубой	*x*	Крест	-.	штрих пунктир
R	Красный	+	Плюс	--	штрих
g	Зеленый	*	звездочка		
b	Синий	*s*	Квадрат		
w	Белый	*d*	Ромб		
K	черный	*v*	треугольник вверх		
		<	треугольник влево		
		>	треугольник вправо		
		p	пятиугольник		
		h	шестиугольник		

Если на одном графике нужно отобразить несколько функций, например, *y*1=*f*(*x*) и *y*2=*f*(*x*), то они вначале вычисляются, а затем выводятся процедурой *plot*(*x*,*y*1,'*s*1',*x*,*y*2,'*s*2'...), в которой в качестве параметров для каждой функции следуют группы <аргумент, функция, тип линии>.

```
y2  = sin(t-.25);
y3  = sin(t-.5);
plot(t,  y, 'b', t,  y2, 'g',  t,  y3, 'r')
```

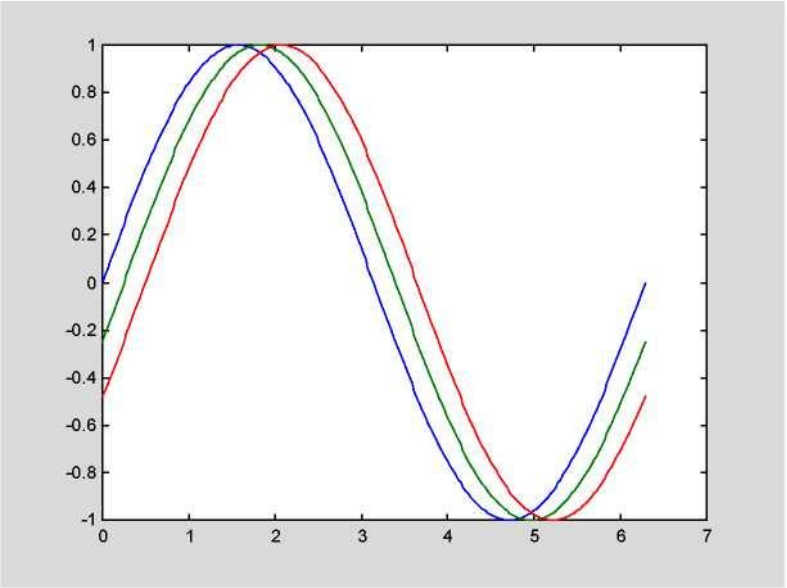


Рис. 1.2. Изображение нескольких функций на одном графике *y*=sin(*x*)

Удобство использования графиков во многом зависит от дополнительных элементов оформления: координатной сетки, подписей к осям, заголовка и легенды. Управление осями

Производится с помощью функции ***axis***. Она имеет несколько возможностей для настройки масштаба, ориентации и коэффициента сжатия.

axis({*xmin xmax ymin ymax*})

Обычно *MATLAB* находит максимальное и минимальное значение и выбирает соответствующий масштаб осей. Функция *axis* заменяет значения по умолчанию предельными значения, вводимыми пользователем. Также можно использовать ключевые слова для управления внешним видом осей. Например:

axis square – создает *x* и *y* оси равной длины,

axis equal – создает отдельные отметки приращений для *x* и *y* осей одинаковой длины.

Тогда, функция: *plot*(*exp*(*i***t*)), следующая либо за *axis square*, либо за *axis equal* превращает овал в правильный круг.

axis auto – возвращает значения по умолчанию и переходит в автоматический режим;

axis on – включает обозначения осей и метки промежуточных делений;

axis off – выключает обозначения осей и метки промежуточных делений;

Замечание: Иногда требуется сравнить поведение двух функций, значения которых сильно отличаются друг от друга. График функции с небольшими значениями практически сливается с осью абсцисс, и установить его вид не удается. В этой ситуации помогает функция ***plotyy***, которая выводит графики в окно с двумя вертикальными осями, имеющими подходящий масштаб.

Сетка наносится командой ***grid on***, подписи к осям размещаются при помощи ***xlabel***, ***ylabel***, заголовок задается командой ***title***. Наличие нескольких графиков на одних осях требует помещения легенды командой ***legend*** с информацией о линиях.

Пример: Следующие команды выводят графики изменения суточной температуры.

```
>> time = [0 4 7 9 10 11 12 13 13.5 14 14.5 15 16 17 18 20 22];
>> temp1 = [14 15 14 16 18 17 20 22 24 28 25 20 16 13 13 14 13];
>> temp2 = [12 13 13 14 16 18 20 20 23 25 25 20 16 12 12 11 10];
>> plot(time, temp1, `ro-`, time, temp2, `go-')
>> grid on
>> title(`Суточные температуры')
>> xlabel(`Время (час.)')
>> ylabel(`Температура (С)')
>> legend('10 мая', '11 мая')
```

Замечание: Дополнительным аргументом legend может быть положение легенды в графическом окне:

-1 – вне графика в правом верхнем углу графического окна;

0 – выбирается лучшее положение в пределах графика так, чтобы как можно меньше перекрывать сами графики;

1 – в верхнем правом углу графика (по умолчанию)

2 – в верхнем левом углу графика

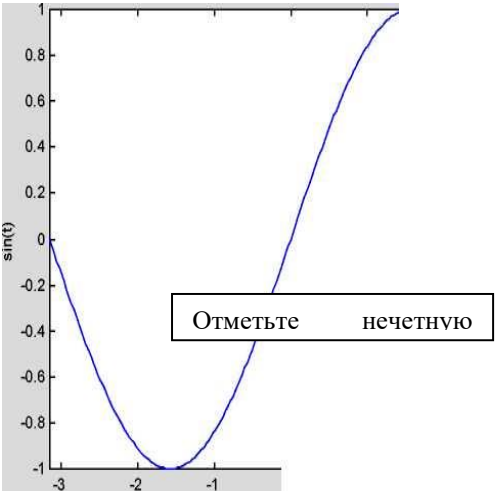
3 – в нижнем левом углу графика

4 – в нижнем левом углу графика.

Использование *TEX*-представления позволяет применять греческие буквы, математические символы и различные шрифты. Следующий пример демонстрирует эту возможность:

```
t  = -pi:pi/100:pi;
y = sin(t) ;
plot(t,y)
axis([-pi pi -1 1])
xlabel( ' -\pi \leq \itt \leq \pi ' )
ylabel( ' sin(t) ' )
```

title(' График функции sin ')
text(-l, -1/3, ' *it*{Отметьте нечетную симметрию} ')



образ 1

Рис. 1.3. График функции *y*=sin(*x*)

Окна изображений

Функция *plot* автоматически открывает новое окно изображения (далее окно), если до этого его не было на экране. Если же оно существует, то *plot* использует его очищая его содержимое.

Для открытия нового окна и выбора его по умолчанию, наберите:

figure

Для того, чтобы сделать существующее окно текущим –

figure(*n*),

где *n* - это номер в заголовке окна. В этом случае результаты всех последующих команд будут выводиться в это окно.

Замечание: Чтобы добавить кривые на существующий график, следует воспользоваться командой *hold on*. Команда ***clf*** очищает все текущее окно. Команда ***cla*** – убирает только график, а оси, заголовок и названия осей оставляет

Подграфики

Функция *subplot* позволяет выводить множество графиков в одном окне или распечатывать их на одном листе бумаги.

subplot({*m*},{*n*},{*p*}) -

Разбивает окно изображений на матрицу m на n подграфиков и выбирает p-ый подграфик текущим. Графики нумеруются вдоль первого в верхней строке, потом во второй и т.д. Например, для того, чтобы представить графические данные в **четырех** разных подобластях окна необходимо выполнить следующее:

```
t = -pi:pi/100:pi;
x = cos(t) ;
y = sin(t) ;
z = 1/sin(t);
k = 1/cos(t) ;

subplot(2,2,1); plot(t,x)
subplot(2,2,2); plot(t,y)
subplot(2,2,3); plot(t,z)
subplot(2,2,4); plot(t,k)
```

Выполните задание к разделу 4: (задание 2 на стр. 25).

Основы трехмерной графики

MATLAB предоставляет различные способы визуализации функций двух переменных – построение трехмерных графиков и линий уровня, параметрически заданных линий и поверхностей.

Для формирования трехмерных графиков необходимо:

– Сгенерировать матрицы с координатами узлов сетки на прямоугольной области определения функции.

– Вычислить функцию в узлах сетки и записать полученные значения в матрицу.

– Использовать одну из графических функций MATLAB.

Генерация сетки

Сетка генерируется при помощи команды ***meshgrid***, вызываемой с двумя аргументами, векторами, элементы которых соответствуют сетке прямоугольной области построения функции. Можно использовать один аргумент, если область построения функции – квадрат.

[*X,Y*] = *meshgrid*(*x,y*) – преобразует область, заданную векторами *x,y* в массивы *X* и *Y*, которые могут быть использованы для вычисления функции двух переменных и построения трехмерных графиков.

Пример:

```
>> [X,Y] = meshgrid(-1:0.5:1, 0:0.5:1)
>> [X,*Y] = meshgrid(-1:0.05:1, 0:0.05:1);
```

[*X,Y*] = *meshgrid*(*x*) – аналогична [*X,Y*] = *meshgrid*(*x,x*)

Трехмерный график с аксонометрией

Команда *plot3*(…) является аналогом команды *plot*(…), но относится к функциям двух переменных *z*(*x,y*). Она строит аксонометрическое изображение трехмерных поверхностей.

plot3(*x,y,z*) – строит массив точек, представленных векторами *x, y* и *z*, соединяя их отрезками прямых.

Пример:

>> [*X,Y*] = *meshgrid*(-3:0.15:3);
>> *Z*=*X*.^2+*Y*.^2;
>> *plot3*(*X,Y,Z*)

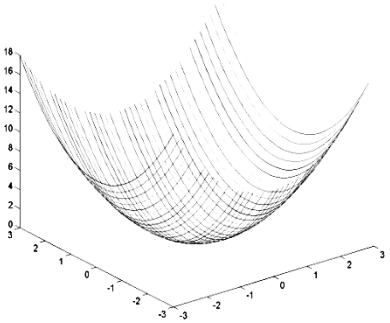


Рис. 1.4. Трехмерный график с аксонометрией

Задание свойств линий и маркеров для *plot3*() производится также как и для *plot*()

plot3(*X,Y,Z,S*)
>> *plot3*(*X,Y,Z,'-o'*)
plot3(*x1,y1,z1,s1, x2,y2,z2,s2,…*)
>> *plot3*(*X,Y,Z,'-k',Y,X,Z,'-k'*)

Наиболее представительными и наглядными являются сетчатые графики поверхностей с заданной или функциональной окраской. В названии их команд присутствует слово *mesh*. Имеется три группы таких команд:

Сетчатый 3D-график с функциональной окраской

mesh(*X,Y,Z,C*) – выводит в окно сетчатую поверхность *Z*(*X,Y*) с цветами узлов поверхности, заданными массивом *C*.

mesh(*X,Y,Z*) – аналог предшествующей команды при *C=Z*.

Пример: Построить график функции двух переменных
z(*x,y*) = 4□sin2□*x*□cos1.5□*y*□(1-*x*2)□*y*□(1-*y*)
на прямоугольной области определения *x*□[-1,1], *y*□[0,1].
>> [*X,Y*] = *meshgrid*(-1:0.05:1, 0:0.05:1);

```
>> Z = 4*sin(2*pi*X).*cos(1.5*pi*Y).*(1-X.^2).*Y.*(1-Y);
>> mesh(X,Y,Z)
```

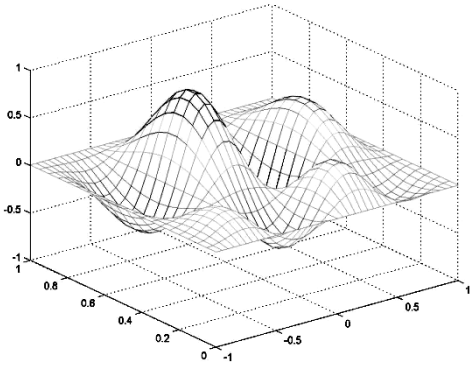


Рис. 1.5. Сетчатый 3D-график с функциональной окраской

Цвет линий соответствует значениям функции. *MATLAB* рисует только видимую часть поверхности. При помощи команды *hidden off* можно сделать каркасную модель «прозрачной», добавив скрытую часть. *MATLAB* имеет несколько функций, возвращающих матричный образ поверхностей. Например, функция *peaks*(*N*) возвращает матричный образ поверхности с рядом пиков и впадин. Такие функции удобно использовать для проверки команд трехмерной графики.

Пример:

```
>> z = peaks(25);
>> mesh(z)
```

Сетчатый 3*D*-график с функциональной окраской и проекцией

Иногда график поверхности полезно объединить с контурным графиком ее проекции на плоскость, расположенным под поверхностью.

meshc(…)

```
>> [X,Y] = meshgrid([-3:0.15:3]);
>> Z=X.^2+Y.^2;
>> meshc(X,Y,Z)
```

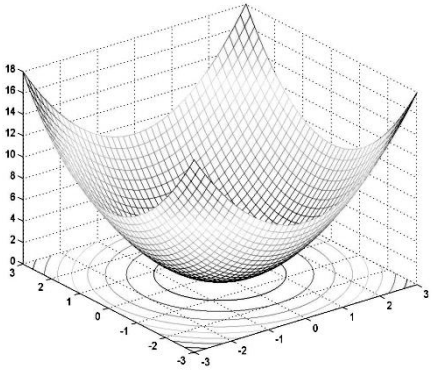


Рис. 1.6. Сетчатый 3D-график с функциональной окраской и проекцией

График такого типа дает лучшее представление об особенностях поверхности.

Поверхностный 3*D*-график с функциональной окраской

Наглядное представление о поверхностях дают графики, использующие функциональную закраску ячеек. Например, цвет окраски поверхности *z*(*x*,*y*) может быть поставлен в соответствие с высотой *z* поверхности с выбором для малых высот темных тонов, а для больших – светлых.

Для построения таких поверхностей используются команды класса *surf*(…).

surf(*X*,*Y*,*Z*,*C*) – строит цветную параметрическую поверхность по данным матриц *X*,*Y* и *Z* с цветом, задаваемым массивом *C*.

surf(*X*,*Y*,*Z*) – аналогична предшествующей команде, где *C*=*Z*

Пример

```
>> [X,Y] = meshgrid([-3:0.15:3]);
>> Z=X.^2+Y.^2;
>> surf(X,Y,Z)
```

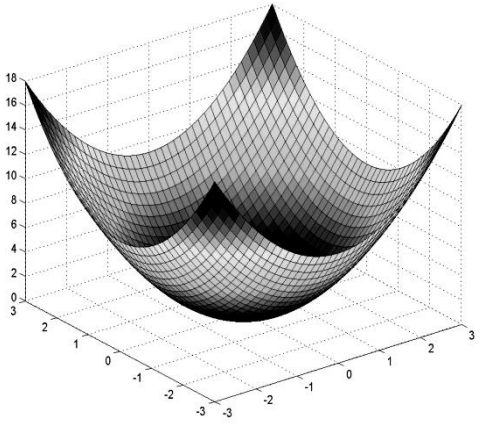


Рис. 1.7. Поверхностный 3D-график с функциональной окраской

>> *colormap*('') – задает окраску тонами определенного цвета

>> *shading interp* – обеспечивает устранение изображения сетки и задает интерполяцию для оттенков цвета объемной поверхности

Применение интерполяции для окраски придает поверхностям и фигурам более реалистичный вид, но фигуры каркасного вида дают более точные количественные данные о каждой точке.

Поверхностный 3*D*-график с функциональной окраской и освещением

Пожалуй, наиболее реалистичный вид имеют графики поверхностей, в которых имитируется освещение от точечного источника света, расположенного в заданном месте координатной системы. Графики имитируют оптические эффекты рассеивания, отражения и зеркального отражения света. Для получения таких эффектов используется команда *surfl*.

surfl(*Z,S*) или *surfl*(*X,Y,Z,S*) – строит графики поверхности с подсветкой от источника света, положение которого в системе декартовых координат задается вектором *S*=[*Sx,Sy,Sz*], а в системе сферических координат – вектором *S*=[*AZ,EL*];

При использовании *surfl* удобно задавать цветовые палитры: *copper, bone, gray, pink,* в которых интенсивность цвета изменяется линейно.

Пример.

>> [*X,Y*] = *meshgrid*(-1:0.05:1, 0:0.05:1);

>> *Z* = 4*sin(2**pi***X*).*cos(1.5**pi***Y*).*(1-*X*.^2).**Y*.*(1-*Y*);

>> *surfl*(*X,Y,Z*)

>> *colormap*(`*copper*')
>> *shading interp*

По умолчанию источник света имеет азимут, больший на 45□, чем наблюдатель, и тот же угол возвышения. Возможно изменение азимута источника. Например, изменим азимут до -90□ по отношению к наблюдателю, а угол возвышения до нуля.

>> [*Az, El*] = *view*;
>> *surfl*(*X,Y,Z*,[*Az*-90,0])
>> *shading interp*

Азимут отсчитывается от оси, противоположной оси *y*, а угол возвышения - от плоскости *xy*.

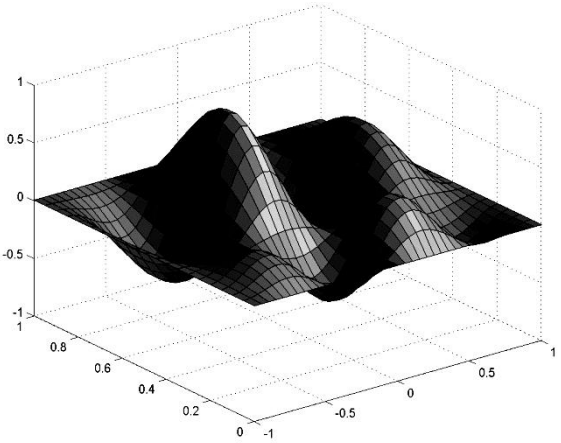


Рис. 1.8. Поверхностный 3D-график с функциональной окраской и освещение.

Выполните задание к разделу 5: (задание 3 на стр. 26).

Программирование в среде MATLAB

Управление потоками

MATLAB имеет пять видов структур управления потоками:

оператор *if*
оператор *switch*
циклы *for*
циклы *while*
оператор *break*

В том случае, когда некоторая группа операций должна повторно выполняться несколько раз, используют операторы цикла.

а) оператор цикла с неизвестным числом повторений (*while*)

while выражение
<оператор>
<оператор>
end

Операторы, указанные в теле цикла, выполняются, пока выражение истинно.

Пример:

```
fact=1;
k=1;
while fact<1e5
fact=fact*k;
k=k+1;
end
```

б) оператор цикла с известным числом повторений (*for*)

for переменная цикла = начальное значение : <приращение> : конечное значение
<оператор>
<оператор>
end

Операторы, указанные в теле цикла, выполняются заданное число раз. При этом переменная цикла последовательно принимает все значения от начального до конечного с приращением после каждого прохода цикла.

```
for i=1:1:100
s=s+5
end
```

Замечание: Хорошим стилем являются отступы при использовании циклов для лучшей читаемости, особенно, когда они вложенные.

```
for i = 1:m
   for j = 1:n
      H(i,j) = 1/(i+j);
            end ;
end
```

в) оператор условного ветвления

if <логическое выражение> <операторы> *end*	*if* <логическое выражение> <операторы> *elseif* <логическое выражение> <операторы> *else* <операторы> *end*
if <логическое выражение> <операторы> *else* <операторы> *end*	

Оператор *else* не содержит логического условия. Инструкции, связанные с ним, выполняются, если предшествующий оператор *if* ложен.

Оператор *elseif* содержит логическое условие, которое вычисляется, если предшествующий оператор *if* ложен. Оператор *elseif* может многократно использоваться внутри оператора условия *if*.

Пример:

if n<0

a=2**n*;

elseif rem(*n*,2)==0;

a=*n*/2;

else

a=(*n*+1)/2; Если *n*>0 *b* нечетное, увеличить на 1 и разделить.

end

г) оператор переключения

switch…case…otherwise…end

Оператор *switch* работает, сравнивая значение вычисленного выражения со значениями групп *case*. Если значение выражения в операторе *switch* совпадает со значением, указанным в *case*, то выполняются соответствующая группа операций до следующего оператора *case* или *otherwise*, или *end*. Оператор *otherwise* выполняется, если <выражение> не совпало ни с одним из значений.

Пример:

switch input_num % выч. выражение

```
case –1                    % значение
disp ('минус один')        % выводит строку text в рабочее окно MATLAB
case 0
disp ('нуль')
case 1
disp ('плюс один')
otherwise
disp ('другое значение')
end
```

д) оператор досрочного выхода из цикла

```
break
```

Оператор *break* позволяет досрочно выходить из циклов *for* или *while.* Во вложенных циклах *break* осуществляет выход только из самого внутреннего цикла.

```
for i=1:1:100
s=s+5
if s>300 then
      break;
end;
end
```

1.6.1 Работа с M-файлами

Если требуется выполнять большую последовательность команд или повторять группу команд для различных значений входных переменных, работа в режиме командной строки становится неудобной. Для облегчения работы *MATLAB* предоставляет возможность организации вычислений в виде так называемых *M*-файлов. *M*-файлы являются обычными текстовыми файлами, которые содержат последовательности команд, операторов, функций и т.д., необходимых для вычислений. В состав системы *MATLAB* входит редактор/отладчик *M*-файлов. Наиболее удобно использовать инструментальную панель командного окна этого редактора/отладчика.

Для создания м-файла в меню *File* основного окна *MatLab* и в пункте *New* выберите подпункт *M-file*. Новый файл открывается в редакторе *M*-файлов. Наберите в редакторе команды, приводящие к построению двух графиков в одном графическом окне.

```
x = [0:0.1:7];
f = exp(-x);
subplot(1, 2, 1)
plot(x, f)
g = sin(x);
```

subplot(1, 2, 2)
plot(*x*, *g*)

Сохраните теперь файл с именем *mydemo.m* в подкаталоге *work* основного каталога *MatLab*, выбрав пункт *Save as* меню *File* редактора. Для запуска на выполнение всех команд, содержащихся в файле, следует выбрать пункт *Run* в меню *Debug*. На экране появится соответствующее графическое окно. Созданный *M*-файл можно как угодно редактировать и повторять все команды заново. При этом графическое окно будет отражать все изменения, которые вы сделали в файле. Для выполнения только части команд из файла, следует их сначала выделить, а затем выполнить из пункта *Evaluate selection* меню *Text*.

В редакторе *M*-файлов может быть открыто одновременно несколько файлов; переход от одного к другому осуществляется при помощи закладок с их именами внизу окна редактора. Открытие существующего *M*-файла производится из пункта *Open* меню *File* рабочей среды, либо редактора *M*-файлов.

Типы М-файлов. Файл-программы и файл-функции

M-файлы бывают двух типов: файл-программы (*Script M-files*), содержащие последовательность команд, и файл-функции (*Function M-files*), в которых описываются функции, определяемые пользователем для своих целей. Файл mydemo, который Вы уже создали, это файл-программа. Все переменные, объявленные в ней, становятся доступными в рабочей среде после ее выполнения. Фактически, созданный *M*-файл является новой командой, которую понимает *MatLab*. Теперь в командной строке достаточно набрать команду *mydemo* и появится графическое окно, соответствующее командам программы *mydemo.m*.

Как правило, *M*-файлы хранятся в каталоге пользователя. Чтобы *MatLab* мог найти их, следует установить пути, указывающее расположение *M*-файла.

Файл-программы являются последовательностью команд и не имеют входных и выходных аргументов. Для использования численных методов и при программировании собственных приложений необходимо уметь составлять файл-функции, которые выполняют действия с входными аргументами и возвращают результат в выходных аргументах.

Имеет смысл один раз написать файл-функцию, чтобы потом вызывать ее всюду, где необходимо.

Откройте в редакторе *M*-файлов новый файл и наберите текст

function f = myfun(x)
f = exp(-*x*)*sqrt((*x*^2+1)/(*x*^4+0.1));

Слово *function* в первой строке означает, что данный файл содержит файл-функцию. Первая строка является заголовком файл-функции, где размещается имя функции *myfun*, один входной аргумент *x* и один выходной – *f*. После заголовка следует тело функции (здесь это одна строка), где и вычисляется ее значение. Теперь сохраните файл в рабочем каталоге (укажите полностью Ваш путь). При сохранении появляется диалоговое окно с уже готовым именем *myfun*. Не изменяйте его.

Теперь созданную функцию можно использовать так же, как и встроенные функции, из командной строки:

>> *y*=*myfun*(1.3)
y = 0.2600

Задания к лабораторной работе № 1

Задание 1

Таблица 1.2

1.Создать матрицу *A*							
Вариант1	Вариант2	Вариант3	Вариант4	Вариант5	Вариант6	Вариант7	Вариант8
-1 2 3 5 7 8 9 1 4 5 6 4 11 1 5 9	8 4 6 9 2 7 4 10 3 8 2 6 -5 3 7 3	8 9 13 2 -6 6 4 5 8 4 6 4 13 1 8 9	8 6 7 5 7 3 7 15 4 8 9 2 8 9 8 5	1 -2 3 5 -7 8 9 1 11 3 2 4 1 1 4 -9	1 5 3 2 6 7 3 6 3 4 6 7 1 5 8 2	-1 2 3 5 3 8 9 1 5 3 6 4 7 7 5 1	1 5 6 4 2 8 9 1 4 8 6 4 4 5 5 -1
Вариант9	Вариант10	Вариант11	Вариант12	Вариант13	Вариант14	Вариант15	Вариант16
-1 4 8 5 7 7 9 2 4 8 6 1 1 1 4 0	-4 4 6 9 2 5 4 10 3 8 -6 6 8 3 7 2	5 4 3 2 -4 6 -4 5 8 5 6 2 3 1 8 91	8 6 7 51 0 7 -3 15 6 8 11 2 8 9 14 5	1 2 -3 15 5 8 -9 1 1 5 2 6 1 11 -5 -7	1 4 13 2 6 7 -4 6 3 -2 6 9 1 15 4 2	-5 2 -5 5 3 2 3 1 1 3 3 4 0 -2 0 1	1 0 6 4 2 5 9 -11 5 9 0 4 6 5 5 -1
Вариант17	Вариант18	Вариант19	Вариант20	Вариант21	Вариант22	Вариант23	Вариант24
0 9 13 2 -6 0 4 3 1 4 2 4 1 1 0 1	1 -3 3 5 -7 8 -6 1 11 0 0 4 0 1 4 -9	-3 21 5 -2 -3 1 3 1 -1 3 2 4 1 0 1 0	-4 5 1 9 2 5 0 10 1 8 -4 6 1 3 1 3	-1 3 6 9 2 5 3 10 5 12 2 6 1 1 1 1	7 5 2 4 2 7 9 17 -4 1 -5 7 1 4 2 1	8 6 7 51 0 7 -3 15 6 8 11 2 8 9 14 5	-1 4 8 5 7 7 9 2 4 8 6 1 1 1 4 0

2. Вычислить сумму элементов матрицы *A* по строкам, по столбцам, а также сумму всех элементов матрицы *A*.	
3. Транспонировать матрицу *A*.	
4.Умножить матрицу *A* на 2.	
5. Найти квадратный корень из элементов матрицы *A*.	
6 .Возвести в квадрат все элементы матрицы *A*.	
7.Создать массив *X*, содержащий 2 строки:	
Вариант 1	1 строка - числа от 1 до 100, в порядке возрастания 2 строка - числа от 1 до 100, в порядке убывания
2	1 строка - нечетные числа от 1 до 100, в порядке возрастания 2 строка – четные числа от 1 до 100, в порядке возрастания
3	1 строка - нечетные числа от 1 до 100, в порядке возрастания 2 строка - нечетные числа от 1 до 100, в порядке убывания
4	1 строка - числа от 1 до 100, кратные 3-м, в порядке возраста-

	ния 2 строка - числа от 1 до 100, кратные 3-м, в порядке убывания
5	1 строка - числа от 1 до 100, кратные 4-м, в порядке возрастания 2 строка - числа от 1 до 100, кратные 4-м, в порядке убывания
6	1 строка - числа от 1 до 100, кратные 5-и, в порядке возрастания 2 строка - числа от 1 до 100, кратные 5-и, в порядке убывания
7	1 строка - числа от 1 до 100, кратные 6-и, в порядке возрастания 2 строка - числа от 1 до 100, кратные 6-и, в порядке убывания
8	1 строка - числа от 1 до 100, кратные 7-и, в порядке возрастания 2 строка - числа от 1 до 100, кратные 7-и, в порядке убывания
9	1 строка - нечетные числа от 1 до 100, в порядке убывания 2 строка – четные числа от 1 до 100, в порядке возрастания
10	1 строка - четные числа от 1 до 100, в порядке убывания 2 строка – нечетные числа от 1 до 100, в порядке убывания
11	1 строка - числа от 1 до 100, кратные 3-м, в порядке убывания 2 строка - числа от 1 до 100, кратные 3-м, в порядке возрастания
12	1 строка - числа от 1 до 100, в порядке убывания 2 строка - числа от 1 до 100, в порядке возрастания
13	1 строка - числа от 1 до 100, кратные 6-и, в порядке убывания 2 строка - числа от 1 до 100, кратные 6-и, в порядке возрастания
14	1 строка - числа от 1 до 100, кратные 7-и, в порядке убывания 2 строка - числа от 1 до 100, кратные 7-и, в порядке убывания
15	1 строка - числа от 1 до 100, кратные 7-и, в порядке убывания 2 строка - числа от 1 до 100, кратные 7-и, в порядке возрастания
16	1 строка - числа от 1 до 100, кратные 7-и, в порядке возрастания 2 строка - числа от 1 до 100, кратные 7-и, в порядке возрастания
17	1 строка - числа от 1 до 100, кратные 4-м, в порядке возрастания 2 строка - числа от 1 до 100, кратные 4-м, в порядке возрастания
18	1 строка - числа от 1 до 100, кратные 4-м, в порядке убывания 2 строка - числа от 1 до 100, кратные 4-м, в порядке убывания
19	1 строка - числа от 1 до 100, кратные 4-м, в порядке убывания 2 строка - числа от 1 до 100, кратные 4-м, в порядке возрастания
20	1 строка - четные числа от 1 до 100, в порядке убывания 2 строка – четные числа от 1 до 100, в порядке убывания

21	1 строка - числа от 1 до 100, кратные 3-м, в порядке возрастания 2 строка - числа от 1 до 100, кратные 3-м, в порядке возрастания
22	1 строка - числа от 1 до 100, в порядке убывания 2 строка - числа от 1 до 100, в порядке убывания
8. Создать m-файл, описывающий все произведенные действия.	

Задание 2. Вычислить значения функций 1 и 2 (табл. 1.3) для аргумента в заданном интервале [*a b*] с шагом *h*. Вывести графики функций одновременно на одном графике в декартовых координатах. Для разных графиков использовать разный тип линий.

Таблица 1.3

№	Функция 1	Функция 2	*a*	*B*	*H*
1	$y = \sin(x)$	$z = \exp(x+3)/5000 - 1$	-2□	2□	□/20
2	$y = \cos(x)$	$z = 0.00025e^{3}-x - 0.6$	-2□	2□	□/20
3	$y = \lvert tg(x)\rvert + 0.1$	$z = (1+x)^{6}$	-2□	2□	□/20
4	$y = (x^{2}-1)/15$	$z = 1+\sin(x)$	-2□	2□	□/20
5	$y = (x^{3}-2)/15$	$z = 5\cos(x)$	-2□	2□	□/20
6	$y = x^{2} - 10$	$z = 0.025\exp(-1.2x)$	-5	5	1
7	$y = 3\sin(x)$	$z=0.015x^{3}$	-5	5	1
8	$y = \lvert\sin(x)\rvert\exp(x/2)$	$z = 5x - x^{1.5}+\sin(x)$	0	5	0.5
9	$y = 0.5x^{3}$	$z = -2\cos(x)$			
10	$y = 2-\sin(x)$	$z = \cos(x)$	-2□	2□	□/20
11	$y = \exp(2x+1)$	$z = \sin(x)$	-2□	2□	□/20
12	$y = (x^{3}-2)/15$	$z = x^{2} - 1$	-2□	2□	□/20
13	$y = 6\cos(x)$	$z = 0.5x^{2}$	-2□	2□	□/20
14	$y = 5x^{2}$	$z = (x^{3}-3)/11$	-2□	2□	□/20
15	$y = \exp(-1.7x)$	$z = \exp(-1.6x)$	-5	5	1
16	$y = 5\cos(x)$	$z = 7\sin(x)$	-5	5	1
17	$y = 1+\cos(x)$	$z = (x^{3}-6)/13$	-5	5	1
18	$y = (x^{3}-3)/10$	$z = 7x^{3}$	-2□	2□	□/20
19	$y = 1+\sin(x)$	$z = \exp(-1.4x)$	-2□	2□	□/20
20	$y = 8\cos(x)$	$z = (x^{2}-8)/33$	-2□	2□	□/20
21	$y = 6x^{2}+5x^{2}$	$z = (x^{3}-2)/4$	-5	5	1
22	$y = 6\cos(x)+\sin(x)$	$y = x^{2}$	-2□	2□	□/20

Задание 3. Вычислить значения функции двух аргументов (табл. 1.4) в заданном диапазоне. Вывести функцию в виде 5 трехмерных графиков разного типа. Вывести функцию в виде 2 контурных графиков разного типа.

Таблица 1.4

№	Функция	Пределы изменения	
		x	y
1	$z=\sin(x)\cos(y)$	от -2□ до 2□	от -2□ до 2□
2	$z=\sin(x/2)\cos(y)$	от -2□ до 2□	от -2□ до 2□
3	$z=\sin(2x)\cos(y)$	от -2□ до 2□	от -2□ до 2□
4	$z = \sin(x)\cos(y/2)$	от -2□ до 2□	от -2□ до 2□
5	$z = \sin(x/2)\cos(2y)$	от -2□ до 2□	от -2□ до 2□
6	$z = \sin(2x)\cos(2y)$	от -2□ до 2□	от -2□ до 2□
7	$z = (1+\sin(x)/x)(\sin(y)/y)$	от -2□ до 2□	от -2□ до 2□
8	$z = (\sin(x)/x)\cos(y)$	от -2□ до 2□	от -2□ до 2□
9	$z = \sin(3x)\cos(y/2)$	от -2□ до 2□	от -2□ до 2□
10	$z = (\sin(3x)+3)\cos(2y)$	от -2□ до 2□	от -2□ до 2□
11	$z = \sin(x/3)\cos(4y)$	от -2□ до 2□	от -2□ до 2□
12	$z = (4+\sin(2x)/x)(\sin(4y)/y)$	от -2□ до 2□	от -2□ до 2□
13	$z=\sin(2x)\cos(3y)$	от -2□ до 2□	от -2□ до 2□
14	$z = (-1+\sin(x)/x)(\sin(y)/3)$	от -2□ до 2□	от -2□ до 2□
15	$z = \sin(x)\cos(4y)$	от -2□ до 2□	от -2□ до 2□
16	$z = (3+\sin(x)/3)(\sin(2y)/y)$	от -2□ до 2□	от -2□ до 2□
17	$z = (1+\sin(3x)/2)(1+\sin(2y)/y)$	от -2□ до 2□	от -2□ до 2□
18	$z=\sin(5x)\cos(1.5y)$	от -2□ до 2□	от -2□ до 2□
19	$z = \sin(5x)\cos(2y)$	от -2□ до 2□	от -2□ до 2□
20	$z = \sin(3x)\cos(3y)$	от -2□ до 2□	от -2□ до 2□
21	$z = \sin(x)\cos(7y)$	от -2□ до 2□	от -2□ до 2□
22	$z = \sin(9x)\cos(4y)$	от -2□ до 2□	от -2□ до 2□

ЧАСТЬ 3. ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

ПРАКТИЧЕСКИЕ ЗАНЯТИЯ

Пример 1. Найти хромосому с максимальным количеством единиц. Допустим, что хромосомы состоят из 12 генов, а популяция насчитывает 8 хромосом. Понятно, что наилучшей будет хромосома, состоящая из 12 единиц.

Выбор исходной популяции хромосом. Необходимо случайным образом сгенерировать 8 двоичных последовательностей длиной12 битов. Это можно достигнуть, например, подбрасыванием монеты (96 раз, при выпадании «орла» приписывается значение 1, а в случае «решки» - 0). Таким образом можно сформировать исходную популяцию:

ch_1 = [111001100101] ch_5 = [010001100100]
ch_2 = [001100111010] ch_6 = [010011000101]
ch_3 = [011101110011] ch_7 = [101011011011]
ch_4 = [001000101000] ch_8 = [000010111100]

Оценка приспособленности хромосом к популяции. Функция приспособленности определяет количество единиц в хромосоме. Ее значения для каждой хромосомы из исходной популяции:

$F(ch_1) = 7$ $F(ch_5) = 4$
$F(ch_2) = 6$ $F(ch_6) = 5$
$F(ch_3) = 8$ **$F(ch_7) = 8$**
$F(ch_4) = 3$ $F(ch_8) = 5$

Хромосомы ch_3 и ch_7 характеризуются наибольшими значениями функции принадлежности. Они считаются наилучшими кандидатами на решение задачи. Если условие остановки алгоритма не выполняется, то на следующем шаге производится селекция хромосом из ткущей популяции.

Селекция хромосом. Селекция производится по методу рулетки. Для каждой из 8 хромосом текущей популяции получаем сектора рулетки, выраженные в процентах (рис. 4):

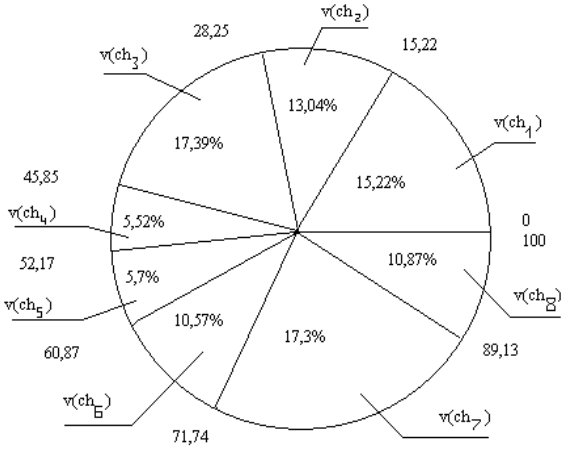


Рис. 4. Принцип работы рулетки.

$V(ch_1) = 15,22$ $V(ch_5) = 8,7$
$V(ch_2) = 13,04$ $V(ch_6) = 10,87$
$V(ch_3) = 17,39$ $V(ch_7) = 17,39$
$V(ch_4) = 6,52$ $V(ch_5) = 10,87$

Розыгрыш с помощью колеса рулетки сводится к случайному выбору числа из интервала [0, 100], указывающего на соответствующий сектор на колесе, т.е. на конкретную хромосому. Допустим, что разыграны следующие 8 чисел:

70 44 9 74 44 86 48 23

Это означает выбор хромосом

ch_7 ch_3 ch_1 ch_7 ch_3 ch_7 ch_4 ch_2

Как видно, хромосома ch_7 была выбрана трижды, а хромосома ch_3 – дважды. Именно эти хромосомы имеют наибольшее значение функции приспособленности. Однако выбрана хромосома ch_4 с наименьшим значением функции приспособленности. Все выбранные таким образом хромосомы включаются в так называемый родительский пул.

Применение генетических операторов. Допустим, что ни одна из отобранных в процессе селекции хромосом не подвергается мутации, и все они составляют популяцию хромосом, предназначенных для скрещивания. Это означает, что вероятность скрещивания $p_c = 1$, а вероятность мутации $p_м = 0$. Допустим, что из этих хромосом случайным образом сформированы пары родителей:

ch_2 и ch_7 ch_1 и ch_7 ch_3 и ch_4 ch_3 и ch_7

Для первой пары случайным образом выбрана точка скрещивания $l_k = 4$, для второй

$l_k = 3$, для третьей $l_k = 11$, для четвертой $l_k = 5$. При этом процесс скрещивания протекает так, как показано на рис. . В результате выполнения оператора скрещивания получается 4 пары потомков.

Если бы при случайном подборе хромосом для скрещивания были объединены, например, ch_3 с ch_3 и ch_4 с ch_7 вместо ch_3 с ch_4 и ch_3 с ch_7, а другие пары остались без изменения, то скрещивание ch_3 с ch_3 дало бы две такие же хромосомы независимо от разыгранной точки скрещивания.. Это означало бы получение двух потомков, идентичных своим родителям. Заметим, что такая ситуация вероятна для хромосом с наибольшим значением функции приспособленности, т.е. именно такие хромосомы получают наибольшие шансы на переход в новую популяцию.

ПЕРВАЯ ПАРА РОДИТЕЛЕЙ		ПЕРВАЯ ПАРА ПОТОМКОВ
[001100111010]		[001111011011]
	Скрещивание	
[101011011011]	⟶	[101000111010]
ВТОРАЯ ПАРА РОДИТЕЛЕЙ		ВТОРАЯ ПАРА ПОТОМКОВ
[111001100101]		[111011011011]
	Скрещивание	
[101011011011]	⟶	[101001100101]
ТРЕТЬЯ ПАРА РОДИТЕЛЕЙ		ТРЕТЬЯ ПАРА ПОТОМКОВ
[011101110011]		[011101110010]
	Скрещивание	
[001000101000]	⟶	[001000101001]
ЧЕТВЕРТАЯ ПАРА РОДИТЕЛЕЙ		ЧЕТВЕРТАЯ ПАРА ПОТОМКОВ
[011101110011]		[011101011011]
	Скрещивание	
[101011011011]	⟶	[101011110011]

Рис.5. Работа ГА.

Формирование новой популяции. После выполнения операции скрещивания мы получаем (согласно рис.) следующую популяцию потомков:

Ch_1 = [001111011011] Ch_5 = [011101110010]
Ch_2 = [101000111010] Ch_6 = [001000101001]
Ch_3 = [111011011011] Ch_7 = [011101011011]
Ch_4 = [101001101101] Ch_8 = [101011110011]

Для отличия от хромосом предыдущей популяции обозначения вновь сформированных хромосом начинаются с большой буквы С.

Согласно блок-схеме генетического алгоритма производится возврат ко второму этапу, т.е. к оценке приспособленности

хромосом из вновь сформированной популяции, которая становится текущей. Значения функций приспособленности этой популяции составляют

$F(Ch_1) = 8$ $F(Ch_5) = 7$
$F(Ch_2) = 6$ $F(Ch_6) = 4$
$F(Ch_3) = 9$ $F(Ch_7) = 8$
$F(Ch_4) = 6$ $F(Ch_8) = 8$

Заметно, что популяция характеризуется более высоким средним значением функции приспособленности, чем популяция родителей. В результате скрещивания получена хромосома Ch_3 с наибольшим значением функции приспособленности, которым не обладала ни одна хромосома из родительской популяции. Однако могло произойти и обратное, поскольку после скрещивания на первой итерации хромосома, которая в родительской популяции характеризовалась наибольшим значением функции приспособленности, могла просто «потеряться». Помимо этого «средняя» приспособленность новой популяции все равно оказалась бы выше предыдущей, а хромосомы с большими значениями функции приспособленности имели бы шансы появиться в следующих поколениях.

Пример 2. Найти максимум функции $f(x) = 2x^2+1$ для x = 0…31.

Для применения ГА необходимо прежде всего закодировать в виде двоичных последовательностей значения x. Очевидно, что числа 0…31 можно представить в двоичной системе счисления. При этом) кодируется как 00000, а 31 – 11111, т.е. это цепочка из 5 битов.. Выберем случайным образом популяцию, состоящую из 6 кодовых последовательностей N=6 (30 раз подбрасываем монету). Допустим, что выбраны хромосомы:

ch_1 = 10011 ch_4= 10101
ch_2= 00011 ch_5= 01000
ch_3= 00111 ch_6 = 11101

Соответствующие им фенотипы – это числа из интервала 0…31.

ch_1 = 19 ch_4= 21
ch_2= 3 ch_5= 8
ch_3= 7 ch_6 = 29

По вышеприведенной формуле рассчитываем значения функции приспособленности

Fch_1 = 723 Fch_4= 883
Fch_2= 19 F ch_5= 129
Fch_3= 99 Fch_6 = 1683 Среднее значение F =589, сумма – 3536

Рассчитываем

$$p_s(ch_i) = \frac{F(ch_i)}{\sum_{i-1}^{N} F(ch_i)} 100\%$$

Pch_1 = 20,45 Pch_4= 24,97
Pch_2= 0,54 P ch_5= 3,65
Pch_3= 2,8 Pch_6 = 47,6

Методом рулетки выбираем 6 хромосом длz репродукции (вращаем рулетку 6 раз) . Допустим, что выбраны числа

97 26 54 13 31 88

Это означает выбор хромосом ch_6 ch_4 ch_6 ch_1 ch_4 ch_6.

Пусть скрещивание выполняется с вероятностью 1. Для скрещивания случайным образом формируем пары ch_1 и ch_4 ch_4 и ch_6 ch_6 и ch_6.

Допустим, что случайным образом выбрана точка скрещивания, рваная 3 для первой пары, а также точка скрещивания , равная 2 для второй пары. Операция скрещивания для третьей пары бесполезна, т.к. дает тот же результат. При условии, что вероятность мутации равна 0, в новую популяцию включаются хромосомы:

Ch_1 = 10001 Ch_4= 11101
Ch_2= 10111 Ch_5= 11101
Ch_3= 10101 Ch_6 = 11101

В результате декодирования получаем числа

Ch_1 = 17 Ch_4= 29
Ch_2= 23 Ch_5= 29
Ch_3= 21 Ch_6 = 29

Функции приспособленности

Fch_1 = 579 Fch_4= 1683
Fch_2= 1059 F ch_5= 1683
Fch_3= 883 Fch_6 = 1683 Среднее значение F =1262, что значительно больше предыдущего значения.

Лабораторная работа №1

Генетические алгоритмы – это метод решения оптимизационных задач, основанный на биологических принципах естественного отбора и эволюции. Генетический алгоритм повторяет определенное количество раз процедуру модификации популяции (набора отдельных решений), добиваясь тем самым получения новых наборов решений (новых популяций). При этом на каждом шаге из популяции выбираются «родительские особи», то есть

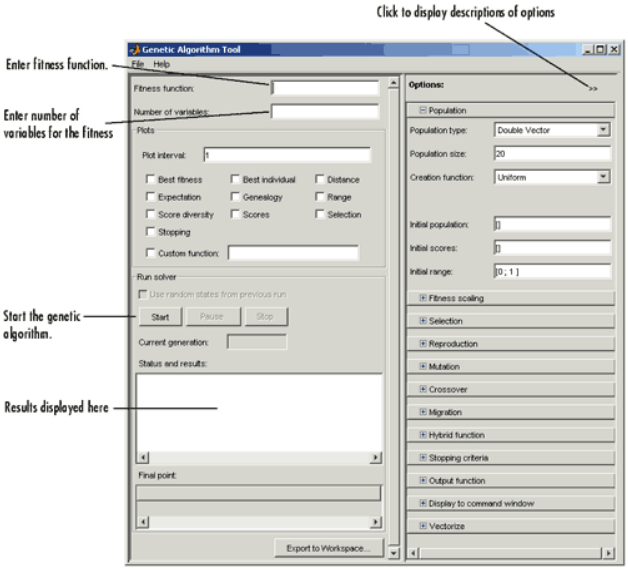
решения, совместная модификация которых (скрещивание) и приводит к формированию новой особи в следующем поколении. Генетический алгоритм использует три вида правил, на основе которых формируется новое поколение: правила отбора, скрещивания и мутации. Мутация позволяет путем внесения изменений в новое поколение избежать попадания в локальные минимумы оптимизируемой функции.
(Под катом основная часть + несколько скриншотов).
Механизм работы с генетическими алгоритмами в среде MATLAB реализован двумя способами:
1. Вызов функции генетических алгоритмов
2. Использование комплекта Genetic Algorithm Tool.
Оба способа поставляются в числе стандартного набора функций и модулей MATLAB Намного более удобным и наглядным является второй способ работы с генетическими алгоритмами в MATLAB, связанный с использованием модуля Genetic Algorithm Tool.

Работа с GENETIC ALGORITHM TOOL

Для запуска пакета Genetic Algorithm Tool следует в командной строке MATLABвыполнить команду gatool. После этого запустится пакет генетическихалгоритмов и на экране (рис.1) появится основное окно утилиты.

Открытие инструментария генетического алгоритма

Для того, что бы открыть инструментарий Генетического алгоритма, следует выполнить следующую команду – gatool в командной строке пакета MATLAB. В результате выполнения команды откроется инструментарий Генетического алгоритма, как это показано на рисунке ниже



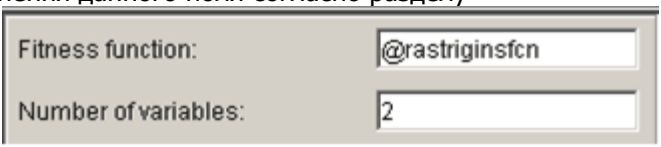
Постановка задачи для инструментария Генетического алгоритма

Для того, что бы ввести необходимую для решения задачу следует заполнить следующие два поля:

Fitness function - подлежащая минимизации функция. Следует ввести указатель на M-файл функции. Number of variables - число независимых переменных для функции пригодности.

Примечание.

Не следует использовать функцию Editor/Debugger для отладки M-файла с целевой функцией в период работы инструментария Генетического алгоритма. Подобные действия могут привести к Java сообщению о выполнении правильной операции в командном окне и воспрепятствуют процессу отладки. Вместо этого следует вызвать целевую функцию непосредственно из командной строки или передать ее в функцию генетического алгоритма ga. Для облегчения процесса отладки следует экспортировать решаемую задачу из инструментария Генетического алгоритма в рабочее пространство MATLAB, как это приведено в разделе Опции генетического алгоритма. Далее на рисунке представлен пример заполнения данного поля согласно разделу



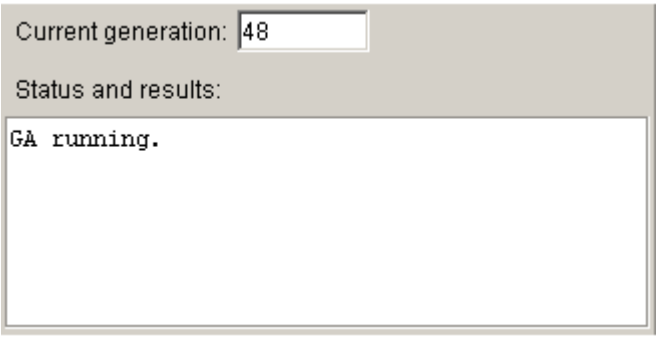
Выполнение генетического алгоритма

Для начала работы с Генетическим алгоритмом следует кликнуть мышкой на кнопку Start на панели Run solver. Далее выполняются следующие действия:

В поле Current generation выводится число текущих поколений.

На панели Status and results отображается сообщение "GA running".

На рисунке представлено состояние поля Current generation и панели Status and results в течение процесса выполнения алгоритма.



По окончании выполнения алгоритма на панели Status and results отображается следующая информация:

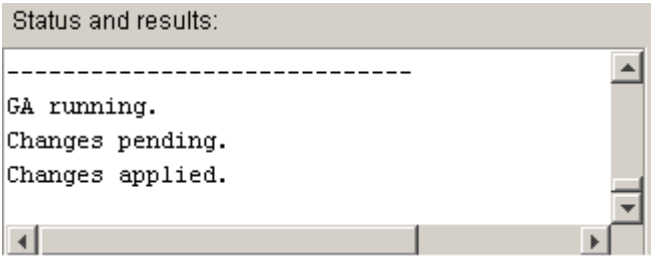
Сообщение "GA terminated"

Значение функции пригодности в точке наилучшего индивидуума для финального поколения.

Причина останова алгоритма.

Координаты конечной точки.

По ходу выполнения алгоритма, возможно, изменить ряд установок по работе с инструментарием Генетического алгоритма. Вносимые изменения влияют на состояние следующего поколения. По мере того, как вносятся изменения и эти изменения влияют на стартовое состояние последующего поколения, то на панели Status and Results можно видеть сообщение Changes pending. При запуске работы со следующим поколением уже можно видеть следующие сообщения Changes applied. Вид этих сообщений будет



Приостановка и остановка алгоритма

При выполнении алгоритма можно выполнить следующие действия:

Кликнуть мышкой на кнопку Pause и временно приостановить выполнение алгоритма. Для возобновления работы алгоритма из режима временного останова для работы с текущим семейством следует кликнуть мышкой на кнопку.

Клик на кнопку Stop останавливает выполнение алгоритма. На панели Status and results отображается значение функции пригодности для наилучшей точки текущего поколения на момент действия на кнопку Stop.

Примечание

Если в состоянии алгоритма после действия кнопки Stop запустить вновь Генетический алгоритм с помощью кнопки Start, то данный алгоритм возобновит свою работу с новым случайно выбранным семейством или с определенным семейством, предварительно выбранным с помощью поля Initial population. Если необходимо восстановить работу алгоритма с точки временного останова, следует пользоваться кнопками Pause и Resume.

Установка критериев останова

В Генетическом алгоритме используются следующие пять критериев останова и которые можно найти в списке опций Stopping criteria и с помощью которых можно установить точку автоматического останова без использования режима ручного останова с использованием кнопки Stop. Работа алгоритма останавливается в случае выполнения одного из следующих условий:

Generations - в алгоритме достигнуто определенное число поколений.

Time - прошло выполнение алгоритма в течение определенного времени в секундах.

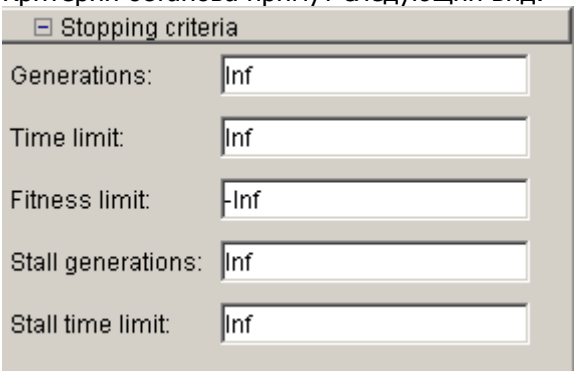
Fitness limit - Значение функции пригодности в текущем поколении стало меньше или равно некоего определенного значения.

Stall generations - в алгоритме проведены вычисления без улучшения функции пригодности в течение определенного числа поколений.

Stall time limit -- в алгоритме проведены вычисления без улучшения функции пригодности в течение определенного времени в секундах.

Если необходимо продолжить вычисления алгоритма до тех пор, пока не будут выполнены действия на кнопках Pause или Stop, то принимаемые по умолчанию значения опции следует установить в следующем виде:

установить Generations как Inf
установить Time как Inf.
установит Fitness limit как -Inf.
установить Stall generations как Inf.
установить Stall time limit как Inf.
Критерии останова примут следующий вид:

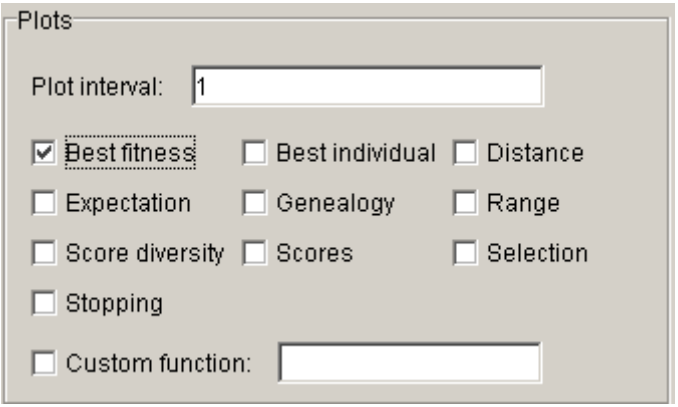


Примечание

Не следует использовать указанные выше установки при вызове Генетического алгоритма с помощью команды ga из командной строки, поскольку невозможно провести останов алгоритма до тех пор, пока не будет выполнена команда Ctrl + C. Взамен этого следует установить предельные значения для Generations или Time.

Графики для отображения

С помощью панели Plots, как это показано далее на рисунке, имеется возможность вывода различных графиков с результатами расчета с помощью генетического алгоритма.



Для формирования соответствующего типа графика следует выставить метку в соответствующий квадрат. Например, если выбрать Best fitness и Best individual и выполнить задачу из раздела

Примечание

При отображении более одного графика, если кликнуть мышкой на определенный график, то можно в отдельном окне получить увеличенную версию этого графика.

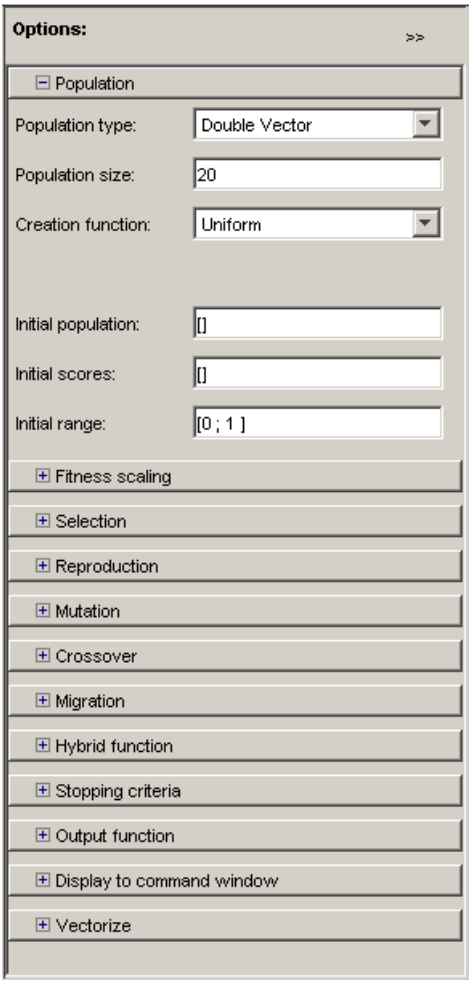
Воспроизведение результатов

Для воспроизведения результатов последнего запуска Генетического алгоритма следует выбрать команду Use random states from previous run контрольного бокса. С помощью этой команды снова переустанавливаются состояния генераторов произвольных чисел, используемых в данном алгоритме для их предыдущих значений. Если не изменять какие-либо установки в Инструментарии Генетического алгоритма, то в следующий раз запуск генетического алгоритма даст те же самые результаты, как и для предыдущего раза.

Естественно, что следует оставлять без выбора опцию Use random states from previous run, что дает определенные преимущества в произвольности Генетического алгоритма. При выборе опции Use random states from previous run в контрольном боксе можно проанализировать итоговые результаты некого специализированного запуска алгоритма или продемонстрировать заданную точность при других расчетах по данному алгоритму.

Установка опций для инструментария генетического алгоритма

С помощью панели Options имеется возможность установить опции Генетического алгоритма, как это показано на рисунке ниже.



Установка опций в качестве переменных в рабочего пространства MATLAB

Опции численного расчета можно установит или непосредственно, если набрать эти значения в соответствующем редакторском боксе, или ввести имена неких переменных в рабочем пространстве MATLAB которое включает в себя значения этих опций. Например, параметр Population size можно установить как 50 двумя способами:

- Ввести "50" в поле Population size
- Ввести "popsize = 50" в подсказке MATLAB и затем ввести в поле Population size

Переменные рабочего пространства MATLAB наиболее удобно использовать тогда, когда опции представляют собой векторы или матрицы большой размерности.

Импорт и экспорт опций и задач

В программном пакете имеются возможности, как экспортировать структуры опций и задач из инструментария Генетического алгоритма в рабочее пространство MATLAB, а так же через некоторое время импортировать их назад в инструментарий. С помощью таких операций имеется возможность сохранять текущие установки и затем в дальнейшем использовать их снова. Также в Генетическом алгоритме с помощью функции ga из командной строки можно экспортировать и затем повторно использовать одну и туже структуру опций.

Экспорту и импорту подлежит следующая информация:

Постановка задача, включая Fitness function и Number of variables.

Текущие специальные опции.

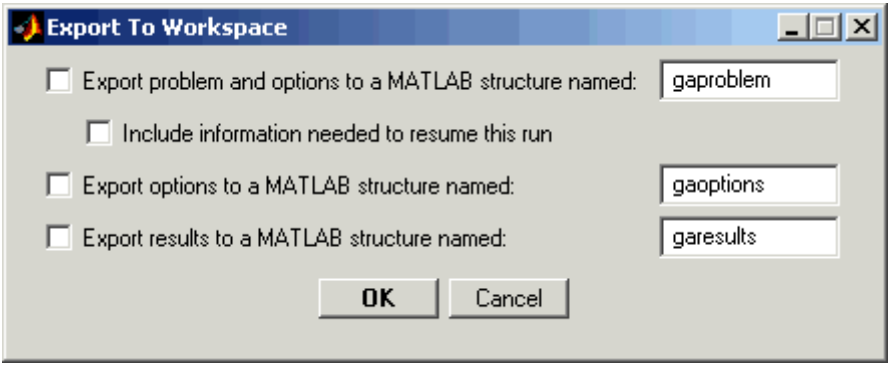
Результаты расчета алгоритма.

Экспорт опций и задач

Имеется возможность, как экспортировать структуры опций и задач из инструментария Генетического алгоритма в рабочее пространство MATLAB, а так же через некоторое время импортировать их назад в инструментарий. Кроме того, в Генетическом алгоритме операции экспорта опций и постановки задач можно выполнить с помощью функции ga - смотри ссылку

Использование опций и задач из инструментария Генетического алгоритма.

Для запуска операции экспорта опций и постановки задач следует кликнуть мышкой на кнопку Export или Export to Workspace из меню File. В результате этих операций откроется диалоговый бокс, как это показано на рисунке ниже.



В диалоговом боксе имеются следующие опции:

Для сохранения, как постановки задач, так и установки для текущих опций следует выделить Export problem and options to a MATLAB structure named и ввести имя данной структуры. В последующем, для возможности импорта этой структуры в инструментарий Генетического алгоритма, все установки для Fitness function, Number of variables и все остальные установки для опций должны быть сохранены в специальные структуры.

Примечание. Если до начала экспорта задачи выбрать Use random states from previous run в панели Run solver, то при операции экспорта в начале последнего запуска выполняемой задачи в инструментарии Генетического алгоритма также выполняется и операция сохранения состояний rand и randn. Далее, в случае выбора параметра Use random states from previous run, при импорте задачи и запуске Генетического алгоритма полностью воспроизводятся результаты выполнения предыдущей задачи.

Если необходимо, что бы в Генетическом алгоритме перед операцией экспорта задачи были восстановлены результаты запуска программы из последнего семейства, то следует выбрать параметр Include information needed to resume this run. Далее при импорте структуры задачи и выполнении команды Start в алгоритме проводится восстановление параметров из конечного семейства предыдущего запуска программы.

Для сохранения принимаемого по умолчанию состояния генетического алгоритма с генерацией стохастического начального семейства, необходимо удалить название семейства в поле Initial population и вставить на его место пустые скобки [].

Примечание. Если была помечена команда Include information needed to resume this run, то тогда выбор Use random states from previous run не оказывает влияния на исходное семейство, создаваемое при импорте задачи и выполнении соответствующего Ге-

нетического алгоритма. Последняя опция предназначена только для воспроизведения результатов в соответствии с началом нового запуска программы, а не восстановления результатов запуска.

Для сохранения только выбранных опций следует выбрать Export options to a MATLAB structure named и ввести некое новое имя для структуры опции.

Для сохранения результатов последнего запуска программы следует выбрать параметр Export results to a MATLAB structure named и ввести некое новое имя для структуры опции.

Пример - выполнение программы ga для экспорта задач

Для экспорта задачи и выполнения функции ga из командной строки следует выполнить следующие шаги:

Кликнуть мышкой на команду Export to Workspace.

В диалоговом боксе Export to Workspace dialog box ввести имя для структуры задачи, например, my_problem в поле Export problems and options to a MATLAB structure named.

В подсказке MATLAB вызвать функцию ga с водным аргументом my_problem:

```
[x fval] = ga(my_problem)
```

В результате чего получим

```
x =0.0027  - 0.0052
fval =0.0068
```

Импорт опции

Для импорта структуры опций из рабочего пространства MATLAB следует выбрать команду Import Options из меню File. В результате выполнения этой команды откроется диалоговый бокс с отображением списка структур опций Генетического алгоритма в рабочем пространстве MATLAB. Далее следует выбрать структуру опции и кликнуть на команду Import, и поле опций в инструментарии Генетического алгоритма будет изменено с учетом значений импортируемых опций.

Структуру опций можно создать двумя способами:

При вызове команды gaoptimset с опцией в качестве выходного аргумента

Путем сохранения текущих опций при экспорте в диалоговый бокс рабочего пространства инструментария.

Импорт задачи

Для импорта задачи, которая предварительно была экспортирована из инструментария Генетического алгоритма, следует выбрать Import Problem из меню File. В результате выполнения этой команды откроется диалоговый бокс с отображением списка структур задач Генетического алгоритма в рабочем пространстве

MATLAB. Далее следует выбрать структуру задач и кликнуть на команду OK, в результате чего будут изменены следующие поля в инструментарии Генетического алгоритма

Fitness function
Number of variables
Linear inequalities
Linear equalities
Bounds
Nonlinear constraint function
Поля в панели Options
Переустановка полей задач

Для переустановки или очистки данных после выполнения задачи в Генетическом алгоритме, следует выбрать поле New Problem из File menu. По этой команде происходит переустановка всех полей Генетического алгоритма на принимаемые по умолчанию опции.

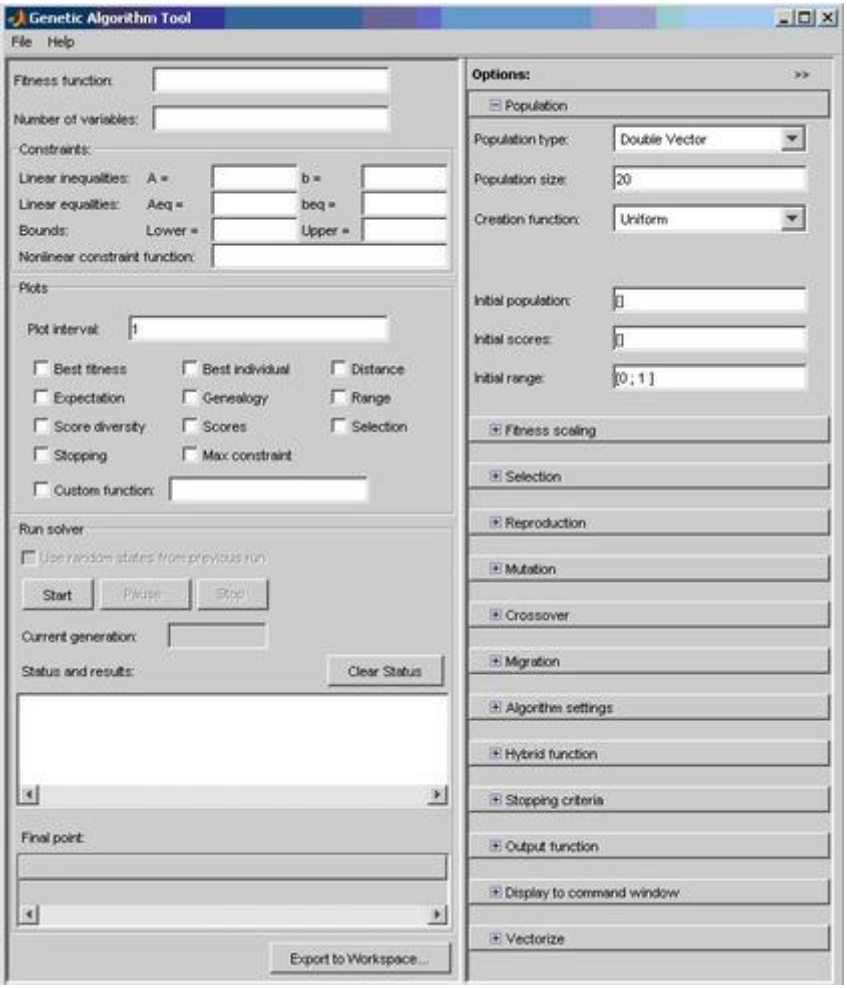


Рис.1.

В поле Fitness function указывается оптимизируемая функция в виде @fitnessfun, где fitnessfun.m – название М-файла, в котором предварительно следует описать оптимизируемую функцию. На всякий случай отмечу, что М-файл создается в среде MATLAB через меню File->New->M-File. Пример описания некоторой функции my_fun в М-файле:

```
function z = my_fun(x)
z = x(1)^2 — 2*x(1)*x(2) + 6*x(1) + x(2)^2 — 6*x(2);
```

Вернемся к основному окну утилиты GATool. В поле Number of variables указывается длина входного вектора оптимизируемой функции. В рассмотренном выше примере функция my_fun имеет входной вектор длины 2.

В панели Constraints можно задать ограничения или ограничивающую нелинейную функцию. В поле Linear inequalities задается линейное ограничение неравенством вида:

$A*x \leq b$.

В поле Linear equalities данной панели задаются линейные ограничения равенством:

$A*x = b$.

В обоих случаях A – некоторая матрица, b – вектор. В поле Bounds в векторном виде задаются нижнее и верхнее ограничения переменных, а в поле Nonlinear constraint function можно задать произвольную нелинейную функцию ограничений. Если в конкретной задаче не требуется задание ограничений, все поля панели Constraints следует оставить незаполненными. Ниже находится панель настройки графиков. Она позволяет выводить различные графики, отображающие информацию о работе генетического алгоритма. На основе этой информации можно менять настройки алгоритма с целью повышения эффективности его работы. Например, *выбор опции* Best Fitness в этой панели позволяет вывести на одном графике лучшее и среднее значение оптимизируемой функции для каждого поколения работы алгоритма.

Подробнее эта панель будет описана ниже наравне с остальными панелями вкладки Options утилиты GATool. Панель Run Solver содержит управляющие элементы (кнопки Start, Pause и Stop для начала, временной и полной остановки работы генетического алгоритма). Также она содержит поля Status and results, в которое выводятся текущие результаты работы запущенного генетического алгоритма, и Final point, в котором выводится значение конечной точки работы алгоритма — наилучшей величины оптимизируемой функции (то есть, искомое значение).

В правой части основного окна утилиты GATool находится панель Options. Она позволяет устанавливать различные настройки для работы генетических алгоритмов. При щелчке мышью по кнопкам [+], которые находятся напротив названия каждого из настраиваемых параметров в панели Options, появляются выпадающие списки (вкладки), содержащие поля для ввода и изменения соответствующих параметров генетического алгоритма.

Основными настраиваемыми параметрами в GATool являются:

1. популяция (вкладка Population);
2. масштабирование (вкладка Fitness Scaling); оператор отбора (вкладка Selection);
3. оператор репродукции (вкладка Reproduction);
4. оператор мутации (вкладка Mutation);
5. оператор скрещивание (вкладка Crossover);
6. перенесение особей между популяциями (вкладка Migration);
7. специальные параметры алгоритма (вкладка Algorithm settings);
8. задание гибридной функции (вкладка Hybrid function);
9. задание критерия остановки алгоритма (вкладка Stopping criteria);
10. вывод различной дополнительной информации по ходу работы

11. генетического алгоритма (вкладка Plot Functions);
12. вывод результатов работы алгоритма в виде новой функции (вкладка Output function);
13. задание набора информации для вывода в командное окно (вкладка Display to command window);
14. способ вычисления значений оптимизированной и ограничивающей функций (вкладка User function evaluation).

Рассмотрим подробнее все вышеперечисленные вкладки панели Options и элементы, которые они содержат. Во вкладке настройки популяций пользователь имеет возможность выбрать тип математических объектов, к которому будут относиться особи всех популяций (двойной вектор, битовая строка или пользовательский тип). При этом стоит учитывать, что использование битовой строки и пользовательских типов накладывают ограничения на перечень допустимых операторов создания, мутации и скрещивания особей. Так, например, при выборе в качестве формы представления особей битовой строки для оператора скрещивания нельзя использовать гибридную функцию или нелинейную ограничивающую функцию.

Также вкладка популяции позволяет настраивать размер популяции (из скольких особей будет состоять каждое поколение) и каким образом будет создаваться начальное поколение (Uniform – если отсутствуют накладываемые ограничения, в противном случае — Feasible population). Кроме того, в рассматриваемой вкладке имеется возможность задать вручную начальное поколение (используя пункт Initial population) или его часть, начальный рейтинг особей (пункт Initial scores), а также ввести ограничительный числовой диапазон, которому должны принадлежать особи начальной популяции (Initial range).

Во вкладке масштабирования (Fitness Scaling) пользователь имеет возможность указать функцию масштабирования, которая конвертирует достигаемые оптимизируемой функцией значения в значения, лежащие в пределах, допустимых для оператора отбора. При выборе в качестве функции масштабирования параметра Rank масштабирование будет приводиться к рейтингу, то есть особям присваивается рейтинговый номер (для лучшей особи – единица, для следующей – двойка, и так далее). Пропорциональное масштабирование (Proportional) задает вероятности пропорционально заданному числовому ряду для особей. При выборе опции Top наибольшее рейтинговое значение присваивается сразу нескольким наиболее выдающимся особям (их число указыва-

ется в виде параметра). Наконец, при выборе масштабирования типа Shift linear имеется возможность указать максимальную вероятность наилучшей особи.

Вкладка Selection позволяет выбрать оператор отбора родительских особей на основе данных из функции масштабирования. В качестве доступных для выбора вариантов оператора отбора предлагаются следующие:

Tournament – случайно выбирается указанное число особей, среди них на конкурсной основе выбираются лучшие;

Roulette – имитируется рулетка, в которой размер каждого сегмента устанавливается в соответствии с его вероятностью;

Uniform – родители выбираются случайным образом согласно заданному распределению и с учетом количества родительских особей и их вероятностей;

Stochastic uniform – строится линия, в которой каждому родителю ставится в соответствие её часть определенного размера (в зависимости от вероятности родителя), затем алгоритм пробегает пот линии шагами одинаковой длины и выбирает родителей в зависимости от того, на какую часть линии попал шаг.

Вкладка Reproduction уточняет каким образом происходит создание новых особей. Пункт Elite count позволяет указать число особей, которые гарантировано перейдут в следующее поколение. Пункт Crossover fraction указывает долю особей, которые создаются путем скрещивания. Остальная доля создается путем мутации.

Во вкладке оператора мутации выбирается тип оператора мутации. Доступны следующие варианты:

Gaussian – добавляет небольшое случайное число (согласно распределению Гаусса) ко всем компонентам каждого вектора-особи;

Uniform – выбираются случайным образом компоненты векторов и вместо них записываются случайные числа из допустимого диапазона;

Adaptive feasible – генерирует набор направлений в зависимости от последних наиболее удачных и неудачных поколений и с учетом налагаемых ограничений продвигается вдоль всех направлений на разную длину;

Custom – позволяет задать собственную функцию.

Вкладка Crossover позволяет выбрать тип оператора скрещивания (одноточечное, двухточечное, эвристическое, арифметическое или рассеянное (Scattered), при котором генерируется

случайный двоичный вектор соответствия родителей). Также имеется возможность задания произвольной (custom) функции скрещивания.

Во вкладке Migration можно настраивать правила, согласно которым особи будут перемещаться между подпопуляциями в пределах одной популяции. Подпопуляции создаются, если в качестве размера популяции указан вектор, а не натуральное значение. В данной вкладке можно указать направление миграции (forward – в следующую подпопуляцию, both – в предыдущую и следующую), долю мигрирующих особей и частоту миграции (сколько поколений проходит между миграциями). Если создание подпопуляций не требуется, эту вкладку всегда стоит оставлять без изменений.

Вкладка специальных опций алгоритма позволяет настраивать параметры решения системы нелинейных ограничений, налагаемых на алгоритм. Значение параметра Initial penalty определяет начальное числовое значение критики алгоритма, Penalty factor используется как множитель этого значения в случаях, когда разработчика не устраивает точность оптимизации или при выходе за границы, определенные во вкладке ограничений. Как правило, эти опции детально настраиваются для решения задач высокой сложности.

Вкладка Hybrid function позволяет задать ещё одну функцию минимизации, которая будет использоваться после окончания работы алгоритма. В качестве возможных гибридных функций доступны следующие встроенные в саму среду MATLAB функции:

none (не использовать гибридную функцию);
fminsearch (поиск минимального из значений);
patternsearch (поиск по образцу);
fminunc (для неограниченного алгоритма);
fmincon (для алгоритма с заданными ограничениями).

Во вкладке критерия остановки (Stopping criteria) указываются ситуации, при которых алгоритм совершает остановку. При этом, настраиваемыми являются следующие параметры:

Generations – максимальное число поколений, после превышения которого произойдет остановка;

Time limit – лимит времени на работу алгоритма;

Fitness limit – если оптимизируемое значение меньше или равно данного лимита, то алгоритм остановится;

Stall generations – количество мало отличающихся поколений, по прошествии которых алгоритм остановится;

Stall time limit – то же, что и предыдущий параметр, но применимо к времени работы алгоритма;

Function tolerance и Nonlinear constraint tolerance – минимальные значения изменений оптимизируемой и ограничивающей функций соответственно, при которых алгоритм продолжит работу.

Особый интерес представляет вкладка Plot Functions, которая позволяет выбирать различную информацию, которая выводится по ходу работы алгоритма и показывает как корректность его работы, так и конкретные достигаемые алгоритмом результаты. Наиболее важными и используемыми для отображения параметрами являются:

Plot interval – число поколений, по прошествии которого происходит очередное обновление графиков;

Best fitness – вывод наилучшего значения оптимизируемой функции для каждого поколения;

Best individual – вывод наилучшего представителя поколения при наилучшем оптимизационном результате в каждом из поколений;

Distance – вывод интервала между значениями особей в поколении;

Expectation – выводит ряд вероятностей и соответствующие им особи поколений;

Genealogy – вывод генеалогического дерева особей;

Range – вывод наименьшего, наибольшего и среднего значений оптимизируемой функции для каждого поколения;

Score diversity – вывести гистограмму рейтинга в каждом поколении;

Scores – вывод рейтинга каждой особи в поколении;

Selection – вывод гистограммы родителей;

Stopping – вывод информации о состоянии всех параметров, влияющих на критерии остановки;

Custom – отображение на графике некоторой указанной пользователем функции.

Вкладка вывода результатов в виде новой функции (Output function) позволяет включить вывод истории работы алгоритма в отдельном окне с заданным интервалом поколений (флаг History to new window и поле Interval соответственно), а также позволяет задать и вывести произвольную выходную функцию, задаваемую в поле Custom function.

Вкладка User function evaluation описывает, в каком порядке происходит вычисление значений оптимизируемой и ограничива-

ющей функций (отдельно, параллельно в одном вызове или одновременно).

Наконец, вкладка Display to command window позволяет настраивать информацию, которая отображается в основном командном окне MATLAB при работе алгоритма. Возможны следующие значения: Off — нет вывода в командное окно, Iterative — вывод информации о каждой итерации работающего алгоритма, Diagnose — вывод информации о каждой итерации и дополнительных сведениях о возможных ошибках и измененных ключевых параметрах алгоритма, Final – выводится только причина остановки и конечное значение.

Лабораторная работа № 9
Оптимизация с использованием генетических алгоритмов

Цели и задачи работы:

Изучение генетических алгоритмов с помощью графического интерфейса. Нахождениеглобального минимума функции одной переменной.

Задание

1. Создать m-файл функции, которую необходимо минимизировать.

Варианты заданий

cos(x), sin(x).

Сохранить этот файл под именем my_fun1.m.

2. Вызвать графический интерфейс: gatool.

В поле fitness function дать имя этому файлу: @my_fun1.

— Задать параметры генетического алгоритма:
— количество особей (Population size) 50,
— поколений (Generations) 100 (во вкладке Stopping criteria),
— начальный отрезок (initial range): [0,1].
— В группе параметров plots установить флажок best fitness.
— Запуск на счет производится с помощью кнопки start.

3. Провести численные эксперименты с целью нахождения параметров генетического алгоритма, обеспечивающих решение с необходимой точностью. Объяснить найденные закономерности.

— Методические указания

— Генетические алгоритмы основаны на моделировании процессов биологической эволюции и относятся к стохастическим методам.

— Терминологический аппарат включает заимствованные из биологии и генетики термины, среди которых наиболее часто используются следующие.

— Хромосома – вектор (vector) чисел или строка, состоит из набора генов. Гены (genes) – элементы хромосомы. Скрещивание (crossover) или кроссинговер – обмен фрагментами двух хромосом. Мутация (mutation) – случайное изменение гена. Особь или индивидуум (individual) – вариант решения задачи (приближенное решение), обычно задается одной хромосомой. Популяция (population) – набор особей. Поколение (generation) – итерация эволюционного процесса поиска решения задачи. В ходе каждой итерации особь оценивается с помощью функции пригодности (fitness function).

— В правой части графического интерфейса gatool приведено краткое объяснение (quick reference) некоторых терминов математического аппарата генетических алгоритмов и дана вспомогательная информация по использованию графического интерфейса.

Лабораторная работа № 10
Задача глобальной оптимизации функции двух переменных

Цели и задачи работы:

— Нахождение глобального минимума функции двух аргументов с помощью генетических алгоритмов.

— Изучение генетических алгоритмов в режиме командной строки Matlab.

Задание

1. Создать m-файл с записью заданной функции, которую необходимо минимизировать:

— $f(x,y) = -\cos^2(2r)\exp(-r^2)$,

— где $r^2 = (x-0.2)^2 + (y-0.7)^2$.

— Сохранить этот файл под именем my_fun2.m.

2. Вызвать графический интерфейс gatool.

— В поле fitness function задать имя файла: @my_fun2.

— Задать самостоятельно параметры генетического алгоритма.

3. Провести численные эксперименты с целью нахождения параметров генетического алгоритма, обеспечивающих решение с необходимой точностью. Проверить правильность нахождения решения.

4. Сохранить задание в m-файле. Для этого в меню File выбрать команду Generate M-file. Запустить созданный файл ga2.m на выполнение в режиме командной строки.
5. Провести численные эксперименты с файлом ga2.m в режиме командной строки: изменить начальное приближение 'PopInitRange' на другое, увеличить или уменьшить популяцию 'PopulationSize', добавить для параметра 'PlotFcns' дополнительные опции @gaplotscores @gaplotscorediversity.

Методические указания

Данная лабораторная работа является усложненным вариантом предыдущей. Файл my_fun2.m может иметь следующий вид:

— **function** y = my_fun2(x)
— r = (**x(**1)-0.2).^2 + (x(2)-0.7).^2;
— y = – **cos(**2*sqrt(r)).^2.*exp(-r);

В некоторых случаях полезна опция History to new function во вкладке Stopping criteria, позволяющая вывести численные значения функции пригодности на каждом шаге.

Важность выполнения пункта 4 заключается в том, что файл ga2.m может в дальнейшем быть модифицирован и включен в текст более сложной программы.

Задать число поколений можно с помощью следующей команды:

options = gaoptimset(options,'Generations'.

ЧАСТЬ 4 НЕЧЕТКАЯ ЛОГИКА В СИСТЕМАХ УПРАВЛЕНИЯ

Лабораторная работа №1
Изучение нечеткой логики управления в приложении fuzzylogic

Цель: Закрепить знания по разделу нечеткие системы управления, систематизировать эти знания, научиться реализовывать модели на ЭВМ.

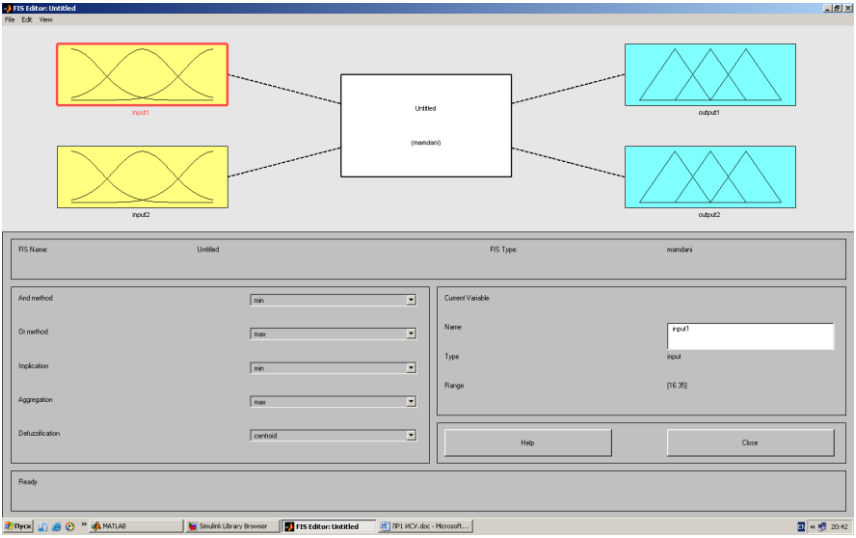
Ход работы

1. Постановка задачи

Определиться с количеством входных, выходных параметров, получить структурную схему системы управления. Реализовать модель управления.

2. Построение модели

Создадим двумерную модель кондиционера:



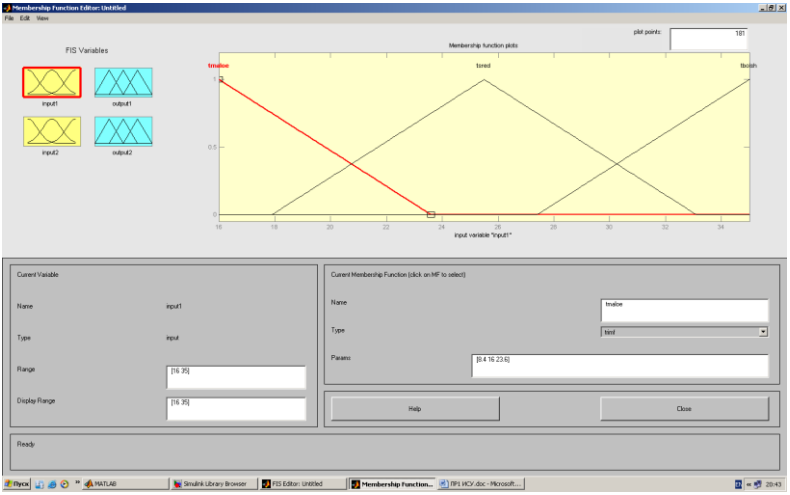
Входные параметры: температура внутри помещения и скорость изменения температуры в помещение

Выходные параметры: скорость вращения вентилятора и расход охлаждающей жидкости.

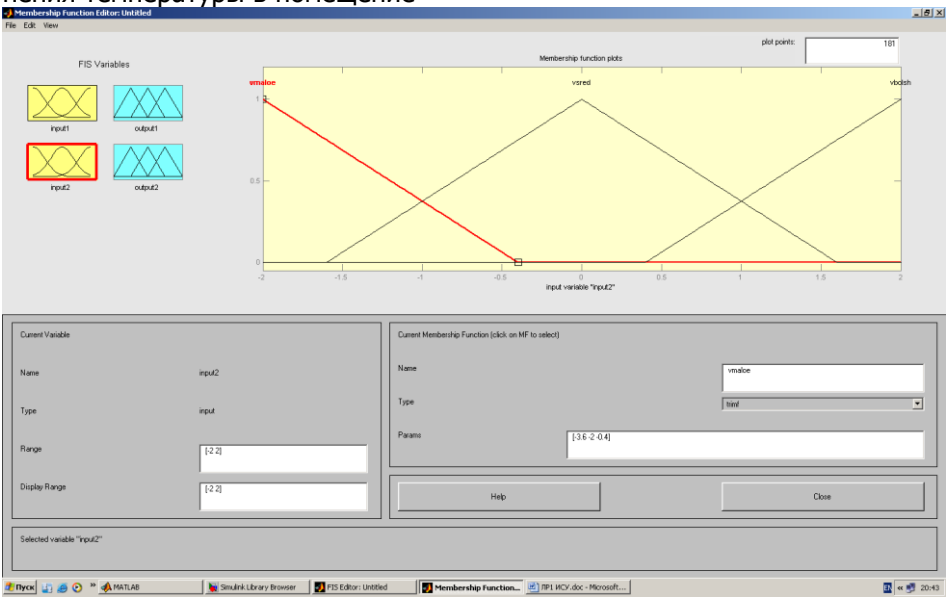
3. Создание термов

Для каждого параметра создадим термы:

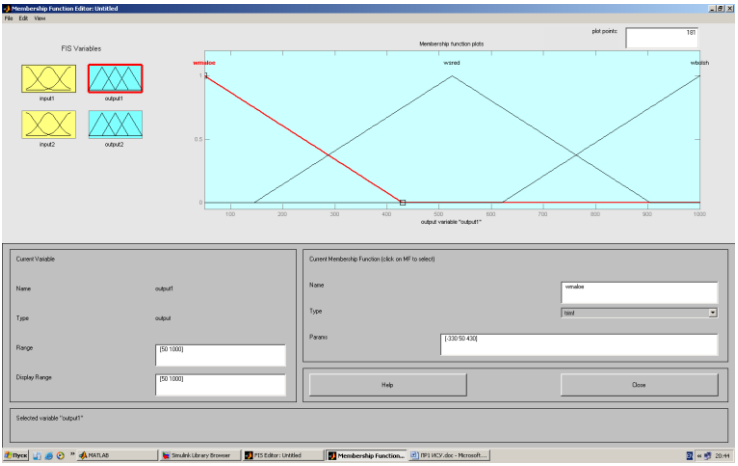
На данном графике указан входной сигнал «температура внутри помещения»



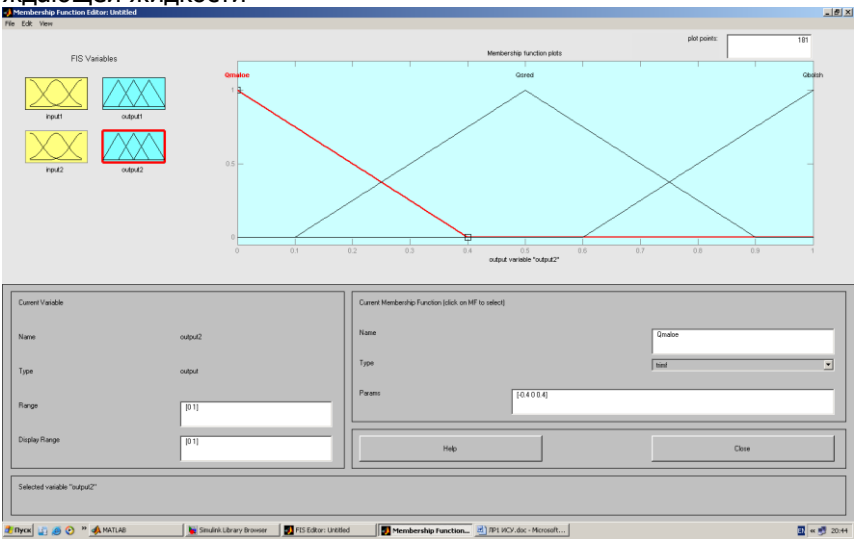
На данном графике указан входной сигнал «скорость изменения температуры в помещение»



На данном графике указан выходной сигнал «скорость вращения вентилятора»

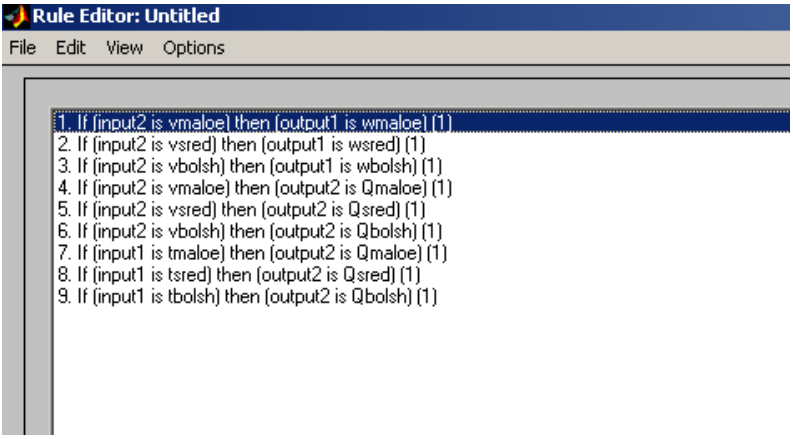


На данном графике указан выходной сигнал «расход охлаждающей жидкости»

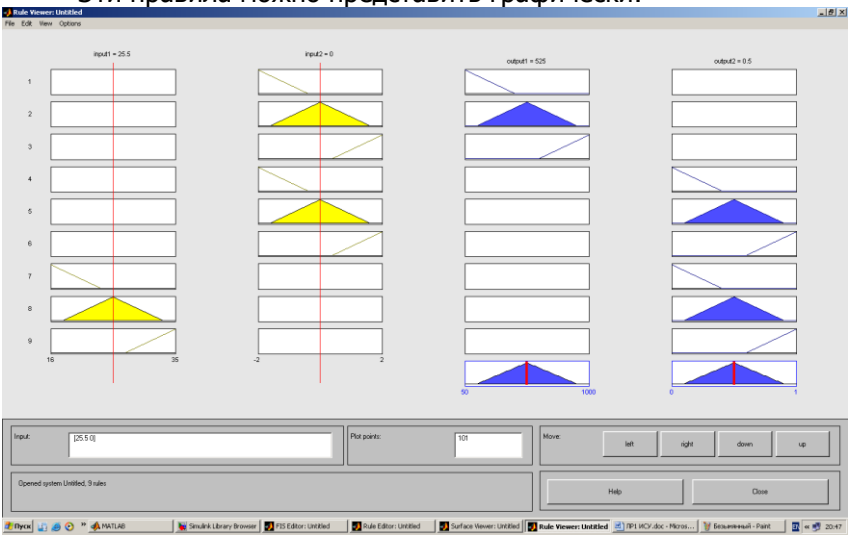


4. Создание правил

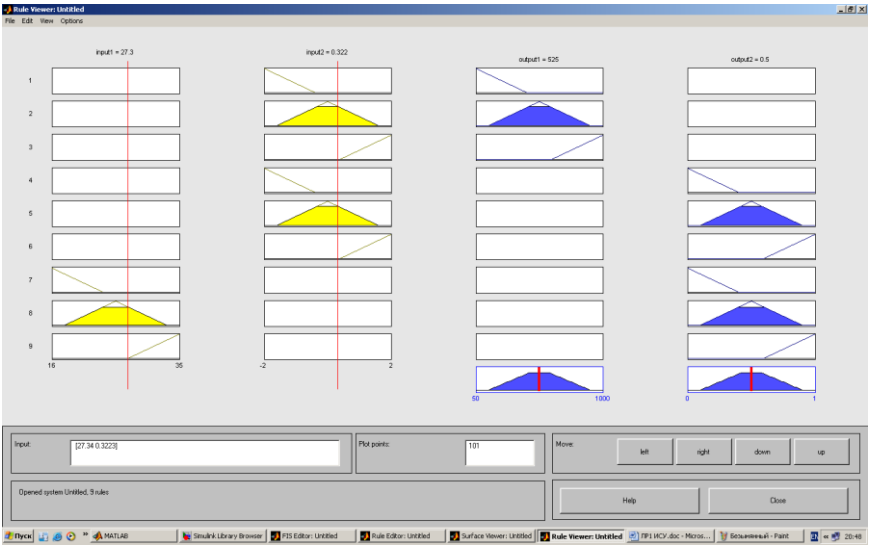
Для различных ситуаций создадим правила задания режима работы



Эти правила можно представить графически:

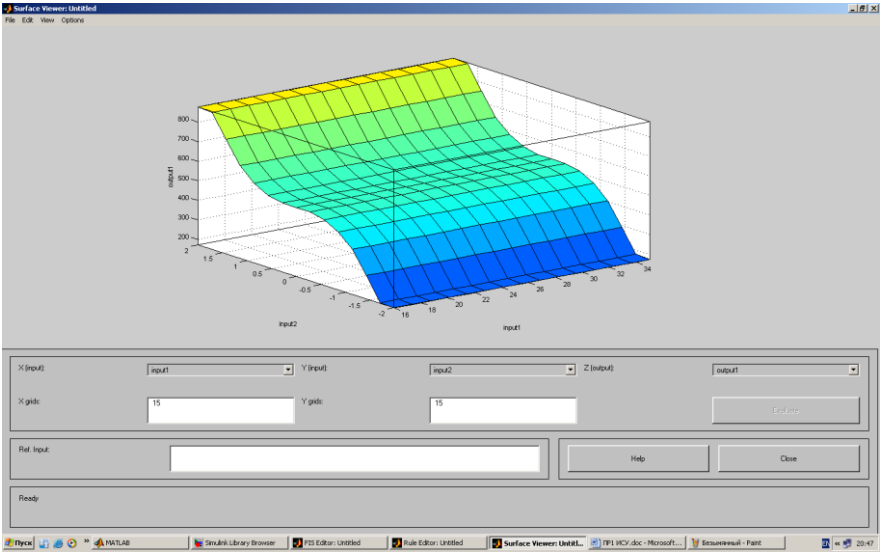


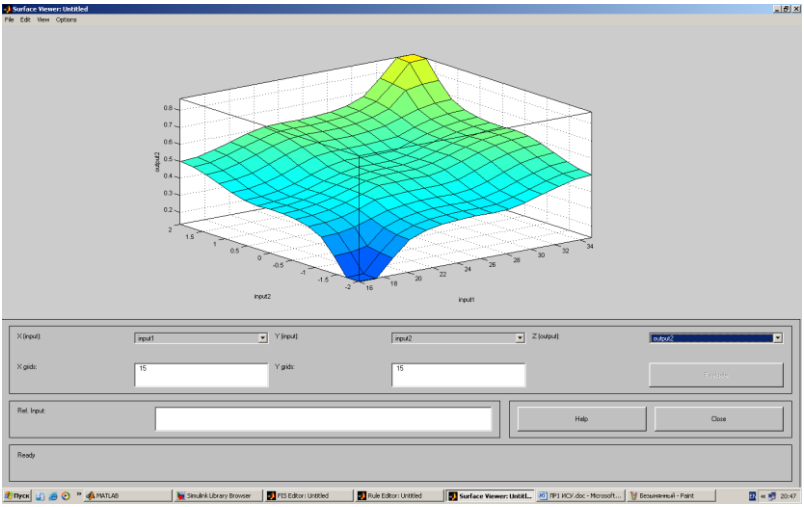
На этом графике мы изменили входные сигналы



5. Анализ полученной модели

Построим графики зависимости скорости вращения вентилятора, температуры в помещение и скорости изменения температуры.





Вывод: закрепили знания по разделу нечеткие системы управления, систематизировали их, научились реализовывать модели на ЭВМ. Модель ведет себя адекватно и отвечает всем требованиям.

Лабораторная работа № 2
Нечеткая система управления Sugeno

Цели и задачи работы:

Моделирование системы управления краном воды душа с помощью нечеткой системы типа Sugeno.

Задание

1. Создать нечеткую систему управления краном горячей или холодной воды душа в соответствии с вариантом. Входные переменные программы: *t* – температура падающей воды душа, *dt* – скорость изменения температуры (изменение температуры воды за 1 секунду). Нечеткая система должна на выходе формировать управляющий сигнал: *u* – угол поворота крана (в радианах или градусах). Нужно использовать упрощенный алгоритм нечеткого вывода (Sugeno 0-го порядка) и центроидный метод дефаззификации.

2. Провести настройку параметров нечеткой системы, корректируя вид функций принадлежности.

Варианты заданий

1. Управлять краном горячей воды. Использовать 9 правил.

2. Управлять краном холодной воды. Использовать 8 правил.

Используемые операторы и команды:

fuzzy – редактор нечеткой системы вывода (FIS-редактор)

Методические указания

В данной системе управления на вход поступает разность температуры падающей воды и температуры комфортного уровня, например, 40^{O} С. Известно, что температура воды при принятии душа иногда меняется по различным причинам, поэтому приходится корректировать положение крана горячей или холодной воды. Следует учесть инерционность действия крана душа, так как после прохождения регулирующего крана вода проходит некоторый путь по шлангу и только после этого падает на человека, который принимает душ. Поэтому большое значение имеет учет скорости изменения температуры воды для выработки управляющего сигнала для крана. Например, если температура в данный момент имеет комфортный уровень 40 ^{o}С, но скорость изменения температуры велика (имеет положительное значение), то угол поворота крана горячей воды следует уменьшить.

Для выполнения п.1 следует создать нечеткую систему типа Sugeno с двумя входами: вход 1 – температура падающей из душа воды, вход 2 – скорость изменения этой температуры. Создаваемая нечеткая система должна быть аналогична той, что была создана ранее при выполнении п.2 лабораторной работы № 3.

Лабораторная работа № 3
Нечеткая экспертная система с алгоритмом вывода Mamdani

Цели и задачи работы:

Построение нечеткой экспертной системы с алгоритмом вывода Mamdani.

Задание

1. Создать нечеткую экспертную систему с алгоритмом вывода Mamdani, которая должна оценить уровень работы предприятия общественного питания. Использовать 2 входа, 1 выход, 3 правила типа «если... то», «если... или…то».

2. Построить усложненный вариант нечеткой экспертной системы для оценки работы предприятия или другого объекта (число входов и правил должно соответствовать заданному варианту). Использовать правила типа «если... то», «если... или…то»,

«если... и…то». Самостоятельно предложить входные переменные и правила вывода.

Варианты заданий:

Оценка квартиры для бюро обмена жилья: 5 входов, 7 правил.

Уровень обслуживания в пассажирском поезде: 4 входа, 6 правил.

Оценка успеваемости студентов: 3 входа, 5 правил.

Методические указания

При решении п.1 могут быть заданы следующие инструкции.

1. Если сервис плохой или еда несвежая, то оценка низкая.
2. Если обслуживание хорошее, то оценка средняя.
3. Если сервис отличный или еда вкусная, то оценка высокая.

При создании новой системы нечеткого вывода по умолчанию создается система с алгоритмом вывода Mamdani. Качество обслуживания и еды будем оценивать по 5-балльной системе.

В пункте меню Edit Variable/Input добавить второй вход (появится второй блок с именем input2). Однократным щелчком левой кнопкой мыши по блоку input1 заменить его имя на «service», input2 – на «food», output1 – на «grade».

Задать функции принадлежности. Для переменной «service» в полях Range и Display Range установить диапазон изменения и отображения этой переменной – от 0 до 5 (подтверждая ввод нажатием клавиши Enter). Удалить заданные по умолчанию функции принадлежности с помощью мыши. Через пункт меню Edit/Add MFs задать функции принадлежности гауссова типа (gaussmf) в количестве 3. Заменить их имена на «bad», «good» и «excellent».

Щелчком мыши по «food» задать диапазон изменения от 0 до 5. После удаления заданных по умолчанию функций принадлежности задать две функции принадлежности трапецеидальной формы trapfm с параметрами [0 0 1 3] и [2 4 5 5] и именами «not fresh» и «nice».

Для выходной переменной «grade» указать диапазон [0 5], задать три функции принадлежности треугольной формы с именами «bad», «good», «excellent».

Конструирование правил реализовано в пункте меню Edit/Rules. В первом и третьем правилах использовать «or» (единица в скобках после каждого правила указывает его «вес», который

нужно оставить равным 1). При вводе второго правила, где отсутствует переменная «food», выбрать опцию none.

Из пункта меню View/Rules вызывать окно графического отображения функционирования системы, где можно задавать значения входных переменных в соответствующих полях, ответ отображается в правой части окна. Можно также перемещать отметку шкалы с помощью мыши.

Из пункта меню View/Surface вызвать отображение двумерной функции оценки работы предприятия общественного питания.

Сохранить созданную систему на диске: **File/Export/To disk**.

При выполнении задания п.2 нужно усложнить созданную экспертную систему: ввести заданное число правил и входов. Экспертная система должна быть достаточно продуманной и иметь практическое значение.

Используемые операторы и команды:

fuzzy – редактор нечеткой системы вывода.

Программное обеспечение: MATLAB 6.5.

ЛАБОРАТОРНАЯ РАБОТА № 4
Работа Fuzzy Logic с блоками Simulink.

Цель работы: приобретение навыков Работы Fuzzy Logic с блоками Simulink.

Теоретическая часть

Система нечёткого вывода, созданные тем или иным образом с помощью пакета Fuzzy Logic Toolbox, допускают интеграцию с инструментами пакета Simulink, что позволяет выполнять моделирование систем в рамках последнего. Рассмотрим это на примере контроля уровня воды в баке [3]. Для построения какой-то собственной моделирующей системы с использованием средств нечеткой логики и блоков Simulink рекомендуется просто скопировать блок Fuzzy Logic Controller из рассмотренной системы sltank (или какого-либо другого демонстрационного примера MATLAB) и поместить его в блок-диаграмму разрабатываемой системы. Из командной строки командой **fuzblock** можно также открыть библиотеку нечетких блоков пакета Simulink, которые могут быть использованы точно так же, при необходимости – с дополнительным редактированием.

Функционирование указанных блоков осуществляется с использованием системной S-функции **sffis.mex.** Запись этой функции такова:

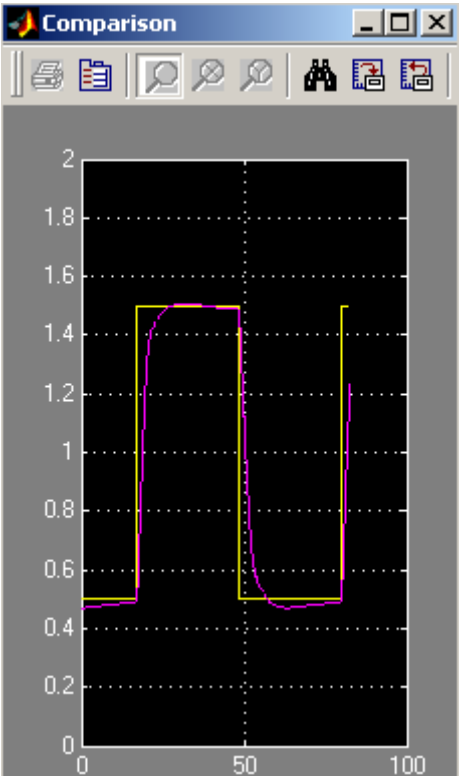


Рис. 3. Результаты моделирования системы управления с нечётким регулятором.

output = sffis(t,x,u,nag,nsmat),

где output – выход нечеткого регулятора, t, x и flag – стандартные аргументы системной функции Simulink, fismat – идентификатор (имя) нечеткой системы вывода, и входной сигнал (вектор входных сигналов) регулятора.

По смыслу данная функция аналогична рассмотренной выше функции evalfis, но она оптимизирована для работы в среде Simulink.

Демонстрационные примеры работы с пакетом **Fuzzy Logic Toolbox.**

Для ознакомления с пакетом Fuzzy Logic Toolbox можно использовать следующие функции (команды) в режиме командной строки:

defuzzdm – обзор методов приведения к четкости (дефазификации),

fcmdemo – демонстрация алгоритма кластеризации Fuzzy c-means (2-D графика),

fuzdemos – демонстрация графического интерфейса пакета Fuzzy Logic,

gasdemo – демонстрация использования аппарата гибридных сетей для решения задачи о выборе автомобиля, наиболее экономичного по расходу топлива,

juggler – демонстрация системы жонглирования мячом с использованием нечеткого регулятора,

invkine – демонстрация нечеткого управления движением робота-манипулятора,

irisfcm – демонстрация алгоритма кластеризации Fuzzy c-means,

noisedm – демонстрация решения задачи фильтрации на основе методов нечеткой логики,

slbb – демонстрация задачи «шар на качелях»,

slcp – демонстрация нечеткой системы управления перевернутым маятником,

sltank – демонстрация системы управления уровнем воды в баке с нечетким регулятором,

sltankrule – то же, что в предыдущем случае, но с дополнительным просмотром нечетких правил,

sltbu – демонстрация функционирования нечеткой системы управления грузовиком.

С каждым из этих примеров связан M-файл, fis-файл или/и блок-диаграмма Simulink, запуск которых производится с помощью одной из указанных команд. Текст пояснений в примерах – на английском языке. Данные примеры доступны и через главное меню MATLAB (пункт Help/Examples and Demos, раздел Toolboxes/Fuzzy Logic). Дополнительную информацию можно получить, используя команду **help fuzzy.**

Пример 1. Контроль уровня воды в баке.

На рис. 1 изображен объект управления в виде бака с водой, к которому подходят две трубы: через одну трубу, снабженную краном, вода втекает в бак, через другую – вытекает. Подачу воды в бак можно регулировать, больше или меньше открывая кран. Расход воды является неконтролируемым и зависит от диа-

метра выходной трубы (он фиксирован) и от текущего уровня воды в баке. Если понимать под выходной (регулируемой) переменной уровень воды, а под регулирующим элементом – кран, то можно отметить, что подобный объект регулирования, с точки зрения его математического описания, является динамическим и существенно нелинейным.

Определим цель управления здесь как установление уровня воды в баке на требуемом (изменяющемся) уровне и попробуем решить соответствующую задачу управления средствами нечеткой логики.

Очевидно, в регулятор, обеспечивающий достижение цели управления, должна поступать информация о несоответствии (разности) требуемого и фактического уровней воды, при этом данный регулятор должен вырабатывать управляющий сигнал на регулирующий элемент (кран).

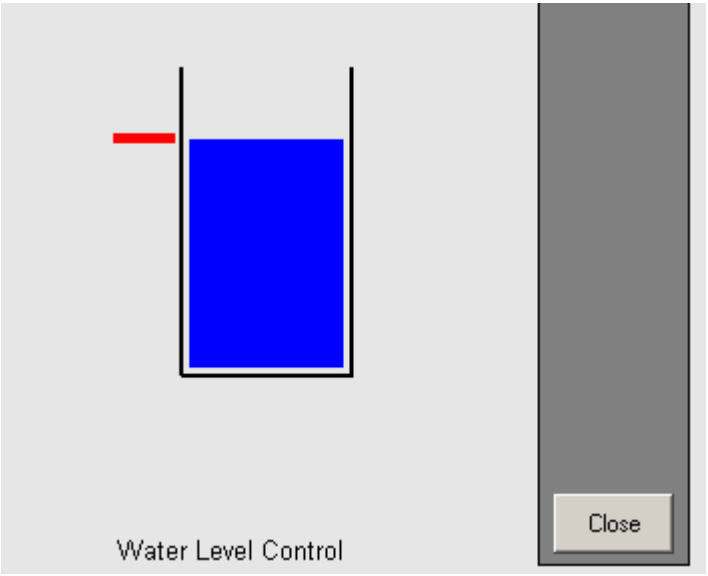


Рис. 1. Схематическое представление объекта управления (бака с водой).

В первом приближении функционирование регулятора можно описать набором из следующих правил:

1. If (level is okay) then (valve is no_change) (1)
2. If (level is low) then (valve is open_fast) (1)
3. If (level is high) then (valve is closefast) (1)

4. If (level is okay) and (rate is positive) then (valve is close_slow) (1)

5. If (level is okay and (rate is negative) then (valve is opens-low) (1),

что в переводе означает:

1. Если (уровень соответствует заданному), то (кран без изменения) (1)
2. Если (уровень низкий), то (кран быстро открыть) (1)
3. Если (уровень высокий), то (кран быстро закрыть) (1)
4. Если (уровень соответствует заданному) и (его прирост – положительный), то (кран надо медленно закрывать) (1)
5. Если (уровень соответствует заданному) и (его прирост – отрицательный), то (кран надо медленно открывать) (1)

Модель **системы управления** уровнем воды в баке с нечетким регулятором, **основанным на** приведенных правилах, является одной из **демонстрационных моделей пакетов Fuzzy Logic и Simulink.**

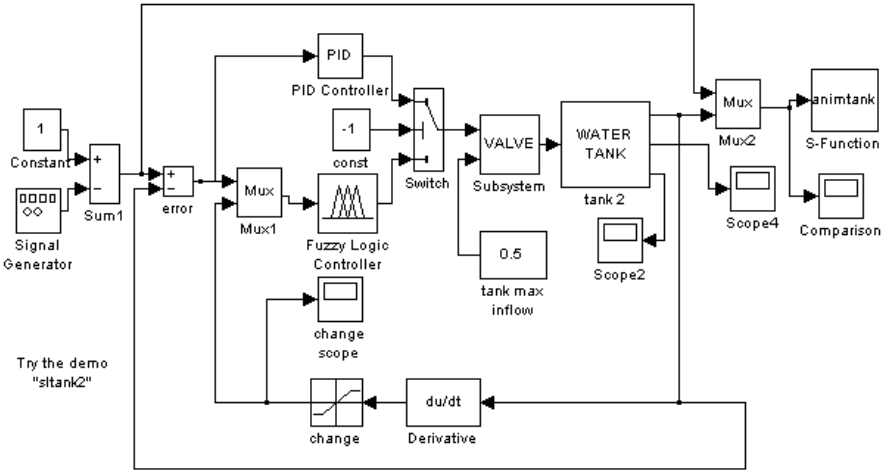


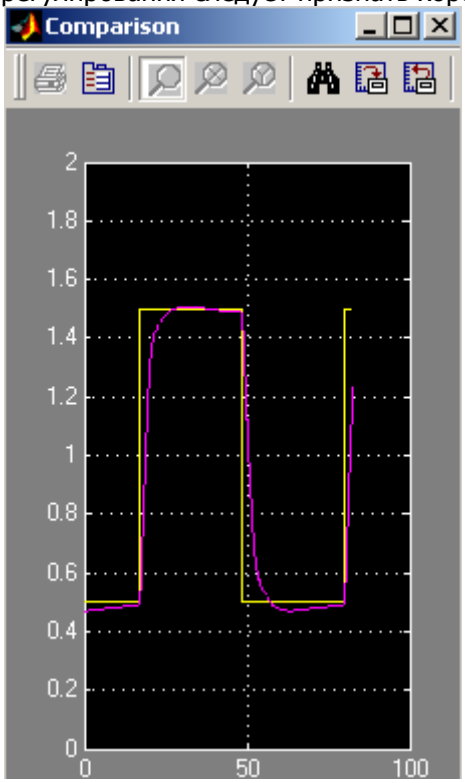
Рис. 2. Блок-диаграмма модели системы управления уровнем воды в баке нечётким регулятором

Ее можно вызвать из командной строки командой **sltank,** что приведет к открытию окна Simulink с блок-диаграммой указанной модели (рис. 2).

Одновременно блок Fuzzy Logic Controller будет сопоставлен с системой нечеткого вывода, записанной в файле tank.fis.

Для изучения процесса функционирования системы необходимо нажать кнопку **Start** панели инструментов блок-диаграммы

модели и дважды щелкнуть левой кнопкой мыши по блоку Comparison (Сравнение). В появившемся окне этого блока будут показаны изменяющиеся во времени сигналы заданного (последовательность импульсов прямоугольной формы) и фактического уровней воды (рис. 3). Как видно, переходный процесс в системе имеет апериодическую форму и заканчивается достаточно быстро, т. е. качество регулирования следует признать хорошим..



Лабораторная работа № 5
Нечеткая аппроксимирующая система

Цели и задачи работы :

Освоение графического пользовательского интерфейса FIS-редактора. Создание нечеткой системы аппроксимации функций.

Задание

1. Создать нечеткую систему типа Sugeno, отображающую таблично заданную зависимость $y = x^3$ в 5 точках:

$x_1=0$, $x_2=0.5$, $x_3=1$, $x_4=1.5$, $x_5=2$;

$y_1=0$, $y_2=0.125$, $y_3=1$, $y_4=3.375$, $y_5=8$.

2. Создать аппроксимирующую нечеткую систему типа Sugeno заданной функции двух аргументов. Параметры должны соответствовать варианту.

Варианты заданий:

$y = x_1(x_2+1)^2$, $x_1 \in [0,4]$, $x_2 \in [0,2]$;

$y = x_1(x_2-1)^2$, $x_1 \in [0,2]$, $x_2 \in [0,2]$.

Используемые операторы и команды:

fuzzy – редактор нечеткой системы вывода (FIS-редактор)

Методические указания

Графический интерфейс редактора нечеткой системы вывода запускается из режима командной строки: fuzzy.

Редактор содержит следующие меню.

File – стандартный набор операций работы с файлами (создание, сохранение, импорт, экспорт).

Edit – редактирование (включение входных и выходных переменных, функций принадлежности, правил нечеткого вывода). Доступ к редактированию может быть также осуществлен щелчком мыши по соответствующей пиктограмме основного окна редактора.

View – визуализация работы системы нечеткого вывода.

С помощью программы-редактора на любом этапе проектирования нечеткой системы в нее можно вносить необходимые коррективы.

Для выполнения задания п.1 нужно выполнить следующие операции. В меню File выбрать New FIS/Sugeno. Выделить щелчком левой кнопки мыши блок input1. В поле Name вместо input1 ввести обозначение аргумента: x. Двойным щелчком по блоку x вызвать окно редактора функций принадлежности – Membership Function Editor. В окне графиков функций принадлежности удалить с помощью выделения мышью и клавиши Delete заданные по умолчанию функции принадлежности. Из меню Edit/Add MFs вызывать диалоговое окно, позволяющее задать тип (MF type) и количество (Number of MFs) функций принадлежности. Выбрать функции принадлежности вида гаусса (gaussmf) в количестве 5. Для выбранных кривых задать имена в поле Name, например, x0, …, x5. Диапазон (Range) – [0, 2]. Нажать кнопку Close.

Выделить щелчком левой кнопки мыши блок output1 и дать в поле Name имя y. Дважды щелкнуть мышью по этому блоку. В

появившемся диалоговом окне в качестве функций принадлежности выбрать постоянные функции принадлежности. Из меню Edit/Add MFs добавить еще 2 функции, доведя их общее число до 5. Задать диапазон (Range): [0, 8]. Изменить имена функций принадлежности на y1,…, y5. Одновременно соответствующие числовые значения y ввести в поле Params. Закрыть окно.

Дважды щелкнуть левой кнопкой мыши по среднему блоку, что приведет к вызову редактора правил (Rule Editor). При вводе каждого правила необходимо выбрать соответствующие поля имен переменных x и y. После нажатия кнопки Add rule введенное правило появится в верхнем окне. Закрыть окно редактора правил. Сохранить созданную аппроксимирующую систему на диске, используя пункт меню File/Export/to Disk под выбранным именем.

Выбрать позицию меню View/Rules. Изменить численное значение входной переменной можно перемещая с помощью курсора красную вертикальную черту или задавая числовое значение в поле Input. Например, для x=1.3 результат аппроксимации 2.53, точное значение равно 2.20. Уровень погрешности для созданной простой нечеткой системы аппроксимации является допустимым. Выбрать пункт меню View/Surface и перейти к окну просмотра кривой y.

Задание п.2 выполняется аналогичным образом, но для двух входов, которые соответствуют двум переменным заданной функции.

Лабораторная работа № 6
Проектирование нечеткой модели управления

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью выполнения лабораторной работы является, получение практических навыков построения нечеткой модели управления контейнерным краном.

Задачи, решаемые в данной работе:

- Получение навыков работы в FIS редакторе программного пакета MATLAB
- Получение трехмерного графика зависимости мощности двигателя, расстояния до цели и угла отклонения троса от вертикали.

-Провести анализ работы модели в соответствии с вариантом задания.

ХОД ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Получить задание у преподавателя из таблицы 1.
2. Разобраться, как строиться нечеткая база правил в соответствии со следующими условиями:

Решение, которое должно быть получено, относится к области управления контейнер краном и предназначено для предотвращения раскачивания груза при подводе его к месту выгрузки.

Контейнерные краны используются при выполнении погрузочно-разгрузочных работ в портах. Кран, соединяется с контейнером тросом и кабина крана вместе с контейнером, перемещается по рельсам в горизонтальном направлении. Когда кран приходит в движение, контейнер начинает раскачиваться и отклоняться от положения равновесия и не находится строго под кабиной крана.

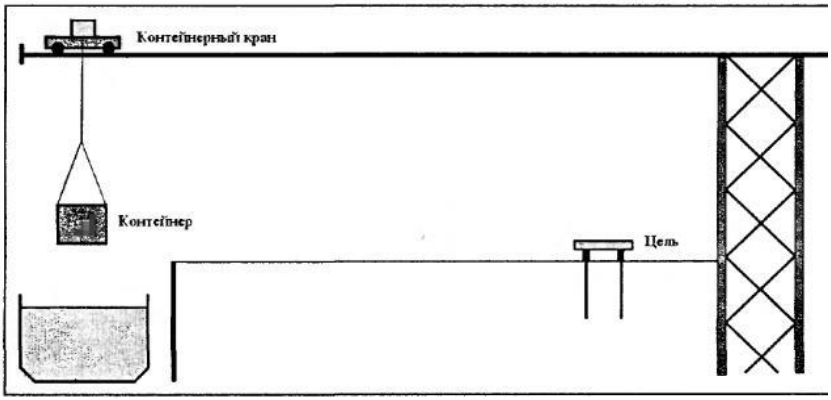


Рис. 1. Схема рабочей зоны контейнерного крана.

Проблема заключается в том, что пока контейнер качается при транспортировке и трос отклоняется от вертикали, контейнер не может быть опущен на погрузочную площадку, в качестве которой используются железнодорожные платформы или другие транспортные средства.

Анализ действий крановщиков операторов, выполняющих управление краном, показывает, что они при работе применяют

Следующие эвристические правила:

1. Движение следует начинать со средней скоростью.
2. Если осуществляется движение и кабина находится дале-ко от места выгрузки, то следует отрегулировать мощность двига-теля так, чтобы контейнер оказался несколько позади ка-бины крана.
3. Если кабина находится не далеко от места выгрузки, то следует уменьшить скорость движения таким образом, что-бы кон-тейнер находился немного впереди кабины крана.
4. Если контейнер находится очень близко к месту выгруз-ки, следует отключить двигатель, чтоб движение продол-жалось только по инерции.
5. Если контейнер находится прямо над позицией це-ли, то следует остановить двигатель и прекратить движение.

Построим базу правил:

• этой целью, преобразуем, рассмотренные выше, 5 эври-стических правил в 6 правилах нечетких продукций:

ПРАВИЛО1: ЕСЛИ "расстояние далекое" И "угол равен нулю "ТО "мощность положительная средняя"

ПРАВИЛО2: ЕСЛИ "расстояние далекое" И "угол отрица-тельный малый "ТО"мощность положительная большая"

ПРАВИЛО3: ЕСЛИ "расстояние далекое" И "угол отрица-тельный большой "ТО" мощность положительная средняя"

ПРАВИЛО 4: ЕСЛИ "расстояние среднее" И "угол отрица-тельный малый "ТО" мощность отрицательная средняя"

ПРАВИЛО5: ЕСЛИ "расстояние близкое" И "угол положи-тельный малый "ТО "мощность положительная средняя"

ПРАВИЛО6: ЕСЛИ "расстояние ноль" И "угол равен нулю "ТО "мощность равна нулю"

• Требуется спроектировать нечеткую систему, выполнив следующую последовательность шагов:

Шаг 1. В программе Matlab откроем **FIS–редактор**, напе-чатав слово **fuzzy** в командной строке.

Шаг 2. В появившемся графическом окне **FIS Editor** вводим вторую входную переменную. Для этого выбираем команду **Addinput** в меню **Edit**.

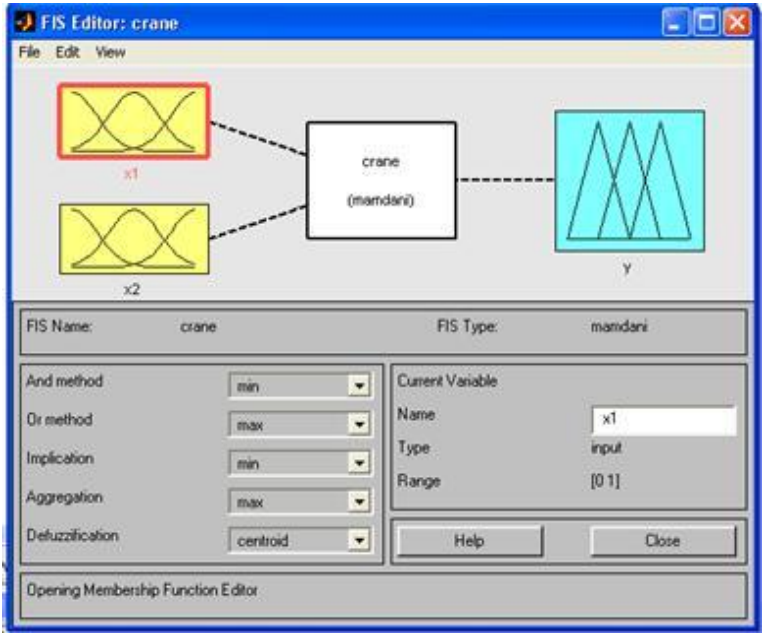


Рис. 2. FIS–редактор с двумя входными переменными.

Шаг 3. Переименуем первую входную переменную. Для этого сделаем щелчок левой кнопкой мыши на блоке input1, введем новое обозначение x1 в поле редактирования имени текущей переменной и нажмем <Enter>.

Шаг 4. Переименуем вторую входную переменную, введя x2 на блоке input2.

Шаг 5. Переименуем выходную переменную. Для этого щелкнем мышкой на блоке output1. Введем новое обозначение y поле редактирования имени текущей переменной; нажмем <Enter>.

Шаг 6. Зададим имя системы. Для этого в меню File выберем в подменю Export команду ToFile и введем имя файла, например, Crane.

Шаг 7. Перейдем в редактор функций принадлежности. Для этого сделаем двойной щелчок левой кнопкой мыши на блоке **x1**

- зададим диапазон изменения переменной **x1**, введя значения (0 45) в поле Range (рис. 3).

Шаг 8. Зададим функции принадлежности переменной **x1**. Для лингвистической оценки этой переменной будем использо-

вать четыре терма с треугольными функциями принадлежности. Для этого зайдем в меню **Edit** выберем команду **RemoveAllMFs** для удаления установленных по умолчанию функций принадлежности. После этого в меню **Edit** выберем команду **AddMFs...**

В появившемся диалоговом окне выберем четыре терма в поле **NumberMFs**. После ввода функций принадлежности введем для каждой функции диапазон изменений (в зависимости от выбранных нами правил расстояния).

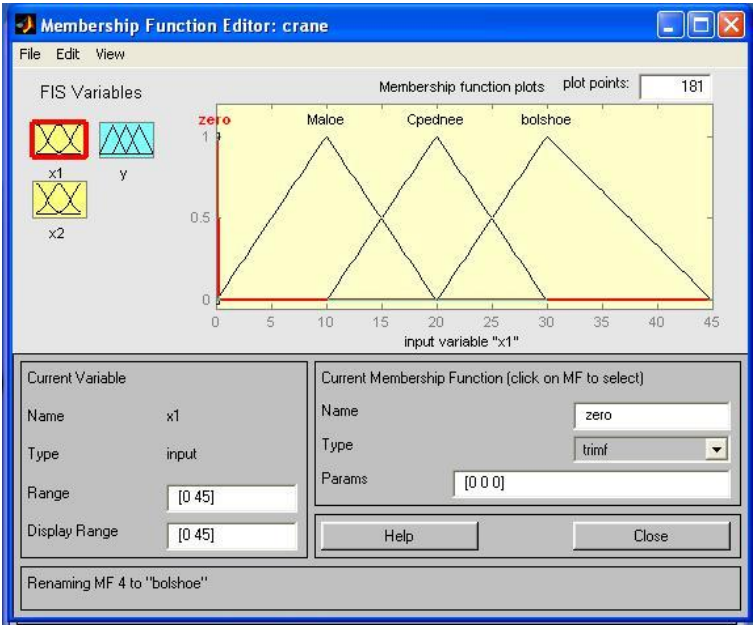


Рис. 3. Функция принадлежности переменной x1.

Шаг 9. Зададим наименования термов переменной **x1**. Для этого щелкнем мышкой по графику первой функции принадлежности (см. рис. 3). График активной функции принадлежности выделяется красной жирной линией. Затем введем наименование терма Ноль (Zero) в поле **Name** и нажмем **<Enter>**. Щелкнем мышкой по графику второй функции принадлежности, введем наименование терма Малое (Maloe) в поле **Name** и нажмем **<Enter>**. Щелкнем мышкой по графику третьей функции принадлежности, введем наименование терма Среднее (Cpednee) в поле **Name** и нажмем **<Enter>**. Щелкнем мышкой по графику

четвертой функции принадлежности, введем наименование терма Большое (bolshoe) в поле **Name** и нажмем **<Enter>**.

Шаг 10. Зададим функции принадлежности переменной **x2**. Для этого активизируем переменную **x2** щелчком мышкой по блоку **x2**. Зададим диапазон изменения переменной **x2**. Для этого введем значения (-15 5) в поле **Range** и нажмем **<Enter>**. Для лингвистической оценки этой переменной будем использовать, как и ранее четыре терма с треугольными функциями принадлежности. Задаем эти функции так же как и для функции **x1**, после чего переходим к следующему шагу.

Шаг 11. По аналогии с шагом 9 зададим следующие наименования термов переменной **x2**: Отрицательный большой (- bolshoi), Отрицательный малый (- Malii), Ноль (Zero), Положительный малый (+ Malii).

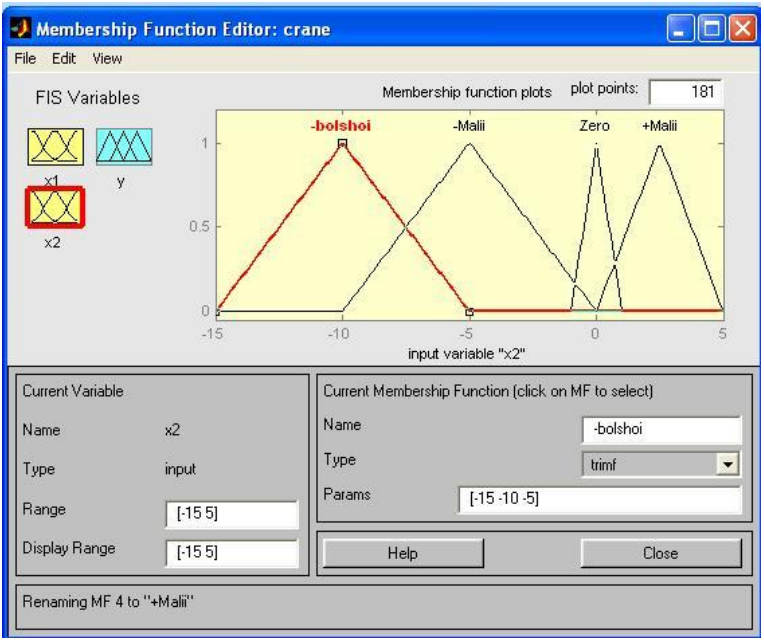


Рис.4. Функция принадлежности x2.

Шаг 12. Зададим функции принадлежности переменной **y**. Для этого щелчком мыши по блоку **y** активизируем переменную **y**. Зададим диапазон изменения переменной **y**. Для этого введем значения (-10 10) в поле **Range** (рис. 5) и нажмем **<Enter>**. Для лингвистической оценки этой переменной будем использовать

пять термов с треугольными функциями принадлежности. Добавляем термы так же, как и в предыдущем случае.

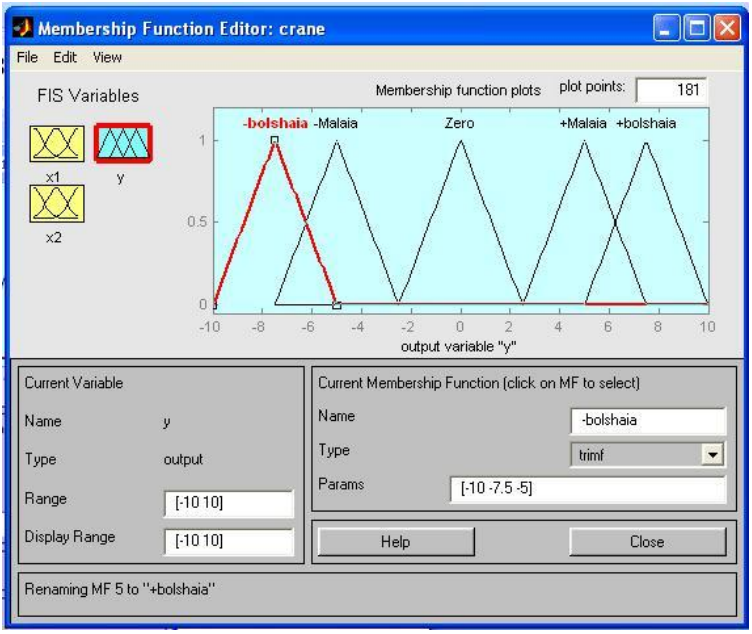


Рис.5. Функция принадлежности переменной y.

Шаг 13. По аналогии с шагом 9, зададим следующие наименования термов переменной **y**: Отрицательная большая (-bolshaia), Отрицательная малая (-Malaia), Ноль (Zero), Положительная малая (+ Malaia), Положительная большая (+ bolshaia). В результате получим графическое окно, изображенное на рис. 5.

Шаг 14. Перейдем в редактор базы знаний **RuleEditor** (Рис.6). Для этого в меню **Edit** выберем команду **Rules..**. Для ввода правила выбираем в меню соответствующую комбинацию термов и нажимаем кнопку **Addrule**.

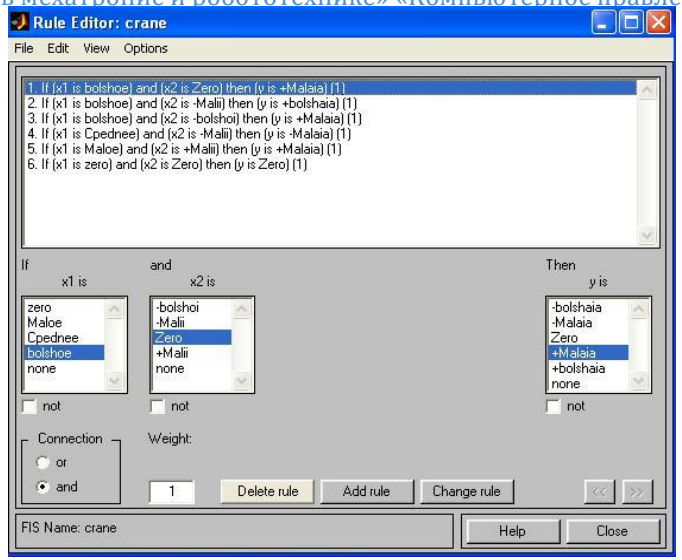


Рис.6. Редактор базы правил.

Шаг 15. Сохраним созданную систему. Для этого в меню **File** выберем в подменю **Export** команду **To File**.

Шаг 16. Откроем окно визуализации нечеткого вывода. Оно активизируется командой **Rules** меню **View** (рис. 7). В поле **Input** указываются ***значения входных переменных (Таблица 1)***, для которых выполняется нечеткий логический вывод.

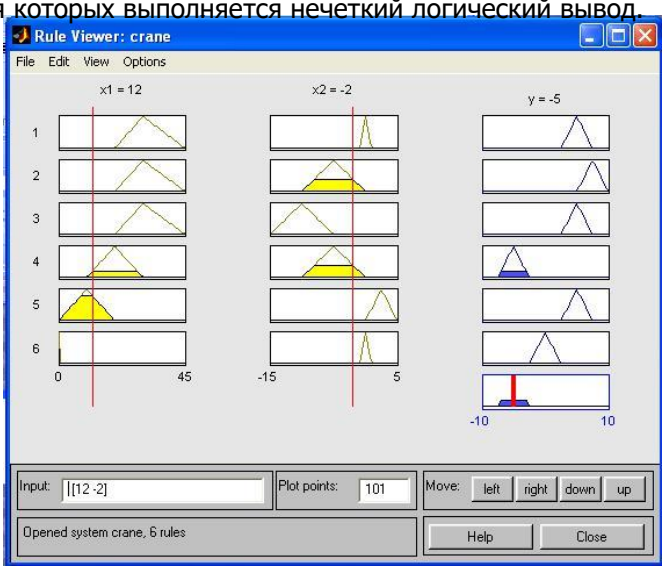


Рис. 7. Визуализация нечеткого вывода Мамдани.

Шаг 17. Выведем изображение поверхности «входы - выход», соответствующей нашей синтезированной нечеткой системе. Окно выводится по команде **Surface** меню **View** (рис.8). При сравнении полученной нами поверхности и составленной базы правил можно сделать выводы относительно качества описания нечеткими правилами моделируемой системы.

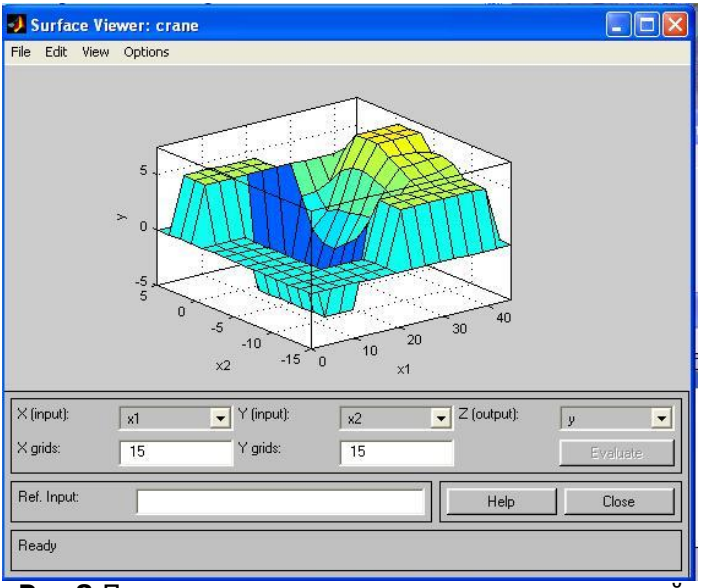


Рис.8 Поверхность «входы-выход» синтезированной нечеткой системы.

Варианты заданий

Вариант №	Расстояние до цели	Угол между контейнером и кабиной крана
1	18	-13
2	22	-7
3	5	3
4	40	-2
5	28	-10
6	3	1
7	12	0
8	27	-8
9	16	-6
10	8	4

ЧАСТЬ 5 НЕЙРОННЫЕ СЕТИ

Лабораторная работа №1

Цель работы

Изучить нейронные сети и системы управления в MATLAB. Обучить регулятор на основе нейронной сети.

Теоретические положения

История нейронных сетей

На заре развития электронно-вычислительной техники в середине XX века среди ученых и конструкторов еще не существовало единого мнения о том, как должна быть реализована и по какому принципу работать типовая электронно-вычислительная машина. Это сейчас мы с Вами изучаем в курсах Основ информатики архитектуру машины фон Неймана, по которой построены практически все существующие сегодня компьютеры. При этом в тех же учебниках ни слова не говорится о том, что в те же годы были предложены принципиально иные архитектуры и принципы действия компьютеров. Одна из таких схем получила название нейросетевого компьютера, или просто нейросети. Первый интерес к нейросетям был обусловлен пионерской работой МакКаллока и Питса, изданной в 1943 году, где предлагалась схема компьютера, основанного на аналогии с работой человеческого мозга. Они создали упрощенную модель нервной клетки – нейрон. Мозг человека состоит из белого и серого веществ: белое – это тела нейронов, а серое – это соединительная ткань между нейронами, или аксоны и дендриты. Мозг состоит примерно из 1011 нейронов, связанных между собой. Каждый нейрон получает информацию через свои дендриты, а передает ее дальше только через единственных аксон, разветвляющийся на конце на тысячи синапсов (рисунок 1).

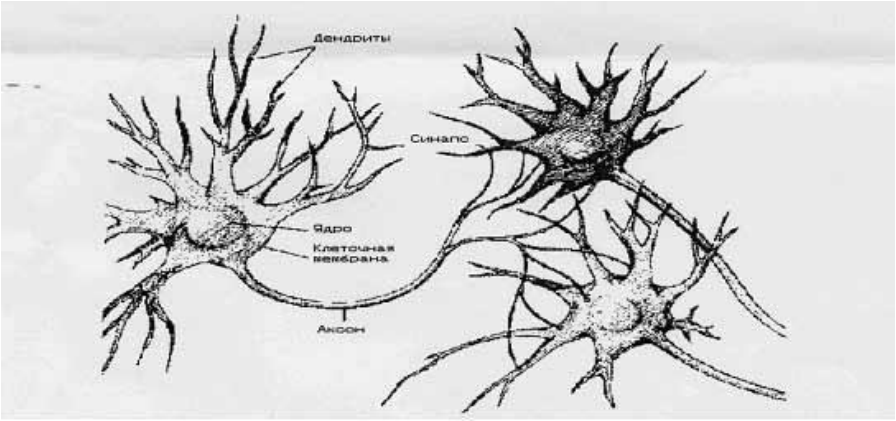


Рисунок 1- Структура нейрона

Простейший нейрон может иметь до 10000 дендритов, принимающих сигналы от других клеток. Таким образом, мозг содержит примерно 1015 взаимосвязей. Если учесть, что любой нейрофизиологический процесс активизирует сразу множество нейронов, то можно представить себе то количество информации или сигналов, которое возникает в мозгу.

Нейроны взаимодействуют посредством серий импульсов, длящихся несколько миллисекунд, каждый импульс представляет собой частотный сигнал с частотой от нескольких единиц до сотен герц. Это невообразимо медленно по сравнению с современными компьютерами, но в тоже время человеческий мозг гораздо быстрее машины может обрабатывать аналоговую информацию, както: узнавать изображения, чувствовать вкус, узнавать звуки, читать чужой почерк, оперировать качественными параметрами. Все это реализуется посредством сети нейронов, соединенных между собой синапсами. Другими словами, мозг -–это система из параллельных процессоров, работающая гораздо эффективнее, чем популярные сейчас последовательные вычисления. Кстати говоря, недавно в одном из журналов я читал, что технология последовательных вычислений подошла к пределу своих технических возможностей, и в настоящее время остро стоит проблема развития методов параллельного программирования и создания параллельных компьютеров. Так что, может быть, нейросети являются только очередным шагом в этом направлении.

Нейронная сеть

Нейросеть (искусственная нейронная сеть - ИНС или просто нейронная сеть) - набор нейронов соединенных между собой

дендритами и аксонами. В искусственно созданных нейросетях внутренние функции у нейронов (еще их называют передаточными) фиксированы, а все параметры определяются значениями весов синапсов.

Входы и выходы нейронов могут быть входами сети, выходами сети или внутренними волокнами.

Процесс работы сети заключается в подаче определенного набора чисел на входы сети и получения закономерного набора на выходе. Для удобства совокупность всех параметров называют вектором. Таким образом при подаче определенного вектора на вход нейросеть выдает закономерный вектор на выходе.

Ход работы

Ниже описаны 3 архитектуры нейронных сетей, которые реализованы в ППП Neural Network Toolbox в виде следующих контроллеров:

- контроллер с предсказанием (NN Predictive Contrloller);
- контроллера на основе модели авторегрессии со скользящим средним (NARMA-L2 Controller);
- контроллера на основе эталонной модели (Model Reference Controller).

Ниже представлено краткое описание каждой из этих архитектур и способы их применения. Применение нейронных сетей для решения задач управления позволяет выделить два этапа проектирования:

- этап идентификации
- этап синтеза закона управления.

На этапе идентификации разрабатывается модель управляемого процесса в виде нейронной сети, которая на этапе синтеза используется для синтеза регулятора. Для каждой из трех архитектур используется одна и та жу 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 автогрессии со скользящим средним регулятор представляет собой достаточно простую рекострукцию 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.

Динамические модели систем управления с нейросетевыми регуляторами размещены в специальном разделе Control System набора блоков NN Blocksets и включает три упомянутые выше модели регуляторов, а также блок построения графиков.

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

РЕГУЛЯТОР С ПРЕДСКАЗАНИЕМ. Этот регулятор использует модель управляемого процесса в виде нейронной сети, для того чтобы предсказать будущие реакции процесса на случайные сигналы управления. Алгоритм оптимизации вычисляет управляющие сигналы, которые минимизируют разность между желаемыми и действительными изменениями сигнала на выходе модели и таким образом оптимизируют управляемый процесс, построение модели управляемого процесса выполняется автономно с использованием нейронной сети, которая обучается в групповом режиме с использованием одного из алгоритмов обучения. Контроллер, реализующий такой регулятор, требует значительного объема вычислений, поскольку для расчета оптимального закона управления оптимизация выполняется на каждом такте управления.

РЕГУЛЯТОР NARMA – L2. Изо всех архитектур этот регулятор требует наименьшего объема вычислений. Данный регулятор – это просто некоторая реконструкция нейросетевой модели управляемого процесса, полученной на этапе 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. 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. Достоинством регуляторов на основе эталонной модели является, то, что они применимы к различным классам управляемых процессов.

Система управления с регулятором на основе эталонной модели

Цель обучения регулятора состоит в том, чтобы движение звена отслеживало выход эталонной модели:

$$\frac{d^2 y_r}{dt^2} = -9y_r - 6\frac{dy_r}{dt} + 9r,$$

где y_r - выход эталонной модели;

r – задающий сигнал на входе модели.

Структурная схема, поясняющая принцип построения системы управления с эталонной моделью показана на рисунке 2.

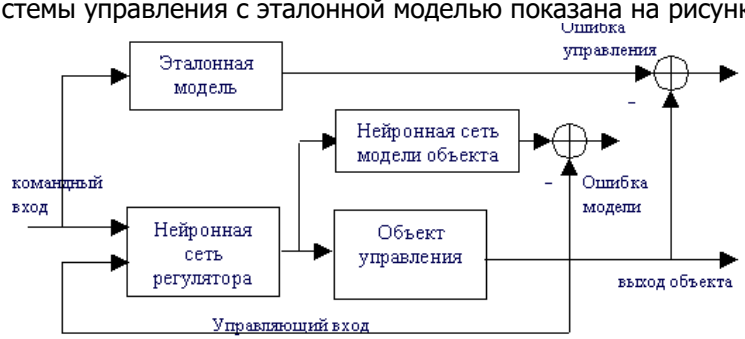


Рисунок 2 - Структурная схема

Соответствующая динамическая модель, реализованная в Simulink, показана на рисунке 3.

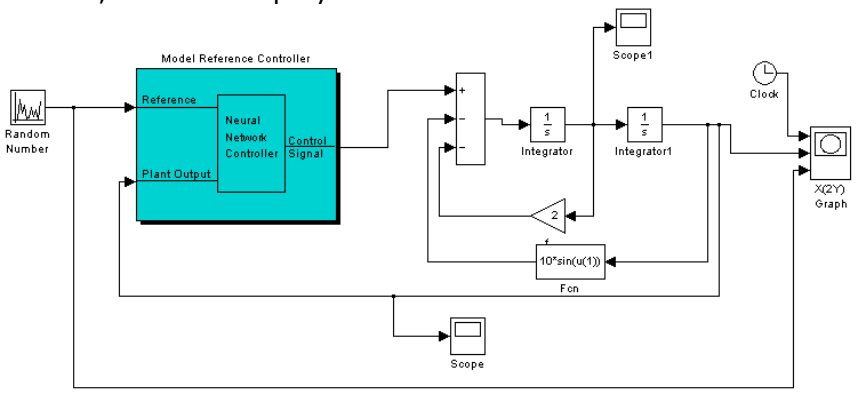


Рисунок 3 - Динамическая модель

Для того, чтобы начать работу необходимо активизировать блок нейросетевого регулятора двойным щелчком мыши. Появится окно, показанное на рисунке 4.

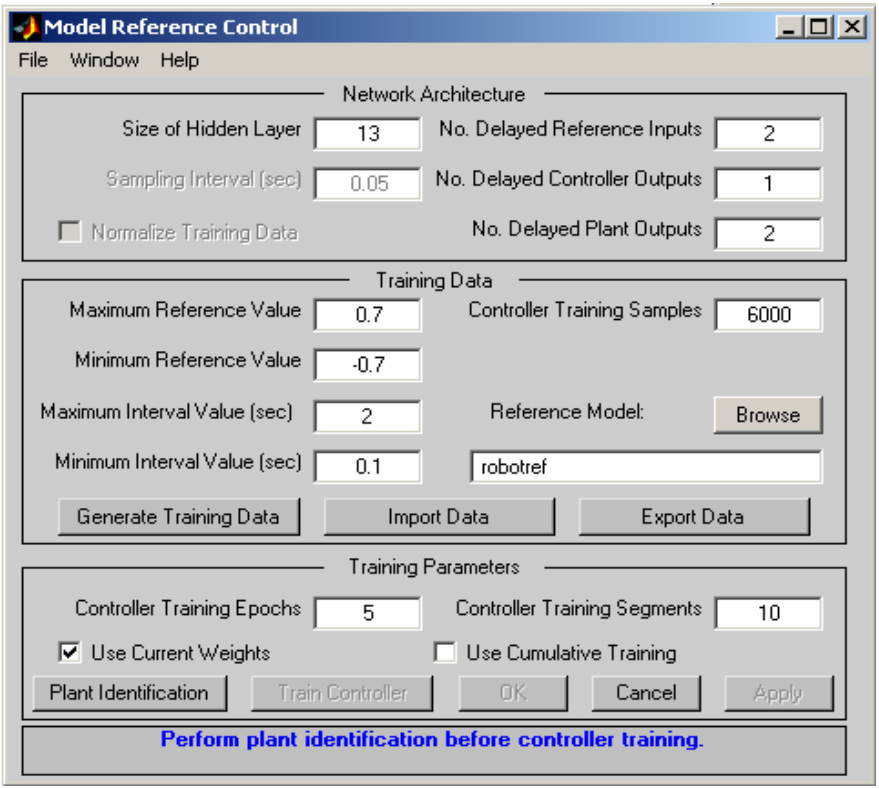


Рисунок 4 - Динамическая модель

Оно выполняет функции графического интерфейса пользователя. Прежде чем установить параметры контроллера, необходимо построить модель управляемого процесса. Это означает, что прежде всего необходимо выполнить идентификацию управляемого процесса, т.е. построить его нейросетевую модель, воспользовавшись специальной процедурой Plant Identification.

Вид окна Plant Identification приведен на рисунке 7.

Это окно универсальное и может быть использовано для построения нейросетевых моделей для любого динамического объекта, который описан моделью Simulink. Для системы управления движением звена манипулятора динамическая модель, реализованная в Simulink и удовлетворяющая уравнению движения звена будет выглядеть, как показано на рисунке 5.

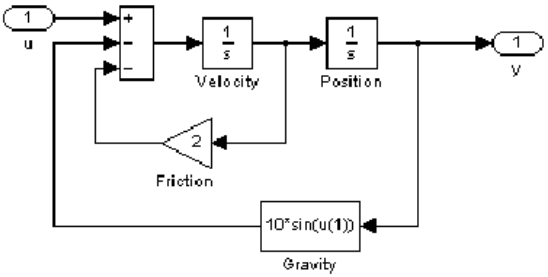


Рисунок 5 - Динамическая модель в Simulink

Процедура идентификации позволяет построить нейронную сеть, которая будет моделировать динамику управляемого процесса. Если модель должна использоваться при настройке контроллера, то её следует создать прежде, чем начнется расчет контроллера. Кроме того, вам может потребоваться создание новой модели объекта, если спроектированный контроллер будет функционировать неудовлетворительно.

Процедура идентификации требует задания следующих параметров:

size of the hidden layer – размер скрытого слоя определяется количеством используемых нейронов;

sampling interval – такт дискретности в секундах определяет интервал между двумя последовательными моментами съема данных;

No. delayed plant inputs – количество элементов запаздывания на входе модели;

No delayed plant outputs – количество элементов запаздывания на выходе модели;

normalize trainig data – окно контроля нормирования обучающих данных к диапозону [0 1] ;

trainig samples – длина обучающей выборки (количество точек съема информации) ;

maximum plant input – максимальное значение входного сигнала;

minimum plant input – минимальное значение входного сигнала;

maximum interval value – максимальный интервал идентификации в секундах;

minimum interval value – минимальный интервал идентификации в секундах;

limit output data – окно контроля, позволяющее ограничить объем выходных данных; только при включенном окне контроля будут доступны два следующих окна редактирования текста;

maximum plant output - максимальное значение выходного сигнала;

minimum plant output - минимальное значение выходного сигнала;

simulink plant model – задание модели Simulink с указанием входных и выходных портов, используемых для построения нейросетевой модели управляемого процесса;

generate trainig data – кнопка запуска процесса генерации обучающей последовательности;

import data – импорт обучающей последовательности из рабочей области или файла данных;

export data – экспорт сгенерированных данных в рабочую область или файл;

trainig epochs – количество циклов обучения;

trainig function – задание обучающей функции;

use current weights – окно контроля , позволяющее подтвердить использование текущих весов нейронной сети;

use validation/testing for training – выбор этих окон контроля будет означать, что 25 процентов данных из обучающей последовательности будет использовано для формирования тестового и контрольного подмножеств соответственно.

Итак, выбор процедуры generate trainig data приведет к тому, что будет запущена программа генерации обучающей последовательности. Программа генерирует обучающие данные путем воздействия случайных ступенчатых воздействий на модель Simulink управляемого процесса. Графики входного и выходного сигналов объекта управления выводятся на экран (рисунок 6).

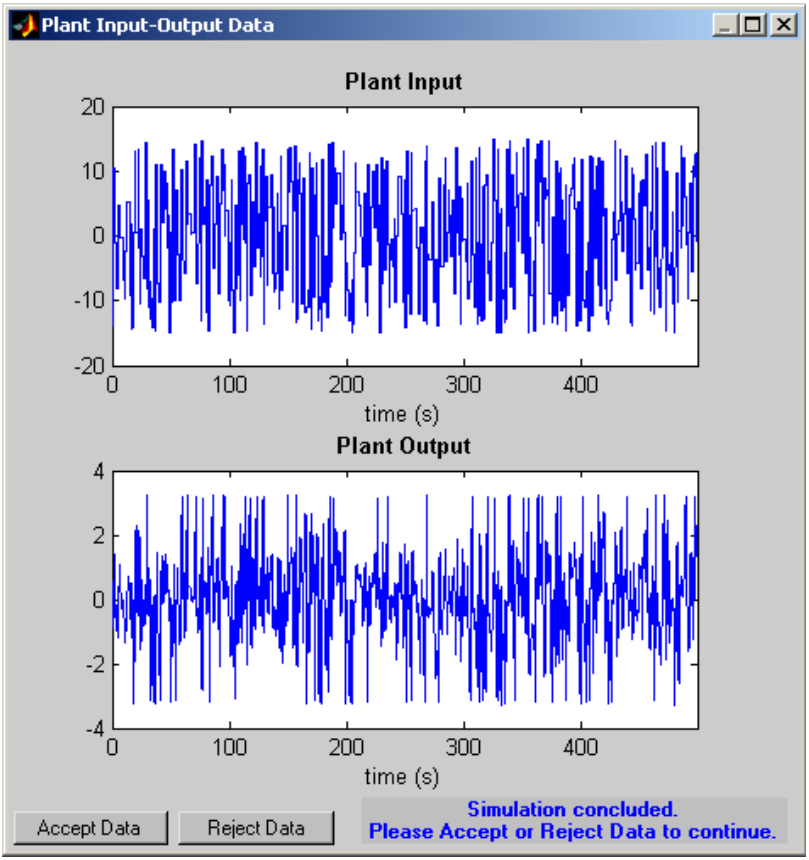


Рисунок 6 - Графики входного и выходного сигналов объекта управления

По завершении генерации обучающей последовательности пользователю предлагается принять сгенерированные данные (Accept Data), либо отказаться от них (Reject Data).

Если вы принимаете данные, приложение возвращает вас к несколько измененному окну Plant Indentification (рисунок 7).

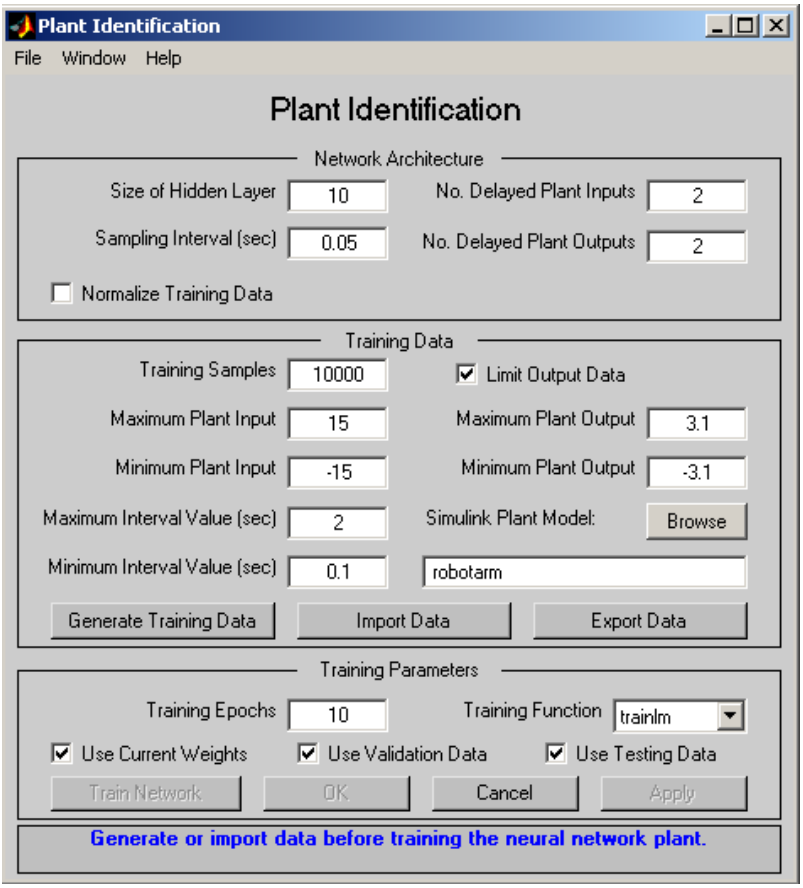


Рисунок 7 - Окно Plant Identification

Здесь часть окон недоступны, а кнопка Generate Trainig Data заменена на кнопку Erase Generated Data, что позволяет удалить сгенерированные данные.

В окне фрейма содержится сообщение «Обучающая последовательность состоит из 10000 замеров. Можно начинать обучение нейронной сети.

Для этого следует воспользоваться кнопкой Train Network (Обучить сеть). Начнется обучение нейросетевой модели. После завершения обучения его результаты отображаются на графиках, как это показано на рисунке 8.

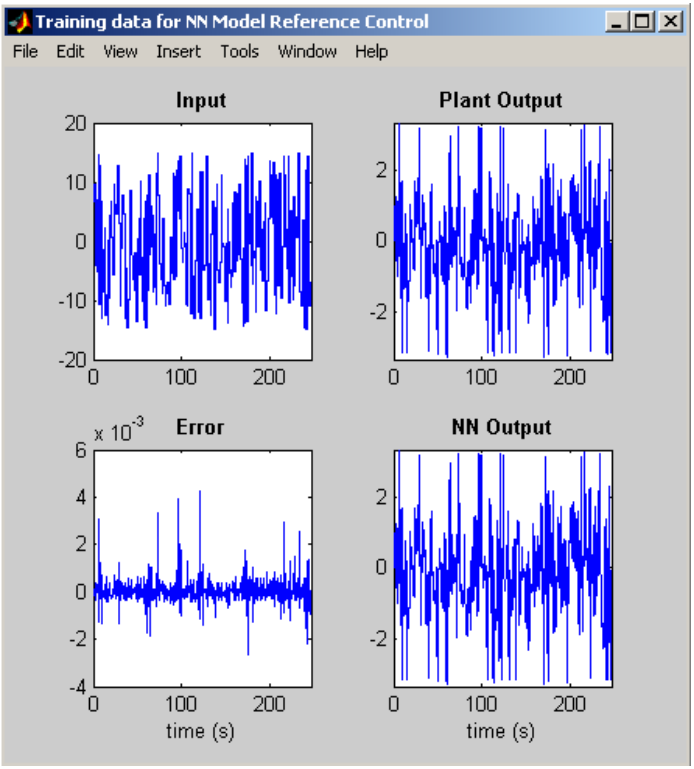


Рисунок 8 - Окно Train data for NN Model Reference Control

Где построены результаты обучения и тестирования на контрольном подмножестве.

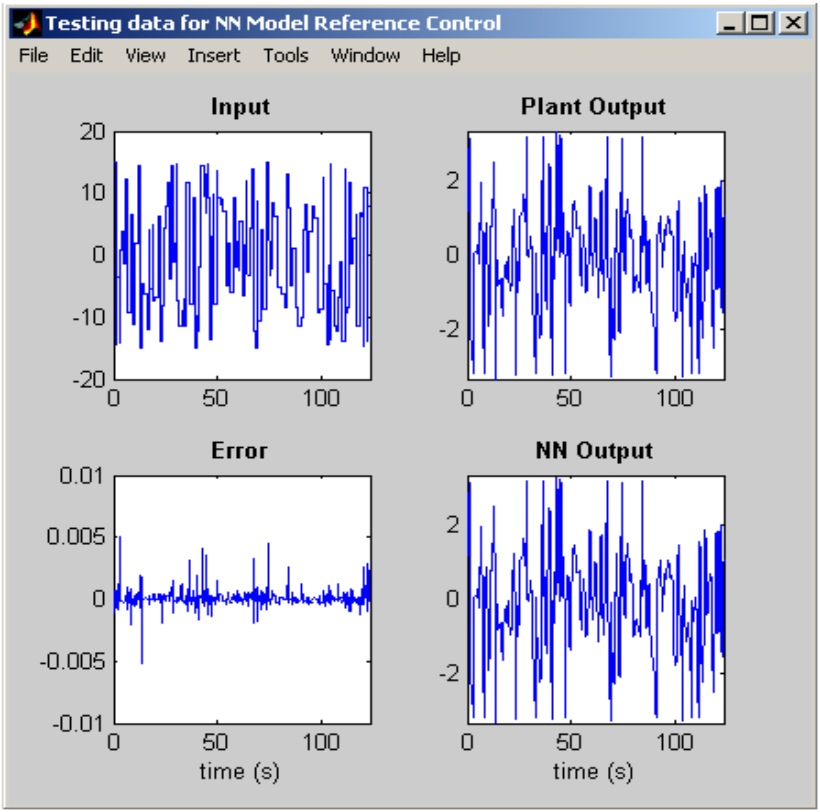


Рисунок 9 - Окно Testing data for NN Model Reference Control

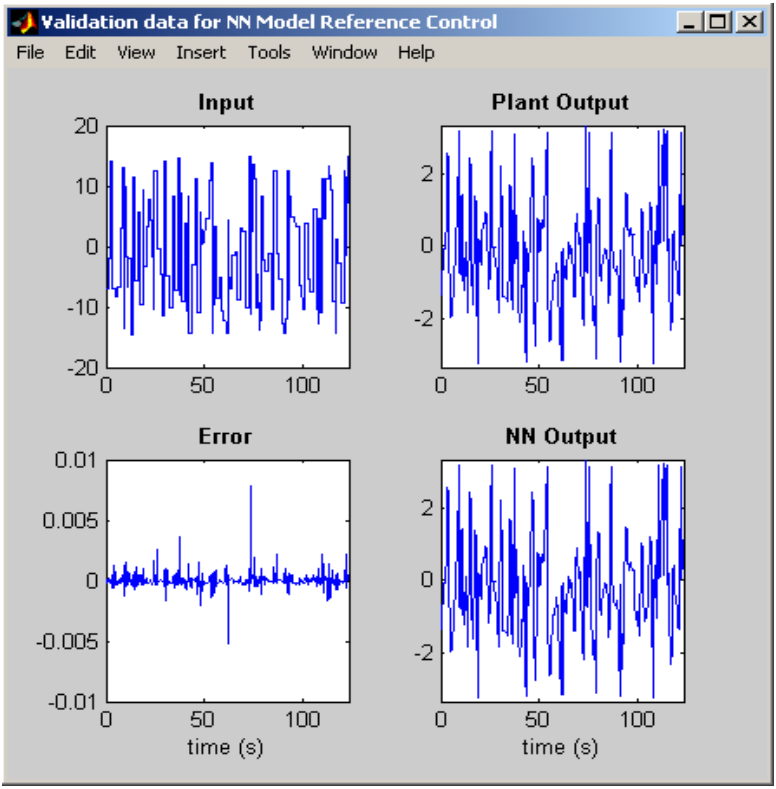


Рисунок 10 - Окно Validation data for NN Model Reference Control

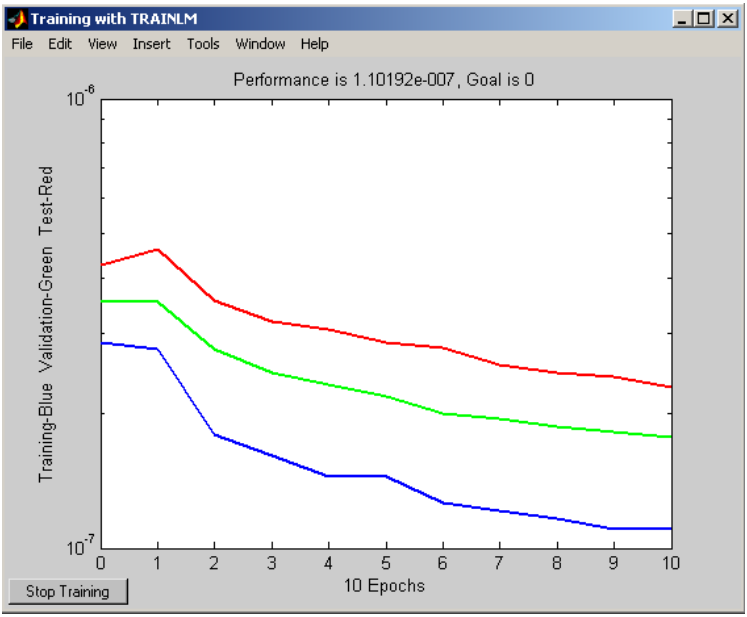


Рисунок 11 - Окно Training with TRAINLM

Текущее состояние отмечено в окне Plant Identification сообщением «Обучение завершено. Мы можем сгенерировать или импортировать новые данные, продолжить обучение или сохранить полученные результаты, выбрав кнопки Ок или Apply.

Лабораторная работа № 2
Использование Simulink при построении нейронных сетей.

Целью работы является изучение основных блоков визуального приложения Simulink программной среде MATLAB 6.0, используемых при построении нейронных сетей, приобретение опыта моделирования нейросетевых систем управления среды MATLAB 6.0.

Теоретическая часть

Библиотека блоков построения нейронных сетей в Simulink.

Пакет Neural Network Toolbox содержит ряд блоков, которые могут быть непосредственно использованы для построения нейронных сетей в среде Simulink, либо применяться вместе с функцией gensim [1].

Для вызова набора блоков, в командной строке необходимо набрать команду neural, после выполнения которой появляется окно вида рис. 1.

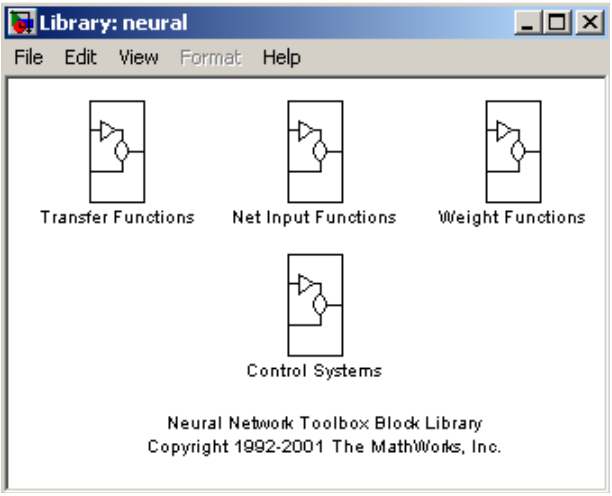


Рис. 1. Основные нейросетевые блоки Simulink

Каждый из представленных на рис. 1 блоков в свою очередь является набором (библиотекой) некоторых блоков.

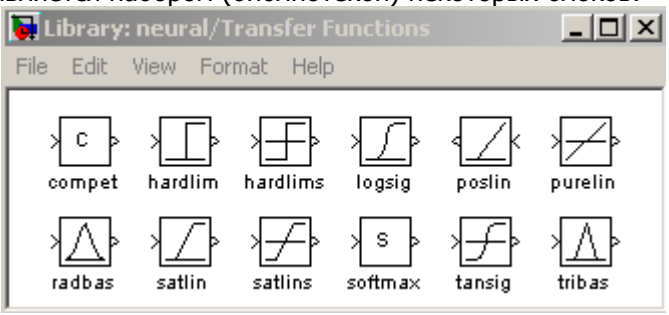


Рис. 2. Библиотека функций активации

Блоки функции активации (Transfer Functions). Двойной щелчок кнопки мыши на блоке Transfer Functions приводит к появлению библиотеки функций активации (рис. 2). Каждый из этих блоков данной библиотеки преобразует подаваемый на него вектор в соответствующий вектор той же размерности (табл. 1).

Таблица 1. Перечень функций активации нейронов

Название	Формула	Область значений
Пороговая	$f(x)=\begin{cases}0, & s<\theta,\\ 1, & s\geq\theta\end{cases}$	0,1
Знаковая (сигнатурная)	$f(x)=\begin{cases}1, & s>\theta,\\ -1, & s\leq\theta\end{cases}$	-1,1
Сигмоидальная (логистиничнская)	$f(x)=\frac{1}{1+e^{-s}}$	(0,1)
Полулинейная	$f(x)=\begin{cases}s, & s>\theta,\\ 0, & s\leq\theta\end{cases}$	$(0,\infty)$
Линейная	$f(x)=s$	$(-\infty,\infty)$
Радиальная базисная (гауссова)	$f(x)=e^{-s^2}$	(0,1)
Полулинейная с насыщением	$f(x)=\begin{cases}0, & s\leq 0,\\ s, & 0<s<1,\\ 1, & s\geq 1\end{cases}$	(0,1)
Линейная с насыщением	$f(x)=\begin{cases}-1, & s\leq -1,\\ s, & -1<s<1,\\ 1, & s\geq 1\end{cases}$	(-1,1)

Гиперболический тангенс (сигмоидальная)	$f(x)=\frac{e^{s}-e^{-s}}{e^{s}+e^{-s}}$	(-1,1)
Треугольная	$f(x)=\begin{cases}1-\lvert s\rvert, & \lvert s\rvert\le\theta,\\ 1, & \lvert s\rvert>1\end{cases}$	(0,1)

Блоки преобразования входов сети. Проводя аналогичную рассмотренной операцию, но с блоком Net Input Functions, придём к библиотеке блоков вида, представленного на рис. 3 [1].

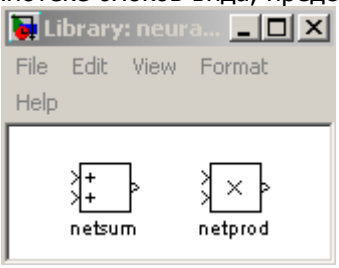


Рис. 3. Библиотека блоков преобразований сигналов

Блоки данной библиотеки реализуют рассмотренные выше функции преобразования входов сети.

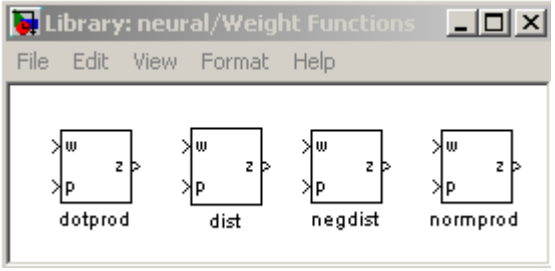


Рис. 4. Библиотека блоков весовых коэффициентов

Блоки весовых коэффициентов. Точно так же (но щелкая левой кнопкой мыши по иконке с надписью Weight Functions) придем к библиотеке блоков (рис. 4), реализующих некоторые функции весов и расстояний.

Отметим, что при задании конкретных числовых значений, при операции со всеми приведенными блоками ввиду особенностей Simulink векторы необходимо представлять как столбцы, а не как строки (как это было до сих пор).

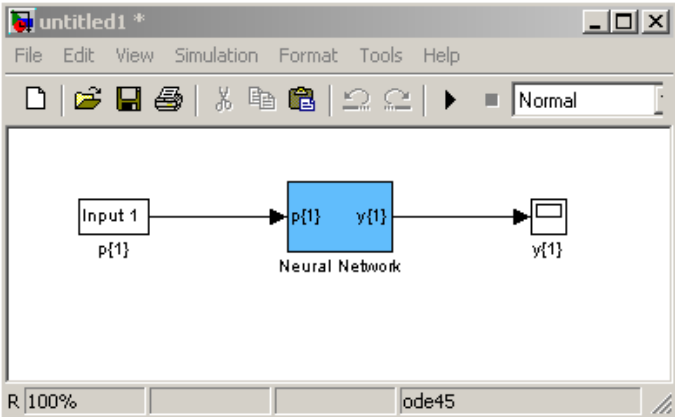


Рис. 5. Созданная нейросетевая модель Simulink

Формирование нейросетевых моделей. Основной функцией для формирование нейросетевых моделей в Simulink является функция **gensim,** записываемая в форме

gensim(net,st),

где net – имя созданной НС, st – интервал дискретизации (если НС не имеет задержек, ассоциированных с ее входами или слоями, значение данного аргумента устанавливается равным -1) [2].

Пример 1. Создание постейшей нейронной сети.

Пусть входной и целевой векторы имеют вид

p = [1 2 3 4 5];

t = [1 3 5 7 9].

Создадим линейную НС и протестируем ее:

» p = [1 2 3 4 5];

» t = [1 3 5 7 9];

» net = newlind(p,t);

» y =sim(net,p)

y = 1.0000 3.0000 5.0000 7.0000 9.0000

Затем запустим Simulink командой

» gensim(net,-1)

Это приведет к открытию блок-диаграммы (рис. 5).

Для проведения тестирования модели щелкнем дважды по левой иконке (с надписью Input 1, т. е. Вход 1), что приведет к открытию диалогового окна (рис. 6).

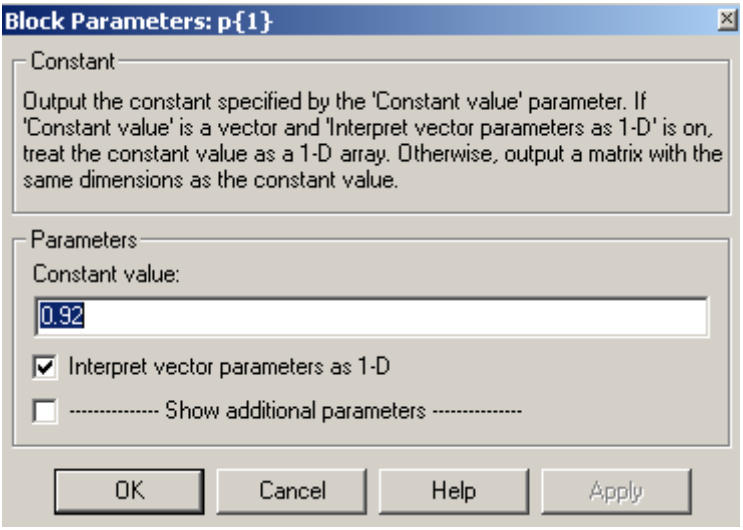


Рис. 6. Диалоговое окно задания входа НС

В данном случае блок Input 1 является стандартным блоком задания константы (Constant). Изменим значение по умолчанию на 2 и нажмем кнопку ОК. Затем нажмем кнопку Start в меню моделирования. Расчет нового значения сетью производится практически мгновенно. Для его вывода необходимо дважды щелкнуть мышью по правой иконке (блоку y(1)). Результат вычислений отображается рис. 7 – он равен 3, как и требуется.

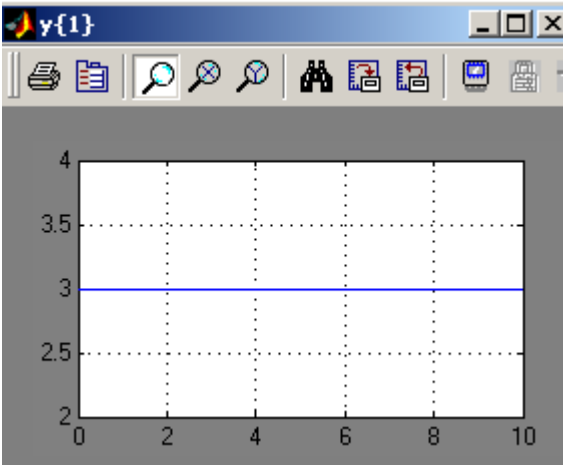


Рис. 7. Окно с выходом НС

Отметим, что, дважды щелкая левой кнопкой мыши по блоку Neural Network, затем – по блоку Layer 1, можно получить детальную графическую информацию о структуре сети (рис. 8).

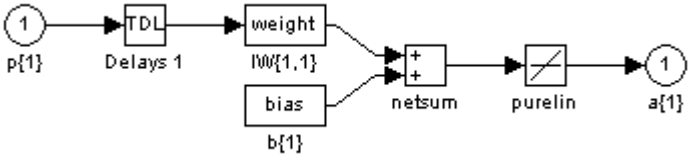


Рис. 8. Структура созданной НС

С созданной сетью можно проводить различные эксперименты, возможные в среде Simulink; вообще с помощью команды gensim осуществляется интеграция созданных нейросетей в блок-диаграммы этого пакета – с использованием имеющихся при этом инструментов моделирования различных систем (например, встраивание нейросетевого регулятора в систему управления и моделирование последней и т. п.).

Лабораторная работа № 3
Элементы математического моделирования неиросетевых систем управления в программной среде MATLAB 6.0

Библиотека инструментов пакета MATLAB 6.0 позволяет моделировать супервизорный режим работы многослойной нейросети в составе системы динамическим объектом, заданным уравнениями состояния. Реализация этого режима обеспечивается нейросетевым контроллером **Model Reference Control,** представляющим собой структуру, которая включает две многослойные нейронные сети: управляющую и идентификационную. Каждая сеть блока содержит один скрытый слой, количество нейронов в котором выбирается в зависимости от сложности задачи управления [2].

Блок **Model Reference Control** находится в библиотеке инструментов MATLAB в папке Neural Network Blockset/Control Systems.

Настройка контроллера осуществляется в два этапа. На первом этапе проводится идентификация объекта управления (обучение идентификационной многослойной нейронной сети (МНС)), а на втором – обучение управляющей МНС так, чтобы по-

ведение замкнутой системы «контроллер – объект» управления было бы идентичным поведению заранее выбранной модели (reference-модели). Таким образом, в данном контроллере реализуется стратегия обучения «с учителем». В процессе обучения поэтапно производится настройка весовых коэффициентов нейронов сети по выбранному алгоритму. Под одним уроком обучения понимается перенастройка всех весовых коэффициентов сети.

При открытии блока **Model Reference Controller** появляется окно настройки управляющей сети. Для того чтобы перейти к идентификации ОУ, необходимо нажать на кнопку Plant Identification, расположенную в левом нижнем углу окна. При этом появится новое окно **Plant Identification** (рис. 9), в котором можно будет задать параметры, необходимые для настройки идентификационной сети.

Идентификация объекта управления происходит в три стадии.

Первая стадия – выбор структуры идентификационной МНС (блок **Network Architecture** на рис. 9). Она включает в себя выбор следующих параметров:

Size of Hidden Layer – количество нейронов в скрытом слое (используется МНС с одним скрытым слоем). Этот параметр следует выбирать с учетом сложности объекта управления. Следует учесть, что сеть с большим количеством нейронов в скрытом слое будет обеспечивать более высокие качественные показатели за меньшее количество уроков, но при этом возрастет сложность вычислений, т. е. один урок будет занимать большее время.

Sampling Interval – период дискретизации (в секундах).

No. Delayed Plan! Inputs – количество элементов задержки на входах идентификационной сети.

No. Delayed Plant Outputs – количество элементов задержки на выходах идентификационной сети.

Normalize Training Data – нормализация данных в процессе обучения. При выборе этого параметра данные, используемые в процессе обучения, будут приведены к отрезку [0; 1].

Вторая стадия – это получение данных для обучения идентификационной сети (блок **Training Data** на рис. 9). На этом этапе возможно импортирование данных из MAT-файла или непосредственно из рабочего пространства MATLAB, а также формирование данных на основе заранее подготовленной модели объекта управления.

Модель объекта управления выполняется в виде отдельного файла SIMULINK и оформляется с помощью выделенных входа и выхода объекта (использовать блоки **Input Port** и **Output Port).**

В процессе формирования данных на основе представленной модели объекта управления проводится эксперимент, в котором на вход этой модели подается последовательность прямоугольных импульсов (в дальнейшем будем называть ее тестовым сигналом), параметры которой задаются следующими значениями:

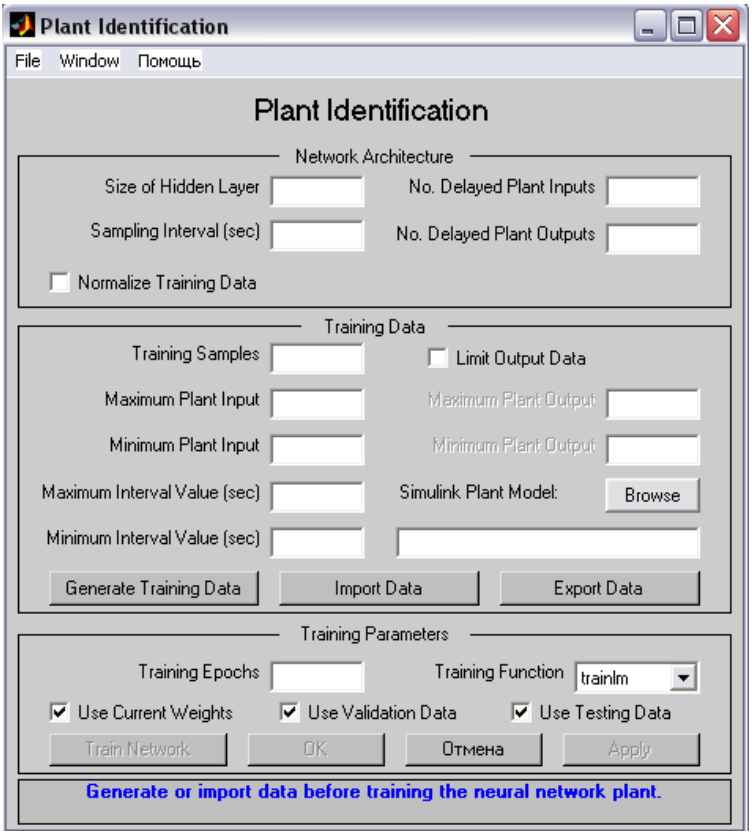


Рис. 9. Настройка процесса идентификации объекта управления

Training Samples – количество дискретных отсчетов в тестовой последовательности. Значение этого параметра следует выбирать исходя из сложности объекта управления.

Maximum Plant Input – максимальная величина сигнала.

Minimum Plant Input – минимальная величина сигнала.

Maximum Interval Value – максимальная длительность одного импульса (в секундах).

Minimum Interval Value – минимальная длительность одного импульса (в секундах).

Эти значения следует выбирать такими, чтобы тестовый сигнал предоставлял бы как можно большую информацию об объекте, т. е. был бы достаточно разнообразным.

Limit Output Data – ограничение выхода модели. При использовании этого параметра будет предложено выбрать минимальное и максимальное значения, которыми будет ограничена реакция объекта управления на тестовый сигнал.

Simulink Plant Model – здесь необходимо указать название модели SIMULINK. При нажатии на кнопку Browse будет предложено указать расположение заранее подготовленного файла с моделью (для корректной работы путь, по которому находится файл, не должен содержать русских символов).

После выбора всех значений необходимо нажать кнопку Generate Training Data. При этом процесс выработка последовательности будет отображен в графическом окне **Plant Input-Output Data.** Процесс можно прервать, нажав на кнопку Stop Simulation.

После окончания будет предложено либо принять данные (Accept Data), либо отклонить их (Reject Data).

Если по каким-то причинам полученные данные не подходят, то при нажатии на кнопку Reject Data окно закроется, и можно будет задать другие параметры тестовой последовательности, другую модель или импортировать данные из MAT-файла или из рабочего пространства MATLAB.

При принятии данных кнопка Generate Training Data заменяется на Erase Generated Data. Таким образом, на любом этапе можно остановиться и сформировать тестовую последовательность заново.

Кнопка Import Data служит для импорта тестовых данных из сохраненного ранее MAT-файла или из рабочего пространства MATLAB. Кнопка Export Data предназначена для экспорта тестовых данных в MAT-файл или в рабочее пространство MATLAB в виде структуры данных или массива.

После получения тестовой последовательности необходимо перейти к заключительной стадии – обучению идентификационной МНС.

Для этого необходимо определить следующие параметры:

Training Epochs – количество уроков обучения. Значение этого параметра необходимо выбирать достаточно большим, чтобы ошибка обучения была бы достаточно мала. Однако, начинав с некоторого момента времени, дальнейшее обучение сети становится практически бесполезным, и поэтому количество уроков не должно быть слишком большим. Для данного примера обучение можно прекратить, когда график ошибки выйдет на длинный пологий участок.

Training Function – выбор алгоритма обучения. Для данного примера рекомендуется использовать алгоритм trainlm.

Use Current Weights – использовать уже полученные весовые коэффициенты. Этот параметр можно выбирать при повторном обучении сети.

Use Validation Testing Data – при выборе этих параметров 25% тестовых данных будет использовано для раннего завершения обучения и/или проверки сети. В нижней части окна синим цветом отображается подсказка.

После выбора всех параметров необходимо нажать на кнопку Train Network (обучение сети). При этом появится новое окно, в котором будет наглядно отображаться ход обучения, т. е. будет показана ошибка обучения для каждого урока. Процесс обучения можно прервать, нажав на кнопку Stop Training.

После завершения процесса обучения будет показано два новых окна – **Training data for NN Model Reference Control** и **Testing data for NN Model Reference Control** (если был выбран параметр Use Testing Data).

После успешного обучения идентификационной сети необходимо перейти к обучению управляющей сети. Для этого в окне Plant Identification нужно нажать на кнопку OK.

Для настройки процесса обучения управляющей сети предназначено окно **Model Reference Control (**рис. 10**)**.

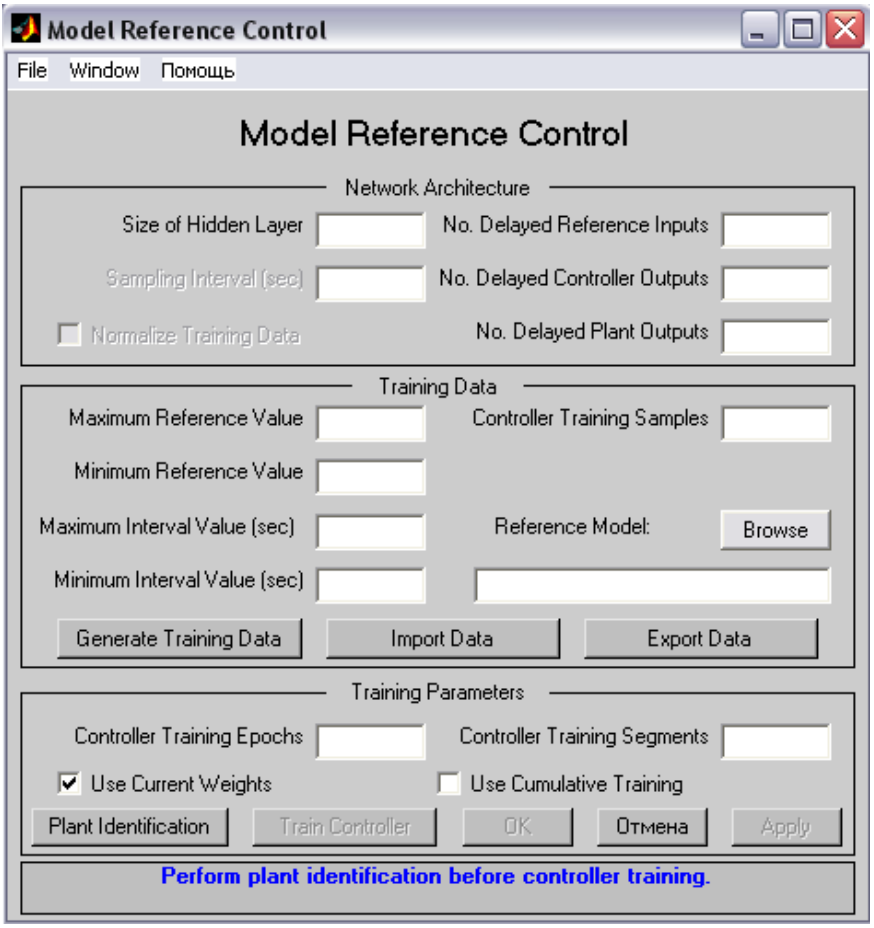


Рис. 10. Настройка процесса обучения управляющей сети

Процесс обучения контроллера также можно разбить на три стадии.

Первая стадия – выбор конфигурации управляющей сети

Здесь параметры Sampling Interval и Normalize Training Data выбираются такими же, как и при обучении идентификационной сети автоматически.

No. Delayed Reference Inputs – количество элементов задержки на входе, на который подается тестовый сигнал.

No. Delayed Controller Outputs – количество элементов задержки на выходе управляющей сети.

No. Delayed Plant Outputs – количество элементов задержки на выходе идентификационной сети.

Вторая стадия – получение данных для обучения.

На этой стадии, как и при обучении идентификационной сети, возможно импортирование данных из MAT-файла или непосредственно из рабочего пространства MATLAB, а также формирование данных на основе заранее подготовленной эталонной (reference) модели.

Reference-модель – это модель, которая, описывает желаемое поведение замкнутой системы контроллер-объект управления. Она задается так же, как и модель объекта управления на этапе идентификации, т. е. в виде файла на графическом языке SIMULINK.

Controller Training Samples – количество дискретных отсчетов а тестовом сигнале. Оно должно быть, как и при идентификации объекта, достаточно большим.

Все параметры, как и сам процесс формирования тестовой последовательности, аналогичны используемым на этапе идентификации объекта управления.

Третья стадия – обучение контроллера (управляющей сети). Для обучения необходимо задать следующие параметры:

Controller Training Epochs – количество уроков. Значение этого параметра нельзя определить заранее, поэтому его необходимо подобрать в процессе обучения.

Controller Training Segments – количество сегментов, на которые будут разделены тестовые данные. При этом для каждого сегмента будет проведено указанное количество уроков. Желательно разбивать данные на сегменты таким образом, чтобы каждый сегмент содержал хотя бы один переходный процесс.

Use Current Weights – использовать текущие весовые коэффициенты. В противном случае весовые коэффициенты будут выбраны случайным образом. Этот параметр следует применять при повторном обучении.

Use Cumulative Training – при выборе этого параметра первоначальное обучение будет выполнено с первым сегментом тестовых данных, а при последующем обучении остальные сегменты будут добавляться к данным, используемым на предыдущем этапе обучения. Таким образом, на финальной стадии обучения будет использован весь набор тестовых данных. Этот параметр следует использовать с осторожностью, так как он увеличивает время обучения.

После задания всех параметров необходимо приступить к обучению управляющей сети, для чего нужно нажать на кнопку Train Controller. При этом откроется окно, в котором будет

наглядно отображаться процесс обучения (так же, как и при обучении идентификационной сети). Процесс обучения можно прервать, нажав на кнопку Stop Training.

Внимание! Процесс обучения может занимать длительное время (до нескольких часов), и при этом программа может не реагировать на нажатие кнопки Stop Training.

После завершения обучения открывается окно **Plant** Response **for** NN Model **Reference Control,** в котором отображаются результаты обучения. Их можно просмотреть более детально, выбрав в меню пункт View/Figure Toolbar. При этом под строкой меню появится панель инструментов, с помощью которой можно манипулировать изображением.

После успешного завершения процесса обучения управляющей сети контроллер готов к использованию в составе системы управления.

Возможные проблемы в процессе моделирования.

1. Процесс обучения идентификационной сети идет медленно или прекращается самопроизвольно, что определяется по тому, что ошибка обучения не уменьшается с какого-либо урока.

Это может происходить в следующих случаях:

- неверно выбрана конфигурация MHC2 – необходимо задать большее число нейронов в скрытом слое;
- неверно задан набор тестовых данных, не «возбуждающих» ОУ, – следует выбрать другие значения параметров в блоке **training data;**
- выбрано слишком малое число уроков обучения и его необходимо увеличить.

2. Процесс обучения управляющего контроллера (сети MHC1) идет крайне медленно и ошибка обучения практически не уменьшается.

Это может происходить в следующих случаях:

- неудачно заданы параметры тестовой последовательности при идентификации объекта и необходимо повторить процесс обучения сети MHC2 с другими параметрами в блоке training data; следует использовать близкие по форме и параметрам тестовые сигналы при обучении обеих MHC;
- выбрано слишком малое число элементов в скрытом слое и его необходимо увеличить.

3. Прямые показатели качества моделируемой системы существенно отличаются от аналогичных показателей для эталонной модели.

Это возможно в следующих случаях:

- идентификация объекта произведена неудачно, поэтому следует повторить настройку MHC2, учитывая ранее допущенные ошибки эксперимента, например, выбор недостаточного числа уроков обучения MHC2, числа нейронов в скрытом слое сети;
- конфигурация управляющей сети MHC1 не позволяет решать возложенную на нее задачу управления и необходимо повторить обучение с увеличенным количеством элементов в скрытом слое [2].

Пример 2. Система автоматического управления с нейросетевым регулятором на основе эталонной модели [3].

В качестве примера моделирования в среде Simulink систем управления с использованием нейросетевых регуляторов используем демонстрационный пример пакета, иллюстрирующий систему управления с эталонной моделью звеном («рукой») робота и вызываемый с помощью пунктов Help/Demos меню MATLAB (см. рис. 11).

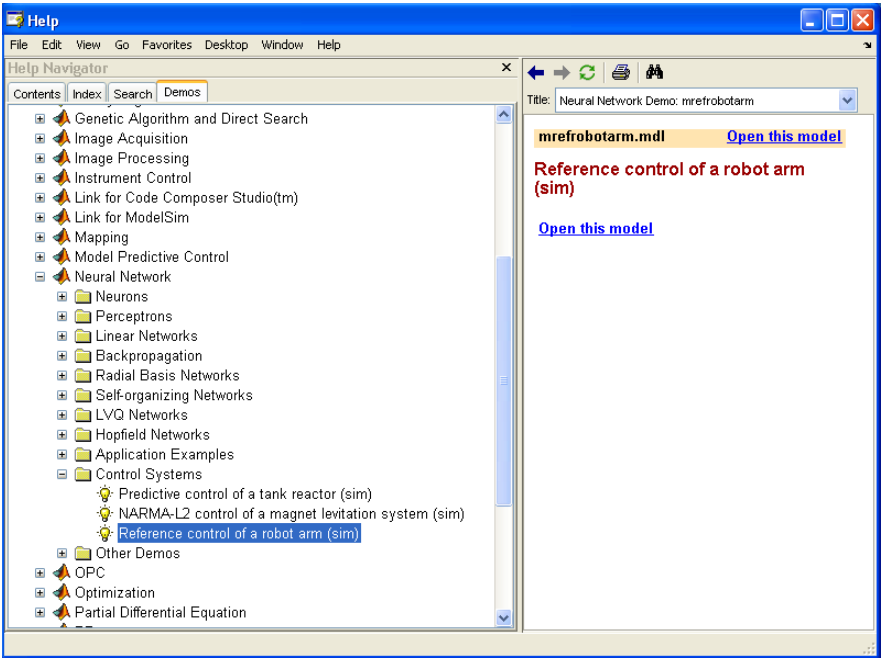


Рис. 11. Окно справки MATLAB

При открытии модели (с помощью мышки пункта Open this model (Отрыть эту модель)) в правой части окна, открывается окно Simulink с названием **mrefrobotarm** (рис. 12).

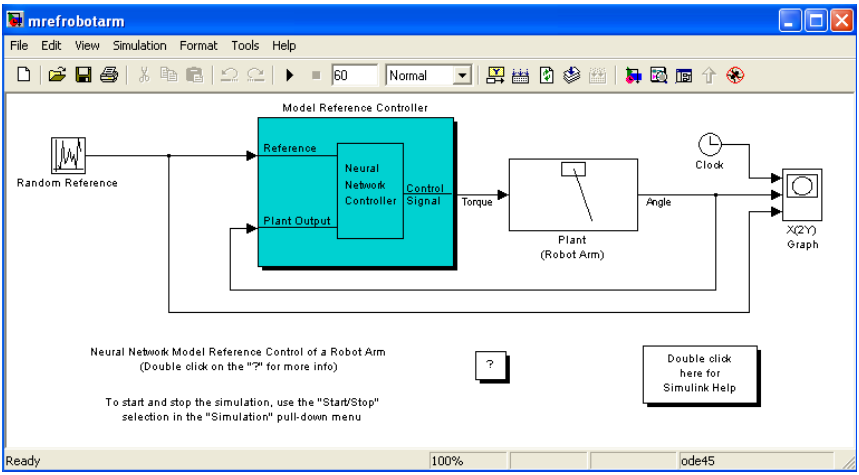


Рис. 12. Структура нейросетевой системы управления в среде Simulink

При реализации системы использованы 2 нейронные сети – для регулятора и для модели объекта управления.

В этом демонстрационном примере цель заключается в управлении движением одного звена робота, схематично показанного на рис. 13.

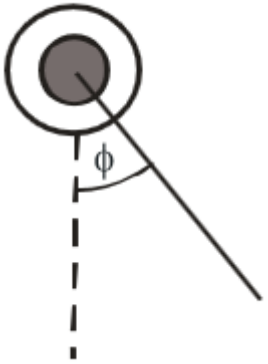


Рис. 13. Схематичное изображение движения звена робота

Управление движения звена представлено соотношением:

$$\frac{d^2\phi}{dt^2} = -10\sin\phi - 2\frac{d\phi}{dt} + u,$$

где φ – угол поворота звена, u – момент, развиваемый двигателем постоянного тока.

Соответствующая динамическая модель, реализаванная в среде Simulink, приведена на рис. 14.

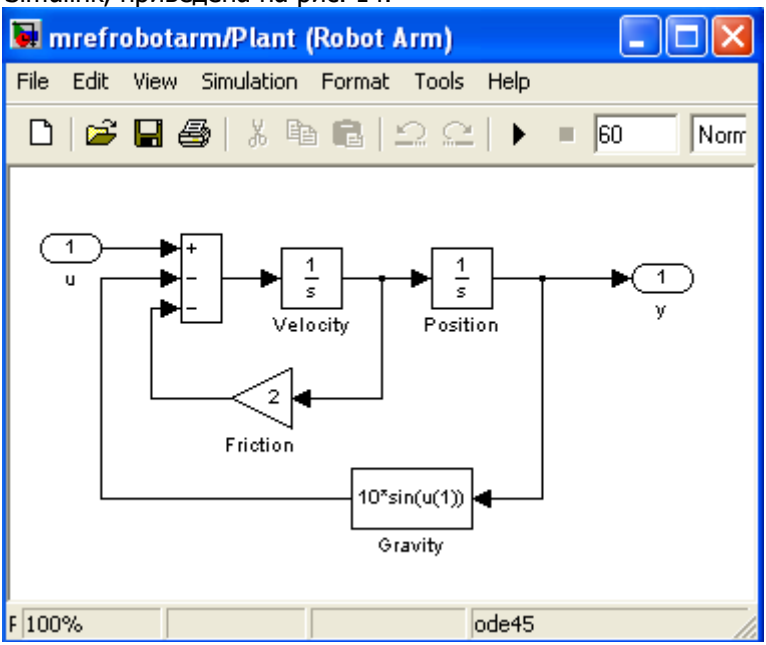


Рис. 14. Структурная динамическая модель звена робота

Цель обучения регулятора состоит в том, чтобы движение звена соответствовало выходу эталонной модели:

$$\frac{d^2 y_r}{dt^2} = -9\sin y_r - 6\frac{dy_r}{dt} + 9r,$$

где y_r – выход эталонной модели, r – задающий сигнал на входе модели.

Структурная схема, поясняющая принцип построения системы управления с эталонной моделью, показана на рис. 15.

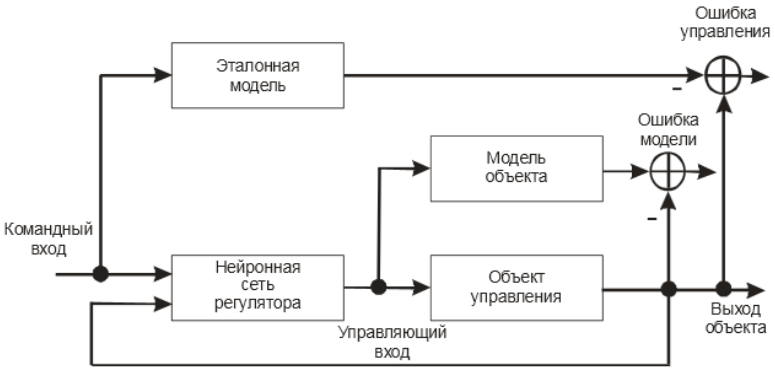


Рис. 15. Система управления с эталонной моделью

На рисунке:
Command Input – задающий сигнал,
Reference Model – эталонная модель,
NN Controller – нейросетевой регулятор,
NN Plant Model – нейронная сеть модели объекта,
Plant – объект управления,
Model Error – ошибка модели,
Control Error – ошибка управления,
Plant Output – выход объекта.

В приведенной структуре следует выделить эталонную модель, которая задает желаемую траекторию движения звена робота, удовлетворяющему второму из приведенных дифференциальных уравнений, а также нейронные сети, реализующие регулятор и объект управления.

В данном случае архитектура НС регулятора описывается профилем 5–13–1 (5входов, 13 нейронов скрытого слоя и один выход), т. е. содержит два слоя нейронов. Из двух слоев состоит и нейронная сеть модели объекта. Линии задержки, используемые для формирования входов НС, имеют такт дискретности 0.05 с.

Для начала работы с демонстрационным примером необходимо активизировать блок **Model Reference Controller** (см. рис. 12) двойным щелчком левой кнопки мыши. Появится окно, показанное на рис. 16.

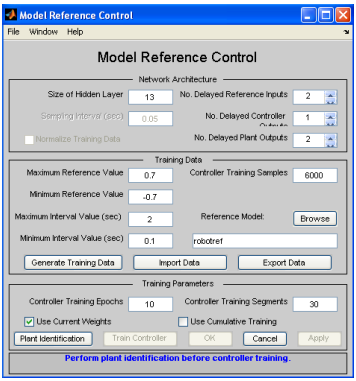


Рис. 16. Окно настроек нейросетевых моделей

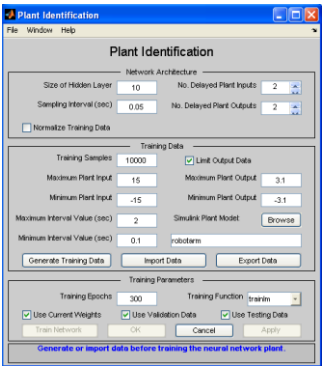


Рис. 17. Окно для построения нейросетевой модели ОУ

Особенность рассматриваемой системы управления состоит в том, что следует построить две нейронные сети – модель объекта управления и самого регулятора. Начнем с построения модели объекта управления, выбрав в окне настроек кнопку Plant Identification. При этом откроется окно **Plant Identification**, которое показано на рис. 17.

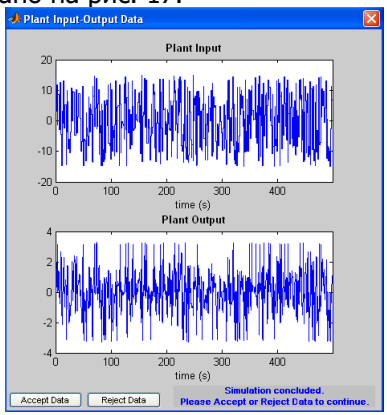


Рис. 18. Сигнал ОУ в режиме его идентификации

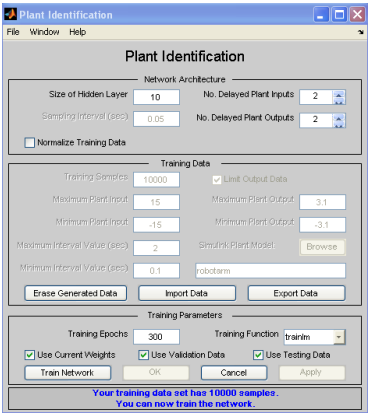


Рис. 19. Модифицированное окно для построения модели ОУ

Для построения нейросетевой модели ОУ следует выбрать кнопку Generate Training Data. Это приведет к тому, что будет запущена программа генерации обучающей последовательности на интервале 500 с для модели рассматриваемого объекта управления. Программа генерирует обучающие данные путем воздействия ряда случайных ступенчатых сигналов на модель Simulink

объекта. Графики входного и выходного сигнала объекта выводятся на экран (рис. 18).

По завершении генерации обучающей последовательности пользователю предлагается либо принять данные (Accept Data), либо отказаться от них (Reject Data).

Примем данные, после чего приложение возвратит нас к несколько измененному окну **Plant Identification** (рис. 19). Здесь часть окна недоступна, а кнопка Generate Training Data заменена на кнопку Erase Generated Data, что позволяет удалить сгенерированные данные.

Теперь можно начать обучение нейронной сети.

Для этого следует воспользоваться кнопкой Train Network (Обучить сеть). Начнется обучение нейросетевой модели. После завершения обучения его результаты отображаются на графиках, как это показано на рис. 20 и 21, где построены соответствующие результаты обучения и тестирования на контрольном множестве.

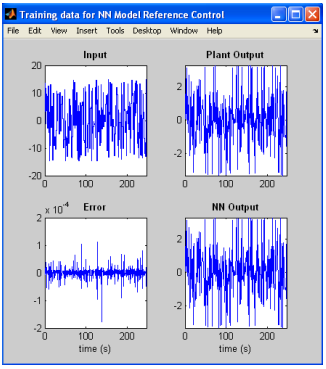


Рис. 20. Результат обучения модели на обучающем множестве

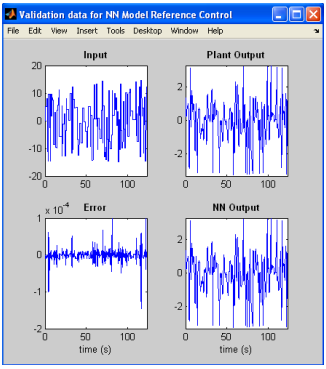


Рис. 21. Результат тестирования модели на контрольном множестве

Текущее состояние отмечено в окне **Plant Identification** (рис. 22) сообщением «Обучение завершено». Вы можете сгенерировать или импортировать новые данные, продолжить обучение или сохранить полученные результаты, выбрав кнопки OK или Apply.

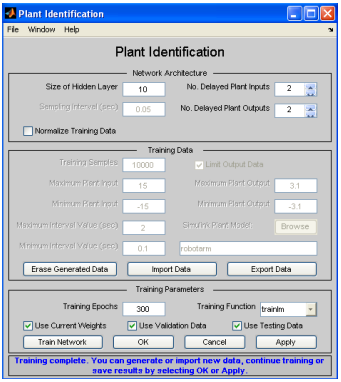


Рис. 22. Окно идентификации объекта с сообщением об окончании процедуры

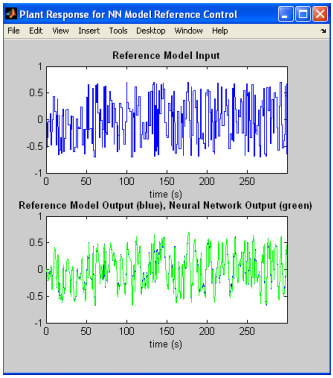


Рис. 23. Выход эталонной модели и объекта управления.

Выберем кнопку OK. В результате параметры нейросетевой модели объекта управления будут введены в блок **Model Reference Controller** системы Simulik, после чего произойдет возврат в окно **Model Reference Controller** (рис. 23).

Теперь для обучения регулятора следует выбрать кнопку Generate Training Data, чтобы сгенерировать обучающие данные. Эти данные в виде графиков появятся в окне **Input-Output Data for NN Model Reference Controller** и снова необходимо подтвердить или опровергнуть эти данные. Если данные приемлемы, то в окне **Model Reference Controller** следует выбрать кнопку Train Controller (Обучить регулятор). После того как обучение закончится, графики выходов эталонной модели и объекта управления выводятся на экран (рис. 24). Заметим, что обучение регулятора занимает, в силу применяемого алгоритма обучения (динамического варианта метода обратного распространения ошибки), весьма значительное время.

По окончании обучения регулятора необходимо нажать на кнопку OK, вернуться к модели Simulik (рис. 12) и начать моделирование, выбрав опцию Start из меню Simulation. Графики эталонного сигнала и выхода объекта управления показаны на рис. 25.

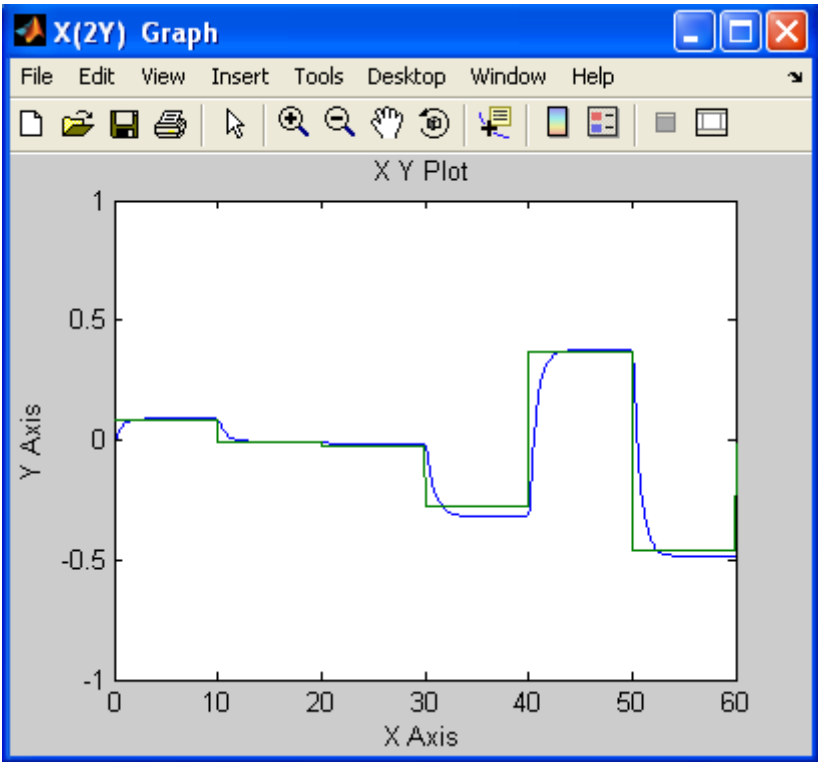


Рис. 25. Графики эталонного сигнала (ступенчатый процесс) и выхода объекта

Из анализа полученных данных следует, что реакция системы на ступенчатые воздействия со случайной амплитудой носит монотонный характер и обрабатывается в пределах 3…4 с. Это свидетельствует о хорошем качестве регулятора Model Reference Controller для управления звеном робота-манипулятора.

Программа работы и методические указания

1) Выполните все примеры, предлагаемые в теоретической части.

2) Для примера 2 исследуйте в моделируемой схеме нейросетевой адаптивной системы влияние параметров нейросети и числа уроков обучения на показатели качества процесса идентификации объекта (первый этап адаптивного управления) и процесса обучения сети MHC1 (второй этап адаптивного управления).

3) Отобразите результаты обучения (для примера 2) для различного числа нейронов в скрытом слое.

Содержание отчета

- цель работы;
- задание;
- краткое описание действий по пунктам;
- графики по всем пунктам программы;

выводы по работе

Лабораторная работа № 4
«Распознавание цифр»

Цель работы.

1. Построение нейронных сетей в среде MATLAB.
2. Исследование возможностей распознавания печатных символов с помощью нейронных сетей.

Работа включает три этапа:

1. Подготовка эталонных (обучающих) образов печатных символов в виде набора графических файлов.
2. Создание и обучение нейронной сети (НС) в среде MATLAB.
3. Распознавание печатных символов с помощью обученной НС.

1. Подготовка эталонных образов.

Набор эталонных образов задается преподавателем. Примером такого набора является последовательность из десяти цифр от 0 до 9. В этом примере число образов M=10. В случае, когда каждый класс образов характеризуется лишь своим эталоном, имеем число классов, также равное M.

Каждый образ формируется в виде графического файла в битовом формате. Тип файла (расширение) определяется используемыми в среде MATLAB типами графических файлов. Рекомендуется использовать расширение ***tif***.

Для создания графических файлов образов удобно использовать среду "***Adobe Photoshop***". В этом случае при создании каждого файла необходимо проделать следующую последовательность операций:

1) создать новый файл, задав его параметры:
   - имя : *XXXX*;
   - ширина: N1 пикселей;
   - высота: N2 пикселей;
   - цветовой режим: битовый формат.

Значения N1, N2=8…20 задаются преподавателем.

2) используя инструменты типа «Кисть», «Ластик» и др. создать требуемый образ символа.
3) с помощью команды «Сохранить как» сохранить созданный образ в виде файла типа ***tif*** в заданной преподавателем папке.

На рис. 1 приведены примеры графических символов цифр при N1=10, N2=12 пикс.

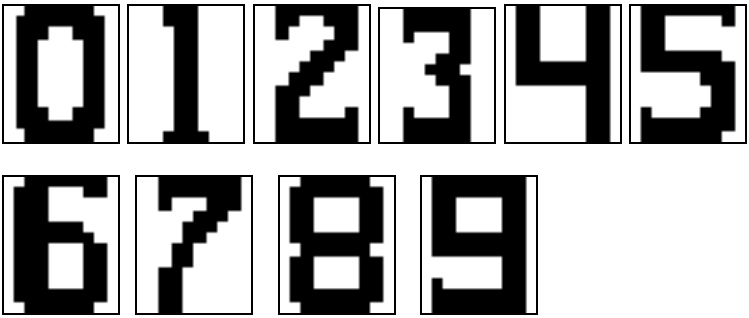


Рис. 1. Примеры графических символов цифр.

2. Создание и обучение НС с среде MATLAB.

На данном этапе выполнение работы в среде MATLAB производится с помощью программы ***sr_newff*** , которая реализует следующие функции:

- формирование числовых массивов эталонных образов, используемых в качестве обучающих;
- подготовка данных, необходимых для создания НС;
- создание НС, задание параметров обучения НС и обучение НС.

Эталонный образ каждого символа представлен в виде вектора-столбца [N,1], число элементов N которого равно числу признаков (иначе говоря, N – размерность пространства признаков). Такой вектор-столбец формируется из двумерного массива-изображения [N1,N2], который, в свою очередь, формируется при считывании графического файла образа с помощью команд:

imread (FILENAME) - процедура чтения графического файла;

X = reshape (A,[N,1]) - процедура преобразования двумерного массива A[N1,N2] в одномерный вектор-столбец X[N,1], где N=N1*N2.

Процедура умножения массива на 1 приводит к смене типа элементов массива с *logical* (для элементов битового формата) на *double*.

Для удовлетворительной работы НС недостаточно формирования лишь одного обучающего образа для каждого класса (типа символа) образов. Это связано с тем, что распознаваемые образы (на этапе работы НС в режиме распознавания) всегда отличаются от обучающих по ряду причин:

- различие шрифтов и стилей печатных символов;
- погрешности сканирования и неточности совмещения символа и окна сканирования;
- низкое качество печати, дефекты бумаги и т.д.

В силу указанных причин для надежного распознавания образов НС следует обучать на достаточно представительном множестве образов, входящих в один и тот же класс.

В программе ***sr_newff*** формирование дополнительных обучающих образов производится путем незначительного искажения эталонных образов, считываемых из графических файлов. Искажение образа-эталона каждого класса реализуется путем добавления к нему равномерного (по площади изображения) шума типа «Соль и перец», представляющего собой случайное искажение отдельных пикселей изображения. Степень искажения характеризуется числом p=[0;1], определяющим долю искаженных пикселей.

Такой подход при формировании образов позволяет, во-первых, быстро получать большое число обучающих образов, и, во-вторых, регулировать (путем изменения значения p) степень разброса множества образов в пределах одного класса.

Подготовка данных, необходимых для создания НС, включает в себя:

1) формирование двумерного массива обучающих образов XR[N,K], каждый столбец которого представляет собой набор N признаков одного образа, а число столбцов K равно числу обучающих образов;
2) формирование двумерного массива желаемых откликов YR[NY,K], где NY – число выходов НС (т.е., число нейронов выходного слоя); K – число обучающих образов. Отклик YR[:,k] (в общем случае – вектор-столбец) соответствует k-му обучающему образу – вектору XR[:,k];

3) формирование двумерного массива R[N,2], определяющего минимальное R(n,1) и максимальное R(n,2) значение n-го признака, n=1,…,N.

Создание НС. В общем случае НС ***net*** создается с помощью команды:

net = nnnnn (P1,P2,…PL), где

nnnnn – тип НС;

P1,…,PL – параметры НС.

В настоящей работе используется НС типа многослойного персептрона newff, которая задается командой:

net = newff (R, [A1 A2 … AL], {F1 F2 … FL}, BTF, PF), где

R - массив минимальных и максимальных значений входных нейронов (признаков);

Ai - число нейронов i-го слоя, начиная с первого скрытого слоя, i=1,…,L;

Fi - функция активации нейронов i-го слоя, по умолчанию 'tansig';

BTF - функция обучения сети, по умолчанию 'trainlm';

PF - критерий остановки, по умолчанию 'mse' (минимум ско).

Дополнительные параметры, используемые при создании сети:

net.performFcn='msereg' - обучение НС производится с помощью метода регуляризации;

net.performParam.ratio=0.1 - значение параметра регуляризации;

net.trainParam.show=5 - число эпох, через которое производится вывод параметров процедуры обучения;

net.trainParam.epochs=500 - максимальное число эпох при обучении НС;

net.trainParam.goal=0.02 - значение целевой функции, по достижении которого процесс обучения НС прекращается.

Процесс обучения НС запускается командой:

net = train (net, XR, YR) .

Для решения задач распознавания печатных символов рекомендуется использовать трехслойную НС (один скрытый слой) с числом нейронов:

N=120 - во входном слое;

A1=20 - в скрытом (промежуточном) слое;

A2=1 - в выходном слое.

При использовании большего числа нейронов процедура обучения НС может занять слишком много времени. Для рекомендованных значений параметров НС (в том числе и дополнительных) и общем числе обучающих образов (для всех заданных классов) K=100…200 время обучения НС составляет 5…20 мин.

3. Распознавание печатных символов с помощью обученной НС.

Работа НС, т.е. формирование отклика Y при входном воздействии в виде вектора-столбца X[N,1] производится командой:

Y = sim (net, X).

В случае, когда желаемый отклик принимает целочисленные значения, рекомендуется использовать округление до ближайшего целого, т.е.

Y = round (sim (net, X)).

Тестирование работы НС при распознавании печатных символов с различной степенью искажения производится с помощью программы ***sr_work*** , исходными данными для которой являются:

SX.tif - имя графического файла образа-эталона;

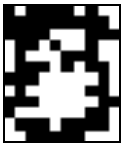
N- число пикселей изображения образа;

NT - число тестируемых образов, полученных путем искажения эталона;

P- доля искаженных пикселей [0; 1].

На рис. 2 представлены некоторые примеры распознавания символов, изображенных на рис. 1, с помощью обученной НС. Обучение проводилось при числе обучающих образов M=10 для каждого вида символа и параметре искажения символов p=0,1.

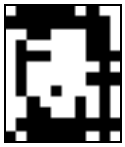
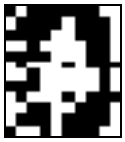
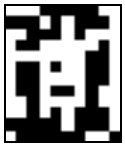




Результат распознавания:

«2» **«3»** **«5»**

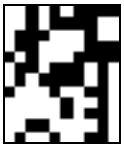
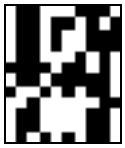
Рис. 2.а. Неверные распознавания символа **«0»**, искаженного 20% шума «Соль и Перец».

Результаты распознавания:

«0» **«0»** **«0»** **«0»**

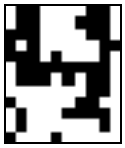
Рис.2.б. Правильные распознавания символа **«0»**, искаженного 20% шума «Соль и Перец».

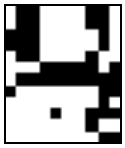




Результат распознавания:

«3» **«5»** **«6»**

Рис. 2.в. Неверные распознавания символа **«4»**, искаженного 20% шума «Соль и Перец».

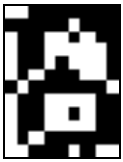
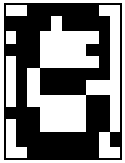
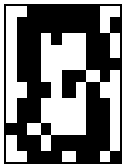






Результаты распознавания:

«4» **«4»** **«4»** **«4»** **«4»**

Рис.2.г. Правильные распознавания символа **«4»**, искаженного 20% шума «Соль и Перец».

Результаты распознавания:

«8» **«8»** **«6»**

Рис.2.д. Результаты распознавания символа **«8»**, искаженного 10% шума «Соль и Перец».

Результаты распознавания символов, представленные на рис. 2 а-г, демонстрируют хорошее распознавание с помощью НС даже при сильном искажении (параметр p>0,1). Для объективной оценки качества работы НС необходимо вычисление вероятностных характеристик распознавания. При правильном выборе пара-

метров обучения сети и использовании не менее 100 обучающих образов можно получить вероятность правильного распознавания символов порядка 0,6…0,9 (в зависимости от вида распознаваемого символа) при параметре искажения p=0,1…0,2.

Порядок выполнения работы.

1. Подготовить графические файлы эталонных образов для символов, заданных преподавателем.
2. В среде MATLAB создать и обучить НС, предназначенную для распознавания печатных символов.
3. Исследовать зависимость качества работы НС от:
   - степени искажения символов (параметр p);
   - числа нейронов в скрытом слое.

Качество работы НС характеризуется вероятностями правильной классификации $P_{np}(i)$ образа i-го класса, i=1,…,M. Оценка вероятностей $P_{np}(i)$ производится по формуле:

$$\hat{P}_{np}(i) = \frac{N_{np}}{N_0},$$

где N_{np} - число правильных распознаваний образа i-го класса; N_0 - общее число распознаваний образов i-го класса. Число N_{np} определяется экспериментально при запуске программы ***sr_work*** при значениях N_0 =10…100.

Лабораторная работа № 5
«Задача распространения образов с использованием нейронных сетей»

Цель работы: Изучить принципы построения и применения методов искусственного интеллекта, в частности искусственных нейронных сетей, в задачи распознавания образов.

1. Теоретическая часть

Под нейронными сетями подразумеваются вычислительные структуры, которые моделируют простые биологические процессы, обычно ассоциируемые с процессами человеческого мозга. Адаптируемые и обучаемые, они представляют собой распараллеленные системы, способные к обучению путем анализа положительных и отрицательных воздействий. Элементарным преобразо-

вателем в данных сетях является *искусственный нейрон,* или просто *нейрон,* названный так по аналогии с биологическим прототипом.

К настоящему времени предложено и изучено большое количество моделей нейроподобных элементов и нейронных сетей.

Нервная система и мозг человека состоят из нейронов, соединенных между собой нервными волокнами. Нервные волокна способны передавать электрические импульсы между нейронами. Все процессы передачи раздражений от нашей кожи, ушей и глаз к мозгу, провесы мышления и управления действиями — все это реализовано в живом организме как передача электрических импульсов между нейронами.

Нейрон (нервная клетка) является особой биологической клеткой, которая обрабатывает информацию (рисунок 1.1). Он состоит из тела и отростков нервных волокон двух типов — *дендритов,* по которым принимаются импульсы, и единственного *аксона,* по которому нейрон может передавать импульс. Тело нейрона включает ядро, которое содержит информацию о наследственных свойствах, и плазму, обладающую молекулярными средствами для производства необходимых нейрону материалов. Нейрон получает сигналы (импульсы) от аксонов других нейронов через дендриты (приемники) и передает сигналы, сгенерированные телом клетки, вдоль своего аксона (передатчика), который в конце разветвляется на волокна. На окончаниях этих волокон находятся специальные образования — *синапсы,* которые влияют на силу импульса.

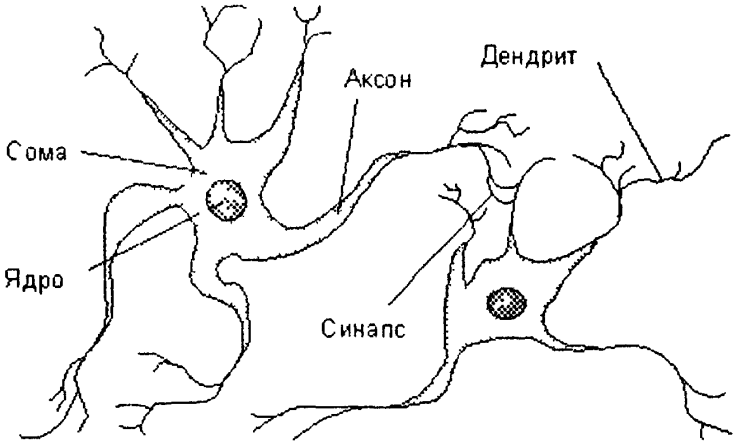


Рисунок1.1 – Взаимосвязь биологических нейронов

Синапс является элементарной структурой и функциональным узлом между двумя нейронами (волокно аксона одного нейрона и дендрит другого). Когда импульс достигает синоптического окончания, высвобождаются определенные химические вещества, называемые нейротрансмиттерами. Нейротрансмиттеры диффундируют через синоптическую щель, возбуждая или затормаживая, в зависимости от типа синапса, способность нейрона-приемника генерировать электрические импульсы. Результативность синапса может настраиваться проходящими через него сигналами, так что синапсы могут обучаться в зависимости от активности процессов, в которых они участвуют. Эта зависимость от предыстории действует как память, которая, возможно, ответственна за память человека. Важно отметить, что веса синапсов могут изменяться со временем, что изменяет и поведение соответствующего нейрона.

Структура и свойства искусственного нейрона

Нейрон — это составная часть нейронной сети (НС). На рисунке 1.2 показана его структура.

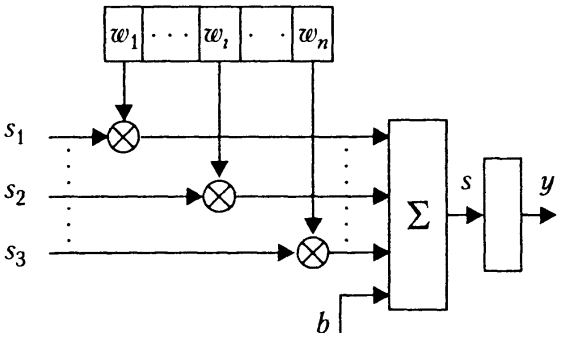


Рисунок 1.2 - Структура искусственного нейрона

В состав нейрона входят умножители (синапсы), сумматор и нелинейный преобразователь. Синапсы осуществляют связь между нейронами и умножают входной сигнал на число, характеризующее силу связи, — вес синапса. Сумматор выполняет сложение сигналов, поступающих по синоптическим связям от других нейронов, и внешних входных сигналов. Нелинейный преобразователь реализует нелинейную функцию одного аргумента — выхода сумматора. Эта функция называется *функцией активации*

или *передаточной функцией* нейрона. Нейрон в целом реализует скалярную функцию векторного аргумента. Математическая модель нейрона описывается соотношениями

$$s = \sum_{i=1}^{n} w_i x_i + b;$$

$$y = f(s),$$

где w_i — вес синапса $(i = 1...n)$;
b — значение смещения;
s — результат суммирования;
x_i — компонент входного вектора (входной сигнал) (i = 1...n);
y — выходной сигнал нейрона;
n — число входов нейрона;
f — нелинейное преобразование (функция активации или передаточная функция).

В общем случае входной сигнал, весовые коэффициенты и значения смещения могут принимать вещественные значения. Выход (y) определяется видом функции активации и может быть как действительным, так и целым. Во многих практических задачах входы, веса и смещения могут принимать лишь некоторые фиксированные значения.

Синоптические связи с положительными весами называют *возбуждающими,* с отрицательными весами — *тормозящими.*

Таким образом, нейрон полностью описывается своими весами w_i и передаточной функцией f(s). Получив набор чисел (вектор) x_i в качестве входов, нейрон выдает некоторое число y на выходе.

Описанный вычислительный элемент можно считать упрощенной математической моделью биологических нейронов — клеток, из которых состоит нервная система человека и животных.

Чтобы подчеркнуть различие нейронов биологических и математических, вторые иногда называют *пейроподобными элементами* или *формальными нейронами.*

Пакет расширения по нейронным сетям Neural Networks Toolbox

Пакет Neural Networks Toolbox (Нейронные сети) содержит средства для проектирования, моделирования, обучения и использования множества известных парадигм аппарата искусственных нейронных сетей (ИНС): от базовых моделей персептрона до самых современных ассоциативных и самоорганизующихся

сетей. Пакет может быть использован для решения множества разнообразных задач, таких как обработка сигналов, нелинейное управление, финансовое моделирование и т. п.

Функция создания сети:

Сеть = newff (minmax(Вход),[Количество нейронов входного слоя, Количество нейронов промежуточного слоя, Количество нейронов выходного слоя],{'Функция активации входного слоя' 'Функция активации промежуточного слоя' 'Функция активации выходного слоя'},'Функция обучения').

Функции активации (передаточные функции)

- compet — функция конкуренции;
- hardlim — пороговая функция активации с порогом Θ = 0;
- hardlims — знаковая или сигнатурная функция активации;
- logsig — сигмоидальная логистическая функция;
- poslin — полулинейная функция;
- purelin — линейная функция активации;
- radbas — радиальная базисная функция;
- satlin — полулинейная функция с насыщением;
- satlins — линейная функция с насыщением;
- softmax — функция

$$\frac{\exp(x_Y)}{\sum_{i=1}^{N}\exp(x_Y)}$$

где *N* — число строк матрицы-аргумента X;

- tansig — сигмоидальная функция (гиперболический тангенс);
- tribas — треугольной функции принадлежности;
- dhardlim — производная пороговой функции активации.

Функции обучения

- trainbfg - функция обучения, реализующая разновидность квазиньютоновского алгоритма обратного распространения ошибки (BFGS);
- trainbr - функция, реализующая так называемый байесовский метод обучения, сущность которого заключается в подстройке весов и смещений сети на основе алгоритма Левенберга—Марквардта. Данный алгоритм минимизирует комбинацию квадратов ошибок и весов с выбором наилучшего варианта (для получения наилучших обобщающих свойств сети). Эта процедура из-

вестна как байесовская регуляризация, откуда следует название метода;

- traincgb - функция обучения НС, реализующая разновидность алгоритма сопряженных градиентов (так называемый метод Powell—Beale);

- traincgf - функция обучения НС, реализующая разновидность алгоритма обратного распространения ошибки в сочетании с методом оптимизации Флетчера— Лоуэлла;

- traincgp - то же, что в предыдущем случае, но с использованием метода Polak—Ribiere;

- traingd - функция, реализующая «классический» алгоритм обратного распространения ошибки;

- traingda - то же, что в предыдущем случае, но с адаптацией коэффициента скорости обучения;

- traingdm - функция, реализующая модифицированный алгоритм обратного распространения ошибки с введенной «инерционностью» коррекции весов и смещений;

- traingdx - функция, реализующая комбинированный алгоритм обучения, объединяющий особенности двух вышеприведенных;

- trainlm - данная функция возвращает веса и смещения НС, используя алгоритм оптимизации Левенберга—Марквардта;

- trainoss - функция, реализующая разновидность алгоритма обратного распространения ошибки с использованием метода секущих;

- trainrp - функция, реализующая разновидность алгоритма обратного распространения ошибки, так называемый *упругий* алгоритм обратного распространения (resilient backpropagation algorithm, RPROP);

- trainscg - данная функция возвращает веса и смещения НС, используя алгоритм масштабируемых сопряженных градиентов;

- trainb - функция, обучающаяся с весом и уклоном, узнающим правила.

- trainc - циклически возрастающее обучение с изучением функций.

- trainr – функция случайно возрастающего обучения с изучением функций.

2. Выполнение работы

1. Ознакомиться с дополнительной литературой и информационными материалами, предоставленными для выполнения работы.
2. Записать в протоколе постановку задачи на разработку и исследование нейронной сети для распознавания 2-х мерных изображений объектов и исходные данные к этой задаче (по указанию преподавателя).
3. С помощью программы **Swear_rasp.exe** сформировать обучающую выборку, необходимую для обучения нейронной сети распознаванию заданных объектов.
4. Программой **Creation_nn.m** указав размерность оптического сенсора и количество нейронов второго слоя создать и обучить нейронную сеть.
5. С помощью программы **Calculation_nn.m** выполнить проверку надежности распознавания, используя файлы с данными для распознавания, получая их программой **Swear_rasp.exe** при различных положениях заданных объектов.
6. Заполнить таблицы результатов распознавания, рассчитав для каждого объекта вероятность его правильного распознавания.

Описать в протоколе выполнявшиеся действия и полученные результаты, сделать выводы по работе.

Таблица 1 – Таблица результатов

п/п	Наименование детали	Количество предъявленных для распознавания образов	Количество правильно распознанных образов	Вероятность правильного распознавания образа	Количество нейронов среднего слоя сети

3 Варианты задания

Таблица 2 – Варианты задания

Вариант	Объекты для распознавания	Размерность сетки датчиков	Шаг перемещения при получе-	Шаг угла поворота при по-	Количество нейронов

			нии выборки	лучении обучающей выборки	среднего слоя нейронной сети
1	Квадрат Треугольник Прямоугольник	5*5 9*9	25	60	5
2	Треугольник Прямоугольник Трапеция	4*5 10*8	50	15	8
3	Прямоугольник Трапеция Уголок	6*6 8*10	20	30	15
4	Трапеция Уголок Параллелограмм	4*4 7*7	50	30	10
5	Уголок Параллелограмм Многоугольник	5*5 10*10	20	60	20
6	Квадрат Параллелограмм Многоугольник	4*6 8*9	50	10	12
7	Квадрат Треугольник Многоугольник	6*4 10*10	25	60	8
8	Квадрат Треугольник Трапеция	4*4 9*9	50	15	10
9	Треугольник Прямоугольник Уголок	5*5 8*8	25	30	14

10	Прямо-угольник Трапеция Параллело-грамм	4*6 8*10	20	90	16
11	Трапеция Уголок Много-угольник	4*5 10*8	25	30	9
12	Квадрат Уголок Параллело-грамм	6*4 5*10	50	15	18
13	Треугольник Параллело-грамм Много-угольник	4*7 8*9	25	30	20
14	Квадрат Прямо-угольник Много-угольник	4*5 10*9	50	10	8
15	Квадрат Прямо-угольник Трапеция	6*6 9*9	20	90	15
16	Треугольник Трапеция Уголок	4*4 8*8	25	60	10
17	Прямо-угольник Уголок Параллело-грамм	5*5 9*9	20	30	11
18	Трапеция Параллело-грамм Много-угольник	4*10 6*6	50	15	15
19	Квадрат	4*5	25	60	7

	Уголок Много-угольник	9*8			
20	Квадрат Трапеция Много-угольник	5*5 8*8	50	30	8
21	Квадрат Треугольник Уголок	4*4 8*10	20	60	14
22	Треугольник Прямо-угольник Параллело-грамм	6*6 7*9	25	30	20
23	Прямо-угольник Трапеция Много-угольник	4*6 8*9	50	10	11
24	Квадрат Трапеция Уголок	5*7 10*10	25	90	16
25	Треугольник Трапеция Уголок	6*6 9*10	20	60	12

4 Содержание отчета

1 Цель работы
2 Задание
3 Таблица результатов
4 Выводы по работе

ЧАСТЬ 6. ЭКСПЕРТНЫЕ СИСТЕМЫ

Лабораторная работы №1. «Разработка экспертной системы управления мобильным роботом в среде CLIPS 6.3»

Цель работы:

- получение практических навыков проектирования экспертных систем для решения задач планирования действий, системы управления мобильным роботом.

- ознакомление с основными принципами программирования на языке CLIPS;

- приобретение навыков работы в программной средеCLIPS 6.3;

- разработка программы по планированию действий технической системой в робототехнике и мехатронике.

Основные теоретические сведения

Порядок работы экспертной системы состоит из трех основных этапов:

1.сопоставление фактов и правил;

2. выбор правила, подлежащего активизации;

3. выполнение действий, предписанных правилом.

Основные элементы программирования в CLIPS

В системеCLIPS(CLanguageIntegratedProductionSystem) используется, диалект языка LISP, направленный на создание экспертных систем, используя синтаксис и стиль программирования схожий с человеческой речью и логикой мышления.CLIPS поддерживает объектно-ориентированную и процедурную парадигму программирования.

Язык CLIPS включает три основных элемента для написания программ:

- Простые типы данных;
- Конструкции для пополнения базы знаний;
- Функции для манипулирования данными.

Простые типы данных

В системе определено восемь базовыхтипов данных: Float (числа с плавающей точкой), integer (целочисленные), symbol (неразрывная последовательность символов), string (набор символов заключенный в двойные кавычки), external-address, fact-addresinstance-name, instance-address.

Для записи чисел используются цифры (0-9), и символы (.), (+), (e).

Разделяющие символы: пробел, табуляция, двойные кавычки, (,), &, |, <, >, ;.

Символ (\) отменяет спец значение следом стоящего символа например: ("a and \"Ъ").

Работа с базой знаний. Факты

Факт представляет собой основную единицу данных (fact-list), используемую правилами. Каждому факту присваивается уникальный идентификатор факта. Индексы нумеруется с нуля.

Идентификатор факта имеет вид: f-n, где n – целое положительное число.

Пример:f-0 (todayisSunday) ;; факт№1 «сегоднявоскресение»

f-1 (weatheriswarm);; факт№2 «погодатеплая»

Факты представимы в двух форматах: позиционном и непозиционном.

В CLIPS существуют следующие зарезервированные слова, которые не могут использоваться как первое поле любого факта: test, and, or, not, declare,logical, object, exists, forall.

Непозиционные факты (шаблонные факты) - позволяют задавать имена каждому полю факта и затем обращаться к ним по имени.

Конструкция имеет общий вид:

(deftemplate<name>
(slot-1)
(slot-2)
…
(slot-N))

где <name> - имя шаблона, (slot-N) - именованное поле (или слот). Так же слоты могут быть ограничены по типу, значению, числовому диапазону, могут содержать значение по умолчанию.

Пример:

(deftemplatestudent ;;объявляем шаблон фактов с именем «student»
"astudentrecord" ;;строка с комментарием "запись о студенте"
(slotname (typeSTRING));; слот содержит шаблон факта с именем «name»
;;имеющий одинаргумент строчного типа

(slotage (typeNUMBER) (default 18))) ;;изначально, значение факта будет равно 18

Каждый слот начинается с ключевого слова slot или field. Слот включает поле данных, например name,тип данных, например STRING. Можно указать и значение по умолчанию (default).

Операции над фактами

В командной срокеCLIPS, чтобы добавить факт в fact-list (assert), удалить (retract), изменить (modify), дублировать (duplicate), вывести список фактов из памяти (facts), очистить базу фактов (clear), удалить массив фактов (undeffacts<listname>), подключить базу фактов (load), загрузить базу фактов в рабочую память(reset)нужно ввести соответствующую команду. Пример:

;;добавить факт «сегодня воскресение» в базу знаний
CLIPS> (assert (todayisSunday))
==>f-1 (todayisSunday) ;;факт успешно добавлен под индексом f-1
<Fact-1>

Команды, assert и retract, используются в программе для изменения базы фактов.Для использования базы фактов, нужно в начале ее подключить, затем загрузить в память.Команда reset сначала очищает базу фактов, а затем включает в нее факты из всех ранее загруженных массивов.

Работа с базой правил. Правила

В языке CLIPS правила имеют следующий формат:
(defrule<имя правила>
<предпосылка_1 >
<предпосылка_m>
=>
<действие_1 >
< действе_n>)

Параметр salience– хранит приоритет факта, принимает целочисленное значение в диапазоне (-10 ООО, 10 000). Поумолчанию равен нулю.

Переменные объявляются конструкцией ?<name>. Если <name> - не заполнено, то вместо "?" может быть сопоставлен любой элемент.

Пример:

(defrulepick-a-chore;;объявляемправилосименем «pick-a-chore»

"Choicedaysforthejob";; комментарий «Выбор дней для работ»
;;еслиусловия
(today is ?day)
(choreis ?job)
;;если условия буду сопоставлены с фактами
;;(todayisSunday) ;;сегодня – «воскресение»
;;(choreiscarwash) ;;случайная работа – «автомойка»
=>
;; то в случае активизации оно включит в базу новый факт (docarwashonSunday)
(assert (do ?job on ?day)))

Аналогично, правило удаление факта из базы
(defrule drop-a-chore ;;правилосименем «drop-a-chore»
(todayis ?day);;если условия буду сопоставлено с фактом (todayisSunday)
;;в переменную ?chore сохраняется указатель на факт (docarwashonSunday)
?chore <- (do ?job on ?day)
=>
(retract ?chore));;факт (docarwashonSunday) удаляетсяизбазы
где "<-" – оператор взятия адреса.

Функции для манипулирования данными. Определение функций

CLIPS поддерживаетпользовательские и системныетипы функций. Системные функции являются встроенными, а пользовательские определяются непосредственно в программе.

Математические и арифметические функции: +, -, *, /, ** (возведение в степень), Abs, Sqrt, Mod, Min, Max.

Пример 1:
(+ 2 5 8)

Логические операторы: | (ИЛИ), ~ (НЕ), & (И)

Оператор testслужит для выполнения условийсравнения,с использованием логических операторов (or, not, and), в левой части правила. (test (or (>?a7) (<?b-1))) ;; при ?a=8 и ?b=-2 – условие выполниться

Пользовательские функции объявляются командой deffunction.

(deffunction<имя функции> (<аргумент_1> ... < аргумент_n>)
<выражение1>
…
<выражение_n>)

Аргументами функции, могут быть данные простых типов, переменные или других функций.

Функция будет возвращать результат последнего выражения в списке.

Пример:

;; объявлена функция с именем betweenc аргументами ?lb, ?value, ?ub
(deffunction between (?l ?v ?r)
(or (> ?l ?v) (> ?v ?r)))

Эта функция возвращает true, если заданное целочисленное значение попало в диапазон между нижним и верхним пределами.

Чтобы присвоить переменной значение используется команда bind.

Пример: (bind ?a 4)

Практическая часть лабораторной работы

Решение задач на планирование

Задача планировщика - определить последовательность действий модуля принятия решений, системы управления мобильным роботом.

Порядоквыполнения

1.Изучить теоретический материал методических указаний.

2. Выполнить пример программы "Робот и ящик" в режиме отладки.(включить просмотр правил и фактов)

3. Модифицировать программу "Робот и ящик" таким образом, чтобы ящик необходимо было передвинуть в комнату С. Программу выполнять пошагово.

4. Составить программу на планирование действий робота согласно индивидуальному варианту задания.

Пример создания.

Рассмотрим ситуацию, когда есть две комнаты (А и В). В комнате «А» находится мобильный работ, а в комнате «В» расположен ящик. Задача робота, переместить ящик из «В» и «А».

Решение.

Зададим шаблон определяющий положение объекта:

```
(deftemplate in ;; имяшаблона «in»
;; Object – имя объекта (ящик или робот)
;; location – название локации (комнаты)
(slotobject (typeSYMBOL)) ;;имеет символьный тип, принимаемого значения
(slotlocation (type SYMBOL)) )
```

Определим шаблон цели действия робота.

```
;; task–задача
;; action – имя действия, которое нужно выполнить
;; object – имя объекта, который нужно переместить
;; from – название локации (комнаты) из какой переместить
;; to – название локации (комнаты) в какую переместить
(deftemplatetask
(slot action (type SYMBOL))
(slot object (type SYMBOL))
(slot from (type SYMBOL))
(slotto (typeSYMBOL)) )
```

На основе шаблонов in и taskзапишем факты о начальном состоянии системы:

```
(deffacts world
(in (object robot) (location RoomA)) ;; роботнаходитсявкомнатеА
(in (object box) (location RoomB)) ;; ящикнаходитсявкомнате B
;;задача: выполнить действие – перемещение ящика из комнаты А в В
(task(actiontransfer) (objectbox) (fromRoomB) (toRoomA)) )
```

Первые два фактазадают начальное положение робота и ящика, третий определяет задачу. Далее зададим необходимые правила поведения роботадля достижения цели. Весь процесс можно разбить на три этапа:

1. перемещение робота в комнату, где находится объект;
2. перемещение робота с объектом в комнату, определенную задачей;
3. завершение программы, если цель достигнута.

Опишем первое правило действия:

```
;; если ящик находится в другой комнате, то переместится туда.
(defrulemove
(task(object ?X) (from ?Y))
(in (object ?X) (location ?Y))
```

;; если робот не в комнате Y, сохраняется ссылка на факт in,

?robot-position <- (in (object robot) (location ~?Y))

=>

;;*то modify*изменит значение поля *location*, факта *in*

(modify ?robot-position (location ?Y)))

;; т. е. робот перемещается в комнату, где находится объект, который необходимо переместить.

Теперь создадим правило перемещения робота с ящиком, в заданную комнату:

(defruletransfer

(task(object ?X) (from ?Y) (to ?Z))*;;получениефактаизбазы*

?object-position <- (in (object ?X) (location ?Y)) *;;еслиящик ?X находитьсявкомнате ?Y*

?robot-position<- (in (objectrobot) (location ?Y))*;;еслироботнаходитьсявтой же комнате*

=>

;; то переместить робота и объект в комнату ?Z

;;то есть изменить у обоих фактов значение поля locationc ?Yна ?Z

(modify ?robot-position (location ?Z))

(modify ?object-position (location ?Z)))

Условием завершения работы будет являться наличие факта, что робот и ящик находятся к заданной комнате, указанной в слоте *to*.

(defrulestop

(declare (salience+1))*;;повышаем приоритет правила, чтобы оно проверялось в первую очередь*

;; если значения полей факта task совпадают со значениями факта in

(task(object ?X) (to ?Y))

(in (object ?X) (location ?Y))

=>

(halt))*;; halt – команда остановки программы*

Для выполнения программы необходимо сохранить код в файл с расширением (*.clp), загрузить программу для выполнения (load), очистить память и загрузить базу знаний (assert), запустить программу (run), по окончанию программы можно просмотреть итоговый список фактов (facts), либо в активировать просмотр фактов (Execution->Watch->Facts).

Варианты индивидуальных заданий.

В помещении,разделённом на комнаты: A,B,C,D,E, находится робот и три ящика. Задача робота собрать все ящики и сложить из них башню в комнате, где изначально находиться робот. Исходные данные указаны в таблице 1.

Таблица 1

№ варианта	Начальное местоположение			
	робота	ящиков		
		1	2	3
1	A	A	B	C
2	B	D	E	A
3	C	B	C	D
4	D	E	A	B
5	E	C	D	E
6	A	A		C
7	B	D	E	
8	C		C	D
9	D	E		B
10	E	A	B	C
11	A	D	E	A
12	B	B		D
13	C	E	A	B
14	D		D	E
15	E	A	B	

**номер задания в списке соответствует порядковому номеру студента в списке группы. Варианты заданий распределяются циклично.*

Содержание отчета.

В отчете необходимо указать: цель работы, описание индивидуального задания и код программы, результаты работы программы, выводы по работе.

Контрольные вопросы:

1. Что такое система, основанная на знаниях?
2. Формы представления фактов?
3. Способ описания непозиционных фактов в CLIPS?

3. Какие механизмы представления знаний поддерживает CLIPS?

3. Что входит в понятие "Коэффициент уверенности"?

4. Что содержит параметр salience?

5. В чем главная особенность языков программирования экспертных систем от других языков программирования?

Вычислительная процедура обучения с экспертом

Математический аппарат

Пусть имеется сложная система (научная, техническая, экономическая и т.д.), которая может характеризоваться определенным набором признаков X_k, $k = 1,2,3,\ldots$. Такой произвольный вектор значений признаков можно трактовать как образ, принадлежащий пространству признаков $\{X\}$.

Множество образов представляется в виде множества векторов, состоящего из k подмножеств или классов:

$z_1=\{X\}_1, \ldots, z_k=\{X\}_k$.

Для процедуры обучения с экспертом исходной информацией служат векторы значений признаков ситуаций по каждому из рассматриваемых эталонных классов и сформированная на основе мнения эксперта обучающая выборка. Исследуя и анализируя указанным образом, ряд таких систем с привлечением эксперта-человека, можно на основании его знаний и личного опыта выстроить классификацию и оценить, к какому из классов принадлежит исследуемый объект. Набор признаков системы, перечисленный выше, эксперт может оценивать либо по 10-бальной системе, либо в пределах от 0 до 1, как в рассматриваемом ниже примере.

На основе полученной информации и с учетом мнения эксперта формируется обучающая выборка, которая представлена в таблице 1.

Таблица 1. Обучающая выборка

Номер объекта	Значения признаков					Классификация эксперта
	z_1	z_2	z_3	...	z_n	
1	1	0,3	1		1	1
2	0,1	0,6		1	1	2
3	0,2	0,8	0,9		0,7	3
4	0,5	1	0,7		0,1	2
⋮						⋮
:						
L	1	1	0,5		0,1	

Задача обучения сводится к разбиению пространства признаков на классы (т.е. к проведению классификации), а задача распознавания сводится к определению класса $z_j = \{X\}_j$, $j=1,...,k$, с помощью векторной нормы:

$$z_k(X): \min_k ||X - \{X\}_k||.$$

В качестве векторной нормы могут быть использованы следующие известные нормы:

- евклидова норма $||X_i - X_{ki}||_E = (\Sigma_i(X_i - X_{ki})^2)^{1/2}$;
- норма расстояния $||X_i - X_{ki}||_M = (\Sigma_i|X_i - X_{ki}|)$;
- норма Чебышева $||X_i - X_{ki}||_C = \max_i|X_i - X_{ki}|$.

В вычислительных процедурах иммунокомпьютинга в качестве аналога расстояния используется понятие энергии связи, основанное на сингулярном разложении матрицы. Энергия связи между объектами A и M представляется следующим образом:

$$\omega_i = - U^T_i M V_i,\ U_i^T U_i = 1,\ V_i^T V_i = 1,\ i = 1, \quad , r,$$

где U_i, V_i – соответственно, правые и левые сингулярные векторы

матрицы A, r – ранг матрицы.

Алгоритм вычислительной процедуры обучения с экспертом состоит их следующих шагов:

Шаг 1. Сворачивание вектора в матрицу. Заданный вектор X размерности $(nx\ 1)$ сворачиваем в матрицу M размерности $n_U x n_V = n$.

Шаг 2. Формируем матрицы $A_1, A_2,\ldots,A_k$ для эталонных классов $c = 1,\ldots,к$ и вычисляем их сингулярные векторы:

$\{U_1, V_1\}$ – для A_1, $\{U_2, V_2\}$ - для A_2, $\{U_к, V_к\}$ - для A_k.

Шаг 3. Распознавание. Для каждого входного образа M вычисляем $к$ значений энергии связи между каждой парой сингулярных векторов:

$\omega_1 = - U^T_1 M V_1$, ….., $\omega_k = - U^T_к M V_к$.

Шаг 4. Определяем класс, к которому принадлежит входной образ M. Минимальное значение энергии связи ω^* определяет этот класс,:

$c = \omega^* = min_c\{ \omega_c \}$.

Лабораторная работа № 2 «Экспертные системы в MATLAB. Часть 1»

Цель работы: создание программного модуля для реализации вычислительной процедуры обучения с экспертом на основе инструментария универсальной системы MATLAB.

Порядок выполнения работы

1. Открыть универсальную систему MATLAB.
2. Задать обучающие матрицы A и матрицу M размерности (4×3).
3. Программно реализовать шаги 1-4 алгоритма вычислительной процедуры обучения с экспертом.
4. Сохранить все результаты выполнения работы в файле на диске.

Пример выполнения лабораторной работы №2

В соответствии с п. 2 формируем обучающие матрицы трех эталонных классов A_i ,$i=1,2,3$ размерности (4 x 3) и анализируемую матрицу M размерности (4 x 3).

```
>> A1=rand(4,3)

A1 =

    0.8147    0.6324    0.9575
    0.9058    0.0975    0.9649
    0.1270    0.2785    0.1576
    0.9134    0.5469    0.9706

>> A2=3*randn(4,3)

A2 =

   -1.2977   -3.4394    0.9819
   -4.9968    3.5727    0.5239
    0.3760    3.5675   -0.5601
    0.8630   -0.1129    2.1774

>> A3=2*randn(4,3)

A3 =

   -1.1766    2.1335    0.5888
    4.3664    0.1186   -2.6724
   -0.2728   -0.1913    1.4286
    0.2279   -1.6647    3.2471
```

Рисунок 1 Обучающие матрицы трёх эталонных классов.

Для сформированных обучающих эталонных матриц A_i, $i=1,2,3$ вычисляем первые левые и правые сингулярные векторы U_i, V_i, $i=1,2,3$.

```
>> M=0.5*rand(4,3)

M =

    0.4786    0.2109    0.3279
    0.2427    0.4579    0.0179
    0.4001    0.3961    0.4246
    0.0709    0.4797    0.4670

>> [U1,S1,V1]=svd(A1)

U1 =

   -0.5821   -0.4285    0.6528    0.2267
   -0.5355    0.7655    0.1390   -0.3284
   -0.1177   -0.4448   -0.0902   -0.8833
   -0.6004   -0.1802   -0.7392    0.2462


S1 =

    2.3951         0         0
         0    0.4444         0
         0         0    0.0400
         0         0         0


V1 =

   -0.6358    0.2773   -0.7204
   -0.3263   -0.9423   -0.0747
   -0.6995    0.1875    0.6896
```

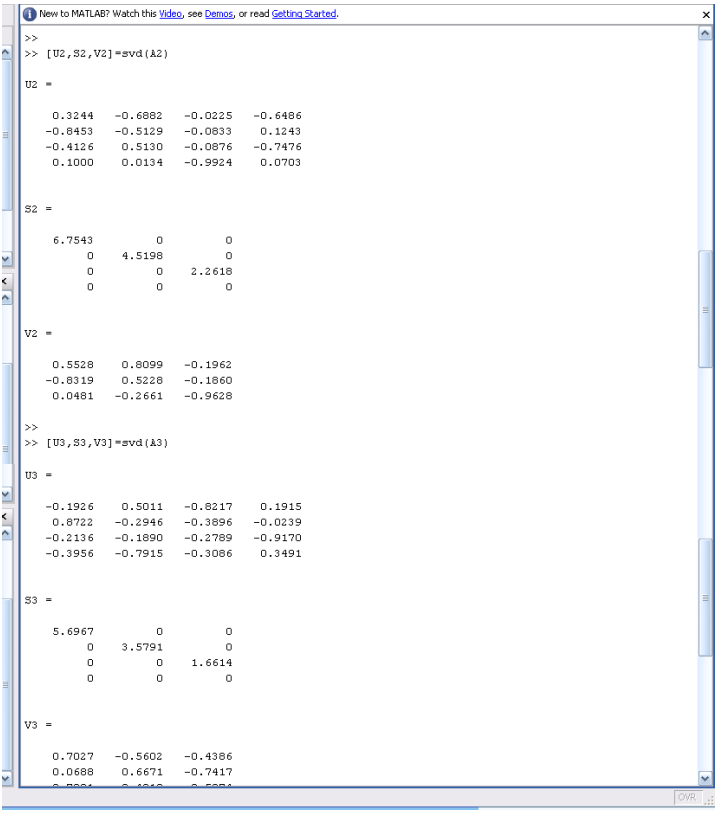


Рисунок 2.

Используя эти компоненты, вычисляем значения энергии связи анализируемой матрицы M с эталонными обучающими матрицами A_i

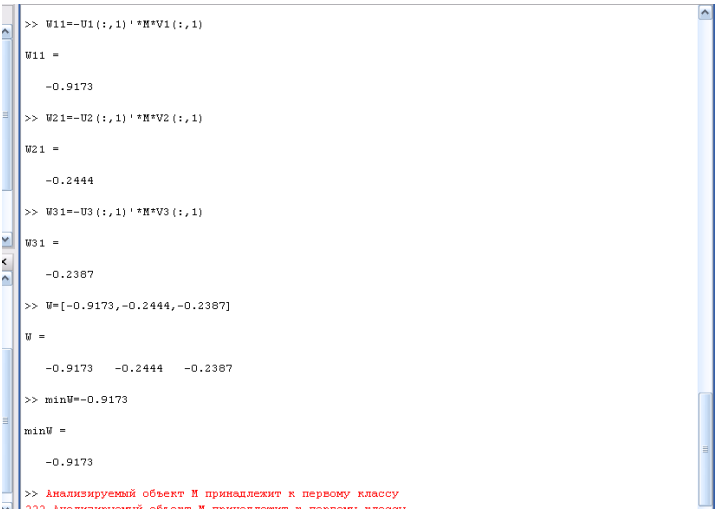


Рисунок 3.

Из вычисленных значений энергии связи формируем множество W=[W_{11}, W_{21},W_{31}], минимальное значение которого определяет принадлежность анализируемого объекта к соответствующему классу.

Вывод: анализируемый объект относится к первому классу.

Контрольные вопросы

1. Что такое энергия связи?
2. Роль эксперта при реализации вычислительной процедуры обучения с экспертом.
3. Что является мерой близости выбранного образа *M* к конкретному классу?

Вычислительная процедура обучения с экспертом

Математическая формализация задачи

Отнесение различных векторов индикаторов к одному классу может рассматриваться как агрегация исходных данных, при этом геометрическая близость (расстояние) между объектами

формализуется с помощью соответствующей векторной нормы. Наиболее распространенной является евклидова норма:

$$\left\| X_i - X_j \right\| = \left(\sum_k \left(X_{ik} - X_{jk} \right)^2 \right)^{1/2}.$$

Следовательно, задача обучения сводится к разбиению пространства индикаторов на классы (т.е. к проведению классификации), а задача распознавания сводится к определению класса (т.е. к проведению кластеризации) $z_j=\{X\}_j$, $j=1,\ldots,k$ с помощью выбранной векторной нормы:

$$Z_k(X) : min_k \left\| X - \{X\}_k \right\|.$$

Как известно, с помощью векторной нормы можно определить понятие расстояния между вектором и классом, а также понятие расстояния между классами.

С точки зрения распознавания образов полагается, что чем меньше значение выбранной нормы, тем сходство между объектами больше.

Определим образ как n-мерный вектор-столбец $X=[x_1, x_2,\ldots,x_n]$, где $x_1, x_2,\ldots,x_n$ являются вещественными числами, и индекс T является символом транспонирования матрицы.

Представим задачу распознавания образовкак отображение $f(X)\rightarrow\{1,\ldots,c\}$ любого образа X в одно из целых чисел $1,\ldots,c$, которые представляют классы.

Задача распознавания образов может быть сформулирована следующим образом:

Дано:

- Число классов c;
- набор из m обучающих образов: X1,….,Xm;;
- класс любого обучающего образа:f(X1)=c1…,f(Xm)=cm;
- произвольный n-мерный вектор P.

Найти:

Класс вектора P: $f(P)=?$.

Для реальных задач исходные данные в самом общем случае являются многомерными и допускают представление в виде массивов (векторов) вещественных и/или целых чисел. Как было отмечено выше, одной из основных особенностей ИК-алгоритма распознавания образов является проекция произвольных данных в пространство ФИС. Такое преобразование обладает следующими преимуществами:

- имеет строгое математическое обоснование в терминах сингулярного разложения матриц;
- существенно снижает размерность данных (до одно- двух- или трехмерного пространства ФИС);
- позволяет наглядно представить и визуализировать любую ситуацию как точку одно- двух- или трехмерного пространства.

Рассмотрим математическое описание основных процедур алгоритма иммунокомпьютинга.

1.1. Обучение

1. Сформировать обучающую матрицу $A=[X_1,...,X_m]^T$ размерности $m \times n$.
2. Вычислить максимальное сингулярное число s, а также левый и правый сингулярные векторы U и V обучающей матрицы по следующей итеративной (эволюционной) схеме:

$$\begin{cases} U_{(0)} = [1 \ldots 1]^T, \\ V_{(k)}{}^T = U_{(k-1)}^T A, \quad V_{(k)} = \dfrac{V}{\|V\|}, \\ U_{(k)} = AV_{(k)}, \quad U_{(k)} = \dfrac{U}{\|U\|}, \\ s_k = U_{(k)}^T A V_{(k)}, \ k = 1,2,..., \end{cases}$$

до выполнения условия $\left| s_{(k)} - s_{(k-1)} \right| < \varepsilon,$

$$s = s_{(k)}, \quad U = U_{(k)}, \quad V = V_k.$$

3. Хранить сингулярное число s.
4. Хранить правый сингулярный вектор V как *антитело-пробу*.
5. Для всякого $i=1,...,m$ хранить компоненту u_i левого сингулярного вектора U (как клетку формальной иммунной сети) и класс c_i, соответствующий обучающему образу X_i.

1.2. Распознавание

6. Для всякого n-мерного образа Z вычисляется *энергия связи* с V:

$$w(Z) = \frac{1}{s} Z^T V$$

(напомним, что *s*–хранимое сингулярное число, а *V*–это хранимый правый сингулярный вектор обучающей матрицы *A*).

7. Выбирается u_i, которая имеет минимальное расстояние (*близость*) с энергией связи *w*:

$$\min_i \left| w - u_i \right|, i = 1, \ldots, m.$$

8. Считать класс c_i искомым классом образа *Z*.

1.3. Замечания

Данное ядро алгоритма распознавания образов использует только максимальное (первое) сингулярное число *s* и соответствующие ему сингулярные векторы *U* и *V*, которые вычисляются на шаге 2 данного алгоритма.

В общем случае, рекомендуется использовать первые три сингулярных числа и соответствующие им сингулярные векторы обучающей матрицы. Они могут быть вычислены по той же итеративной схеме (шаг 2) с использованием вычислительной процедуры метода исчерпывания:

1.Вычислить максимальное сингулярное число s_1 и соответствующие ему сингулярные вектора U_1 и V_1 обучающей матрицы на шаге 2.

2.Сформировать матрицу $A_2=A-s_1U_1V_1$ и вычислить ее максимальное сингулярное число s_2 и соответствующие ему сингулярные векторы U_2 и V_2 посредством шага 2.

3.Сформировать матрицу $A_3=A - s_2U_2V_2$ и вычислить ее максимальное сингулярное число s_3 и соответствующие ему сингулярные векторы U_3 и V_3 посредством шага 2.

Затем вычислить 3 значения энергии связи с помощью шага 6:

$$w_1(Z) = \frac{1}{s_1} Z^T V_1, \quad w_2(Z) = \frac{1}{s_2} Z^T V_2, \quad w_3(Z) = \frac{1}{s_3} Z^T V_3.$$

Определить класс c_i на шаге 8 посредством определения минимального расстояния в трехмерном Евклидовом пространстве на шаге 7:

$$\min_i \sqrt{\left(w_1 - [u_i]_1\right)^2 + \left(w_2 - [u_i]_2\right)^2 + \left(w_3 - [u_i]_3\right)^2},$$

где $[u_i]_1, [u_i]_2, [u_i]_3$ являются *i*-ми компонентами левых сингулярных векторов U_1, U_2, U_3.

Отметим, что шаги 2 и 3 над матрицами A_2 и A_3 обеспечивают более точную аппроксимацию обучающей матрицы, согласно следующему свойству сингулярного разложения:

$$A = s_1 U_1 V_1^T + s_2 U_2 V_2^T + s_3 U_3 V_3^T + \ldots + s_r U_r V_r^T,$$

где r –ранг матрицы *A*.

Следует также отметить, что данный алгоритм иммунокомпьютинга может быть рассмотрен как «иммунный» алгоритм, так как любой образ может быть представлен как частный случай формального протеина и его распознавание основывается на энергии связи с антителом формального протеина.

Лабораторная работа № 3 «Экспертные системы в MATLAB. Часть 2»

Цель работы: создание программного модуля для реализации вычислительной процедуры обучения с экспертом на основе инструментария универсальной системы MATLAB.

Порядок выполнения работы

1. Открыть универсальную систему MATLAB.
2. Задать исходную матрицу*A* соответствующейразмерности и матрицу анализируемого объекта *M*. Программно реализовать шаги *1-8* алгоритма вычислительной процедуры обучения с экспертом.
3. Сохранить все результаты выполнения работы в файле на диске.

Пример выполнения лабораторной работы №3

Рассмотрим решение задачи классификации (задачи обучения с экспертом) на примере классификации легковых автомобилей.

Пусть имеется обучающая выборка в виде матрицы *R* размерности (3×33). Эта матрица в качестве элементов имеет инди-

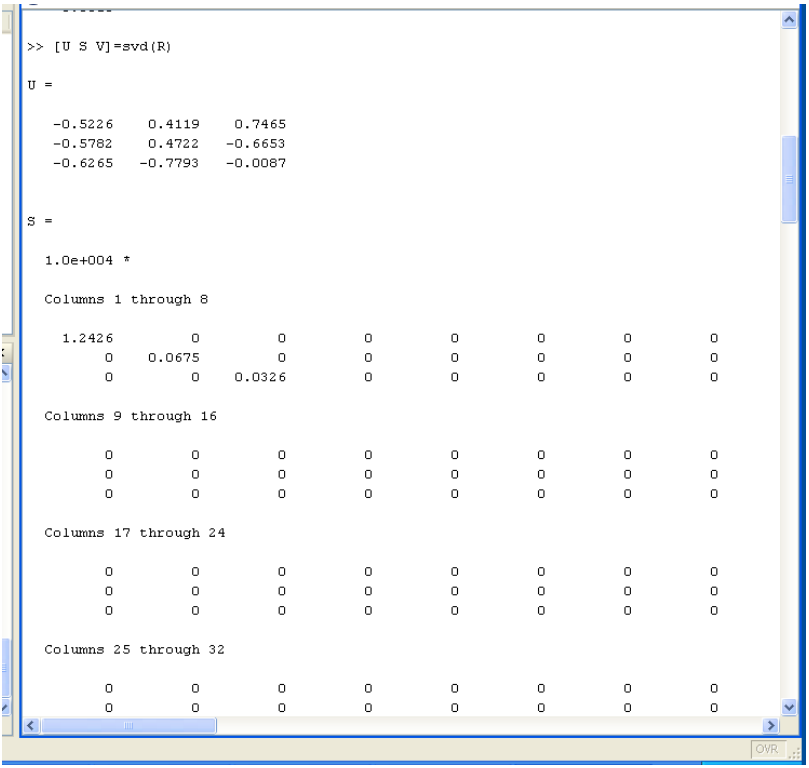
каторы, характеризующие три класса легковых автомобилей (второй, пятый и шестой):

```
R =

  1.0e+003 *

  Columns 1 through 8

    2.0000    0.0020    1.1400    1.6500    0.2020    0.0101    0.3500    4.1500
    1.9970    0.0030    1.3900    1.9700    0.2170    0.0095    0.5500    4.7950
    1.9980    0.0030    1.5400    2.1400    0.2360    0.0081    0.5250    5.0350

  Columns 9 through 16

    1.7350    1.4250    2.5150    1.5150    0.1400    0.0020    0.0010    1.7810
    1.8100    1.4500    2.7600    1.5400    0.1200    0.0030    0.0020    1.7810
    1.8800    1.4400    2.8800    1.5950    0.1200    0.0020    0.0020    2.7710

  Columns 17 through 24

    0.0103    0.0020    0.0200    0.1250    0.1700    0.0020    0.0010    0.0020
    0.0095    0.0020    0.0200    0.1500    0.2100    0.0020    0.0010    0.0030
    0.0106    0.0030    0.0300    0.1930    0.2800    0.0020    0.0010    0.0030

  Columns 25 through 32

    0.0020    0.0020    0.0020    0.0020    0.0118    0.0064    0.0084    0.0010
    0.0020    0.0020    0.0020    0.0020    0.0113    0.0066    0.0084    0.0010
    0.0020    0.0040    0.0020    0.0020    0.0142    0.0074    0.0099    0.0010

  Column 33

    0.0560
    0.0710
    0.0810
```

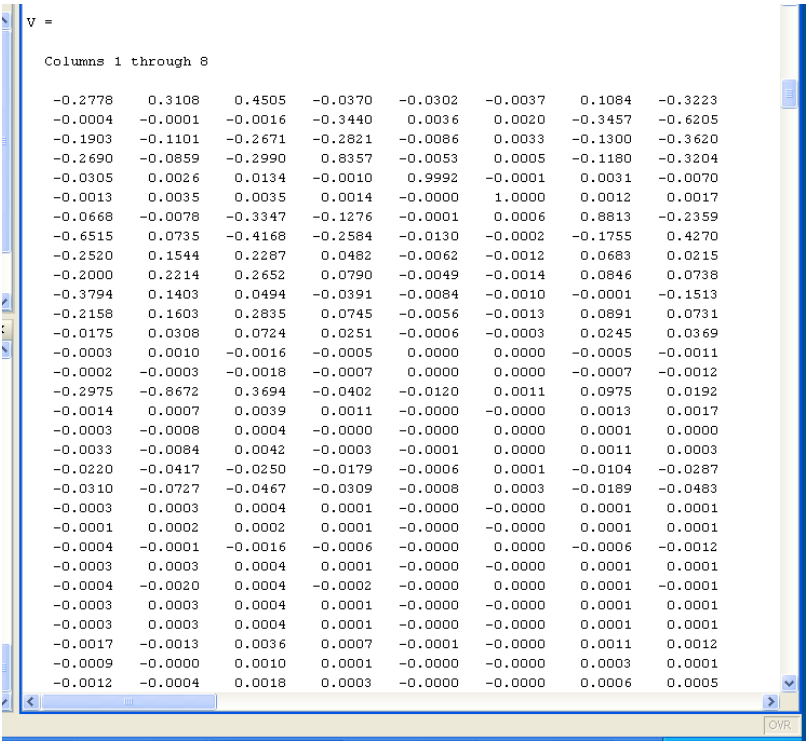
Рисунок 4. Обучающая выборка в виде матрицы *R* размерности (3×33).

Сингулярное разложение обучающей матрицы R осуществляется с помощью оператора: [USV]=svd(R).

Ниже представлены матрицы компонент сингулярного разложения:



```
>> [U S V]=svd(R)

U =

   -0.5226    0.4119    0.7465
   -0.5782    0.4722   -0.6653
   -0.6265   -0.7793   -0.0087


S =

  1.0e+004 *

  Columns 1 through 8

    1.2426         0         0         0         0         0         0         0
         0    0.0675         0         0         0         0         0         0
         0         0    0.0326         0         0         0         0         0

  Columns 9 through 16

         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0

  Columns 17 through 24

         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0

  Columns 25 through 32

         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
```



V =

Columns 1 through 8

-0.2778	0.3108	0.4505	-0.0370	-0.0302	-0.0037	0.1084	-0.3223
-0.0004	-0.0001	-0.0016	-0.3440	0.0036	0.0020	-0.3457	-0.6205
-0.1903	-0.1101	-0.2671	-0.2821	-0.0086	0.0033	-0.1300	-0.3620
-0.2690	-0.0859	-0.2990	0.8357	-0.0053	0.0005	-0.1180	-0.3204
-0.0305	0.0026	0.0134	-0.0010	0.9992	-0.0001	0.0031	-0.0070
-0.0013	0.0035	0.0035	0.0014	-0.0000	1.0000	0.0012	0.0017
-0.0668	-0.0078	-0.3347	-0.1276	-0.0001	0.0006	0.8813	-0.2359
-0.6515	0.0735	-0.4168	-0.2584	-0.0130	-0.0002	-0.1755	0.4270
-0.2520	0.1544	0.2287	0.0482	-0.0062	-0.0012	0.0683	0.0215
-0.2000	0.2214	0.2652	0.0790	-0.0049	-0.0014	0.0846	0.0738
-0.3794	0.1403	0.0494	-0.0391	-0.0084	-0.0010	-0.0001	-0.1513
-0.2158	0.1603	0.2835	0.0745	-0.0056	-0.0013	0.0891	0.0731
-0.0175	0.0308	0.0724	0.0251	-0.0006	-0.0003	0.0245	0.0369
-0.0003	0.0010	-0.0016	-0.0005	0.0000	0.0000	-0.0005	-0.0011
-0.0002	-0.0003	-0.0018	-0.0007	0.0000	0.0000	-0.0007	-0.0012
-0.2975	-0.8672	0.3694	-0.0402	-0.0120	0.0011	0.0975	0.0192
-0.0014	0.0007	0.0039	0.0011	-0.0000	-0.0000	0.0013	0.0017
-0.0003	-0.0008	0.0004	-0.0000	-0.0000	0.0000	0.0001	0.0000
-0.0033	-0.0084	0.0042	-0.0003	-0.0001	0.0000	0.0011	0.0003
-0.0220	-0.0417	-0.0250	-0.0179	-0.0006	0.0001	-0.0104	-0.0287
-0.0310	-0.0727	-0.0467	-0.0309	-0.0008	0.0003	-0.0189	-0.0483
-0.0003	0.0003	0.0004	0.0001	-0.0000	-0.0000	0.0001	0.0001
-0.0001	0.0002	0.0002	0.0001	-0.0000	-0.0000	0.0001	0.0001
-0.0004	-0.0001	-0.0016	-0.0006	-0.0000	0.0000	-0.0006	-0.0012
-0.0003	0.0003	0.0004	0.0001	-0.0000	-0.0000	0.0001	0.0001
-0.0004	-0.0020	0.0004	-0.0002	-0.0000	0.0000	0.0001	-0.0001
-0.0003	0.0003	0.0004	0.0001	-0.0000	-0.0000	0.0001	0.0001
-0.0003	0.0003	0.0004	0.0001	-0.0000	-0.0000	0.0001	0.0001
-0.0017	-0.0013	0.0036	0.0007	-0.0001	-0.0000	0.0011	0.0012
-0.0009	-0.0000	0.0010	0.0001	-0.0000	-0.0000	0.0003	0.0001
-0.0012	-0.0004	0.0018	0.0003	-0.0000	-0.0000	0.0006	0.0005

Рисунок 5. Матрицы компонент сингулярного разложения

Анализируемый объект представляется в виде вектора M:

M=[2000 3 1375 1925 222 10.1 445 4547 1766 1428 2650 1528 106 3 2 1781 9.5 2 20 150 210 2 1 3 2 2 2 2 11.3 6.4 9.2 1 71].

Для этого вектора M вычисляем значения энергии связи и расстояние до соответствующих точек для 2, 5, 6, классов в пространстве ФИС.

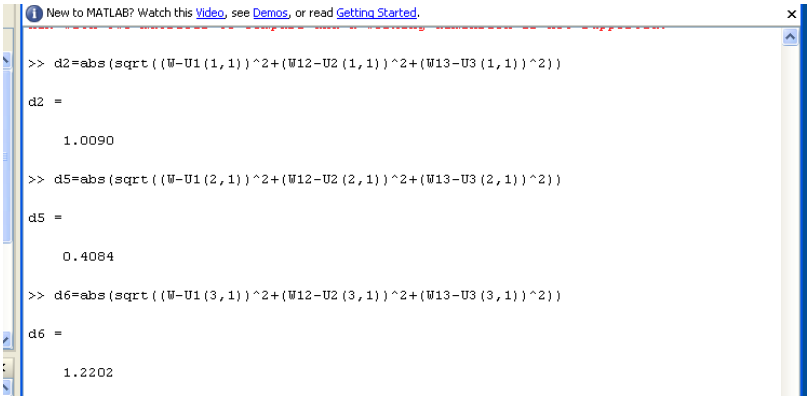


Рисунок 6. Результаты вычисленийзначений энергии

Из полученного множества значений расстояний от точки в пространстве ФИС, характеризуемой анализируемый объект до точек, характеризующих соответственно, классы 2, 5, 6:

d={d2,d5,d6}={1.0090, 0.4082, 1.2202}

выбираем наименьшее значение, которое равно 0.4082. Это значение определяет принадлежность анализируемого объекта к 5 классу.

Вывод: анализируемый объект принадлежит к 5 классу.

Контрольные вопросы

1. Какая норма была использована при проведении классификации?
2. Сущность вычислительной процедуры автоматической классификации.
3. Можно ли назвать рассмотренную процедуру обучения с экспертом интеллектуальной процедурой?

СПИСОК ЛИТЕРАТУРЫ

1. Леоненков А.В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. – СПб: БХВ-Петербург, 2005. –218-221с.

2. Прикладные нечеткие системы: Перевод с япон./ К. Асаи, Д. Ватада, С. Иваи и др.; под ред. Т. Тэрано, К. Асаи, М. Сугено.: Мир, 1993.

3. Дьяконов В., Круглов В. Математические пакеты расширения MATLAB: специальный справочник. – СПб.: Питер, 2001.

4. Заде Л. Понятие лингвистической переменной и ее применение к принятию приближенных решений. М.: Мир, 1976. 165 с.

5. CLIPSatoolforbuildingexpertsystems [электронный источник] - Режим доступа: http://clipsrules.sourceforge.net, свободный.

6. Джозеф Джарратано - Экспертные системы. Принципы разработки и программирование, 4е изд – 2007

7. В.Л. Афонин, В.А. Макушин – Интеллектуальные робототехнические системы. Курс лекций. Учебное пособие. Москва. 2005.

8. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. СПб.: Питер, 2001. с. 384.

9. А. П. Частиков, Д. Л. Белов, Т. А. Гаврилова - Разработка экспертных систем. Среда CLIPS. БХВ-Петербург. 2003