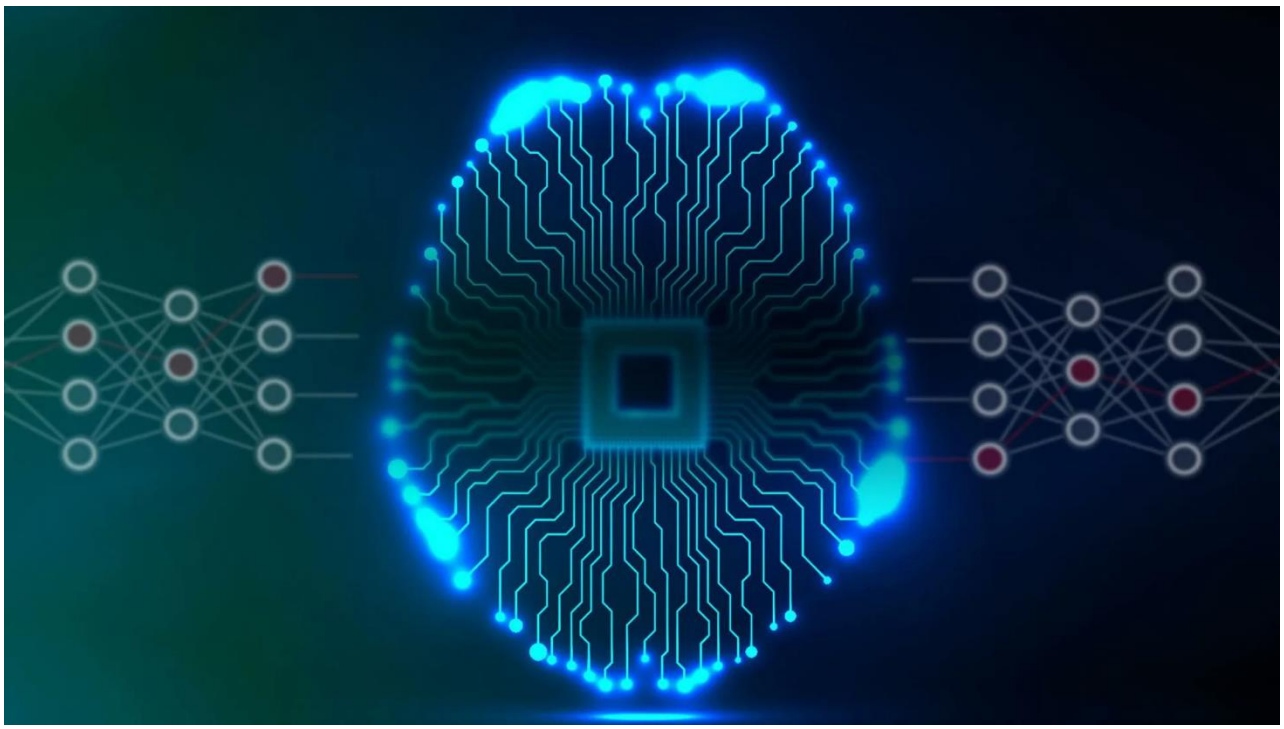
МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ
ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ
имени А.А. ЕЖЕВСКОГО

Институт экономики, управления и прикладной информатики
Кафедра информатики и математического моделирования

Нечеткая логика и нейронные сети
Учебное пособие



Молодежный
2023

УДК 004.032.26(075.8)
Н 59

Печатается по решению научно-методического совета Иркутского государственного аграрного университета имени А.А. Ежевского (протокол № 3 от 25.12. 2023 г.)

Составитель:
канд. тех. наук, доцент М. Н. Полковская

Рецензенты:
Руководитель группы автоматизации управления финансами департамента консолидации управленческой отчетности и автоматизации ООО «ИНК» Т. Р. Галимзянов;
канд. тех. наук, доцент кафедры электроснабжения и физики Иркутского ГАУ А. Ю. Логинов

Н 59 **Нечеткая логика и нейронные сети :** учебное пособие для студентов направления подготовки 09.03.03 Прикладная информатика / Иркут. гос. аграр. ун-т им. А. А. Ежевского ; сост. М. Н. Полковская. - Молодежный : Изд-во ИрГАУ, 2023. - 116 с.

УДК 004.032.26(075.8)

Учебно-методическое пособие содержит теоретический материал и практические задания, посвященные проектированию моделей нечеткого вывода Мамдани и ANFIS. Кроме того, в практической части содержатся задания, посвященные обучению нейронной сети на основе реальных данных с помощью средств программы Matlab. Задания, описанные в практической части, выполняются по вариантам, предлагаемым преподавателем. Помимо этого, приведены различные типовые задания (тесты), позволяющие оценить качество освоения изученного теоретического и практического материала.

Работа предназначена для бакалавров направления подготовки 09.03.03 Прикладная информатика. Кроме того, она может быть полезна магистрантам и аспирантам, занимающимся изучением нечеткой логики и нейронных сетей.

Оглавление

1. ЦЕЛЬ И ЗАДАЧИ

Цель освоения дисциплины:

- сформировать способности ориентироваться во всем многообразии методов построения нейронных сетей;
- использовать терминологию, относящуюся к нейронным сетям;
- использовать терминологию, относящуюся к нечеткой логике;
- разрабатывать архитектуру основных нейронных сетей;
- создавать алгоритмы обучения основных классов нейронных сетей.

Основные задачи освоения дисциплины:

- Основные задачи освоения дисциплины: сформировать способности ориентироваться во всем многообразии методов построения нейронных сетей; использовать терминологию, относящуюся к нейронным сетям; использовать терминологию, относящуюся к нечеткой логике; разрабатывать архитектуру основных нейронных сетей; создавать алгоритмы обучения основных классов нейронных сетей.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Дисциплина «Нечеткая логика и нейронные сети; 09.03.03 - Прикладная информатика; Прикладная информатика в АПК; (ФГОС3++);» находится в дисциплин по выбору б1.в.дв.3 Б1.В.ДВ.03 учебного плана по направлению подготовки 09.03.03 Прикладная информатика. Дисциплина изучается в 7 семестре.

3. ТРЕБОВАНИЯ К УСЛОВИЯМ РЕАЛИЗАЦИИ ДИСЦИПЛИНЫ (ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, СООТНЕСЕННЫХ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ)

Изучение данной дисциплины направлено на формирование у обучающихся следующих компетенций, соотнесенных с индикаторами достижения компетенций:

Код компетенции	**Результаты освоения ОП**	**Индикаторы компетенции**	**Перечень планируемых результатов обучения по дисциплине**
ПК-1	Способность проводить обследование организаций, выявлять информационные потребности пользователей, формировать требования к информационной системе.	ИД-3ПК-1 Применяет методику проведения об-следования организации и выявления информационных потребностей пользователей	Знать: методику проведения обследования организации и выявления информационных потребностей пользователей Уметь: выявлять информационные потребности пользователей Владеть: методикой проведения обследования организации и выявления информационных потребностей пользователей
ПК-3	Способность проектировать	ИД-1ПК-3 Использует ме-	Знать: методологии и средства проектирования

	информационные системы по видам обеспечения.	тодологии и средства проектирования ИС	ИС Уметь: использовать методоло-гии и средства проектирования ИС Владеть: методологией и сред-ствами проектирования ИС

4. ТИПОВЫЕ КОНТРОЛЬНЫЕ ЗАДАНИЯ ИЛИ ИНЫЕ МАТЕРИАЛЫ, НЕОБХОДИМЫЕ ДЛЯ ОЦЕНКИ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ (ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ) ПО ДИСЦИПЛИНЕ, ХАРАКТЕРИЗУЮЩИХ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ И (ИЛИ) ДЛЯ ИТОГОВОГО КОНТРОЛЯ СФОРМИРОВАННОСТИ КОМПЕТЕНЦИИ

4.1. Примерный перечень вопросов к зачету для оценивания результатов обучения в виде ЗНАНИЙ (ПК-1, ПК-3).

1. Что такое семантическая сеть и для чего ее применяют?
2. В чем состоит идея создания семантической сети?
3. Каким образом представляются данные в семантической сети?
4. Существуют ли ограничения на число связей элементов, свойств и сложность при построении семантической сети?
5. Какие отношения предложены в качестве операторов отношения для группировки вершин?
6. Что представляет из себя фрейм, его составные части?
7. Что такое слот и из каких частей он состоит?
8. Для чего служат имя фрейма и имя слота?
9. Для чего служат указатели наследования?
10. для чего служат указание типа данных, демон?
11. Для чего служат присоединенная процедура и значение слота?
12. Что такое правила продукции и в чем их сущность?
13. В чем отличие прямой цепочки рассуждений от обратной цепочки рассуждений?
14. Из каких частей состоит продукционная система?
15. Значение и применение частей продукционной системы для представления знаний?
16. Чем отличаются «прямая» и «обратная» цепочки рассуждений?

17. Какие виды правил существуют?
18. Как контролируется вывод правил из БЗ?
19. Как учитывается достоверность заключительной части правила?
20. Что такое экспертная система?
21. Как функционирует экспертная система? Какие функции выполняет каждый элемент системы?
22. К какому типу относятся ядра продукций в разработанной Вами ЭС?
23. Приведите пример одного продукционного правила, соответствующего разработанной базе знаний.
24. Как осуществляется приобретение знаний в разработанной ЭС?
25. В чем отличие однозначных и альтернативных продукционных правил?
26. Что такое нечеткая логика? В каких приложениях используется?
27. Операции над нечеткими множествами.
28. Способы задания нечеткого множества.
29. Функция принадлежности. Типы функций принадлежности.
30. Алгоритмы нечеткого вывода. Алгоритм Мамдани.
31. Алгоритм Сугено.
32. Экспертные системы на основе использования нечеткой логики.
33. Гибридные сети.

4.2. Примерный перечень простых контрольных заданий к зачету в форме тестов для оценивания результатов обучения в виде УМЕНИЙ.

Комплексное компьютерное тестирование

Тест 1- Тип ответа: Одиночный выбор (ПК-1, ПК-3**)**

1. Временной признак учитывается в экспертных системах:

a)динамических

b)детерминированных

c)аналитических

2. Выберите наиболее точное определение базы знаний:

a)совокупность правил принятия решений

b)совокупность единиц знаний, отражающих факты и зависимости фактов

c)совокупность описаний объектов и их связей

3. Экспертная система состоит из:

a)интеллектуального интерфейса

b)базы знаний

c)механизма вывода заключений

d)интеллектуального интерфейса, базы знаний и механизма вывода заключений

4. Центральным компонентом экспертной системы является:

a)БД

b)Интеллектуальный интерфейс

c)БЗ

5)Наибольшую стоимость имеет:

a)база знаний

b)механизм вывода

c)интеллектуальный интерфейс

d)механизм приобретения знаний

6)Процедура, выполняющая интерпретацию запроса пользователя к БЗ и формирующая ответ в удобной для него форме, – это:

a)механизм объяснения

b)интеллектуальный интерфейс

c)механизм приобретения знаний

d)механизм вывода

7)Механизм вывода:

a)обосновывает решение

b)формирует решение

c)выполняет решение

d)формирует и выполняет решение

8 Свойство автоматических систем брать на себя отдельные функции интеллекта человека, например, выбирать и принимать оптимальные решения на основе ранее полученного опыта и рационального анализа внешних воздействий – это…

a) искусственный интеллект;
b) нечеткая логика;
c) нейронные сети.

9 В нечеткой логике используется величина степень истинности, принимающая любые значения из бесконечного множества от

a) -∞ до +∞
b) -1 до 1
c) 0 до 1

10 Ориентированный граф, вершины которого отображают некоторые понятия, а дуги – отношения между ними– это…

a) фреймовая модель
b) семантическая сеть
c) продукционная модель

11 Переменная, значениями которой являются не числа, а слова естественного языка, называемые термами – это…

a) лингвистическая переменная
b) логическая переменная
c) продукция

12 Термин "искусственный интеллект" предложил...

a) Д. Маккартни

b) А. Тьюринг

c) Г. Розенблатт

d) П. МакКаллок

13 Начало исследований в области искусственного интеллекта относится:

a) конец 40-х годов 20 века

b) конец 60-х годов 20 века

c) конец 70-х годов 20 века

d) конец 50-х годов 20 века

14 Что есть у нейрона?

a. Один вход, много выходов

b. Много входов, много выходов

c. Один вход, один выход

d. Много входов, один выход

15 Требуется ли алгоритмизация процесса решения задач при построении нейронных сетей?

a. Нет, и в этом - главное преимущество нейронных сетей

b. Да, но в данном случае она производится экспертами

c. Да, иначе сеть не построить

16 Каково главное преимущество нейронных сетей?

a. Простота реализации

b. Возможность самообучения

c. Нелинейность

Тест 2 – Установить соответствие или порядок действий (ПК-1, ПК-3**)**

1)Установите соответствие:

1. системы, основанные на прецедентах
2. многоагентные системы
3. гипертекстовые системы

Варианты:

a)динамические экспертные системы

b)самообучающиеся ИС

c)системы с интеллектуальным интерфейсом

2) Установите соответствие:

1. индуктивные системы
2. классифицирующие системы
3. контекстные системы помощи

Варианты:

a)экспертные системы

b)самообучающиеся ИС

c)системы с интеллектуальным интерфейсом

3) Установите соответствие:

1. многоагентные системы
2. нейросетевые системы
3. системы с когнитивной графикой

Варианты:

a)экспертные системы

b)самообучающиеся ИС

c)системы с интеллектуальным интерфейсом

4) Установите соответствие:

1. интеллектуальные базы данных
2. динамические системы
3. нейронные сети

Варианты:

a)экспертные системы

b)самообучающиеся ИС

c)системы с интеллектуальным интерфейсом

5) Установите соответствие:

1. системы интеллектуального анализа данных
2. гипертекстовые системы
3. динамические системы

Варианты:

a)экспертные системы

b)самообучающиеся ИС

c)системы с интеллектуальным интерфейсом

6) Установите соответствие:

1. системы, основанные на прецедентах
2. гипертекстовые системы
3. классифицирующие системы

Варианты:

a) экспертные системы
b) самообучающиеся ИС

c) системы с интеллектуальным интерфейсом

7) Установите соответствие:

1. системы с естественно-языковым интерфейсом
2. системы интеллектуального анализа данных
3. классифицирующие системы

Варианты:

a)экспертные системы

b)самообучающиеся ИС

c)системы с интеллектуальным интерфейсом

8)Установите соответствие:

1. ИИС, предназначенная для поиска неявной информации в базе данных или тексте для произвольных запросов, составляемых на ограниченном естественном языке
2. ИИС, предназначенная для решения слабоформализуемых задач на основе накапливаемого в базе знаний опыта работы эксперта в проблемной области
3. ИИС, предназначенная для автоматического формирования единиц знаний на основе примеров реальной практики

Варианты:

a)экспертная система

b)система с интеллектуальным интерфейсом

c)самообучающаяся система

9)Установите порядок этапов проектирование экспертной системы:

a)концептуализация проблемной области

b)идентификация проблемной области

c)реализация экспертной системы

d)формализация базы знаний

e)тестирование экспертной системы

10.Укажите соответствие для всех 5 вариантов ответа:

1) информационно-поисковая система	a) Информационная библиотечная система
2) управляющая информационная система	b) Медицинские информационные системы
3) интеллектуальная информационная система	c) Компьютеризированная продажа железнодорожных билетов
	d) Система бухгалтерского учета
	e) Система оперативного планирования выпуска продукции

11.Установите соответствие

1) Интеллектуальные базы данных	a) Динамические системы
2) Системы с интеллектуальным интерфейсом	b) Экспертные системы
3) Нейронные сети	c) Самообучающиеся ИС

12.Установите соответствие:

1) системы, основанные на прецедентах	a) самообучающиеся ИС
2) многоагентные системы	b) динамические экспертные системы
3) гипертекстовые системы	c) системы с интеллектуальным интерфейсом

13.Установите соответствие:

1) индуктивные системы	a) самообучающиеся ИС
2) классифицирующие системы	b) экспертные системы
3) контекстные системы помощи	c) системы с интеллектуальным интерфейсом

14.Установите соответствие:

1) многоагентные системы	a) экспертные системы
2) нейросетевые системы	b) самообучающиеся ИС
3) системы с когнитивной графикой	c) системы с интеллектуальным интерфейсом

15. Установите соответствие:

1) системы с естественно-языковым интерфейсом	a) системы с интеллектуальным интерфейсом
2) системы интеллектуального анализа данных	b) самообучающиеся ИС
3) классифицирующие системы	c) экспертные системы

16. Установите соответствие определений логического вывода

1) от общего к частному	a) дедуктивный вывод
2) от частного к общему	b) индуктивный вывод
3) от частного к частному	c) абдуктивный вывод

Тест 3 - укажите 2 и более варианта ответа (ОПК-2, 3, 4, 8**)**

1)К системам с интеллектуальным интерфейсом относят:

а)интеллектуальные базы данных

b)системы, основанные на прецедентах

c)гипертекстовые системы

d)прикладные программы

e)системы когнитивной графики

2)Назовите основные компоненты экспертной системы:

a)СУБД

b)интеллектуальный интерфейс

c)механизм вывода

d)прикладная программа

e)механизм объяснения

f)база знаний

g)программа вывода результата

h)механизм приобретения знаний

3)В инструментальную среду экспертной системы обязательно входят:

a)механизм вывода знаний

b)механизм доступа к данным

c)механизм приобретения знаний

d)механизм интервьюирования экспертов

e)механизм тестирования знаний

f)механизм объяснения

g)интеллектуальный интерфейс

h)интерфейс с информационной системой

4)В состав экспертной системы входят:

a)механизм приобретения знаний

b)база знаний

c)механизм вывода заключений

d)база данных

5. В области искусственного интеллекта решаются следующие задачи (несколько вариантов):

a) Представление знаний и разработка систем, основанных на знаниях

b) Разработка естественно-языковых интерфейсов и машинный перевод

c) Игры и творчество

d) Распознавание образов

e) Разработка баз данных

f) Вычислительные задачи

6. В создании ЭС участвует …

a) заказчик;

b) инженер по знаниям;

c) пользователь;

d) эксперт;

e) учредитель.

7. Признаками определения интеллектуальности информационной системы являются:

a) самообучаемость

b) коммуникативность

c) эффективность

d) решение сложных задач

e) нет правильного ответа

8. Основные преимущества продукционных систем?

a) простота и гибкость выделения знаний;

b) зависимость знаний от программы поиска;

c) простота наращивания базы знаний;

d) мощный механизм математического вывода;

e) возможность трассировки “цепочки рассуждений”;

9. Какие НЕ-факторы изучаются в системах представления знаний?

a) Неполнота

b) Неточность

c) Незнание

d) Неоднозначность

e) Неэффективность

10. Что такое экспертные системы?

a) Система, реализующая метод экспертных оценок

b) Система, принимающая решения

c) Система, принимающая решения на уровне эксперта-профессионала

d) Система, основанная на знаниях специалистов

11. Укажите, при каком из условий разработка ЭС оправдана:

a) Задача требует оперирование символами

b) Задача требует эвристических решений

c) Задача не слишком проста

d) Задача представляет практический интерес

e) Задача имеет размеры, допускающие реализацию

12. ЭС получили широкое распространение потому, что человеческие знания:

a) Дорогие

b) Труднопередаваемые

c) Легко документируемые

d) Непрочные

13. В чем состоят интеллектуальные функции технологических систем.

a) Коррекция и выбор режимов обработки

b) Расчет погрешности рассогласования между программным и реальным движением инструмента

c) Задание программного движения инструмента

d) Выбор цели и коррекция программы движения инструмента.

14. Особенности интеллектуальных систем, отличающие их от адаптивных систем.

a) Наличие модели внешней среды

b) Коррекция ошибки движения

c) Приспособление к внешней информации

d) Выбор цели на основе проигрывания ситуации на модели внешней среды

15. Функции адаптивных систем.

a) Наличие модели внешней среды

b) Коррекция ошибки между программным и реальным движениями

c) Приспособление к информации о внешней среде

d) Выбор цели на основе проигрывания ситуации на модели внешней среды

16. Степенью принадлежности элемента x называется:

a) характеристика, показывающая в какой степени x является элементом данного нечеткого множества

b) значение функции принадлежности, вычисленной на аргументе x

c) вероятность обладания элементом x свойством, характеризующим данное нечеткое множество

17. Лингвистическая переменная называется структурированной, если:

a) ее семантическое правило можно задать алгоритмически
b) ее синтаксическое правило можно задать алгоритмически
c) ее терм-множество можно задать перечислением
d) ее терм-множество образует некоторую алгебраическую структуру

ОСНОВЫ НЕЧЕТКОЙ ЛОГИКИ. ЛОГИКО-ЛИНГВИСТИЧЕСКИЕ МОДЕЛИ

1.1. Источники неопределенности знаний и данных

Во многих реальных приложениях приходится сталкиваться с ситуацией, описываемой неточной информацией. Множество источников неопределенности используемой информации в большинстве случаев можно разделить на две категории: 1) Недостаточно полное знание предметной области. 2) Недостаточная информация о конкретной ситуации. Знания о предметной области могут быть неясными или неполными, могут использоваться недостаточно четко сформулированные концепции или недостаточно изученные явления. Например, в диагностике психических заболеваний существует несколько отличающихся теорий о происхождении и симптоматике шизофрении.

Располагая неполным знанием, мы не можем уверенно предсказать, какой эффект даст то или иное действие. Например, терапия, использующая новые препараты дает совершенно неожиданные результаты. И, наконец, даже когда мы располагаем достаточно полной теорией предметной области, эксперт может посчитать, что эффективно использовать не точные, а эвристические методы. Так, методика устранения неисправности в электронном блоке путем замены подозрительных узлов оказывается значительно более эффективной, чем скрупулезный анализ цепей в поиске детали, вышедшей из строя. Но помимо неточных знаний, неопределенность может быть внесена неточными или ненадежными данными о конкретной ситуации.

Любой сенсор имеет ограниченную разрешающую способность, при составлении отчетов могут быть допущены ошибки или в них могут попасть недостоверные сведения. На практике далеко не всегда можно получить полные ответы на поставленные вопросы и, хотя можно воспользоваться различного рода дополнительной информации, например, о пациенте с помощью дорогостоящих процедур или хирургическим путем, такие методики

используются крайне редко из-за высокой стоимости и рискованности. Помимо всего прочего, существует еще и фактор времени. Не всегда есть возможность быстро получить необходимые данные, когда ситуация требует принятия срочного решения. Если работа ядерного реактора вызывает подозрение, вряд ли кто-нибудь будет ждать окончания всего комплекса проверок, прежде чем принимать решение о его остановке.

Суммируя все вышесказанное, отметим, что эксперты пользуются неточными методами по двум главным причинам: 1) точных методов не существует; 2) точные методы существуют, но не могут быть применены на практике из-за отсутствия необходимого объема данных или невозможности их накопления по соображениям стоимости, риска или из-за отсутствия времени на сбор необходимой информации.

1.2. Представление неопределенности знаний и данных

Большинство исследователей в области искусственного интеллекта давно пришли к единому мнению, что неточные методы играют важную роль в разработке экспертных систем, но много споров вызывает вопрос, какие именно методы должны использоваться. До последнего времени многие соглашались с утверждениями Мак-Карти и Хайеса, что теория вероятностей не является адекватным инструментом для решения задач представления неопределенности знаний и данных [McCarthy and Hayes,1969].

Выдвигались следующие аргументы в пользу такого мнения:

• теория вероятности не дает ответа на вопрос, как комбинировать вероятности с количественными данными;

• назначение вероятности определенным событиям требует информации, которой мы просто не располагаем;

• непонятно, как количественно оценивать такие часто встречающиеся на практике понятия, как «в большинстве случаев», «в редких случаях», или такие приблизительные оценки, как «старый» или «высокий»;

• применение теории вероятности требует «слишком много чисел», что вынуждает инженеров давать точные оценки тем параметрам, которые они не могут оценить.

Все эти соображения породили новые формальные аппарат для работы с неопределенностями, который получил название нечеткая логика (fuzzy logic). То знание, которое эксперт использует при оценке каких-либо параметров, обычно базируется скорее на отношениях между классами данных и классами гипотез, чем на отношениях между отдельными данными и конкретными гипотезами. Большинство методик решения проблем в той или иной сфере включает классификацию данных (сигналов, симптомов и т.п.), которые рассматриваются как конкретные представители некоторых более общих категорий. Редко когда эти более общие категории могут быть четко очерчены. Конкретный объект может обладать частью характерных признаков определенной категории, а частью не обладать, принадлежность конкретного объекта к определенному классу может быть размыта.

Предложенная Заде [Zadeh, 1965] теория нечетких множеств (fuzzy set theory) представляет собой формализм, предназначенный для формирования суждений о таких категориях и принадлежащих к ним объектах. Эта теория лежит в основе теории нечеткой логики (fuzzy logic) [Zadeh, 1975]. Привлекательность нечеткой логики для проектировщиков экспертных систем состоит в ее близости к естественному языку. Таким терминам, как «быстрый», «немного», чаще всего дается интерпретация на основе повседневного опыта и интуиции. Это упрощает процесс инженерии знаний, поскольку подобные суждения человека– эксперта можно непосредственно преобразовать в выражения нечеткой логики.

1.3. Нечеткая логика и ее применение. Нечеткий вывод.

Классическая или булева логика имеет один существенный недостаток - с ее помощью невозможно описать ассоциативное мышление человека. Она

оперирует только двумя понятиями: ИСТИНА и ЛОЖЬ, исключая любые промежуточные значения. Аналогично этому булева логика не признает ничего кроме единиц и нулей. Все это хорошо для вычислительных машин, но попробуйте представить весь окружающий вас мир только в черном и белом цвете, вдобавок исключив из языка любые ответы на вопросы, кроме ДА и НЕТ. В такой ситуации вам можно только посочувствовать. Решить эту проблему и призвана нечеткая логика. С термином «лингвистическая переменная» можно связать любую физическую величину, для которой нужно иметь больше значений, чем только ДА и НЕТ. В этом случае вы определяете необходимое число термов и каждому из них ставите в соответствие некоторое значение описываемой физической величины. Для этого значения степень принадлежности физической величины к терму будет равна единице, а для всех остальных значений - в зависимости от выбранной функции принадлежности.

Например, можно ввести переменную ВОЗРАСТ и определить для нее термы ЮНОШЕСКИЙ, СРЕДНИЙ и ПРЕКЛОННЫЙ. Обсудив с экспертами значения конкретного возраста для каждого терма, вы с полной уверенностью можете избавиться от жестких ограничений логики Аристотеля. Одним из основных методов представления знаний в экспертных системах являются продукционные правила, позволяющие приблизиться к стилю мышления человека. Любое правило продукций состоит из посылок и заключения. Возможно Основы нечеткой логики, логико-лингвистические модели стр. 6 наличие нескольких посылок в правиле, в этом случае они объединяются посредством логических связок И, ИЛИ. Обычно продукционное правило записывается в виде:

«ЕСЛИ (посылка_1) (связка) (посылка_2) (св) …(св) (посылка_n), ТО (заключение)».

Главным же недостатком продукционных систем остается то, что для их функционирования требуется наличие полной информации о системе. Нечеткие системы управления основаны на правилах продукционного типа, однако в

качестве посылки и заключения в правиле используются лингвистические переменные, что позволяет избежать ограничений, присущих классическим продукционным правилам.

В основу функционирования нечетких систем управления положен механизм нечеткого вывода, который рассматривается подробно в соответствующей главе работы. Результат нечеткого логического вывода является нечетким, а физическое исполнительное устройство не способно воспринять такую команду. Необходимы специальные математические методы, позволяющие переходить от нечетких значений величин к вполне определенным.

Основные из этих математических методов также рассмотрены в работе. Нечеткие системы управления уже достаточно давно используются на практике. За прошедшее с 1965 года время нечеткая логика прошла путь от почти антинаучной теории, практически отвергнутой в Европе и США, до банальной ситуации конца девяностых годов, когда в Японии в широком ассортименте появились «нечеткие» бритвы, пылесосы, фотокамеры. Началом практического применения теории нечетких множеств можно считать 1975г., когда Мамдани и Ассилиан (Mamdani and Assilian) построили первый нечеткий контролер для управления простым паровым двигателем. В 1982 Холмблад и Остергад (Holmblad and Osregaad) разработали первый промышленный нечеткий контроллер, который был внедрен в управление процессом обжига цемента на заводе в Дании. Успех первого промышленного контролера, основанного на нечетких лингвистических правилах “Если - то” привел к всплеску интереса к теории нечетких множеств среди математиков и инженеров. Несколько позже Бартоломеем Коско (Bart Kosko) была доказана теорема о нечеткой аппроксимации (Fuzzy Approximation Theorem), согласно которой любая математическая система может быть аппроксимирована системой, основанной на нечеткой логике. Другими словами, с помощью естественноязыковых высказываний-правил “Если - то”, с последующей их

формализацией средствами теории нечетких множеств, можно сколько угодно точно отразить произвольную взаимосвязь “входы-выход” без использования сложного аппарата дифференциального и интегрального исчислений, традиционно применяемого в управлении и идентификации. Сам термин «fuzzy» так прочно вошел в жизнь, что на многих языках он даже не переводится.

В России в качестве примера можно вспомнить рекламу стиральных машин и микроволновых печей фирмы Samsung, обладающих искусственным интеллектом на основе нечеткой логики. Тем не менее, столь масштабный скачок в развитии нечетких систем управления не случаен. Простота и дешевизна их разработки заставляет проектировщиков все чаще прибегать к этой технологии. Системы, основанные на нечеткой логике, разработаны и успешно внедрены в таких областях, как управление технологическими процессами, управление транспортом, медицинская диагностика, техническая диагностика, финансовый менеджмент, биржевое прогнозирование, распознавание образов.

Спектр приложений очень широкий - от видеокамер и бытовых стиральных машин до средств наведения ракет ПВО и управления боевыми вертолетами. Практический опыт разработки систем нечеткого логического вывода свидетельствует, что сроки и стоимость их проектирования значительно меньше, чем при использовании традиционного математического аппарата, при этом обеспечивается требуемый уровень прозрачности моделей. Многие современные задачи управления просто не могут быть решены классическими методами из-за очень большой сложности математических моделей, их описывающих.

На рисунке 1 показаны области наиболее эффективного применения современных технологий управления. Как видно, классические методы управления хорошо работают при полностью детерминированном объекте управления и детерминированной среде, а для систем с неполной информацией

и высокой сложностью объекта управления оптимальными являются нечеткие методы управления.

Рисунок 1 – Области наиболее эффективного применения современных технологий управления

(В правом верхнем углу рисунка приведена еще одна современная технология управления - с применением искусственных нейронных сетей, но мы не станем столь глубоко вдаваться в достижения ученых.)

Лингвистическая переменная

Лингвистической называется переменная, принимающая значения из множества слов или словосочетаний некоторого естественного или искусственного языка. Множество допустимых значений лингвистической переменной называется терммножеством. Термом (term) называется любой элемент терм–множества. В теории нечетких множеств терм формализуется

нечетким множеством с помощью функции принадлежности. Задание значения переменной словами, без использования чисел, для человека более естественно.

Ежедневно мы принимаем решения на основе лингвистической информации типа: "очень высокая температура"; "длительная поездка"; "быстрый ответ"; "красивый букет"; "гармоничный вкус" и т.п. Психологи установили, что в человеческом мозге почти вся числовая информация вербально перекодируется и хранится в виде лингвистических термов. Понятие лингвистической переменной играет важную роль в нечетком логическом выводе и в принятии решений на основе приближенных рассуждений. Формально, лингвистическая переменная определяется следующим образом:

Опр: Лингвистической переменной называется набор , где β - наименование лингвистической переменной; T - множество ее значений (терм-множество), представляющих собой наименования нечетких переменных, областью определения каждой из которых является множество X. Множество T называется базовым терм - множеством лингвистической переменной; G - синтаксическая процедура, позволяющая оперировать элементами терм-множества T, в частности, генерировать новые термы (значения).

Множество $T \cup G(T)$, где G(T) - множество сгенерированных термов, называется расширенным терм - множеством лингвистической переменной; M - семантическая процедура, позволяющая превратить каждое новое значение лингвистической переменной, образуемое процедурой G, в нечеткую переменную, т.е. сформировать соответствующее нечеткое множество. Значениями лингвистической переменной являются нечеткие множества, символами которых являются слова и предложения в естественном или формальном языке, служащие, как правило, некоторой элементарной характеристикой явления.

Пример: Рассмотрим лингвистическую переменную с именем "температура в комнате". Тогда оставшуюся четверку , можно определить так: 1) универсальное множество ; 2) терм-множество T={"холодно", "комфортно",

"жарко"} с такими функциями принадлежностями: 3) синтаксические правила , порождающее новые термы с использованием квантификаторов "и","или", "не", "очень","более-менее" и других; 4) M будет являться процедурой, ставящей каждому новому терму в соответствие нечёткое множество из X по правилам: если термы A и B имели функции принадлежности μA(u) и μB(u) соответственно, то новые термы будут иметь следующие функции принадлежности, заданные в таблице 1.

Таблица 1 – Функции принадлежности

Квантификатор	**Функция принадлежности** ($u \in U$)
не t	$1 - \mu_t(u)$
очень t	$(\mu_t(u))^2$
более-менее t	$\sqrt{\mu_t(u)}$
A и B	$\max(\mu_A(x), \mu_B(x))$
A или B	$\min(\mu_A(x), \mu_B(x))$

Графики функций принадлежности термов "холодно", "не очень холодно" и других лингвистической переменной "температура в комнате" показаны на рисунке 2.

Понятие нечеткого множества

Значительный шаг в направлении развития теории нечетких множеств сделал профессор Калифорнийского университета Лотфи А. Заде (1965 г. - публикация работы “Fuzzy Sets”). Заде расширил понятие множества, допустил, что характеристическая функция (функция принадлежности элемента множеству) может принимать любые значения в интервале [0, 1]. Такие множества были названы им нечеткими (fuzzy).

Пусть E – универсальное множество, x – элемент E, P – некоторое свойство. Обычное (четкое) множество A универсального множества E, элементы которого удовлетворяют свойству P, определяются как множество упорядоченных пар

$$A = \{\mu_A(x)/x\},$$

где $\mu_A(x)$ - характеристическая функция, принимающая значение 1, если x удовлетворяет свойству P, и 0 – в противном случае.

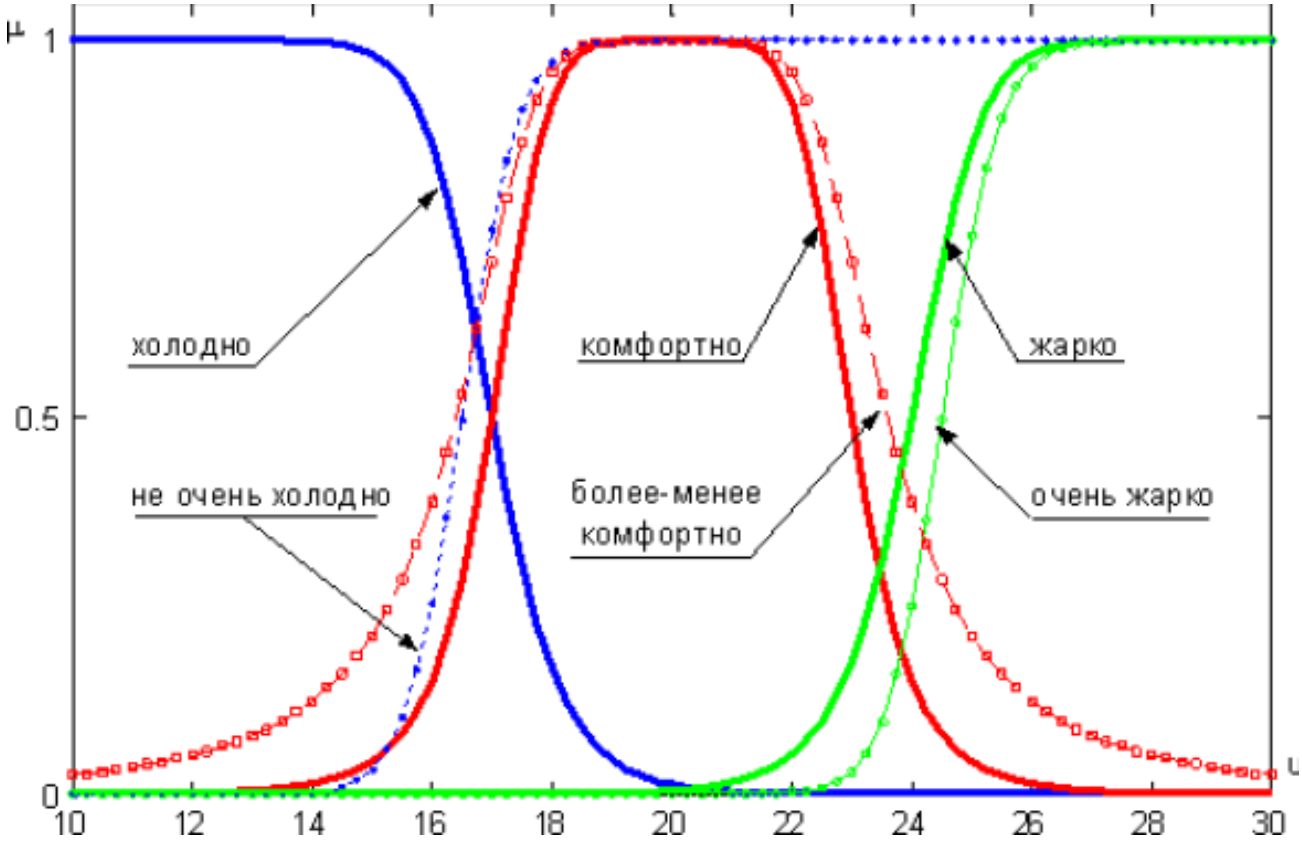


Рисунок 2 – Графики функций принадлежности термов "холодно", "не очень холодно" и других лингвистической переменной "температура в комнате"

В теории нечетких множеств для элементов x из E нет однозначного ответа «да/нет» относительно свойства P. В связи с этим нечеткое множество A универсального множества E определяется как множество упорядоченных пар с функцией принадлежности $\mu_A(x)$, принимающей значение в некотором упорядоченном множестве M (например, M=[0, 1]).

Функция принадлежности указывает степень (или уровень) принадлежности элемента x подмножеству A. Множество M называют множеством принадлежностей. Если M={0, 1},то нечеткое подмножество A может рассматриваться как обычное или четкое множество.

Примеры записи нечеткого множества.

1 Пусть E ={x_1, x_2, x_3, x_4, x_5}; A – нечеткое множество, для которого $\mu_A(x_1)$ =0,3; $\mu_A(x_2)$=0; $\mu_A(x_3)$=1; $\mu_A(x_4)$=0,6; $\mu_A(x_5)$=0,9.

Тогда A можно представить в виде

A = {0,3/x_1; 0/x_2; 1/x_3; 0,6/x_4; 0,9/x_5}.

2 Пусть E = {1, 2, 3, … 100} и соответствует понятию «возраст», тогда нечеткое множество «молодой» может быть определено с помощью выражения

$$\mu_{"молодой"} = \begin{cases} 1, & x \in [1,25], \\ \dfrac{1}{1+\left(\dfrac{x-25}{5}\right)^2}, & x > 25. \end{cases}$$

В приведенных выше примерах использованы прямые методы определения функций принадлежности, когда эксперт задает значение $\mu_A(x)$ для каждого $x \subset E$. Такой способ используется для измеряемых понятий (скорость, температура и т.д.) или когда выделяют полярные значения. Разновидностью прямого способа задания функции принадлежности является групповой метод, когда группе экспертов предлагают сделать оценку того или иного явления, например, оценить: «этот человек лысый» или нет, - тогда количество утвердительных ответов, деленное на общее число экспертов, дает значение $\mu_{лысый}$ для данного лица.

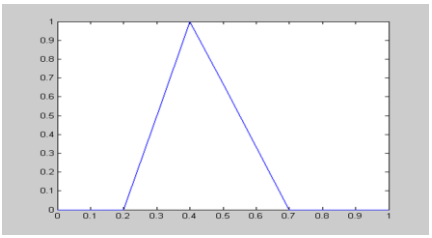
Кроме указанных способов задания функций принадлежности используют также типовые формы функций принадлежности (рисунок 15).

Аналитическая запись некоторых типовых функций принадлежности:

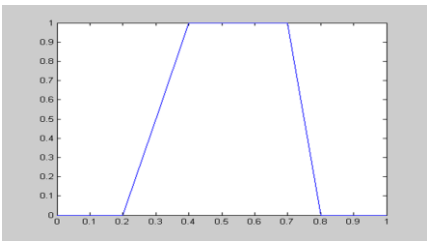
– треугольная - $\mu(x) = \max\left(\min\left(\dfrac{x-a}{b-a}, \dfrac{c-x}{c-b}\right), 0\right)$, a, b, c - определяют параметры функции;

– гауссова - $\mu(x) = e^{-\left(\frac{x-a}{b}\right)^2}$, a, b - параметры функции принадлежности;

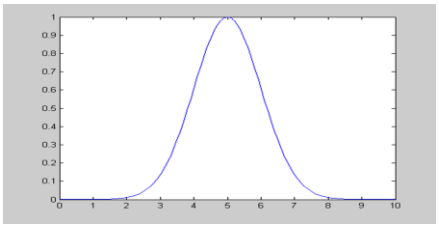
– сигмоидная - $\mu(x)=\frac{1}{1+\exp(-a(x-b))}$, a,b - параметры функции принадлежности.



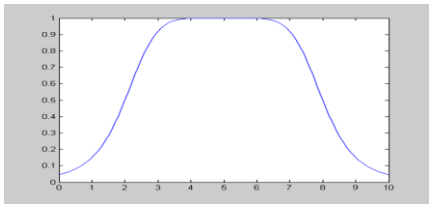
а) треугольная (trimf)



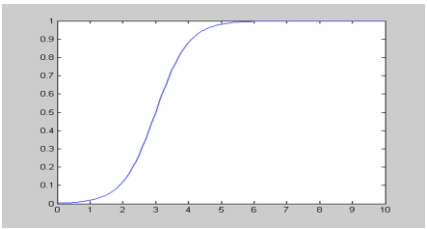
б) трапецеидальные (trapmf)



в) гауссова (gaussmf)



г) обобщенная колоколообразная (gbellmf)



д) сигмоидная (sigmf)

Рисунок 3 – Примеры типовых форм функций принадлежности (ниже в скобке приводится обозначение функции в среде MATLAB)

Основные операции над нечеткими множествами

Включение

Пусть A и B нечеткие множества на универсальном множестве E. A содержится в B, если $\forall x \in E \ \ \mu_A(x) \le \mu_B(x)$. Обозначается $A \subset B$.

Равенство

A и B равны, если $\forall x \in E \ \ \mu_A(x) = \mu_B(x)$. Обозначение: $A=B$.

Дополнение

Пусть M=[0,1], A и B дополняют друг друга, если $\forall x \in E \ \mu_A(x) = 1 - \mu_B(x)$. Обозначается $B = \overline{A}$ или $A = \overline{B}$.

Пересечение

$A \cap B$ - наименьшее нечеткое подмножество, содержащееся одновременно в A и B: $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)) = \mu_A(x) \tilde{\wedge} \mu_B(x)$.

Или, по другому, определение в классе треугольных норм (t – норма).

Типичными t - нормами являются:

- операция min как нечеткое логическое произведение с нечеткими переменными X1, X2: $\mathbf{X_1} \tilde{\wedge} \mathbf{X_2} = \mathbf{min(X_1, X_2)}$;

- алгебраическое произведение $\mathbf{X_1} \tilde{\wedge} \mathbf{X_2} = \mathbf{X_1} \cdot \mathbf{X_2}$ [2, 3].

Объединение

$A \cup B$ - наибольшее нечеткое подмножество, включающее как *A*, так и *B*, с функцией принадлежности:

$\mu_{\text{À} \cup \hat{\text{A}}}(\tilde{\text{o}}) = \max(\mu_{\text{A}}(\text{x}), \mu_{\text{B}}(\text{x})) = \mu_{\text{A}}(\text{x}) \tilde{\vee} \mu_{\text{B}}(\text{x})$.

Или, по-другому, определение в классе треугольных конорм (s – норма).

Типичными s - нормами являются:

- операция нечеткого логического сложения $\mathbf{X_1} \tilde{\vee} \mathbf{X_2} = \mathbf{max(X_1, X_2)}$;

- алгебраическая сумма $\mathbf{X_1} \tilde{\mathbf{V}} \mathbf{X_2} = \mathbf{X_1} + \mathbf{X_2} - \mathbf{X_1 X_2}$,

где «~» означает нечеткую логическую операцию.

Разность

A-B= $A \cap \overline{B}$ с функцией принадлежности: $\mu_{\text{À} - \hat{\text{A}}}(\tilde{\text{o}}) = \min(\mu_{\text{A}}(\text{x}), 1 - \mu_{\text{B}}(\text{x}))$.

Нечеткие отношения

Нечеткое n-мерное отношение определяется как нечеткое подмножество R на E, принимающее свои значения в M. В случае n=2 и M=[0,1] нечетким отношением R между множествами X=E_1 и Y=E_2 будет называться функция R: (X, Y) → [0,1], которая ставит в соответствие каждой паре элементов (x,y) ∈ *X* x *Y* величину $\mu_R(x, y) \in [0,1]$.

Обозначение: $x \in X,\ y \in Y : xRy$.

Алгебраические операции над нечеткими отношениями аналогичны операциям с нечетким множествам.

Композиция (свертка) двух нечетких отношений

Пусть R_1 – нечеткое отношение R_1: $(X * Y) \rightarrow [0,1]$ между X и Y, и R_2 – нечеткое отношение R_2: $(Y * Z) \rightarrow [0,1]$ между Y и Z. Нечеткое отношение между X и Z обозначается $R_1 \bullet R_2$ и определяется как

$$\mu_{R_1 \bullet R_2}(x, y) = \bigvee_y \left[\mu_{R_1}(x, y) \wedge \mu_{R_2}(y, z)\right] = \max\left(\min\left(\mu_{R_1}(x, y), \mu_{R_2}(y, z)\right)\right),$$

где символ $\bigvee_y$ - обозначает операцию выбора наибольшего по y значения и называется (max-min)-сверткой отношений R_1 и R_2.

Пример. Пусть заданы отношения R_1 и R_2.

Таблица 2 – Отношение R_1

R_1	y_1	y_2	y_3
x_1	0,1	0,7	0,4
x_2	1	0,5	0

Таблица 3 – Отношение R_2

R_2	z_1	z_2	z_3	z_4
y_1	0,9	0	1	0,2
y_2	0,3	0,6	0	0,9
y_3	0,1	1	0	0,5

$$\mu_{R_1 \bullet R_2}(x_1, y_1) = \left[\mu_{R_1}(x_1, y_1) \wedge \mu_{R_2}(y_1, z_1)\right] \vee \left[\mu_{R_1}(x_1, y_2) \wedge \mu_{R_2}(y_2, z_1)\right] \vee$$
$$\vee \left[\mu_{R_1}(x_1, y_3) \wedge \mu_{R_2}(y_3, z_1)\right] =$$
$$= (0,1 \wedge 0,9) \vee (0,7 \wedge 0,3) \vee (0,4 \wedge 0,1) = 0,1 \vee 0,3 \vee 0,1 = 0,3.$$

Таблица 4 – Отношения R_1 и R_2

$R_1 \cdot R_2$	z_1	z_2	z_3	z_4
x_1	0,3	0,6	0,1	
x_2	0,9	0,5	1	

Нечеткие выводы

В экспертных и управляющих системах механизм нечетких выводов в своей основе имеет базу знаний, формируемую специалистами предметной области в виде совокупности нечетких предикатных правил вида:

$П_1$: если x есть A_1, то y есть B_1,

$П_2$: если x есть A_2, то y есть B_2,

…

$П_n$: если x есть A_n, то y есть B_n,

где x – входная переменная, y – переменная вывода, A и B – функции принадлежности, определенные на x и y соответственно.

Знания эксперта $A \rightarrow B$ отражает нечеткое причинное отношение предпосылки и заключения, поэтому его называют нечетким отношением:

$R = A \rightarrow B$,

где «$\rightarrow$» - нечеткая импликация.

Отношение R можно рассматривать как нечеткое подмножество прямого произведения $X \times Y$ полного множества предпосылок X и заключений Y. Таким образом, процесс получения (нечеткого) результата вывода B' с использованием данного наблюдения A' и значения $A \rightarrow B$ можно представить в виде

$B' = A' \bullet R = A' \bullet (A \rightarrow B)$.

Алгоритм нечеткого вывода

1 ***Нечеткость*** (фаззификация, fuzzification). Функции принадлежности, определенные для входных переменных, применяются к их фактическим значениям для определения степени истинности каждой предпосылки каждого правила).

2 ***Логический вывод.*** Вычисленное значение истинности для предпосылок каждого правила применяется к заключениям каждого правила. Это приводит к одному нечеткому подмножеству, которое будет назначено

переменной вывода для каждого правила. В качестве правил логического вывода используются только операции min (минимума) или prod (умножение).

3 ***Композиция***. Нечеткие подмножества, назначенные для каждой переменной вывода (во всех правилах), объединяются вместе, чтобы сформировать одно нечеткое подмножество для каждой переменной вывода. При подобном объединении обычно используются операции max (максимум) или sum (сумма).

4 ***Дефаззификация*** – приведение к четкости (defuzzification). Преобразование нечеткого набора выводов в число.

Алгоритмы нечеткого вывода Мамдани (Mamdani)

Пусть заданы два нечетких правила:

$П_1$: если x есть A_1 и y есть B_1, тогда z есть C_1,

$П_2$: если x есть A_2 и y есть B_2, тогда z есть C_2.

1 ***Нечеткость***. Находят степени принадлежности для предпосылок каждого правила: $A_1(x_0)$, $A_2(x_0)$, $B_1(y_0)$, $B_2(y_0)$.

2 ***Нечеткий вывод***. Определяют уровни «отсечения» для предпосылок каждого правила (операция min):

$\alpha_1 = A_1(x_0) \wedge B_1(y_0)$,

$\alpha_2 = A_2(x_0) \wedge B_2(y_0)$,

определяют усеченные функции принадлежности

$C'_1 = (\alpha_1 \wedge C_1(z))$,

$C'_2 = (\alpha_2 \wedge C_2(z))$.

3 ***Композиция***. Производится объединение найденных усеченных функций (операция max), получают нечеткое подмножество для переменной выхода с функцией принадлежности:

$$\mu_{\Sigma}(z) = C(z) = C'_1(z) \vee C'_2(z) = (\alpha_1 \wedge C_1(z)) \vee (\alpha_2 \wedge C_2(z)).$$

4 ***Дефаззификация***. Приведение к четкости (определение z_0), например, *центроидным* методом (как центр тяжести для кривой $\mu_{\Sigma}(z)$):

$$z_0 = \frac{\int_{\Omega} z\mu_{\Sigma}(z)dz}{\int_{\Omega} \mu_{\Sigma}(z)dz}.$$

Алгоритм иллюстрируется на рисунке 4.

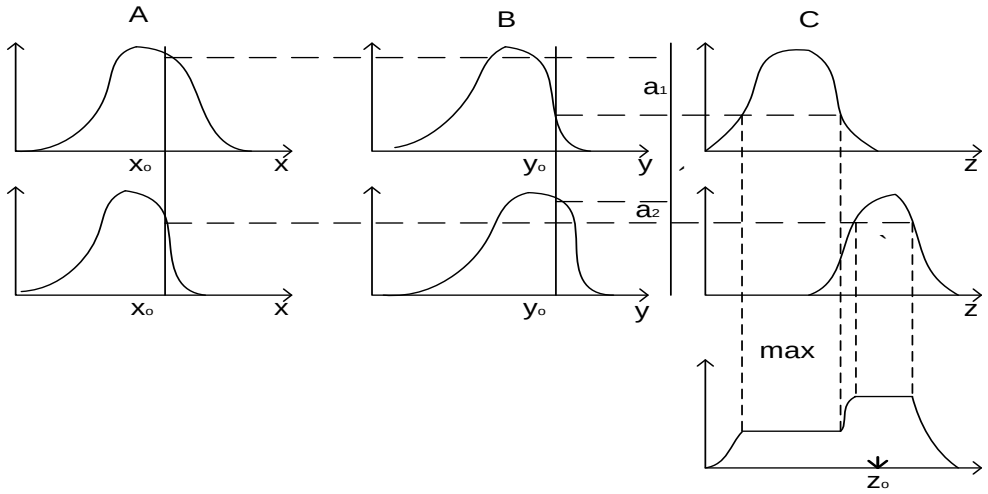


Рисунок 4 – Графическая интерпретация алгоритма Мамдани

Алгоритмы нечеткого вывода Сугено (Sugeno) и Такаги (Takagi)

Сугено (Sugeno) и Такаги (Takagi) использовали набор правил в следующей форме:

$П_1$: если x есть A_1 и y есть B_1, тогда $z=a_1x+b_1y$,

$П_2$: если x есть A_2 и y есть B_2, тогда $z=a_2x+b_2y$.

1 Первый этап – аналогично алгоритму Мамдани.

2 Определение $\alpha_1 = A_1(x_0) \wedge B_1(y_0)$, $\alpha_2 = A_2(x_0) \wedge B_2(y_0)$ и индивидуальные выходы правил:

$z^*_1 = a_1x_0 + b_1y_0$,

$z^*_2 = a_2x_0 + b_2y_0$.

3 Определяется значение переменной вывода:

$$z_0 = \frac{\alpha_1 z_1^* + \alpha_2 z_2^*}{\alpha_1 + \alpha_2}.$$

Представленная форма правил иллюстрирует алгоритм Сугено 1-го порядка. Если правила записаны в форме:

$П_1$: если x есть A_1 и y есть B_1, тогда $z=c_1$,

$П_2$: если x есть A_2 и y есть B_2, тогда $z=c_2$,

то задан алгоритм Сугено 0-го порядка.

Иллюстрация алгоритма Сугено 0-го порядка представлена на рисунке 5.

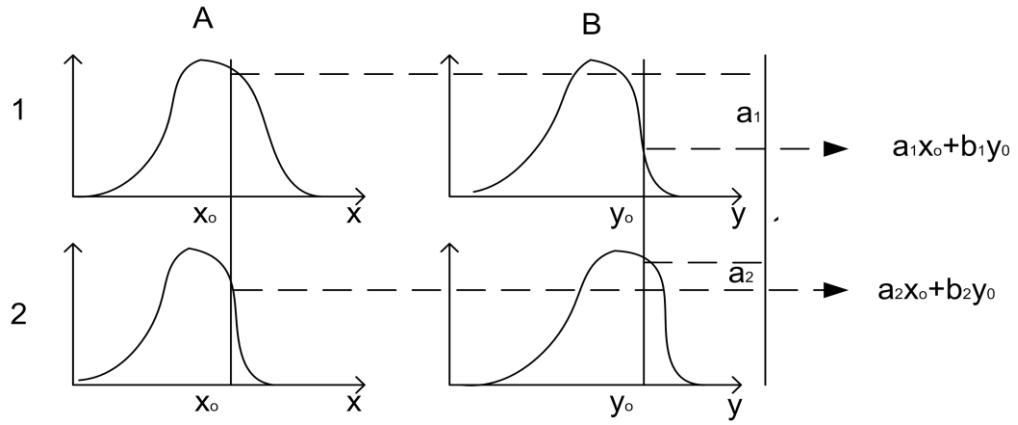


Рисунок 5 – Графическая интерпретация алгоритма Сугено

Перечислим наиболее известные методы дефаззификации.

Метод Максимума – выбирается тот элемент нечеткого множества, который имеет наивысшую степень принадлежности этому множеству.

Если этот элемент не является единственным, т.е. функция принадлежности $\mu_A(y_1)$ имеет несколько локальных максимумов, или если имеется максимальное «плато», то выбор среди элементов, имеющих наивысшую степень принадлежности множеству, осуществляется на основе некоторого критерия.

Метод левого (правого) максимума – выбирается наименьшее (наибольшее) из чисел y_1, y_2, .. ,y_n, имеющих наивысшую степень принадлежности нечеткому множеству.

Метод среднего из максимумов – в качестве искомого четкого значения y_o принимается среднее арифметическое координат локальных максимумов $y_o = \frac{1}{n}\sum_{1}^{n} y_i$.

Возможные методы дефаззификации и их обозначения в MATLAB приведены на рисунке 6: наименьший максимум (smallest of max, som), наибольший максимум (largest of max, lom), средний максимум (mean of max, mom), бисекторный (bisector of area), рассмотренный выше центроидный (centroid).

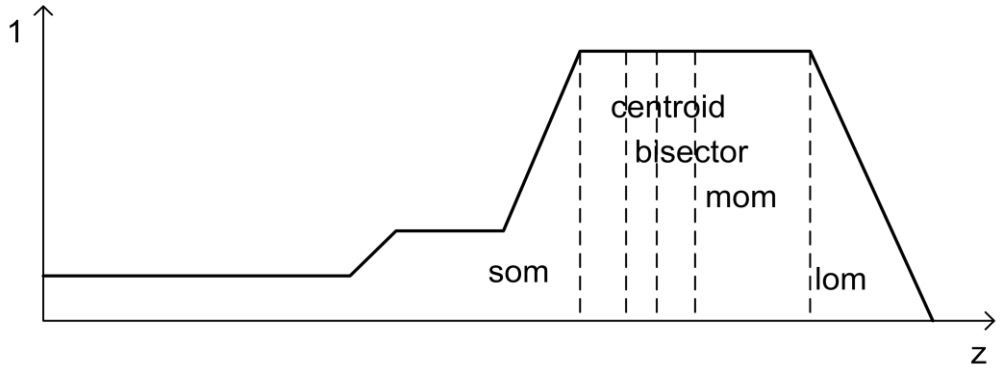


Рисунок 6 – Иллюстрация примеров фаззификации z

Типовое практическое задание к лабораторной работе № 2

Проектирование системы нечеткого вывода в среде MATLAB

Построение нечеткой аппроксимирующей функции $y=x^2$

Исходные данные для аппроксимации представлены в таблице 5.

Таблица 5 – Значения x и y

X	-1	-0,6	0	0,4	1
Y	1	0,36	0	0,16	1

1 Командой **fuzzy** из режима командной строки запускается основная интерфейсная программа пакета Fuzzy Logic – редактор нечеткой системы вывода.

2 В меню **File** выбрать команду New Sugeno FIS.

3 Выбрать входной элемент системы *input1* и ввести в поле *Name* обозначение входной переменной (*x)*.

4 Войти в режим редактирования функции принадлежности – *Membership Function Editor* (двойное нажатие левой клавиши мыши). Выбором **Edit/Add MFs** (добавить функцию принадлежности) задать тип функции принадлежности (gaussmf) и количество (5).

5 Установить диапазон (*Range*) изменения *x* от -1 до 1 (определяется заданным диапазоном *x*).

6 Совместить ординаты максимумов функций принадлежности с значениями аргумента *x*.

Шаг выполняется двумя способами: выделить редактируемую функцию принадлежности, перетащить мышью кривую функции принадлежности; более точную установку проводят заданием числовых значений параметров функции принадлежности в поле *Params* (первое значение определяет размах кривой, второе – положение центра).

В поле *Name* вводится имя функции принадлежности (1- bn, 2 – n, 3 (центральная) – z, 4 – p, 5 – bp).

Выйти из редактора функций принадлежности – *Close*.

7 Ввести в поле название выходной переменной *output1* (*y)*, войти в режим редактирования функций принадлежности.

8 Задать вид функции принадлежности для выходной переменной. Предлагается выбрать в качестве функции принадлежности линейные (*linear*) или постоянные (*constant*) – в зависимости от алгоритма Сугено (1-го или 0-го порядка). В поставленной задаче необходимо выбрать постоянные функции принадлежности с общим числом 4 (по числу различающихся значений *y*). Нажать Ok.

9 Установить диапазон (*Range*) – [0, 1]. Изменить значения (*Params*) и задать для выходных переменных имена (*Name*) соответственно 0; 0,16; 0,36; 1. Закрыть редактор.

10 Перейти в редактор привил (*Rule Editor*) (дважды щелкнуть на средний белый квадрат разрабатываемой структуры системы нечеткого вывода). При

вводе каждого правила необходимо обозначить соответствие между каждой функцией принадлежности аргумента *x* и числовым значением *y*. Кривая, обозначенная bn, соответствует *y=1*, для чего выбирается в левом поле (с заголовком *x is*) вариант bn, а в правом – 1 и нажимается кнопка *Add rule*.

Введенное правило появится в окне правил в виде

If (x is bn) then (y is 1) (1).

Аналогично вводятся все правила (всего 5).

Закрыть окно редактора правил и вернуться в окно FIS-редактора.

Построение системы закончено.

Сохранить на диске (**File/Save to disk as**) созданную систему. Перейти в редактор функций принадлежности (**View/Edit Membership function**). Из окна редактора командой можно перейти в окно просмотра правил (**View rules**), просмотра поверхности (**View surface**).

В окне просмотра правил иллюстрируется процесс принятия решения (вычисления *y*). Красная вертикальная черта, пересекающая графики в правой части окна, которую можно перемещать с помощью мыши, позволяет изменять значения переменной входа (либо вводят значение с клавиатуры в поле *Input*), при этом соответственно изменяется значение выхода.

Просмотр кривой *y(x)* осуществляется командой **View/View surface**. Большая погрешность аппроксимации заданной зависимости объясняется малым числом экспериментов.

С помощью рассмотренных редакторов на любом этапе проектирования нечеткой модели можно внести необходимые коррективы, например задание пользовательской функции принадлежности.

В рассмотренном примере использованы следующие операции, устанавливаемые по умолчанию в алгоритме Сугено:

- логический вывод организуется с помощью операции логического умножения (prod);

- композиция организуется с помощью операции логической суммы (вероятностного ИЛИ, probor);

- приведение к четкости (дефаззификация) организуется дискретным вариантом центроидного метода (взвешенным средним, wtaver).

Исследование автоматической системы управления с fuzzy-регулятором

Открыть файл *model2.mdl* каталога *Demo* из запущенного приложения Matlab. На экране должна быть изображена следующая структура АСР (рис. 7).

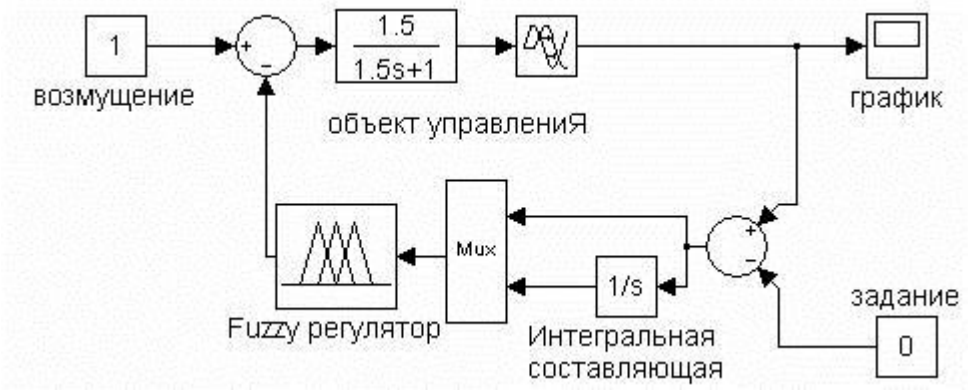


Рисунок 7 – Модель одноконтурной системы автоматического регулирования с ПИ-подобным fuzzy-регулятором

Теперь при помощи инструментов графического интерфейса пользователя (GUI) пакета "Fuzzy Logic Toolbox" создадим нечёткую систему, реализующую типовой аналоговый ПИ-регулятор. Заметим, что с помощью пакета "Fuzzy Logic Toolbox" можно строить нечеткие системы двух типов - Мамдани и Сугэно. Остановимся на системе типа Мамдани. Командой fuzzy в окне MATLAB вызываем окно Редактора фази-инференционной системы (Fuzzy Inference System Editor), выбираем тип системы - Мамдани, задаём два входа - для пропорциональной и интегральной составляющих и называем входные переменные, например, x1 и x2, а выходную - y.

Из данного окна вызываем окно Редактора функций принадлежности (Membership Function Editor) двойным щелчком мыши по изображению переменной x1 или при помощи меню Edit. Здесь для лингвистического описания каждой переменной выберем семь треугольных термов (NB,NM,NS,ZE,PS,PM,PB). Термы выходной переменной лучше выбирать непересекающимися. Это повысит чёткость регулирования. В этом же окне зададим диапазоны изменения переменных:

Для входных переменных регулятора рекомендуются симметричные диапазоны изменения, при этом

$$x_1 \in \left[-\frac{1}{k_1};\frac{1}{k_1}\right] \text{ и } x_2 \in \left[-\frac{1}{k_0};\frac{1}{k_0}\right],$$

где $К_1$, $К_0$ – оптимальные настройки пропорциональной и интегрирующей частей ПИ – регулятора в смысле какого-либо критерия.

Для выходной переменной регулятора диапазон изменения рекомендуется брать в виде $y \in [0;C]$, где верхняя граница С при единичном ступенчатом воздействии варьируется от 1.1 до 2, чтобы выходной сигнал регулятора мог компенсировать это возмущение. По мере увеличения значения С уменьшается динамическая ошибка, но возрастают время регулирования и число колебаний переходного процесса. Поэтому рекомендуется С принимать равным 2, когда наблюдается оптимальное соотношение между величиной динамической ошибки, времени регулирования и количеством колебаний.

Теперь необходимо сформировать базу правил fuzzy-регулятора. В основу положен способ, предложенный в литературе. Линейный непрерывный ПИ-регулятор с дифференциальным уравнением

$$y(t) = k_D \cdot \varepsilon(t) + \frac{1}{T_E}\int_0^t \varepsilon(t)\cdot dt$$

можно заменить близким по стратегии и логике управления fuzzy-регулятором, если в качестве его выходной переменной рассматривать приращение

управляющего воздействия Δy. Тогда ПИ закон регулирования можно представить в следующей дифференциальной форме:

$$\frac{dy(t)}{dt} = k_D \cdot \frac{d\varepsilon(t)}{dt} + \frac{1}{T_I} \cdot \varepsilon(t),$$

или в разностной форме:

$$\Delta y(k) = y(k) - y(k-1) = k_D \cdot \Delta\varepsilon(k) + \frac{\Delta t}{T_I} \cdot \varepsilon(k).$$

Таким образом, для входных переменных ε(k) и Δε(k) и выходной Δy(k) может быть синтезирован fuzzy-регулятор, реализующий нелинейный закон

$$\Delta y(k) = F\left[\Delta\varepsilon(k),\ \varepsilon(k)\right]$$

и эквивалентный в определённом смысле ПИ-регулятору.

Для нашего случая x_1 соответствует сигналу рассогласования ε(k), x_2 соответствует приращению сигнала рассогласования Δε(k), а y соответствует Δy(k). Лингвистические правила для такого ПИ-подобного fuzzy-регулятора приведены в таблице 6.

Таблица 6 – Лингвистические правила для ПИ fuzzy-регулятора

Δε \ ε	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NB	NB	NM	NS	ZE
NM	NB	NB	NB	NM	NS	ZE	PS
NS	NB	NB	NM	NS	ZE	PS	PM
ZE	NB	NM	NS	ZE	PS	PM	PB
PS	NM	NS	ZE	PS	PM	PB	PB
PM	NS	ZE	PS	PM	PB	PB	PB
PB	ZE	PS	PM	PB	PB	PB	PB

Вызываем окно Редактора правил (Rule Editor) в меню Edit или нажатием Ctrl+3 и заполняем список правилами из таблицы 5. Правила формируются по типу: ЕСЛИ … И …, ТО…. Можно посмотреть пространство управления, вызвав окно Просмотра пространства управления(Surface Viewer) из меню View или комбинацией клавиш Ctrl+6. Полученный файл сохраним под именем **fuzzy1.fis**.

В окне параметров блока Fuzzy Logic Controller укажем имя файла **fuzzy1.** В окне модели в меню **File** выберем пункт **Model Properties**. В открывшемся окне выберем вкладку **CallBacks** и в поле **Model pre-load function** напишем:

fuzzy1=readfis('fuzzy1').

Данная команда будет каждый раз при открытии файла модели помещать файл **fuzzy1.fis** в WorkSpace (рабочее пространство системы MATLAB). Это необходимо для нормального функционирования модели. Стоит заметить, что при внесении изменений в fis-файл нужно помещать его исправленную версию в WorkSpace либо при помощи пункта Export/To Workspace меню File, либо комбинацией клавиш Ctrl+T, либо каждый раз закрытием и открытием файла модели.

В диалоговом окне **Simulation Parameters** меню **Simulation** во вкладке **Advanced** для опции **Boolean logic signals** необходимо установить значение **off.** При этом блоки логики будут допускать переменные в форме с плавающей точкой.

Запуск модели – **Simulation/Start** (Ctrl+T).

В папке Demo находится файл с правилами нечеткого вывода. Загрузка файла осуществляется из окна редактора правил Rule Viewer: во вкладке **File/Import From Disk** открыть файл fuzzy1.fsi. Затем нужно поместить правила в WorkSpace системы (описано выше).

Работу разработанного fuzzy-регулятора можно сравнить с аналоговой системой регулирования (рис. 8).

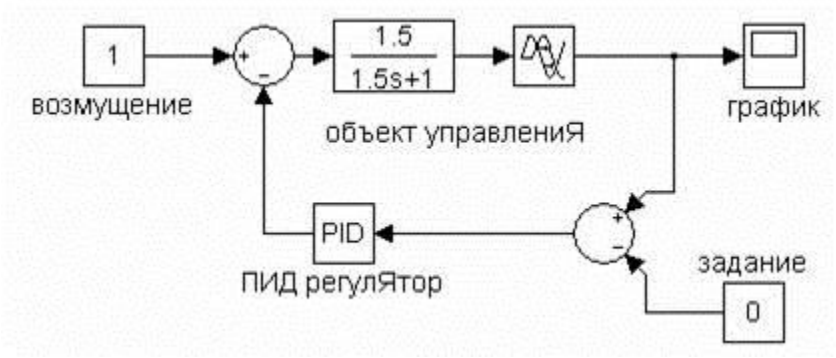


Рисунок 8 – Модель одноконтурной системы автоматического регулирования

Дифференциальное уравнение ПИД-регулятора имеет вид:

$$y(t) = k_D \cdot \varepsilon(t) + \frac{1}{T_E}\int_0^t \varepsilon(t) \cdot dt + T_Д \cdot \frac{d\varepsilon(t)}{dt}.$$

Для исследования ПИ-регулятора в блоке настроек регулятора необходимо в поле параметра $T_Д$ (настройка дифференцирующей части ПИД-регулятора) установить значение 0.

НЕЙРОННЫЕ СЕТИ. КЛАССИФИКАЦИЯ НЕЙРОННЫХ СЕТЕЙ

Нейронная сеть — это последовательность нейронов, соединенных между собой синапсами.

Структура нейронной сети пришла в мир программирования из биологии.

Благодаря такой структуре машина обретает способность анализировать и даже запоминать различную информацию.

Нейронные сети также способны не только анализировать входящую информацию, но и воспроизводить ее из своей памяти.

Нейросеть это машинная интерпретация мозга человека, в котором находятся миллионы нейронов передающих информацию в виде электрических импульсов.

Нейронные сети используются для решения сложных задач, которые требуют аналитических вычислений подобных тем, что делает человеческий мозг.

Самыми распространенными применениями нейронных сетей является:

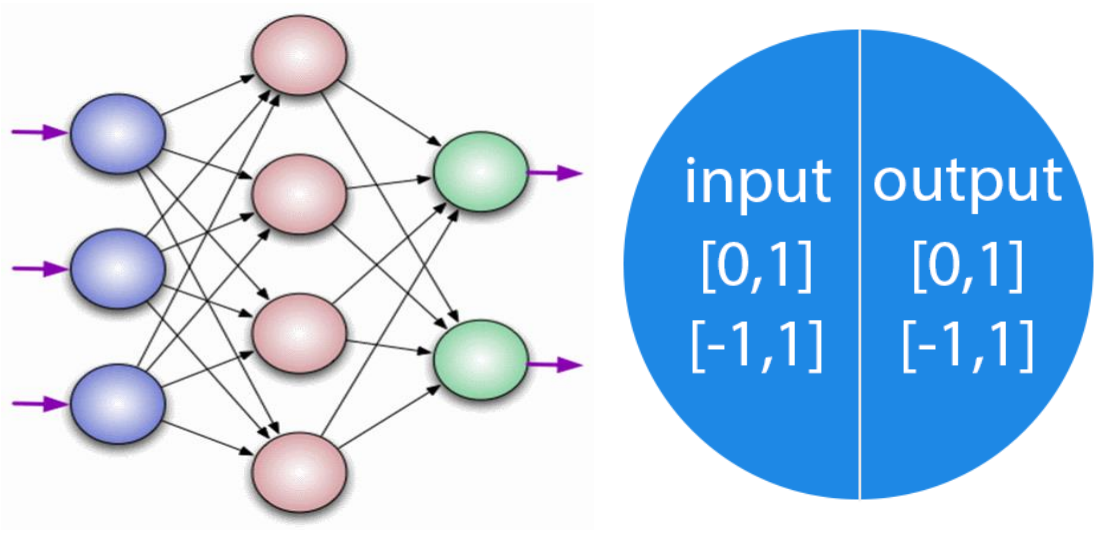
Классификация — распределение данных по параметрам. Например, на вход дается набор людей и нужно решить, кому из них давать кредит, а кому нет. Эту работу может сделать нейронная сеть, анализируя такую информацию как: возраст, платежеспособность, кредитная история и тд.

Предсказание — возможность предсказывать следующий шаг. Например, рост или падение акций, основываясь на ситуации на фондовом рынке.

Распознавание — в настоящее время, самое широкое применение нейронных сетей. Используется в Google, когда вы ищете фото или в камерах телефонов, когда оно определяет положение вашего лица и выделяет его и многое другое.

Нейрон — это вычислительная единица, которая получает информацию, производит над ней простые вычисления и передает ее дальше.

Они делятся на три основных типа: входной (синий), скрытый (красный) и выходной (зеленый) (рис. 9а).



а) б)

Рисунок 9 – Типы и параметры нейронов

У каждого из нейронов есть 2 основных параметра: входные данные (input data) и выходные данные (output data).

В случае входного нейрона: input=output. В остальных, в поле input попадает суммарная информация всех нейронов с предыдущего слоя, после чего, она нормализуется, с помощью функции $f(x)$ и попадает в поле output.

Нейроны оперируют числами в диапазоне [0,1] или [-1,1].

Для обработки чисел, которые выходят из данного диапазона необходимо разделить 1 на это число.

Этот процесс называется нормализацией, и он очень часто используется в нейронных сетях.

Синапс это связь между двумя нейронами.

У синапсов есть 1 параметр — вес.

Благодаря ему, входная информация изменяется, когда передается от одного нейрона к другому.

Допустим, есть 3 нейрона, которые передают информацию следующему. Тогда у нас есть 3 веса, соответствующие каждому из этих нейронов. У того

нейрона, у которого вес будет больше, та информация и будет доминирующей в следующем нейроне (пример — смешение цветов).

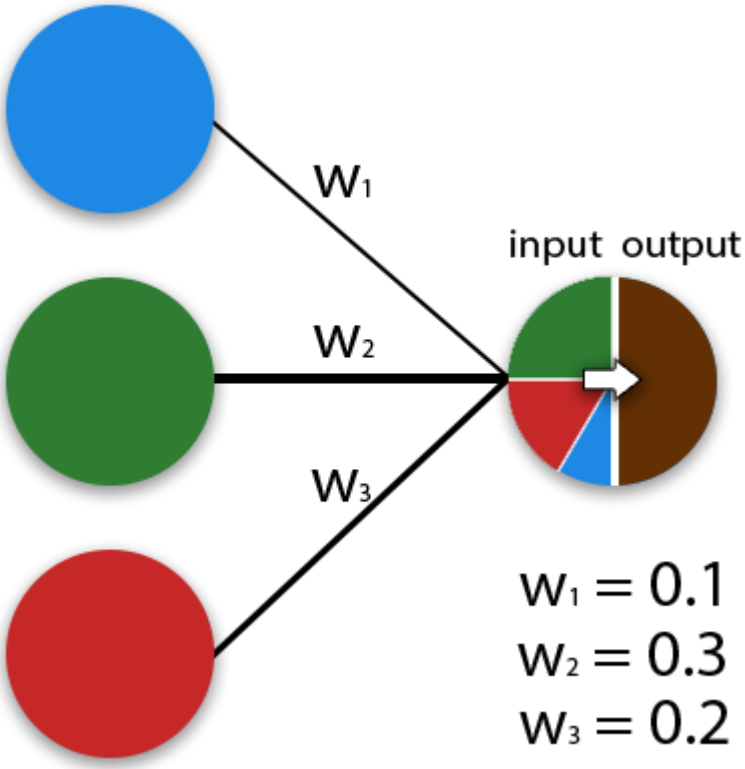


Рисунок 10 – Пример формирование нейронной сети

Функция активации — это способ нормализации входных данных (мы уже говорили об этом ранее).

То есть, если на входе у вас будет большое число, пропустив его через функцию активации, вы получите выход в нужном вам диапазоне.

Функций активации достаточно много поэтому мы рассмотрим самые основные: Линейная, Сигмоид (Логистическая) и Гиперболический тангенс.

Главные их отличия — это диапазон значений.

Сигмоид – самая распространенная функция активации, ее диапазон значений [0,1].

Именно на ней показано большинство примеров в сети, также ее иногда называют логистической функцией. Соответственно, если в вашем случае присутствуют отрицательные значения (например, акции могут идти не только

вверх, но и вниз), то вам понадобиться функция которая захватывает и отрицательные значения.

Гиперболический тангенс. Имеет смысл использовать гиперболический тангенс, только тогда, когда ваши значения могут быть и отрицательными, и положительными, так как диапазон функции [-1,1]. Использовать эту функцию только с положительными значениями нецелесообразно так как это значительно ухудшит результаты вашей нейросети.

Тренировочный сет — это последовательность данных, которыми оперирует нейронная сеть. В нашем случае исключающего или (xor) у нас всего 4 разных исхода то есть у нас будет 4 тренировочных сета: 0xor0=0, 0xor1=1, 1xor0=1,1xor1=0.

Итерация – это своеобразный счетчик, который увеличивается каждый раз, когда нейронная сеть проходит один тренировочный сет. Другими словами, это общее количество тренировочных сетов пройденных нейронной сетью.

При инициализации нейронной сети эта величина устанавливается в 0 и имеет потолок, задаваемый вручную. Чем больше эпоха, тем лучше натренирована сеть и соответственно, ее результат. Эпоха увеличивается каждый раз, когда мы проходим весь набор тренировочных сетов, в нашем случае, 4 сетов или 4 итераций.

Ошибка — это процентная величина, отражающая расхождение между ожидаемым и полученным ответами. Ошибка формируется каждую эпоху и должна идти на спад. Если этого не происходит, значит, вы что-то делаете не так. Ошибку можно вычислить разными путями, но мы рассмотрим лишь три основных способа: Mean Squared Error (далее MSE), Root MSE и Arctan.

У Arctan, ошибка, почти всегда, будет больше, так как он работает по принципу: чем больше разница, тем больше ошибка. У Root MSE будет наименьшая ошибка, поэтому, чаще всего, используют MSE, которая сохраняет баланс в вычислении ошибки.

ЛАБОРАТОРНАЯ РАБОТА 1. МЕТОДЫ НЕЧЕТКОЙ ЛОГИКИ

Задача: сколько раз в месяц нужно поливать цветы в зависимости от температуры (°С) в помещении и ширины листьев растения (см).

1. Для загрузки основного fis-редактора напечатаем слова **fuzzy** в командной строке. После этого откроется новое графическое окно (рис. 11).

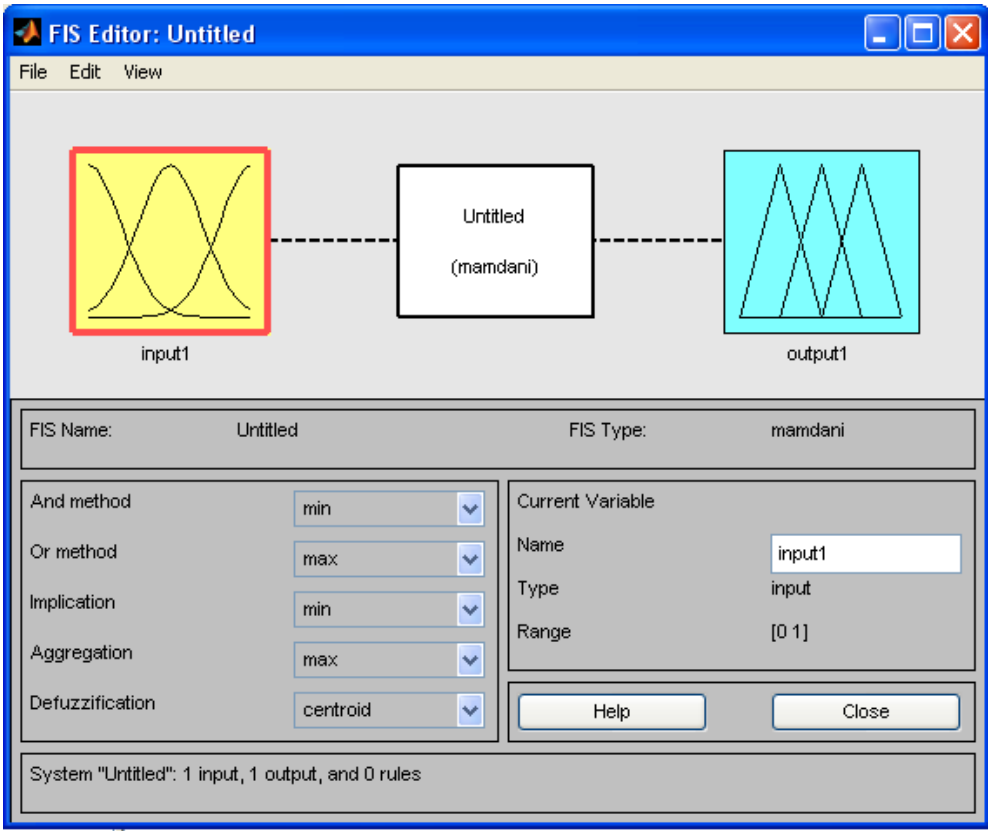


Рисунок 11 – Окно основного fis-редактора

2. Добавим вторую входную переменную. Для этого в меню **Edit** выбираем команду **Add variable → input**.

3. Переименуем первую входную переменную. Для этого сделаем один щелчок левой кнопкой мыши на блоке **input1**, введем новое обозначение **temperature (температура)** в поле редактирования имени текущей переменной.

4. Переименуем вторую входную переменную. Для этого сделаем один щелчок левой кнопкой мыши на блоке **input2**, введем новое обозначение **leaves (листья)** в поле редактирования имени текущей переменной.

5. Переименуем выходную переменную. Для этого сделаем один щелчок левой кнопкой мыши на блоке **output1**, введем новое обозначение **watering (полив)** в поле редактирования имени текущей переменной.

6. Зададим имя системы. Для этого в меню **File** выбираем в подменю **Export** команду **To workspace** и вводим имя файла **flowers** (рис. 12).

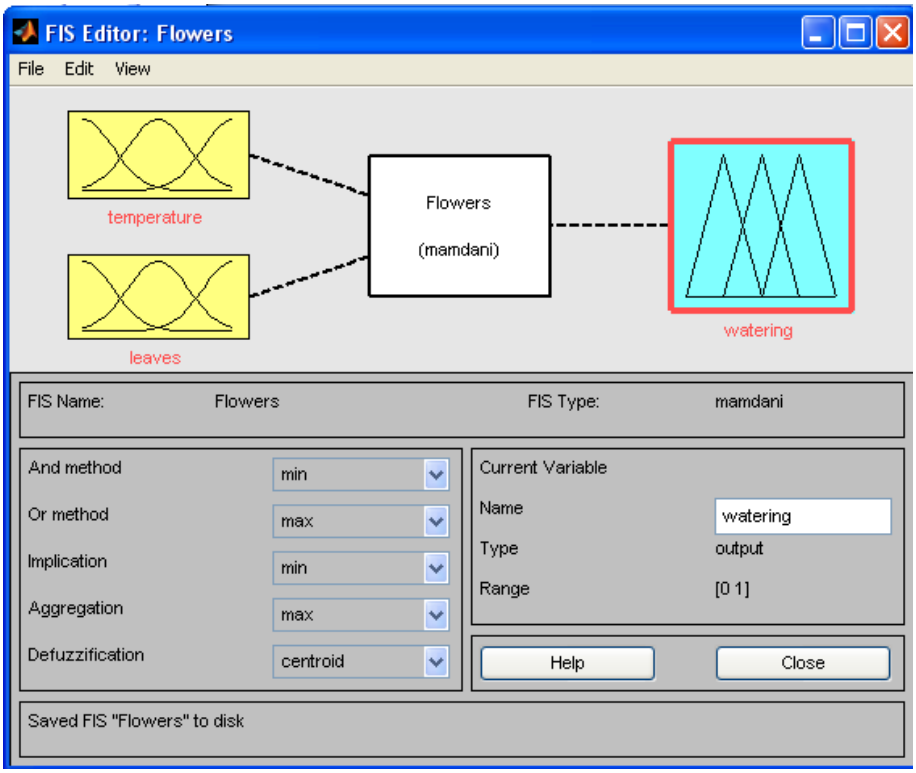


Рисунок 12 – Окно присвоения имени системы

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

8. Зададим диапазон изменения переменной **temperature**. Для этого напечатаем **10 30** в поле **Range.**

9. Зададим функции принадлежности переменной **temperature**. Для лингвистической оценки этой переменной будем использовать 3 терма с треугольными функциями принадлежности. Для этого в меню **Edit** выберем команду **Add MFs...** В результате появиться диалоговое окно выбора типа и количества функций принадлежностей. По умолчанию это 3 терма с треугольными функциями принадлежности.

10. Зададим наименования термов переменной **temperature**. Для этого делаем один щелчок левой кнопкой мыши по графику первой функции принадлежности. Затем вводим наименование терма **low (низкая)** в поле **Name**. Затем делаем один щелчок левой кнопкой мыши по графику второй функции принадлежности и вводим наименование терма, например **Normal (нормальная)** в поле **Name.** Еще раз делаем один щелчок левой кнопкой мыши по графику третьей функции принадлежности и вводим наименование терма **High (высокая)** в поле **Name.** В поле **Params** мы можем задать численные параметры для графиков. В результате получим результат, приведенный на рисунке 13.

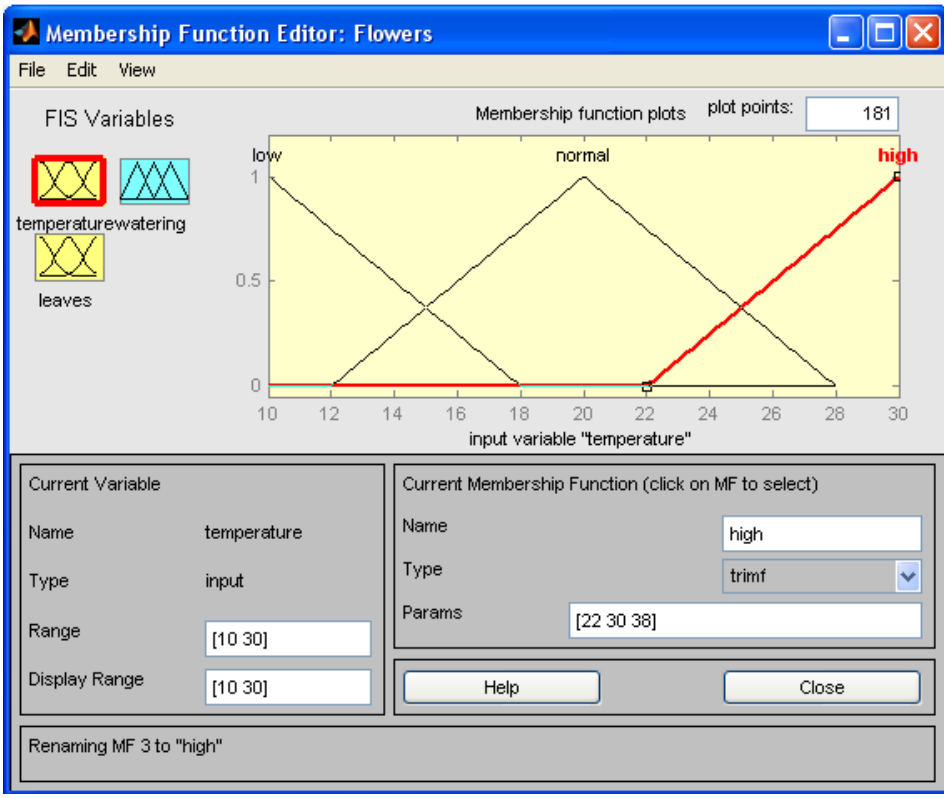


Рисунок 13 – Окно настройки параметров

11. Зададим функции принадлежности переменной **leaves** (рис. 14). По аналогии зададим следующие наименования термов **tight (узкие), middle(средние), wide (широкие).**

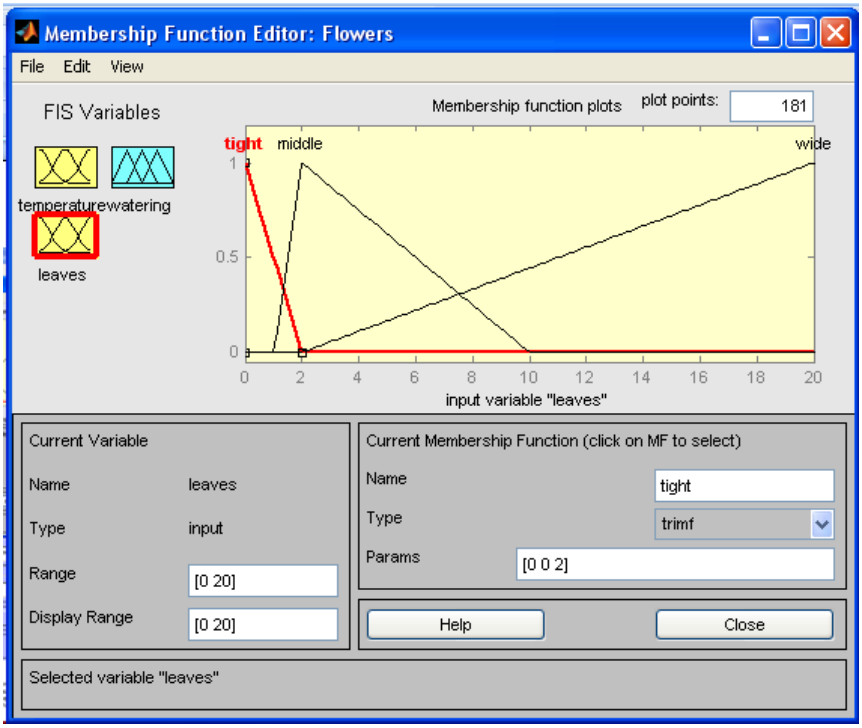


Рисунок 14 – Окно настройки переменной **leaves**

12. Зададим функции принадлежности переменной **watering** (рис. 15). Зададим следующие наименования термов: **seldom(редко), gently(умеренно), often(часто).**

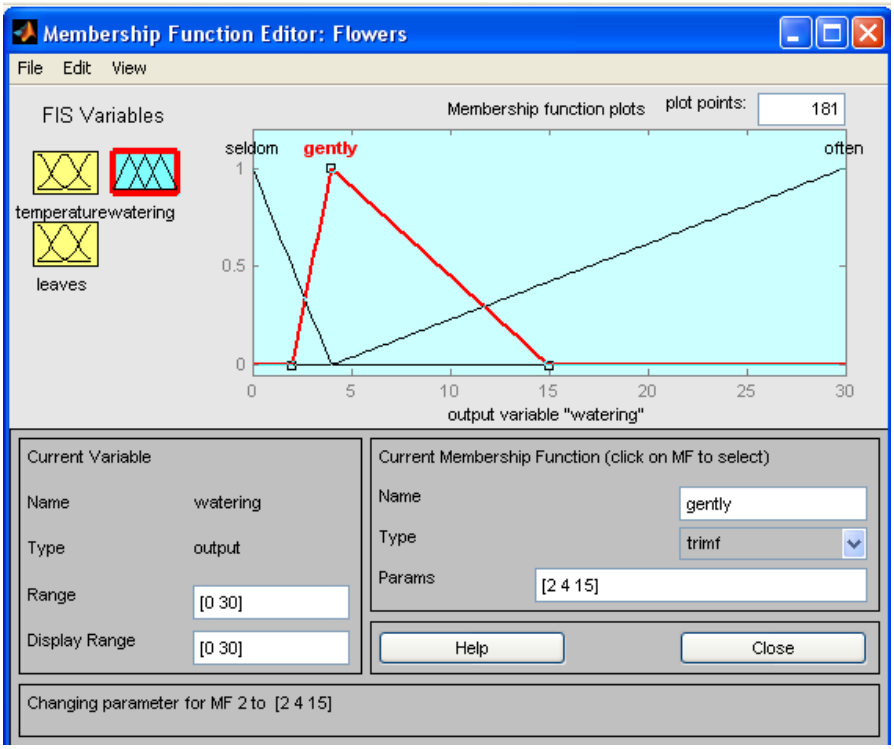


Рисунок 15 – Окно настройки переменной **watering**

13. Перейдем в редактор базы знаний **RuleEditor**. Для этого выберем в меню **Edit** выберем команду **Rules...**.

14. Сформулируем следующие правила, представленные на рисунке 16.

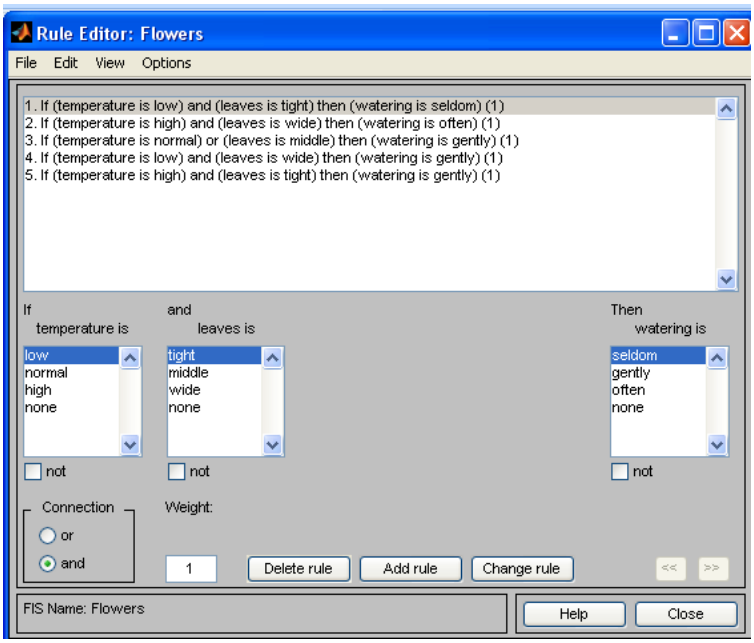


Рисунок 16 – Окно настройки правил

15. Сохраним созданную систему. Для этого в меню **File** выбираем в подменю **Export** команду **To workspace**.

16. В окне **Rules...** меню **View в** поле **Input** укажем значения входных переменных, для которых выполняется логический вывод (рис. 17).

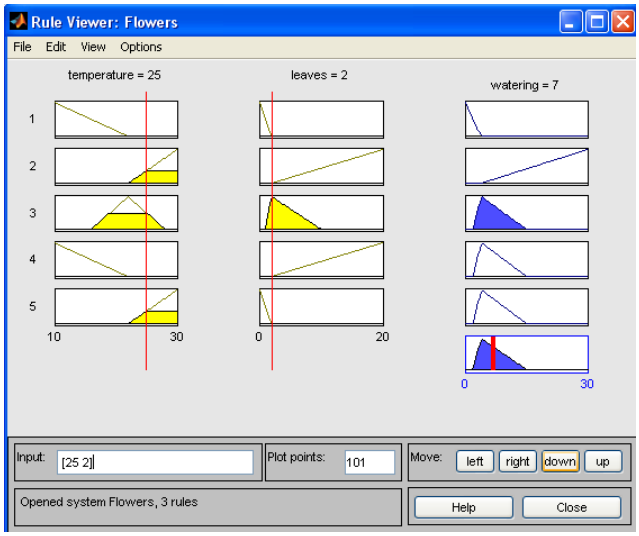


Рисунок 17 – Окно настройки входных переменных

Таким образом, при температуре 25°C и ширине листьев 2 см цветы нужно поливать 7 раз в месяц.

17. На следующем рисунке приведена поверхность “входы-выход”, соответствующая синтезированной нечеткой системе. Для вывода этого окна необходимо использовать команду **View surface...** меню **View.**

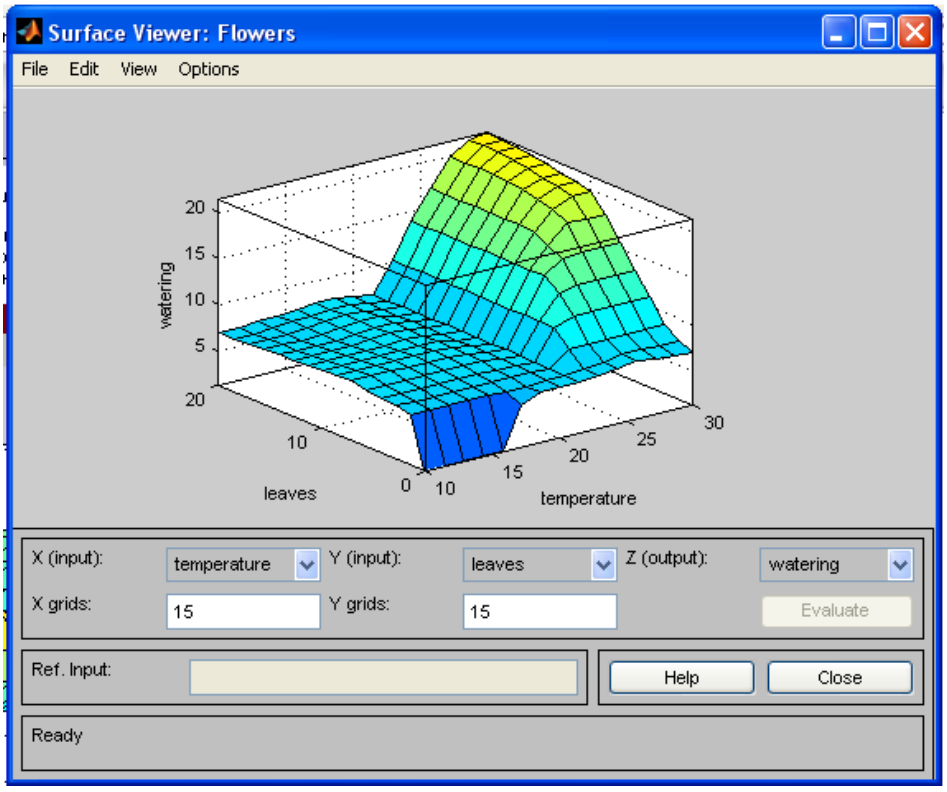


Рисунок 18 – Поверхность “входы-выход”

ЛАБОРАТОРНАЯ РАБОТА №2. 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 (рис. 19):

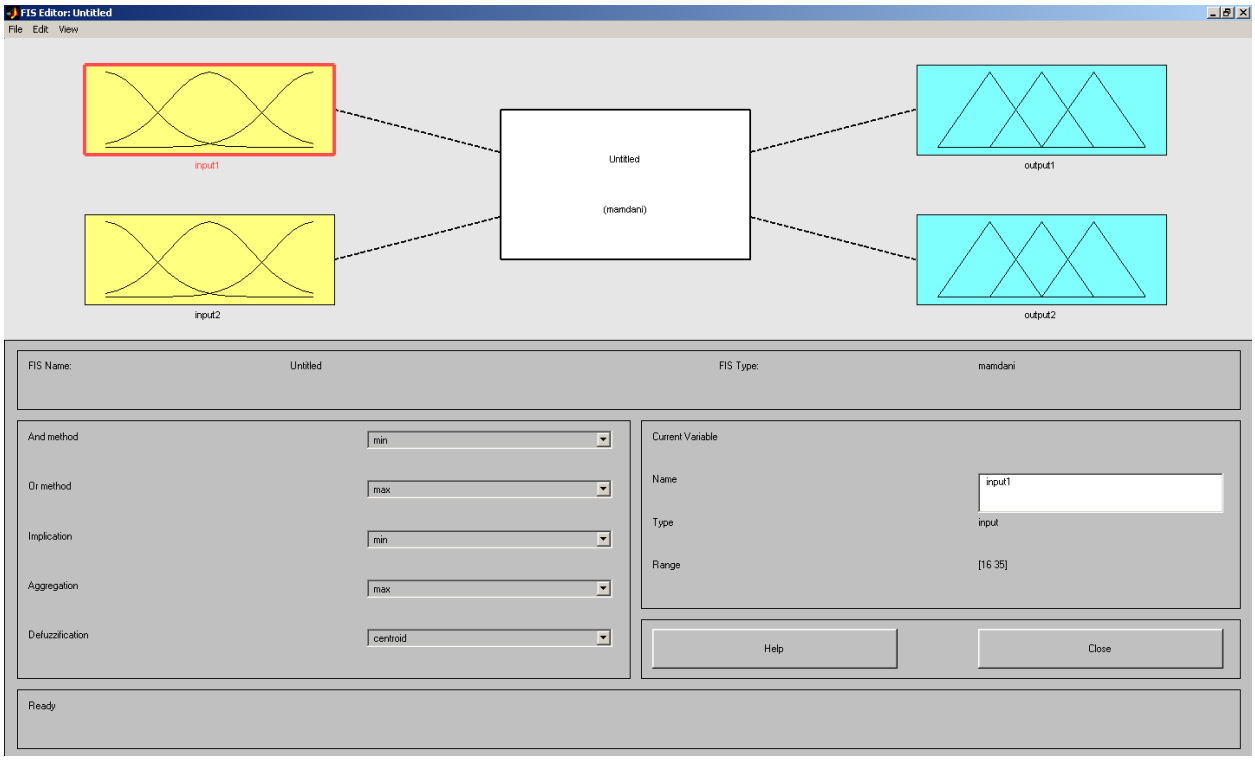


Рисунок 19 – Окно создания модели

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 рисунке 20 указан входной сигнал «температура внутри помещения».

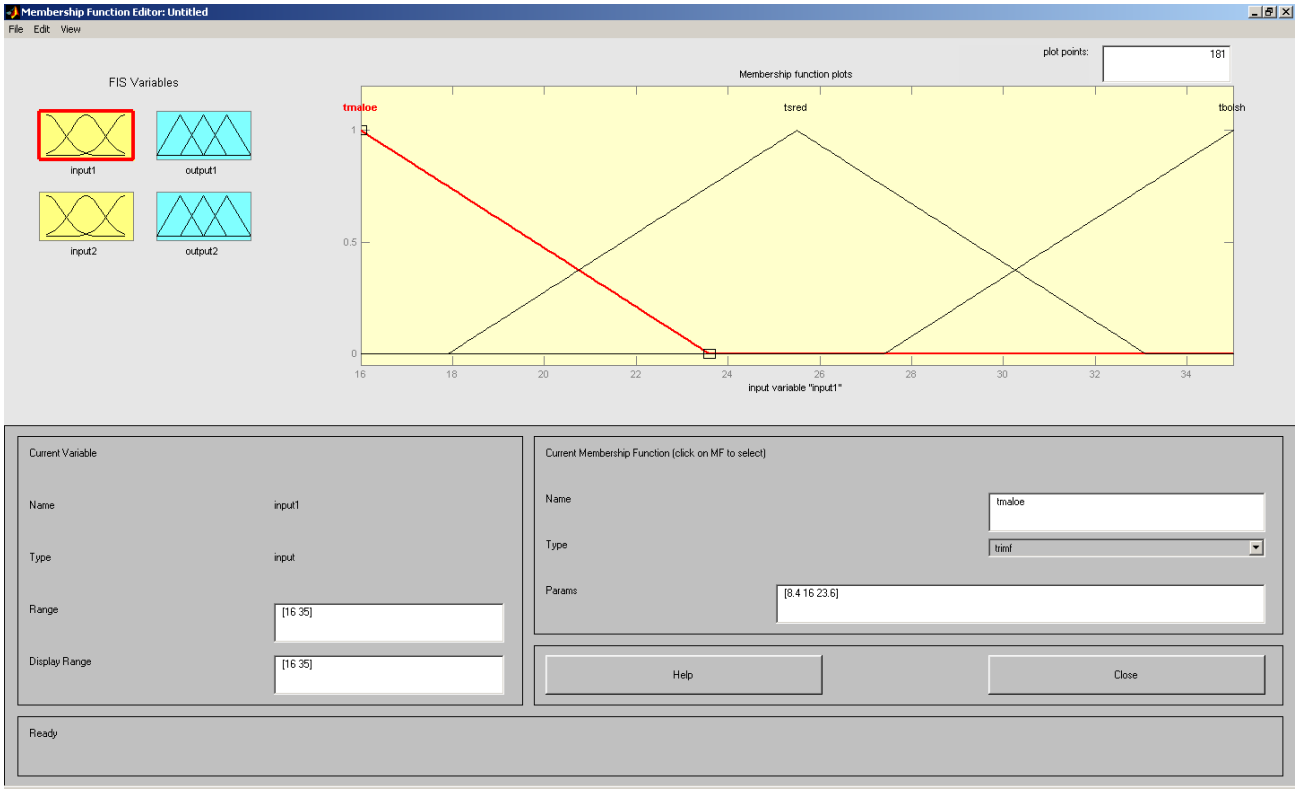


Рисунок 20 – Входной сигнал «температура внутри помещения»

На рисунке 21 указан входной сигнал «скорость изменения температуры в помещение».

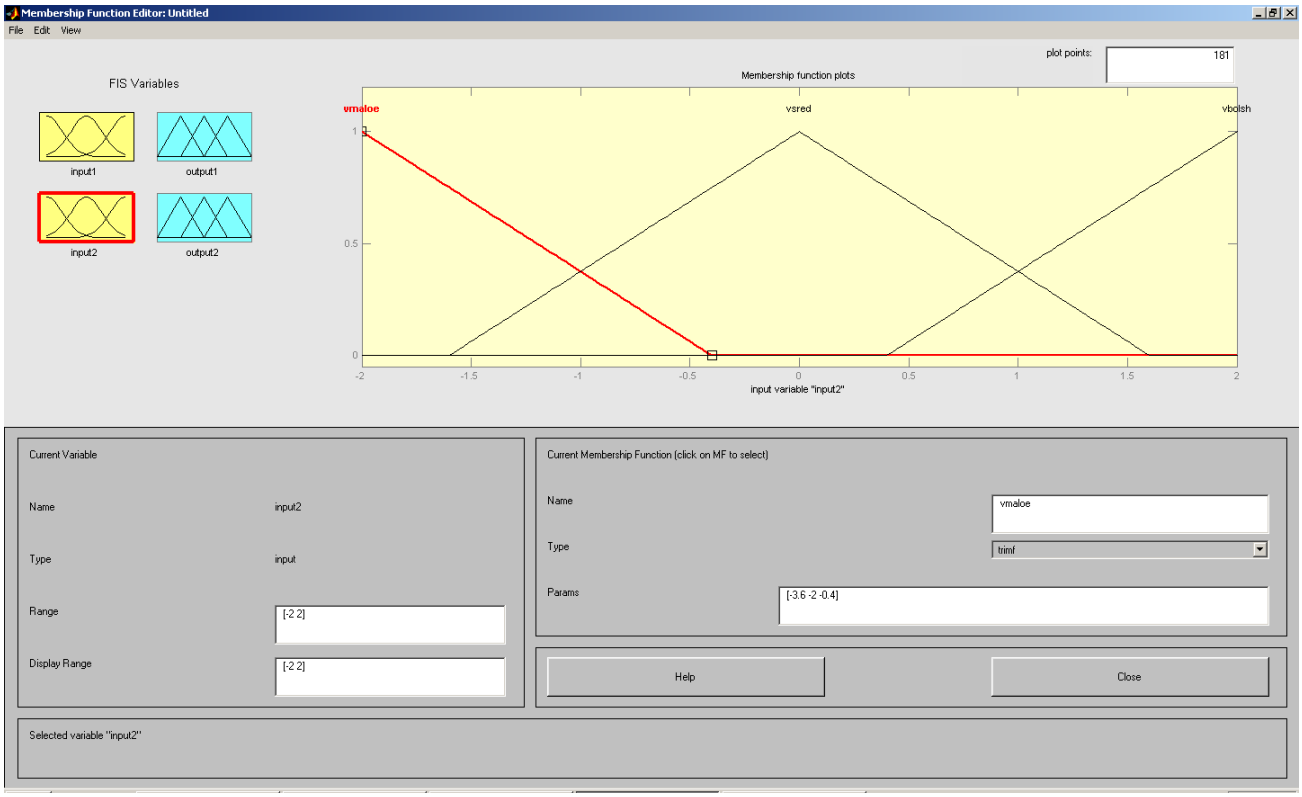


Рисунок 21 – Входной сигнал «скорость изменения температуры в помещение»

На рисунке 22 указан выходной сигнал «скорость вращения вентилятора».

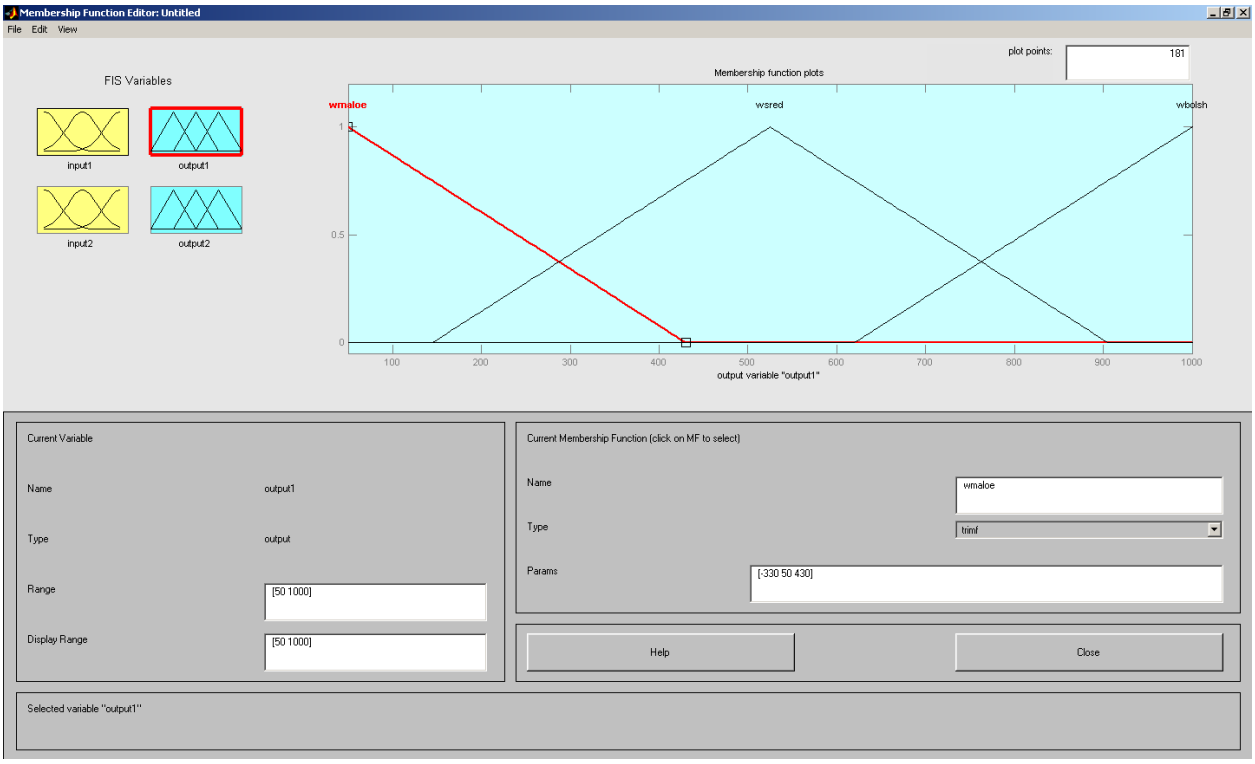


Рисунок 22 – Выходной сигнал «скорость вращения вентилятора»

На рисунке 23 указан выходной сигнал «расход охлаждающей жидкости».

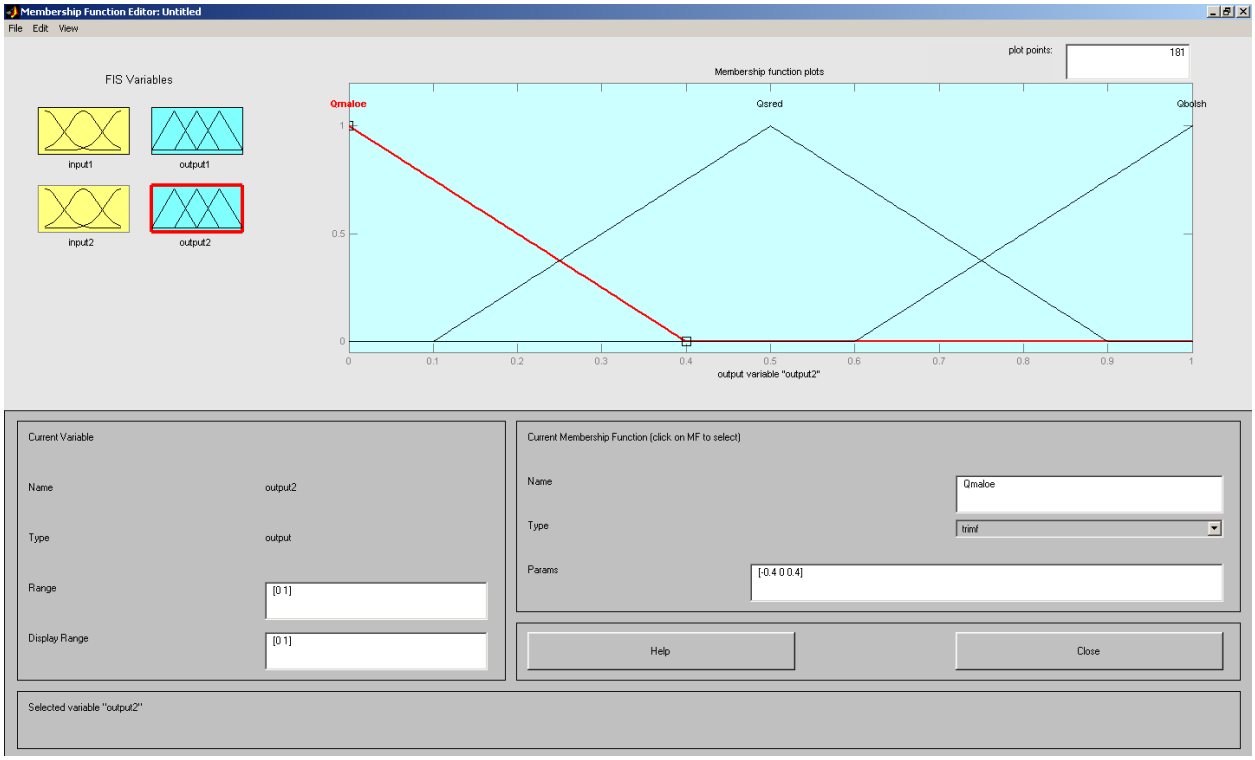


Рисунок 23 – Окно создания модели

4. Создание правил

Для различных ситуаций создадим правила задания режима работы (рис. 24).

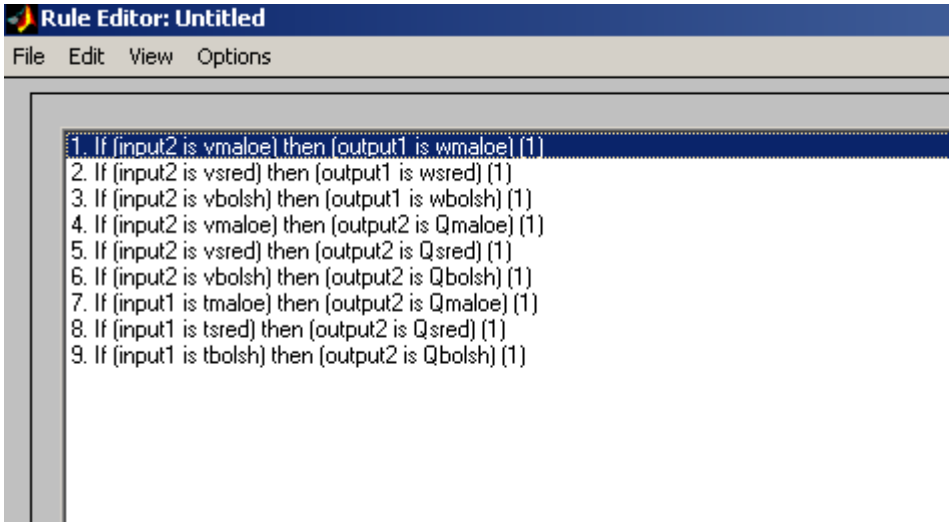


Рисунок 24 – Правила задания режима работы

Эти правила можно представить графически (рис. 25).

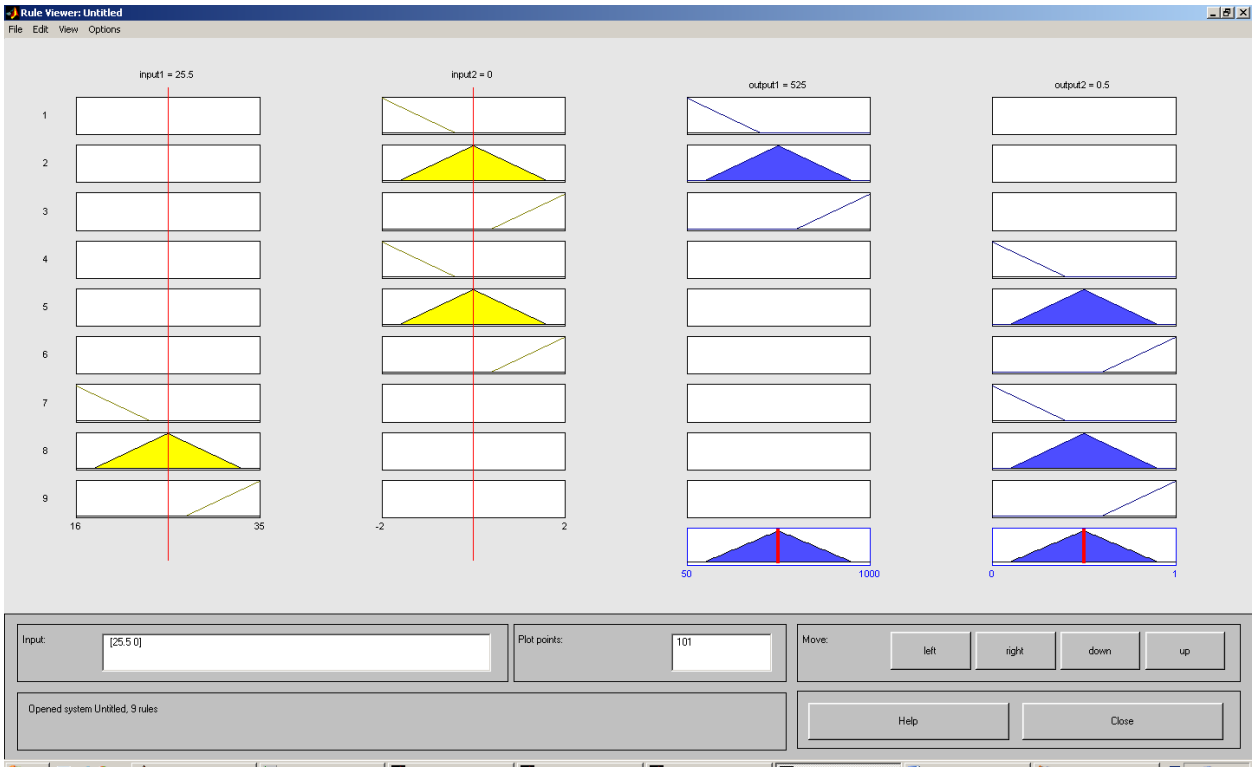


Рисунок 25 – Графическое представление правил

На рисунке 26 представлены измененные входные сигналы.

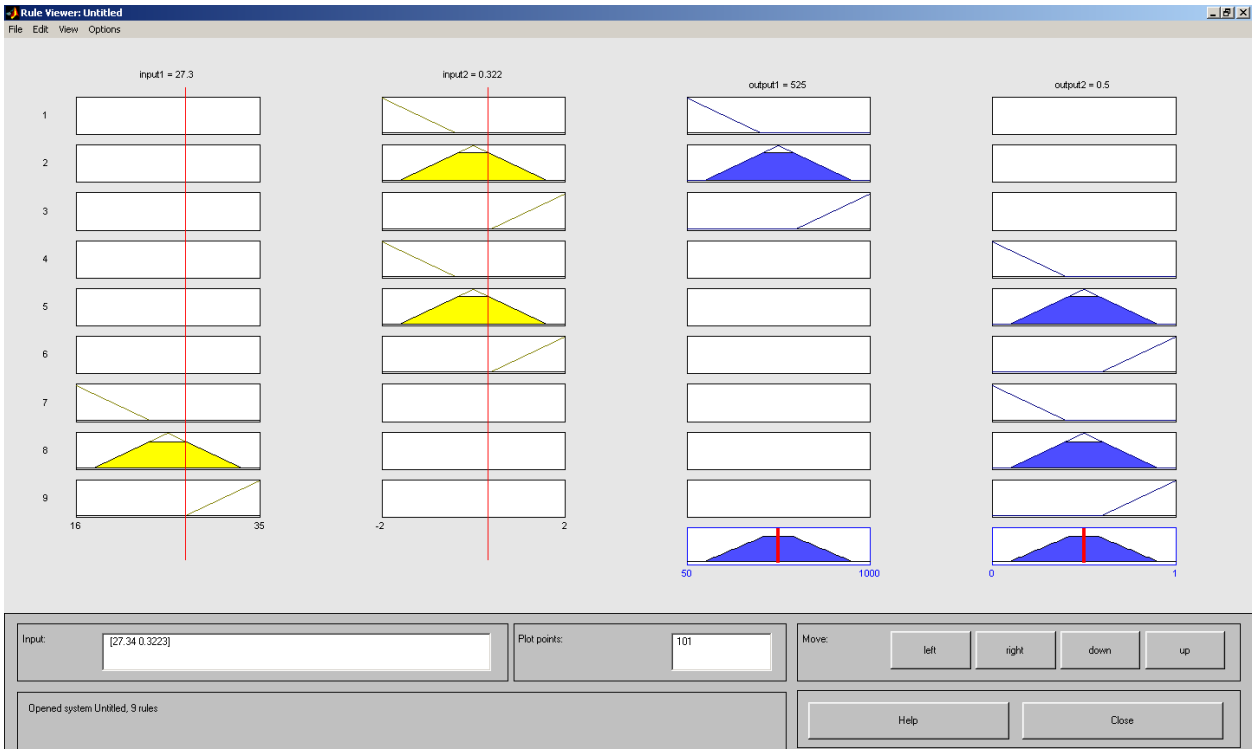


Рисунок 26 – Измененные входные сигналы

5. *Анализ полученной модели*

Построим графики зависимости скорости вращения вентилятора, температуры в помещение и скорости изменения температуры (рис. 27, 28).

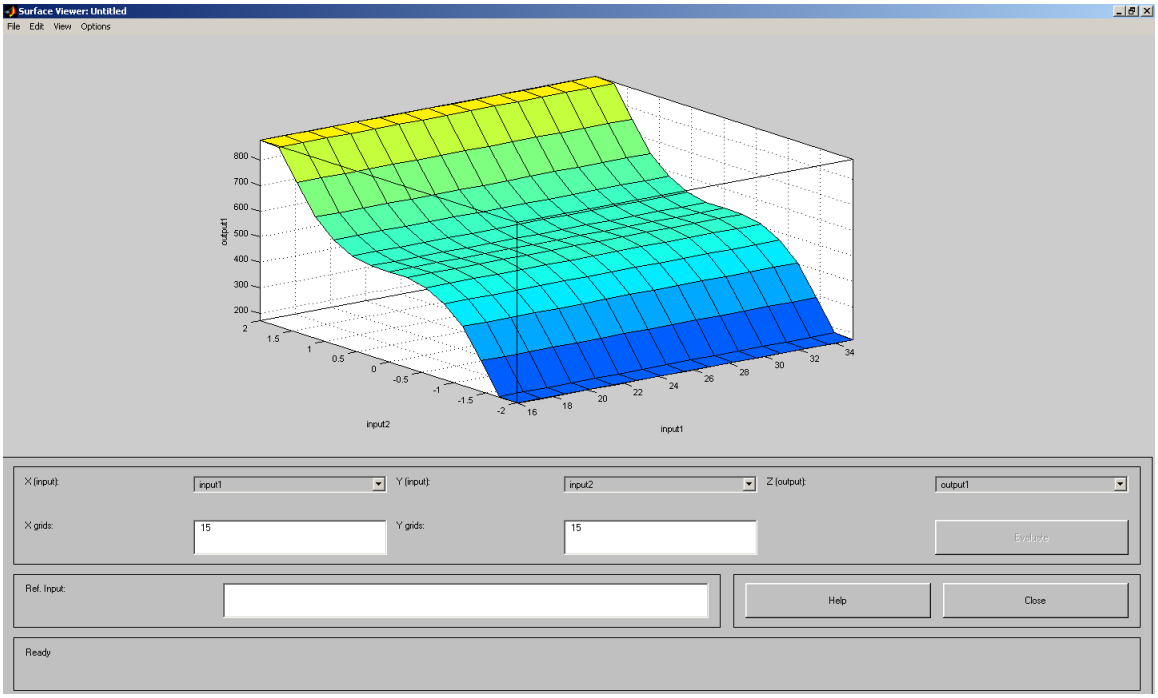


Рисунок 27 – График зависимости скорости вращения вентилятора, температуры в помещение и скорости изменения температуры

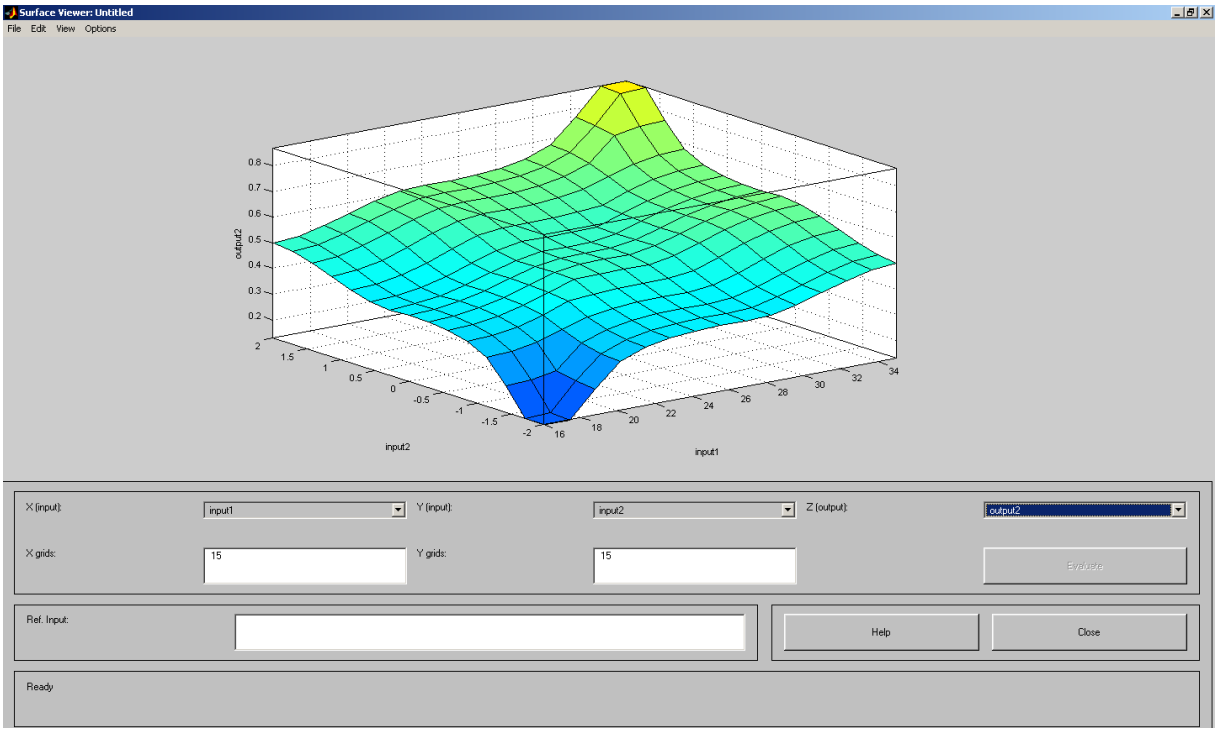
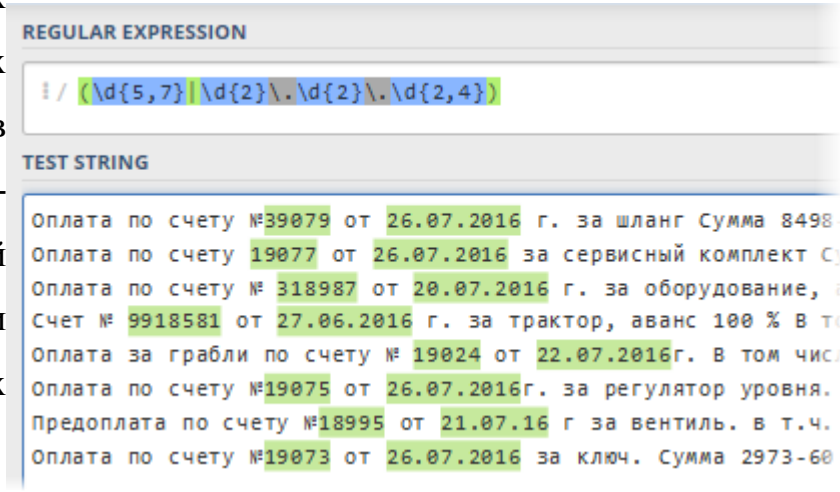


Рисунок 28 – График зависимости скорости вращения вентилятора, температуры в помещение и скорости изменения температуры

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

ЛАБОРАТОРНАЯ РАБО № 3. АНАЛИЗ ТЕКСТА РЕГУЛЯРНЫМИ ВЫРАЖЕНИЯМИ (REGEXP) В EXCEL

Одной из самых трудоемких и неприятных задач при работе с текстом в Excel является парсинг - разбор буквенно-цифровой "каши" на составляющие и извлечение из нее нужных нам фрагментов. Например:



- извлечение почтового индекса из адреса (хорошо, если индекс всегда в начале, а если нет?)
- нахождение номера и даты счета из описания платежа в банковской выписке
- извлечение ИНН из разношерстных описаний компаний в списке контрагентов
- поиск номера автомобиля или артикула товара в описании и т.д.

Обычно во подобных случаях, после получасового муторного ковыряния в тексте вручную, в голову начинают приходить мысли как-то автоматизировать этот процесс (особенно если данных много). Решений тут несколько и с разной степенью сложности-эффективности:

- Использовать встроенные текстовые функции Excel для поиска-нарезки-склейки текста: ЛЕВСИМВ (LEFT), ПРАВСИМВ (RIGHT), ПСТР (MID), СЦЕПИТЬ (CONCATENATE) и ее аналоги, ОБЪЕДИНИТЬ (JOINTEXT), СОВПАД (EXACT) и т.д. Этот способ хорош, если в тексте есть четкая логика (например, индекс всегда в начале адреса). В противном случае формулы существенно усложняются и, порой,

дело доходит даже до формул массива, что сильно тормозит на больших таблицах.

- Использование оператора проверки текстового подобия Like из Visual Basic, обернутого в пользовательскую макро-функцию. Это позволяет реализовать более гибкий поиск с использованием символов подстановки (*,#,? и т.д.) К сожалению, этот инструмент не умеет извлекать нужную подстроку из текста - только проверять, содержится ли она в нем.

Кроме вышеперечисленного, есть еще один подход, очень известный в узких кругах профессиональных программистов, веб-разработчиков и прочих технарей - это регулярные выражения (Regular Expressions = RegExp = "регэкспы" = "регулярки"). Упрощенно говоря, RegExp - это язык, где с помощью специальных символов и правил производится поиск нужных подстрок в тексте, их извлечение или замена на другой текст. Регулярные выражения - это очень мощный и красивый инструмент, на порядок превосходящий по возможностям все остальные способы работы с текстом. Многие языки программирования (C#, PHP, Perl, JavaScript...) и текстовые редакторы (Word, Notepad++...) поддерживают регулярные выражения.

Microsoft Excel, к сожалению, не имеет поддержки RegExp по-умолчанию "из коробки", но это легко исправить с помощью VBA. Откройте редактор Visual Basic с вкладки Разработчик (Developer) или сочетанием клавиш Alt+F11. Затем вставьте новый модуль через меню Insert - Module и скопируйте туда текст вот такой макрофункции:

```
1   Public Function RegExpExtract(Text As String, Pattern As String, Optional
2   Item As Integer = 1) As String
3       On Error GoTo ErrHandl
4       Set regex = CreateObject("VBScript.RegExp")
5       regex.Pattern = Pattern
6       regex.Global = True
```

```
7         If regex.Test(Text) Then
8             Set matches = regex.Execute(Text)
9             RegExpExtract = matches.Item(Item - 1)
10            Exit Function
11        End If
12    ErrHandl:
13        RegExpExtract = CVErr(xlErrValue)
      End Function
```

Теперь можно закрыть редактор Visual Basic и, вернувшись в Excel, опробовать нашу новую функцию. Синтаксис у нее следующий:

=RegExpExtract(Txt ; Pattern ; Item)

где

- Txt - ячейка с текстом, который мы проверяем и из которого хотим извлечь нужную нам подстроку
- Pattern - маска (шаблон) для поиска подстроки
- Item - порядковый номер подстроки, которую надо извлечь, если их несколько (если не указан, то выводится первое вхождение)

Самое интересное тут, конечно, это Pattern - строка-шаблон из спецсимволов "на языке" RegExp, которая и задает, что именно и где мы хотим найти. Вот самые основные из них - для начала:

Паттерн	Описание
.	Самое простое - это точка. Она обозначает любой символ в шаблоне на указанной позиции.
\s	Любой символ, выглядящий как пробел (пробел, табуляция или перенос строки).
\S	Анти-вариант предыдущего шаблона, т.е. любой НЕпробельный символ.
\d	Любая цифра

\D	Анти-вариант предыдущего, т.е. любая НЕ цифра
\w	Любой символ латиницы (A-Z), цифра или знак подчеркивания
\W	Анти-вариант предыдущего, т.е. не латиница, не цифра и не подчеркивание.
[*символы*]	В квадратных скобках можно указать один или несколько символов, разрешенных на указанной позиции в тексте. Например ст[уо]л будет соответствовать любому из слов: *стол* или *стул*. Также можно не перечислять символы, а задать их диапазоном через дефис, т.е. вместо [ABDCDEF] написать [A-F]. или вместо [4567] ввести [4-7]. Например, для обозначения всех символов кириллицы можно использовать шаблон [а-яА-ЯёЁ].
[^*символы*]	Если после открывающей квадратной скобки добавить символ "крышки" ^, то набор приобретет обратный смысл - на указанной позиции в тексте будут разрешены все символы, кроме перечисленных. Так, шаблон [^ЖМ]уть найдет *Путь* или *Суть* или *Забудь*, но не *Жуть* или *Муть*, например.
|	Логический оператор ИЛИ (OR) для проверки по любому из указанных критериев. Например (счет|счёт|invoice) будет искать в тексте любое из указанных слов. Обычно набор вариантов заключается в скобки.
^	Начало строки
$	Конец строки
\b	Край слова

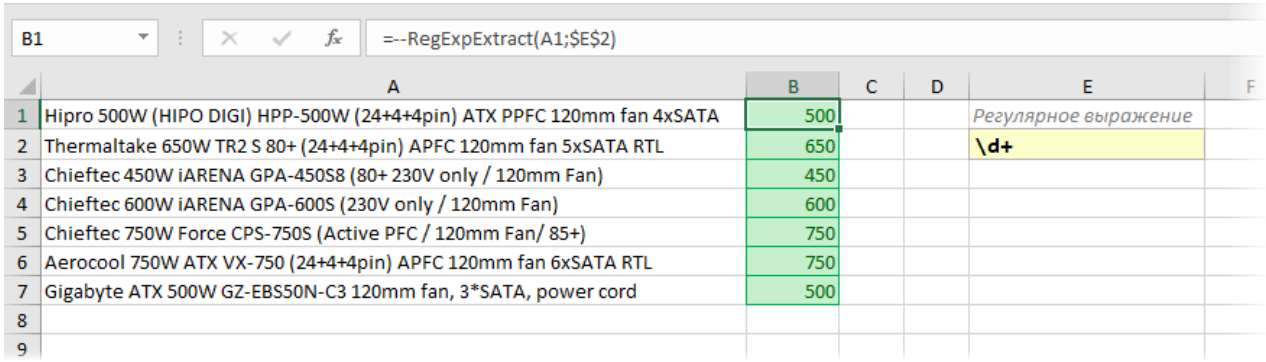
Если мы ищем определенное количество символов, например, шестизначный почтовый индекс или все трехбуквенные коды товаров, то на помощь нам приходят *квантификаторы* или *кванторы* - специальные выражения, задающие количество искомых знаков. Квантификаторы применяются к тому символу, что стоит перед ним:

Квантор	Описание
?	Ноль или одно вхождение. Например .? будет означать один любой символ или его отсутствие.
+	Одно или более вхождений. Например \d+ означает любое количество цифр (т.е. любое число от 0 до бесконечности).
*	Ноль или более вхождений, т.е. любое количество. Так \s* означает любое количество пробелов или их отсутствие.
{число} или {число1,число2}	Если нужно задать строго определенное количество вхождений, то оно задается в фигурных скобках. Например \d{6} означает строго шесть цифр, а шаблон \s{2,5} - от двух до пяти пробелов

Теперь давайте перейдем к самому интересному - разбору применения созданной функции и того, что узнали о паттернах на практических примерах из жизни.

Извлекаем числа из текста

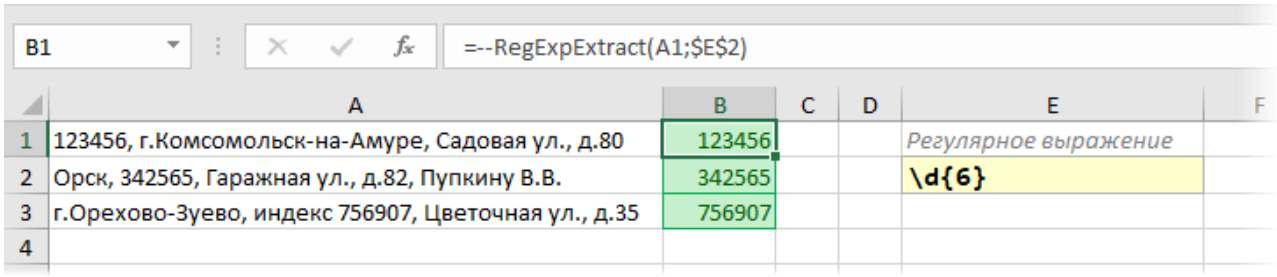
Для начала разберем простой случай - нужно извлечь из буквенно-цифровой каши первое число, например мощность источников бесперебойного питания из прайс-листа:



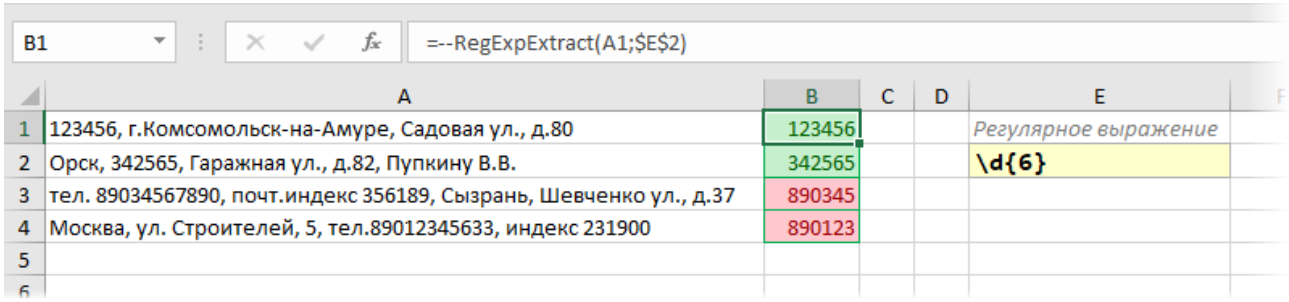
Логика работы регулярного выражения тут простая: \d - означает любую цифру, а квантор + говорит о том, что их количество должно быть одна или больше. Двойной минус перед функцией нужен, чтобы "на лету" преобразовать извлеченные символы в полноценное число из числа-как-текст.

Почтовый индекс

На первый взгляд, тут все просто - ищем ровно шесть цифр подряд. Используем спецсимвол \d для цифры и квантор {6} для количества знаков:

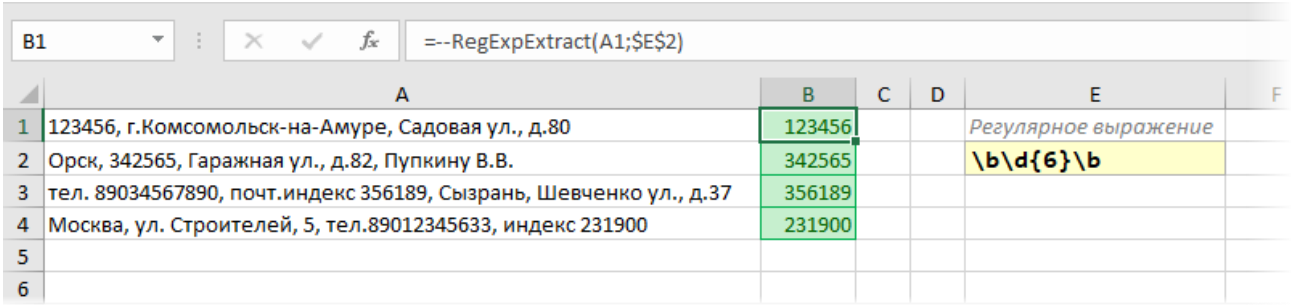


Однако, возможна ситуация, когда левее индекса в строке стоит еще один большой набор цифр подряд (номер телефона, ИНН, банковский счет и т.д.) Тогда наша регулярка выдернет из нее первых 6 цифр, т.е. сработает некорректно:



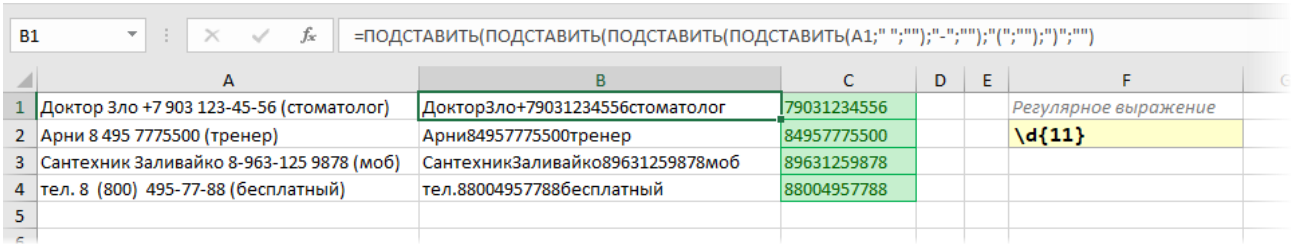
Чтобы этого не происходило, необходимо добавить в наше регулярное выражение по краям модификатор \b означающий конец слова. Это даст понять

Excel, что нужный нам фрагмент (индекс) должен быть отдельным словом, а не частью другого фрагмента (номера телефона):



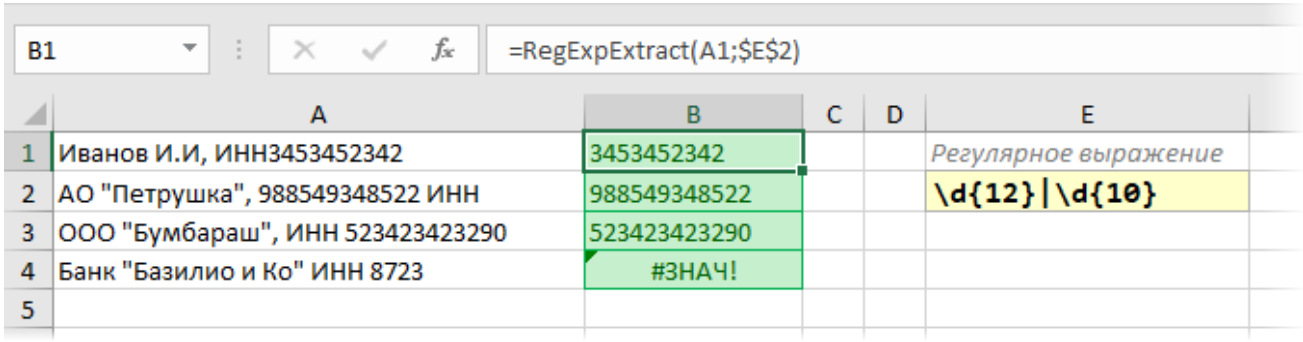
Телефон

Проблема с нахождением телефонного номера среди текста состоит в том, что существует очень много вариантов записи номеров - с дефисами и без, через пробелы, с кодом региона в скобках или без и т.д. Поэтому, на мой взгляд, проще сначала вычистить из исходного текста все эти символы с помощью нескольких вложенных друг в друга функций ПОДСТАВИТЬ (SUBSTITUTE), чтобы он склеился в единое целое, а потом уже примитивной регуляркой \d{11} вытаскивать 11 цифр подряд:

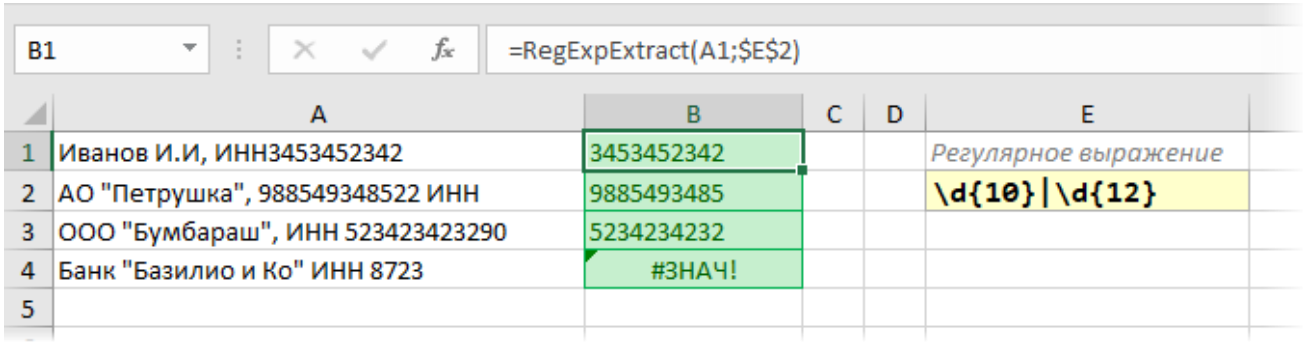


ИНН

Тут чуть сложнее, т.к. ИНН (в России) бывает 10-значный (у юрлиц) или 12-значный (у физлиц). Если не придираться особо, то вполне можно удовлетвориться регуляркой \d{10,12}, но она, строго говоря, будет вытаскивать все числа от 10 до 12 знаков, т.е. и ошибочно введенные 11-значные. Правильнее будет использовать два шаблона, связанных логическим ИЛИ оператором | (вертикальная черта):



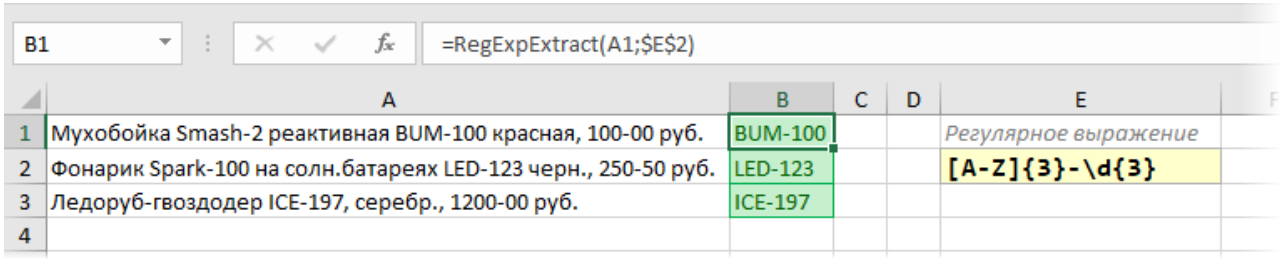
Обратите внимание, что в запросе мы сначала ищем 12-разрядные, и только потом 10-разрядные числа. Если же записать нашу регулярку наоборот, то она будет вытаскивать для всех, даже длинных 12-разрядных ИНН, только первые 10 символов. То есть после срабатывания первого условия дальнейшая проверка уже не производится:



Это принципиальное отличие оператора | от стандартной экселевской логической функции ИЛИ (OR), где от перестановки аргументов результат не меняется.

Артикулы товаров

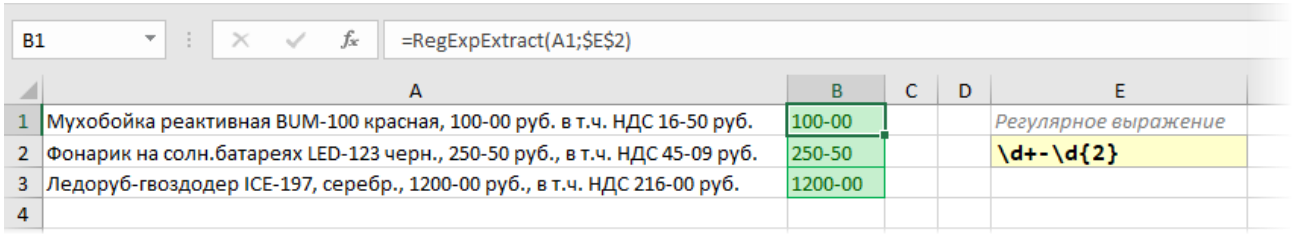
Во многих компаниях товарам и услугам присваиваются уникальные идентификаторы - артикулы, SAP-коды, SKU и т.д. Если в их обозначениях есть логика, то их можно легко вытаскивать из любого текста с помощью регулярных выражений. Например, если мы знаем, что наши артикулы всегда состоят из трех заглавных английских букв, дефиса и последующего трехразрядного числа, то:



Логика работы шаблона тут проста. [A-Z] - означает любые заглавные буквы латиницы. Следующий за ним квантор {3} говорит о том, что нам важно, чтобы таких букв было именно три. После дефиса мы ждем три цифровых разряда, поэтому добавляем на конце \d{3}

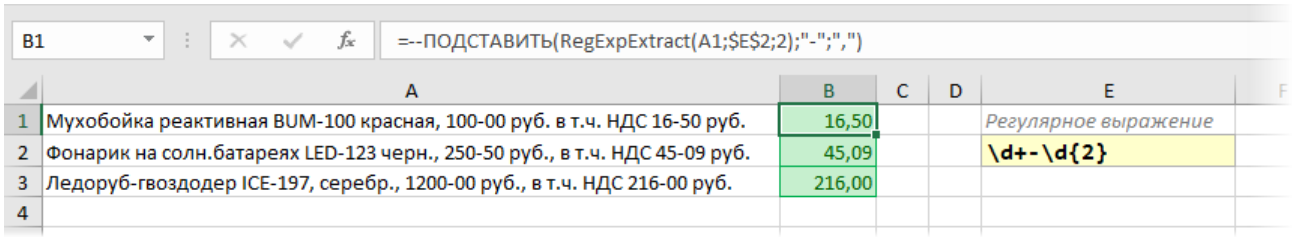
Денежные суммы

Похожим на предыдущий пункт образом, можно вытаскивать и цены (стоимости, НДС...) из описания товаров. Если денежные суммы, например, указываются через дефис, то:



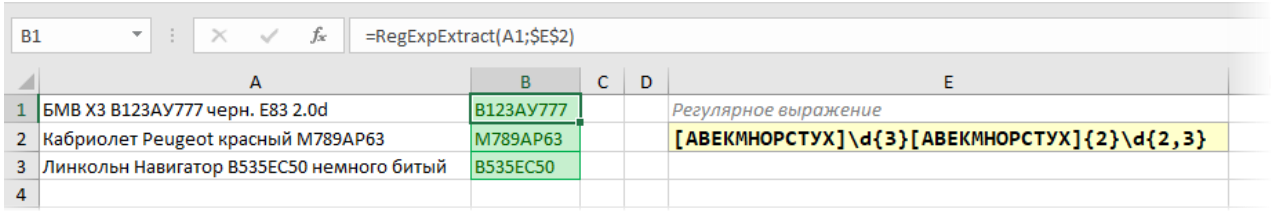
Паттерн \d с квантором + ищет любое число до дефиса, а \d{2} будет искать копейки (два разряда) после.

Если нужно вытащить не цены, а НДС, то можно воспользоваться третьим необязательным аргументом нашей функции RegExpExtract, задающим порядковый номер извлекаемого элемента. И, само-собой, можно заменить функцией ПОДСТАВИТЬ (SUBSTITUTE) в результатах дефис на стандартный десятичный разделитель и добавить двойной минус в начале, чтобы Excel интерпретировал найденный НДС как нормальное число:



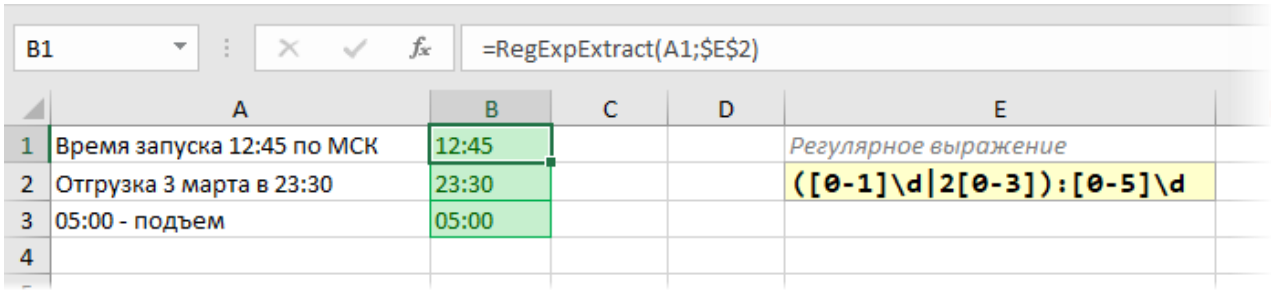
Автомобильные номера

Если не брать спецтранспорт, прицепы и прочие мотоциклы, то стандартный российский автомобильный номер разбирается по принципу "буква - три цифры - две буквы - код региона". Причем код региона может быть 2- или 3-значным, а в качестве букв применяются только те, что похожи внешне на латиницу. Таким образом, для извлечения номеров из текста нам поможет следующая регулярка:



Время

Для извлечения времени в формате ЧЧ:ММ подойдет такое регулярное выражение:



После двоеточия фрагмент [0-5]\d, как легко сообразить, задает любое число в интервале 00-59. Перед двоеточием в скобках работают два шаблона, разделенных логическим ИЛИ (вертикальной чертой):

- [0-1]\d - любое число в интервале 00-19
- 2[0-3] - любое число в интервале 20-23

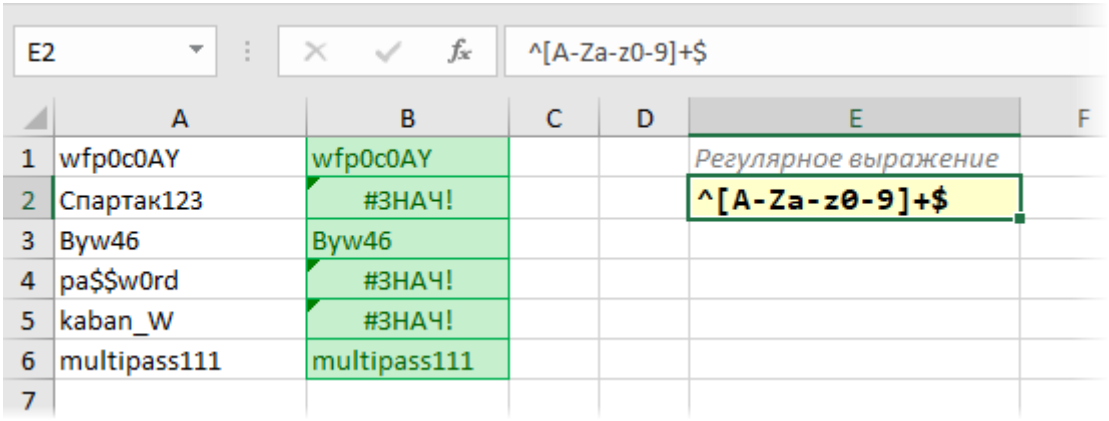
К полученному результату можно применить дополнительно еще и стандартную Excel'евскую функцию ВРЕМЯ (TIME), чтобы преобразовать его в понятный программе и пригодный для дальнейших расчетов формат времени.

Проверка пароля

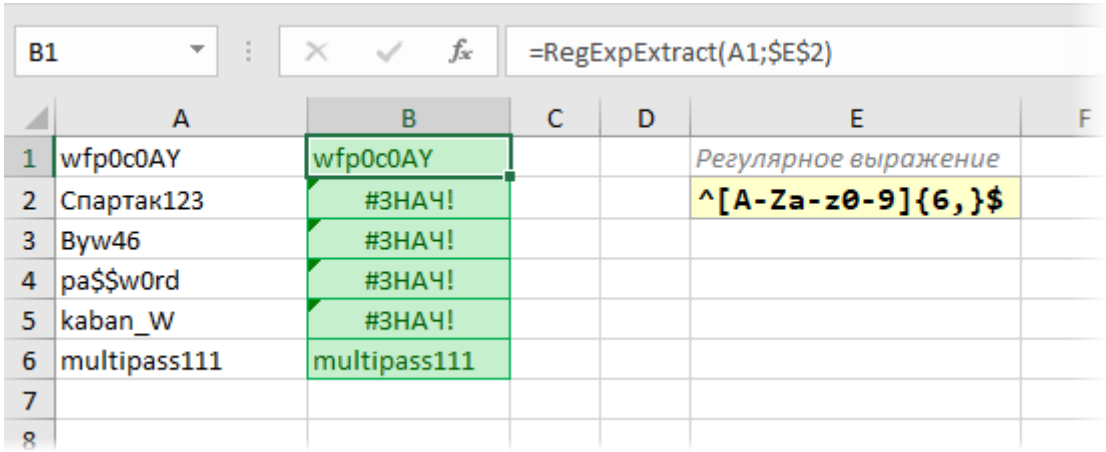
Предположим, что нам надо проверить список придуманных пользователями паролей на корректность. По нашим правилам, в паролях могут

быть только английские буквы (строчные или прописные) и цифры. Пробелы, подчеркивания и другие знаки препинания не допускаются.

Проверку можно организовать с помощью вот такой несложной регулярки:

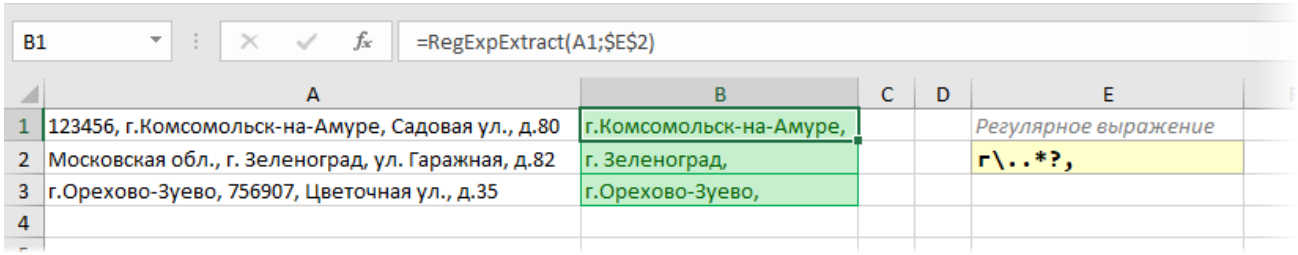


По сути, таким шаблоном мы требуем, чтобы между началом (^) и концом ($) в нашем тексте находились только символы из заданного в квадратных скобках набора. Если нужно проверить еще и длину пароля (например, не меньше 6 символов), то квантор + можно заменить на интервал "шесть и более" в виде {6,}:



Город из адреса

Допустим, нам нужно вытащить город из строки адреса. Поможет регулярка, извлекающая текст от "г." до следующей запятой:

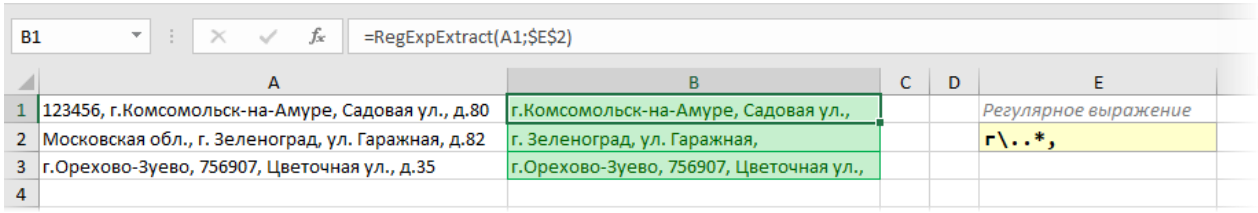


Давайте разберем этот шаблон поподробнее.

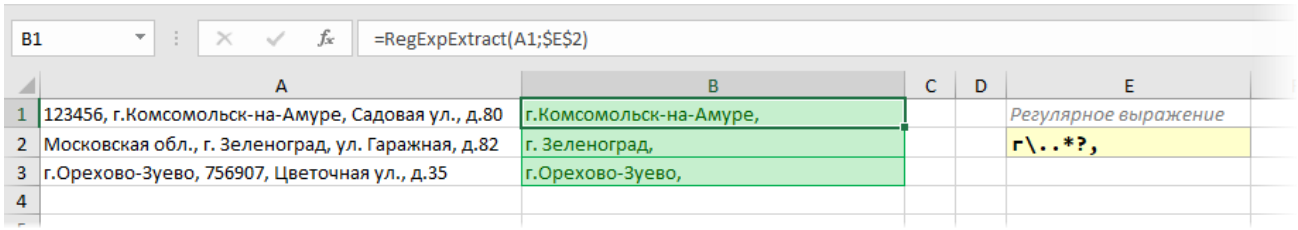
Если вы прочитали текст выше, то уже поняли, что некоторые символы в регулярных выражениях (точки, звездочки, знаки доллара и т.д.) несут особый смысл. Если же нужно искать сами эти символы, то перед ними ставится обратная косая черта (иногда это называют *экранированием*). Поэтому при поиске фрагмента "г." мы должны написать в регулярке г\. если ищем плюсик, то \+ и т.д.

Следующих два символа в нашем шаблоне - точка и звездочка-квантор - обозначают любое количество любых символов, т.е. любое название города.

На конце шаблона стоит запятая, т.к. мы ищем текст от "г." до запятой. Но ведь в тексте может быть несколько запятых, правда? Не только после города, но и после улицы, дома и т.д. На какой из них будет останавливаться наш запрос? Вот за это отвечает вопросительный знак. Без него наша регулярка вытаскивала бы максимально длинную строку из всех возможных:

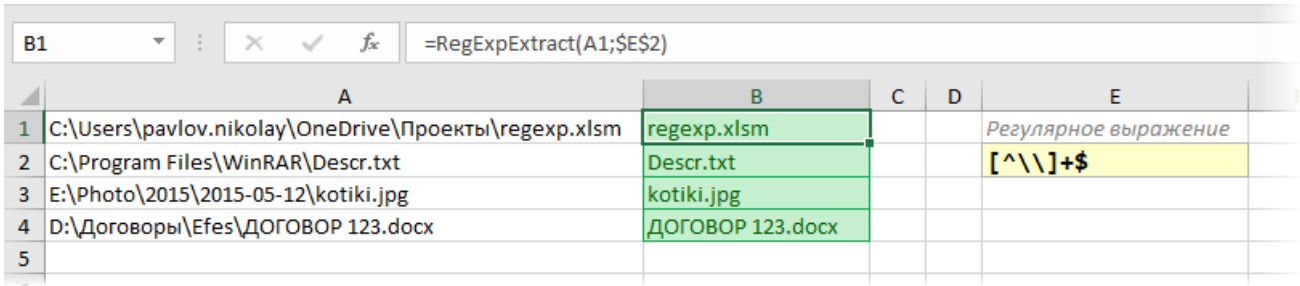


В терминах регулярных выражений, такой шаблон является "жадным". Чтобы исправить ситуацию и нужен вопросительный знак - он делает квантор, после которого стоит, "скупым" - и наш запрос берет текст только до первой встречной запятой после "г.":



Имя файла из полного пути

Еще одна весьма распространенная ситуация - вытащить имя файла из полного пути. Тут поможет простая регулярка вида:



Тут фишка в том, что поиск, по сути, происходит в обратном направлении – от конца к началу, т.к. в конце нашего шаблона стоит $, и мы ищем все, что перед ним до первого справа обратного слэша. Бэкслэш заэкранирован, как и точка в предыдущем примере.

ЛАБОРАТОРНАЯ РАБОТА № 4. АВТОМАТИЗАЦИЯ РАСЧЕТОВ В СРЕДЕ MATLAB

Цель работы - изучить возможности программной среды; получить навыки расчетов различной сложности и графического представления данных; получение навыков по программированию в данной среде.

Содержание лабораторной работы

Освоение интерфейса и входного языка программы, определение возможностей и ограничений.

Расчеты и визуализация результатов.

Основные приемы программирования. Создание сценариев и функций.

Общие положения

Система MATLAB предлагается разработчиками (фирма Math Works, Inc.) как лидирующий на рынке, в первую очередь, в системе военнопромышленного комплекса, в аэрокосмической отрасли и автомобилестроении, язык программирования высокого уровня для технических вычислений с большим числом стандартных пакетов (прикладных программ).

Система MATLAB вобрала в себя не только передовой опыт развития и компьютерной реализации численных методов, накопленный за последние три десятилетия, но и весь опыт становления математики за всю историю человечества. Популярности системы способствует ее мощное расширение Simulink, предоставляющее удобные и простые средства, в том числе визуальное объектно-ориентированное программирование, для моделирования линейных и нелинейных динамических систем, а также множество других пакетов расширения системы.

Система MATLAB выполняет сложные и трудоемкие операции над векторами и матрицами даже в режиме прямых вычислений без какого-либо программирования. Ею можно пользоваться как мощнейшим калькулятором, в котором наряду с обычными арифметическими и алгебраическими действиями

могут использоваться такие сложные операции, как инвертирование матрицы, вычисление ее собственных значений и принадлежащих им векторов, решение систем линейных уравнений, вывод графиков двумерных и трехмерных функций и многое другое.

Обычные числа и переменные в MATLAB рассматриваются как матрицы размером 1x1, что дает единообразные формы и методы проведения операций над обычными числами и массивами. Данная операция обычно называется векторизацией. Векторизация обеспечивает и упрощение записи операций, производимых одновременно над всеми элементами векторов и матриц, и сущест11 венное повышение скорости их выполнения. Это также означает, что большинство функций может работать с аргументами в виде векторов и матриц. При необходимости вектора и матрицы преобразуются в массивы, и значения вычисляются для каждого их элемента.

MATLAB - расширяемая система, и ее легко приспособить к решению нужных вам классов задач. Ее огромное достоинство заключается в том, что это расширение достигается естественным путем и реализуется в виде так называемых т-файлов (с расширением .m). Иными словами, расширения системы хранятся на жестком диске компьютера и в нужный момент вызываются для использования точно так же, как встроенные в MATLAB (внутренние) функции и процедуры. Дополнительный уровень системы образуют ее пакеты расширения (toolbox). Они позволяют быстро ориентировать систему на решение задач в той или иной предметной области: в специальных разделах математики, в физике и в астрономии, в области нейтронных сетей и средств телекоммуникаций, в математическом моделировании, проектировании событийно-управляемых систем и т. д. Благодаря этому MATLAB обеспечивает высочайший уровень адаптации к решению задач конечного пользователя. Более подробно справку о возможностях данной среды можно найти в источниках /1-12/ и электронных учебниках кафедры в лаборатории "Информационных систем".

Выполнение лабораторной работы

Освоение интерфейса и входного языка программы

1. Запустить программу MATLAB. Система MATLAB создана таким образом, что любые (подчас весьма сложные) вычисления можно выполнять в режиме прямых вычислений, то есть без подготовки программы.

2. Ознакомиться с основными диалоговыми панелями и меню программы.

3. В окне Команды вычислить выражения: 5+80-л; sin(5); esin(45)+1/cos(25). Особенности вычислений:

• для указания ввода исходных данных используется символ »;

• для блокировки вывода результата вычислений некоторого выражения после него надо установить знак;

• если не указана переменная для значения результата вычислений, то MATLAB назначает такую переменную с именем ans;

• знаком присваивания является привычный математикам знак равенства =, а не комбинированный знак :=, как во многих других языках программирования и математических системах;

• результат вычислений выводится в строках вывода (без знака »);

• встроенные функции (например, sin) записываются строчными буквами, и их аргументы указываются в круглых скобках;

• если вводимое математическое выражение длиннее одной строки, то в этом случае часть выражения можно перенести на новую строку с помощью знака многоточия "...".

• символьные (тестовые выражения) записываются в апострофах;

4. Ввести два вектора v1=[1 3 5 8] и v2=[5 67 44 45]. Произвести произведение и деление векторов.

Формирование упорядоченных числовых последовательностей. Такие последовательности нужны для создания векторов или значений абсциссы при по - строении графиков. Для этого в MATLAB используется оператор:

<Начальное значение>: <Шаг>: <Конечное значение>.

5. Сформировать вектор V3 от 0 до 50 с шагом 3.

6. Сформировать таблицу M=[1 2 3: 4 5 6; 7 8 9]. Возможен ввод элементов матриц и векторов в виде арифметических выражений, содержащих любые доступные системе функции.

7. Ввести вектор V= [2+2/(3+4) exp(5) sqrt(lQ)].

Расчеты и визуализация результатов

1. Произвести вычисление выражений:

$$f1 = x^{z} - y + \frac{\cos z}{2};\ f2 = \frac{x + \sin y^{z}}{z^{3}};\ f3 = x^{z} + \sqrt{y + \ln z};\ f4 = \frac{|x - \sqrt{y}|}{z^{2}};$$

$$f5 = \frac{tg\,x + \sqrt{y}}{z^{3}};\ f6 = \frac{\sqrt{x + y^{3}}}{\ln z};\ f7 = \frac{\sqrt{e^{x} + y^{2}}}{z};\ f8 = \left|e^{x} + \frac{y^{2}}{z}\right|;$$

$$f9 = \log_3 z + x + \sqrt[3]{y};\ f10 = \frac{tg\,y + \arccos z}{x};\ f11 = \overline{|x - 1{,}2|};\ f12 = \frac{\lg x + \pi y}{z - 8{,}31}.$$

2. Выполнить построение двухмерных графиков. Для построения используется команда plot(f(x)). Аргументы функции и диапазон их изменения должен быть определен заранее. Графики MATLAB строит в отдельных окнах, называемых графическими окнами.

3. Произвести построение графиков следующих функций: f(x,y)=(x -y); f(x)=x -20-x+5; f(x)=x-cos(x); f(x)=x-sin(x); f(x)=x-sin(1/x)

4. Построить несколько графиков в одном графическом окне при помощи команды: plot(a1,f1,a2,f2,a3,f3), где al, a2, a3 - векторы аргументов функций, f1, f2, f3 - векторы значений функций, графики которых строятся в одном окне.

5. Построить графики с помощью команды fplott ('f(x)', [xmin xmax]).

6. Построить столбчатый график функции e x x с помощью команды bar(x,f(x));

7. Построить график с указанием погрешности каждой точки: x=-2:0.1:2; y=erf(x); e = rand(size(x))/10; errorbar(x,y,e);

8. Построить график дискретных отсчетов: x = 0:0.1:4; y = sin(xA2).*exp(-x); stem(x,y).

9. Построить график в полярных координатах: t=0:.01:2*pi; polar(t,abs(sin(2*t).*cos(2*t))).

10. Построить трехмерный график командой plot(x,y,z), где x,y,z - векторы. Ввести: [x,y] = meshgrid([-3:0.15:3]); z=x.A2+y.A2; plot3(x,y,z).

11. Постройте функцию z=x.A2+y.A2 при помощи команды mesh(z)

12. Построить график командой surf(z): z=peaks(25); surf(z); colormap(jet).

13. Построить графики функций:

$$G(a,b) = \begin{vmatrix} \lceil (5 + 2 \cdot \cos(a)) \cdot \cos(b) \rceil \\ (5 + 2 \cdot \cos(a)) \cdot \sin(b) \\ \lfloor \quad \quad \rfloor \end{vmatrix}; Z(x,y)=x^2+y^2-30.$$

Создание сценариев и функций

Программами в системе MATLAB являются m-файлы текстового формата, содержащие запись программ в виде программных кодов. Язык программирования системы MATLAB имеет следующие средства:

• данные различного типа;

• константы и переменные;

• операторы, включая операторы математических выражений;

• встроенные команды и функции;

• функции пользователя;

• управляющие структуры;

• системные операторы и функции;

• средства расширения языка.

Файл-сценарий, именуемый также Script-файлом, является просто записью серии команд без входных и выходных параметров.

Он имеет следующую структуру:

%основной комментарий;

%дополнительный комментарий; .

Комментарии - любой текст, служащий для пояснения команд, не влияющий на результаты расчетов и начинается всегда знаком %!

Важны следующие свойства файлов-сценариев:

• они не имеют входных и выходных аргументов;

• работают с данными из рабочей области;

• в процессе выполнения не компилируются;

• представляют собой зафиксированную в виде файла последовательность операций, полностью аналогичную той, что используется в сессии.

1. Создайте новый m-файл, через меню файл). Введите код построения графика:

```
%Plot with color red
   %Строит график синусоиды линией красного цвета
   %c выведенной масштабной сеткой в интервале [xmin.
   xmax] x=xmin:0.1:xmax;
   plot(x.sin(x).'r')
   grid on.
```

2. Сохраните файл-сценарий с именем на test_skript1. Запустите на выполнение данный файл.

М-файл-функция является типичным объектом языка программирования системы MATLAB. Одновременно он является полноценным модулем с точки зрения структурного программирования, поскольку содержит входные и выходные параметры и использует аппарат локальных переменных. Структура такого модуля с одним выходным параметром выглядит следующим образом:

```
function var=f_name(Список_параметров)
%Основной комментарий
%Дополнительный комментарий
Тело файла с любыми
выражениями var=выражение.
```

М-файл-функция имеет следующие свойства:

• он начинается с объявления function, после которого указывается имя переменной van - выходного параметра, имя самой функции и список ее входных параметров;

• функция возвращает свое значение и может использоваться в виде name (Список_параметров) в математических выражениях;

• все переменные, имеющиеся в теле файла-функции, являются локальными, т. е. действуют только в пределах тела функции;

• файл-функция является самостоятельным программным модулем, который общается с другими модулями через свои входные и выходные параметры;

• правила вывода комментариев те же, что у файлов-сценариев;

• файл-функция служит средством расширения системы MATLAB;

• при обнаружении файла-функции он компилируется и затем исполняется, а созданные машинные коды хранятся в рабочей области системы MATLAB.

Последняя конструкция var = выражение вводится, если требуется, чтобы функция возвращала результат вычислений.

3. Создайте новый m-файл, через меню файл(Ейе). Введите код М-файл функции:

```
function f1=f1()
R=input('Ввведите одну из цифр:1,2,3');
        if R==1 % A complex plot of f(z) = z^3
                % Three maxima at the cube roots of 1
           z=cplxgrid(20);
           cplxmap(z,z.^3);
        elseif R==2
           % A complex plot of f(z) = (z^4-1)^(1/4)
           % Four zeros at the fourth roots of
           1 z=cplxgrid(20); cplxmap(z,(z.^4-
           1).^(1/4));
        elseif R==3
           % A complex plot of f(z) = z^(1/3)
         % The Riemann surface for the cube root
         cplxroot(3,15);
        end
end.
```

4. Сохраните М-файл-функцию с именем на test_function1. Запустите на выполнение данный файл.

Контрольные вопросы

1. Структура интерфейса программы, назначение окон.

2. Принципы взаимодействия с другими программами.
3. Принципы ввода математических и текстовых выражений.
4. Работа с переменными и константами, форматы представления данных.
5. Основные приемы вычислений с таблицами и матрицами.
6. Графические возможности программы.
7. Принципы построения графиков.
8. Экспорт графических изображений.
9. Язык программирования MATLAB: назначение, особенности синтаксиса, ограничения.
10. Файлы, сценарии, функции, назначение, применение, отличительные черты.

ЛАБОРАТОРНАЯ РАБОТА №5. ПРОЕКТИРОВАНИЕ СИСТЕМ УПРАВЛЕНИЯ В СРЕДЕ MATLAB

Цель работы — познакомиться с пакетами расширений Fuzzy Logic Toolbox и Neural Networks Toolbox среды MATLAB; получить навыки построения систем управления на основе методов нечеткой логики; построить апроксиматоры на основе нейросетей.

Содержание лабораторной работы

1. Освоение интерфейса пакета расширений Fuzzy Logic Toolbox, определение возможностей и ограничений.

2. Создание системы управления светофорного объекта на основе нечеткой логики.

3. Ознакомление с приложениями MATLABa, построенных на пакете расширений Neural Networks Toolbox.

4. Построение апроксиматора на основе нейросетевой технологии.

Общие положения

Теоретические основы применения "нечеткой логикой" (Fuzzy logic)

Решение качественных или смешанных задач в настоящее время является наиболее сложным, т.к. необходимо решать задачи, не поддающиеся полностью или частично формализации для применения стандартной логики. Поэтому качественные задачи пока в основном решаются человеком, но есть уже и опыт решения их нейронными сетями или экспертными системами.

Управление техническими или иными объектами в большинстве своем являются задачами количественными. Аналитически управление объектом сводятся к решению одного ли системы дифференциальных уравнений.

Существуют специальные прикладные программы для моделирования динамических систем таких, как Visimm. При достаточно точном описании процесса системой уравнений ее целесообразно решать аналитическим или

численным методом. Эффективно такие вычисления производятся в среде MathCAD, MATLAB.

Однако зачастую аналитическое описание объекта (математическая модель) или недостаточно верно, или решение носит сложный характер. В этих случаях прибегают к методам моделирования, анализа, построенных на основе нейросетей, нечеткой логики, генетических алгоритмах, то есть методах искусственного интеллекта (ИИ).

Нечеткая логика возникла как наиболее удобный способ построения систем управления метрополитенами и сложными технологическими процессами, а также нашла применение в бытовой электронике, диагностических и других экспертных системах. Несмотря на то, что математический аппарат нечеткой логики впервые был разработан в США, активное развитие данного метода началось в Японии, и новая волна вновь достигла США и Европы.

Термин fuzzy (англ. нечеткий, размытый - произносится 'фаззи') является ключевым понятием. Нечеткая логика является многозначной логикой, что позволяет определить промежуточные значения для таких общепринятых оценок, как да|нет, истинно|ложно, черное|белое и т.п. Выражения подобные таким, как "слегка тепло" или "довольно холодно" становится возможно формулировать математически и обрабатывать на компьютерах. Нечеткая логика появилась в 1965 в работах Лотфи А. Задэ (Lotfi A. Zadeh), профессора технических наук Калифорнийского университета в Беркли.

Рассмотрим базовые понятия "нечеткой логики". Самым главным понятием систем, основанных на нечеткой логике, является понятие нечеткого (под)множества.

Нечеткое множество - это такое множество, которое образуется путем введения обобщенного понятия принадлежности, т.е. расширения двухэлементного множества значений функции принадлежности {0,1} до отрезка [0,1]. Это означает, что переход от полной принадлежности объекта

множеству к его полной непринадлежности происходит не скачком, как в обычных "четких" множествах, а плавно, постепенно, причем степень принадлежности элемента множеству выражается числом из интервала [0,1].

Таким образом, нечеткое множество $\bar{A} = \{(x, \mu A(x))\}$ определяется математически как совокупность упорядоченных пар, составленных из элементов x множества X и соответствующих им степеней принадлежности $\mu A(x)$ или непосредственно в виде функции $\mu A: X \rightarrow [0,1]$.

Функцией принадлежности (membership function) называется функция, которая позволяет вычислить степень принадлежности произвольного элемента универсального множества к нечеткому множеству. Графическое представление функции принадлежности называется термом.

Нечеткое число - это нечеткое подмножество универсального множества действительных чисел, имеющее нормальную и выпуклую функцию принадлежности, то есть такую, что а) существует такое значение носителя, в котором функция принадлежности равна единице, а также б) при отступлении от своего максимума влево или вправо функция принадлежности убывает.

Понятие нечеткого множества - это попытка математической формализации нечеткой информации для построения математических моделей. В основе этого понятия лежит представление о том, что составляющие данное множество элементы, обладающие общим свойством, могут обладать этим свойством в различной степени и, следовательно, принадлежать к данному множеству с различной степенью.

Из вышесказанного можно сделать следующие выводы:

1) нечеткие множества описывают неопределенные понятия (быстрый бегун, горячая вода, жаркая погода);

2) нечеткие множества допускают возможность частичной принадлежности к ним (пятница - частично выходной день (укороченный), погода скорее жаркая);

3) степень принадлежности объекта к нечеткому множеству определяется соответствующим значением функции принадлежности на интервале [0,1]; 18

4) функция принадлежности ставит в соответствие объекту (или логической переменной) значение степени его принадлежности к нечеткому множеству. Для наглядности приведем функцию принадлежности (терм) множества молодых людей.

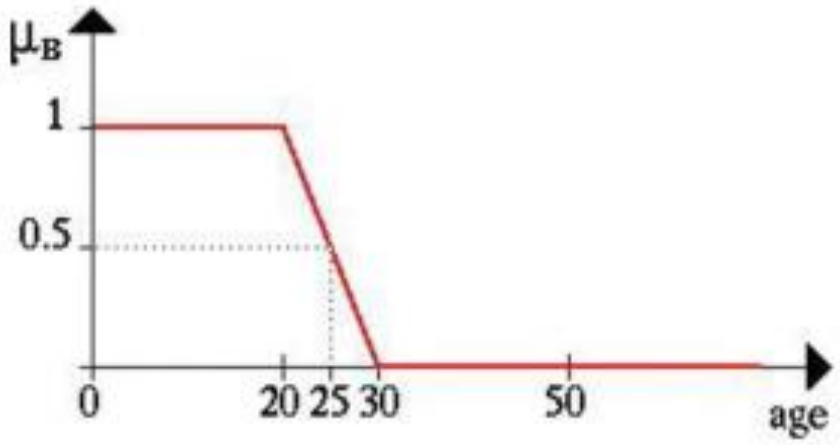


Рисунок 29 - Функция принадлежности $\mu_{\hat{A}}$ множества B

Из рисунка 29 видно, что 25-летние все еще молоды со степенью принадлежности 0,5. Определим базовые операции (действия) над нечеткими множествами (числами). Аналогично действиям с обычными множествами нам потребуется определить пересечение, объединение и отрицание нечетких множеств. В своей самой первой работе по нечетким множествам Л. А. Задэ предложил оператор минимума для пересечения и оператор максимума для объединения двух нечетких множеств. Легко видеть, что эти операторы совпадают с обычными (четкими) объединением и пересечением, только рассматриваются степени принадлежности 0 и 1. Чтобы пояснить это, приведем несколько примеров. Пусть A - нечеткое число (интервал) от 5 до 8 и B – нечеткое число около 4, как показано на рисунке 30.

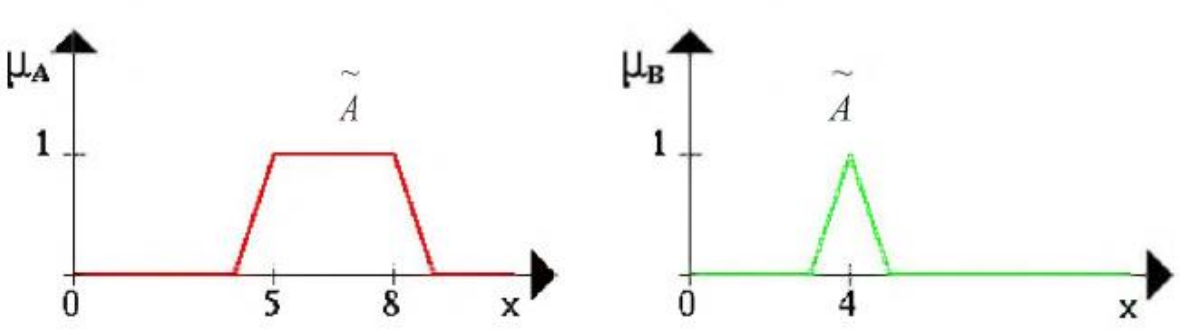


Рисунок 30 - Графическое представление двух нечетких чисел A и B

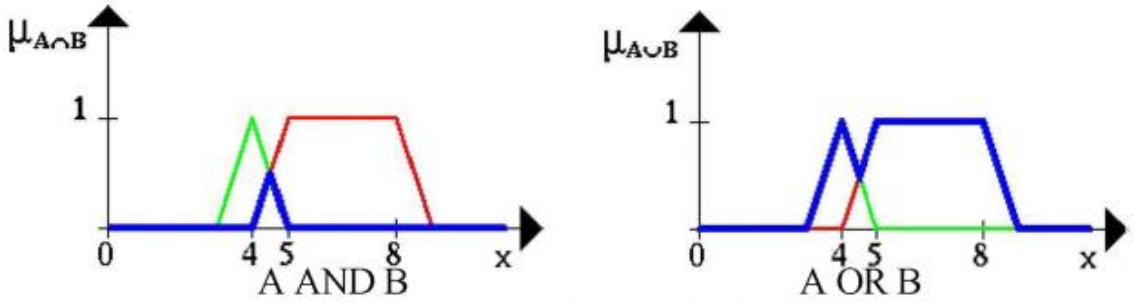


Рисунок 31 - Логические операции (AND, OR) с нечеткими числами A и B

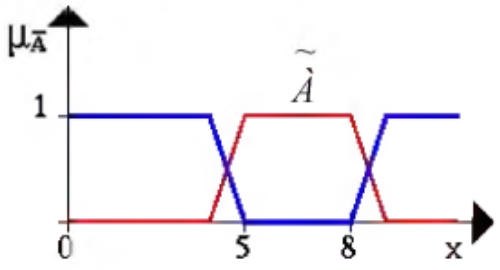


Рисунок 32 - Логическое отрицание (NOT) нечеткого числа A

Фаззификацией (fuzzification) называется процедура преобразование "четкой" точки $\overline{\tilde{O}} = (\tilde{o}_1,...\tilde{o}_n) \in \tilde{O}$ в нечеткое множество числа A из X. Другими словами нахождение значений функции принадлежности нечеткого числа в заданной точке. На этом этапе происходит установление соответствия между численным значением входной переменной и значением функции принадлежности соответствующего ей терма входной лингвистической переменной. Так, фаззификация числа 25 (рисунок 29) дает нам значение 0,5.

Дефаззификацией (defuzzification) называется процедура преобразования нечеткого множества в четкое число.

В теории нечетких множеств процедура дефаззификации аналогична нахождению характеристик положения (математического ожидания, моды, медианы) случайных величин в теории вероятности. Простейшим способом выполнения процедуры дефаззификации является выбор четкого числа, соответствующего максимуму функции принадлежности. Однако пригодность этого способа ограничивается лишь одноэкстремальными функциями принадлежности. Для многоэкстремальных функций принадлежности в Fuzzy Logic Toolbox запрограммированы такие методы дефаззификации:

• Centroid - центр тяжести;

• Bisector - медиана;

• LOM (Largest Of Maximums) - наибольший из максимумов;

• SOM (Smallest Of Maximums) - наименьший из максимумов;

• Mom (Mean Of Maximums) - центр максимумов.

Нечеткой базой знаний (fuzzy knowledge base) о влиянии факторов X=x1 .. xn на значение параметра Y называется совокупность логических высказываний типа:

ЕСЛИ (x1=μ1) И (x2=μ2),…И (xn=μn) ИЛИ (x1=μ1), ИЛИ (x2=μ2)…,ИЛИ (xk=μk),
ТО yi=μj,
где μ1…μn- нечеткий терм, которым оцениваются входные переменные x1... xn;
μj - нечеткий терм, которым оцениваются выходные переменные y1... yn.

Графически систему на основе нечеткой логики можно представить в виде блоков (рисунок 33). Рассмотрим разработку подобных систем в среде MATLAB. MATLAB - матричная лаборатория - наиболее развитая система программирования для научно-технических расчетов, дополненная к настоящему времени несколькими десятками более частных приложений, относящихся к 20 вычислительной математике, обработке информации,

конструированию электронных приборов, экономике и ряду других разделов прикладной науки.

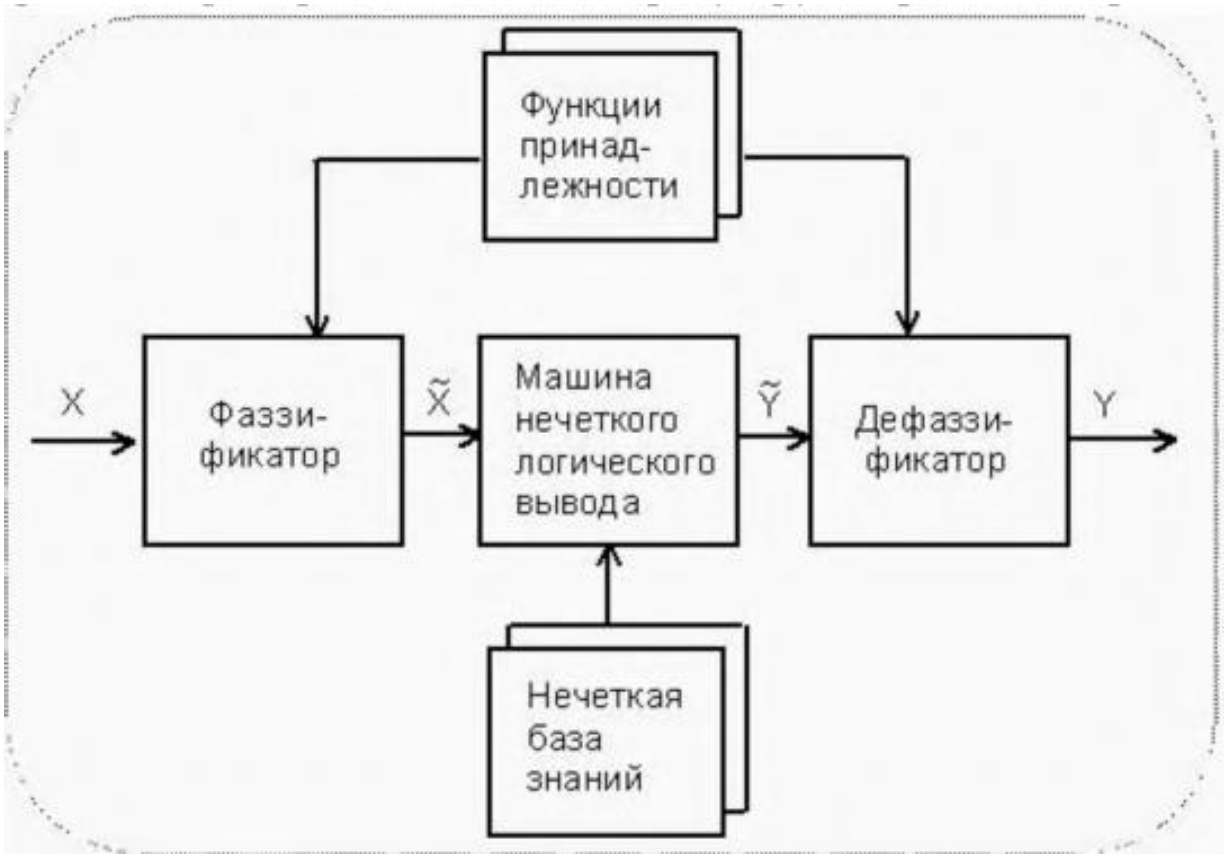


Рисунок 33 - Структурная схема системы, использующей "нечеткую логику" X - входной четкий вектор; O - вектор нечетких множеств, соответствующий входному вектору X; Y - результат логического вывода в виде вектора нечетких множеств; Y - выходной четкий вектор.

MATLAB - система программирования высокого уровня, работающая как интерпретатор и включающая большой набор инструкций (команд) для выполнения самых разнообразных вычислений, задания структур данных и графического представления информации. Команды эти разбиты на тематические группы, расположенные в различных директориях системы.

Fuzzy Logic Toolbox - это пакет прикладных программ, входящих в состав среды MATLAB. Он позволяет создавать системы нечеткого логического вывода и нечеткой классификации в рамках среды MATLAB, с возможностью их интегрирования в Simulink.

Базовым понятием Fuzzy Logic Toolbox является FIS-структура - система нечеткого вывода (Fuzzy Inference System). FIS-структура содержит все необходимые данные для реализации функционального отображения “входы-выходы” на основе нечеткого логического вывода согласно схеме, приведенной на рисунке 33.

Fuzzy Logic Toolbox содержит следующие категории программных инструментов:

• функции;

• интерактивные модули с графическим пользовательским интерфейсом (с GUI);

• блоки для пакета Simulink;

• демонстрационные примеры.

Модуль fuzzy позволяет строить нечеткие системы двух типов - Мамдани и Сугэно. В системах типа Мамдани база знаний состоит из правил вида “Если 21 x1=низкий и x2=средний, то y=высокий”. В системах типа Сугэно база знаний состоит из правил вида “Если x1=низкий и x2=средний, то y=a0+a1x1+a2x2". Таким образом, основное отличие между системами Мамдани и Сугэно заключается в разных способах задания значений выходной переменной в правилах, образующих базу знаний. В системах типа Мамдани значения выходной переменной задаются нечеткими термами, в системах типа Сугэно - как линейная комбинация входных переменных.

Теоретические основы нейросетевых технологий

Искусственные нейронные сети индуцированы биологией, так как они состоят из элементов, функциональные возможности которых аналогичны большинству элементарных функций биологического нейрона. Эти элементы затем организуются по способу, который может соответствовать (или не соответствовать) анатомии мозга. Несмотря на такое поверхностное сходство, искусственные нейронные сети демонстрируют удивительное число свойств,

присущих мозгу. Например, они обучаются на основе опыта, обобщают знание и извлекают существенные свойства из поступающей информации.

Искусственные нейронные сети могут менять свое поведение в зависимости от внешней среды. Этот фактор в большей степени, чем любой другой, ответствен за тот интерес, который они вызывают. После предъявления входных сигналов (возможно, вместе с требуемыми выходами) они самонастраиваются, чтобы обеспечивать требуемую реакцию.

Искусственный нейрон имитирует в первом приближении свойства биологического нейрона. На вход искусственного нейрона поступает некоторое множество сигналов, каждый из которых является выходом другого нейрона. Каждый вход умножается на соответствующий вес, аналогичный синаптической силе, и все произведения суммируются, определяя уровень активации нейрона.

На рисунке 34 представлена модель, реализующая эту идею. Хотя сетевые парадигмы весьма разнообразны, в основе почти всех их лежит эта конфигурация. Здесь множество входных сигналов, обозначенных x1, x2,..., xn, поступает на искусственный нейрон. Эти входные сигналы, в совокупности обозначаемые вектором X, соответствуют сигналам, приходящим в синапсы биологического нейрона. Каждый сигнал умножается на соответствующий вес wh w2,..., wn, и поступает на суммирующий блок, обозначенный E. Каждый вес соответствует «силе» одной биологической синаптической связи, (множество весов в совокупности обозначается вектором W). Суммирующий блок, соответствующий телу биологического элемента, складывает взвешенные входы алгебраически, создавая выход, который мы будем называть NET. В векторных обозначениях это может быть компактно записано следующим образом:

NET = XW.

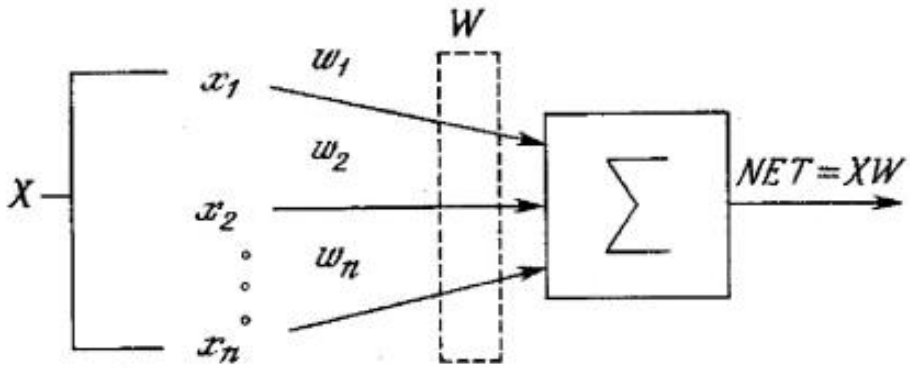


Рисунок 34 - Искусственный нейрон

Сигнал NET далее, как правило, преобразуется активационной функцией F и дает выходной нейронный сигнал OUT. Активационная функция может быть обычной линейной функцией

$$OUT = K(NET),$$

где К - постоянная, пороговой функции;

OUT = 1, если NET > T,

OUT = 0 в остальных случаях,

где Т - некоторая постоянная пороговая величина, или же функцией, более точно моделирующей нелинейную передаточную характеристику биологического нейрона и представляющей нейронной сети большие возможности (рис. 35).

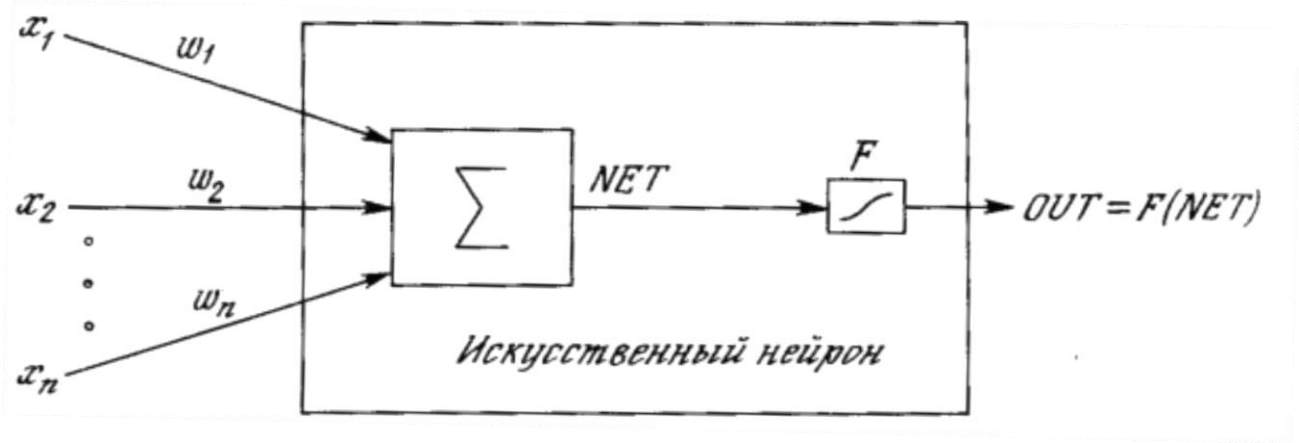


Рисунок 35 – Искусственный нейрон с активационной функцией

На рисунке 35 блок, обозначенный F, принимает сигнал NET и выдает сигнал OUT. Если блок F сужает диапазон изменения величины NET так, что при любых значениях NET значения OUT принадлежат некоторому конечному интервалу, то F называется «сжимающей» функцией. В качестве «сжимающей» функции часто используется логистическая или «сигмоидальная» (S-образная) функция. Эта функция математически выражается как $F(x) = 1/(1 + e^{-x})$. Таким образом,

$$OUT = \frac{1}{1+e^{-NET}}$$

По аналогии с электронными системами активационную функцию можно считать нелинейной усилительной характеристикой искусственного нейрона. Коэффициент усиления вычисляется как отношение приращения величины OUT к вызвавшему его небольшому приращению величины NET.

Признаки, при наличии которых задачу целесообразно решать с применением нейросетевых технологий:

• отсутствие алгоритма и математической модели решения задачи;

• имеются данные экспериментальных исследований;

• проблема характеризуется большими объемами входной информации, причем данные носят вероятностный или нечеткий характер, зашумлены, искажены или противоречивы.

Любая нейронная сеть может быть представлена в виде схемы, (рис. 36).

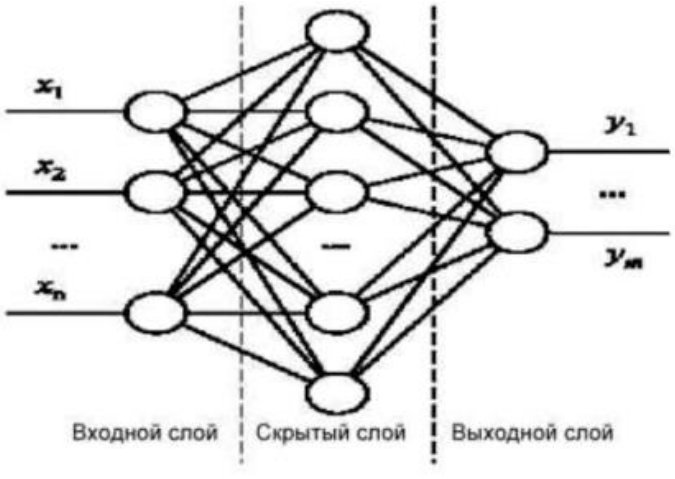


Рисунок 36 - Структура трехслойной нейронной сети

После построения сети требуется ее обучение. Схема обучения представлена на рисунке 37.

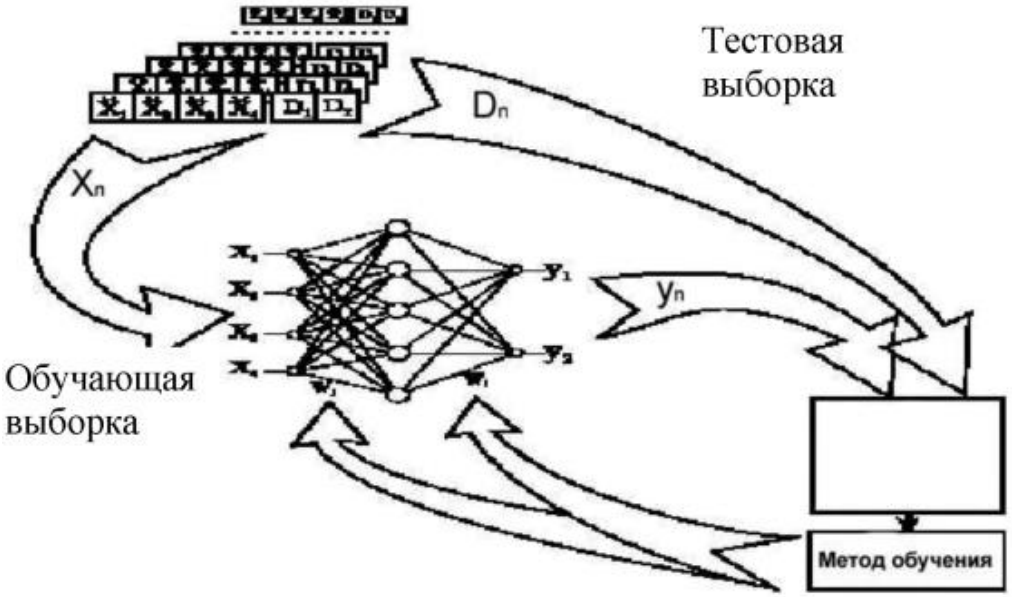


Рисунок 37 - Структура обучения нейронной сети

Методы (алгоритмы) обучения НС делятся на два класса:

• обучение с учителем - сети предъявляются значения как входных, так и желательных выходных сигналов, и она по некоторому внутреннему алгоритму подстраивает веса своих синаптических связей.

• обучение без учителя - выходы НС формируются самостоятельно, а веса изменяются по алгоритму, учитывающему только входные и производные от них сигналы. После обучения сеть готова к работе, то есть решения задач классификации, аппроксимации и др.

Выполнение лабораторной работы

1. Создания системы управления светофорного объекта.

Обычно для таких задач выбирают систему типа Мамдани.

Постановка задачи. Система управления должна регулировать время работы светофора в режиме зеленого света в зависимости от количества подъезжающих к перекрестку машин (рисунок 38).

Решение задачи. Для работы нечеткого светофора на перекрестке улиц Север-Юг (СЮ) и Запад-Восток (ЗВ) необходимо установить 8 датчиков (рисунок 38), которые считают проехавшие мимо них машины.

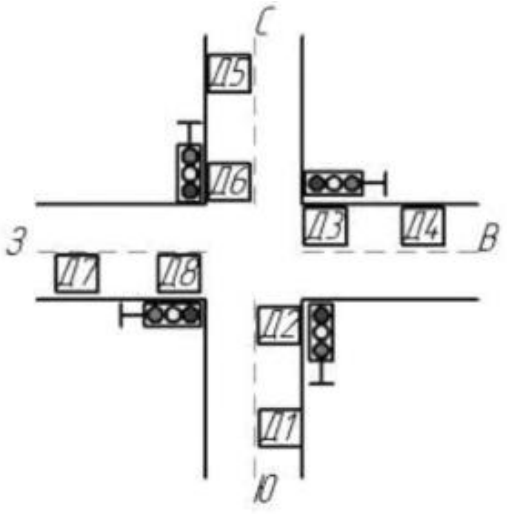


Рисунок 38 - Расположение датчиков на перекрестке Светофор использует разности показаний четырех пар датчиков: (Д1-Д2), (Д3-Д4), (Д5-Д6) и (Д7-Д8).

Таким образом, если для улицы СЮ горит зеленый свет, машины проезжают перекресток и показания двух пар датчиков равны: Д1=Д2, Д5=Д6, а, следовательно, их разность равна нулю. В это же время на улице ЗВ перед светофором останавливаются машины, которые успели проехать только Д4 и Д7.

В результате можно рассчитать суммарное количество автомобилей на этой улице следующим образом: (Д4-Д3)+(Д7-Д8)=(Д4- 0)+(Д7-0)=Д4+Д7.

Поскольку работа светофора зависит от числа машин на обеих улицах и текущего времени зеленого света, для нашей системы предлагается использовать 3 входа:

• число машин на улице СЮ по окончании очередного цикла (интенсивность СЮ);

• число машин на улице ЗВ по окончании цикла (интенсивность ЗВ);

• время зеленого света нечеткого светофора, сек. (фаза светофора).

Шаг1. Для загрузки основного fis-редактора напечатаем слова fuzzy в командной строке MATLAB. После этого откроется нового графическое окно, по-казанное на рисунке 39.

Шаг 2. Добавим вторую и третью входную переменную. Для этого в меню Edit выбираем команду Add input.

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

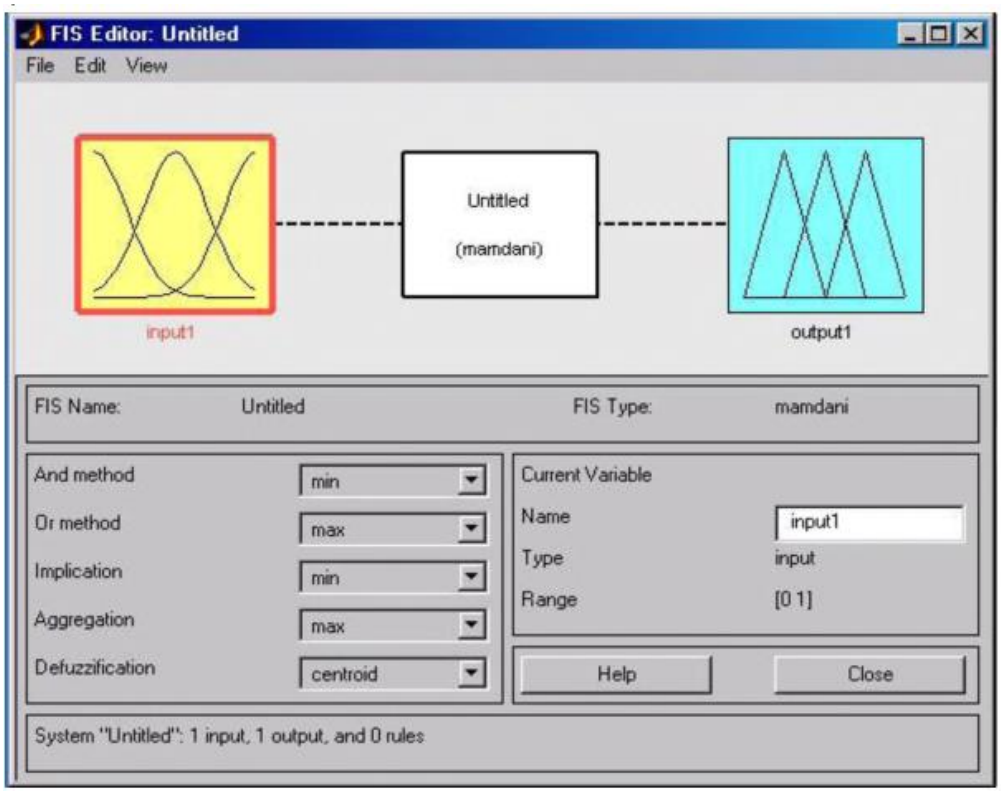


Рисунок 39 - Окно редактора FIS-Editor

Шаг 4. Переименуем остальные входные переменные. Для этого сделаем один щелчок левой кнопкой мыши на блоке input2, введем новое обозначение интенсивностью в поле редактирования имени текущей переменной и нажмем . Аналогично переименовываем третью переменную.

Шаг 5. Переименуем выходную переменную. Для этого сделаем один щелчок левой кнопкой мыши на блоке output 1, введем новое обозначение изменить-фазу в поле редактирования имени текущей переменной и нажмем.

Шаг 6. Зададим имя системы. Для этого в меню File выбираем в подменю Export команду To disk и вводим имя файла, например, svetofor.

Шаг 7. Перейдем в редактор функций принадлежности. Для этого сделаем двойной щелчок левой кнопкой мыши на блоке фаза-светофора.

Шаг 8. Зададим диапазон изменения переменной фаза-светофора. Для этого напечатаем 0 50 в поле Range (рисунок 40).

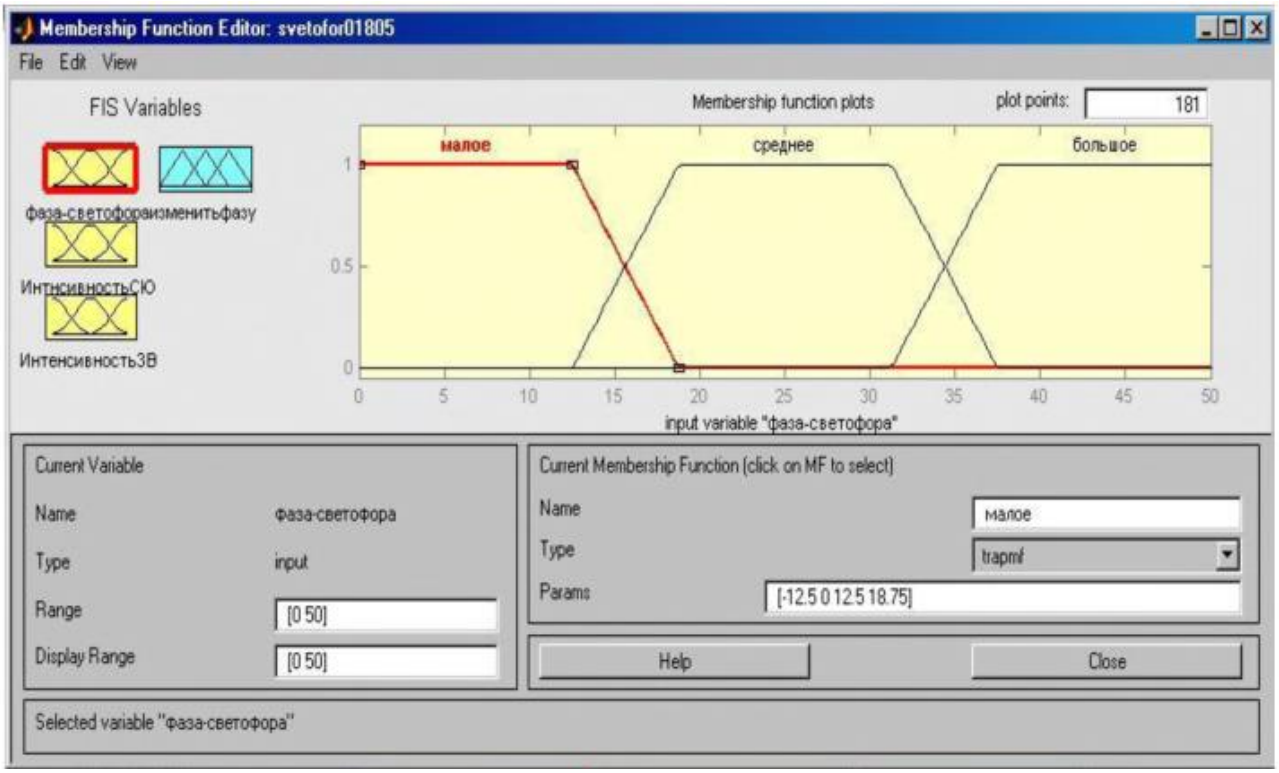


Рисунок 40 - Функция принадлежности фазы зеленого сигнала светофора

Шаг 9. Зададим функции принадлежности переменной фаза-светофора. Для лингвистической оценки этой переменной будем использовать 3 терма с трапециидальными функциями принадлежности. Для этого в меню Edit выберем команду Add MFs... В результате появится диалоговое окно выбора

типа и количества функций принадлежностей. По умолчанию, это 3 терма с треугольными функциями принадлежности. Поэтому просто нажимаем.

Шаг 10. Зададим наименования термов переменной фаза-светофора. Для этого делаем один щелчок левой кнопкой мыши по графику первой функции принадлежности (рисунок 40). Затем вводим наименование терма, например, 26 малое, в поле Name и нажмем. Затем делаем один щелчок левой кнопкой мыши по графику второй функции принадлежности и вводим наименование терма, например, Среднее, в поле Name и нажмем . Еще раз делаем один щелчок левой кнопкой мыши по графику третьей функции принадлежности и вводим наименование терма, например, Высокое, в поле Name и нажмем . В результате получим графическое окно, изображенное на рисунке 40.

Шаг 11. Зададим функции принадлежности переменной интенсивностью. Для лингвистической оценки этой переменной будем использовать 5 термов с трапециидальными функциями принадлежности. Для этого активизируем переменную интенсивностью с помощью щелчка левой кнопки мыши на блоке интенсивностью. Зададим диапазон изменения переменной интенсивностью. Для этого напечатаем 0 90 в поле Range (рисунке 41) и нажмем. Затем в меню Edit выберем команду Add MFs.... В появившимся диалоговом окне выбираем тип функции принадлежности trapmf в поле MF type и 5 термов в поле Number of MFs. После этого нажимаем.

Шаг 12. По аналогии с шагом 10 зададим следующие наименования термов переменной интенсивностьЗВ: Очень малое, малое, среднее, большое, очень большое.

Шаг 13. Зададим функции принадлежности переменной изменитьфазу. Для лингвистической оценки этой переменной будем использовать 3 терма с гауссовкой функциями принадлежности. Для этого активизируем переменную изменитьфазу с помощью щелчка левой кнопки мыши на блоке изменитьфазу. Зададим диапазон изменения переменной изменитьфазу. Для этого напечатаем -20 20 в поле Range (рис. 42) и нажмем. Затем в меню Edit выберем команду Add

MFs.... В появившимся диалоговом окне выбираем 3 термов в поле Number of MFs.

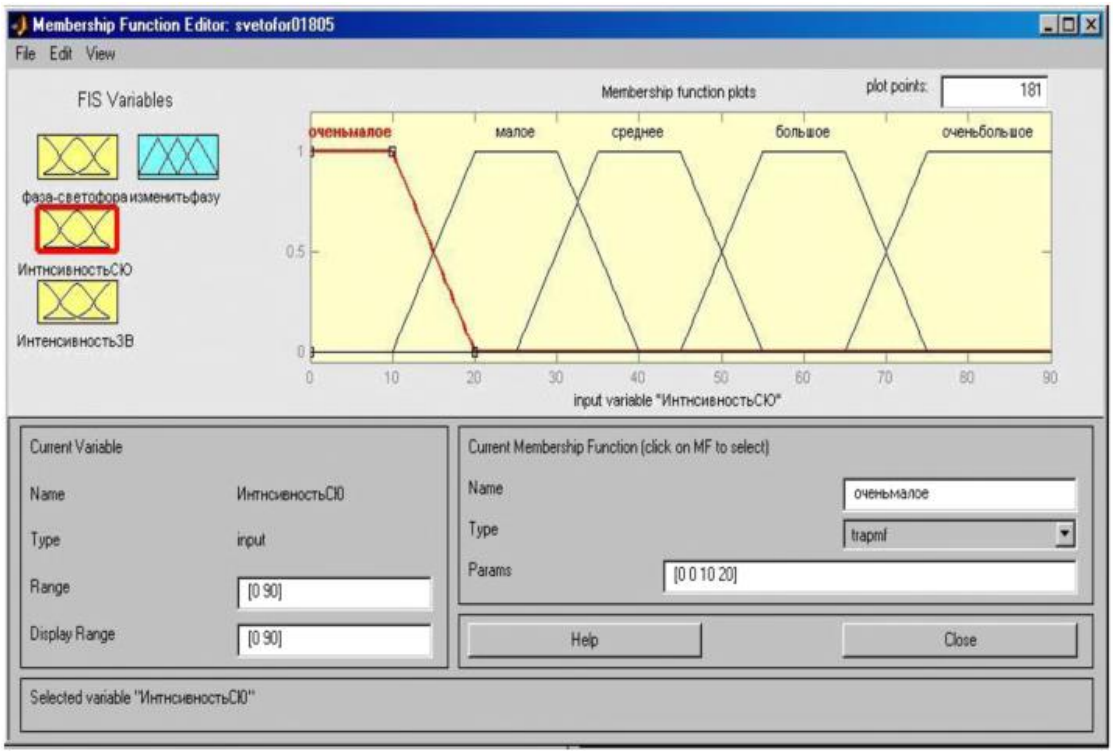


Рисунок 41 - Функция принадлежности интенсивности движения по направлению СЮ

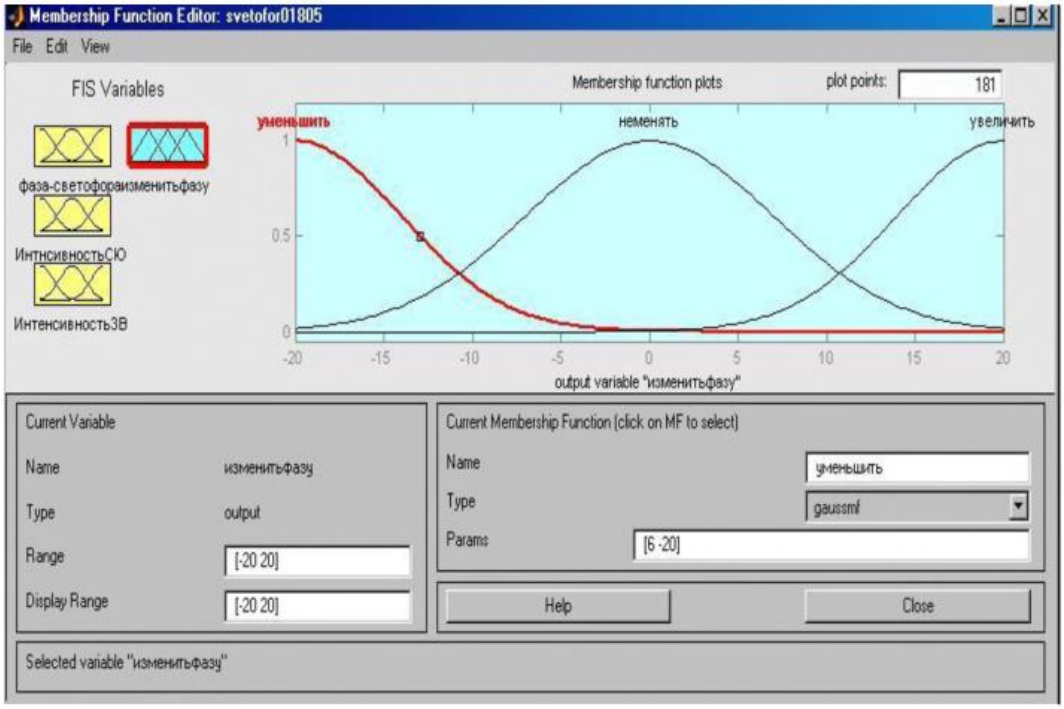


Рисунок 42 - Функция принадлежности выхода системы (изменение фазы зеленого сигнала светофора, сек.)

Шаг 14. По аналогии с шагом 10 зададим следующие наименования термов переменной y: уменьшить, не менять, Увеличить. В результате получим графическое окно, изображенное на рисунке 42.

Шаг 15. Перейдем в редактор базы знаний RuleEditor. Для этого выберем в меню Edit, команду Edit rules....

Шаг 16. На основе теоретических и экспериментальных данных о работе светофорных объектов сформулируем правила работы системы: Если время зеленого света на улице СЮ=большое, & число машин на улице СЮ=малое, & число машин на улице ЗВ=большое, ТО время зеленого света = уменьшить.

Аналогично формулируем остальные правила. Число правил должно быть достаточным для эффективной работы системы в заданных условиях.

Для нашей задачи это количество составляет 50 правил (рисунок 43). Для ввода правила необходимо выбрать в меню соответствующую комбинацию термов и нажать кнопку Add rule. На рисунке 43 изображено окно редактора базы знаний после ввода всех девяти правил. Число, приведенное в скобках в конце каждого правила представляет собой весовым коэффициент соответствующего правила.

Шаг 17. Сохраним созданную систему. Для этого в меню File выбираем в подменю Export команду To disk.

На рисунке 44 приведено окно визуализации нечеткого логического вывода. Это окно активизируется командой View rules... меню View. В поле Input указываются значения входных переменных, для которых выполняется логический вывод.

На рисунке 45 приведена поверхность “входы-выход”, соответствующая синтезированной нечеткой системе. Для вывода этого окна необходимо использовать команду View surface... меню View.

Ознакомление с приложениями MATLABa построенных на пакете расширений Neural Networks Toolbox

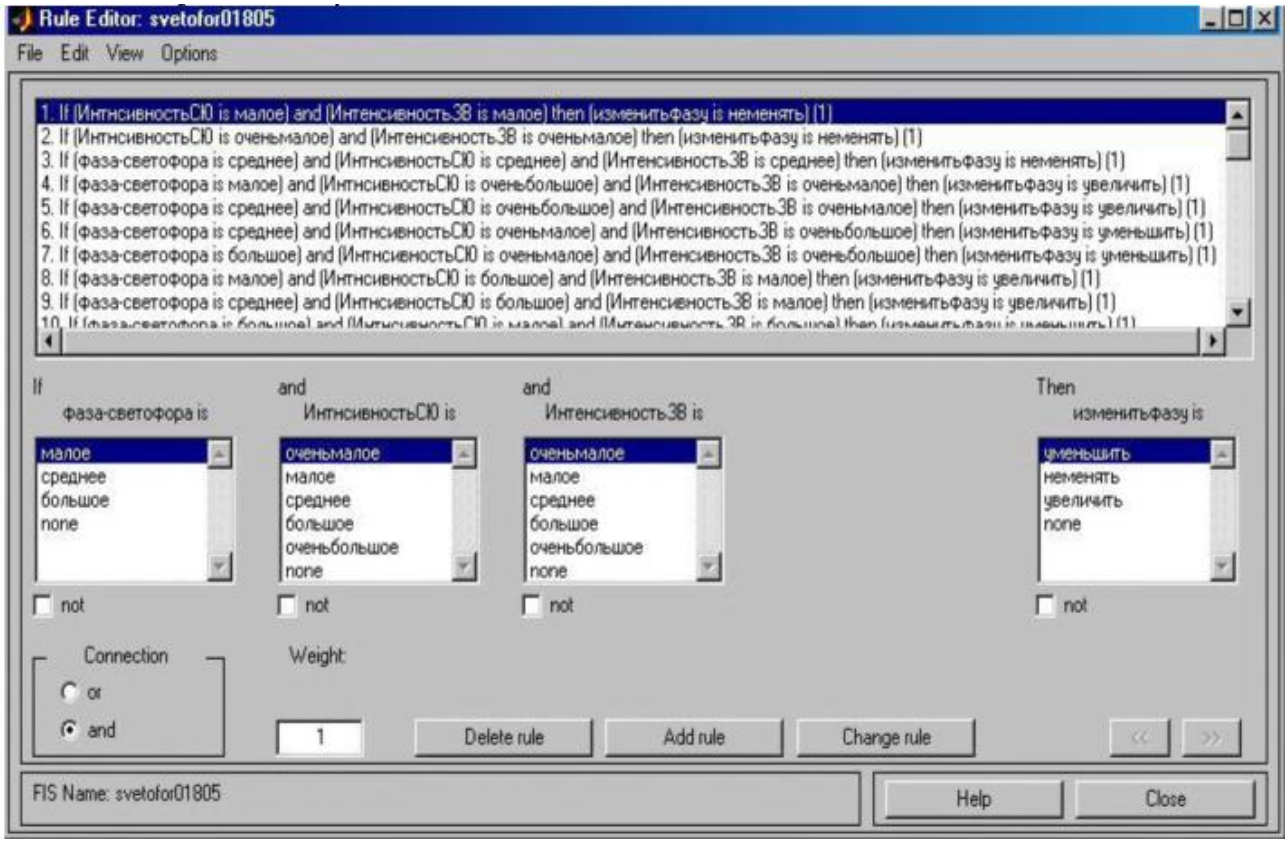


Рисунок 43 - Редактор базы логических правил

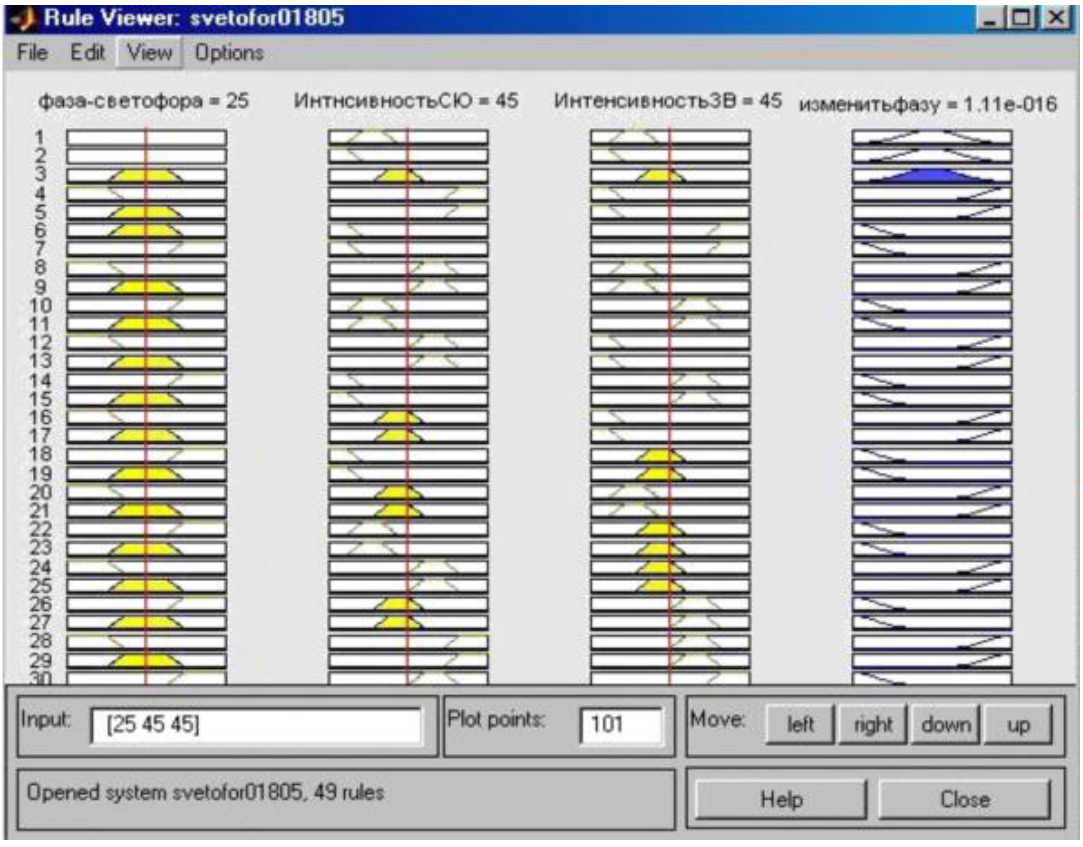


Рисунок 44 - Просмотр логических правил в браузере FIS редакторе

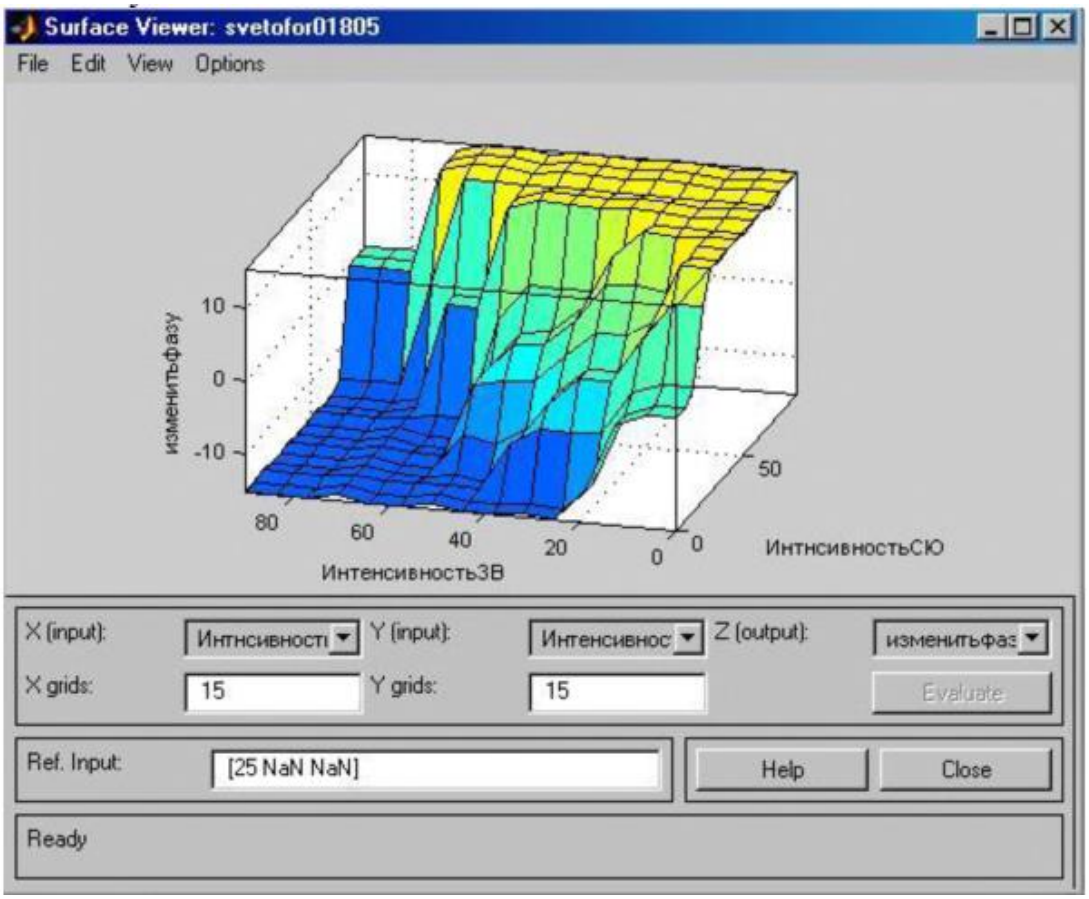


Рисунок 45 - Характеристика нечеткого регулятора по параметрам интенсивность движения по направлениям СЮ - ВЗ 30

Ознакомьтесь с демонстрационными примерами использования пакета Neural Networks Toolbox. В окне Редактор(ЕШ1ог) выберите в списке демонстрации, затем инструменты (instrument), далее Neural Networks Toolbox.

Построение нейросетевого апроксиматора

1. Создайте новый m-файл, через меню файл(File). Введите код нейросети по аппроксимации функции:

```
%определение функций
time = 1:0.01:2.5;
X = sin(sin(time).*time*10);
P = con2seq(X);
T = con2seq(2*[0 X(1:(end-1))] + X);
```

```
%графическое представление исходных сигналов
plot(time,cat(2,P{:}),time,cat(2,T {:}),'--');
title('Input and Target Signals');
xlabel('Time');
legend({'Input','T arget'});
%создание нейросети
net = newlin([-3 3],1,[0 1],0.1);
%обучение сети и вывод результата
[net,Y,E,Pf]=adapt(net,P,T);
plot(time,cat(2,Y{:}),'b', ...
time,cat(2,T{:}),'r', ...
time,cat(2,E{:}),'g',[1 2.5],[0 0],'k');
legend({'Output','Target','Error'});
%обучение сети и вывод результата
[net,Y,E,Pf]=adapt(net,P,T);
plot(time,cat(2,Y{:}),'b', ...
time,cat(2,T{:}),'r', ...
time,cat(2,E{:}),'g',[1 2.5],[0 0],'k');
legend({'Output','Target','Error'});
```

2. Сохраните файл-сценарий с именем на test_NN1. Запустите на выполнение данный файл.

Контрольные вопросы

1. Структура интерфейса пакета Fuzzy Logic Toolbox.

2. Принципы построения систем моделирования и экспертных систем на основе нечеткой логике в пакете Fuzzy Logic Toolbox.

3. Этапы построения нечетких регуляторов.

4. Принципы построения базы правил нечетких систем.

5. Принципы построения систем на базе нейросетей.

6. Структура, виды нейросетей, сферы назначения.

7. Прядок построения нейросетевого апроксиматора.

ЛАБОРАТОРНАЯ РАБОТА № 6. ПРОЕКТИРОВАНИЕ СИСТЕМ УПРАВЛЕНИЯ НА ОСНОВЕ ГИБРИДНЫХ НЕЙРОНЕЧЕТКИХ СИСТЕМ В СРЕДЕ MATLAB

Цель работы - познакомиться с пакетами расширений Adaptive Neuro-Fuzzy Inference System ANFIS (адаптивная нейро-нечеткая система) среды MATLAB; получить навыки построения систем прогнозирования на основе нейронечетких систем.

Содержание лабораторной работы

1 Освоение интерфейса пакета расширений ANFIS, определение возможностей и ограничений.

2 Ознакомление с приложениями MATLAB, построенными в пакете расширений ANFIS.

3 Создание системы прогнозирования на основе ANFIS.

Общие положения

Основным компонентом рассмотренных ранее средств системы MATLAB в рамках пакета Fuzzy Logic Toolbox является нечеткая база (знаний) правил, которая занимает центральное место в процедурах нечеткого вывода. В то же время существуют целые классы прикладных задач, в которых выявление и построение правил невозможно или связано с серьезными трудностями концептуального характера. К таким задачам относятся задачи распознавания образов, экстраполяции и интерполяции функциональных зависимостей, классификации и прогнозирования, нелинейного и ситуационного управления, а также интеллектуального анализа данных (Data Mining).

Общей особенностью подобных задач является существование некоторой зависимости или отношения, связывающего входные и выходные переменные модели системы, представляемой в форме так называемого "черного ящика".

При этом выявление и определение данной зависимости в явном теоретикомножественном или аналитическом виде не представляется возможным либо по причине недостатка информации о моделируемой проблемной области, либо сложности учета многообразия факторов, оказывающих влияние на характер данной взаимосвязи.

Для конструктивного решения подобных задач разработан специальный математический аппарат, получивший название нейронных сетей. Достоинством моделей, построенных на основе нейронных сетей, является возможность получения новой информации о проблемной области в форме некоторого прогноза. При этом построение и настройка нейронных сетей осуществляется посредством их обучения на основе имеющейся и доступной информации.

Недостатком нейронных сетей является представление знаний о проблемной области в специальном виде, которое может существенно отличаться от возможной содержательной интерпретации существующих взаимосвязей и отношений.

Нечеткие нейронные сети или гибридные сети по замыслу их разработчиков призваны объединить в себе достоинства нейронных сетей и систем нечеткого вывода. С одной стороны, они позволяют разрабатывать и представлять модели систем в форме правил нечетких продукций, которые обладают наглядностью и простотой содержательной интерпретации. С другой стороны, для по - строения правил нечетких продукций используются методы нейронных сетей, что является более удобным и менее трудоемким процессом для системных аналитиков. В последнее время аппарат гибридных сетей повсеместно признается специалистами как один из наиболее перспективных для решения слабо или плохо структурированных задач прикладного системного анализа.

Система MATLAB имеет в своем составе ANFIS-редактор. ANFIS является аббревиатурой Adaptive Neuro-Fuzzy Inference System - (адаптивная

нейронечеткая система). ANFIS-редактор позволяет автоматически синтезировать из экспериментальных данных нейро-нечеткие сети. Нейро-нечеткую сеть можно рассматривать как одну из разновидностей систем нечеткого логического вывода типа Сугэно. При этом функции принадлежности синтезированных систем настроены (обучены) так, чтобы минимизировать отклонения между результатами нечеткого моделирования и экспериментальными данными. Применяя эту систему можно эффективно разрабатывать системы прогнозирования. Внешний вид ANFIS-редактора представлен на рисунке 46.

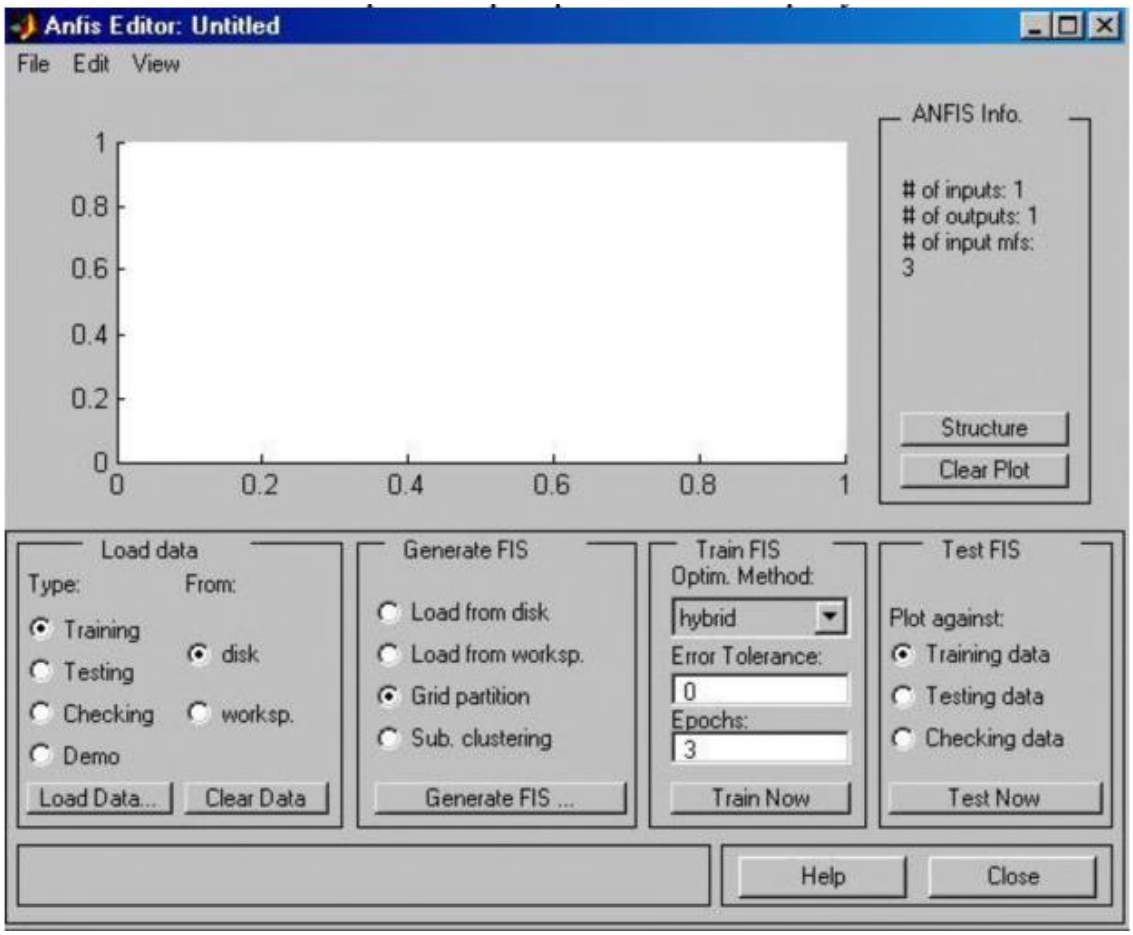


Рисунок 46 - ANFIS-редактор в среде MATLAB

Выполнение лабораторной работы

Освоение интерфейса ANFIS-редактора.

1. Запустить MATLAB.

2. Загрузить ANFIS-редактор с помощью команды anfisedit. В результате выполнения этой команды появится графическое окно, изображенное на рисунке 46.

3. Изучить меню и области интерфейса редактора ANFIS.

Ознакомление с демонстрационными примерами MATLAB, построенными в пакете расширений Fuzzy Logic

1. Ознакомьтесь с демонстрационными примерами использования пакета Fuzzy Logic. В окне Редактор (Editor) выберите в списке демонстрации, затем инструменты (instrument), далее Fuzzy Logic.

Проектирование с системы прогнозирования

1. Загрузить ANFIS-редактор с помощью команды anfisedit. Проектирование систем прогнозирования производится в несколько этапов:

• загрузка данных с диска (обучающая выборка);

• генерирование структуры нечеткой системы (FIS);

• обучение нейросети системы;

• тестирование и дообучение системы прогнозирования.

2. Загрузить обучающую выборку через область загрузка данных(Load data), файл anfis1.dat.

3. Сохранить созданную систему. Для этого в меню File выбираем в подменю Export команду To disk.

4. Произвести генерирование нечеткой (FIS) системы через область генерация нечеткой структуры (Generate FIS). Необходимо выбрать число термов входных переменных, их вид, а также тип выходной переменной.

5. Произвести обучение нейросети гибридной системы, при этом нужно выбрать метод обучения, ошибку обучения и количество эпох при обучении. Первые два параметра подбираются по результатам обучения сети, а количество эпох должно быть достаточными для обучения сети с заданными параметрам. Следует начинать с 50 эпох. В дальнейшем увеличивают количество эпох до величины, при которой ошибка обучения не изменяется.

6. Просмотреть построенную структуру системы, используя область свойств системы^О ТК info).

7. Изучить правила нечеткой системы прогнозирования и кривую прогноза через меню (edit и view).

8. Произвести анализ полученной системы и сделать выводы.

Контрольные вопросы

1. Гибридные нейро-нечеткие системы, назначение, сферы применения.

2. Структура интерфейса Adaptive Neuro-Fuzzy Inference System, Fuzzy Logic Toolbox.

3. Принципы построения систем прогнозирования в пакете Adaptive Neuro - Fuzzy Inference System, Fuzzy Logic Toolbox.

4. Этапы построения гибридных систем и их тестирование.

Список литературы

1. www.matlab.ru [Электронный ресурс]: Консультационный Центр MATLAB - Режим доступа: http//www.matlab.ru, свободный. - Загл. с экрана. - Яз. русский, англ.

2. Барский А.Б. Введение в нейронные сети / А.Б. Барский – Электрон. текстовые данные.– М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.– 358 с.

3. Борисов В.В. Нечеткие модели и сети: монография/ В.В. Борисов, В.В. Круглов, А.С. Федулов – Электрон. текстовые данные.– М.: Горячая линия - Телеком, 2012.– 284 с.

4. Галушкин А.И. Нейронные сети. Основы теории [Электронный ресурс]: монография/ Галушкин А.И.– Электрон. текстовые данные.– М.: Горячая линия - Телеком, 2012.– 496 с.

5. Дьяконов В. MATHCAD 8/2000: Специальный справочник - СПб.: Питер, 2001. - 592 с.

6. Дьяконов В. MATLAB с пакетами расширений / В. Дьяконов, И. Абраменкова, В. Круглов // под ред. Проф. В.П. Дьяконова. - М.: Нолидж. - 2001. - 880с.

7. Дьяконов В. Математические пакеты расширения MATLAB: Специальный справочник / В. Дьяконов, В. Круглов //. - СПб.: Питер, 2001. - 480 с.

8. Заде Л. Понятие лингвистической переменной и его применение для принятия приближенных решений /Л. Заде //. М.: Мир, 1976. -165с.

9. Интеллектуальные системы: методические указания к лабораторным работам для студентов бакалавриата, обучающихся по направлению подготовки 01.03.04 «Прикладная математика»/ – Электрон. текстовые данные.– М.: Московский государственный строительный университет, Ай Пи Эр Медиа, ЭБС АСВ, 2015.– 57 с.

10. Интеллектуальные системы: учебное пособие/ А.М. Семенов [и др.].– Электрон. текстовые данные.– Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2013.– 236 с.

11. Круглов В.В. Нечёткая логика и искусственные нейронные сети / В.В. Круглов, М.И. Дли, Р.Ю. Голунов. – М.: ФИЗМАТЛИТ – 2001.- 221 с.

12. Круглов В.В. Искусственные нейронные сети. Теория и практика / В.В. Круглов, В.В. Борисов. – М.: Горячая линия – Телеком, 2002. – 382 с.

13. Леоненков А.В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. - СПб.: БХВ - Питербург, 2003. - 736 с.

14. Макаров Е.Г. Инженерные расчеты в MathCAD: Учебный курс. - СПб.: Питер, 2003. - 448 с.

15. Макеева А.В. Основы нечеткой логики. Учебное пособие для Вузов. – Н. Новгород: ВГИПУ, 2009. – 59 с.

16. Нечеткие множества в моделях управления и искусственного интеллекта / Под ред. Д.А. Поспелова. - М.: Наука, 1986. -312 с.

17. Пегат А. Нечеткое моделирование и управление / А. Пегат ; пер. с англ. — М. : БИНОМ. Лаборатория знаний, 2009. — 798 с

18. Потемкин В.Г. Вычисления в среде MATLAB.- М.: Диалог - МИФИ, 2004. - 720 с.

19. Ротштейн А. П. Интеллектуальные технологии идентификации: нечёткая логика, генетические алгоритмы, нейронные сети / А. П. Ротштейн. – Винница : УНИВЕРСУМ-Винница, 1999. – 320 с.

20. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы/ Рутковская Д., Пилиньский М., Рутковский Л.– Электрон. текстовые данные.– М.: Горячая линия - Телеком, 2013.– 384 с.

21. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И. Д. Рудинского. - М.: Горячая линия - Телеком, 2006. - 452 с.

22. Сысоев Д.В. Введение в теорию искусственного интеллекта: учебное пособие/ Сысоев Д.В., Курипта О.В., Проскурин Д.К.– Электрон. текстовые данные.– Воронеж: Воронежский государственный архитектурно-строительный университет, ЭБС АСВ, 2014.– 171 с.

23. Трахтенгерц Э.А. Компьютерная поддержка принятия решений. - М.: СИНТЕГ, 1998. -376 с.

24. Чернов В.Г. Основы теории нечетких множеств : учеб. пособие / В.Г. Чернов ; Владим. гос. ун-т.- Владимир : Изд-во Владим. гос. ун-та, 2010. – 96 с.

25. Штовба С.Д. Проектирование нечетких систем средствами MATLAB. – М.: Горячая линия – Телеком, 2007. – 288 с.

26. Яхъяева Г.Э. Нечеткие множества и нейронные сети: учебное пособие/ Яхъяева Г.Э.– Электрон. текстовые данные.– М.: БИНОМ. Лаборатория знаний, ИнтернетУниверситет Информационных Технологий (ИНТУИТ), 2008.– 316 с

учебное издание

Нечеткая логика и нейронные сети

Учебное пособие

Составитель:
Полковская Марина Николаевна

Лицензия на издательскую деятельность
ЛР № 070444 от 11.03.98 г.
Подписано в печать 2023 г.

Издательство Иркутский государственный
аграрный университет им. А.А.Ежевского
664038, Иркутская обл., Иркутский р-н,
пос. Молодежный