ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет автоматики и вычислительной техники
Кафедра «Электронные вычислительные машины»

РОСТОВЦЕВ В.С.

Теория и применение нечеткой логики

Учебное пособие

Киров
2016

УДК 004.896(07)

Р785

Допущено к изданию методическим советом факультета

автоматики и вычислительной техники ФГБОУ ВПО «ВятГУ» в качестве учебного пособия для магистрантов направления 09.04.01 «Информатика и вычислительная техника» всех профилей подготовки, всех форм обучения.

Рецензент

доцент кафедры АТ ФГБОУ ВО «ВятГУ»,

кандидат технических наук

В.И. Семёновых

Ростовцев В.С.

Р785 Теория и применение нечеткой логики: учебное пособие. - Киров: Изд-во ВятГУ, 2016. -112 с.

Учебное пособие предназначено для студентов 1 курса очного отделения подготовки магистрантов по направлению 09.04.01 «Информатика и вычислительная техника» при выполнении лабораторных работ и самостоятельной работы в процессе изучения дисциплины «Теория и применение нечеткой логики».

Тех. редактор Е. В. Кайгородцева

ОГЛАВЛЕНИЕ

Введение

В 1970 году Беллман и Заде опубликовали статью "Decision - Making in Fuzzy Environment", которая послужила отправной точкой для большинства работ по нечеткой теории принятия решений. В той статье рассматривается процесс принятия решений в условиях неопределенности, когда цели и ограничения заданы нечеткими множествами [1,2,3,4].

Интерес к теории нечетких множеств постоянно возрастает. Об этом может свидетельствовать экспоненциальный рост публикаций в этой области за последние тридцать. Начиная с 90-х годов в области нечеткой логики акценты исследований сместились с общетеоретических - к прикладным инженерным.

В настоящее время издаются более десяти специализированных журналов по нечетким множествам и системам, среди которых “Fuzzy Sets and Systems” (издательство Elsevier Science), “Journal of Intelligent and Fuzzy Systems” (издательство IOS Press), “Intenational Journal of Uncertainty, Fuzziness and Knowledge-Based Systems” (издательство World333 Scientific), IEEE Transactions on Fuzzy Systems (издательство IEEE). Огромное количество журналов публикуют статьи по нечетким множествам, среди которых отметим следующие русскоязычные “Кибернетика и системный анализ”, “Известия РАН. Теория и системы управления”, “Автоматика и телемеханика”, “Автоматика и вычислительная техника”.

Принятие решения - это выбор альтернативы, которая одновременно удовлетворяет и нечетким целям, и нечетким ограничениям. В этом смысле, цели и ограничения являются симметричными относительно решения, что стирает различия между ними и позволяет представить решение как слияние нечетких целей и ограничений.

При принятии решений по схеме Беллмана-Заде не делается никакого различия между целью и ограничениями [3].

В настоящее время большую актуальность приобретает использование систем нечеткой логики для решения объемных, трудно формализуемых задач в различных предметных областях, причем области применения нечеткой логики постоянно расширяются. Нечеткие системы находят широкое применение в медицине и экономике, в автомобильной, аэрокосмической и транспортной промышленности, в области изделий бытовой техники, в сфере финансов, анализа и принятия управленческих решений и многих других. В результате, возникает необходимость в разработке адекватных моделей, эффективных алгоритмов и реализующих их программных комплексов.

Недостаток создания нечетких систем заключается в том, что набор правил вывода устанавливается человеком-экспертом. Построение окончательной модели требует зачастую объединения знаний специалистов многих областей и многократного тестирования системы. Экспертам, помимо правил, необходимо определиться с типами и параметрами функций принадлежности нечетких множеств, заданных на совокупности входных и выходных параметров, а это требует достаточно длительного и утомительного процесса подбора указанных величин.

Основное преимущество нечетких систем в отличие от нейронных сетей (НС) заключается в том, что знания в этих системах представляются в форме легко понимаемых человеком гибких логических конструкций, таких, как «IF … - THEN …». Кроме того, необходимо отметить, что, в соответствии с теоремой, доказанной Б. Коско (1993), любая математическая функция может быть аппроксимирована системой, основанной на нечеткой логике, следовательно, такие системы являются универсальными. Основные трудности при использовании нечетких систем

на практике связаны с априорным определением правил и построением функции принадлежности для каждого значения лингвистических переменных, описывающих структуру объекта, которые обычно проектировщик выполняет вручную.

Поскольку вид и параметры функций принадлежности выбираются субъективно, они могут быть не вполне адекватны реальной действительности.

Построение интеллектуальных систем на основе принципов синергизма является одним из перспективных направлений в области информатики. Наиболее важным свойством таких систем является самоорганизация. Под самоорганизующейся системой понимается такая система, которая способна сама в процессе взаимодействия ее компонентов и внешней среды перестраивать свои связи и порождать новую организацию с целью эффективного развития. Такая система характеризуется гибкостью и наличием обратных связей, большой степенью свободы компонентов и различными формами их коллективного поведения.

К настоящему моменту времени в результате развития принципов синергетики стали более понятны не только процессы организации интеллектуальных систем с лучшим качеством, но и появились новые методы, позволяющие создавать такие системы. Эти методы, дающие возможность системам, в которых они используются, обучаться и адаптироваться к внешней среде, обрабатывать неполную и неточную информацию и т.д. Новая идея интеграции этих технологий с традиционными системами искусственного интеллекта на основе принципов синергизма позволит создавать гибридные интеллектуальные системы, в которых будут использоваться преимущества рассматриваемых технологий и компенсироваться недостатки каждой из них.

Наиболее перспективными моделями, используемыми в интеллектуальных системах поддержки принятия решений в этом плане, являются модели на основе нечеткой логики и нейросетевой технологии. Объединение этих технологий в рамках единой системы позволяет создать следующие типы гибридных моделей [1]:

– нейросетевые нечеткие модели, в которых нейросетевая технология используется как инструмент в нечетких логических системах (для автоматического формирования функций принадлежности, определения нечетких отношений и т.д.);

– нечеткие нейронные сети, в которых с помощью аппарата нечеткой логики осуществляется фаззификация отдельных элементов нейросетевых моделей (нейронов, межнейронных связей, разделения на кластеры и т.д.);

– нечетко - нейросетевые гибридные модели, в которых объединение осуществляется на уровне методов для настройки одного из элементов модели.

Многообещающим представляется третий вид интеграции, при котором исходные модели не модифицируются, а лишь более "тонко" подстраиваются для более четкого выполнения некоторых функций.

1. ЭЛЕМЕНТЫ ТЕОРИИ НЕЧЕТКОЙ ЛОГИКИ

1.1. История развития нечетких систем

В Японии исследования в области нечеткой логики получили широкую финансовую поддержку. В Европе и США усилия были направлены на то, чтобы сократить огромный отрыв от японцев. Например, агентство космических исследований NASA стало использовать нечеткую логику в маневрах стыковки.

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 возможно формулировать математически и обрабатывать на компьютерах. Нечеткая логика появилась в 1965 в работах Лотфи А. Заде (Lotfi A. Zadeh), профессора технических наук Калифорнийского университета в Беркли [1,2,3,4].

Прежде чем нечеткий подход к моделированию сложных систем получил признание во всем мире, прошло не одно десятилетие с момента зарождения теории нечетких множеств. И на этом пути развития нечетких систем принято выделять три периода.

Первый период (конец 60-х–начало 70 гг.) характеризуется развитием теоретического аппарата нечетких множеств (Л. Заде, Э. Мамдани, Беллман). Во втором периоде (70–80-е годы) появляются первые практические результаты в области нечеткого управления сложными техническими системами (парогенератор с нечетким управлением). Одновременно стало уделяться внимание вопросам построения экспертных систем, построенных на нечеткой логике, разработке нечетких

контроллеров. Нечеткие экспертные системы для поддержки принятия решений находят широкое применение в медицине и экономике. Наконец, в третьем периоде, который длится с конца 80-х годов и продолжается в настоящее время, появляются пакеты программ для построения нечетких экспертных систем, а области применения нечеткой логики заметно расширяются. Она применяется в автомобильной, аэрокосмической и транспортной промышленности, в области изделий бытовой техники, в сфере финансов, анализа и принятия управленческих решений и многих других.

Широкое распространение нечеткой логики по миру началось после доказательства в конце 80-х Бартоломеем Коско знаменитой теоремы FAT (Fuzzy Approximation Theorem). В бизнесе и финансах нечеткая логика получила признание после того как в 1988 году экспертная система на основе нечетких правил для прогнозирования финансовых индикаторов единственная предсказала биржевой крах. И количество успешных фаззи-применений в настоящее время исчисляется тысячами.

Очевидной областью внедрения алгоритмов нечеткой логики являются всевозможные экспертные системы, в том числе:

– нелинейный контроль за процессами (производство);

– самообучающиеся системы (или классификаторы), исследование рисковых и критических ситуаций;

– распознавание образов;

– финансовый анализ (рынки ценных бумаг);

– исследование данных (корпоративные хранилища);

– совершенствование стратегий управления и координации действий, например, сложное промышленное производство.

В Японии это направление переживает настоящий бум. Здесь функционирует специально созданная лаборатория Laboratory for

International Fuzzy Engineering Research (LIFE). Программой этой организации является создание более близких человеку вычислительных устройств. LIFE объединяет 48 компаний в числе которых находятся: Hitachi, Mitsubishi, NEC, Sharp, Sony, Honda, Mazda, Toyota и др. Кроме японских участников LIFE можно выделить: IBM, Fuji Xerox. К деятельности LIFE проявляет интерес NASA.

Мощь и интуитивная простота нечеткой логики как методологии разрешения проблем гарантирует ее успешное использование во встроенных системах контроля и анализа информации. При этом происходит подключение человеческой интуиции и опыта оператора.

В отличие от традиционной математики, требующей на каждом шаге моделирования точных и однозначных формулировок закономерностей, нечеткая логика предлагает совершенно иной уровень мышления, благодаря которому творческий процесс моделирования происходит на наивысшем уровне абстракции, при котором постулируется лишь минимальный набор закономерностей.

Нечеткие числа, получаемые в результате “не вполне точных измерений”, во многом аналогичны распределениям теории вероятностей, но свободны от присущих последним недостатков: малое количество пригодных к анализу функций распределения, необходимость их принудительной нормализации, соблюдение требований аддитивности, трудность обоснования адекватности математической абстракции для описания поведения фактических величин. В пределе, при возрастании точности, нечеткая логика приходит к стандартной, Булевой. По сравнению с вероятностным методом, нечеткий метод позволяет резко сократить объем производимых вычислений, что, в свою очередь, приводит к увеличению быстродействия нечетких систем.

Недостатками нечетких систем являются:

– отсутствие стандартной методики конструирования нечетких систем;

– невозможность математического анализа нечетких систем существующими методами;

– применение нечеткого подхода по сравнению с вероятностным не приводит к повышению точности вычислений.

В настоящее время нечеткая логика рассматривается как стандартный метод моделирования и проектирования. В 1997 году язык нечеткого управления FCL внесен в Международный стандарт программируемых контроллеров IEC 1131-7.

Системы на нечетких множествах разработаны и успешно внедрены в таких областях, как: медицинская диагностика, техническая диагностика, финансовый менеджмент, управление персоналом, биржевое прогнозирование, распознавание образов, разведка ископаемых, выявление мошенничества, управление компьютерными сетями, управление технологическими процессами, управление транспортом, поиск информации в Интернете, радиосвязь и телевидение. Спектр приложений очень широкий от бытовых видеокамер, пылесосов и стиральных машин до средств наведения ракет противовоздушной обороны и управления боевыми вертолетами. По-прежнему лидирует Япония, в которой выпущено свыше 4800 «нечетких» патентов (в США около 1700 патентов).

Практический опыт разработки систем на нечетких множествах свидетельствует, что сроки и стоимость их проектирования значительно ниже, чем при использовании традиционного математического аппарата, при этом обеспечивается требуемые уровни качества. Это объясняется тем, что:

– нечеткая логика позволяет по экспертным знаниям быстро разработать прототип технического устройства с последующим усложнением его функциональности;

– модель на основе нечеткого логического вывода прозрачнее (проще для понимания), чем аналогичная модель на дифференциальных, разностных или иных уравнениях;

– нечеткие модели проще реализовать аппаратным способом, обеспечивая при этом распараллеливание вычислений.

Области применения систем, в которых используется нечеткая логика:

– выбор эффективного варианта инвестиционного проекта в условиях наличия неопределенности (невозможность предсказания инфляции, спроса, технических и ресурсных показателей самого проекта).

– автоматическое управление воротами плотины на гидроэлектростанциях;

– упрощенное управление роботами;

– наведение телекамер при трансляции спортивных событий;

– замена экспертов при анализе работы биржи;

– предотвращение нежелательных температурных флуктуаций в системах кондиционирования воздуха;

– эффективное и стабильное управление автомобильными двигателями;

– управление экономичной скоростью автомобилей;

– улучшение эффективности и оптимизация промышленных систем управления;

– позиционирование приводов в производстве полупроводников wafer-steppers;

– оптимизированное планирование автобусных расписаний;

– системы архивации документов;

– системы прогнозирования землетрясений;

– медицина: диагностика рака;

– сочетание методов нечеткой логики и нейронных сетей;

– распознавание рукописных символов в карманных компьютерах (записных книжках);

– распознавание движения изображения в видеокамерах;

– автоматическое управление двигателем пылесосов с автоматическим определением типа поверхности и степени засоренности;

– управление освещенностью в камкодерах;

– компенсация вибраций в камкодерах;

– однокнопочное управление стиральными машинами;

– распознавание рукописных текстов, объектов, голоса;

– вспомогательные средства полета вертолетов;

– моделирование судебных процессов;

– САПР производственных процессов;

– управление скоростью линий и температурой при производстве стали;

– управление метрополитенами для повышения удобства вождения, точности остановки и экономии энергии;

– оптимизация потребления бензина в автомобилях;

– повышение чувствительности и эффективности управления лифтами;

– повышение безопасности ядерных реакторов.

Реализация нечеткой системы включает в среднем 50 правил, 6 лингвистических переменных. В результате внедрения нечеткой системы кондиционирования воздуха в помещении сокращается время ее создания на 40 процентов к стандартному решению, обеспечивается поддержка температуры при наличии возмущающих факторов (открытые окна и т.п.), экономия энергии составляет до 24 процентов.

Пример: Система кондиционирования воздуха Mitsubishi.

Промышленная система кондиционирования воздуха, обеспечивающая гибкую реакцию на изменения окружающих условий.

Реализация: 50 правил; 6 лингвистических переменных; Разрешение: 8 бит; Входные значения: температура в комнате, температура стены и мгновенные значения этих сигналов.

Разработка: 4 дня на создание прототипа; 20 дней на тестирование и интеграцию; 80 дней на оптимизацию на реальных тестовых объектах; Реализация в виде чисто программного комплекса на стандартном микроконтроллере.

Результаты применения системы кондиционирования воздуха Mitsubishi: сокращение времени и трудоемкости разработки до 40 процентов к стандартному решению; поддержка температуры при наличии возмущающих факторов (открытые окна и т.п.) существенно улучшена; используется небольшое число датчиков; экономия энергии - 24 процента.

Использование нечеткого управления рекомендуется в следующих случаях:

- для очень сложных процессов, когда не существует простой математической модели;
- для нелинейных процессов высоких порядков;
- для обработки лингвистически экспертных знаний.

Использование нечеткого управления не рекомендуется в следующих случаях:

- приемлемый результат может быть получен с помощью общей теории управления;
- уже существует формализованная и адекватная математическая модель;
- проблема не разрешима.

1.2. Основные понятия в области нечетких множеств

Математическая теория нечетких множеств (fuzzy sets) и нечеткая логика (fuzzy logic) являются обобщениями классической теории множеств и классической формальной логики. Данные понятия были впервые предложены американским ученым Лотфи Заде (Lotfi Zadeh) в 1965 году. Основной причиной появления новой теории стало наличие нечетких и приближенных рассуждений при описании человеком процессов, систем, объектов [1,2,3,4,5]. Следует использовать общепринятую терминологию.

Нечеткое множество (fuzzy sets) – множество, элементы которого принадлежат ему, в той или иной степени.

Нечеткая логика (fuzzy logic) – умозаключение с использованием нечетких множеств или нечетких правил.

Нечеткое правило (fuzzy rule) – условное высказывание вида "ЕСЛИ X есть A, ТО У есть B", где A и B нечеткие множества. Нечеткое правило образует связь между нечеткими множествами.

Нечеткая система (fuzzy system) – множество нечетких правил, преобразующих входные данные в выходные.

Американским ученым Лотфи Заде (Lotfi Zadeh) в 1994 году введен еще один термин «мягкие вычисления», представляющий сложную компьютерную методологию [1,2], компонентами которой являются:

– нечеткая логика (приближенные вычисления, грануляция информации, вычисление на словах);

– нейрокомпьютинг (обучение, адаптация, классификация, системное моделирование и идентификация);

– генетические вычисления (синтез, настройка и оптимизация с помощью систематизированного случайного поиска и эволюции);

– вероятностные вычисления (управление неопределенностью, сети доверия, хаотические системы, предсказание).

Интерпретация «мягких вычислений» может быть следующая: *«мягкие вычисления — это нечеткие системы + нейронные сети+ +генетические алгоритмы»*.

Свойства мягкой интеллектуальной системы обеспечиваются ее компонентами. Мягкие системы, такие, как нечеткие нейронные сети с генетической настройкой параметров, демонстрируют взаимное усиление достоинств и нивелирование недостатков отдельных методов.

Нечёткие множества. Понятие нечеткого множества - эта 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. При таком подходе высказывания типа “такой-то элемент принадлежит данному множеству” теряют смысл, поскольку необходимо указать “насколько сильно” или с какой степенью конкретный элемент удовлетворяет свойствам данного множества.

Нечетким множеством (fuzzy set) на универсальном множестве называется совокупность пар (μ_A (u), u), где μ_A (u), - степень принадлежности элемента $u \in U$ к нечеткому множеству A. Степень принадлежности - это число из диапазона [0.1]. Чем выше степень принадлежности, тем с большей мерой элемент универсального множества соответствует свойствам нечеткого множества.

Функцией принадлежности (*membership function)* называется функция, которая позволяет вычислить степень принадлежности произвольного элемента универсального множества к нечеткому множеству [1].

Если универсальное множество состоит из конечного количества элементов U= {u_1, u_2, …, u_k}, тогда нечеткое множество А записывается в виде:

$A = \sum_{i=1}^{k} \mu A\,(ui)/\,u\,i$,

где μ_A (u) - степень принадлежности элемента $u \in U$ к нечеткому множеству А;

u_i - элемент множества U.

В нечетких системах элемент может частично принадлежать к любому множеству. Значения функции принадлежности являются рациональными числами из интервала [0,1], где 0 означает отсутствие принадлежности к множеству, а 1 - полную принадлежность. Конкретное значение функции принадлежности называется степенью или коэффициентом принадлежности. Эта степень может быть определена явным образом в виде функциональной зависимости. Например, μ_A (x) = exp (- ($\frac{x-3}{0.2}$)^2), либо дискретно - путем задания конечной последовательности значений $x \in \{xn\}$ в виде:

$$A(x) = \left\{ \frac{\mu(x1)}{x1}, \frac{\mu(x2)}{x2}, ..., \frac{\mu(x3)}{x3} \right\}.$$

Например, для последовательности дискретных значений переменной x, равных x1= 7, x2 = 8, x3 = 9, x4 = 10, x5 = 11, x6 = 12, x7 = 13, их коэффициент принадлежности к числам, близким 10, может быть определен в виде

$$A(x) = \left\{ \frac{0.1}{7}, \frac{0.3}{8}, \frac{0.8}{9}, \frac{1.0}{10}, \frac{0.8}{11}, \frac{0.3}{12}, \frac{0.1}{13}, \right\}. \quad -$$

В теории нечетких множеств, помимо переменных цифрового типа, существуют лингвистические переменные с приписываемыми им значениями. Пусть переменная x обозначает температуру (x = "температура"). Можно определить нечеткие множества "отрицательная",

"близкая к нулю", "положительная", характеризуемые функциями принадлежности $\mu_{\text{отрицательная}}(x)$, $\mu_{\text{близкое\_к\_нулю}}(x)$, $\mu_{\text{положительная}}(x)$. Так же как обычная переменная может принимать различные значения, лингвистическая переменная "температура" может принимать различные μ лингвистические значения. В нашем примере это: "отрицательная", "близкая к нулю" и "положительная". Следовательно, лингвистическое выражение может иметь вид: "температура отрицательная", "температура, близкая к нулю", "температура положительная". На рис.1.1 приведена графическая иллюстрация функции принадлежности переменной x = T (где T означает температуру) для трех названных множеств значений температуры. Непрерывными линиями обозначена классическая (точная) принадлежность, а пунктирными линиями - нечеткая принадлежность. Можно отметить, что функция нечеткой принадлежности является непрерывным приближением пороговой функции точной принадлежности.

Каждое нечеткое множество имеет определенный носитель. Носителем множества *Supp(A)* является подмножество тех элементов A, для которых коэффициент принадлежности к A не равен нулю, т.е. Supp (A) = {x, μA(x) > 0}. На рис.1.1 носителем множества "близкая к нулю" является множество температур в интервале от минус 4 до плюс 4°C [6].

В нечеткой логике основополагающим является понятие лингвистической переменной, значениями которой являются не числа, а слова естественного языка, называемые термами.

Лингвистической переменной (ЛП) называется переменная, значением которой являются нечеткие множества, описываемые в форме слов или предложений на естественном или искусственном языке. Формально ЛП задается набором множеств:

{X, T(X), U, G, M},

где X- название переменной;

T(X)- терм-множество переменной X;

U- универсальное множество;

G – синтаксическое правило, порождающее названия значений переменной X;

M- семантическое правило, которое ставит в соответствие каждому значению ЛП ее смысл.

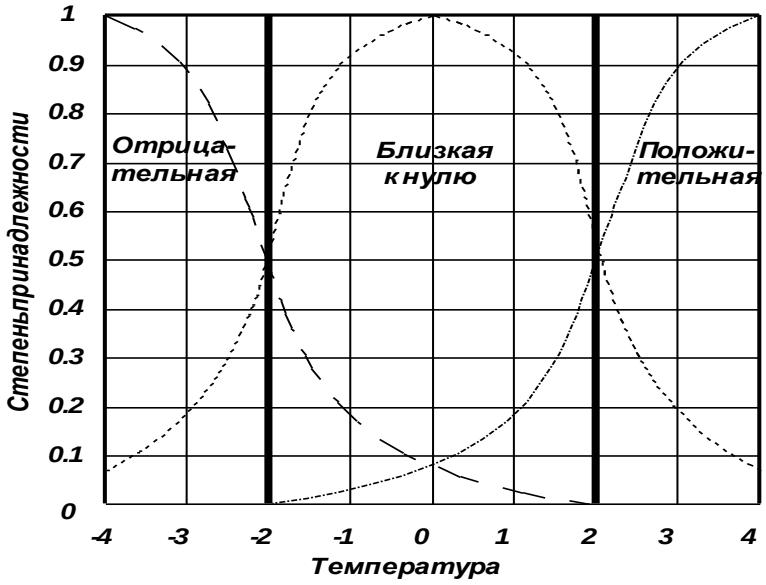


Рис. 1.1. Иллюстрация понятия принадлежности температуры

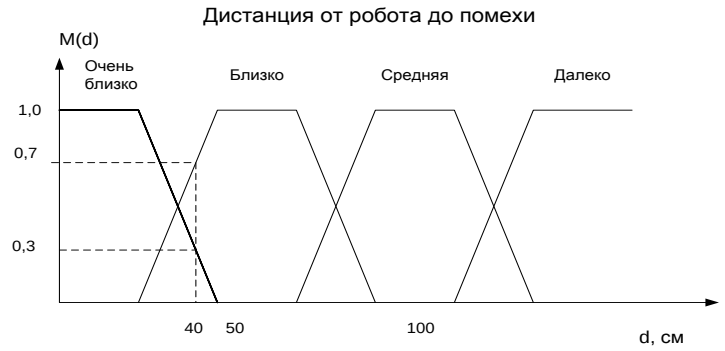
Пример. X=ОЦЕНКА T(ОЦЕНКА)={неудовлетворительно, удовлетворительно, хорошо, отлично}. U=[0,100].

Например, нечеткое понятие как ЦЕНА АКЦИИ представляет наименование лингвистической переменной. Для нее базовое терм-множество будет состоять из трех нечетких переменных: НИЗКАЯ, УМЕРЕННАЯ, ВЫСОКАЯ. Область рассуждений определяется в виде X=[100;200] единиц [1,2].

Для каждого лингвистического терма из базового терм-множества T должна быть построена функция принадлежности. Количество термов в лингвистической переменной обычно не превышает семи.

Например, в случае управления мобильным роботом можно ввести две лингвистические переменные: ДИСТАНЦИЯ (расстояние до помехи) и НАПРАВЛЕНИЕ (угол между продольной осью робота и направлением на помеху).

Рассмотрим лингвистическую переменную ДИСТАНЦИЯ. Значениями ее можно определить термы ДАЛЕКО, СРЕДНЯЯ, БЛИЗКО и ОЧЕНЬ БЛИЗКО. Для физической реализации лингвистической переменной необходимо определить точные физические значения термов этой переменной. Пусть переменная ДИСТАНЦИЯ может принимать любое значение из диапазона от нуля до бесконечности. Согласно положениям теории нечетких множеств, каждому значению расстояния из указанного диапазона может быть поставлено в соответствие некоторое число от нуля до единицы, которое определяет степень принадлежности данного физического расстояния (допустим 40 см) к тому или иному терму лингвистической переменной ДИСТАНЦИЯ.



Лингвистическая переменная «ДИСТАНЦИЯ» включает 4 терма
«ДИСТАНЦИЯ»= {очень близко, близко, средняя, далеко }

Рис.1.2. Лингвистическая переменная «ДИСТАНЦИЯ»

Степень принадлежности определяется так называемой функцией принадлежности M(d), где d - расстояние до помехи. В нашем примере

расстоянию 40 см можно задать степень принадлежности к терму ОЧЕНЬ БЛИЗКО, равную 0,7, а к терму БЛИЗКО – 0,3 (рис.1.2). Конкретное определение степени принадлежности может проходить только при работе с экспертами.

Переменной НАПРАВЛЕНИЕ, которая может принимать значения в диапазоне от 0 до 360 градусов, зададим термы ЛЕВОЕ, ПРЯМО И ПРАВОЕ.

Теперь необходимо задать выходные переменные. В рассматриваемом примере достаточно одной, которая будет называться РУЛЕВОЙ УГОЛ. Она может содержать термы: РЕЗКО ВЛЕВО, ВЛЕВО, ПРЯМО, ВПРАВО, РЕЗКО ВПРАВО. Связь между входом и выходом запоминается в виде нечетких правил (рис.1.3).

Направление	Дистанция			
	Очень близко	Близко	Средняя	Далеко
Правое	Резко влево	Резко влево	Влево	Прямо
Прямо	Резко влево	Влево	Влево	Прямо
Левое	Резко вправо	Резко вправо	Вправо	Прямо

Рис.1.3. База нечетких правил

Каждая запись, представленная на рис.1.3, соответствует своему нечеткому правилу: «Если ДИСТАНЦИЯ БЛИЗКО и НАПРАВЛЕНИЕ ПРАВОЕ, тогда РУЛЕВОЙ УГОЛ РЕЗКО ВЛЕВО».

Таким образом, мобильный робот с нечеткой логикой будет работать по следующему принципу: данные с сенсоров о расстоянии до помехи и направлении на нее будут фаззифицированы, обработаны согласно табличным правилам, дефаззифицированы, и полученные данные в виде управляющих сигналов поступят на привод робота.

Характеристикой нечеткого множества выступает функция принадлежности (Membership Function). μF C(x) – степень принадлежности значения «x» к нечеткому множеству C, представляющая собой обобщение понятия характеристической функции обычного множества. Тогда нечетким множеством C называется множество упорядоченных пар вида C={μFc(x)/x}, μFc(x) принадлежит интервалу от 0 до 1. Значение μFc(x)=0 означает отсутствие принадлежности к множеству, 1 – полную принадлежность.

Задание лингвистической переменной можно осуществить в дискретной или непрерывной форме.

Дискретная форма:

A(x)={μ(x1) /x1, μ(x2) /x2, …, μ(xn) /xn}.

Непрерывная форма в виде функциональной зависимости.

$\mu A(x) = \exp(-(x-3)/0,2)^2$.

Пример 1. Для множества чисел от x1=7, x2=8, …, x7=13 степень принадлежности их к числам, близким к 10, выглядит таким образом:

A(x)={0,1/7; 0.3/8; 0.8/9; 1.0/10; 0.8/11; 0.3/12; 0.1/13}.

Примером нечеткого множества может служить формализация неточного определения ГОРЯЧИЙ ЧАЙ. В качестве x (область рассуждений) будет выступать шкала температуры в градусах Цельсия. Очевидно, что она будет изменяется от 0 до 100 градусов. Нечеткое множество для понятия ГОРЯЧИЙ ЧАЙ может выглядеть следующим образом:

C= {0/0; 0/10; 0/20; 0,15/30; 0,30/40; 0,60/50; 0,80/60; 0,90/70; 1/80; 1/90; 1/100}.

Например, чай с температурой 60 градусов Цельсия, принадлежит к множеству ГОРЯЧИЙ ЧАЙ со степенью принадлежности 0,80. Для одного человека чай с температурой 60 градусов Цельсия может оказаться горячим, для другого – не слишком горячим. Именно в этом и проявляется нечеткость задания соответствующего множества.

В нечеткой логике есть возможность строить выражения, на основе которых можно получать новые значения нечетких (следовательно, и лингвистических) переменных.

Применение правил нечеткого вывода позволяет получить описание предметной области на основе имеющихся у эксперта знаниях о ней. Благодаря операциям фаззификации и дефаззификации имеется возможность обработки данных, общая схема которой представляется таким образом:

1. По имеющемуся набору данных, с помощью оператора фаззификации провести перевод этих данных в нечеткий формат.
2. Пользуясь имеющейся базой нечетких правил, преобразовать нечеткие переменные.
3. Получившийся набор нечетких переменных подвергнуть дефаззификации и получить четкое значение.

Часть выходных данных можно передать через обратную связь на вход, если того требует алгоритм управления.

Дефаззификацией (defuzzification) называется процедура преобразования нечеткого множества в четкое число. В теории нечетких множеств процедура дефаззификации аналогична нахождения 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. Наиболее широко распространенными являются следующие функции 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.

Свойства нечетких множеств и операции над ними подробно описаны в [1,2,3].

1.3. Элементы нечёткой логики

Нечеткая логика — это обобщение традиционной аристотелевой логики на случай, когда истинность рассматривается как лингвистическая переменная, принимающая значения типа: "очень истинно", "более-менее истинно", "не очень ложно" и т.п. Указанные лингвистические значения представляются нечеткими множествами.

Лингвистической называется переменная, принимающая значения из множества слов или словосочетаний некоторого естественного или искусственного языка. Множество допустимых значений лингвистической переменной называется терм-множеством. Задание значения переменной словами, без использования чисел, для человека более естественно. Ежедневно люди принимают 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.

Лингвистическая переменная задается пятеркой <X, T, U, G, M>, где X - имя переменной; T - терм-множество, каждый элемент которого (терм) представляется как нечеткое множество на универсальном множестве U; G-синтаксические правила, часто в виде грамматики, порождающие название термов; M - семантические правила, задающие функции принадлежности нечетких термов, порожденных синтаксическими правилами G.

Особое место в нечеткой логике занимает лингвистическая переменная "истинность". В классической логике истинность может принимать только два значения: истинно и ложно. В нечеткой логике истинность "размытая". Нечеткая истинность определяется аксиоматически, причем разные авторы делают это по-разному. Интервал [0,1] используется как универсальное множество для задания лингвистической переменной "истинность". Обычная, четкая истинность может быть представлена нечеткими множествами – синглтонами. В этом случае четкому понятию истинно будет соответствовать функция принадлежности

$$\mu_{истинно}(u) = \begin{cases} 0, u \neq 1 \\ 1, u = 1 \end{cases}$$

а четкому понятию ложно

$$\mu_{ложно}(u) = \begin{cases} 0, u \neq 0 \\ 1, u = 1 \end{cases}, \quad u \in [0,1]$$

Для задания нечеткой истинности Заде предложил такие функции принадлежности термов "истинно" и "ложно":

$$\mu_{истинно}(u)=\begin{cases}0, & 0\le u\le a\\ 2\left(\dfrac{u-a}{1-a}\right)^2, & a<u<\dfrac{a+1}{2}\\ 1-2\left(\dfrac{u-1}{1-a}\right)^2 & \dfrac{a+1}{2}<u\le 1\end{cases},$$

$$\mu_{ложно}(1-u)=\mu_{истинно}(1-u)\ \ u\in[0,1],$$

где a $\in [0,1]$ – параметр, определяющий носители нечетких множеств "истинно" и "ложно". Для нечеткого множества "истинно" носителем будет интервал (a,1], а для нечеткого множества "ложно" – [0, a).

Функции принадлежности нечетких термов "истинно" и "ложно" изображены на рис.1.4. Они построены при значении параметра a=0,4.

Как видно, графики функций принадлежности термов "истинно" и "ложно" представляют собой зеркальные отображения.

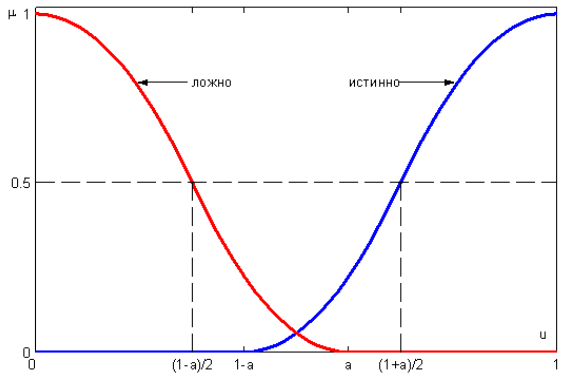


Рис.1.4. Лингвистическая переменная "истинность" по Л.Заде

С помощью операций $\cup$ (ИЛИ) и $\cap$ (И) нечеткую базу знаний можно переписать в более компактном виде:

$$\bigcup_{p=1}^{k_j}\left[\bigcap_{i=1}^{n}\left(x_i = a_i^{j_p}\right)\right] \to y = d_j, \quad j = 1,\ldots,m,$$

где $a_l^{j_p}$ - нечеткий терм, которым оценивается переменная xi в строчке с номером jp, p=1,..,kj;

kj- количество строчек-конъюнкций, в которых выход y оценивается нечетким термом dj, j=1,…,m;

m количество термов, используемых для лингвистической оценки выходного параметра y.

Нечетким логическим выводом (fuzzy logic inference) называется получение заключения в виде нечеткого множества, соответствующего текущим значениях входов, с использованием нечеткой базы знаний и нечетких операций. Иначе, нечеткий логический вывод может быть определен как аппроксимация зависимости y=f(x1, x2,…,xn) с помощью нечеткой базы знаний и операций над нечеткими множествами.

Основу нечеткого логического вывода составляет композиционное правило Заде.

Правило *modus ponens* обеспечивает вывод заключения "B есть истинно", если известно, что "A есть истинно" и существует правило "Если A, то B" (A и B- четкие логические утверждения).

Однако если прецедент отсутствует, то правило modus ponens не обеспечит вывод даже приближенного заключения. Даже в случае, когда известно, что близкое к A утверждение A' является истинным, правило modus ponens не может быть применено. Одним из возможных способов принятия решений при неопределенной информации является применение нечеткого логического вывода.

1.4. Математические операции над нечеткими множествами

На нечетких множествах, рассматриваемых как обобщение обычных множеств, можно определить ряд математических операций, являющихся обобщением аналогичных операций, выполняемых на "четких" множествах. К ним среди прочих относятся [1,2,3]:

Логическая сумма нечетких множеств $A \cup B$

$$\mu A \cup B(x) = \mu A(x) \cup \mu B(x) = Max[A(x), B(x)],$$

где знак $\cup$ обозначает оператор *Max*.

График объединения нечетких множеств приведен на рис 1.5.

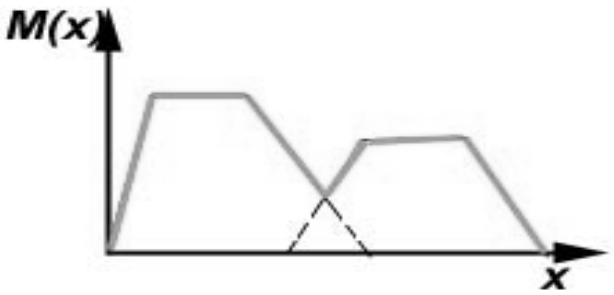


Рис.1.5. Объединение нечетких множеств

Логическое произведение множеств $A \cap B$

$$\mu A \cap B(x) = \mu A(x) \cap \mu B(x) = Min[A(x), B(x)]$$

где знак $\cap$ обозначает оператор min.

График пересечения приведен на рис.1.6.

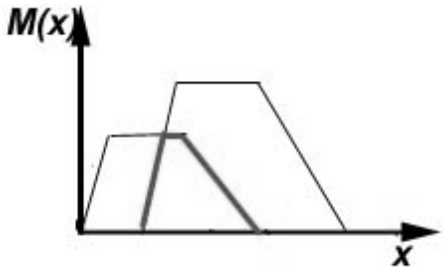


Рис.1.6. Пересечение нечетких множеств

Отрицание множества $\overline{A}$

$$\mu\overline{A}(x) = 1 - \mu A(x)$$

График дополнения приведен на рис.1.7.

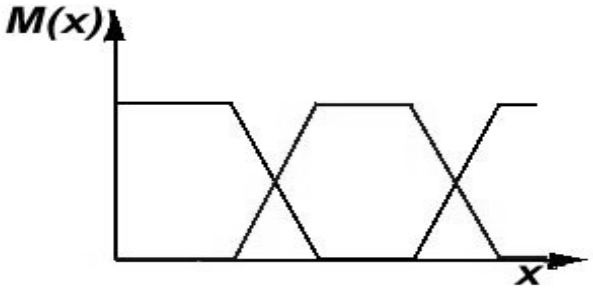


Рис.1.7. Дополнение (отрицание) нечеткого множества

В отличие от обычных (четких) множеств, где отрицание элементов, принадлежащих к множеству, дает пустое множество, *отрицание нечеткого множества* определяет непустое множество, состоящее из элементов, функции принадлежности которых также определены на интервале [0,1].

Высотой d нечеткого множества A называется максимальное значение функции принадлежности этого множества $d=\max \mu_A(x)$ $x \in X$.

Если d = 1, то нечеткое множество называется нормальным.

Равенство множеств A и B. Нечеткие множества A(x) и B(x) равны между собой, когда для всех элементов x_i, обоих множеств выполняется условие $\mu_A(x_i)=\mu_B(x_i)$.

Кардинальное число нечеткого множества A равно сумме коэффициентов принадлежности всех элементов к этому множеству $\mu(A)=\sum_i^n \mu A(xi)$.

Сечением α нечеткого множества A называется подмножество

A α , содержащее те элементы множества A, для которых $\mu A(x)>\alpha$ (слабое сечение) и $\mu A(x)>=\alpha$ (сильное сечение) при $\alpha = [0,1]$.

Операция концентрации CON (A)

$$\mu con(x) = [\mu A(x)]^2$$

Эта операция весьма часто выполняется при действиях с лингвистической переменной, в которых она отождествляется с понятием «очень».

Графики размытия и концентрации показаны на рис.1.8.

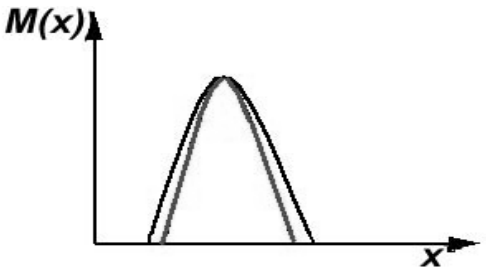


Рис.1.8. Размытие (концентрация) нечеткого множества

Операция растяжения DIL(A)

$$\mu dil(x) = [\mu A(x)]^{0.5}$$

Лингвистическое значение этой операции формулируется как "примерно" либо "приблизительно".

*Алгебраическое произведение двух множеств A * B*

$$\mu A * B(x) = \mu A(x) * \mu B(x)$$

Ограниченная сумма двух нечетких множеств A | +| B

$$\mu_{A|+|B}(x) = \min\{1, \mu A(x) + \mu B(x)\}$$

Ограниченная разница двух нечетких множеств A | - | B

$$\mu_{A|-|B}(x) = \max\{0, \mu A(x) - \mu B(x)\}.$$

Ограниченное произведение двух нечетких множеств A||B*

$$\mu_{A|\cdot|B}(x) = \max\{0, \mu A(x) + \mu B(x) - 1\}$$

Нормализация множества NORM (A)

$$\mu norm(x) = \frac{\mu A(x)}{\max\{\mu A(x)\}}.$$

Определенные на нечетких множествах операции обладают свойствами ассоциативности, коммутативности и дистрибутивности, причем эти свойства понимаются следующим образом:

- ассоциативность: (A* B) *C = A* (B*C);
- коммутативность: A*B = B* A (за исключением ограниченной разности);
- дистрибутивность: A* (B ° C) = (A*B) ° (A*C).

Здесь операторы * и ° обозначают любую определенную выше операцию на нечетких множествах. Из свойств нечетких множеств следует, что в отличие от произведения обычных множеств логическое произведение множества и его отрицание не обязательно образуют пустое множество, что можно записать в виде:

$$A \cap \overline{A} \neq 0 .$$

Точно так же и логическая сумма нечеткого множества A и его отрицание не образуют полное множество U, что можно записать $A \cup \overline{A} \neq U$.

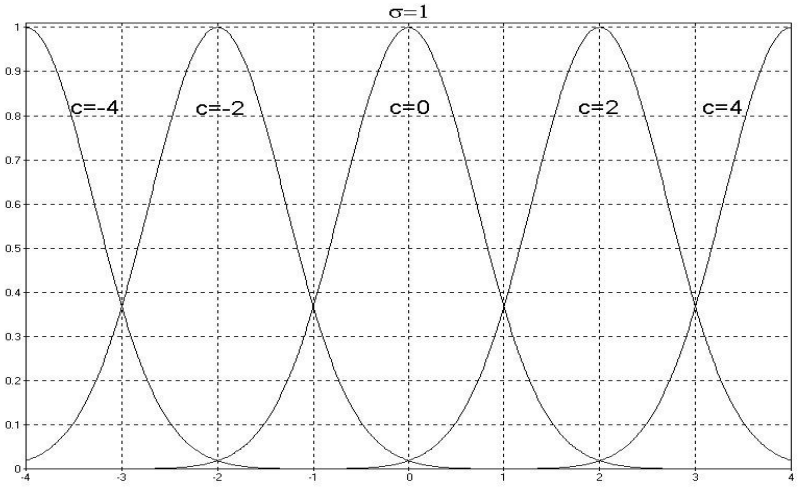
1.5. Методы фаззификации

Фаззификатор преобразует N-мерный входной вектор X =[x1, x2 , ..., xN]T в нечеткое множество A, характеризуемое функцией принадлежности μ(x) с четкими переменными. Несмотря на то, что нечеткие системы могут иметь функции принадлежности произвольной структуры, с практической точки зрения наибольшей популярностью пользуются функции гауссовского типа, а также треугольные и трапецеидальные функции [1,2,3].

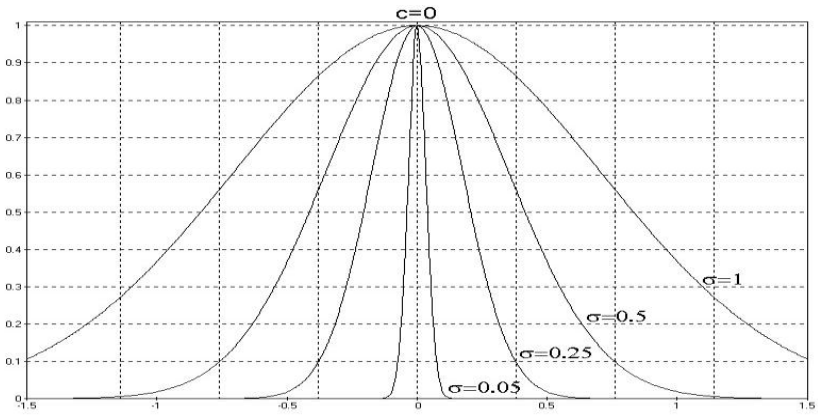
Общая форма обобщённой гауссовской функции для переменной x с центром c и вариацией σ и параметром b для множества F имеет вид:

$$\mu_A(x) = \exp\left[-\left(\frac{x-c}{\sigma}\right)^{2b}\right]$$

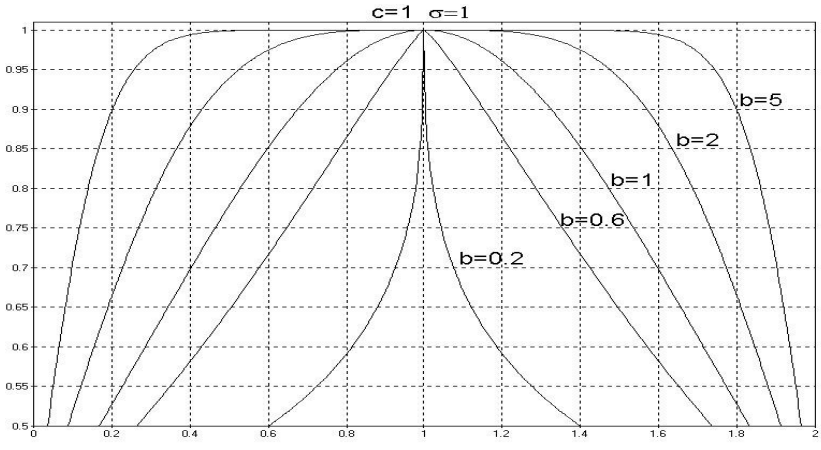
На рис.1.9. представлена формы типовых гауссовских функций при различных параметрах c, σ и b, причем на рис.1.9а показано влияние размещения центра c, на рис.1.9б – влияние значения σ, а на рис.1.9в – влияние значения b. Параметр c обозначает центр нечеткого множества, а его изменение соответствует смещению функции принадлежности по горизонтальной оси. Параметр σ, иногда называемый коэффициентом широты, отвечает за форму функции. Чем меньше его значение, тем больше крутизна функции. Значение параметра b существенным образом влияет на форму кривой. При соответствующем подборе показателя степени b она может определять, как функцию Гаусса, так и треугольную или трапецеидальную функцию. Значение b = 1, очевидно, соответствует стандартной гауссовской функции.



а)



б)



в)

а) влияние размещения центра c при σ = 1 и b=1;

б) влияние значения σ при постоянных значениях c = 0 и b=1;

в) влияние значения b при постоянных значениях c = 1 и σ=1

Рис.1.9. Иллюстрация влияния параметров функции Гаусса

Помимо гауссовской функции принадлежности, на практике часто применяется симметричная треугольная функция, которую можно записать в виде

$$\mu_A(x) = \begin{cases} 1 - \dfrac{|x - c|}{d} & \text{для } x \in [c - d, c + d] \\ 0 & \text{для остальных} \end{cases}$$

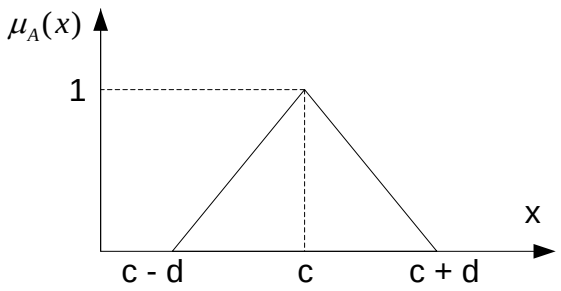


Рис.1.10. Симметричная треугольная функция принадлежности

Интерпретация центральной точки c и ширины d для треугольной функции представлена на рисунке. Эта функция тоже нормирована и принимает единичное значение в точке c.

Обобщением треугольной функции является трапецеидальная функция принадлежности, форма и обозначение которой приведены на рис.1.11.

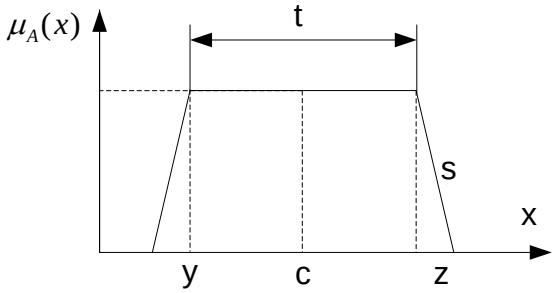


Рис.1.11. Трапецеидальная функция принадлежности

Если определить y = c – (t/2) – (1/s), z = c + (t/2) + (1/s), где s обозначает угол наклона, то трапецеидальная функция описывается зависимостью:

$$\mu_{A^{(i)}}(x_i) = \begin{cases} 0 & \text{для } x > z \text{ или } x < y \\ 1 & \text{для } c - t/2 \le x \le c + t/2 \\ s(z - x) & \text{для } c + t/2 \le x \le z \\ s(z - y) & \text{для } y \le x \le c - t/2 \end{cases}$$

Выбор значения t = 0 редуцирует трапецеидальную функцию до треугольной формы.

1.6. Методы дефаззификации

Дефаззификатор трансформирует нечеткое множество в полностью детерминированное точечное решение y. Нечеткое множество представляет зависимость μ(y)=μA→B(y) как функцию от выходной переменной y. Преобразование этого множества в единственное точечное решение возможно многими способами. Наиболее известны среди них:

- дефаззификация относительно центра области

$$y_c = \frac{\int_y \mu(y) y dy}{\int_y \mu(y) dy}$$

- либо в дискретной форме

$$y_c = \frac{\sum_i \mu(y_i) y_i}{\sum_i \mu(y_i)}$$

- дефаззификация относительно среднего центра

$$y_c = \frac{\sum_i \mu(y_{ci}) y_{ci}}{\sum_i \mu(y_{ci})}$$

где ycі обозначает центр і-го нечеткого правила, а $\mu(y_{ci})$ - значение функции принадлежности, соответствующей этому правилу;

– дефаззификация относительно среднего максимума

$$y_M = \frac{\sum_{i=1}^{m} y_i}{m}$$

где m обозначает количество точек переменной y, в которых $\mu(y_{ci})$ достигает максимального значения. Если функция μ(y) имеет максимальное значение только в одной точке y max, то yM =y $_{max}$. Если μ(y) достигает своих максимальных значений между yl и yp, то yM=1/2(yl + yp).

– в форме выбора минимального из максимальных значений y ys - наименьшее значение y, для которого {μ (y) = max};
– дефаззификация в форме выбора максимального из максимальных значений y.

yl – наибольшее значение y, для которого {μ (y) = max}.

На практике чаще всего применяется дефаззификация относительно среднего центра. Например, пусть представлен нечеткий сигнал, полученный после агрегирования двух правил вывода.

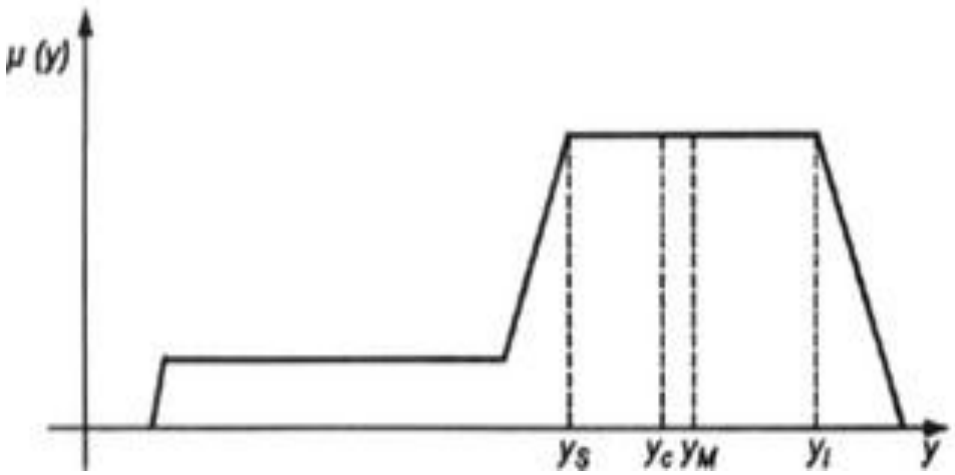


Рис.1.12. Влияния различных способов дефаззификации на решение

1.7. Меры нечеткости

Для определения степени нечеткости множества введено понятие меры нечеткости, определяемой различием множества A и его отрицания ~A.

Наиболее популярна мера Егера, в соответствии с которой степень нечеткости нечеткого множества A в метрике p обозначается FUZp(A) и определяется по формуле

$FUZp(A)= 1/n^{1/p} [1- Dp(A, \sim A)]$,

где Dp(A, ~A) – это мера расстояния между множествами A и ~A, содержащими n элементов. При p=1 это метрика Хемминга, а при p=2 это метрика Евклида.

Для метрики Хемминга $FUZ_1(A)= [1- D_1(A, \sim A)]* 1/n$.

Для метрики Эвклида $FUZ_2(A)= [1- D_2(A, \sim A)]* 1/\sqrt{n}$.

$D_1 (A, \sim A) = \sum_{i=1}^{n} |2 * \mu A(xi) - 1|$ Метрика Хемминга

$D_2 (A, \sim A) = \sum_{i=1}^{n} (2 * \mu A(xi) - 1)^2$. Метрика Евклида

Пример. Нечеткое множество A определяется дискретным способом.

A = {0.1/x1; 0.5/x2; 0.8/x3; 1.0/x4; 0.8/x5; 0.5/x6; 0.1/x7}

~A= {0.9/x1; 0.5/x2; 0.2/x3; 0.0/x4; 0.2/x5; 0.5/x6; 0.9/x7}

$FUZ_1 (A) = 1-1/7(0.8+0+0.6+1+0.6+0+0.8) =0.457$

$FUZ_2 (A) = 1-1/\sqrt{7} (0.64+0+0.36+1+0.36+0+0.64) =0.347$

Другую меру нечеткости предложил Коско на основе кардинального числа нечеткого множества

$FUZ (A) =M (A \cap \sim A) / M (A \cup \sim A)$,

где M(A) обозначает кардинальное число нечеткого множества.

FUZ(A)=(0,1+0,5+0,2+0+0,2+0,5+0,1) / (0,9+0,5+0,8+1+0,8+0,5+0,9) =0,296.

Нечеткость и вероятность. Вероятность наступления события относится к будущему. Например, 8 ноября в течение последних 100 лет день был снежным 82 раза. Вероятность снега в этот день равна 0,82. Когда соответствующий день наступит, то данное событие либо произойдет, либо нет, и в этот момент понятие вероятности утрачивает смысл. Например, на начало дня еще не ясно, будет ли день снежным, а когда день прошел, то мы будем говорить о свершении события, а не о его вероятности.

Понятие нечеткости оценивается другим способом. Оно измеряется степенью, с которой событие принадлежит некоторому множеству А. Например, 8 ноября в течение 15 часов шел снег, то его степень принадлежности к множеству снежных дней можно определить, как 15/24. С этой точки зрения понятие нечеткости относится не только к прошлому, но и к настоящему, и к будущему.

Контрольные вопросы

1. Области применения нечетких систем.
2. Преимущества нечетких систем.
3. Мера нечеткости Егера в метрике Хемминга.
4. Мера нечеткости Егера в метрике Евклида.
5. Мера нечеткости Коско.
6. Математические операции над нечеткими множествами/
7. Кардинальное число.
8. Операции концентрации и растяжения.
9. Операторы объединения и пересечения.
10. Нечеткость и вероятность.
11. Понятие нечеткой системы.
12. Понятие нечеткого множества.

13. Понятие лингвистической переменной и термов.

14. Степень принадлежности для дискретной и непрерывной формы.

15.Понятие нечеткого правила.

16. Понятие фаззификации и методы фаззификации.

17. Понятие дефаззификации и методы дефаззификации.

18. Интерпретация «мягких вычислений».

2. ПРОЕКТИРОВАНИЕ НЕЧЕТКИХ СИСТЕМ

2.1. Проектирование нечетких систем в среде MatLab

Fuzzy Logic Toolbox – встроенная в Matlab совокупность функций, обеспечивающая набор средств, позволяющих создавать нечеткие системы:

− создавать и редактировать нечеткие системы внутри среды Matlab;

− встраивать нечеткую подсистему в Simulink (поставляется с Matlab) при моделировании общей системы;

− построить нечеткую систему в Matlab в виде процедуры, вызываемой из программы, написанной на языке Си.

Пакет позволят создавать экспертные системы на основе нечеткой логики, проводить кластеризацию нечеткими алгоритмами, а также проектировать нечеткие нейросети [1,3,4].

Fuzzy Logic Toolbox содержит следующие категории программных инструментов:

– функции;

– интерактивные модули с графическим пользовательским интерфейсом;

– блоки для пакета Simulink;

– демонстрационные примеры;

– Си-код машины нечеткого логического вывода.

Первая категория программных инструментов данного пакета содержит функции, которые могут быть вызваны непосредственно путем набора имени функции в командном окне (command line) или из собственных пользовательских приложений. Большинство из этих функций представляют собой матлабовские функции в виде m-файлов. В этом случае пользователь может посмотреть запрограммированные в этих функциях алгоритмы, а также редактировать и корректировать эти файлы.

Вторая категория программных инструментов пакета содержит диалоговые модули, которые обеспечивают доступ к большинству функций через графический интерфейс. Кроме того, эти модули обеспечивают удобную среду для проектирования, исследования и внедрения систем на основе нечеткого логического вывода.

Третья категория программных инструментов пакета Fuzzy Logic Toolbox содержит модули, которые обеспечивают интеграцию систем нечеткого логического вывода с пакетом Simulink.

Четвертая категория программных инструментов содержит демонстрационные примеры использования инструментов пакета Fuzzy Logic Toolbox.

В Fuzzy Logic Toolbox включены два файла fismain.c и fis.c, содержащие исходные коды на языке Си автономной машины нечеткого

логического вывода. Эти файлы позволяют загружать FIS-файл и файл исходных данных, а также выполнять нечеткий логический вывод. Кроме того, машина нечеткого логического вывода может быть встроена во внешние модули.

В MatLab заложены стандартные типы моделей систем нечеткой логики (Мамдани, Сугено), поддержка логических связок И, ИЛИ и НЕ в настраиваемых правилах, а так же множество встроенных функций принадлежности, в частности:

- сигмовидная;
- гауссова;
- колоколообразной формы;
- S-функция принадлежности;
- трапециевидная;
- треугольная и др.

Также в пакете MatLab есть ANFIS-редактор, который позволяет автоматически синтезировать из экспериментальных данных нейро-нечеткие сети. Редактор ANFIS позволяет создавать или загружать конкретную модель адаптивной системы нейро-нечеткого вывода, выполнять ее обучение, визуализировать структуру, изменять и настраивать ее параметры, использовать настроенную сеть для получения результатов нечеткого вывода.

2.2. Основные правила вывода в нечеткой логике

В традиционной логике решения об истинности одних суждений выносятся на основании истинности других суждений. Подобный вывод

задается в виде схемы: над горизонтальной чертой записываются все суждения, на основании которых принимается решение, а под чертой – полученный результат. Схема корректного вывода обладает тем свойством, что поскольку истинны все суждения над чертой, то истинно также и суждение под чертой.

Правило *modus ponens* определяется следующей схемой вывода:

Условие Импликация	A A → B
Вывод	B

Правило *modus tollens* определяется следующей схемой вывода:

Условие Импликация	~B A → B
Вывод	~A

Допустим, что присутствующие в правилах *modus ponens и modus tollens* суждения характеризуются некоторыми нечеткими множествами.

Нечеткое правило *modus ponens* определяется следующей схемой вывода:

Условие Импликация	x это A IF x это A THEN y это B
Вывод	y это B

Здесь A и B – нечеткие множества, а «x», «y» – лингвистические переменные.

Пример.

Условие Импликация	Скорость автомобиля большая Если скорость автомобиля очень большая, то уровень шума высокий
Вывод	Уровень шума в автомобиле не очень высокий

В приведенной схеме условие, импликация и вывод – неточные утверждения. В качестве лингвистических переменных использованы x– скорость автомобиля, y – уровень шума.

Нечеткое множество T1= {«малая», «средняя», «большая», «очень большая»} включает 4 терма лингвистической переменной x. Нечеткое множество T2= {«малый», «средний», «не очень высокий», «высокий»} включает 4 терма лингвистической переменной y.

Нечеткая импликация A→B равнозначна некоторому нечеткому отношению R⊆ $X \times Y$ с функцией принадлежности $\mu_R(x, y)$. В частном случае, когда T-норма имеет тип min, формула принимает вид:

$$\mu_B(x,y) = \sup_{x \subseteq X} \{ \mu_A( )^* \mu_{A \to B}(x,y) \}$$

Теперь допустим, что применяется импликация A→B по правилу *modus ponens,* а нечеткое множество A (условие) последовательно принимает значения:

A= «очень A», причем $\mu_{A'}(x) = \mu^2_A(x)$;

A= «почти A», причем $\mu_{A'}(x) = \mu^{1/2}_A(x)$;

A= «не A», причем $\mu_A(x) = 1 - \mu_{A'}(x)$.

Здесь нечеткое множество «очень A» определяется при помощи операции концентрации, нечеткое множество «почти A» – при помощи операции растяжения, а нечеткое множество «не A» – при помощи операции дополнения.

Однако, из предпосылки «x это не A» нельзя сделать вывод об y.

Правило *modus tollens* определяется следующей схемой вывода:

Условие Импликация	Не В А → В
Вывод	Не А

Здесь А и В – нечеткие множества, а «х», «у» – лингвистические переменные.

Условие Импликация	Уровень шума в автомобиле не очень высокий Если скорость автомобиля очень большая, то уровень шума высокий
Вывод	Скорость автомобиля не очень большая

Основой для проведения операции нечеткого логического вывода является база правил, содержащая нечеткие высказывания в форме «если-то» и функции принадлежности для соответствующих лингвистических термов. Базовое правило вывода типа «если – то» называется нечеткой импликацией, принимающей форму «если х это А, то у это В», где А и В – это лингвистические значения, идентифицированные нечетким способом через соответствующие функции принадлежности для переменных х и у. Импликацию записывают так же в сокращенном виде А→В.

На базу правил существуют следующие ограничения:

– существует хотя бы одно правило для каждого лингвистического терма выходной переменной;

– для любого терма входной переменной имеется хотя бы одно правило, в котором этот терм используется в качестве предпосылки (левая часть правила).

В противном случае имеет место неполная база нечетких правил.

В общем случае механизм логического вывода включает четыре этапа: введение нечеткости (фаззификация), нечеткий вывод, композиция и

приведение к четкости, или дефаззификация. На рис.2.1 представлена структурная схема системы нечеткого логического вывода.

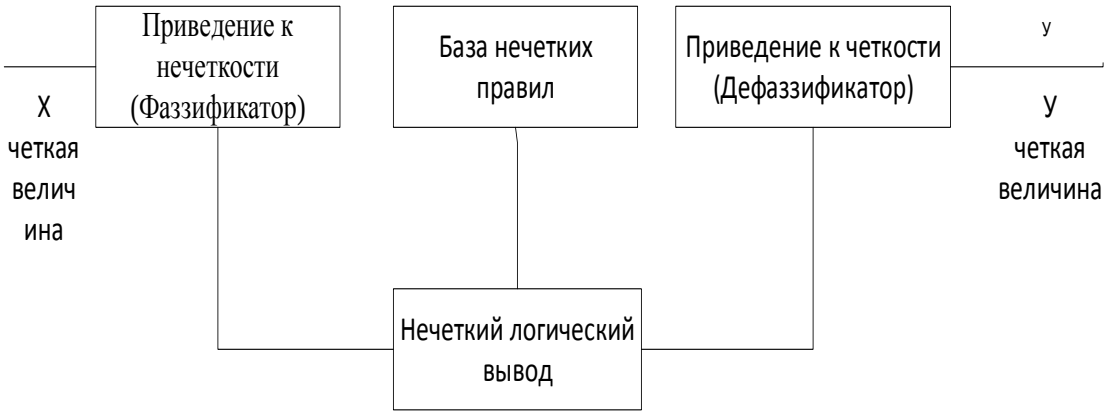


Рис.2.1. Структурная схема системы нечеткого логического вывода

База правил представляет собой набор нечетких «IF-THEN» конструкций, в которых условия и следствия подразумевают использование лингвистических переменных. Наиболее часто в прикладных нечетких системах используется формат правил с множеством входов и одним выходом.

Блок вывода представляет ядро системы нечеткой логики, используемое для моделирования приближенных рассуждений и процесса принятия решения человеком в сложных ситуациях.

Дефаззификатор трансформирует нечеткое множество в полностью детерминированное точечное решение.

Для реализации систем на базе нечетких правил разработано множество алгоритмов нечеткого вывода. Алгоритмы нечеткого вывода различаются главным образом видом используемых правил, логических операций и разновидностью метода дефаззификации.

2.3. Понятие нечеткой импликации

Представим различные способы нечеткой импликации.

1. *Правило типа minimum (правило Мамдани)*

$\mu_{A\to B}(x,y)= \mu_R(x,y)= \mu_A(x) \cap \mu_B(y)= \min[\mu_A(x), \mu_B(y)]$.

2. *Правило типа «произведение» (правило Ларсена)*

$\mu A_{\to B}(x,y)= \mu_R(x,y)= \mu_A(x) \cdot \mu_B(y)$.

3. *Правило Лукашевича*

$\mu_{A\to B}(x,y)= \mu_R(x,y)= 1 \cap [1- \mu_A(x) + \mu_B(y)] =\min[1, 1- \mu_A(x) + \mu_B(y)]$.

4. *Правило типа max-min (правило Заде)*

$\mu A_{\to B}(x,y)= \mu_R(x,y)=\max \{\min [\mu_A(x), \mu_B(y)], 1- \mu_A(x)\}$.

5. *Бинарное правило*

$\mu_{A\to B}(x,y)= \mu_R(x,y)=\max [1 -\mu_A(x), \mu_B(y)]$.

6. *Правило Гогуэна*

$\mu_{A\to B}(x,y)= \mu_R(x,y)=\min\{1, \frac{\mu B(y)}{\mu A(x)}\}$.

7. *Правило Шарпа*

$$\mu_{A\to B}(x,y)= \mu_R(x,y)= \begin{cases} 1, & \text{если } \mu A(x) \le \mu B(y)], \\ 0, & \text{если } \mu A(x) > \mu B(y)]. \end{cases}$$

8. *Правило Геделя*

$$\mu_{A\to B}(x,y)= \mu_R(x,y)= \begin{cases} 1, & \text{если } \mu A(x) \le \mu B(y)], \\ \mu B(y), & \text{если } \mu A(x) > \mu B(y)]. \end{cases}$$

9. *Вероятностное правило*

$\mu_{A\to B}(x,y)= \mu_R(x,y)= \min[1, 1- \mu_A(x) + \mu_A(x)\ \mu_B(y)]$.

10. *Правило ограниченной суммы*

$\mu_{A\to B}(x,y)= \mu_R(x,y)=\min\{1,[\mu_A(x) + \mu_B(y)]\}$.

В теории нечетких множеств разработан общий подход к выполнению операторов пересечения, объединения и дополнения, реализованный в так называемых треугольных нормах и конормах. Для

реализации операций пересечения и объединения – наиболее распространенные случаи треугольной нормы (t-нормы) и треугольной конормы (t-конормы или s– нормы) [1].

Треугольной нормой (t-нормой) называется бинарная операция ∩ на единичном интервале [0,1]× [0,1]→[0,1], удовлетворяющая следующим аксиомам для любых a,b,c ∈[0,1]:

- a ∩ 1 = a (граничное условие);
- a ∩ c ≥ a ∩ b, если c ≥ b (монотонность);
- a ∩ b = b ∩ a (коммуникативность);
- a ∩ (b ∩ c) = (a ∩ b) ∩ c (ассоциативность).

Часто используют следующие t-нормы: min(a,b) – пересечение по Заде; (a– b) – вероятностное пересечение(умножение); max(a + b– 1, 0) – пересечение по Лукашевичу.

Треугольной конормой (s-нормой) называется бинарная операция ∩ на единичном интервале [0,1]× [0,1]→[0,1], удовлетворяющая следующим аксиомам для любых a,b,c ∈[0,1]:

- a ∪ 0 = a (граничное условие);
- a ∪ c ≥ a ∪ b, если c ≥ b (монотонность);
- a ∪ b = b ∪ a (коммуникативность);
- a ∪ (b ∪ c) = (a ∪ b) ∪ c (ассоциативность).

Часто используют следующие t-нормы: max(a,b) – объединение по Заде; (a+b– ab) – вероятностное ИЛИ; $min(a$ + b, 1)0) – j,] – объединение по Лукашевичу.

2.4. Алгоритм нечеткой модели Мамдани

Алгоритм нечеткого вывода Мамдани – это наиболее распространенный способ логического вывода в нечетких системах. В нем

используется минимаксная композиция нечетких множеств. Данный механизм включает в себя следующую последовательность действий:

1. Процедура фаззификации: определяются степени истинности, т.е. значения функций принадлежности для левых частей каждого правила (предпосылок). Для базы правил с m правилами обозначим степени истинности как $A_{ik}(x_k)$, i=1, 2,..., m, k=1, 2,..., n.

2. Нечеткий вывод. Сначала определяются уровни "отсечения" для левой части каждого из правил: $alfa_i = \min_i (A_{ik}(x_k))$.

3. Далее находятся "усеченные" функции принадлежности:

$B_i^*(y) = \min_i (alfa_i, B_i(y))$.

4. Композиция, или объединение полученных усеченных функций, для чего используется композиция нечетких множеств по критерию максимума: $\mu_F(y) = \max_i (B_i^*(y))$.

где $\mu_F(y)$ – функция принадлежности итогового нечеткого множества.

5. Дефаззификация, или приведение к четкости, например, методом среднего центра, или центроидным методом.

Фаззификатор выполняет функцию преобразования четких значений x входных переменных в нечеткое значение, выраженное степенью принадлежности каждого терма лингвистической переменной.

Обобщенная функциональная структура системы, приведенная на рис.2.1, может быть представлена в расширенной форме, которая в явном виде демонстрирует правила нечеткого вывода так, как это изображено на рис.2.2.

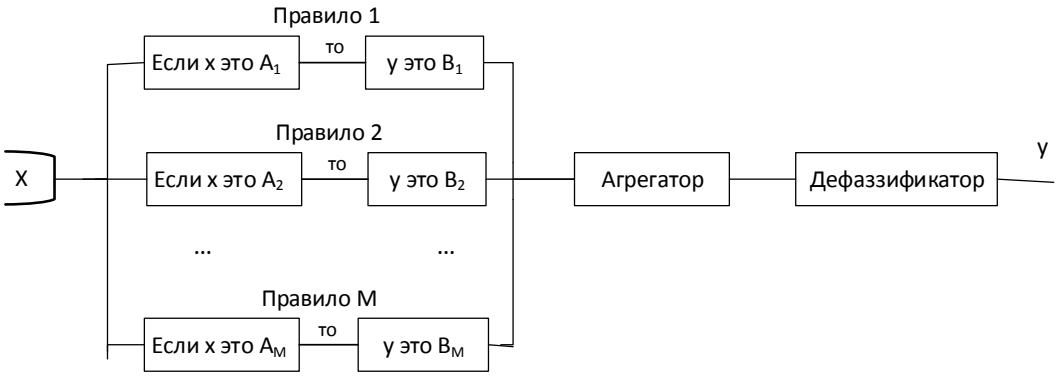


Рис.2.2 – Организация вывода в нечеткой системе при наличии M правил вывода

Поскольку допускается применение множества нечетких правил, в ней также предусмотрен блок агрегирования, чаще всего реализуемый в виде логического сумматора (оператор Max). Как правило, в модели присутствуют следующие операторы:

- оператор логического или арифметического произведения для определения результирующего уровня активации, в котором учитываются все компоненты вектора *x* условия;

- оператор логического или арифметического произведения для определения значения функции принадлежности для всей импликации A→B*;*

- оператор логической суммы как агрегатор равнозначных результатов импликации многих правил;

- оператор дефаззификации, трансформирующий нечеткий результат $\mu(y)$ в четкое значение выходной переменной *y.*

На рис.2.3 графически приведена иллюстрация процесс нечеткого вывода по Мамдани для двух входных переменных и двух нечетких правил.

Способ агрегирования двух правил нечеткого вывода при существовании двух значений переменных x_1 и x_2. Логическое произведение (оператор Min) используется как для агрегирования нечетких правил

относительно конкретных переменных x_i (i = 1, 2), образующих вектор X, так и на уровне импликации A→B для одиночных правил вывода.

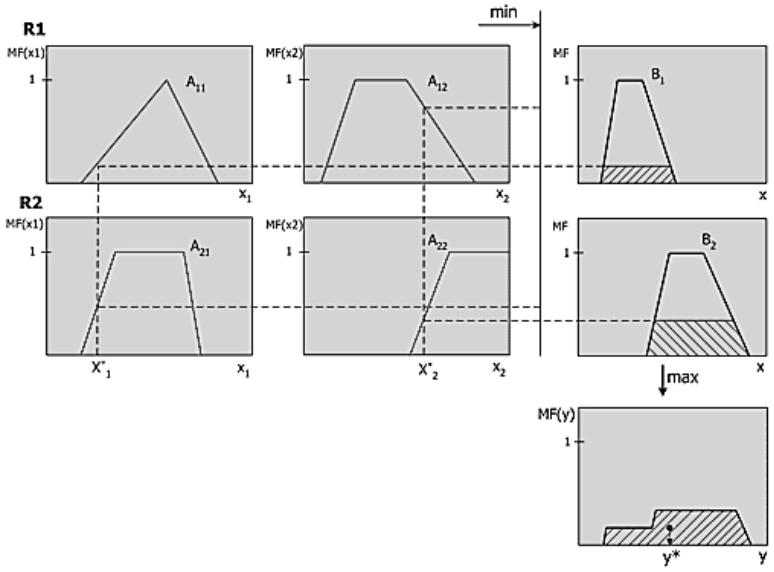


Рис.2.3. Иллюстрация нечеткого вывода по алгоритму Мамдани

Агрегирование импликаций, касающихся правил 1 и 2, проводится с использованием логической суммы (оператор Max). В правой нижней части рисунка представлен нечеткий результат в виде функции принадлежности переменной *y*. Получение четкого значения *y*, соответствующего четким значениям входных переменных x_1 и x_2, требует применения процедуры дефаззификации. Геометрический смысл такого значения – центр «тяжести» для кривой $\mu_F(y)$.

2.5. Алгоритм нечеткой модели Сугено

Алгоритм нечеткого вывода Сугено предлагает замену дефаззификации на вычисление значения функции.

Сугено использовал набор правил в следующей форме:

$П_1$: если *x* есть A_1 и *y* есть B_1, то $z_1 = a_1 x + b_1 y$,

$П_2$: если *x* есть A_2 и *y* есть B_2, то $z_2 = a_2 x + b_2 y$.

Алгоритм Сугено включает следующие шаги (рис.2.4):

1. Введение нечеткости как в алгоритме Мамдани.

2. Нечеткий вывод. Находятся уровни отсечения для предпосылок каждого правила – как в алгоритме Мамдани и индивидуальные выходы правил по формулам:

$z_1^*=a_1x_0+b_1y_0$,

$z_2^*=a_2x_0+b_2y_0$.

3. Определяется четкое значение переменной вывода:

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

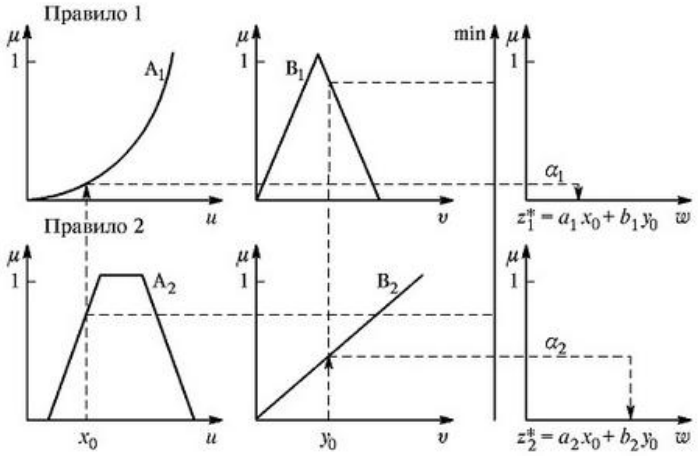


Рис.2.4. Схема нечеткого вывода по алгоритму Сугено

2.6. Пример проектирования системы нечёткого вывода типа Мамдани в Fuzzy Logic Toolbox

1. Создание системы нечёткого вывода типа Мамдани, которая моделирует зависимость $y=x_1^2 \sin(x_2-1)$ в области $x_1\in[-7\ 3]$, $x_2\in[-4{,}4\ 1{,}7]$.

2. Проектирование проводится на основе трёхмерного изображения указанной зависимости

% Построение графика функции y=x1^2 * sin(x2-1)

```
% в области x1∈[-7  3], x2∈[-4,4 1,7].
n=15; % количество точек дискретизации
x1=linspace (-7 3 n);  x2=linspace (-4,4  1,7  n);
y=zeros(n,n);
for j=1:n
y(j, :)=x1.^2*sin(x2(j)-1);
end
surf(x1,x2,y)
xlabel(`x_1`);   ylabel(`x_2`); zlabel(`y`);
title(`искомая зависимость`)
```

3. Задание 7 нечётких правил

Если X1= «низкий» И X2= «низкий», ТО y= «высокий»;

Если X1= «низкий» И X2= «средний», ТО y= «низкий»;

Если X1= «низкий» И X2= «высокий», ТО y= «высокий»;

Если X1= «средний», ТО y= «средний»;

Если X1= «высокий» И X2= «низкий», ТО y= «выше среднего»

Если X1= «высокий» И X2= «средний», ТО y= «ниже среднего»

Если X1= «высокий» И X2= «высокий», ТО y= «выше среднего»

4. Шаги проектирования нечёткой системы

Шаг1. Ввести слово fuzzy в командной строке. Появится окно рис.2.5.

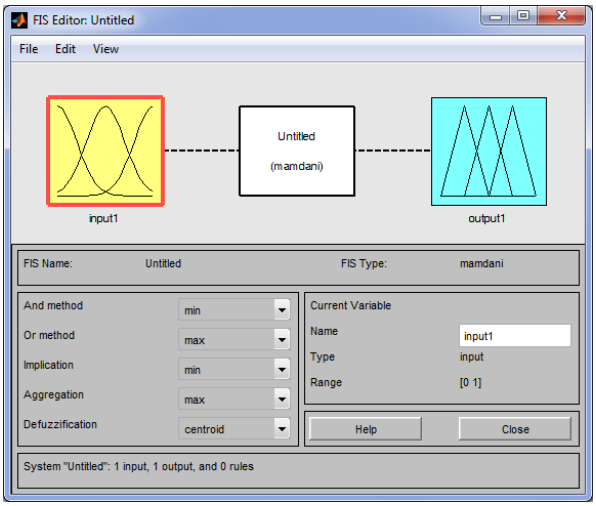


Рис.2.5. Основное окно проектирования нечеткой системы

Шаг2. Добавить вторую входную переменную, для чего в меню Edit выбрать команду Add input.

Шаг 3-5. Переименовать входные переменные x1,x2 и выходную переменную y (рис.2.6).

Шаг6. Сохранить, выбрав в меню File команду Export To File.

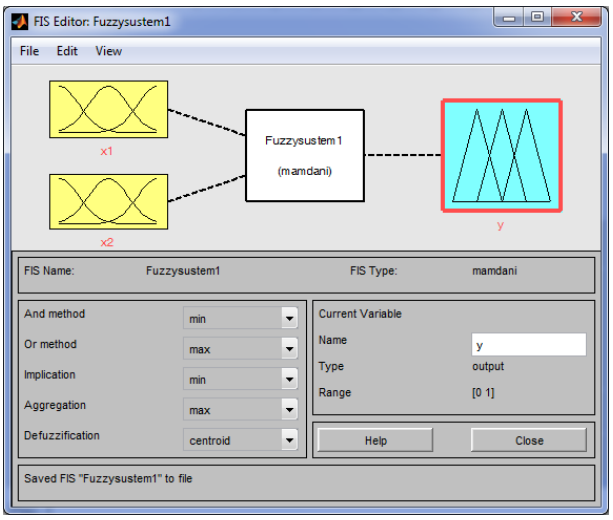


Рис.2.6. Основное окно проектирования нечеткой системы

с двумя переменными

Шаг7. Переход в редактор функций принадлежности, для чего сделать двойной щелчок на блоке X1.

Шаг8. Задать диапазон изменения переменной X1, напечатав -7 3 в поле Range.

Шаг 9. Задать функции принадлежности переменной X1. Для лингвистической оценки этой переменной будем использовать три терма с треугольными функциями принадлежности. Эти функции установлены по умолчанию.

Шаг10-12. Задать наименования термов переменной X1. Для этого щелкнем по графику первой функции принадлежности. График активированной функции изображается жирной красной линией. Затем вводится наименование терма НИЗКИЙ в поле Name.Аналогично для других термов переменной X1 и X2.

Шаг13. Задать функции принадлежности переменной y. Для лингвистической оценки этой переменной будем использовать 5 термов с функцией Гаусса.

Для этого активизировать переменную y, щёлкнув по блоку y. Задать диапазон изменения y от -50 до 50 в поле Range и нажать Enter. Затем в меню Edit выбрать команду All MFs для удаления установленных по умолчанию функций принадлежности.

Затем в меню Edit выбрать команду Add MFs и в появившемся окне выбрать функцию gaussmf в поле MF type и 5 термов в поле Number of MFs. После этого активировать функцию y, щёлкнув по графику.

Шаг14. По аналогии с шагом 10 задать 5 термов: низкий, средний, ниже среднего, выше среднего, высокий (рис.2.7).

Шаг 15. Перейти в редактор базы знаний RuleEditor, для чего в меню Edit выбрать команду Rules.

Шаг16. Для ввода правила выбирать соответствующую комбинацию термов и нажать кнопку Add rule (рис.2.8).

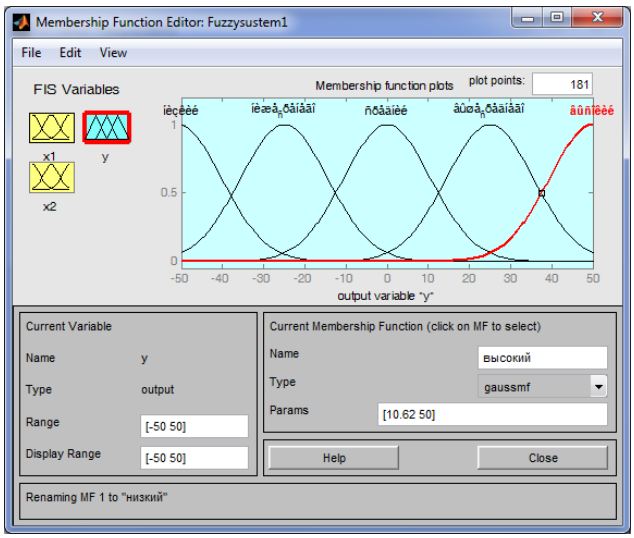


Рис.2.8. Иллюстрация 5 термов

Шаг17. Сохранить созданную систему. Для этого в меню File выбирать команду Export To File.

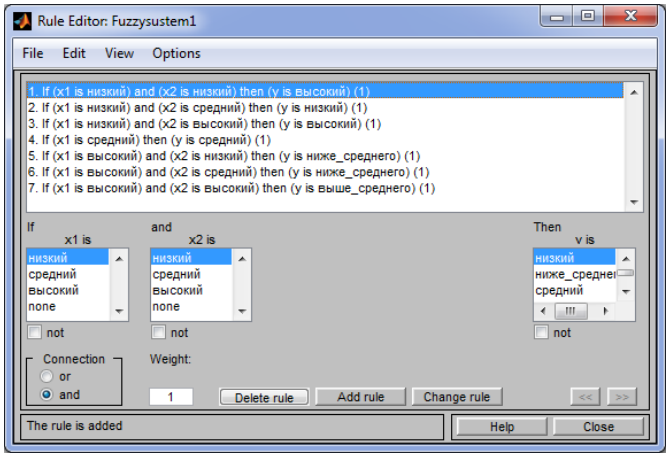


Рис.2.9. Иллюстрация выбора термов

Шаг 18. В меню View выбрать команду Rules (рис.2.10).

Шаг19. В поле input указывается значение входных переменных, для которых производится нечёткий логический вывод. Поверхность «входы-выходы» выводится по команде Surface в меню View (рис.2.11).

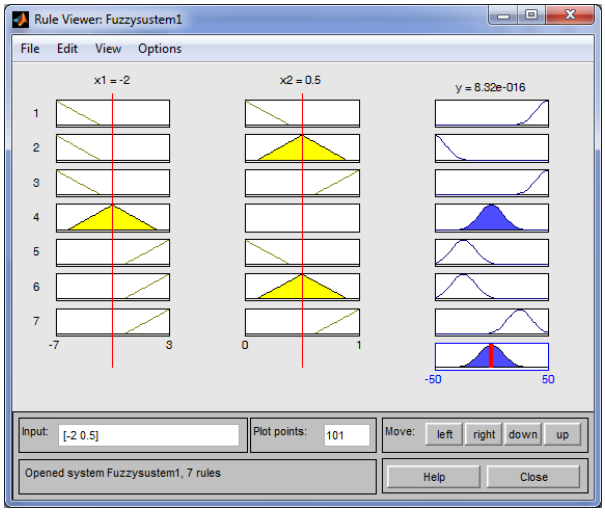


Рис.2.10. Иллюстрация выбора команды Rules

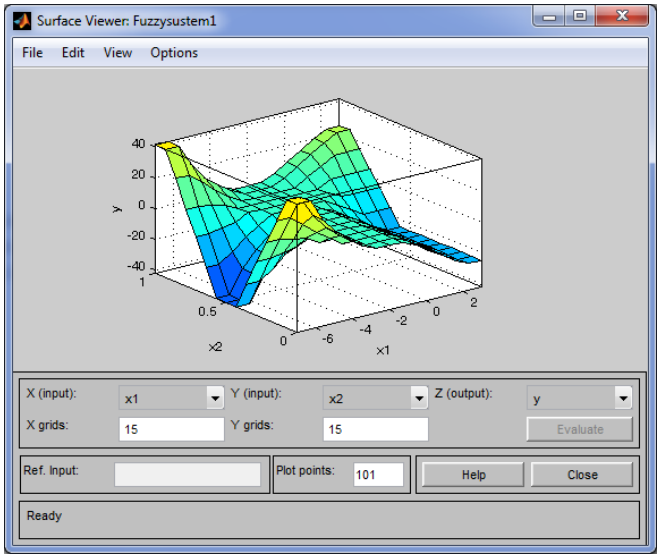


Рис.2.11. Иллюстрация выбора команды Surface

2.7. Пример проектирования системы нечёткого вывода типа Сугено в Fuzzy Logic Toolbox

Создание системы нечёткого вывода типа Сугено, которая моделирует зависимость $y=x_1^2 \sin(x_2-1)$ в области $x_1 \in [-7\ \ 3]$, $x_2 \in [-4,4\ \ 1,7]$.

Проектирование проводится на основе трёхмерного изображения указанной зависимости

```
% Построение графика функции y=x1^2 * sin(x2-1)
% в области x1∈[-7 3], x2∈[-4,4 1,7].
n=15; % количество точек дискретизации
x1=linspace (-7 3 n);  x2=linspace (-4,4  1,7  n);
y=zeros(n,n);
for j=1:n
y(j, :)=x1.^2*sin(x2(j)-1);
end
surf(x1,x2,y)
xlabel(`x_1`);   ylabel(`x_2`); zlabel(`y`);
title(`искомая зависимость`)
```

Задание семи нечетких правил

Если X1= «низкий» И X2= «низкий», ТО y= 50;

Если X1= «низкий» И X2= «средний», ТО y=4x1-x2;

Если X1= «низкий» И X2= «высокий», ТО y=50;

Если X1= «средний» , ТО y= 0;

Если X1= «высокий» И X2= «низкий», ТО y=2x1-2x2-3;

Если X1= «высокий» И X2= «высокий», ТО y=2 *x1+2* x2+1

Шаги проектирования нечеткой системы Сугено.

Шаг1 Ввести слово fuzzy в командной строке. Появится главное окно проектирования нечеткой системы Сугено (рис.2.12).

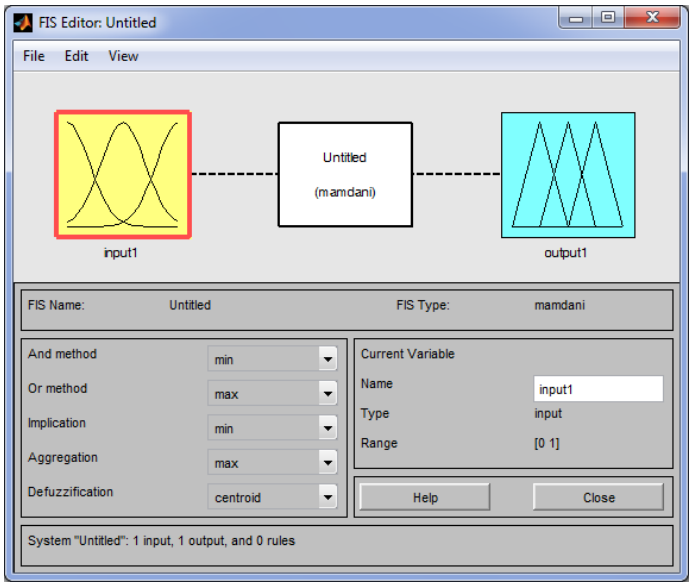


Рис.2.12. Главное окно проектирования нечеткой системы Сугено

Шаг 2. В меню File выбрать команду New Fis… Sugeno.

Шаг3. Добавим входную переменную x2, для чего в меню Edit выбрать команду Add input.

Шаг4. Переименуем входы и выход системы.

Шаг5. Сохраним нечёткую систему. В меню File выбрать команду Export To File.

Шаг6. Зададим терм-множество и функции принадлежности входных переменных.

Шаг7. Переход в редактор функций принадлежности, для чего сделать двойной щелчок на блоке X1.

Шаг8. Задать диапазон изменения переменной X1, напечатав -7 3 в поле Range.

Шаг 9. Задать функции принадлежности переменной X1. Для лингвистической оценки этой переменной будем использовать три терма с

треугольными функциями принадлежности. Эти функции установлены по умолчанию.

Шаг10-12. Задать наименования термов переменной X1. Для этого щелкнем по графику первой функции принадлежности. График активированной функции изображается жирной красной линией. Затем вводится наименование терма НИЗКИЙ в поле Name.Аналогично для других термов переменной X1 и X2.

Шаг13. Задать заключения правил, для чего:

- активизировать переменную y, щелкнув по блоку «y»;
- в правом верхнем углу появилось обозначение трёх функций принадлежности, каждая из которых соответствует одной линейной зависимости между входами и выходом.

В базе Сугено указаны 5 разных зависимостей (рис.2.13). Поэтому добавляем ещё 2 заключения правил, выбрав в меню Edit выбрать команду Add Mfs, а в окне Number of Mfs значение 2 и нажать ОК.

Шаг14. Зададим наименования и параметры линейных зависимостей (мышкой по mf1 и ввести наименование в поле Name). Например, 50 и тип зависимости Constant в менюType, а в поле Params вводим значение 50.

Аналогично для второго mf2 введем наименование 4*x1-x2, укажем линейный тип зависимости через опцию Linear в меню Type и введем параметры зависимости 4 -1 0 в поле Params.

Для линейной зависимости порядок, следующий: первый параметр – коэффициент при первой переменной; второй параметр –коэффициент при второй переменной; последний параметр-свободный член зависимости.

Для третьего заключения mf3 введем наименование, например, 0, укажем тип зависимости- константа и введем параметр 0.

Для четвертого заключения mf4 введем наименование, например, 2*x1-2*x2-3, укажем линейный тип зависимости и введем параметры 2 -2 -3.

Для пятого заключения mf5 введем наименование, например, 2*x1+2*x2+1, укажем линейный тип зависимости и введем параметры 2 2 1.

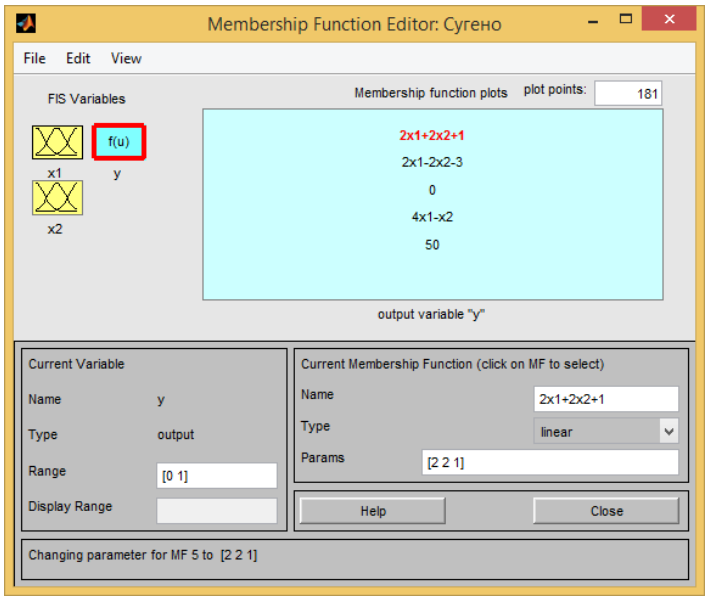


Рис.2.13. Заключения правил Сугено

Шаг 15.Для ввода правил выбираем соответствующую комбинацию, выбрав команду Add rule в меню Edit.

В меню View выбираем команду Rules (рис.2.14)

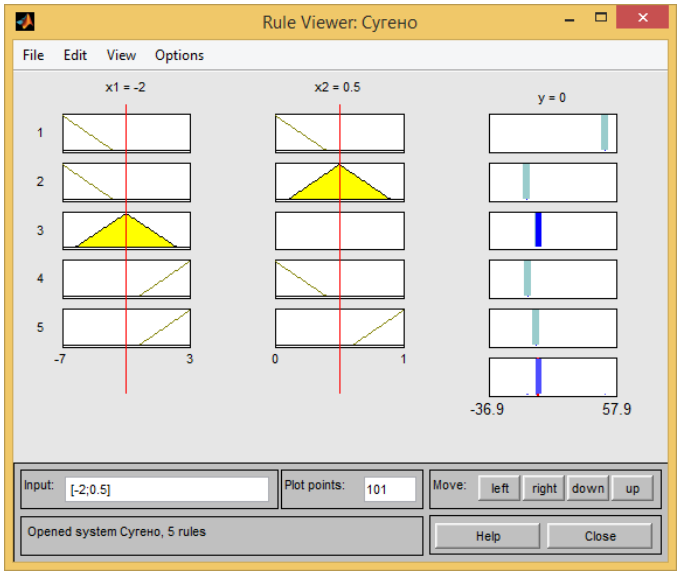


Рис.2.14. Иллюстрация выбора команды Rules в меню View

Изменяя значения x1 и x2 (положение красной черты) получаем новые значения y.

Шаг 17. В меню View выбираем команду Surface. Получаем поверхность «входы-выходы» (рис.2.15)

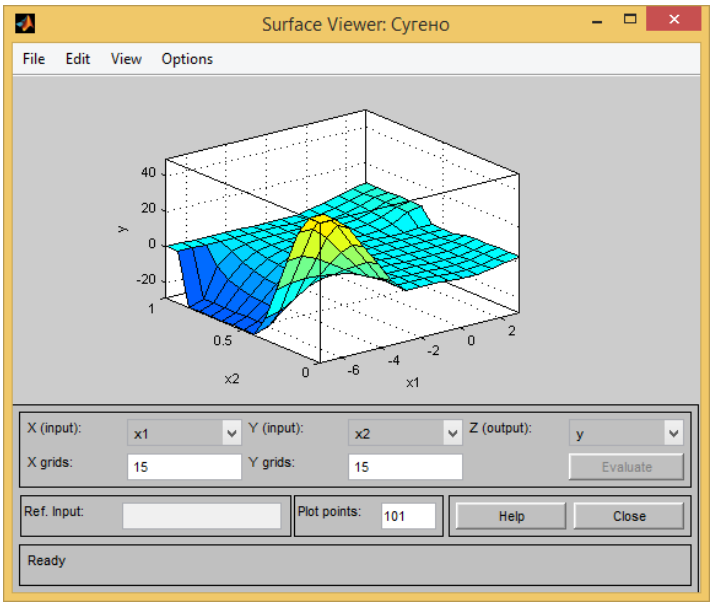


Рис.2.15. Иллюстрация выбора команды Surface

2.8. Изменение параметров нечеткой системы Сугено

Для одного из тестовых примеров разработано 9 правил нечеткой системы, максимально приближенной к реальной, поочередно изменялись функция принадлежности переменных, а также методы И, ИЛИ и методы дефаззификации, при варьировании которых оценивалась ошибка аппроксимации.

Рассмотрено одиннадцать функций принадлежности переменных:

– *trimf* (задает функцию принадлежности в форме треугольника);

– *trapmf* (применяется для задания ассиметричных функций принадлежности переменных, наиболее возможные значения которых принимаются на некотором интервале);

– *gbellmf* и *gaussmf* (применяются для задания гладких симметричных функций принадлежности);

– *gauss2mf*, *dsigmf* и *psigmf* (применяются для задания гладких ассиметричных функций принадлежности);

– *sigmf* (применяется для задания монотонных функций принадлежности);

– *pimf* (применяется для задания ассиметричных функций принадлежности с плавным переходом от пессимистической к оптимистической оценки нечеткого числа);

– *smf* (применяется для задания неубывающих функций принадлежности с насыщением);

– *zmf* (применяется для задания невозрастающих функций принадлежности с насыщением).

Было установлено, что наибольшей точностью обладает функция *gaussmf*.

При изменении методов связки *AND (prod),* для метода *AND (min) E* было установлено, что наибольшей точностью для данного примера обладает метод *AND (min),* который дает минимальную ошибку аппроксимации.

При изменении методов связки OR (max), OR (probor) были получены одинаковые ошибки аппроксимации, так как среди разработанных нечетких правил нет ни одного, где связкой между переменными служила бы операция ИЛИ, то есть изменение метода OR никак не влияет на спроектированную систему нечеткого вывода.

При изменении метода Deffuzzification было установлено,что наибольшей точностью обладает метод Deffuzzification (wtaver)– взвешенное среднее.

Таким образом, построенння нечеткая система типа Сугено по сравнению с нечеткой системой Мамдани обладает наименьшей ошибкой аппроксимации и алгоритм Сугено более точен, чем алгоритм Мамдани для данного примера.

Контрольные вопросы

1. В чем главное отличие нечеткой системы Сугено от нечеткой системы Мамдани?
2. Какие этапы алгоритмов Мамдани и Сугено совпадают?
3. Какие методы дефаззификации применяются в нечеткой системе Мамдани и Сугено?
4. Что такое t– норма?
5. Что такое s– норма?
6. Формула вероятностного ИЛИ.
7. Сравнить нечеткие базы знаний Мамдани и Сугено.
8. Взвешенное среднее (центр тяжести) как результат дефаззификации.
9. Взвешенная сумма как результат дефаззификации.
10. Алгоритм Мамдани.
11. Методы дефаззификации в алгоритме Мамдани.

12. Понятие нечеткой импликации.

13. Методы нечеткой импликации.

14. Нечеткое правило modus ponens.

15. Нечеткое правило modus tollens.

3. ПРОЕКТИРОВАНИЕ ГИБРИДНОЙ НЕЙРО-НЕЧЕТКОЙ МОДЕЛИ КЛАССА ANFIS

3.1. Основы нейросетевых нечетких систем

Нейросетевой нечеткой системой называется такая система, в которой отдельные элементы нечеткости (функции принадлежности, логические операторы, отношения) и алгоритмы вывода реализуются с помощью нейронной сети.

Основное преимущество нечетких систем в отличие от НС:

1. Знания в этих системах представляются в форме легко понимаемых человеком гибких логических конструкций, таких, как

   «IF … - THEN …».

2. Б. Коско (1993) Теорема: «любая математическая функция может быть аппроксимирована системой, основанной на нечеткой логике, следовательно, такие системы являются универсальными».

Основное преимущество нейросетевого подхода состоит в следующем. С помощью НС имеется возможность выявления закономерностей в данных, их обобщение, т.е. извлечение знаний из данных, а основной недостаток– невозможность непосредственно (в явном виде, а не в виде вектора весовых коэффициентов межнейронных связей) представить функциональную зависимость между входом и выходом исследуемого объекта.

Наглядно нейро-сетевую реализацию демонстрирует следующий пример. Рассмотрим возможность построения функций принадлежности

для трех значений некоторой лингвистической переменной: «мало (S-Small)», «средне (M-Medium)», «много (L-Large)». Нейросетевая реализация функций принадлежности приведена на рис.3.1, а график функций принадлежности – на рис.3.2.

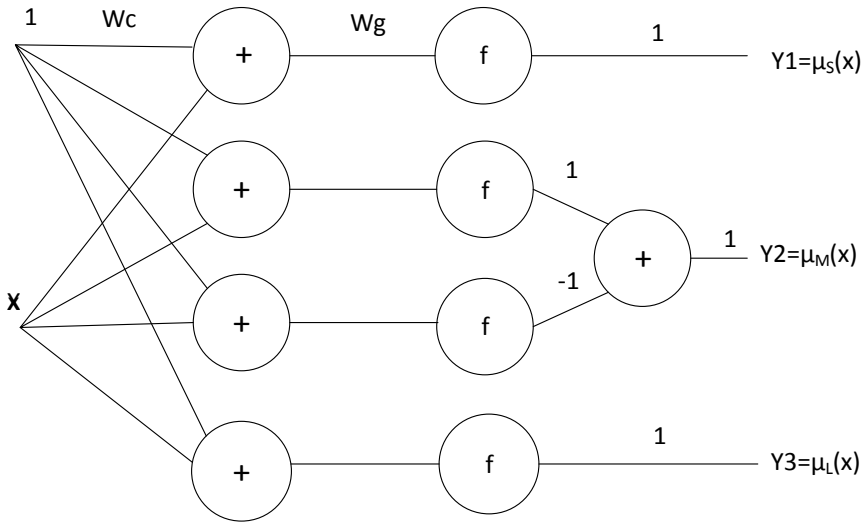


Рис.3.1. Нейросетевая реализация функций принадлежности

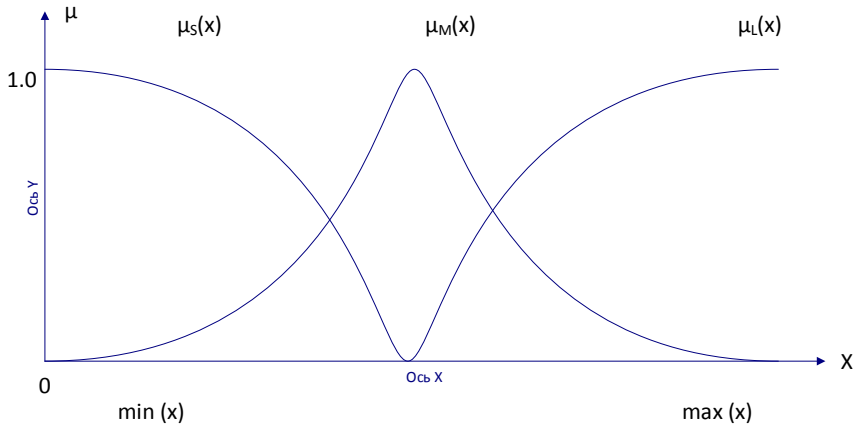


Рис.3.2. График функций принадлежности

На рис.3.1 параметр wc – весовой коэффициент смещения и w_g – вес суммарного сигнала на входе нелинейного преобразователя. Путем подстройки весовых коэффициентов формируется функция

принадлежности. Выходы y_1, y_2, y_3 определяют величины соответствующих функций принадлежности –$\mu_S(x)$, $\mu_M(x)$, $\mu_L(x)$. Узлы со знаком «+» суммируют сигналы входов нейронов, а узлы с символом f реализуют сигмовидные функции.

Реализация нечетких И (AND), ИЛИ (OR) и других операторов в нейросетевом логическом базисе дает основу для построения нейросетевых нечетких моделей, т.е. можно задать функцию активации, реализующую min-оператор для нечеткой операции AND, max-оператор для нечеткой операции OR. «Мягкий» (softmin) оператор может быть использован для замены оригинального нечеткого оператора И:

$$(a \cap b) = softmin(a, b) = \frac{ae^{-ka} + be^{-kb}}{e^{-a} + e^{-b}}.$$

Две главные проблемы, возникающие при организации вывода на основе нечеткой логики, которые более эффективно можно реализовать на основе нейросетевого подхода (рис.3.3):

1. Первая проблема связана с определением функций принадлежности, использующихся в условной части правил.
2. Вторая проблема – с выбором одного правила, определяющего решение, из совокупности правил.

Правила нечеткого вывода имеют формат

Rs: IF X=(x1, x2,…,xn) is As THEN y_s=HCs(x1,x2,…,xn).

Условная часть правила IF использует r правил вывода (s=1, 2,…,r). As- нечеткое множество условной части каждого правила. HCs-структура нейросетевой модели (многослойный персептрон) со входами x1,x2,…,xn и выходом y_s.

Для каждого правила своя НС. Для определения функций принадлежности условной части правила используется НСs(μ) – многослойный персептрон. Нейронная сеть НСs(μ) обучается на входных данных обучающей выборки.

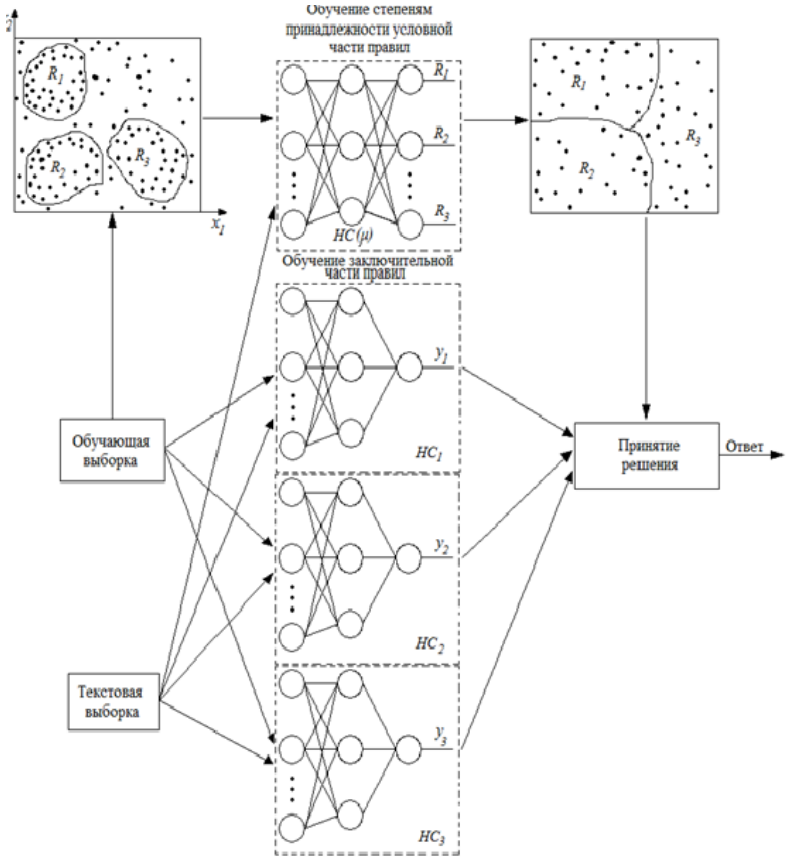


Рис.3.3. Иллюстрация нейросетевого подхода

Выходы обученной НС рассматриваются как величины функций принадлежности нечетких множеств в условной части IF правил, т.к. величина выходного сигнала определяет принадлежность данных к каждому правилу.

Алгоритм процедуры нечеткого вывода на основе нейросетевой модели:

1. Формирование обучающей No и тестовой N_T выборки; N= No + N_T – общее число примеров из базы данных.

2. Кластеризация обучающей выборки. Обучающая выборка делится на r классов Rs (по числу правил), где s=1, 2,…, r. Каждая обучающая подвыборка для класса Rs определяется парой (x^s_i,y^s_i), где i=1,2,…,Ns, а Ns– число примеров в обучающей выборке для класса Rs. Разделение n–мерного входного пространства на r в данном случае означает, что число правил вывода равно r.

3. Обучение HCs(μ). Для каждого входного вектора X определим вектор функций принадлежности к правилу Mi такой, что ms i=1 и mk i=0 для k≠s . Нейронная сеть HCs(μ) с n входами и r выходами обучается на парах (Xi, Mi), следовательно, после обучения и тестирования такая сеть будет способна определить степень принадлежности для каждого входного вектора, принадлежащего к классу Rs. Таким образом, функция принадлежности к части IF правила определяется как выходная величина обученной НС.

4. На четвертом шаге алгоритма процедуры нечеткого вывода на основе нейросетевой модели – это обучение HCs. Обучающая выборка с входами xi1s, xi2s,…, xins и выходной величиной yi; i=1,2,…,Ns подается на вход и выход HCs, которая является нейросетевой моделью части THEN в Rs. С помощью тестовой выборки вычисляется ошибка обобщения:

Esn=$\sum_{i=1}^{NT}\{yi - ms(Xi)mAs(Xi)\}$^2,

где ms(Xi) – наблюдаемый выход HCs. Если Es< δ, где δ – априори заданная величина, HCs – обучена.

5. Пятый шаг алгоритма процедуры нечеткого вывода — это принятие решения на основе нейросетевой модели. Для заданного входного

вектора производится вычисление выходной величины по аналогии с формулой дефаззификации:

$$Yt^*=\frac{\sum_{s=1}^{r} mAs(Xi)ms(Xi)}{\sum_{s=1}^{r} mAs(Xi)}.$$

3.2. Нейро-нечеткая сеть Такаги-Сугено-Канга

Наибольшую популярность среди нечетких систем адаптивного типа приобрела модель вывода Такаги-Сугено-Канга (TSK).

Обобщенную схему вывода модели TSK при использовании M правил и N переменных xj можно представить в виде

IF (x_1 IS $A_1(1)$) AND (x_2 IS $A_2(1)$),..., AND (x_n IS $A_n(1)$),

THEN y1 = p10 +$\sum_{j=1}^{N} p1jxj$

… … …

IF (x_1 IS $A_1(M)$) AND (x_2 IS $A_2(M)$),..., AND (x_n IS $A_n(M)$),

THEN y1 = p10 +$\sum_{j=1}^{N} pMjxj$

Условие IF (xi IS Ai) реализуется функцией фаззификации, которая представляется обобщенной функцией Гаусса отдельно для каждой переменной xi:

$$\mu_A(x_i)=\frac{1}{1+\left(\frac{x_i-c_i}{\sigma_i}\right)^{2b_i}}, \qquad (3.1)$$

где μ_A (xi) представляет оператор A_i. В нечетких сетях целесообразно задавать это условие в форме алгебраического произведения, из которого следует, что для k-го правила вывода

$$\mu_A^{(k)}(x) = \prod_{j=1}^{N} \left[\frac{1}{1 + \left(\frac{x_i - c_j^{(k)}}{\sigma_j^{(k)}} \right)^{2b_j^{(k)}}} \right]. \qquad (3.2)$$

При M правилах вывода агрегирование выходного результата сети производится по формуле

$$y = \sum_{i=1}^{M} \frac{w_i}{\sum_{j=1}^{N} w_j} \left(p_{i0} + \sum_{j=1}^{N} p_{ij} x_j \right), \qquad (3.3)$$

которую можно представить в виде

$$y(x) = \frac{1}{\sum_{k=1}^{N} w_k} \sum_{k=1}^{M} w_k y_k(x), \qquad (3.4)$$

где $yk(x) = pk0 + \sum_{j=1}^{N} pkjxj$. Присутствующие в этом выражении веса wk интерпретируются как значимость компонентов μ^k_A(xi), определенных формулой (3.1). При этом условии формуле (3.4) можно сопоставить многослойную структуру сети (рис.3.3), состоящей из пяти слоев.

Первый слой выполняет раздельную фаззификацию каждой переменной xi (i=1,2, … , N), определяя для каждого k-го правила вывода значение коэффициента принадлежности $\mu_A^{(k)}(x_i)$ в соответствии с применяемой функцией фаззификации. Это параметрический слой с параметрами cj(k), $\sigma_j^{(k)}$, bj(k), подлежащими адаптации в процессе обучения.

Второй слой выполняет агрегирование отдельных переменных xi, определяя результирующее значение коэффициента принадлежности для вектора x (уровень активизации правила вывода) в соответствии с формулой (3.2). Это слой непараметрический.

Третий слой представляет собой генератор функции TSK, вычисляющий значения где $yk(x) = pk0 + \sum_{j=1}^{N} pkjxj.$

В этом слое также производится умножение сигналов yk(x) на значения wk, сформированные на предыдущем слое. Это параметрический слой, в котором адаптации подлежат линейные веса pkj для k = 1,2, … ,M и j = 1,2, … ,N, определяющие функцию следствия модели TSK.

Четвертый слой составляют два нейрона-сумматора, один из которых рассчитывает взвешенную сумму сигналов yk(x), а второй определяет сумму весов $\sum_{k=1}^{M} wk$. Это непараметрический слой.

Последний, пятый слой, состоящий из единственного выходного нейрона,- это нормализующий слой, в котором веса подвергаются нормализации в соответствии с формулой (3.4). Выходной сигнал y(x) определяется выражением, соответствующим зависимости (3.5):

$$y(x) = f(x) = \frac{f_1}{f_2}. \qquad (3.5)$$

Из приведенного описания следует, что нечеткая сеть TSK содержит только два параметрических слоя (первый и третий), параметры которых уточняются в процессе обучения. Параметры первого слоя будем называть нелинейными параметрами, поскольку они относятся к нелинейной функции (2.1), а параметры третьего слоя – линейными весами, т.к. они относятся к параметрам pkj линейной функции TSK.

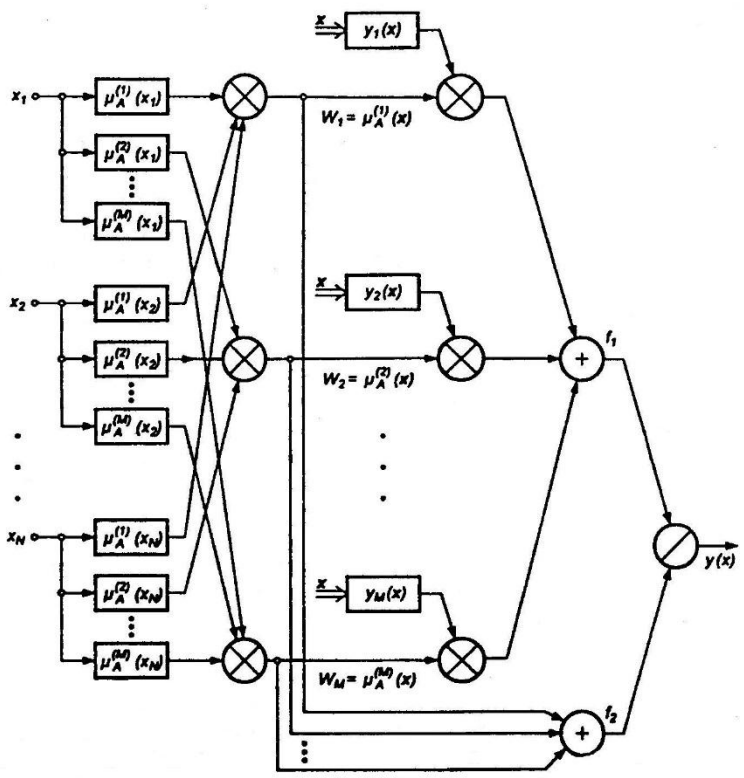


Рис. 3.3. Структура нечеткой нейронной сети TSK

При уточнении функциональной зависимости (3.4) для сети TSK получаем:

$$y(x) = \frac{1}{\sum_{k=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(k)}(x_j)\right]}\sum_{k=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(k)}(x_j)\right]\left[p_{k0}+\sum_{j=1}^{N}p_{kj}x_j\right] . \qquad (3.6)$$

Если принять, что в конкретный момент времени параметры условия зафиксированы, то функция y(x) является линейной относительно переменных xi (i=1,2, … , N).

При наличии N входных переменных каждое правило формирует N+1 переменных pj(k) линейной зависимости TSK. При M правилах вывода это дает M(N+1) линейных параметров сети. В свою очередь, каждая функция принадлежности использует три параметра (c, s, b), подлежащих адаптации. Если принять, что каждая переменная xi характеризуется собственной функцией принадлежности, то при M правилах вывода мы получим 3MN нелинейных параметров. В сумме это дает M(4N+1) линейных и нелинейных параметров, значения которых должны подбираться в процессе обучения сети.

3.3. Гибридная сеть как адаптивная система нейро-нечеткого вывода

Гибридная сеть представляет собой многослойную нейронную сеть специальной структуры без обратных связей, в которой используются обычные (не нечеткие) сигналы, веса и функции активации, а выполнение операции суммирования основано на использовании фиксированной T-нормы, T-конормы или некоторой другой непрерывной операции. При этом значения входов, выходов и весов гибридной нейронной сети представляют собой вещественные числа из отрезка [0, 1].

Основная идея, положенная в основу модели гибридных сетей, заключается в том, чтобы использовать существующую выборку данных для определения параметров функций принадлежности, которые лучше всего соответствуют некоторой системе нечеткого вывода. При этом для

нахождения параметров функций принадлежности используются известные процедуры обучения нейронных сетей.

В пакете Fuzzy Logic Toolbox системы MATLAB[3] гибридные сети реализованы в форме так называемой адаптивной системы нейро-нечеткого вывода ANFIS. ANFIS является аббревиатурой Adaptive Neuro-Fuzzy Inference System – (адаптивная нейро-нечеткая 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. Загрузка ANFIS-редактора осуществляется по команде anfisedit. С одной стороны, гибридная сеть ANFIS представляет собой нейронную сеть с единственным выходом и несколькими входами, которые представляют собой нечеткие лингвистические переменные. При этом термы входных лингвистических переменных описываются стандартными для системы MATLAB функциями принадлежности, а термы выходной переменной представляются линейной или постоянной функцией принадлежности.

С другой стороны, гибридная сеть ANFIS представляет собой систему нечеткого вывода FIS типа Сугено нулевого или первого порядка, в которой каждое из правил нечетких продукций имеет постоянный вес, равный 1.

3.4. Моделирование и реализация нейро-нечеткой сети в среде MATLAB

В пакете Fuzzy Logic Toolbox системы MATLAB гибридные сети реализованы в форме адаптивных систем нейро-нечеткого вывода ANFIS [3]. При этом разработка и исследование гибридных сетей оказывается возможной:

– в интерактивном режиме с помощью специального графического редактора адаптивных сетей, получившего название редактора ANFIS;

– в режиме командной строки с помощью ввода имен соответствующих функций с необходимыми аргументами непосредственно в окно команд системы MATLAB.

Редактор ANFIS позволяет создавать или загружать конкретную модель адаптивной системы нейро-нечеткого вывода, выполнять ее обучение, визуализировать ее структуру, изменять и настраивать ее параметры, а также использовать настроенную сеть для получения результатов нечеткого вывода.

В результате выполнения этой команды появится графическое окно (рис. 3.4.). На этом же рисунке указаны функциональные области ANFIS-редактора, описание которых приведено ниже.

ANFIS-редактор содержит 3 верхних меню - File, Edit и View, область визуализации, область свойств ANFIS, область загрузки данных, область генерирования исходной системы нечеткого логического вывода, область обучения, область тестирования, область вывода текущей информации, а также кнопки Help и Close, которые позволяют вызвать окно справки и закрыть ANFIS-редактор, соответственно.

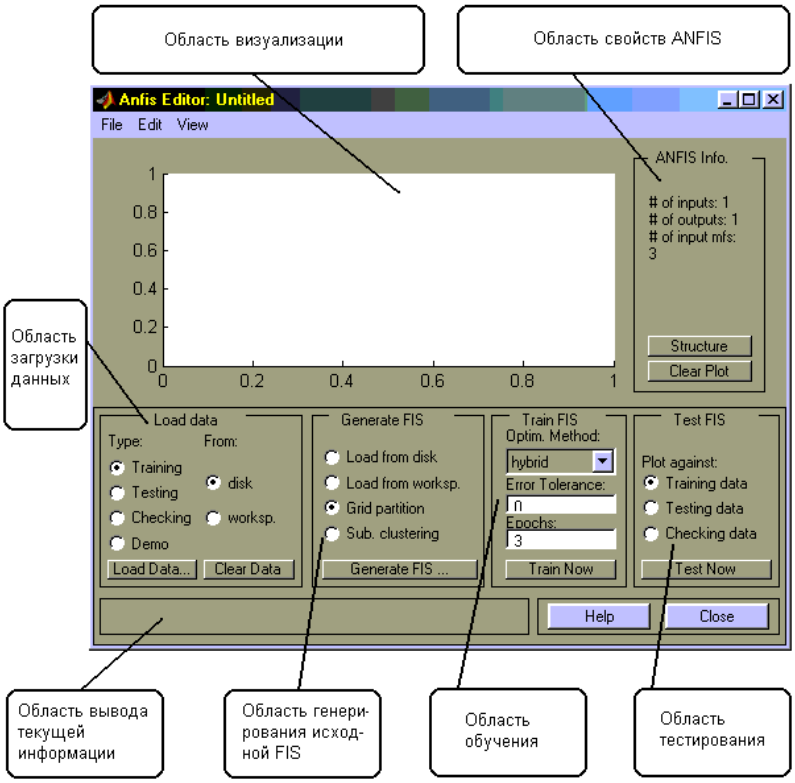


Рис.3.4. Основное окно ANFIS-редактора

Меню File и View одинаковые для всех GUI-модулей, используемых с системами нечеткого логического вывода. Меню Edit. Общий вид меню приведен на рис.3.5.

Рис.3.5. Меню Edit

Команда Undo отменяет ранее совершенное действие. Выполняется также по нажатию Ctrl+Z.

Команда FIS Properties… открывает FIS-редактор. Эта команда может быть также выполнена нажатием Ctrl+1.

Команда Membership Functions… открывает редактор функций принадлежностей. Эта команда может быть также выполнена нажатием Ctrl+2.

Команда Rules… открывает редактор базы знаний. Эта команда может быть также выполнена нажатием Ctrl+3.

Команда Anfis… открывает ANFIS-редактор. Эта команда может быть также выполнена нажатием Ctrl+3. Заметим, что данная команда, запущенная из ANFIS-редактора не приводит к выполнению каких-либо действий, так этот редактор уже открыт. Однако, в меню Edit других GUI-модулей, используемых с системами нечеткого логического вывода, добавляется команда Anfis…, позволяющая открыть ANFIS-редактор из этих модулей.

Область визуализации предназначена для вывода двух типов информации:

- при обучении системы – кривой обучения в виде графика зависимости ошибки обучения от порядкового номера итерации;
- при загрузке данных и тестировании системы – экспериментальных данных и результатов моделирования.

Экспериментальные данные и результаты моделирования выводятся в виде множества точек в двумерном пространстве. При этом по оси абсцисс откладывается порядковый номер строчки данных в выборке (обучающей, тестирующей или контрольной), а по оси ординат - значение выходной переменной для данной строчки выборки. Используются следующие маркеры:

- голубая точка (.) – тестирующая выборка;
- голубая окружность (o) – обучающая выборка;

– голубой плюс (+) – контрольная выборка;

– красная звездочка (*) – результаты моделирования.

Область свойств ANFIS. В области свойств ANFIS (ANFIS info) выводится информация о количестве входных и выходных переменных, о количестве функций принадлежностей для каждой входной переменной, а также о количестве строчек в выборках. В этой области расположены две кнопки: Structure и Clear Plot.

Нажатие кнопки Structure открывает новое графическое окно, в котором система нечеткого логического вывода представляется в виде нейро-нечеткой сети.

В качестве иллюстрации приведена нейро-нечеткая сеть, содержащая четыре входных переменных и одну выходную (рис.3.6). В этой системе по три лингвистических терма используется для оценки каждой из входных переменных и четыре терма для выходной.

Нажатие кнопки Clear Plot позволяет очистить область визуализации.

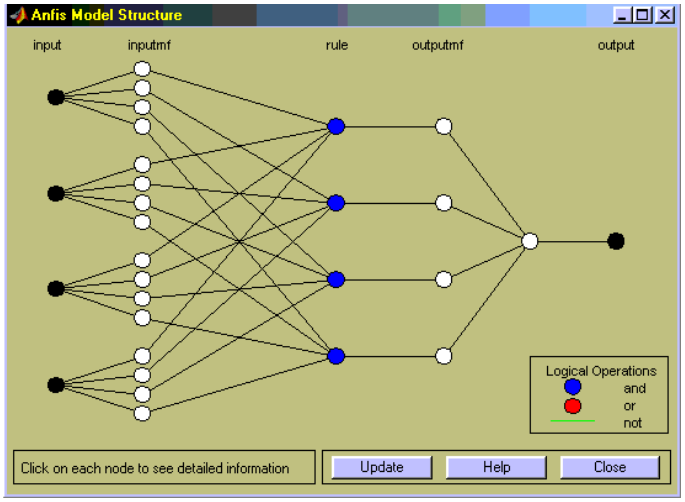


Рис. 3.6. Пример структуры нейро-нечеткой сети

Область загрузки данных. В области загрузки данных (Load data) расположены:

меню выбора типа данных (Type), содержащее альтернативы:

- Traning - обучающая выборка;
- Testing - тестирующая выборка;
- Checking - контрольная выборка;
- Demo - демонстрационный пример;

меню выбора источника данных (From), содержащее альтернативы:

- disk – диск;
- worksp. - рабочая область MatLab;

кнопка загрузки данных Load Data…, по нажатию которой появляется диалоговое окно выбора файла, если загрузка данных происходит с диска, или окно ввода идентификатора выборки, если загрузка данных происходит из рабочей области;

кнопка очистки данных Clear Data.

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

Область генерирования исходной системы нечеткого логического вывода.

В области генерирования (Generate FIS) расположено меню выбора способа создания исходной системы нечеткого логического вывода. Меню содержит следующие альтернативы:

- Load from disk – загрузка системы с диска;

– Load from worksp. – загрузка системы из рабочей области MatLab;

– Grid partition - генерирование системы по методу решетки (без кластеризации);

– Sub. clustering – генерирование системы по методу субкластеризации.

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

При выборе Load from disk появляется стандартное диалоговое окно открытия файла.

При выборе Load from worksp. появляется стандартное диалоговое окно ввода идентификатора системы нечеткого логического вывода.

При выборе Grid partition появляется окно ввода параметров метода решетки (рис. 3.7), в котором нужно указать количество термов для каждой входной переменной и тип функций принадлежности для входных и выходной переменных.

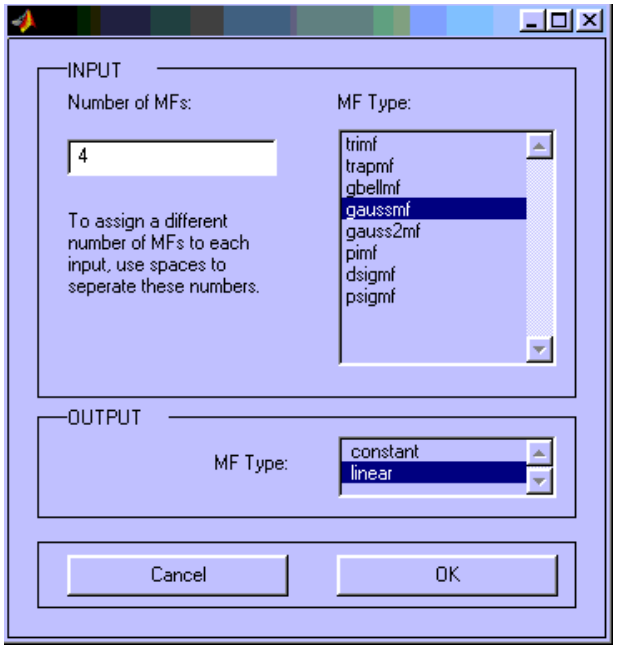


Рис. 3.7. Окно ввода параметров для метода решетки

При выборе Sub. clustering появляется окно ввода следующих параметров метода субкластеризации (рис. 3.8):

- Range of influence – уровни влияния входных переменных;
- Squash factor – коэффициент подавления;
- Accept ratio – коэффициент, устанавливающий во сколько раз потенциал данной точки должен быть выше потенциала центра первого кластера для того, чтобы центром одного из кластеров была назначена рассматриваемая точка;
- Reject ratio –коэффициент, устанавливающий во сколько раз потенциал данной точки должен быть ниже потенциала центра первого кластера, чтобы рассматриваемая точка была исключена из возможных центров кластеров.

Область обучения. В области обучения (Train FIS) расположены меню выбора метода оптимизации (Optim. method), поле задания требуемой точности обучения (Error tolerance), поле задания количества итераций обучения (Epochs) и кнопка Train Now, нажатие которой запускает режим обучение. Промежуточные результаты обучения выводятся в область визуализации и в рабочую область MatLab.

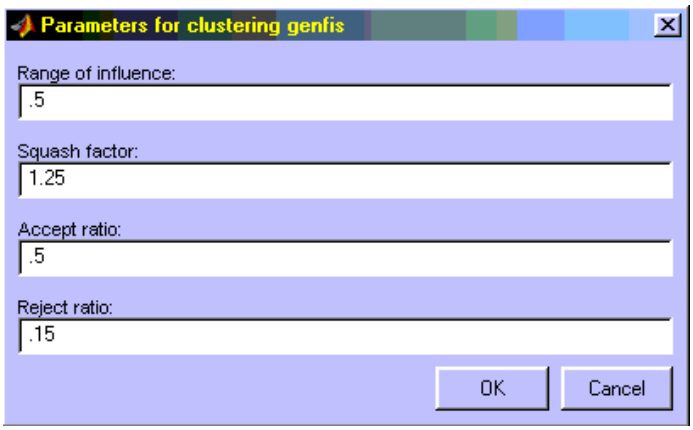


Рис.3.8. Окно ввода параметров для метода субкластеризации

В ANFIS-редакторе реализованы два метода обучения:

– backpropa – метод обратного распространения ошибки, основанный на идеях метода наискорейшего спуска;

– hybrid – гибридный метод, объединяющий метод обратного распространения ошибки с методом наименьших квадратов.

Область тестирования. В области тестирования (Test FIS) расположены меню выбора выборки и кнопка Test Now, по нажатию по которой происходит тестирование нечеткой системы с выводом результатов в область визуализации.

Область вывода текущей информации. В этой области выводится наиболее существенная текущая информация, например, сообщения об окончании выполнении операций, значение ошибки обучения или тестирования и т.п.

3.5. Пример нейро-нечеткой модели

Пример. Имеются исходные данные индекса РТС за период с 01.03.2004 по 30.04.2004. Требуется построить нейро-нечеткую сеть и спрогнозировать значение индекса на 1.05.2004.

Общая последовательность процесса разработки модели гибридной сети может быть представлена в следующем виде.

1.*Подготовка файла с обучающими данными*. Целесообразно воспользоваться редактором электронных таблиц MS Excel. Обучающую выборку необходимо сохранить во внешнем файле с расширением *.dat.

Рекомендуется в MS Excel создать файл в формате *.txt, а затем переименовать файл с расширением *.dat. При этом в файле с обучающими данными необходимо запятые заменить на точки.

Пример «Фрагмент обучающей выборки для анализа и прогнозирования индекса РТС». Алгоритм прогнозирования подразумевает то, что каждое последующее значение рассчитывается на основе нескольких предыдущих (табл. 3.1).

2. *Открыть редактор ANFIS. Загрузить файл с обучающими данными*.

Кнопка загрузки данных Load Data, по нажатию которой появляется диалоговое окно выбора файла, если загрузка данных происходит с диска, или окно ввода идентификатора выборки, если загрузка данных происходит из рабочей области.

Внешний вид редактора ANFIS с загруженными обучающими данными изображен на рис. 3.9.

3. *После подготовки и загрузки обучающих данных можно сгенерировать структуру системы нечеткого вывода* FIS типа Сугено, которая является моделью гибридной сети в системе Matlab. Для этой цели следует воспользоваться кнопкой Generate FIS в нижней части рабочего окна редактора. При этом две первые опции относятся к предварительно созданной структуре гибридной сети, а две последних – к форме разбиения входных переменных модели.

Перед генерацией структуры системы нечеткого вывода типа Сугено после вызова диалогового окна свойств зададим для каждой из входных переменных по три лингвистических терма, а в качестве типа их функций принадлежности выберем треугольные функции.

Таблица 3.1

Фрагмент данных для обучения нейро-нечеткой сети

Первая входная переменная	Вторая входная переменная	Третья входная переменная	Выходная переменная
688.72	686.21	667.27	669.26
686.21	667.27	669.26	673.25
667.27	669.26	673.25	688.68
669.26	673.25	688.68	680.86
673.25	688.68	680.86	671.33
688.68	680.86	671.33	669.55
680.86	671.33	669.55	676.20
671.33	669.55	676.20	680.57
669.55	676.20	680.57	698.70
676.20	680.57	698.70	708.59
680.57	698.70	708.59	721.81
698.70	708.59	721.81	712.88
708.59	721.81	712.88	713.14
721.81	712.88	713.14	705.16
712.88	713.14	705.16	706.71
713.14	705.16	706.71	716.55

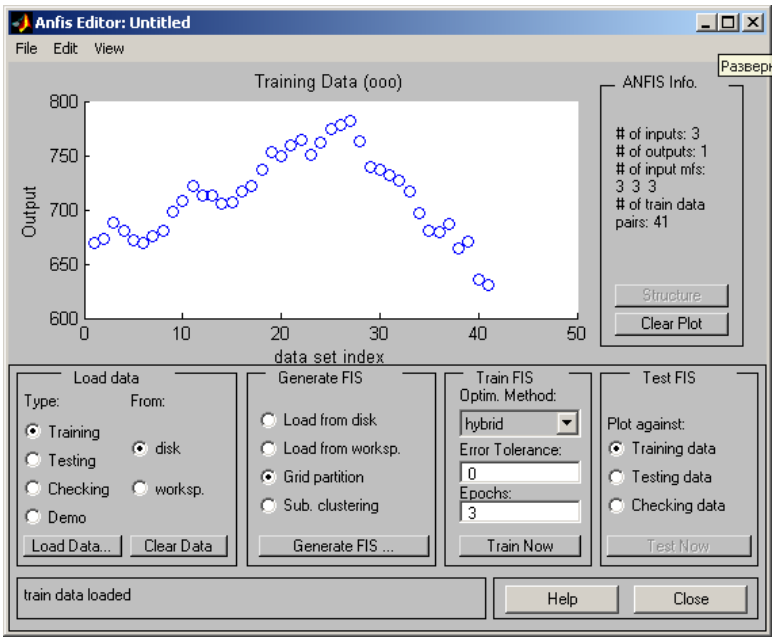


Рис. 3.9. Графический интерфейс редактора ANFIS после загрузки обучающих данных

После нажатия кнопки Generate FIS вызывается диалоговое окно с указанием числа и типа функций принадлежности для отдельных термов входных переменных и выходной переменной (рис. 3.10).

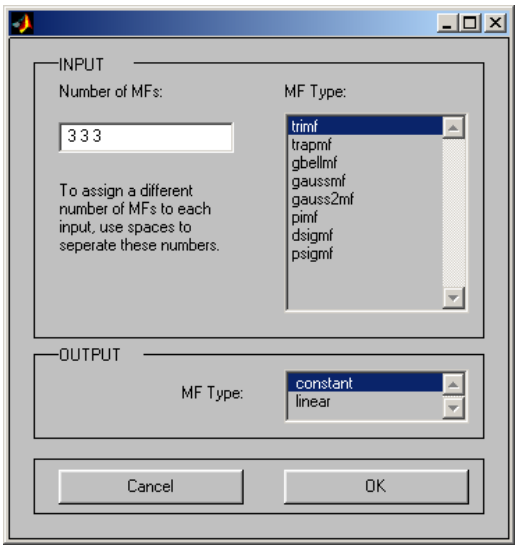


Рис.3.10. Диалоговое окно для задания количества и типа функций принадлежности

4. *После генерации структуры гибридной сети можно визуализировать ее структуру*, для чего следует нажать кнопку Structure в правой части графического окна (рис. 3.11).

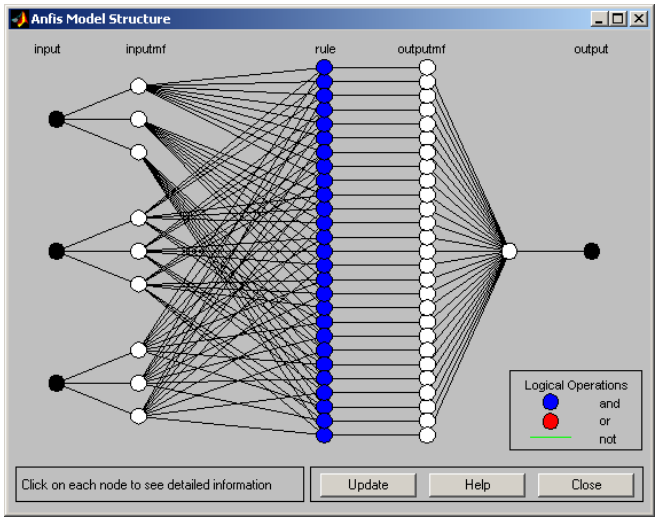


Рис. 3.11. Структура сгенерированной системы нечеткого вывода

Для рассматриваемого примера система нечеткого вывода содержит три входных переменных с тремя термами каждая, 27 правил нечетких продукций, одну выходную переменную с 27 термами.

5. Перед обучением гибридной сети необходимо задать параметры обучения, для чего следует воспользоваться следующей группой опций в правой нижней части рабочего окна:

– выбрать метод обучения гибридной сети – обратного распространения (backpropo) или гибридный (hybrid), представляющий собой комбинацию метода наименьших квадратов и метода убывания обратного градиента.

– установить уровень ошибки обучения (Error Tolerance) – по умолчанию значение 0. (изменять не рекомендуется)

– задать количество циклов обучения (Epochs) – по умолчанию значение 3 (рекомендуется увеличить для рассматриваемого примера задать его значение равным 40).

Для обучения сети следует нажать кнопку Train now. При этом ход процесса обучения иллюстрируется в окне визуализации в форме графика зависимости ошибки от количества циклов обучения (рис. 3.12).

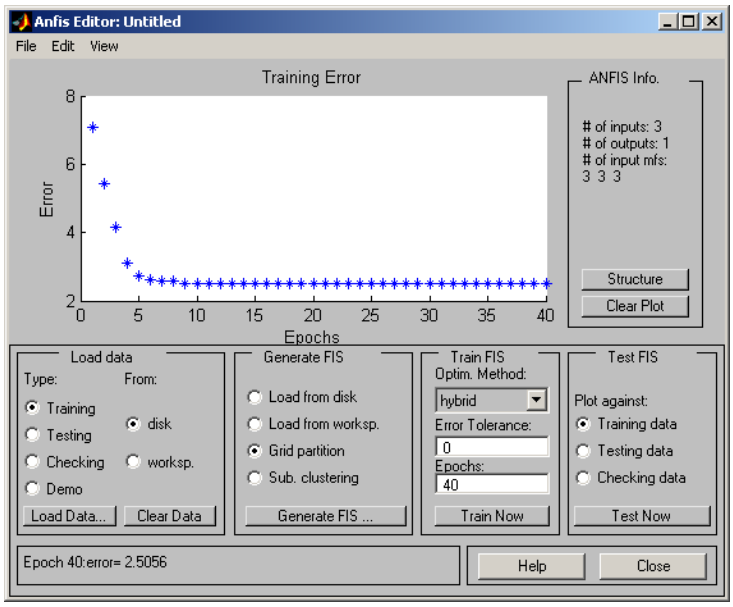


Рис.3.12. График зависимости ошибок обучения от количества циклов обучения

Дальнейшая настройка параметров построенной и обученной гибридной сети может быть выполнена с помощью стандартных графических средств пакета Fuzzy Logic Toolbox. Для этого рекомендуется сохранить созданную систему нечеткого вывода во внешнем файле с расширением *.fis, после чего следует загрузить этот файл в редактор систем нечеткого вывода FIS.

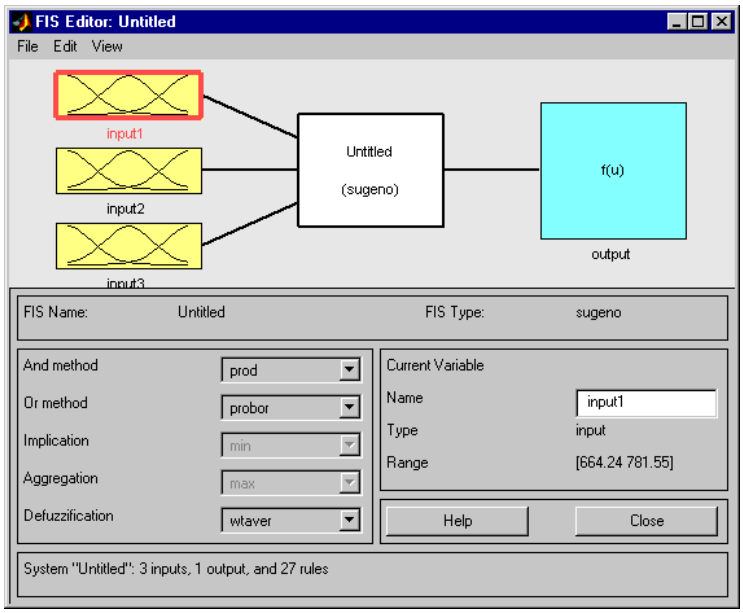


Рис.3.13. Графический интерфейс редактора FIS для сгенерированной системы нечеткого вывода

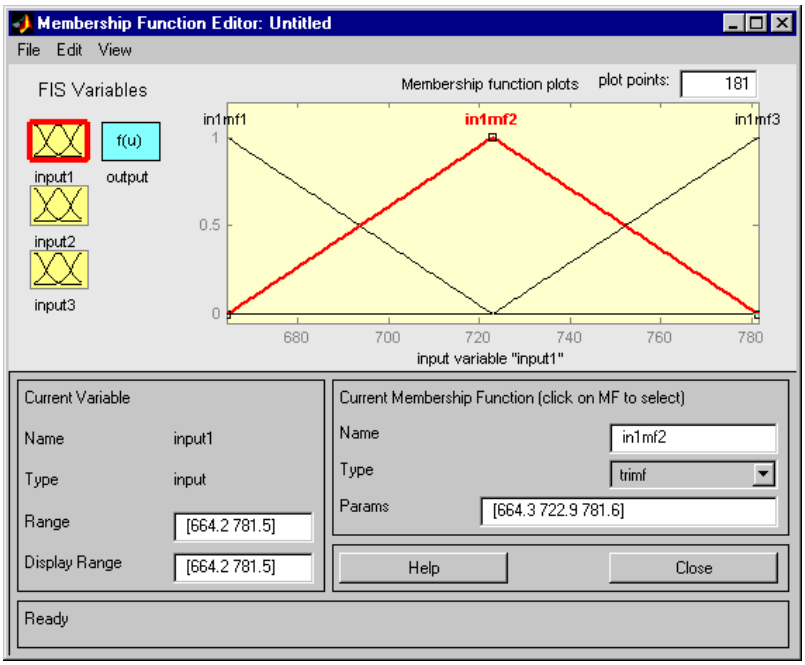


Рис.3.14. Графический интерфейс редактора функций принадлежности построенной системы нечеткого вывода

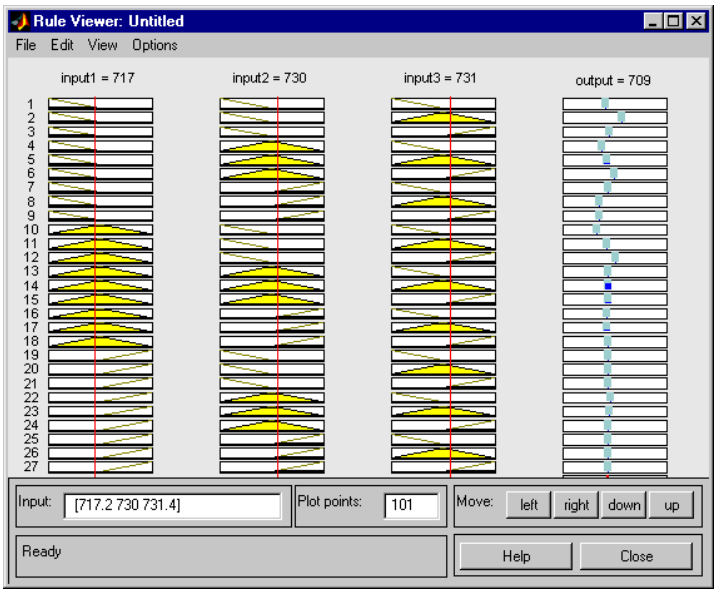


Рис.3.15. Графический интерфейс просмотра правил сгенерированной системы нечеткого вывода

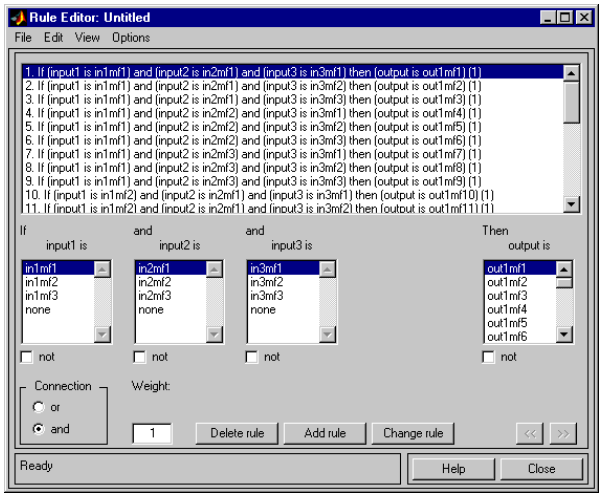


Рис. 3.16. Фрагмент базы нечетких правил

Для проверки адекватности построенной нечеткой модели гибридной сети можно спрогнозировать курс доллара на определенный день, воспользовавшись функцией evalfis.

Проверка на корректность с использованием функции evalfis приведена ниже.

```
>> fis = readfis('C:\Users\student\Documents\MATLAB\final-final.fis');
```

```
>> tip = evalfis([-3 -4.4], fis)
tip =

   41,32

>> fis = readfis('C:\Users\student\Documents\MATLAB\final-final.fis');
>> tip = evalfis([3 1.7], fis)
tip =

   2.6548

>> fis = readfis('C:\Users\student\Documents\MATLAB\final-final.fis');
>> tip = evalfis([0 -3], fis)

tip =

  0.278
```

3.6. Изменение параметров гибридной сети

Для построения системы, максимально приближенной к реальной, требуется провести экспериментальное моделирование, поочередно изменив функцию принадлежности переменных (как входных, так и выходной), количество термов входных переменных, а также используемые методы И, ИЛИ и дефаззификации. При варьировании перечисленных параметров оценивается ошибка аппроксимации (E) и выбирается набор оптимальных параметров нейро-нечеткой сети.

Для анализа ошибки аппроксимации необходимо рассмотреть восемь функций принадлежности входных переменных:

trimf (задает функцию принадлежности в форме треугольника);

trapmf (применяется для задания ассиметричных функций принадлежности переменных, наиболее возможные значения которых принимаются на некотором интервале);

gbellmf и gaussmf (применяются для задания гладких симметричных функций принадлежности);

gauss2mf, dsigmf и psigmf (применяются для задания гладких ассиметричных функций принадлежности);

pimf (применяется для задания ассиметричных функций принадлежности с плавным переходом от пессимистической к оптимистической оценки нечеткого числа).

Ниже приведены результаты экспериментов. Низкую точность показало применение функций pimf и trapmf (ошибка аппроксимации E> 20). Немногим лучшие, но также низкие результаты достигаются при использовании функций dsigmf и psigmf (E ≈ 18). Средние результаты показывают функции trimf, gaussmf и gauss2mf (4<E<8). Наилучшие результаты получены при использовании функции gbellmf (E = 1,382).

Рассмотрено две функции для выходной переменной: const – «постоянная» – (E = 1,382) и linear – «линейная» - (E = 0,03), вторая из которых оказалась существенно точнее.

Ошибка E = 0,03 получена при использовании трех термов для каждой из входных переменных. Использование четырех термов дало ничтожное улучшение (E = 0,008). Затраты на реальную реализацию такой системы нецелесообразны, так как они дают сравнительно малое улучшение при резком усложнении структуры. Поэтому было принято решение не принимать структуру с четырьмя термами как наилучшую.

Метод AND целесообразен, если среди разработанных нечетких правил есть такие, где связкой между переменными служит операция И. В данном случае рассмотрены методы AND (prod) – «умножение» – (E = 0,03) и AND (min) – «минимум» – (E = 261,264). Очевидно, что метод AND (prod) точнее.

Метод OR целесообразен, если среди разработанных нечетких правил есть такие, где связкой между переменными служит операция ИЛИ. В данном случае таких правил нет, поэтому два рассмотренных метода OR (max) – «умножение» – и OR (probor) – «вероятностное ИЛИ» – дали одинаковые погрешности (E = 0,03), причем они равны предыдущему полученному минимальному значению, что доказывает отсутствие влияния данного метода на построение конечной системы.

Рассмотрено два метода дефаззификации: Deffuzzification (wtsum) – «взвешенная сумма» и Deffuzzification (wtaver) – «взвешенное среднее» – (E = 0,03). Наименьшая погрешность достигается при использовании второго из указанных методов.

При изменении функции выходной переменной были получены следующие ошибки аппроксимации:

- для функции const E = 1,382;
- для функции linear E = 0,03.

Для тестируемой задачи наибольшей точностью обладает функция linear.

Путем изменения количества термов была построена система, дающая минимальную ошибку аппроксимации. При увеличении количества термов для каждого входного порта до 5 термов, а выходного до 9 термов среднеквадратичная ошибка уменьшается, но возрастают затраты на реализацию нейро-сетевой системы.

Таким образом, построенная нейро-нечеткая система с наименьшей ошибкой аппроксимации E = 0,03. Этот результат намного лучше, чем в системе типа Мамдани и типа Сугено. Исходя из этого, можно сделать вывод, что использование для заданной задачи аппроксимации нейро-нечеткой системы предпочтительнее.

Контрольные вопросы

1. Дайте определение нейро-нечеткой сети.
2. Каково предназначение сетей нейро-нечеткого вывода?
3. В чем преимущества использования нейро-нечетких сетей?
4. Охарактеризуйте структуру нейро-нечеткой сети.
5. Опишите процесс разработки нейро-нечеткой сети в среде MATLAB.
6. Как проверить адекватность построенной нейро-нечеткой сети?
7. Какие возможности по визуализации результатов моделирования предоставляет система MATLAB?
8. Какие варьируемые параметры нейро-нечеткой системы влияют на ошибку аппроксимации?
9. Какой метод дефаззификации лучше и почему?
10. Какой метод AND, OR необходимо применять для решения задачи аппроксимации?

4. ЛАБОРАТОРНЫЕ РАБОТЫ

4.1. Лабораторная работа №1. Основные приемы работы в MatLab

Цель лабораторной работы: приобретение навыков основы работы с программой MATLAB и основными командами задания векторов, матриц, вычисления функций, построения графиков, решения задач аппроксимации.

4.1.1. Построение графиков функции одной переменной

Рассмотрим пример построения графика одной переменной $y(x)=e^{-x} * \sin 10x$, определённой на отрезке [0,1].

Вывод отображения функции в виде графика состоит из следующих этапов:

1. Задание вектора значений аргумента x.
2. Вычисление вектора значений y(x).
3. Вызов команды *plot* для построения графика.

Команды для задания вектора x и вычисления функции лучше завершить точкой с запятой для подавления вывода в командное окно их значений.

После команды *plot* точку с запятой ставить не требуется, так как она ничего не выводит в командное окно. Для поэлементного умножения необходимо использовать точку со звёздочкой (.*).

```
>>x=0:0.05:1;

>>y=exp(-x).*sin(10*x);
```

>>plot(x,y)

После выполнения команд на экране появится окно Figure1 с графиком функции.

Сравнение нескольких функций удобно производить, отобразив их графики на одних осях. Например, на отрезке [-1, -0,3] построить графики функций

$f(x)=\sin(1/x^2)$, $g(x)=\sin(1.2/x^2)$.

>> x=-1: 0.005: -0.3;

>>f=sin(x.^-2);

>> g=sin(1.2*x.^-2);

>>plot(x,f,x,g)

Графики можно вывести разным цветом. Команда *plot* позволяет задать стиль и цвет линий, например,

>>plot(x,f,'k-'x,g,'k:')

Первый график выводится сплошной чёрной линией, а второй - чёрной пунктирной. Параметр k обозначает цвет линии чёрный, дефис-сплошную линию, а двоеточие – пунктирную линию (табл.4.1).

Например, для построения первого графика красными точечными маркерами без линии, а второго графика пунктирной черной линией следует использовать команду

>>plot (x, f, 'r.', x, g, 'k:').

Таблица 4.1

Параметры формирования графика

Параметр цвета	Цвет линий графика	Тип маркера	Назначение маркера	Параметр линии	Тип линии
y	жёлтый	.	точка	-	сплошная
m	розовый	◦	кружок	:	пунктирная
c	голубой	×	крестик	-.	штрихпунктирная
r	красный	+	знак «плюс»	--	штриховая
g	зелёный	*	звёздочка		
b	синий	s	квадрат		
w	белый	d	ромб		
k	чёрный	v	треугольник		

Пример трёхмерного изображения функции y=x1^3*sin(x2-4) представлен на рис.4.1.

```
%Построение графика функции y=x1^3*sin(x2-4)
%в области x1[-7 3], x2[-3.0 1.7]
 n=15%Количество точек дискретизации
x1=linspace(-7,3,n)
```

```
x2=linspace(-3.0,1.7,n)
y=zeros(n,n)
for j=1:n
   y(j,:)=x1.^3*sin(x2(j)-4)
end
surf(x1,x2,y)
xlabel('x_1');
ylabel('x_2');
zlabel('y');
title('Искомая зависимость')
```

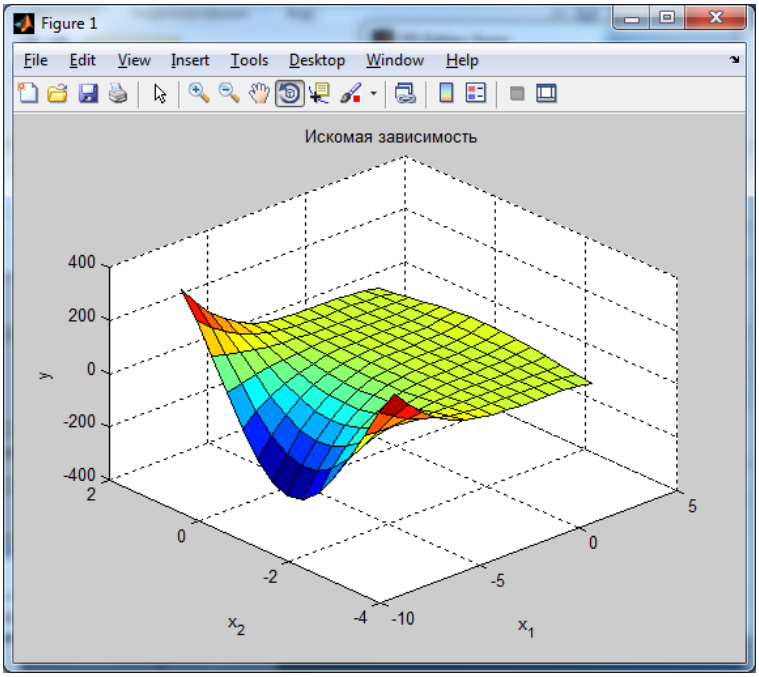


Рис.4.1. Пример трёхмерного изображения

4.1.2. Двумерные массивы, матрицы

Ввод матрицы размером два на три. Матрицу можно рассматривать как вектор-столбец из двух элементов, каждый из которых является вектор-

строкой длиной три. Поэтому строки при наборе разделяются точкой с запятой.

A=[3 1 -1; 2 4 3];

A=

3 1 -1

2 4 3

Другие способы ввода квадратной матрицы.

>>B=[4 3 -1

2 7 0

-5 1 2]

По окончанию набора нажмите Enter.

Ещё один способ ввода матриц состоит в том, что матрицу можно трактовать как вектор-строку, каждый элемент которой является вектором-столбцом.

Рассмотрим матрицу C размером два на три

$$\begin{matrix} 3 & -1 & 7 \\ 4 & 2 & 0 \end{matrix}$$

>>C=[[3;4] [-1;2] [7;0]]

C=

$$\begin{matrix} 3 & -1 & 7 \\ 4 & 2 & 0 \end{matrix}$$

Для проверки откройте окно Workspace или наберите в командной строке whos.

4.1.3. Обращение к элементам матриц

Доступ к элементам матриц производится с помощью двух индексов-номера строки и номера столбца, заключённых в круглые скобки.

```
>>C(2,3)
ans=
0
```

Элементы матриц могут входить в состав выражений:

```
>>C(1,1)+C(2,2)+C(2,3)
ans=
5
```

4.1.4. Сложение, вычитание, умножение, транспонирование и возведение в степень

При использовании матричных операций следует помнить, что для сложения или вычитания матрицы должны быть одного размера, а при перемножении число столбцов первой матрицы обязательно равняется числу строк второй матрицы. Сложение и вычитание осуществляется с помощью знаков плюс и минус.

```
>>S=A– B
S=
0 -2 8
2 -2 -3
```

>>S=A+B

S=

6 0 6

6 6 3

Для умножения матриц предназначена «звёздочка».

>>P=A*B

P=

-25 9 11

20 26 -4

Умножение матрицы на число.

>>P=A*3

P=

9 3 -3

6 12 9

Транспонирование матрицы, как и вектора, производится с помощью апострофа (‘). Для вещественных чисел можно использовать точку с апострофом (.‘).

$$B = \begin{matrix} 4 & 3 & -1 \\ 2 & 7 & 0 \\ -5 & 1 & 2 \end{matrix}$$

>>B’

$$\begin{matrix} 4 & 2 & -5 \\ 3 & 7 & 1 \\ -1 & 0 & 2 \end{matrix}$$

Возведение квадратной матрицы в целую степень производится с помощью команды ^.

>>B2=B1^2

B2=

$$\begin{matrix} 27 & 32 & -6 \\ 22 & 55 & -2 \\ -28 & -6 & 9 \end{matrix}$$

Проверьте полученный результат, умножив матрицу саму на себя.

Задание. Вычислите выражение $(A+C)B^3(A-C)^T$

Учтите приоритет операций, сначала выполняется транспонирование, затем возведение в степень, потом умножение, а сложение и вычитание производится в последнюю очередь.

>>(A+C)*B^3*(A-C)’

4.1.5. Перемножение матрицы и вектора

Поскольку вектор-столбец или вектор-строка в MatLab являются матрицами, у которых один из размеров равен единице, то все вышеописанные операции применимы и для умножения матрицы на вектор, или вектор-строки на матрицу.

Например, вычисление выражения

$$[1\ 3\ -2]\begin{vmatrix} 2 & 0 & 1 \\ -4 & 8 & -1 \\ 0 & 9 & 2 \end{vmatrix}\begin{bmatrix} -8 \\ 3 \\ 4 \end{bmatrix}$$

Можно осуществить следующим образом:

```
>>a=[1  3  -2];

>>B=[2  0  1; -4  8  -1; 0  9  2];

>>c=[-8; 3; 4];

>>a*B*c

ans=

74
```

4.1.6. Вычисление функций

Пример вычисления функции, параметры которой приведены в табл.4.1. Функция y=f(x) задана выражением

$$y=\frac{1+sin^2(b^3+x^3)}{\sqrt[3]{b^3}+x^3}$$

Таблица 4.1.

Параметры задачи

Функция	b	x_n	x_k	Δx
y=f(x)	2.5	1.28	3.28	0.4

```
b = 2.5

xn = 1.28

xk = 3.28

xd = 0.4

x1 = 1.1

x2 = 2.4
```

```
x3 = 3.6

x4 = 1.7

x5 = 3.9

%Решение задачи (рис.4.2)

X = xn:xd:xk

Y = (1 + sin(b.^3 + X.^3).^2)./(b^3+X.^3).^(1/3)

b =

    2.5000

xn =

    1.2800

xk =

    3.2800

xd =

    0.4000

x1 =

    1.1000

x2 =

2.4000

x3 =

    3.6000

x4 =
```

```
    1.7000

x5 =

    3.9000

X =

    1.2800    1.6800    2.0800    2.4800    2.8800    3.2800

Y =

0.6965    0.7313    0.4253    0.4024    0.5702    0.3679
```

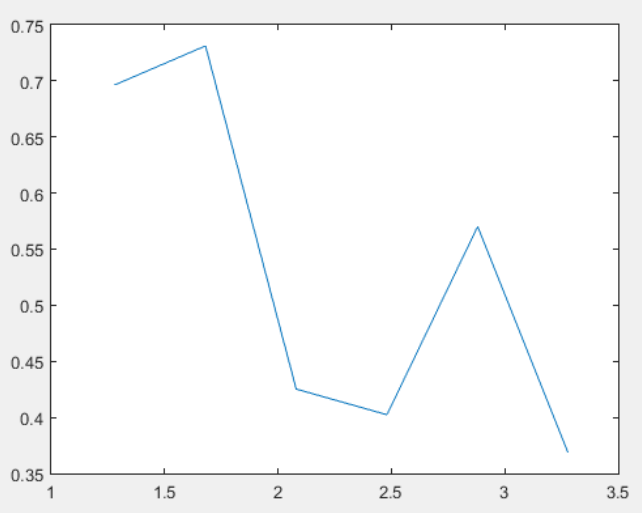


Рис. 4.2. График вычисления y(x)

4.1.7. Аппроксимация функций

Пример аппроксимации функции. Требуется получить эмпирические формулы и оценить их погрешность для функции y=f(x), заданной в таблице 4.2.

Таблица 4.2.

Данные для аппроксимации функции

xi	-3	-2	-1	0	1	2	3
yi	-0.71	-0.01	0.51	0.82	0.88	0.51	0.49

X = [-3, -2, -1, 0, 1, 2, 3]

Y = [-0.71, -0.01, 0.51, 0.82, 0.88, 0.51, 0.49]

plot(X,Y,'o')

X =

-3 -2 -1 0 1 2 3

Y =

-0.7100 -0.0100 0.5100 0.8200 0.8800 0.5100 0.4900

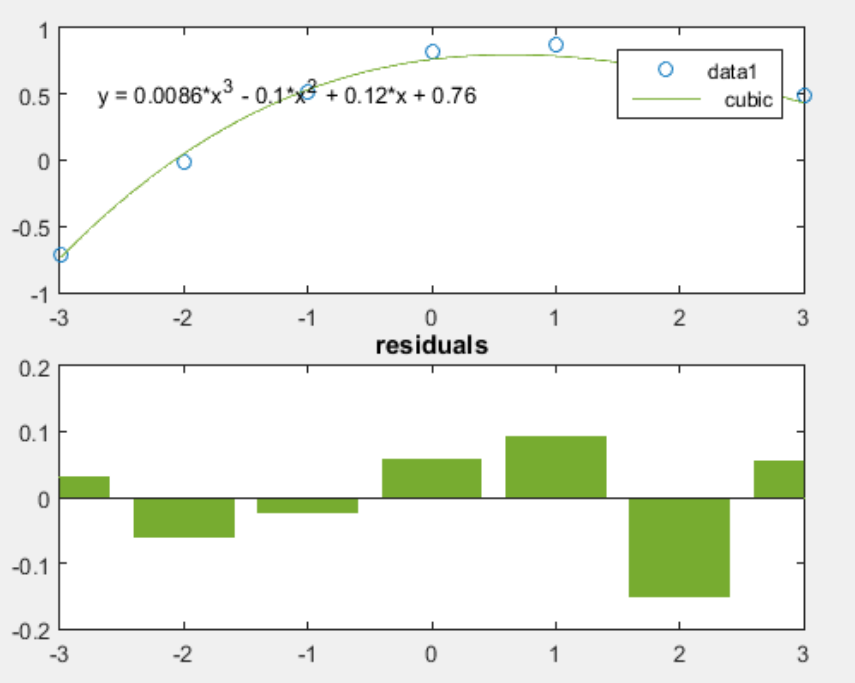


Рис.4.3.Решение задания аппроксимации функции

В результате аппроксимации функции получена зависимость y=0.0085*x^3 -0.1*x^2+0.12*x+0.76 (рис.4.3).

4.1.8. Вычислить среднеквадратичную ошибку аппроксимации

Оценить ошибку аппроксимации в MatLab можно с помощью специальной среднеквадратичной ошибки по формуле:

$$E= \frac{1}{n}\sum_{i=1}^{n}(Y1i - Y2i)^2,$$

где i - номер наблюдения, $Y1_i$ - текущее наблюдение реальной функции, $Y2_i$ - текущее наблюдение аппроксимированной функции, полученные при одном и том же входном векторе
n - общее количество наблюдений.

4.1.9. Задание по лабораторной работе

Изучите материалы по основам работы в MATLAB, приведённые в настоящем учебном пособии и учебной литературе [3]. Сохраните результаты своей работы в *Workspace, для чего* выберите в меню *File* пункт *SaveWorkspace As*. Данный файл включите в ваш отчёт по результатам выполнения лабораторной работы, который в электронном виде представьте преподавателю. Конкретные параметры и функции для выполнения лабораторной работы выдаются преподавателем.

Последовательность шагов выполнения лабораторной работы.

1. Вычисление значений арифметических выражений.
2. Вычисление суммы векторов. Векторы, *a* и *b* хранятся в двумерных массивах.
3. Вычисление минимума, максимума и среднего арифметического векторов.

4. Умножение векторов одинаковой длины.

5. Сортировка элементов вектора по возрастанию и убыванию.

6. Поэлементное возведение в степень вектора, умножение его на число и деление вектора на число.

7. Вычисление скалярного произведения заданных векторов. Вычисление выражения от произведения вектора-строки на матрицу и на вектор-столбец.

8. Ввести матрицу и вывести заданный элемент матрицы.

9. Вывод значений функции y(x) в виде таблицы на интервале [a, b] с заданным шагом.

10. Построение графика функции одной переменной, определённой на интервале. График функции y(x) вывести в виде штриховой красным цветом, а f(x) – черным цветом и сплошной линией.

11. Построение графика функции двух переменных, определённых на заданных интервалах.

12. Вычисление значение функции для заданных параметров на интервале.

13. Выполнение аппроксимации функции, заданной таблично.

14. Вычисление среднеквадратичной ошибки аппроксимации.

4.2. Лабораторная работа №2. Разработка нечеткой модели Мамдани для задачи аппроксимации функции

Цель лабораторной работы: изучение и усвоение методов моделирования и принципов функционирования систем нечёткого вывода типа Мамдани, а также приобретение навыков по конструированию нечетких систем в среде MATLAB.

Порядок выполнения лабораторной работы

5. Для создания системы нечёткого вывода типа Мамдани, которая моделирует зависимость, указанную в варианте заданий.
6. Построить график функции, указанной в варианте
7. Выполнить моделирование со всеми типами функций принадлежности и выбрать ее оптимальный тип по критерию минимума среднеквадратичного отклонения.
8. Для алгоритма Мамдани выполнить моделирование со следующими параметрами:

– метод агрегации (максимум, сумма, вероятностное «ИЛИ»);

– метод дефаззификации (центр тяжести, медиана, наибольший из максимумов);

– метод «И» (минимум, умножение (вероятностное «И»));

– метод «ИЛИ» (максимум, вероятностное «ИЛИ»);

– метод импликации (минимум, умножение);

9. Выбрать оптимальные параметры нечеткой системы, перечисленные в п.4, по критерию минимума среднеквадратичного отклонения
10. По результатам проектирования системы нечёткого вывода в Fuzzy Logic Toolbox составить отчет в электронном виде, включив в него результаты промежуточных этапов, а также графики моделируемой и аппроксимированной функции, сформировать выводы.

4.3. Лабораторная работа №3. Разработка нечеткой модели Сугено для задачи аппроксимации функции

Цель лабораторной работы: изучение и усвоение методов моделирования и принципов функционирования систем нечёткого вывода

типа Сугено, а также приобретение навыков по конструированию нечетких систем в среде MATLAB.

Порядок выполнения лабораторной работы

1. Для создания системы нечёткого вывода типа Сугено, которая моделирует зависимость, указанную в варианте заданий.

2. Построить график функции, указанной в варианте

3. Выполнить моделирование со всеми типами функций принадлежности и выбрать ее оптимальный тип по критерию минимума среднеквадратичного отклонения.

4. Для алгоритма Сугено выполнить моделирование со следующими параметрами:

5. метод агрегации (максимум, сумма, вероятностное «ИЛИ»);

6. метод «И» (минимум, умножение (вероятностное «И»));

7. метод «ИЛИ» (максимум, вероятностное «ИЛИ»);

8. метод импликации (минимум, умножение);

9. Выбрать оптимальные параметры нечеткой системы, перечисленные в п.4, по критерию минимума среднеквадратичного отклонения

10. По результатам проектирования системы нечёткого вывода в Fuzzy Logic Toolbox составить отчет в электронном виде, включив в него результаты промежуточных этапов, а также графики моделируемой и аппроксимированной функции, сформировать выводы.

4.4. Лабораторная работа №4. Разработка нейро-нечеткой модели для задачи аппроксимации функции

Цель лабораторной работы: изучение и усвоение методов моделирования и принципов функционирования нейро-нечетких сетей, в

том числе при решении задач экономического прогнозирования, а также приобретение навыков по конструированию нейро-нечетких сетей в среде MATLAB.

Порядок выполнения работы

1. Подготовить файл с обучающими данными с расширением *.dat.
2. Загрузить файл с обучающими данными в редактор ANFIS.
3. Сгенерировать структуру системы нечеткого вывода FIS типа Сугено
4. Произвести обучение нейро-нечеткой сети, предварительно задав параметры обучения
5. Проверить адекватность нить проверку адекватности построенной нечеткой модели гибридной сети.

Литература

1. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы [Текст] / Д. Рутковская, М. Пилиньский, Л. Рутковский; пер. с польского. И. Д. Рудинского. - М.: Горячая линия-Телеком, 2004. – 452 с.: ил.

2. Яхъяева, Г. Э. Нечеткие множества и нейронные сети [Текст] / Г.Э. Яхъяева; Лаб. знаний, Интернет-ун-т информ. технологий - ИНТУИТ.ру. – М.: БИНОМ, 2006.

3. Штовба С.Д., "Проектирование нечетких систем средствами MATLAB. М.: Горячая линия – Телеком, 2007. – 288с." [Электронный ресурс]. http://matlab.exponenta.ru/fuzzylogic/book6/index.php

4. Леоненков, А.В. Нечеткое моделирование в среде MATLAB и fuzzyTECH [Текст]. – СПб.: БХВ-Петербург, 2003. – 736 с.