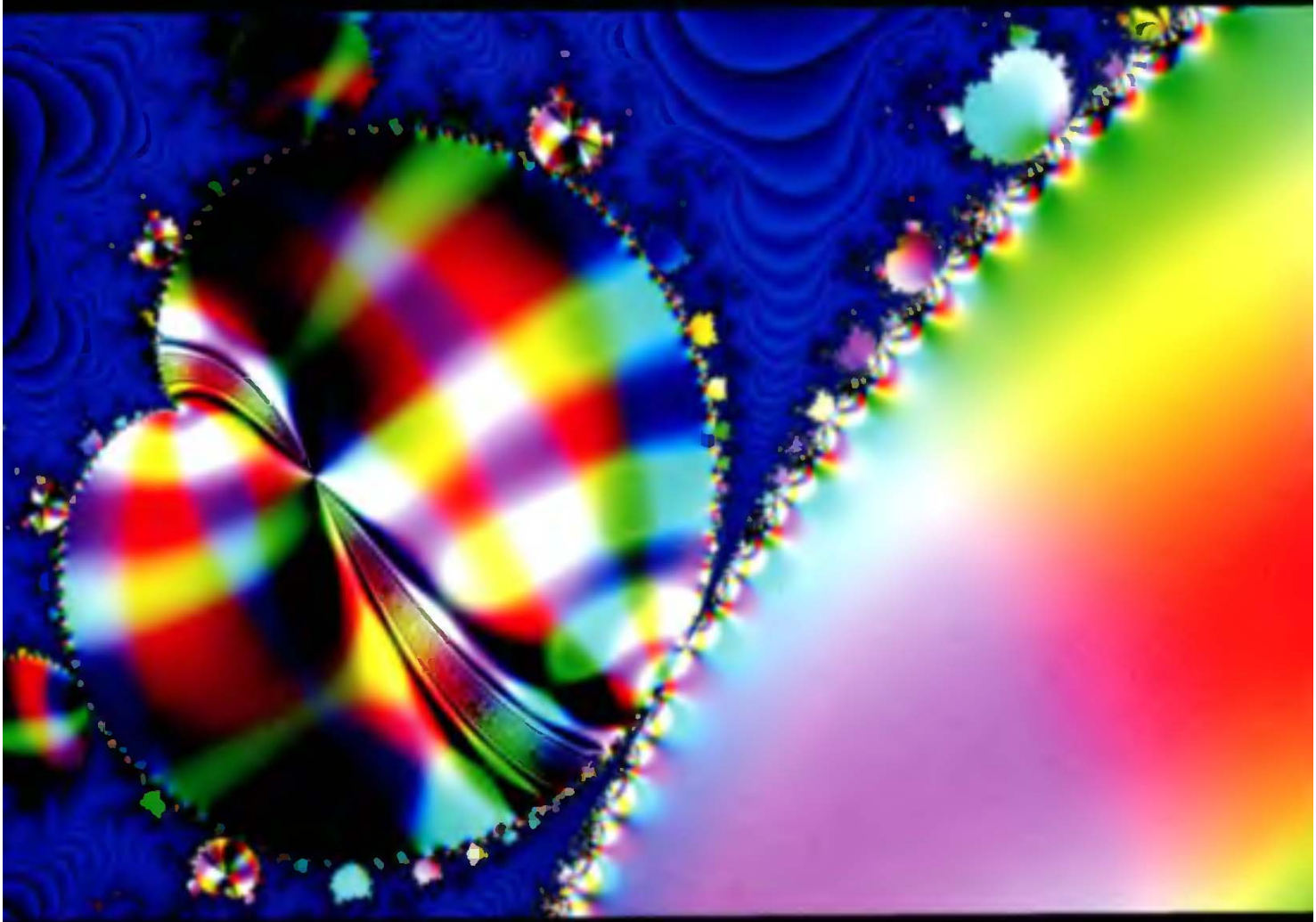
С. Д. Штовба

Проектирование нечетких систем средствами MATLAB



С. Д. Штовба

Проектирование нечетких систем средствами MATLAB

Москва
Горячая линия – Телеком
2007

Штовба С. Д.

Проектирование нечетких систем средствами MATLAB. – М.: Горячая линия – Телеком, 2007. – 288 с., ил.

ISBN 5-93517-359-X.

Рассмотрены вопросы проектирования нечетких систем в пакете Fuzzy Logic Toolbox вычислительной среды MATLAB. Даны необходимые сведения в области теории нечетких множеств и нечеткой логики. Приведен теоретический материал по проектированию нечетких систем. Изложены теория нечеткой идентификации, методы нечеткой кластеризации и их применение для экстракции нечетких правил, а также метод принятия решений в нечетких условиях на основе слияния целей и ограничений. Рассмотрены авторские расширения пакета для проектирования нечетких классификаторов, построения иерархических нечетких систем, обучения нечетких баз знаний типа Мамдани, а также для логического вывода при нечетких исходных данных. Книга может использоваться как учебное пособие к университетским курсам по интеллектуальным системам, искусственному интеллекту, теории принятия решений и методам идентификации.

Для проектировщиков систем, будет полезна научным сотрудникам, аспирантам и студентам старших курсов, интересующимся применением теории нечетких множеств в управлении, идентификации, обработке сигналов, а также разработчикам интеллектуальных систем поддержки принятия решений в медицине, биологии, социологии, экономике, политике, спорте и в других областях.

ББК 22.18

Адрес издательства в Интернет WWW.TECHBOOK.RU
e-mail: radios_hl@mtu-net.ru

Научное издание

Штовба Сергей Дмитриевич

Проектирование нечетких систем средствами MATLAB

Редактор А. С. Попов
Корректор Е. Н. Федоров
Компьютерная верстка Е. В. Кормаковой
Обложка художника В. Г. Ситникова

ЛР № 071825 от 16 марта 1999 г.
Подписано в печать 25.09.06. Формат 70×100/16.
Усл. изд. л. 24. Тираж 2000 экз. Изд. № 6359
Заказ N 5

5-93517-359-X

Отпечатано в типографии ООО "ТИЛЬ-2004", 1007023, Москва,
ул. Электрозаводская д. 21

Памяти моей бабушки
Штовбы Христины Ивановны

Предисловие

О нечетких множествах научный мир узнал 40 лет назад из статьи Л.Заде «Fuzzy Sets» в журнале «Information and Control». Теория нечетких множеств, воспринятая поначалу весьма скептически, сегодня стала модным и эффективным методом моделирования в условиях неопределенности. Нечеткие системы, т.е. системы, использующие нечеткие множества типа «высокая температура», «предсказуемый политик», «ажиотажный спрос», успешно работают в различных областях инженерии, экономики, медицины, биологии, психологии, политики и др. Только в одной Японии используется более 17,7 тыс. патентов, связанных с нечеткими множествами и нечеткими системами. Наибольшее применение нечеткие системы нашли в управлении техническими объектами и технологическими процессами. Например, нечеткие контроллеры на четырех нефтеперерабатывающих заводах Idemitsu Kosan Co. обеспечивают ежегодный экономический эффект более 200 млн японских иен. Изделия с нечетким управлением выпускаются массовыми партиями. Например, японская фирма Omron произвела свыше 7,4 млн нечетких измерителей кровяного давления на 740 млн долл. Немецкая корпорация Siemens продала более двух миллионов стиральных машин с нечетким управлением. Среди нечеткого ассортимента Siemens выделим нечеткие контроллеры для автомобильных трансмиссий Porsche, Peugeot и Hyundai, нечеткие путеводители для навигационных систем Opel и Peugeot, а также систему управления воздушной подушкой, которая использует правила нечеткой классификации, чтобы определить занято сидение в автомобиле пассажиром или багажом.

Популярность теории нечетких множеств в проектировании объясняется тем, что нечеткие системы разрабатываются быстрее, они получаются проще и дешевле четких аналогов. Экспертные знания легко внедрить в нечеткие системы, что позволяет быстро создавать прототипы изделий с прозрачными, т.е. понятными для человека, алгоритмами функционирования. Разработанные в последнее десятилетие методы обучения позволяют настроить нечеткую систему для обеспечения требуемых уровней качества функционирования. Аппаратная реализация нечетких алгоритмов несложная, при этом можно распараллелить вычисления.

Книга раскрывает вопросы проектирования нечетких систем в пакете Fuzzy Logic Toolbox вычислительной среды MATLAB 7 фирмы MathWorks Inc. MATLAB – это вычислительная среда, объединяющая 40 пакетов прикладных инженерных и математических программ, число которых каждый год увеличивает-

. Программный код пакетов открыт, поэтому пользователь может не только про-отреть алгоритмы, но и модифицировать их под свои запросы. Сегодня ATLAB рассматривается как важный элемент инженерного и математического разования в университетах Западной Европы, США, Канады, Израиля и других ран. На многих западных инженерно-исследовательских фирмах он использу-ся как базовый инструмент моделирования и проектирования. В новом веке зна-ия MATLAB стали востребованными и на постсоветской территории.

Книга состоит из четырех глав.

Первая глава является кратким введением в теорию нечетких множеств и не-ткую логику. В ней приводятся базовые теоретические сведения, необходимые ля осмысленной эксплуатации нечетких систем. Для повышения усваиваемости ретического материала в главе помещены многочисленные примеры.

Вторая глава содержит теоретический материал по проектированию нечет-их систем. Она состоит из трех разделов. В первом разделе излагается теория не-ткой идентификации, т.е. методы построения нечетких моделей по результатам аблюдений. Рассматриваются настройка нечетких баз знаний Мамдани и Сугено ля моделирования непрерывных нелинейных зависимостей, а также обучение ечетких классификаторов по экспертно-экспериментальной информации. Во тором разделе рассматриваются методы нечеткой кластеризации и их примене-ие для экстракции нечетких <Если – то> правил из экспериментальных данных. третьем разделе описывается принятие решений в нечетких условиях на основе лияния целей и ограничений по принципу Беллмана–Заде. Раздел иллюстриру-тся нечетким многокритериальным анализом проектов создания бренда.

Третья глава является руководством проектировщика нечетких систем в па-ете Fuzzy Logic Toolbox вычислительной среды MATLAB. Помимо традицион-ого для подобных руководств перевода с английского описания функций и GUI-модулей пакета, в главе значительное внимание уделено демо-примерам, ил-юстрирующим основные этапы проектирования нечетких систем различного на-значения с помощью Fuzzy Logic Toolbox. Для быстрого старта в начале главы из-ожены пошаговые примеры законченного проектирования нечетких систем, тре-ующего минимальных знаний по пакету. Описано взаимодействие Fuzzy Logic oolbox с другими пакетами расширения, а также использование разработанных ечетких систем вне среды MATLAB.

Четвертая глава содержит авторские разработки, расширяющие пакет Fuzzy ogic Toolbox для проектирования нечетких классификаторов, построения иерар-ических нечетких систем, обучения нечетких баз знаний типа Мамдани, а также ля логического вывода при нечетких исходных данных.

Книга завершается списком интернет-ресурсов по нечетким системам.

Теоретический материал книги использовался при чтении курса лекций «Интел-ектуальные технологии» студентам пятого курса Винницкого национального техни-еского университета в 1999–2005 гг. Практический материал книги апробирован на сайтах русскоязычных пользователей системы MATLAB *http://www.matlab.ru* и *http://matlab.exponenta.ru* в разделе Fuzzy Logic Toolbox. Высокий интерес поль-ователей сайтов к этому разделу подвигнул автора к написанию настоящей книги.

Книга предназначена для проектировщиков систем. Она будет полезной научным сотрудникам, аспирантам и студентам старших курсов, интересующимся применением теории нечетких множеств в управлении, идентификации, принятии решений, обработке сигналов и в других областях. Автор стремился излагать материал без сложных математических выкладок, поэтому изучение основной части книги будет по силам студентам первого и второго курсов технического университета. Однако, некоторые разделы требуют алгоритмико-математических знаний, которые обычно получают студенты к концу третьего курса. Предполагается, что читатель имеет базовые навыки работы в среде MATLAB на уровне материала 73-страничной книги «Начало работы с MATLAB», которая выставлена на сайте *http://matlab.expopnenta.ru* в разделе «MATLAB».

Автор благодарит доктора технических наук, профессора А.П.Ротштейна из Иерусалимского политехнического института за формирование интереса к теории нечетких множеств, навыков научной работы в области интеллектуальных технологий и нацеленности на использование вычислительной среды MATLAB, кандидата физико-математических наук С.Б.Ильичеву из SoftLine Inc. за предложение вести раздел Fuzzy Logic Toolbox на сайте *http://www.matlab.ru*, Корнея Еспосито из MathWorks Inc. за предоставление лицензии на MATLAB, а также ректора Винницкого национального технического университета академика Б.И.Мокина за поддержку исследований по нечетким системам.

. Программный код пакетов открыт, поэтому пользователь может не только про-ютреть алгоритмы, но и модифицировать их под свои запросы. Сегодня ATLAB рассматривается как важный элемент инженерного и математического разования в университетах Западной Европы, США, Канады, Израиля и других ран. На многих западных инженерно-исследовательских фирмах он использу-ся как базовый инструмент моделирования и проектирования. В новом веке зна-ия MATLAB стали востребованными и на постсоветской территории.

Книга состоит из четырех глав.

Первая глава является кратким введением в теорию нечетких множеств и не-ткую логику. В ней приводятся базовые теоретические сведения, необходимые ля осмысленной эксплуатации нечетких систем. Для повышения усваиваемости оретического материала в главе помещены многочисленные примеры.

Вторая глава содержит теоретический материал по проектированию нечет-их систем. Она состоит из трех разделов. В первом разделе излагается теория не-еткой идентификации, т.е. методы построения нечетких моделей по результатам аблюдений. Рассматриваются настройка нечетких баз знаний Мамдани и Сугено ля моделирования непрерывных нелинейных зависимостей, а также обучение ечетких классификаторов по экспертно-экспериментальной информации. Во тором разделе рассматриваются методы нечеткой кластеризации и их примене-ие для экстракции нечетких <Если – то> правил из экспериментальных данных. В третьем разделе описывается принятие решений в нечетких условиях на основе лияния целей и ограничений по принципу Беллмана–Заде. Раздел иллюстриру-тся нечетким многокритериальным анализом проектов создания бренда.

Третья глава является руководством проектировщика нечетких систем в па-кете Fuzzy Logic Toolbox вычислительной среды MATLAB. Помимо традицион-ого для подобных руководств перевода с английского описания функций и GUI-модулей пакета, в главе значительное внимание уделено демо-примерам, ил-юстрирующим основные этапы проектирования нечетких систем различного на-значения с помощью Fuzzy Logic Toolbox. Для быстрого старта в начале главы из-ожены пошаговые примеры законченного проектирования нечетких систем, тре-бующего минимальных знаний по пакету. Описано взаимодействие Fuzzy Logic Toolbox с другими пакетами расширения, а также использование разработанных нечетких систем вне среды MATLAB.

Четвертая глава содержит авторские разработки, расширяющие пакет Fuzzy Logic Toolbox для проектирования нечетких классификаторов, построения иерар-хических нечетких систем, обучения нечетких баз знаний типа Мамдани, а также для логического вывода при нечетких исходных данных.

Книга завершается списком интернет-ресурсов по нечетким системам.

Теоретический материал книги использовался при чтении курса лекций «Интел-лектуальные технологии» студентам пятого курса Винницкого национального техни-ческого университета в 1999–2005 гг. Практический материал книги апробирован на сайтах русскоязычных пользователей системы MATLAB *http://www.matlab.ru* и *http://matlab.expopnenta.ru* в разделе Fuzzy Logic Toolbox. Высокий интерес поль-

Книга предназначена для проектировщиков систем. Она будет полезной научным сотрудникам, аспирантам и студентам старших курсов, интересующимся применением теории нечетких множеств в управлении, идентификации, принятии решений, обработке сигналов и в других областях. Автор стремился излагать материал без сложных математических выкладок, поэтому изучение основной части книги будет по силам студентам первого и второго курсов технического университета. Однако, некоторые разделы требуют алгоритмико-математических знаний, которые обычно получают студенты к концу третьего курса. Предполагается, что читатель имеет базовые навыки работы в среде MATLAB на уровне материала 73-страничной книги «Начало работы с MATLAB», которая выставлена на сайте *http://matlab.expopnenta.ru* в разделе «MATLAB».

Автор благодарит доктора технических наук, профессора А.П.Ротштейна из Иерусалимского политехнического института за формирование интереса к теории нечетких множеств, навыков научной работы в области интеллектуальных технологий и нацеленности на использование вычислительной среды MATLAB, кандидата физико-математических наук С.Б.Ильичеву из SoftLine Inc. за предложение вести раздел Fuzzy Logic Toolbox на сайте *http://www.matlab.ru*, Корнея Еносито из MathWorks Inc. за предоставление лицензии на MATLAB, а также ректора Винницкого национального технического университета академика Б.И.Мокина за поддержку исследований по нечетким системам.

Глава 1. КРАТКИЙ КУРС ТЕОРИИ НЕЧЕТКИХ МНОЖЕСТВ

В главе излагаются теоретические знания по нечетким множествам, необходимые для осмысленного использования пакета Fuzzy Logic Toolbox вычислительной среды MATLAB. Теоретический материал первой главы базируется на книгах [2, 7, 12, 17, 24, 46]. Все примеры являются оригинальными.

1.1. ИСТОРИЧЕСКИЙ ЭКСКУРС

История нечетких множеств начинается с 1965 г., когда профессор Заде (Zadeh) из Калифорнийского университета в Беркли опубликовал основополагающую статью «Fuzzy Sets» в журнале «Information and Control» [44]. Прилагательное «fuzzy», которое переводится на русский как «нечеткий», «размытый», «ворсистый», «пушистый», введено в название новой теории, чтобы дистанцировать ее от традиционной четкой математики и аристотелевой логики, оперирующих с четкими понятиями «принадлежит – не принадлежит», «истина – ложь». Концепция нечеткого множества зародилась у Заде *«как неудовлетворенность математическими методами классической теории систем, которая вынуждала добиваться искусственной точности, неуместной во многих системах реального мира, особенно в так называемых гуманистических системах, включающих людей»* [4]. Понятие *нечеткого множества* – эта попытка формализации лингвистической информации для построения математических моделей. В основе этого понятия лежит представление о том, что составляющие данное множество элементы, обладающие общим свойством, могут обладать им в различной степени и, следовательно, принадлежать к этому 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 элемент удовлетворяет свойствам множества [7].

Интерес к теории нечетких множеств постоянно усиливается, о чем свидетельствует экспоненциальный рост публикаций в этой области за последние 30 лет (табл. 1.1). Сегодня выпускаются более 10 специализированных международных журналов по теории и применению нечетких множеств, среди которых «Fuzzy Sets and Systems» и «Applied Soft Computing» издательства Elsevier Science,

Таблица 1.1

Количество публикаций, в названиях которых встречаются слова «fuzzy» и «fuzzy control» (данные BISC – Berkeley Initiative in Soft Computing)

Период	База данных INSPEC (технические науки)		База данных MathSciNet (математические науки)	
	«fuzzy»	«fuzzy control»	«fuzzy»	«fuzzy control»
1970–1979	569	38	443	0
1980–1989	2404	214	2465	39
1990–1999	23207	4689	5483	335
2000–22.12.2004	14172	>508	3960	Нет данных

«Journal of Intelligent and Fuzzy Systems» издательства IOS Press, «Intenational Journal of Uncertainty, Fuzziness and Knowledge-Based Systems» издательства World Scientific, «IEEE Transactions on Fuzzy Systems» издательства IEEE, «Soft Computing» и «Fuzzy Optimization and Decision Making» издательства Springer. Много международных журналов публикуют статьи по нечетким множествам, среди них русскоязычные «Кибернетика и системный анализ», «Известия РАН. Теория и системы управления», «Автоматика и телемеханика», «Автоматика и вычислительная техника».

Началом практического применения теории нечетких множеств считают 1973 г., когда Мамдани (Mamdani) и Ассилиан (Assilian) из Лондонского колледжа Королевы Мэри построили первый *нечеткий контроллер* для лабораторной модели парового двигателя [33]. Концепцию первого нечеткого контроллера составляют идеи нечеткого логического вывода и нечеткого алгоритма, изложенные Заде в 1973 г. в статье [45]. Первый промышленный нечеткий контроллер заработал в конце 1970-х годов в Дании – Холмблад (Holmblad) и Остергард (Ostergaard) внедрили нечеткую логику в управление процессом обжига цемента [28]. В 1980-х годах европейские и американские инженерные и научные сообщества весьма скептически восприняли новую теорию. Зато на Востоке нечеткая логика пошла «на ура». Для людей, воспитанных на восточной философии, с ее неоднозначными и расплывчатыми категориями, нечеткая логика сразу стала своей, родной.

В Японии первый нечеткий контроллер разработал Сугено (Sugeno) в 1983 г. по заказу фирмы Fuji El. для системы очистки воды. Четыре года спустя фирма Hitachi разработала нечеткую систему управления движением электропоезда в метро г.Сендай. На 1990 г. в Японии было зарегистрировано 30 патентов, связанных с нечеткой логикой. В начале 1990-х годов японцы поставили нечеткую логику «на конвейер» – началось серийное производство бытовых приборов с нечетким управлением: камеры с автоматической фокусировкой (Canon), кондиционеры воздуха (Mitsubishi), стиральные машины (Panasonic и Matshushita). Тогда же фирмы Honda и Nissan разработали автоматическую трансмиссию с нечетким управлением, а фирма Toshiba – нечеткий контроллер лифта. Нечеткая логика становится маркетинговым оружием на японском рынке – fuzzy-товары раскупаются быстрее.

В 1994 г. Коско (Kosko) доказал теорему о нечеткой аппроксимации [31], согласно которой, любая математическая система может быть аппроксимирована системой на нечеткой логике. Следовательно, с помощью естественно-языковых высказываний <Если – то>, с последующей их формализацией средствами теории нечетких множеств, можно сколь угодно точно отразить произвольную взаимосвязь «входы – выход» без использования сложного аппарата дифференциального и интегрального исчислений, традиционно применяемого в управлении и идентификации. Практические успехи нечеткого управления получили теоретическое обоснование.

Сегодня нечеткая логика рассматривается как стандартный метод моделирования и проектирования. В 1997 г. язык нечеткого управления (Fuzzy Control Language) 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, логистика, поиск информации в Интернете, радиосвязь и телевидение. Спектр приложений очень широкий – от бытовых видеокамер, пылесосов и стиральных машин до средств наведения ракет ПВО и управления боевыми вертолетами и самолетами. По-прежнему лидирует Япония, в которой выпущено свыше 4800 «нечетких» патентов (для сравнения, в США их около 1700). Практический опыт разработки систем на нечетких множествах свидетельствует, что сроки и стоимость их проектирования значительно ниже, чем при использовании традиционного математического аппарата, при этом обеспечиваются требуемые уровни качества. Отец нечеткой логики Лотфи Заде как-то по этому поводу заметил, что *«почти всегда можно сделать такой же самый продукт без нечеткой логики, но с нечеткой будет быстрее и дешевле»* [27].

1.2. НЕЧЕТКИЕ МНОЖЕСТВА

В разделе приводится определение нечетких множеств, рассматриваются их свойства, правила выполнения нечетких теоретико-множественных операций а также способы построения функций принадлежности на основе экспертной информации.

1.2.1. ОСНОВНЫЕ ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Определение 1. *Нечетким множеством* $\tilde{A}$ на универсальном множестве U называется совокупность пар $(\mu_{\tilde{A}}(u), u)$, где $\mu_{\tilde{A}}(u)$ – степень принадлежности элемента $u \in U$ нечеткому множеству $\tilde{A}$. *Степень принадлежности* – это число из диапазона [0, 1]. Чем выше степень принадлежности, тем большей мерой элемент универсального множества соответствует свойствам нечеткого множества.

Определение 2. *Функцией принадлежности* называется функция, позволяющая для произвольного элемента универсального множества вычислить степень его принадлежности нечеткому множеству.

Если универсальное множество является конечным $U = \{u_1, u_2, \ldots, u_k\}$, тогда нечеткое множество $\tilde{A}$ записывается так:

$$\tilde{A} = \sum_{i=1}^{k} \mu_{\tilde{A}}(u_i)/u_i \quad \text{или} \quad \tilde{A} = (\mu_{\tilde{A}}(u_1)/u_1, \mu_{\tilde{A}}(u_2)/u_2, \ldots, \mu_{\tilde{A}}(u_k)/u_k).$$

В случае непрерывного множества U используют такое обозначение: $\tilde{A} = \int_{u \in U} \mu_{\tilde{A}}(u)/u$. Знаки $\sum$ и $\int$ в этих формулах означают совокупность пар $\mu_{\tilde{A}}(u)$ и u.

Пример 1.1. Представить в виде нечеткого множества понятие «мужчина среднего роста» на универсальном множестве {155, 160, 165, 170, 175, 180, 185, 190}.

Одно из возможных решений выглядит так:

$\tilde{A} = (0/155,\ 0{,}1/160,\ 0{,}3/165,\ 0{,}8/170,\ 1/175,\ 1/180,\ 0{,}5/185,\ 0/190)$.

Обычное (четкое) множество A можно определить через характеристическую функцию $\varphi_A(u)$, принимающую одно из двух значений: 0 – если u не принадлежит множеству ($u \notin A$), и 1 – если принадлежит ($u \in A$). Следовательно, четкое множество можно рассматривать как предельный случай нечеткого множества, функция принадлежности которого принимает лишь бинарные значения.

Определение 3. *Лингвистической переменной* называется переменная, значениями которой могут быть слова или словосочетания некоторого естественного языка.

Для человека привычнее задавать значения переменной не числами, а словами. Ежедневно мы принимаем решения на основе лингвистической информации типа: «очень высокая температура»; «утомительная 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 слов.

Определение 4. *Терм-множеством* называется множество всех возможных значений лингвистической переменной.

Определение 5. *Термом* называется любой элемент терм-множества. Терм задается нечетким множеством посредством функции принадлежности.

Пусть переменная «скорость автомобиля» может принимать значения *«низкая»*, *«средняя»*, *«высокая»* и *«очень высокая»*. В этом случае лингвистической переменной является *«скорость автомобиля»*, термами – лингвистические оценки *«низкая»*, *«средняя»*, *«высокая»* и *«очень высокая»*, которые и составляют терм-множество.

1.2.2. СВОЙСТВА НЕЧЕТКИХ МНОЖЕСТВ

Определение 6. *Высотой нечеткого множества* называется верхняя грань его функции принадлежности $\text{height}(\tilde{A}) = \sup_{u \in U} \mu_{\tilde{A}}(u)$. Для дискретного универсального множества супремум становится максимумом, а значит, высотой нечеткого множества будет максимум степеней принадлежности его элементов.

Определение 7. *Нечеткое множество* $\tilde{A}$ называется *нормальным*, если его высота равна единице. Нечеткое множество, не являющееся нормальным, называют *субнормальным*. *Нормализация*, т.е. преобразование субнормального нечеткого множества $\tilde{A}'$ к нормальному $\tilde{A}$ определяется так:

$$\tilde{A} = \text{norm}(\tilde{A}') \Leftrightarrow \mu_{\tilde{A}}(u) = \frac{\mu_{\tilde{A}'}(u)}{\text{height}(\tilde{A}')}, \quad \forall u \in U.$$

В качестве примера на рис. 1.1 показана нормализация нечеткого множества $\tilde{A}'$ с функцией принадлежности $\mu_{\tilde{A}'}(u) = \frac{0,6}{1+(10-u)^2}$.

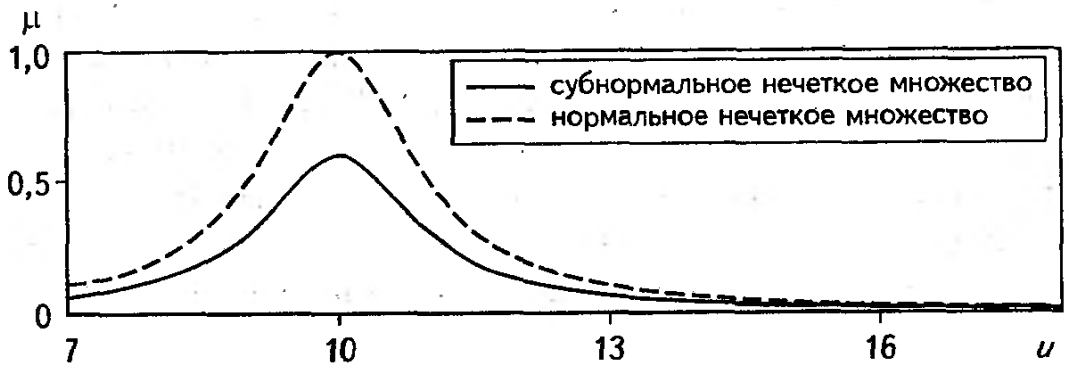


Рис. 1.1. Нормализация субнормального нечеткого множества

Определение 8. *Носителем нечеткого множества* $\tilde{A}$ называется четкое подмножество универсального множества U, элементы которого имеют ненулевые степени принадлежности:

$$\text{supp}(\tilde{A}) = \{u : \mu_{\tilde{A}}(u) > 0\}.$$

Определение 9. *Нечеткое множество* называется *пустым*, если его носитель является пустым множеством.

Определение 10. *Ядром нечеткого множества* $\tilde{A}$ называется четкое подмножество универсального множества U, элементы которого имеют степени принадлежности, равные единице:

$$\text{core}(\tilde{A}) = \{u : \mu_{\tilde{A}}(u) = 1\}.$$

Определение 11. α-*сечением* (α-*срезом или множеством* α-*уровня*) нечеткого множества $\tilde{A}$ называется четкое подмножество универсального множества U, элементы которого имеют степени принадлежности, большие или равные α:

$$A_\alpha = \{u : \mu_A(u) \geq \alpha\}, \quad \alpha \in [0;\ 1].$$

Носитель и ядро можно рассматривать как сечения нечеткого множества на нулевом и единичном α-уровне. Ядро субнормального нечеткого множества пустое. На рис. 1.2 проиллюстрированы определения носителя, ядра, α-сечения и α-уровня нечеткого множества.

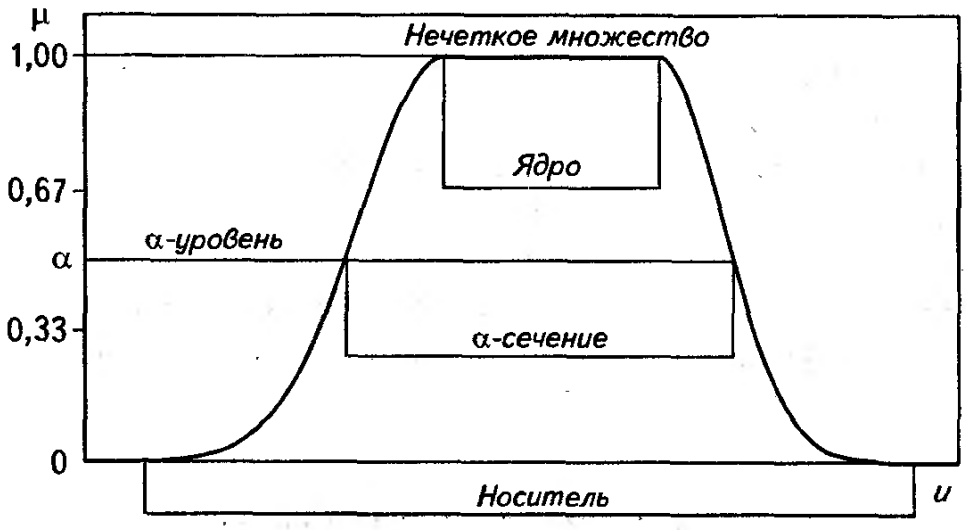


Рис. 1.2. Ядро, носитель, α-уровень и α-сечение нечеткого множества

Пример 1.2. Найти носитель, ядро и α-сечение (при α = 0,25) нечеткого множества $\tilde{A}$ из примера 1.1.

По определениям 8, 10 и 11 получаем:

- носитель нечеткого множества – $\text{supp}(\tilde{A}) = \{160, 165, 170, 175, 180, 185\}$;
- ядро нечеткого множества – $\text{core}(\tilde{A}) = \{175, 180\}$;
- α-сечение нечеткого множества – $A_{0,25} = \{165, 170, 175, 180, 185\}$.

Определение 12. *Нечеткое множество* $\tilde{A}$ называется *выпуклым*, если:

$$\mu_A(\lambda u_1 + (1-\lambda)u_2) \geq \min(\mu_A(u_1),\ \mu_A(u_2)), \quad u_1,\ u_2 \in U, \quad \lambda \in [0,\ 1].$$

Альтернативное определение: нечеткое множество будет выпуклым, если все его α-сечения – выпуклые множества. Примеры выпуклого и невыпуклого нечетких множеств показаны на рис. 1.3.

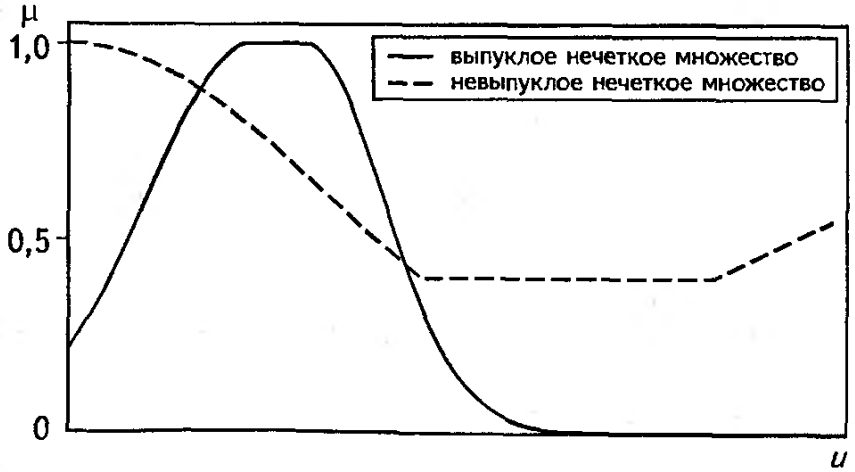


Рис. 1.3. Выпуклое и невыпуклое нечеткие множества

Определение 13. Нечеткие множества $\tilde{A}$ и $\tilde{B}$ равны ($\tilde{A}=\tilde{B}$), если $\mu_A(u)=\mu_B(u), \forall u \in U$.

Определение 14. *Дефаззификацией* называется преобразование нечеткого множества в четкое число.

В теории нечетких множеств дефаззификация аналогична нахождению характеристик положения случайных величин (математического ожидания, моды, медианы) в теории вероятностей. Простейшим способом дефаззификации является выбор четкого числа с максимальной степенью принадлежности. Пригодность этого способа ограничивается лишь одноэкстремальными функциями принадлежности. Для многоэкстремальных функций принадлежности применяются такие методы дефаззификации: центр тяжести (Centroid); медиана (Bisector); центр максимумов (Mean of Maximums); наибольший из максимумов (Largest of Maximums); наименьший из максимумов (Smallest of Maximums). Расчетные формулы этих методов дефаззификации для нечетких множеств на непрерывных и дискретных носителях приведены в табл. 1.2.

Таблица 1.2

Дефаззификация различными методами

Метод дефаззификации	Нечеткое множество	
	$\tilde{A}=\int\limits_{u\in[\underline{u},\bar{u}]}\mu_A(u)/u$	$\tilde{A}=(\mu_A(u_1)/u_1,\ldots,\mu_A(u_k)/u_k).$
Центр тяжести	$\dfrac{\int\limits_{\underline{u}}^{\bar{u}} u\,\mu_A(u)\,du}{\int\limits_{\underline{u}}^{\bar{u}} \mu_A(u)\,du}$	$\dfrac{\sum\limits_{i=1}^{k} u_i\,\mu_A(u_i)}{\sum\limits_{i=1}^{k}\mu_A(u_i)}$
Медиана	Найти такое число a, чтобы: $\int\limits_{\underline{u}}^{a}\mu_A(u)\,du=\int\limits_{a}^{\bar{u}}\mu_A(u)\,du$	$\min(u_j)$ для $\forall j:\ \sum\limits_{i=1}^{j}\mu_A(u_i)\ge\frac{1}{2}\sum\limits_{i=1}^{k}\mu_A(u_i)$
Центр максимумов	$\dfrac{\int\limits_G u\,du}{\int\limits_G du}$, где $G=\arg\sup\limits_{u\ u\in\mathrm{supp}\,\tilde{A}}(\mu_A(u))$	$\dfrac{\sum\limits_{u_j\in G} u_j}{\lvert G\rvert}$
Наименьший из максимумов	$\min(G)$	$\min(G)$
Наибольший из максимумов	$\max(G)$	$\max(G)$

Пример 1.3. Провести дефаззификацию нечеткого множества «мужчина среднего роста» из примера 1.1 по методу центра тяжести.

По формуле из табл. 1.2 получаем:

$$A=\frac{0\cdot155+0{,}1\cdot160+0{,}3\cdot165+0{,}8\cdot170+1\cdot175+1\cdot180+0{,}5\cdot185+0\cdot190}{0+0{,}1+0{,}3+0{,}8+1+1+0{,}5+0}=175{,}4.$$

1.2.3. ОПЕРАЦИИ НАД НЕЧЕТКИМИ МНОЖЕСТВАМИ

Нечеткие теоретико-множественные операции объединения, пересечения и дополнения могут быть обобщены из теории обычных множеств. В теории нечетких множеств степень принадлежности может принимать значения из интервала [0, 1], а не ограничена бинарными значениями 0 и 1, как в обычной теории множеств. Поэтому, нечеткие теоретико-множественные операции могут быть определены по-разному. Ясно, что выполнение нечетких операций объединения, пересечения и дополнения над обычными множествами должно дать такие же результаты, как и при использовании традиционных (канторовских) теоретико-множественных операций. Ниже приведены определения нечетких теоретико-множественных операций, предложенные Заде.

Определение 15. *Дополнением нечеткого множества* $\widetilde{A}$, заданного на U, называется нечеткое множество $\overline{A}$ с функцией принадлежности $\mu_{\overline{A}}(u) = 1 - \mu_{A}(u)$ для всех $u \in U$. На рис. 1.4 показан пример выполнения операции нечеткого дополнения.

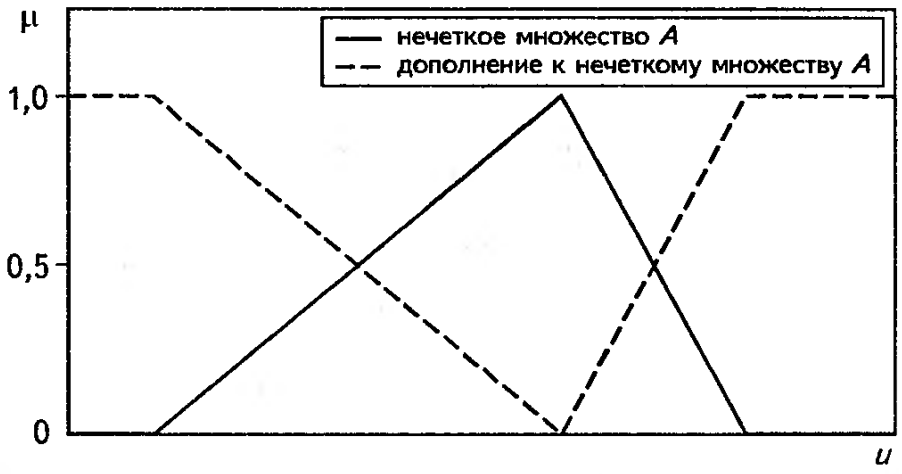


Рис. 1.4. Дополнение нечеткого множества

Определение 16. *Пересечением нечетких множеств* $\widetilde{A}$ и $\widetilde{B}$, заданных на U, называется нечеткое множество $\widetilde{C} = \widetilde{A} \cap \widetilde{B}$ с функцией принадлежности $\mu_C(u) = \min(\mu_A(u), \mu_B(u))$ для всех $u \in U$.

Определение 17. *Объединением нечетких множеств* $\widetilde{A}$ и $\widetilde{B}$, заданных на U, называется нечеткое множество $D = \widetilde{A} \cup \widetilde{B}$ с функцией принадлежности $\mu_D(u) = \max(\mu_A(u), \mu_B(u))$ для всех $u \in U$.

Определения обобщенных операций нечеткого пересечения и объединения – треугольной нормы (t-нормы) и треугольной конормы (t-конормы или s-нормы) приведены ниже.

Определение 18. *Треугольной нормой (t-нормой)* называется бинарная операция $\wedge$ на единичном интервале $[0, 1] \times [0, 1] \to [0, 1]$, удовлетворяющая следующим аксиомам для любых $a, b, c \in [0, 1]$:

1) $a \wedge 1 = a$ (граничное условие);

2) $a \wedge b \le a \wedge c$, если $b \le c$ (монотонность);

3) $a \wedge b = b \wedge a$ (коммутативность);

4) $a \wedge (b \wedge c) = (a \wedge b) \wedge c$ (ассоциативность).

Часто используют такие t-нормы:

$\min(a, b)$ – пересечение по Заде;

$a - b$ – вероятностное пересечение (умножение);

$\max(a + b - 1, 0)$ – пересечение по Лукасевичу.

Определение 19. *Треугольной конормой (s-нормой)* называется бинарная операция $\vee$ на единичном интервале $[0, 1] \times [0, 1] \to [0, 1]$, удовлетворяющая следующим аксиомам для любых $a, b, c \in [0, 1]$:

1) $a \vee 0 = a$ (граничное условие);

2) $a \vee b \le a \vee c$, если $b \le c$ (монотонность);

3) $a \vee b = b \vee a$ (коммутативность);

4) $a \vee (b \vee c) = (a \vee b) \vee c$ (ассоциативность).

Часто используют такие s-нормы:

$\max(a, b)$ – объединение по Заде;

$a + b - ab$ – вероятностное ИЛИ;

$\min(a + b, 1)$ – объединение по Лукасевичу.

Рис. 1.5. Пересечение нечетких множеств с использованием различных t-норм

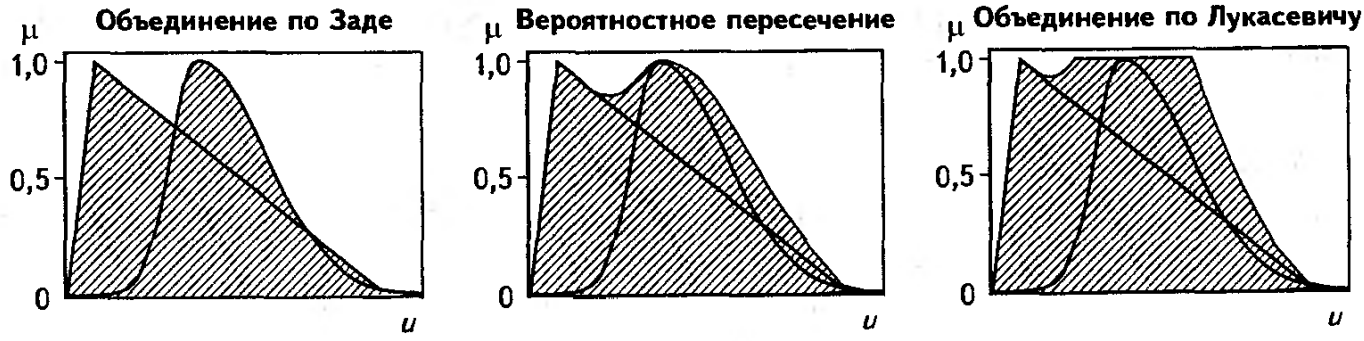


Рис. 1.6. Объединение нечетких множеств с использованием различных s-норм

Примеры пересечения и объединения нечетких множеств по указанным треугольным нормам показаны на рис. 1.5 и 1.6. Наиболее известные треугольные нормы сведены в табл. 1.3.

Таблица 1.3

Примеры треугольных норм [38]

$a \wedge b$ (t-норма)	$a \vee b$ (s-норма)	Параметры
$\min(a, b)$	$\max(a, b)$	–
ab	$a+b-ab$	–
$\max(a+b-1, 0)$	$\min(a+b, 1)$	–
$\begin{cases} a, \text{ если } b=1 \\ b, \text{ если } a=1 \\ 0, \text{ если } a, b \neq 1 \end{cases}$	$\begin{cases} a, \text{ если } b=0 \\ b, \text{ если } a=0 \\ 1, \text{ если } a, b \neq 0 \end{cases}$	–
$\dfrac{ab}{\gamma+(1-\gamma)(a+b-ab)}$	$\dfrac{a+b-(2-\gamma)ab}{1-(1-\gamma)ab}$	$\gamma > 0$
$\dfrac{ab}{\max(a, b, \alpha)}$	$\dfrac{a+b-ab-\min(a, b, 1-\alpha)}{\max(1-a, 1-b, \alpha)}$	$\alpha \in [0, 1]$
$\left(1+\sqrt[\lambda]{\left(\frac{1}{a}-1\right)^{\lambda}+\left(\frac{1}{b}-1\right)^{\lambda}}\right)^{-1}$	$\left(1+\sqrt[\lambda]{\left(\frac{1}{a}-1\right)^{-\lambda}+\left(\frac{1}{b}-1\right)^{-\lambda}}\right)^{-1}$	$\lambda > 0$
$1-\sqrt[\lambda]{(1-a)^{\lambda}+(1-b)^{\lambda}-(1-a)^{\lambda}(1-b)^{\lambda}}$	$\sqrt[\lambda]{a^{\lambda}+b^{\lambda}-a^{\lambda}b^{\lambda}}$	$\lambda > 0$
$\max(1-\sqrt[\rho]{(1-a)^{\rho}+(1-b)^{\rho}}, 0)$	$\min(1-\sqrt[\rho]{a^{\rho}+b^{\rho}}, 1)$	$\rho \geq 1$
$\log_{\omega}^{1+(\omega^{a}-1)(\omega^{b}-1)/(\omega-1)}$	$1-\log_{\omega}^{1+(\omega^{1-a}+\omega^{1-b})/(\omega-1)}$	$\omega > 0,\ \omega \neq 1$
$\max\left(\dfrac{a+b-1+\lambda ab}{1+\lambda}, 0\right)$	$\min(a+b+\lambda ab, 1)$	$\lambda \geq -1$

1.2.4. ФУНКЦИИ ПРИНАДЛЕЖНОСТИ

Практическое использование теории нечетких множеств предполагает наличие функций принадлежностей, которыми описываются лингвистические термы «низкий», «средний», «высокий» и т.п. Задача построения функций принадлежности ставится следующим образом. Даны два множества: множество термов $L = \{l_1, l_2, \ldots, l_m\}$ и универсальное множество $U = \{u_1, u_2, \ldots, u_n\}$. Нечеткое множество $\tilde{l}$ для задания лингвистического терма l_j на универсальном множестве U представляется в виде:

$$\tilde{l}_j = \left(\frac{\mu_{l_j}(u_1)}{u_1}, \frac{\mu_{l_j}(u_2)}{u_2}, \ldots, \frac{\mu_{l_j}(u_n)}{u_n} \right), \quad j = \overline{1,\ m}.$$

Необходимо определить степени принадлежностей элементов множества U к элементам из множества L, т.е. найти $\mu_{l_j}(u_i)$ для всех $j = \overline{1,\ m}$ и $i = \overline{1,\ n}$.

Ниже рассматриваются два метода построения функций принадлежности. Первый метод основан на статистической обработке мнений группы экспертов.

Второй метод базируется на парных сравнениях, выполняемых одним экспертом. В конце подраздела приводятся аналитические выражения, которые часто используются для аппроксимации функций принадлежностей, построенных по экспертной информации.

При построении функций принадлежности по первому методу каждый эксперт заполняет анкету, в которой указывает свое мнение о наличии у элементов u_i $(i=\overline{1,\ n})$ свойств нечеткого множества $\tilde{l}_j$ $(j=\overline{1,\ m})$. Анкета имеет следующий вид:

	u_1	u_2	…	u_n
$\tilde{l}_1$				
$\tilde{l}_2$				
…				
$\tilde{l}_m$				

Введем обозначения: K – количество экспертов; $b^k_{j,i}$ – мнение k-го эксперта о наличии у элемента u_i свойств нечеткого множества $\tilde{l}_j$, $k=\overline{1,K}$, $i=\overline{1,n}$, $j=\overline{1,m}$. Будем считать, что экспертные оценки бинарные, т.е., $b^k_{j,i}\in\{0;\ 1\}$, где 1 указывает на наличие у элемента u_i свойств нечеткого множества $\tilde{l}_j$, а 0 – на их отсутсвие. По результатам анкетирования степени принадлежности нечеткому множеству $\tilde{l}_j$ рассчитываются так:

$$\mu_{\tilde{l}_j}(u_i)=\frac{1}{K}\sum_{k=\overline{1,K}} b^k_{j,i},\quad i=\overline{1,\ n}. \tag{1.1}$$

Таблица 1.4

Результаты опроса экспертов

	Терм	[160, 165)	[165, 170)	[170, 175)	[175, 180)	[180, 185)	[185, 190)	[190, 195)	[195, 200)
Эксперт 1	Низкий	1	1	1	0	0	0	0	0
	Средний	0	0	1	1	1	0	0	0
	Высокий	0	0	0	0	0	1	1	1
Эксперт 2	Низкий	1	1	1	0	0	0	0	0
	Средний	0	0	1	1	0	0	0	0
	Высокий	0	0	0	0	1	1	1	1
Эксперт 3	Низкий	1	0	0	0	0	0	0	0
	Средний	0	1	1	1	1	1	0	0
	Высокий	0	0	0	0	0	1	1	1
Эксперт 4	Низкий	1	1	1	0	0	0	0	0
	Средний	0	0	0	1	1	1	0	0
	Высокий	0	0	0	0	0	0	1	1
Эксперт 5	Низкий	1	1	0	0	0	0	0	0
	Средний	0	1	1	1	0	0	0	0
	Высокий	0	0	0	1	1	1	1	1

Пример 1.4. Построить функции принадлежности термов «низкий», «средний», «высокий», используемых для лингвистической оценки переменной «рост мужчины». Результаты опроса пяти экспертов сведены в табл. 1.4.

Результаты обработки экспертных мнений сведены в табл. 1.5. Числа над пунктирной линией соответствуют количеству голосов, отданных экспертами за принадлежность нечеткому множеству соответствующего элемента универсального множества. Числа под пунктирной линией – степени принадлежности, рассчитанные по формуле (1.1). Графики функций принадлежностей показаны на рис. 1.7.

Таблица 1.5

Результаты обработки мнений экспертов

Терм	[160, 165)	[165, 170)	[170, 175)	[175, 180)	[180, 185)	[185, 190)	[190, 195)	[195, 200)
Низкий	5	4	3	0	0	0	0	0
	1	0,8	0,6	0	0	0	0	0
Средний	0	2	4	5	3	2	0	0
	0	0,4	0,8	1	0,6	0,4	0	0
Высокий	0	0	0	1	2	4	5	5
	0	0	0	0,2	0,4	0,8	1	1

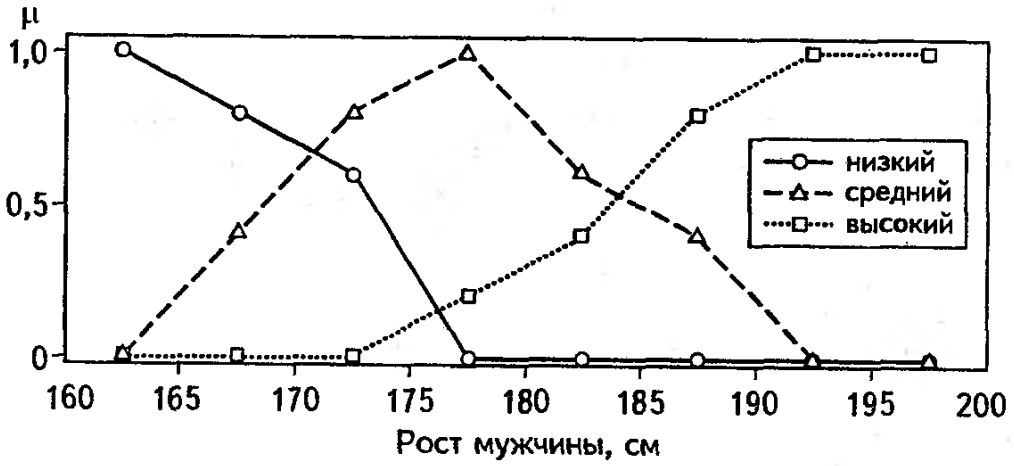


Рис. 1.7. Функции принадлежности нечетких множеств (из примера 1.4)

При построении функций принадлежности по второму методу для каждой пары элементов универсального множества эксперт оценивает преимущество одного элемента над другим по отношению к свойству нечеткого множества. Такие парные сравнения удобно представлять следующей матрицей:

$$\mathbf{A} = \begin{array}{c} \\ u_1 \\ u_2 \\ \dots \\ u_n \end{array} \begin{array}{c} \begin{array}{cccc} u_1 & u_2 & \dots & u_n \end{array} \\ \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & & \dots & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \end{array},$$

где a_{ij} – уровень преимущества элемента u_i над u_j $(i,\ j = \overline{1,\ n})$, определяемый по девятибалльной шкале Саати [19]:

1 – если преимущество элемента u_i над элементом u_j *отсутствует*;
3 – если преимущество u_i над u_j *слабое*;
5 – если преимущество u_i над u_j *существенное*;
7 – если преимущество u_i над u_j *явное*;
9 – если преимущество u_i над u_j *абсолютное*;
2, 4, 6, 8 – *промежуточные* сравнительные оценки: 2 – *почти слабое* преимущество, 4 – *почти существенное* преимущество, 6 – *почти явное* преимущество и 8 – *почти абсолютное* преимущество.

Матрица парных сравнений является диагональной ($a_{ij} = 1,\ i = \overline{1,\ n}$) и обратно симметричной ($a_{ij} = 1 / a_{ij},\ i, j = \overline{1,\ n}$).

Степени принадлежности принимают равными соответствующим координатам собственного вектора $W = (w_1,\ w_2, \ldots,\ w_n)^T$ матрицы парных сравнений A:

$$\mu(u_1) = w_i,\ \ i = \overline{1,\ n}. \tag{1.2}$$

Собственный вектор находят из следующей системы уравнений:

$$\begin{cases} \mathbf{A}W = \lambda_{\max} W, \\ w_1 + w_2 + \ldots + w_n = 1, \end{cases} \tag{1.3}$$

где $\lambda_{\max}$ – максимальное собственное значение матрицы A.

Пример 1.5. Построить функцию принадлежности нечеткого множества «высокий мужчина» на универсальном множестве {170, 175, 180, 185, 190, 195}.

Предположим, что известны такие парные сравнения:

- отсутствие преимущества 195 над 190;
- существенное преимущество 195 над 180;
- абсолютное преимущество 195 над 170;
- почти существенное преимущество 190 над 180;
- почти абсолютное преимущество 190 над 170;
- существенное преимущество 185 над 175;
- слабое преимущество 195 над 185;
- почти абсолютное преимущество 195 над 175;
- слабое преимущество 190 над 185;
- явное преимущество 190 над 175;
- почти существенное преимущество 185 над 180;
- почти явное преимущество 185 над 170;
- слабое преимущество 180 над 175;
- почти существенное преимущество 180 над 170;
- почти слабое преимущество 175 над 170.

Парные сравнения запишем следующей матрицей:

$$\mathbf{A} = \begin{array}{c} \\ 170 \\ 175 \\ 180 \\ 185 \\ 190 \\ 195 \end{array} \begin{array}{c} \begin{array}{cccccc} 170 & 175 & 180 & 185 & 190 & 195 \end{array} \\ \begin{bmatrix} 1 & 1/2 & 1/4 & 1/6 & 1/8 & 1/9 \\ 2 & 1 & 1/3 & 1/5 & 1/7 & 1/8 \\ 4 & 3 & 1 & 1/4 & 1/4 & 1/5 \\ 6 & 5 & 4 & 1 & 1/3 & 1/3 \\ 8 & 7 & 4 & 3 & 1 & 1 \\ 9 & 8 & 5 & 3 & 1 & 1 \end{bmatrix} \end{array}.$$

Собственные значения матрицы парных сравнений A равны:
6,2494;
0,0318 + 1,2230*i*;
0,0318 – 1,2230*i*;
–0,1567 + 0,2392*i*;
–0,1567 – 0,2392*i*;
0,0004.

Следовательно, $\lambda_{max} = 6{,}2494$. Степени принадлежности, найденные по формулам (1.3) и (1.2), приведены в табл. 1.6. Нечеткое множество получилось субнормальным. Для его нормализации разделим все степени принадлежности на максимальное значение, т.е. на 0,3494. Графики функций принадлежности субнормального и нормального нечетких множеств «высокий мужчина» изображены на рис. 1.8. Разница между λ_{max} и n служит мерой несогласованности парных сравнений эксперта. В примере 1.5 $\lambda_{max} = 6{,}2494$, а $n = 6$. Следовательно, мера несогласованности равна 0,2494.

Таблица 1.6

Функции принадлежности нечеткого множества «высокий мужчина»

u_i	170	175	180	185	190	195
$\mu_{\text{«высокий мужчина»}}(u_i)$ для субнормального нечеткого множества	0,0284	0,0399	0,0816	0,1754	0,3254	0,3494
$\mu_{\text{«высокий мужчина»}}(u_i)$ для нормального нечеткого множества	0,0813	0,1141	0,2335	0,5021	0,9314	1,0000

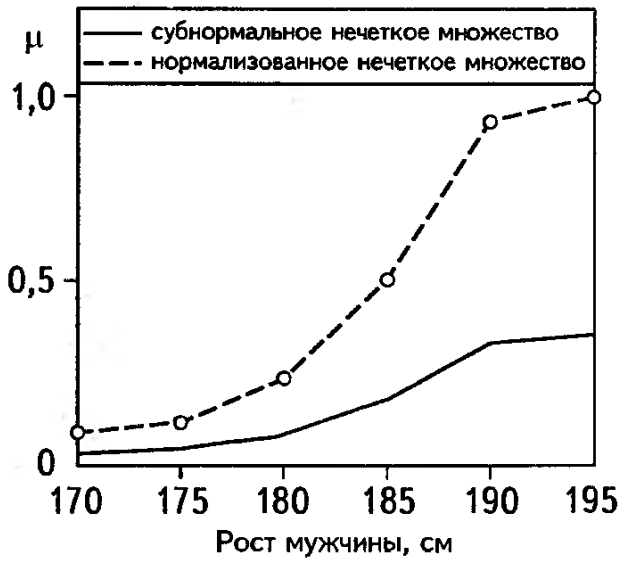


Рис. 1.8. Функции принадлежности нечеткого множества «высокий мужчина» (к примеру 1.5)

Функции принадлежности удобно задавать в параметрической форме. В этом случае задача построения функции принадлежности сводится к определению ее параметров. Обычно функции принадлежности имеют 2, 3 или 4 параметра. Наибольшее распространение получили треугольная, трапециевидная, гауссова и сигмоидная функции принадлежности. Для представления четких чисел в виде нечетких множеств применяется синглтонная функция принадлежности. Аналитические выражения и графики этих функций принадлежности приведены в табл. 1.7 и на рис. 1.9.

Таблица 1.7

Популярные параметрические функции принадлежности

Наименование функции	Аналитическое выражение	Интерпретация параметрв
Треугольная	$\mu(u)=\begin{cases}0, & u\le a \text{ или } u\ge c\\ \dfrac{u-a}{b-a}, & a<u\le b\\ \dfrac{c-u}{c-b}, & b<u<c\end{cases}$	(a, c) – носитель нечеткого множества – пессимистическая оценка нечеткого числа; b – координата максимума – оптимистическая оценка нечеткого числа
Трапециевидная	$\mu(u)=\begin{cases}0, & u\le a \text{ или } u\ge d\\ \dfrac{u-a}{b-a}, & a\le u\le b\\ 1, & b\le u\le c\\ \dfrac{d-u}{d-c}, & c\le u\le d\end{cases}$	(a, d) – носитель нечеткого множества – пессимистическая оценка нечеткого числа; $[b, c]$ – ядро нечеткого множества – оптимистическая оценка нечеткого числа
Гауссова	$\mu(u)=\exp\left(-\dfrac{(u-b)^2}{2c^2}\right)$	b – координата максимума c – коэффициент концентрации
Сигмоидная	$\mu(u)=\dfrac{1}{1+\exp(-a(u-c))}$	a – коэффициент крутизны c – координата перехода через 0,5
Синглтонная	$\mu(u)=\begin{cases}1, & u=a\\ 0, & u\ne a\end{cases}$	a – четкое число, представляемое в виде нечеткого множества

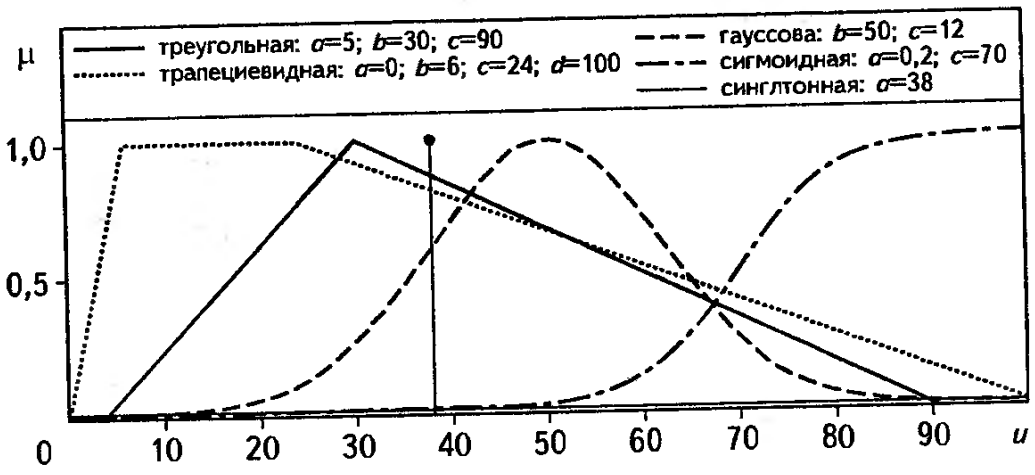


Рис. 1.9. Графики популярных функций принадлежности

1.3. НЕЧЕТКАЯ АРИФМЕТИКА

В разделе рассматриваются способы расчета значений четких функций от нечетких аргументов. Материал основывается на понятиях нечеткого числа и принципа нечеткого обобщения. В конце раздела приводятся правила выполнения арифметических операций над нечеткими числами.

Определение 20. *Нечетким числом* называется выпуклое нормальное нечеткое множество с кусочно-непрерывной функцией принадлежности, заданное на множестве действительных чисел. Например, нечеткое число «около 10» можно задать следующей функцией принадлежности:

$$\mu_{\text{«около 10»}}(u) = \frac{1}{1+(u-10)^2}.$$

Определение 21. *Нечеткое число* $\tilde{A}$ называется *положительным*, если $\mu_{\tilde{A}}(u) = 0, \forall u < 0$. Аналогично, *нечеткое число* $\tilde{A}$ называется отрицательным, если $\mu_{\tilde{A}}(u) = 0, \forall u > 0$.

Определение 22. *Принцип нечеткого обобщения Заде.* Если $y = f(x_1, x_2, \ldots, x_n)$ – функция от n независимых аргументов $x_1, x_2, \ldots, x_n$, которые заданы нечеткими числами $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n$, соответственно, то значением функции $\tilde{y} = f(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$ называется нечеткое число $\tilde{y}$ с функцией принадлежности:

$$\mu_{\tilde{y}}(y^*) = \sup_{\substack{y^* = f(x_1^*, x_2^*, \ldots, x_n^*) \\ x_i^* \in \operatorname{supp}(\tilde{x}_i),\ i=1,n}} \min_{i=1,n}(\mu_{\tilde{x}_i}(x_i^*)).$$

По принципу обобщения рассчитывается нечеткое число, соответствующее значению четкой функции от нечетких аргументов. Различие между функциями от четких и нечетких аргументов иллюстрирует рис. 1.10.

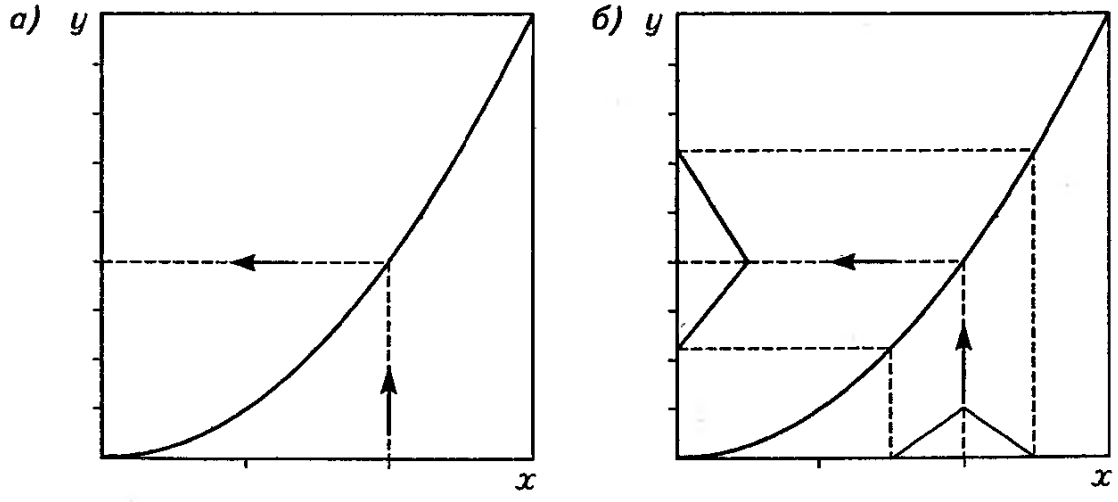


Рис. 1.10. Функции от четкого (*а*) и нечеткого (*б*) аргументов $y = f(x)$

Компьютерно-ориентированная реализация принципа нечеткого обобщения Заде осуществляется по следующему алгоритму.

Шаг 1. Зафиксировать значение $y = y^*$.

Шаг 2. Найти все n-ки $\{x^*_{1,j}, x^*_{2,j}, \ldots, x^*_{n,j}\}$, $j = \overline{1, P}$, удовлетворяющие условиям $y^* = f(x^*_{1,j}, x^*_{2,j}, \ldots, x^*_{n,j})$ и $x^*_{i,j} \in \mathrm{supp}(\tilde{x}_i)$, $i = \overline{1, n}$.

Шаг 3. Степень принадлежности элемента y^* нечеткому числу $\tilde{y}$ вычислить по формуле: $\mu_{\tilde{y}}(y^*) = \max\limits_{j=\overline{1,P}} \min\limits_{i=\overline{1,n}} (\mu_{\tilde{x}_i}(x^*_{i,j}))$.

Шаг 4. Проверить условие «Взяты все элементы y^*?». Если «Да», то перейти к шагу 5, иначе зафиксировать новое значение y^* и перейти к шагу 2.

Шаг 5. Конец.

В приведенном алгоритме предполагается, что нечеткие числа заданы на дискретных носителях. Обычно нечеткие аргументы $\tilde{x}_i, i = \overline{1, n}$ задаются кусочно-непрерывными функциями принадлежности

$$\tilde{x}_i = \int\limits_{x_i \in \mathbf{R}} \mu_{\tilde{x}_i}(x_i) / x_i .$$

Для вычисления значений функции $\tilde{y} = f(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$ нечеткие аргументы дискретизуют, т.е. представляют в виде

$$\tilde{x}_i = \left(\frac{\mu_{\tilde{x}_i}(x_{i,1})}{x_{i,1}}, \frac{\mu_{\tilde{x}_i}(x_{i,2})}{x_{i,2}}, \ldots, \frac{\mu_{\tilde{x}_i}(x_{i,k})}{x_{i,k}} \right), \; i = \overline{1, n}.$$

Количество дискрет k выбирают так, чтобы обеспечить требуемую точность вычислений. На выходе алгоритма получается нечеткое множество, также заданное на дискретном носителе. Результирующую кусочно-непрерывную функцию принадлежности нечеткого числа $\tilde{y}$ получают как верхнюю огибающую найденных точек $(y^*, \mu_{\tilde{y}}(y^*))$.

Пример 1.6. Нечеткие числа $\tilde{x}_1$ и $\tilde{x}_2$ заданы следующими трапециевидными функциями принадлежности:

$$\mu_{\tilde{x}_1}(x) = \begin{cases} 0, & \text{если } x < 1 \text{ или } x > 4 \\ x - 1, & \text{если } x \in [1, 2] \\ 1, & \text{если } x \in [2, 3] \\ 4 - x, & \text{если } x \in [3, 4] \end{cases} \quad \text{и} \quad \mu_{\tilde{x}_2}(x) = \begin{cases} 0, & \text{если } x < 2 \text{ или } x > 8 \\ x - 2, & \text{если } x \in [2, 3] \\ 1, & \text{если } x \in [3, 4] \\ 2 - 0{,}25x, & \text{если } x \in [4, 8] \end{cases}$$

Необходимо найти нечеткое число $\tilde{y} = \tilde{x}_1 \tilde{x}_2$ по принципу обобщения Заде.

Зададим нечеткие аргументы на четырех дискретах (точках). Будем использовать носитель $\{1, 2, 3, 4\}$ для $\tilde{x}_1$ и $\{2, 3, 4, 8\}$ – для $\tilde{x}_2$. Тогда нечеткие аргументы запишем так: $\tilde{x}_1 = (0/1, 1/2, 1/3, 0/4)$ и $\tilde{x}_2 = (0/2, 1/3, 1/4, 0/8)$. Процесс перемножения нечетких чисел $\tilde{x}_1$ и $\tilde{x}_2$ сведен в табл. 1.8. Каждый столбец таблицы соответствует одной итерации алгоритма нечеткого обобщения. Результирующее нечеткое множество представлено первой и последней строчками таблицы. В первой строке записаны элементы универсального множества, а в последней строке – степени их принадлежности к значению выражения $\tilde{x}_1 \tilde{x}_2$. В результате получаем $\tilde{y} = (0/2, 1/6, 1/12, 0/32)$.

Таблица 1.8

Умножение нечетких чисел (к примеру 1.6)

$y^* = x_1^* x_2^*$	2	3	4		6		8			9	12		16		24	32
x_1^*	1	1	1	2	2	3	1	2	4	3	3	4	2	4	3	4
x_2^*	2	3	4	2	3	2	8	4	2	3	4	3	8	4	8	8
$\mu_{\tilde{x}_1}(x_1^*)$	0	0	0	1	1	1	0	1	0	1	1	0	1	0	1	0
$\mu_{\tilde{x}_2}(x_2^*)$	0	1	1	0	1	0	0	1	0	1	1	1	0	1	0	0
$\min(\mu_{\tilde{x}_1}(x_1^*), \mu_{\tilde{x}_2}(x_2^*))$	0	0	1	0	1	0	0	1	0	1	1	0	0	0	0	0
$\mu_{\tilde{y}}(y^*)$	0	0	0		1		1			1	1		0		0	0

Предположим, что тип функции принадлежности результата $\tilde{y}$ будет таким же, как и аргументов $\tilde{x}_1$ и $\tilde{x}_2$, т.е. трапециевидным. Тогда функция принадлежности задается выражением:

$$\mu_{\tilde{y}}(y) = \begin{cases} 0, & \text{если } y < 2 \text{ или } y > 32 \\ (y-2)/3, & \text{если } y \in [1,2] \\ 1, & \text{если } y \in [6,9] \\ (32-y)/23, & \text{если } y \in [3,4] \end{cases}$$

На рис. 1.11*а* показан результат операции $\tilde{y} = \tilde{x}_1 \tilde{x}_2$ при задании нечетких множителей на четырех дискретах. Результирующая трапециевидная функция принадлежности показана сплошной жирной линией. Исследуем, как изменится результат нечеткого обобщения при увеличении числа дискрет, на которых задаются аргумеиты. Нечеткие числа $\tilde{y}$ при задании аргументов $\tilde{x}_1$ и $\tilde{x}_2$ на семи, десяти и шестнадцати дискретах показаны на рис. 1.11 *б*, *в*, *г*. Маркеры в виде точек соответствуют элементам нечеткого множества $\tilde{y}$, найденным по принципу обобщения. Сплошной жирной линией показана верхняя огибающая этих точек, т.е. аппроксимированная функция принадлежности нечеткого числа $\tilde{y}$. Функция принадлежности результата имеет форму криволинейной трапеции, немного выгнутой влево.

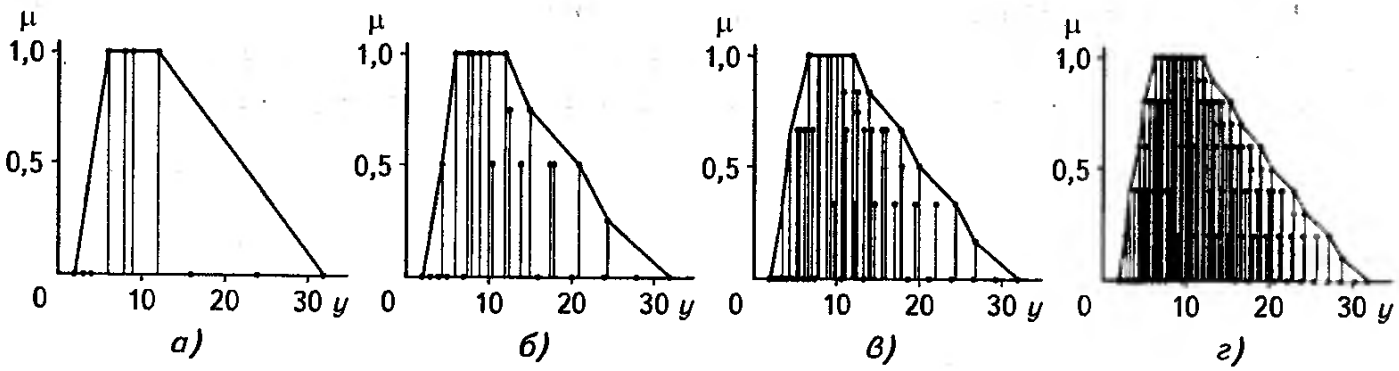


Рис. 1.11. Функция принадлежности результата (к примеру 1.6)

а – для четырех дискрет; *б* – для семи дискрет; *в* – для десяти дискрет; *г* – для шестнадцати дискрет

Применение принципа обобщения Заде сопряжено с двумя трудностями:

- большой объем вычислений – количество элементов результирующего нечеткого множества, которые необходимо обработать, равно $k_1 k_2 \ldots k_n$, где

k_i – мощность (количество элементов) носителя i-го нечеткого аргумента $(i = \overline{1, n})$;

- необходимо строить верхнюю огибающую элементов результирующего нечеткого множества.

Более практичным является уровневый принцип обобщения. В этом случае нечеткие числа задают множествами α-сечений:

$$\tilde{x} = \bigcup_{\alpha \in [0;1]} (\underline{x}_\alpha, \overline{x}_\alpha),$$

где $\underline{x}_\alpha$, $\overline{x}_\alpha$ – соответственно минимальное и максимальное значение на α-уровне.

Определение 23. *α-уровневый принцип обобщения.* Если $y = f(x_1, x_2, \ldots, x_n)$ – функция от n независимых аргументов x_i, $i = \overline{1, n}$, которые заданы нечеткими числами

$$\tilde{x}_i = \bigcup_{\alpha \in [0;1]} (\underline{x}_{i,\alpha}, \overline{x}_{i,\alpha}),$$

то значением функции $\tilde{y} = f(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$ называется нечеткое число

$$\tilde{y} = \bigcup_{\alpha \in [0;1]} (\underline{y}_\alpha, \overline{y}_\alpha),$$

где:

$$\underline{y}_\alpha = \inf_{x_{i,\alpha} \in [\underline{x}_{i,\alpha}, \overline{x}_{i,\alpha}],\ i = \overline{1,n}} (f(x_{1,\alpha}, x_{2,\alpha}, \ldots, x_{n,\alpha})), \tag{1.4}$$

$$\overline{y}_\alpha = \sup_{x_{i,\alpha} \in [\underline{x}_{i,\alpha}, \overline{x}_{i,\alpha}],\ i = \overline{1,n}} (f(x_{1,\alpha}, x_{2,\alpha}, \ldots, x_{n,\alpha})). \tag{1.5}$$

Применение α-уровневого принципа обобщения сводится к решению для каждого α-уровня следующей задачи оптимизации: найти максимальное и минимальное значения функции $y = f(x_1, x_2, \ldots, x_n)$ при условии, что аргументы могут принимать значения из соответствующих α-сечений. Количество α-уровней выбирают так, чтобы обеспечить необходимую точность вычислений.

Пример 1.7. Решить задачу из примера 1.6, применяя α-уровневый принцип обобщения.

Будем использовать два α-уровня: 0 и 1. Тогда нечеткие аргументы задаются так:

$\tilde{x}_1 = (1; 4)_0 \cup (2; 3)_1$ и $\tilde{x}_2 = (2; 8)_0 \cup (3; 4)_1$.

По α-уровневому принципу обобщения получаем:

$\tilde{y} = (2; 32)_0 \cup (6; 12)_1$.

На рис. 1.12*а* изображен результат умножения $\tilde{y} = \tilde{x}_1 \tilde{x}_2$: α-сечения показаны тонкими горизонтальными линиями; кусочно-линейная аппроксимация функции принадлежности нечеткого числа $\tilde{y}$ показана жирной линней. Исследуем, как изменится результат нечеткого обобщения при увеличении количества α-уровней. Нечеткие числа $\tilde{y}$ при задании аргументов $\tilde{x}_1$ и $\tilde{x}_2$ на трех, пяти и восьми α-уровнях показаны на рис. 1.12 *б, в, г*. Сравнивая рис. 1.11 и 1.12, видим, что результаты обобщения по определениям 22 и 23 близки. В предельном случае, при бесконечно большом количестве дискрет, результаты обобщения по этим принципам будут эквивалентными.

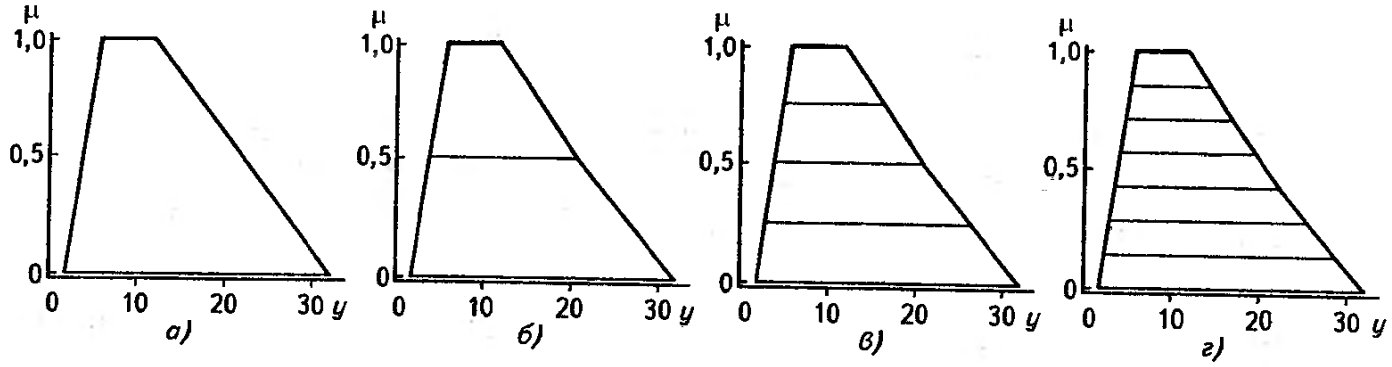


Рис. 1.12. Функция принадлежности результата (к примеру 1.7)

а – для двух α-уровней; *б* – для трех α-уровней; *в* – для пяти α-уровней; *г* – для восьми α-уровней

С помощью α-уровневого принципа обобщения можно легко получить правила выполнения арифметических операций над нечеткими числами. Эти правила для положительных нечетких чисел приведены в табл. 1.9. В общем виде правила выполнения арифметических операций можно записать следующим образом. Обозначим арифметическую операцию символом «*». Тогда для каждого α-уровня результат операции $\tilde{y} = \tilde{x}_1 * \tilde{x}_2$ рассчитывается по формулам:

Таблица 1.9

Правила выполнения арифметических операций для положительных нечетких чисел (для каждого α-уровня)

Арифметическая операция	$\underline{y}$	$\overline{y}$
$\tilde{y} = \tilde{x}_1 + \tilde{x}_2$	$\underline{x}_1 + \underline{x}_2$	$\overline{x}_1 + \overline{x}_2$
$\tilde{y} = \tilde{x}_1 - \tilde{x}_2$	$\underline{x}_1 - \overline{x}_2$	$\overline{x}_1 - \underline{x}_2$
$\tilde{y} = \tilde{x}_1 \tilde{x}_2$	$\underline{x}_1 \underline{x}_2$	$\overline{x}_1 \overline{x}_2$
$\tilde{y} = \tilde{x}_1 / \tilde{x}_2$	$\underline{x}_1 / \overline{x}_2$	$\overline{x}_1 / \underline{x}_2$

$$\underline{y} = \min(\underline{x}_1 * \underline{x}_2,\ \overline{x}_1 * \overline{x}_2,\ \underline{x}_1 * \overline{x}_2,\ \overline{x}_1 * \underline{x}_2),\quad \overline{y} = \max(\underline{x}_1 * \underline{x}_2,\ \overline{x}_1 * \overline{x}_2,\ \underline{x}_1 * \overline{x}_2,\ \overline{x}_1 * \underline{x}_2).$$

Обратим внимание, что операция деления может быть выполнена, только когда делитель является либо положительным, либо отрицательным нечетким числом. В противном случае происходит деление на ноль.

Рассмотренные принципы обобщения требуют, чтобы аргументы функции $y = f(x_1, x_2, \ldots, x_n)$ были независимыми. В случае взаимодействующих аргументов необходимо ввести соответствующие ограничения в задачи оптимизации (1.4) и (1.5).

Определение 24. *α-уровневый принцип обобщения при взаимодействующих аргументах.* Пусть $y = f(x_1, x_2, \ldots, x_n)$ – функция от n аргументов, которые связаны соотношениями $g(x_1, x_2, \ldots, x_n) = 0$ и $h(x_1, x_2, \ldots, x_n) > 0$. Значением функции $\tilde{y} = f(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$ называется нечеткое число $\tilde{y} = \bigcup\limits_{\alpha \in [0,1]} (\underline{y}_\alpha, \overline{y}_\alpha)$, такое, что:

$$\underline{y}_\alpha = \inf_{\substack{x_{i,\alpha} \in [\underline{x}_{i,\alpha}, \overline{x}_{i,\alpha}],\ i=\overline{1,n} \\ g(x_{1,\alpha}, x_{2,\alpha}, \ldots, x_{n,\alpha}) = 0, \\ h(x_{1,\alpha}, x_{2,\alpha}, \ldots, x_{n,\alpha}) > 0}} (f(x_{1,\alpha}, x_{2,\alpha}, \ldots, x_{n,\alpha})),$$

$$\overline{y}_\alpha = \sup_{\substack{x_{i,\alpha} \in [\underline{x}_{i,\alpha}, \overline{x}_{i,\alpha}],\ i=\overline{1,n} \\ g(x_{1,\alpha}, x_{2,\alpha}, \ldots, x_{n,\alpha}) = 0, \\ h(x_{1,\alpha}, x_{2,\alpha}, \ldots, x_{n,\alpha}) > 0}} (f(x_{1,\alpha}, x_{2,\alpha}, \ldots, x_{n,\alpha})).$$

1.4. НЕЧЕТКИЕ ОТНОШЕНИЯ

В разделе дается определение нечеткого отношения, рассматриваются свойства нечетких отношений и правила выполнения операций над ними.

Определение 25. *Нечетким отношением* $\tilde{R}$ на множествах $X_1, X_2, \ldots, X_n$ называется нечеткое подмножество декартова произведения $X_1 X_2 \ldots X_n$. Степень принадлежности $\mu_{\tilde{R}}(x_1, x_2, \ldots, x_n)$ показывает уровень выполнения отношения $\tilde{R}$ между элементами $(x_1, x_2, \ldots, x_n)$, $x_i \in X_i$, $i = \overline{1, n}$.

В дальнейшем будем рассматривать только бинарные нечеткие отношения, которые задаются на декартовом произведении двух множеств. Обозначим эти множества через X и Y. Тогда задание бинарного нечеткого отношения $\tilde{R}$ на $X \times Y$ состоит в указании всех троек $(x, y, \mu_{\tilde{R}}(x, y))$, где $x \in X$, $y \in Y$, $(x, y) \in X \times Y$, $\mu_{\tilde{R}}(x, y) \in [0, 1]$.

Определение 26. Носителем нечеткого отношения $\tilde{R}$ на множествах X и Y называется подмножество декартова произведения $X \times Y$ вида:

$$\mathrm{supp}\tilde{R} = \{(x, y) : (x, y) \in X \times Y,\ \mu_{\tilde{R}}(x, y) > 0\}.$$

Носитель нечеткого отношения можно рассматривать как обычное отношение, связывающее все пары $(x, y) \in X \times Y$, для которых степень выполнения нечеткого отношения $\tilde{R}$ не равна нулю. Более полезным является использование α-сечений нечеткого отношения, определения которых аналогичны определениям множеств α-уровня (см. определение 11).

Определение 27. *α-сечением нечеткого отношения* $\tilde{R}$ на $X \times Y$ называется обычное отношение, связывающее все пары $(x, y) \in X \times Y$, для которых степень выполнения нечеткого отношения $\tilde{R}$ не меньше α:

$$R_\alpha = \{(x, y) : (x, y) \in X \times Y,\ \mu_{\tilde{R}}(x, y) \geq \alpha\}.$$

Пример 1.8. Задать нечеткое отношение $x \approx y$ («x приблизительно равно y»).

Пусть $x \in \{0, 1, 2, 3\}$ и $y \in \{0, 1, 2, 3\}$. Тогда нечеткое отношение можно задать матрицей вида:

$$\tilde{R} = \begin{array}{c} \begin{matrix} 0 & 1 & 2 & 3 \leftarrow y \end{matrix} \\ \begin{bmatrix} 1 & 0{,}5 & 0{,}2 & 0{,}1 \\ 0{,}5 & 1 & 0{,}6 & 0{,}3 \\ 0{,}2 & 0{,}6 & 1 & 0{,}8 \\ 0{,}1 & 0{,}3 & 0{,}8 & 1 \end{bmatrix} \end{array} \begin{matrix} x \downarrow \\ 0 \\ 1 \\ 2 \\ 3 \end{matrix}$$

При непрерывных множествах $X = [0, 3]$ и $Y = [0, 3]$ нечеткое отношение можно задать следующей функцией принадлежности:

$$\mu_{\tilde{R}}(x, y) = e^{-0{,}2(x-y)^2}.$$

Нечеткие отношения $x \approx y$ на дискретных и непрерывных носителях изображены на рис. 1.13.

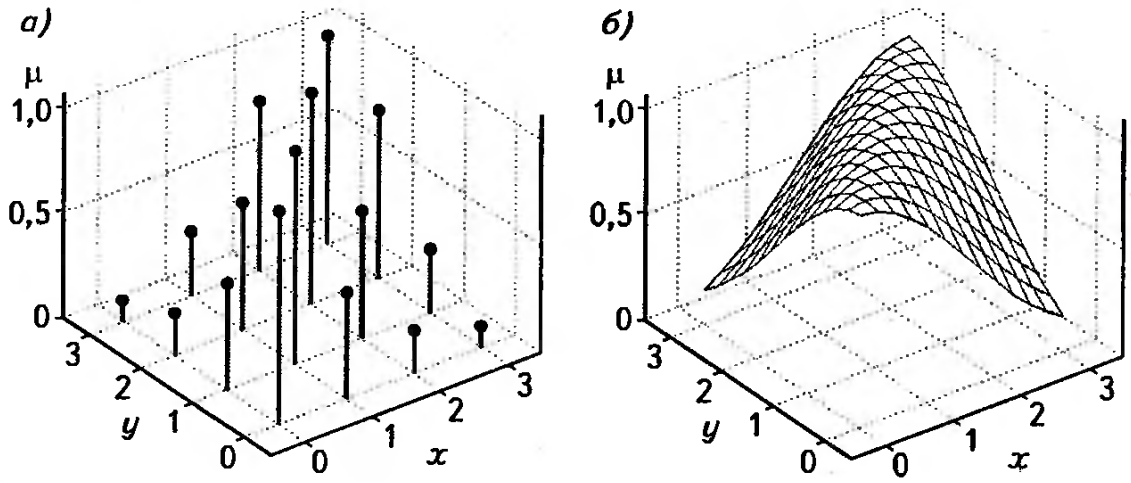


Рис. 1.13. Нечеткое отношение «*x* приблизительно равно *y*»

а – отношение на дискретных множествах; *б* – отношение на непрерывных множествах

Пример 1.9. Задать нечеткое отношение $x << y$ («*x* намного меньше, чем *y*»).
Пусть $x, y \in \{0, 1, 2, 3\}$. Тогда нечеткое отношение можно задать такой матрицей:

$$\widetilde{R} = \begin{matrix} 0 & 1 & 2 & 3 \leftarrow y \\ \end{matrix}$$

$$\begin{matrix} x\downarrow \\ \end{matrix}$$

$$\begin{bmatrix} 0 & 0{,}2 & 0{,}6 & 1 \\ 0 & 0 & 0{,}2 & 0{,}6 \\ 0 & 0 & 0 & 0{,}2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix}$$

Для непрерывных множеств $X = [0, 3]$ и $Y = [0, 3]$ это нечеткое отношение можно определить следующей функцией принадлежности:

$$\mu_{\widetilde{R}}(x, y) = \begin{cases} 0, & \text{если } x \geq y \\ \dfrac{1}{1+5/(x-y)^4}, & \text{если } x < y. \end{cases}$$

Нечеткие отношения $x << y$ на дискретных и непрерывных множествах изображены на рис. 1.14.

Нечеткие отношения позволяют не только зафиксировать сам факт отношения, но и указывать степень его выполнения, что важно для многих практических задач.

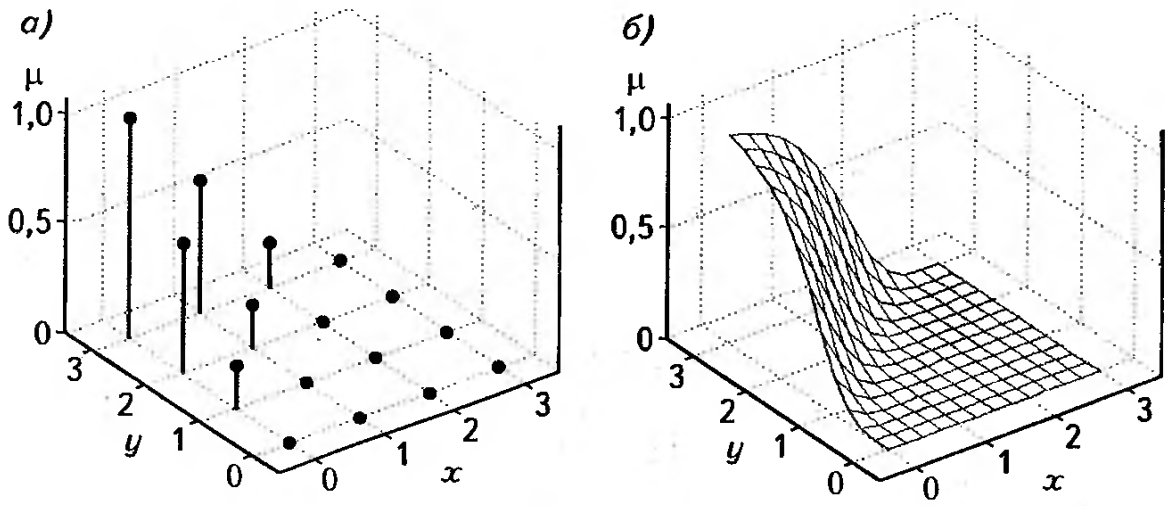


Рис. 1.14. Нечеткое отношение «*x* намного меньше, чем *y*»

а – отношение на дискретных множествах; *б* – отношение на непрерывных множествах

Пример 1.10. Задать отношение «схожий менталитет» для следующих национальностей: украинцы (У), чехи (Ч), австрийцы (А), немцы (Н).

Использование обычного (не нечеткого) отношения позволяет выделить только одну пару наций со схожими менталитетами – немцев и австрийцев. Такое отношение не учитывает, что немецкий менталитет ближе к чешскому, чем к украинскому. Нечеткое отношение позволяет легко представить эти знания. Один из вариантов такого отношения задан следующей матрицей:

$$\widetilde{R} = \begin{array}{c} \begin{matrix} \text{У} & \text{Ч} & \text{А} & \text{Н} \end{matrix} \\ \begin{bmatrix} 1 & 0{,}5 & 0{,}2 & 0{,}1 \\ 0{,}5 & 1 & 0{,}6 & 0{,}3 \\ 0{,}2 & 0{,}6 & 1 & 0{,}8 \\ 0{,}1 & 0{,}3 & 0{,}8 & 1 \end{bmatrix} \end{array} \begin{matrix} \text{У} \\ \text{Ч} \\ \text{А} \\ \text{Н} \end{matrix}$$

Нечеткое отношение $\widetilde{R}$ на дискретном носителе $X \times Y$ можно интерпретировать как нечеткий граф. Вершинами нечеткого графа являются элементы множеств X и Y. Степень принадлежности $\mu_{\widetilde{R}}(x, y)$ показывает, насколько сильно вершины $x \in X$ и $y \in Y$ связаны между собой.

Определение 28. Нечеткое отношение $\widetilde{R}$ на $X \times X$ называется *рефлексивным*, если для любого $x \in X$ выполняется равенство $\mu_{\widetilde{R}}(x, x) = 1$. В случае конечного множества X все элементы главной диагонали матрицы рефлексивного отношения $\widetilde{R}$ равны 1. Примером рефлексивного нечеткого отношения может быть отношение «приблизительно равны».

Определение 29. Нечеткое отношение $\widetilde{R}$ на $X \times X$ называется *антирефлексивным*, если для любого $x \in X$ выполняется равенство $\mu_{\widetilde{R}}(x, x) = 0$. В случае конечного множества X все элементы главной диагонали матрицы $\widetilde{R}$ равны 0. Примером антирефлексивного нечеткого отношения может быть отношение «значительно больше».

Определение 30. Нечеткое отношение $\widetilde{R}$ на $X \times Y$ называется *симметричным*, если для любой пары $(x, y) \in X \times Y$ выполняется равенство $\mu_{\widetilde{R}}(x, y) = \mu_{\widetilde{R}^{-1}}(y, x)$. Матрица симметричного нечеткого отношения является симметричной.

Определение 31. Нечеткое отношение $\widetilde{R}$ на $X \times Y$ называется *асимметричным*, если выражение $\mu_{\widetilde{R}}(x, y) > 0 \Rightarrow \mu_{\widetilde{R}}(y, x) = 0$ справедливо для любой пары $(x, y) \in X \times Y$. Примером асимметричного нечеткого отношения может служить отношение «намного больше».

Определение 32. Нечеткие отношения $\widetilde{R}$ и $\widetilde{R}^{-1}$ на $X \times Y$ называются *обратными*, если для любой пары $(x, y) \in X \times Y$ выполняется равенство $\mu_{\widetilde{R}}(x, y) = \mu_{\widetilde{R}^{-1}}(y, x)$. Примером обратных нечетких отношений может служить пара «намного больше» – «намного меньше».

Операции над нечеткими отношениями, также как и нечеткие теоретико-множественные операции, могут выполняться различными способами. Ниже приводятся определения операций над нечеткими отношениями с использованием треугольных норм (см. подраздел 1.2.3).

Определение 33. *Пересечением нечетких отношений* $\widetilde{A}$ и $\widetilde{B}$, заданных на ...нечеткое отношение $\widetilde{C} = \widetilde{A} \cap \widetilde{B}$ с функцией принадлежности:

$$\mu_{\widetilde{C}}(x, y) = \mu_{\widetilde{A}}(x, y) \wedge \mu_{\widetilde{B}}(x, y), \quad (x, y) \in X \times Y,$$

где $\wedge$ – t-норма.

Определение 34. *Объединением нечетких отношений* $\widetilde{A}$ и $\widetilde{B}$, заданных на $X \times Y$, называется нечеткое отношение $\widetilde{D} = \widetilde{A} \cup \widetilde{B}$ с функцией принадлежности:

$$\mu_{\widetilde{D}}(x, y) = \mu_{\widetilde{A}}(x, y) \vee \mu_{\widetilde{B}}(x, y), \quad (x, y) \in X \times Y,$$

где $\vee$ – s-норма.

Пересечение и объединение нечетких отношений $x \approx y$ и $x << y$ показаны на рис. 1.15. В качестве треугольных норм использовались операции минимума и максимума.

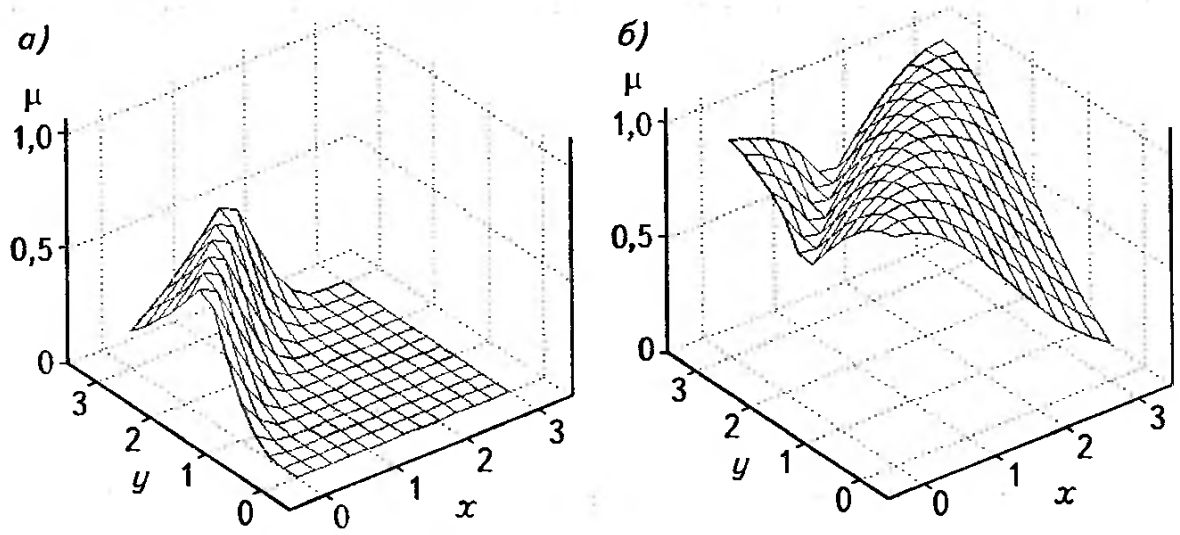


Рис. 1.15. Операции над нечеткими отношениями (из примеров 1.8 и 1.9)

а – пересечение нечетких отношений; *б* – объединение нечетких отношений

Определение 35. *Дополнением нечеткого отношения* $\widetilde{R}$, заданного на $X \times Y$, называется нечеткое отношение $\widetilde{R}'$ с функцией принадлежности:

$$\mu_{\widetilde{R}'}(x, y) = 1 - \mu_{\widetilde{R}}(x, y), \quad (x, y) \in X \times Y,$$

Определение 36. *Максминной композицией* (произведением) нечетких отношений $\widetilde{A}$ и $\widetilde{B}$, заданных на $X \times Z$ и $Z \times Y$, называется нечеткое отношение $\widetilde{G} = \widetilde{A} \circ \widetilde{B}$ на $X \times Y$ с функцией принадлежности

$$\mu_{\widetilde{G}}(x, y) = \sup_{z \in Z} \min(\mu_{\widetilde{A}}(x, z), \ \mu_{\widetilde{B}}(z, y)),$$
$$(x, y) \in X \times Y, \quad (x, z) \in X \times Z, \quad (z, y) \in Z \times Y.$$

В случае конечных множеств X, Y, Z матрица нечеткого отношения $\widetilde{G} = \widetilde{A} \circ \widetilde{B}$ рассчитывается как максминное произведение матриц $\widetilde{A}$ и $\widetilde{B}$. Эта операция выполняется как обычное произведение матриц, в котором поэлементное умножение заменено операцией минимума, а суммирование – операцией максимума. Аналогично определяются операции минимаксной и максимультипликативной композиции. Композиция играет ключевую роль в нечетком логическом выводе.

Пример 1.11. Заданы нечеткие отношения

$$\tilde{A}=\begin{bmatrix}0{,}1 & 0{,}2\\ 0{,}8 & 1\end{bmatrix} \text{ и } \tilde{B}=\begin{bmatrix}0{,}6 & 0{,}4\\ 0{,}5 & 0{,}3\end{bmatrix}.$$

Тогда максминной ($\tilde{G}_1$), минимаксиой ($\tilde{G}_2$) и максимультипликативной ($\tilde{G}_3$) композициям этих нечетких отношений соответствуют такие матрицы:

$$\tilde{G}_1=\begin{bmatrix}0{,}2 & 0{,}2\\ 0{,}6 & 0{,}4\end{bmatrix};\ \tilde{G}_2=\begin{bmatrix}0{,}5 & 0{,}3\\ 0{,}8 & 0{,}8\end{bmatrix};\ \tilde{G}_3=\begin{bmatrix}0{,}1 & 0{,}06\\ 0{,}5 & 0{,}32\end{bmatrix}.$$

Определение 37. Нечеткое отношение $\tilde{R}$ на $X\times Y$ называется *транзитивным*, если $\tilde{R}\circ\tilde{R}\subseteq\tilde{R}$. Следовательно, $\mu_{\tilde{R}}(x,y)\geq\mu_{\tilde{R}\circ\tilde{R}}(x,y)$ для любой пары $(x,y)\in X\times Y$.

Определение 38. *Транзитивным замыканием* $\tilde{R}_{\text{т}}$ нечеткого отношения $\tilde{R}$ называется следующее отношение:

$$\tilde{R}_{\text{т}}=\tilde{R}\cup\tilde{R}^2\cup\tilde{R}^3\cup\ldots\cup\tilde{R}^n\cup\ldots,$$

где $\tilde{R}^n=\underbrace{\tilde{R}\circ\tilde{R}\circ\ldots\circ\tilde{R}}_{n\text{ раз}}$.

1.5. НЕЧЕТКАЯ ЛОГИКА

В разделе рассматривается нечеткая логика – обобщение традиционной аристотелевой логики на случай, когда истинность рассматривается как лингвистическая переменная, принимающая значения «очень истинно», «более-менее истинно», «не очень ложно» и т.п. Эти лингвистические значения представляются нечеткими множествами.

1.5.1. ЛИНГВИСТИЧЕСКИЕ ПЕРЕМЕННЫЕ

Напомним, что лингвистической называется переменная, принимающая значения из множества слов или словосочетаний некоторого естественного языка. Понятие лингвистической переменной играет важную роль в нечетком логическом выводе и в принятии решений на основе приближенных рассуждений. Формально лингвистическая переменная описывается следующей пятеркой:

$$\langle x,\ T,\ U,\ G,\ M\rangle,$$

где x – имя переменной; T – терм-множество, каждый элемент которого задается нечетким множеством 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 из G.

Пример 1.12. Рассмотрим лингвистическую переменную с именем x = «температура в комнате». Тогда оставшуюся четверку $\langle T,\ U,\ G,\ M\rangle$ можно определить так:

- универсальное множество: $U = [12, 35]$;
- терм-множество: T = {«холодно», «комфортно», «жарко»} с такими функциями принадлежностями:

$$\mu_{\text{«холодно»}}(u) = \frac{1}{1+\left|\frac{u-12}{6}\right|^{12}};\quad \mu_{\text{«комфортно»}}(u) = \frac{1}{1+\left|\frac{u-20}{3}\right|^{8}};\quad \mu_{\text{«жарко»}}(u) = \frac{1}{1+\left|\frac{u-33}{8}\right|^{12}};\quad u \in U;$$

- синтаксические правила G, порождающие новые термы с использованием квантификаторов «не», «очень» и «более-менее»;
- семантические правила M, заданные в табл. 1.10.

Графики функций принадлежности термов «холодно», «не очень холодно», «комфортно», «более-менее комфортно», «жарко» и «очень жарко» лингвистической переменной «температура в комнате» показаны на рис. 1.16.

Таблица 1.10

Правила модификации функций принадлежности (к примеру 1.12)

Квантификатор	Функция принадлежности
Не t	$1-\mu_t(u)$
Очень t	$(\mu_t(u))^2$
Более-менее t	$\sqrt{\mu_t(u)}$

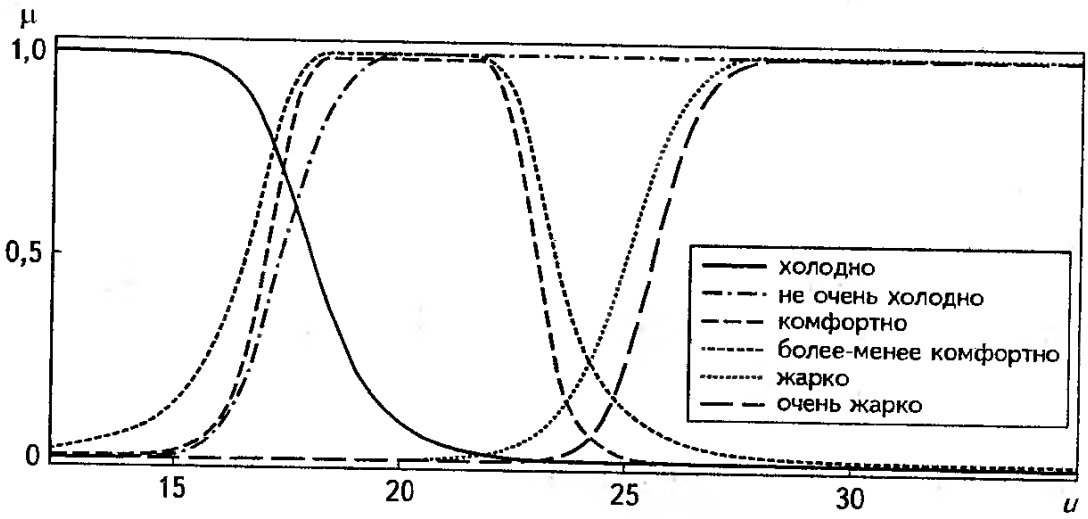


Рис. 1.16. Лингвистическая переменная «температура в комиате» (из примера 1.12)

1.5.2. НЕЧЕТКАЯ ИСТИННОСТЬ

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 лингвистической переменной «истинность». Обычная, четкая истинность может быть представлена нечеткими множествами–синглтонами. В этом случае четким понятиям «истинно» и «ложно» будут соответствовать такие функции принадлежности:

$$\mu_{\text{«истинно»}}(u) = \begin{cases} 0, & u \neq 1 \\ 1, & u = 1 \end{cases} \quad \text{и} \quad \mu_{\text{«ложно»}}(u) = \begin{cases} 0, & u \neq 0 \\ 1, & u = 0; \end{cases} \quad u \in [0, 1].$$

Для нечеткой истинности Заде предложил следующие функции принадлежности термов «истинно» и «ложно»:

$$\mu_{\text{«истинно»}}(u)=\begin{cases}0, & 0\le u\le a\\ 2\left(\dfrac{u-a}{1-a}\right)^2, & a<u\le\dfrac{a+1}{2}\\ 1-2\left(\dfrac{u-1}{1-a}\right)^2, & \dfrac{a+1}{2}<u\le 1\end{cases};\quad \mu_{\text{«ложно»}}(u)=\mu_{\text{«истинно»}}(1-u);\ u\in[0,\,1],$$

где $a\in[0,\,1]$ – параметр, задающий носители нечетких множеств «истинно» и «ложно». Для нечеткого множества «истинно» носителем будет интервал $(a,\,1]$, а для нечеткого множества «ложно» – $[0,\,a)$. Графики функций принадлежности нечетких термов «истинно» и «ложно» при $a=0{,}4$ изображены на рис. 1.17. Они являются зеркальными отображениями.

Для нечетких значений «истинно» и «ложно» Балдвин (Baldwin) предложил такие функции принадлежности:

$$\mu_{\text{«истинно»}}(u)=u;\ \mu_{\text{«ложно»}}(u)=1-u;\ u\in[0,\,1].$$

Квантификаторы «более-менее» и «очень» применяют к нечеткими значениям «истинно» и «ложно», получая термы «очень ложно», «более-менее ложно», «более-менее истинно», «очень истинно», «очень, очень истинно», «очень, очень ложно» и т.п. Функции принадлежности новых термов рассчитывают с использованием операций концентрации и растяжения нечетких множеств, что соответствует возведению функции принадлежности в степень 2 и 1/2 соответственно:

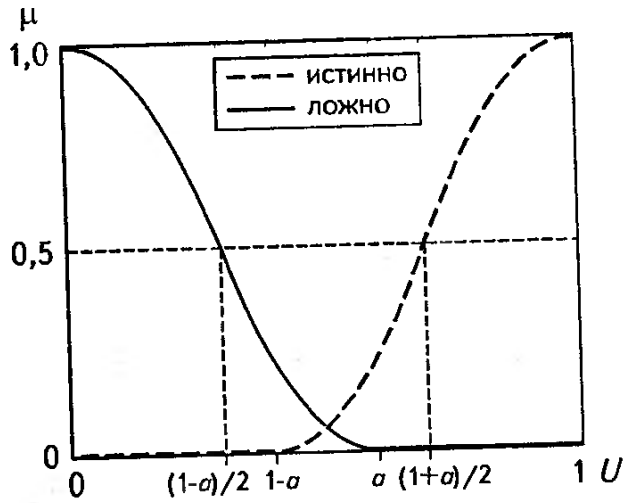


Рис. 1.17. Лингвистическая переменная «истинность» по Заде

$$\mu_{\text{«очень ложно»}}(u)=(\ldots\mu_{\text{«ложно»}}(u))^2;$$

$$\mu_{\text{«очень, очень ложно»}}(u)=(\ldots\mu_{\text{«очень ложно»}}(u))^2;$$

$$\mu_{\text{«более-менее ложно»}}(u)=(\ldots\mu_{\text{«ложно»}}(u))^{1/2};$$

$$\mu_{\text{«более-менее истинно»}}(u)=(\ldots\mu_{\text{«истинно»}}(u))^{1/2};$$

$$\mu_{\text{«очень истинно»}}(u)=(\ldots\mu_{\text{«истинно»}}(u))^2;$$

$$\mu_{\text{«очень, очень истинно»}}(u)=(\ldots\mu_{\text{«очень истинно»}}(u))^2.$$

Графики этих функций принадлежности показаны на рис. 1.18.

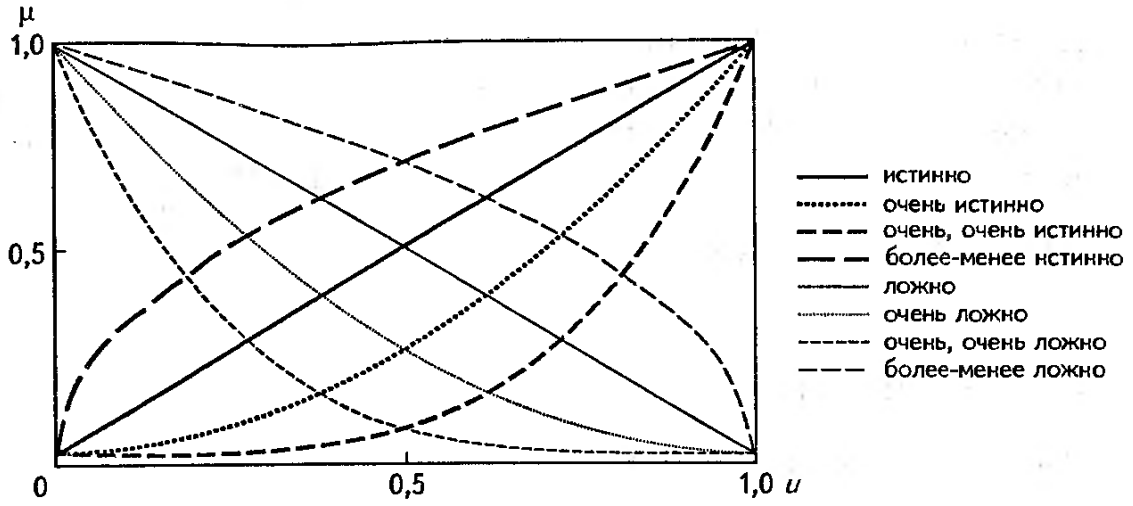


Рис. 1.18. Лингвистическая переменная «истинность» по Балдвину

1.5.3. НЕЧЕТКИЕ ЛОГИЧЕСКИЕ ОПЕРАЦИИ

Вначале кратко напомним основные положения обычной (булевой) логики. Рассмотрим два утверждения A и B, каждое из которых может быть истинным или ложным, т.е. принимать значения «1» или «0». Для двух утверждений A и B существует $(2^2)^2 = 16$ различных логических операций, из которых содержательно интерпретируются следующие пять: И ($\wedge$), ИЛИ ($\vee$), исключающее ИЛИ ($\oplus$), импликация ($\Rightarrow$) и эквивалентность ($\Leftrightarrow$). Таблицы истинности для этих операций сведены в табл. 1.11.

Таблица 1.11

Таблицы истинности булевой логики

A	B	$A \wedge B$	$A \vee B$	$A \oplus B$	$A \Rightarrow B$	$A \Leftrightarrow B$
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	0
1	1	1	1	0	1	1

Предположим, что логическое утверждение может принимать не два значения истинности, а три, например: «истинно», «ложно» и «неопределенно». В этом случае логика будет не двухзначной, а трехзначной. Общее количество бинарных операций, а, следовательно, и таблиц истинности, в трехзначной логике равно $(3^2)^3 = 729$.

Нечеткая логика является разновидностью многозначной логики, в которой значения истинности задаются лингвистическими переменными или термами лингвистической переменной «истинность». Правила выполнения нечетких логических операций получают из булевых логических операций с помощью принципа обобщения.

Определение 39. Обозначим нечеткие логические переменные через $\tilde{A}$ и $\tilde{B}$, а функции принадлежности, задающие истинностные значения этих переменных через $\mu_{\tilde{A}}(u)$ и $\mu_{\tilde{B}}(u)$, $u \in [0, 1]$. *Нечеткие логические операции* И ($\wedge$), ИЛИ ($\vee$), НЕ ($^-$) и импликация ($\Rightarrow$) определяются следующим образом [46]:

$$\mu_{\tilde{A}\wedge\tilde{B}}(u) = \min(\mu_{\tilde{A}}(u),\ \mu_{\tilde{B}}(u));$$

$$\mu_{\tilde{A}\vee\tilde{B}}(u) = \max(\mu_{\tilde{A}}(u),\ \mu_{\tilde{B}}(u));$$

$$\mu_{\bar{\tilde{A}}}(u) = 1 - \mu_{\tilde{A}}(u);$$

$$\mu_{\tilde{A}\Rightarrow\tilde{B}}(u) = \max(1 - \mu_{\tilde{A}}(u),\ \mu_{\tilde{B}}(u)).$$

В многозначной логике логические операции могут быть заданы таблицами истинности. В нечеткой логике количество возможных значений истинности может быть бесконечным, поэтому в общем виде табличное представление логических операций невозможно. Однако таблицей можно задать нечеткую логическую операцию при ограниченном числе истинностных значений, например, для терм-множества {«истинно», «очень истинно», «не истинно», «более-менее ложно», «ложно»}. Для логики с тремя нечеткими значениями истинности {T – «истинно», F – «ложно», $T+F$ – «неизвестно»} Заде предложил следующие лингвистические таблицы истинности.

$\tilde{A}$	$\tilde{B}$	$\bar{\tilde{A}}$	$\tilde{A}\wedge\tilde{B}$	$\tilde{A}\vee\tilde{B}$
T	T	F	T	T
T	F	F	F	T
T	$T+F$	F	$T+F$	T
F	T	T	F	T
F	F	T	F	F
F	$T+F$	T	F	$T+F$
$T+F$	T	$T+F$	$T+F$	T
$T+F$	F	$T+F$	F	$T+F$
$T+F$	$T+F$	$T+F$	$T+F$	$T+F$

Применяя правила из определения 39, можно расширить таблицы нечеткой истинности для большего количества термов. Как это сделать, рассмотрим на следующем примере.

Пример 1.13. Заданы нечеткие истинностные значения:

«истинно» = (0/0; 0/0,2; 0,25/0,4; 0,5/0,6; 0,9/0,8; 1/1);

«более-менее истинно» = (0/0; 0/0,2; 0,5/0,4; 0,7/0,6; 0,95/0,8; 1/1);

«почти истинно» = (0/0; 0,05/0,2; 0,4/0,4; 0,7/0,6; 1/0,8; 0,8/1).

По определению 39 находим нечеткую истинность выражения «почти истинно ИЛИ истинно»:

«почти истинно» $\vee$ «истинно» = (0/0; 0,05/0,2; 0,4/0,4; 0,7/0,6; 1/0,8; 1/1).

Сравнивая полученное нечеткое множество с нечетким множеством «более-менее истинно», видим, что они почти равны, значит:

«почти истинно» ∨ «истинно» = «более-менее истиино».

Результатом нечетких логических операций часто бывает нечеткое множество, неэквивалентное ни одному из ранее введенных значений истинности. В таком случае среди них надо найти наиболее похожее на результат выполнения нечеткой логической операции. Другими словами, необходимо провести *лингвистическую аппроксимацию*, которая может рассматриваться как аналог аппроксимации эмпирического статистического распределения данных стандартными функциями распределения случайных величин.

$\tilde{A}$	$\tilde{B}$	$\tilde{A} \wedge \tilde{B}$	$\tilde{A} \vee \tilde{B}$
Ложно	Ложно	Ложно	Ложно
Истинно	Ложно	Ложно	Истинно
Истинно	Истинно	Истинно	Истинно
Неопределенно	Ложно	Ложно	Неопределенно
Неопределенно	Истинно	Неопределенно	Истинно
Неопределенно	Неопределенно	Неопределенно	Неопределенно
Истинно	Очень истинно	Истинно	Очень истинно
Истинно	Более-менее истинно	Более-менее истинно	Истинно

В качестве примера выше приведены предложенные Балдвином лингвистические таблицы истинности для нечетких значений из рис. 1.18.

1.6. НЕЧЕТКИЙ ЛОГИЧЕСКИЙ ВЫВОД

В разделе рассматриваются основные алгоритмы нечеткого логического вывода: композиционное правило вывода Заде, алгоритм Мамдани, алгоритм Такаги-Сугено, вывод по синглтонной базе знаний и нечеткий логический вывод для задач классификации. Показывается, как использовать нечеткий логический вывод для моделирования объектов с непрерывным и дискретным выходом. Рассматриваются иерархические нечеткие системы и нейро-нечеткие сети.

1.6.1. ЛОГИЧЕСКИЙ ВЫВОД

Обычный булевый логический вывод базируется на следующих тавтологиях:

- модус поненс: $[A \wedge (A \Rightarrow B)] \Rightarrow B$;
- модус толленс: $[(A \Rightarrow B) \wedge B] \Rightarrow A$;
- силлогизм: $[(A \Rightarrow B) \wedge (B \Rightarrow C)] \Rightarrow (A \Rightarrow C)$;
- контрапозиция: $(A \Rightarrow B) \Rightarrow (\overline{B} \Rightarrow \overline{A})$.

В четком логическом выводе наиболее часто применяется правило модус поненс, которое для логических утверждений A и B можно записать так:

Посылка	A есть истинно
Импликация	Если A, то B
Логический вывод	B есть истинно

Модус поненс выводит заключение «B есть истинно», если известно, что «A есть истинно» и существует правило «Если A, то B». Если прецедент отсутствует, то модус поненс не сможет вывести никакого, даже приближенного заключения. Когда известно, что близкое к A утверждение A' является истинным, модус поненс также не может быть применен. Одним из возможных способов принятия решений при неопределенной информации является нечеткий логический вывод.

1.6.2. ОСНОВЫ НЕЧЕТКОГО ЛОГИЧЕСКОГО ВЫВОДА

Определение 40. *Нечеткий логический вывод* – это аппроксимация зависимости «входы – выход» на основе лингвистических высказываний <Если – то> и логических операций над нечеткими множествами.

Нечеткий логический вывод применяется при моделировании объектов с непрерывным и с дискретным выходом. Объекты с непрерывным выходом (рис. 1.19*а*) соответствуют задачам аппроксимации гладких функций, возникающим в прогнозировании, многокритериальном анализе, управлении техническими объектами и т.п. Объекты с дискретным выходом (рис. 1.19*б*) соответствуют задачам классификации в медицинской и технической диагностике, в распознавании образов, в ситуационном управлении и при принятии решений в других областях.

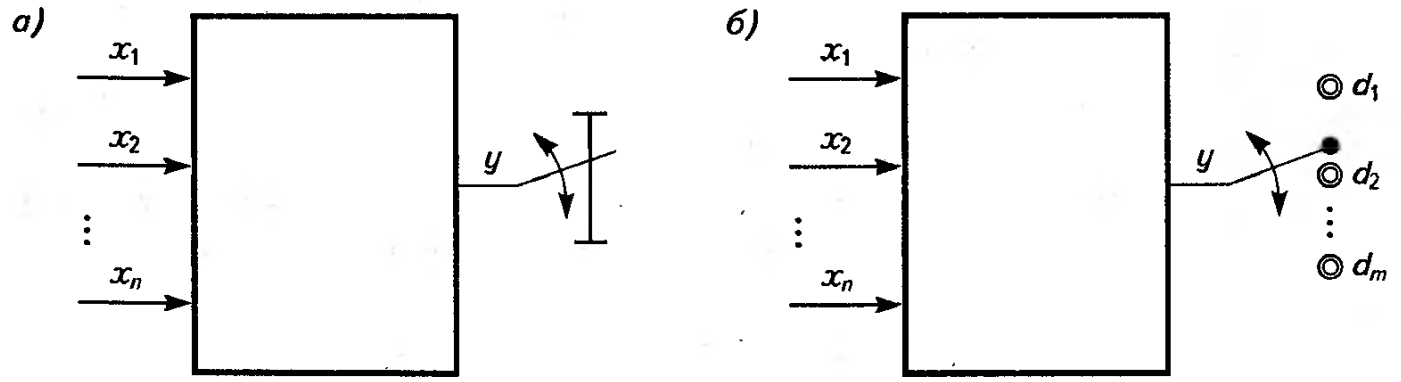


Рис. 1.19. Объекты с непрерывным выходом (*а*) и с дискретным выходом (*б*) [12]

Типовая структура системы нечеткого вывода показана на рис. 1.20. Она содержит такие модули:

- *фаззификатор*, преобразующий фиксированный вектор влияющих факторов (X) в вектор нечетких множеств $\tilde{X}$, необходимых для нечеткого вывода;

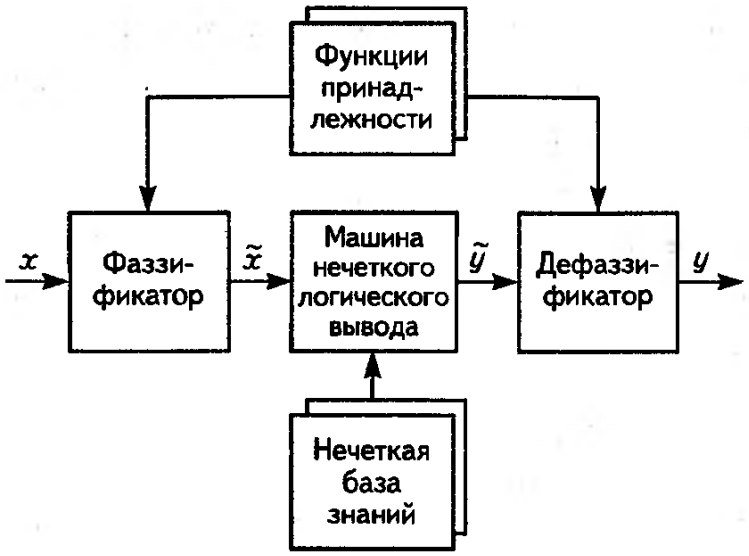


Рис. 1.20. Система нечеткого логического вывода

- *нечеткая база знаний*, содержащая информацию о зависимости $Y = f(X)$ в виде лингвистических правил <Если – то>;
- *функции принадлежности*, используемые для представления лингвистических термов в виде нечетких множеств;
- *машина нечеткого логического вывода*, которая на основе правил базы знаний определяет значение выходной переменной в виде нечеткого множества $\tilde{Y}$, соответствующего нечетким значениям входных переменных $(\tilde{X})$;
- *дефаззификатор*, преобразующий выходное нечеткое множество $\tilde{Y}$ в четкое число Y.

1.6.3. НЕЧЕТКИЕ БАЗЫ ЗНАНИЙ

Определение 41. *Нечеткой базой знаний* называется совокупность нечетких правил <Если – то>, задающих взаимосвязь между входами и выходами исследуемого объекта. Формат нечетких правил такой:

ЕСЛИ <посылка правила>, ТО <заключение правила>.

Посылка правила, или *антецедент* представляет собой утверждение типа «x есть низкий», где «низкий» – это терм, заданный нечетким множеством на универсальном множестве лингвистической переменной x. Квантификаторы «очень», «не», «почти», «более-менее» и другие могут использоваться для модификации термов антецедента.

Заключение правила, или *консеквент* – это факт типа «y есть d», в котором значение выходной переменной может задаваться:

- нечетким термом – «y есть высокий»;
- классом решений – «y есть бронхит»;
- четкой константой – «$y = 5$»;
- четкой функцией от входных переменных – «$y = 5 + 4x$».

Если выходная переменная задана нечетким множеством, тогда правило может быть представлено нечетким отношением. Для нечеткого правила «Если x есть $\tilde{A}$, то y есть $\tilde{B}$» нечеткое отношение $\tilde{R}$ задается на декартовом произведении $U_x \times U_y$, где U_x и U_y – универсальные множества входной и выходной переменной. Для расчета нечеткого отношения можно применять нечеткую импликацию или t-норму. При использовании в качестве t-нормы операции минимума расчет нечеткого отношения $\tilde{R}$ осуществляется так:

$$\mu_{\tilde{R}}(x, y) = \min(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(y)), \quad (x, y) \in U_x \times U_y.$$

Пример 1.14. Следующая нечеткая база знаний описывает зависимость между возрастом водителя (x) и возможностью дорожно-транспортного происшествия (y):

ЕСЛИ x = «молодой», ТО y = «высокая»;
ЕСЛИ x = «средний», ТО y = «низкая»;
ЕСЛИ x = «очень старый», ТО y = «высокая».

Функции принадлежности термов изображены на рис. 1.21. Тогда правнлам базы знаний будут соответствовать нечеткие отношения из рис. 1.22.

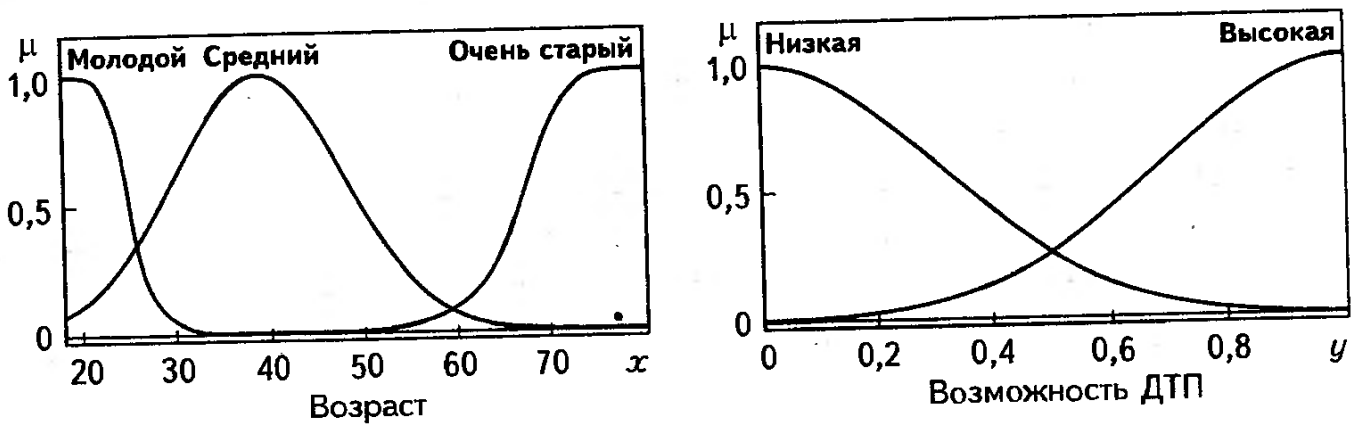


Рис. 1.21. Функции принадлежности термов (из примера 1.14)

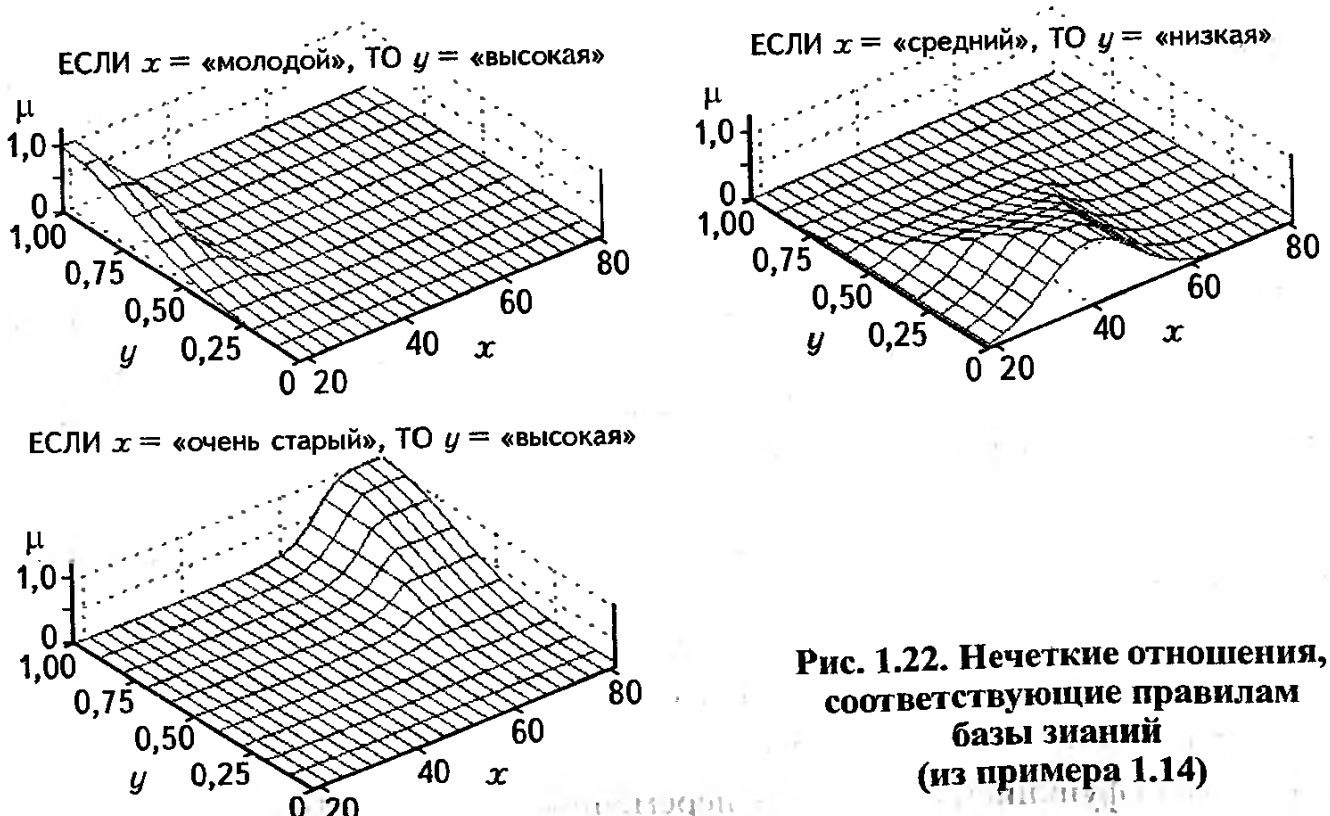


Рис. 1.22. Нечеткие отношения, соответствующие правилам базы зианий (из примера 1.14)

Многомерные зависимости «входы – выходы» задают нечеткими правилами с логическими операциями И и ИЛИ. Например, «Если x_1 = «низкий» И x_2 = «средний» И x_3 = «очень высокий», то y = «высокий». Нечеткую базу знаний, связывающую входы $X = (x_1, x_1, \ldots, x_n)$ с выходом y, представим следующим образом:

$$(x_1 = \tilde{a}_{1j} \quad \Theta_j \quad x_2 = \tilde{a}_{2j} \quad \Theta_j \ldots \Theta_j \quad x_n = \tilde{a}_{nj}) \Rightarrow y = d_j, \quad j = \overline{1, m},$$

где $\tilde{a}_{ij}$ – нечеткий терм, которым оценивается переменная x_i в j-м правиле, $j = \overline{1, m}$; d_i – заключение j-го правила; m – количество правил в базе знаний; Θ_j – логическая операция, связывающая фрагменты антецедента j-го правила (ей может быть логическая операция И или ИЛИ); $\Rightarrow$ – нечеткая импликация.

Если правила содержат только логические операции И, тогда нечеткую базу знаний удобно записывать в виде табл. 1.12.

Таблица 1.12

Нечеткая база знаний

ЕСЛИ				ТО
x_1	x_2	…	x_n	y
$\tilde{a}_{11}$	$\tilde{a}_{21}$	…	$\tilde{a}_{n1}$	d_1
$\tilde{a}_{12}$	$\tilde{a}_{22}$	…	$\tilde{a}_{n2}$	d_2
…	…	…	…	…
$\tilde{a}_{1m}$	$\tilde{a}_{2m}$	…	$\tilde{a}_{nm}$	d_m

Меру уверенности эксперта в адекватности правил учитывают через весовые коэффициенты. Нечеткую базу знаний с весовыми коэффициентами запишем так:

$$(x_1 = \tilde{a}_{1j} \quad \Theta_j \quad x_2 = \tilde{a}_{2j} \quad \Theta_j \ldots \Theta_j \quad x_n = \tilde{a}_{nj} \text{ с весом } w_j) \Rightarrow y = d_j, \quad j = \overline{1, m},$$

где $w_j \in [0, 1]$ – весовой коэффициент j-го правила.

1.6.4. КОМПОЗИЦИОННОЕ ПРАВИЛО НЕЧЕТКОГО ВЫВОДА ЗАДЕ

Определение 42. *Композиционное правило нечеткого вывода Заде.* Если известно нечеткое отношение $\tilde{R}$ между x и y, то при нечетком значении входной переменной $x = \tilde{A}$ нечеткое значение выходной переменной y определяется так:

$$y = \tilde{A} \circ \tilde{R},$$

где знак «∘» – максминная композиция.

Пример 1.15. Дано нечеткое правило «Если $x = \tilde{A}$, то $y = \tilde{B}$», где $\tilde{A} = (0/1; 0{,}1/2; 0{,}5/3; 0{,}8/4; 1/5)$ и $\tilde{B} = (1/5; 0{,}8/10; 0{,}4/15; 0{,}2/20)$. Определить значение выходной переменной y при $x = \tilde{C} = (0{,}3/1; 0{,}5/2; 1/3; 0{,}7/4; 0{,}4/5)$.

Вначале рассчитаем нечеткое отношение, соответствующее правилу «Если $x = \widetilde{A}$, то $y = \widetilde{B}$», применяя в качестве t-нормы операцию минимума:

$$\widetilde{R} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0{,}1 & 0{,}1 & 0{,}1 & 0{,}1 \\ 0{,}5 & 0{,}5 & 0{,}4 & 0{,}2 \\ 0{,}8 & 0{,}8 & 0{,}4 & 0{,}2 \\ 1 & 0{,}8 & 0{,}4 & 0{,}2 \end{bmatrix}.$$

Теперь по формуле $y = \widetilde{C} \circ \widetilde{R}$ рассчитаем нечеткое значение выходной переменной:

$$y = (0{,}3;\ 0{,}5;\ 1;\ 0{,}7;\ 0{,}4) \circ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0{,}1 & 0{,}1 & 0{,}1 & 0{,}1 \\ 0{,}5 & 0{,}5 & 0{,}4 & 0{,}2 \\ 0{,}8 & 0{,}8 & 0{,}4 & 0{,}2 \\ 1 & 0{,}8 & 0{,}4 & 0{,}2 \end{bmatrix} = (0{,}7/5;\ 0{,}7/10;\ 0{,}4/15;\ 0{,}2/20).$$

Композиционное правило Заде составляет основу разнообразных алгоритмов нечеткого логического вывода, которые рассматриваются ниже.

1.6.5. НЕЧЕТКИЙ ЛОГИЧЕСКИЙ ВЫВОД МАМДАНИ

Нечеткий вывод Мамдани выполняется по такой базе знаний:

$$(x_1 = \widetilde{a}_{1j}\ \ \Theta_j\ \ x_2 = \widetilde{a}_{2j}\ \ \Theta_j \ldots \Theta_j\ \ x_n = \widetilde{a}_{nj}\ \text{с весом}\ w_j) \Rightarrow y = \widetilde{d}_j, \quad j = \overline{1, m},$$

в которой все значения входных и выходной переменных заданы нечеткими множествами.

База знаний Мамдани может трактоваться как разбиение пространства влияющих факторов на подобласти с размытыми границами, внутри которых функция отклика принимает нечеткое значение. Правило в базе знаний представляет собой «*информационный сгусток*», отражающий одну из особенностей зависимости «входы – выход». Такие «*сгусти насыщенной информации*» или «*гранулы знаний*» могут рассматриваться как аналог вербального кодирования, которое, как установили психологи, происходит в человеческом мозге при обучении. Возможно поэтому формирование нечеткой базы знаний типа Мамдани обычно не вызывает трудностей у эксперта.

Введем следующие обозначения, необходимые для дальнейшего изложения материала:

- $\mu_j(x_i)$ – функция принадлежности входа $x_i \in [\underline{x}_i,\ \overline{x}_i]$ нечеткому терму $\widetilde{a}_{ij}$, т.е. $\widetilde{a}_{ij} = \int\limits_{x_i \in [\underline{x}_i,\ \overline{x}_i]} \mu_j(x_i) / x_i$;
- $\mu_{d_j}(y)$ – функция принадлежности выхода $y \in [\underline{y},\ \overline{y}]$ нечеткому терму $\widetilde{d}_j$, т.е. $\widetilde{d}_j = \int\limits_{y \in [\underline{y},\ \overline{y}]} \mu_{d_j}(y) / y$.

Степень выполнения посылки j-го правила для текущего входного вектора $X^* = (x_1^*, x_2^*, \ldots, x_n^*)$ рассчитывают так:

$$\mu_j(X^*) = w_j(\mu_j(x_1^*)\ \chi_j\ \mu_j(x_2^*)\ \chi_j \ldots \chi_j\ \mu_j(x_n^*)), \quad j = \overline{1, m},$$

где χ_j обозначает t-норму, если в j-м правиле базы знаний используется логическая операция И (Θ_j = И), и соответствует s-норме при Θ_j = ИЛИ. В нечетком выводе Мамдани треугольные нормы обычно реализуются операциями минимума (t-норма) и максимума (s-норма).

Результат нечеткого вывода можно представить в виде:

$$\tilde{y}^* = \left(\frac{\mu_1(X^*)}{\tilde{d}_1}, \frac{\mu_2(X^*)}{\tilde{d}_2}, \ldots, \frac{\mu_m(X^*)}{\tilde{d}_m} \right). \tag{1.6}$$

Особенность этого нечеткого множества заключается в том, что его носителем является множество нечетких термов $\{\tilde{d}_1, \tilde{d}_2, \ldots, \tilde{d}_m\}$. Для перехода к нечеткому множеству на носителе $|\underline{y}, \bar{y}|$ выполним операции импликации и агрегирования.

В результате логического вывода по j-му правилу базы знаний получаем такое нечеткое значение выходной переменной y:

$$\tilde{d}_j^* = \mathrm{imp}(\tilde{d}_j, \mu_j(X^*)), \quad j = \overline{1, m}, \tag{1.7}$$

где imp – импликация, которая в нечетком выводе обычно реализуются операцией минимума, т.е. «срезанием» функции принадлежности $\mu_{d_j}(y)$ по уровню $\mu_j(X^*)$. Математически это записывается так:

$$\tilde{d}_j^* = \int_{y \in [\underline{y}, \bar{y}]} \min(\mu_j(X^*), \mu_{d_j}(y)) / y.$$

Результат логического вывода по всей базе знаний находят агрегированием нечетких множеств (1.7):

$$\tilde{y}^* = \mathrm{agg}(\tilde{d}_1^*, \tilde{d}_2^*, \ldots, \tilde{d}_m^*),$$

где agg – агрегирование нечетких множеств, которое обычно реализуют операцией максимума.

Иллюстрацией этой формулы служит рис. 1.23, где агрегируются три нечетких множества.

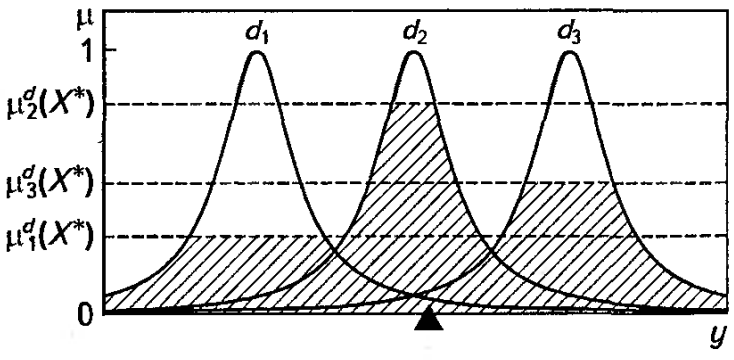


Рис. 1.23. Импликация и агрегирование в нечетком выводе Мамдани

Четкое значение выхода y, соответствующее входному вектору X^*, определяется через дефаззификацию нечеткого множества $\tilde{y}$. Наиболее часто применяется дефаззификация по методу центра тяжести (см. табл. 1.2). На рис. 1.23 резуль-

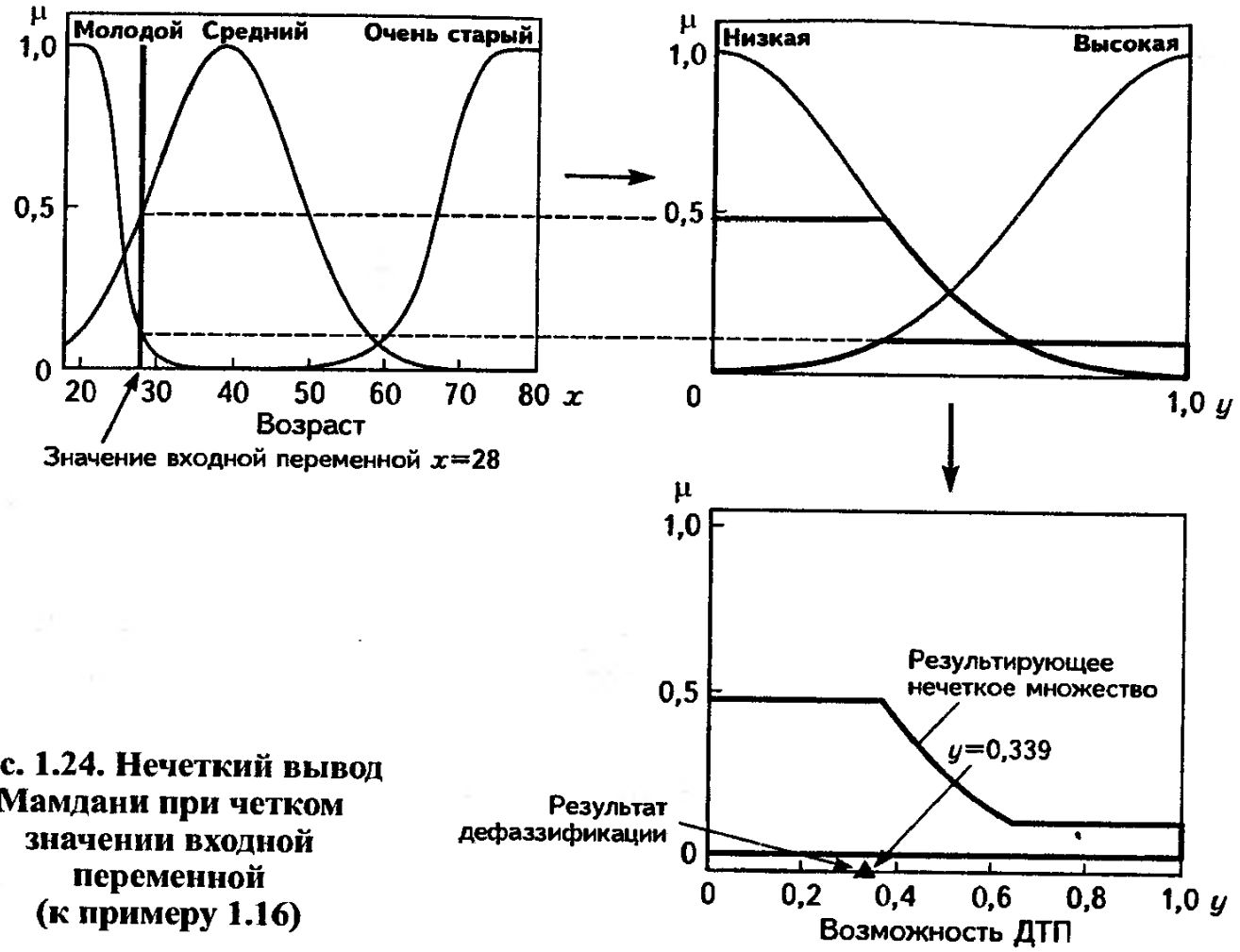


Рис. 1.24. Нечеткий вывод Мамдани при четком значении входной переменной (к примеру 1.16)

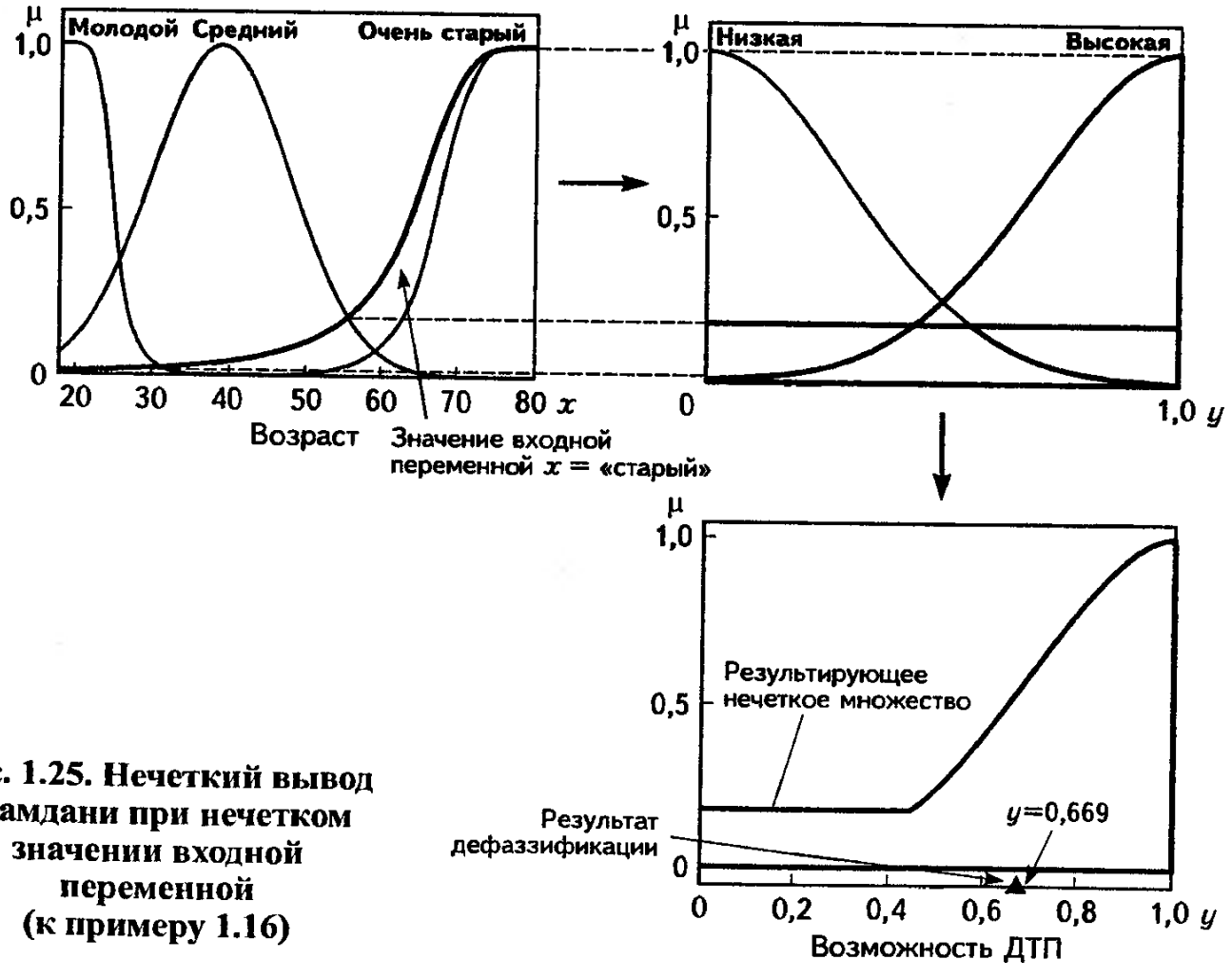


Рис. 1.25. Нечеткий вывод Мамдани при нечетком значении входной переменной (к примеру 1.16)

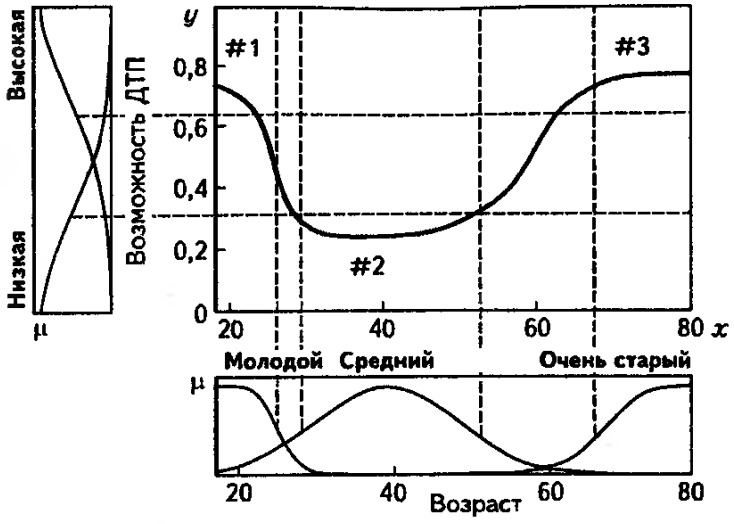


Рис. 1.26. Зависимость «вход – выход» для нечеткой базы знаний (к примеру 1.16)

тат дефаззификации обозначен символом ▲, что соответствует четкому значению выходной переменной.

Пример 1.16. По базе знаний из примера 1.14 выполнить два нечетких вывода при значениях входной переменной $x = 28$ и $x =$ «старый».

Выполнение нечеткого вывода при значениях входной переменной $x = 28$ и $x =$ «старый» показано на рис. 1.24 и 1.25. Импликация реализована операцией минимума, а агрегирование – операцней максимума. Дефаззификация проводнлась по методу центра тяжести. На рнс. 1.26 показана зависимость «вход – выход», описанная нечеткой базой знаний из примера 1.14. Участки графика, соответствующие первому, второму и третьему правилам базы знанний, обозначены на рисунке символами #1, #2 и #3.

1.6.6. НЕЧЕТКИЙ ЛОГИЧЕСКИЙ ВЫВОД СУГЕНО

Нечеткий логический вывод Сугено выполняется по следующей нечеткой базе знаний:

$$(x_1 = \tilde{a}_{1j} \ \Theta_j \ x_2 = \tilde{a}_{2j} \ \Theta_j \ldots \Theta_j \ x_n = \tilde{a}_{nj}) \Rightarrow$$

$$\Rightarrow y = b_{j0} + b_{j1}x_1 + b_{j2}x_2 + \ldots + b_{jn}x_n, \quad j = \overline{1, m},$$

где b_{j0}, b_{j1}, ..., b_{jn} – некоторые действительные числа.

База знаний Сугено аналогична базе знаний Мамдани за исключением заключений правил, которые задаются не нечеткими термами, а линейной функцией от входов:

$$d_j = b_{j0} + \sum_{i=\overline{1,n}} b_{ji}x_i.$$

Правила в базе знаний Сугено являются своего рода переключателями с одного линейного закона «входы – выход» на другой, тоже линейный. Границы подобластей размытые, следовательно, одновременно могут выполняться несколько линейных законов, но с различными степенями. В базе знаний Сугено нет весовых коэффициентов, так как они были бы линейно зависимы с заключениями правил.

Степени принадлежности входного вектора $X^* = (x_1^*, x_2^*, \ldots, x_n^*)$ к значениям $d_j = b_{j0} + \sum_{i=\overline{1,n}} b_{ji}x_i^*$ рассчитывают следующим образом:

$$\mu_{d_j}(X^*)=\mu_j(x_1^*)\ \chi_j\ \mu_j(x_2^*)\ \chi_j\ldots\chi_j\ \mu_j(x_n^*),\quad j=\overline{1,m}.$$

В алгоритме Сугено наиболее часто применяется вероятностное ИЛИ как *s*-норма и произведение как *t*-норма. В результате вывода по всей базе знаний получаем нечеткое множество $\tilde{y}$, соответствующее входному вектору X^*:

$$\tilde{y}=\left(\frac{\mu_{d_1}(X^*)}{d_1},\ \frac{\mu_{d_2}(X^*)}{d_2},\ \ldots,\ \frac{\mu_{d_m}(X^*)}{d_m}\right).$$

Результирующее значение выхода y находят как суперпозицию линейных законов, выполняемых в данной точке $X^*=(x_1^*,\ x_2^*,\ \ldots,\ x_n^*)$ n-мерного факторного пространства. Для этого нечеткое множество $\tilde{y}$ дефаззифицируют, находя взвешенное среднее

$$y=\frac{\sum\limits_{j=\overline{1,m}}\mu_{d_j}(X^*)d_j}{\sum\limits_{j=\overline{1,m}}\mu_{d_j}(X^*)}$$

или взвешенную сумму

$$y=\sum_{j=\overline{1,m}}\mu_{d_j}(X^*)d_j.$$

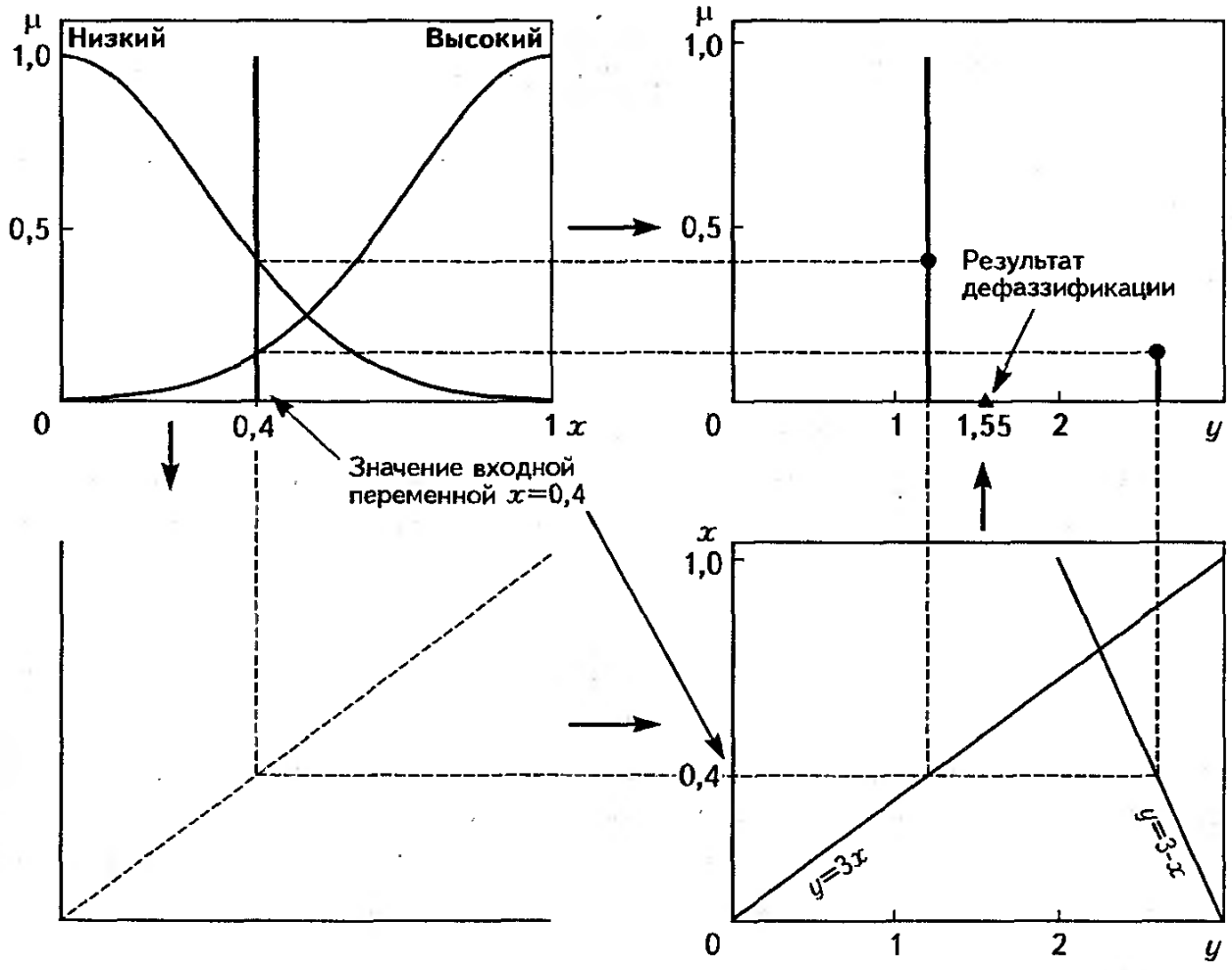


Рис. 1.27. Выполнение нечеткого вывода Сугено (к примеру 1.17)

Пример 1.17. Известна нечеткая база знаний из двух правил:

ЕСЛИ x = «низкий», ТО $y = 3x$;

ЕСЛИ x = «высокий», ТО $y = 3 - x$.

Функции принадлежности термов заданы следующими выражениям:

$$\mu_{\text{«низкий»}}(x) = \exp\left(-\frac{x^2}{0{,}18}\right)$$

и $$\mu_{\text{«высокий»}}(x) = \exp\left(-\frac{(x-1)^2}{0{,}18}\right),$$

$x \in [0, 1]$.

Необходимо выполнить нечеткий вывод при значении входной переменной $x = 0{,}4$.

Выполнение нечеткого вывода показано на рис. 1.27. Дефаззнификация проводилась по методу центра тяжести (взвешенного среднего). На рис. 1.28 показана зависимость «вход – выход» для приведенной выше нечеткой базы знаний. Участки графика, соответствующие первому и второму правилу базы знаний, обозначены на этом рисунке символами #1 и #2.

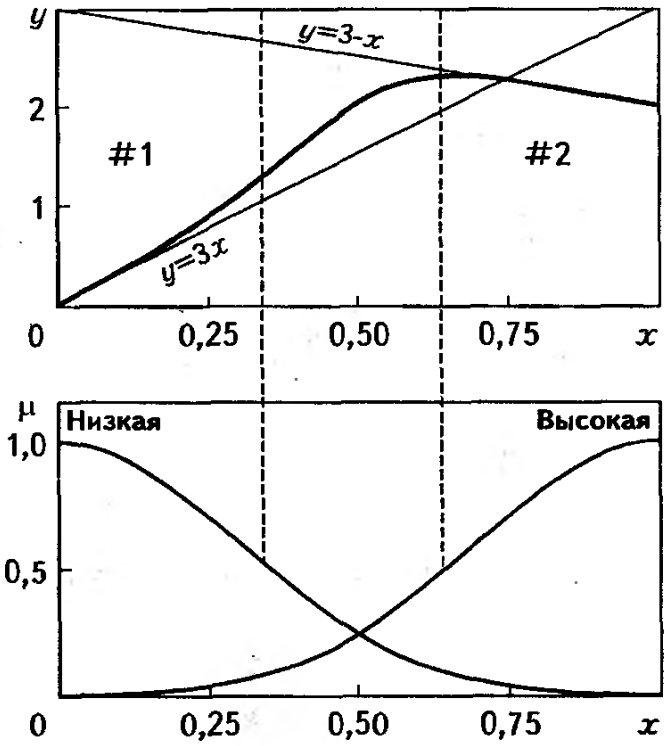


Рис. 1.28. Зависимость «вход – выход» для нечеткой базы знаний (из примера 1.17)

1.6.7. НЕЧЕТКИЙ ЛОГИЧЕСКИЙ ВЫВОД ПО СИНГЛТОННОЙ БАЗЕ ЗНАНИЙ

Рассмотрим нечеткую базу знаний, в которой посылки заданы нечеткими множествами, а заключения правил – четкими числами:

$$(x_1 = \tilde{a}_{1j} \ \Theta_j \ x_2 = \tilde{a}_{2j} \ \Theta_j \ldots \Theta_j \ x_n = \tilde{a}_{nj}) \Rightarrow y = d_j, \quad j = \overline{1, m},$$

где d_j – некоторые действительные числа.

Синглтонная база знаний может рассматриваться как частный случай базы знаний Мамдани. Четкое число d_j, задающее заключение j-го правила, можно рассматривать как нечеткое множество с синглтонной функцией принадлежности (см. табл. 1.2). Синглтонная нечеткая база знаний может также трактоваться как частный случай базы знаний Сугено. Точнее говоря, она эквивалентна нечеткой базе знаний Сугено нулевого порядка, в которой коэффициенты при входных переменных в линейных законах «входы – выход» равны нулю. Поэтому нечеткий вывод осуществляется по формулам из предыдущего подраздела 1.6.6. В синглтонной базе знаний нет весовых коэффициентов, так как они были бы линейно зависимы с заключениями правил.

Пример 1.18. Известна нечеткая база знаний:

ЕСЛИ x_1 = «низкий» И x_2 = «низкий», ТО $y = 5$;

ЕСЛИ x_1 = «низкий» И x_2 = «высокий», ТО $y = -6$;

ЕСЛИ x_1 = «высокий» И x_2 = «низкий», ТО $y = -1$;
ЕСЛИ x_1 = «высокий» И x_2 = «высокий», ТО $y = 10$.

Функции принадлежности термов входных переменных показаны на рис. 1.29.

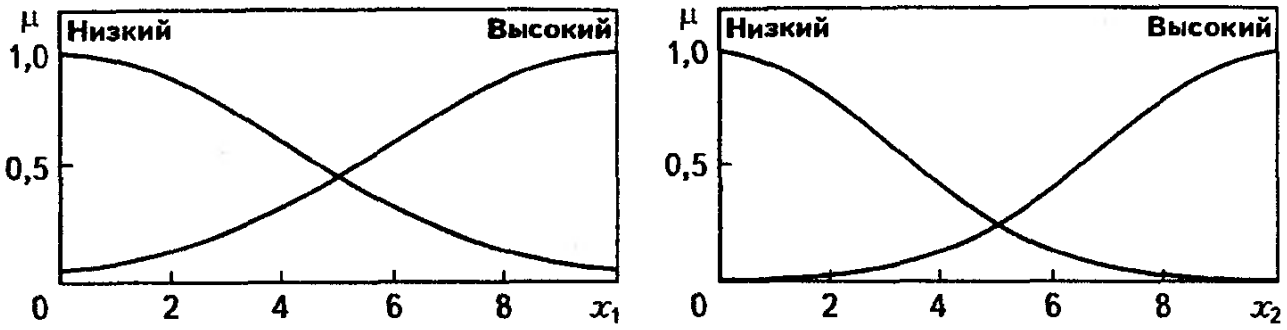


Рис. 1.29. Функции принадлежности входных переменных (к примеру 1.18)

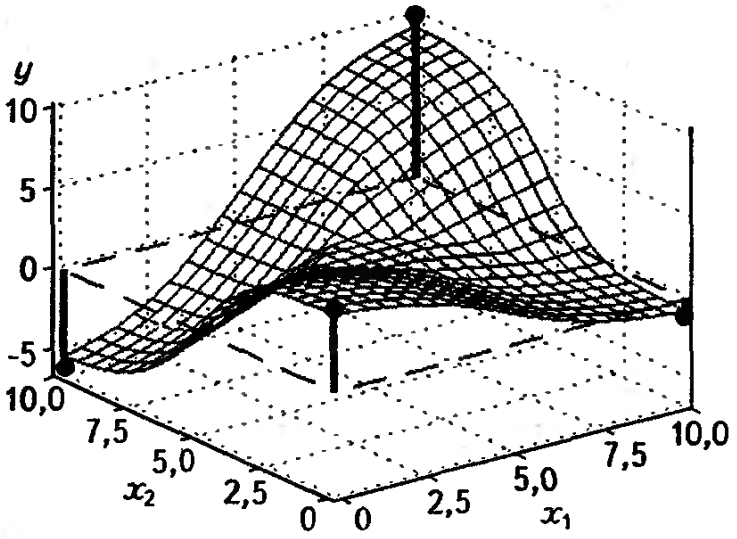


Рис. 1.30. Поверхность «входы – выход» (из примера 1.18)

На рис. 1.30 изображена поверхность «входы – выход» для приведенной нечеткой базы знаний при реализации *t*-нормы операцией минимума. На этом рисунке вертикальные столбики показывают заключения правил базы знаний. Нечеткий вывод аппроксимирует четкие значения заключений правил на все пространство входных переменных. Поверхность как бы натягивается на эти столбики – четкие значения заключений правил.

1.6.8. НЕЧЕТКИЙ ЛОГИЧЕСКИЙ ВЫВОД ДЛЯ ЗАДАЧ КЛАССИФИКАЦИИ

Задача классификации состоит в отнесении объекта, заданного вектором информативных признаков $X = (x_1, x_2, \ldots, x_n)$, к одному из наперед описанных классов $\{d_1, d_2, \ldots, d_m\}$. Классификация соответствует отображению вида:

$$X = (x_1, x_2, \ldots, x_n) \to y \in \{d_1, d_2, \ldots, d_m\}.$$

Для классификации необходима нечеткая база знаний вида [12]:

ЕСЛИ $x_1 = a_{1,j1}$ И $x_2 = a_{2,j1}$ И...И $x_n = a_{n,j1}$, с весом w_{j1}
ИЛИ $x_1 = a_{1,j2}$ И $x_2 = a_{2,j2}$ И...И $x_n = a_{n,j2}$, с весом w_{j2}
...,
ИЛИ $x_1 = a_{1,jk_j}$ И $x_2 = a_{2,jk_j}$ И...И $x_n = a_{n,jk_j}$, с весом w_{jk_j}
ТО $y = d_j$, $j = \overline{1,m}$,

где $a_{j,jp}$ – нечеткий терм, которым оценивается переменная x_i в правиле с номером jp, $p = \overline{1, k_j}$; k_j – количество правил, описывающих класс d_j.

Степени принадлежности объекта $X^* = (x_1^*, x_2^*, \ldots, x_n^*)$ классам d_j рассчитывают так:

$$\mu_{d_j}(X^*) = \max_{p=\overline{1,k_j}} w_{jp} \min_{i=\overline{1,n}} (\mu_{jp}(x_i^*)), \quad j = \overline{1, m},$$

где $\mu_{jp}(x_i)$ – функция принадлежности входа x_i нечеткому терму $a_{i,jp}$.

В качестве решения выбирают класс с максимальной степенью принадлежности:

$$y^* = \arg\max_{\{d_1, d_2, \ldots, d_m\}} (\mu_{d_1}(X^*), \mu_{d_2}(X^*), \ldots, \mu_{d_m}(X^*)).$$

Пример 1.19. Известна нечеткая база знаний из четырех правил:
ЕСЛИ x_1 = «низкий» И x_2 = «низкий», ТО y = класс 1;
ЕСЛИ x_1 = «средний» И x_2 = «высокий», ТО y = класс 2;
ЕСЛИ x_1 = «высокий» И x_2 = «высокий», ТО y = класс 3;
ЕСЛИ x_1 = «высокий» И x_2 = «низкий», ТО y = класс 2.

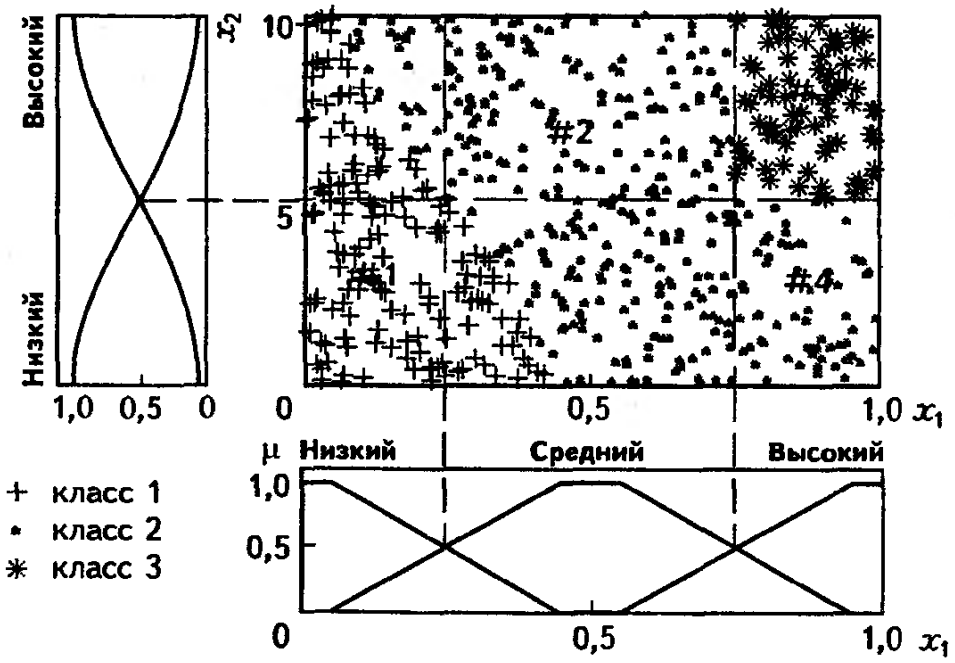


Рис. 1.31. Классификация по нечеткой базе знаний (из примера 1.19)

На рис. 1.31 приведены результаты классификации 600 объектов при реализации t-нормы операцией минимума и агрегирования операцией максимума. Области, соответствующие первому, второму, третьему и четвертому правилам базы знаний, обозначены на рисунке символами #1, #2, #3 и #4.

1.6.9. ИЕРАРХИЧЕСКИЕ СИСТЕМЫ НЕЧЕТКОГО ЛОГИЧЕСКОГО ВЫВОДА

Для моделирования многомерных зависимостей «входы – выход» целесообразно использовать иерархические системы нечеткого вывода. В таких системах выход одной базы знаний подается на вход другой, более высокого уровня иерархии. В иерархических базах знаний отсутствуют обратные связи. На рис. 1.32 приведен пример иерархической нечеткой системы, моделирующей зависимость $=f(x_1, x_2, x_3, x_4, x_5, x_6)$ с помощью трех баз знаний. Эти базы знаний описывают зависимости $y_1 = f_1(x_1, x_2)$, $y_2 = f_2(x_4, x_5, x_6)$ и $y = f_3(y_1, x_3, y_2)$.

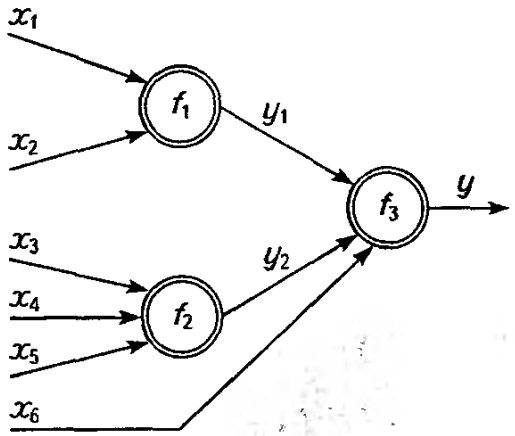


Рис. 1.32. Пример иерархической системы нечеткого вывода

Применение иерархических нечетких баз знаний позволяет преодолеть «проклятие размерности». При большом количестве входов эксперту трудно описать нечеткими правилами причинно-следственные связи. Это обусловлено тем, что в оперативной памяти человека может одновременно храниться не более 7±2 понятий–признаков [34]. Следовательно, количество входных переменных в одной базе знаний не должно превышать это магическое число. Наш опыт создания нечетких экспертных систем свидетельствует, что хорошие базы знаний получаются, когда количество входов не превышает пяти. Поэтому, при большем количестве входных переменных необходимо их иерархически классифицировать с учетом приведенных выше рекомендаций. Выполнение такой классификации не составляет трудностей для эксперта, так как процесс переработки информации человеком происходит по иерархическому принципу [5].

Еще одно преимущество иерархических баз знаний – компактность. Небольшим количеством нечетких правил в иерархических базах знаний можно адекватно описать многомерные зависимости «входы – выход». Проиллюстрируем этот тезис следующим примером. Пусть для лингвистической оценки переменных используется по пять термов. Тогда максимальное количество правил для задания зависимости $y=f(x_1, x_2, x_3, x_4, x_5, x_6)$ с помощью одной базы знаний будет равным $5^6 = 15\,625$. Конечно, для адекватного описания зависимости «входы – выход» необходимо значительно меньше нечетких правил, предположим 2–3% от общего количества. Тогда база знаний будет состоять из 300–500 правил. Для иерархической базы знаний (рис. 1.32), описывающей ту же зависимость, максимальное количество правил будет равным $5^2 + 5^3 + 5^3 = 275$, хотя обычно хватает и 25–35 правил. Причем это короткие правила с двумя–тремя входными переменными.

При нечетком выводе по иерархической базе знаний процедуры дефаззификации и фаззификации для промежуточных переменных y_1 и y_2 (см. рис. 1.32) не

выполняются. Результат логического вывода в виде нечеткого множества (1.6) напрямую передается в машину нечеткого вывода следующего уровня иерархии. Поэтому для промежуточных переменных в иерархических нечетких базах знаний достаточно задать только терм-множества, без описания функций принадлежностей. С примерами использования иерархических нечетких баз знаний для решения прикладных задач можно ознакомиться в следующей литературе:

- медицинская диагностика, прогнозирование числа заболеваний, оценка качества проектов, расследование дорожно-транспортных происшествий, управление технологическим процессом биоконверсии [37, 12];
- диагностирование трещин кирпичных конструкций [8, 9, 10, 39];
- прогнозирование результатов футбольных чемпионатов [11];
- управление динамическими объектами [24].

1.6.10. НЕЙРО-НЕЧЕТКИЕ СЕТИ

Систему нечеткого логического вывода можно представить в виде *нейро-нечеткой сети* – нейронной сети прямого распространения сигнала особого типа. История нейро-нечетких сетей начинается с 1991 г., когда Янг (Jang) предложил ANFIS-модель (*Adaptive-Network-based Fuzzy Inference System*) [30]. Архитектура нейро-нечеткой сети изоморфна нечеткой базе знаний. В нейро-нечетких сетях используются дифференцируемые реализации треугольных норм (умножение и вероятностное ИЛИ), а также гладкие функции принадлежности. Это позволяет применять для настройки нейро-нечетких сетей быстрые алгоритмы обучения нейронных сетей, основанные на методе обратного распространения ошибки. Ниже описываются архитектура и правила функционирования каждого слоя ANFIS-сети. Материал базируется на работе [35].

ANFIS реализует систему нечеткого вывода Сугено в виде пятислойной нейронной сети прямого распространения сигнала. Назначение слоев следующее:

- *первый слой* – термы входных переменных;
- *второй слой* – антецеденты (посылки) нечетких правил;
- *третий слой* – нормализация степеней выполнения правил;
- *четвертый слой* – заключения правил;
- *пятый слой* – агрегирование результата, полученного по различным правилам.

Входы сети в отдельный слой не выделяются. На рис. 1.33 изображена ANFIS-сеть с двумя входными переменными (x_1 и x_2) и четырьмя нечеткими правилами. Для лингвистической оценки входной переменной x_1 используется три терма, для переменной x_2 – два терма.

Введем следующие обозначения, необходимые для дальнейшего изложения:

- $x_1, x_2, \ldots, x_n$ – входы сети;
- y – выход сети;

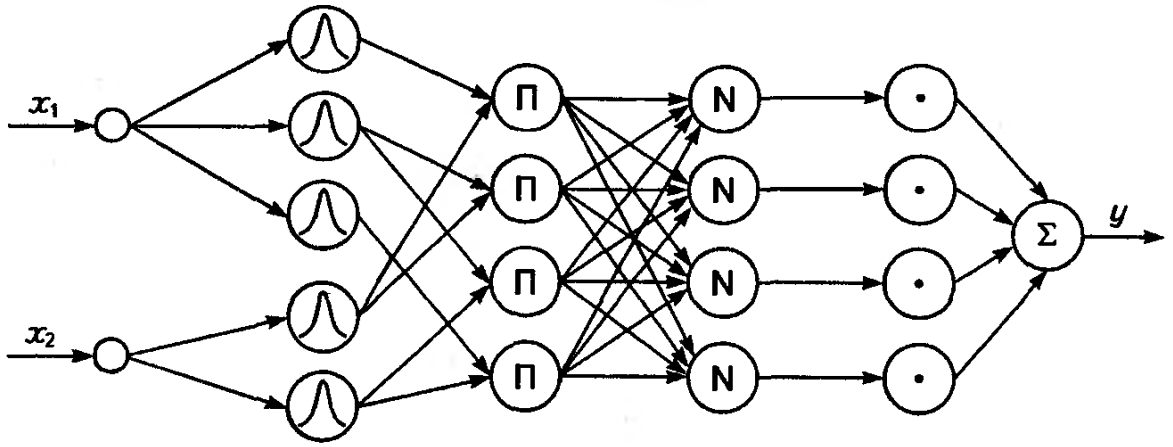


Рис. 1.33. Пример нейро-нечеткой сети

- R_r: Если $x_1 = a_{1,r}$ И...И $x_n = a_{n,r}$, то $y = b_{0,r} + b_{1,r}x_1 + \ldots + b_{n,r}x_n$ – нечеткое правило с порядковым номером r;
- m – количество правил, $r = \overline{1, m}$;
- $a_{i,r}$ – нечеткий терм с функцией принадлежности $\mu_r(x_i)$, применяемый для лингвистической оценки переменной x_i в r-ом правиле ($r = \overline{1, m}$, $i = \overline{1, n}$);
- $b_{q,r}$ – коэффициенты в заключении r-го правила ($r = \overline{1, m}$, $q = \overline{0, n}$).

ANFIS-сеть функционирует следующим образом.

Слой 1. Каждый узел первого слоя представляет один терм с колоколообразной функцией принадлежности. Входы сети x_1, x_2, ..., x_n соединены только со своими термами. Количество узлов первого слоя равно сумме мощностей терм-множеств входных переменных. На выход узла подается степень принадлежности значения входной переменной соответствующему нечеткому терму:

$$\mu_r(x_i) = \frac{1}{1 + \left| \frac{x_i - c}{a} \right|^{2b}},$$

где a, b и c – настраиваемые параметры функции принадлежности.

Слой 2. Количество узлов второго слоя равно m. Каждый узел этого слоя соответствует одному нечеткому правилу. Узел второго слоя соединен с теми узлами первого слоя, которые формируют антецеденты соответствующего правила. Следовательно, каждый узел второго слоя может принимать от 1 до n сигналов. Выходом узла является степень выполнения правила, которая рассчитывается как произведение входных сигналов. Обозначим выходы узлов этого слоя через τ_r, $r = \overline{1, m}$.

Слой 3. Количество узлов третьего слоя также равно m. Каждый узел этого слоя рассчитывает относительную степень выполнения нечеткого правила по формуле:

$$\tau_r^* = \frac{\tau_r}{\sum\limits_{j=\overline{1,m}} \tau_j}.$$

Слой 4. Количество узлов четвертого слоя также равно m. Каждый узел соединен с одним узлом третьего слоя, а также со всеми входами сети (на рис. 1.33 связи с входами не показаны). Узел четвертого слоя рассчитывает вклад одного нечеткого правила в выход сети по такой формуле:

$$y_r = \tau_r^*(b_{0,r} + b_{1,r}x_1 + \ldots + b_{n,r}x_n).$$

Слой 5. Единственный узел этого слоя суммирует вклады всех правил:

$$y = y_1 + \ldots y_r \ldots + y_m .$$

Типовые процедуры обучения нейронных сетей могут быть применены для настройки ANFIS-сети, так как в ней используеются только дифференцируемые функции. Обычно применяется комбинация градиентного спуска в виде алгоритма обратного распространения ошибки и метода наименьших квадратов. Алгоритм обратного распространения ошибки настраивает параметры антецедентов правил, т.е. функций принадлежности. Методом наименьших квадратов оцениваются коэффициенты заключений правил, так как они линейно связаны с выходом сети. Каждая итерация процедуры настройки выполняется в два этапа. На первом этапе на входы подается обучающая выборка и по невязке между желаемым и действительным поведением сети методом наименьших квадратов находятся оптимальные параметры узлов четвертого слоя. На втором этапе остаточная невязка передается с выхода сети на входы и методом обратного распространения ошибки модифицируются параметры узлов первого слоя. При этом найденные на предыдущем этапе коэффициенты заключений правил не изменяются. Итерационная процедура настройки продолжается, пока невязка превышает заранее установленное значение. Для настройки функций принадлежностей, кроме метода обратного распространения ошибки, могут использоваться и другие алгоритмы оптимизации, например, метод Левенберга–Марквардта.

Глава 2. ТЕОРИЯ ПРОЕКТИРОВАНИЯ НЕЧЕТКИХ СИСТЕМ

В главе рассматриваются методы построения нечетких систем на основе идентификации нелинейных зависимостей нечеткими базами знаний, нечеткой кластеризации и подхода Беллмана–Заде по достижению нечеткой цели при нечетких ограничениях.

2.1. ИДЕНТИФИКАЦИЯ НЕЛИНЕЙНЫХ ЗАВИСИМОСТЕЙ НЕЧЕТКИМИ БАЗАМИ ЗНАНИЙ

Идентификация нелинейных зависимостей, т.е. построение их моделей по результатам наблюдений, является важной задачей в технике, экономике, политике, медицине, спорте и в других областях [10, 23]. Работы по нечеткой идентификации нелинейных зависимостей интенсивно проводятся за рубежом с 1990-х годов. Среди русскоязычных публикаций выделим работы профессора Ротштейна [6, 12, 13, 14], в которых разработан метод двухэтапной идентификации нелинейных зависимостей с помощью нечетких баз знаний. На первом этапе выполняется структурная идентификация. Она представляет собой формирование нечеткой базы знаний, которая грубо отражает нелинейную взаимосвязь «входы – выход» с помощью лингвистических правил <Если – то>. Эти правила генерируются экспертом либо получаются в результате экстракции нечетких знаний из экспериментальных данных. На втором этапе происходит параметрическая идентификация исследуемой зависимости путем нахождения таких параметров нечеткой базы знаний, которые минимизируют отклонение результатов нечеткого моделирования от экспериментальных данных. Настраиваемыми параметрами являются веса правил и параметры функций принадлежности нечетких термов.

Раздел посвящен настройке параметров нечеткой базы знаний по экспериментальным данным, т.е. обучению с учителем. Вначале излагаются методы идентификации нечеткими базами знаний Мамдани и Сугено применительно к объектам с непрерывным выходом, что соответствует задачам прогнозирования, многокритериального анализа и управления техническими объектами. Затем рассматривается идентификация нечеткими базами знаний для объектов с дискрет-

ным выходом, что соответствует медицинской и технической диагностике, ситуационному управлению и другим задачам классификации. Материал раздела базируется на публикациях [12, 13, 20, 37, 41].

2.1.1. НАСТРОЙКА НЕЧЕТКОЙ БАЗЫ ЗНАНИЙ МАМДАНИ

Предполагается, что модель зависимости $y = f(X)$ задана нечеткой базой знаний Мамдани. Будем считать, что существует также обучающая выборка из M пар экспериментальных данных, связывающих входы $X_r = (x_{r,1}, x_{r,2}, \ldots, x_{r,n})$ с выходом y исследуемой зависимости:

$$(X_r, y_r), \quad r = \overline{1, M}, \tag{2.1}$$

где $X_r = (x_{r,1}, x_{r,2}, \ldots, x_{r,n})$ – входной вектор в r-й паре обучающей выборки и y_r – соответствующий выход.

Введем следующие обозначения:

- P – вектор параметров функций принадлежности термов входных и выходной переменных;
- W – вектор весовых коэффициентов правил базы знаний;
- $F(P, W, X_r)$ – результат вывода по нечеткой базе знаний Мамдани с параметрами (P, W) при значении входов X_r. Нечеткий вывод осуществляется по формулам подраздела 1.6.5.

Согласно методу наименьших квадратов, настройка нечеткой базы знаний Мамдани сводится к следующей задаче математического программирования: *найти такой вектор* (P, W), *чтобы*

$$\mathrm{RMSE} = \sqrt{\frac{1}{M} \sum_{r=\overline{1,M}} (y_r - F(P, W, X_r))^2} \to \min. \tag{2.2}$$

В этой задаче оптимизации на управляемые переменные P обычно налагают ограничения, обеспечивающие линейную упорядоченность элементов терм-множеств. Такие ограничения не позволяют алгоритмам оптимизации сделать, например, нечеткое множество «низкий» больше «высокого». Кроме того, ядра нечетких множеств не должны выходить за пределы диапазонов изменения соответствующих переменных. Такими ограничениями обеспечивается прозрачность нечеткой базы знаний после настройки, т.е. возможность содержательной интерпретации правил. Что касается вектора W, то его координаты должны находиться в диапазоне [0, 1]. Если к уровню интерпретабельности базы знаний предъявляются высокие требования, то веса правил не настраивают, оставляя их равными 1. Возможен и промежуточный вариант, когда весовые коэффициенты могут принимать значения 0 или 1. В этом случае нулевое значение весового коэффициента эквивалентно исключению правила из нечеткой базы знаний.

Задача (2.2) может быть решена различными технологиями оптимизации, среди которых часто применяется метод наискорейшего спуска, квазиньютоновские методы и генетические алгоритмы.

Пример 2.1. Результаты наблюдений зависимости $y = f(x_1, x_2)$ представлены трехмерным графиком (рис. 2.1) и обучающей выборкой из 20 пар «входы – выход» (табл. 2.1). Необходимо идентифицировать зависимость нечеткой базой знаний Мамдани и сравнить полученную модель с эталонной зависимостью $y = x_1^2 \sin(x_2 - 1)$ в области $x_1 \in [-7; 3]$ и $x_2 \in [-4{,}4; 1{,}7]$. Адекватность нечеткой модели необходимо проверить по критерию (2.2) на тестовой выборке из 1000 случайно сгенерированных пар «входы – выход». Выборки данных показаны на рис. 2.2. Данные доступны по адресу *www.vinnitsa.com/shtovba*.

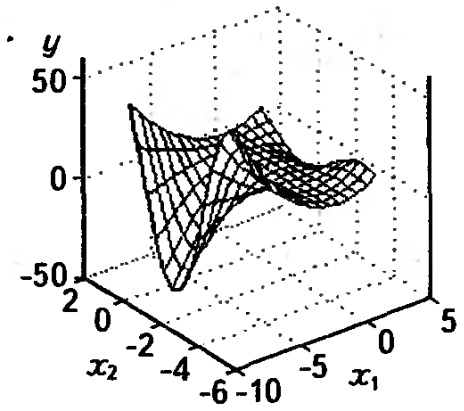


Рис. 2.1. Поверхность «входы – выход» (к примеру 2.1)

Таблица 2.1

Обучающая выборка (к примеру 2.1)

x_1	x_2	y
–6,5	–4,0	40,51
–6,3	–1,0	–36,09
–4,0	1,5	7,67
2,9	–4,3	7,00
–5,0	0	–21,04
–2,0	–3,0	3,02
3,0	–1,5	–5,39
2,9	1,0	0
–5,8	–1,8	–11,27
–5,0	–1,0	–22,73
0,5	1,6	0,14
–4,5	1,0	0
–7,0	1,1	4,90
–6,0	0,5	–17,26
–5,0	–0,1	–22,28
–1,5	0,7	–0,66
–0,9	–4,0	0,78
–5,7	–4,4	25,11
–5,2	–1,6	–13,94
–6,5	–4,0	40,51

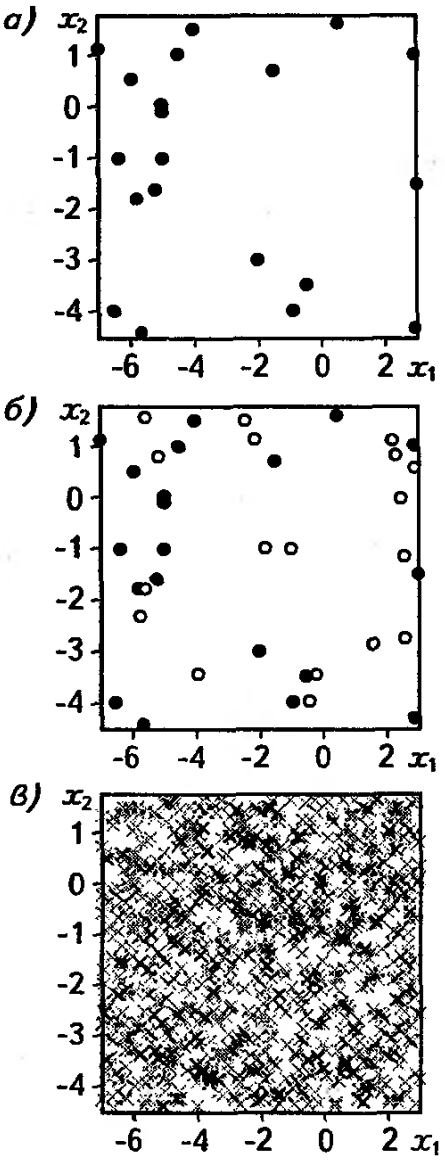


Рис. 2.2. Распределения данных обучающей и тестовой выборок (к примеру 2.1)

а – обучающая выборка из 20 точек; *б* – обучающая выборка из 40 точек; *в* – тестовая выборка

Входы и выход нечеткой модели будем рассматривать как лингвистические переменные, значения которых определяются из следующих терм-множеств: {«низкий», «средний», «высокий»} для x_1 и x_2, и {«низкий», «ниже среднего», «средний», «выше среднего», «высокий»} для y. Термы представим нечеткими множествами с гауссовыми функциями принадлежности. Выбор гауссовой функции принадлежности обусловлен ее достаточной гибкостью и простотой – она задается лишь двумя параметрами. Это сокращает размерность задачи оптимизации при настройке нечеткой базы знаний. Графики исходных функций принадлежности приведены на рис. 2.3*а*.

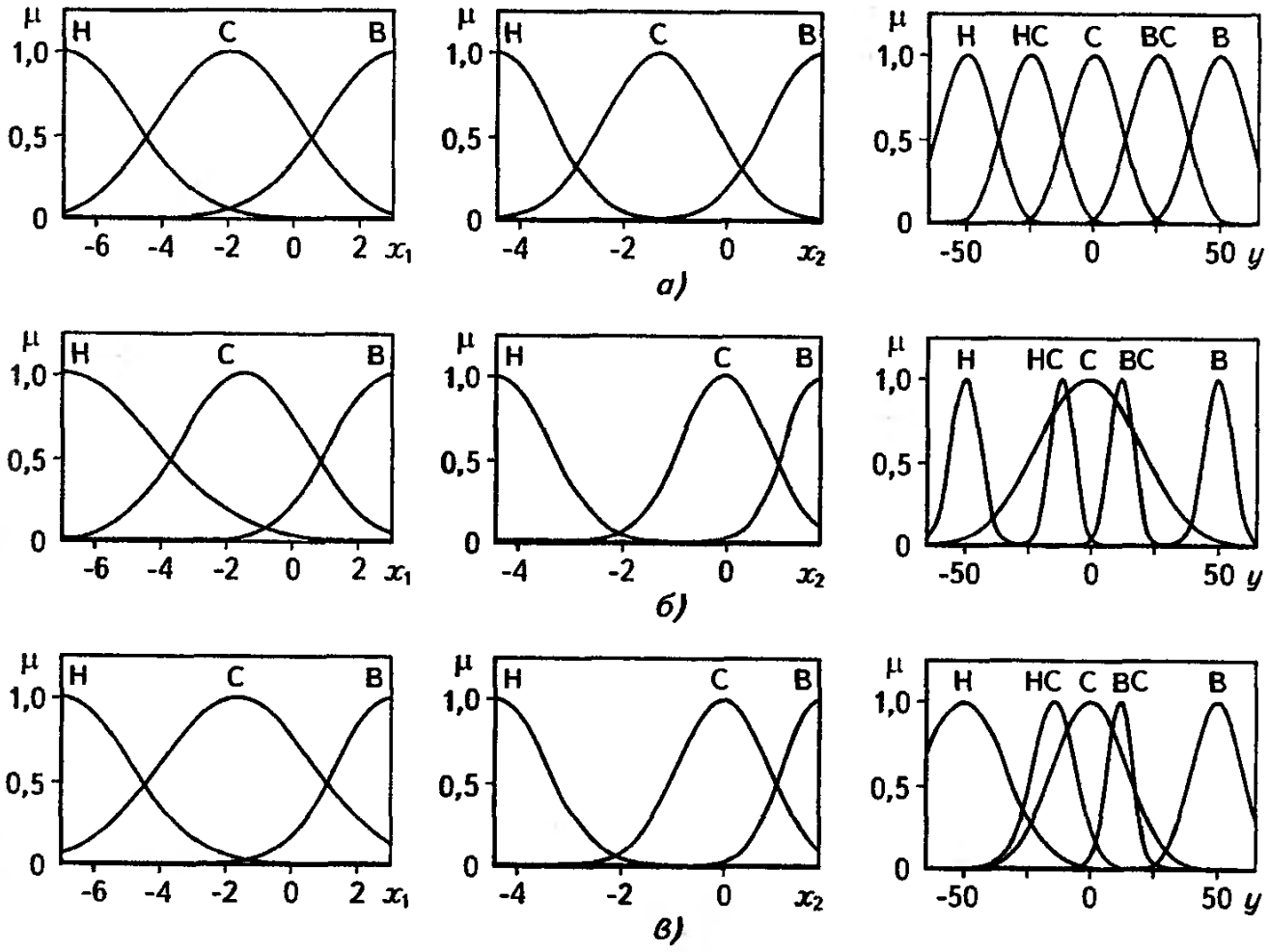


Рис. 2.3. Функции принадлежности нечеткой модели Мамдани (к примеру 2.1)

Н – низкий, НС – ниже среднего, С – средний, ВС – выше среднего, В – высокий

а – до обучения; *б* – после обучения на выборке из 20 точек; *в* – после обучения на выборке из 40 точек

Нечеткая база знаний сгенерирована экспертом визуально на основе рис. 2.1. Она состоит из семи правил, которые сведены в табл. 2.2. В качестве t-нормы выберем максимум. Дефаззификацию будем проводить по методу центра тяжести, так как он обеспечивает наилучшие показатели точности и скорости настройки нечеткой базы знаний [15].

Поверхность «входы – выход» исходной нечеткой модели показана на рис. 2.4*а*. Как видно из этого рисунка, до настройки нечеткая модель отражает

Таблица 2.2

Нечеткая база знаний Мамдани (к примеру 2.1)

x_1	x_2	y	Веса правил (W)		
			до настройки	после настройки на выборке из 20 точек	после настройки на выборке из 40 точек
Низкий	Низкий	Высокий	1	1	1
Низкий	Средний	Низкий	1	1	1
Низкий	Высокий	Высокий	1	1	1
Средний	–	Средний	1	1	1
Высокий	Низкий	Выше среднего	1	0,921	0,977
Высокий	Средний	Ниже среднего	1	0,655	0,479
Высокий	Высокий	Выше среднего	1	0,657	0,105

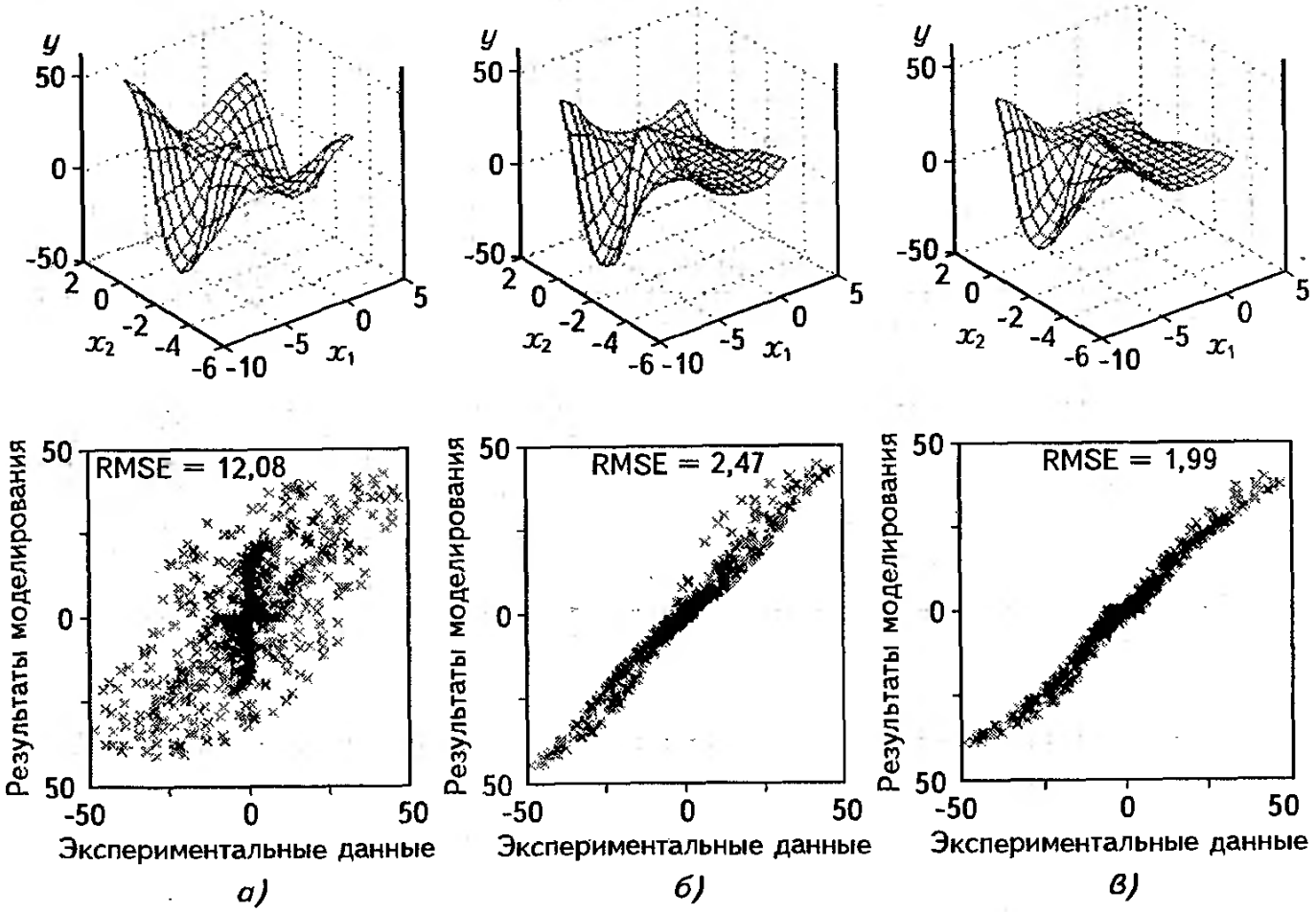


Рис. 2.4. Тестирование нечетких моделей Мамдани (к примеру 2.1)

а – до обучения; *б* – после обучения на выборке из 20 точек; *в* – после обучения на выборке из 40 точек

основные особенности идентифицируемой зависимости. Тестирование модели (рис. 2.4*а*) показывает, что невязка между экспериментальными данными и результатами нечеткого моделирования достаточно большая – 12,08.

Будем настраивать следующие 19 параметров нечеткой базы знаний:

- весовые коэффициенты 5, 6 и 7-го правил;
- коэффициенты концентраций функций принадлежности термов переменных x_1, x_2 и y;
- координаты максимумов функций принадлежностей термов «средний» входных переменных;
- координаты максимумов функций принадлежностей некрайних термов «ниже среднего», «средний» и «выше среднего» выходной переменной.

Координаты максимумов функций принадлежностей крайних термов «низкий» и «высокий» не настраиваются, потому что они совпадают с границами диапазонов изменения переменных. Не будем изменять и весовые коэффициенты первых четырех правил базы знаний, так как их истинность не вызывает сомнений.

Настройку нечеткой модели осуществим квазиньютоновским методом Бройдена–Флетчера–Голфарбда–Шэнно на протяжении 10 итераций. Найденные в результате настройки оптимальные функции принадлежности нечетких термов и веса правил показаны на рис. 2.3*б* и в табл. 2.2, соответственно. В результате настройки невязка на обучающей выборке уменьшилась с 15,33 до 1,95. Поверхность «входы – выход» настроенной нечеткой модели показана на рис. 2.4*б*. Как видно из этого рисунка, после настройки нечеткая модель хорошо отражает поведение идентифицируемой зависимости. На это также указывает и малое значение невязки на тестовой выборке, равное 2,47. Обратим внимание на то, что обучающая выборка содержит всего 20 пар «входы – выход», что всего на единицу больше количества настраиваемых параметров. Несмотря на это, результаты настройки хорошие, что объясняется качественной базой знаний. Фактически, исходная нечеткая база знаний (см. табл. 1.1) уже является грубой моделью идентифицируемой зависимости, которая на этапе настройки лишь доучивается – ее параметры изменяются незначительно.

Таблица 2.3

Вторая часть обучающей выборки (к примеру 2.1)

x_1	x_2	y
–0,9707	–1,0156	–0,8506
–2,3957	1,5092	2,7976
–2,0723	1,1424	0,6094
–0,4011	–3,9513	0,1563
2,5206	–0,0211	–5,4172
2,9372	0,5834	–3,4909
–3,8911	–3,4195	14,4961
2,3062	0,8303	–0,8983
–5,5338	1,5846	16,8988
–5,5749	–1,7864	–10,8077
2,2481	1,1186	0,5979
–0,2027	–3,4439	0,0396
1,5938	–2,8412	1,6358
–0,2027	–3,4439	0,0396
1,5938	–2,8412	1,6358
–5,7498	–2,3377	6,4412
2,6389	–2,7556	4,0120
–1,8163	–0,9672	–3,0431
–5,1735	0,7909	–5,5551
2,6040	–1,1476	–5,6837

Исследуем, как изменится качество настройки при увеличении обучающей выборки. Добавим в обучающую выборку 20 пар «входы – выход» из табл. 2.3. Распределение данных из обучающей выборки показано на рис. 2.2*б*: точками указаны данные из табл. 2.1, а светлыми кружками – новые данные из табл. 2.3. Функции принадлежности нечетких термов и веса правил после 15 итераций обучения показаны на рис. 2.3*в* и в табл. 2.2, соответственно. Увеличение обучающей выборки обеспечило снижение ошибки тестирования до значения 1,99 (см. рис. 2.4*в*).

На рис. 2.5 показаны кривые обучения нечеткой модели. Они отражают зависимости ошибок идентификации (2.2) на обучающей и тестовой выборках от объема (M) самой обучающей выборки. Каждая точка кривых обучения рассчитывалась как среднее значение результатов экспериментов для 10 разных обучающих выборок. Для исследования процесса настройки нечеткой модели использовались обучающие выборки из 20, 30, ..., 110 пар «входы – выход». В этих выборках значения входов генерировались случайно, а выход рассчитывался по формуле эталонной зависимости. Как видно из рис. 2.5, с увеличением объема обучающей выборки уменьшается невязка на тестовой выборке, а также сокращается разница между невязками на обучающей и тестовой выборках.

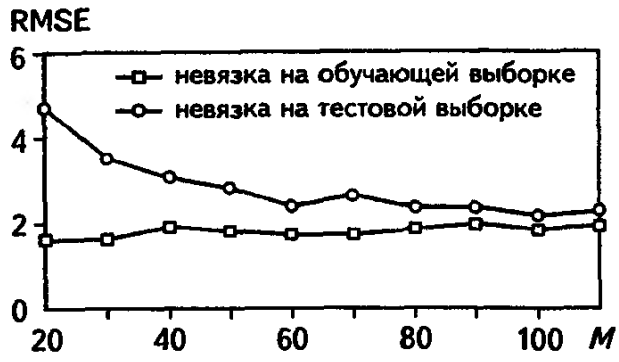


Рис. 2.5. Кривые обучения нечеткой модели Мамдани (к примеру 2.1)

2.1.2. НАСТРОЙКА НЕЧЕТКОЙ БАЗЫ ЗНАНИЙ СУГЕНО

Введем следующие обозначения:

- P – вектор параметров функций принадлежностей термов входных переменных;
- B – вектор коэффициентов линейных функций в заключениях правил базы знаний Сугено.
- $F(P, B, X_r)$ – результат вывода по нечеткой базе знаний Сугено с параметрами (P, B) для входного вектора X_r из выборки (2.1). Нечеткий вывод осуществляется по формулам подраздела 1.6.6.

Параметрическая идентификация нечеткой базы знаний Сугено сводится к следующей задаче математического программирования: *найти такой вектор* (P, B), *чтобы*

$$\sqrt{\frac{1}{M}\sum_{r=\overline{1,M}}(y_r - F(P, B, X_r))^2} \to \min. \qquad (2.3)$$

На практике задачу (2.3) решают как стандартными алгоритмами оптимизации, например, методом Левенберга–Марквардта, так и специально разработанными быстрыми алгоритмами на основе фильтра Калмана и метода обратного распространения ошибки. Алгоритм на основе фильтра Калмана [41] оптимизирует только линейные параметры нечеткой базы знаний – коэффициенты в заключениях правил. ANFIS-алгоритм [30], представляющий собой комбинацию методов наискорейшего спуска и обратного распространения ошибки, оптимизирует как линейные, так и нелинейные параметры нечеткой базы знаний. При этом треугольные нормы должны задаваться дифференцируемыми функциями.

Ниже излагаются три метода идентификации зависимости «входы – выход» с помощью нечеткой базы знаний Сугено. Первый метод использует фильтр Калмана для настройки коэффициентов заключений нечетких правил. Метод предложен японскими учеными Такаги (Takagi) и Сугено (Sugeno) в статье [41], идеи которой послужили основой теории нечеткой идентификации. Второй метод настраивает коэффициенты в заключениях нечетких правил по градиентному алгоритму оптимизации. Третий метод позволяет одновременно настраивать и заключения правил, и функции принадлежности термов входных переменных. Он базируется на идеях обратного распространения ошибки в нейронных сетях. Дальнейшее изложение материала подраздела основывается на работах [41, 43].

Вначале рассмотрим применение фильтра Калмана для настройки нечеткой базы знаний Сугено. Задачу настройки будем рассматривать как нахождение таких коэффициентов в заключениях нечетких правил $\mathbf{B} = (b_{10}, b_{20}, \ldots, b_{m0}, b_{11}, b_{21}, \ldots, b_{m1}, \ldots, b_{1n}, b_{2n}, \ldots, b_{mn})^T$, которые обеспечивают минимум квадратичной невязки:

$$\sum_{r=\overline{1,M}} (y_r - y_r^f)^2 \to \min, \tag{2.4}$$

где y_r^f – результат вывода по нечеткой базе знаний с параметрами $\mathbf{B}$ при значении входов из r-й строчки выборки (X_r).

Входному вектору $X_r = (x_{r,1}, x_{r,2}, \ldots, x_{r,n})$ соответствует такой результат нечеткого вывода:

$$y_r^f = \frac{\sum_{j=\overline{1,m}} \mu_{d_j}(X_r) d_j}{\sum_{j=\overline{1,m}} \mu_{d_j}(X_r)}, \tag{2.5}$$

где $d_j = b_{j0} + \sum_{r=\overline{1,n}} b_{ji} x_{ri}$ – заключение j-го правила; $\mu_{d_j}(X_r)$ – степень выполнения j-го правила (см. подраздел 1.6.6).

Относительную степень выполнения j-го правила для входного вектора X_r обозначим через

$$\beta_{rj} = \frac{\mu_{d_j}(X_r)}{\sum\limits_{k=\overline{1,m}} \mu_{d_k}(X_r)}.$$

Тогда (2.5) можно переписать в виде:

$$y_r^f = \sum_{j=\overline{1,m}} \beta_{rj} d_j = \sum_{j=\overline{1,m}} \beta_{rj} b_{j0} + \beta_{rj} b_{j1} x_{r1} + \beta_{rj} b_{j2} x_{r2} + \ldots + \beta_{rj} b_{jn} x_{rn}.$$

Введем следующие обозначения:

$$\mathbf{Y}^f = (y_1^f, y_2^f, \ldots, y_M^f)^T;$$

$$\mathbf{Y} = (y_1, y_2, \ldots, y_M)^T;$$

$$\mathbf{A} = \begin{bmatrix} \beta_{11}, \ldots, \beta_{1m}, & x_{11}\beta_{11}, \ldots, x_{11}\beta_{1m}, & \ldots, & x_{1n}\beta_{11}, \ldots, x_{1n}\beta_{1m} \\ \beta_{M1}, \ldots, \beta_{Mm}, & x_{M1}\beta_{11}, \ldots, x_{M1}\beta_{1m}, & \ldots, & x_{Mn}\beta_{M1}, \ldots, x_{Mn}\beta_{Mm} \end{bmatrix}.$$

Тогда задача настройки (2.4) в матричной форме ставится следующим образом: *найти вектор* $\mathbf{B}$, *чтобы*

$$E = (\mathbf{Y} - \mathbf{Y}^f)^T (\mathbf{Y} - \mathbf{Y}^f) \to \min, \qquad (2.6)$$

где $\mathbf{Y}^f = \mathbf{AB}$.

Минимальное значение квадратичной невязки E достигается при $\mathbf{Y}^f = \mathbf{Y}$, что соответствует решению уравнения

$$\mathbf{Y} = \mathbf{AB}. \qquad (2.7)$$

Для реальных задач количество настраиваемых параметров меньше объема выборки данных, т.е. $m(n+1) < M$. Следовательно, уравнение (2.7) не имеет точного решения. В этом случае решение можно найти через псевдоинверсию матрицы $\mathbf{A}$:

$$\mathbf{B} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y}. \qquad (2.8)$$

Трудности при решении (2.8) связаны с возможной сингулярностью матрицы $(\mathbf{A}^T\mathbf{A})$. В работе [41] вектор $\mathbf{B}$ находится с помощью фильтра Калмана по следующему итерационному алгоритму:

$$\mathbf{B}^{<k+1>} = \mathbf{B}^k + \mathbf{U}^{<k+1>}\mathbf{A}^{<k+1>}(y_{k+1} - (\mathbf{A}^{<k+1>})^T\mathbf{B}^{<k>}), \quad k = \overline{0,\ M-1},$$

$$\mathbf{U}^{<k+1>} = \mathbf{U}^{<k>} - \frac{\mathbf{U}^{<k>}\mathbf{A}^{<k+1>}(\mathbf{A}^{<k+1>})^T\mathbf{U}^{<k>}}{1 + (\mathbf{A}^{<k+1>})^T\mathbf{U}^{<k>}\mathbf{A}^{<k+1>}}$$

при начальных условиях $\mathbf{B}^{<0>} = 0$ и $\mathbf{U}^{<0>} = u\,\mathbf{I}$, где u – большое положительное число, $\mathbf{I}$ – единичная матрица; <...> – число в угловых скобках указывает номер итерации.

Количество итераций алгоритма идентификации равно объему выборки экспериментальных данных. На первом шаге алгоритма координаты вектора $\mathbf{B}$ настраивают по первой строчке выборки данных. На второй итерации координаты вектора $\mathbf{B}$ подстраивают под вторую пару «входы – выход» и т.д.

Существуют различные алгоритмы нечеткой идентификации, в которых подстройка параметров нечеткой базы знаний Сугено также происходит итерационно по значению «мгновенной ошибки». Ниже рассматривается один такой алгоритм, основанный на градиентном методе оптимизации.

Обозначим «мгновенную ошибку» для r-й строчки выборки данных через $E^{<r>} = (y_r - y_r^f)^2 = e^2$. Тогда, согласно градиентному методу Коши, новые значения управляемых переменных рассчитываются по формуле:

$$b_{ji}^{<r+1>} = b_{ji}^{<r>} - \alpha \frac{\partial E^{<r>}}{\partial b_{ji}}\bigg|_{b_{ji}=b_{ji}^{<r>}}, \quad i = \overline{1,n}, \ j = \overline{1,m}, \tag{2.9}$$

где $\alpha > 0$ – длина шага, которая задает скорость обучения.

При малых значениях параметра α обучение будет медленным. При больших значениях этого параметра возникает опасность переобучения, когда на каждой итерации алгоритма нечеткая модель будет настраиваться только на текущую пару «входы – выход», забывая при этом предыдущий опыт.

Частная производная в (2.9) имеет простое аналитическое выражение:

$$\frac{\partial E^{<r>}}{\partial b_{ji}} = 2e\beta_{rj}x_{ri}.$$

С учетом этого перепишем правило обучения (2.9) в следующем виде:

$$b_{ji}^{<r+1>} = b_{ji}^{<r>} - 2e\beta_{rj}x_{ri}, \quad i = \overline{1,\ n}, \ j = \overline{1,\ m}. \tag{2.10}$$

Идентификацию параметров нечеткой базы знаний с использованием правила (2.10) представим следующим алгоритмом:

Шаг 1. Установить параметры алгоритма: E^* – допустимая квадратичная невязка и z^* – максимальное количество эпох обучения.

Шаг 2. Рассчитать β_{rj} – относительные степени выполнения заключений правил для каждой строчки обучающей выборки, $r = \overline{1, M}$, $j = \overline{1, m}$.

Шаг 3. Установить счетчик итераций обучения и счетчик эпох обучения в единицы: $r = 1$ и $z = 1$.

Шаг 4. Установить начальные значения настраиваемых параметров, например, $b_{ji}^{<r>} = 0, i = \overline{1,n}, \ j = \overline{1,m}$.

Шаг 5. Рассчитать значение «мгновенной ошибки» для r-й пары данных из выборки и пересчитать значения настраиваемых параметров по формуле (2.10).

Шаг 6. Проверить условие «$r < M$?», если «Да», то увеличить счетчик итераций $r = r + 1$ и перейти к шагу 5.

Шаг 7. Рассчитать значение квадратичной невязки на всей выборке данных на z-й эпохе обучения $E^{<z>}$.

Шаг 8. Проверить условие «$E^{<z>} \le E^*$?», если «Да», то перейти к шагу 10.

Шаг 9. Проверить условие «$z < z^*$?», если «Да», то увеличить счетчик эпох $z = z + 1$, установить счетчик итераций обучения в единицу $r = 1$ и перейти к шагу 5.

Шаг 10. Конец.

В приведенном алгоритме используется два критерия останова: первый – по достижению допустимой невязки и второй – по превышению заданного количества эпох обучения. На протяжении одной эпохи осуществляется итерационная подстройка параметров по каждой паре «входы – выход». Таким образом, за одну эпоху обучения шаг 5 алгоритма выполняется ровно M раз. Обучение можно также прекратить, если за одну эпоху настраиваемые параметры практически не изменяются: $\max\limits_{j=\overline{1,m},\, i=\overline{1,n}} \left| b_{ji}^{<r+1>} - b_{ji}^{<r>} \right| < \Delta b^*$ или невязка почти не уменьшается: $E^{<z>} - E^{<z+1>} < \Delta E^*$, где Δb^* и ΔE^* – малые величины.

До сих пор рассматривалась настройка только заключений правил. Если ввести некоторые ограничения на вид нечеткой базы знаний, то по методу обратного распространения ошибки можно получить достаточно простые правила настройки параметров посылок правил. Как это сделать, рассмотрим на примере вывода правил обучения функций принадлежности нечеткой базы знаний Сугено нулевого порядка.

Предположим, что известна следующая нечеткая база знаний Сугено нулевого порядка:

$$\text{Если } x_1 = \tilde{a}_{1j} \text{ и } x_2 = \tilde{a}_{2j}, \ldots, x_n = \tilde{a}_{nj}, \quad \text{то } y = b_j,\ j = \overline{1, m},$$

где b_j – некоторые действительные числа, $j = \overline{1, m}$.

Потребуем, чтобы каждый терм входной переменной использовался только в одном правиле. Чтобы получить дифференцируемые соотношения «входы – выход», будем использовать гауссовы функции принадлежности:

$$\mu_j(x_i) = \exp\left(-\frac{1}{2} \left(\frac{x_i - c_{ij}}{\sigma_{ij}} \right)^2 \right),$$

где $\mu_j(x_i)$ – функция принадлежности нечеткого множества $\tilde{a}_{ij}$; c_{ij} и σ_{ij} – параметры функции принадлежности – координата максимума и коэффициент концентрации.

Входному вектору $X_r = (x_{r1}, x_{r2}, \ldots, x_{rn})$ соответствует такой результат нечеткого вывода:

$$y_r^f = \beta_{rj} \sum_{j=\overline{1,m}} (b_{j0} + b_{j1} x_{r1} + b_{j2} x_{r2} + \ldots + b_{jn} x_{rn}),$$

где β_{rj} – относительная степень выполнения j-го правила для входного вектора X_r.

При использовании в качестве t-нормы умножения относительная степень выполнения j-го правила рассчитывается так:

$$\beta_{rj} = \frac{\mu_j(x_{r1})\mu_j(x_{r2})\ldots\mu_j(x_{rn})}{\sum\limits_{k=\overline{1,m}}\mu_k(x_{r1})\mu_k(x_{r2})\ldots\mu_k(x_{rn})} = \frac{\exp\left(-\frac{1}{2}\sum\limits_{i=\overline{1,n}}\left(\frac{x_{ri}-c_{ij}}{\sigma_{ij}}\right)^2\right)}{\sum\limits_{k=\overline{1,m}}\exp\left(-\frac{1}{2}\sum\limits_{i=\overline{1,n}}\left(\frac{x_{ri}-c_{ik}}{\sigma_{ik}}\right)^2\right)}.$$

Настраиваемыми параметрами при идентификации будут коэффициенты в заключениях правил $\mathbf{B} = (b_1, b_2, \ldots, b_m)$ и параметры функций принадлежности нечетких термов: координаты максимумов $\mathbf{C} = (c_{11}, c_{21}, \ldots, c_{n1}, c_{21}, c_{22}, \ldots, c_{n2}, \ldots, c_{1m}, c_{2m}, \ldots, c_{nm})$ и коэффициенты концентраций $\mathbf{S} = (\sigma_{11}, \sigma_{21}, \ldots, \sigma_{n1}, \sigma_{21}, \sigma_{22}, \ldots, \sigma_{n2}, \ldots, \sigma_{1m}, \sigma_{2m}, \ldots, \sigma_{nm})$. Задача настройки состоит в нахождении вектора (**B**, **C**, **S**), обеспечивающего выполнение (2.6).

Находя частные производные «мгновенной ошибки» $E^{<r>}$ по управляемым переменным b_j, c_{ij} и σ_{ij}, получаем такие правила обучения:

$$b_j^{<r+1>} = b_j^{<r>} - \alpha\frac{\partial E^{<r>}}{\partial b_j} = b_j^{<r>} - 2\alpha e\beta_{rj}, \tag{2.11}$$

$$c_i^{<r+1>} = c_{ij}^{<r>} - \alpha\frac{\partial E^{<r>}}{\partial c_{ij}} = c_{ij}^{<r>} - \alpha e\beta_{rj}(b_j^{<r>} - y_r)\frac{x_{ri} - c_i^{<r>}}{(\sigma_{ij}^{<r>})^2}, \tag{2.12}$$

$$\sigma_{ij}^{<r+1>} = \sigma_{ij}^{<r>} - \alpha\frac{\partial E^{<r>}}{\partial \sigma_{ij}} = \sigma_{ij}^{<r>} - \alpha e\beta_{rj}(b_j^{<r>} - y_r)\frac{(x_{ri} - c_{ij}^{<r>})^2}{(\sigma_{ij}^{<r>})^3}, \tag{2.13}$$

где $i = \overline{1, n}$, $j = \overline{1, m}$, $r = \overline{1, M}$.

Для каждой пары данных «входы – выход» обучение происходит в два этапа. На первом этапе осуществляется нечеткий вывод для текущего входного вектора. На втором этапе вычисляется «мгновенная ошибка» – разница между полученным и ожидаемым значениями выхода, и модифицируются параметры нечеткой базы знаний. При представлении нечеткой модели в виде нейро-нечеткой сети первый этап обучения можно сопоставить с прямым распространением сигнала по сети, а второй этап – с обратным распространением ошибки.

В правила обучения (2.11)–(2.13) входят одни и те же множители, что позволяет быстро пересчитывать значения настраиваемых параметров. При функциях принадлежности других типов правила обучения получаются не такими простыми. Обучение функций принадлежностей по формулам (2.11)–(2.13) по сравнению с настройкой заключений правил по формуле (2.10) является более трудоемкой вычислительной процедурой в связи с пересчетом относительных степеней выполнения правил после каждой итерации алгоритма обучения.

Пример 2.2. Идентифицировать нелинейную зависимость из примера 2.1 нечеткой базой знаний Сугено. Сравнить результаты идентификации нечеткими базами знаний Мамдани и Сугено.

Для лингвистической оценки входных переменных будем использовать по два нечетких терма с гауссовой функцией принадлежности. Графики исходных функций принадлежности изображены на рис. 2.6*а*. Исходная нечеткая база знаний из четырех правил приведена в табл. 2.4. По графику идентифицируемой зависимости (рис. 2.1) трудно оценить коэффициенты в заключениях правил, поэтому в исходной базе знаний они приняты равными нулю. При любых значениях входов нечеткая модель выдает нуль. Поверхность «входы – выход» исходной нечеткой модели приведена на рис.2.7*а*.

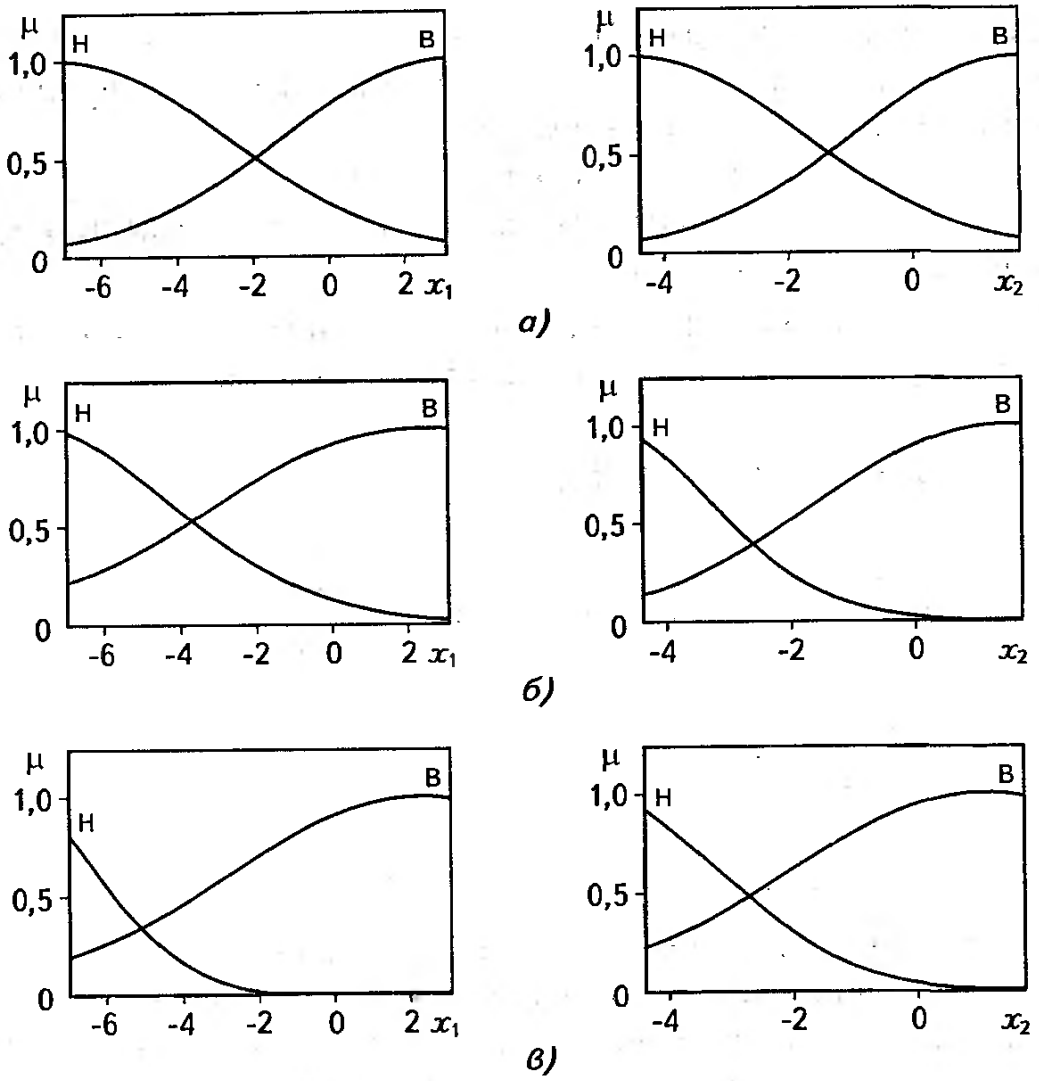


Рис. 2.6. Функции принадлежности нечеткой модели Сугено (к примеру 2.2)

Н – низкий, В – высокий

а – до обучения; *б* – после обучения на выборке из 20 точек; *в* – после обучения на выборке из 40 точек

Будем настраивать 20 параметров нечеткой базы знаний: по три коэффициента в заключениях каждого из четырех нечетких правил и по два параметра функций принадлежности для каждого из четырех термов входных переменных. После 100 итераций ANFIS-алгоритма ошибка тестирования уменьшилась с 13,2 до 5,69. На рис. 2.7*б* показана поверхность «входы – выход», соответствующая нечеткой базе знаний Сугено, настроенной на выборке из 20 точек. Как видно из рисунка, настроенная нечеткая модель отражает лишь основные особенности идентифицируемой зависимости. Низкое качество идентификации связано с малым объе-

Таблица 2.4

Нечеткая база знаний Сугено (к примеру 2.2)

x_1	x_2	y		
		исходная	после настройки на выборке из 20 точек	после настройки на выборке из 40 точек
Низкий	Низкий	$0+0x_1+0x_2$	$460{,}8+1{,}43x_1+85{,}15x_2$	$642{,}3+8{,}85x_1+103{,}79x_2$
Низкий	Высокий	$0+0x_1+0x_2$	$-47{,}51-0{,}33x_1+40{,}22x_2$	$-82{,}62-2{,}2x_1+58{,}8x_2$
Высокий	Низкий	$0+0x_1+0x_2$	$-93{,}11+3{,}89x_1-20{,}17x_2$	$25{,}81+2{,}13x_1+4{,}62x_2$
Высокий	Высокий	$0+0x_1+0x_2$	$4{,}97-0{,}9x_1+3{,}21x_2$	$-3{,}61-0{,}73x_1+3{,}25x_2$

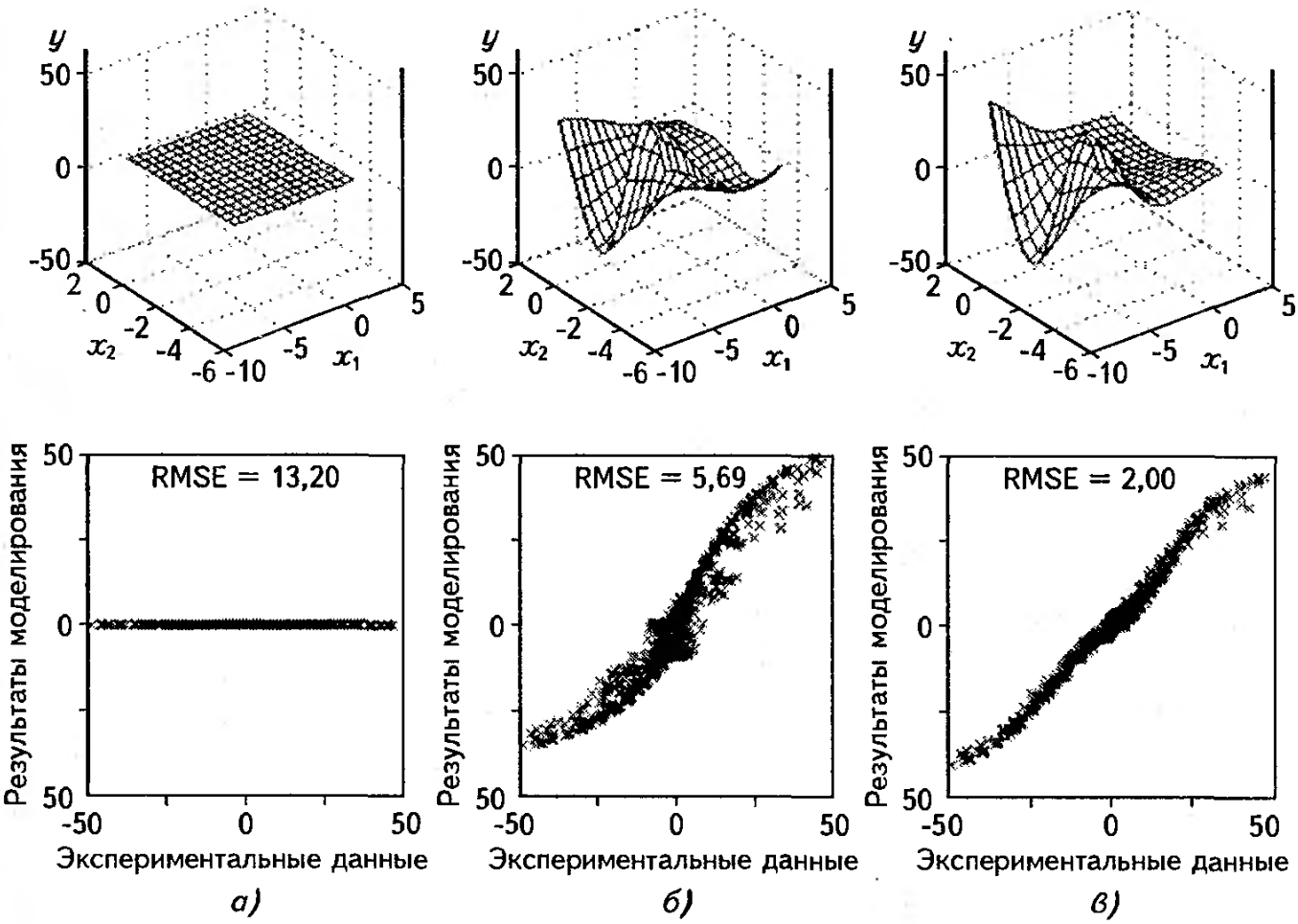


Рис. 2.7. Тестирование нечетких моделей Сугено (к примеру 2.2)

мом обучающей выборки – количество пар «входы – выход» равно числу настраиваемых параметров. Качество идентификации можно существенно улучшить, увеличивая обучающую выборку. Результаты настройки нечеткой базой знаний Сугено по обучающей выборке из 40 пар «входы – выход» приведены на рис. 2.7*в*.

На рис. 2.8 показаны кривые обучения нечеткой модели Сугено. Каждая точка кривых обучения рассчитывалась как среднее значение результатов экспериментов для 10 разных обучающих выборок. Обучение проводилось на протяжении 100 итераций ANFIS-алгоритма. Как видно из рис. 2.8, объем обучающей выборки существенно влияет на качество идентификации нечеткой моделью Сугено.

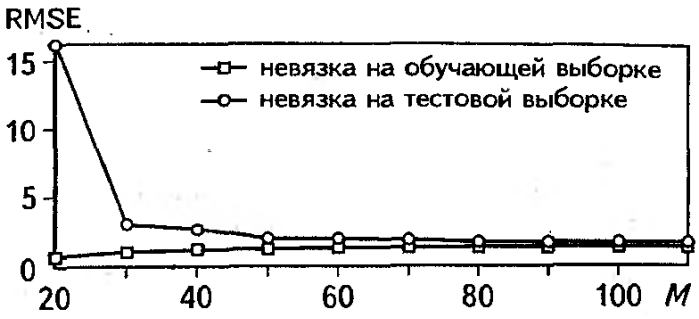


Рис. 2.8. Кривые обучения нечеткой модели Сугено (к примеру 2.2)

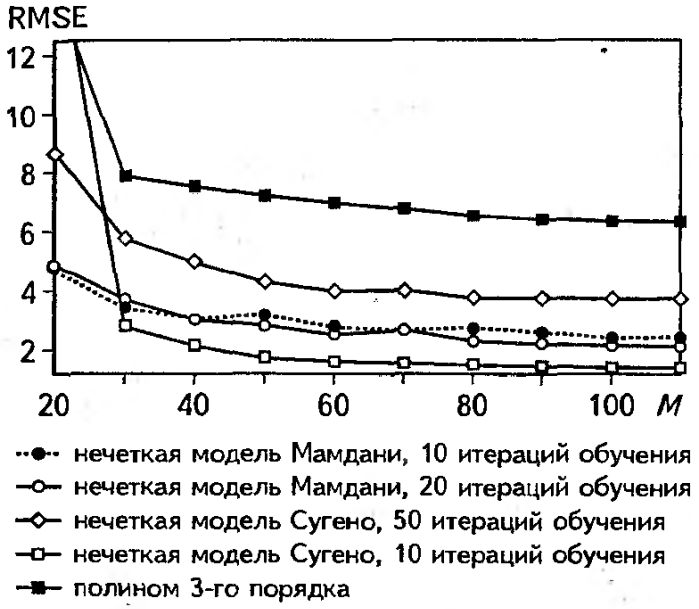


Рис. 2.9. Кривые обучения для нечетких баз знаний Мамдани и Сугено (к примеру 2.2)

Для сравнения результатов идентификации нечеткими базами знаний Мамдани и Сугено на рис. 2.9 приведены кривые обучения при разной продолжительности обучения. Для примера приведена также кривая обучения при аппроксимации полиномом третьей степени. При малых обучающих выборках качество идентификации существенно выше для нечеткой базы знаний Мамдани. Это объясняется тем, что исходная, основанная на экспертных высказываниях, нечеткая модель Мамдани уже отражает основные особенности идентифицируемой зависимости. При малых обучающих выборках настройка не должна быть продолжительной, чтобы избежать переобучения. С увеличением обучающей выборки лучшее качество идентификации обеспечивает база знаний Сугено. Хорошие результаты получаются, когда объем обучающей выборки в 2–3 раза превышает количество настраиваемых параметров. При больших обучающих выборках наступает эффект насыщения – точность идентификации практически не улучшается с увеличением числа наблюдений.

Обратим внимание, что после настройки база знаний Мамдани остается прозрачной, ее параметры – функции принадлежности легко интерпретируются лингвистическими термами. Для баз знаний Сугено типовое явление – сложность содержательной интерпретации ее параметров. В качестве примера укажем на заключения правил в табл. 2.4, а также на функцию принадлежности терма «низкий» переменной x_1 на рис. 2.7*в*. Поэтому для задач, где более важна точность идентификации, целесообразным будет использование нечетких баз знаний Сугено, а для задач, где более важным является объяснение и обоснование принятого решения, нечеткие базы знаний Мамдани будут иметь преимущество.

2.1.3. НАСТРОЙКА НЕЧЕТКОЙ БАЗЫ ЗНАНИЙ ДЛЯ ЗАДАЧ КЛАССИФИКАЦИИ

Настройка представляет собой нахождение таких параметров функций принадлежностей термов входных переменных и весовых коэффициентов правил, которые минимизируют отклонение между желаемым и действительным поведением нечеткого классификатора на обучающей выборке. Критерий близости можно определить различными способами.

Первый способ заключается в выборе в качестве критерия настройки процента ошибок классификации на обучающей выборке. Введем следующие обозначения:

- P – вектор параметров функций принадлежности термов входных и выходной переменных;
- W – вектор весовых коэффициентов правил базы знаний;
- $F(X_r, P, W)$ – результат вывода по нечеткой базе с параметрами (P, W) при значении входов X_r. Нечеткий логический вывод для задач классификации описан в подразделе 1.6.8.

Настройка нечеткого классификатора сводится к следующей задаче оптимизации: *найти такой вектор (P, W), чтобы*

$$\frac{100\%}{M}\sum_{r=\overline{1,M}}\Delta_r \to \min, \tag{2.14}$$

где Δ_r – ошибка классификации объекта X_r:

$$\Delta_r = \begin{cases} 1, & \text{если } y_r \neq F(X_r, P, W), \\ 0, & \text{если } y_r = F(X_r, P, W). \end{cases}$$

Преимущества критерия настройки (2.14) заключаются в его простоте и ясной содержательной интерпретации. Процент ошибок широко используется как критерий обучения различных систем распознавания образов. Целевая функция задачи оптимизации (2.14) принимает дискретные значения. Это затрудняет использование градиентных методов оптимизации, так как на протяженных плато целевой функции алгоритмы оптимизации «застревают». Особенно трудно подобрать подходящие параметры градиентных алгоритмов (например, приращения аргументов для расчета частных производных) при небольшой выборке данных.

Второй способ использует в качестве критерия настройки расстояние между результатом вывода в виде нечеткого множества

$$\left(\frac{\mu_{d_1}(X)}{d_1}, \frac{\mu_{d_2}(X)}{d_2}, \ldots, \frac{\mu_{d_m}(X)}{d_m}\right)$$

и значением выходной переменной в обучающей выборке. Для этого выходную переменную y в обучающей выборке (2.1) фаззифицируют следующим образом [12, 37]:

$$\left.\begin{array}{ll}\tilde{y}=(1/d_1,\ 0/d_2,\ldots,\ 0/d_m), & \text{если } y=d_1\\ \tilde{y}=(0/d_1,\ 1/d_2,\ldots,\ 0/d_m), & \text{если } y=d_2\\ \ldots & \\ \tilde{y}=(0/d_1,\ 0/d_2,\ldots,\ 1/d_m), & \text{если } y=d_m\end{array}\right\}. \qquad (2.15)$$

В этом случае настройка нечеткого классификатора сводится к следующей задаче оптимизации [12, 37]: *найти такой вектор (P, W), чтобы*

$$\sqrt{\frac{1}{M}\sum_{r=1}^{M}\sum_{j=1}^{m}(\mu_{d_j}(y_r)-\mu_{d_j}(X_r,P,W))^2}\to\min, \qquad (2.16)$$

где $\mu_{d_j}(y_r)$ – степень принадлежности значения выходной переменной y в r-й паре обучающей выборке к решению d_j в соответствии с (2.15); $\mu_{d_j}(X_r,P,W)$ – степень принадлежности выхода нечеткой модели с параметрами (P, W) к решению d_j, определяемая по формуле (1.8) при значениях входов из r-й пары обучающей выборки (X_r).

Целевая функция в задаче (2.16) не имеет протяженных плато, поэтому она пригодна к оптимизации градиентными методами. Однако результаты оптимизации иногда неудовлетворительные: нечеткая база знаний, минимизирующая (2.16), не всегда обеспечивает также и минимум ошибок классификации. Это объясняется тем, что точки, близкие к границам раздела классов, вносят почти одинаковый вклад в критерий настройки как при правильной, так и при ошибочной классификации.

Третий способ наследует достоинства предыдущих способов. Идея заключается в том, чтобы вклад ошибочно классифицированных объектов в критерий настройки увеличивать посредством умножения расстояния

$$\sqrt{\sum_{j=1}^{m}(\mu_{d_j}(y_r)-\mu_{d_j}(X_r,P,W))^2}$$

на штрафной коэффициент. В результате задача оптимизации принимает следующий вид [22]:

$$\sqrt{\frac{1}{M}\sum_{r=1}^{M}(\Delta_r\cdot\text{penalty}+1)\sum_{j=1}^{m}(\mu_{d_j}(y_r)-\mu_{d_j}(X_r,P,W))^2}\to\min, \qquad (2.17)$$

где $\text{penalty}>0$ – штрафной коэффициент.

Ограничения на управляемые переменные в сформулированных задачах оптимизации описаны в подразделе 2.1.1. Задачи (2.14), (2.16) и (2.17) могут быть решены различными технологиями оптимизации, среди которых часто применяют метод наискорейшего спуска, квазиньютоновские методы и генетические алгоритмы.

Параметры функций принадлежности и веса правил можно настраивать одновременно или по отдельности. При настройке только весов правил объем вычислений можно значительно сократить, так как входящие в (1.8) степени принад-

лежности $\mu_{jp}(x_i^*)$ не зависят от W. Для этого в начале оптимизации надо рассчитать степени выполнения правил при единичных весовых коэффициентах ($w_{jp} = 1$) для каждого объекта из обучающей выборки:

$$g_{jp}(X_r) = \min_{i=\overline{1,n}} \mu_{jp}(x_{ri}), \quad j = \overline{1,m}, \ p = \overline{1,k_j}, \ r = \overline{1,M}.$$

Для новых весовых коэффициентов степени принадлежности объекта X_r классам d_j пересчитываются так [22]:

$$\mu_{d_j}(X_r) = \max_{p=\overline{1,k_j}} w_{jp} g_{jp}(X_r), \quad j = \overline{1,m}. \tag{2.18}$$

Пример 2.3. Рассматривается объект с двумя входами $x_1, x_2 \in [0,10]$ и одним выходом y, который может принимать одно из трех дискретных значений $\{d_1, d_2, d_3\}$ в соответствии с решающими правилами:

$$y = \begin{cases} d_1, \text{ если } x_2 < \dfrac{14{,}6}{2{,}25 + (x_1 - 6{,}5)^2}, \\ d_2, \text{ если } \dfrac{14{,}6}{2{,}25 + (x_1 - 6{,}5)^2} < x_2 < 2{,}2\sqrt{x_1} + 3, \\ d_3, \text{ если } x_2 > 2{,}2\sqrt{x_1} + 3. \end{cases} \tag{2.19}$$

Разделяющие кривые, а также обучающая и тестовая выборки показаны на рис. 2.10. Выборки содержат 80 и 5000 пар «входы – выход» соответственно. В выборках значения входов выбирались случайно, а значения выхода рассчитывались по формуле (2.19).

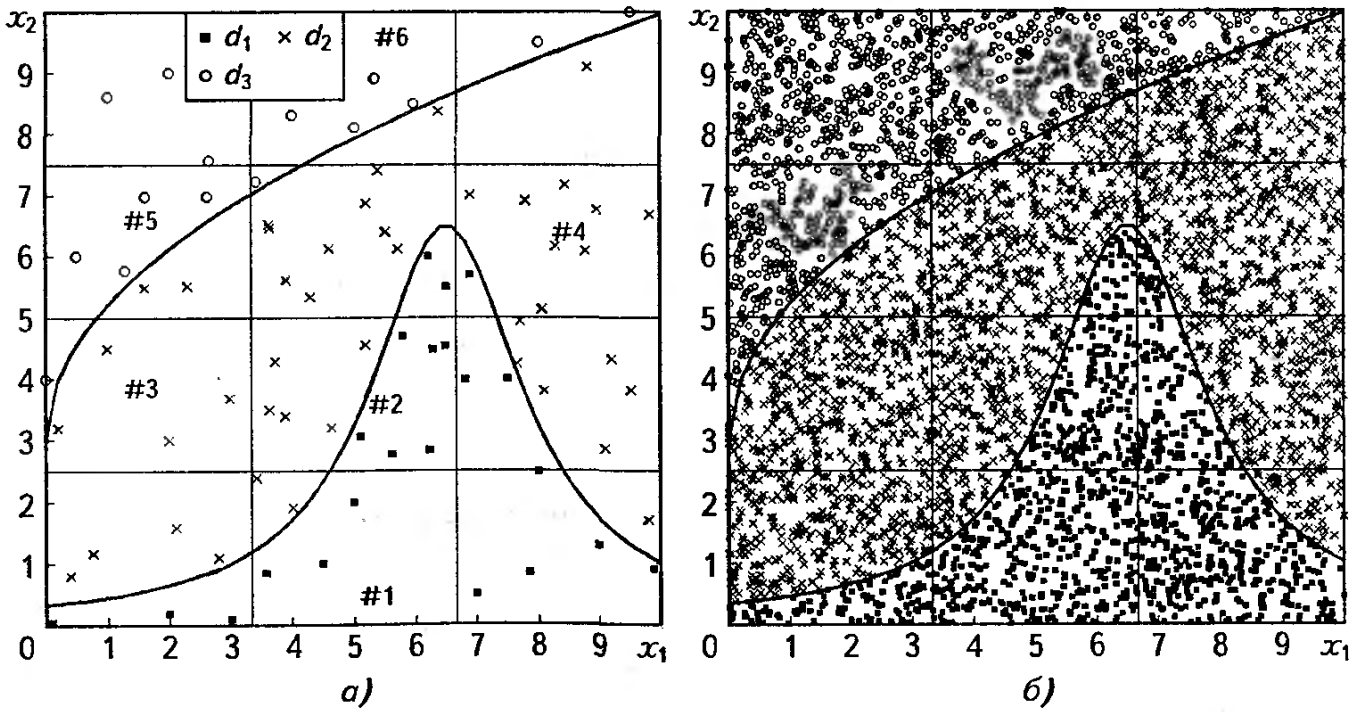


Рис. 2.10. Обучающая (*а*) и тестовая (*б*) выборки (к примеру 2.3)

На основе этих выборок спроектируем нечеткий классификатор. Входы нечеткого классификатора будем рассматривать как лингвистические переменные, значения которых определяются из следующих терм-множеств: {«низкий», «средний», «высокий»} для

x_1 и {«низкий», «ниже среднего», «выше среднего», «высокий»} для x_2. Формализацию термов осуществим симметричной гауссовой функцией принадлежности (рис. 2.11*а*). До настройки коэффициенты концентраций всех функций принадлежности равны 2.

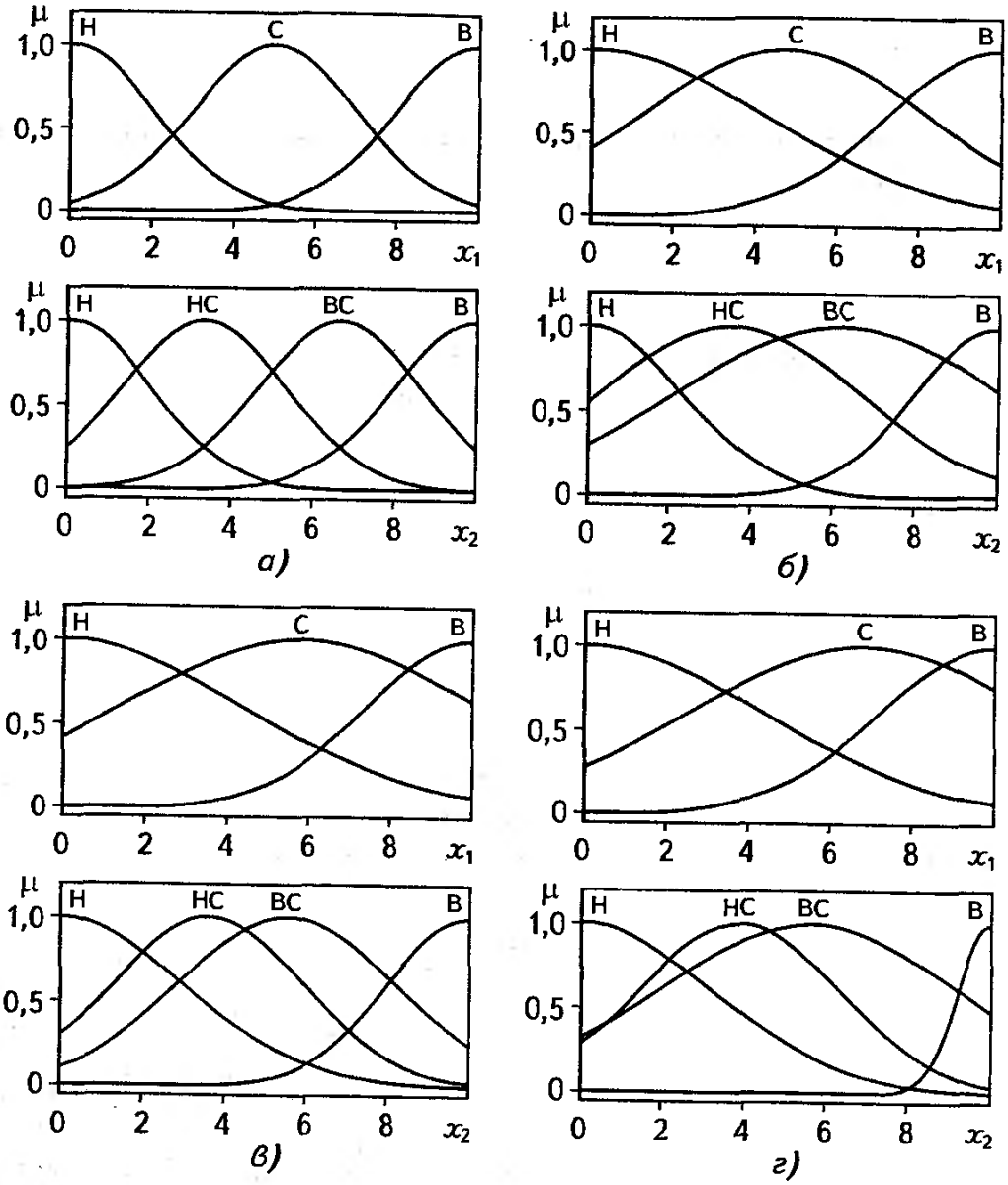


Рис. 2.11. Функции принадлежности (к примеру 2.3)

Н – низкий, НС – ниже среднего, С – средний, ВС – выше среднего, В – высокий
а – исходный классификатор; *б* – классификатор I; *в* – классификатор II; *г* – классификатор III

Таблица 2.5

Нечеткая база знаний (к примеру 2.3)

x_1	x_2	y	Веса правил, W			
			До настройки	Классификатор I	Классификатор II	Классификатор III
Средний	Низкий	d_1	1	0,767	0,751	0,779
Средний	Ниже среднего	d_1	1	0,513	0,395	0,539
Низкий	Ниже среднего	d_2	1	1,000	1,000	1,000
Высокий	Выше среднего	d_2	1	0,570	1,000	0,783
Низкий	Выше среднего	d_3	1	0,816	0,490	0,697
Средний	Высокий	d_3	1	0,020	0,016	1,000

Координаты максимумов выбраны так, чтобы разбить интервал [0, 10] на три (для x_1) и на четыре (для x_2) равные части. По рис. 2.10 эксперт сгенерировал шесть нечетких правил классификации, которые сведены в табл. 2.5. Области действия правил обозначены на рис. 2.10 символами #1, #2, ..., #6.

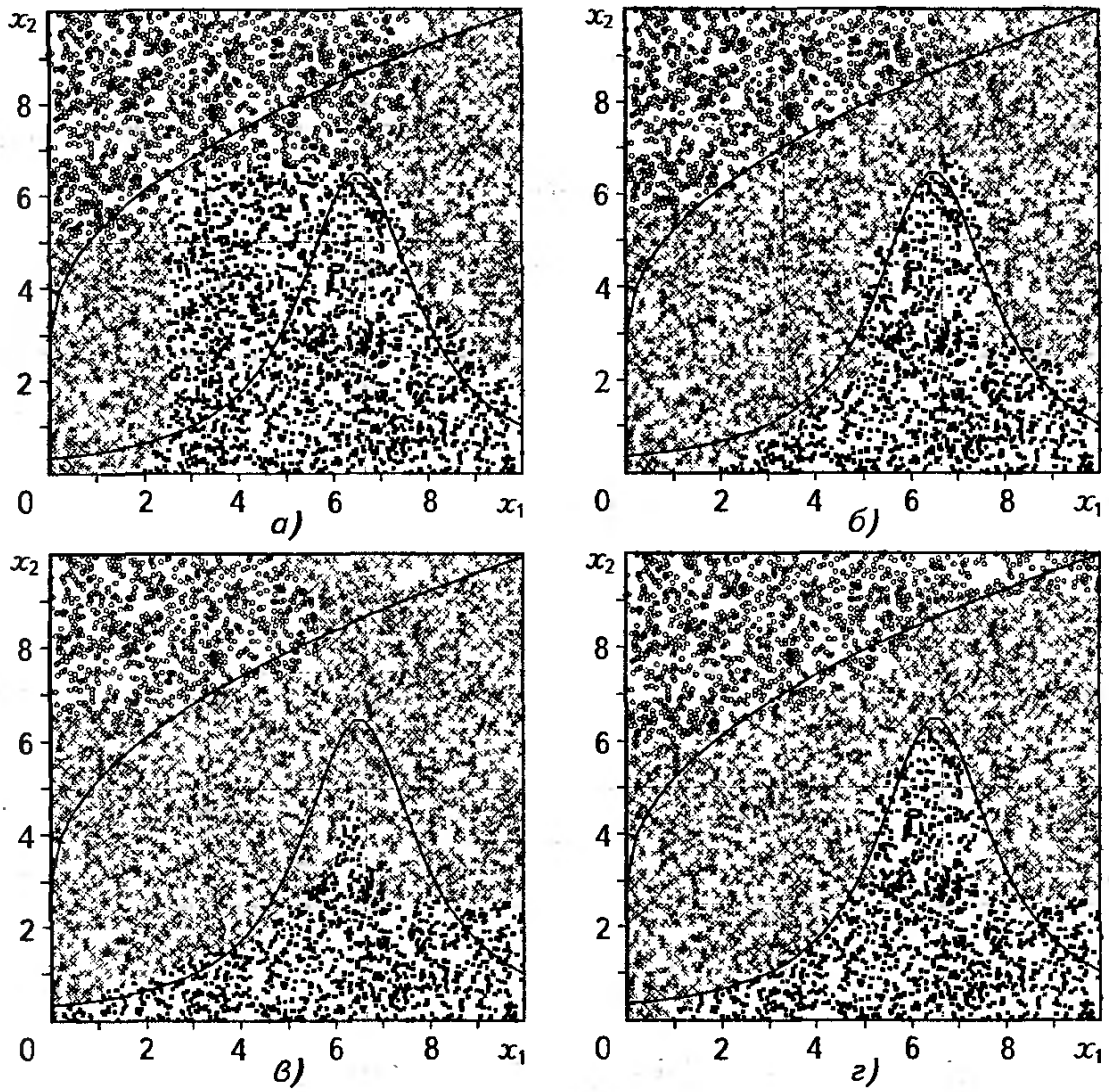


Рис. 2.12. Классификация на тестовой выборке (к примеру 2.3)

а – исходный классификатор; *б* – классификатор I; *в* – классификатор II; *г* – классификатор III

Исходная нечеткая база знаний с шестью правилами грубо отражает нелинейные разделяющие кривые – на тестовой выборке ошибочно классифицировано 26,6% объектов (рис. 2.12*а*). После настройки весов правил количество ошибок уменьшилось до 15%, однако безошибочность нечеткого классификатора остается на уровне следующего простого дерева решений:

```
Если ((x1>1.2929) & (x2≤1)),                                  то y = d1,
Если ((x1>4.6335) & (x2>1) & (x1≤7.5) & (x2≤6)),               то y = d1,
Если ((x2>8.3607) & (x1>5.3301)),                             то y = d3,
Если ((x1≤5.3301) & (x2>6.9107)),                             то y = d3,
Если ((x1≤1.2929) & (x2>3.4988) & (x2≤6.9107)),               то y = d3,
Иначе,                                                          y = d2.
```

Низкая безошибочность нечеткого классификатора после настройки весов правил объясняется «плохими» функциями принадлежности. Поэтому необходимо модифициро-

вать не только веса правил, но и функции принадлежности. Будем настраивать следующие 16 параметров нечеткого классификатора:

- три координаты максимумов функций принадлежностей термов «средний», «ниже среднего» и «выше среднего»;
- семь коэффициентов концентраций функций принадлежностей термов входных переменных;
- шесть весовых коэффициентов правил базы знаний.

Таблица 2.6

Результаты тестирования классификаторов (к примеру 2.3)

Классификатор	Критерий I, %	Критерий II	Критерий III (penalty = 9)	Безошибочность на тестовой выборке, %
Исходный	33,75	0,658	1,997	26,60
Классификатор I	6,25	0,727	1,013	9,78
Классификатор II	18,75	0,646	1,221	16,42
Классификатор III	8,75	0,676	1,018	9,22
Дерево решений	7,50	–	–	15,24

Результаты настройки с использованием различных критериев приведены на рис. 2.11 и в табл. 2.5. Использовались такие критерии настройки: критерий I – формула (2.14); критерий II – формула (2.16); критерий III – формула (2.17). Результаты тестирования классификаторов сведены в табл. 2.6. Классификатор I настроен по критерию I, классификатор II – по критерию II и классификатор III – по критерию III. Критерии I и III обеспечивают лучшую настройку нечеткого классификатора, однако следует помнить, что подбор подходящих параметров градиентных алгоритмов при оптимизации по критерию I часто отнимает много времени.

2.2. НЕЧЕТКАЯ КЛАСТЕРИЗАЦИЯ

Кластеризация – это объединение объектов в группы (кластеры) на основе схожести признаков для объектов одной группы и отличий между объектами из разных групп, что соответствует обучению без учителя. Большинство алгоритмов кластеризации не опирается на традиционные для статистических методов допущения – они могут использоваться в условиях почти полного отсутствия информации о законах распределения данных. Кластеризацию проводят для объектов с количественными (числовыми), качественными или смешанными признаками. В разделе рассматриваются методы кластеризация для объектов с количественными признаками: алгоритмы четких и нечетких *c*-средних и алгоритм горной кластеризации. В конце раздела показывается, как использовать результаты кластеризации для синтеза нечетких баз знаний. Материал раздела базируется на литературных источниках [24, 26, 43, 46].

2.2.1. ВВЕДЕНИЕ В КЛАСТЕРИЗАЦИЮ

Исходной информацией для кластеризации является матрица наблюдений:

$$\mathbf{X}=\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & & & \\ x_{M1} & x_{M2} & \dots & x_{Mn} \end{bmatrix},$$

каждая строчка которой представляет собой значения n признаков одного из M объектов кластеризации. Кластеризация состоит в разбиении объектов из **X** на несколько подмножеств (кластеров), в которых объекты между собой более схожи, чем с объектами из других кластеров. В метрическом пространстве «схожесть» обычно определяют через расстояние. Расстояние может рассчитываться как между исходными объектами – строчками матрицы **X**, так и от этих объектов к прототипам центров кластеров. Зачастую координаты прототипов заранее неизвестны, их находят одновременно с разбиением данных на кластеры.

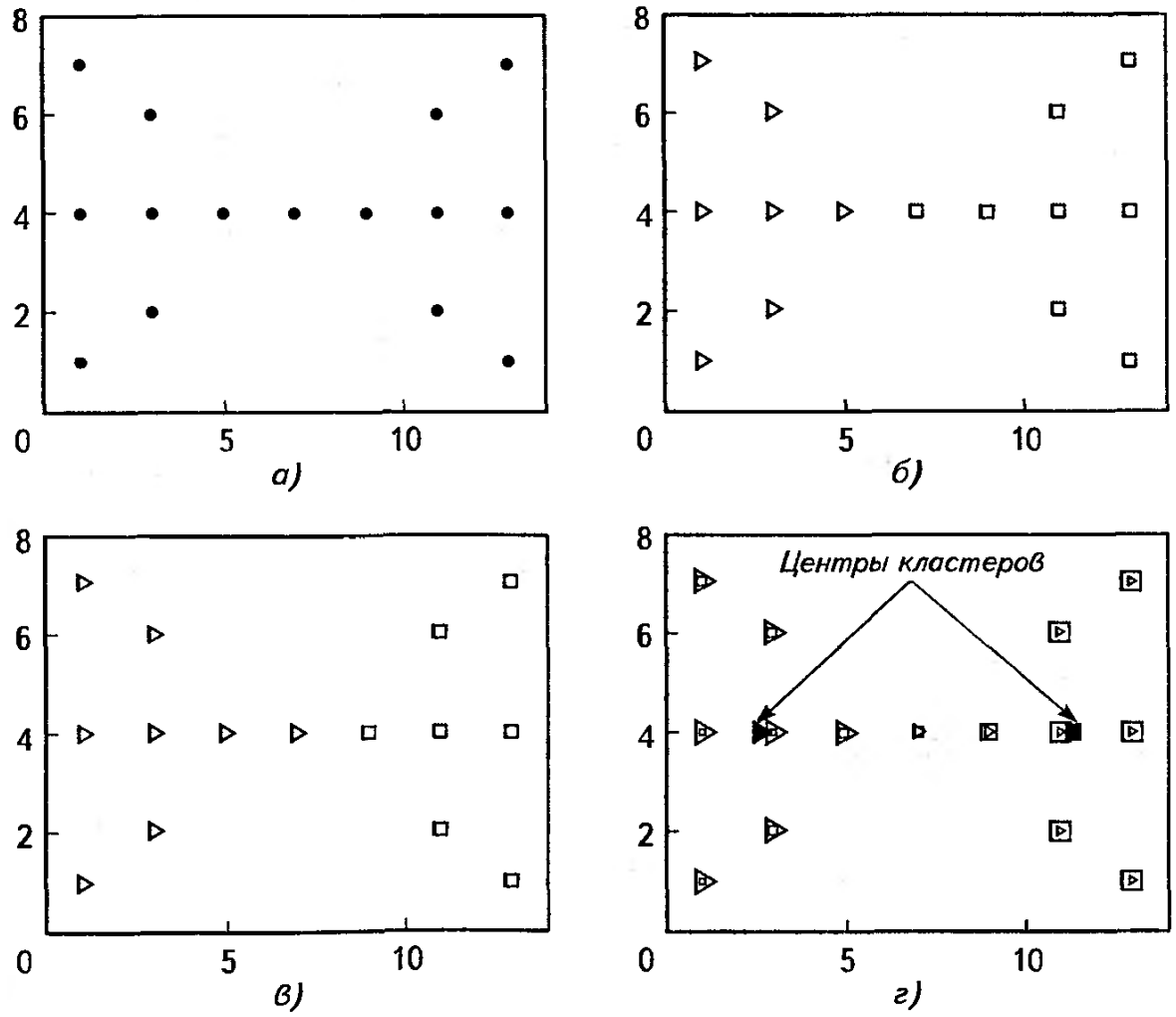


Рис. 2.13. Сравнение четкой и нечеткой кластеризации «бабочки» (к примеру 2.4)

а – исходные данные; *б* – четкая кластеризация I; *в* – четкая кластеризация II; *г* – нечеткая кластеризация

Существует множество методов кластеризации, которые можно классифицировать на четкие и нечеткие. Четкие методы кластеризации разбивают исходное множество объектов **X** на несколько непересекающихся подмножеств. При этом любой объект из **X** принадлежит только одному кластеру. Нечеткие методы кластеризации позволяют одному и тому же объекту одновременно принадлежать нескольким, или даже всем кластерам, но с разными степенями. Нечеткая кластеризация во многих ситуациях более «естественна», чем четкая, например, для объектов, расположенных на границе кластеров. Проиллюстрируем этот тезис на «бабочке» – хорошо известном в теории кластеризации примере.

Пример 2.4. «Бабочка» представляет собой 15 объектов, двумерное изображение которых напоминает одноименное насекомое (рис. 2.13*а*). При четкой кластеризации (рис. 2.13*б* и *в*) получаются два кластера из семи и восьми объектов. На рисунке объекты первого кластера обозначены треугольниками, а второго – квадратами. Симметричная «бабочка» при четкой кластеризации разбивается на два несимметричных кластера. При нечеткой кластеризации (рис. 2.13*г*) проблемный восьмой объект, расположенный в центре «бабочки», одновременно принадлежит двум симметричным кластерам с одной и той же степенью. На этом рисунке размер маркеров пропорционален степени принадлежности объекта кластеру.

Методы кластеризации также классифицируются по тому, определено ли количество кластеров заранее или нет. В последнем случае количество кластеров находят по распределению исходных данных в ходе выполнения алгоритма.

Дальнейшее изложение материала организовано следующим образом: в начале рассматриваются алгоритмы *c*-средних, разбивающие данные на наперед заданное число кластеров, затем описывается алгоритм горной кластеризации, который не требует задания количества кластеров, и в заключении показывается, как использовать результаты кластеризации для экстракции нечетких правил из данных.

2.2.2. КЛАСТЕРИЗАЦИЯ АЛГОРИТМАМИ С-СРЕДНИХ

В начале рассматриваются ключевые идеи четкой кластеризации алгоритмом *c*-средних, затем базовый нечеткий алгоритм *c*-средних и в заключении основные пути улучшения нечеткой кластеризации.

2.2.2.1. Четкая кластеризация алгоритмом *c*-средних

При кластеризации алгоритмом *c*-средних множество **X** разбивается на подмножества A_i, $i=\overline{1, c}$ со следующими свойствами:

$$\bigcup_{i=\overline{1,c}} A_i = \mathbf{X}; \tag{2.20}$$

$$A_i \cap A_j = \varnothing, \quad i, j = \overline{1, c}, \quad i \neq j; \tag{2.21}$$

$$\varnothing \subset A_i \subset \mathbf{X}, \quad i = \overline{1, c}. \tag{2.22}$$

Условие (2.20) указывает, что все объекты должны быть распределены по кластерам. При этом каждый объект должен принадлежать только одному кластеру (условие (2.21)) и ни один из кластеров не может быть пустым или содержать все объекты (условие (2.22)). Количество кластеров $c \in \{2, 3, \ldots, M-1\}$ задается до начала работы алгоритма.

Задачу кластеризации удобно формулировать, используя характеристическую функцию. Характеристическая функция принимает значение 0, если элемент не принадлежит кластеру, и 1, если элемент принадлежит кластеру. С использованием характеристической функции кластеры описываются следующей матрицей разбиения:

$$\mathbf{U} = [\varphi_{ki}], \quad \varphi_{ki} \in \{0, 1\}, \quad k = \overline{1, M}, \quad i = \overline{1, c},$$

где k-я строчка матрицы **U** указывает на принадлежность объекта $X_k = (x_{k1}, x_{k2}, \ldots, x_{kn})$ кластерам $A_1, A_2, \ldots, A_c$.

Матрица **U** должна обладать следующими свойствами:

$$\sum_{i=\overline{1,c}} \varphi_{ki} = 1, \quad k = \overline{1, M}; \tag{2.23}$$

$$0 < \sum_{k=\overline{1,M}} \varphi_{ki} < M, \quad i = \overline{1, c}. \tag{2.24}$$

Для оценки качества разбиения используется критерий разброса, показывающий сумму расстояний от объектов до центра своего кластера. Для евклидового пространства этот критерий вида [3]:

$$\sum_{i=\overline{1,c}} \sum_{X_k \in A_i} \left\| V_i - X_k \right\|^2, \tag{2.25}$$

где $A_i = \{X_p: \ \varphi_{pi} = 1, \ p = \overline{1, M}\}$ – i-й кластер; $V_i = \frac{1}{|A_i|} \sum_{X_k \in A_i} X_k$ – центр i-го кластера; $|A_i|$ – количество объектов кластера A_i.

Кластеризацию объектов **X** можно сформулировать как следующую задачу оптимизации: найти матрицу **U**, минимизирующую значение критерия (2.25). Дискретный характер четкого разбиения обуславливает негладкость целевой функции, что усложняет нахождение оптимальной кластеризации.

2.2.2.2. Базовый алгоритм нечетких *c*-средних

Нечеткие кластеры опишем следующей матрицей нечеткого разбиения:

$$\mathbf{F} = [\mu_{ki}], \quad \mu_{ki} \in [0, 1], \quad k = \overline{1, M}, \quad i = \overline{1, c},$$

в которой k-я строчка содержит степени принадлежности объекта $X_k = (x_{k1}, x_{k2}, \ldots, x_{kn})$ кластерам $A_1, A_2, \ldots, A_c$. Единственным отличием матриц **F** и **U** является то, что при нечетком разбиении степень принадлежности объекта к кластеру прини-

мает значения из интервала [0, 1], а при четком – из двухэлементного множества {0, 1}. Аналогичные (2.23)–(2.24) условия для матрицы нечеткого разбиения записываются так [26]:

$$\sum_{i=\overline{1,c}} \mu_{ki} = 1, \quad k = \overline{1, M}; \tag{2.26}$$

$$0 < \sum_{k=\overline{1,M}} \mu_{ki} < M, \quad i = \overline{1, c}. \tag{2.27}$$

Нечеткое разбиение позволяет просто решить проблему объектов, расположенных на границе двух кластеров – им назначают одинаковые степени принадлежностей, например, по 0,5. Недостаток нечеткого разбиения проявляется при работе с объектами, удаленными от центров всех кластеров. Удаленные объекты имеют мало общего с любым из кластеров, поэтому интуитивно хочется назначить им малые степени принадлежности. Однако по условию (2.26) сумма степеней принадлежности равна единице, как у удаленных, так и у близких к центрам кластеров объектов. Для устранения этого недостатка можно использовать возможностное разбиение, при котором произвольный объект из **X** должен принадлежать хотя бы одному кластеру. Для этого ослабляют условие (2.26) следующим образом:

$$\exists i: \ \mu_{ki} > 0, \ \forall k.$$

Для оценки качества нечеткого разбиения используется следующий критерий разброса [26]:

$$\sum_{i=\overline{1,c}} \sum_{k=\overline{1,M}} (\mu_{ki})^m \left\| V_i - X_k \right\|^2, \tag{2.28}$$

где $V_i = \dfrac{\sum_{k=\overline{1,M}} (\mu_{ki})^m X_k}{\sum_{k=\overline{1,M}} (\mu_{ki})^m}$ – центры нечетких кластеров; $m \in (1, \infty)$ – экспоненциальный вес.

Наиболее известный и часто применяемый метод минимизации критерия (2.28) – алгоритм нечетких *c*-средних. Он базируется на методе неопределенных множителей Лагранжа. Алгоритм находит локальный оптимум, поэтому выполнение его из различных начальных точек может привести к разным результатам.

Алгоритм нечетких *c*-средних [26]:

Шаг 1. Установить параметры алгоритма: c – количество кластеров; m – экспоненциальный вес; ε – параметр останова алгоритма.

Шаг 2. Случайным образом сгенерировать матрицу нечеткого разбиения **F**, удовлетворяющую условиям (2.26)–(2.27).

Шаг 3. Рассчитать центры кластеров по формуле:

$$V_i = \frac{\sum\limits_{k=\overline{1,N}} (\mu_{ki})^m X_k}{\sum\limits_{k=\overline{1,N}} (\mu_{ki})^m}, \quad i = \overline{1, c}.$$

Шаг 4. Рассчитать расстояния между объектами из **X** и центрами кластеров:

$$D_{ki} = \sqrt{\|X_k - V_i\|^2}, \quad k = \overline{1, M}, \quad i = \overline{1, c}.$$

Шаг 5. Пересчитать элементы матрицы нечеткого разбиения для всех $k = \overline{1, M}$ и $i = \overline{1, c}$:

$$\text{если } D_{ki} > 0, \text{ то } \mu_{ki} = \frac{1}{\left(D_{ik}^2 \sum\limits_{j=\overline{1,c}} \frac{1}{D_{jk}^2} \right)^{1/(m-1)}};$$

$$\text{если } D_{ki} = 0, \text{ то } \mu_{ki} = \begin{cases} 1, & \text{если } j = i, \\ 0, & \text{если } j \neq i, \end{cases} \quad j = \overline{1, c}.$$

Шаг 6. Проверить условие $\|\mathbf{F} - \mathbf{F}^*\|^2 < \varepsilon$, где $\mathbf{F}^*$ – матрица нечеткого разбиения на предыдущей итерации алгоритма. Если «Да», то перейти к шагу 7, иначе – к шагу 3.

Шаг 7. Конец.

Пример 2.5. Данные представлены следующей таблицей:

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_1	1	1	1	3	3	3	5	7	9	11	11	11	13	13	13
x_2	1	4	7	2	4	6	4	4	4	2	4	6	1	4	7

Графическое изображение этих данных представляет собой «бабочку» (рис. 2.13*а*). Необходимо выполнить кластеризацию данных по алгоритму нечетких *c*-средних.

Установим такие параметры алгоритма: $c = 2$, $m = 2$ и $\varepsilon = 10^{-5}$. После восьми итераций получим следующее нечеткое разбиение:

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
μ_{k1}	0,909	0,976	0,909	0,947	0,998	0,947	0,879	0,500	0,121	0,053	0,002	0,053	0,091	0,024	0,091
μ_{k2}	0,091	0,024	0,091	0,053	0,002	0,053	0,121	0,500	0,879	0,947	0,998	0,947	0,909	0,976	0,909

Значение критерия (2.28) для этого нечеткого разбиения равно 82,94. Результаты нечеткой кластеризации показаны на рис 2.13*г* (размер маркеров пропорционален степени принадлежности объекта кластеру). Трехмерные изображения нечетких кластеров приведены на рис. 2.14.

В алгоритме нечетких *c*-средних самым важным параметром является количество кластеров. Правильно выбрать количество кластеров для реальных задач без какой-либо априорной информации о структурах в данных достаточно сложно. Существует два формальных подхода к выбору числа кластеров.

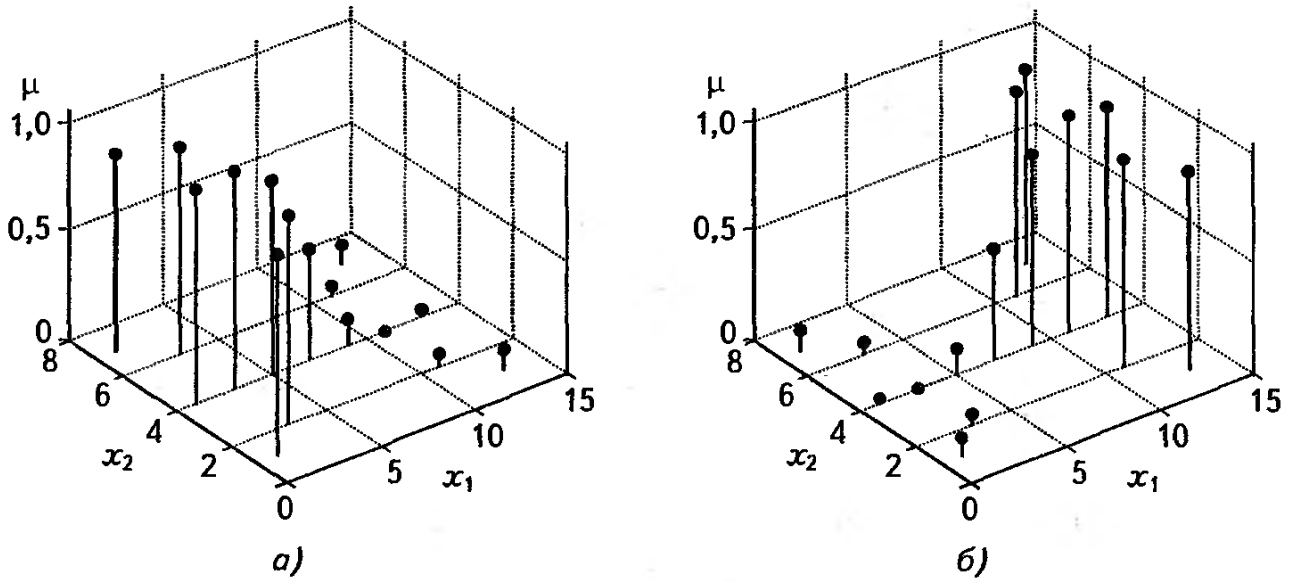


Рис. 2.14. Нечеткие кластеры (к примеру 2.5)

а – нечеткий кластер №1; *б* – нечеткий кластер №2

Первый подход основан на критерии компактности и отделимости полученных кластеров. Логично предположить, что при правильном выборе количества кластеров данные будут разбиты на компактные и хорошо отделимые друг от друга группы. В противном случае кластеры, вероятно, не будут компактными и хорошо отделимыми. Существует несколько критериев оценки компактности кластеров, однако вопрос о том, как формально и достоверно определить правильность выбора количества кластеров для произвольного набора данных все еще остается открытым. Для алгоритма нечетких *c*-средних в [24] рекомендуется использовать индекс Хей-Бени (Xei-Beni), предложенный в [42]:

$$\chi = \frac{\sum_{i=1,c}\sum_{k=1,M}(\mu_{ki})^m \left\|X_k - V_i\right\|^2}{M \min_{i\neq j}\left(\left\|X_k - V_i\right\|^2\right)}.$$

Второй подход предлагает начинать кластеризацию при достаточно большом числе кластеров, а затем последовательно объединять схожие смежные кластеры. При этом используются различные формальные критерии схожести кластеров.

Экспоненциальный вес (m) в алгоритме нечетких *c*-средних задает уровень нечеткости, размазанности получаемых кластеров. Чем больше m, тем нечеткое разбиение размазаннее. При $m \to \infty$ элементы матрицы **F** приближаются к $1/c$, следовательно, все объекты будут принадлежать ко всем кластерам с одной и той же степенью. Экспоненциальный вес позволяет при формировании координат центров кластеров усилить влияние объектов с большими степенями принадлежности и уменьшить влияние объектов с малыми степенями принадлежности. На сегодня не существует теоретически обоснованного правила выбора значения экспоненциального веса. Обычно устанавливают $m = 2$.

2.2.2.3. Обобщения алгоритма нечетких c-средних

В базовом алгоритме нечетких c-средних расстояние между объектом $X=(x_1, x_2, ..., x_n)$ и центром кластера $V=(v_1, v_2, ..., v_n)$ рассчитывается через стандартную евклидову норму: $D^2=\|X-V\|^2$. В кластерном анализе применяются и другие нормы, среди которых часто используется диагональная норма и норма Махалонобиса. В общем виде норму можно задать через симметричную положительно определенную матрицу **B** размером $n \times n$:

$$\|X-V\|_{\mathbf{B}}^2=(X-V)\mathbf{B}(X-V)^T,$$

где T – операция транспонирования.

Для евклидовой нормы матрица **B** представляет собой единичную матрицу:

$$\mathbf{B}=\begin{bmatrix}1 & 0 & \cdots & 0\\ 0 & 1 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & 1\end{bmatrix}.$$

При евклидовой норме кластеры выделяются в виде гиперсфер.

Для диагональной нормы матрица **B** задается следующим образом:

$$\mathbf{B}=\begin{bmatrix}\omega_1 & 0 & \cdots & 0\\ 0 & \omega_2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \omega_n\end{bmatrix}.$$

Элементы главной диагонали матрицы интерпретируются как веса координат. Диагональная норма позволяет выделить кластеры в виде гиперэллипсоидов, ориентированных вдоль координатных осей.

Для нормы Махалонобиса матрица **B** рассчитывается через ковариационную матрицу от **X**:

$$\mathbf{B}=\mathbf{R}^{-1},$$

где $\mathbf{R}=\frac{1}{M}\sum_{k=\overline{1,M}}(X_k-\bar{X})^T(X_k-\bar{X})$ – ковариационная матрица; $\bar{X}=\frac{1}{M}\sum_{k=\overline{1,M}}X_k$ – вектор средних значений данных.

При норме Махаланобиса кластеры получаются в виде гиперэллипсоидов, оси которых могут быть ориентированы в произвольных направлениях.

Примеры изолиний различных норм показаны на рис. 2.15. На рис. 2.16 приведен пример кластеризации методом нечетких c-средних при евклидовом расстоянии: слева изображены объекты кластеризации; справа показаны результаты нечеткой кластеризации. Центры нечетких кластеров обозначены символами «+».

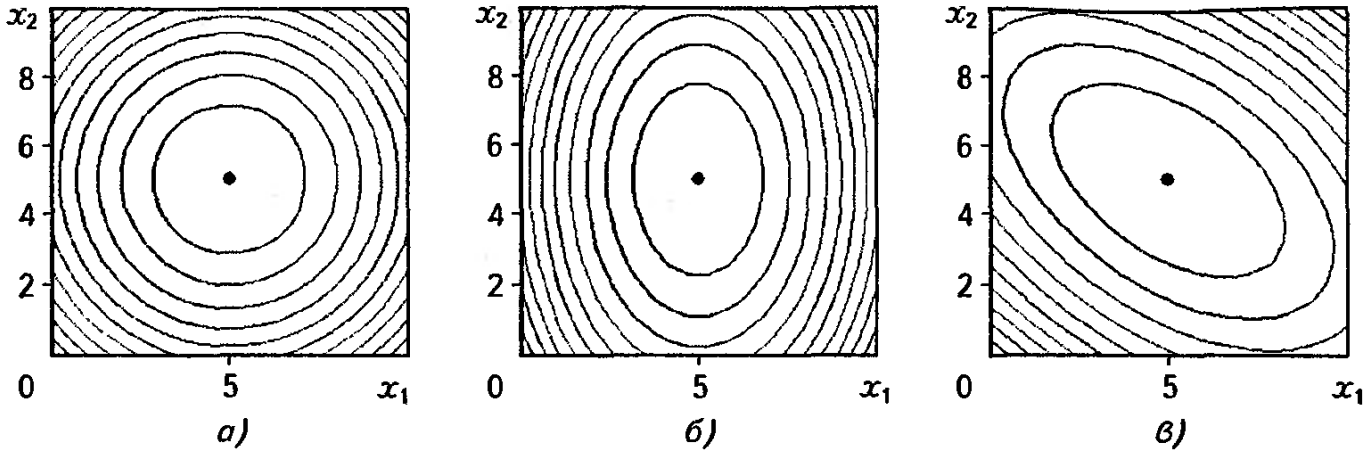


Рис. 2.15. Изолинии различных норм

а – евклидова норма; *б* – диагональная норма; *в* – норма Махаланобиса

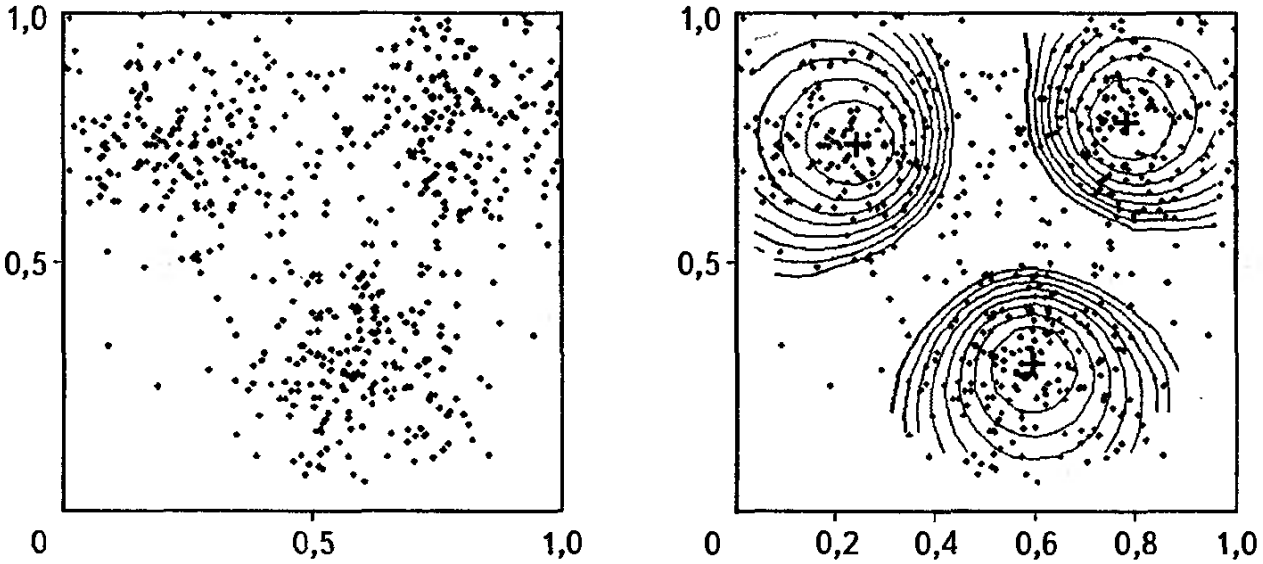


Рис. 2.16. Нечеткая кластеризация при евклидовой норме

Восемь изолиний функций принадлежности нечетких кластеров построены для значений 0,67; 0,71; 0,75; 0,79; 0,83; 0,87; 0,91 и 0,95.

Для некоторых наборов данных «глазная кластеризация» позволяет выделить скопления объектов в виде различных геометрических фигур: сфер, эллипсоидов разной ориентации, цепочек и т.п. В результате кластеризации по алгоритмам с фиксированной нормой форма всех кластеров получается одинаковой. Алгоритмы кластеризации как бы «навязывают» данным неприсущую им структуру, что приводит не только к неоптимальным, но иногда и к принципиально неправильным результатам. Для устранения этого недостатка предложено несколько методов, среди которых выделим алгоритм Густавсона–Кесселя (Gustafson–Kessel) [29].

Алгоритм Густавсона–Кесселя использует адаптивную норму для каждого кластера, т.е. для каждого i-го кластера существует своя норм-порождающая матрица $\mathbf{B}_i$. По этому алгоритму выделяются кластеры различной геометрической формы, так как оптимизируются не только координаты центров кластеров и матрица нечеткого разбиения, но также и норм-порождающие матрицы. Критерий

оптимальности (2.28) линеен относительно $\mathbf{B}_i$, поэтому для получения ненулевых решений вводятся некоторые ограничения на норм-пораждающие матрицы. В алгоритме Густавсона–Кесселя это следующее ограничение на значение определителя норм-пораждающих матриц:

$$|\mathbf{B}_i| = \beta_i, \quad \beta_i > 0, \quad i = \overline{1, c}.$$

Оптимальное решение находят посредством метода неопределенных множителей Лагранжа. Алгоритм Густавсона–Кесселя обладает значительно большей вычислительной трудоемкостью по сравнению с алгоритмом нечетких *c*-средних.

2.2.3. КЛАСТЕРИЗАЦИЯ ГОРНЫМ АЛГОРИТМОМ

Горный алгоритм кластеризации предложен Ягером (Yager) и Филевым (Filev) в 1993 г. Особенностью алгоритма является то, что он не требует задания количества кластеров. Изложение горного алгоритма кластеризации базируется на книге [43].

На первом шаге горной кластеризации определяют точки, которые могут быть центрами кластеров. На втором шаге для каждой такой точки рассчитывается значение потенциала, показывающего возможность формирования кластера в ее окрестности. Чем гуще расположены объекты в окрестности прототипа кластера, тем выше значение его потенциала. На третьем шаге итерационно выбираются центры кластеров среди точек с максимальными потенциалами. Рассмотрим горный алгоритм кластеризации подробнее.

На *первом шаге* необходимо сформировать потенциальные центры кластеров. Для алгоритма горной кластеризации число потенциальных центров кластеров (Q) должно быть конечным. Такими центрами могут быть как объекты кластеризации (строчки матрицы **X**), так и особые точки факторного пространства. В последнем случае диапазоны изменения входных признаков разбивают на несколько интервалов. Проведя через точки разбиения прямые, параллельные координатным осям, получаем решеточный гиперкуб. Узлы этой решетки и будут соответствовать центрам потенциальных кластеров. Обозначим через q_r – количество значений, которые могут принимать центры кластеров по r-й координате, $r = \overline{1, n}$. Тогда количество возможных кластеров будет равно: $Q = q_1 q_2 \ldots q_n$.

На *втором шаге* алгоритма рассчитывается потенциал центров кластеров по следующей формуле:

$$P(Z_h) = \sum_{k=\overline{1,M}} \exp(-\alpha D(Z_h, X_k)), \quad h = \overline{1, Q},$$

где $Z_h = (z_{1,h}, z_{2,h}, \ldots, z_{n,h})$ – потенциальный центр h-го кластера, $h = \overline{1, Q}$; α – положительная константа; $D(Z_h, X_k)$ – расстояние между потенциальным центром кластера (Z_h) и объектом кластеризации (X_k). В евклидовом пространстве это расстояние рассчитывается по формуле:

$$D(Z_h, X_k) = \sqrt{\|Z_h - X_k\|^2}.$$

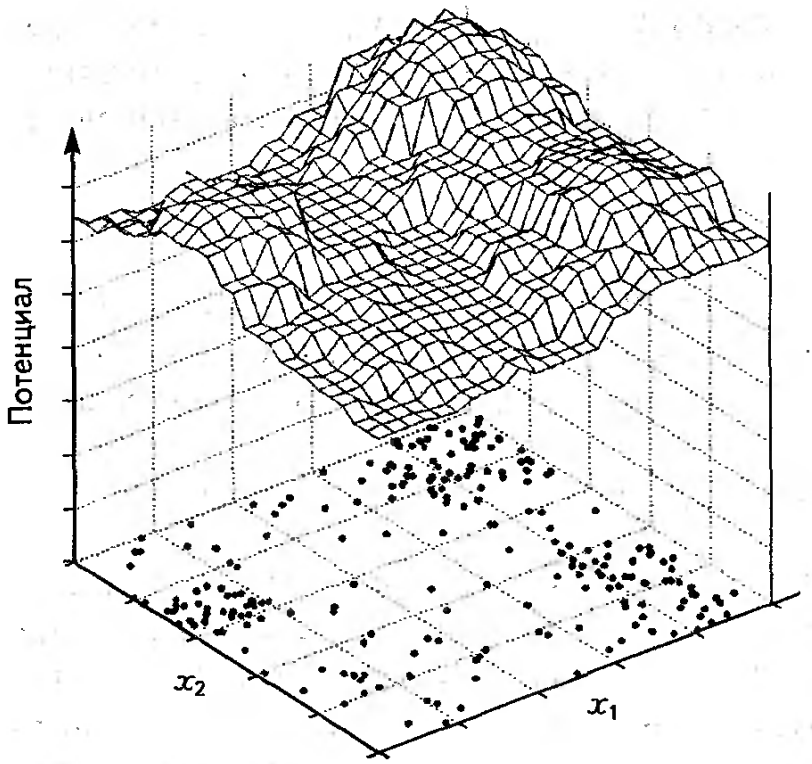


Рис. 2.17. Распределение потенциала при кластеризации горным алгоритмом

Когда объекты кластеризации заданы двумя признаками ($n=2$), графическое изображение распределения потенциала будет представлять собой поверхность, напоминающую горный рельеф (рис. 2.17). Отсюда и название – горный алгоритм кластеризации.

На *третьем шаге* алгоритма в качестве центров кластеров выбирают координаты «горных вершин». Центром первого кластера назначают точку с наибольшим потенциалом. Обычно, наивысшая вершина окружена несколькими достаточно высокими пиками. Назначение центром следующего кластера точки с максимальным потенциалом среди оставшихся вершин привело бы к выделению большого числа близко расположенных центров кластеров. Следовательно, перед выбором следующего кластерного центра необходимо исключить влияние только что найденного кластера. Для этого значения потенциала оставшихся возможных центров кластеров пересчитывается следующим образом: от текущих значений потенциала вычитают вклад центра только что найденного кластера (поэтому кластеризацию по этому методу иногда называют субтрактивной, от англ. *subtractive* – вычитание). Перерасчет потенциала происходит по формуле:

$$P_2(Z_h)=P_1(Z_h)-P_1(V_1)\exp(-\beta D(Z_h,V_1)),\quad Z_h\neq V_1,\quad h=\overline{1,Q},$$

где $P_1(.)$ – потенциал на 1-й итерации; $P_2(.)$ – потенциал на 2-й итерации; β – положительная константа; V_1 – центр первого найденного кластера:

$$V_1=\arg\max_{Z_1,Z_2,\ldots,Z_Q}(P_1(Z_1),\ P_1(Z_2),\ \ldots,\ P_1(Z_Q)).$$

Центр второго кластера определяется по максимальному значению обновленного потенциала:

$$V_2=\arg\max_{Z_h:\ Z_h\neq V_1,\ h=\overline{1,Q}}(P_2(Z_1),\ P_2(Z_2),\ \ldots,\ P_2(Z_Q)).$$

Затем снова пересчитывается значение потенциалов:

$$P_3(Z_h)=P_2(Z_h)-P_2(V_2)\exp(-\beta D(Z_h,V_2)),\quad Z_h\neq V_1,\quad Z_h\neq V_2,\quad h=\overline{1,Q}.$$

Итерационная процедура выделения центров кластеров продолжается до тех пор, пока максимальное значение потенциала превышает некоторый порог. Кластеризация по горному алгоритму не является нечеткой, однако ее часто используют при синтезе нечетких правил из данных.

2.2.4. СИНТЕЗ НЕЧЕТКИХ ПРАВИЛ ПО РЕЗУЛЬТАТАМ КЛАСТЕРИЗАЦИИ

Обозначим через $V_1, V_2, ..., V_c$ центры кластеров, найденные в результате горной кластеризации. Для упрощения выкладок примем, что центры кластеров заданы двумя координатами: $V_i = (x_i, y_i), i = \overline{1, c}$. Задача состоит в синтезе нечетких правил, связывающих вход (x) с выходом (y).

Центру кластера V_i $(i = \overline{1, c})$ ставится в соответствие одно нечеткое правило [43]:

Если $x = \tilde{x}_i$, то $y = \tilde{y}_i$,

в котором нечеткие термы интерпретируются так: $\tilde{x}_i$ – «около x_i» и $\tilde{y}_i$ – «около y_i». Функции принадлежностей этих нечетких термов задаются гауссовой кривой:

$$\tilde{x}_i = \exp\left(-\frac{1}{2}\left(\frac{x - x_i}{\beta}\right)^2\right), \quad \tilde{y}_i = \exp\left(-\frac{1}{2}\left(\frac{y - y_i}{\beta}\right)^2\right), \quad i = \overline{1, c},$$

где β – параметр алгоритма горной кластеризации.

Нечеткие правила можно синтезировать и по результатам нечеткой кластеризации. Пусть объекты кластеризации имеют два признака ($n = 2$). Тогда результаты нечеткого разбиения можно представить трехмерной поверхностью. Для построения такой поверхности следует для каждого объекта отложить по осям абсцисс и ординат значения признаков, а по оси аппликат – степень принадлежности объекта нечеткому кластеру. Количество поверхностей равно числу кластеров (n). На рис. 2.14 показаны функции принадлежности нечетких кластеров из примера 2.5. Они напоминают нечеткие отношения (сравни с рис. 1.15 и рис. 1.23). Следовательно, каждому кластеру можно поставить в соответствие одно нечеткое правило.

По результатам нечеткой кластеризации можно синтезировать нечеткие правила различных баз знаний: синглтонной, Мамдани и Сугено. Функции принадлежности термов в посылках правила получают проецированием степеней принадлежности соответствующего кластера (строчек матрицы нечеткого разбиения **F**) на оси входных переменных. Затем полученные множества степеней принадлежностей аппроксимируют подходящими параметрическими функциями принадлежности. В качестве заключения правила синглтонной базы знаний выбирают координату центра кластера. Заключения правил базы знаний Мамдани находят также как и функции принадлежности термов входных переменных. Заключения правил базы знаний Сугено находят по методу наименьших квадратов. При кластеризации с использованием нормы Махалонобиса в качестве заключений правил типа Сугено могут быть выбраны уравнения длинных осей гиперэллипсоидов.

Пример 2.6. Данные для кластеризации, а также центры двух нечетких кластеров изображены на рис. 2.18. На этом рисунке выведены изолинии для следующих значений функций принадлежности: 0,95; 0,90; 0,85; 0,80; 0,75; 0,70 и 0,65. Функции принадлежности нечетких кластеров также показаны двумя трехмерными поверхностями. Они интерпретируются с функциями принадлежности термов, показанными на рис. 2.19, следующими нечеткими правилами:

ЕСЛИ x = «низкий», ТО y = «низкий»,

ЕСЛИ x = «высокий», ТО y = «высокий».

Функции принадлежности получены проецированием поверхностей с рис. 2.18 на оси x и y. Маркеры на графиках функций принадлежности соответствует одному объекту кластеризации.

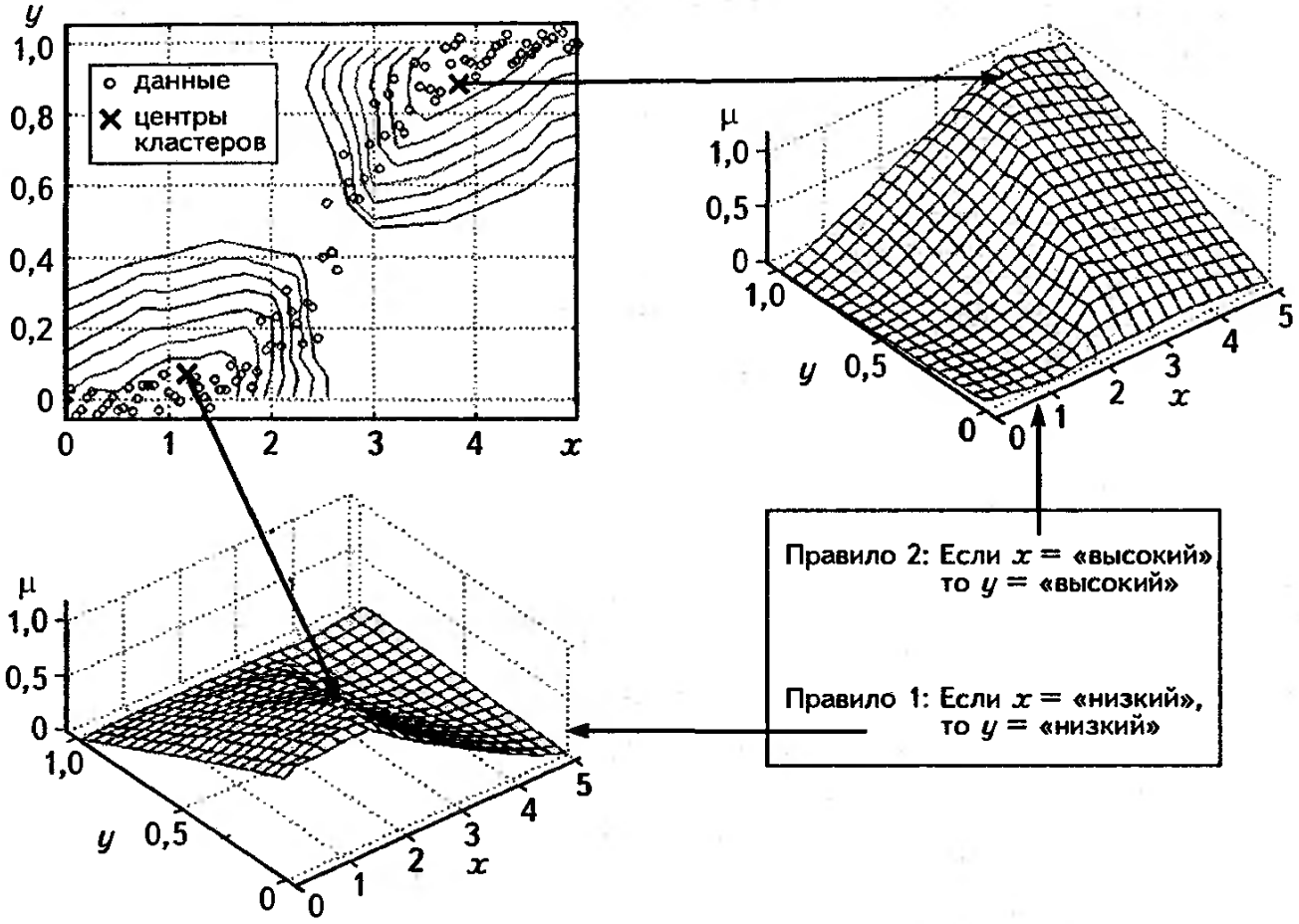


Рис. 2.18. Экстракция нечетких правил через кластеризацию (к примеру 2.6)

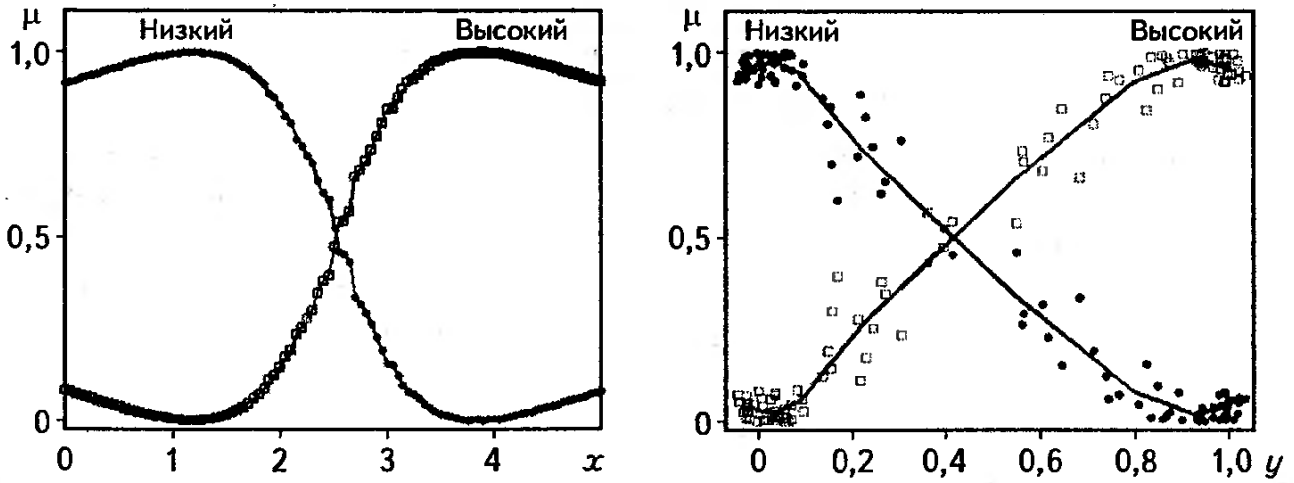


Рис. 2.19. Функции принадлежности, полученные посредством кластеризации данных примера 2.6

2.3. ПРИНЯТИЕ РЕШЕНИЙ В НЕЧЕТКИХ УСЛОВИЯХ ПО СХЕМЕ БЕЛЛМАНА–ЗАДЕ

В 1970 г. Беллман и Заде опубликовали статью *«Decision – Making in Fuzzy Environment»* ([25], русский перевод [1]), 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. Принятие решения – это выбор альтернативы, которая одновременно удовлетворяет и нечетким целям, и нечетким ограничениям. В этом смысле цели и ограничения являются симметричными относительно решения. Это стирает различия между ними и позволяет представить решение как слияние нечетких целей и ограничений. В разделе излагаются основы теории принятия решений в нечетких условиях по схеме Беллмана–Заде с примером нечеткого многокритериального анализа вариантов бренд-проекта. При написании раздела использованы источники [1, 12, 16, 18, 46].

2.3.1. НЕЧЕТКИЕ ЦЕЛИ, ОГРАНИЧЕНИЯ И РЕШЕНИЯ

Пусть $X = \{x\}$ – множество альтернатив. Нечеткую цель $\widetilde{G}$ будем отождествлять с нечетким множеством $\widetilde{G}$ в X. Например, если альтернативами являются действительные числа, т.е. $X = \mathbf{R}$, и нечеткая цель сформулирована как «x должно быть около 10», то ее можно представить нечетким множеством с такой функцией принадлежности:

$$\mu_G(x) = \frac{1}{1+(x-10)^2}, \quad x \in X. \tag{2.29}$$

Аналогичным образом нечеткое ограничение $\widetilde{C}$ определяется как некоторое нечеткое множество на универсальном множестве X. Например, нечеткое ограничение «x должно быть значительно больше 8» при $X = \mathbf{R}$ можно представить нечетким множеством с такой функцией принадлежности:

$$\mu_C(x) = \begin{cases} 0, & \text{если } x < 5, \\ \dfrac{1}{1+\exp(-0{,}8(x-8))}, & \text{если } x \geq 5. \end{cases} \tag{2.30}$$

Нечеткое решение $\widetilde{D}$ также определяется как нечеткое множество на универсальном множестве альтернатив X. Функция принадлежности этого нечеткого множества показывает, насколько хорошо решение удовлетворяет *нечетким целям* И *нечетким ограничениям*. Логической операции И, которая связывает цели с ограничениями, соответствует пересечение нечетких множеств. Следовательно, решение – это пересечение нечеткой цели с нечетким ограничением:

$$\widetilde{D} = \widetilde{G} \cap \widetilde{C}. \tag{2.31}$$

Пример 2.7. Нечеткая цель $\tilde{G}$ и нечеткое ограничение $\tilde{C}$ сформулированы так:
$\tilde{G}$: «x должно быть около 10» и $\tilde{C}$: «x должно быть значительно больше 8».

Функции принадлежности нечетких множеств $\tilde{G}$ и $\tilde{C}$ заданы выражениями (2.29) и (2.30). Нечеткое решение $\tilde{D}$ найдем по формуле (2.31). Учитывая, что пересечению нечетких множеств соответствует операция минимума над функций принадлежности, получаем:

$$\mu_D(x)=\begin{cases}0, & \text{если } x<5,\\ \min\left(\dfrac{1}{1+\exp(-0{,}8(x-8))},\dfrac{1}{1+(x-10)^2}\right), & \text{если } x\geq 5.\end{cases}$$

Взаимосвязь между нечеткими целью, ограничением и решением показана на рис. 2.20. Цель и ограничения конфликтуют между собой, поэтому в нечетком множестве $\tilde{D}$ нет ни одного элемента со степенью принадлежности, равной 1. Значит, не существует альтернативы, которая полностью удовлетворяет и цели, и ограничению. В качестве четкого решения в таких случаях обычно выбирают альтернативу с максимальной степенью принадлежности нечеткому множеству $\tilde{D}$.

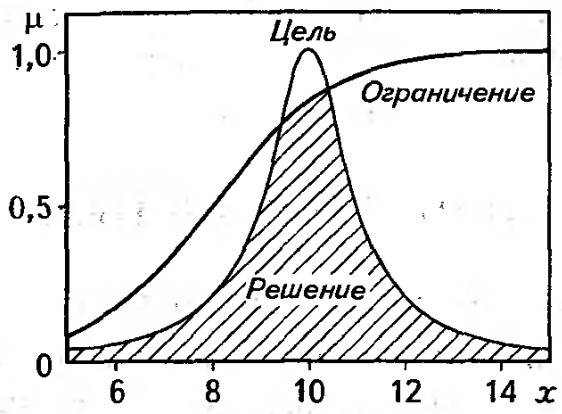


Рис. 2.20. Принятие решения по принципу Беллмана–Заде (к примеру 2.7)

При принятии решений по схеме Беллмана–Заде не делается никакого различия между целью и ограничениями. Всякое разделение на цель и ограничения является условным: в формуле (2.31) можно поменять местами цель с ограничением, при этом решение не изменится. В традиционной теории принятия решений подобные замены функции предпочтения на ограничение недопустимы. Однако и здесь прослеживается некоторое скрытое сходство между целями и ограничениями. Оно становится явным при использовании метода неопределенных множителей Лагранжа и штрафных функций, когда цель и ограничения объединяются в одну функцию.

В общем случае, когда имеется n целей и m ограничений, результирующее решение по схеме Беллмана–Заде определяется пересечением всех целей и ограничений [1]:

$$\tilde{D}=\tilde{G}_1\cap\tilde{G}_2\cap\ldots\cap\tilde{G}_n\cap\tilde{C}_1\cap\tilde{C}_1\cap\ldots\cap\tilde{C}_m$$

и соответственно

$$\mu_D=\mu_{G_1}\wedge\mu_{G_2}\wedge\ldots\wedge\mu_{G_n}\wedge\mu_{C_1}\wedge\mu_{C_2}\wedge\ldots\wedge\mu_{C_m}.$$

До сих пор предполагалось, что все цели и ограничения, входящие в $\tilde{D}$, имеют одинаковую важность. Более привычная ситуация, в которой удовлетворение одним целям и (или) ограничениям важнее, чем другим. Обозначим через $\alpha_i\in(0,1)$ – коэффициент относительной важности i-й цели, а через $\beta_j\in(0,1)$ – коэффициент относительной важности j-го ограничения:

$$\sum_{i=\overline{1,n}}\alpha_i + \sum_{j=\overline{1,m}}\beta_j = 1.$$

Тогда функция принадлежности решения определяется так:

$$\mu_D = (\mu_{G_1})^{\alpha_1} \wedge (\mu_{G_2})^{\alpha_2} \wedge \ldots \wedge (\mu_{G_n})^{\alpha_n} \wedge (\mu_{C_1})^{\beta_1} \wedge (\mu_{C_2})^{\beta_2} \wedge \ldots \wedge (\mu_{C_m})^{\beta_m}. \quad (2.32)$$

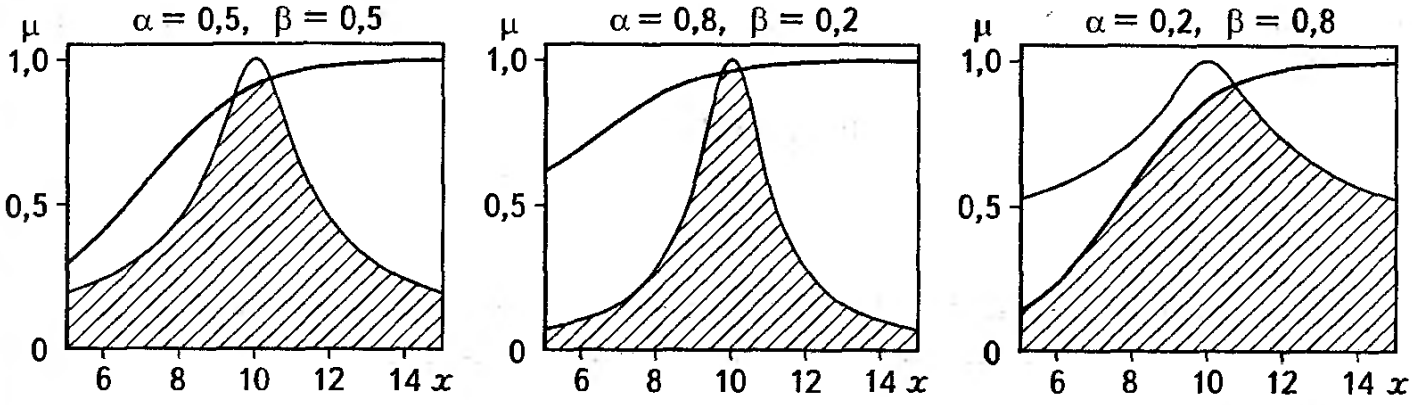


Рис. 2.21. Принятие решений при разной важности цели и ограничения (к примеру 2.7)

Чем меньше коэффициент относительной важности, тем соответствующее нечеткое множество цели или ограничения становится более размазанным и, следовательно, его роль в принятии решения снижается. На рис. 2.21 показаны нечеткие решения при различных коэффициентах важности цели и ограничения из примера 2.7.

2.3.2. НЕЧЕТКИЙ МНОГОКРИТЕРИАЛЬНЫЙ АНАЛИЗ ВАРИАНТОВ

Будем считать известными:

- $\mathbf{P} = \{P_1, P_2, \ldots, P_k\}$ – множество вариантов, которые подлежат многокритериальному анализу;
- $\mathbf{G} = \{G_1, G_2, \ldots, G_n\}$ – множество критериев, по которым оценивают варианты.

Задача многокритериального анализа состоит в упорядочивании элементов множества **P** по критериям из множества **G**.

Пусть $\mu_{G_i}(P_j)$ – число в диапазоне [0, 1], которым оценивается вариант $P_j \in \mathbf{P}$ по критерию $G_i \in \mathbf{G}$: чем больше число $\mu_{G_i}(P_j)$, тем лучше вариант P_j по критерию G_i, $i = \overline{1, n}$, $j = \overline{1, k}$. Тогда, критерий G_i можно представить нечетким множеством $\widetilde{G}_i$ на универсальном множестве вариантов **P** [12, 16]:

$$G_i = \left\{ \frac{\mu_{G_i}(P_1)}{P_1}, \frac{\mu_{G_i}(P_2)}{P_2}, \ldots, \frac{\mu_{G_i}(P_k)}{P_k} \right\}, \quad (2.33)$$

где $\mu_{G_i}(P_j)$ – степень принадлежности элемента P_j нечеткому множеству $\widetilde{G}_i$.

Находить степени принадлежности нечеткого множества (2.33) удобно методом построения функций принадлежности на основе парных сравнений. Он описан в подразделе 1.2.4. При использовании этого метода необходимо сформировать матрицы парных сравнений вариантов по каждому критерию. Общее количество таких матриц равно количеству критериев.

Наилучшим вариантом будем тот, который одновременно лучший по всем критериям. Нечеткое решение $\widetilde{D}$ находится как пересечения частных критериев [12, 46]:

$$\widetilde{D} = \widetilde{G}_1 \cap \widetilde{G}_2 \cap \ldots \cap \widetilde{G}_n = \left\{ \frac{\min\limits_{i=\overline{1,n}} \mu_{G_i}(P_1)}{P_1}, \frac{\min\limits_{i=\overline{1,n}} \mu_{G_i}(P_2)}{P_2}, \ldots, \frac{\min\limits_{i=\overline{1,n}} \mu_{G_i}(P_k)}{P_k} \right\}. \quad (2.34)$$

Согласно с полученным нечетким множеством $\widetilde{D}$, наилучшим вариантом следует считать тот, у которого наибольшая степень принадлежности:

$$D = \arg\ \max(\mu_D(P_1),\ \mu_D(P_2), \ldots,\ \mu_D(P_k)).$$

При неравновесных критериях степени принадлежности нечеткого множества $\widetilde{D}$ находят так [12, 46]:

$$\mu_D(P_j) = \min_{i=\overline{1,n}} (\mu_{G_i}(P_j))^{\alpha_i}, \quad j = \overline{1, k}, \quad (2.35)$$

где α_i – коэффициент относительной важности критерия G_i, $\alpha_1 + \alpha_1 + \ldots + \alpha_n = 1$.

Показатель степени α_i в формуле (2.35) концентрирует нечеткое множество $\widetilde{G}_i$ в соответствии с мерой важности критерия G_i. Коэффициенты относительной важности критериев могут быть определены различными методами, например, с помощью парных сравнений по шкале Саати [19].

2.3.3. НЕЧЕТКИЙ МНОГОКРИТЕРИАЛЬНЫЙ АНАЛИЗ БРЕНД-ПРОЕКТОВ

В качестве примера принятия решений в нечетких условиях по схеме Беллмана–Заде рассмотрим сравнение четырех проектов ($P_1 \div P_4$) выведения бренда на рынок [18]. Для оценки проектов воспользуемся такими критериями:

G_1 – уровень проработки проекта;

G_2 – ожидаемый эффект;

G_3 – риски;

G_4 – скорость вывода бренда;

G_5 – перспективы развития бренда;

G_6 – стоимость проекта.

Экспертные сравнения проектов по критериям $G_1 \div G_6$ приведены в табл. 2.7. По каждому критерию сравнивались шесть пар проектов.

Таблица 2.7

Парные сравнения проектов по шкале Саати

Критерий	Экспертные парные сравнения	
G_1	Отсутствует преимущество P_1 над P_2 Слабое преимущество P_1 над P_3 Существенное преимущество P_1 над P_4	Слабое преимущество P_2 над P_3 Существенное преимущество P_2 над P_4 Слабое преимущество P_3 над P_4
G_2	Слабое преимущество P_1 над P_2 Существенное преимущество P_1 над P_3 Явное пренмущество P_1 над P_4	Почти слабое преимущество P_2 над P_3 Слабое преимущество P_2 над P_4 Почти слабое преимущество P_3 над P_4
G_3	Существенное преимущество P_1 над P_2 Отсутствует пренмущество P_1 над P_3 Явное преимущество P_1 над P_4	Слабое преимущество P_2 над P_4 Существенное преимущество P_3 над P_2 Явное преимущество P_3 над P_4
G_4	Слабое преимущество P_2 над P_1 Отсутствует преимущество P_2 над P_4 Слабое преимущество P_4 над P_1	Существенное преимущество P_3 над P_1 Почти слабое преимущество P_3 над P_2 Слабое преимущество P_3 над P_4
G_5	Слабое преимущество P_2 над P_1 Отсутствует преимущество P_2 над P_3 Слабое преимущество P_3 над P_1	Существенное преимущество P_4 над P_1 Слабое преимущество P_4 над P_2 Почти слабое преимущество P_4 над P_3
G_6	Явное преимущество P_2 над P_1 Слабое преимущество P_2 над P_3 Отсутствует преимущество P_2 над P_4	Слабое преимущество P_3 над P_1 Явное преимущество P_4 над P_1 Слабое преимущество P_4 над P_3

Экспертным высказываниям соответствуют следующие матрицы парных сравнений:

$$\mathbf{A}(G_1)=\begin{bmatrix}1 & 1 & 3 & 5\\ 1 & 1 & 3 & 5\\ 1/3 & 1/3 & 1 & 3\\ 1/5 & 1/5 & 1/3 & 1\end{bmatrix};\qquad \mathbf{A}(G_2)=\begin{bmatrix}1 & 3 & 5 & 7\\ 1/3 & 1 & 2 & 3\\ 1/5 & 1/2 & 1 & 2\\ 1/7 & 1/3 & 1/2 & 1\end{bmatrix};$$

$$\mathbf{A}(G_3)=\begin{bmatrix}1 & 5 & 1 & 7\\ 1/5 & 1 & 1/5 & 3\\ 1 & 5 & 1 & 7\\ 1/7 & 1/3 & 1/7 & 1\end{bmatrix};\qquad \mathbf{A}(G_4)=\begin{bmatrix}1 & 1/3 & 1/5 & 1/3\\ 3 & 1 & 1/2 & 1\\ 5 & 2 & 1 & 3\\ 3 & 1 & 1/3 & 1\end{bmatrix};\qquad (2.36)$$

$$\mathbf{A}(G_5)=\begin{bmatrix}1 & 1/3 & 1/3 & 1/5\\ 3 & 1 & 1 & 1/3\\ 3 & 1 & 1 & 1/2\\ 5 & 3 & 2 & 1\end{bmatrix};\qquad \mathbf{A}(G_6)=\begin{bmatrix}1 & 1/7 & 1/3 & 1/7\\ 7 & 1 & 3 & 1\\ 3 & 1/3 & 1 & 1/3\\ 7 & 1 & 3 & 1\end{bmatrix}.$$

В каждой матрице шесть элементов соответствуют парным сравнениям из табл. 2.7. Остальные элементы найдены с учетом того, что матрица парных сравнений является диагональной и обратно симметричной (подробнее см. подраздел 1.2.4).

Применяя формулы (1.2) и (1.3) к матрицам парных сравнений (2.36), получаем следующие нечеткие множества:

$$\widetilde{G}_1 = \left\{\frac{0,39}{P_1}, \frac{0,39}{P_2}, \frac{0,15}{P_3}, \frac{0,07}{P_4}\right\}; \quad \widetilde{G}_2 = \left\{\frac{0,59}{P_1}, \frac{0,22}{P_2}, \frac{0,12}{P_3}, \frac{0,07}{P_4}\right\};$$

$$\widetilde{G}_3 = \left\{\frac{0,42}{P_1}, \frac{0,11}{P_2}, \frac{0,42}{P_3}, \frac{0,05}{P_4}\right\}; \quad \widetilde{G}_4 = \left\{\frac{0,08}{P_1}, \frac{0,23}{P_2}, \frac{0,48}{P_3}, \frac{0,21}{P_4}\right\}; \quad (2.37)$$

$$\widetilde{G}_5 = \left\{\frac{0,08}{P_1}, \frac{0,23}{P_2}, \frac{0,48}{P_3}, \frac{0,21}{P_4}\right\}; \quad \widetilde{G}_6 = \left\{\frac{0,06}{P_1}, \frac{0,40}{P_2}, \frac{0,14}{P_3}, \frac{0,40}{P_4}\right\}.$$

Из (2.37) следует, что проект P_1 является лучшим по критериям G_1, G_2 и G_3, проект P_2 – по критериям G_1 и G_6, проект P_3 – по критериям G_3 и G_4, а проект P_4 – по критериям G_5 и G_6. Поэтому выбор проекта будет зависить от важности критериев.

Для расчета коэффициентов относительной важности критериев воспользуемся экспертным методом парных сравнений. Будем считать известными следующие лингвистические парные сравнения важности критериев:

- *почти существенное* преимущество G_1 над G_4;
- *отсутствует* преимущество G_1 над G_5;
- *слабое* преимущество G_1 над G_6;
- *слабое* преимущество G_2 над G_1;
- *почти слабое* преимущество G_2 над G_3;
- *почти сильное* преимущество G_2 над G_4;
- *слабое* преимущество G_2 над G_5;
- *существенное* преимущество G_2 над G_6;
- *почти слабое* преимущество G_3 над G_1;
- *существенное* преимущество G_3 над G_4;
- *почти слабое* преимущество G_3 над G_5;
- *слабое* преимущество G_3 над G_6;
- *слабое* преимущество G_5 над G_4;
- *почти слабое* преимущество G_5 над G_6;
- *почти слабое* преимущество G_6 над G_4.

Экспертным высказываниям соответствует следующая матрица парных сравнений:

$$\mathbf{A}=\begin{bmatrix} 1 & 1/3 & 1/2 & 4 & 1 & 3 \\ 3 & 1 & 2 & 6 & 3 & 5 \\ 2 & 1/2 & 1 & 5 & 2 & 3 \\ 1/4 & 1/6 & 1/5 & 1 & 1/3 & 1/2 \\ 1 & 1/3 & 1/2 & 3 & 1 & 2 \\ 1/3 & 1/5 & 1/3 & 2 & 1/2 & 1 \end{bmatrix}.$$

По формулам (1.2) и (1.3) находим коэффициенты относительной важности критериев G_1, G_2, ..., G_6:

$$\alpha_1=0,15;\quad \alpha_2=0,34;\quad \alpha_3=0,26;\quad \alpha_4=0,05;\quad \alpha_5=0,13;\quad \alpha_6=0,07,$$

что означает наибольшую важность при принятии решения ожидаемого эффекта (G_2) и рисков (G_3).

По формуле (2.35) получаем такие нечеткие множества:

$$\widetilde{G}_1^{\alpha_1}=\left\{\frac{0,39^{0,15}}{P_1},\frac{0,39^{0,15}}{P_2},\frac{0,15^{0,15}}{P_3},\frac{0,07^{0,15}}{P_4}\right\}=\left\{\frac{0,868}{P_1},\frac{0,868}{P_2},\frac{0,753}{P_3},\frac{0,667}{P_4}\right\};$$

$$\widetilde{G}_2^{\alpha_2}=\left\{\frac{0,59^{0,34}}{P_1},\frac{0,22^{0,34}}{P_2},\frac{0,12^{0,34}}{P_3},\frac{0,07^{0,34}}{P_4}\right\}=\left\{\frac{0,835}{P_1},\frac{0,596}{P_2},\frac{0,490}{P_3},\frac{0,409}{P_4}\right\};$$

$$\widetilde{G}_3^{\alpha_3}=\left\{\frac{0,42^{0,26}}{P_1},\frac{0,11^{0,26}}{P_2},\frac{0,42^{0,26}}{P_3},\frac{0,05^{0,26}}{P_4}\right\}=\left\{\frac{0,797}{P_1},\frac{0,552}{P_2},\frac{0,797}{P_3},\frac{0,456}{P_4}\right\};$$

$$\widetilde{G}_4^{\alpha_4}=\left\{\frac{0,08^{0,05}}{P_1},\frac{0,23^{0,05}}{P_2},\frac{0,48^{0,05}}{P_3},\frac{0,21^{0,05}}{P_4}\right\}=\left\{\frac{0,894}{P_1},\frac{0,936}{P_2},\frac{0,969}{P_3},\frac{0,933}{P_4}\right\};$$

$$\widetilde{G}_5^{\alpha_5}=\left\{\frac{0,08^{0,13}}{P_1},\frac{0,23^{0,13}}{P_2},\frac{0,48^{0,13}}{P_3},\frac{0,21^{0,13}}{P_4}\right\}=\left\{\frac{0,717}{P_1},\frac{0,813}{P_2},\frac{0,823}{P_3},\frac{0,909}{P_4}\right\};$$

$$\widetilde{G}_6^{\alpha_6}=\left\{\frac{0,06^{0,07}}{P_1},\frac{0,40^{0,07}}{P_2},\frac{0,14^{0,07}}{P_3},\frac{0,40^{0,07}}{P_4}\right\}=\left\{\frac{0,813}{P_1},\frac{0,938}{P_2},\frac{0,871}{P_3},\frac{0,938}{P_4}\right\}.$$

Пересечение этих нечетких множеств дает такие степени принадлежности нечеткого решения $\widetilde{D}$:

$$\mu_D(P_1)=\min(0,868,\ 0,835,\ 0,797,\ 0,894,\ 0,717,\ 0,813)=0,717;$$

$$\mu_D(P_2)=\min(0,868,\ 0,596,\ 0,552,\ 0,936,\ 0,813,\ 0,938)=0,552;$$

$$\mu_D(P_3) = \min(0{,}753,\ 0{,}490,\ 0{,}797,\ 0{,}969,\ 0{,}823,\ 0{,}871) = 0{,}490;$$

$$\mu_D(P_4) = \min(0{,}667,\ 0{,}409,\ 0{,}456,\ 0{,}933,\ 0{,}909,\ 0{,}938) = 0{,}409.$$

В результате получаем нечеткое множество

$$\widetilde{D} = \left\{ \frac{0{,}717}{P_1}, \frac{0{,}552}{P_2}, \frac{0{,}490}{P_3}, \frac{0{,}409}{P_4} \right\},$$

которое свидетельствует о преимуществе проекта P_1 над остальными. Таким образом, проект P_1 лучше других одновременно удовлетворяет все критерии с учетом их важности. Нечеткие множества, показывающие, насколько полно проекты $P_1 \div P_4$ удовлетворяют критериям $G_1 \div G_6$, запишем так:

$$\widetilde{P}_1 = \left\{ \frac{0{,}868}{G_1}, \frac{0{,}835}{G_2}, \frac{0{,}797}{G_3}, \frac{0{,}894}{G_4}, \frac{0{,}717}{G_5}, \frac{0{,}813}{G_6} \right\};$$

$$\widetilde{P}_2 = \left\{ \frac{0{,}868}{G_1}, \frac{0{,}596}{G_2}, \frac{0{,}552}{G_3}, \frac{0{,}936}{G_4}, \frac{0{,}813}{G_5}, \frac{0{,}938}{G_6} \right\};$$

$$\widetilde{P}_3 = \left\{ \frac{0{,}753}{G_1}, \frac{0{,}490}{G_2}, \frac{0{,}797}{G_3}, \frac{0{,}969}{G_4}, \frac{0{,}823}{G_5}, \frac{0{,}871}{G_6} \right\};$$

$$\widetilde{P}_4 = \left\{ \frac{0{,}667}{G_1}, \frac{0{,}409}{G_2}, \frac{0{,}456}{G_3}, \frac{0{,}933}{G_4}, \frac{0{,}909}{G_5}, \frac{0{,}938}{G_6} \right\}.$$

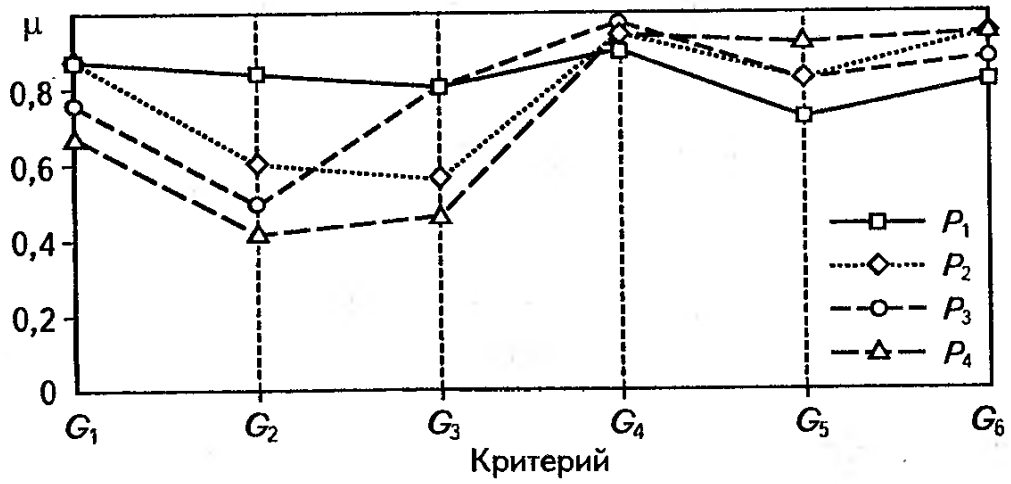


Рис. 2.22. Сравнение проектов $P_1 \div P_4$ с учетом важности критериев $G_1 \div G_6$

Графики функций принадлежности этих нечетких множеств изображены на рис. 2.22. Из рисунка видно, что по маловажным критериям $\widetilde{G}_4$ и $\widetilde{G}_6$ расстояние между проектами незначительное. В то же время по приоритетным критериям $\widetilde{G}_2$ и $\widetilde{G}_3$ разница между проектами существенная.

2.3.4. «ЧТО – ЕСЛИ». АНАЛИЗ ВАРИАНТОВ

При многокритериальном анализе часто возникает вопрос: «Что необходимо изменить в некоторой альтернативе, чтобы она стала наилучшей?» Для ответа на него надо знать, насколько чувствительно принятое решение к экспертным парным сравнениям. В подразделе описывается методика анализа чувствительности, предложенная в [18]. Идея методики состоит в определении, *что* будет за решение, *если* изменить одно из парных сравнений.

При изменении одного из парных сравнений вариантов необходимо обеспечить непротиворечивость остальных. Например, изменяется a_{ij} – уровень преимущества варианта P_i над вариантом P_j. Тогда в матрице парных сравнений необходимо изменить и элемент a_{ji}, так как они связаны зависимостью $a_{ji} = 1 / a_{ij}$. Кроме того, возможны изменения значений в уровнях преимущества варианта P_i над другими, которым соответствуют элементы a_{ir} и $a_{ri} = 1 / a_{ir} (r = \overline{1,\ k}, r \neq i, r \neq j)$ матрицы парных сравнений. Ниже рассматриваются четыре ситуации, когда новое значение элемента требует корректирования элемента a_{ij} матрицы парных сравнений.

1. Пусть преимущество варианта P_i над P_j сильнее, чем над P_r, т.е. $a_{ji} > a_{jr}$. Тогда в непосредственном парном сравнении вариант P_i не должен превосходить вариант P_r, следовательно $a_{ir} \leq 1$. Математически запишем это таким правилом:

$$\text{если } a_{ji} > a_{jr}, \text{ тогда } a_{ir} := \min(1,\ a_{ir}). \tag{2.38}$$

2. Пусть преимущество варианта P_i над P_j сильнее, чем преимущество варианта P_r над P_j, т.е. $a_{ij} > a_{rj}$. Тогда в непосредственном парном сравнении вариант P_r не должен превосходить P_i, следовательно $a_{ir} \geq 1$. Запишем это следующим правилом:

$$\text{если } a_{ij} > a_{rj}, \text{ тогда } a_{ir} := \max(1,\ a_{ir}). \tag{2.39}$$

3. Пусть вариант P_i лучше P_j ($a_{ij} > 1$), а вариант P_j лучше P_r ($a_{jr} > 1$). Тогда вариант P_i не будет лучше, чем P_r, следовательно $a_{ir} > 1$. При этом a_{ir} – уровень преимущества P_i над P_r, должен быть не меньше, чем a_{ij} и a_{jr}. Запишем это таким правилом:

$$\text{если } a_{ij} > 1 \text{ и } a_{jr} > 1, \text{ тогда } a_{ir} := \max(a_{ij},\ a_{jr},\ a_{ir}). \tag{2.40}$$

4. Пусть вариант P_i хуже P_j ($a_{ij} < 1$), а вариант P_j хуже P_r ($a_{jr} < 1$). Тогда вариант P_i должен быть хуже, чем P_r, следовательно $a_{ir} < 1$. При этом a_{ri} – уровень преимущества P_r над P_i, должен быть не меньше, чем $a_{ji} = 1 / a_{ij}$ и $a_{rj} = 1 / a_{jr}$. Запишем это следующим правилом:

$$\text{если } a_{ij} < 1 \text{ и } a_{jr} < 1, \text{ тогда } a_{ir} := \min(a_{ij},\ a_{jr},\ a_{ir}). \tag{2.41}$$

Ниже приводится пошаговая методика «Что – Если» анализа вариантов, использующая правила (2.38)–(2.41).

Шаг 1. Обозначить анализируемый вариант через P_i.

Шаг 2. Выявить критерий, по которому можно улучшить вариант P_i, и обозначить этот критерий через G_u.

Шаг 3. Определить вариант, с которым удобно сравнивать вариант P_i по критерию G_u. Обозначить этот вариант-аналог через P_j.

Шаг 4. Изменить по шкале Саати значение элемента a_{ij} в матрице парных сравнений $\mathbf{A}(G_u)$.

Шаг 5. Рассчитать значение элемента a_{ji} в матрице парных сравнений $\mathbf{A}(G_u)$ по формуле $a_{ji} = 1/a_{ij}$.

Шаг 6. Пересчитать значения элементов a_{ir} и a_{rj} $(r = \overline{1, k}, r \neq i, r \neq j)$ по правилам (2.38)–(2.41).

Шаг 7. Обеспечить обратную симметричность матрицы $\mathbf{A}(G_u)$.

Шаг 8. Рассчитать новые степени принадлежности нечеткого множества $\widetilde{G}_u$ по формулам (1.2) и (1.3).

Шаг 9. Провести многокритериальный анализ вариантов по формулам (2.34)–(2.35) и зафиксировать принятое решение.

Шаг 10. Повторить шаги 4–9 для всех возможных парных сравнений вариантов P_i и P_j по критерию G_u.

Пример 2.8. Данные для многокритериального анализа бренд-проектов прнведены в предыдущем подразделе. Необходимо установить, каким должен быть проект P_3, чтобы он стал наилучшим.

Проект P_3 имеет третий ранг; проекты P_1 и P_2 лучше его. Предположим, что можно улучшить проект P_3 по критерию G_2. Промоделируем, как повлияет на принятие решения изменение уровня преимущества проекта P_3 над P_1 с текущего значения «существенное преимущество P_1 над P_3» до оценки «слабое преимущество P_3 над P_1». Для этого поменяем значение элемента a_{31} матрицы парных сравнений $\mathbf{A}(G_2)$ с 1/5 на 1/4, 1/3, 1/2, 1, 2 и 3, и проведем расчеты по описанной выше методике. Результаты расчетов сведены в табл. 2.8 и изображены графиками зависимости решения от изменения парного сравнения a_{31} (рис. 2.23).

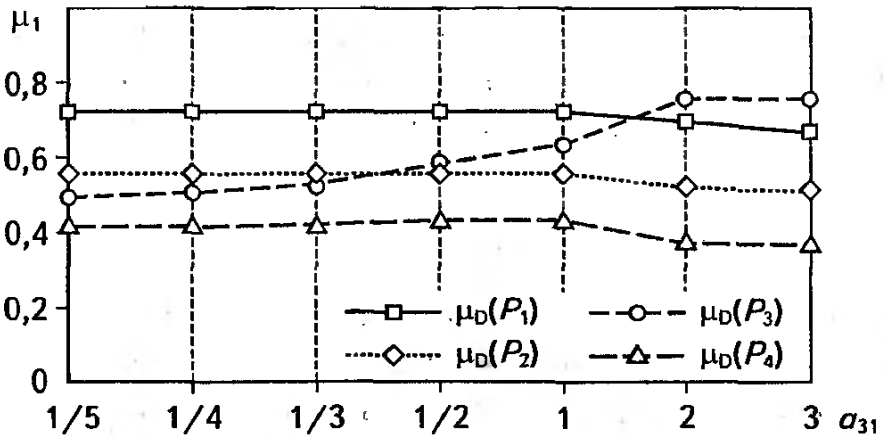


Рис. 2.23. Результаты «Что – Если» анализа бренд-проектов

Из рисунка видно, что проект P_3 станет вторым по рангу, когда по критерию G_2 преимущество P_1 над P_3 будет меньше слабого ($a_{31} > 1/3$). Проект P_3 станет наилучшим, когда он будет хоть чуть-чуть превосходить проект P_1 по критерию G_2 ($a_{31} > 1$).

Таблица 2.8

Расчеты зависимости принятия решения от изменения парного сравнения a_{31} по критерию G_2 (к примеру 2.8)

a_{31}	$A(G_2)$	$\tilde{G}_2$	$\tilde{D}$	D
1/5	$\begin{bmatrix} 1 & 3 & 5 & 7 \\ 1/3 & 1 & 2 & 3 \\ 1/5 & 1/2 & 1 & 2 \\ 1/7 & 1/3 & 1/2 & 1 \end{bmatrix}$	$\left\{\frac{0,59}{P_1}, \frac{0,22}{P_2}, \frac{0,12}{P_3}, \frac{0,07}{P_4}\right\}$	$\left\{\frac{0,717}{P_1}, \frac{0,552}{P_2}, \frac{0,490}{P_3}, \frac{0,409}{P_4}\right\}$	P_1
1/4	$\begin{bmatrix} 1 & 3 & 4 & 7 \\ 1/3 & 1 & 2 & 3 \\ 1/4 & 1/2 & 1 & 2 \\ 1/7 & 1/3 & 1/2 & 1 \end{bmatrix}$	$\left\{\frac{0,57}{P_1}, \frac{0,23}{P_2}, \frac{0,13}{P_3}, \frac{0,07}{P_4}\right\}$	$\left\{\frac{0,717}{P_1}, \frac{0,552}{P_2}, \frac{0,504}{P_3}, \frac{0,412}{P_4}\right\}$	P_1
1/3	$\begin{bmatrix} 1 & 3 & 3 & 7 \\ 1/3 & 1 & 2 & 3 \\ 1/3 & 1/2 & 1 & 2 \\ 1/7 & 1/3 & 1/2 & 1 \end{bmatrix}$	$\left\{\frac{0,55}{P_1}, \frac{0,23}{P_2}, \frac{0,15}{P_3}, \frac{0,07}{P_4}\right\}$	$\left\{\frac{0,717}{P_1}, \frac{0,552}{P_2}, \frac{0,522}{P_3}, \frac{0,415}{P_4}\right\}$	P_1
1/2	$\begin{bmatrix} 1 & 3 & 2 & 7 \\ 1/3 & 1 & 1 & 3 \\ 1/2 & 1 & 1 & 2 \\ 1/7 & 1/3 & 1/2 & 1 \end{bmatrix}$	$\left\{\frac{0,52}{P_1}, \frac{0,20}{P_2}, \frac{0,20}{P_3}, \frac{0,08}{P_4}\right\}$	$\left\{\frac{0,717}{P_1}, \frac{0,552}{P_2}, \frac{0,582}{P_3}, \frac{0,423}{P_4}\right\}$	P_1
1	$\begin{bmatrix} 1 & 3 & 1 & 7 \\ 1/3 & 1 & 1 & 3 \\ 1 & 1 & 1 & 2 \\ 1/7 & 1/3 & 1/2 & 1 \end{bmatrix}$	$\left\{\frac{0,45}{P_1}, \frac{0,21}{P_2}, \frac{0,26}{P_3}, \frac{0,08}{P_4}\right\}$	$\left\{\frac{0,717}{P_1}, \frac{0,552}{P_2}, \frac{0,631}{P_3}, \frac{0,428}{P_4}\right\}$	P_1
2	$\begin{bmatrix} 1 & 3 & 1/2 & 7 \\ 1/3 & 1 & 1/3 & 3 \\ 2 & 3 & 1 & 7 \\ 1/7 & 1/3 & 1/7 & 1 \end{bmatrix}$	$\left\{\frac{0,33}{P_1}, \frac{0,14}{P_2}, \frac{0,48}{P_3}, \frac{0,05}{P_4}\right\}$	$\left\{\frac{0,689}{P_1}, \frac{0,512}{P_2}, \frac{0,753}{P_3}, \frac{0,368}{P_4}\right\}$	P_3
3	$\begin{bmatrix} 1 & 3 & 1/3 & 7 \\ 1/3 & 1 & 1/3 & 3 \\ 3 & 3 & 1 & 7 \\ 1/7 & 1/3 & 1/7 & 1 \end{bmatrix}$	$\left\{\frac{0,30}{P_1}, \frac{0,13}{P_2}, \frac{0,52}{P_3}, \frac{0,05}{P_4}\right\}$	$\left\{\frac{0,660}{P_1}, \frac{0,505}{P_2}, \frac{0,753}{P_3}, \frac{0,364}{P_4}\right\}$	P_3

Глава 3. ПАКЕТ FUZZY LOGIC TOOLBOX

В вычислительную среду MATLAB интегрированы десятки пакетов прикладных инженерных и математических программ, одним из них является Fuzzy Logic Toolbox. В главе описывается пакет Fuzzy Logic Toolbox v.2.1 вычислительной системы MATLAB 7, предназначенный для проектирования и исследования систем на нечеткой логике. Материал главы организован следующим образом: вначале рассматриваются основные возможности пакета, затем показывается пошаговый процесс проектирования систем нечеткого вывода с применением GUI-модулей пакета и из командной строки, далее приводятся руководства пользователя по GUI-модулям, описываются демо-примеры и функции пакета. В конце главы рассматриваются форматы данных пакета и описывается взаимодействие Fuzzy Logic Toolbox с Simulink. При написании главы использовалась документация фирмы MathWorks Inc. [27].

Пред изучением главы читателям, у которых нет базовых навыков работы в среде MATLAB, рекомендуется ознакомиться с книгой «Начало работы с MATLAB», которая выставлена на сайте *http://matlab.expopnenta.ru* в разделе «MATLAB».

3.1. СТРУКТУРА И ВОЗМОЖНОСТИ ПАКЕТА

Пакет Fuzzy Logic Toolbox поддерживает все фазы разработки нечетких систем, включая синтез, исследование, проектирование, моделирование и внедренис в режиме реального времени. Встроенные GUI-модули пакета создают интуитивно понятную среду, обеспечивающую легкое продвижения по всем ступенькам проектирования нечетких систем. Функции пакета реализуют большинство современных нечетких технологий, включая нечеткий логический вывод, нечеткую кластеризацию и адаптивную нейро-нечеткую настройку (ANFIS). Fuzzy Logic Toolbox, как и все пакеты расширения системы MATLAB, открыт для пользователя: можно просмотреть алгоритмы, модифицировать исходный код, добавить собственные функции принадлежности или процедуры дефаззификации. Ключевыми особенностями пакета Fuzzy Logic Toolbox являются:

- специализированные GUI-модули для создания систем нечеткого вывода;
- реализация популярных алгоритмов нечеткого вывода Мамдани и Сугено;
- библиотека функций принадлежности;
- настройка функций принадлежности ANFIS-алгоритмом;
- экстракция нечетких правил с помощью кластеризации данных;
- возможность внедрения систем нечеткого вывода в Simulink через модуль «нечеткий контроллер»;
- Си-код алгоритмов нечеткого вывода, позволяющий использовать спроектированные нечеткие системы вне среды MATLAB.

Fuzzy Logic Toolbox включает следующие GUI-модули:

- Fuzzy Inference System Editor – редактор общих свойств системы нечеткого вывода. Позволяет установить количество входов и выходов системы, выбрать тип системы (Мамдани или Сугено), метод дефаззификации, реализации логических операций, а также вызвать другие GUI-модули, работающие с системами нечеткого вывода;
- Membership Function Editor – редактор функций принадлежности. Редактор выводит на экран графики функций принадлежности входных и выходных переменных. Позволяет выбрать количество термов для лингвистической оценки входных и выходных переменных, а также задать тип и параметры функций принадлежности каждого терма;
- Rule Editor – редактор нечеткой базы знаний. Позволяет задавать и редактировать нечеткие правила в лингвистическом, логическом и индексном форматах. Редактирование правил осуществляется выбором необходимого сочетания термов из меню;
- Rule Viewer – браузер нечеткого вывода. Визуализирует выполнение нечеткого вывода по каждому правилу, получение результирующего нечеткого множества и его дефаззификацию;
- Surface Viewer – браузер поверхности «входы – выход» нечеткой системы. Выводит графики зависимости выходной переменной от любых двух входных переменных;
- ANFIS Editor – редактор нейро-нечеткой сети. Позволяет синтезировать и настроить нейро-нечеткие сети по выборке данных «входы – выход». Для настройки используется алгоритм обратного распространения ошибки или его комбинация с методом наименьших квадратов;
- Findcluster – инструмент субтрактивной кластеризации по горному методу и нечеткой кластеризации по алгоритму нечетких *c*-средних. Позволяет найти центры кластеров данных, которые используются для экстракции нечетких правил.

Все GUI-модули, за исключением Findcluster, динамически обмениваются данными и могут быть вызваны друг из друга (рис. 3.1).

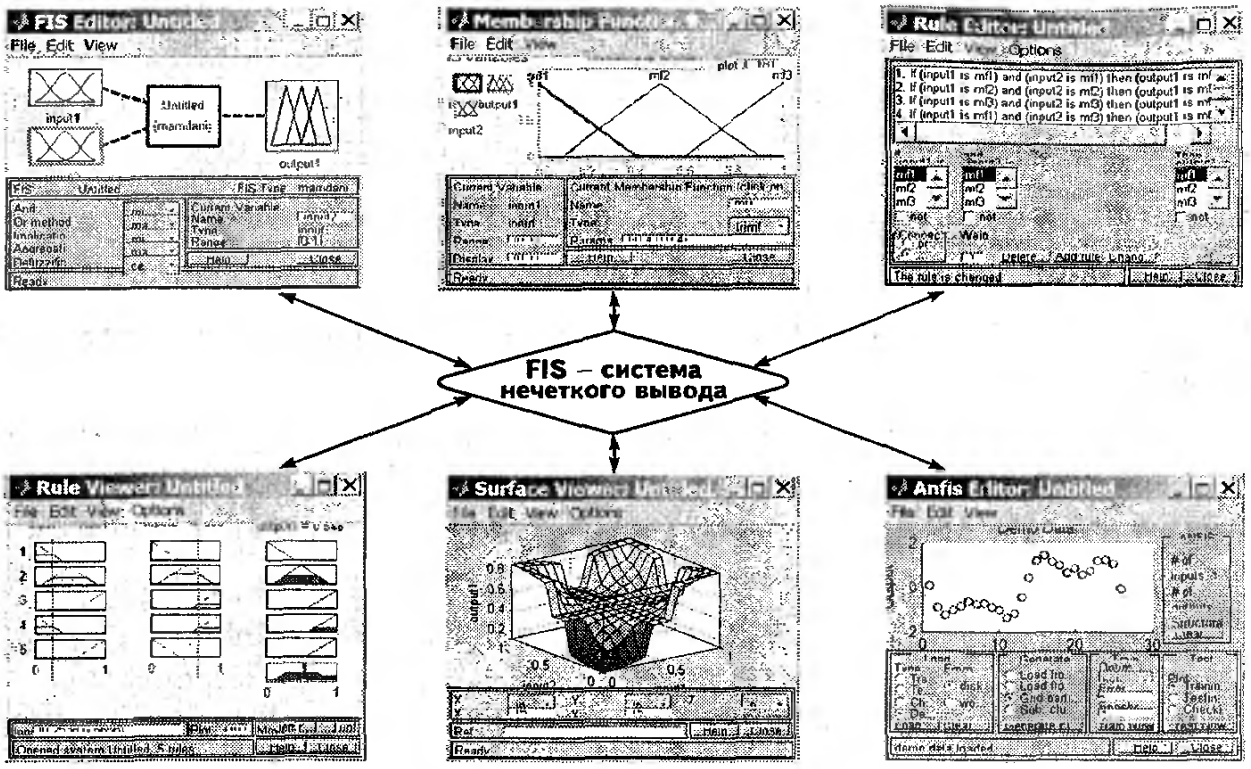


Рис. 3.1. Взаимодействие GUI-модулей нечеткого вывода

Кроме GUI-модулей, Fuzzy Logic Toolbox содержит библиотеку функций для разработки нечетких систем из командной строки, а также для написания программ, автоматизирующих проектирование и исследование нечетких систем. Ниже приведены названия и краткое описание основных функций пакета (аббревиатура FIS обозначает систему нечеткого вывода):

addmf	–	добавление функции принадлежности в FIS;
addrule	–	добавление правил в FIS;
addvar	–	добавление переменной в FIS;
anfis	–	настройка FIS типа Сугено с помощью технологии ANFIS;
convertfis	–	преобразование FIS-матрицы из Fuzzy Logic Toolbox v.1 в FIS-структуру для Fuzzy Logic Toolbox v.2;
defuzz	–	дефаззификация нечеткого множества;
discfis	–	дискретизация функций принадлежности нечетких термов из FIS;
distfcm	–	расчет расстояния по Евклиду;
dsigmf	–	функция принадлежности в виде разности двух сигмоидных функций;
evalfis	–	выполнение нечеткого логического вывода;
evalmf	–	вычисление значений произвольной функции принадлежности;
evalmmf	–	расчет степеней принадлежностей для нескольких функций принадлежностей;
fcm	–	кластеризация алгоритмом нечетких *c*-средних;
fuzarith	–	нечеткий калькулятор;
gauss2mf	–	двухсторонняя гауссова функция принадлежности;
gaussmf	–	гауссова функция принадлежности;
gbellmf	–	обобщенная колоколообразная функция принадлежности;

genfis1	–	генерирование из данных исходной FIS типа Сугено без применения кластеризации;
genfis2	–	генерирование из данных исходной FIS типа Сугено через субтрактивную кластеризацию;
genparam	–	генерирования начальных параметров функций принадлежности для ANFIS-обучения.
gensurf	–	вывод поверхности «входы – выход», соответствующей FIS;
getfis	–	получение свойств FIS;
initfis	–	генерирование исходной матрицы нечеткого разбиения;
isfis	–	проверка структуры данных системы нечеткого вывода;
mam2sug	–	преобразование FIS типа Мамдани в FIS типа Сугено;
mf2mf	–	пересчет параметров встроенных функций принадлежности различных типов;
newfis	–	создание новой FIS;
parsrule	–	вставка в FIS правил, заданных предложениями на естественном языке;
pimf	–	Пи-подобная функция принадлежности;
plotfis	–	вывод основных параметров FIS в виде графической схемы;
plotmf	–	вывод графиков функций принадлежности термов одной переменной;
probor	–	вероятностная реализация логической операции ИЛИ;
psigmf	–	функция принадлежности в виде произведения двух сигмоидных функций;
readfis	–	загрузка FIS из файла;
rmmf	–	удаление функции принадлежности нечеткого терма из FIS;
rmvar	–	удаление переменной из FIS;
setfis	–	назначение свойств FIS;
showfis	–	вывод на экран FIS-структуры в текстовом формате;
showrule	–	вывод базы знаний FIS;
sigmf	–	сигмоидная функция принадлежности;
smf	–	*s*-подобная функция принадлежности;
subclust	–	субтрактивная кластеризация;
sugmax	–	расчет диапазона изменения выходной переменной в FIS типа Сугено;
trapmf	–	трапециевидная функция принадлежности;
trimf	–	треугольная функция принадлежности;
writefis	–	сохранение FIS на диске;
zmf	–	*z*-подобная функция принадлежности.

Пакет Fuzzy Logic Toolbox позволяет внедрить разработанные системы нечеткого вывода в динамические модели пакета Simulink. Для этой цели служит Simulink-модуль Fuzzy Logic Controller – нечеткий контроллер. Для быстрого нечеткого вывода в пакете Simulink оптимизирован код функции sffis, возможности которой аналогичны функции evalfis. С использованием Real-Time Workshop можно сгенерировать эффективный код нечеткого вывода. Fuzzy Logic Toolbox со-

держит два десятка демо-примеров, иллюстрирующих возможности пакета по созданию нечетких систем в различных областях: идентификации, управлении, прогнозировании, обработке сигналов и др.

3.2. БЫСТРЫЙ СТАРТ

В разделе описывается пошаговый процесс проектирования систем нечеткого вывода в Fuzzy Logic Toolbox. Цель раздела – как можно быстрее подготовить читателя к началу работы с пакетом. Вначале рассматривается разработка нечетких систем Мамдани и Сугено с использованием GUI-модулей пакета. Затем описывается синтез из данных нечетких систем с помощью ANFIS-редактора и в режиме командной строки. Изложение материала ведется на примерах законченного проектирования нечетких систем.

3.2.1. РАЗРАБОТКА НЕЧЕТКОЙ СИСТЕМЫ ТИПА МАМДАНИ

Создадим систему нечеткого вывода типа Мамдани, которая моделирует зависимость $y = x_1^2 \sin(x_2 - 1)$ в области $x_1 \in [-7, 3]$, $x_2 \in [-4{,}4, 1{,}7]$. Проектирование осуществим на основе трехмерного изображения указанной зависимости (рис. 3.2), которое построено следующей программой:

```
%Построение графика функции y = x1^2*sin(x2-1)
%в области x1є[-7, 3] и x2є[-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')
```

Поверхности на рис. 3.2 поставим в соответствие следующие семь нечетких правил:

1) ЕСЛИ x_1 = «низкий» И x_2 = «низкий», ТО y = «высокий»;
2) ЕСЛИ x_1 = «низкий» И x_2 = «средний», ТО y = «низкий»;
3) ЕСЛИ x_1 = «низкий» И x_2 = «высокий», ТО y = «высокий»;
4) ЕСЛИ x_1 = «средний», ТО y = «средний»;
5) ЕСЛИ x_1 = «высокий» И x_2 = «низкий», ТО y = «выше среднего»;
6) ЕСЛИ x_1 = «высокий» И x_2 = «средний», ТО y = «ниже среднего»;
7) ЕСЛИ x_1 = «высокий» И x_2 = «высокий», ТО y = «выше среднего».

Проектирование нечеткой системы состоит в выполнении следующей последовательности шагов.

Шаг 1. Открыть FIS-редактор, напечатав слово `fuzzy` в командной строке. После этого появится новое графическое окно, показанное на рис. 3.3.

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

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

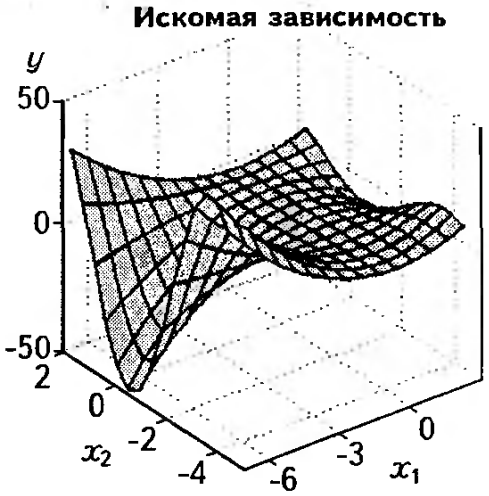


Рис. 3.2. График функции $y = x_1^2 \sin(x_2 - 1)$

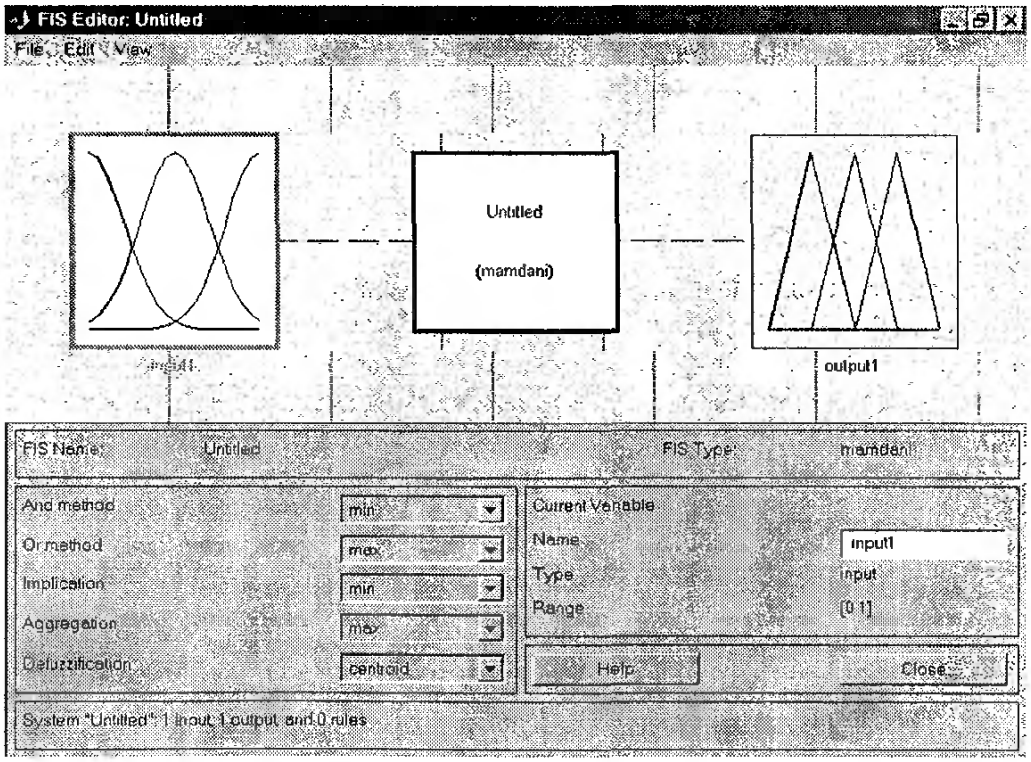


Рис. 3.3. FIS Editor

Шаг 4. Переименуем вторую входную переменную. Для этого щелкнем мышкой на блоке **input2**, введем новое обозначение `x2` в поле редактирования имени текущей переменной и нажмем <Enter>.

Шаг 5. Переименуем выходную переменную. Для этого щелкнем мышкой на блоке **output1**, введем новое обозначение `y` в поле редактирования имени текущей переменной и нажмем <Enter>.

Шаг 6. Зададим имя системы. Для этого в меню **File** выберем в подменю **Export** команду **To Disk...** и введем имя файла, например, `first`.

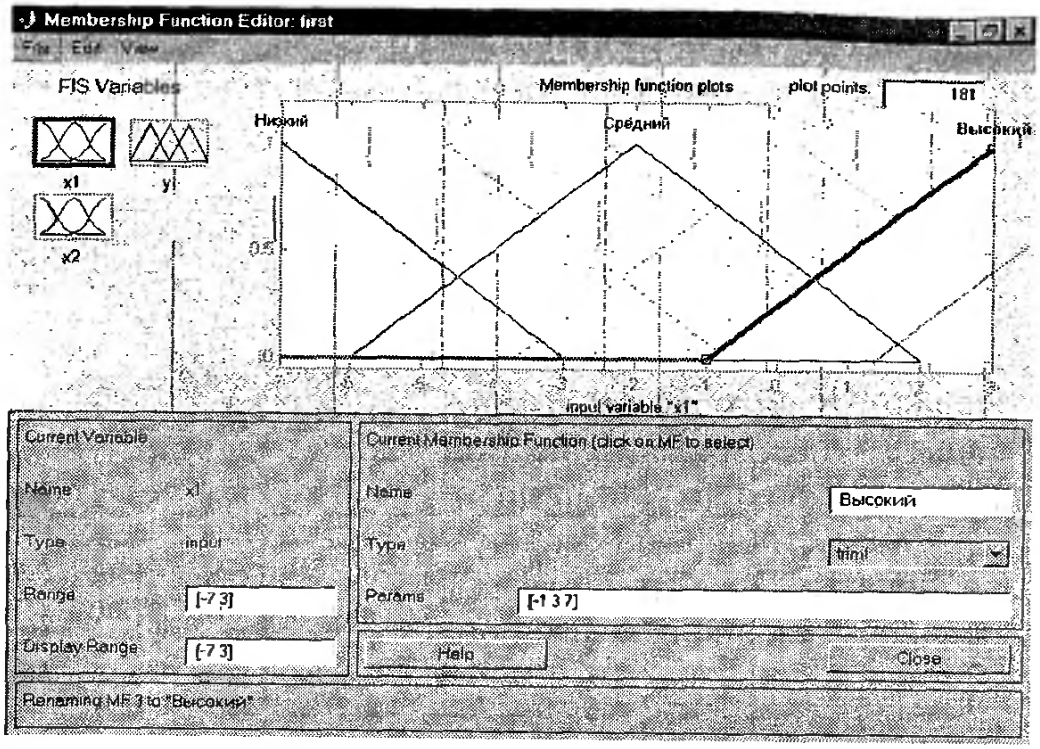


Рис. 3.4. Функции принадлежности переменной x_1 в Membership Function Editor

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

Шаг 8. Зададим диапазон изменения переменной x_1, напечатав `-7 3` в поле **Range** (рис. 3.4).

Шаг 9. Зададим функции принадлежности переменной x_1. Для лингвистической оценки этой переменной будем использовать три терма с треугольными функциями принадлежности. Эти функции установлены по умолчанию, поэтому переходим к следующему шагу.

Шаг 10. Зададим наименования термов переменной x_1. Для этого щелкнем мышкой по графику первой функции принадлежности (см. рис. 3.4). График активной функции принадлежности выделяется красной жирной линией. Затем введем наименование терма `Низкий` в поле **Name** и нажмем <Enter>. Щелкнем мышкой по графику второй функции принадлежности, введем наименование терма `Средний` в поле **Name** и нажмем <Enter>. Щелкнем мышкой по графику третьей функции принадлежности, введем наименование терма `Высокий` в поле **Name** и нажмем <Enter>. В результате получим графическое окно, изображенное на рис. 3.4.

Шаг 11. Зададим функции принадлежности переменной x_2. Для этого активизируем переменную x_2 щелчком мышкой по блоку **x2**. Зададим диапазон изменения переменной x_2. Для этого напечатаем `-4.4 1.7` в поле **Range** (рис. 3.5) и нажмем <Enter>. Для лингвистической оценки этой переменной будем использовать три терма с треугольными функциями принадлежности. Они установлены по умолчанию, поэтому переходим к следующему шагу.

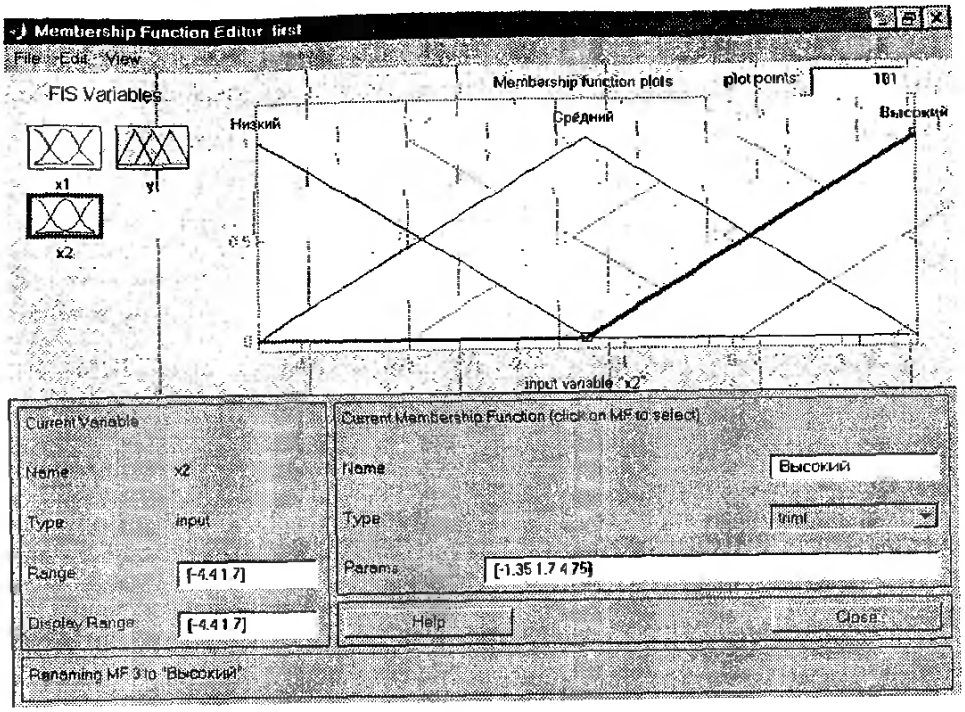


Рис. 3.5. Функции принадлежности переменной x_2 в Membership Function Editor

Шаг 12. По аналогии *с шагом 10* зададим следующие наименования термов переменной x_2: `Низкий`, `Средний`, `Высокий`. В результате получим графическое окно, изображенное на рис. 3.5.

Шаг 13. Зададим функции принадлежности переменной *y*. Для лингвистической оценки этой переменной будем использовать пять термов с гауссовыми функциями принадлежности. Для этого щелчком мыши по блоку **y** активизируем переменную *y*. Зададим диапазон изменения переменной *y*. Для этого напечатаем `-50 50` в поле **Range** (рис. 3.6) и нажмем <Enter>. Затем в меню **Edit** выберем команду

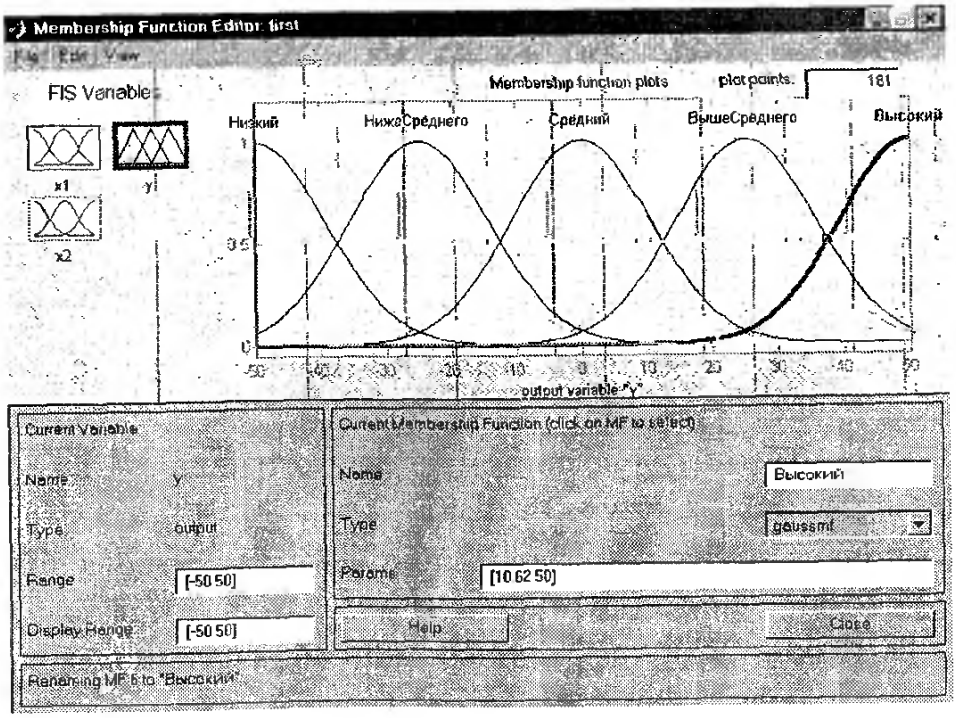


Рис. 3.6. Функции принадлежности переменной *y* в Membership Function Editor

Remove All MFs для удаления установленных по умолчанию функций принадлежности. После этого в меню **Edit** выберем команду **Add MFs...**. В появившемся диалоговом окне выберем тип функции принадлежности **gaussmf** в поле **MF type** и пять термов в поле **Number of MFs**. После ввода функций принадлежности редактор активизирует первую входную переменную, поэтому для продолжения работы щелкнем мышкой по пиктограмме **y**.

Шаг 14. По аналогии *с шагом 10* зададим следующие наименования термов переменной *y*: `Низкий`, `Ниже среднего`, `Средний`, `Выше среднего`, `Высокий`. В результате получим графическое окно, изображенное на рис. 3.6.

Шаг 15. Перейдем в редактор базы знаний RuleEditor. Для этого в меню **Edit** выберем команду **Rules...**.

Шаг 16. Для ввода правила выбираем в меню соответствующую комбинацию термов и нажимаем кнопку **Add rule**. На рис. 3.7 изображено окно редактора базы знаний после ввода всех семи правил. В конце правил в скобках указаны весовые коэффициенты.

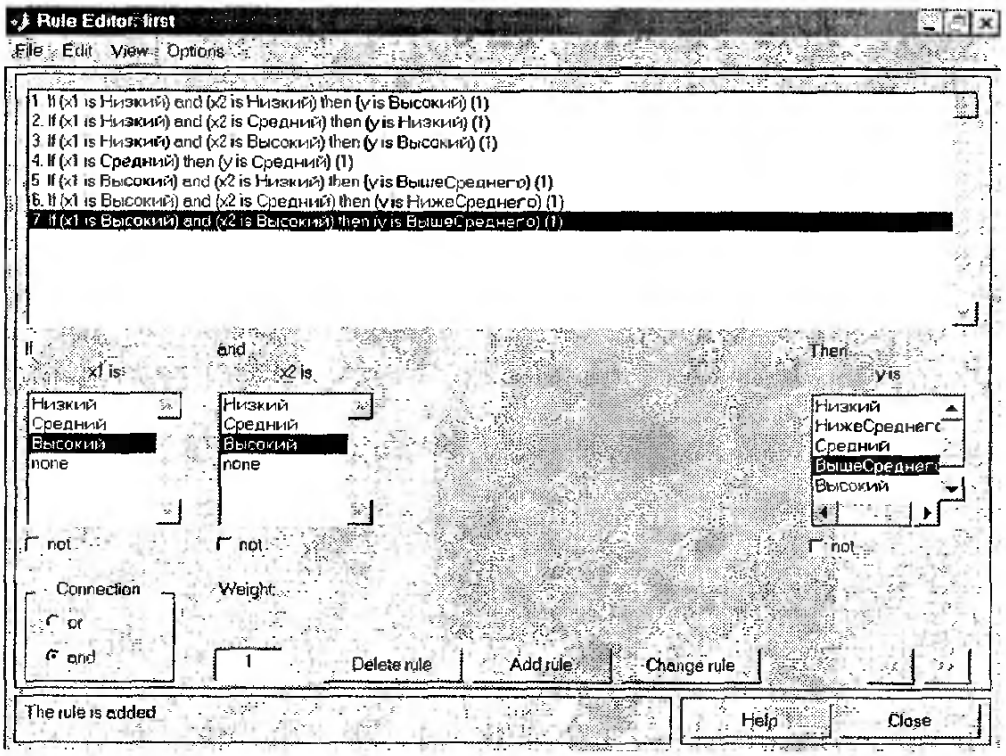


Рис. 3.7. Нечеткая база знаний Мамдани в Rule Editor

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

На рис. 3.8 приведено окно визуализации нечеткого вывода. Окно активизируется командой **Rules** меню **View**. В поле **Input** указываются значения входных переменных, для которых выполняется нечеткий логический вывод.

На рис. 3.9 приведена поверхность «входы – выход», соответствующая синтезированной нечеткой системе. Окно выводится по команде **Surface** меню **View**.

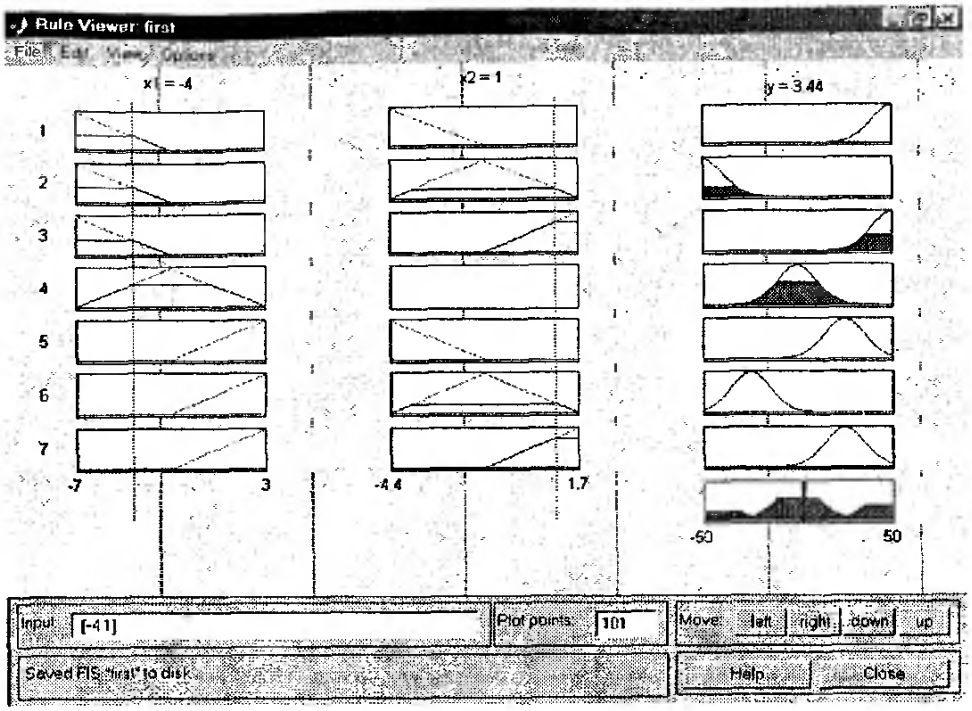


Рис. 3.8. Визуализация нечеткого вывода Мамдани в Rule Viewer

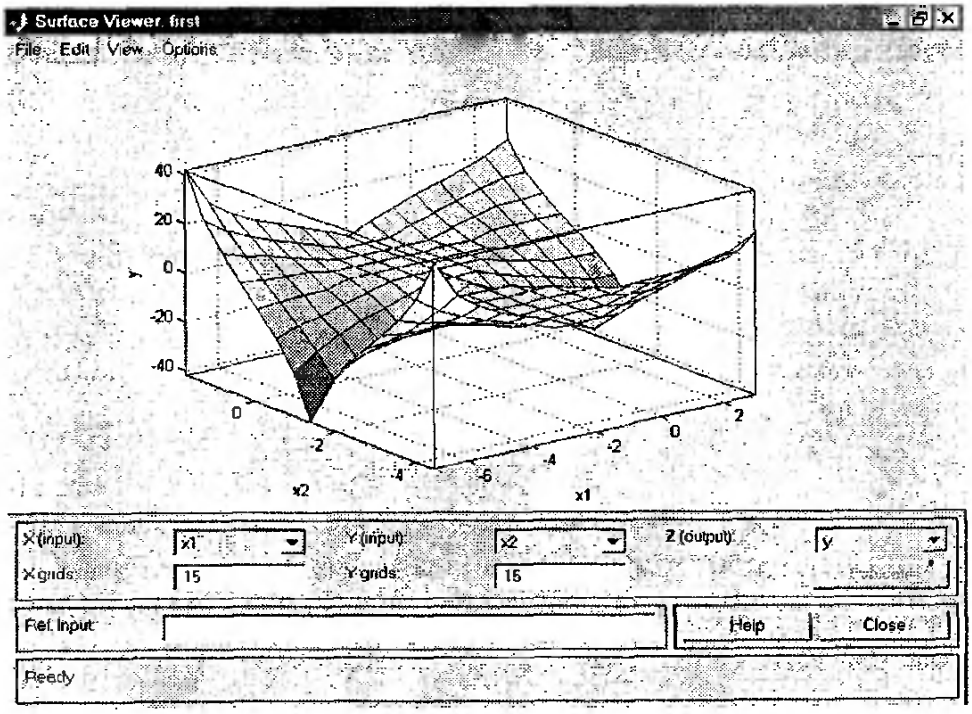


Рис. 3.9. Поверхность «входы – выход» для базы знаний Мамдани в Surface Viewer

Сравнивая поверхности на рис. 3.2 и на рис. 3.9, можно сделать вывод, что нечеткие правила описывают особенности моделируемой нелинейной зависимости.

3.2.2. РАЗРАБОТКА НЕЧЕТКОЙ СИСТЕМЫ ТИПА СУГЕНО НА ОСНОВЕ ЭКСПЕРТНЫХ ЗНАНИЙ

Создадим систему нечеткого вывода типа Сугено, моделирующую зависимость $y = x_1^2 \sin(x_2 - 1)$ в области $x_1 \in [-7, 3]$, $x_2 \in [-4{,}4, 1{,}7]$. График зависимости приведен на рис. 3.2. Смоделируем эту зависимость следующей базой знаний:

1) ЕСЛИ x_1 = «низкий» И x_2 = «низкий», ТО $y = 50$;
2) ЕСЛИ x_1 = «низкий» И x_2 = «средний», ТО $y = 4x_1 - x_2$;
3) ЕСЛИ x_1 = «низкий» И x_2 = «высокий», ТО $y = 50$;
4) ЕСЛИ x_1 = «средний», ТО $y = 0$;
5) ЕСЛИ x_1 = «высокий» И x_2 = «низкий», ТО $y = 2x_1 - 2x_2 - 3$;
6) ЕСЛИ x_1 = «высокий» И x_2 = «высокий», ТО $y = 2x_1 + 2x_2 + 1$.

Создание системы нечеткого вывода типа Сугено состоит в выполнении следующей последовательности шагов.

Шаг 1. Загрузим FIS-редактор, напечатав слова `fuzzy` в командной строке. После этого откроется графическое окно, показанное на рис. 3.3.

Шаг 2. Выберем тип системы. Для этого в меню **File** выбираем в подменю **New FIS...** команду **Sugeno**.

Шаг 3. Добавим вторую входную переменную. Для этого в меню **Edit** выберем команду **Add input**.

Шаг 4. Переименуем входы и выход системы. Для этого выполним *шаги 3–5* предыдущего алгоритма.

Шаг 5. Зададим имя системы. Для этого в меню **File** выберем в подменю **Export** команду **To disk** и введем имя файла, например, `second`.

Шаг 6. Зададим терм-множества и функции принадлежности входных переменных. Для этого выполним *шаги 7–12* предыдущего алгоритма.

Шаг 7. Зададим заключения правил. Для этого щелчком мыши по блоку **y** активизируем переменную *y*. В правом верхнем угле появилось обозначение трех функций принадлежности, каждая из которых соответствует одной линейной зависимости между входами и выходом. В базе знаний Сугено указаны пять различных зависимостей. Поэтому добавим еще два заключения правил, выбрав из меню **Edit** команду **Add Mfs...** Затем в появившемся диалоговом окне в поле **Number of MFs** выберем **2** и нажмем кнопку **OK**.

Шаг 8. Зададим наименования и параметры линейных зависимостей. Для этого щелкнем мышкой по наименованию первого заключения **mf1**. Затем в поле **Name** печатаем наименование зависимости, например, `50`, и устанавливаем тип зависимости – константа через опцию **Constant** в меню **Type**. После этого введем значение параметра `50` в поле **Params**.

Аналогично, для второго заключения **mf2** введем наименование, например, `4x1-x2`, укажем линейный тип зависимости «входы – выход» через опцию **Linear** в меню **Type** и введем параметры зависимости `4 -1 0` в поле **Params**. Для линейной зависимости порядок параметров следующий: первый параметр – коэффициент при первой переменной, второй – при второй и т.д., и 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, `2x1-2x2-3`, укажем линейный тип зависимости и введем параметры `2 -2 -3`.

Для пятого заключения **mf5** введем наименование, например, `2x1+2x2+1`, укажем линейный тип зависимости и введем параметры `2 2 1`. В результате получим графическое окно, изображенное на рис. 3.10.

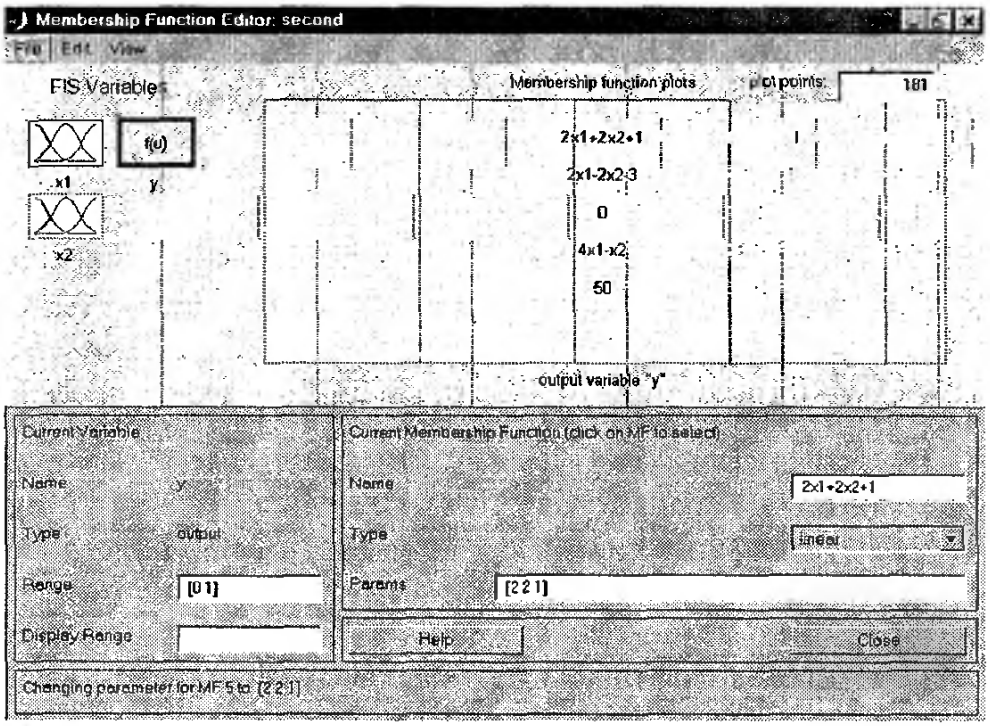


Рис. 3.10. Заключения правил Сугено в Membership Function Editor

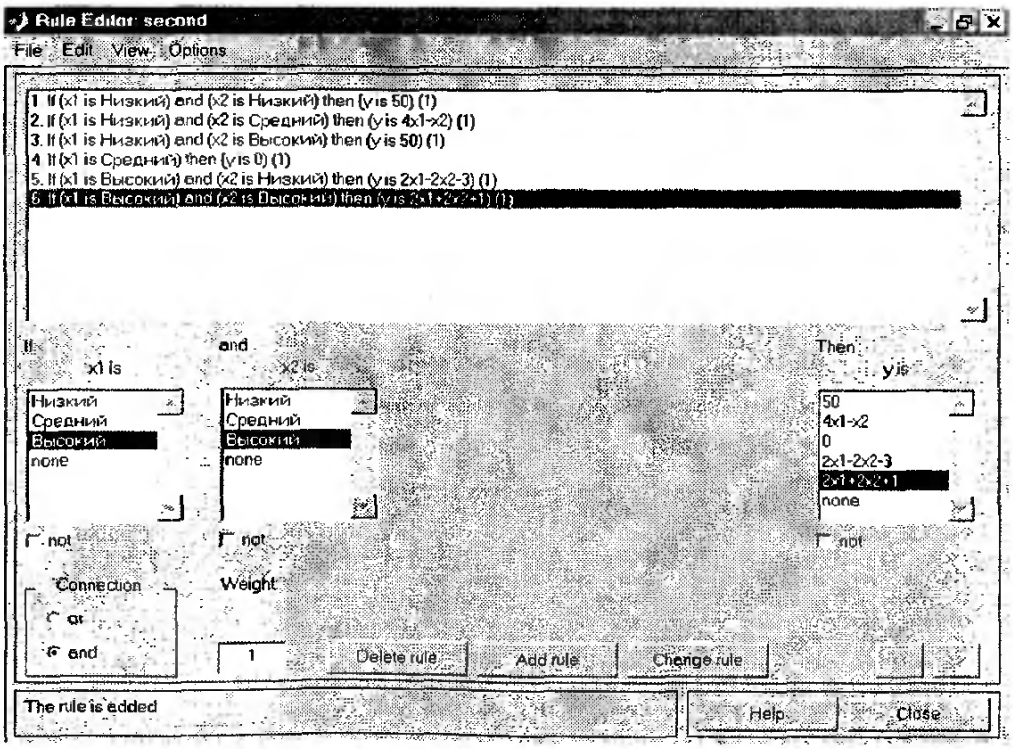


Рис. 3.11. Нечеткая база знаний Сугено в Rule Editor

Шаг 9. Перейдем в редактор базы знаний Rule Editor через команду **Rules...** меню **Edit**. Для ввода правил необходимо выбрать соответствующую комбинацию посылок и заключений правил и нажать кнопку **Add rule**. На рис. 3.11 изображено окно редактора базы знаний после ввода всех шести правил.

На рис. 3.12 приведено окно визуализации нечеткого вывода. Окно активизируется командой **Rules** меню **View**. На рис. 3.13 изображена поверхность «входы – выход», соответствующая синтезированной нечеткой системе. Окно выводится по команде **Surface** меню **View**. Сравнивая поверхности на рис. 3.2, рис. 3.9 и

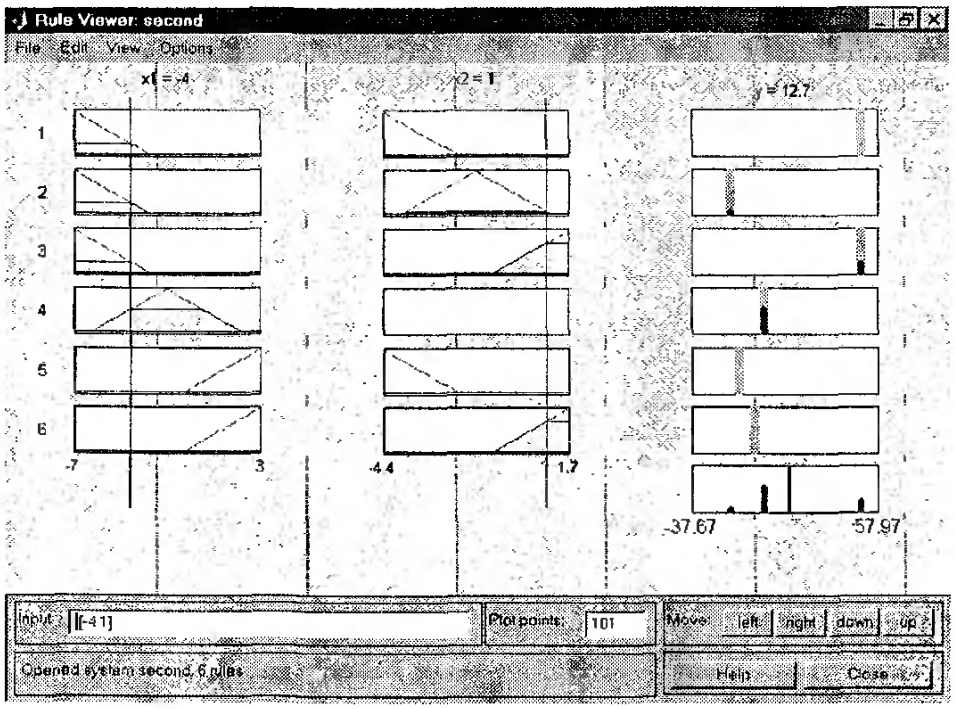


Рис. 3.12. Визуализация нечеткого вывода Сугено в Rule Viewer

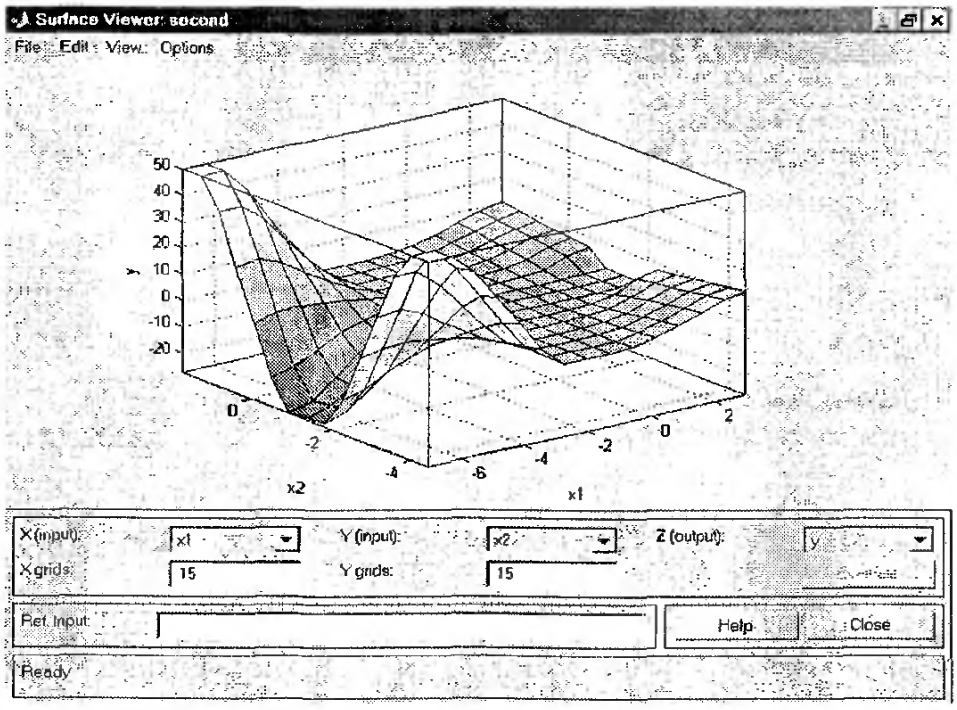


Рис. 3.13. Поверхность «входы – выход» для базы знаний Сугено в Surface Viewer

рис. 3.13, можно сделать вывод, что нечеткие правила достаточно хорошо описывают сложную нелинейную зависимость. Модель типа Сугено более точная, однако подобрать подходящие заключения правил не всегда просто.

3.2.3. ЭКСТРАКЦИЯ ИЗ ДАННЫХ НЕЧЕТКОЙ СИСТЕМЫ СУГЕНО С ПОМОЩЬЮ ANFIS-РЕДАКТОРА

В подразделе рассматривается процедура автоматического построения нечеткой системы типа Сугено из экспериментальных данных. Материал излагается на примере моделирования зависимости ширины лепестка ириса от длины чашелистика и длины лепестка.

Экспериментальные данные записаны в файле iris.dat, который содержит результаты измерения 150 ирисов. Ширина лепестка записана в четвертом столбце, длина чашелистика – в первом столбце, а длина лепестка – в третьем. Зависимость между этими признаками показана на рис. 3.14. Точки соответствуют цветкам. Поверхность получена аппроксимацией трехмерных данных командой `griddata`.

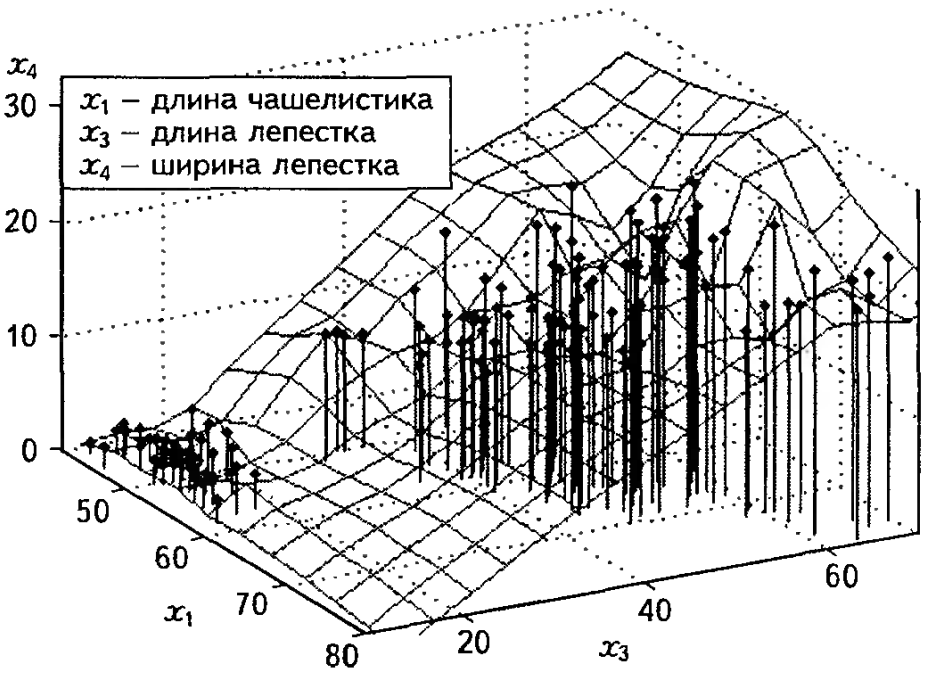


Рис. 3.14. Зависимость ширины лепестка ириса от длины чашелистика и длины лепестка

В обучающую выборку включим 120 ирисов, а в тестовую – 30. Для этого выполним следующие команды, по которым в рабочей области формируются обучающая выборка (`tr_set`) и тестовая выборка (`test_set`):

```
ir = load('iris.dat');
iris_number = length(ir);
test_index = [6:10:iris_number 7:10:iris_number];
tr_index = setdiff(1:iris_number, test_index);
tr_set = ir(tr_index, [1 3 4]);
test_set = ir(test_index, [1 3 4]).
```

Теперь командой `anfisedit` загрузим ANFIS-редактор, в котором сгенерируем и обучим нечеткую систему типа Сугено. Для загрузки обучающей выборки в окне ANFIS-редактора (рис. 3.15) в поле **Load data** выберем опции **Training** (меню **Type:**) и **worksp.** (меню **From:**) и нажмем кнопку **Load Data...**. В результате появится окно (рис. 3.16), в котором напечатаем идентификатор обучающей выборки `tr_set` и нажмем **OK**. Для загрузки тестовой выборки установим опцию **Testing** в меню **Type**, нажмем **Load Data...** и в появившемся окне напечатаем `test_set`. После этих действий ANFIS-редактор будет выглядеть так, как показано на рис. 3.17. Данные обучающей выборки показаны окружностями, а тестовой выборки – точками.

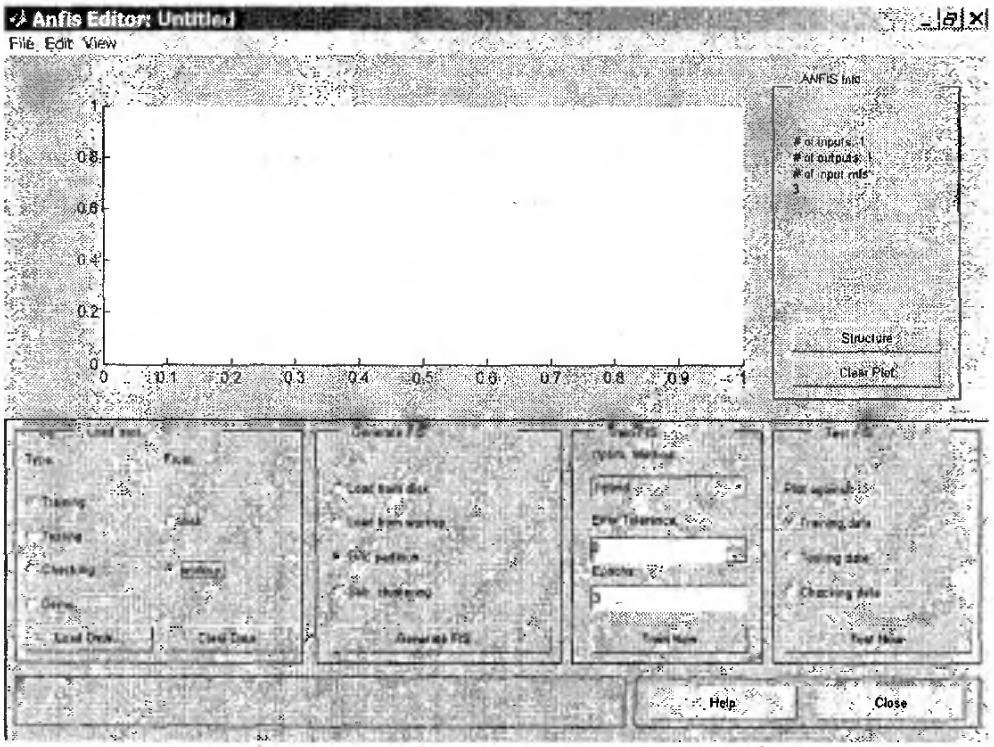


Рис. 3.15. Основное окно ANFIS-редактора

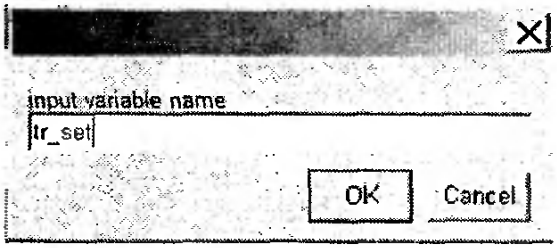


Рис. 3.16. Окно ввода имени переменной в ANFIS-редакторе

Создадим исходную систему нечеткого вывода, нажав кнопку **Generate FIS...**. По умолчанию используется простой метод решетчатого разбиения (Grid partition), согласно которому функции принадлежности нечетких термов равномерно распределяются внутри диапазона изменения данных. База знаний содержит все возможные варианты правил. Коэффициенты в заключениях правил принимаются равными нулю.

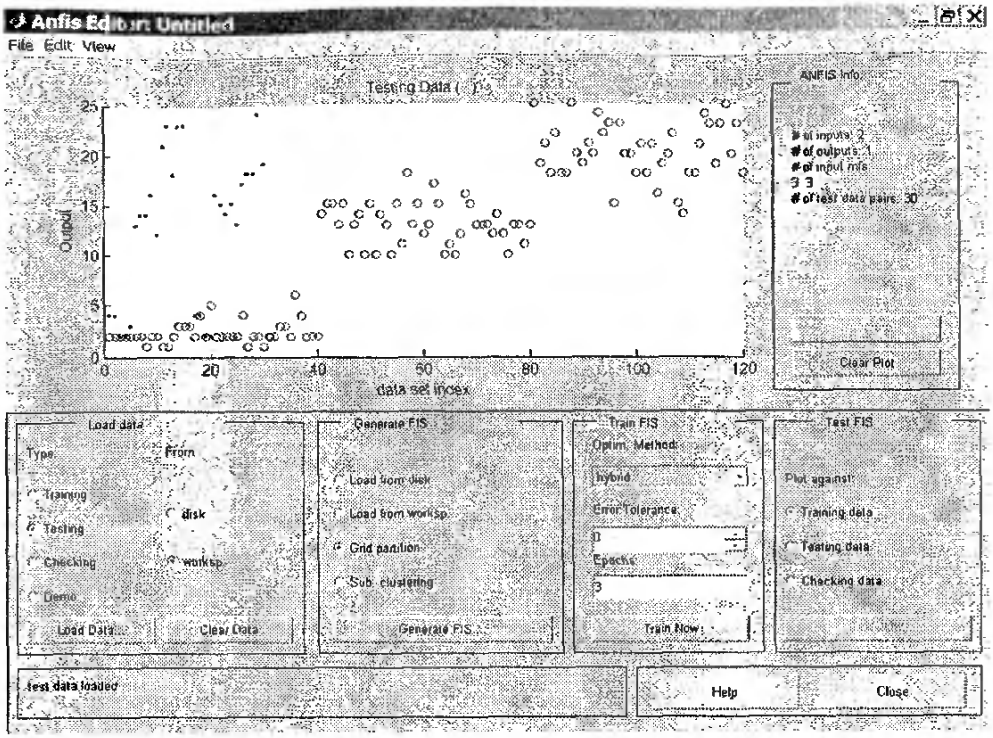


Рис. 3.17. ANFIS-редактор после загрузки обучающей и тестовой выборок

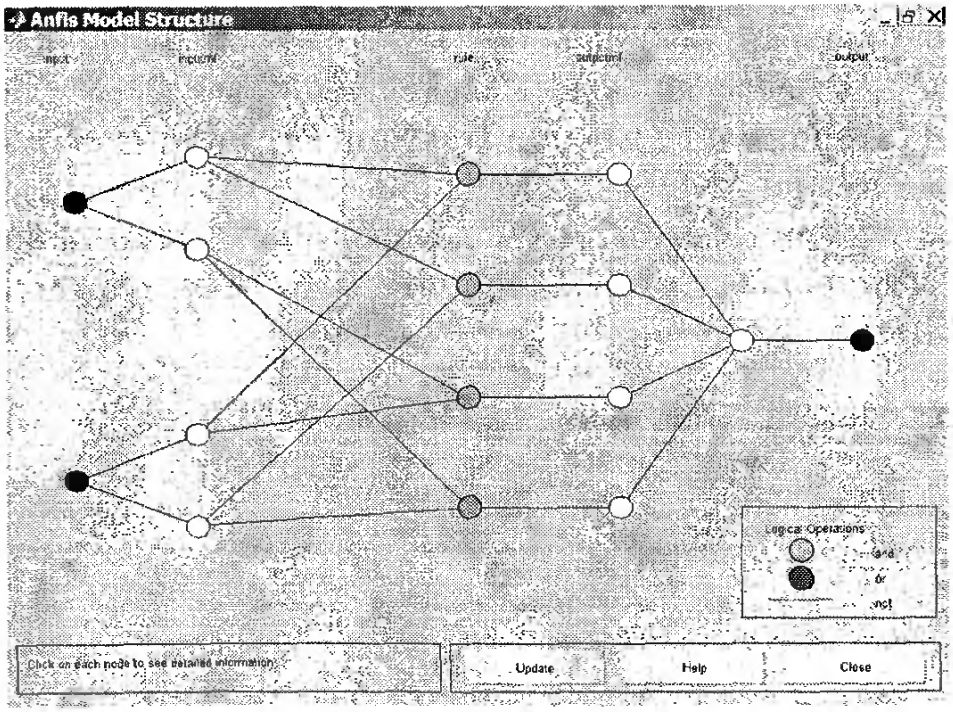


Рис. 3.18. Архитектура синтезированой нейро-нечеткой сети

Теперь с нечеткой системой можно работать, используя GUI-модули пакета Fuzzy Logic Toolbox. Вызов этих модулей происходит через меню **Edit** и **View**. Для изображения синтезированной системы в виде нейро-нечеткой сети (рис. 3.18) необходимо нажать кнопку **Structure** в поле **ANFIS Info**.

Проведем обучение нечеткой системы на протяжении 25 итераций. Для этого в поле **Epochs**: меню **Train FIS** напечатаем 25 и нажмем кнопку **Train Now**. Динамика обучения выводится в основном окне ANFIS-редактора (рис. 3.19): ось абсцисс соответствует итерациям алгоритма обучения, а ось ординат – зна-

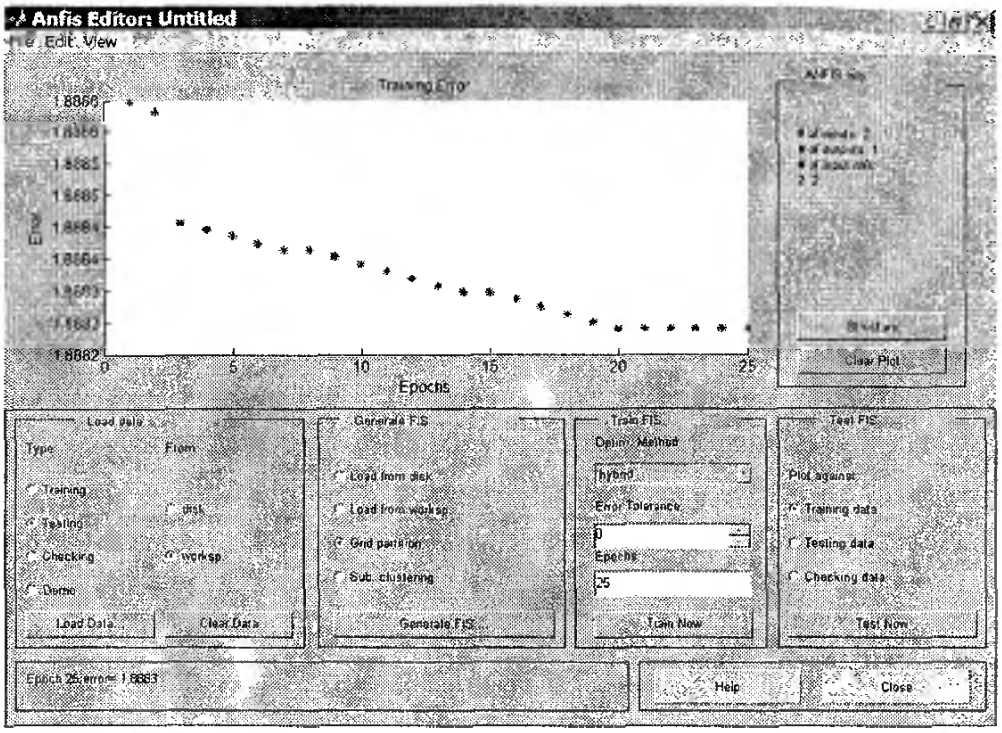


Рис. 3.19. Динамика обучения нечеткой системы

чениям средней квадратичной ошибки. После настройки ошибка обучения составляет 1,89. Для проверки модели на тестовой выборке установим в поле **Test FIS** опцию **Testing data** и нажмем **Test Now**. Результаты тестирования выводятся в основном окне ANFIS-редактора (рис. 3.20). Экспериментальные данные показаны точками, а результаты моделирования – звездочками. Ошибка на тестовой выборке составляет 1,87. Сохранить на диске нечеткую систему можно через меню **File**.

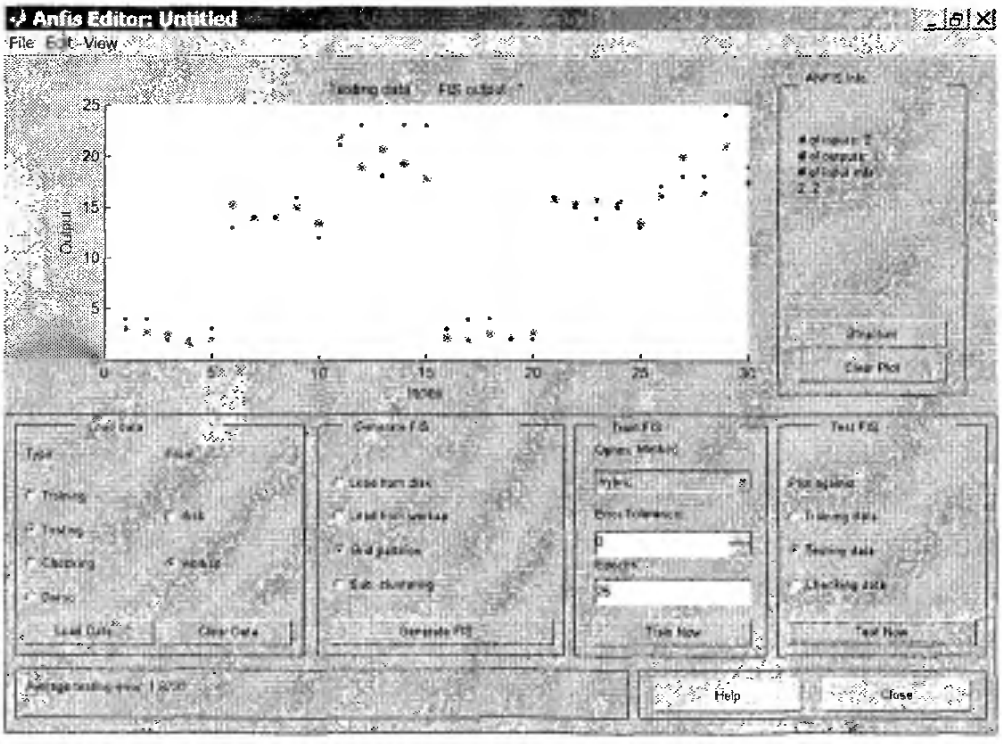


Рис. 3.20. Тестирование нечеткой системы

Сравним результаты нечеткой идентификации с линейной и квадратичной регрессиями. Идентификацию параметров линейной модели $x_4 = a_0 + a_1 x_1 + a_2 x_3$ осуществим следующими командами:

```
N = length(tr_set);
NN = length(test_set);
disp('Линейная модель:')
X = [ones(N,1) tr_set(:, 1) tr_set(:, 2)];
A = X\tr_set(:, 3)
%Расчет невязки на обучающей выборке
Out_model = X*A;
rmg_tr = norm(tr_set(:, 3)-Out_model)/sqrt(N)
%Расчет невязки на тестовой выборке
X_test = [ones(NN,1) test_set(:, 1) test_set(:, 2)];
Out_model = X_test*A;
rmg_ch = norm(test_set(:, 3)-Out_model)/sqrt(NN).
```

В результате получаем модель:

$$x_4 = 0,373 - 0,098x_1 + 0,458x_3,$$

ошибки которой на обучающей и тестовой выборке составляют 2,02 и 2,08 соответственно.

Для идентификации параметров нелинейной модели $x_4 = a_0 + a_1 x_1 + a_2 x_3 + a_3 x_1^2 + a_4 x_3^2 + a_5 x_1 x_3$ выполним следующие команды:

```
disp('Квадратичная модель:')
X = [ones(N, 1) tr_set(:, 1) tr_set(:, 2) tr_set(:, 1).^2...
tr_set(:, 2).^2 tr_set(:, 1).*tr_set(:, 2)];
A = X\tr_set(:, 3)
%Расчет невязки на обучающей выборке
Out_model = X*A;
rmg_tr = norm(tr_set(:, 3)-Out_model)/sqrt(N)
%Расчет невязки на тестовой выборке
X_test = [ones(NN, 1) test_set(:, 1) test_set(:, 2)...
test_set(:, 1).^2 test_set(:, 2).^2 test_set(:, 1).*test_set(:, 2)];
Out_model = X_test*A;
rmg_ch = norm(test_set(:, 3)-Out_model)/sqrt(NN)
```

В результате получаем модель:

$$x_4 = -25,564 + 0,87x_1 + 0,314x_3 - 0,008x_1^2 + 0,004x_3^2 - 0,002x_1x_3,$$

ошибки которой на обучающей и тестовой выборках составляют 1,94 и 1,99 соответственно. Таким образом, нечеткая модель точнее линейной и квадратичной регрессий описывает зависимость ширины лепестка ириса от длины чашелистика и длины лепестка.

3.2.4. ЭКСТРАКЦИЯ НЕЧЕТКОЙ СИСТЕМЫ В РЕЖИМЕ КОМАНДНОЙ СТРОКИ

В подразделе рассматривается двухэтапная процедура проектирования нечеткой системы Сугено. На первом этапе синтезируются нечеткие правила из экспериментальных данных с использованием субтрактивной кластеризации. На втором этапе настраиваются параметры нечеткой модели с помощью ANFIS-алгоритма. Материал излагается на примере построения нечеткой системы, моделирующей зависимость числа пригородных автомобильных поездок (y) от следующих демографических показателей: количество жителей (x_1); количество домов (x_2); количество автомобилей (x_3); средний уровень доходов на один дом (x_4); уровень занятости населения (x_5).

Для загрузки данных напечатаем команду `tripdata` , по которой в рабочей области создается обучающая выборка из 75 пар данных «входы – выход» (переменные `datin` и `datout`) и тестовая выборка из 25 данных (переменные `chkdatin` и `chkdatout`). Экспериментальные данные собраны из 100 транспортных зон округа Ньюкасл. На рис. 3.21 приведена обучающая выборка в виде однофакторных зависимостей «входы – выход». Рисунок построен следующими командами:

```
for i = 1:5
    subplot(3, 5, i)
    plot(datin(:, i), datout, 'kx')
    xlabel(['x_' num2str(i)])
    ylabel('y');
end
```

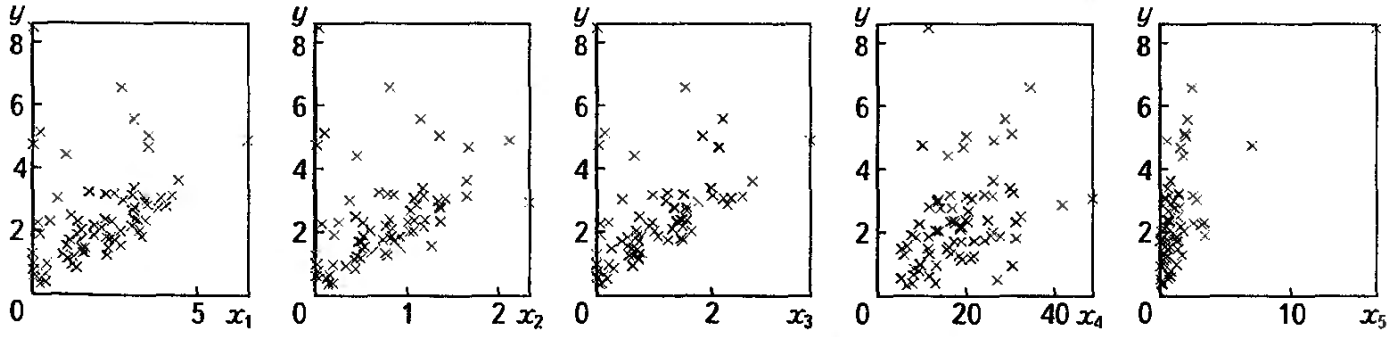


Рис. 3.21. Обучающая выборка для идентификации зависимости числа автомобильных поездок от демографических показателей района

С помощью функции genfis2 синтезируем из данных нечеткую модель Сугено с использованием субтрактивного метода кластеризации. Это быстрый однопроходный метод без итерационных процедур оптимизации. При вызове этой функции необходимо указать радиусы кластеров, которые определяют, как далеко от центра кластера могут находиться его элементы. Значения радиусов должны быть из диапазона [0, 1]. Как правило, хорошие нечеткие базы знаний получаются когда радиусы находятся в диапазоне [0,2, 0,5]. Радиусы кластеров задаются третьим

аргументом функции genfis2. Будем считать, что в кластерном анализе все координаты являются равноважными, поэтому значение этого аргумента можно задать скаляром. Следующая команда вызывает функцию genfis2 при значении радиусов кластера, равных 0,5:

```
fis = genfis2(datin, datout, 0,5)
```

В результате синтезируется нечеткая модель Сугено первого порядка с тремя правилами. Для вычисления ошибки моделирования на обучающей выборке вызовем следующие команды:

```
fuzout = evalfis(datin, fis);
trnRMSE = norm(fuzout-datout)/sqrt(length(fuzout))
```

В результате получим, что значение невязки (2.3) равно: **trnRMSE = 0,5276**. Теперь проверим, как работает модель вне точек обучения, т.е. на тестовой выборке. Для этого выполним аналогичные команды для тестовой выборки:

```
chkfuzout = evalfis(chkdatin, fis);
chkRMSE = norm(chkfuzout-chkdatout)/sqrt(length(chkfuzout))
```

Значение ошибки на тестовой выборке равно: **chkRMSE = 0,6170**. Неудивительно, что ошибка на тестовой выборке больше, чем на обучающей. Сравним экспериментальные данные из тестовой выборки с результатами нечеткого моделирования, применяя такие команды:

```
plot(chkdatout)
hold on
plot(chkfuzout, 'o')
hold off
legend('экспериментальные данные', 'нечеткое моделирование')
```

Результаты выполнения этих команд показаны на рис. 3.22 *а*. Видно, что нечеткая модель описывает тенденцию в данных. Однако в отдельных случаях расхождения между экспериментальными данными и результатами моделирования значительные.

Попытаемся улучшить модель с помощью ANFIS-обучения. Зададим относительно небольшое количество итераций обучения – 50. Для обучения будем использовать только обучающую выборку с последующей проверкой настроенной нечеткой модели на тестовой выборке. Обучение осуществим командой `fis2=anfis([datin datout], fis, [50 0 0,1])`.

Рассчитаем ошибки моделирования на обучающей и тестовой выборках после обучения:

```
fuzout2 = evalfis(datin, fis2);
trnRMSE2 = norm(fuzout2-datout)/sqrt(length(fuzout2))
chkfuzout2 = evalfis(chkdatin, fis2);
chkRMSE2 = norm(chkfuzout2-chkdatout)/sqrt(length(chkfuzout2))
```

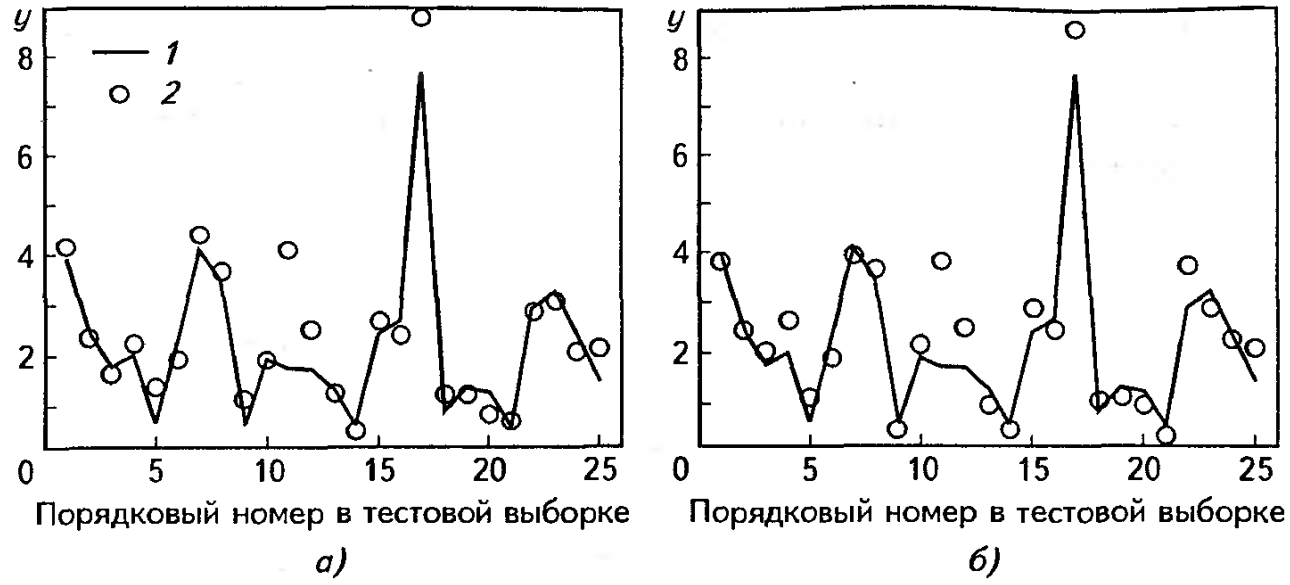


Рис. 3.22. Тестирование нечетких моделей

а – после субтрактивной кластеризации; *б* – после субтрактивной кластеризации
1 – экспериментальные данные; *2* – нечеткое моделирование

Значения ошибок равны: **trnRMSE2 = 0,3407** и **chkRMSE2 = 0,5836**. Качество моделирования значительно возросло на обучающей выборке. В то же время адекватность модели на тестовой выборке улучшилась несущественно. На это указывает и графики на рис. 3.22*б*, построенные командами:

```
plot(chkdatout)
hold on
plot(chkfuzout2, 'o')
legend('экспериментальные данные', 'нечеткое моделирование')
hold off
```

Исследуем, как зависит адекватность нечеткой модели от длительности обучения. Для этого вызовем функции anfis в следующем формате:

```
[fis3, trnErr, stepSize, fis4, chkErr] = anfis([datin datout],...
fis, [200 0 0.1], [], [chkdatin chkdatout])
```

Третий входной аргумент этот функции задает 200 итераций обучения. Длинный список выходных аргументов возвращает протокол обучения модели: размер шага (`stepSize`), ошибки моделирования на обучающей (`trnErr`) и тестовой (`chkErr`) выборках на каждой итерации алгоритма. После 200 итераций алгоритма минимальные ошибки на обучающей и тестовой выборках равны **0,326566** и **0,582545** соответственно. На первый взгляд кажется, что длительное обучение дало хорошие результаты – ошибки предыдущих моделей были больше. Однако указанные минимальные значения ошибок соответствует не одной и той же модели, а двум: `fis3` – нечеткая модель с минимальной ошибкой на обучающей выборке и `fis4` – нечеткая модель с минимальной ошибкой на тестовой выборке. Посмотрим, на каких итерациях алгоритма обучения получены модели

с минимальными ошибками. Для этого выведем кривые обучения с помощью команд:

```
plot(1:200, trnErr, '.k-', 1:200, chkErr, 'b-');
legend ('ошибка на обучающей выборке',...
'ошибка на тестовой выборке');
xlabel('итерации алгоритма обучения');
ylabel('RMSE')
```

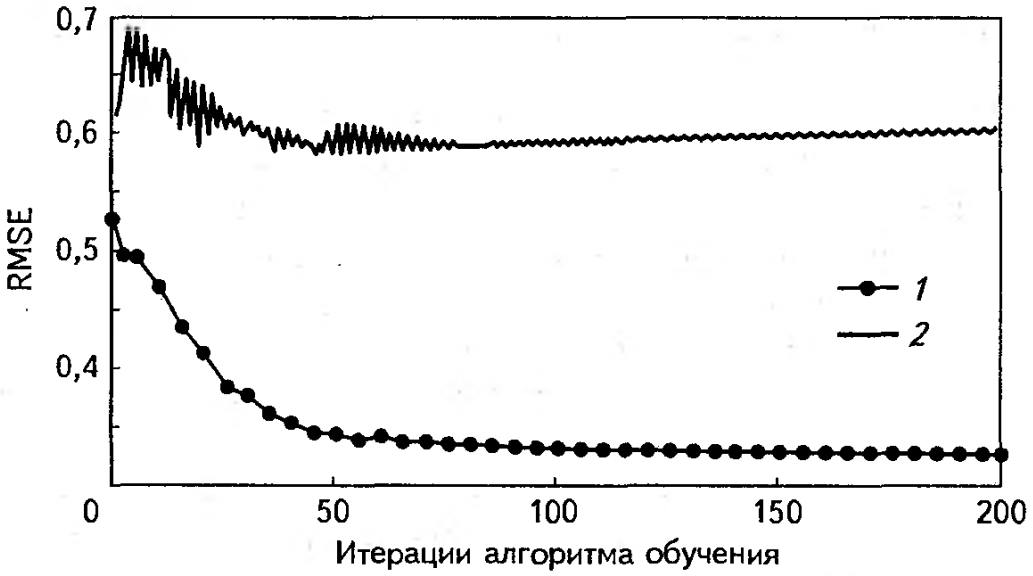


Рис. 3.23. Зависимости ошибок нечеткого моделнрования от количества итераций обучения

1 – ошибка на обучающей выборке; *2* – ошибка на тестовой выборке

Динамика обучения (рис. 3.23) показывает, что ошибка на тестовой выборке имеет тенденцию к уменьшению на протяжении около 50 итераций и достигает наименьшего значения на 52-й итерации алгоритма обучения. После этого ошибка на тестовой выборке начинает немного возрастать, хотя функция anfis продолжает минимизировать ошибку на обучающей выборке на протяжении всех 200 итераций. Начиная с 52-й итерации проявляется эффект переобучения, состоящий в потере генерализирующих свойств модели. Переобученная модель очень хорошо отражает реальность, представленную обучающей выборкой. Вне точек обучения адекватность такой модели низкая – результаты моделирования сильно отличаются от экспериментальных данных. Для предотвращения переобучения обычно используют тактику «раннего останова» – прекращение обучения при возрастании ошибки на тестовой выборке.

3.3. GUI-МОДУЛИ

В разделе приводятся руководства пользователю по работе с GUI-модулями пакета Fuzzy Logic Toolbox:

- Fuzzy Inference System Editor – редактор общих свойств системы нечеткого вывода;

- Membership Function Editor – редактор функций принадлежности;
- Rule Editor – редактор нечеткой базы знаний;
- Rule Viewer – браузер нечеткого вывода;
- Surface Viewer – браузер поверхности «входы – выход»;
- ANFIS Editor – редактор нейро-нечеткой сети;
- Findcluster – инструмент кластеризации данных.

3.3.1. FUZZY INFERENCE SYSTEM EDITOR

Редактор общих свойств системы нечеткого вывода – Fuzzy Inference System Editor (FIS-редактор) предназначен для создания, сохранения, загрузки и вывода на печать систем нечеткого вывода, а также для редактирования следующих свойств: тип системы; наименование системы; количество входных и выходных переменных; наименование входных и выходных переменных; параметры нечеткого вывода. Загрузить FIS-редактор можно командами `fuzzy`, `fuzzy(fis)` или `fuzzy('fis_file_name')`. При вызове функции fuzzy с входным аргументом загружается нечеткая система `fis` из рабочей области MATLAB или из файла `fis_file_name.fis`. Интерактивное графическое окно модуля с указанием функционального назначения основных полей графического окна приведено на рис. 3.24. Внизу окна FIS-редактора, как и всех GUI-модулей, располо-

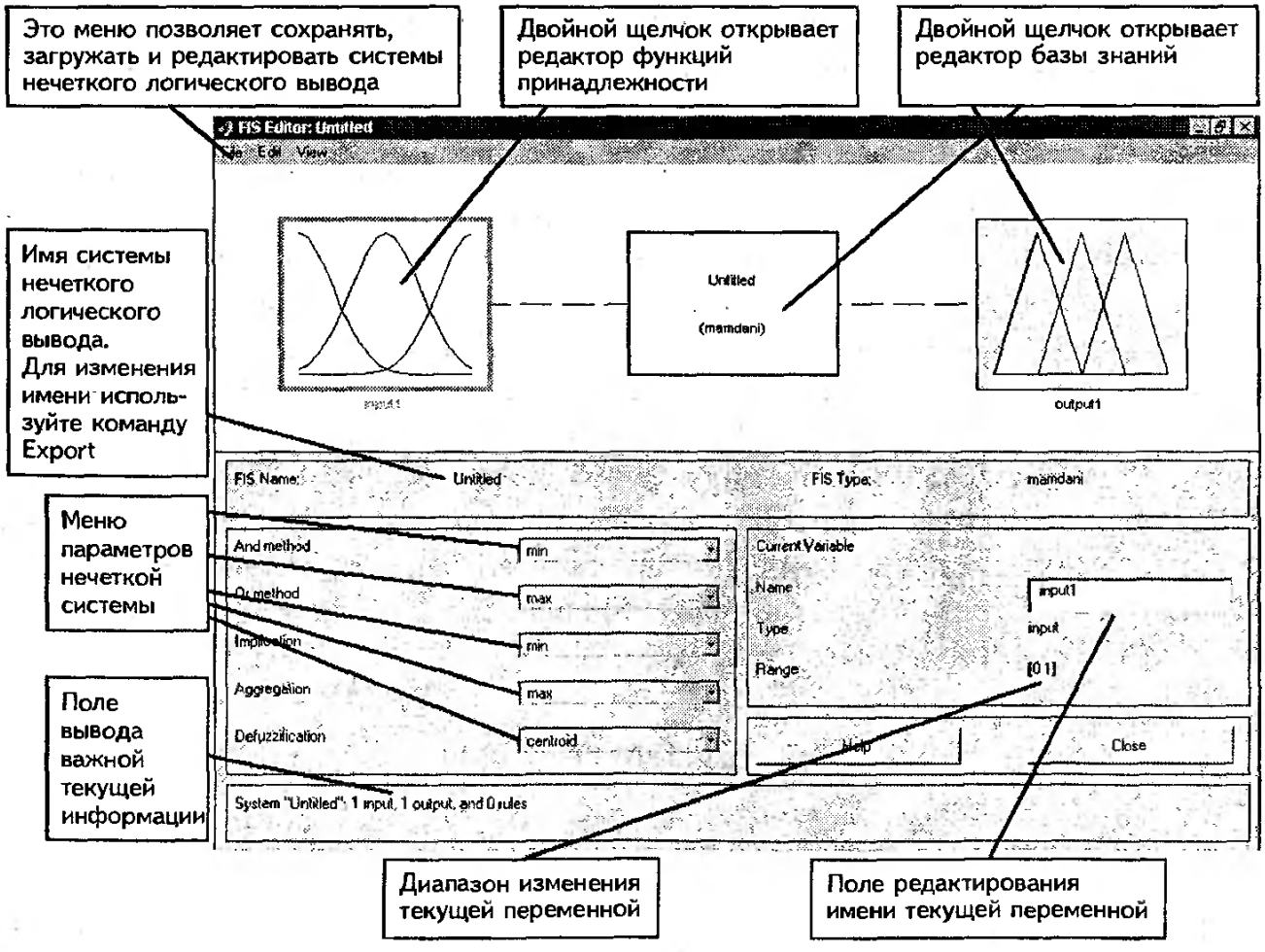


Рис. 3.24. Назначение основных полей FIS-редактора

жены кнопки **Help** и **Close** для вызова окна справки и завершения программы соответственно.

FIS-редактор содержит восемь меню. Это три общесистемных меню (**File**, **Edit**, **View**) и пять меню выбора параметров нечеткого вывода (**And Method**, **Or Method**, **Implication**, **Aggregation** и **Defuzzification**).

3.3.1.1. Меню File

Меню **File** (рис. 3.25) является общим для всех GUI-модулей, используемых с системами нечеткого вывода.

По команде **New FIS...** создается новая система нечеткого вывода. При выборе этой команды появятся две альтернативы: **Mamdani** и **Sugeno**, задающие тип нечеткой системы. Создать систему типа Мамдани можно также нажатием Ctrl+N.

По команде **Import** загружается ранее созданная система нечеткого вывода. При выборе этой команды появятся две альтернативы **From Workspace...** и **From Disk...**, загружающие нечеткую систему из рабочей области MATLAB и с диска соответственно. При выборе команды **From Workspace...** появится диалоговое окно, в котором необходимо указать идентификатор системы нечеткого вывода, находящейся в рабочей области MATLAB. При выборе команды **From Disk...** появится диалоговое окно (рис. 3.26), в котором необходимо указать имя файла системы нечеткого вывода. Файлы систем нечеткого вывода имеют расширение .fis. Загрузить нечеткую систему с диска можно также нажатием Ctrl+O или командой `fuzzy FIS_name`, где `FIS_name` – имя файла системы нечеткого вывода.

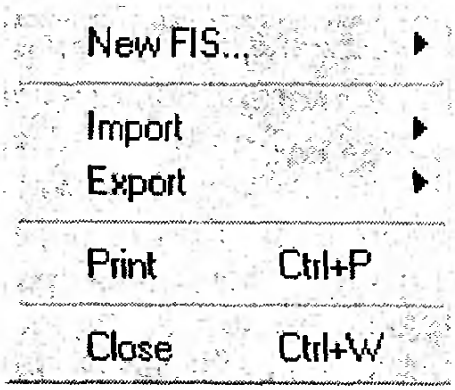


Рис. 3.25. Меню File

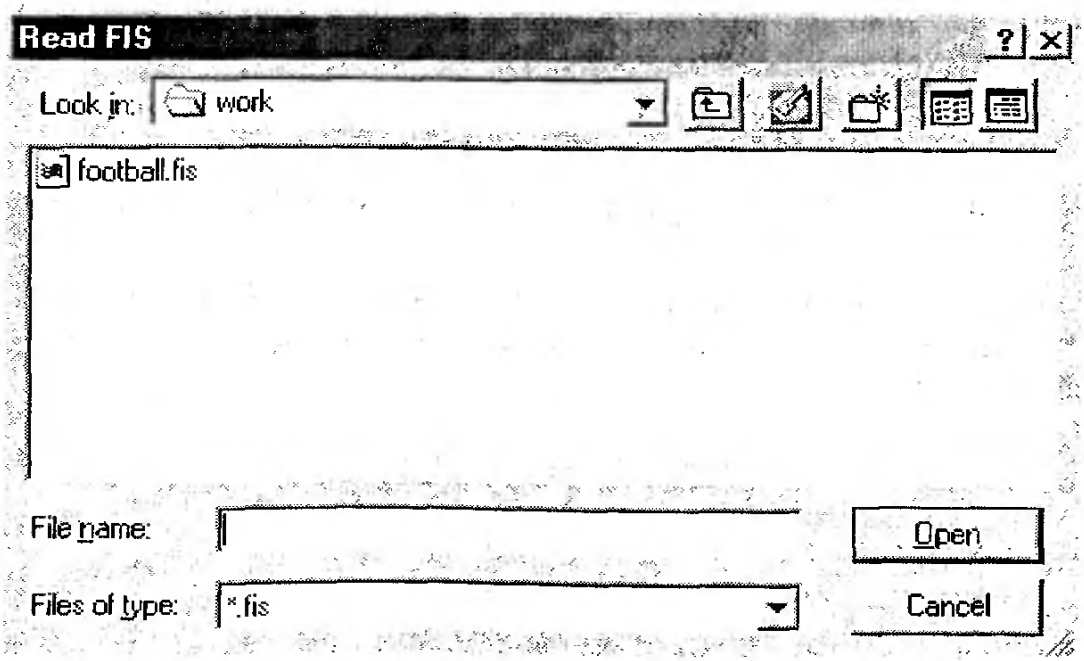


Рис. 3.26. Окно загрузки системы нечеткого вывода с диска

При выборе команды **Export** появятся две альтернативы **To Workspace...** и **To disk...**, которые позволяют скопировать систему нечеткого вывода в рабочую область MATLAB и на диск соответственно. По команде **To Workspace...** появится окно ввода идентификатора, под которым нечеткая система будет доступна в рабочей области MATLAB. По команде **To Disk...** появится диалоговое окно, в котором необходимо указать имя файла системы нечеткого вывода. Скопировать нечеткую систему в рабочую область и на диск можно также нажатием Ctrl+T и Ctrl+S соответственно.

По команде **Print** печатается копия графического окна. Выполняется также нажатием Ctrl+P.

По команде **Close** графическое окно закрывается. Окно может быть закрыто нажатием Ctrl+W или щелчком левой кнопки мыши по кнопке Close.

3.3.1.2. Меню Edit

Меню **Edit** показано на рис. 3.27.

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

По команде **Add Variable...** в систему нечеткого вывода добавляется одна переменная. При выборе этой команды появляются две альтернативы **Input** и **Output**, добавляющие входную и выходную переменные.

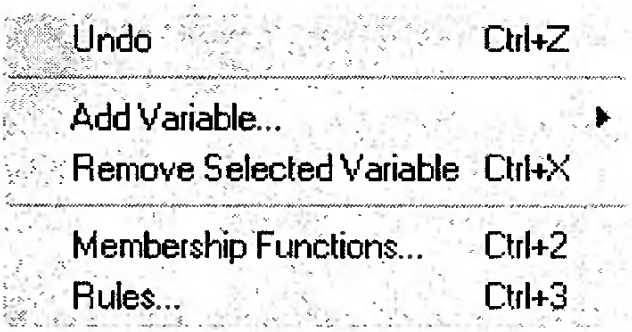


Рис. 3.27. Меню Edit FIS-редактора

Команда **Remove Selected Variable** удаляет текущую переменную из системы. Признаком текущей переменной является красная окантовка ее пиктограммы. Назначение текущей переменной происходит щелчком левой кнопки мыши по ее пиктограмме. Удалить текущую переменную можно также нажатием Ctrl+X.

Команда **Membership Functions...** открывает редактор функций принадлежностей. Выполняется также нажатием Ctrl+2.

Команда **Rules...** открывает редактор базы знаний. Выполняется также нажатием Ctrl+3.

При работе с нечеткой системой типа Сугено в меню **Edit** входит еще команда **Anfis...**, открывающая редактор нейро-нечеткой сети. Команда выполняется также по нажатию Ctrl+4.

3.3.1.3. Меню View

Меню **View** (рис. 3.28) является общим для всех GUI-модулей, используемых с системами нечеткого вывода. Оно позволяет открыть окно визуализации нечеткого вывода (команда **Rules** или нажатие клавиш

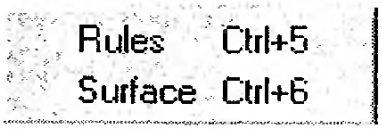


Рис. 3.28. Меню View

Ctrl+5) и окно вывода поверхности «входы – выход», соответствующей нечеткой системе (команда **Surface** или нажатие клавиш Ctrl+6).

3.3.1.4. Меню And method, Or method, Implication и Aggregation

Меню **And method** позволяет установить следующие реализации логической операции И: **min** – минимум и **prod** – умножение.

Меню **Or method** позволяет установить следующие реализации логической операции ИЛИ: **max** – максимум и **probor** – вероятностное ИЛИ.

Меню **Implication** позволяет установить следующие импликации в нечетком выводе Мамдани: **min** – минимум и **prod** – умножение. Использование умножения в качестве импликации позволяет реализовать в пакете Fuzzy Logic Toolbox нечеткий вывод Ларсена.

Меню **Aggregation** позволяет установить в нечетком выводе Мамдани следующие реализации операции объединения функций принадлежности выходной переменной (агрегирования): **max** – максимум, **sum** – ограниченная сумма и **probor** – вероятностное ИЛИ.

Для использования других способов выполнения этих операций необходимо выбрать команду **Custom...** и в появившемся окне напечатать имя *m*-функции, реализующей операцию.

3.3.1.5. Меню Defuzzification

Через меню **Defuzzification** выбирается метод дефаззификации. Для системы типа Мамдани запрограммированы следующие методы: **centroid** – центр тяжести; **bisector** – медиана; **lom** – наибольший из максимумов; **som** – наименьший из максимумов и **mom** – среднее из максимумов. Для системы типа Сугено запрограммированы такие методы: **wtaver** – взвешенное среднее и **wtsum** – взвешенная сумма. Можно установить собственный метод дефаззификации. Для этого необходимо выбрать команду **Custom...** и в появившемся окне напечатать имя *m*-функции, выполняющей дефаззификацию.

3.3.2. MEMBERSHIP FUNCTION EDITOR

Редактор функций принадлежности (Membership Function Editor) предназначен для задания мощности терм-множеств, термов, типов и параметров функций принадлежности входных и выходных переменных. По умолчанию для каждой переменной установлены по три терма с треугольными функциями принадлежности. Редактор функций принадлежности вызывается из любого GUI-модуля командой **Membership Functions...** меню **Edit** или нажатием клавиш Ctrl+2. В FIS-редакторе открыть редактор функций принадлежности можно также двойным щелчком левой кнопкой мыши по полю входной или выходной переменной. Загрузить редактор функций принадлежности можно также командами `mfedit`, `mfedit(fis)` и `mfedit ('fis_file_name')`. При вызове функции mfedit с входным аргументом загружается нечеткая система `fis` из рабочей области

MATLAB или файла `fis_file_name.fis`. Окно редактора функций принадлежности с указанием функционального назначения основных полей приведено на рис. 3.29.

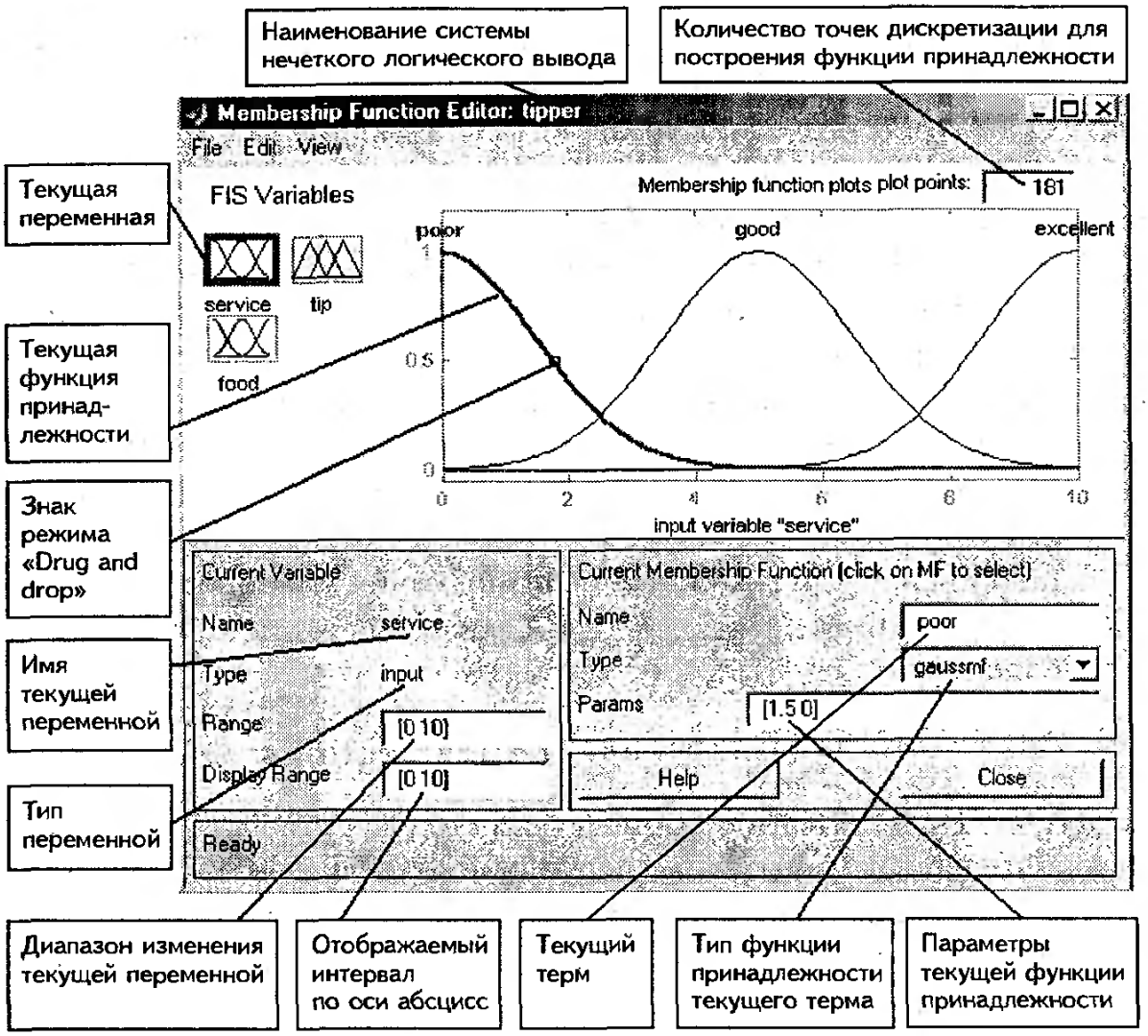


Рис. 3.29. Назначение основных полей редактора функций принадлежности

Редактор функций принадлежности содержит четыре поля ввода информации:

- **Range** – диапазон изменения текущей переменной;
- **Display Range** – окно показа функции принадлежности;
- **Name** – наименование нечеткого множества;
- **Params** – параметры функции принадлежности.

Параметры функции принадлежности можно подбирать, изменяя ее график в режиме «Drug and drop». Для этого необходимо установить мышь на знаке режима «Drug and drop» (см. рис. 3.29), нажать на левую кнопку и, не отпуская ее, изменить форму функции принадлежности. При этом параметры функции принадлежности пересчитываются автоматически.

Редактор функций принадлежности содержит три меню – **File**, **Edit** и **View**. Меню **File** и **View** описаны в пунктах 3.3.1.1 и 3.3.1.3. Меню **Edit** показано на рис. 3.30.

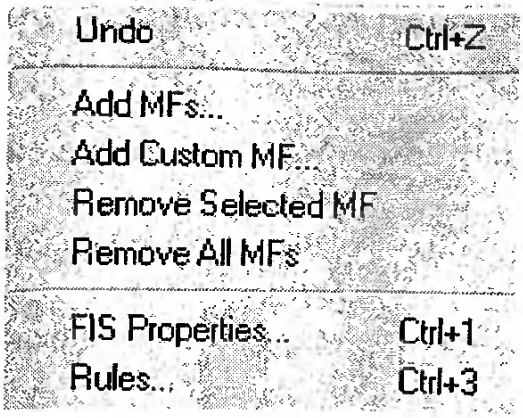


Рис. 3.30. Меню Edit редактора функций принадлежности

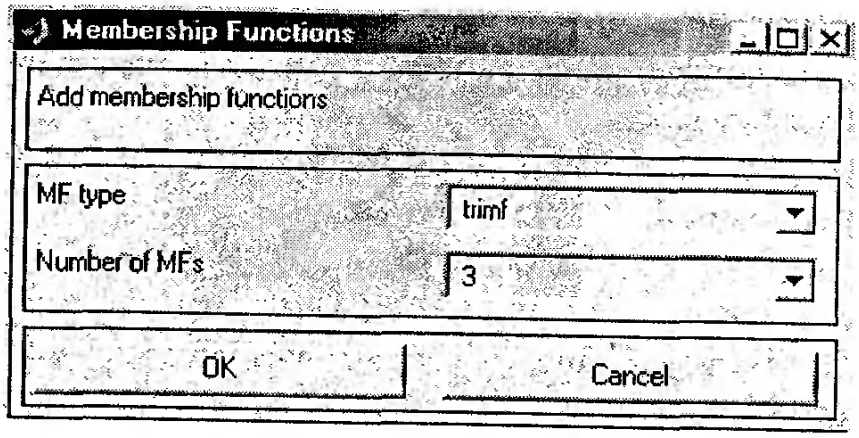


Рис. 3.31. Выбор количества термов и типа функций принадлежности

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

Команда **Add MFs...** добавляет термы в терм-множество текущей переменной. При выборе этой команды появится диалоговое окно (рис. 3.31), в котором выбирается тип функции принадлежности и количество термов. Параметры функций принадлежности устанавливаются автоматически таким образом, чтобы равномерно покрыть диапазон из поля **Range**. При изменении диапазона в окне **Range** параметры функций принадлежности масштабируются. Список встроенных типов функций принадлежности приведен на рис. 3.32.

Команда **Add Custom MF...** добавляет нечеткий терм с невстроенной функцией принадлежности. После выбора этой команды появится графическое окно (рис. 3.33), в котором необходимо напечатать терм в поле **MF name**, имя *m*-файла функции принадлежности в поле **M-File function name** и ее параметры в поле **Parameter list**. Количество параметров не должно превышать 4.

Команда **Remove Selected MF** удаляет выделенный терм из терм-множества текущей переменной. Признаком текущей переменной является красная окантов-

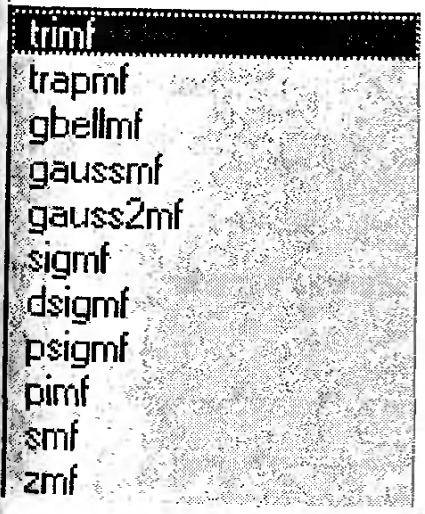


Рис. 3.32. Выбор типа функций принадлежности в меню MF type

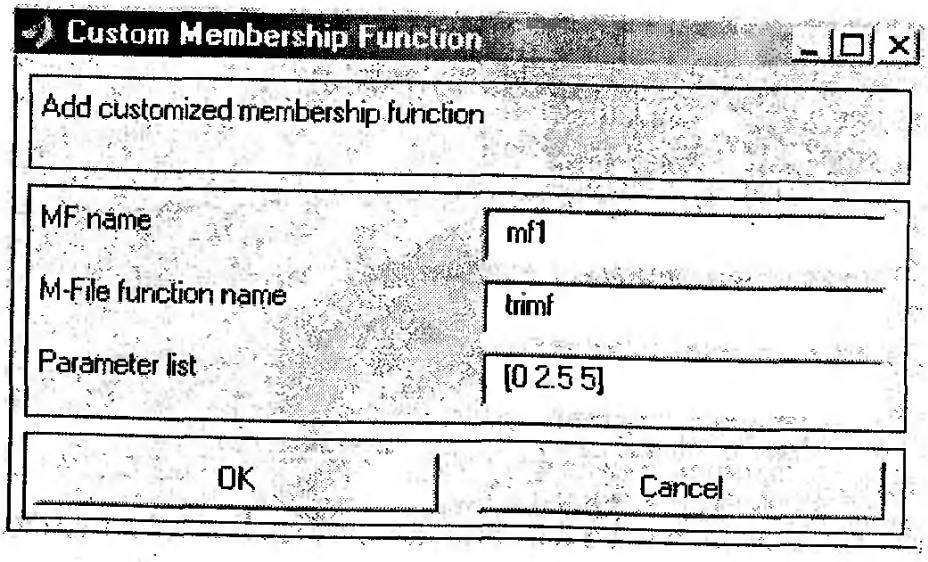


Рис. 3.33. Окно добавления нечеткого терма с невстроенной функцией принадлежности

ка ее пиктограммы. Признаком выделенного терма является красный цвет его функции принадлежности. Терм выделяется щелчком левой кнопкой мыши по графику функции принадлежности.

Команда **Remove All MFs** удаляет все термы из терм-множества текущей переменной.

Команда **FIS Properties...** открывает FIS-редактор. Выполняется также нажатием Ctrl+1.

Команда **Rules...** открывает редактор базы знаний. Выполняется также нажатием Ctrl+3.

При работе с нечеткой системой типа Сугено в меню **Edit** входит еще команда **Anfis...**, открывающая редактор нейро-нечеткой сети. Команда выполняется также по нажатию Ctrl+4.

3.3.3. RULE EDITOR

Редактор базы знаний (Rule Editor) предназначен для формирования и модификации нечетких правил. Редактор базы знаний может быть вызван из любого GUI-модуля, используемого с системами нечеткого вывода, командой **Rules...** меню **Edit** или нажатием клавиш Ctrl+3. В FIS-редакторе открыть редактор базы зна-

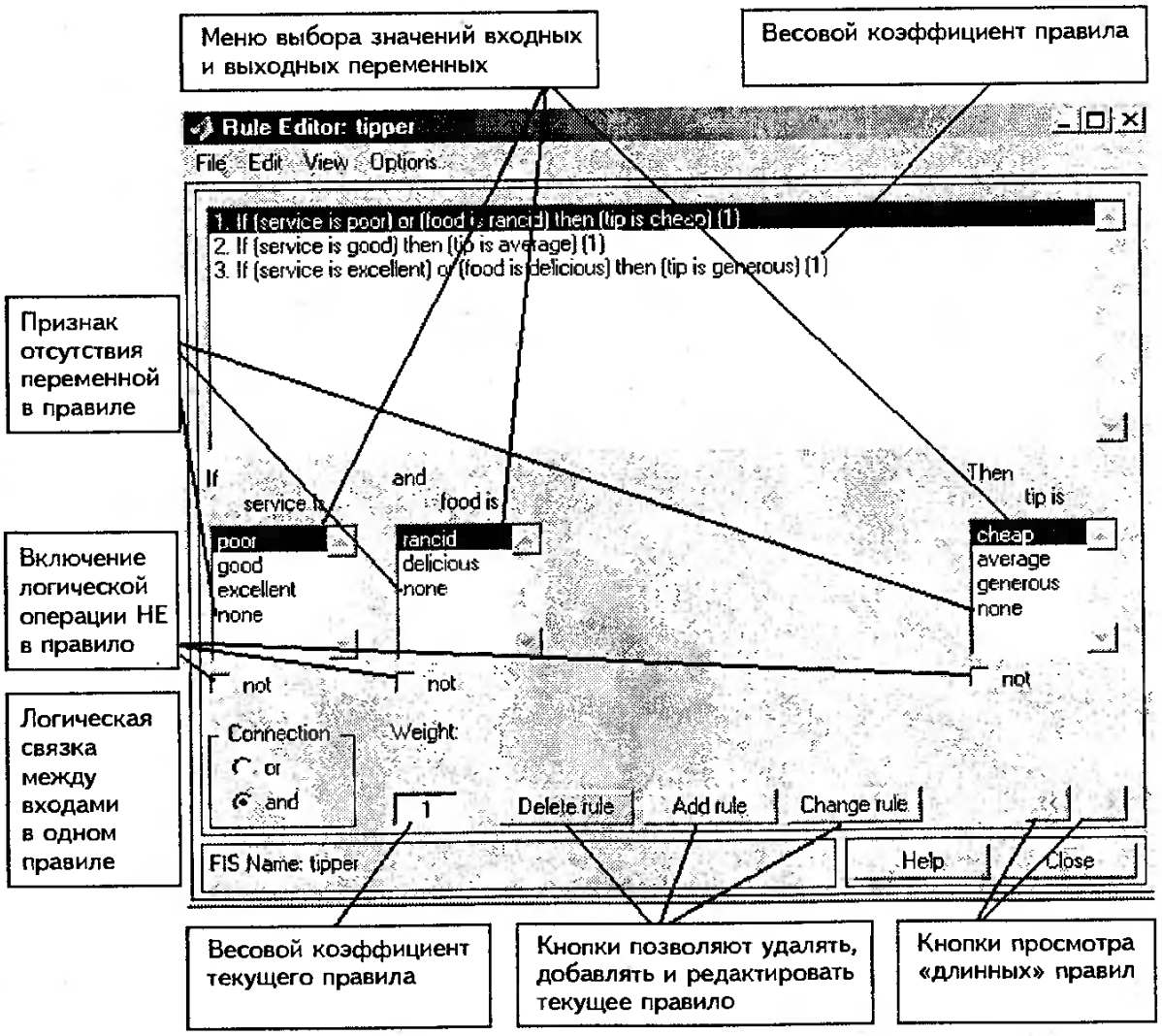


Рис. 3.34. Назначение основных полей редактора базы знаний

ний можно двойным щелчком мыши по прямоугольнику с названием системы нечеткого вывода, который находится в центре окна. Открыть редактор базы знаний можно также командами `ruleedit(fis)` и `ruleedit('fis_file_name')`, по которым загружается нечеткая система `fis` из рабочей области MATLAB или файла `fis_file_name.fis`. Окно редактора базы знаний с указанием функционального назначения основных полей приведено на рис. 3.34.

Для ввода нового правила в базу знаний необходимо мышкой подобрать соответствующую комбинацию лингвистических термов входных и выходных переменных, выбрать тип логической связки (И или ИЛИ) между переменными внутри правила, установить наличие или отсутствие логической операции НЕ для каждой лингвистической переменной, ввести значение весового коэффициента правила и нажать кнопку **Add Rule**. Если истинность нечеткого правила не изменяется при любом значении некоторой входной переменной, т.е. когда переменная не влияет на результат нечеткого вывода в данной области факторного пространства, то в качестве лингвистического значения этой переменной выбирают **none**. По умолчанию установлены следующие параметры правил:

- логическая связка переменных внутри правила – И;
- логическая операция НЕ – отсутствует;
- значение весового коэффициента правила – 1.

Для удаления правила из базы знаний необходимо щелкнуть по нему мышкой и нажать кнопку **Delete Rule**. Для модификации правила необходимо щелкнуть по нему мышкой, установить необходимые параметры правила и нажать кнопку **Edit Rule**.

Редактор базы знаний содержит четыре системных меню: **File**, **Edit**, **View** и **Options**, меню выбора термов входных и выходных переменных, поля установки логических операций И, ИЛИ, НЕ и весов правил, а также кнопки редактирования и просмотра правил. Меню **File** и **View** одинаковы для всех GUI-модулей, используемых с системами нечеткого вывода. Они описаны в пунктах 3.3.1.1 и 3.3.1.3.

3.3.3.1. Меню Edit

Меню **Edit** показано на рис. 3.35.

Рис. 3.35. Меню Edit редактора базы знаний

Команда **Undo** отменяет ранее совершенное действие. Выполняется также нажатием Ctrl+Z.

Команда **FIS Properties...** открывает FIS-редактор. Выполняется также нажатием Ctrl+1.

Команда **Membership Functions...** открывает редактор функций принадлежностей. Выполняется также нажатием Ctrl+2.

При работе с нечеткой системой типа Сугено в меню **Edit** входит еще команда **Anfis...**, открывающая редактор нейро-нечеткой сети. Команда выполняется также нажатием Ctrl+4.

3.3.3.2. Меню Options

Через меню **Options** выбирают язык и формат правил базы знаний (рис. 3.36).

При выборе команды **Language** появится список языков: **English** – английский, **Deutsch** – немецкий и **Francais** – французский.

Рис. 3.36. Меню Options редактора базы знаний

При выборе команды **Format** появится список возможных форматов правил базы знаний: **Verbose** – лингвистический; **Symbolic** – логический; **Indexed** – индексный. Базы знаний демо-системы нечеткого вывода Tipper в разных форматах показаны на рис. 3.34, 3.37 и 3.38.

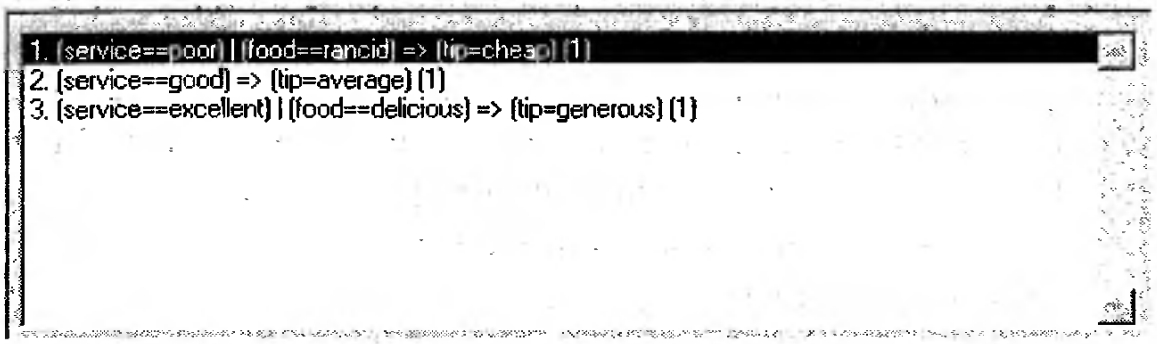


Рис. 3.37. База знаний в логическом формате

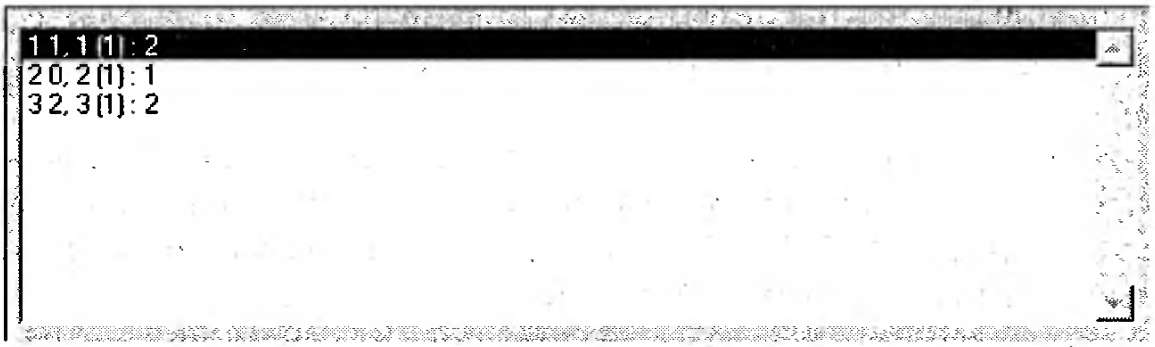


Рис. 3.38. База знаний в индексном формате

3.3.4. ANFIS EDITOR

Редактор нейро-нечеткой сети (ANFIS Editor) позволяет автоматически синтезировать из экспериментальных данных нейро-нечеткие сети и настраивать их. Нейро-нечеткую сеть можно рассматривать как одно из представлений систем нечеткого вывода типа Сугено. ANFIS-редактор может быть вызван из любого GUI-модуля, используемого с системами нечеткого вывода Сугено, командой **Anfis...** меню **Edit** или нажатием клавиш Ctrl+4. Загрузка ANFIS-редактора осуществляется также командами `anfisedit`, `anfisedit(fis)` и `anfisedit('fis_file_name')`. При вызове функции anfisedit с входным аргументом загружается нечеткая система `fis` из рабочей области MATLAB или из файла `fis_file_name.fis`. В результате выполнения этой команды появится интерактивное графическое окно, приведенное на рис. 3.39. На этом рисунке также

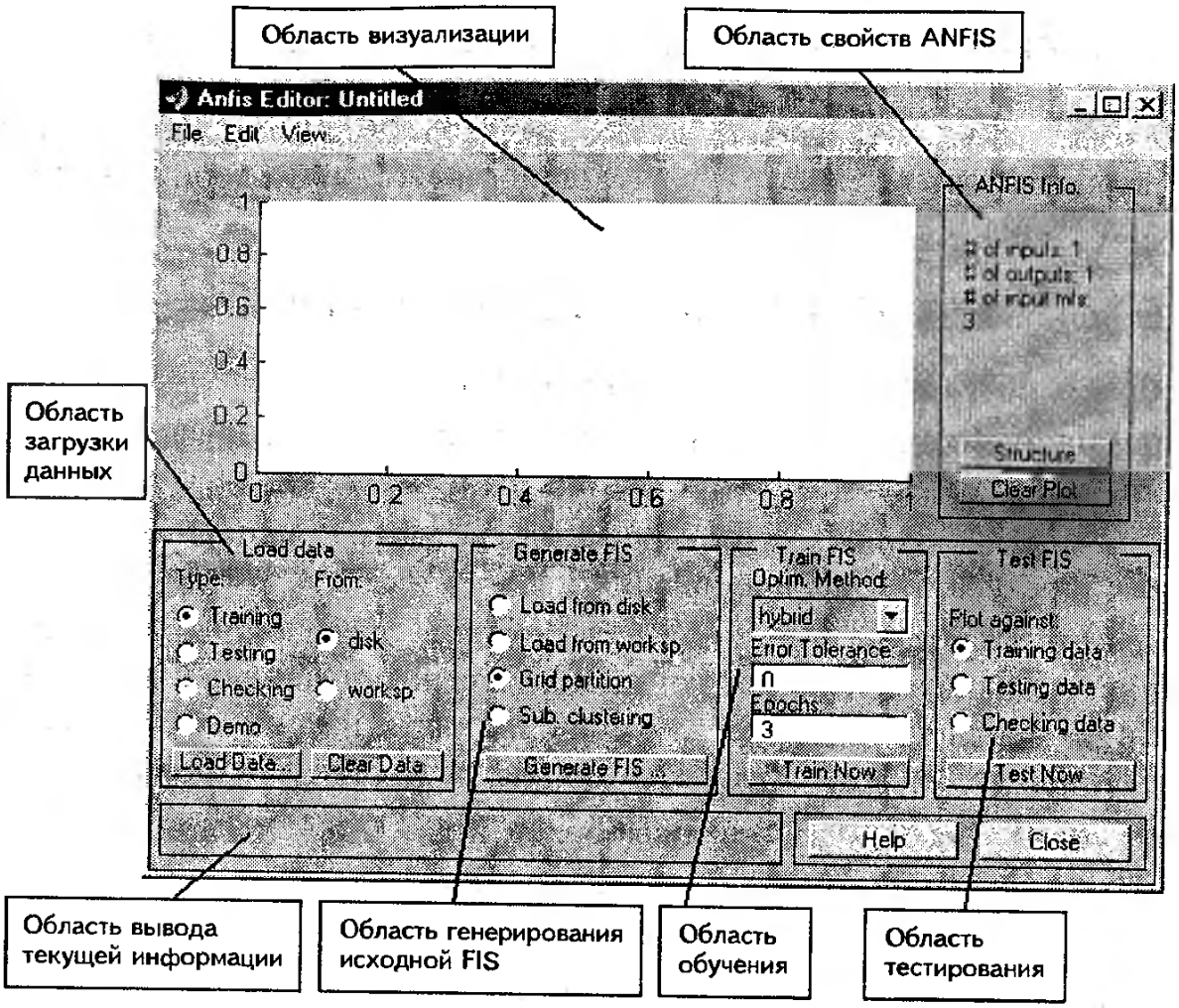


Рис. 3.39. Назначения основных полей ANFIS-редактора

указаны функциональные назначения основных полей, описание которых приведено ниже.

ANFIS-редактор содержит три верхних меню: **File**, **Edit** и **View**, область визуализации, область свойств ANFIS, область загрузки данных, область генерирования исходной системы нечеткого вывода, область обучения, область тестирования, область вывода текущей информации, а также кнопки **Help** и **Close**. Меню **File** и **View** одинаковы для всех GUI-модулей, используемых с системами нечеткого вывода. Они описаны в пунктах 3.3.1.1 и 3.3.1.3.

3.3.4.1. Меню Edit

Меню **Edit** показано на рис. 3.40.

Команда **Undo** отменяет ранее совершенное действие. Выполняется также нажатием Ctrl+Z.

Команда **FIS Properties...** открывает FIS-редактор. Выполняется также нажатием Ctrl+1.

Команда **Membership Functions...** открывает редактор функций принадлежностей. Выполняется также нажатием Ctrl+2.

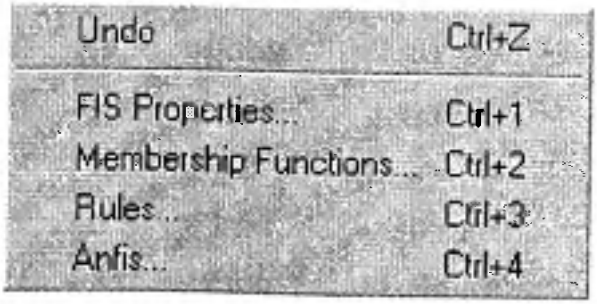


Рис. 3.40. Меню Edit ANFIS-редактора

Команда **Rules...** открывает редактор базы знаний. Выполняется также нажатием Ctrl+3.

Команда **Anfis...** открывает ANFIS-редактор. Эта команда, запущенная из ANFIS-редактора, не приводит к выполнению каких-либо действий, так как он уже открыт.

3.3.4.2. Область визуализации

В область визуализации выводится информация двух типов:

- при настройке системы – динамика обучения в виде графика зависимости ошибки обучения от номера итерации;
- при загрузке данных и тестировании системы – экспериментальные данные и результаты моделирования.

Данные выводятся в виде множества точек в двухмерном пространстве. По оси абсцисс откладывается порядковый номер строчки данных в обучающей, тестовой или контрольной выборке, а по оси ординат – значение выходной переменной для данной строчки выборки. Используются следующие маркеры:

- голубая точка (•) – тестовая выборка;
- голубая окружность (○) – обучающая выборка;
- голубой плюс (+) – контрольная выборка;
- красная звездочка (*) – результаты моделирования.

3.3.4.2. Область свойств ANFIS

В области свойств ANFIS (**ANFIS info**) выводится число входных и выходных переменных, количество функций принадлежностей для каждой входной переменной, а также объемы выборок данных. В области расположены две кнопки

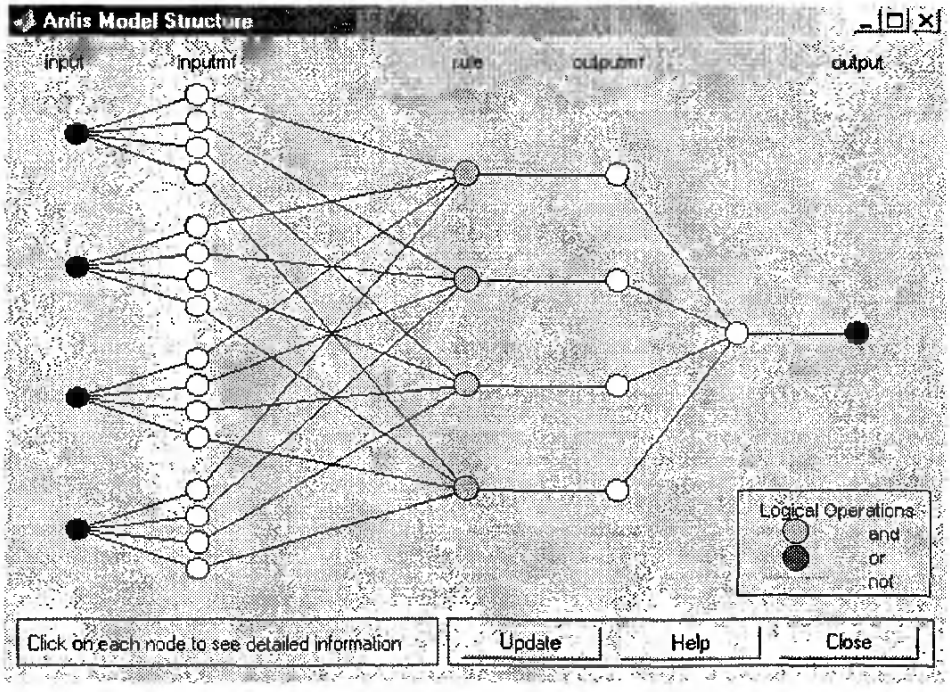


Рис. 3.41. Пример нейро-нечеткой сети в ANFIS-редакторе

Structure и **Clear Plot**. Нажатие кнопки **Clear Plot** очищает область визуализации. Нажатие кнопки **Structure** открывает новое окно, в котором система нечеткого вывода изображается нейро-нечеткой сетью. В качестве примера на рис. 3.41 приведена нейро-нечеткая сеть с четырьмя входами и одним выходом. В этой сети для лингвистической оценки каждой входной переменной используется три терма, а для выходной переменной – четыре терма.

3.3.4.3. Область загрузки данных

В области загрузки данных (**Load data**) расположены:

- меню выбора типа данных (**Type**), содержащее альтернативы: **Traning** – обучающая выборка; **Testing** – тестовая выборка; **Checking** – контрольная выборка; **Demo** – демо-набор данных;
- меню выбора источника данных (**From**), содержащее альтернативы: **disk** – диск; **worksp.** – рабочая область MATLAB;
- кнопка загрузки данных **Load Data...**, по нажатию которой появится диалоговое окно выбора файла при загрузке с диска или окно ввода идентификатора выборки при загрузке данных из рабочей области;
- кнопка очистки данных **Clear Data**.

Примечание. В одном сеансе работы ANFIS-редактора можно загружать данные только одного формата с одним и тем же количеством входных переменных.

3.3.4.4. Область генерирования исходной системы нечеткого вывода

В области генерирования (**Generate FIS**) расположены меню выбора способа создания исходной системы нечеткого 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 алгоритму решетчатого разбиения (без кластеризации);

Sub. clustering – генерирование системы через субтрактивную кластеризацию по горному методу.

В области также расположена кнопка **Generate**, по нажатию которой генерируется исходная нечеткая система.

При выборе опции **Load from disk** появится стандартное диалоговое окно открытия файла.

При выборе опции **Load from worksp.** появится стандартное диалоговое окно ввода идентификатора системы нечеткого вывода.

При выборе опции **Grid partition** появится окно ввода параметров решетчатого разбиения (рис. 3.42), в котором указывается количество термов для каждой входной переменной и тип функций принадлежности для входных и выходной переменных.

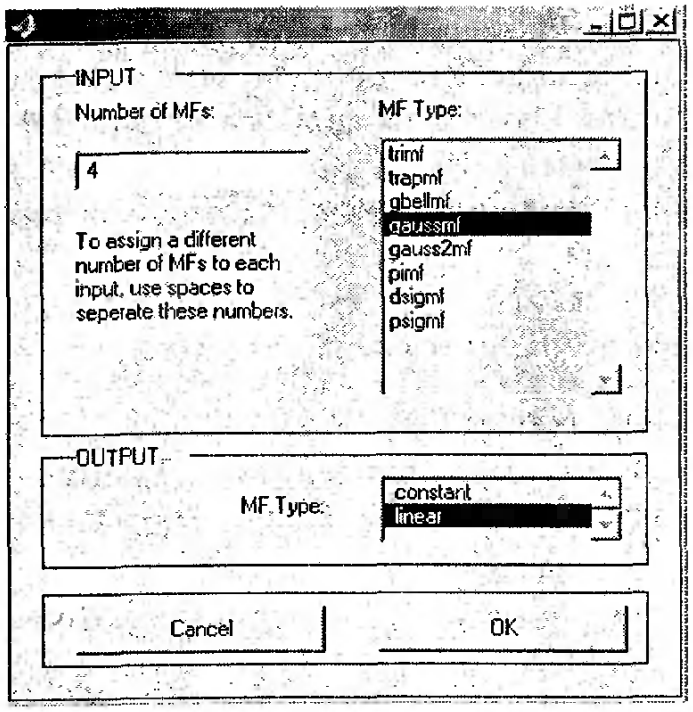


Рис. 3.42. Окно ввода параметров решетчатого разбиения

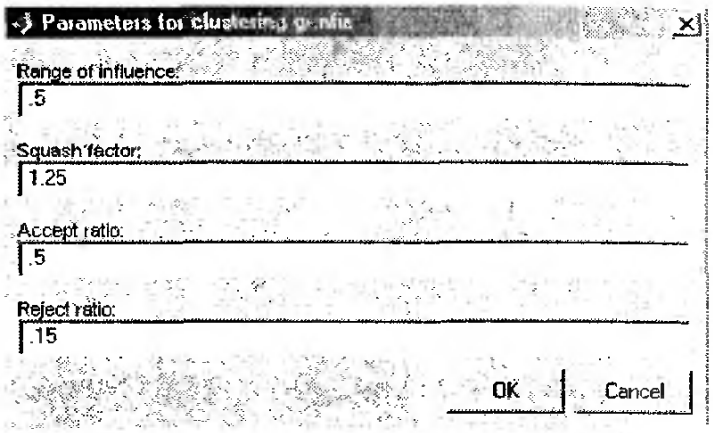


Рис. 3.43. Окно ввода параметров субтрактивной кластеризации

При выборе опции **Sub. clustering** появится окно ввода следующих параметров субтрактивной кластеризации (рис. 3.43):

- **Range of influence** – вектор радиусов, определяющий соотношения проекций кластеров на координатные оси;
- **Squash factor** – коэффициент подавления, задающий размер кластера: чем больше значение коэффициента, тем больше объектов в окрестности центра кластера будут включены в него;
- **Accept ratio** – коэффициент принятия, устанавливающий, во сколько раз потенциал данной точки должен быть выше потенциала центра первого кластера для того, чтобы она рассматривалась как возможный центр следующего кластера;
- **Reject ratio** – коэффициент отторжения, устанавливающий, во сколько раз потенциал данной точки должен быть ниже потенциала центра первого кластера, чтобы рассматриваемая точка была исключена из возможных центров кластеров.

Параметры алгоритма субтрактивной кластеризации рассмотрены в описании функции subclust.

3.3.4.5. Области обучения, тестирования и вывода текущей информации

В области обучения (**Train FIS**) расположены меню выбора метода обучения (**Optim. method**), поле задания требуемой точности обучения (**Error tolerance**), поле задания количества итераций обучения (**Epochs**) и кнопка **Train Now** для на-

чала обучения. Промежуточные результаты обучения выводятся в область визуализации и в рабочую область MATLAB. В ANFIS-редакторе реализованы два метода обучения: **backpropa** – метод обратного распространения ошибки, основанный на идеях наискорейшего спуска; **hybrid** – гибридный алгоритм, объединяющий методы обратного распространения ошибки и наименьших квадратов.

В области тестирования (**Test FIS**) расположены меню выборок данных и кнопка **Test Now**, по нажатии которой нечеткая система тестируется с выводом результатов в область визуализации.

В области вывода текущей информации отображаются сообщения об окончании выполнения операций, значение ошибки обучения или тестирования и т.п.

3.3.5. RULE VIEWER

Браузер Rule Viewer иллюстрирует ход нечеткого вывода по каждому правилу, получение результирующего нечеткого множества и выполнение процедуры дефаззификации. Rule Viewer вызывается из любого GUI-модуля, используемого

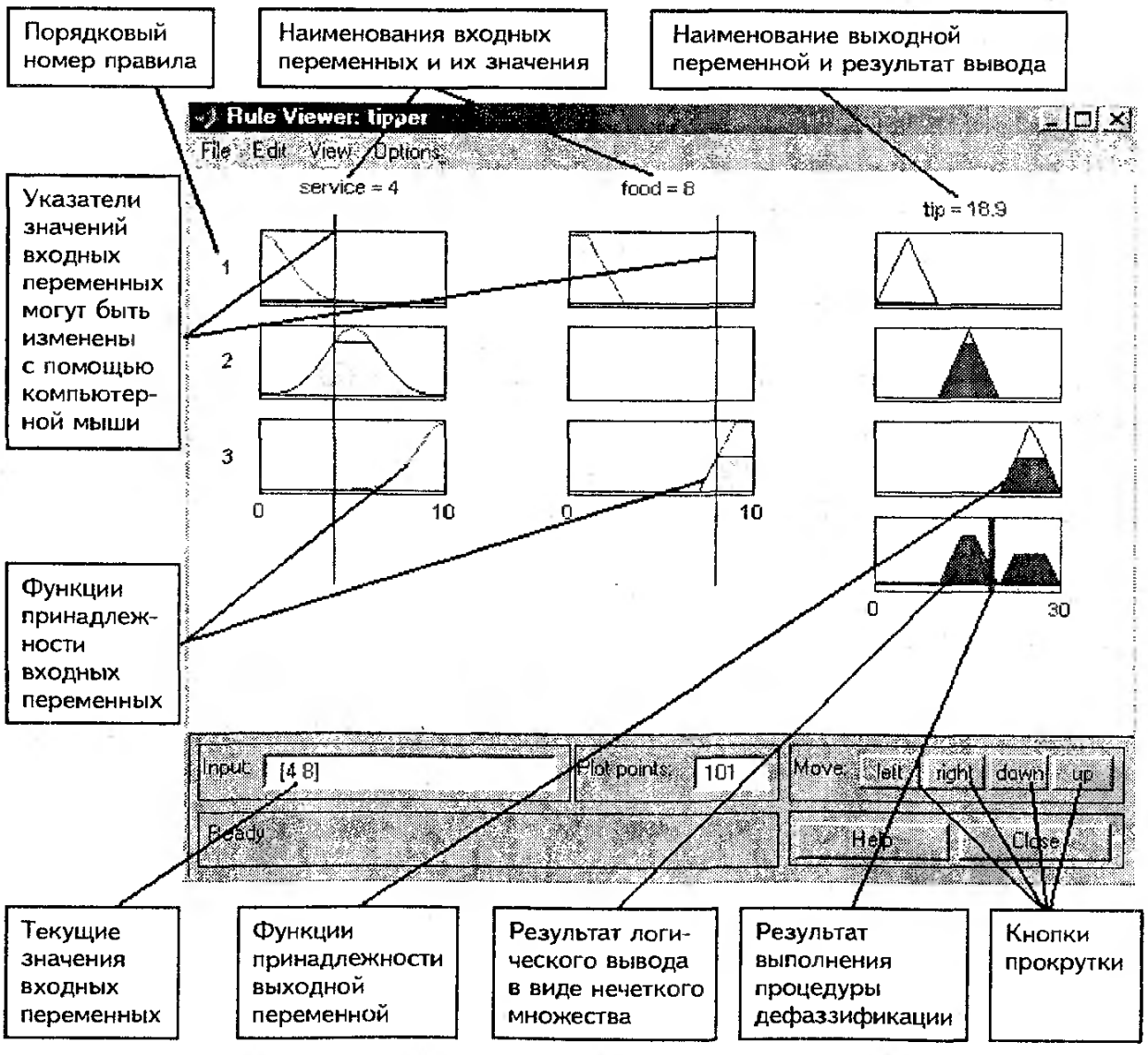


Рис. 3.44. Визуализация нечеткого вывода для демо-системы tipper в Rule Viewer

с системами нечеткого вывода, командой **Rules** меню **View** или нажатием клавиш Ctrl+5. Открыть модуль можно также командой `ruleview('fis_file_name')`, по которой загружается нечеткая система из файла `fis_file_name.fis`. Окно браузера Rule Viewer с указанием функционального назначения основных полей показано на рис. 3.44.

Правило базы знаний представляется последовательностью горизонтально расположенных прямоугольников. Для вывода правила в формате **Rule Editor** необходимо щелкнуть мышкой по порядковому номеру правила. На рис. 3.44 первые две колонки прямоугольников соответствуют посылкам правил, а последний ряд прямоугольников – заключениям правил. Пустой прямоугольник в визуализации второго правила означает, что в этом правиле посылка по переменной **food** отсутствует (**food is none**). Заливка графиков функций принадлежностей входных переменных указывает на степень принадлежности значений входов нечетким термам данного правила, заливка графика функции принадлежности выходной переменной показывает результат логического вывода по данному правилу в виде нечеткого множества. Результирующее нечеткое множество, соответствующее логическому выводу по всем правилам, показано в нижнем прямоугольнике последнего столбца графического окна. В этом же прямоугольнике жирная вертикальная линия соответствует четкому значению после дефаззификации.

Ввод значений входных переменных осуществляется двумя способами: набором чисел на клавиатуре в поле **Input**; перемещением мышкой вертикальных линий-указателей. В последнем случае мышь устанавливают на данной линии, нажимают левую кнопку мыши и не отпуская ее перемещают указатель на нужную позицию. Новое значения входной переменной будет пересчитано автоматически и выведено в окне **Input**.

Rule Viewer содержит четыре меню (**File**, **Edit**, **View** и **Options**), два поля ввода информации (**Input** и **Plot points**) и кнопки прокрутки изображения влево–вправо (**left–right**) и вверх–вниз (**up–down**). В поле **Plot points** задается количество точек дискретизации для построения графиков функций принадлежности. Значение по умолчанию – 101. Меню **File** и **View** одинаковы для всех GUI-модулей используемых с системами нечеткого вывода. Они описаны в пунктах 3.3.1.1 и 3.3.1.3.

Меню **Edit** приведено на рис. 3.45.

Команда **FIS Properties...** открывает FIS-редактор. Выполняется также нажатием Ctrl+1.

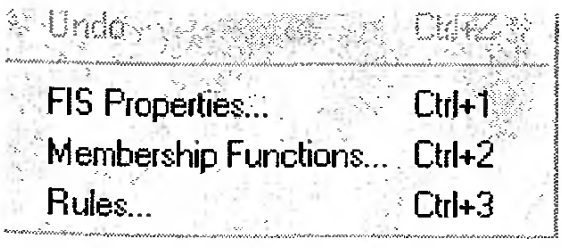


Рис. 3.45. Меню Edit браузера Rule Viewer

Команда **Membership Functions...** открывает редактор функций принадлежностей. Выполняется также нажатием Ctrl+2.

Команда **Rules...** открывает редактор базы знаний. Выполняется также нажатием Ctrl+3.

При работе с нечеткой системой типа Сугено в меню **Edit** входит еще команда **Anfis...**, открывающая редактор нейро-нечеткой сети. Команда выполняется также нажатием Ctrl+4.

Меню **Options** содержит команду **Format**, которая задает формат вывода выбранного правило в нижней части графического окна. Возможные форматы правил: **Verbose** – лингвистический; **Symbolic** – логический и **Indexed** – индексный.

3.3.6. SURFACE VIEWER

Браузер Surface Viewer выводит поверхность «входы – выход» нечеткой системы. Модуль позволяет вывести график зависимости любой выходной переменной от произвольных двух (или одной) входных переменных. Surface Viewer вызывается из любого GUI-модуля, используемого с системами нечеткого вывода, командой Surface меню View или нажатием клавиш Ctrl+6. Открыть модуль можно также командой `surfview('fis_file_name')`, по которой загружается нечеткая система из файла `fis_file_name.fis`. Окно браузера Surface Viewer с указанием функционального назначения основных полей показано на рис. 3.46.

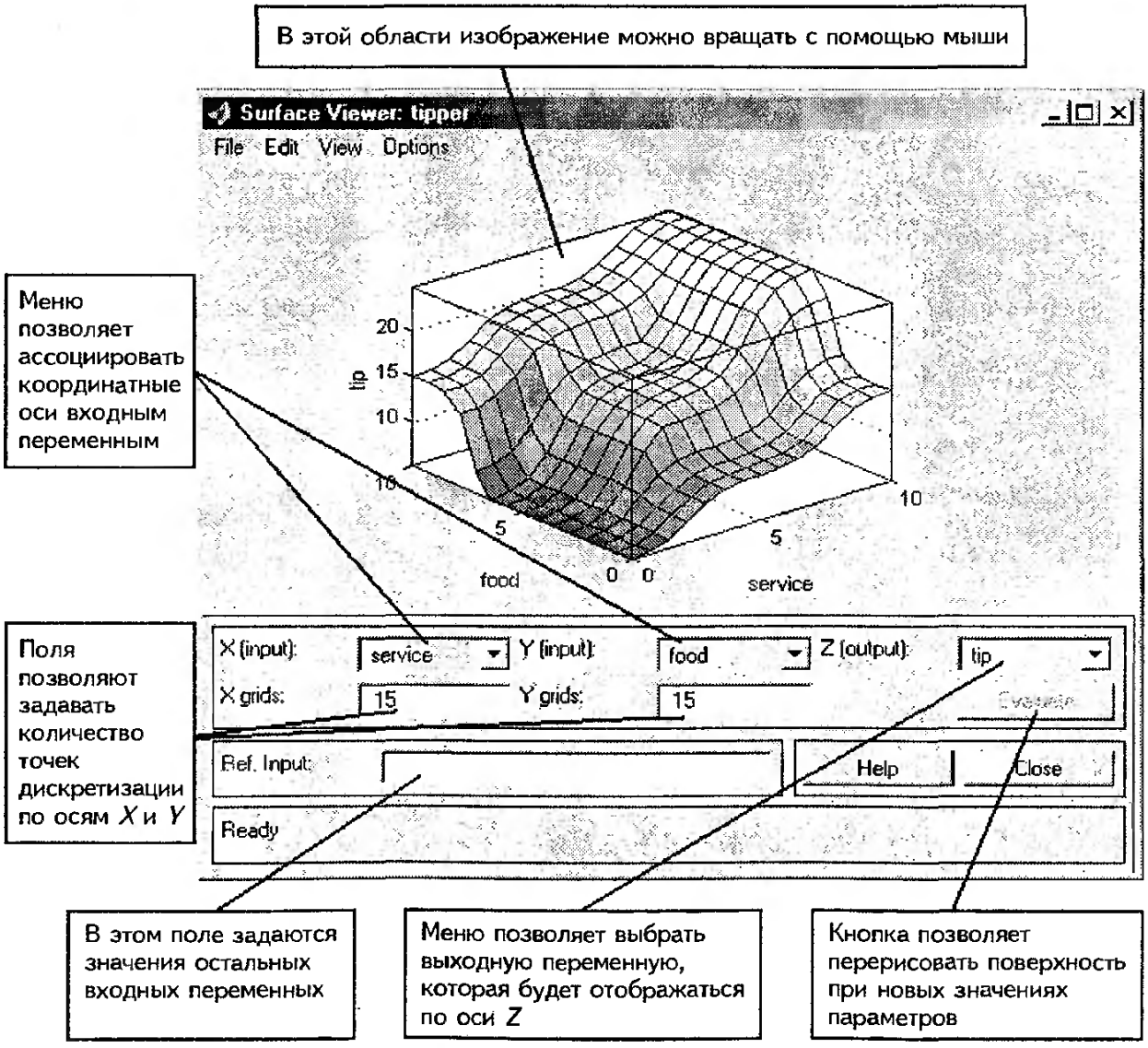


Рис. 3.46. Визуализация поверхности «входы – выход» для демо-системы tipper в Surface Viewer

В браузере Surface Viewer можно вращать поверхность «входы – выход» с помощью мыши, для чего установить мышь на поверхности «входы – выход», нажать на левую кнопку и не отпуская ее повернуть график на требуемый угол.

Surface Viewer содержит четыре верхних меню (**File**, **Edit**, **View** и **Options**), три меню выбора координатных осей (**X (input)**, **Y (input)** и **Z (output)**), три поля ввода информации (**X girds**, **Y girds** и **Ref. Input**), кнопку **Evaluate** для построения поверхности при новых параметрах. Меню **File** и **View** одинаковы для всех GUI-модулей, используемых с системами нечеткого вывода. Они описаны в пунктах 3.3.1.1 и 3.3.1.3. Меню **Edit** браузера Surface Viewer такое же, как и у Rule Viewer (см. предыдущий подраздел).

3.3.6.1. Меню Options

Меню **Options** изображено на рис. 3.47.

Команда **Plot** задает формат вывода поверхности «входы – выход» через меню, показанное на рис. 3.48. На рис. 3.49 приведены поверхности «входы – выход» всех поддерживаемых форматов для демо-системы tipper.

Команда **Color Map** задает палитру цветов поверхности «входы – выход». Возможные значения: **default** – палитра, установленная по умолчанию; **blue** – холодная сине-голубая палитра; **hot** – теплая палитра из черного, красного, желтого и белого цветов; **HSV** – насыщенная палитра из желтого, зеленого, циана, голубого, маженты и красного цветов.

Команда **Always evaluate** позволяет установить/отменить режим автоматического, т.е. без нажатия кнопки **Evaluate**, перерисовывания поверхности «входы – выход» при любом изменении параметров нечеткой системы.

Рис. 3.47. Меню Options браузера Surface Vie-

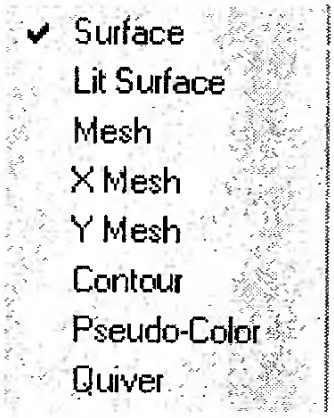


Рис. 3.48. Меню Plot браузера Surface Viewer

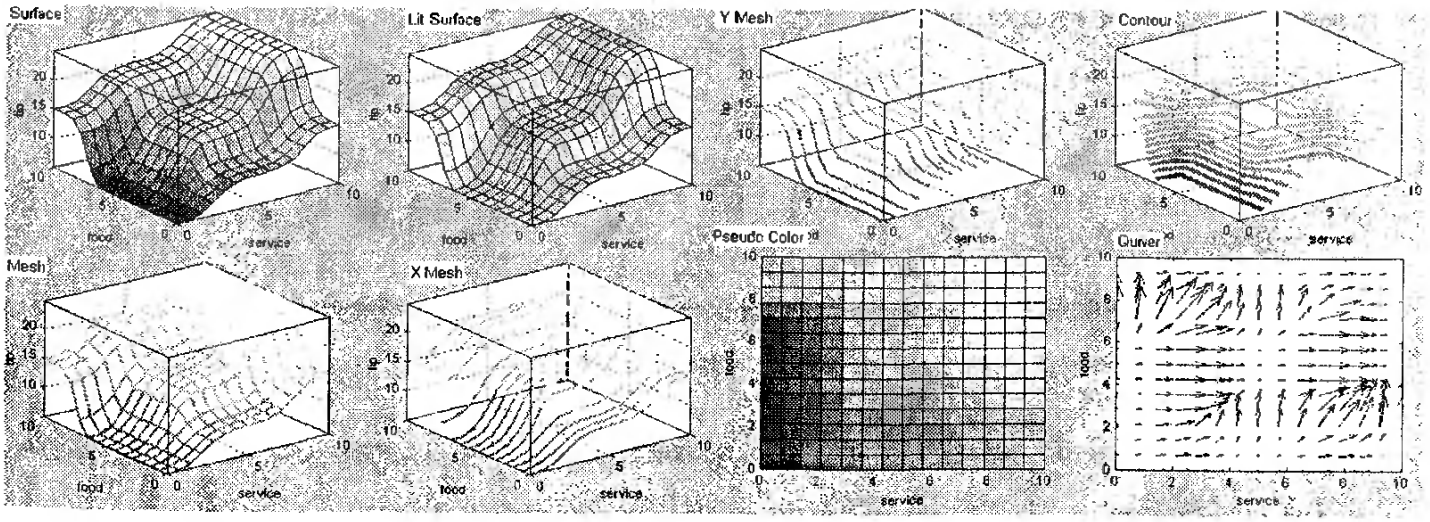


Рис. 3.49. Поверхности «входы – выход» различных форматов в браузере Surface Viewer

3.3.6.2. Меню координатных осей

Через меню **X (input)**, **Y (input)** и **Z (output)** осям координат ставятся в соответствие входы и выходы нечеткой системы. Входы отображаются по осям **X** и **Y**, а выходы только по оси **Z**. В Surface Viewer можно вывести однофакторные зависимостей «вход – выход». Для этого в меню второй координатной оси (**X (input)** или **Y (input)**) необходимо выбрать значение **none**.

3.3.6.3. Поля ввода информации

В полях **X girds** и **Y girds** задаются количества точек дискретизации по осям **X** и **Y** для построения поверхности «входы – выход». По умолчанию количество дискрет равно 15.

В поле **Ref. Input** задаются значения входов нечеткой системы, которые не ассоциированы с осями координат. Значения по умолчанию равны серединам интервалов изменения переменных.

3.3.7. FINDCLUSTER

GUI-модуль Findcluster позволяет найти центры кластеров многомерных данных по алгоритму нечетких *c*-средних и горному алгоритму субтрактивной кластеризации. Загрузка модуля Findcluster осуществляется по команде `findcluster` или `findcluster('file_name.dat')`. В последнем случае в GUI-модуль Findcluster загружаются данные для кластеризации из файла `'file_name.dat'`. Основное графическое окно модуля Findcluster с указанием функционального назначения областей приведено на рис. 3.50.

Модуль Findcluster содержит семь верхних типовых меню графического окна (**File**, **Edit**, **View**, **Insert**, **Tools**, **Windows** и **Help**), области визуализации, загрузки данных и кластеризации, а также вывода текущей информации, в котором появляются сообщения о состоянии модуля, номере итерации алгоритма кластеризации, значениях целевой функции и т.п.

3.3.7.1. Область визуализации

В области визуализации в двумерном пространстве выводятся экспериментальные данные (объекты кластеризации) и найденные центры кластеров. Для данных используется маркер в виде красной окружности (○), а для центров кластеров – маркер в виде черной точки (•). В области визуализации также расположены меню выбора координатных осей **X-axis** и **Y-axis**, позволяющие ассоциировать признаки объектов с осями абсцисс и ординат соответственно.

3.3.7.2. Область загрузки данных

Область загрузки данных расположена в правом верхнем углу окна. В ней находится кнопка **Load Data...**, по нажатию которой с диска загружаются данные через типовое окно открытия файла. Данные в файле должны быть записаны построчно, т.е. каждому объекту кластеризации соответствует одна строка файла данных.

3.3.7.3. Область кластеризации

В области кластеризации пользователь может выбрать алгоритм кластеризации, установить его параметры, провести кластеризацию и сохранить координаты центров кластеров в файле. В области кластеризации расположены меню и три кнопки.

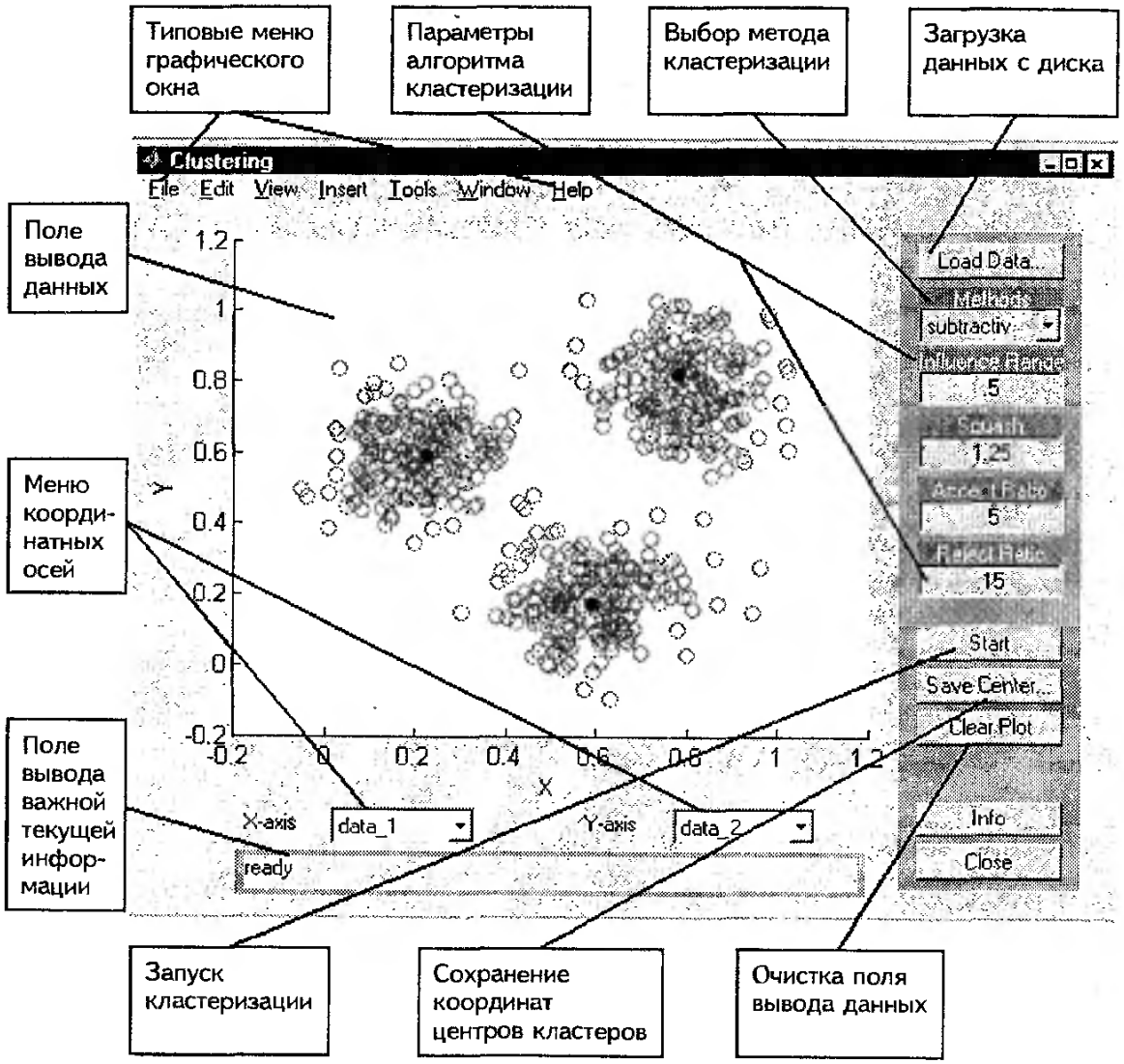


Рис. 3.50. Назначения основных полей модуля Findcluster

Меню **Method...** позволяет выбрать один из двух алгоритмов кластеризации: **subtractiv** – алгоритм субтрактивной кластеризации; **fcm** – алгоритм нечетких *c*-средних. При субтрактивной кластеризации графическое окно модуля Findcluster показано на рис. 3.50. В этом случае пользователь может установить значения следующих параметров: **Influence Range** – важность признаков, **Squash** – коэффициент подавления, **Accept Ratio** – коэффициент принятия и **Reject Ratio** – коэффициент отторжения. Смысл этих параметров объяснен в описании функции subclust. При выборе алгоритма нечетких *c*-средних область кластеризации принимает вид, изображенный на рис. 3.51. В этом случае пользователь может установить значения следующих параметров: **Cluster Num.** – количество кластеров;

Max Iteration # – максимальное количество итераций алгоритма; **Min** – минимально допустимое значение улучшения целевой функции за одну итерацию алгоритма; **Exponent** – значения экспоненциального веса. Дополнительная информация об этих параметрах приведена в описании алгоритма нечетких *c*-средних.

По нажатии кнопки **Star** запускается кластеризация. При использовании алгоритма нечетких *c*-средних координаты центров кластеров выводятся в окне визуализации после каждой итерации. При использовании субтрактивного алгоритма открывается дополнительное окно с динамикой кластеризации. Координаты центров кластеров выводятся по окончании выполнения алгоритма.

По нажатии кнопки **Save Center...** координаты найденных центров кластеров сохраняются на диск через типовое окно записи данных в файл. Координаты центров записываются в таком же формате, что и данные для кластеризации, т.е. в каждой строке хранятся координаты одного центра.

По нажатии кнопки **Clear Plot** поле вывода данных очищается.

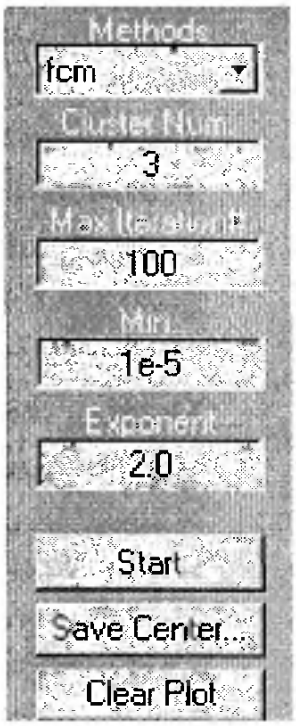


Рис. 3.51. Область кластеризации модуля Findcluster при использовании алгоритма нечетких *c*-средних

3.4. ДЕМО-ПРИМЕРЫ

В разделе описываются все демо-примеры пакета Fuzzy Logic Toolbox, иллюстрирующие применение теории нечетких множеств для управления техническими объектами, идентификации динамических систем, прогнозирования, обработки сигналов и для решения других задач. Коды демо-примеров открыты, поэтому на их основе можно быстро разработать собственные системы решения аналогичных задач.

3.4.1. ЗАПУСК ОСНОВНЫХ ДЕМО-ПРИМЕРОВ

Запуск демо-примеров происходит по набору имени соответствующей программы в командной строке MATLAB. Основные демо-примеры можно запустить: командой **Demos** в системном меню **Help**; командой `demo`; программой fuzdemos.

Демонстрационная программа fuzdemos выводит на экран меню для запуска следующих демо-примеров пакета Fuzzy Logic Toolbox:

- **ANFIS: Noise cancellation** – адаптивное подавление шумов с помощью ANFIS;
- **ANFIS: Time-series prediction** – предсказание временного ряда Маккся – Глэсса с помощью ANFIS;

- **ANFIS: Gas mileliage prediction** – прогнозирование топливной эффективности автомобиля с помощью ANFIS;
- **Fuzzy c-means clustering** – кластеризация алгоритмом нечетких *c*-средних;
- **Subtractive clustering** – субтрактивная кластеризация;
- **Ball juggler** – жонглирование шариком с помощью теннисной ракетки;
- **Inverse kinematics** – инверсная кинематика робота-манипулятора;
- **Defuzzification** – дефаззификация различными методами;
- **Membership function gallery** – галерея функций принадлежности;
- **Water Tank (sim)** – нечеткое управление уровнем воды (необходим Simulink);
- **Water Tank with Rule View (sim)** – нечеткое управление уровнем воды с демонстрацией базы знаний (необходим Simulink);
- **Cart and pole (sim)** – нечеткое управление системой «перевернутый маятник на тележке» (необходим Simulink);
- **Cart and two poles (sim)** – нечеткое управление системой «два перевернутых маятника на тележке» (необходим Simulink);
- **Ball and beam (sim)** – нечеткое управление системой «шарик на коромысле» (необходим Simulink);

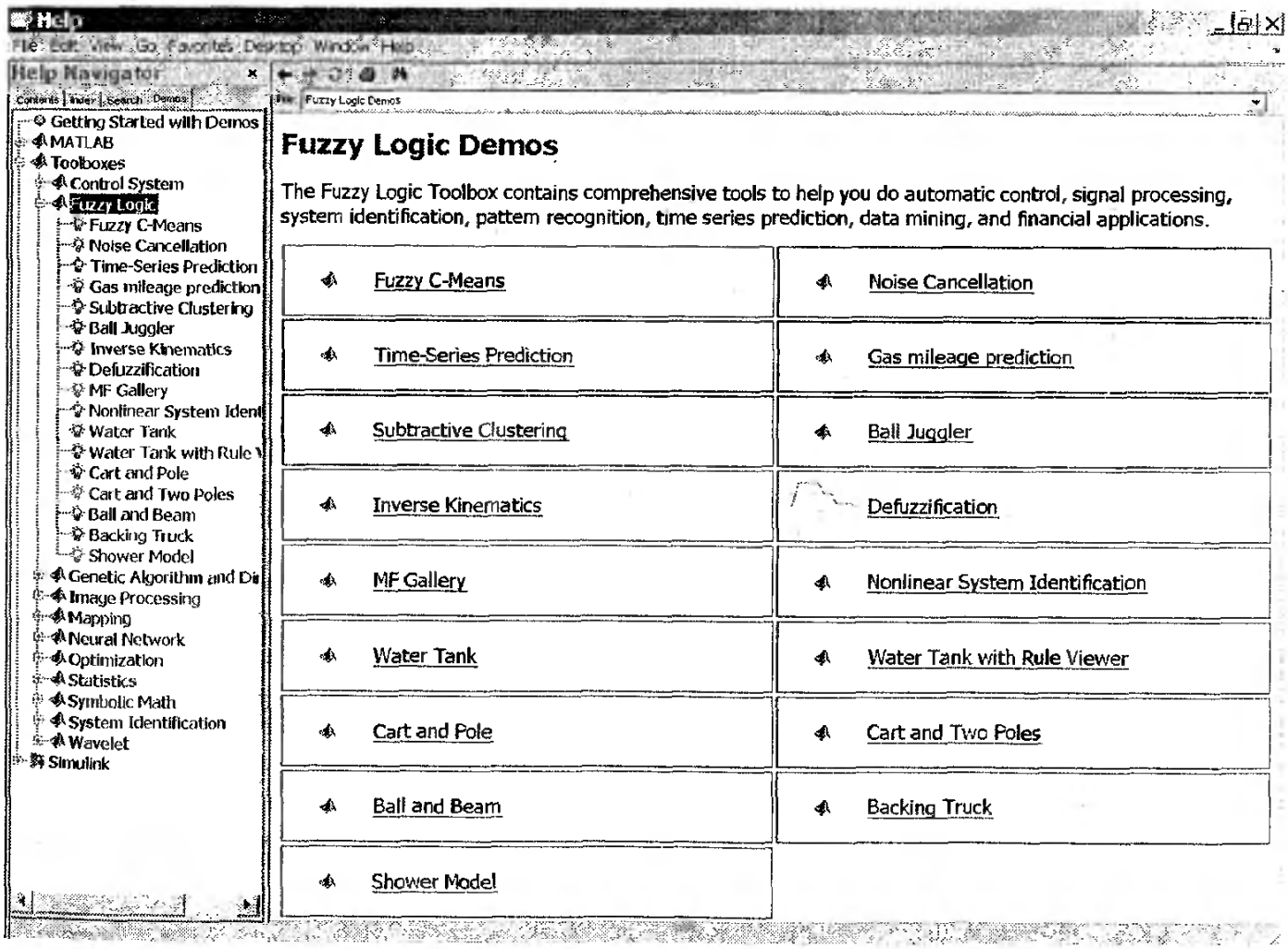


Рис. 3.52. Меню основных демо-примеров пакета Fuzzy Logic Toolbox

- **Backing truck (sim)** – нечеткое управление парковкой грузовика (необходим Simulink).

Для запуска демо-примера необходимо выбрать мышкой соответствующий пункт меню и нажать кнопку **Run this demo**. Нажатие кнопки **Close** завершает работу программы fuzdemos.

В MATLAB 7 демо-примеры удобно просматривать через специальное демо-меню (рис. 3.52). Для его вызова необходимо вначале открыть меню всех демо-примеров, используя команды demo или через опцию **Demos** в системном меню **Help**. Затем в меню **Help Navigator** выбрать опцию **Fuzzy Logic**. Для активизации демо-примера надо щелкнуть мышкой по его названию. На экране появится описание демо-примера. Для запуска демонстрации необходимо нажать кнопку **Run this demo**, для просмотра программного кода – кнопку **View code**.

3.4.2. ПРЕДСКАЗАНИЕ ТОПЛИВНОЙ ЭФФЕКТИВНОСТИ АВТОМОБИЛЯ

Демонстрационная программа gasdemo иллюстрирует применение ANFIS для предсказания топливной эффективности автомобиля. Топливная эффективность (MPG – miles per gallon) определяется как количество миль, проезжаемых автомобилем на одном галлоне топлива. При выполнении программы gasdemo на экран выводится девять слайдов. Для управления демонстрацией каждый слайд имеет такие кнопки:

- **Next>>** – показ следующего слайда (на первом слайде вместо этой кнопки расположена кнопка **Start>>** – запуск демонстрации);
- **Reset** – возврат на первый слайд;
- **View Code** – просмотр кода программы;
- **Close** – завершение работы программы.

На первых двух слайдах приводится следующее описание задачи. Прогнозирование топливной эффективности автомобиля является типовой задачей нелинейного регрессионного анализа. Прогнозирование MPG осуществляется по следующим параметрам автомобиля: число цилиндров, литраж, мощность, масса, ускорение и год выпуска. Экспериментальные данные записаны в файле auto-gas.dat. Фрагмент выборки данных показан на рис. 3.53. Данные доступны через веб-репозитарий задач автоматического обучения Калифорнийского университета в Ирвине (*http://www.ics.edu/~mlearn/MLRepository.html*). На втором слайде указано, что для построения модели прогнозирования топливной эффективности экспериментальные данные разделены на обучающую и тестовую выборки. В обучающую выборку попали нечеткие строки файла, а в тестовую – четные.

Слайды с третьего по пятый описывают выбор наилучшего набора входных переменных нечеткой модели прогнозирования с помощью функции exhsrch. Для сокращения времени перебора нечеткие модели с разными входами обучаются за одну итерацию ANFIS-алгоритма.

Car Name	Input Attributes						Output
	No. of Cylinders	Displacement	Horsepower	Weight	Acceleration	Year	MPG
Chevrolet Chevelle Malibu	8	307	130	3504	12	70	18
Plymouth Duster	6	198	95	2833	15,5	70	22
Fiat 128	4	90	75	2108	15,5	74	24
Oldsmobile Cutlass Supreme	8	260	110	4060	19	77	17
Toyota Tereel	4	89	62	2050	17,3	81	37,7
Honda Accord	4	107	75	2205	14,5	82	36
Ford Ranger	4	120	79	2625	18,6	82	28

Рис. 3.53. Фрагмент выборки для прогнозирования топливной эффективности автомобиля

На третьем слайде описывется процедура оценки информативности параметров автомобиля на основе грубых моделей «вход – выход». Слайд сопровождается дополнительным графическим окном (рис. 3.54), сгенерированным командой

```
exhsrch(1, trn_data, chk_data, input_name).
```

В этом окне приведены результаты тестирования шести нечетких моделей с такими входами: **Weight** – масса автомобиля; **Disp** – литраж; **Power** – мощность; **Cylinder** – количество цилиндров; **Year** – год выпуска; **Acceler** – ускорение (время разгона до 60 миль/ч). Значения невязки RMSE (Root Mean Squared Error) на обучающей и тестовой выборках обозначены соответственно кружками и звездочками. Численные значения невязок выводятся в рабочую область MATLAB.

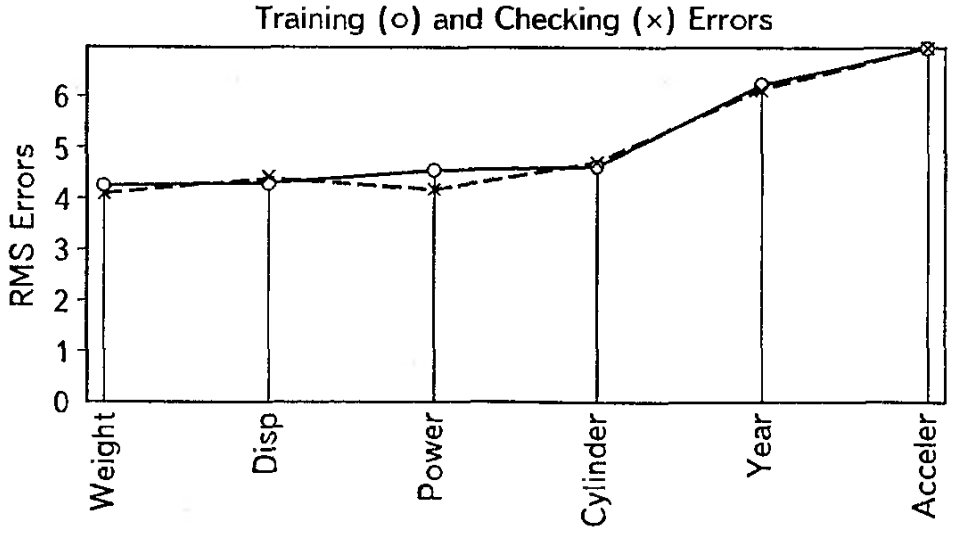


Рис. 3.54. Тестирование грубых моделей «один вход – один выход»

Как видно из графика, наиболее информативным признаком является масса автомобиля, а вторым по рангу – литраж. Невязки на обучающей и тестовой выборках одного порядка, следовательно, можно добавить в модель еще одну входную переменную. Интуитивно мы могли бы просто выбрать массу и литраж автомобиля, так как для них значения ошибок обучения минимальны. Однако такой директивный подход не гарантирует, что выбранная ANFIS-модель с думя входами обеспечит максимальную точность прогнозирования.

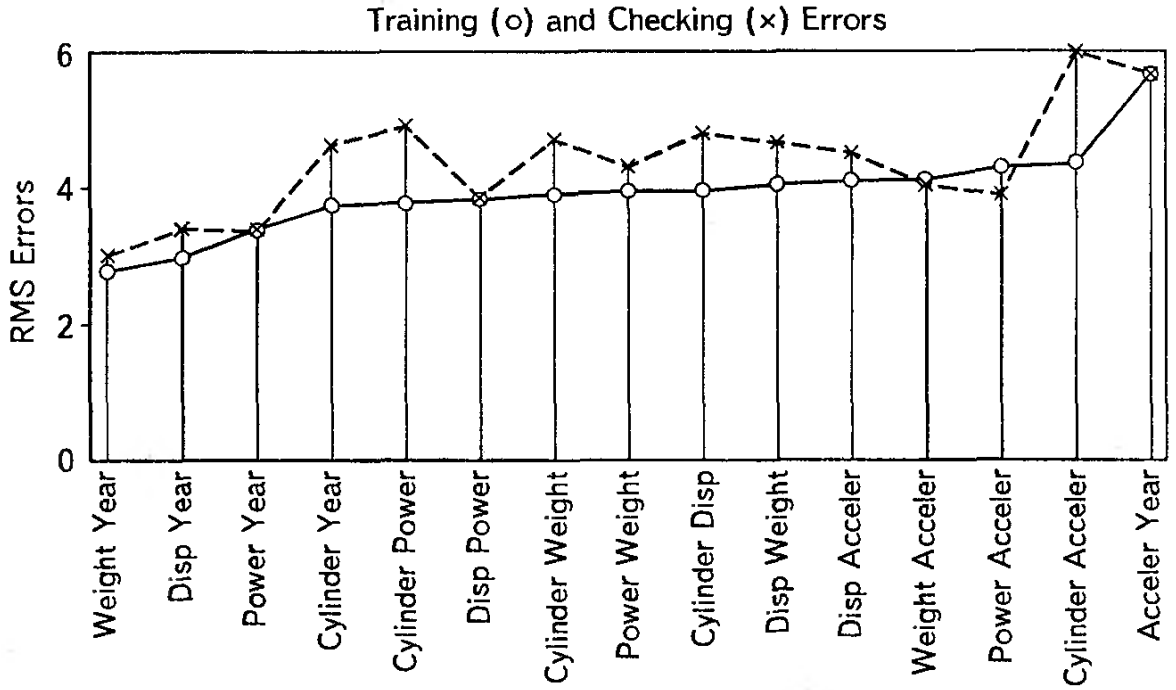


Рис. 3.55. Тестирование грубых моделей «два входа – один выход»

На четвертом слайде описана процедура оценки информативности пар входных переменных. Слайд сопровождается дополнительным графическим окном (рис. 3.55), в котором приводятся результаты тестирования 15 синтезированных моделей «два входа – один выход». Окно является результатом выполнения команды

```
input_index = exhsrch(2, trn_data, chk_data, input_name).
```

Наиболее информативной парой признаков являются масса автомобиля и год выпуска (что не противоречит житейскому опыту). Разница невязок на обучающей и тестовой выборках увеличивается, что сигнализирует о приближении эффекта переусложнения модели.

На пятом слайде исследуется информативность троек входных переменных. В дополнительном окне (рис. 3.56) приводятся результаты тестирования 20 синтезированных моделей «три входа – один выход». Наилучшее прогнозирование обеспечивает модель с такими входными переменными: масса автомобиля, ускорение и год выпуска. Минимальные невязки на обучающей и на тестовой выборках существенно не уменьшились по сравнению с наилучшей моделью «два входа – один выход». Следовательно, добавление еще одной входной переменной к паре масса автомобиля и год выпуска не слишком повысит точность прогнозирова-

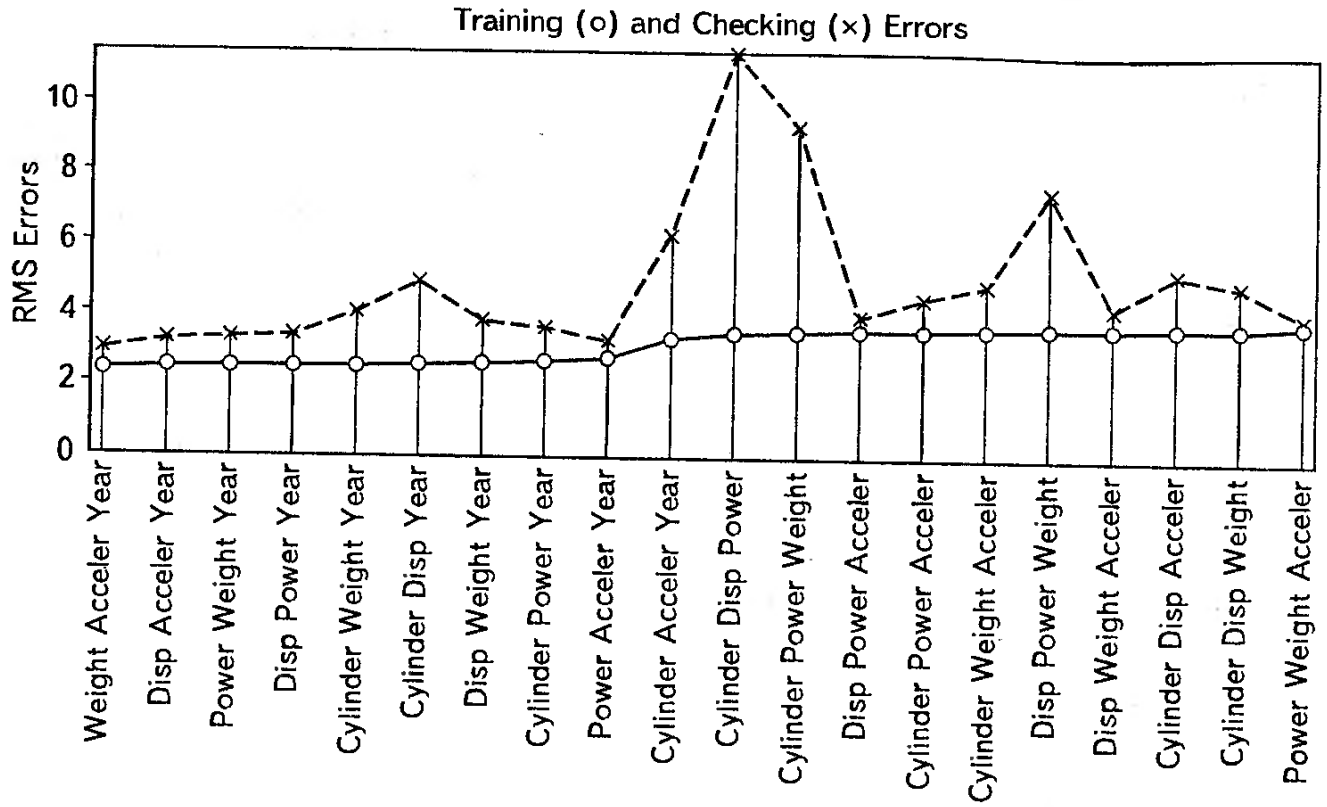


Рис. 3.56. Тестирование грубых моделей «три входа – один выход»

ния топливной эффективности автомобиля. Для обеспечения лучших генерализирующих свойств предпочтительно использование более простых моделей, поэтому для дальнейших исследований выбирается модель «два входа – один выход».

На шестом слайде изображена поверхность «входы – выход» (рис. 3.57) для лучшей двухвходовой ANFIS-модели, учитывающей массу и год выпуска автомобиля. Поверхность является нелинейной и монотонной, причем прогнозируемый уровень топливной эффективности уменьшается с увеличением массы автомобиля и возрастает с увеличением года выпуска. Для рассматриваемой ANFIS-модели значения RMSE на обучающей и тестовой выборках равны 2,766 и 2,995 соответственно. Для сравнения, обычная линейная регрессионная модель, учитывающая все параметры автомобиля, обеспечивает значения RMSE 3,452 и 3,444 на обучающей и тестовой выборках соответственно.

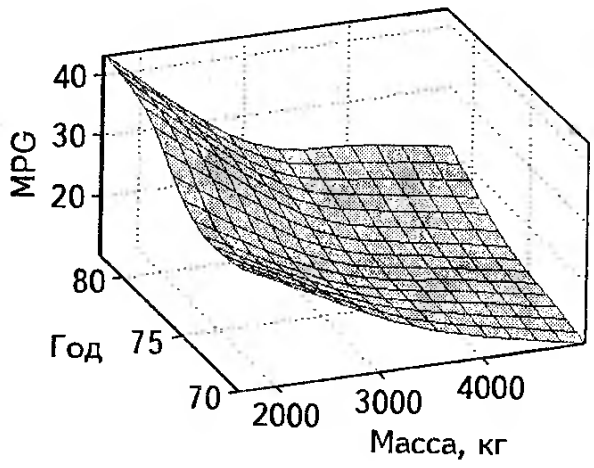


Рис. 3.57. Поверхность «входы – выход» для наилучшей нечеткой модели после одной итерации обучения

На седьмом слайде приведены результаты обучения выбранной двухвходовой модели. На предыдущих этапах модели-кандидаты обучались с помощью функции exhsrch только на протяжении одной итерации ANFIS-алгоритма. Это сделано для быстрого выбора подходящего множества входных переменных. Сейчас, когда входные переменные выбраны, можно потратить

больше времени на обучение. На седьмом слайде показана динамика обучения нечеткой модели в виде зависимостей ошибок обучения (нижняя кривая) и тестирования (верхняя кривая) от количества итераций алгоритма (рис. 3.58). Минимум ошибки тестирования достигается на 53-й итерации алгоритма. Затем ошибка тестирования возрастает, что указывает на эффект переобучения, т.е. на потерю моделью свойств обобщения. Точка минимума отмечена на графике окружностью.

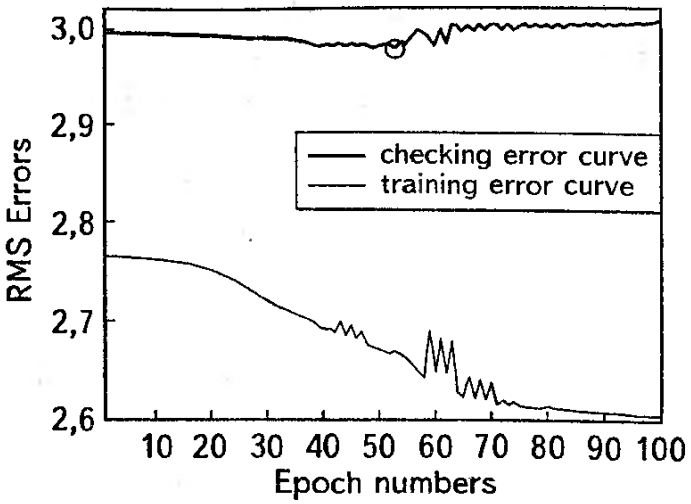


Рис. 3.58. Динамика обучения нечеткой модели

На восьмом слайде показана поверхность «входы – выход» нечеткой модели с минимальной ошибкой тестирования (рис. 3.59). Для этой модели ошибки на обучающей и тестовой выборках равны 2,670 и 2,978, что меньше, чем у первоначальной модели. Однако при рассмотрении дальнего угла поверхности обнаруживается парадоксальный эффект – для новых автомобилей топливная эффективность возрастает с увеличением массы автомобиля. Такое неожиданное поведение модели объясняется отсутствием в обучающей выборке данных в этой области факторного пространства.

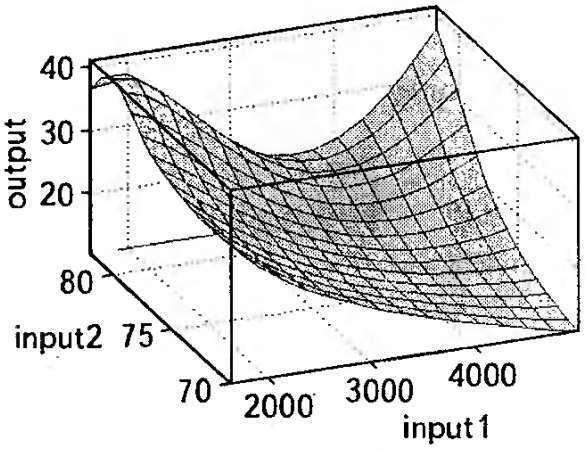


Рис. 3.59. Поверхность «входы – выход» нечеткой модели после обучения

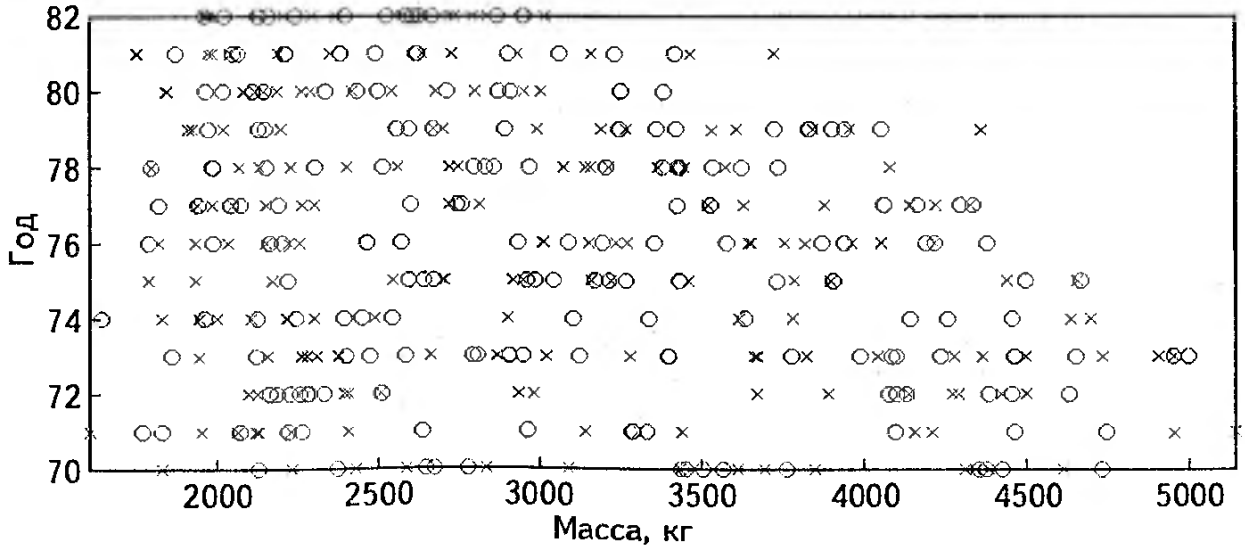


Рис. 3.60. Обучающая (кружки) и тестовая (крестики) выборки

На девятом слайде (см. рис. 3.60) приведено распределение экспериментальных данных из обучающей и тестовой выборок. В правом верхнем углу факторного пространства отсутствуют данные, что и является причиной парадоксального поведения ANFIS-модели. Используя модели, синтезированные с помощью ANFIS, как, впрочем, и с помощью любой другой технологии автоматического обучения, необходимо помнить, что они адекватно могут описывать лишь закономерности, представленные репрезентативными выборками данных.

3.4.3. НЕЛИНЕЙНОЕ ШУМОПОДАВЛЕНИЕ

Демонстрационная программа noisedm иллюстрирует применение технологии ANFIS для нелинейного шумоподавления. При выполнении программы noisedm на экран выводится 10 слайдов.

На первом слайде выводится аннотация программы noisedm. Для запуска демонстрации необходимо нажать кнопку **Start>>**.

На втором слайде показан гипотетический информационный сигнал **x** длительностью 6 с (рис. 3.61). Сигнал задан следующим фрагментом программы:

```
time = (0:0.01:6); x = sin(40./(time+0.01));
```

На третьем слайде указано, что информационный сигнал **x** не может быть измерен без наложения сигнала **n2**. Сигнал **n2** представляет собой нелинейно-искаженный шум **n1** (рис. 3.62), который определен как: `n1 = randn(size(time))`.

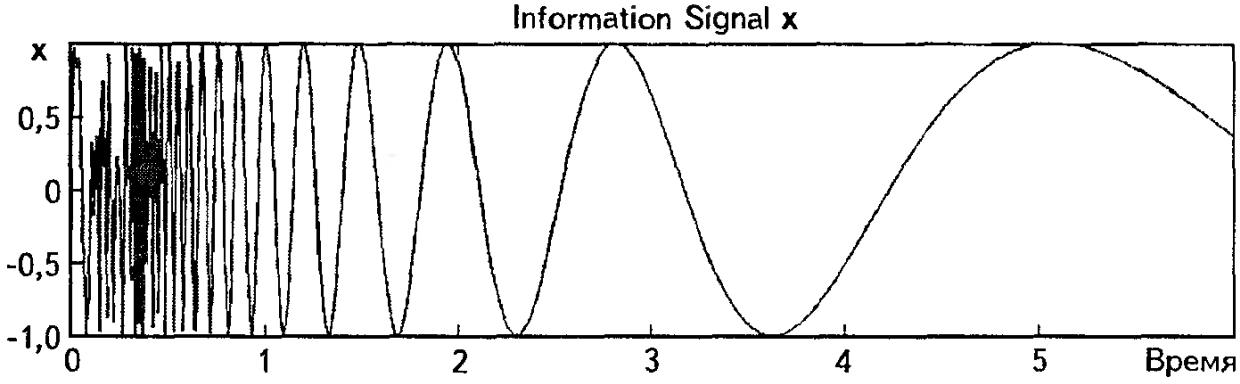


Рис. 3.61. Информационный сигнал без шума

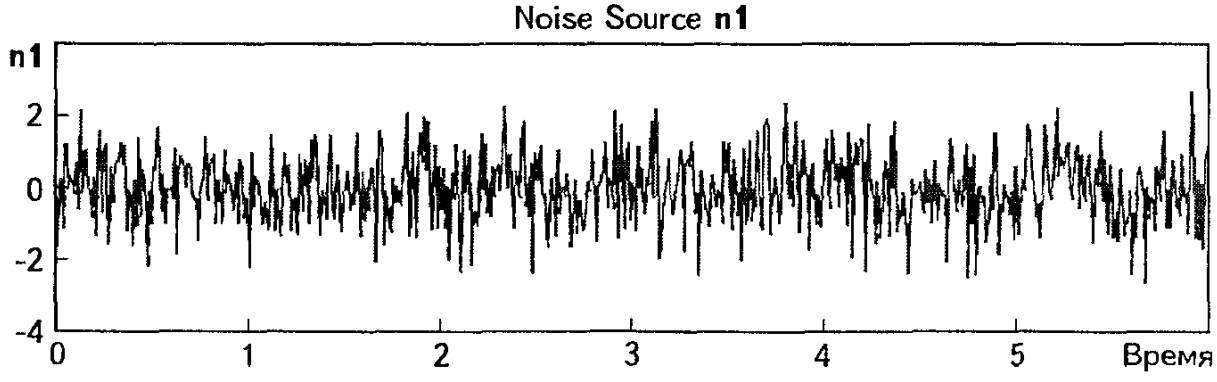


Рис. 3.62. Неискаженный шум

На четвертом слайде показана нелинейная характеристика канала связи (рис. 3.63). При прохождении по этому каналу шум **n1** искажается в шум **n2** по следующему нелинейному закону: `n2(k) = 4*sin(n1(k))*n1(k-1)/(1+n1(k-1)^2)`.

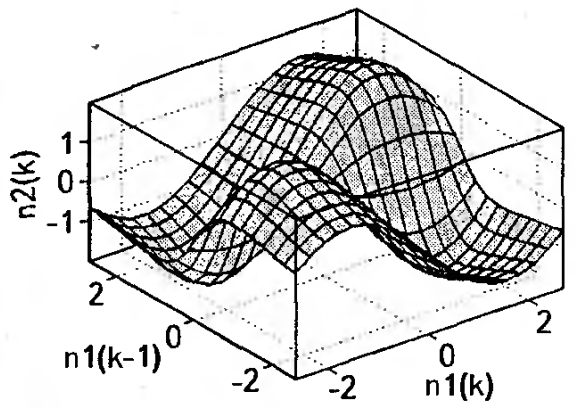


Рис. 3.63. Характеристика канала связи

На пятом слайде показаны исходный и искаженный шумы (рис. 3.64). Хотя сигналы **n1** и **n2** связаны нелинейной зависимостью, показанной на рис. 3.63, визуально это заметить трудно.

На шестом слайде показан измеренный сигнал **m** (рис. 3.65), полученный сложением исходного информационного сигнала **x** с искаженым шумом **n2**. Наша цель состоит в выделении из зашумленного сигнала **m** исходного информационного сигнала **x**. Обратим

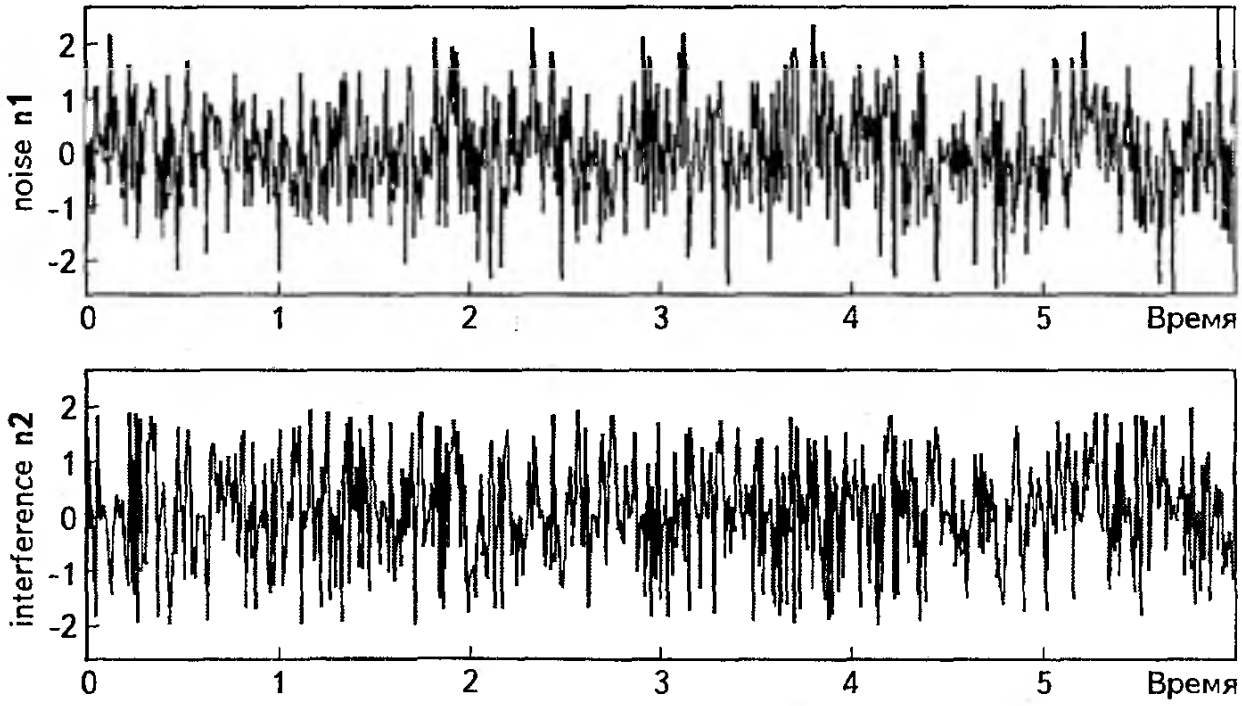


Рис. 3.64. Шум до (вверху) и после (внизу) прохождения через канал связи

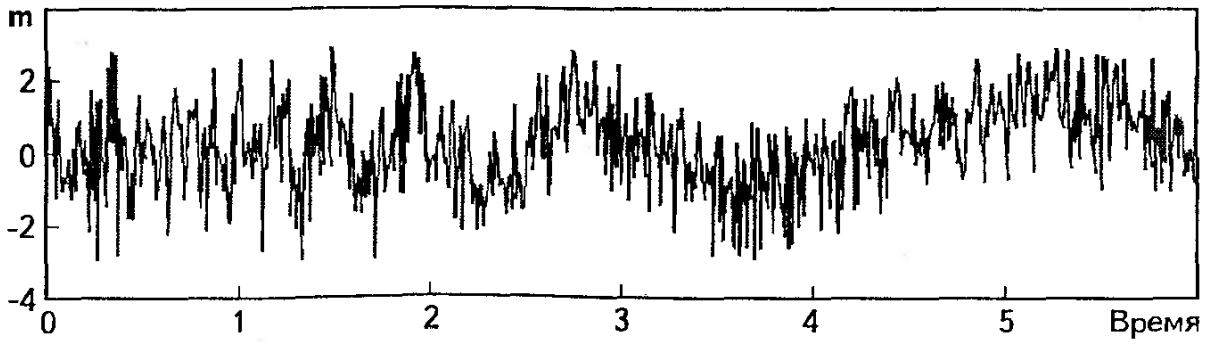


Рис. 3.65. Измеренный сигнал

внимание на то, что шум **n2** неизвестен. Доступной информацией является измеренный сигнал **m** и неискаженный шум **n1**.

На седьмом слайде указано, что для идентификации нелинейной зависимости между сигналами **n1** и **n2** будет использоваться функция anfis. Хотя сигнал **n2** неизвестен, для обучения можно использовать сигнал **m** как зашумленную версию сигнала **n2**. Таким образом, при решении идентификации информационный сигнал **x** будет трактоваться как шум. Будем предполагать, что порядок нелинейности канала связи известен (в нашем случае он равен 2), поэтому необходимо использовать нечеткую модель с двумя входами.

На восьмом слайде приведена программа обучения ANFIS-модели. В нечеткой модели используется по два терма для каждого входа. Следовательно, база знаний состоит из четырех правил. Общее количество настраиваемых параметров равно 24, из которых 12 линейных и 12 нелинейных. Длина шага при обучении установлена равной 0,2. Нечеткая модель обучается на протяжении 10 итераций. Информация о процессе обучения модели выводится в командное окно MATLAB.

На девятом слайде приведены искомый и обнаруженный шумы. На рис. 3.66 искомый шум показан вверху, а обнаруженный – внизу. Шум выделяется функцией evalfis с помощью обученной нечеткой модели. Напомним, что сигнал **n2** на момент моделирования является неизвестным.

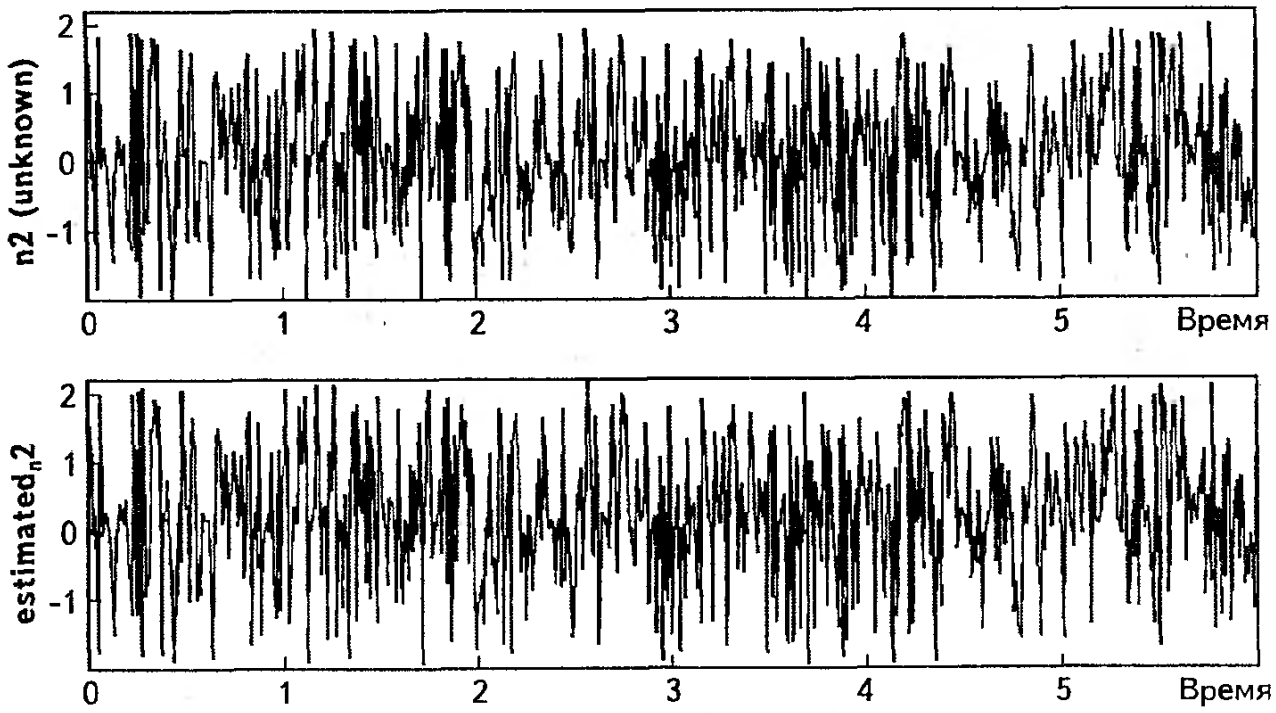


Рис. 3.66. Искомый и обнаруженный шумы

На десятом слайде показан измеренный сигнал после удаления шума. Отфильтрованный сигнал **estimated_x** получен вычитанием из измеренного сигнала **x** шума **estimated_n**, обнаруженного нечеткой моделью. Для сравнения на рис. 3.67 показаны искомый информационный сигнал **x** и отфильтрованный сигнал **estimated_x**.

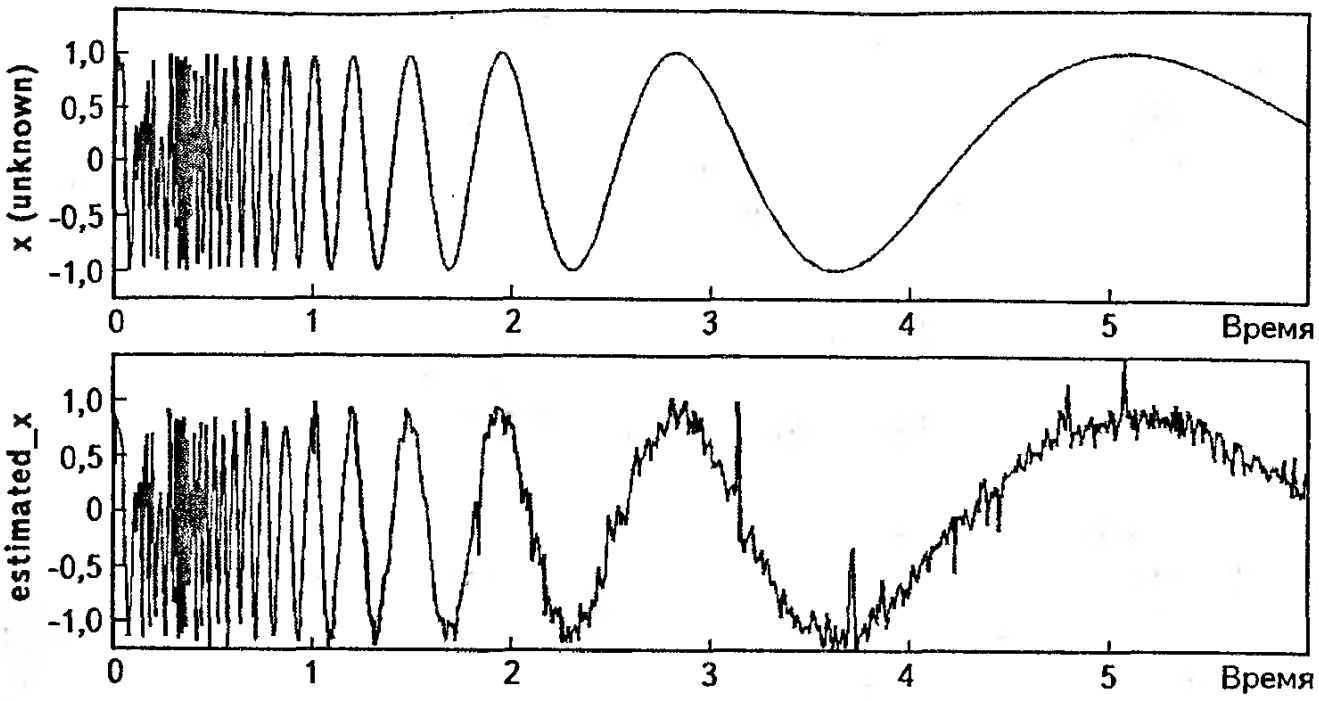


Рис. 3.67. Искомый и отфильтрованный сигналы

Как видно из рисунка, даже без длительного обучения нечеткая модель достаточно хорошо удаляет шумы из измеренного сигнала m.

3.4.4. ПРЕДСКАЗАНИЕ ВРЕМЕННОГО РЯДА

Демонстрационная программа mgtsdemo иллюстрирует применение технологии ANFIS для предсказания временного ряда Маккея – Глэсса (Mackey – Glass). При выполнении программы mgtsdemo на экран выводится девять слайдов.

На первом слайде выводится аннотация программы mgtsdemo. Для запуска демонстрации необходимо нажать кнопку **Start>>**.

На втором слайде описывается временной ряд Маккея – Глэсса, фрагмент которого показан на рис. 3.68. Дифференциальное уравнение этого ряда имеет вид:

$$\frac{dx(t)}{dt} = \frac{0,2x(t-\tau)}{1+x^{10}(t-\tau)} - 0,1x(t).$$

При значениях $x(0) = 1,2$ и $\tau = 17$ ряд получается непериодическим и неконвергентным, динамика которого очень чувствительна к начальным условиям. Временной ряд на рис. 3.68 получен в предположении, что $x(t) = 0$ при $t < 0$.

На третьем слайде указано, что ANFIS-технология будет использована для построения модели, предсказывающей значение ряда через шесть интервалов времени $x(t+6)$ на основе предыдущих четырех наблюдений временного ряда: $x(t-18)$, $x(t-12)$, $x(t-6)$ и $x(t)$. Следовательно, необходимо использовать такой формат обучающей выборки: $[x(t-18)\ \ x(t-12)\ \ x(t-6)\ \ x(t)\ \ x(t+6)]$. В интервале от $t = 118$ до $t = 1117$ собрано 1000 пар данных в указанном формате. Первые 500 пар используются как обучающая выборка, а остальные – как тестовая. На тре-

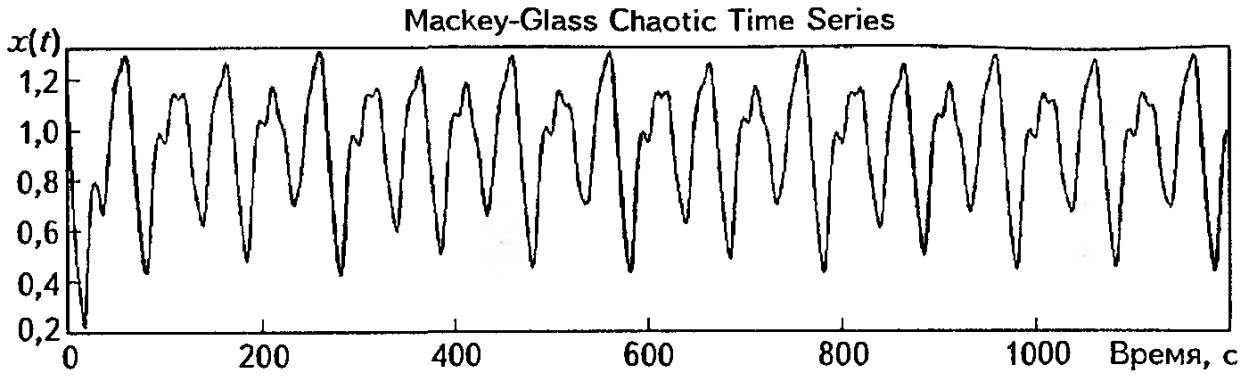


Рис. 3.68. Временной ряд Маккея – Глэсса

тьем слайде показан график временного ряда на промежутке [118; 1117] т.е. на интервале, из которого сформированы обучающая и тестовая выборки.

На четвертом слайде из обучающей выборки `trn_dat` генерируется исходная нечеткая модель по команде `fismat=genfis1(trn_data)`. Для лингвистической оценки входных переменных используется по два терма с колоколообразными функциями принадлежности, графики которых (рис. 3.69*а*) приведены на четвертом слайде.

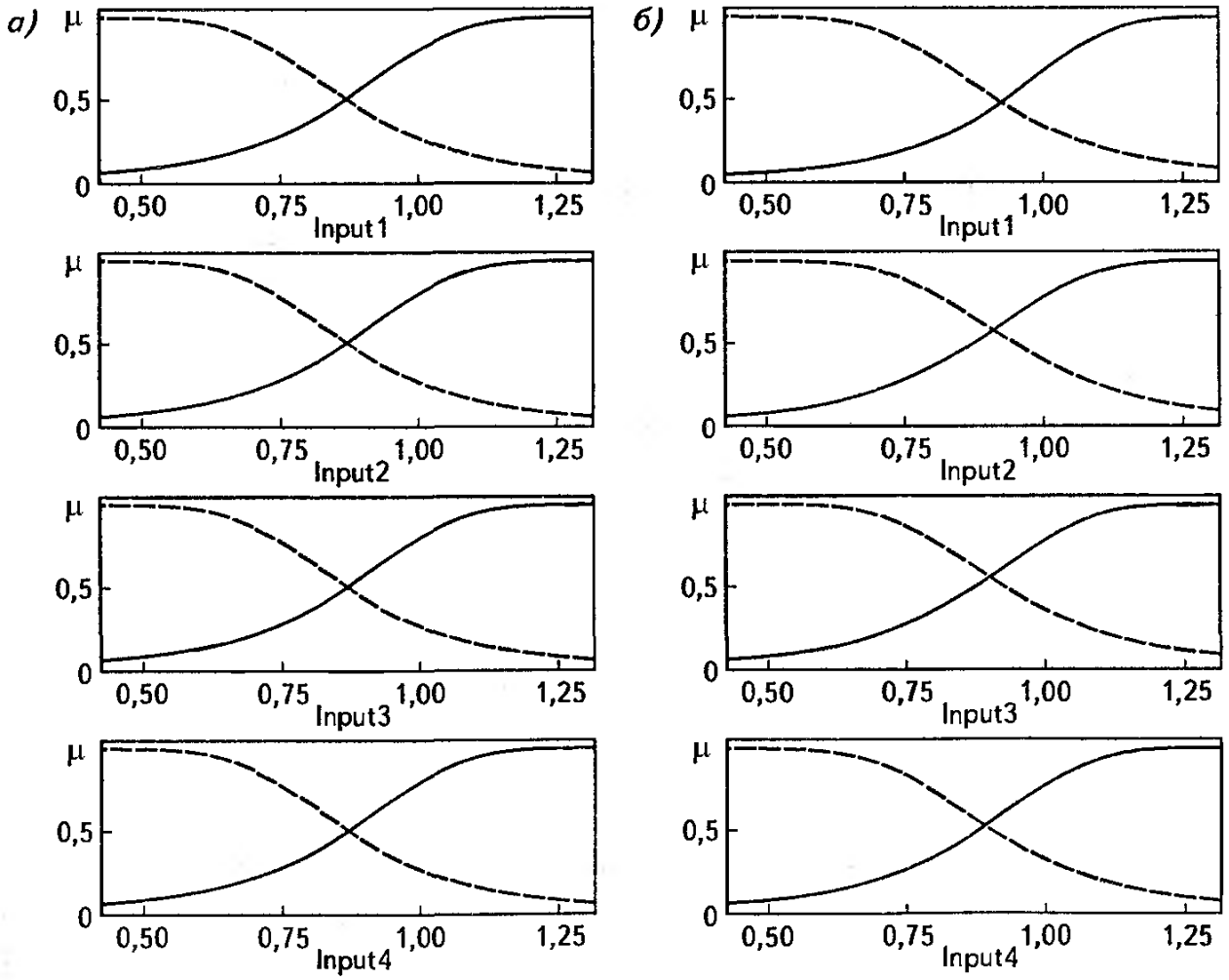


Рис. 3.69. Функции принадлежности нечеткой модели предсказания временного ряда

а – до обучения; *б* – после обучения

На пятом слайде указано, что сгенерированная нечеткая база знаний содержит $2^4 = 16$ правил. Количество настраиваемых параметров нечеткой модели равно 104: 24 нелинейных и 80 линейных. Такое число настраиваемых параметров хорошо сбалансировано с объемом обучающей выборки, составляющей 500 пар «входы – выход». Обучение нечеткой модели осуществляется по команде:

```
[trn_fismat,trn_error] = anfis(trn_data, fismat, [], [],...
chk_data).
```

На десять итераций обучения затрачивается 4 мин. на компьютере SUN SPARC II workstation. Для экономии времени нечеткая модель в демо-показе не обучается; результаты обучения напрямую загружаются в память.

На шестом слайде приведены графики оптимальных функций принадлежности (рис. 3.69*б*), полученные после 10 итераций алгоритма обучения. Они не сильно изменились по сравнению с исходными функциями принадлежности. Очевидно, что основное обучение произошло путем изменения линейных параметров нечеткой модели. Нелинейные параметры обычно сильно изменяются при продолжительной настройке, которая может быть выполнена позже.

На седьмом слайде показана динамика обучения нечеткой модели – графики изменения ошибки RMSE (рис. 3.70) на обучающей (*1*) и тестовой (*2*) выборках. Обратим внимание, что ошибка на тестовой выборке меньше, чем на обучающей. Такой эффект нетипичен не только для ANFIS-обучения, но и для нелинейной аппроксимации вообще. Возможно, процесс обучения еще далек от завершения.

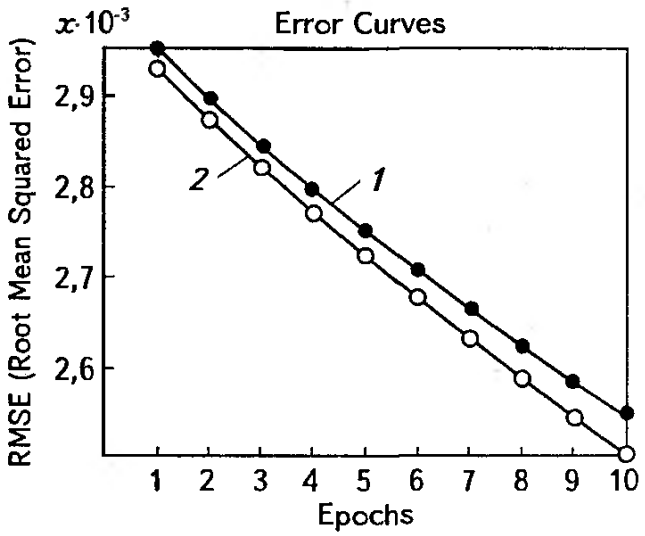


Рис. 3.70. Динамика обучения нечеткой модели предсказания временного ряда

На восьмом слайде показаны исходный временной ряд Маккея – Глэсса и ряд, предсказанный обученной нечеткой моделью. Разница между графиками на рис. 3.71 такая незначительная, что визуально заметить ее невозможно. Для наблюдения ошибки предсказания необходимо изменить масштаб.

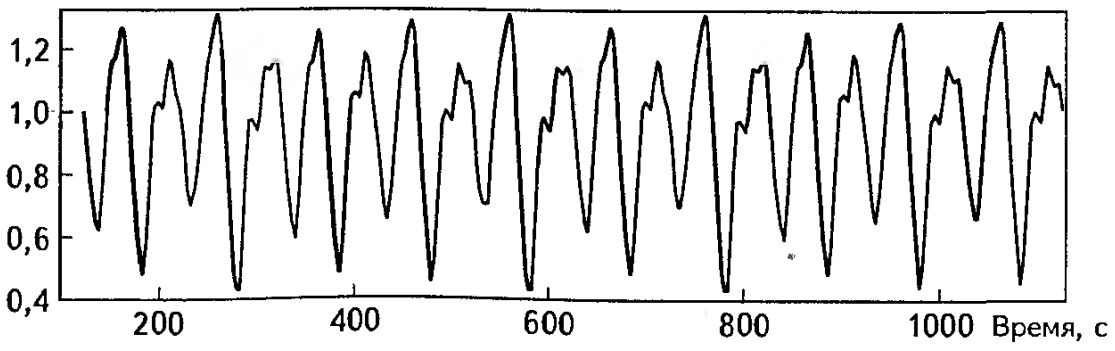


Рис. 3.71. Исходный и предсказанный временные ряды

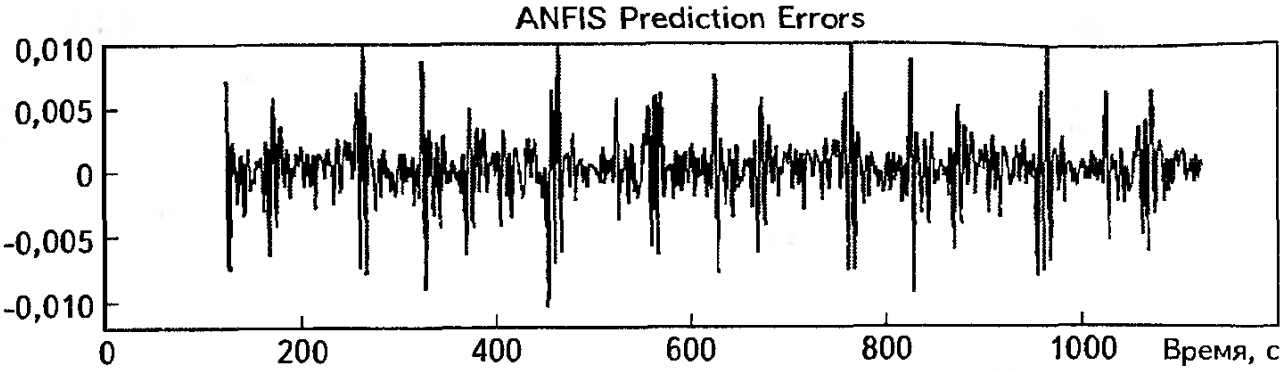


Рис. 3.72. Ошибка предсказания временного ряда нечеткой моделью после обучения

На девятом слайде показан график ошибки предсказания обученной нечеткой модели (рис. 3.72). Обратим внимание, что масштаб этого слайда почти в 100 раз крупнее предыдущего. Напомним, что обучение нечеткой модели было прервано после 10 итераций. Ожидается, что ошибку предсказания можно уменьшить дальнейшим обучением нечеткой модели.

3.4.5. ПРОГНОЗИРОВАНИЕ КОЛИЧЕСТВА АВТОМОБИЛЬНЫХ ПОЕЗДОК

Демонстрационная программа trips иллюстрирует применение субтрактивной кластеризации для синтеза нечеткой модели прогнозирования количества автомобильных поездок. Субтрактивная кластеризация может использоваться как быстрый метод синтеза нечетких правил из данных. Кроме того, этот метод может рассматриваться как своего рода препроцессинг для ANFIS-алгоритма – синтезированная нечеткая модель является начальной точкой для обучения. Преимуществом кластеризации при синтезе нечеткой модели является то, что правила базы знаний получаются объектно-ориентированными. Это снижает возможность «комбинаторного взрыва» – катастрофического увеличения объема базы знаний при большом числе входных переменных. При выполнении демонстрационной программы trips на экран выводится семь слайдов.

На первом слайде выводится аннотация программы trips. Для запуска демонстрации необходимо нажать кнопку **Start>>**.

На втором слайде указано, что задача состоит в идентификации зависимости числа заявок на автомобильные поездки из региона от следующих демографических показателей местности: количество жителей; количество домов; количество автомобилей; уровень доходов; уровень занятости населения. На слайде обучающая выборка показана в виде пяти графиков значений входных переменных.

На третьем слайде сообщается, что с помощью субтрактивной кластеризации из данных генерируется система нечеткого вывода типа Сугено. Для этого используется функция genfis2. Сгенерированная нечеткая система позволяет рассчитать значение выходной переменной на основе данных о пяти входных переменных. База знаний системы содержит всего четыре нечетких правила. Структура сгенерированной нечеткой системы показана на рис. 3.73.

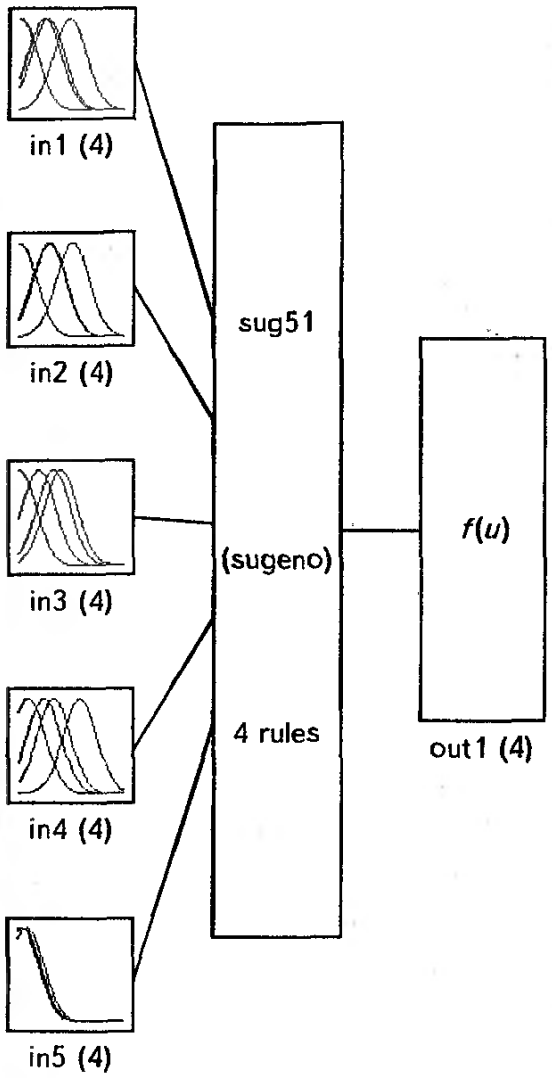


Рис. 3.73. Структура нечеткой системы прогнозирования количества автомобильных поездок

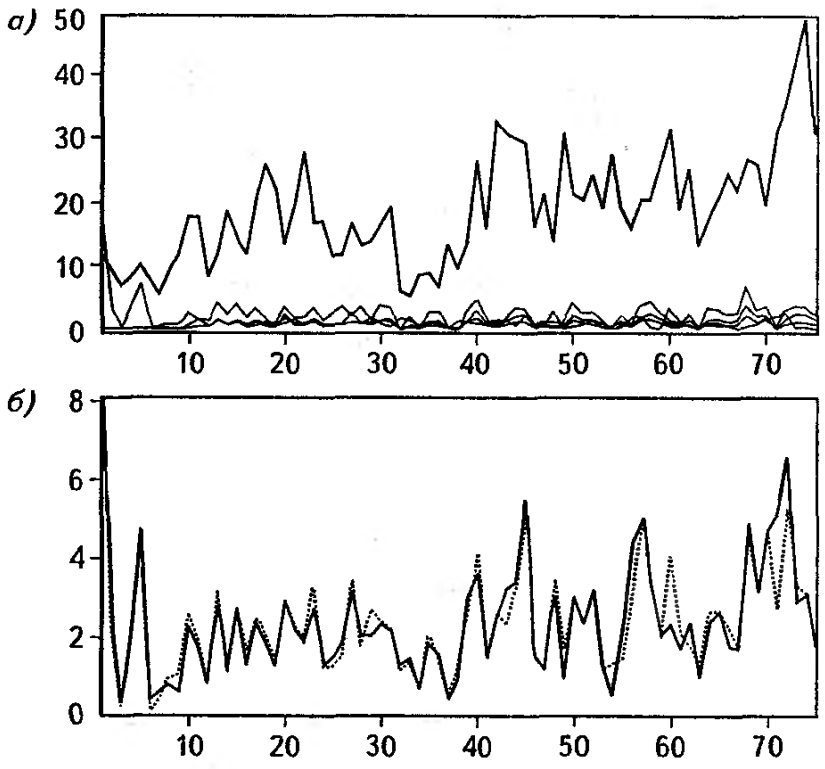


Рис. 3.74. Тестирование системы на обучающей выборке

На четвертом слайде (рис. 3.74) показаны графики тестирования сгенерированной системы на обучающей выборке из 75 пар «входы – выход». На рис. 3.74*а* изображены пять графиков значений входных переменных из обучающей выборки, на рис. 3.74*б* – экспериментальные и модельные значения выходной переменной. Видно, что нечеткая модель хорошо описывает идентифицируемую зависимость.

На пятом слайде сравниваются действительные и модельные значения выходной переменной (рис. 3.75). Маркер в виде крестика соответствует одной паре данных из обучающей выборки. По оси абсцисс откладываются действительные значения выходной переменной, а по оси ординат – модельные значения для одного и того же набора входных данных. В идеальном случае все маркеры расположились бы на диагонали первого квадранта.

На шестом слайде проверяются генерализирующие свойства синтезированной нечеткой модели на тестовой выборке из 25 пар «входы – выход», неиспользованных при идентификации. Результаты тестирования в виде графика «действительные – модельные значения» показаны на рис. 3.75*б*. Видно, что модель хорошо работает и вне точек обучения. Обратим внимание, что используется грубая,

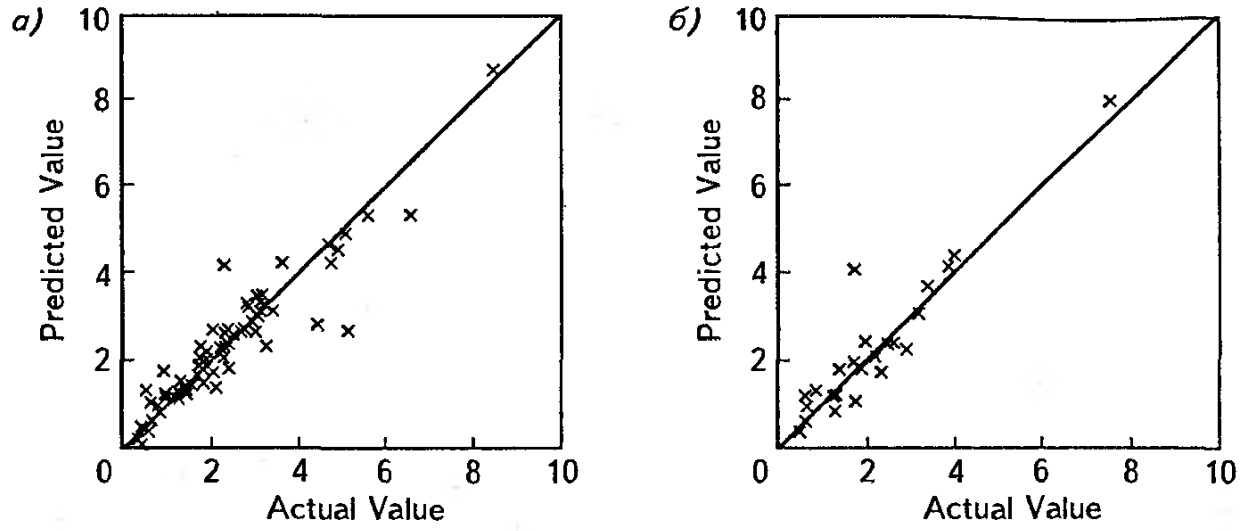


Рис. 3.75. График «действительные – модельные значения»

а – обучающая выборка; *б* – тестовая выборка

ненастроенная нечеткая модель, сгенерированная из экспериментальных данных алгоритмом субтрактивной кластеризации.

На седьмом слайде написано, что кластеризация может быть эффективной технологией обработки больших массивов данных. Основная идея кластеризации заключается в группировании данных в сгустки полезной информации, дающих компактную модель исследуемого объекта. Данная демонстрационная программа показывает способ достижения высокой точности прогнозирования в условиях многофакторной зависимости «входы – выход» без обучения нечетких моделей.

3.4.6. ИДЕНТИФИКАЦИЯ ПРОЦЕССА НАГРЕВА ВОЗДУХА В ФЕНЕ

Демонстрационная программа drydemo иллюстрирует применение технологии ANFIS для идентификации нелинейных динамических систем на примере процесса нагрева воздуха в фене. Результаты нечеткой идентификации сравниваются с линейной ARX-моделью. Для работы drydemo необходим пакет System Identification Toolbox. При выполнении демонстрационной программы drydemo на экран выводится 10 слайдов.

Рис. 3.76. Схема процесса нагрева воздуха

На первом слайде выводится аннотация программы и схема процесса нагрева воздуха (рис. 3.76). Для запуска демонстрации необходимо нажать кнопку **Start>>**.

На втором слайде указано, что данные для идентификации зависимости собраны по методике из главы 17 книги [30] с помощью «Feedback's Process Trainer PT 326» – лабораторного устройства для изучения процессов с обратной связью. Устройство функционирует как фен для сушки волос: воздух нагнетается

во внутрь и проходит через нагревательный элемент, расположенный на входе устройства. Температура воздуха измеряется термопарой на выходе устройства. На рис. 3.77 приведена выборка данных для идентификации зависимости между $u(k)$ – напряжением на нагревательном элементе – проволочном резисторе и $y(k)$ – температурой воздуха на выходе.

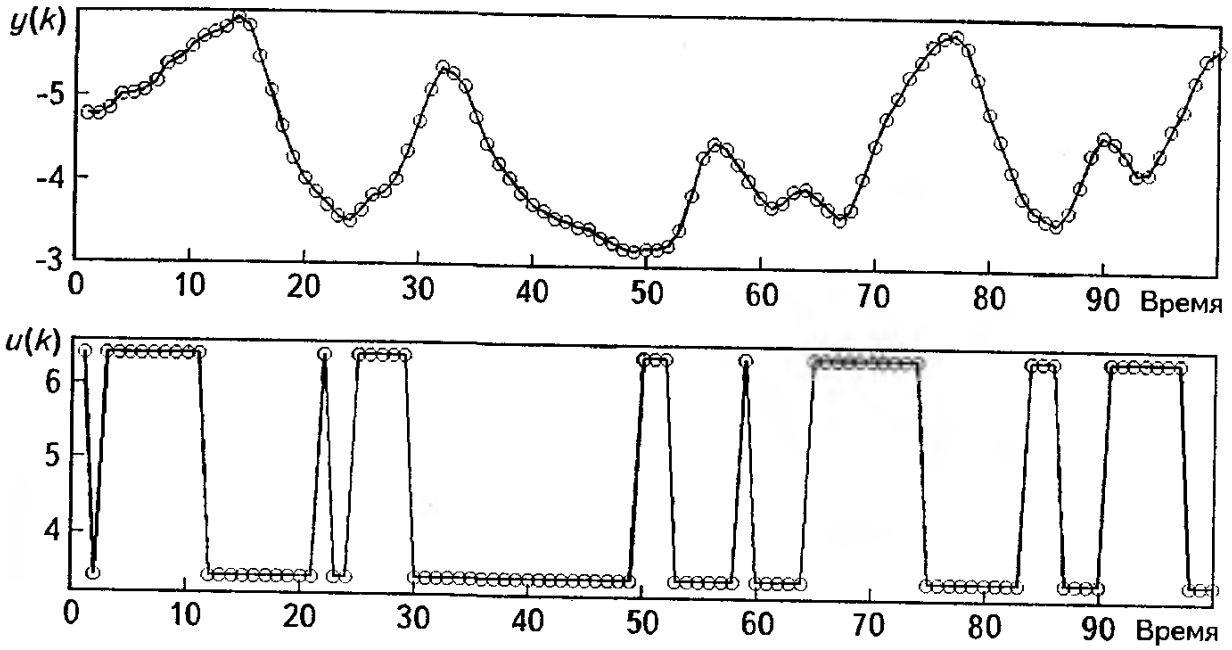


Рис. 3.77. Временные диаграммы температуры воздуха (y) и напряжения на резисторе (u)

На третьем слайде описано, что 1000 пар данных «вход – выход» собраны с интервалом стробирования 0,08 с. Напряжение на резисторе $u(k)$ выбиралось случайно из двух значений 3,41 В и 6,41 В. Вероятность смены уровня напряжения на каждом временном отсчете равнялась 0,2. Графики напряжения на резисторе и температуры воздуха на выходе фена для первых 100 временных отсчетов показаны на рис. 3.77. Полная выборка данных приведена в System Identification Toolbox.

На четвертом слайде описывается типовой процесс построения ARX-модели т.е. поиск линейной зависимости вида:

$$y(k)+a_1 y(k-1)+\ldots+a_m y(k-m)=b_1 u(k-d)+\ldots+b_n u(k-d-n+1),$$

где a_i $(i=1, \ldots, m)$ и b_j $(j=1, \ldots, n)$ – линейные параметры, определяемые методом наименьших квадратов.

Структура ARX-модели однозначно определяется тремя числами m, n и d. Для нахождения ARX-модели фена экспериментальные данные были разделены на две выборки: обучающую ($k = 1, \ldots, 300$) и тестовую ($k = 301, \ldots, 600$). Поиск наилучшей структуры модели выполнен полным перебором комбинаций значений (m, n, d), при котором параметры изменялись от 1 до 10 независимо друг от друга. Наилучшая ARX-модель получена при $m = 5$, $n = 10$ и $d = 2$. Значение RMSE для этой модели равно 0,1122 на обучающей выборке и 0,0749 – на тестовой вы-

борке. Прогнозирование температуры воздуха наилучшей ARX-моделью приведено на рис. 3.78. В командном окне выводится время, затраченное компьютером на синтез оптимальной ARX-модели.

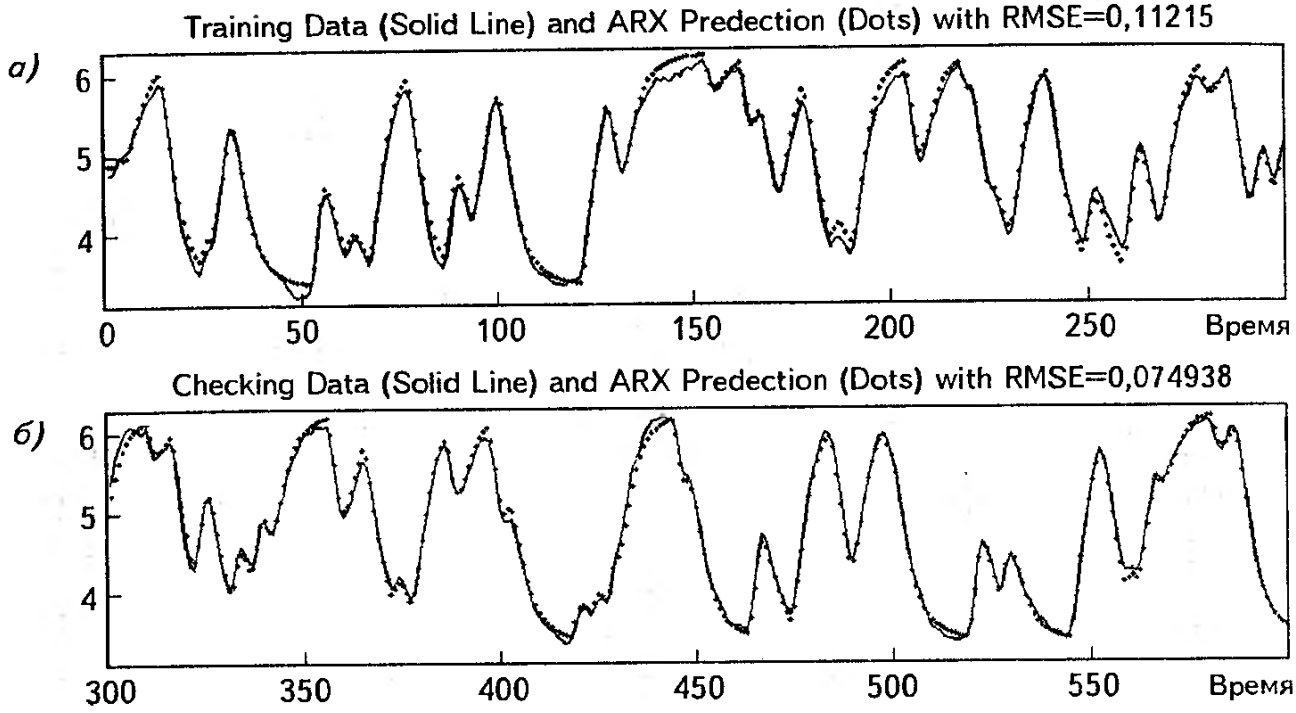


Рис. 3.78. Сравнение экспериментальных данных с результатами ARX-моделирования на обучающей (*а*) и тестовой (*б*) выборках

На пятом слайде подчеркивается, что ARX-модель является линейной; ее основное преимущество – высокая скорость структурной и параметрической идентификации. Как видно из графиков на рис. 3.78, точность прогнозирования достаточно высокая. Однако, если бы нелинейная модель обеспечила лучшую точность, то мы могли бы остановить свой выбор на ней. Попробуем синтезировать ANFIS-модель, чтобы удостоверится, можно ли обеспечить более высокую точность идентификации на основе нечеткого вывода.

На шестом слайде написано, что первым шагом идентификации с помощью ANFIS-технологии является выбор входных переменных нечеткой модели. Для сокращения времени идентификации предположим, что на выходную переменную $y(k)$ могут оказать влияние 10 кандидатов в входные переменные: $y(k-1)$, $y(k-2)$, $y(k-3)$, $y(k-4)$, $u(k-1)$, $u(k-2)$, $u(k-3)$, $u(k-4)$, $u(k-5)$ и $u(k-6)$. Входные переменные модели выбираются эвристически с помощью последовательного поиска вперед, реализованного функцией seqsrch. При таком поиске на каждом шаге в модель добавляется одна входная переменная, обеспечивающая минимальное значение RMSE. Результаты поиска показаны в новом графическом окне (рис. 3.79). На этом рисунке маркеры в виде кружков соответствуют значениям ошибки на обучающей выборке, а в виде крестиков – значениям ошибки на тестовой выборке. В качестве входов модели выбраны $y(k-1)$, $u(k-3)$ и $u(k-4)$. Для этой модели значения RMSE равны 0,0609 на обучающей выборке и 0,0604 на

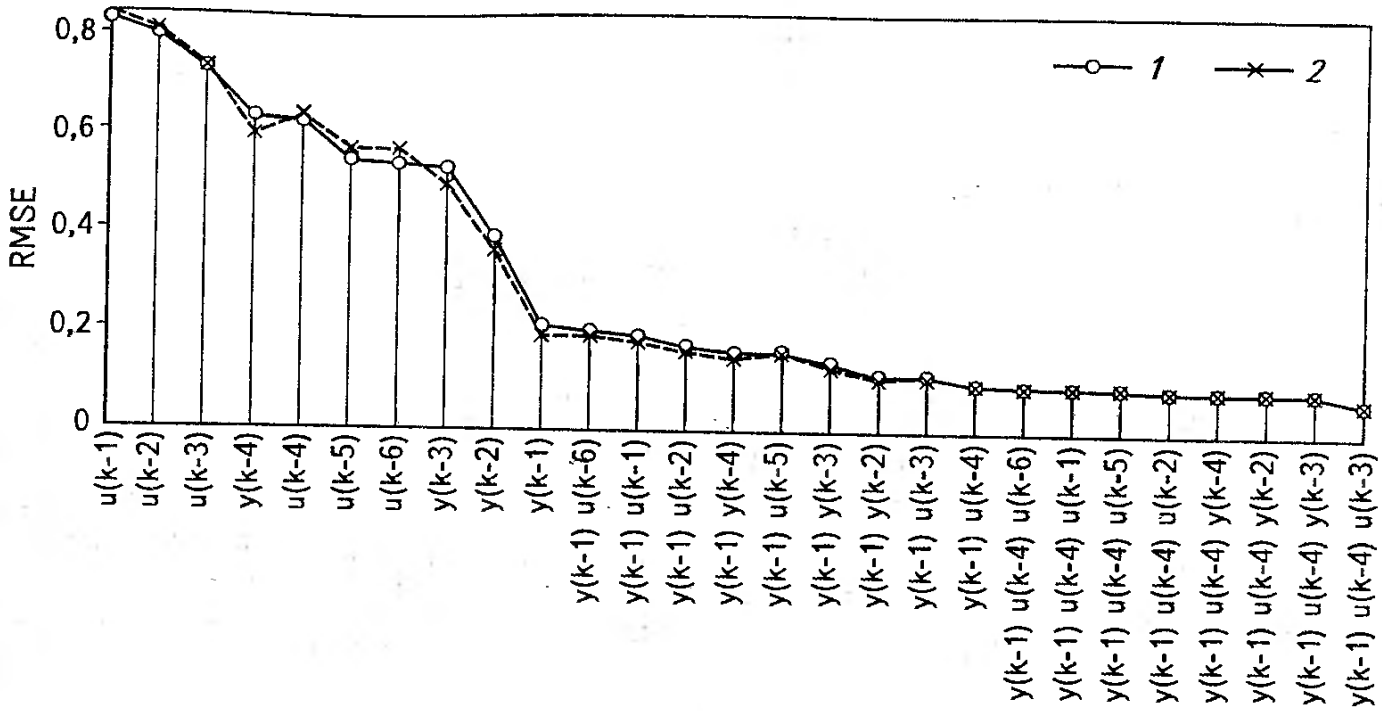


Рис. 3.79. Селекция входов нейро-нечеткой модели методом последовательного поиска вперед

1 – обучающая выборка; *2* – тестовая выборка

тестовой. В командном окне выводится протокол селекции входных переменных с указанием времени поиска.

На седьмом слайде указано, что селекцию входов модели можно осуществить функцией exhsrch, выполняющей полный перебор комбинаций переменных–кандидатов. Такая селекция занимает продолжительное время, например, при выборе трех входных переменных из десяти возможных необходимо синтезировать $C(10, 3) = 20$ нейро-нечетких моделей–кандидатов.

На восьмом слайде написано, что согласно теории идентификации динамических систем входными переменными модели не могут быть элементы только одного из двух следующих множеств: $Y = \{y(k-1), y(k-2), y(k-3), y(k-4)\}$ и $U = \{u(k-1), u(k-2), u(k-3), u(k-4), u(k-5), u(k-6)\}$. Разумным путем угадывания хорошего набора входов нейро-нечеткой модели будет выбор двух элементов из множества Y и одного из множества U. В этом случае общее количество моделей–кандидатов будет $C(4,2) \cdot 6 = 36$, что намного меньше, чем при полном переборе. Протокол работы описанного метода селекции входов выводится в командном окне. Результаты поиска показаны в новом графическом окне (рис. 3.80). В качестве входов модели выбраны $y(k-1)$, $y(k-2)$ и $u(k-3)$. Для этой модели значения RMSE равны 0,0474 на обучающей выборке и 0,0485 на тестовой. Точность моделирования существенно выше, чем при идентификации линейной ARX-моделью или ANFIS-моделью с селекцией входов методом последовательного поиска вперед.

На девятом слайде показаны результаты ANFIS моделирования температуры воздуха на обучающей и тестовой выборках (рис. 3.81). Очевидно, что точность нечеткого моделирования значительно выше, чем для ARX-метода (см. рис. 3.78).

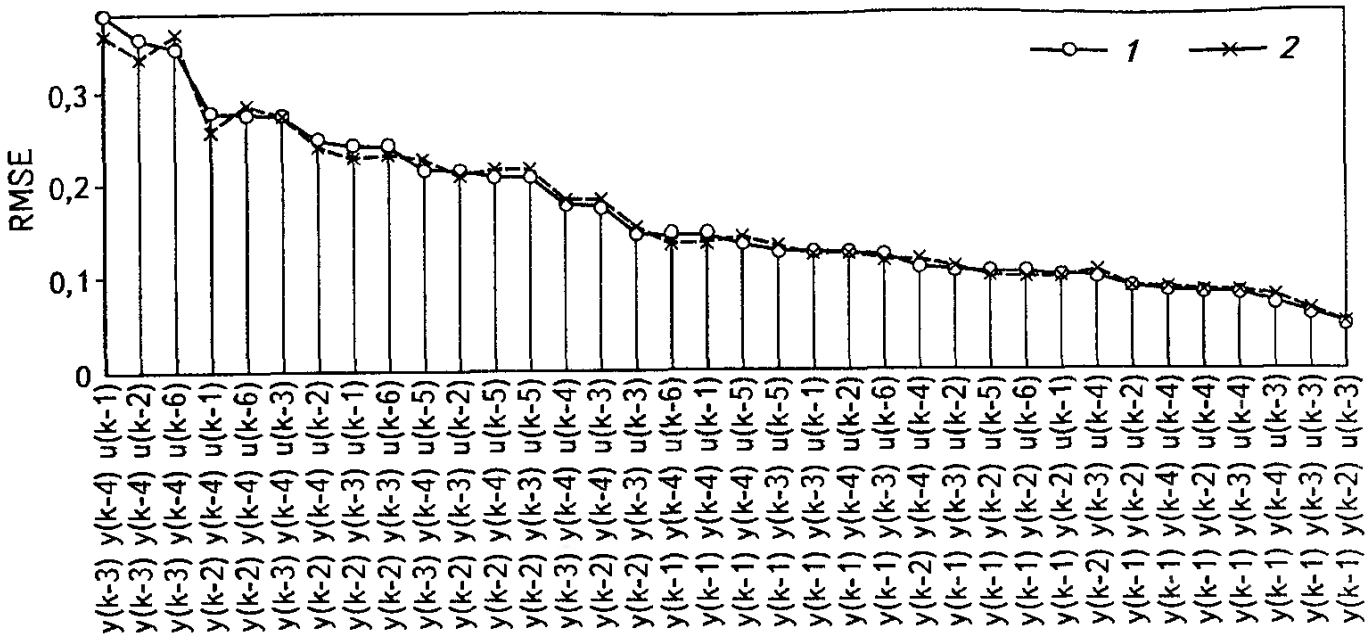


Рис. 3.80. Селекция входов нейро-нечеткой модели методом ограниченного перебора

1 – обучающая выборка; *2* – тестовая выборка

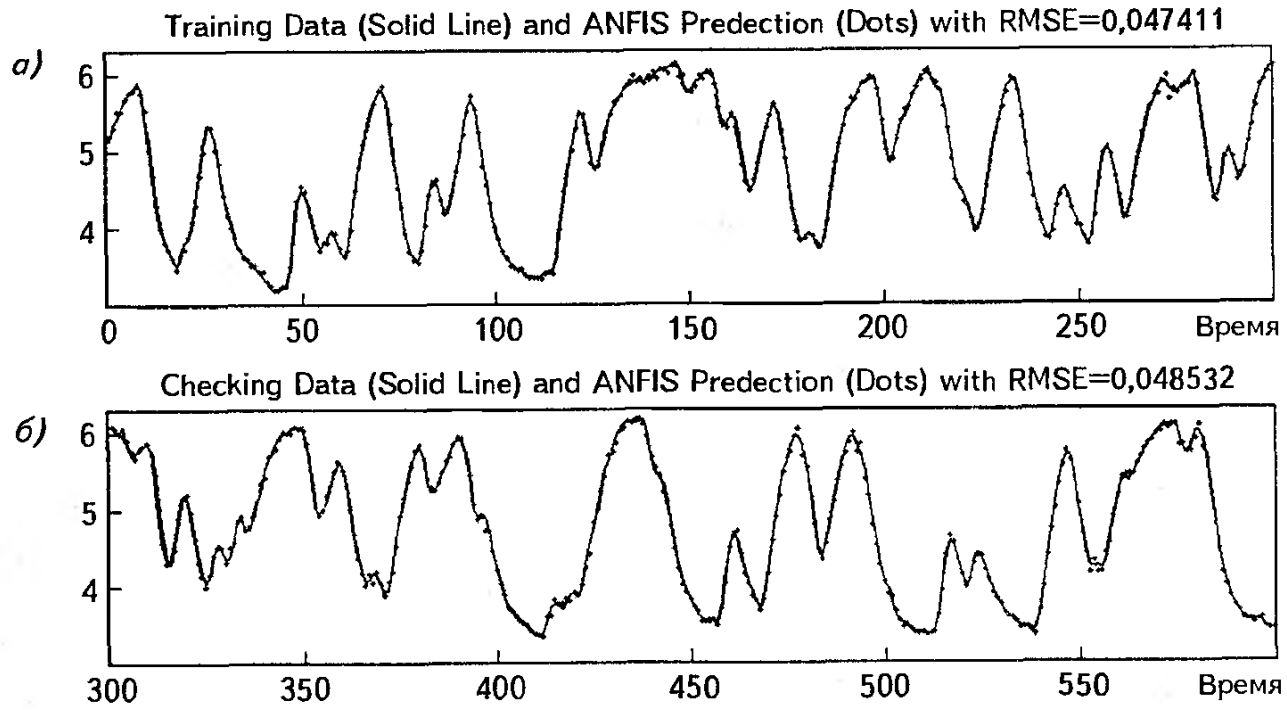


Рис. 3.81. Сравнение экспериментальных данных с результатами ANFIS-моделирования на обучающей (*а*) и тестовой (*б*) выборках

На десятом слайде сравниваются рассмотренные методы идентификации (табл. 3.1). ARX-идентификация самая быстрая, но и точность моделирования значительно хуже, чем для наиболее медленного ANFIS-подхода с селекцией методом ограниченного перебора. Следовательно, для задач, где главным критерием является время идентификации, правильным будет выбор ARX-модели.

Таблица 3.1

Сравнение методов идентификации процесса нагрева воздуха

Метод идентификации	ARX	ANFIS с селекцией методом последовательного поиска вперед	ANFIS с селекцией методом ограниченного перебора
Количество входов модели	14	3	3
RMSE на обучающей выборке	0,1122	0,0609	0,0474
RMSE на тестовой выборке	0,0749	0,0604	0,0485
Количество линейных параметров модели	15	32	32
Количество нелинейных параметров модели	0	18	18
Время идентификации на компьютере PentiumPro (200 МГц, 64 Мб), с	1,6	10,3	21,5

Для задач с высокими требованиями к точности идентификации использование нелинейных ANFIS-моделей будет оправданным.

3.4.7. ЖОНГЛИРОВАНИЕ ТЕННИСНЫМ ШАРИКОМ

Демонстрационная программа juggler иллюстрирует применение нечеткого контроллера для управления теннисной ракеткой при жонглировании шариком. Задача жонглирования состоит в выборе такого угла наклона теннисной ракетки, который обеспечивал бы только вертикальное перемещение шарика возле заданной точки. Угол наклона ракетки можно изменять при каждом соударении с шариком. Ракетка автоматически перемещается в точку пересечения траектории шарика с горизонталью.

После запуска программы juggler появится диалоговое окно, показанное на рис. 3.82. Шарик изображен кружком, теннисная ракетка – прямоугольником. Точка, в окрестности которой необходимо обеспечить вертикальное перемещение шарика, указана серым треугольником. Шарик упруго отражается от ракетки, а также от границ графического окна и белой горизонтальной линии.

Анимация запускается нажатием кнопки **Start Animation...**, которая расположена внизу графического окна. Для вывода траекторий движения шарика и ракетки необходимо установить флажок в окне **Show Trails**. Нажатие кнопки **Clear Trails** стирает следы шарика и ракетки. Управлять углом наклона ракетки можно вручную либо с помощью нечеткого контроллера. Способ управления выбирается через меню **Controller**, содержащее следующие альтернативы: **Fuzzy** – нечеткий контроллер и **Human** – человек–оператор. После начала анимации в нижней части окна появятся две кнопки: **Stop** – прекращение анимации и **Pause...** – пауза в анимации. По нажатии кнопки **Pause...** появятся две новые

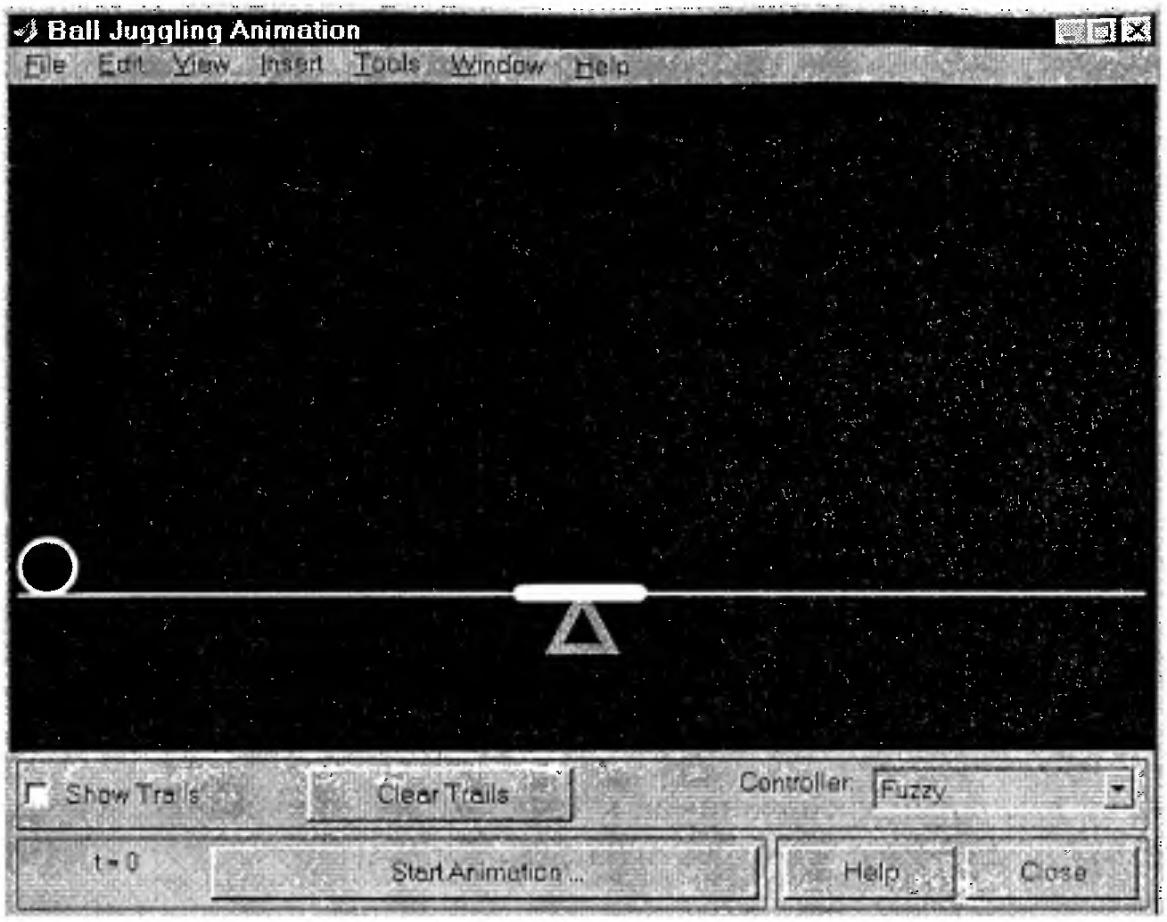


Рис. 3.82. Демонстрационная программа juggler

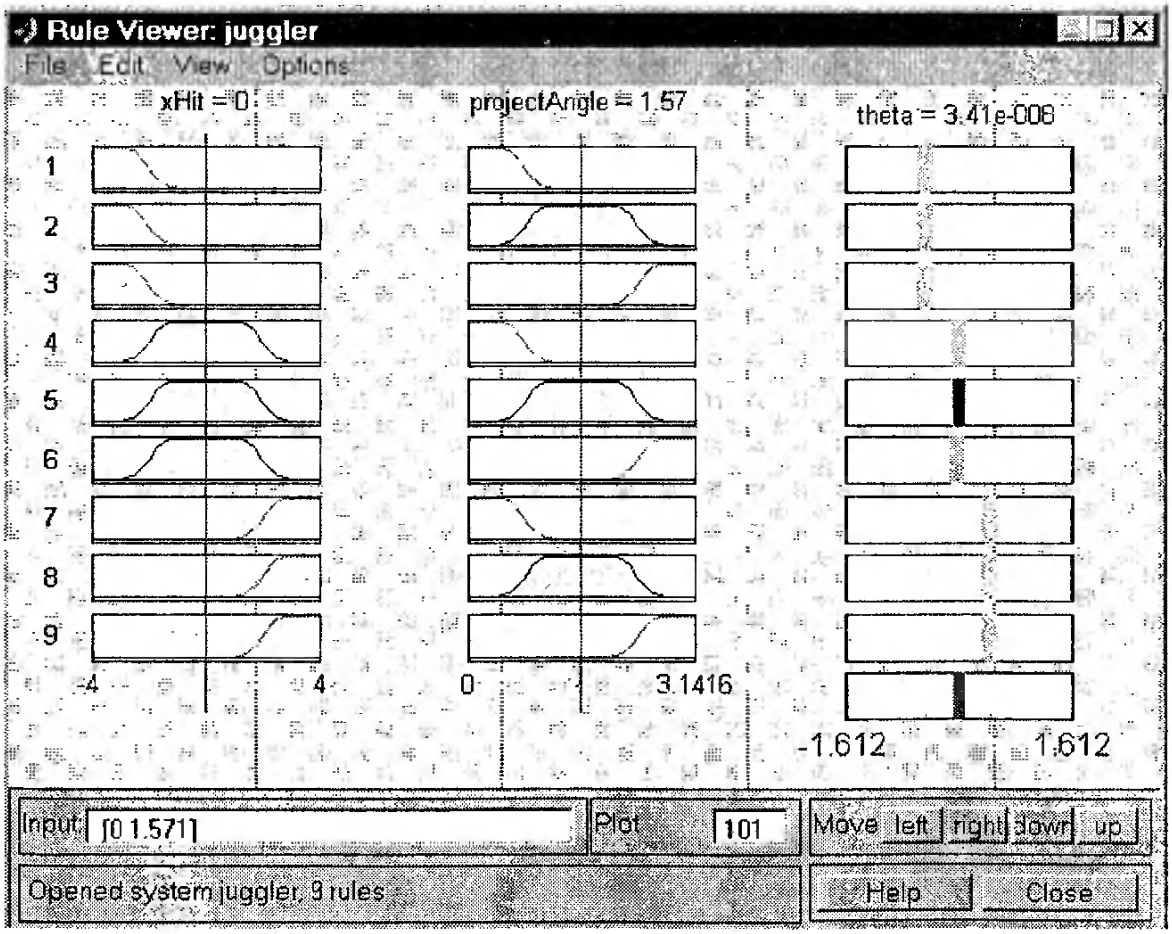


Рис. 3.83. База правил нечеткого контроллера теннисной ракеткой в браузере RuleViewer

кнопки: **Continue** – продолжение анимации и **Step** – выполнение анимации в пошаговом режиме.

Контроллер реализован системой нечеткого вывода Сугено. Она имеет один выход **theta** – угол наклона ракетки и два входа: **xHit** – расстояние от точки пересечения траектории шарика с горизонталью до заданной точки жонглирования и **projectAngle** – угол наклона траектории шарика.

Для лингвистической оценки входных переменных используется по три терма с колоколообразными функциями принадлежности. Нечеткое управление осуществляется по девяти правилам, приведенным на рис. 3.83 в окне модуля RuleViewer. При нечетком управлении перемещение точки жонглирования (серого треугольника) происходит случайно через каждые 200 шагов анимации. Количество пройденных шагов анимации выводится в нижнем левом углу графического окна.

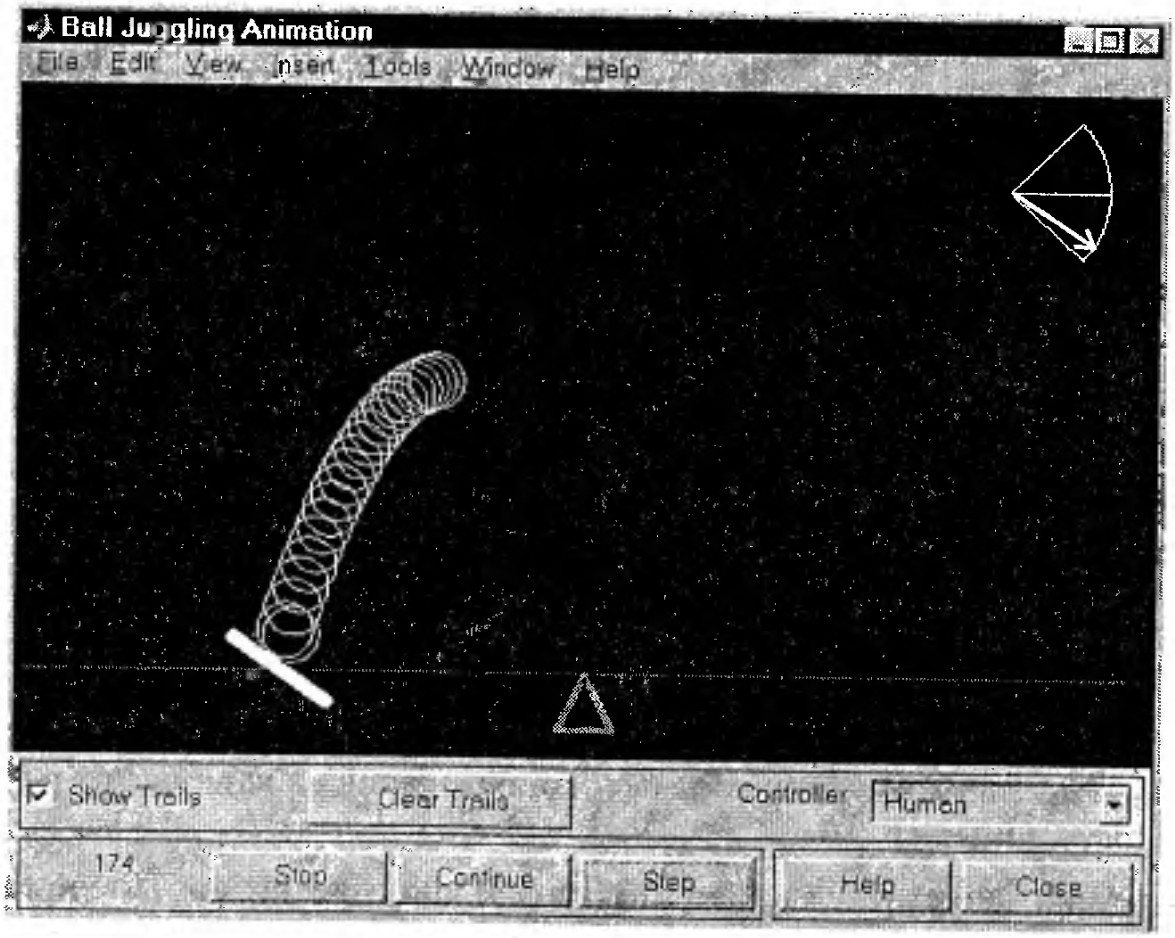


Рис. 3.84. Управление теннисной ракеткой человеком-оператором

При ручном управлении ракеткой в верхнем правом углу графического окна (рис. 3.84) появится «рулевое колесо». Для изменения угла наклона ракетки необходимо переместить мышкой белую линию со стрелкой «рулевого колеса».

3.4.8. УДЕРЖАНИЕ ШАРИКА НА КОРОМЫСЛЕ

Демонстрационная программа slbb иллюстрирует применение нечеткого контроллера для удержания шарика на коромысле. Задача состоит в выборе таких значений силы, прилагаемой к концам коромысла, которые бы обеспечили удержание шарика в состоянии неустойчивого равновесия в заданной точке коромысла. В переходном процессе значение силы может изменяться неоднократно. Для работы программа slbb необходим пакет Simulink.

После запуска программы slbb появится окно пакета Simulink с симулинк-моделью для этого демо-примера (рис. 3.85).

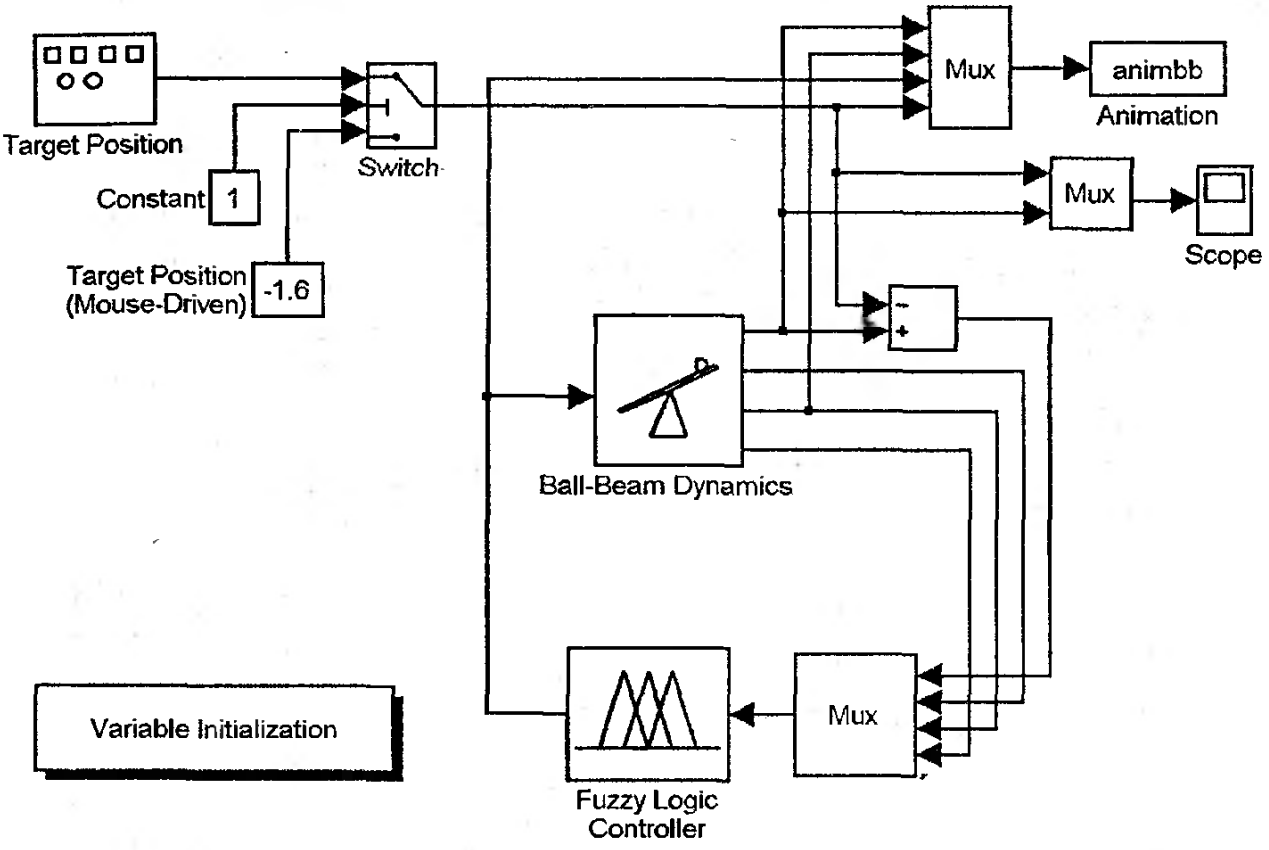


Рис. 3.85. Симулинк-модель системы «шарик на коромысле»

Анимация запускается командой **Start** меню **Simulation**. После этого откроется диалоговое окно (рис. 3.86), в котором будет происходить анимация. Шарик изображен кружком, коромысло – прямоугольником, опора коромысла – серым треугольником. Цель, т.е. точка, в окрестности которой необходимо удерживать шарик, указана белым треугольником. Управляющая сила показана стрелкой, длина которой пропорциональна величине силы. Анимация запускается автоматически после открытия окна. Для вывода траекторий движения шарика и коромысла необходимо установить флажок в окне **Show Trails**. Очистка окна от траекторий происходит нажатием кнопки **Clear Trails**. После начала анимации в нижней части окна появятся две кнопки: **Stop** – прекращение анимации и **Pause...** – пауза в анимации. Через меню **Target Position** назнача-

ются такие законы перемещение цели: **Sinosoid Wave** – синусоидальная волна; **Square Wave** – прямоугольная волна (меандр); **Saw Wave** – пилообразная волна; **Mouse-Driven** – определяемый пользователь с помощью мыши. В последнем случае для перемещения цели необходимо установить мышку в середине белого треугольника, нажать на левую кнопку и не отпуская ее переместить цель в нужное место.

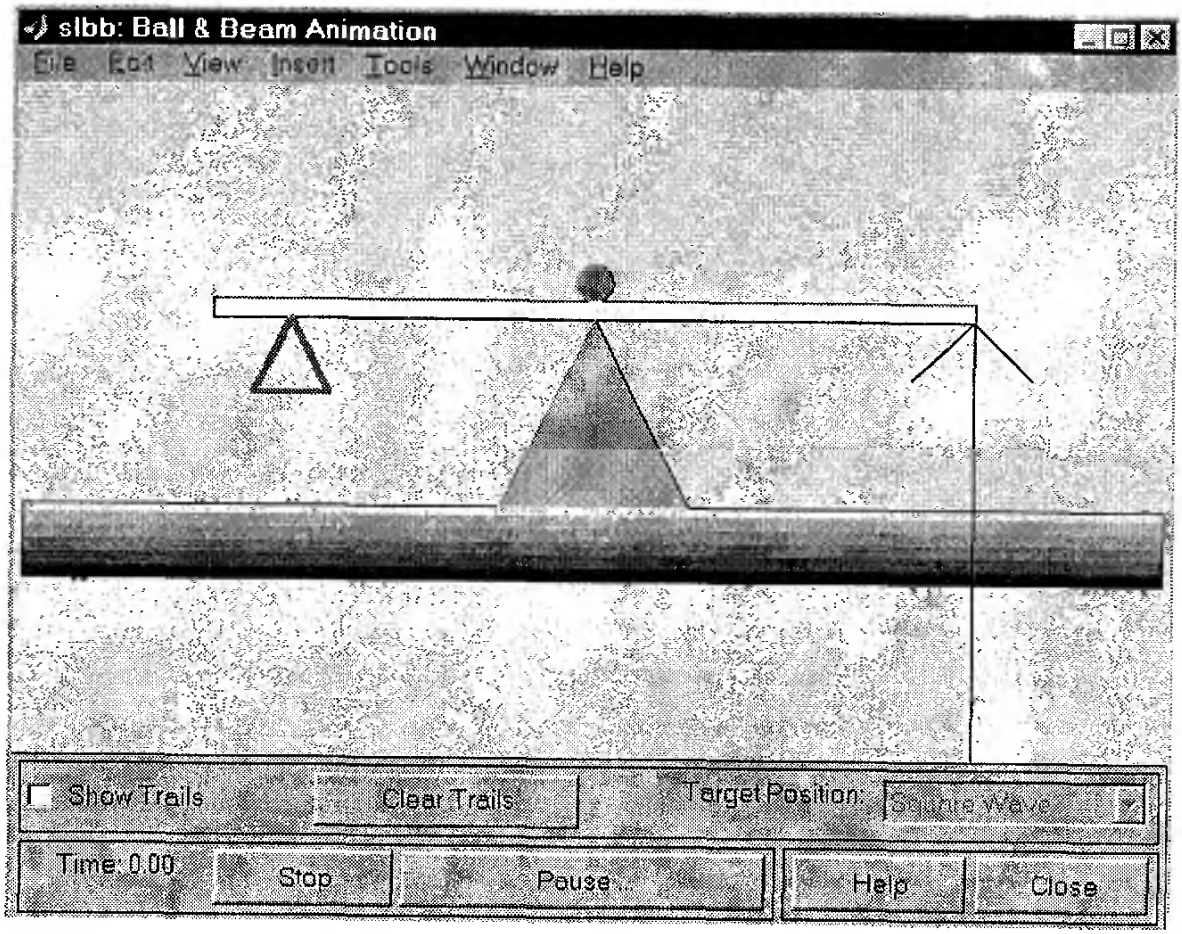


Рис. 3.86. Анимационное окно программы slbb

Управление системой «шарик на коромысле» осуществляет нечеткий контроллер Сугено с четырьмя входами. Для лингвистической оценки входных переменных используется по два терма с колоколообразными функциями принадлежности. База знаний содержит 16 правил. Для просмотра нечеткого контроллера в симулинк-формате необходимо щелкнуть правой кнопкой мыши по блоку **Fuzzy Logic Controller** (см. рис. 3.85) и в появившемся меню выбрать команду **Look under mask**. Затем в появившемся графическом окне **Link: slbb/Fuzzy Logic Controller** щелкнуть правой кнопкой мыши по блоку **FIS Wizard** и снова в появившемся меню выбрать команду **Look under mask**.

Демонстрационная программа **slbb** также выводит окно **Scope** с графиками изменения во времени положения шарика на коромысле и координаты цели. Траектория шарика изображается серой линией, а траектория цели – белой линией. Примеры траекторий при разных законах перемещения цели показаны на рис. 3.87.

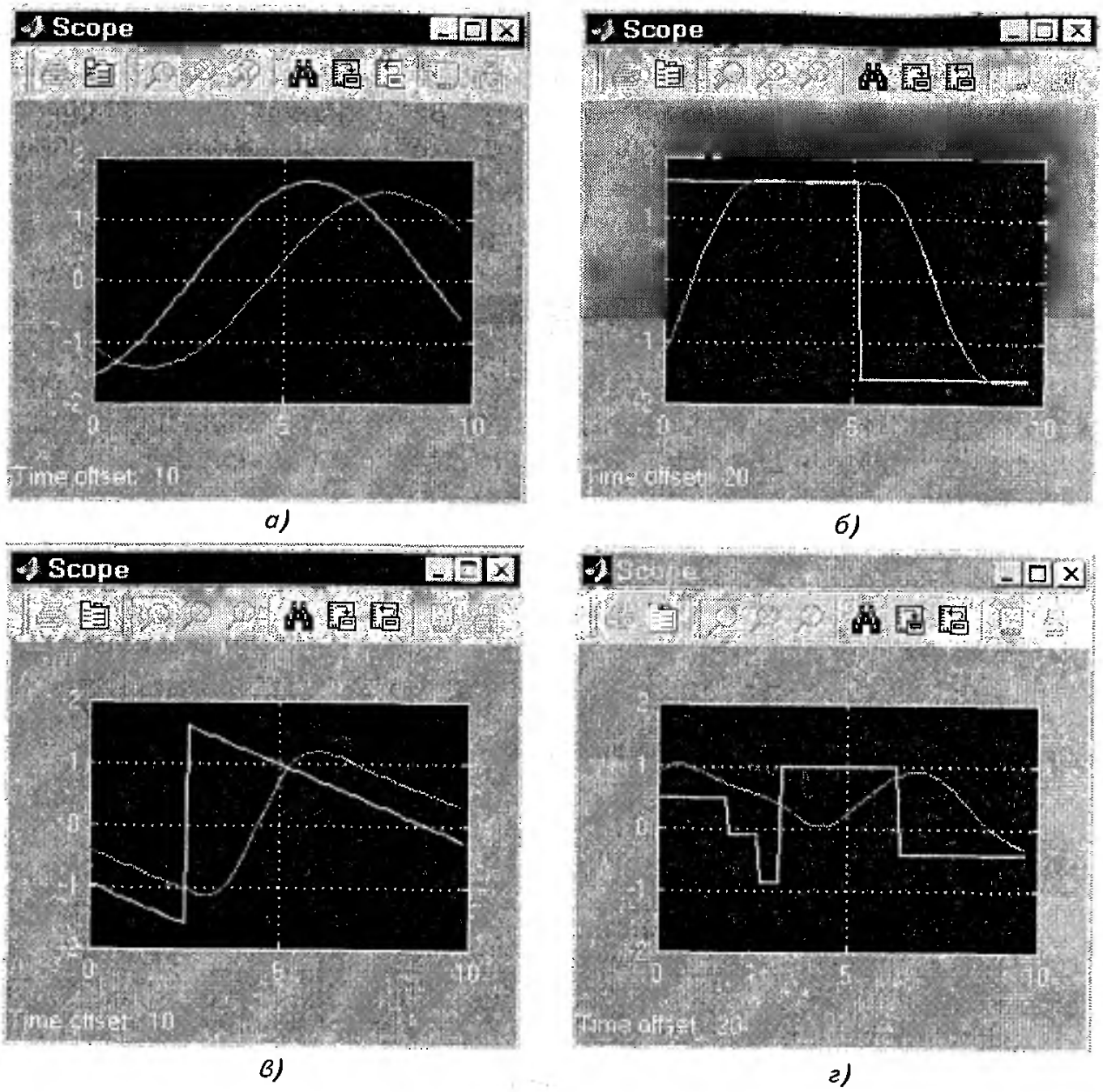


Рис. 3.87. Графики движений шарика на коромысле при разных законах перемещений цели

а – синусоидальная волна; *б* – меандр; *в* – пилообразная волна; *г* – заданный пользователем

3.4.9. ПАРКОВКА ГРУЗОВИКА

Демонстрационная программа sltbu иллюстрирует применение нечеткого контроллера для управления парковкой грузовика. Задача управления заключается в выборе таких положений рулевого колеса, которые обеспечивали бы перемещение грузовика в заданное место парковки. При парковке скорость грузовика поддерживается постоянной. Для работы программ sltbu необходим пакет Simulink.

После запуска программы sltbu появится окно пакета Simulink с симулинк-моделью для этого демо-примера (рис. 3.88).

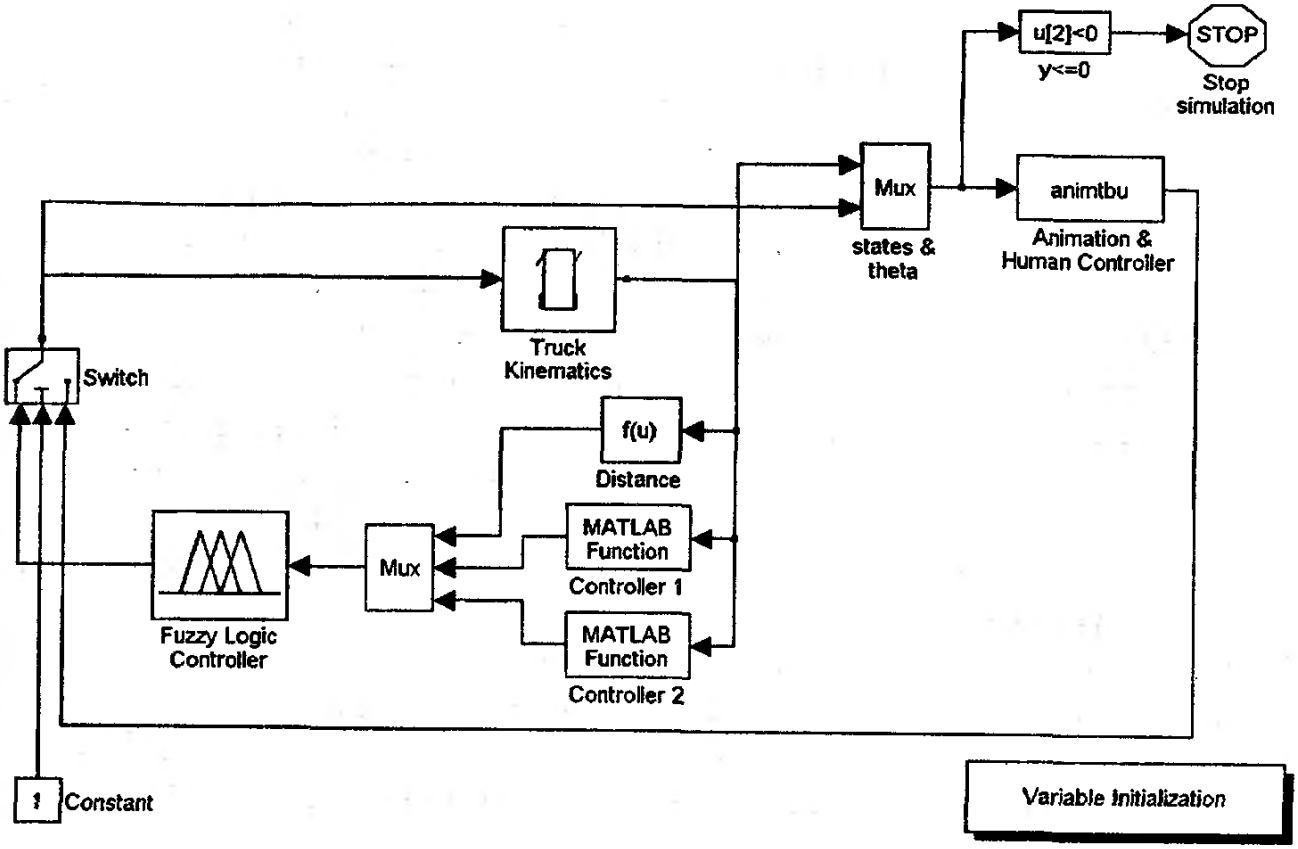


Рис. 3.88. Симулинк-модель системы «Грузовик с нечетким контроллером парковки»

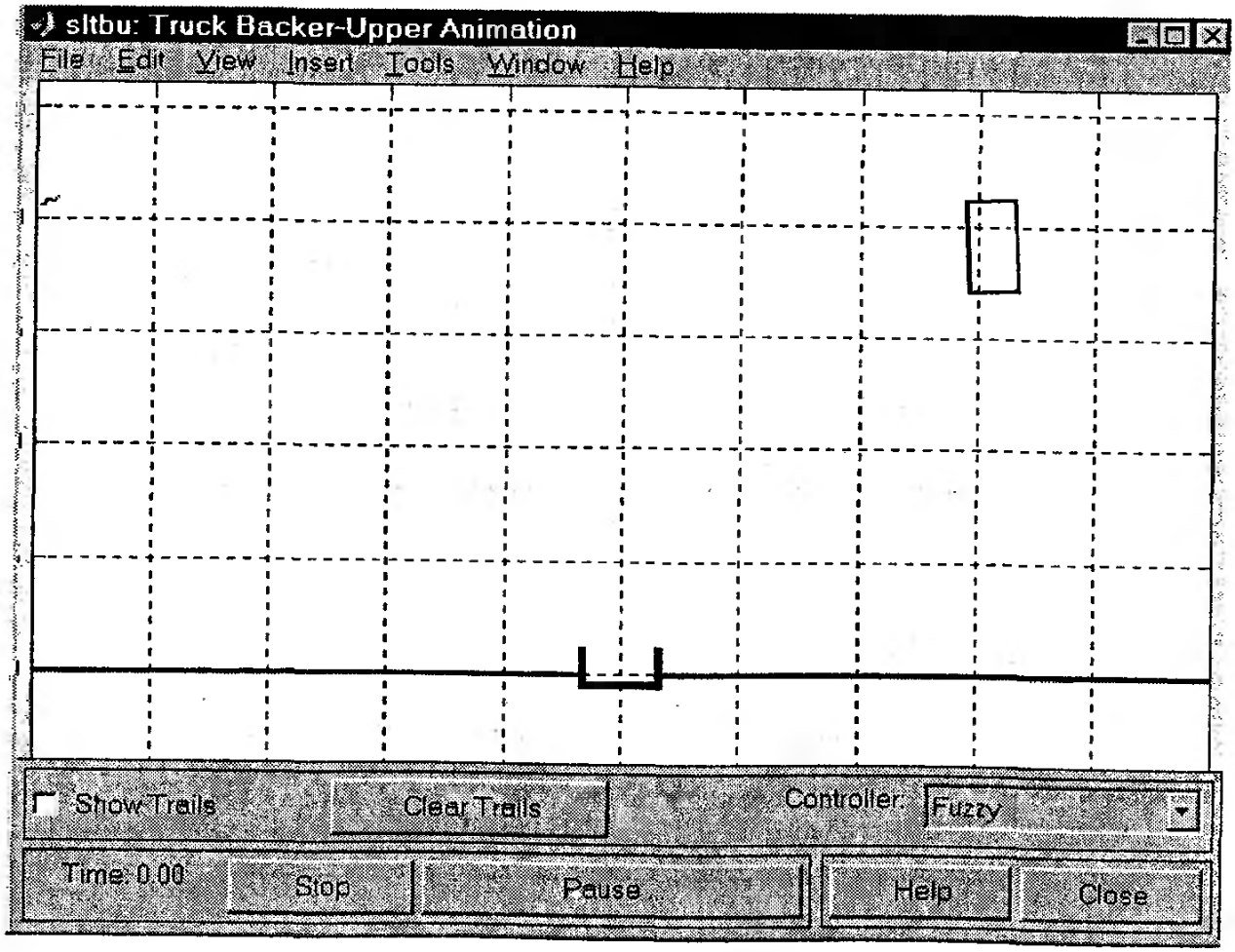


Рис. 3.89. Анимационное окно программы sltbu

Для запуска анимации необходимо выбрать команду **Start** меню **Simulation**. После этого появится окно анимации (рис. 3.89). Грузовик изображен прямоугольником, место парковки – черной перевернутой буквой «П», бордюр – горизонтальной линией. Центр грузовика не должен пересекать бордюр. Запуск анимации осуществляется автоматически по открытии окна. Для повторного запуска анимации необходимо нажать кнопку **Start Simulation...**. Для вывода траекторий грузовика необходимо установить флажок в окне **Show Trails**. Удаление следов грузовика происходит нажатием кнопки **Clear Trails**. После начала анимации в нижней части графического окна появятся две кнопки: **Stop** – прекращение анимации и **Pause...** – пауза в анимации. По нажатию кнопки **Pause...** появляются две новые кнопки: **Continue** – продолжение анимации и **Step** – выполнение анимации в пошаговом режиме. Способ управления парковкой грузовика выбирается через меню **Controller**, которое содержит следующие альтернативы: **Fuzzy** – нечеткий контроллер и **Human** – человек–оператор.

Нечеткий контроллер реализован системой нечеткого вывода Сугено. Система имеет выход control – угол поворота руля и три входа: distance – расстояния до точки парковки, control1 – угол поворота руля вблизи места парковки и control2 – угол поворота руля вдали от места парковки. Нечеткая база знаний содержит такие два правила:

ЕСЛИ distance = near, ТО control = control1;
ЕСЛИ distance = far, ТО control = control2.

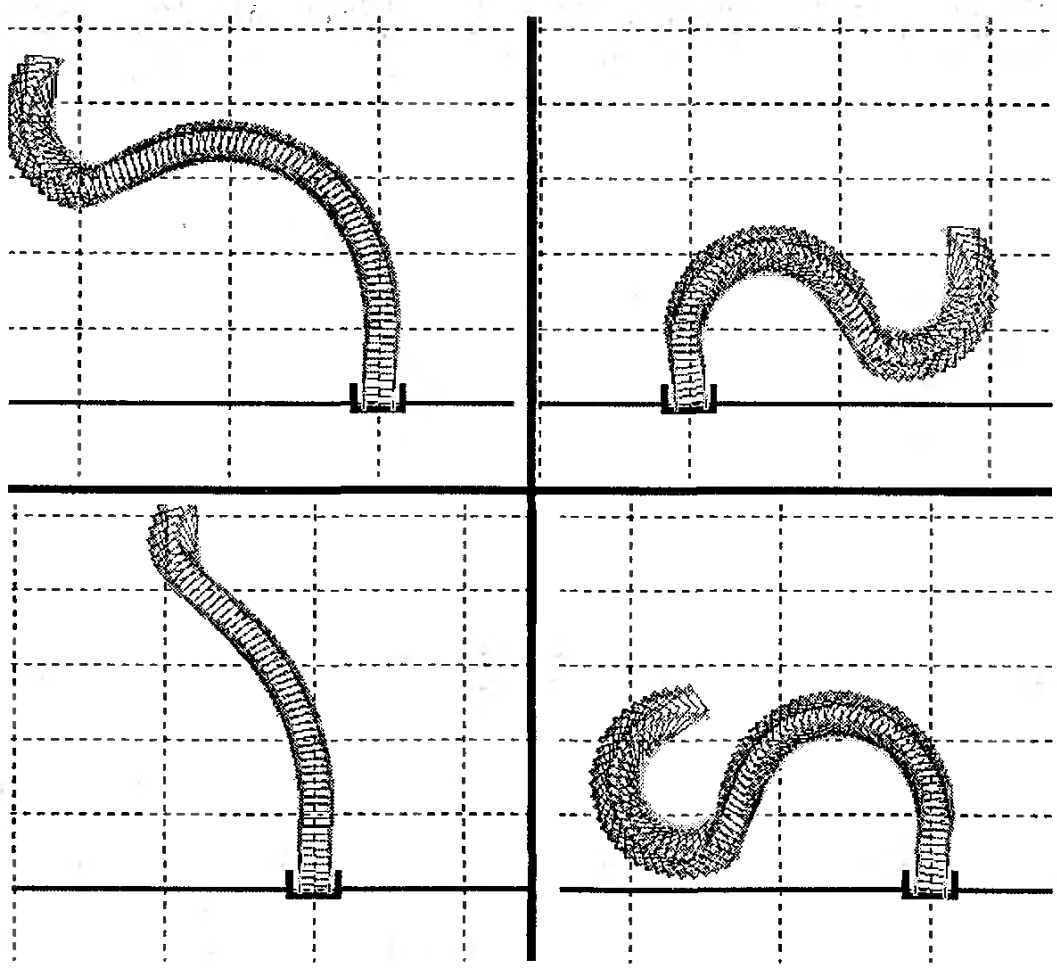


Рис. 3.90. Типовые траектории парковки грузовика с нечетким контроллером

Значение переменной control1 (control2), т.е. угол поворота руля, если бы грузовик находился близко (далеко) от точки парковки рассчитывается по несложному алгоритму. Нечеткий контроллер определяет, какое управление выбрать: ближнее – control1 или дальнее – control2. Для формализации нечетких термов near (рядом) и far (далеко) используются *z*- и *s*-подобные функции принадлежности соответственно. Когда грузовик находится недалеко и неблизко от места парковки, угол поворота руля рассчитывается как промежуточное значение между control1 и control2 по алгоритму нечеткого вывода Сугено. Примеры траекторий парковки из разных стартовых позиций приведены на рис. 3.90. В стартовые позиции грузовик перетаскивается мышкой.

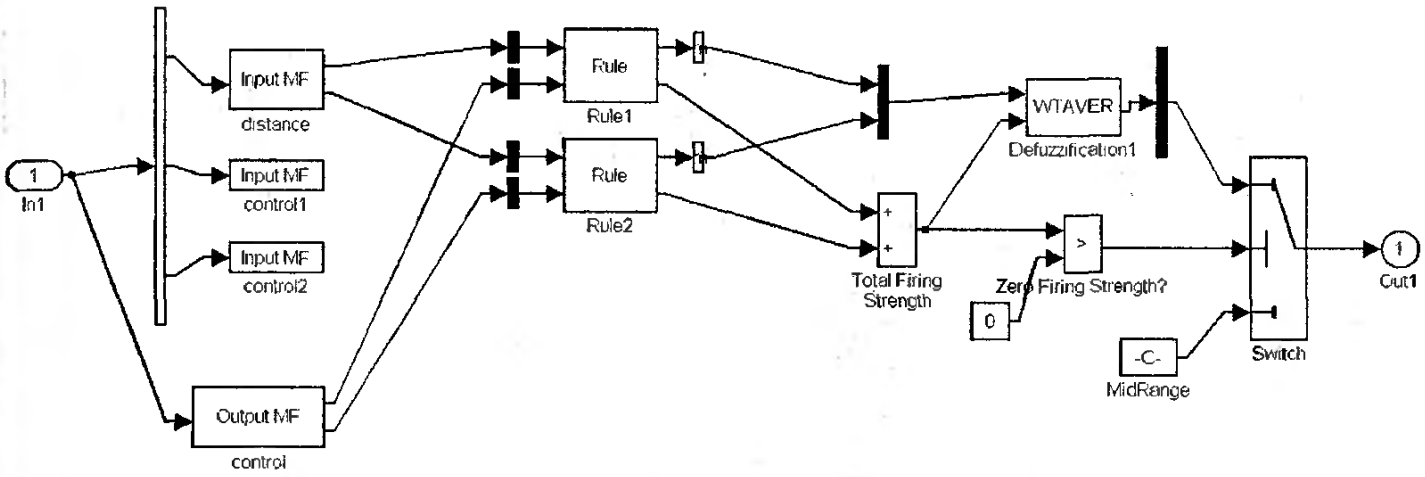


Рис. 3.91. Нечеткий контроллер парковки грузовика в симулинк-формате

Для просмотра нечеткого контроллера в симулинк-формате (рис. 3.91) необходимо щелкнуть правой кнопкой мыши по блоку **Fuzzy Logic Controller** (см. рис. 3.88) и в появившемся меню выбрать команду **Look under mask**. Затем в новом графическом окне **Link: sltbu/Fuzzy Logic Controller** щелкнуть правой кнопкой мыши по блоку **FIS Wizard** и снова в появившемся меню выбрать команду **Look under mask**.

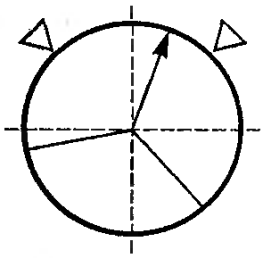


Рис. 3.92. Рулевое колесо при ручной парковке грузовика

При ручном управлении парковкой в окне анимации выводится «рулевое колесо», изображенное на рис. 3.92. Изменение угла поворота происходит по перемещению мышкой стрелки «рулевого колеса».

3.4.10. РЕГУЛЯТОР ВОДЫ В БАКЕ

Демонстрационные программы sltank и sltankrule иллюстрируют использование нечеткого контроллера для регулирования уровня воды в баке. Программа sltank отличается от sltankrule только тем, что в ней отсутствует окно браузера ViewRuler с правилами нечеткого управления.

Объект регулирования представляет собой бак, в который одновременно втекает и из которого вытекает вода. Задача регулирования состоит в выборе таких положений вентиля, которые обеспечили бы поддержание заданного уровня воды в баке. Объект является нелинейным, так как расход воды пропорционален квадрату высоты столба жидкости в баке. Для работы программ sltank и sltankrule необходим пакет Simulink.

После запуска программы sltankrule появится окно пакета Simulink, содержащее симулинк-модель (рис. 3.93) этого демо-примера.

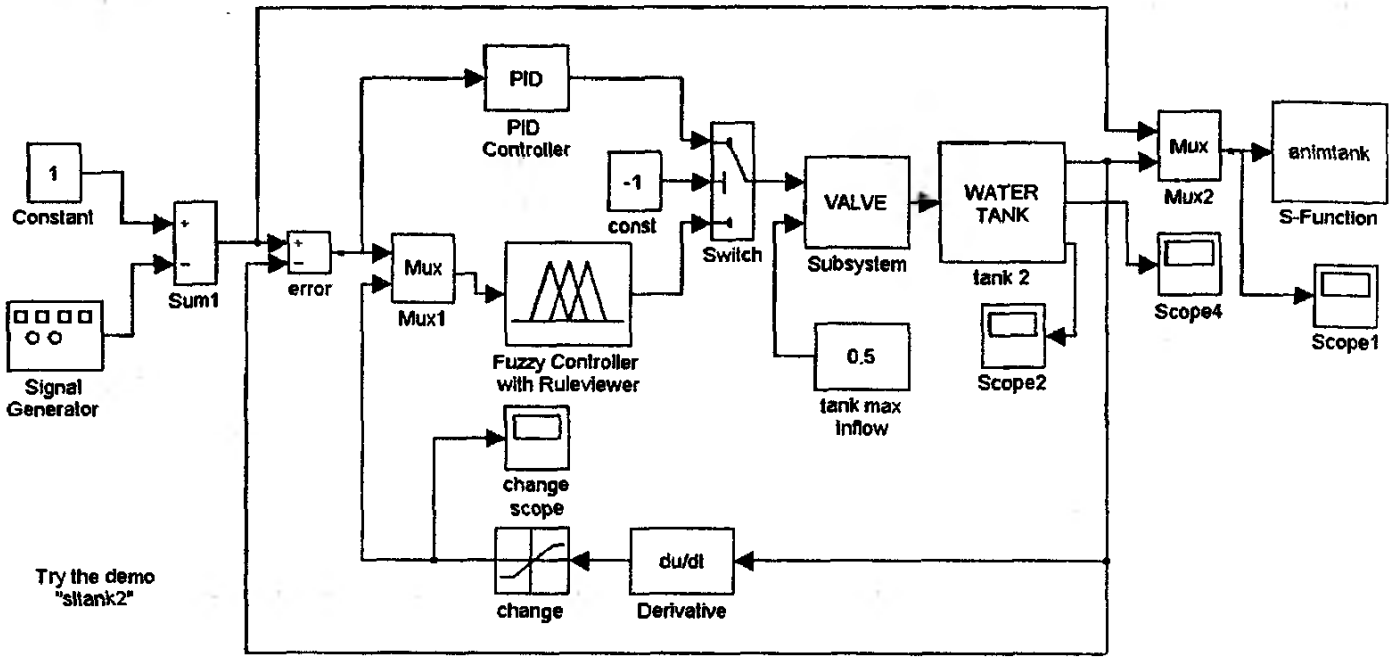


Рис. 3.93. Симулинк-модель системы «Водяной бак с нечетким контроллером»

Запуск анимации осуществляется командой **Start** меню **Simulation**. После этого появится анимационное окно (окно **Tank Demo** на рис. 3.94). Текущий уровень воды показан серым цветом. Необходимый уровень воды отмечен черной линией. Анимация запускается автоматически по открытию окна. Также автоматически открывается окно правил нечеткого контроллера (верхнее окно на рис. 3.94), иллюстрирующее принятие решений по управлению в текущий момент времени.

Нечеткий контроллер реализован системой нечеткого вывода с двумя входами: разница между требуемым и текущим уровнями воды и скорость изменения этой разницы. Временные диаграммы требуемого и текущего уровней воды показаны в окне **Scope1** (рис. 3.94) белой и серой линиями соответственно. В окне **change scope** приведена временная диаграмма скорости изменения разности требуемого и текущего уровней воды. Эти окна открываются по щелчку мышкой по пиктограммам **Scope1** и **change scope** в симулинк-модели системы «Водяной бак с нечетким контроллером». Пользователь может аналогичным образом вывести на экран временные диаграммы расхода воды (**Scope4**) и сигнала переполнения бака (**Scope2**). Примеры этих диаграмм показаны на рис. 3.95.

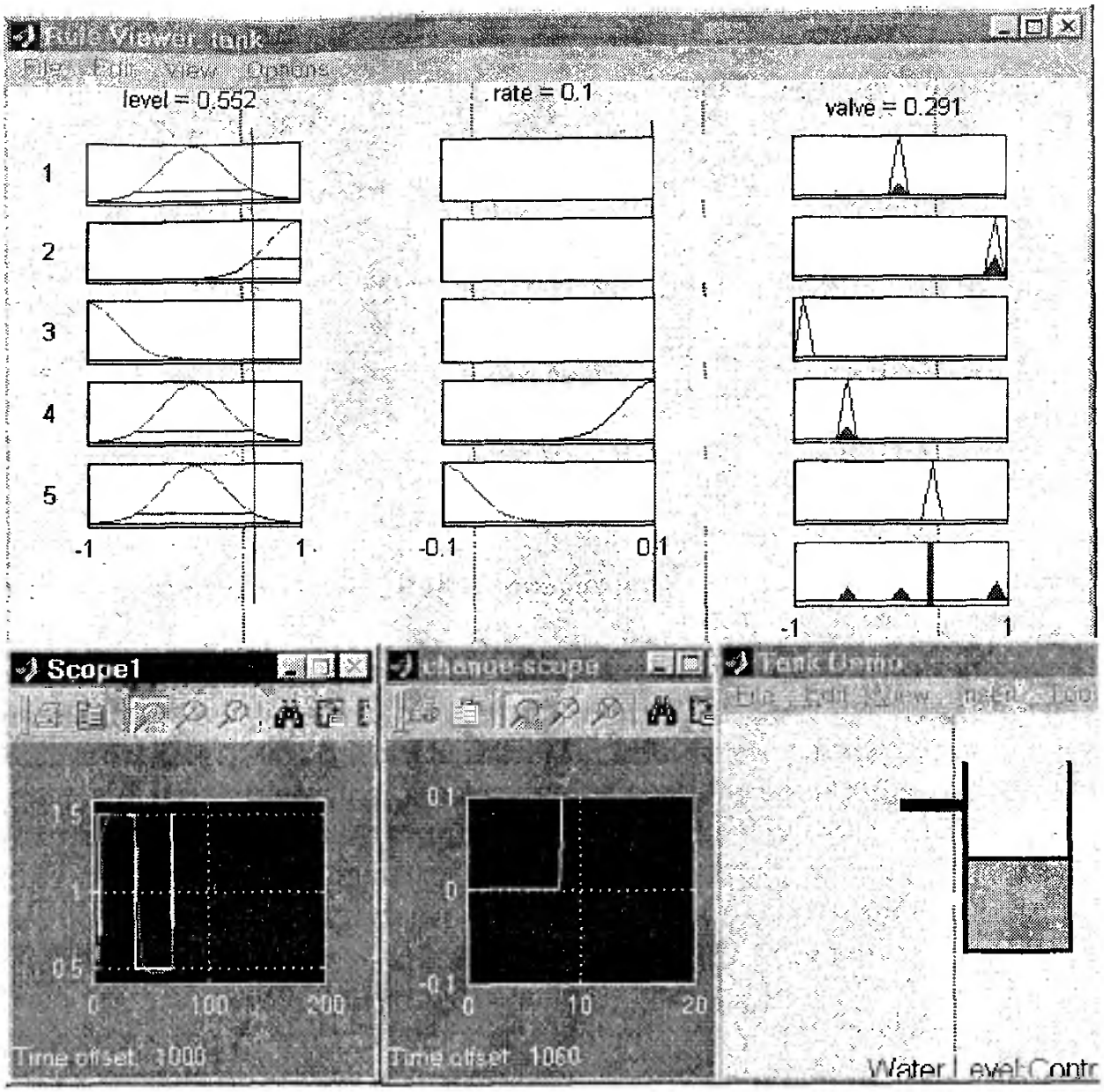
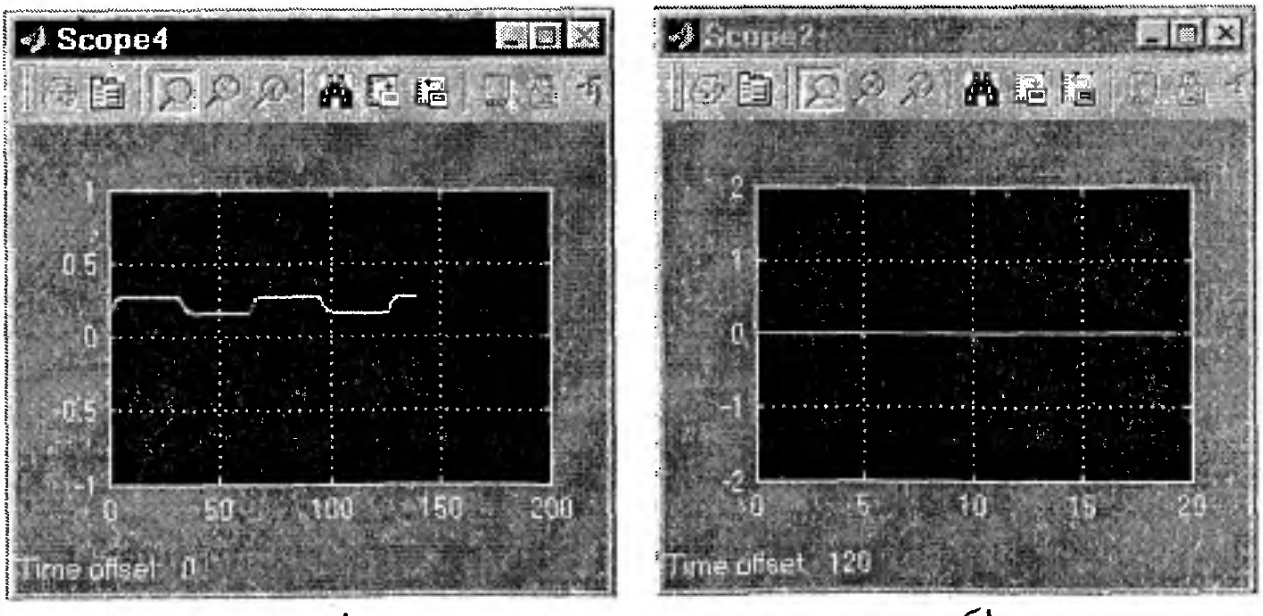


Рис. 3.94. Основные окна системы «Водяной бак с нечетким контроллером»



а) *б)*

Рис. 3.95. Временные диаграммы расхода воды (*а*) и сигнала переполнения бака (*б*)

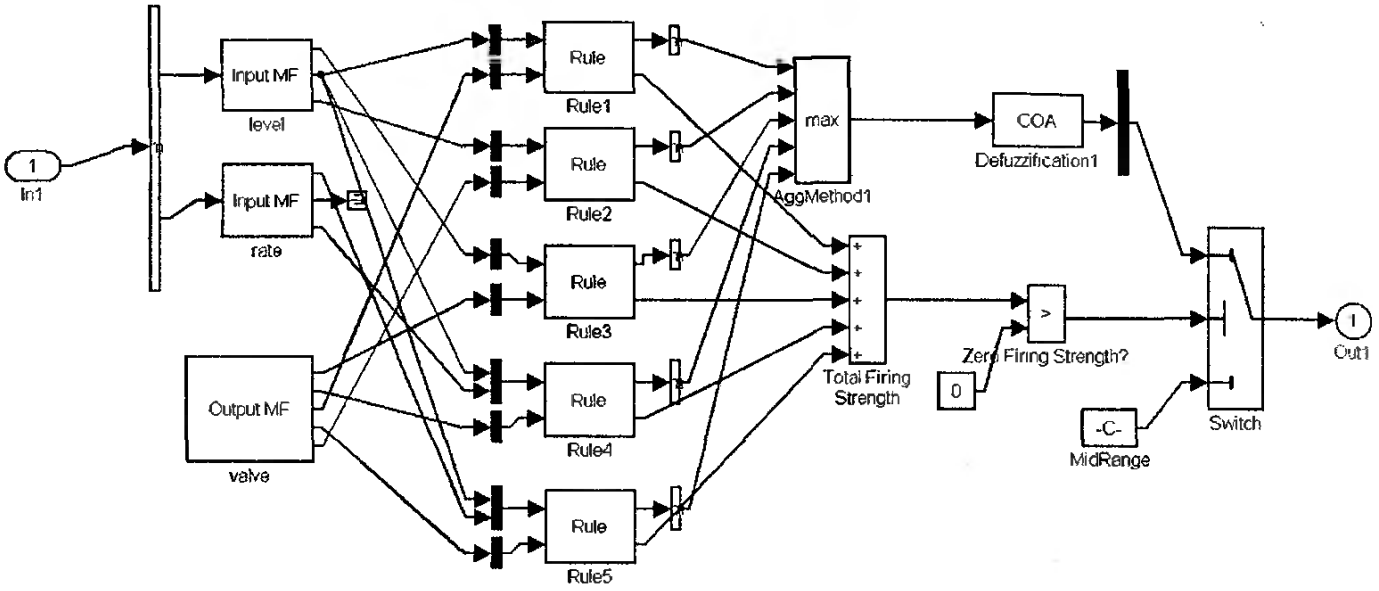


Рис. 3.96. Нечеткий контроллер уровня воды в баке в симулинк-формате

Нечеткий контроллер использует пять правил для расчета управляющего воздействия. Для лингвистической оценки входной переменной level (разница между требуемым и текущим уровнями воды) используются три терма с гауссовыми функциями принадлежности, для оценки входной переменной rate (скорость изменения разницы между требуемым и текущим уровнями воды) – два терма с гауссовыми функциями принадлежности и для выходной переменной valve (изменение положения вентиля) – пять термов с треугольными функциями принадлежности. Для просмотра нечеткого контроллера в симулинк-формате (рис. 3.96) необходимо щелкнуть правой кнопкой мыши по блоку **Fuzzy Controller** (см. рис. 3.93) и в появившемся меню выбрать команду **Look under mask**. Затем в новом графическом окне **Link: sltankrule/Fuzzy Logic Controller with Ruleviewer*** щелкнуть правой кнопкой мыши по блоку **Fuzzy Logic Controller** и снова в появившемся меню выбрать команду **Look under mask**. Затем в новом графическом окне **Link: sltankrule/Fuzzy Logic Controller with Ruleviewer/Fuzzy Logic Controller*** щелкнуть правой кнопкой мыши по блоку **FIS Wizard** и снова в появившемся меню выбрать команду **Look under mask**.

3.4.11. УПРАВЛЕНИЕ ДУШЕМ

Демонстрационная программа shower иллюстрирует применение нечеткого контроллера для управления душем. Задача управления состоит в установке таких положений вентилей холодной и горячей воды смесителя, которые обеспечили бы требуемые уровни температуры и расхода воды. Для работы программы shower необходим пакет Simulink.

После запуска программы shower появится окно с симулинк-моделью (рис. 3.97) для этого демо-примера.

Анимация запускается командой **Start** меню **Simulation**. После этого в графических окнах **temp scope** и **flow scope** выводятся временные диаграммы темпера-

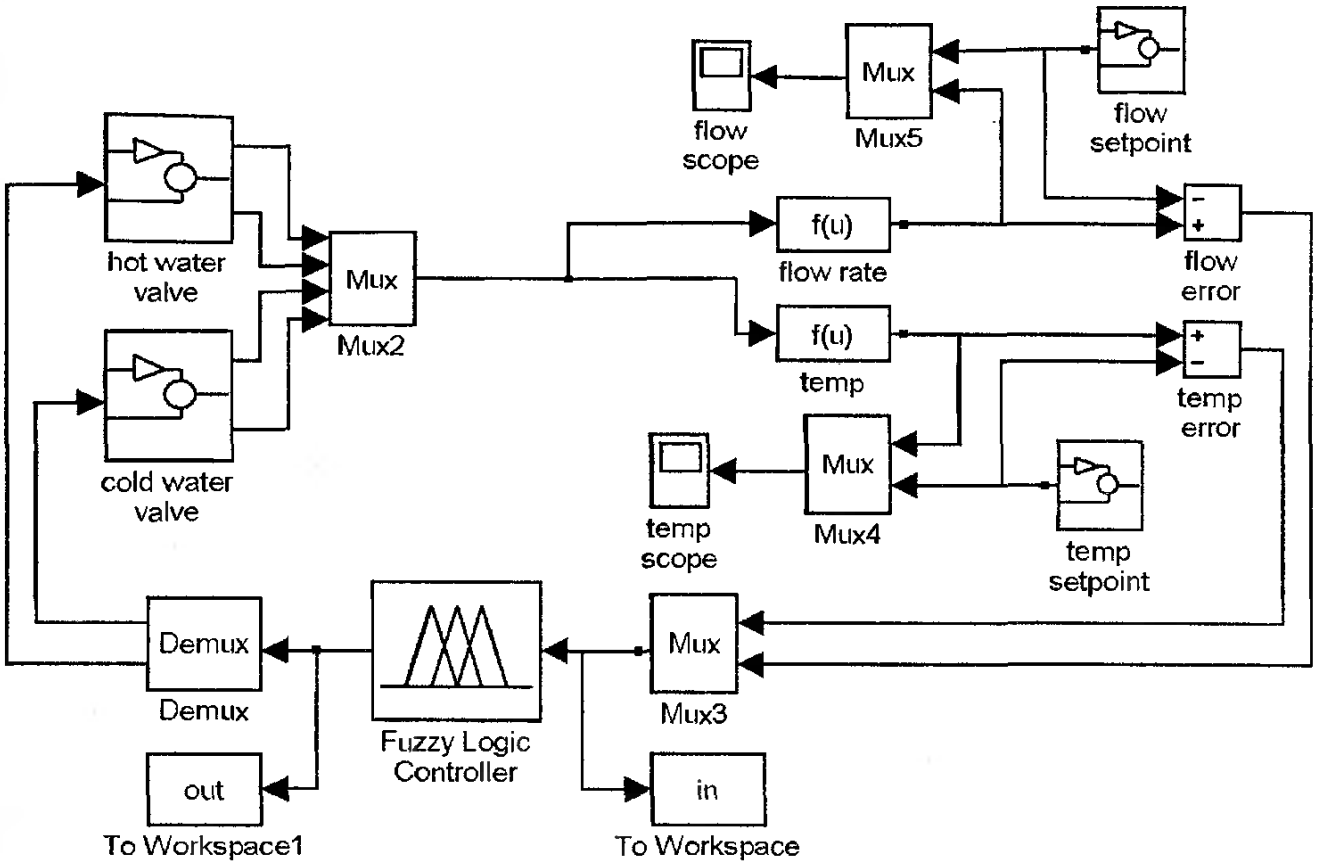


Рис. 3.97. Симулинк-модель системы «Душ с нечетким контроллером»

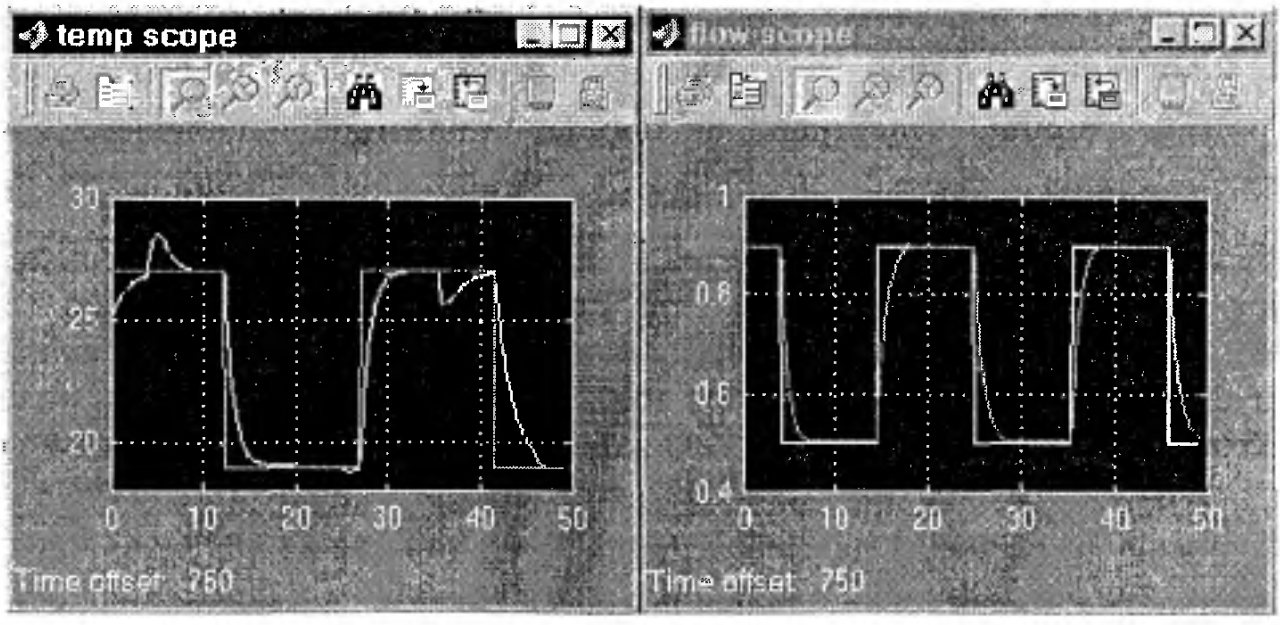


Рис. 3.98. Типовые временные диаграммы температуры и расхода воды

туры и расхода воды в душе (рис. 3.98). Требуемые значения температуры и расхода воды показаны серыми линиями, а текущие значения – белыми линиями. Моделирование системы «душ с нечетким контроллером» происходит при следующих параметрах:

- температура холодной воды – 10°C;
- температура горячей воды – 30°C;

- закон изменения требуемого расхода воды – прямоугольная волна с частотой 0,3 рад/с;
- закон изменения требуемой температуры воды – прямоугольная волна с частотой 0,21432 рад/с.

Нечеткий контроллер душа реализован системой нечеткого вывода Мамдани с двумя входами и двумя выходами. Входными переменными являются: **temp** – разница между текущей и требуемой температурой воды и **flow** – разница между текущим и требуемым расходом воды. Выходными переменными являются: **cold** – изменение положения вентиля холодной воды и **hot** – изменение положения вентиля горячей воды. Для лингвистической оценки входных переменных используется по три терма. Крайние термы заданы трапециевидными функциями принадлежности, а остальные – треугольными. Для лингвистической оценки выходных переменных используется по пять термов с треугольными функциями принадлежности. База знаний содержит такие правила:

If (temp is cold) and (flow is soft) then (cold is openSlow) and (hot is openFast);

If (temp is cold) and (flow is good) then (cold is closeSlow) and (hot is openSlow);

If (temp is cold) and (flow is hard) then (cold is closeFast) and (hot is closeSlow);

If (temp is good) and (flow is soft) then (cold is openSlow) and (hot is openSlow);

If (temp is good) and (flow is good) then (cold is steady) and (hot is steady);

If (temp is good) and (flow is hard) then (cold is closeSlow) and (hot is closeSlow);

If (temp is hot) and (flow is soft) then (cold is openFast) and (hot is openSlow);

If (temp is hot) and (flow is good) then (cold is openSlow) and (hot is closeSlow);

If (temp is hot) and (flow is hard) then (cold is closeSlow) and (hot is closeFast).

Для просмотра нечеткого контроллера в симулинк-формате необходимо щелкнуть правой кнопкой мыши по блоку **Fuzzy Logic Controller** (см. рис. 3.97) и в появившемся меню выбрать команду **Look under mask**. Затем в новом окне **Link: shower/Fuzzy Logic Controller*** щелкнуть правой кнопкой мыши по блоку **FIS Wizard** и снова в появившемся меню выбрать команду **Look under mask**.

3.4.12. УДЕРЖАНИЕ ПЕРЕВЕРНУТОГО МАЯТНИКА НА ТЕЛЕЖКЕ

Демонстрационные программы slcp, slcp1 и slcpp1 иллюстрируют применение нечеткого контроллера для перемещения неустойчивой системы «Перевернутый маятник на тележке» в заданную точку. Объект управления представляет собой стержень, нижний конец которого шарнирно закреплен на тележке. Тележка может перемещаться вдоль плоскости вращения стержня. Задача состоит в приложении к тележке таких сил, которые бы обеспечили удержание маятника в вертикальном положении. Удержание в равновесии системы «Перевернутый маятник на тележке» является классической задачей теории автоматического управления.

К ней сводятся задачи управления многими реальными объектами, например, ракетой или буксиром, толкающим впереди себя баржи. Программы slcp, slcp1 и slcpp1 демонстрируют применение нечеткого управления не только для удержания маятника в вертикальном положении, но и для перемещения неустойчивой системы в заданную точку. Объект управления в программе slcp содержит один маятник фиксированной длины, в программе slcp1 – один маятник переменной длины, в программе slcpp1 – два маятника. Для работы этих программ необходим пакет Simulink.

После запуска программы slcp появится окно пакета Simulink с симулинк-моделью (рис. 3.99) для этого демо-примера.

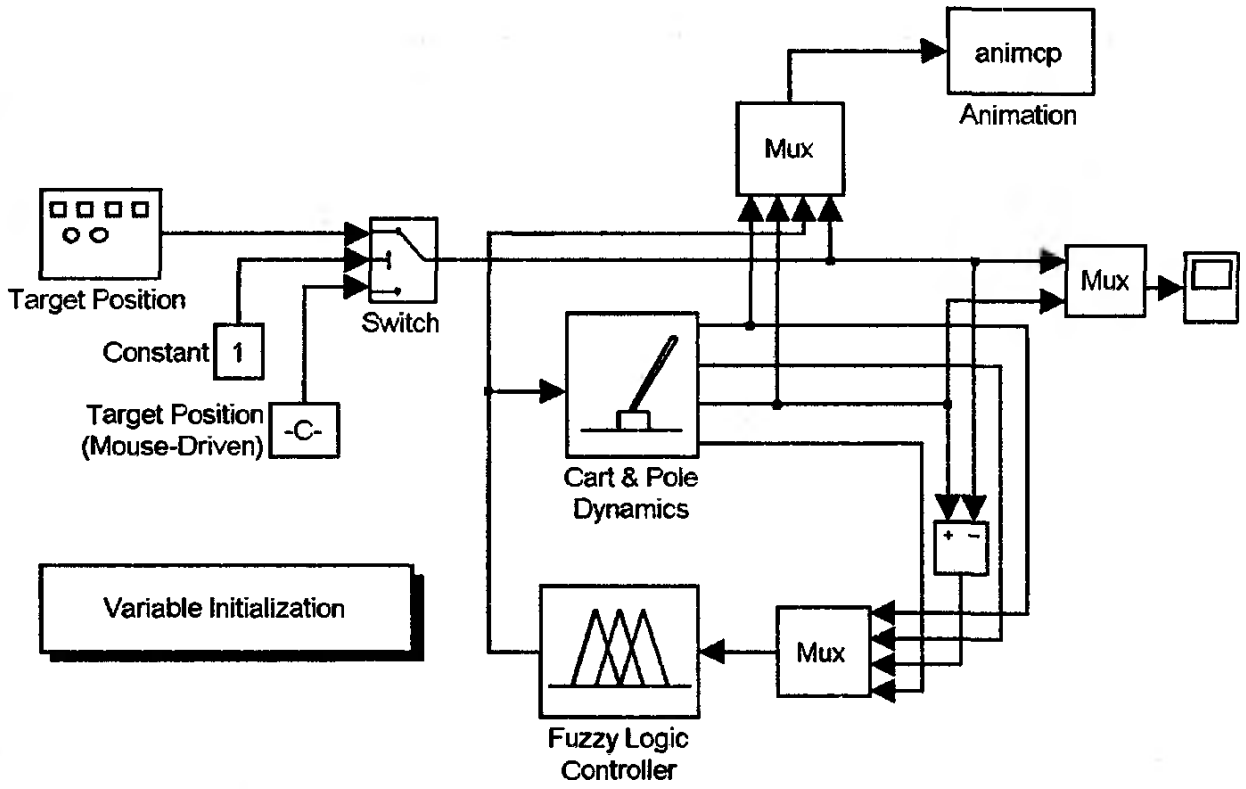


Рис. 3.99. Симулинк-модель системы «Перевернутый маятник на тележке»

Анимация запускается командой **Start** меню **Simulation**. После этого появится анимационное окно (рис. 3.100). Маятник изображен белым прямоугольником, телсжка – серым прямоугольником. Цель, т.е. точка, в окрестности которой необходимо удерживать маятник, указана треугольником. Управляющая сила показана стрелкой, длина которой пропорциональна величине силы. Для вывода траекторий движения маятника и тележки необходимо установить флажок в окне **Show Trails**. Очистка графического окна от траекторий происходит нажатием кнопки **Clear Trails**. После начала анимации в нижней части графического окна появляются две кнопки: **Stop** – прекращение анимации и **Pause...** – пауза в анимации. Через меню **Target Position** назначаются следующие законы перемещения цели: **Sinusoid Wave** – синусоидальная волна; **Square Wave** – прямоугольная волна (меандр); **Saw Wave** – пилообразная волна; **Random Wave** – случайный; **Mouse Driven** – определяемый пользователем с помощью мышки. В последнем случае

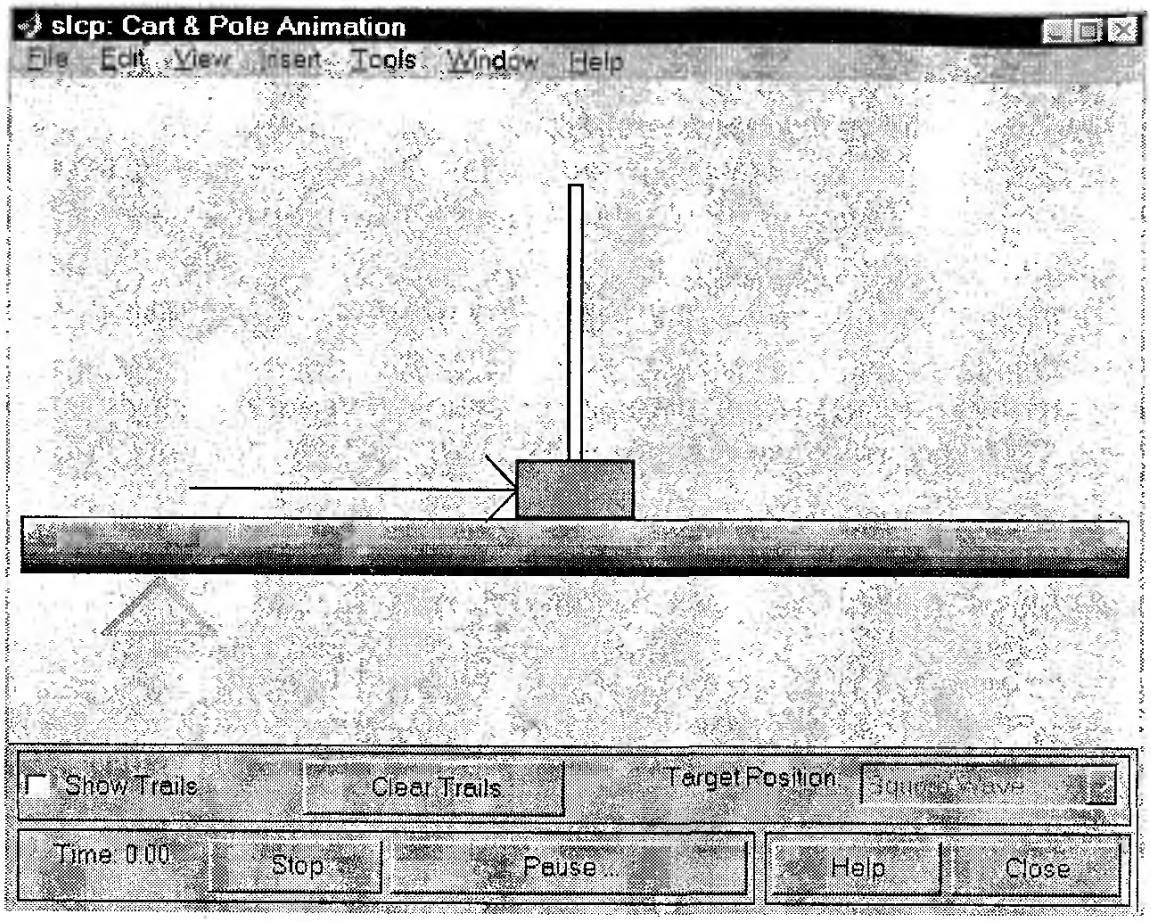


Рис. 3.100. Анимационное окно программы slcp

необходимо установить мышку в середине треугольника, нажать на левую кнопку и, не отпуская ее, переместить цель в нужное место.

Системой «Перевернутый маятник на тележке» управляет нечеткий контроллер Сугено с четырьмя входами: угол наклона маятника; скорость изменения угла наклона маятника; расстояние от центра тележки до цели; скорость тележки. Для лингвистической оценки входных переменных используется по два терма с колоколообразными функциями принадлежности. База знаний содержит 16 правил. Для просмотра нечеткого контроллера в симулинк-формате необходимо щелкнуть правой кнопкой мыши по блоку **Fuzzy Logic Controller** (см. рис. 3.99) и в появившемся меню выбрать команду **Look under mask**. Затем в новом окне **Link: slcp/Fuzzy Logic Controller** щелкнуть правой кнопкой мыши по блоку **FIS Wizard** и выбрать команду **Look under mask**.

Демонстрационная программа slcp выводит траектории цели и центра тележки в графическом окне **Scope**. Примеры этих траекторий при разных законах перемещения цели показаны на рис. 3.101. Траектория тележки показана серой линией, траектория цели – белой линией. Эти окна выводятся по щелчку мышкой по пиктограмме **Scope** (крайняя справа фигура на рис. 3.99).

Симулинк-модель системы «Перевернутый маятник переменной длины на тележке», используемая демонстрационной программой slcp1, показана на рис. 3.102. Эта модель отличается от предыдущей (см. рис. 3.99) наличием блоков, задающих закон изменения длины маятника в виде синусоидальной волны с частотой 6 рад/с.

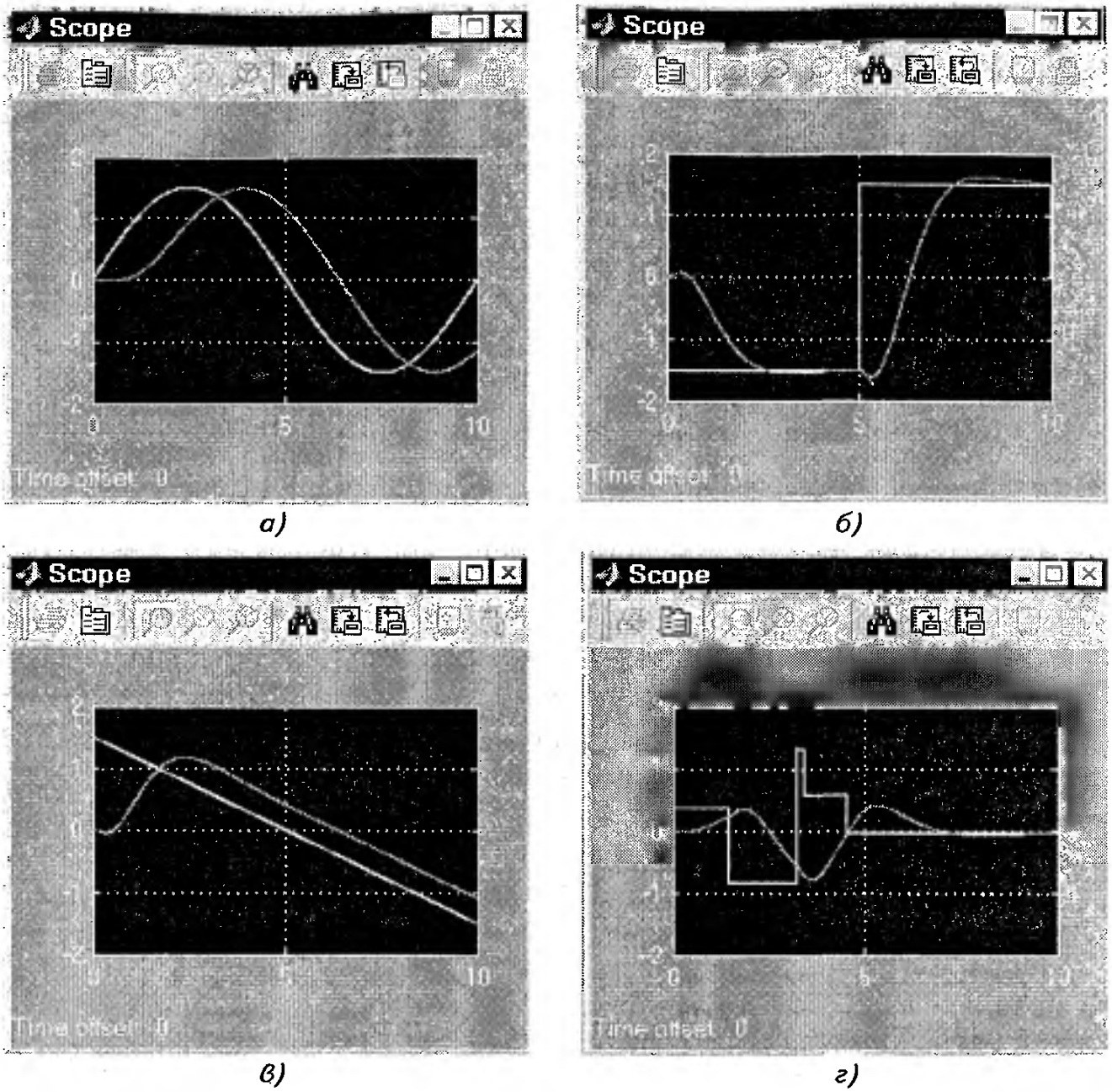


Рис. 3.101. Графики движений тележки при разных законах перемещения цели

а – синусоидальная волна; *б* – меандр; *в* – пилообразная волна; *г* – заданный пользователем

Системой «Перевернутый маятник переменной длины на тележке» управляет нечеткий контроллер Сугено (рис. 3.103) с пятью входами: угол наклона маятника, скорость изменения угла наклона маятника, расстояние от центра тележки до цели, скорость тележки и длина маятника. Управление происходит по трем правилам, которые определяют закон перемещения тележки при коротком, среднем и длинном маятнике. Термы «короткий», «средний» и «длинный» представлены нечеткими множествами с гауссовыми функциями принадлежности.

Примеры траекторий тележки при перемещении цели по синусоидальной и прямоугольной волнам показаны на рис. 3.104. Как и в случае маятника с постоянной длиной, нечеткий контроллер обеспечивает хорошее управление неустойчивым объектом.

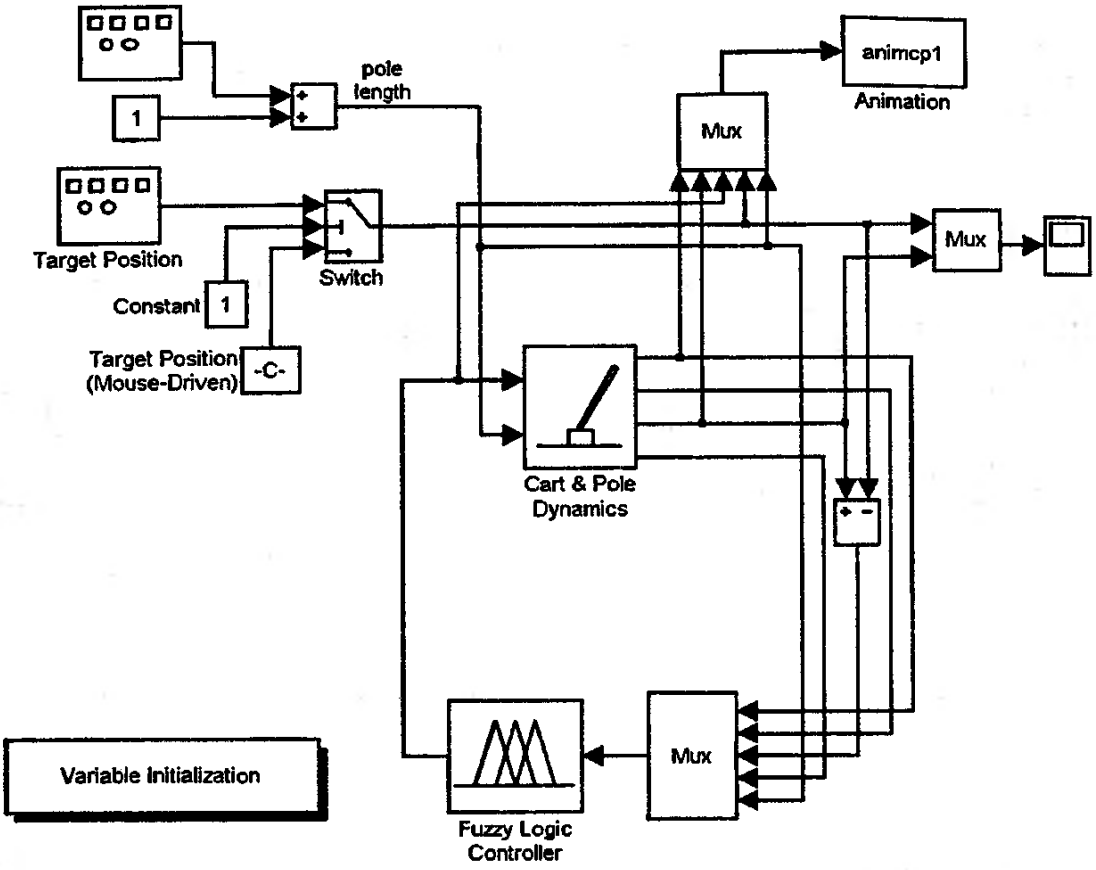


Рис. 3.102. Симулинк-модель системы «Перевернутый маятник переменной длины на тележке»

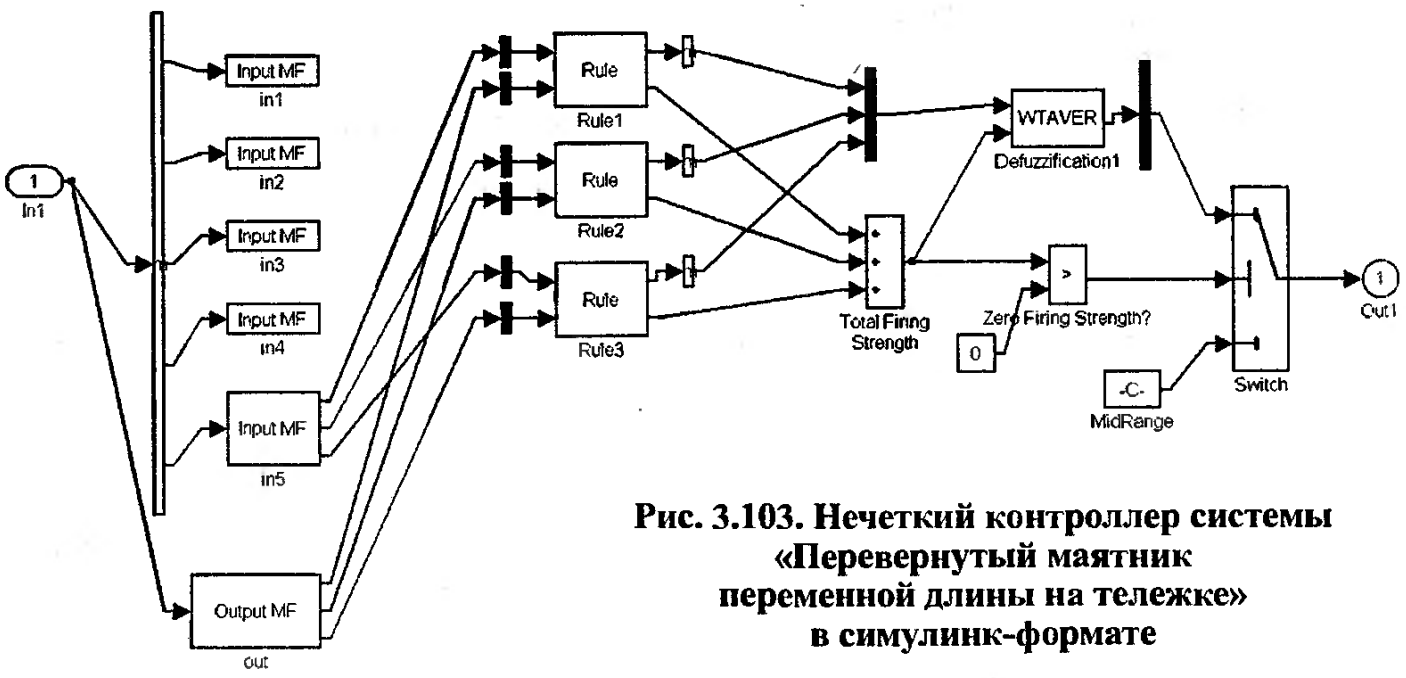
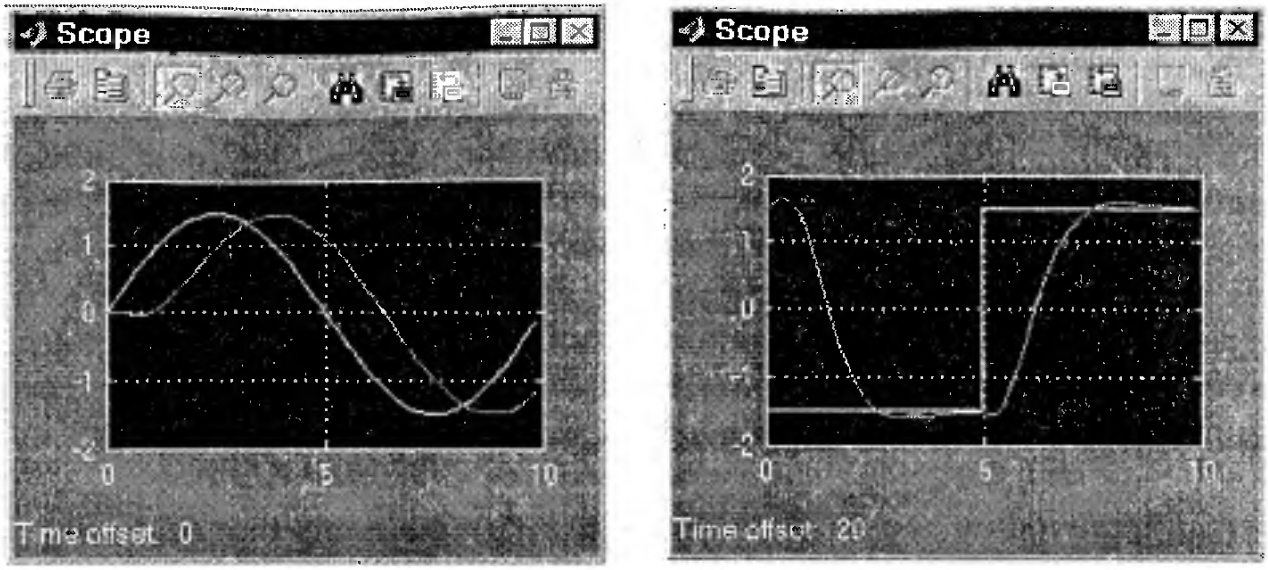


Рис. 3.103. Нечеткий контроллер системы «Перевернутый маятник переменной длины на тележке» в симулинк-формате

Симулинк-модель системы «Два перевернутых маятника на тележке» демо-программы slcpp1 показана на рис. 3.105. Маятники на тележке расположены параллельно. Длина одного из них изменяется во времени. Таким образом, рассматриваемый объект управления отличается от предыдущего наличием дополнительного маятника фиксированной длины.



а) *б)*

Рис. 3.104. Графики движений тележки при разных законах перемещения цели

а – синусоидальная волна; *б* – меандр

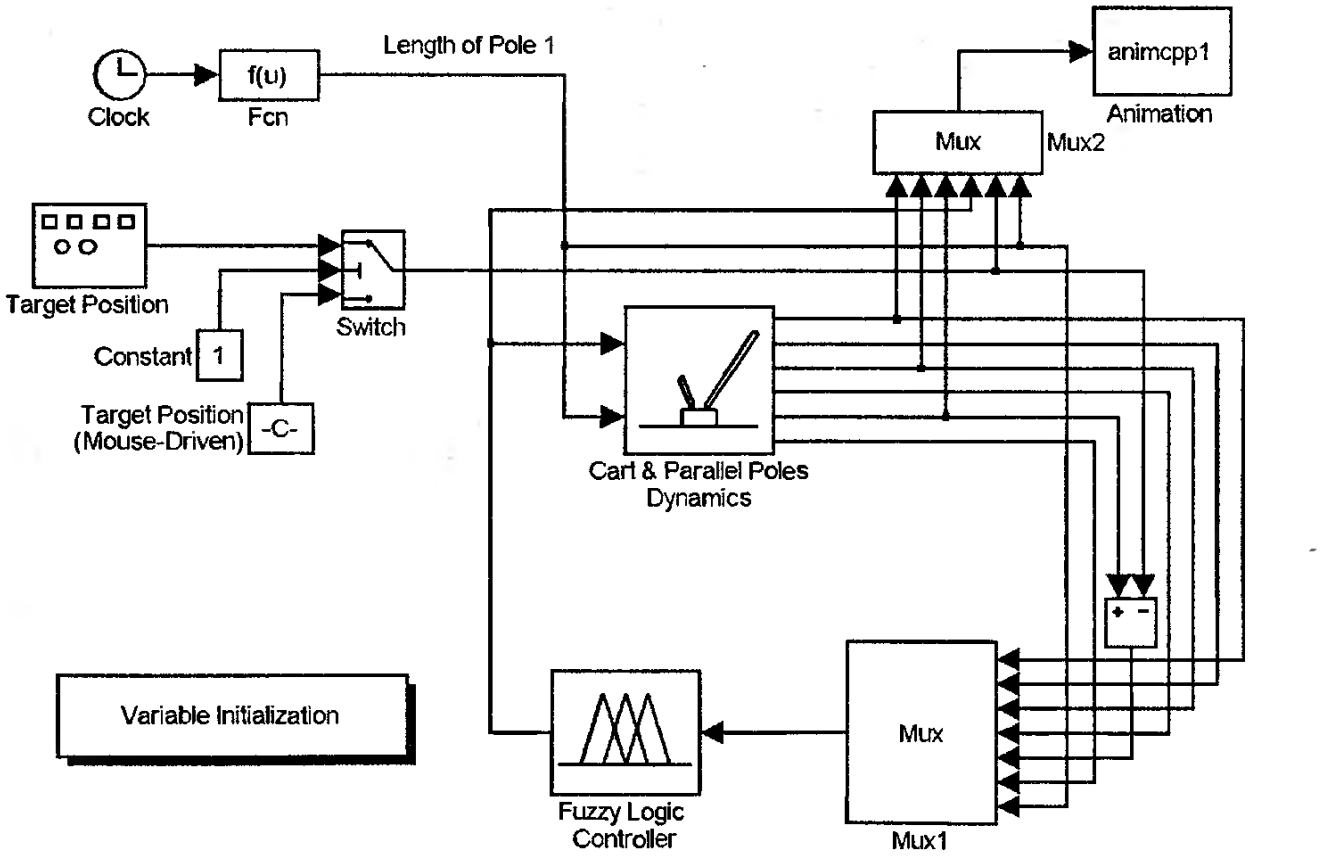


Рис. 3.105. Симулинк-модель системы «Два перевернутых маятника на тележке»

Системой «Два перевернутых маятника на тележке» управляет нечеткий контроллер Сугено с шестью входами. База знаний содержит 11 правил, которые определяют законы перемещения тележки при различных длинах маятника. Управления системой «Два перевернутых маятника на тележке» достаточно сложная задача для такого нечеткого контроллера – часто тележка с маятниками перемещается за пределы экрана.

3.4.13. УПРАВЛЕНИЕ РУКОЙ РОБОТА–МАНИПУЛЯТОРА

Демонстрационная программа invkine иллюстрирует применение нечетких контроллеров для управления рукой робота–манипулятора. Робот–манипулятор представляет собой двухзвенный шарнирно-соединенный механизм. Один конец первого звена шарнирно соединен со столом, а второй конец соединен со вторым звеном. Рука робота перемещается в одной плоскости. Траектория рабочего элемента робота зависит от углов поворота в шарнирных соединениях. Задача управления состоит в выборе таких углов поворота в шарнирных соединениях, которые обеспечивали бы попадание рабочего элемента робота в заданную точку.

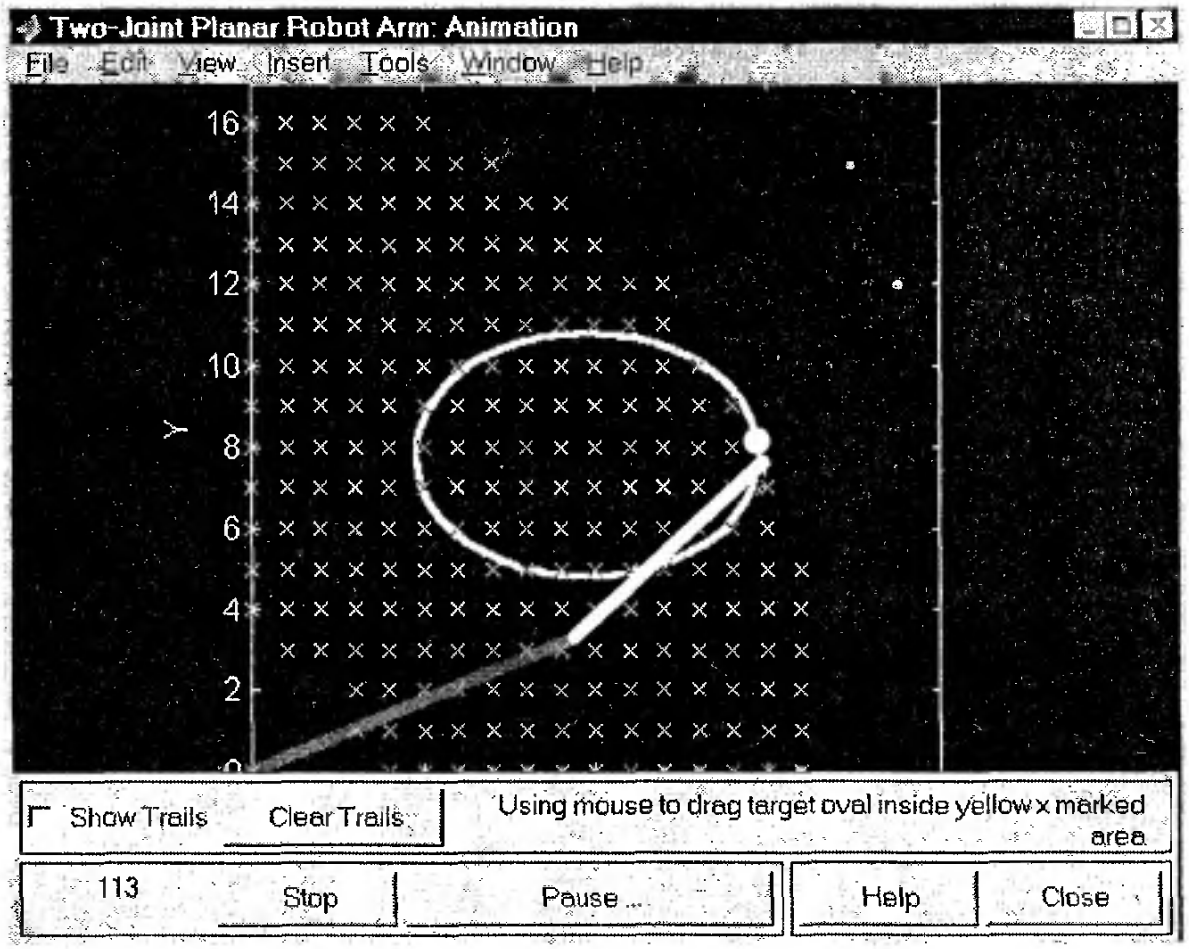


Рис. 3.106. Демонстрационная программа invkine

После запуска программы invkine появится диалоговое окно, изображенное на рис. 3.106. Первое звено руки робота показано серым прямоугольником, второе – белым. Требуемая траектория движения рабочего элемента робота изображена овалом. Точка, в которой должен находиться рабочий элемент робота, показана белым кружком. Область движения руки робота задается перемещением мышкой овала, который соответствует требуемой траектории. Рука робота будет двигаться по этому овалу только внутри области, отмеченной крестиками. Каждый крестик соответствует одной точке из обучающей выборки, которая использовалась при настройке нечетких контроллеров.

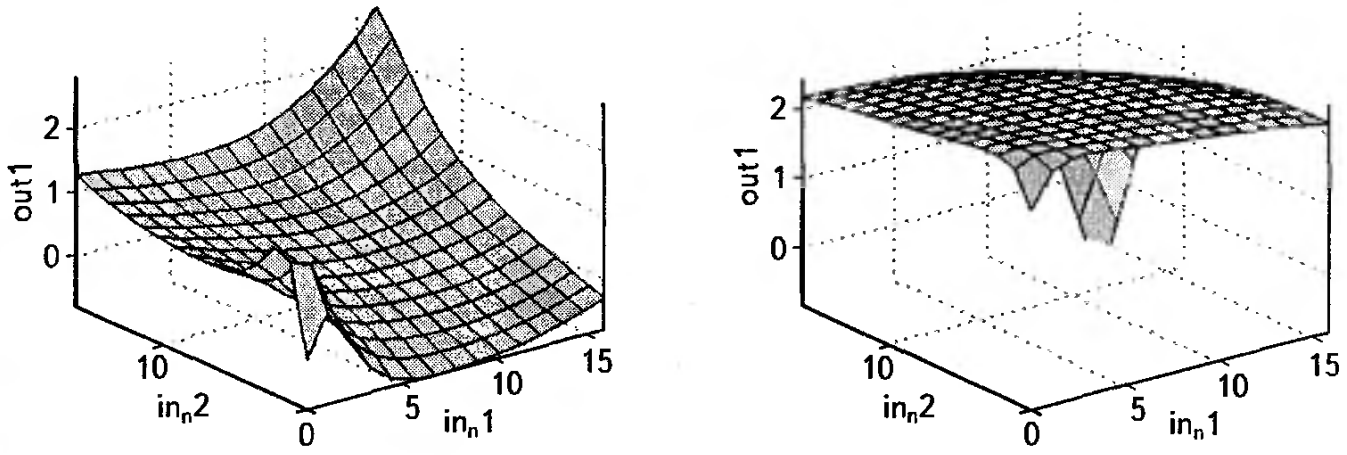


Рис. 3.107. Поверхности «входы – выход» для нечетких контроллеров робота–манипулятора

Нажатие кнопки **Start Animation...** запускает анимацию. Для вывода траекторий движения руки робота необходимо установить флажок в окне **Show Trails**. Очистка графического окна от траекторий происходит по нажатию кнопки **Clear Trails**.

Роботом управляют два нечетких контроллера Сугено. Первый контроллер рассчитывает угол поворота первого звена руки робота (серого прямоугольника), а второй контроллер – второго звена (белого прямоугольника). Входными переменными для нечетких контроллеров являются координаты точки, в которой должен находиться рабочий элемент робота. Для лингвистической оценки входных переменных используется по три терма с колоколообразными функциями принадлежности. Нечеткое управление в каждом контроллере происходит по девяти правилам. Поверхности «входы – выход» для нечетких контроллеров робота–манипулятора показаны на рис. 3.107. Видно, что при попарно малых и попарно больших значениях координат цели нечеткие контроллеры выдают неадекватные управляющие воздействия. Это объясняется нерепрезентативными обучающими выборками, в которых отсутствуют данные с такими координатами. Для настройки каждого контроллера использовались обучающие выборки из 229 пар «входы – выход», которые показаны на рис. 3.106 крестиками. При установке овала за пределы зоны обучения нечеткие контроллеры не обеспечивают движения руки робота по требуемой траектории. В пределах зоны обучения траектория руки робота, управляемого нечеткими контроллерами, практически не отличается от требуемой.

3.4.14. КЛАСТЕРИЗАЦИЯ АЛГОРИТМОМ НЕЧЕТКИХ С-СРЕДНИХ

Демонстрационная программа fcmdemo иллюстрирует применение алгоритма нечетких *c*-средних для задач кластеризации. Программа fcmdemo выводит на экран интерактивное окно (рис. 3.108), позволяющее пользователю выбрать набор данных для кластеризации, установить параметры алгоритма нечетких

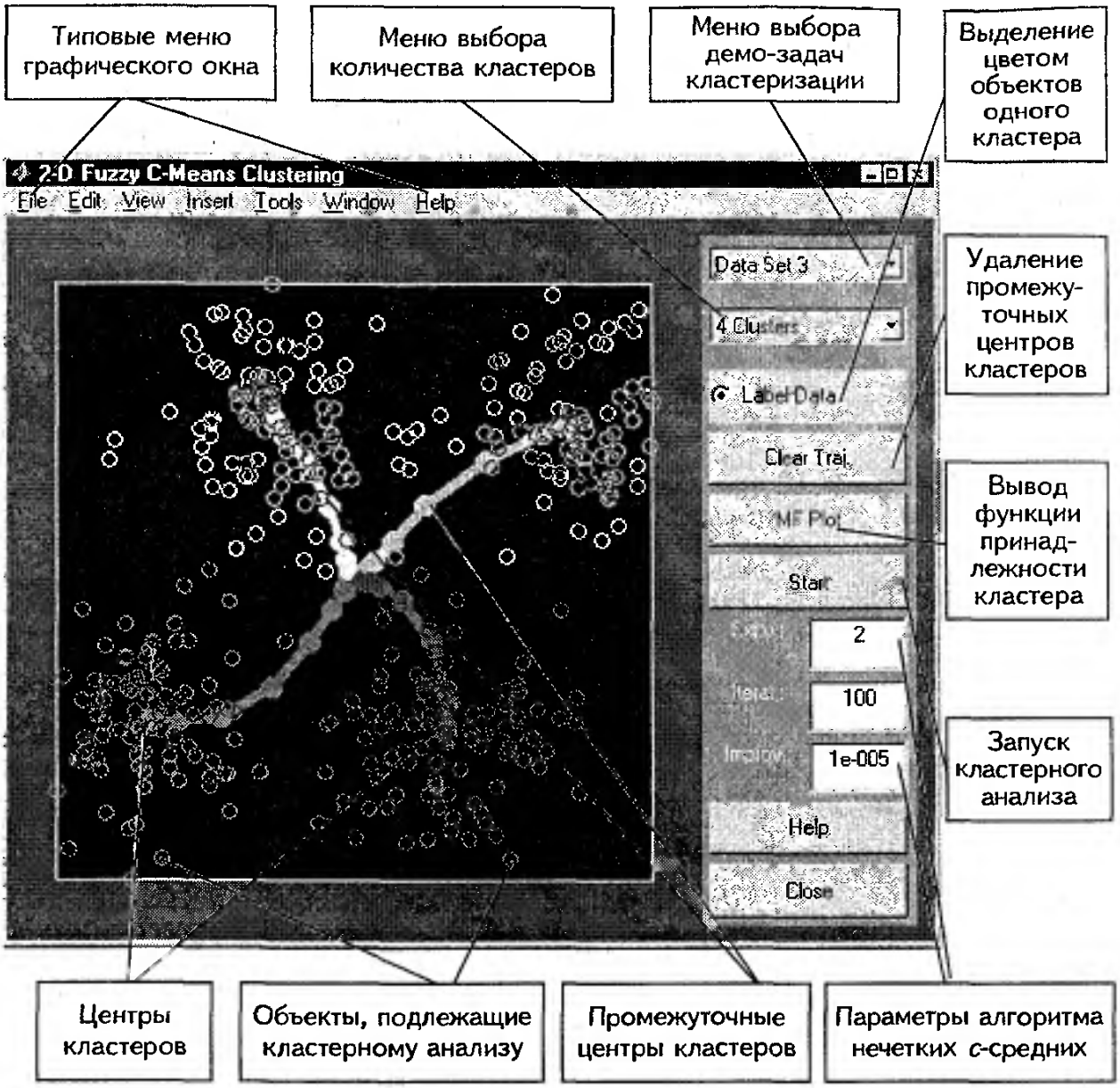


Рис. 3.108. Окно демонстрационной программы fcmdemo

c-средних, вывести результаты кластеризации, в том числе и графики функций принадлежности нечетких кластеров. Значения целевой функции на каждой итерации алгоритма нечеткой кластеризации выводятся в рабочую область MATLAB.

Графическое окно **fcmdemo** содержит семь верхних типовых меню (**File**, **Edit**, **View**, **Insert**, **Tools**, **Windows** и **Help**), область визуализации, меню выбора данных, меню установки параметров алгоритма нечетких *c*-средних, кнопку запуска кластеризации (**Start**), а также кнопки **Info** и **Close**, которые вызывают окно справки и завершают программу соответственно.

Меню выбора данных расположено в правом верхнем углу графического окна. Пользователь может выбрать один из пяти демо-наборов данных **Data Set 1** – **Data Set 5** или загрузить свои данные. Демо-наборы генерируются программой fcmdemo при каждой загрузке данных. Для загрузки своих данных необходимо выбрать опцию **Custom...** и затем в типовом окне открытия файла указать соответствующий файл данных. Данные должны быть записаны в файле построчно, т.е.

каждый объект необходимо описать одной строкой, содержащей значения двух признаков.

В центре графического окна расположена область визуализации, в которой выводятся объекты кластеризации и найденные центры кластеров. Маркеры в виде окружностей соответствуют объектам кластеризации, а маркеры в виде дисков – центрам кластеров. Кластеры выделяются различными цветами. Вначале центры всех кластеров располагаются в середине области визуализации. Затем, во время выполнения алгоритма, центры нечетких кластеров пошагово перемещаются из середины к «своим» местам. Траектории центров кластеров изображаются сплошными линиями. После каждой итерации алгоритма нечетких *c*-средних цвета объектов изменяются согласно их принадлежности кластерам.

Для управления визуализацией используются следующие кнопки: **Label Data** – разрешение/подавление выделения цветом принадлежности объектов кластерам; **Clear Traj.** – удаление траекторий движения центров нечетких кластеров; **MF Plot** – вывод функции принадлежности центра нечеткого кластера. Для вывода графика функции принадлежности необходимо мышкой выбрать центр кластера и нажать кнопку **MF Plot**. Пример функции принадлежности центра кластера показан на рис. 3.109.

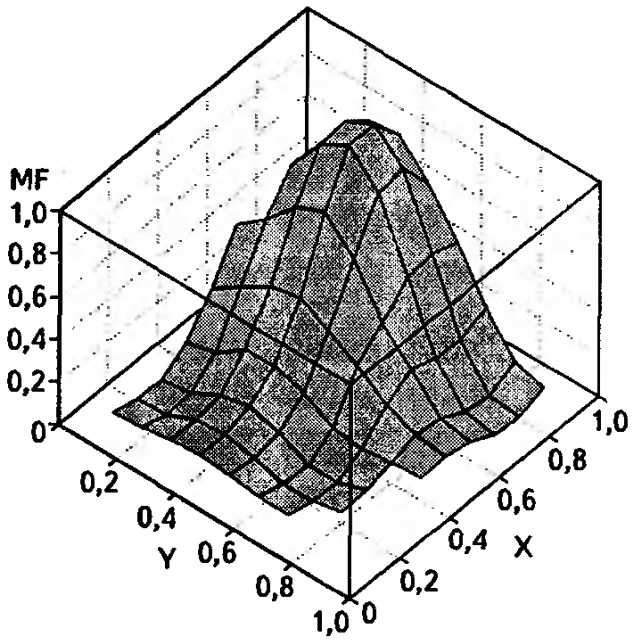


Рис. 3.109. Фуикция принадлежности нечеткого кластера в программе fcmdemo

Пользователь может выбрать через меню количество кластеров (от 2 до 9), а также установить значения следующих параметров алгоритма нечетких *c*-средних: экспоненциальный вес (поле **Expo**); максимальное количество итераций алгоритма (поле **Iterat.**); минимально допустимое улучшение целевой функции за одну итерацию алгоритма (поле **Improv.**).

3.4.15. КЛАСТЕРИЗАЦИЯ ИРИСОВ

Демонстрационная программа irisfcm иллюстрирует применение алгоритма нечетких *c*-средних для кластеризации ирисов. Этот алгоритм реализован функцией fcm. Задача классификации ирисов предложена Р. Фишером в 1936 г. С тех пор она часто используется в качестве полигона для тестирования различных методов распознавания образов, машинного обучения, экстракции знаний, статистического анализа данных и т.п. Задача классификации ирисов ставится следующим образом. По известным значениям четырех признаков цветка необходимо отнести

ирис к одному из трех классов: 1 – Iris Setosa, 2 – Iris Versicolor; 3 – Iris Virginica. При принятии решения используются такие признаки: x_1 – длина чашелистика; x_2 – ширина чашелистика; x_3 – длина лепестка; x_4 – ширина лепестка. Исходные данные для классификации ирисов записаны в файле iris.dat. Файл содержит 150 строк, каждая из которых соответствует одному ирису. Информация об ирисе представлена пятеркой чисел: первые четыре числа соответствуют значениям признаков, а пятое число – классу ириса. На рис. 3.110 приведены 2D-распределения ирисов по классам. Цветы Iris Setosa линейно отделимы от остальных классов ирисов по признаку длина лепестка или ширина лепестка, а также по любому набору из двух признаков. В тоже время ирисы оставшихся классов Iris Versicolor и Iris Virginica неотделимы линейно один от другого ни по одному из шести возможных наборов из двух признаков.

Задача кластеризации ирисов состоит в нахождении центров кластеров, т.е. в определении значений признаков типовых ирисов каждого класса. Кластеризация осуществляется по алгоритму нечетких *c*-средних со следующими параметрами:

- экспоненциальный вес – 2,0;
- количество кластеров – 3;
- максимальное количество итераций алгоритма – 100;
- минимальное улучшения целевой функции за одну итерацию – 0,000001.

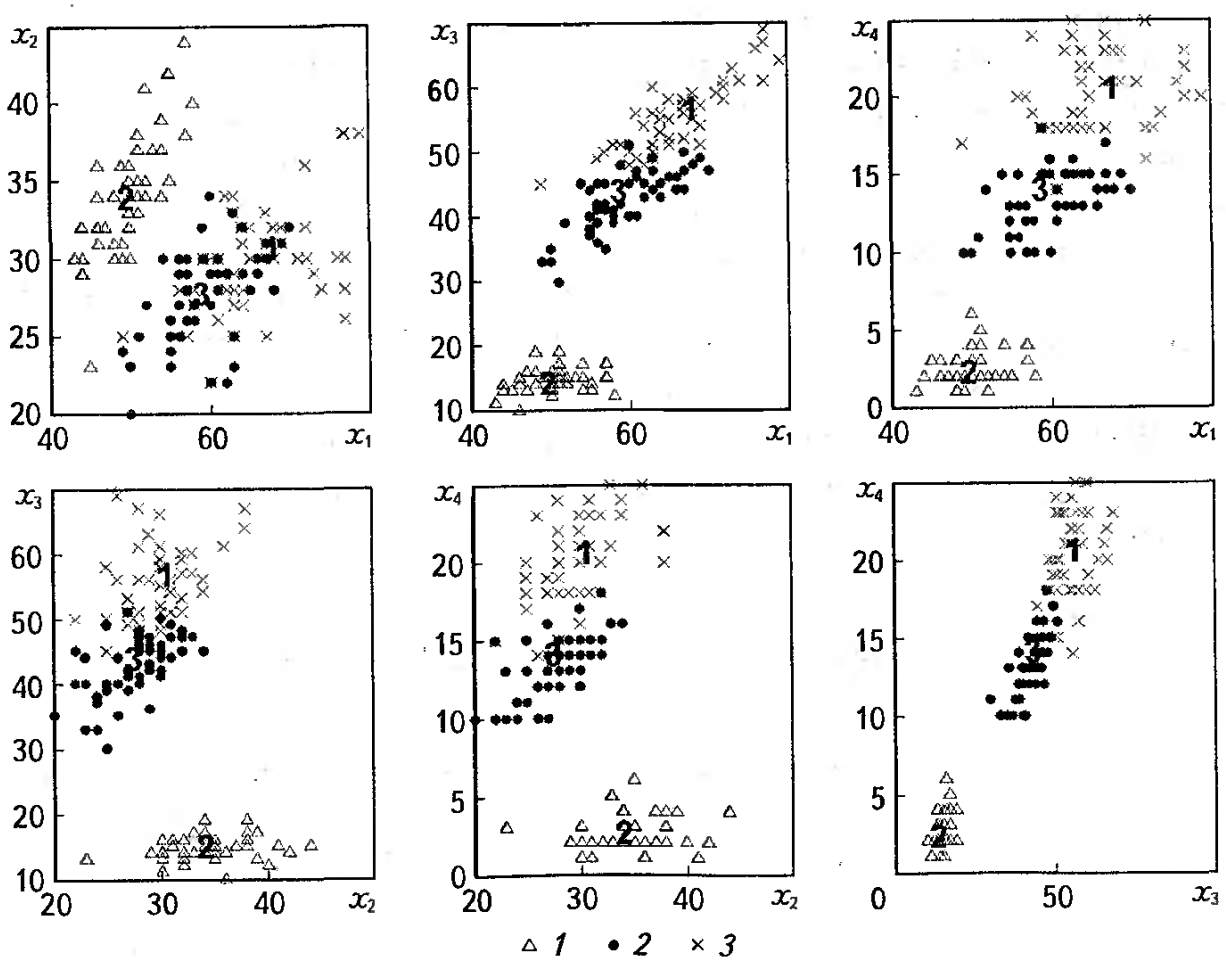


Рис. 3.110. Кластериый анализ ирисов

1 – Iris Setosa; *2* – Iris Versicolor; *3* – Iris Virginica

Во время работы демо-программа irisfcm выводит следующую информацию:

- центры кластеров на каждой итерации алгоритма. Центры кластеров выводятся в виде цифр «1», «2» и «3» на 2D-диаграммы распределения ирисов по классам (см. рис. 3.110). Цифра «1» соответствует типовому ирису класса Iris Virginica, «2» – Iris Setosa, «3» – Iris Versicolor;
- динамику обучения, в виде графика целевой функции на итерациях алгоритма (рис. 3.111);
- значения целевой функции на каждой итерации алгоритма.

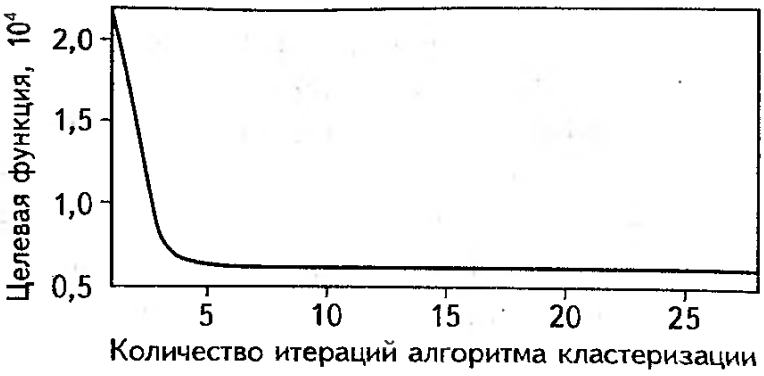


Рис. 3.111. Зависимость качества кластеризации от количества итераций алгоритма

3.4.16. МЕТОДЫ ДЕФАЗЗИФИКАЦИИ

Демонстрационная программа defuzzdm выводит на экран графическое окно (рис. 3.112) с результатами дефаззификации двух нечетких чисел по методам центра тяжести (centroid), медианы (bisector), наибольшего из максимумов (lom), наименьшего из максимумов (som) и центра максимумов (mom). На практике ча-

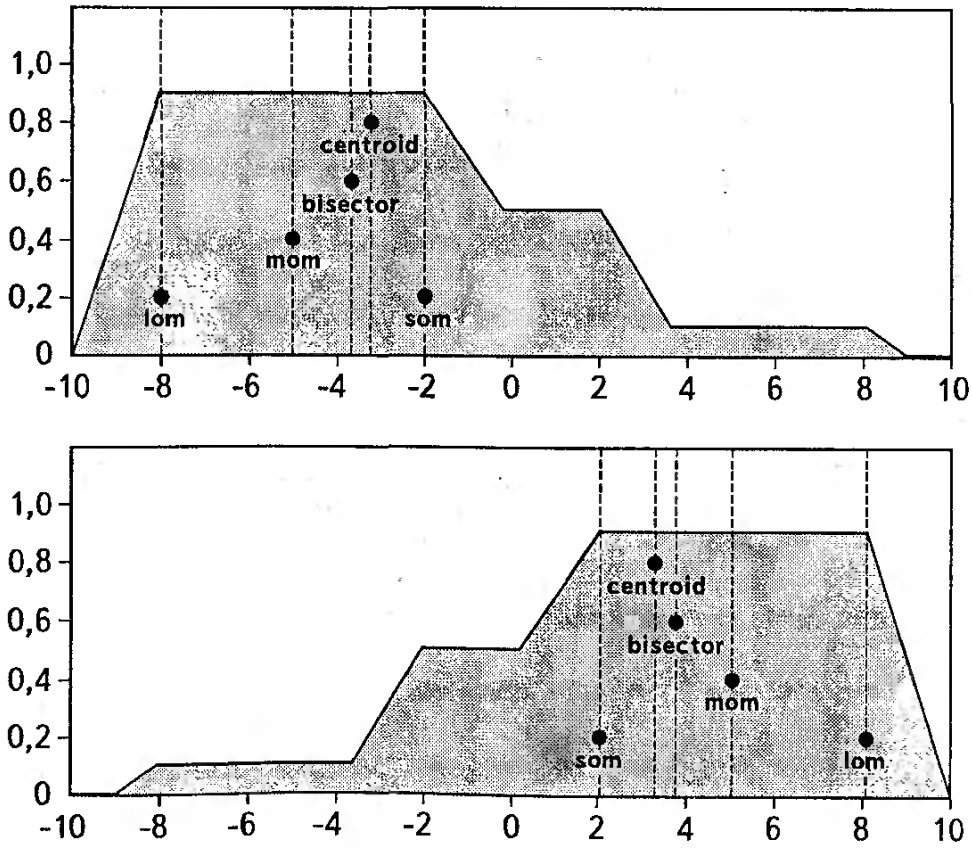


Рис. 3.112. Дефаззификация различными методами

ще применяется дефаззификация по методу центра тяжести. Используя этот метод, необходимо помнить, что диапазон четких (после дефаззификации) значений выходной переменной будет всегда уже интервала, на котором она определена. Этого недостатка лишен второй по популярности метод дефаззификации – центр максимумов. При использовании метода центра максимумов следует учитывать, что результат дефаззификации не чувствителен к вкладу правил с малыми степенями выполнения. Он зависит лишь от правил с максимальными степенями выполнения. Во многих случаях существует только одно такое правило, которое и определяет результат нечеткого вывода.

3.4.17. ГАЛЕРЕЯ ФУНКЦИЙ ПРИНАДЛЕЖНОСТИ

Демонстрационная программа mfdemo выводит на экран окно (рис. 3.113) с графиками всех запрограммированных в Fuzzy Logic Toolbox параметрических функций принадлежности:

- trapmf – трапециевидная функция принадлежности;
- gbellmf – обобщенная колоколообразная функция принадлежности;
- trimf – треугольная функция принадлежности;
- gaussmf – симметричная гауссова функция принадлежности;
- gauss2mf – двухсторонняя гауссова функция принадлежности;
- smf – *s*-подобная функция принадлежности;
- zmf – *z*-подобная функция принадлежности;

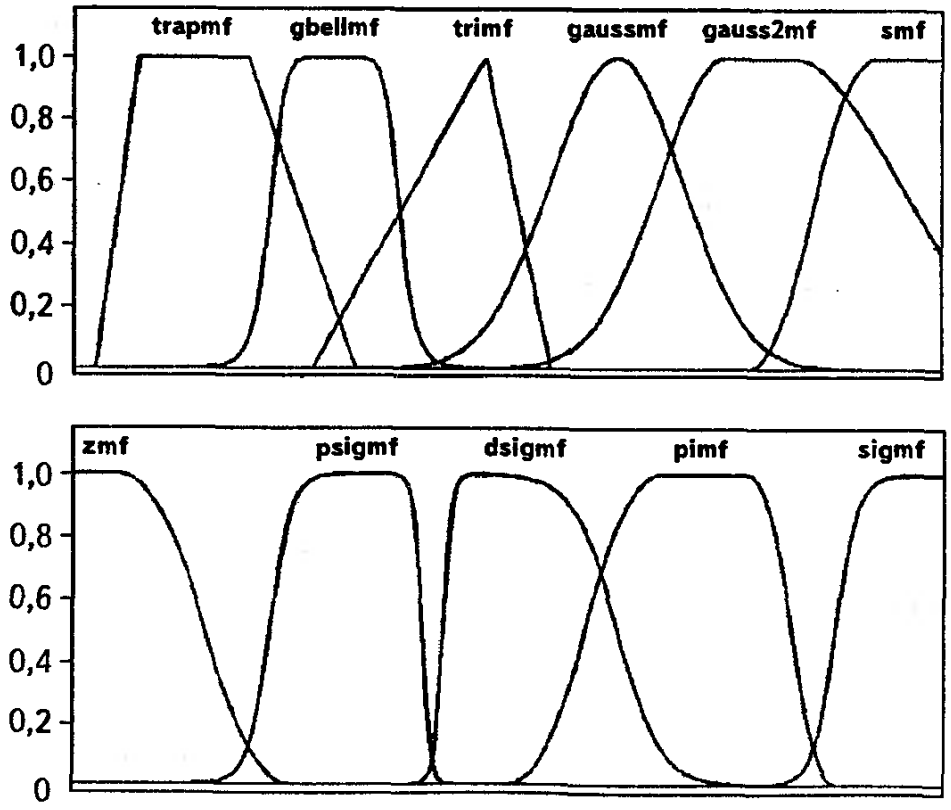


Рис. 3.113. Галерея функций принадлежности

- psigmf – произведение двух сигмоидных функций принадлежности;
- dsigmf – функция принадлежности в виде разности между двумя сигмоидными функциями;
- pimf – пи-подобная функция принадлежности;
- sigmf – сигмоидная функция принадлежности.

3.4.18. КАЛЬКУЛЯТОР ЧАЕВЫХ

Нечеткая система tipper.fis моделирует выделение чаевых официанту в американском ресторане. Для загрузки системы необходимо набрать команду fuzzy tipper.

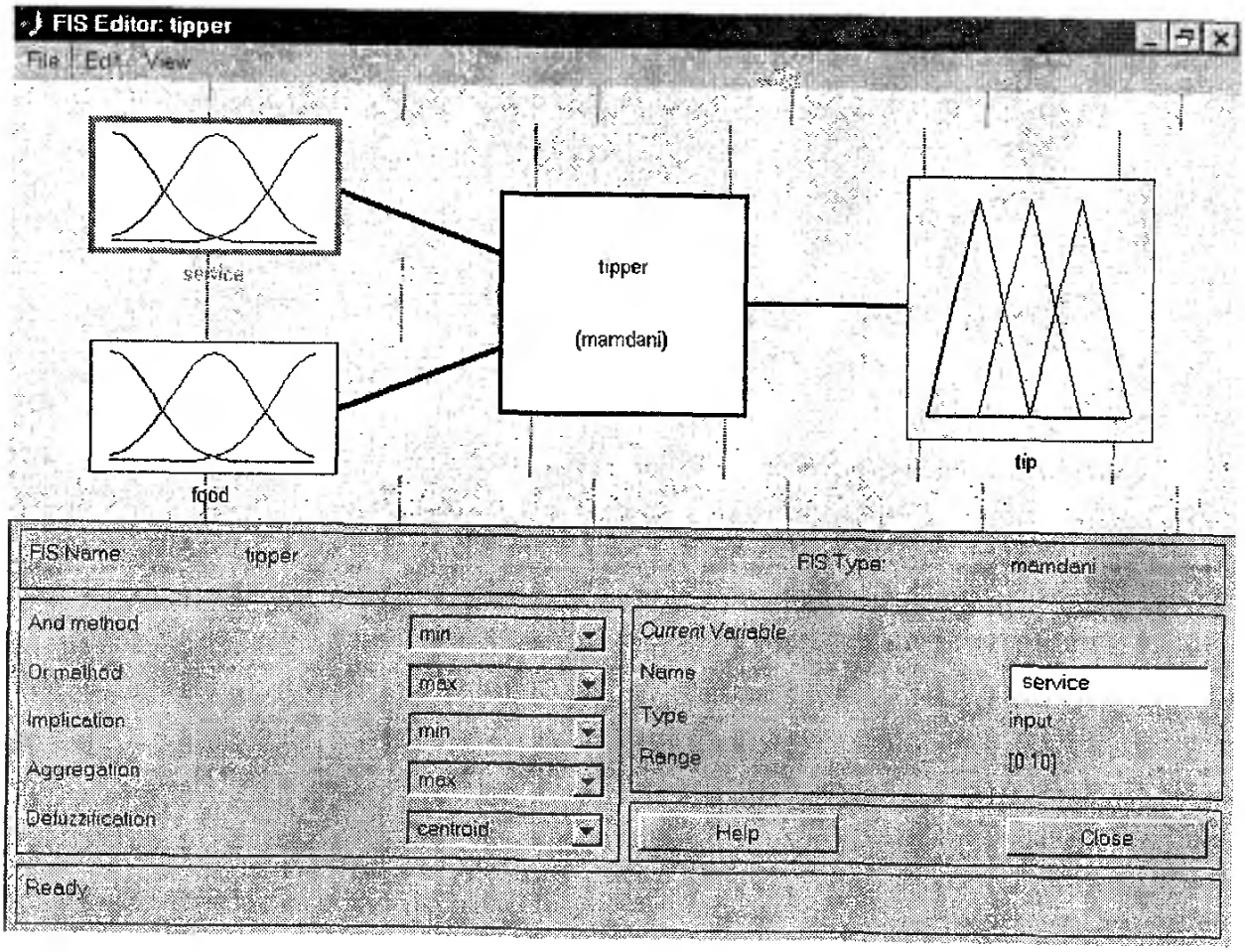


Рис. 3.114. Калькулятор чаевых tipper в FIS-редакторе

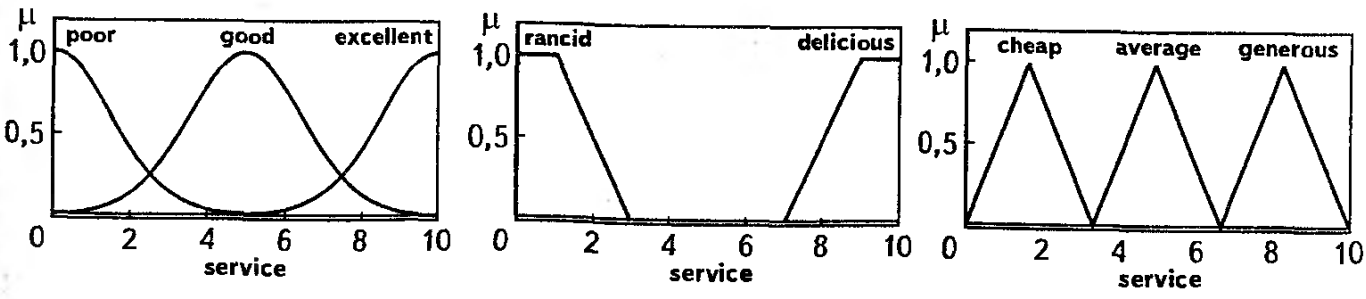


Рис. 3.115. Функции принадлежности переменных демо-системы tipper

Структура нечеткой системы tipper показана на рис. 3.114. Чаевые (**tip**) зависят от уровня обслуживания (**service**) и качества еды (**food**). Уровень обслуживания оценивается тремя термами «плохой» (**poor**), «хороший» (**good**) и «отличный» (**excellent**). Качество еды оценивается двумя термами «пригоревшая» (**rancid**) и «вкусная» (**delicious**). Чаевые могут быть «низкими» (**cheap**), «средними» (**average**) и «щедрыми» (**generous**). Функции принадлежности нечетких термов показаны на рис. 3.115.

Процент чаевых расчитывается по алгоритму Мамдани по следующей базе знаний:

If (service is poor) or (food is rancid) then (tip is cheap);

If (service is good) then (tip is average);

If (service is excellent) or (food is delicious) then (tip is generous).

Поверхности «входы – выход» нечеткой системы tipper приведены на рис. 3.49.

Fuzzy Logic Toolbox содержит также файлы: tippersg.fis – калькулятор чаевых на основе нечеткого вывода Сугено и tipper1.fis – калькулятор чаевых, учитывающий только уровень обслуживания.

3.5. СПРАВОЧНИК ФУНКЦИЙ ПАКЕТА FUZZY LOGIC TOOLBOX

В разделе описываются все функции пакета Fuzzy Logic Toolbox, предназначенные для работы с нечеткими системами. Для каждой функции указаны назначение, синтаксис, входные и выходные переменные, а также пример использования. При описании функций используется аббревиатура «FIS» – система нечеткого вывода.

addmf

Назначение Добавление функции принадлежности в FIS.

Синтаксис
```
fis = addmf(fis, varType, varIndex, mfName,...
mfType, mfParams)
```

Описание Функцию принадлежности можно добавить только к системе нечеткого вывода, существующей в рабочей области MATLAB. Система нечеткого вывода должна каким-то образом быть загружена в рабочую область или создана с помощью функции newfis. Функция addmf имеет шесть входных аргументов:

1) `fis` – идентификатор системы нечеткого вывода в рабочей области MATLAB;

2) `varType` – тип переменной, к которой добавляется функция принадлежности. Допустимые значения: `'input'` – входная переменная и `'output'` – выходная переменная;

3) `varIndex` – порядковый номер переменной, к которой добавляется функция принадлежности;

4) `mfName` – наименование добавляемой функции принадлежности (терм). Задается строкой символов;

5) `mfType` – тип добавляемой функции принадлежности. Задается строкой символов;

6) `mfParams` – вектор параметров добавляемой функции принадлежности.

В нечеткой системе функции принадлежности нумеруются в той последовательности, в которой они добавлены функцией addmf, т.е. первая добавленная функция принадлежности всегда имеет порядковый номер 1. С помощью addmf нельзя добавить функцию принадлежности к несуществующей переменной. Добавить переменную в FIS можно функцией addvar.

Пример

```
fis = addmf(fis, 'input', 1, 'низкий', 'trapmf',...
[150, 155, 165, 170])
```

В терм-множество первой входной переменной нечеткой системы `fis` добавляется терм `'низкий'`, заданный трапециевидной функцией принадлежности с параметрами `[150, 155, 165, 170]`.

addrule

Назначение Добавление нечетких правил в FIS.

Синтаксис

```
fis = addrule(fis, ruleList)
```

Описание Правила можно добавить только к существующей в рабочей области MATLAB нечеткой системе. Функция addrule имеет два входных аргумента:

1) `fis` – идентификатор системы нечеткого вывода в рабочей области MATLAB;

2) `ruleList` – матрица добавляемых правил в индексном формате. Количество строк матрицы `ruleList` равно числу добавляемых правил, т.е. каждая строка соответствует одному правилу. Количество столбцов матрицы равно $m+n+2$, где $m(n)$ – количество входных (выходных) переменных `fis`. Первые m столбцов соответствуют входным переменным; они задают посылки правил. Элементы этих столбцов содержат порядковые номера термов входных переменных. Число 0 указывает, что соответствующая переменная в правиле не задана, т.е. ее значение равно none. Следующие n столбцов соответствуют выходным переменным, т.е. задают заключения правил. Элементы этих столбцов содержат порядковые номера термов выходных переменных. Предпоследний столбец матрицы содержит весовые коэффициенты правил, значения которых должны быть в диапазоне [0, 1]. Последний столбец матрицы задает логические связки между перемен-

ными внутри правил: 1 соответствует логической операции И, 2 – логической операции ИЛИ.

Пример

```
fis = addrule(fis, [1 1 1 1 1; 1 2 2 0.5 1])
```

В базу знаний нечеткой системы `fis` добавляются два следующих правила:

Если вход1 = MF1 и вход2 = MF1, то выход1 = MF1 с весом 1;

Если вход1 = MF1 и вход2 = MF2, то выход1 = MF2 с весом 0,5,

где MF1 и MF2 – термы с порядковыми номерами 1 и 2.

addvar

Назначение Добавление переменной в FIS.

Синтаксис

```
fis = addvar(fis, varType, varName, varBound)
```

Описание Переменную можно добавить только к системе нечеткого вывода, существующей в рабочей области MATLAB. Функция addvar имеет четыре входных аргумента:

1) `fis` – идентификатор системы нечеткого вывода в рабочей области MATLAB;

2) `varType` – тип добавляемой переменной. Допустимые значения: `'input'` – входная переменная и `'output'` – выходная переменная;

3) `varName` – наименование добавляемой переменной. Задается строкой символов;

4) `varBound` – диапазон изменения добавляемой переменной.

Нумерация переменных в `fis` соответствует порядку их добавления функцией addvar, т.е. первая добавленная переменная имеет порядковый номер 1. Входные и выходные переменные нумеруются независимо.

Пример

```
fis = addrule(fis, 'input', 'Рост', [155 205])
```

В нечеткую систему `fis` добавляется входная переменная `'Рост'`, заданная на интервале `[155 205]`.

anfis

Назначение Настройка систем нечеткого вывода типа Сугено.

Синтаксис

```
[fis, error] = anfis(trndata)
[fis, error] = anfis(trndata, initfis)
[fis, error, stepsize] = anfis(trndata, initfis,...
trnopt, dispopt, [], optmethod)
[fis, error, stepsize, chkfis, chkerror] =...
anfis(trndata, initfis, trnopt, dispopt, chkdata)
```

Описание Это основная mex-функция настройки нечетких систем типа Сугено. Настройка представляет собой итерационную процедуру нахождения таких параметров функций принадлежности и коэффициентов в заключениях правил, которые минимизируют расхождения между результатами логического вывода и экспери-

ментальными данными, т.е. между действительным и желаемым поведением системы. Экспериментальные данные для настройки нечеткой системы задаются обучающей выборкой. Функция anfis применяется при проектировании систем типа SISO (один вход – один выход) и типа MISO (много входов – один выход). Функция anfis не работает при нестандартных, т.е. запрограммированных пользователем, типах функций принадлежности и методах дефаззификации.

В основу функции anfis положен один из первых методов создания нейро-нечетких систем для аппроксимации функций, предложенный Янгом в 1991 г. Для настройки параметров нечеткой системы функция anfis использует метод обратного распространения ошибки или гибридный алгоритм обучения. Функция anfis может использовать тестовую выборку для проверки адекватности нечеткой системы.

Функция anfis может иметь шесть входных аргументов:

1) `trndata` – обучающая выборка в виде матрицы, каждая строчка которой задает пару данных «входы – выход». Последний столбец матрицы соответствует вектору значений выхода, а остальные столбцы – входным данным;

2) `initfis` – исходная система нечеткого вывода Сугено нулевого или первого порядка. При отсутствии аргумента `initfis` функция anfis вызовет функцию genfis1 для создания типовой нечеткой системы. В типовой системе будет по две гауссовых функции принадлежности для каждой входной переменной. Для создания системы с иным количеством функций принадлежности необходимо задать их число аргументом `initfis`. Если все переменные оцениваются одинаковым количеством термов, тогда достаточно указать только одно значение;

3) `trnopt` – вектор параметров настройки:

`trnopt(1)` – количество итераций (значение по умолчанию – 10);

`trnopt(2)` – допустимая ошибка обучения (значение по умолчанию – 0);

`trnopt(3)` – начальная длина шага (значение по умолчанию – 0,01);

`trnopt(4)` – коэффициент уменьшения длины шага (значение по умолчанию – 0,9);

`trnopt(5)` – коэффициент увеличения длины шага (значение по умолчанию – 1,1);

4) `dispopt` – вектор, указывающий какие промежуточные результаты обучения выводить в командное окно MATLAB:

`dispopt(1)` – ANFIS-информация: количество функций принадлежности входных и выходной переменных и т.п.;

`dispopt(2)` – ошибка обучения;

`dispopt(3)` – длина шага, которая выводится при его изменении;

`dispopt(4)` – окончательный результат.

Для вывода информации на экран необходимо установить соответствующую координату в 1. По умолчанию выводятся все промежуточные результаты;

5) `chkdata` – тестовая выборка, используемая для исследования генерализирующих свойств нечеткой системы, т.е. ее способности выдавать правильные результаты вне точек обучения. Формат тестовой выборки такой же, как и обучающей. Тестовая выборка особенно важна для задач обучения с большим количеством входов и (или) для задач с зашумленными данными. Обучающая и тестовая выборки должны быть репрезентативными;

6) `optmethod` – метод оптимизации, используемый для настройки. Допустимые значения: 1 – гибридный алгоритм и 0 – метод обратного распространения ошибки. По умолчанию применяется гибридный алгоритм, использующий метод обратного распространения ошибки для настройки функций принадлежности входных переменных и метод наименьших квадратов для настройки коэффициентов в заключениях правил.

Функция anfis может возвратить пять выходных аргументов:

1) `fis` – настроенная система нечеткого вывода, параметры которой минимизируют ошибку на обучающей выборке;

2) `error` – ошибки обучения на каждой итерации, рассчитанные по формуле (2.3) на обучающей выборке;

3) `stepsize` – длины шага алгоритма оптимизации на каждой итерации;

4) `chkfis` – настроенная система нечеткого вывода, параметры которой минимизируют ошибку на тестовой выборке. Для получения такой системы необходимо задать пятый входной аргумент функции anfis – тестовую выборку (`chkdata`);

5) `chkerror` – ошибки тестирования на каждой итерации, рассчитываемые по формуле (2.3) на тестовой выборке.

Настройка прекращается по достижению требуемой ошибки обучения или после выполнения указанного количества итераций.

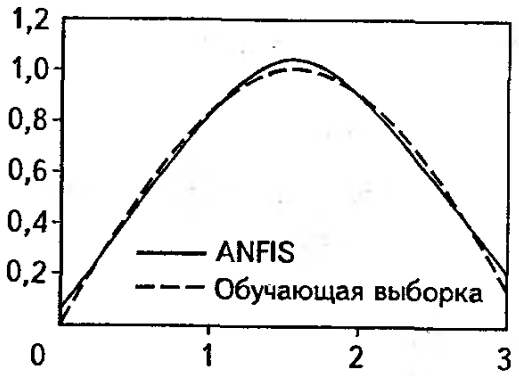
Функция anfis может быть вызвана с одним входным аргументом – обучающей выборкой и одним выходным аргументом – системой нечеткого вывода, параметры которой настроены по обу-

чающей выборке. При инициализации только части аргументов значения остальных необходимо задать как [] или NaN. Значения по умолчанию могут быть изменены непосредственным редактированием файла anfis.m.

Пример

```
x = (0: 0.05: 3)'; y = sin(x); trndata = [x y];
[fis, error, stepsize] = anfis(trndata)
out = evalfis(x, fis);
plot(x, y, 'bo', x, out, 'k-');
legend('Обучающая выборка', 'ANFIS')
```

Синтезируется и настраивается система нечеткого вывода `fis`, моделирующая зависимость $y = \sin(x)$ в диапазоне [0, 3].



convertfis

Назначение Преобразование FIS-матрицы из Fuzzy Logic Toolbox v.1 в FIS-структуру для Fuzzy Logic Toolbox v.2.

Синтаксис

```
fis_new = convertis(fis_old)
```

Описание Преобразовывает систему нечеткого вывода, заданную матрицей `fis_old` в формате Fuzzy Logic Toolbox v.1, в структуру `fis_new` в соответствии с форматом Fuzzy Logic Toolbox v.2.

defuzz

Назначение Дефаззификация нечеткого множества.

Синтаксис

```
crisp = defuzz(x, mf, method)
```

Описание Выполняет дефаззификацию, т.е. преобразование нечеткого множества в четкое число. Функция defuzz имеет три входных аргумента:

1) `x` – универсальное множество, на котором задано нечеткое множество;

2) `mf` – вектор степеней принадлежности элементов множества `x` нечеткому множеству, подлежащему дефаззификации;

3) `method` – метод дефаззификации. Допустимые значения: centroid – центр тяжести; bisector – медиана; mom – центр максимумов; som – наименьший из максимумов; lom – наибольший из максимумов.

Методы дефаззификации описаны в табл. 1.2. Если метод дефаззификации отличается от указанных, тогда он должен быть реализован *m*-функцией. В этом случае значения аргументов `x` и `mf` передаются этой функции для выполнения дефаззификации.

Пример

```
x = 0: 0.1: 10';
mf = trapmf(x, [0, 2, 4, 10]);
crisp = defuzz(x, mf, 'centroid')
```

Дефаззификация нечеткого множества с трапециевидной функцией принадлежности с параметрами [0, 2, 4, 10], заданного на универсальном множестве {0; 0,1; 0,2; ...; 10}.

discfis

Назначение Дискретизация функций принадлежности нечетких множеств из FIS.

Синтаксис

```
[XI, YI, XO, YO, R] = discfis(fis, numPts)
```

Описание Функция используется для ускорения нечеткого вывода посредством дискретизации функций принадлежности всех нечетких множеств, входящих в FIS. Функция discfis может иметь два входных аргумента:

1) `fis` – система нечеткого вывода;

2) `numPts` – необязательный входной аргумент, задающий количество точек дискретизации функций принадлежности. Значение по умолчанию равно 181. Это означает, что все нечеткие множества представляются в виде 181 пары чисел «элемент универсального множества – степень принадлежности».

Функция discfis возвращает пять выходных аргументов:

1) `XI` – матрица универсальных множеств, на которых заданы нечеткие термы входных переменных. Размер матрицы `numPts`×Ninp, где Ninp – количество термов для лингвистической оценки входных переменных;

2) `YI` – матрица степеней принадлежности элементов матрицы `XI` соответствующим нечетким термам. Размер матрицы `numPts`×Ninp. Первый столбец матрицы содержит степени принадлежности первому терму первой входной переменной, а последний столбец – последнему терму последней входной переменной;

3) `XO` – матрица универсальных множеств, на которых заданы нечеткие термы выходных переменных. Размер матрицы `numPts`×Nout, где Nout – количество термов для лингвистической оценки выходных переменных;

4) `YO` – матрица степеней принадлежности элементов матрицы `XO` соответствующим термам. Размер матрицы

numPts×Nout. Первый столбец матрицы содержит степени принадлежности первому терму первой выходной переменной, а последний столбец – последнему терму последней выходной переменной. Для нечеткой системы типа Сугено матрица YO является нулевой;

5) R – список правил базы знаний в индексном формате.

Пример

```
fis = readfis('tipper');
[XI, YI, XO, YO, R] = discfis(fis)
```

Дискретизация функций принадлежностей демо-системы нечеткого вывода tipper.

distfcm

Назначение Расчет расстояния по Евклиду.

Синтаксис

```
out = distfcm(center, data)
```

Описание Рассчитывает евклидово расстояние между центром кластера и объектом. Функция имеет два входных аргумента:

1) center – матрица, каждая строка которой задает координаты центра одного кластера;

2) data – матрица, каждый столбец которой задает координаты одного объекта.

Функция distfcm возвращает матрицу расстояний out размером $M \times N$, где M – количество центров кластеров; N – количество объектов. Элемент out(i, j) соответствует расстоянию между i-м кластером и j-м объектом, т.е. расстоянию между center(i, :) и data(:, j).

Пример

```
center = [1 1; 2 2];
data = [0 1 1 4; 1 2 3 0]';
out = distfcm(center, data)
```

Расчет расстояний по Евклиду между центрами кластеров (1, 1) и (2, 2) и объектами с координатами (0, 1), (1, 2), (1, 3) и (4, 0).

dsigmf

Назначение Функция принадлежности в виде разности двух сигмоидных функций.

Синтаксис

```
y = dsigmf(x, params)
```

Описание Задает функцию принадлежности в виде разности двух сигмоидных функций по формуле:

$$\mu(x) = \frac{1}{1+e^{-a_1(x-c_1)}} - \frac{1}{1+e^{-a_2(x-c_2)}}.$$

Применяется для задания гладких асимметричных функций принадлежности.

Функция dsigmf имеет два входных аргумента:

1) x – вектор, для координат которого рассчитываются степени принадлежности;

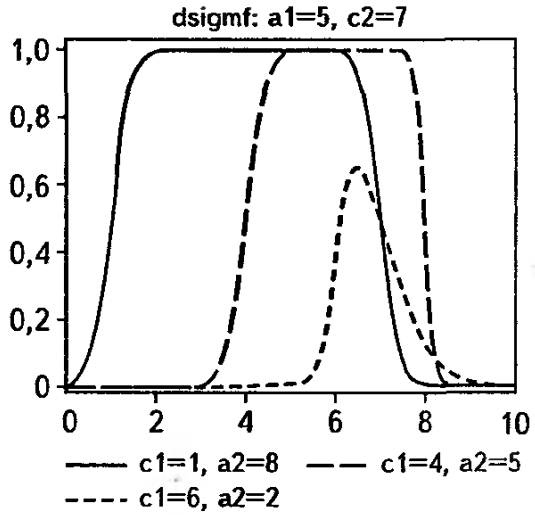
2) params – параметры функции принадлежности. Порядок задания параметров: [a1 c1 a2 c2].

Функция dsigmf возвращает вектор степеней принадлежности.

Пример

```
x = 0: 0.25: 10;
y1 = dsigmf(x, [5 1 8 7]);
y2 = dsigmf(x, [5 4 5 7]);
y3 = dsigmf(x, [5 6 2 7]);
plot(x, y1, 's-', x, y2, '.:', x, y3, '+–');
ylim([0 1.05]); title('dsigmf: a1=5, c2=7');
legend('c1=1, a2=8', 'c1=4, a2=5', 'c1=6, a2=2')
```

Построение графиков функций принадлежности в виде разности между двумя сигмоидными функциями с разными параметрами на интервале [0, 10].



evalfis

Назначение Нечеткий логический вывод.

Синтаксис

```
output = evalfis(input, fis)
output = evalfis(input, fis, numPts)
[output, IRR, ORR, ARR] = evalfis(input, fis)
[output, IRR, ORR, ARR] = evalfis(input, fis, numPts)
```

Описание Выполняет нечеткий вывод. Функция evalfis может иметь три входных аргумента:

1) input – матрица значений входных переменных, для которых необходимо выполнить нечеткий вывод. Размер матрицы $M \times N$, где N – количество входных переменных; M – количество входных данных. Каждая строчка матрицы представляет один вектор значений входных переменных;

2) fis – система нечеткого вывода;

3) numPts – необязательный аргумент, задающий количество точек дискретизации функций принадлежности. Значение

по умолчанию равно 101, которое означает, что все нечеткие множества представляются в виде 101 пары чисел «элемент универсального множества – степень принадлежности». При уменьшении точек дискретизации возрастает скорость логического вывода и уменьшается точность вычислений.

Функция evalfis может иметь четыре выходных аргумента:

1) `output` – матрица значений выходных переменных, получаемая в результате нечеткого вывода для данных `input`. Размер матрицы $M \times L$, где M – количество входных данных; L – количество выходных переменных в `fis`;

2) `IRR` – матрица степеней принадлежности входных значений нечетким термам из базы знаний. Размер матрицы $NR \times N$, где NR – количество правил в `fis`; N – количество входных переменных;

3) `ORR` – матрица размером `numPts`×($NR \times L$). Каждый столбец матрицы содержит функцию принадлежности выходной переменной, получаемую в результате вывода по одному правилу. Функция принадлежности дискретизуется на `numPts` точках и представляется вектором степеней принадлежности;

4) `ARR` – матрица размером `numPts`×L, содержащая функции принадлежности выходных переменных в результате нечеткого вывода по всей базе знаний. Функции принадлежности дискретизуются на `numPts` точках и представляются вектором степеней принадлежности.

Аргументы `IRR`, `ORR` и `ARR` являются необязательными, они содержат промежуточные результаты нечеткого вывода. При задании нескольких входных данных значения этих аргументов рассчитываются только для последнего вектора данных. Аргументы используются для отслеживания процесса логического вывода или при реализации нестандартной процедуры нечеткого вывода.

Пример

```
fis = readfis('tipper');
tip = evalfis([3 8], fis)
```

Первая строчка загружает демо-систему нечеткого вывода tipper, предназначенную для определения процента чаевых в ресторане. Вторая строчка рассчитывает размер чаевых, когда service = 3 и food = 8.

evalmf

Назначение Вычисление значений произвольной функции принадлежности.

Синтаксис

```
y = evalmf(x, params, type)
```

Описание Вычисляет значения произвольной функции принадлежности. Функцию evalmf вызывают с тремя входными аргументами:

1) x – вектор, для координат которого рассчитываются степени принадлежности;

2) params – параметры функции принадлежности, порядок которых определяется ее типом;

3) type – тип функции принадлежности. Значение аргумента можно задать строчкой символов или числом: 1 – 'trimf'; 2 – 'trapmf'; 3 – 'gaussmf'; 4 – 'gauss2mf'; 5 – 'sigmf'; 6 – 'dsigmf'; 7 – 'psigmf'; 8 – 'gbellmf'; 9 – 'smf'; 10 – 'zmf'; 11 – 'pimf'. При задании другого типа функции принадлежности предполагается, что она реализована соответствующим *m*-файлом.

Функция evalmf возвращает вектор степеней принадлежности координат вектора x.

Пример

```
x = 0: 0.1: 10;
%Расчет значений треугольной функции принадлежности:
y = evalmf(x, [0 3 9], 1);
plot(x, y);
title('Треугольная функция принадлежности с
%параметрами [0 3 9]')
```

evalmmf

Назначение Расчет степеней принадлежностей для нескольких функций принадлежности.

Синтаксис

```
y = evalmmf(x, params, types)
```

Описание Вычисляет значения нескольких функций принадлежности нечетких множеств, заданных на одном и том же универсальном множестве. Функция evalmmf может иметь три входных аргумента:

1) x – вектор, для координат которого рассчитываются степени принадлежности;

2) params – матрица параметров функций принадлежности. Первая строка матрицы определяет параметры первой функ-

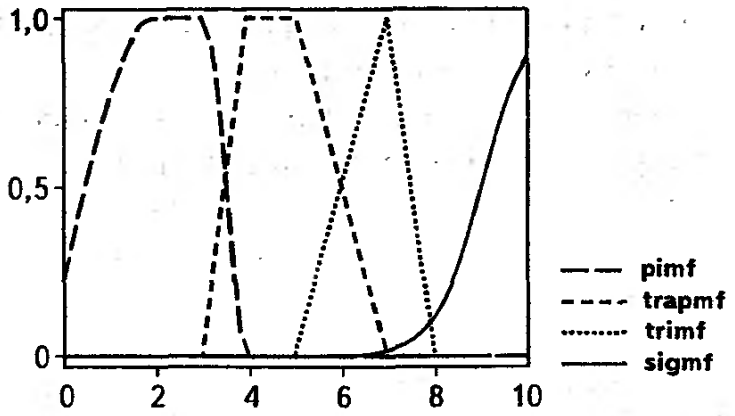
ции принадлежности, вторая строка – параметры второй функции принадлежности и т.д.;

3) `types` – матрица типов функций принадлежности. Первая строка матрицы задает тип первой функции принадлежности, вторая строка – тип второй функции принадлежности и т.д. Тип функции принадлежности можно задать строчкой символов или числом: 1 – 'trimf'; 2 – 'trapmf'; 3 – 'gaussmf'; 4 – 'gauss2mf'; 5 – 'sigmf'; 6 – 'dsigmf'; 7 – 'psigmf'; 8 – 'gbellmf'; 9 – 'smf'; 10 – 'zmf'; 11 – 'pimf'. При задании другого типа функции принадлежности предполагается, что она реализована соответствующим m-файлом.

Функция evalmmf возвращает матрицу y, содержащую степени принадлежности координат вектора x. Первая строка матрицы содержит значения первой функции принадлежности, вторая строка – значения второй функции принадлежности и т.д.

Пример

```
x = 0: 0.2: 10;
params = [-1 2 3 4; 3 4 5 7; 5 7 8 0; 2 9 0 0];
type = str2mat('pimf', 'trapmf', 'trimf', 'sigmf');
%Расчет значений пи-подобной, трапециевидной,
%треугольной и сигмоидной функций принадлежности
%для координат вектора x:
mf = evalmmf(x, params, type);
plot(x', mf'); legend(type)
```



fcm

Назначение Кластеризация алгоритмом нечетких *c*-средних.

Синтаксис

```
[V, M, obj_fcn] = fcm(X, c)
[V, M, obj_fcn] = fcm(X, c, options)
```

Описание Выполняет кластеризацию данных по алгоритму нечетких *c*-средних. Функция fcm может иметь три входных аргумента:

1) `X` – матрица данных для кластеризации. Каждая строка матрицы соответствует одному объекту кластеризации;

2) `c` – количество кластеров. Количество кластеров должно быть больше 1 и меньше числа объектов, заданных матрицей `X`;

3) options – необязательный аргумент, устанавливающий параметры алгоритма нечеткой кластеризации:

options(1) – экспоненциальный вес (значение по умолчанию – 2,0);

options(2) – максимальное количество итераций алгоритма кластеризации (значение по умолчанию – 100);

options(3) – минимально допустимое улучшение целевой функции за одну итерацию алгоритма (значение по умолчанию – 0,00001);

options(4) – опция вывода промежуточных результатов кластеризации (значение по умолчанию: 1 – вывод промежуточных результатов).

Для использования значений по умолчанию необходимо напечатать NaN в соответствующей координате вектора options.

Алгоритм нечеткой кластеризации останавливается после выполнения заданного числа итераций или когда улучшение целевой функции за одну итерацию меньше указанного минимального значения.

Функция fcm имеет три выходных аргумента:

1) V – матрица координат найденных центров нечетких кластеров. Каждая строка матрицы соответствует центру одного кластера;

2) M – матрица степеней принадлежности объектов кластерам. Каждая строка матрицы соответствует функции принадлежности одного кластера;

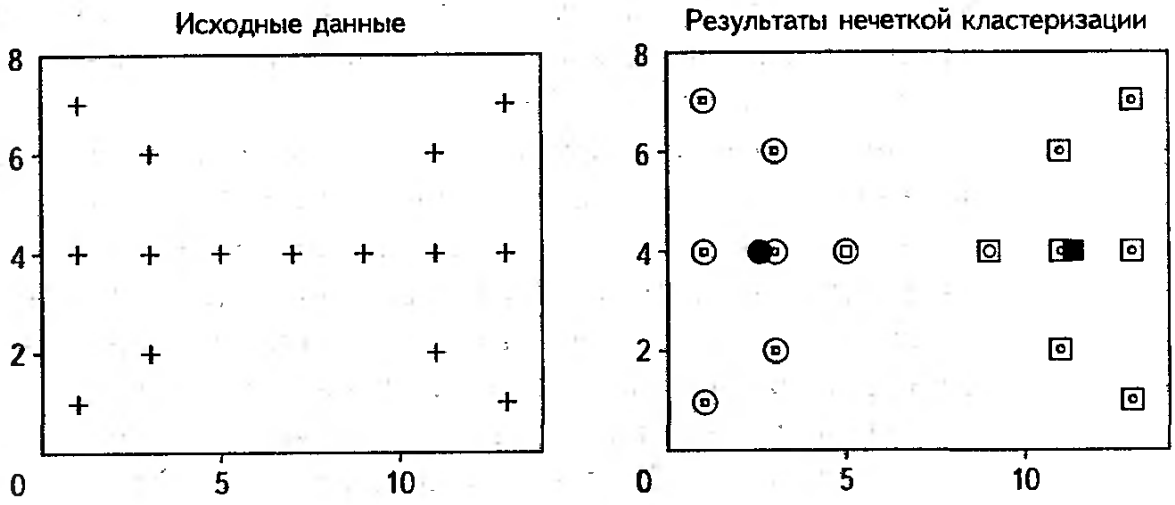
3) obj_fcn – вектор значений целевой функции на каждой итерации алгоритма кластеризации.

Пример

```
X = [1 1; 1 4; 1 7; 3 2; 3 4; 3 6; 5 4; 7 4; 9 4;...
11 2; 11 4; 11 6; 13 1; 13 4; 13 7];
[V M opt] = fcm(X, 2)
subplot(1, 2, 1);
plot(X(:, 1), X(:, 2), 'k+', 'markersize', 6)
title('Исходные данные'); xlim([0 14]); ylim([0 8])
subplot(1, 2, 2);
%Центры кластеров:
plot(V(1, 1), V(1, 2), 'ro', 'markersize', 8,...
'markerfacecolor', 'r')
hold on
plot(V(2, 1), V(2, 2), 'bs', 'markersize', 8,...
'markerfacecolor', 'b')
%Вывод результатов нечеткой кластеризации:
for i = 1:15;
plot(X(i, 1), X(i, 2), 'ro', 'markersize', M(1, i)*8+2);
plot(X(i, 1), X(i, 2), 'bs', 'markersize', M(2, i)*8+2);
end
```

```
xlim([0 14]); ylim([0 8]);
title('Результаты нечеткой кластеризации')
```

Проводится кластеризация объектов, образующих фигуру типа «бабочка». На рисунке размер маркера пропорционален степени принадлежности объекта нечеткому кластеру. Расположенный в центре объект имеет одинаковые степени принадлежности к двум кластерам. Центры нечетких кластеров указаны залитыми маркерами.



fuzarith

Назначение Нечеткий калькулятор.

Синтаксис

```
c = fuzarith(x, a, b, operator)
```

Описание Выполняет арифметические операции сложения, вычитания, умножения и деления над нечеткими числами. Функция fuzarith имеет четыре входных аргумента:

1) `x` – универсальное множество, на котором заданы нечеткие числа;

2) `a` – первый операнд в виде вектора степеней принадлежности элементов универсального множества первому нечеткому множеству. Другими словами, аргументы `x` и `a` образуют первое нечеткое число;

3) `b` – второй операнд в виде вектора степеней принадлежности элементов универсального множества второму нечеткому множеству, т.е. аргументы `x` и `b` образуют второе нечеткое число;

4) `operator` – арифметическая операция: 'sum' – сложение; 'sub' – вычитание; 'prod' – умножение; 'div' – деление.

Выходным аргументом функции fuzarith является вектор степеней принадлежности элементов универсального множества `x` результату выполнения нечеткой арифметической операции. Длины векторов `x`, `a`, `b` и `c` должны совпадать.

Нечеткие арифметические операции выполняются по следующему алгоритму:

- преобразование нечетких чисел–операндов в α-уровневые нечеткие множества;
- выполнение арифметической операции для каждого α-уровня по принципу обобщения;
- преобразование результирующего нечеткого числа к традиционному виду.

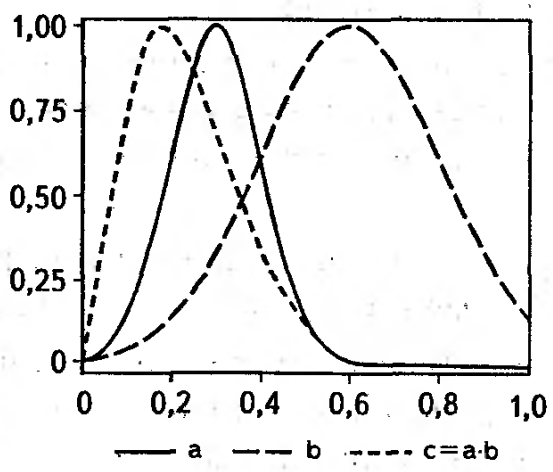
Функция fuzarith использует стандартные процедуры интерполяции для выполнения указанных выше преобразований. Количество α-уровней равно 101.

Деление не выполняется, если носитель нечеткого множества содержит как отрицательные, так и положительные числа, чтобы не делить на ноль. При выполнении арифметических операций над нечеткими данными, как правило, носитель результирующего нечеткого числа отличен от носителей нечетких операндов. Функция fuzarith выводит степени принадлежности результата только для универсального множества нечетких операндов, т.е. для множества x. Для получения всего результирующего нечеткого числа необходимо добавить в заголовок функции fuzarith одну выходную переменную `intervalC`. Эта переменная содержит результат арифметической операции в виде нечеткого множества в α-уровневом разложении.

Пример

```
x = 0: 0.025: 1;
a = gaussmf(x, [0.1 0.3]); b = gaussmf(x, [0.2 0.6]);
c = fuzarith(x, a, b, 'prod');
plot(x, a, '-o', x, b, ':+', x, c);
legend('a', 'b', 'c=a*b')
```

Рассчитывается нечеткое число c как произведение нечетких чисел a и b с гауссовыми функциями принадлежности, заданными на универсальном множестве {0, 0,025, ..., 1}. Графики функций принадлежности показаны на рисунке.



gauss2mf

Назначение Двухсторонняя гауссова функция принадлежности.

Синтаксис `y = gauss2mf(x, params)`

Описание Задает функцию принадлежности в виде следующей комбинации двух гауссовых кривых:

если $b_1 < b_2$,

$$\text{то } \mu(x) = \begin{cases} \exp((x-b_1)^2/(-2c_1^2)), & x < b_1; \\ 1, \quad b_1 \le x \le b_2; \\ \exp((x-b_2)^2/(-2c_2^2)), & x > b_2; \end{cases}$$

если $b_1 > b_2$,

$$\text{то } \mu(x) = \begin{cases} \exp((x-b_1)^2/(-2c_1^2)), & x < b_2; \\ \exp((x-b_1)^2/(-2c_1^2))\exp((x-b_2)^2/(-2c_2^2)), & b_2 \le x \le b_1; \\ \exp((x-b_2)^2/(-2c_2^2)), & x > b_1. \end{cases}$$

Если $b_1 < b_2$, то параметры функции принадлежности геометрически интерпретируются следующим образом:

b_1 и b_2 – нижняя и верхняя границы ядра нечеткого множества;

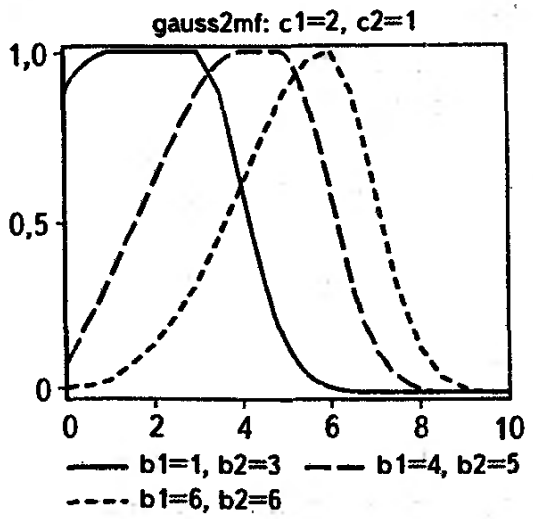
c_1 и c_2 – коэффициенты концентраций левой и правой ветвей графика функции принадлежности.

При $b_1 > b_2$ нечеткое множество получается субнормальным.

Функция gauss2mf применяется для задания гладких асимметричных функций принадлежности. Функция gauss2mf имеет два входных аргумента:

1) `x` – вектор, для координат которого рассчитываются степени принадлежности;

2) `params` – вектор параметров функции принадлежности. Порядок задания параметров: `[c1 b1 c2 b2]`.



Функция gauss2mf возвращает степени принадлежности координат вектора x.

Пример

```
x = 0: 0.5: 10;
y1 = gauss2mf(x, [2 1 1 3]);
y2 = gauss2mf (x, [2 4 1 5]);
y3 = gauss2mf(x, [2 6 1 6]);
plot(x, y1, 'o-', x, y2, '+:', x, y3, '.-');
title('gauss2mf, c1=2, c2=1');
legend('b1=1, b2=3', 'b1=4, b2=5', 'b1=6, b2=6')
```

Построение графиков двухсторонних гауссовых функций принадлежности с различными параметрами на интервале [0, 10].

gaussmf

Назначение Гауссова функция принадлежности.

Синтаксис

```
y = gaussmf(x, params)
```

Описание Задает функцию принадлежности в виде симметричной гауссовой кривой согласно формуле:

$$\mu(x) = e^{-\frac{(x-b)^2}{2c^2}}.$$

Параметры функции принадлежности геометрически интерпретируются следующим образом:

b – координата максимума функции принадлежности;

c – коэффициент концентрации функции принадлежности.

Функция gaussmf применяется для задания гладких симметричных функций принадлежности. Функция gaussmf имеет два входных аргумента:

1) x – вектор, для координат которого рассчитываются степени принадлежности;

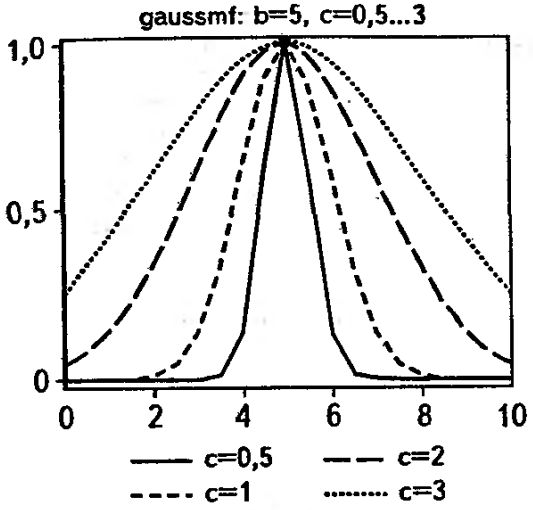
2) params – вектор параметров функции принадлежности. Порядок задания параметров: [c b].

Функция gaussmf возвращает степени принадлежности координат вектора x.

Пример

```
x = 0: 0.5: 10;
y1 = gaussmf(x, [0.5 5]);
y2 = gaussmf(x, [1 5]);
y3 = gaussmf(x, [2 5]);
y4 = gaussmf(x, [3 5]);
plot(x, y1, 'o-', x, y2, '+:', x, y3, '.-', x,...
y4, '-^')
title('gaussmf, b=5, c=0.5...3');
legend('c=0.5', 'c=1', 'c=2', 'c=3')
```

Построение графиков симметричных гауссовых функций принадлежности с различными коэффициентами концентрации.



gbellmf

Назначение Обобщенная колоколообразная функция принадлежности.

Синтаксис `y = gbellmf(x, params)`

Описание Задает функцию принадлежности в виде симметричной кривой в форме колокола. Эта функция задается формулой

$$\mu(x)=\frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}},$$

параметры которой геометрически интерпретируются следующим образом:

a – коэффициент концентрации функции принадлежности;

b – коэффициент крутизны функции принадлежности ($b>0$);

c – координата максимума функции принадлежности.

Функция gbellmf применяется для задания гладких симметричных функций принадлежности. Функция gbellmf имеет два входных аргумента:

1) `x` – вектор, для координат которого рассчитываются степени принадлежности;

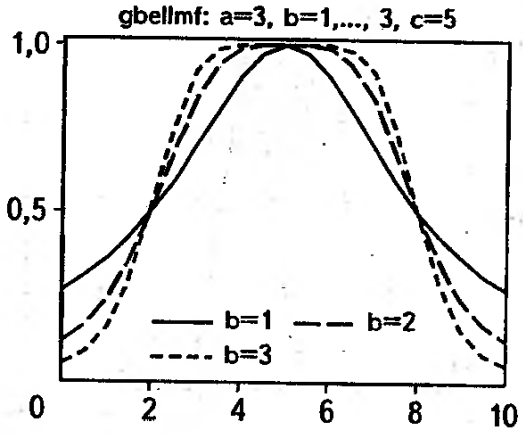
2) `params` – вектор параметров функции принадлежности. Порядок задания параметров: `[a b c]`.

Функция gbellmf возвращает степени принадлежности координат вектора `x`.

Пример

```
x = 0: 0.5: 10;
y1 = gbellmf(x, [3 1 5]); y2 = gbellmf(x, [3 2 5]);
y3 = gbellmf(x, [3 3 5]);
plot(x, y1, 'o-', x, y2, '+:', x, y3, '.-')
title('gbellmf, a=3, b=1, ..., 3, c=5');
legend('b=1', 'b=2', 'b=3')
```

Построение графиков колоколообразных функций принадлежности с различными коэффициентами крутизны.



genfis1

Назначение Генерирование из данных исходной FIS типа Сугено без использования кластеризации.

Синтаксис

```
fis = genfis1(data, numMFs, inmftype, outmftype)
```

Описание Генерирует из данных систему нечеткого вывода типа Сугено по методу решетчатого разбиения. Функции принадлежностей входных переменных выбираются таким образом, чтобы равномерно покрыть диапазоны изменения данных. Объем базы знаний определяется как произведение мощностей терм-множеств входных переменных, следовательно, генерируется все возможные правила. Коэффициенты в заключениях правил принимаются равными нулю. Это означает, что при любых значениях входных переменных на выходе системы будет нулевое значение. Полученная система нечеткого вывода не отражает представленные данными закономерности. Она является исходной системой для обучения посредством ANFIS, в результате которого закономерности, заложенные в данных, будут идентифицированы.

Функция genfis1 может иметь до четырех входных аргументов:

1) `data` – матрица исходных данных, каждая строчка которой является парой «входы – выход»;

2) `numMFs` – необязательный аргумент, задающий количество термов для оценки входных переменных. Если количество термов одинаковое для всех переменных, тогда достаточно задать скалярное значение этого аргумента. Значение по умолчанию – 2;

3) `inmftype` – необязательный аргумент, задающий типы функций принадлежности нечетких термов входных переменных. Значение этого аргумента в виде одной строки символов указывает на то, что используются функции принадлежности одного типа. Массив строк задает тип функций принадлежностей для каждой входной переменной. По умолчанию используется обобщенная колоколообразная функция принадлежности;

4) `outmftype` – необязательный аргумент, задающий тип заключений нечетких правил Сугено. Допустимые значения: 'linear' – линейная (значение по умолчанию) и 'constant' – константа.

Функция возвращает систему нечеткого логического вывода типа Сугено.

Пример

```
data = [rand(10,1) 10*rand(10,1)-5 rand(10,1)];
fis = genfis1(data, [3 7], char('pimf', 'trimf'))
```

Генерирование системы нечеткого вывода Сугено из данных `data`. Система использует три терма с пи-подобными функциями принадлежности для оценки первой входной переменной и семь термов с треугольными функциями принадлежности для оценки второй входной переменной.

genfis2

Назначение

Генерирование из данных исходной FIS типа Сугено посредством субтрактивной кластеризации.

Синтаксис

```
fis = genfis2(Xin, Xout, radii, xBounds, options)
```

Описание

Генерирует из данных нечеткую систему типа Сугено с применением субтрактивной кластеризации. При использовании данных только с одной выходной переменной результат выполнения функции genfis2 может рассматриваться как исходная нечеткая система для ANFIS-обучения.

Экстракция нечетких правил происходит в два этапа. Вначале функция subclust находит посылки нечетких правил. Затем по методу наименьших квадратов рассчитываются заключения правил. В результате получается система нечеткого вывода с базой правил, покрывающей всю предметную область.

Функция genfis2 может иметь до пяти входных аргументов, первые три из которых обязательны:

1) `Xin` – матрица, каждая строчка которой содержит значения входных переменных;

2) `Xout` – матрица, каждая строчка которой содержит значения выходных переменных;

3) `radii` – вектор, определяющий размеры области действия правил по каждой координате. Координаты вектора `radii` должны находиться в диапазоне [0, 1], так как функция subclust масштабирует данные на единичный гиперкуб. Если значение `radii` задано скаляром, тогда все координаты считаются равноважными;

4) `xBounds` – матрица диапазонов изменения данных, необходимая для их масштабирования на единичный гиперкуб. Каждый столбец матрицы задает диапазон изменения данных по одной координате, т.е. размер матрицы равен $2 \times p$, где

p – количество входных и выходных переменных. Если аргумент `xBounds` не задан, тогда диапазоны изменения данных рассчитываются по фактическим значениям матриц `Xin` и `Xout`;

5) `options` – вектор параметров кластерного анализа, информация о которых приведена в описании функции subclust.

Пример

```
x = 2*rand(100, 2); y = x(:, 1).^2+2*x(:, 2);
fis = genfis2(x, y, 0.5)
y_fis = evalfis(x, fis);
plot(y, y_fis, 'r.')
hold on
min_y = min(y); max_y = max(y);
plot([min_y max_y], [min_y max_y], '-k')
xlabel('Экспериментальные данные');
ylabel('Нечеткое моделирование')
```

В первой строчке примера задаются 100 пар «входы – выход», связанных зависимостью $y = x_1^2 + 2x_2$. Затем генерируется система нечеткого вывода, которая идентифицирует эту зависимость по представленным данным. На рисунке показано желаемое (сплошная линия) и действительное (черные точки) поведения нечеткой модели. Как видно из рисунка, даже без использования технологии обучения ANFIS, нечеткая модель, синтезированная функцией genfis2, хорошо описывает данные.

genparam

Назначение Генерирование исходных параметров функций принадлежности для ANFIS-обучения.

Синтаксис

```
mf_param = genparam(data, mf_n, mf_type)
```

Описание Генерирует параметры функций принадлежности таким образом, чтобы ядра нечетких множеств размещались через равные промежутки внутри интервалов изменения данных. Функция genparam может иметь три входных аргумента:

1) `data` – матрица данных «входы – выход» (обязательный аргумент). Каждая строка данных содержит одну пару «входы – выход». Последний столбец матрицы содержит значения выходной переменной, а остальные – значения входных переменных;

2) `mf_n` – вектор, задающий количество термов для каждой входной переменной. Если аргумент задан скаляром, то количество термов принимается одинаковым для всех входных переменных. Значение по умолчанию – 2;

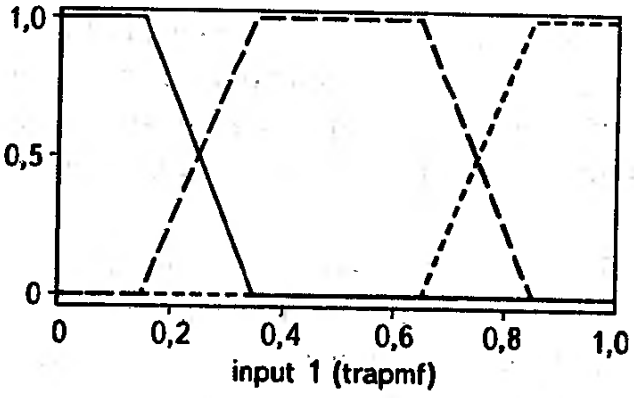
3) `mf_type` – матрица строк, задающих типы функций принадлежности входных переменных. Если аргумент задан одной строкой, тогда все функции принадлежности будут иметь один и тот же тип. Значение по умолчанию: 'gbellmf' – обобщенная колоколообразная функция принадлежности. Не поддерживаются функции принадлежности 'sigmf', 'smf' и 'zmf'.

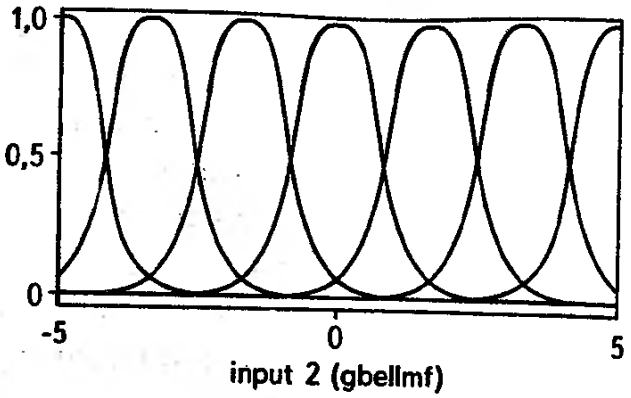
Функция genparam возвращает матрицу `mf_param`, каждая строчка которой содержит параметры одной функции принадлежности. Первая строка матрицы соответствует параметрам первой функции принадлежности первой входной переменной, вторая – второй функции принадлежности первой входной переменной и т.д. и последняя строка – последней функции принадлежности последней входной переменной.

Пример

```
NumData = 1000;
data = [rand(NumData, 1)...
10*rand(NumData, 1)-5 rand(NumData, 1)];
NumMf = [3 7]; MfType = str2mat('trapmf', 'gbellmf');
MfParams = genparam(data, NumMf, MfType);
NumInput = size(data, 2)-1;
range = [min(data)' max(data)'];
FirstIndex = [0 cumsum(NumMf)];
for i = 1:NumInput;
  subplot(1, NumInput, i);
  x = linspace(range(i, 1), range(i, 2), 100);
  index = FirstIndex(i) + 1:FirstIndex(i) + NumMf(i);
  mf = evalmmf(x, MfParams(index, :), MfType(i, :));
  plot(x, mf', 'linewidth', 1);
  xlabel(['input 'num2str(i)' ('MfType(i, :)')']);
end
```

Генерирование трех треугольных и семи колоколообразных функций принадлежностей из 1000 пар случайно полученных данных «входы – выход».





gensurf

Назначение Генерирование поверхности «входы – выход», соответствующей FIS.

Синтаксис

```
gensurf(fis)
gensurf(fis, inputs, output)
gensurf(fis, inputs, output, grids)
gensurf(fis, inputs, output, grids, refinput)
[x y z] = gensurf(...)
```

Описание Выводит поверхность «входы – выход», соответствующую системе нечеткого вывода. Функция gensurf может иметь до пяти входных аргументов:

1) `fis` – система нечеткого логического вывода;

2) `inputs` – вектор порядковых номеров входных переменных нечеткой системы, которым ставятся в соответствие оси абсцисс и ординат. Значения по умолчанию – [1 2]. Если задан номер только одной входной переменной, тогда строится график однофакторной зависимости «вход – выход»;

3) `output` – порядковый номер выходной переменной нечеткой системы, выводимой по оси аппликат. По умолчанию выбирается выходная переменная с первым порядковым номером;

4) `grids` – количество дискрет по осям абсцисс и ординат для построения поверхности. Если количество точек одинаково по обеим осям, тогда достаточно указать только одно значение. Значение по умолчанию – 15;

5) `refinput` – вектор значений входных переменных, неассоциированных с координатными осями. Аргумент необходим, если количество входных переменных больше двух. Длина вектора равна количеству входных переменных. Координаты вектора, соответствующие переменным, для которых строится поверхность, должны иметь значения `NaN`. Остальные координаты вектора `refinput` фиксируют значения входных переменных, неассоциированных с координатными осями. По умолчанию значения этих координат вектора

refinput равны серединам диапазонов изменения переменных.

Функция gensurf может иметь три выходных аргумента x, y, z, которые задают координаты поверхности «входы – выход» по осям абсцисс (x), ординат (y) и аппликат (z).

Пример

```
a = readfis('tipper')
gensurf(a)
```

Выводится поверхность «входы – выход» для нечеткой демо-системы tipper, моделирующей зависимость размера чаевых от качества блюд и уровня сервиса в ресторане.

getfis

Назначение Получение свойств FIS.

Синтаксис

```
out = getfis(fis)
out = getfis(fis, fisprop)
out = getfis(fis, vartype, varindex, varprop)
out = getfis(fis, vartype, varindex, 'mf',...
mfindex, mfprop)
```

Описание Выдает информацию о нечеткой системе. Наименования свойств можно задать в любом регистре. Функция getfis может иметь один, два, четыре или шесть входных аргументов:

1) fis – система нечеткого вывода;

2) fisprop – наименование свойства нечеткой системы. Допустимые значения:

'AggMethod' – реализация операции агрегирования;

'AndMethod' – реализация логической операции И;

'DefuzzMethod' – метод дефаззификации;

'ImpMethod' – реализация импликации;

'InLabels' – наименования входных переменных;

'InMfLabels' – термы выходных переменных;

'InMfParams' – параметры функций принадлежности входных переменных;

'InMfs' – количество функций принадлежности входных переменных;

'InMfTypes' – типы функций принадлежности входных переменных;

'InRange' – диапазоны изменения входных переменных;

'Name' – наименование системы нечеткого вывода;

'NumInputMfs' – количество функций принадлежности входных переменных;

'NumInputs' – количество входных переменных;

'NumOutMfs' – количество функций принадлежности выходных переменных;

'NumOutputs' – количество выходных переменных;

'NumRules' – количество правил в базе знаний;
'OrMethod' – реализация логической операции ИЛИ;
'OutLabels' – наименования входных переменных;
'OutMfLabels' – термы выходных переменных;
'OutMfParams' – параметры функций принадлежности выходных переменных;
'OutMfs' – количество функций принадлежности выходных переменных;
'OutMfTypes' – типы функций принадлежности выходных переменных;
'OutRange' – диапазоны изменения выходных переменных;
'RuleList' – база знаний в индексном формате;
'Type' – тип системы нечеткого вывода;

3) `vartype` – тип переменной. Допустимые значения: 'input' – входная переменная; 'output' – выходная переменная;

4) `varindex` – порядковый номер переменной;

5) `varprop` – наименование свойства переменной. Допустимые значения:
'Name' – наименование переменной;
'Range' – диапазон изменения переменной;
'NumMfs' – количество функций принадлежности;
'MfLabels' – список термов, используемых для лингвистической оценки переменной;

6) `mfindex` – порядковый номер функции принадлежности переменной;

7) `mfprop` – наименование свойства функции принадлежности. Допустимые значения:
'Name' – наименование функции принадлежности (терм);
'Type' – тип функции принадлежности;
'Params' – параметры функции принадлежности.

Пример

```
a = readfis('tipper')
out = getfis(a)
```

Выводится список основных свойств демо-системы нечеткого вывода «Tipper».

initfis

Назначение Генерирование исходной матрицы нечеткого разбиения.

Синтаксис

```
U = initfcm(cluster_n, data_n)
```

Описание Генерирует матрицу степеней принадлежности для начала итерационной процедуры поиска центров кластеров по алгоритму нечетких *c*-средних. Функция имеет два входных аргумента:

1) `cluster_n` – количество кластеров;
2) `data_n` – количество объектов кластеризации.

Функция initfcm возвращает матрицу нечеткого разбиения U размером `cluster_n` × `data_n`. Сумма по столбцам элементов этой матрицы, т.е. сумма степеней принадлежности объекта ко всем кластерам, равна единице.

Пример

```
%Генерирование матрицы степеней принадлежности
%десяти объектов к трем кластерам:
initfcm(3, 10)
```

isfis

Назначение Проверка структуры данных системы нечеткого вывода.

Синтаксис

```
tf = isfis(fis)
```

Описание Проверяет, является ли структура `fis` системой нечеткого вывода. Точнее говоря, проверяется, содержит ли `fis` следующие поля структуры нечеткого вывода: 'name', 'type', 'andMethod', 'orMethod', 'defuzzMethod', 'impMethod', 'aggMethod' и 'rule'. Если `fis` содержит все указанные поля, тогда считается, что она является системой нечеткого вывода, и функция isfis возвращает 1. В противном случае возвращается 0.

mam2sug

Назначение Преобразование FIS типа Мамдани в FIS типа Сугено.

Синтаксис

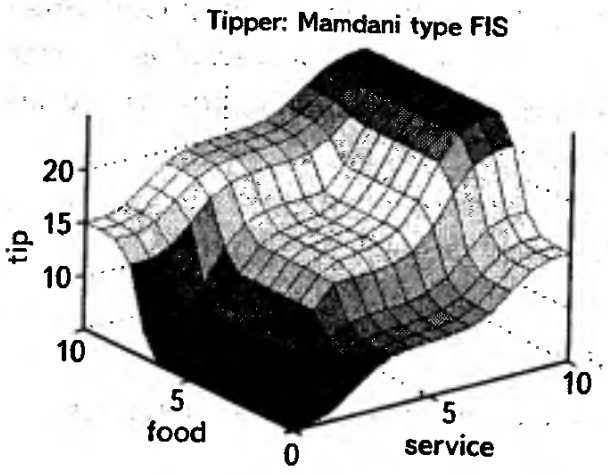
```
sug_fis = mam2fis(mam_fis)
```

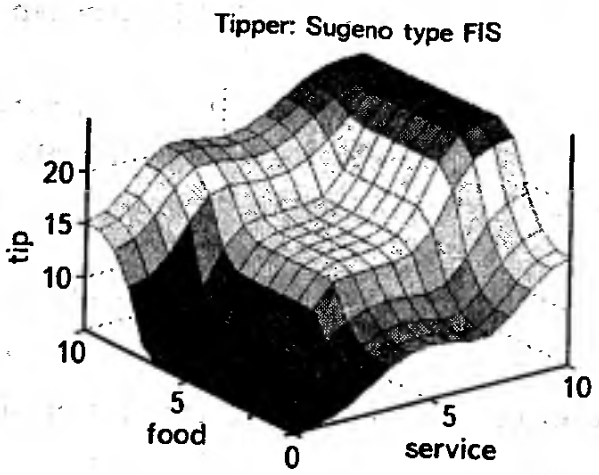
Описание Преобразовывает систему нечеткого вывода Мамдани (`mam_fis`) в систему Сугено (`sug_fis`). Нечеткая система может иметь несколько входов и выходов. В результате выполнения функции mam2sug получается система типа Сугено нулевого порядка с тем же количество правил, что и в исходной системе. Посылки правил не меняются. В качестве заключений правил используются числа, соответствующие максимуму функций принадлежности выходных переменных в исходной системе.

Пример

```
mam_fis = readfis('tipper');
sug_fis = mam2sug(mam_fis);
subplot(1, 2, 1)
gensurf(mam_fis); title('Tipper: Mamdani type FIS');
subplot(1, 2, 2)
gensurf(sug_fis); title('Tipper: Sugeno type FIS')
```

Загружается в рабочую область нечеткая демо-система «Tipper», задающая зависимость размера чаевых от качества пищи и уровня сервиса в ресторане. С помощью функции mam2sug преобразовывается исходная система типа Мамдани в систему типа Сугено. На рисунке показаны поверхности «входы – выход» этих систем.





mf2mf

азначение Пересчет параметров встроенных функций принадлежности различных типов.

интаксис `outParams = mf2mf(inParams, inType, outType)`

писание Пересчитывает параметры одного встроенного типа функции принадлежности в параметры другого типа. Функция mf2mf имеет три входных аргумента:

1) `inParams` – вектор параметров исходной функции принадлежности;

2) `inType` – тип исходной функции принадлежности;

3) `outType` – тип новой функции принадлежности.

Допустимые значения типов функций принадлежности: 'trimf'; 'trapmf'; 'pimf'; 'gaussmf'; 'gauss2mf'; 'gbellmf'; 'sigmf'; 'dsigmf'; 'smf'; 'psigmf'; 'zmf'. Описание этих типов функций принадлежности приведено в подразделе 3.4.16.

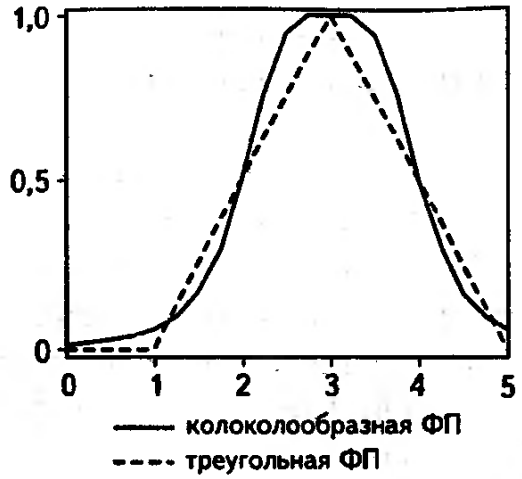
Функция возвращает вектор параметров новой функции принадлежности.

Многократное использование функции mf2mf может привести к некоторому изменению формы функции принадлежности. Например, если полученные параметры функции принадлежности обратно пересчитать в параметры функции принадлежности исходного типа, то иногда конечные параметры не будут равны исходным.

имер

```
x = 0: 0.25: 5;
mfp1 = [1 2 3];
mfp2 = mf2mf(mfp1, 'gbellmf', 'trimf');
plot(x, gbellmf(x, mfp1), '-o', x, trimf(x, mfp2), ':+')
legend('Колоколообразная ФП', 'Треугольная ФП')
```

Преобразование обобщенной колоколообразной функции принадлежности в треугольную.



newfis

Назначение Создание новой FIS.

Синтаксис

```
fis = newfis(name, type, andMethod, orMethod,...
impMethod, aggMethod, defuzzMethod)
```

Описание Создает в рабочей области новую систему нечеткого вывода. Функция newfis может иметь до семи входных аргументов:

1) `name` – наименование системы нечеткого вывода;

2) `type` – тип системы нечеткого вывода. Допустимые значения: 'mamdani' – система типа Мамдани (значение по умолчанию); 'Sugeno' – система типа Сугено;

3) `andMethod` – реализация логической операции И. Значения по умолчанию: минимум ('min') – для системы типа Мамдани; произведение ('prod') – для системы типа Сугено;

4) `orMethod` – реализация логической операции ИЛИ. Значения по умолчанию: максимум ('max') – для системы типа Мамдани; вероятностное ИЛИ ('probor') – для системы типа Сугено;

5) `impMethod` – реализация импликации. Значение по умолчанию: 'min' – минимум;

6) `aggMethod` – реализация агрегирования. Значение по умолчанию: 'max' – максимум;

7) `defuzzMethod` – метод дефаззификации. Значения по умолчанию: центр тяжести ('centroid') – для системы типа Мамдани; взвешенное среднее ('wtaver') – для системы типа Сугено.

Пример

```
a = newfis('new_fuzzy_system')
```

В рабочей области создается структура a, содержащая нечеткую систему с именем `'new_fuzzy_system'`. Значения всех параметров системы устанавливаются по умолчанию.

parsrule

значение Вставка в FIS правил, заданных предложениями на естественном языке.

нтаксис

```
outfis = parsrule(infis, inrulelist, ruleformat, lang)
[outfis, outrulelist, errorstr] =...
parsrule(infis, inrulelist, ruleformat, lang)
```

исание Вводит правила в нечеткую базу знаний. При этом удаляются все ранее существующие в базе знаний правила. Функция parsrule может иметь до четырех входных аргументов, первые два из которых обязательные:

1) `infis` – исходная система нечеткого вывода;

2) `inrulelist` – матрица, каждая строчка которой определяет одно нечеткое правило <Если – То>. Правила можно задавать предложениями на английском, немецком и французских языках, а также в символьном и индексном форматах. При задании правил на естественном языке необходимо использовать следующие ключевые слова:

для английского языка: «if», »and», «or», «then», «is», «not»;

для французского языка: «si», «et», «ou», «alors», «est», «n'est_pas»;

для немецкого языка: «senn», «und», «oder», «dann», «ist», «nicht», которые эквиваленты русским «если», «и», «или», «то», «есть» и «не», соответственно. Весовой коэффициент указывается в конце правила. По умолчанию он равен 1. Нечеткая база знаний задается;

3) `ruleformat` – формат правил. Допустимые значения: 'verbose' – словесный; 'symbolic' – символьный; 'indexed' – индексный. Значение по умолчанию – 'verbose'. Примеры нечетких правил в различных форматах приведены в подразделе 3.3.3. В формате 'verbose' нельзя использовать терм 'none' в качестве значений переменной. Для задания «коротких» правил необходимо из правила исключить наименования соответствующих переменных;

4) `lang` – язык представления правил в формате 'verbose'. Допустимые значения: 'english' – английский; 'francais' – французский; 'deutsch' – немецкий. По умолчанию установлен английский язык.

Функция parsrule может иметь до трех выходных аргументов:

1) `outfis` – система нечеткого вывода с новыми правилами;

2) `outrulelist` – список правил системы `outfis`. Список представляет собой матрицу целых положительных чисел, соответствующих правилам `inrulelist`. Для преобразования чисел в символы необходимо использовать функцию

char. Вошедшие в `outfis` корректно заданные правила в этом списке имеют порядковый номер. Некорректные правила обозначены символом '#'.

3) `errorstr` – список ошибок задания правил.

Пример

```
infis = readfis('tipper');
r1 = 'If service is good then tip is average';
r2 = 'If service is poor and food is rancid then
%tip is cheap';
r3 = 'If service is excellent and food is delicious
%then tip is generous';
inrulelist = [r1; r2; r3];
outfis = parsrule(infis, inrulelist)
```

Загружается в рабочую область нечеткая демо-система «Tipper». Затем формируется новая база знаний со следующими правилами:

«If service is good then tip is average»;

«If service is poor and food is rancid then tip is cheap»;

«If service is excellent and food is delicious then tip is generous'».

pimf

Назначение Пи-подобная функция принадлежности.

Синтаксис

```
y = pimf(x, params)
```

Описание Задает функцию принадлежности в виде криволинейной трапеции. Эта функция задается как произведение *s*- и *z*-подобных функций принадлежности:

```
pimf(x, [a, b, c, d]) = smf(x, [a, b]). *zmf(x, [c, d])
```

Если `b > c`, то параметры функции принадлежности интерпретируются так:

`[a, d]` – носитель нечеткого множества;

`[b, c]` – ядро нечеткого множества.

При $b > c$ нечеткое множество получается субнормальным.

Функция pimf применяется для задания асимметричных функций принадлежности с плавным переходом от пессимистической к оптимистической оценке нечеткого числа. Функция pimf имеет два входных аргумента:

1) `x` – вектор, для координат которого рассчитываются степени принадлежности;

2) `params` – вектор параметров функции принадлежности. Порядок задания параметров `[a b c d]`.

Функция pimf возвращает степени принадлежности координат вектора `x`.

Пример

```
x = 0: 0.5: 10;
y1 = pimf(x, [0 0.5 3 9]);
y2 = pimf(x, [0 4 5.5 9]);
```

parsrule

Назначение Вставка в FIS правил, заданных предложениями на естественном языке.

Синтаксис

```
outfis = parsrule(infis, inrulelist, ruleformat, lang)
[outfis, outrulelist, errorstr] =...
parsrule(infis, inrulelist, ruleformat, lang)
```

Описание Вводит правила в нечеткую базу знаний. При этом удаляются все ранее существующие в базе знаний правила. Функция parsrule может иметь до четырех входных аргументов, первые два из которых обязательные:

1) `infis` – исходная система нечеткого вывода;

2) `inrulelist` – матрица, каждая строчка которой определяет одно нечеткое правило <Если – То>. Правила можно задавать предложениями на английском, немецком и французских языках, а также в символьном и индексном форматах. При задании правил на естественном языке необходимо использовать следующие ключевые слова:

для английского языка: «if», »and», «or», «then», «is», «not»;

для французского языка: «si», «et», «ou», «alors», «est», «n'est_pas»;

для немецкого языка: «senn», «und», «oder», «dann», «ist», «nicht», которые эквиваленты русским «если», «и», «или», «то», «есть» и «не», соответственно. Весовой коэффициент указывается в конце правила. По умолчанию он равен 1. Нечеткая база знаний задается;

3) `ruleformat` – формат правил. Допустимые значения: 'verbose' – словесный; 'symbolic' – символьный; 'indexed' – индексный. Значение по умолчанию – 'verbose'. Примеры нечетких правил в различных форматах приведены в подразделе 3.3.3. В формате 'verbose' нельзя использовать терм 'none' в качестве значений переменной. Для задания «коротких» правил необходимо из правила исключить наименования соответствующих переменных;

4) `lang` – язык представления правил в формате 'verbose'. Допустимые значения: 'english' – английский; 'francais' – французский; 'deutsch' – немецкий. По умолчанию установлен английский язык.

Функция parsrule может иметь до трех выходных аргументов:

1) `outfis` – система нечеткого вывода с новыми правилами;

2) `outrulelist` – список правил системы `outfis`. Список представляет собой матрицу целых положительных чисел, соответствующих правилам `inrulelist`. Для преобразования чисел в символы необходимо использовать функцию

char. Вошедшие в `outfis` корректно заданные правила в этом списке имеют порядковый номер. Некорректные правила обозначены символом '#'.

3) `errorstr` – список ошибок задания правил.

Пример

```
infis = readfis('tipper');
r1 = 'If service is good then tip is average';
r2 = 'If service is poor and food is rancid then
%tip is cheap';
r3 = 'If service is excellent and food is delicious
%then tip is generous';
inrulelist = [r1; r2; r3];
outfis = parsrule(infis, inrulelist)
```

Загружается в рабочую область нечеткая демо-система «Tipper». Затем формируется новая база знаний со следующими правилами:

«If service is good then tip is average»;

«If service is poor and food is rancid then tip is cheap»;

«If service is excellent and food is delicious then tip is generous'».

pimf

Назначение Пи-подобная функция принадлежности.

Синтаксис

```
y = pimf(x, params)
```

Описание Задает функцию принадлежности в виде криволинейной трапеции. Эта функция задается как произведение *s*- и *z*-подобных функций принадлежности:

`pimf(x, [a, b, c, d]) = smf(x, [a, b]). *zmf(x, [c, d])`

Если `b > c`, то параметры функции принадлежности интерпретируются так:

`[a, d]` – носитель нечеткого множества;

`[b, c]` – ядро нечеткого множества.

При $b > c$ нечеткое множество получается субнормальным.

Функция pimf применяется для задания асимметричных функций принадлежности с плавным переходом от пессимистической к оптимистической оценке нечеткого числа. Функция pimf имеет два входных аргумента:

1) `x` – вектор, для координат которого рассчитываются степени принадлежности;

2) `params` – вектор параметров функции принадлежности. Порядок задания параметров `[a b c d]`.

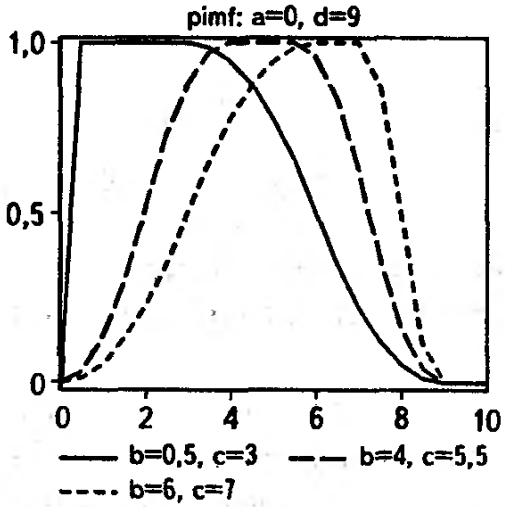
Функция pimf возвращает степени принадлежности координат вектора `x`.

Пример

```
x = 0: 0.5: 10;
y1 = pimf(x, [0 0.5 3 9]);
y2 = pimf(x, [0 4 5.5 9]);
```

```
y3 = pimf(x, [0 6 7 9]);
plot(x, y1, 'o-', x, y2, '+:', x, y3, '.-')
title('pimf, a=0, d=9')
legend('b=0.5, c=3', 'b=4, c=5,5', 'b=6, c=7')
```

Построение графика пи-подобной функции принадлежности для нечетких множеств с ядрами [0,5, 3], [4, 5,5] и [6, 7].



plotfis

Назначение Вывод основных параметров FIS в виде графической схемы.

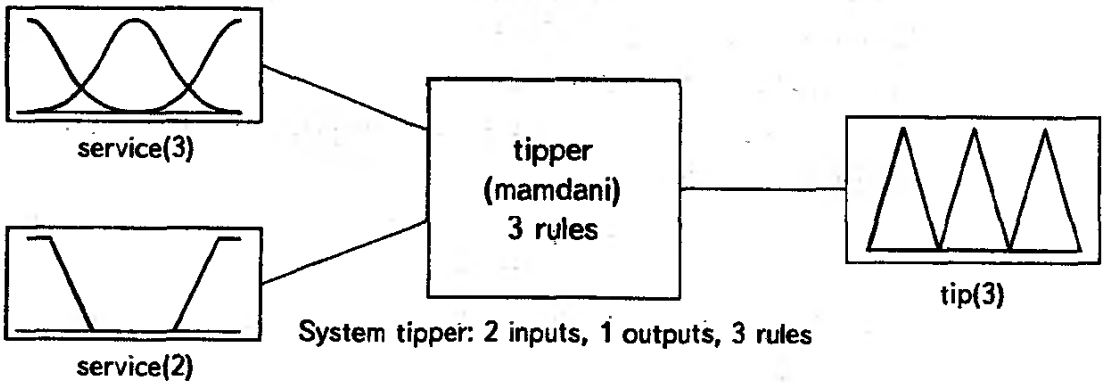
Синтаксис `plotfis(fis)`

Описание Выводит графическую схему системы нечеткого вывода `fis`. Входы системы изображаются в левой части окна, выходы – в правой части, в центре – база знаний. В окне выводятся наименование и тип нечеткой системы, количество термов и количество правил. Также изображаются графики функций принадлежности всех нечетких термов.

Пример

```
a = readfis('tipper');
plotfis(a)
```

Вывод схемы нечеткой демо-системы «Tipper».



plotmf

Назначение Вывод графиков функций принадлежности термов одной переменной.

Синтаксис
```
plotmf(fis, varType, varIndex, numPts)
[x, y] = plotmf(fis, varType, varIndex, numPts)
```

Описание Выводит графики функций принадлежности термов одной переменной нечеткой системы. Графики функций принадлежности выделяются разными цветами. Функция plotmf может иметь три или четыре входных аргумента:

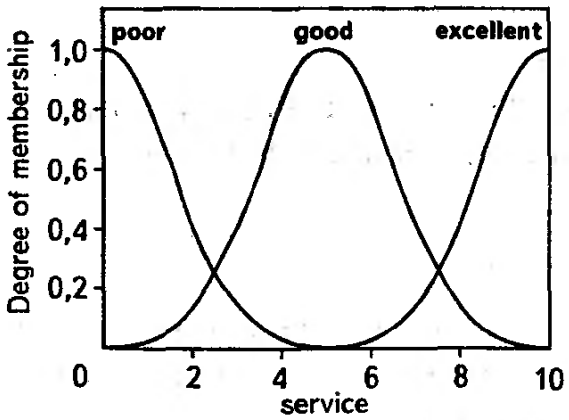
1) `fis` – система нечеткого вывода;
2) `varType` – тип переменной. Допустимые значения: 'input' – входная переменная; 'output' – выходная переменная;
3) `varIndex` – порядковый номер переменной. Входные и выходные переменные нумеруются независимо;
4) `numPts` – число дискрет, по которым строятся графики функций принадлежности. Значение по умолчанию – 181.

Функция plotmf может иметь два выходных аргумента:

1) `x` – матрица абцисс-координат для всех графиков функций принадлежности;
2) `y` – матрица значений функций принадлежности, соответствующих `x`.

При вызове функции plotmf с выходными аргументами графики функций принадлежности не выводятся.

Пример
```
a = readfis('tipper');
plotmf(a, 'input', 1)
```
Вывод графиков функций принадлежности термов первой входной переменной демо-системы нечеткого вывода «Tipper».



probor

Назначение Вероятностная реализация логической операции ИЛИ.

Синтаксис
```
y = probor(x)
```

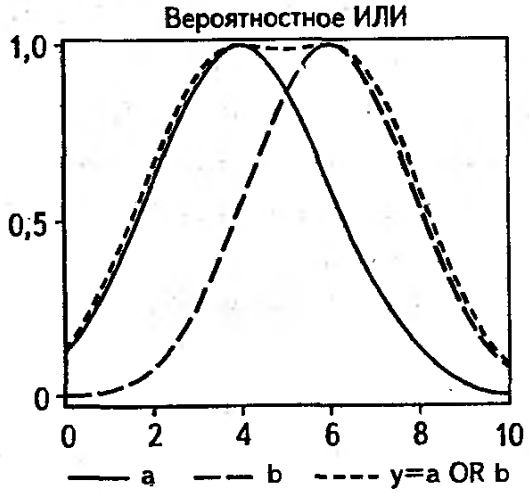
Описание Вероятностная реализация логической операции ИЛИ над числами a и b задается формулой $y(a, b) = a + b - ab$.

Входной аргумент x задает матрицу степеней принадлежности. Логическая операция ИЛИ выполняется над степенями принадлежности, записанными в одном столбце матрицы x. Функция probor возвращает вектор y с результатами логической операции ИЛИ.

Пример

```
x = 0: 0.5: 10;
a = gaussmf(x, [2 4]); b = gaussmf(x, [1.8 6]);
y = probor([a; b]);
plot(x, a, 'o:', x, b, 's:' , x, y, '.-')
legend('a', 'b', 'y = a OR b');
title('Вероятностное ИЛИ')
```

Выполнение операции «вероятностное ИЛИ» над двумя нечеткими множествами с гауссовыми функциями принадлежности.



psigmf

Назначение Функция принадлежности в виде произведения двух сигмоидных функций.

Синтаксис

```
y = psigmf(x, params)
```

Описание Задает функцию принадлежности в виде следующего произведения двух сигмоидных функций:

$$\mu(x)=\frac{1}{1+e^{-a_1(x-c_1)}}\frac{1}{1+e^{-a_2(x-c_2)}}.$$

Функция psigmf применяется для задания гладких асимметричных функций принадлежности. Функция psigmf имеет два входных аргумента:

1) x – вектор, для координат которого рассчитываются степени принадлежности;

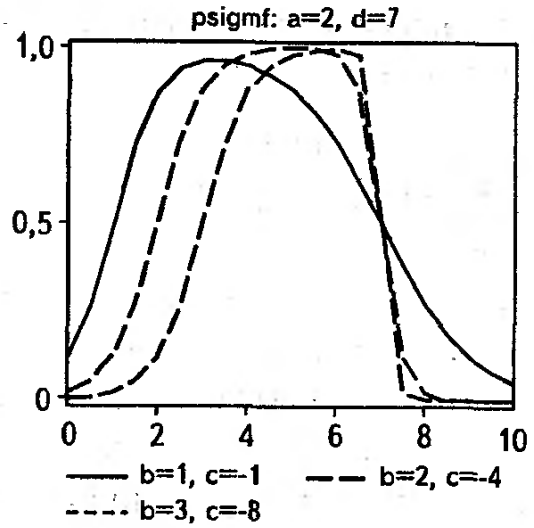
2) params – вектор параметров функции принадлежности. Порядок задания параметров: [a1 c1 a2 c2].

Функция psigmf возвращает степени принадлежности координат вектора x.

Пример

```
x = 0: 0.5: 10;
y1 = psigmf(x, [2 1 -1 7]);
y2 = psigmf(x, [2 2 -4 7]);
y3 = psigmf(x, [2 3 -8 7]);
plot(x, y1, 'o-', x, y2, '+:', x, y3, '.-');
title('psigmf, a=2, d=7');
legend('b=1, c=-1', 'b=2, c=-4', 'b=3, c=-8')
```

Построение графиков функций принадлежности в виде произведения двух сигмоидных функций с параметрами `[2 1 -1 7]`, `[2 2 -4 7]` и `[2 3 -8 7]`.



readfis

Назначение Загрузка FIS из файла.

Синтаксис

```
fis = readfis
fis = readfis('fisfile')
```

Описание Загружает в рабочую область систему нечеткого вывода из файла `fisfile.fis`. Выходным аргументом функции readfis является структура системы нечеткого вывода `fis`. При вызове функции readfis без входного аргумента появляется типовое окно открытия файла.

Пример

```
a = readfis('tipper')
```

Загрузка в рабочую область нечеткой демо-системы «Tipper».

rmmf

Назначение Удаление нечеткого терма из FIS.

Синтаксис

```
fis = rmmf(fis, varType, varIndex, 'mf', mfIndex)
```

Описание Удаляет терм из нечеткой системы. Функция rmmf имеет пять входных аргументов:

1) `fis` – система нечеткого вывода;

2) `varType` – тип переменной. Допустимые значения: 'input' – входная переменная; 'output' – выходная переменная;

3) varIndex – порядковый номер переменной. Входные и выходные переменные нумеруются независимо;

4) 'mf' – константа;

5) mfIndex – порядковый номер удаляемого терма, используемого с переменной с порядковым номером varIndex.

При попытке удаления терма, задействованного в базе знаний, запрашивается подтверждение действия.

Пример

```
a = readfis('tipper');
b = rmmf(a, 'input', 1, 'mf', 3)
```

Удаление терма «excellent» из терм-множества, используемого для лингвистической оценки входной переменной «service» в нечеткой демо-системе «Tipper».

rmvar

Назначение Удаление переменной из FIS.

Синтаксис

```
fis2 = rmvar(fis, varType, varIndex)
[fis2, errorStr] = rmvar(fis, varType, varIndex)
```

Описание Удаляет переменную из системы нечеткого вывода. Функция rmmf вызывается с тремя входными аргументами:

1) fis – система нечеткого вывода;

2) varType – тип переменной. Допустимые значения: 'input' – входная переменная; 'output' – выходная переменная;

3) varIndex – порядковый номер удаляемой переменной. Входные и выходные переменные нумеруются независимо.

Функция rmmf автоматически модифицирует базу знаний таким образом, чтобы она соответствовала количеству входных и выходных переменных. Перед удалением переменной необходимо из базы знаний исключить все правила, в которых она фигурирует. В противном случае появится запрос на подтверждение удаления.

Функция rmmf может иметь два выходных аргумента:

1) fis2 – нечеткая система без переменной с порядковым номером varIndex;

2) errorStr – строка ошибок при работе функции rmmf.

Пример

```
a = readfis('tipper');
b = rmvar(a, 'input', 1)
```

Удаление входной переменной «service» из нечеткой демо-системы «Tipper».

setfis

Назначение Назначение свойств FIS.

Синтаксис

```
fis2 = setfis(fis, vartype, varindex, varprop,...
propvalue)
fis2 = setfis(fis, vartype, varindex, 'mf',...
mfindex, mfprop, propvalue)
```

Описание

Устанавливает новые свойства системы нечеткого вывода. Функция setfis может иметь три, пять или семь входных аргументов:

1) `fis` – исходная система нечеткого вывода;

2) `fisprop` – наименование свойства нечеткой системы, значение которого будет изменено. Допустимые наименования:

'AggMethod' – реализация агрегирования;

'AndMethod' – реализация логической операции И;

'DefuzzMethod' – метод дефаззификации;

'ImpMethod' – реализация импликации;

'InLabels' – наименования входных переменных;

'InMfParams' – параметры функций принадлежности входных переменных;

'Name' – наименование системы нечеткого логического вывода;

'OrMethod' – реализация логической операции ИЛИ;

'OutLabels' – наименования входных переменных;

'OutMfParams' – параметры функций принадлежности выходных переменных;

'Type' – тип системы нечеткого вывода (Мамдани или Сугено);

'RuleList' – список правил нечеткой базы знаний;

3) `propvalue` – новое значение свойства нечеткой системы;

4) `vartype` – тип переменной, свойство которой будет изменено. Допустимые значения: 'input' – входная переменная; 'output' – выходная переменная;

5) `varindex` – порядковый номер переменной, свойство которой будет изменено;

6) `varprop` – наименование изменяемого свойства переменной. Допустимые значения:

'Name' – наименование переменной;

'Range' – диапазон изменения переменной;

'NumMfs' – количество термов;

7) `'mf'` – константа, указывающая, что будут изменены нечеткие термы;

8) `mfindex` – порядковый номер терма в терм-множестве, используемого для лингвистической оценки переменной;

9) `mfprop` – наименование изменяемого свойства нечеткого терма. Допустимые наименования:

'Name' – наименование функции принадлежности (терм);

'Type' – тип функции принадлежности;

'Params' – параметры функции принадлежности.

Наименования свойств можно задавать в любом регистре.

Функция setfis возвращает систему нечеткого вывода `fis2` с новым значением соответствующего свойства.

Пример

```
fis = readfis('tipper');
fis = setfis(fis, 'DefuzzMethod', 'mom')
```

Установка метода дефаззификации «средний из максимумов» (`'mom'`) в нечеткой демо-системе «Tipper».

sffis

Назначение Оптимизированная под Simulink функция нечеткого вывода.

Синтаксис

```
output = sffis(t, x, u, flag, fis)
```

Описание Это mex-файл, специально оптимизированный для использования в пакете Simulink. Функция sffis выполняет нечеткий вывод аналогично функции evalfis. Функция sffis имеет пять входных аргументов:

- `t`, `x`, и `flag` – стандартные аргументы *s*-функций пакета Simulink;
- `u` – вектор значений входных переменных, для которых осуществляется нечеткий вывод;
- `fis` – система нечеткого вывода.

Функция возвращает результат нечеткого вывода.

showfis

Назначение Вывод на экран свойств FIS.

Синтаксис

```
showfis(fis)
```

Описание Выводит на экран рабочей области значения следующих свойств нечеткой системы `fis`:

- Name – наименование системы нечеткого вывода;
- Type – тип системы нечеткого вывода;
- Inputs/Outputs – количество входных и выходных переменных;
- NumInputMFs – мощности терм-множеств входных переменных;
- NumOutputMFs – мощности терм-множеств выходных переменных;
- NumRules – количество правил в нечеткой базе знаний;
- AndMethod – реализация логической операции И;
- OrMethod – реализация логической операции ИЛИ;
- ImpMethod – реализация импликации;
- AggMethod – реализация агрегирования;
- DefuzzMethod – метод дефаззификации;
- InLabels – наименования входных переменных;
- OutLabels – наименования выходных переменных;
- InRange – диапазоны изменения входных переменных;
- OutRange – диапазоны изменения выходных переменных;
- InMFLabels – терм-множества входных переменных;

OutMFLabels – терм-множества выходных переменных;

InMFTypes – типы функций принадлежности термов входных переменных;

OutMFTypes – типы функций принадлежности термов выходных переменных;

InMFParams – параметры функций принадлежности термов входных переменных;

OutMFParams – параметры функций принадлежности термов выходных переменных;

Rule Antecedent – посылки правил;

Rule Consequent – заключения правил;

Rule Weigth – весовые коэффициенты правил;

Rule Connection – логические связки переменных внутри правил.

Пример

```
fis = readfis('tipper');
showfis(fis)
```

Вывод на экран значений свойств нечеткой демо-системы «Tipper».

1. Name	tipper	23. OutMFLabels	cheap
2. Type	mamdani	24.	average
3. Inputs/Outputs	[2 1]	25.	generous
4. NumInputMFs	[3 2]	26. InMFTypes	gaussmf
5. NumOutputMFs	3	27.	gaussmf
6. NumRules	3	28.	gaussmf
7. AndMethod	min	29.	trapmf
8. OrMethod	max	30.	trapmf
9. ImpMethod	min	31. OutMFTypes	trimf
10. AggMethod	max	32.	trimf
11. DefuzzMethod	centroid	33.	trimf
12. InLabels	service	34. InMFParams	[1.5 0 0 0]
13.	food	35.	[1.5 5 0 0]
14. OutLabels	tip	36.	[1.5 10 0 0]
15. InRange	[0 10]	37.	[0 0 1 3]
16.	[0 10]	38.	[7 9 10 10]
17. OutRange	[0 30]	39. OutMFParams	[0 5 10 0]
18. InMFLabels	poor	40.	[10 15 20 0]
19.	good	41.	[20 25 30 0]
20.	excellent	42. Rule Antecedent	[1 1]
21.	rancid	43.	[2 0]
22.	delicious	44.	[3 2]

showrule

Назначение Вывод базы знаний FIS.

Синтаксис
```
outStr = showrule(fis, ruleIndex, ruleFormat, lang)
```

Описание Функция возвращает переменную `outStr`, содержащую список правил базы знаний системы нечеткого вывода `fis`. Если выходной аргумент функции не задан, то список правил выводится на экран рабочей области. Функция showrule может иметь до четырех входных аргументов:

1) `fis` – система нечеткого вывода;
2) `ruleIndex` – порядковые номера выводимых правил. Задается в виде вектора натуральных чисел. По умолчанию выводятся все правила базы знаний;
3) `ruleFormat` – формат вывода правил. Допустимые значения: 'verbose' – словесный (установлен по умолчанию); 'symbolic' – символьный; 'indexed' – индексный;
4) `lang` – язык представления правил в формате 'verbose'. Допустимые значения: 'english' – английский (установлен по умолчанию); 'francais' – французский; 'deutsch' – немецкий.

Пример
```
%Вывод на экран базы знаний нечеткой демо-системы
%«Tipper», которая содержит такие правила:
%1) если обслуживание = плохое или пища =
%пригоревшая, то чаевые = малые;
%2) если обслуживание = хорошее, то чаевые = средние;
%3) если обслуживание = отличное и пища = вкусная,
%то чаевые = щедрые;
fis = readfis('tipper');
showrule(fis)
```

sigmf

Назначение Сигмоидная функция принадлежности.

Синтаксис
```
y = sigmf(x, params)
```

Описание Задает сигмоидную функцию принадлежности по формуле:

$$\mu(x)=\frac{1}{1+e^{-a(x-c)}}.$$

Параметры функции принадлежности геометрически интерпретируются так:

a – коэффициент крутизны функции принадлежности;

c – координата перегиба функции принадлежности.

Функция sigmf применяется для задания монотонных функций принадлежности. Функция sigmf имеет два входных аргумента:

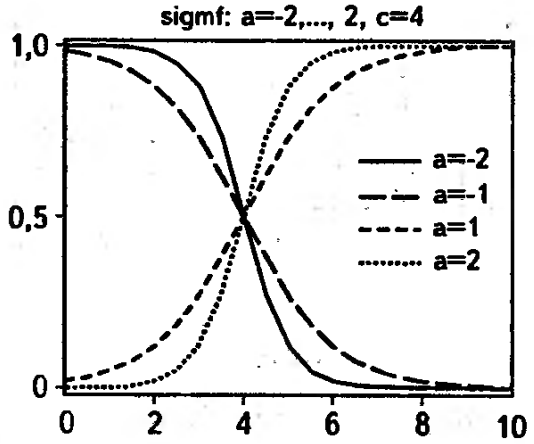
1) `x` – вектор, для координат которого рассчитываются степени принадлежности;
2) `params` – вектор параметров функции принадлежности. Порядок задания параметров: `[a c]`.

Функция sigmf возвращает степени принадлежности координат вектора x.

Пример

```
x = 0: 0.5: 10;
y1 = sigmf(x, [-2 4]);
y2 = sigmf(x, [-1 4]);
y3 = sigmf (x, [1 4]);
y4 = sigmf(x, [2 4]);
plot(x, y1, 'o-', x, y2, '+:', x, y3, '.-', x,...
y4, '--^')
title('sigmf: a=-2, ..., 2, c=4');
legend('a=-2', 'a=-1', 'a=1', 'a=2')
```

Построение графиков сигмоидных функций принадлежности с различными коэффициентами крутизны на интервале [0, 10].



smf

Назначение *s*-подобная функция принадлежности.

Синтаксис

```
y = smf(x, params)
```

Описание Задает *s*-подобную двухпараметрическую функцию принадлежности. Параметры функции принадлежности задают интервал, внутри которого функция нелинейно возрастает от 0 до 1. Функция smf применяется для задания неубывающих функций принадлежности с насыщением, представляющих нечеткие множества типа «очень высокий». Функция smf имеет два входных аргумента:

1) x – вектор, для координат которого рассчитываются степени принадлежности;

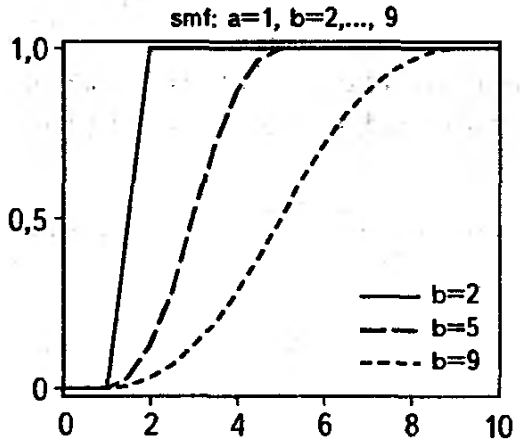
2) params – вектор параметров, которые задают интервал $[a, b]$ возрастания функции принадлежности. Если $b \le a$, то функция принадлежности получается в виде единичной ступеньки, проходящей через точку $(a + b) / 2$. Порядок задания параметров: $[a, b]$.

Функция smf возвращает степени принадлежности координат вектора x.

Пример

```
x = 0: 0.5: 10;
y1 = smf(x, [1 2]);
y2 = smf(x, [1 5]);
y3 = smf(x, [1 9]);
plot(x, y1, 'o-', x, y2, '+:', x, y3, '.-')
title('smf, a=1, b=2,..., 9');
legend('b=2', 'b=5', 'b=9')
```

Построение графиков s-подобных функций принадлежности с параметрами [1 2], [1 5] и [1 9].



subclust

Назначение Субтрактивная кластеризация.

Синтаксис `[centers, sigmas] = subclust(X, radii, xBounds,... options)`

Описание Вычисляет центры кластеров данных по улучшенному горному алгоритму субтрактивной кластеризации. Количество кластеров определяется во время работы алгоритма по распределению данных. Функция subclust может иметь до четырех входных аргументов, первые два из которых обязательны:

1) `X` – данные для кластерного анализа. Каждая строка матрицы `X` задает один объект кластеризации;

2) `radii` – вектор, определяющий размеры кластеров по каждой координате. Координаты вектора `radii` должны находиться в диапазоне [0, 1] в связи с тем, что при кластеризации данные `X` масштабируются на единичный гиперкуб. Как правило, малые значения `radii` приводят к тому, что функция subclust находит много мелких кластеров. Обычно хорошие результаты кластерного анализа получаются при значениях `radii` из диапазона [0,2, 0,5]. Если аргумент `radii` задан скаляром, тогда все координаты считаются равноважными;

3) `xBounds` – матрица диапазонов изменения входных данных, необходимая для масштабирования матрицы `X` на единичный гиперкуб. Каждая строка матрицы задает диапазон

изменения данных по одной координате. Если аргумент xBounds не задан, тогда диапазоны изменения входных данных рассчитываются по фактическим значениям матрицы X;

4) options – вектор параметров кластерного анализа:

options (1) – коэффициент подавления. Значение options(1)*radii используется для определения объектов, близких к центру кластера. Эти объекты считаются принадлежащими данному кластеру и исключаются из дальнейшего рассмотрения в кластерном анализе. Чем больше значение коэффициента подавления, тем больше соседних объектов будут принадлежать кластеру. Значение коэффициента подавления по умолчанию равно 1,25;

options(2) – коэффициент принятия. Используется как критерий назначения объекта центром кластера. Если отношение значений максимального потенциала текущего центра кластера к потенциалу центра первого кластера больше коэффициента принятия, тогда текущий объект рассматривается как потенциальный центр нового кластера. Чем больше значение коэффициента принятия, тем больше кластеров будет найдено. Значение коэффициента принятия по умолчанию равно 0,5;

options(3) – коэффициент отторжения. Используется как критерий исключения объекта из списка потенциальных центров кластеров. Если отношение значений максимального потенциала текущего центра кластера к потенциалу центра первого кластера меньше коэффициента принятия, тогда текущий объект дополнительно проверяется по коэффициенту отторжения. В случае, когда это отношение больше коэффициента отторжения и рассматриваемый объект расположен далеко от уже найденных центров кластеров, тогда он рассматривается как центр нового кластера. В противном случае, указанный объект исключается из дальнейшего рассмотрения как потенциальный центр кластера. Чем меньше значение коэффициента отторжения, тем больше отдаленных друг от друга кластеров может быть найдено. Значение коэффициента отторжения должно быть меньше значения коэффициента принятия. Значение коэффициента отторжения по умолчанию равно 0,15;

options(4) – управление выводом на экран промежуточных результатов кластерного анализа. Установленное по умолчанию нулевое значение этого параметра подавляет вывод промежуточных результатов.

Функция subclust может иметь до двух выходных аргументов:

1) centers – матрица центров найденных кластеров. Каждая строка матрицы представляет координаты центра одного кластера;

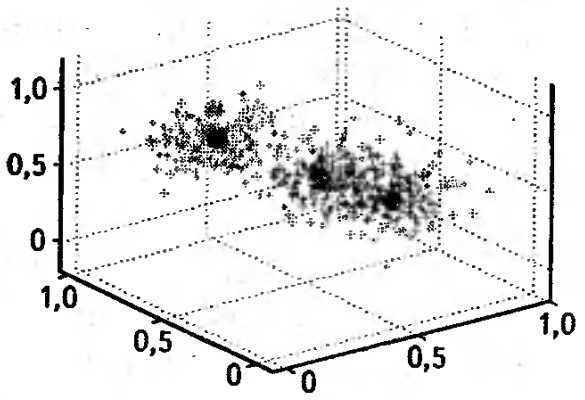
2) sigmas – вектор радиусов кластеров.

При вызове функции subclust без выходных аргументов координаты центров кластеров выводятся на экран.

Пример

```
data = load('clusterdemo.dat');
centers = subclust(data, 0.3)
plot3(data(:, 1), data(:, 2), data(:, 3), 'm+',...
'markersize', 3);
hold on
plot3(centers(:, 1), centers(:, 2), centers(:, 3),...
'k.', 'markersize', 22);
grid on
```

Кластерный анализ данных из файла clusterdemo.dat. Центры найденных кластеров указаны кружками.



sugmax

Назначение

Расчет диапазонов изменения выходных переменных системы нечеткого вывода Сугено.

Синтаксис

```
out = sugmax(fis)
```

Описание

Рассчитывает диапазоны изменения выходных переменных системы нечеткого вывода типа Сугено, заданной аргументом fis. Каждая строка матрицы out соответствует одной выходной переменной. В первом столбце матрицы записываются нижние, а во втором – верхние границы диапазонов изменения выходных переменных.

Пример

```
fis = readfis('juggler');
out = sugmax(fis)
```

Расчет диапазона изменения выходной переменной нечеткой демо-системы juggler.

trapmf

Назначение Трапециевидная функция принадлежности.

Синтаксис `y = trapmf(x, params)`

Описание Задает функцию принадлежности в форме трапеции:

$$\mu(x)=\begin{cases}0, & x\le a,\\ \dfrac{x-b}{b-a}, & a\le x\le b,\\ 1, & b\le x\le c,\\ \dfrac{d-x}{d-c}, & c\le x\le d,\\ 0, & d\le x.\end{cases}$$

Параметры трапециевидной функции принадлежности интерпретируются так:

[*a*, *d*] – носитель нечеткого множества (пессимистическая оценка значения переменной);

[*b*, *c*] – ядро нечеткого множества (оптимистическая оценка значения переменной).

Функция trapmf применяется для задания асимметричных функций принадлежности нечетких множеств, наиболее возможные значения которых достигаются на некотором интервале. Функция trapmf имеет два входных аргумента:

1) `x` – вектор, для координат которого рассчитываются степени принадлежности;

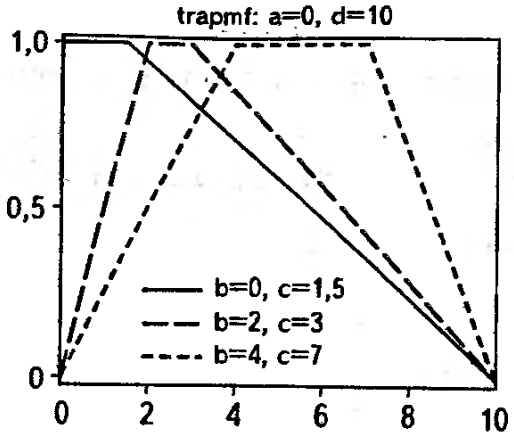
2) `params` – вектор параметров функции принадлежности. Порядок задания параметров: [*a b c d*]. Параметры функции принадлежности должны удовлетворять условию $a\le b\le c\le d$.

Функция trapmf возвращает степени принадлежности координат вектора `x`.

Пример

```
x = 0: 0.5: 10;
y1 = trapmf(x, [0 0 1.5 10]);
y2 = trapmf(x, [0 2 3 10]);
y3 = trapmf(x, [0 4 7 10]);
plot(x, y1, 'o-', x, y2, '+:', x, y3, '.-')
title('trapmf, a=0, d=10')
legend('b=0, c=1.5', 'b=2, c=3', 'b=4, c=7')
```

Построение трапециевидных функций принадлежности нечетких множеств с различными оптимистическими оценками.



trimf

Назначение Треугольная функция принадлежности.

Синтаксис

```
y = trimf(x, params)
```

Описание Задает функцию принадлежности в виде треугольника. Эта простая и наиболее часто применяемая функция принадлежности, которая задается следующим аналитическим выражением:

$$\mu(x)=\begin{cases}0, & x\le a,\\ \dfrac{x-a}{b-a}, & a\le x\le b,\\ \dfrac{c-x}{c-b}, & b\le x\le c,\\ 0, & c\le x.\end{cases}$$

Параметры треугольной функции принадлежности интерпретируются так:

$[a, c]$ – диапазон изменения переменной;

b – наиболее возможное значение переменной.

Функция trimf имеет два входных аргумента:

1) x – вектор, для координат которого рассчитываются степени принадлежности;

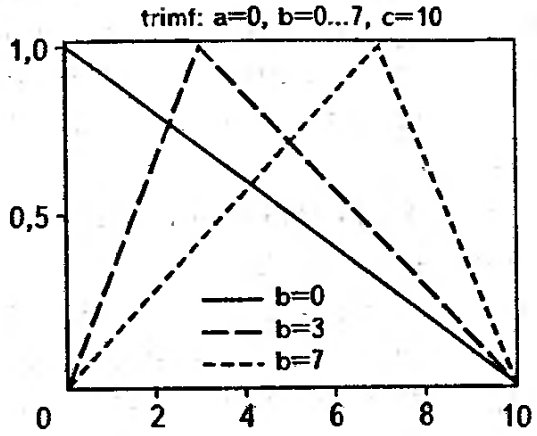
2) `params` – вектор параметров функции принадлежности. Порядок задания параметров: $[a\ b\ c]$. Параметры функции принадлежности должны удовлетворять условию $a \le b \le c$.

Функция trimf возвращает степени принадлежности координат вектора x.

Пример

```
x = 0: 0.5: 10;
y1 = trimf(x, [0 0 10]);
y2 = trimf(x, [0 3 10]);
y3 = trimf(x, [0 7 10]);
plot(x, y1, 'o-', x, y2, '+:', x, y3, '.-')
title('trimf, a=0, b=0...7, c=10');
legend('b=0', 'b=3', 'b=7')
```

Построение графиков треугольных функций принадлежности с параметрами `[0 0 10]`, `[0 3 10]` и `[0 7 10]`.



writefis

Назначение Сохранение FIS на диске.

Синтаксис `writefis(fis, filename, 'dialog')`

Описание Сохраняет систему нечеткого вывода на диске. Функция writefis может иметь до трех входных аргументов:

1) `fis` – идентификатор системы нечеткого вывода в рабочей области (обязательный аргумент);

2) `filename` – имя файла, в котором будет сохранена система нечеткого вывода;

3) `'dialog'` – константа, вызывающая типовое диалоговое окно записи файла на диск. В этом окне в качестве имени файла используется значение аргумента `filename`. Пользователь может изменить имя файла, а также указать папку, в которую будет производиться запись.

При вызове функции writefis с одним аргументом будет открыто типовое диалоговое окно записи файла на диск. Однако в отличие от вызова функции с тремя аргументами имя файла по умолчанию в этом окне установлено не будет.

При вызове функции writefis с двумя аргументами диалоговое окно записи файла на диск не появится. Система будет сохранена в текущей папке. Расширение «.fis» будет добавлено к имени файла в случае, если оно не было задано аргументом `filename`.

Пример

```
fis = readfis('tipper');
writefis(fis, 'tipper_copy')
```

Запись демо-системы нечеткого вывода «Tipper» в файл tipper_copy.fis.

zmf

Назначение *z*-подобная функция принадлежности.

Синтаксис `y = zmf(x, params)`

Описание

Задает *z*-подобную двухпараметрическую функцию принадлежности. Параметры функции принадлежности определяют интервал, внутри которого функция нелинейно убывает от 0 до 1. Функция zmf применяется для задания невозрастающих функций принадлежности с насыщением, представляющих нечеткие множества типа «очень низкий». Функция `zmf` имеет два входных аргумента:

1) `x` – вектор, для координат которого рассчитываются степени принадлежности;

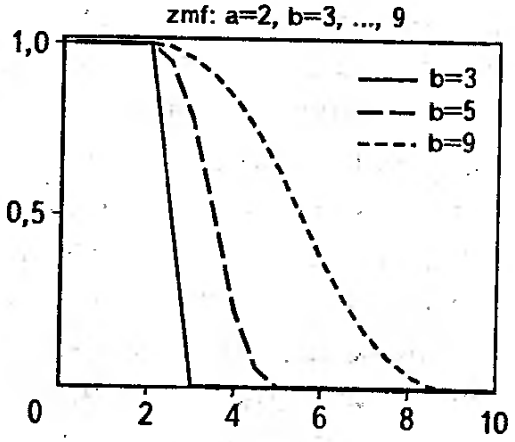
2) `params` – вектор параметров, которые задают интервал $[a, b]$ убывания функции принадлежности. Если $b \le a$, то функция принадлежности получается в виде единичной ступеньки, проходящей через точку $(a + b) / 2$. Порядок задания параметров: $[a, b]$.

Функция `zmf` возвращает степени принадлежности координат вектора `x`.

Пример

```
x = 0: 0.5: 10;
y1 = zmf(x, [2 3]);
y2 = zmf(x, [2 5]);
y3 = zmf(x, [2 9]);
plot(x, y1, 'o-', x, y2, '+:', x, y3, '.-')
title('zmf, a=2, b=3,..., 9')
legend('b=3', 'b=5', 'b=9')
```

Построение графиков *z*-подобных функций принадлежности с параметрами `[2 3]`, `[2 5]` и `[2 9]`.



3.6. СТРУКТУРЫ ДАННЫХ

В разделе описываются структуры данных пакета Fuzzy Logic Toolbox, используемые для представления систем нечеткого вывода в рабочей области, хранения их на диске, а также для представления данных для ANFIS-обучения и нечеткой кластеризации.

3.6.1. СТРУКТУРА ДАННЫХ СИСТЕМЫ НЕЧЕТКОГО ВЫВОДА

Система нечеткого вывода представляется в рабочей области MATLAB структурой данных, изображенной на рис. 3.116. Существует два способа загрузки FIS в рабочую область:

- считывание с диска с помощью функции readfis;
- передача из GUI-модулей через команду **To workspace** подменю **Export** меню **File**.

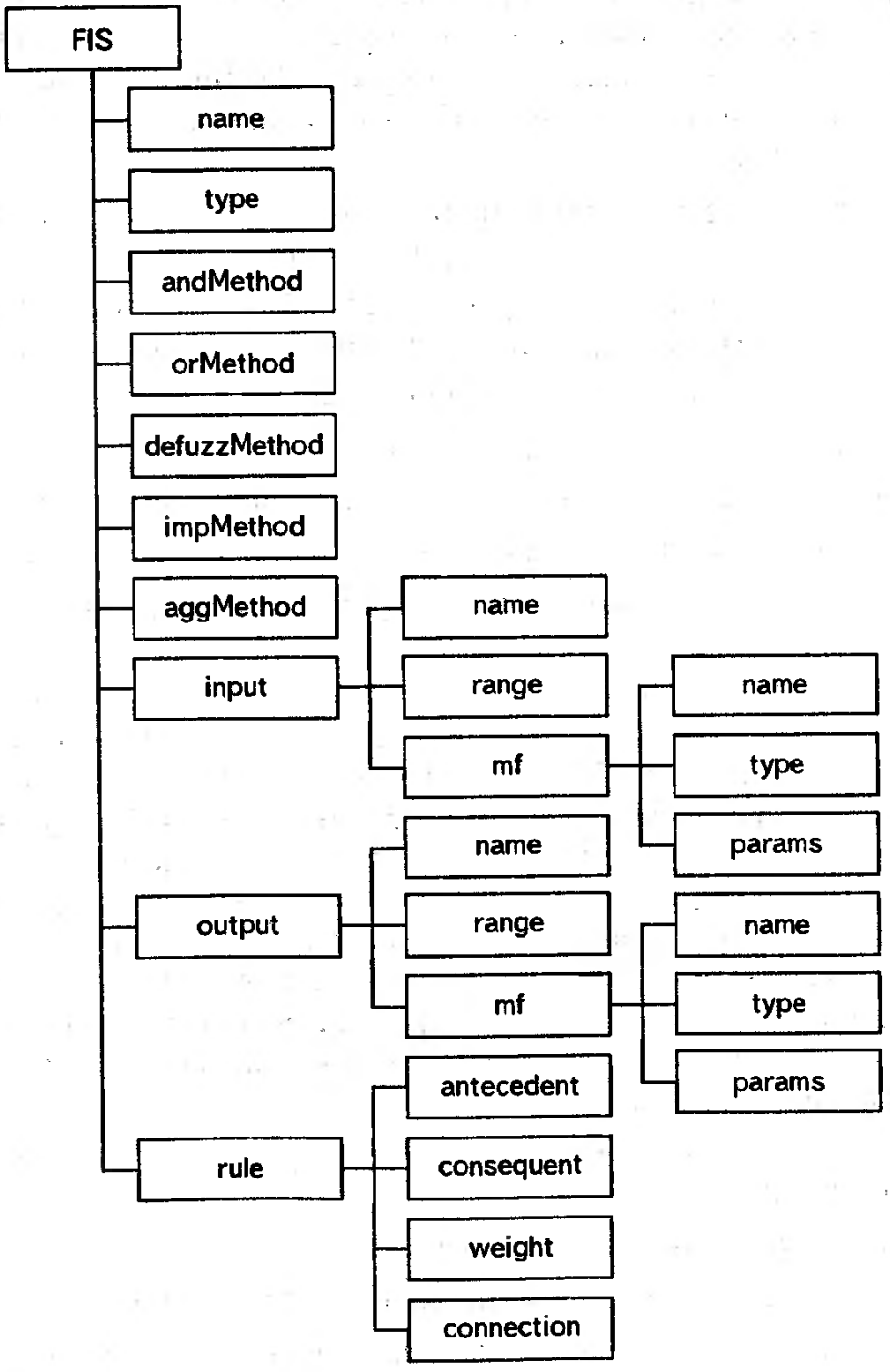


Рис. 3.116. Структура данных FIS

Поля структуры данных системы нечеткого вывода содержат следующую информацию:

- `name` – наименование системы нечеткого вывода;
- `type` – тип системы. Допустимые значения: 'Mamdani' и 'Sugeno';
- `andMethod` – реализация логической операции И. Запрограммированные реализации: 'min' – минимум и 'prod' – умножение;
- `orMethod` – реализация логической операции ИЛИ. Запрограммированные реализации: 'max' – максимум и 'probor' – вероятностное ИЛИ;
- `defuzzMethod` – метод дефаззификации. Запрограммированные методы для систем типа Мамдани: 'centroid' – центр тяжести; 'bisector' – медиана; 'lom' – наибольший из максимумов; 'som' – наименьший из максимумов; 'mom' – среднее из максимумов. Запрограммированные методы для систем типа Сугено: 'wtaver' – взвешенное среднее и 'wtsum' – взвешенная сумма;
- `impMethod` – реализация импликации. Запрограммированные реализации: 'min' – минимум и 'prod' – умножение;
- `aggMethod` – реализация агрегирования. Запрограммированные реализации: 'max' – максимум; 'sum' – сумма и 'probor' – вероятностное ИЛИ;
- `input` – массив входных переменных системы;
- `input.name` – наименование входной переменной;
- `input.range` – диапазон изменения входной переменной;
- `input.mf` – массив функций принадлежности входной переменной;
- `input.mf.name` – наименование функции принадлежности входной переменной;
- `input.mf.type` – тип функции принадлежности входной переменной. Запрограммированные функции: 'dsigmf' – функция принадлежности в виде разности между двумя сигмоидными функциями; 'gauss2mf' – двухсторонняя гауссова функция принадлежности; 'gaussmf' – гауссова функция принадлежности; 'gbellmf' – обобщенная колоколообразная функция принадлежности; 'pimf' – пи-подобная функция принадлежности; 'psigmf' – произведение двух сигмоидных функций принадлежности; 'sigmf' – сигмоидная функция принадлежности; 'smf' – *s*-подобная функция принадлежности; 'trapmf' – трапециевидная функция принадлежности; 'trimf' – треугольная функция принадлежности; 'zmf' – *z*-подобная функция принадлежности;
- `input.mf.params` – массив параметров функции принадлежности входной переменной;
- `output` – массив выходных переменных;
- `output.name` – наименование выходной переменной;
- `output.range` – диапазон изменения выходной переменной;
- `output.mf` – массив функций принадлежности выходной переменной;

- `output.mf.name` – наименование функции принадлежности выходной переменной;
- `output.mf.type` – тип функции принадлежности выходной переменной. Запрограммированные функции для системы типа Мамдани описаны в поле `input.mf.type`. Для системы Сугено заключения правил могут быть заданы константами ('constatnt') или линейной комбинацией входных переменных ('linear');
- `output.mf.params` – массив параметров функции принадлежности выходной переменной;
- `rule` – массив правил нечеткой базы знаний;
- `rule.antecedent` – посылки правила. Указываются порядковые номера термов в порядке записи входных переменных. Число 0 указывает на то, что значение соответствующей входной переменной не влияет на истинность правила;
- `rule.consequent` – заключения правила. Указываются порядковые номера термов в порядке записи выходных переменных. Число 0 указывает на то, что правило не распространяется на соответствующую выходную переменную;
- `rule.weight` – вес правила. Задается числом из диапазона [0, 1];
- `rule.connection` – логическая связка переменных внутри правила: 1 – логическое И; 2 – логическое ИЛИ.

Для доступа к свойствам системы нечеткого вывода достаточно указать имя соответствующего поля. Например, команда `FIS_NAME.rule(1).weight = 0.5` устанавливает вес первого правила в 0,5, команда `length(FIS_NAME.rule)` определяет количество правил в базе знаний, а команда `FIS_NAME.input(1).mf(1).name = 'Низкий'` переименовывает первый терм первой входной переменной в `Низкий`.

3.6.2. СТРУКТУРА ФАЙЛА СИСТЕМЫ НЕЧЕТКОГО ВЫВОДА

Системы нечеткого вывода хранятся на диске в виде fis-файлов – текстовых файлов специального формата. Функции readfis и writefis используются для чтения и записи этих файлов. Иногда удобно модифицировать системы нечеткого вывода путем изменения их fis-файлов в текстовом редакторе без вызова GUI-модулей Fuzzy Logic Toolbox. Редактировать fis-файлы надо осторожно, так как при изменении некоторых строчек файла требуются исправления нескольких других. Например, если при редактировании удален терм, то необходимо убедиться, что в базе знаний отсутствуют правила с этим термом.

Формат fis-файла описан в табл. 3.2 на примере файла нечеткой демо-системы tipper.fis. Правила базы знаний представлены в индексном формате.

Таблица 3.2

Формат fis-файла (на примере файла tipper.fis)

Строки файла	Описание
% $Revision: 1,1$	Комментарий
[System]	Метка общих параметров системы нечеткого вывода
Name = 'tipper'	Наименование системы
Type = 'mamdani'	Тип системы нечеткого вывода
NumInputs = 2	Количество входных переменных системы
NumOutputs = 1	Количество выходных переменных системы
NumRules = 3	Количество правил в базе знаний
AndMethod = 'min'	Реализация логической операции И
OrMethod = 'max'	Реализация логической операции ИЛИ
ImpMethod = 'min'	Реализация импликации
AggMethod = 'max'	Реализация агрегирования
DefuzzMethod = 'centroid'	Метод дефаззификации
[Input1]	Метка первой входной переменной
Name = 'service'	Наименование первой входной переменной
Range = [0 10]	Диапазон изменения
NumMF s = 3	Количество термов
MF1 = 'poor':'gaussmf', [1.5 0]	Первый терм, тип функции принадлежности и ее параметры
MF2 = 'good':'gaussmf', [1.5 5]	Второй терм, тип функции принадлежности и ее параметры
MF3 = 'excellent':'gaussmf', [1.5 10]	Третий терм, тип функции принадлежности и ее параметры
[Input2]	Метка второй входной переменной
Name = 'food'	Наименование второй входной переменной
Range = [0 10]	Диапазон изменения
NumMFs = 2	Количество термов
MF1 = 'rancid':'trapmf', [0 0 1 3]	Первый терм, тип функции принадлежности и ее параметры
MF2 = 'delicious':'trapmf', [7 9 10 10]	Второй терм, тип функции принадлежности и ее параметры
[Output1]	Метка первой выходной переменной
Name = 'tip'	Наименование выходной переменной
Range = [0 30]	Диапазон изменения
NumMFs = 3	Количество термов
MF1 = 'cheap':'trimf', [0 5 10]	Первый терм, тип функции принадлежности и ее параметры
MF2 = 'average':'trimf', [10 15 20]	Второй терм, тип функции принадлежности и ее параметры
MF3 = 'generous':'trimf', [20 25 30]	Третий терм, тип функции принадлежности и ее параметры
Rules]	Метка правил базы знаний
1, 1 (1) : 2 0, 2 (1) : 1 2, 3 (1) : 2	Правила базы знаний в индексном формате

3.6.3. СТРУКТУРЫ ДАННЫХ ДЛЯ ANFIS-ОБУЧЕНИЯ И КЛАСТЕРИЗАЦИИ

Выборки данных для ANFIS-обучения и кластеризации задаются матрицами, каждая строчка которых соответствует признакам одного объекта. При обучении нейро-нечетких сетей первый столбец матрицы данных соответствует значениям первой входной переменной, второй столбец матрицы – значениям второй входной переменной и т.д. Последний столбец матрицы данных содержит значения выходной переменной. На диске выборки данных записываются в аналогичном формате в виде текстовых файлов.

3.7. ВЗАИМОДЕЙСТВИЕ С ДРУГИМИ ПАКЕТАМИ

В разделе описывается, как использовать системы нечеткого вывода при моделировании динамических объектов в Simulink и как применять спроектированные в Fuzzy Logic Toolbox нечеткие системы вне программной среды MATLAB.

3.7.1. БЛОКИ ДЛЯ ПАКЕТА SIMULINK

Взаимодействие с пакетом Simulink происходит через библиотеку fuzblock (рис. 3.117), которая содержит следующие блоки:

- Fuzzy Logic Controller – нечеткий контроллер;
- Fuzzy Logic Controller with Ruleviewer – нечеткий контроллер с выводом окна **RuleViewer** во время моделирования в пакете Simulink;
- Membership Functions – библиотека симулинк-блоков для следующих функций принадлежностей и реализаций логических операций:

  Diff. Sigmoidal MF – разница двух сигмоидных функций принадлежности;

  Gaussian MF – гауссова функция принадлежности;

  Gaussian 2MF – двухсторонняя гауссова функция принадлежности;

  Generalized Bell MF – обобщенная колоколообразная функция принадлежности;

  Pi-shaped MF – пи-подобная функция принадлежности;

  Probabilistic OR – вероятностная реализация логической операции ИЛИ;

  Probabilistic Rule Agg – вероятностная реализация агрегирования;

  Prod. Sigmoidal MF – произведение двух сигмоидных функций принадлежности;

  S-shaped MF – *s*-подобная функция принадлежности;

  Sigmoidal MF – сигмоидная функция принадлежности;

  Trapezoidal MF – трапециевидная функция принадлежности;

  Triangular MF – треугольная функция принадлежности;

  Z-shaped MF – *z*-подобная функция принадлежности.

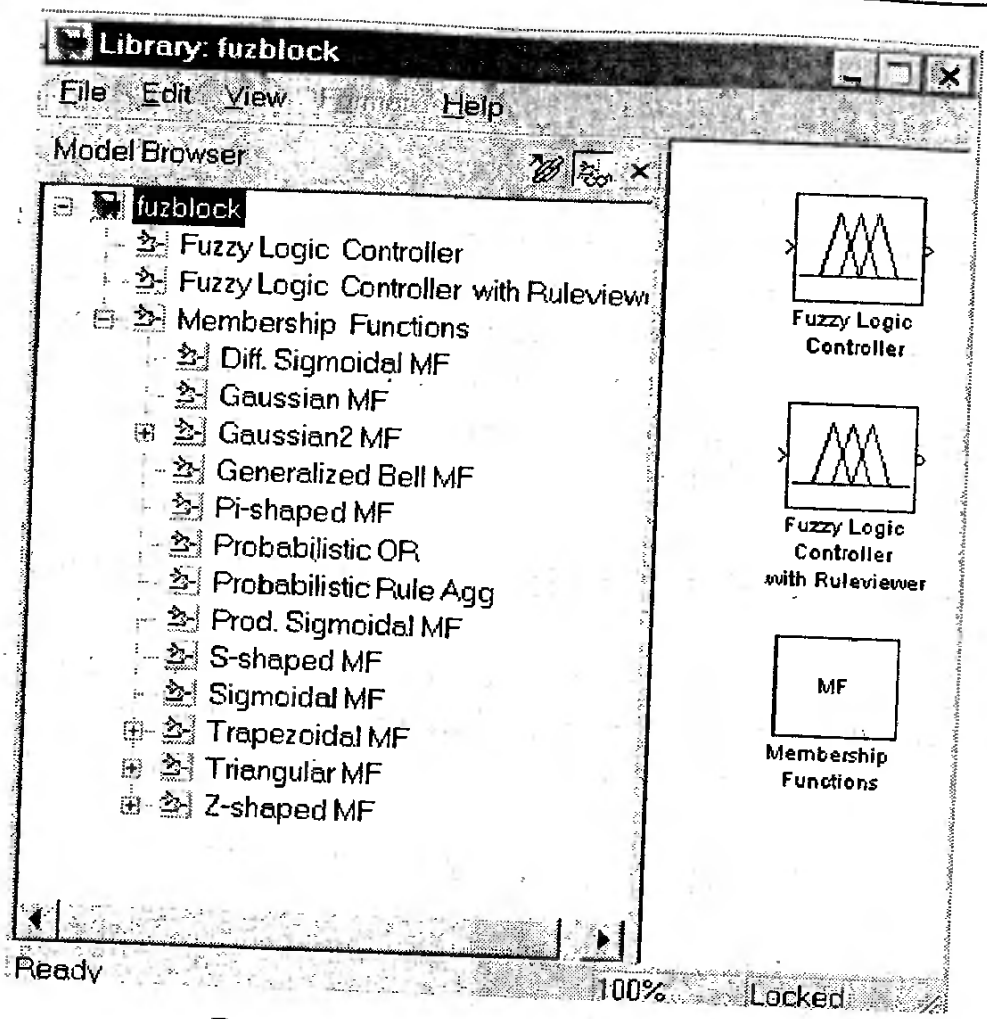


Рис. 3.117. Библиотека fuzblock

Для включения системы нечеткого вывода в симулинк-модуль необходимо крыть fuzblock командой `fuzblock` или через опцию **Fuzzy Logic Toolbox** в nulink Library Browser. Затем выбрать блок Fuzzy Logic Controller или Fuzzy gic Controller with Ruleviewer, сделать двойной щелчок по этому блоку и в поя-ившемся диалоговом окне ввести имя файла или наименование переменной из очей области, которые соответствуют системе нечеткого вывода. Если нечет-система имеет несколько входов, тогда в симулинк-модели необходимо соеди-ть их вместе до ввода в нечеткий контроллер. Аналогично, если нечеткая систе-имеет несколько выходов, тогда выходные сигналы блока будут представлены ой мультиплексной линией.

Для большинства нечетких систем fuzblock автоматически генерирует иерар-скую модель, состоящую из симулинк-модулей (например, см. рис. 3.85). оматический синтез модели происходит с помощью FIS Wizard. Синтезиро-ные модели состоят только из встроенных симулинк-модулей, поэтому нечет-вывод выполняется очень быстро, даже если модель получается громоздкой. Wizard генерирует симулинк-модули, если нечеткая система содержит только оенные функции принадлежности. Кроме того, должны использоваться такие изации логических операций: ИЛИ – max; И – min и prod; импликация – min и ; агрегирование – max. Если создать нечеткую систему из симулинк-блоков не гся, тогда используется sffis – специально оптимизированная под Simulink ция нечеткого вывода.

3.7.2. СИ-КОД МАШИНЫ НЕЧЕТКОГО ЛОГИЧЕСКОГО ВЫВОДА

В Fuzzy Logic Toolbox включены два файла `fismain.c` и `fis.c`, содержащие исходные коды автономной машины нечеткого вывода на языке Си. Эти файлы позволяют загрузить fis-файл, файл исходных данных и выполнить нечеткий вывод. Машина нечеткого вывода может быть встроена во внешние модули.

При вызове функции fismain необходимо указать два входных аргумента: файл данных, для которых будет выполняться нечеткий вывод, и файл системы нечеткого вывода (fis-файл). Для создания исполняемого кода си-файлы должны быть откомпилированы.

В качестве примера рассмотрим выполнение логического вывода нечеткой демо-системой, заданной файлом `mam21.fis`. Сначала сгенерируем данные для нечеткого вывода, используя следующие команды MATLAB:

```
[x, y] = meshgrid(-5:5, -5:5);
input_data = [x(:), y(:)];
save fis_in input_data - ascii
```

В результате на диск запишется ASCII-файл `fis_in` с входными данными в виде таблицы размером 121×2. Каждая строка таблицы соответствует одному входному вектору. Теперь выполним нечеткий вывод с помощью программы `fismain`. Синтаксис вызова `fismain` такой же, как и функции `evalfis`, за исключением того, что все матрицы и структуры заменены именами файлов. Для вызова `fismain` в системе UNIX необходимо напечатать:

```
%fismain fis_in mam21.fis > fis_out
```

После этого на диске появится файл `fis_out` с результатами нечеткого вывода. Файл содержит 121 строчку, в каждой из которых записан результат нечеткого вывода для одного входного вектора.

Для сравнения результатов нечетких выводов, выполняемых матлабовским mex-файлом `evalfis` и программой `fismain`, выполним в среде MATLAB следующий сценарий:

```
fis = readfis('mam21');
matlab_out = evalfis(input_data, fis);
load fis_out
max(max(abs(matlab_out - fis_out)))
```

В результате получим значение максимального абсолютного отклонения между результатами нечетких выводов. Отклонения имеет порядок 10^{-13}, точное значение которого зависит от аппаратных средств (длины разрядной сетки).

При использовании Си-кода машины нечеткого логического вывода необходимо помнить, что:

- Си-код совместим с ANSI- и K&R-стандартами на язык Си;
- Си-код не поддерживает определяемые пользователем функции принадлежности и реализации логических операций И, ИЛИ, импликации и агре-

гирования. Таким образом, доступными являются встроенные реализации этих операций и 11 типовых функций принадлежности;

- `fismain.c` содержит только одну функцию – main(). Она довольно плохо документирована, поэтому ее легкая адаптация для других приложений проблематична;
- для добавления нового типа функций принадлежности или для изменения механизма вывода необходимо внести изменения в файл `fis.c`, который содержит все необходимые функции для выполнения нечеткого вывода;
- для компьютеров Macintosh программа `fismain` по умолчанию будет использовать следующие файлы: `fismain.in` – для исходных данных; `fismain.fis` – для системы нечеткого вывода; `fismain.out` – для результатов нечеткого вывода. Имена этих файлов заданы в файле `fismain.c`; они могут быть изменены по необходимости.

Глава 4. РАСШИРЕНИЕ ПАКЕТА FUZZY LOGIC TOOLBOX

В главе представлены авторские расширения пакета Fuzzy Logic Toolbox, позволяющие: 1) настраивать нечеткие базы знаний типа Мамдани; 2) синтезировать нечеткие модели из экспериментальных данных с помощью нечеткой кластеризации; 3) осуществлять нечеткий логический вывод при нечетких значениях входов; 4) проектировать нечеткие классификаторы; 5) создавать иерархические нечеткие системы. Код, описываемых в главе программ, доступен на *www.vinnitsa.com/shtovba*. В главе использованы материалы работ [20, 21, 22, 40].

4.1. НАСТРОЙКА НЕЧЕТКИХ МОДЕЛЕЙ МАМДАНИ СРЕДСТВАМИ OPTIMIZATION TOOLBOX

Пакет Fuzzy Logic Toolbox обеспечивает настройку нечетких моделей типа Сугено посредством функции anfis и GUI-модуля anfisedit. Теоретические сведения по настройке нечетких моделей Мамдани приведены в подразделе 2.1.1. Для настройки нечетких моделей типа Мамдани в среде MATLAB предлагается использовать функции пакета оптимизации Optimization Toolbox [36]. Для этого необходимо запрограммировать целевую функцию задачи настройки (2.2) и написать сценарий оптимизации с использованием функций Optimization Toolbox, например, fmincon. Как это сделать, показано ниже на примере проектирования нечеткой системы прогнозирования топливной эффективности автомобиля.

Задача прогнозирования топливной эффективности автомобиля описана в подразделе 3.4.2. Прогнозирование топливной эффективности будем проводить по двум параметрам автомобиля: масса и год выпуска. В FIS-редакторе создадим нечеткую систему с двумя входами (Weight – масса автомобиля и Year – год выпуска) и одним выходом MPG – топливная эффективность автомобиля. Установим такие диапазоны изменения входов: [1613, 5140] – для Weight и [70, 82] для Year, что соответствует минимальным и максимальным значениям в выборке данных. Диапазон изменения выхода установим такой: [5, 50]. Он немного перекрывает значения MPG в выборке данных, так как при дефаззификации по методу центра тяжести фактический интервал изменения выхода нечеткой системы всегда уже диапазона, указанного в свойствах выходной переменной. Для лингвистической оценк

переменной Weight будем использовать термы «легкий» и «тяжелый», для переменной Year – термы «старый» и «новый», и для переменной MPG – термы «низкая», «средняя» и «высокая». В качестве функций принадлежностей выберем гауссовы кривые с установленными по умолчанию параметрами (рис. 4.1*а*). Взаимосвязь «входы – выход» опишем такой базой знаний:

ЕСЛИ Weight = «легкий» И Year = «новый», ТО MPG = «высокая»;

ЕСЛИ Weight = «тяжелый» И Year = «старый», ТО MPG = «низкая»;

ЕСЛИ Weight = «легкий» И Year = «старый», ТО MPG = «средняя».

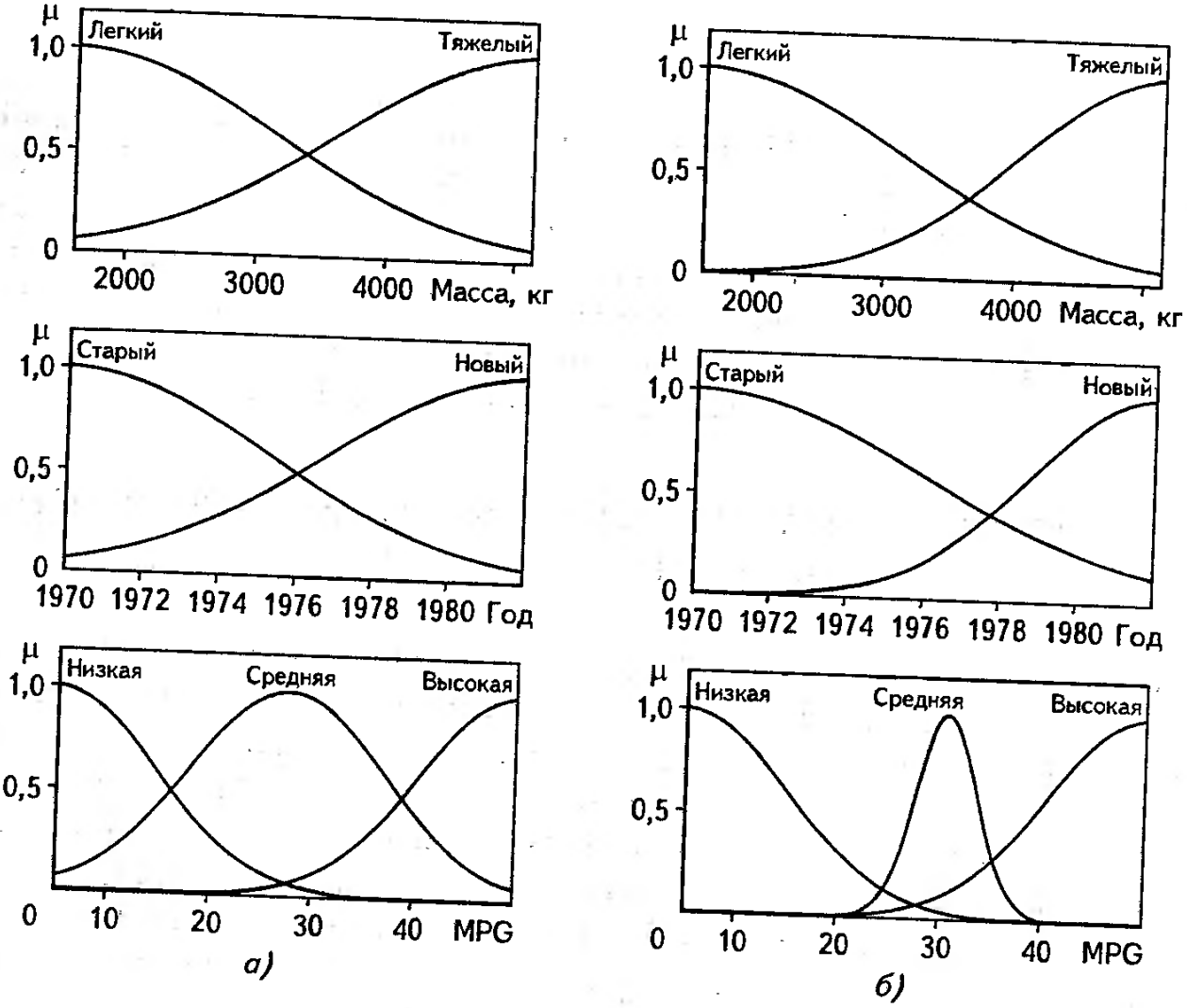
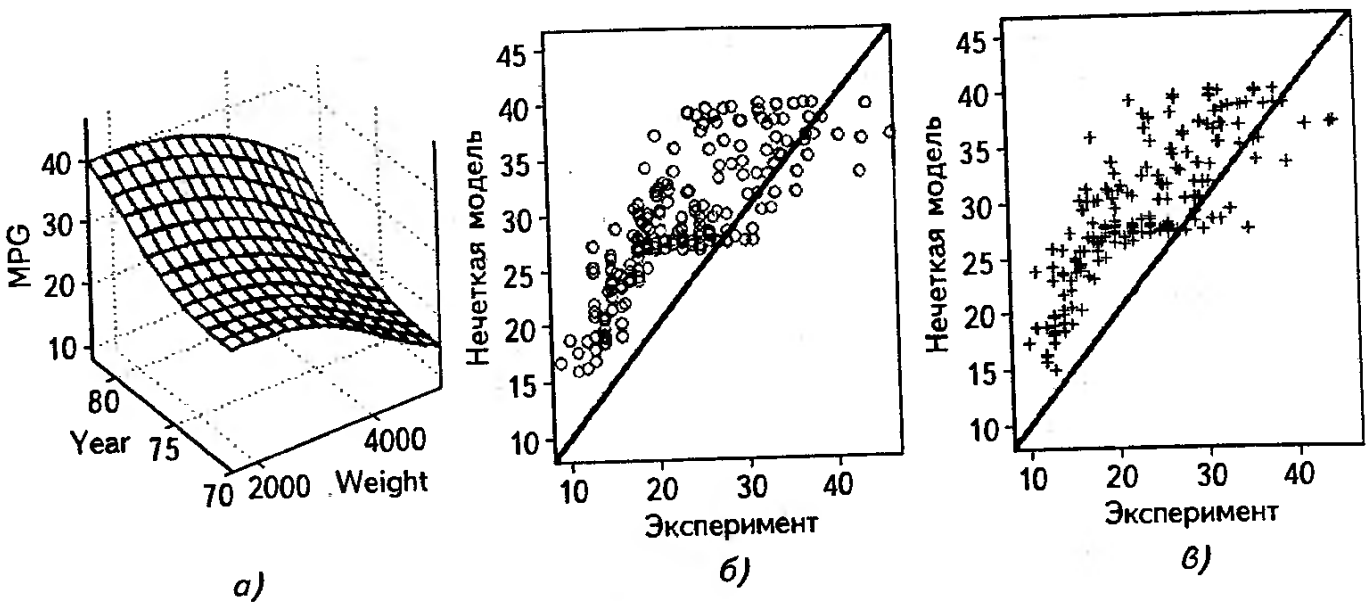


Рис. 4.1. Функции принадлежности для прогнозирования MPG

а – до настройки; *б* – после настройки

Поверхность «входы – выход» исходной нечеткой системы показана на ис. 4.2*а*. До настройки нечеткая модель плохо описывает исследуемую зависи-ость: ошибка RMSE на обучающей выборке равна 7,51, а на тестовой – 7,37. а рис. 4.2*б* и рис. 4.2*в* сравниваются экспериментальные и модельные результа-ы, из которых видно, что в большинстве случаев предсказанная топливная эф-ективность выше действительной.

Будем настраивать 11 параметров нечеткой базы знаний: семь коэффициентов центраций функций принадлежности нечетких термов входных и выходных



а) *б)* *в)*

Рис. 4.2. Тестирование нечеткой системы прогнозирования MPG до настройки

а – нечеткая модель; *б* – проверка на обучающей выборке; *в* – проверка на тестовой выборке

переменных, координату максимума функции принадлежности терма «средняя» и три весовых коэффициента правил. Коэффициенты концентраций функций принадлежности ограничим такими интервалами:

- [750, 2000] – для нечетких термов «легкий» и «тяжелый»;
- [3, 8] – для нечетких термов «старый» и «новый»;
- [3, 20] – для нечетких термов «низкая», «средняя» и «высокая».

Сценарий настройки, использующий функцию нелинейной оптимизации fmincon пакета Optimization Toolbox, а также файлы целевой функции и функции установки новых параметров нечеткой системы приведены ниже.

```
%СЦЕНАРИЙ НАСТРОЙКИ НЕЧЕТКОЙ СИСТЕМЫ ТИПА МАМДАНИ
%ДЛЯ ПРОГНОЗИРОВАНИЯ MPG.
%Serhiy D. Shtovba %$Revision: 1.1 $ $Date: 2004/05/30
%Формирование обучающей (tr_data) и тестовой (ch_data) выборок:
[data, input_name] = loadgas;
%M_INP = data(:, [4 6 7]);
tr_data = M_INP(1:2:end, :);
ch_data = M_INP(2:2:end, :);
%Загрузка исходной нечеткой системы:
fis = readfis('mpg_mamdani0.fis');
num_rule = length(fis.rule); %<— Количество правил
%НАСТРАИВАЕМЫЕ ПАРАМЕТРЫ::::::::::::::::::::::::::::::
%Коэффициенты концентраций: 2 для Weight (*0.001), 2
%для Year (*0.1) и 3 для MPG (*0.1)(параметры умножены
%на соответствующие коэффициенты, чтобы все управляемые
%переменные были одного порядка:
vlb_c = [.750   .750  .3  .3   .3  .3   .3]; %нижняя граница
c0    =  [1.500 1.500 .5  .5  1.0 1.0 1.0]; %начальная точка
```

```
vub_c = [2.000  2.000  .8  .8  2.0 2.0 2.0]; %верхняя граница
%Координата максимума функции принадлежности второго
%терма (средняя) переменной MPG (*0.1):
vlb_b = 1,5;                    %нижняя граница
b0 = 2,7;                       %начальная точка
vub_b = 4,0;                    %верхняя граница
%Веса правил:
vlb_w(1:num_rule) = 0;          %нижняя граница
w0(1:num_rule) = 1;             %начальная точка
vub_w(1:num_rule) = 1;          %верхняя граница
%Объединение настраиваемых параметров в один вектор:
vlb = [vlb_c vlb_b vlb_w];      %нижняя граница
x0 = [c0 b0 w0];                %начальная точка
vub = [vub_c vub_b vub_w];      %верхняя граница
%ПАРАМЕТРЫ ОПТИМИЗАЦИИ:
options = [];
options = optimset('Display', 'iter');
options.DiffMinChange = 0.0001;
options.DiffMaxChange = 0.2;
options.LargeScale = 'off';
options.MaxIter = 25;
%Оптимизация:
[xopt, delta] = fmincon(@ob_fun_mpg, x0, [], [], [],...
[], vlb, vub, [], options, fis, tr_data(:, 1:2),...
tr_data(:, 3));
%Расчет невязки на тестовой выборке:
fis = change_fis_mpg(xopt, fis);
out_ch = evalfis(ch_data(:, 1:2), fis);
rmg_ch = norm(out_ch-ch_data(:, 3))/sqrt(length(ch_data));
writefis(fis, 'mpg_mamdani_trained.fis');
%Запись настроеной нечеткой системы
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

function delta = ob_fun_mpg(x, fis, inp, target);
%РАСЧЕТ НЕВЯЗКИ ПРИ НОВЫХ ПАРАМЕТРАХ x НЕЧЕТКОЙ СИСТЕМЫ fis.
%Установка новых параметров нечеткой системы:
fis = change_fis_mpg(x, fis);
%Нечеткий вывод:
out_tr = evalfis(inp, fis);
%Расчет невязки:
delta = norm(out_tr - target)/sqrt(length(target))
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

function fis_new = change_fis_mpg(x, fis_old);
%УСТАНОВКА НОВЫХ ПАРАМЕТРОВ x НЕЧЕТКОЙ СИСТЕМЫ fis_old
fis_new = fis_old;
fis_new.input(1).mf(1).params(1) = 1000*x(1);
fis_new.input(1).mf(2).params(1) = 1000*x(2);
```

```
fis_new.input(2).mf(1).params(1)  = 10*x(3);
fis_new.input(2).mf(2).params(1)  = 10*x(4);
fis_new.output(1).mf(1).params(1)  = 10*x(5);
fis_new.output(1).mf(2).params(1)  = 10*x(6);
fis_new.output(1).mf(2).params(1)  = 10*x(7);
fis_new.output(1).mf(2).params(2)  = 10*x(8);
fis_new.rule(1).weight = x(9);
fis_new.rule(2).weight = x(10);
fis_new.rule(3).weight = x(11)
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

Функции принадлежности после настройки показаны на рис. 4.1б. Весов коэффициенты получились такими: 0,153 – для первого правила, 1 – для второго 0,076 – для третьего. Протокол настройки нечеткой системы приведен ниже:

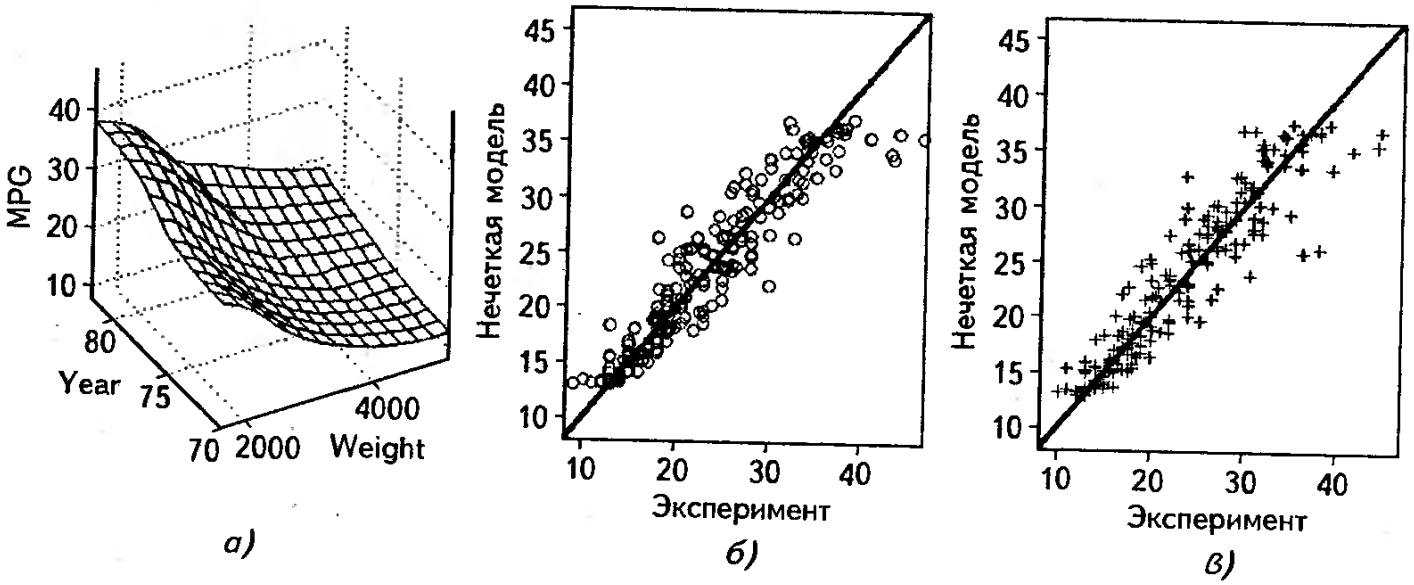
Iter	F-count	f(x)	max constraint	Step-size	Directional derivative	First-ord optimalit	Procedu
0	12	7.58002	0				
1	26	4.20333	0	0.5	2.75	14.1	
2	40	3.62348	0	0.5	1.6	3.21	
3	54	3.53554	0	0.5	2.13	3.87	
4	68	3.28778	-0.0008742	0.5	0.212	1.54	
5	83	3.16848	-0.02117	0.25	-0.0358	0.55	
6	98	3.14958	-0.01588	0.25	0.175	1.33	
7	111	3.10668	0	1	0.0109	0.24	
8	124	3.0949	0	1	0.199	0.88	
9	138	3.06397	0	0.5	0.275	1.06	
10	152	3.03244	-0.00707	0.5	0.31	3.05	
11	167	2.96174	-0.005303	0.25	0.474	1.02	
12	180	2.93601	0	1	0.226	3.24	
13	193	2.89311	0	1	0.154	2.05	
14	207	2.88763	0	0.5	0.141	2.08	
15	220	2.88342	0	1	0.0857	1.19	
16	233	2.84532	0	1	0.0267	0.87	
17	246	2.82986	0	1	0.00255	0.47	
18	259	2.82452	0	1	-0.00399	0.4	
19	272	2.81701	0	1	-0.00185	0.0	
20	286	2.81533	0	0.5	-0.00052	0.3	
21	300	2.81307	0	0.5	-0.00457	0.6	
22	317	2.81237	0	0.0625	0.00232	0.9	
23	330	2.80666	0	1	-0.00368	0.4	
24	343	2.80026	0	1	-0.000173	0.2	
25	356	2.79792	0	1	-0.00118	0.0	
26	369	2.79726	0	1	0.000105	0.1	

```
Maximum number of iterations exceeded;
increase OPTIONS.MaxIter
xopt = 1,5050  1,1024  0,5871  0,3224  1,0013  1,0000...
0,3000  3,0570  0,1534  1,0000  0,0762
delta = 2,7979
rmg_ch = 2,8842
ans = mpg_mamdani_trained
```

Таблица 4.1

Сравнение моделей прогнозирования топливной эффективности автомобиля

Модель	RMSE на обучающей выборке	RMSE на тестовой выборке
Модель Мамдани	2,7979	2,8842
Модель Сугено	2,6965	2,9779
Линейная модель	3,5007	3,3373



а) *б)* *в)*

Рис. 4.3. Тестирование нечеткой системы прогнозирования MPG до настройки

а – нечеткая модель; *б* – проверка на обучающей выборке; *в* – проверка на тестовой выборке

В результате настройки ошибка прогнозирования MPG на обучающей выборе снизилась до уровня RMSE = 2,7979. В табл. 4.1 сравниваются настроенная модель Мамдани, наилучшая модель Сугено из демо-примера gasdemo и оптимальная инейная регрессионная модель MPG = –15,374 – 0,007Weight + 0,077Year. идно, что нечеткая модель Мамдани аппроксимирует лучше. Обратим внимание, что нечеткая модель Сугено значительно завышает топливную эффективность новых тяжелых автомобилей, информация о которых не попала в выборки анных (см. рис. 3.59). Тестирование нечеткой системы Мамдани после настройки (рис. 4.3*б* и рис. 4.3*в*) свидетельствует, что база знаний из трех нечетких правил орошо описывает взаимосвязь массы и года выпуска автомобиля с показателем опливной эффективности на всем факторном пространстве.

4.2. ЭКСТРАКЦИЯ НЕЧЕТКИХ МОДЕЛЕЙ МАМДАНИ ЧЕРЕЗ НЕЧЕТКУЮ КЛАСТЕРИЗАЦИЮ

В Fuzzy Logic Toolbox реализована экстракция из экспериментальных данных нечетких моделей Сугено. Ниже предлагаются программы экстракции из данных нечетких моделей типа Мамдани. Для синтеза базы правил применяется нечеткая кластеризация алгоритмом *c*-средних, который в Fuzzy Logic Toolbox реализован функцией fcm. Теория синтеза из данных нечетких <Если – то> правил описана в подразделе 2.2.4.

Процедура экстракции нечеткой базы состоит в выполнении следующей последовательности шагов:

- кластеризация данных алгоритмом нечетких *c*-средних;
- отображения каждого нечеткого кластера в одно нечеткое правило типа <Если – то>;
- аппроксимация многомерной функции принадлежности каждого нечеткого кластера функциями принадлежности термов входных и выходных переменных.

Для выполнения последнего шага будем использовать асимметричные функции принадлежности: треугольную (trimf) и двухстороннюю гауссову (gauss2mf). Координаты максимумов функций принадлежности принимаются равными центрам нечетких кластеров. Поиск оптимальных параметров функций принадлежности проведем по методу наименьших квадратов отдельно для левой и правой частей графика. Под левой и правой частями понимаются ветви графика функции принадлежности по обе стороны от координаты максимума.

Для экстракции из данных нечетких моделей типа Мамдани разработана программа genfis3. Для ее работы необходимы функции mf_tuning и rule_extractor. Описание этих программ приведено ниже.

genfis3

Назначение Экстракция из данных нечеткой модели типа Мамдани.

Синтаксис
```
fis = genfis3(data_set, N_rules, options_fcm,...
N_points, multi_ratio, mf_type)
```

Описание Генерирует из данных нечеткую модель типа Мамдани (`fis`) с помощью кластеризации по алгоритму нечетких *c*-средних. Для нечеткой кластеризации используется функция fcm. Для синтеза нечеткой модели используется функция rule_extractor. Функция genfis3 может иметь от двух до шести входных аргументов:

1) `data_set` – обучающая выборка. В каждой строчке записан один объект. Размер: Количество_объектов × Количество_признаков. Выходной переменной соответствует последний столбец матрицы;

2) `N_rules` – количество правил базы знаний;

3) options_fcm – параметры алгоритма нечетких *c*-средних. Подробнее смотри в описании функции fcm;

4) N_points – количество точек дискретизации, по которым ищутся оптимальные параметры функций принадлежности. Значение по умолчанию – 100;

5) multi_ratio – коэффициент, задающий ширину поиска оптимальных параметров функций принадлежности. Значение по умолчанию – 3;

6) mf_type – тип функции принадлежности. Допустимые значения: 'trimf' – треугольная и 'gauss2mf' – двухсторонняя гауссова. Значение по умолчанию – 'gauss2mf'.

mf_tuning

значение Аппроксимация асимметричной функцией принадлежности степеней принадлежности объектов нечеткому кластеру.

интаксис
```
new_params = mf_tuning(x, mu, params, N_points,...
multi_ratio, mf_type)
```

писание Подбирает параметры функции принадлежности, чтобы наилучшим образом описать нечеткий кластер. Как критерий близости используется средняя квадратичная невязка. Оптимальные параметры подбираются перебором. Функция genfis3 может иметь от трех до шести входных аргументов:

1) x – элементы универсального множества. Размер: 1 × Количество_объектов_кластеризации;

2) mu – вектор степеней принадлежности элементов x нечеткому кластеру. Размер: 1 × Количество_объектов_кластеризации;

3) params – вектор исходных параметров треугольной функции принадлежности. Координатами вектора являются нижняя граница универсального множества, координата центра кластера и верхняя граница универсального множества;

4) N_points – количество точек дискретизации, по которым ищутся оптимальные параметры функций принадлежности. Значение по умолчанию – 100;

5) multi_ratio – коэффициент, задающий ширину поиска оптимальных параметров функций принадлежности. Значение по умолчанию – 3;

6) mf_type – тип функции принадлежности. Допустимые значения: 'trimf' – треугольная и 'gauss2mf' – двухсторонняя гауссова. Значение по умолчанию – 'gauss2mf'.

Функция genfis3 возвращает оптимальные параметры функции принадлежности типа mf_type.

rule_extractor

Назначение Экстракция нечеткой модели типа Мамдани по результатам нечеткой кластеризации

Синтаксис
```
fis = rule_extractor(data_set, cluster_center,...
mu_cluster, N_points, multi_ratio, mf_type)
```

Описание Генерирует из данных нечеткую модель типа Мамдани (`fis`) по результатам нечеткой кластеризации. Для подбора параметров функций принадлежности используется функция mf_tuning. Функция genfis3 может иметь от трех до шести входных аргументов:

1) `data_set` – обучающая выборка. В каждой строчке записан один объект. Размер: Количество_объектов × Количество_признаков. Выходной переменной соответствует последний столбец матрицы;

2) `cluster_center` – координаты центров нечетких кластеров. Размер: Количество_кластеров × Количество_признаков;

3) `mu_cluster` – матрица степеней принадлежности объектов кластерам. Размер: Количество_кластеров × Количество_объектов;

4) `N_points` – количество точек дискретизации, по которым ищутся оптимальные параметры функций принадлежности. Значение по умолчанию – 100;

5) `multi_ratio` – коэффициент, задающий ширину поиска оптимальных параметров функций принадлежности. Значение по умолчанию – 3;

6) `mf_type` – тип функции принадлежности. Допустимые значения: `'trimf'` – треугольная и `'gauss2mf'` – двухсторонняя гауссова. Значение по умолчанию – `'gauss2mf'`.

Рассмотрим применение функции genfis3 для синтеза нечеткой системы прогнозирования топливной эффективности автомобиля. Условия задачи приведены в подразделе 3.4.2. Прогнозирование топливной эффективности осуществим по массе автомобиля и году создания модели. Для этого выполним следующие команды:

```
%Формирование обучающей и тестовой выборок:
[data, input_name] = loadgas;
M_INP = data(:, [4 6 7]);
tr_set = M_INP(1:2:end, :);
test_set = M_INP(2:2:end, :);
%Синтез нечеткой модели:
N_rules = 3;
N_points = 300;
multi_ratio = 3;
fis = genfis3(tr_set, N_rules, [], N_points, multi_ratio)
```

В результате получим нечеткую модель прогнозирования топливной эффективности, база знаний которой содержит следующие правила:

ЕСЛИ input1 = Inp1MF1 И input2 = Inp2MF1, ТО output = OutMF1;

ЕСЛИ input1 = Inp1MF2 И input2 = Inp2MF2, ТО output = OutMF2;

ЕСЛИ input1 = Inp1MF3 И input2 = Inp2MF3, ТО output = OutMF3.

Синтезированные функции принадлежности показаны на рис. 4.4. На этом рисунке маркеры соответствуют степеням принадлежностям объектов нечетким кластерам, а сплошные линии – синтезированным функциям принадлежности.

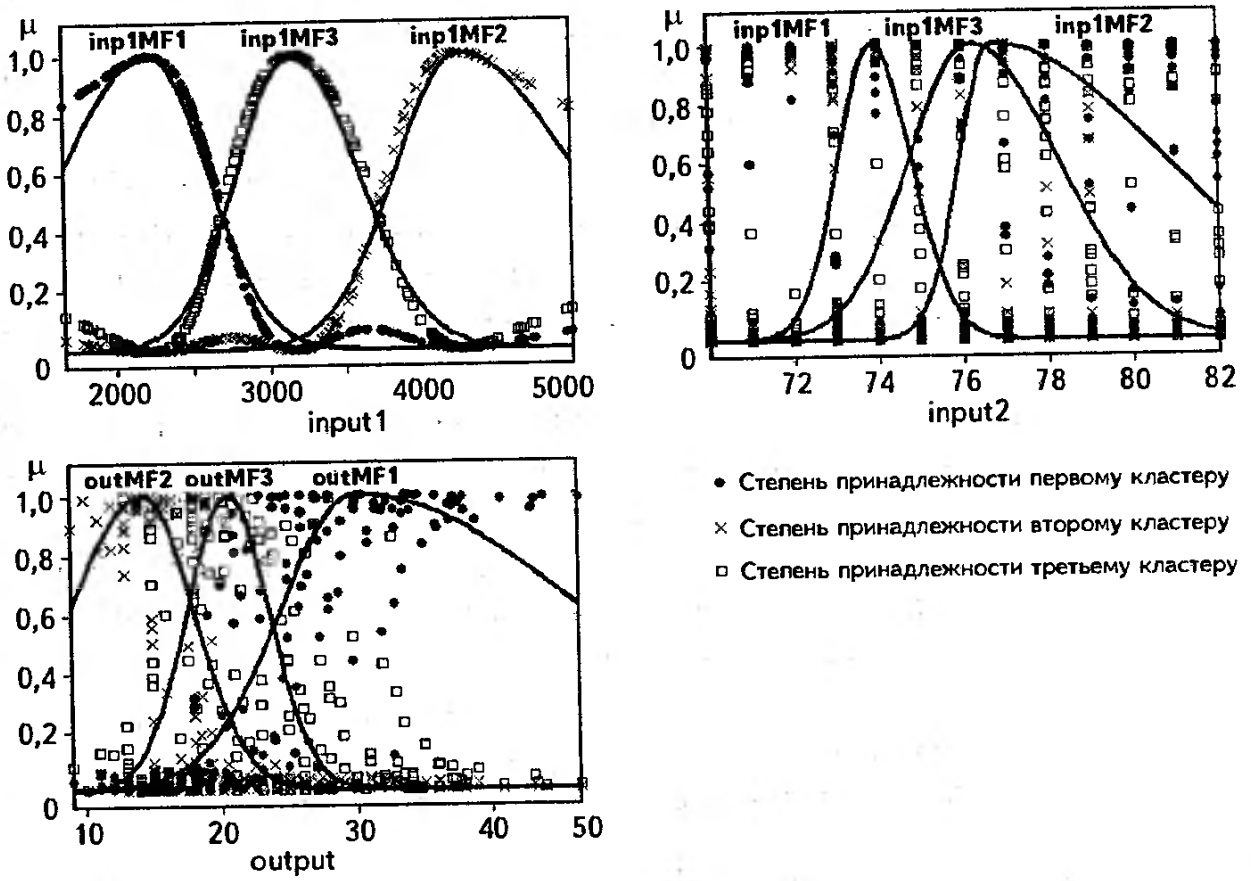


Рис. 4.4. Синтезированные программой genfis3 функции принадлежности нечеткой системы прогиозирования топливной эффективности автомобиля

При этих функциях принадлежности нечеткая база знаний содержательно интерпретируется так:

ЕСЛИ «масса» = «легкая» И «модель» = «новая»,
ТО «топливная_эффективность» = «высокая»;

ЕСЛИ «масса» = «тяжелая» И «модель» = «старая»,
ТО «топливная_эффективность» = «низкая»;

ЕСЛИ «масса» = «средняя» И «модель» = «средняя»,
ТО «топливная_эффективность» = «средняя».

Нечеткая модель получилась прозрачной, ее правила не противоречат здравому смыслу. Проверим точность прогнозирования, выполнив следующие команды:

```
out_tr = evalfis(tr_set(:,1:2), fis);
out_test = evalfis(test_set(:,1:2), fis);
rmg_tr = norm(out_tr - tr_set(:, 3))/sqrt(length(tr_set))
rmg_test = norm(out_test - test_set(:, 3))/sqrt(length(test_set))
```

В результате получаем, что ошибка RMSE на обучающей выборке составляет 4,68, а на тестовой выборке – 4,69. Напомним, что ошибки прогнозирования нечеткой модели из предыдущего раздела, которая создана по экспертным правилам, составляют 7,51 и 7,37. Таким образом, синтезированная программой genfis3 нечеткая модель прогнозирует лучше, чем экспертная нечеткая модель. Для повышения точности нечеткую модель можно обучить. Как это сделать – описано в предыдущем разделе.

4.3. ПРОЕКТИРОВАНИЕ НЕЧЕТКИХ КЛАССИФИКАТОРОВ

Пакет Fuzzy Logic Toolbox предоставляет широкий набор инструментов для проектирования и исследования систем нечеткого вывода с непрерывным выходом. Ниже показывается, как расширить Fuzzy Logic Toolbox для проектирования нечетких классификаторов, т.е. нечетких систем с дискретным выходом (см. рис. 1.19*б*). В начале раздела приводится программа нечеткой классификации, использующая функции пакета Fuzzy Logic Toolbox, затем показывается, как настраивать нечеткие классификаторы с помощью пакета Optimization Toolbox, и в конце предлагается пакет программ для быстрой настройки весов правил нечеткого классификатора.

Реализовывать нечеткий классификатор (см. подраздел 1.6.8) будем через систему нечеткого вывода Сугено. Классам решений $\{d_1, d_2, \ldots, d_m\}$ поставим в соответствие термы выходной переменной; наименование класса решений зададим как элемент терм-множества выходной переменной. Параметры заключений правил (параметры «функций принадлежности» выходной переменной) могут быть произвольными, так как они не влияют на результат классификации. Проектировать системы нечеткого вывода типа Сугено удобно в FIS-редакторе.

Для выполнения нечеткой классификации разработана функция fuzzy_classifier, текст которой приведен ниже.

```
function [decision, mf_grades] = fuzzy_classifier(x, fis, type)
%FUZZY_CLASSIFIER выполняет нечеткий логический вывод
%для задач классификации:
%x - входной вектор (признаки классифицируемого объекта);
%fis - система нечеткого логического вывода;
%type - формат решения:
%'number' - порядковый номер класса (значение по умолчанию);
%'name' - наименование класса;
%decision - результат классификации объекта x;
%mf_grades - вектор степеней принадлежности каждому классу.
%Требуемые ресурсы: Fuzzy Logic Toolbox v.2.X
```

```
%$Revision: 1.6 $ $Date: 2004/03/12

%Проверка входных аргументов:
if nargin < 2
   error ('Необходимо задать 2 или 3 входных аргумента');
end
[tmp1 tmp2] = size(x);
if tmp1 ~= 1
   error('Классификация выполняется только для одного объекта');
end
if isfis(fis) == 0;
   error('Второй аргумент должен быть системой нечеткого вывода');
end
if lower(fis.type(1:6)) ~= 'sugeno'
   error('Система нечеткого вывода должна быть типа Сугено');
end
if nargin == 2
   type = 'number'; % <— Установка значения по умолчанию;
end

%Основная часть:
number_of_decisions = length(fis.output(1).mf);
number_of_rules = length(fis.rule);
[a, b, c, d] = evalfis(x, fis);
%<— Нечеткий логический вывод
%Номера правил с максимальной степенью выполнения:
rule_num_with_max_fulfilment = find(d == max(d));
%Количество таких правил:
number_rules_with_max = length(rule_num_with_max_fulfilment);
if number_rules_with_max > 1
   %Если таких правил несколько, то возможно объект попал
   %на границу классов:
   colision_flag = zeros(1, number_of_decisions);
   for i = 1:number_rules_with_max;
      index = rule_num_with_max_fulfilment(i);
      %Помечаем решения с максимальной степенью принадлежности:
      colision_flag(fis.rule(index).consequent) = 1;
   end
   %Суммируем степени принадлежности проблемного объекта
   %по разным правилам:
   sum_mf = zeros(1,number_of_decisions);
   for i = 1:number_of_rules
      index = fis.rule(i).consequent;
      sum_mf(index) = sum_mf(index) + d(i);
   end
```

```
        %Оставляем только классы с максимальной степенью
        %принадлежности и выбираем класс по максимальной сумме
        %степеней принадлежности:
        [tmp1 number_of_the_class] = max(sum_mf.*colision_flag);
    else number_of_the_class =...
    = fis.rule(rule_num_with_max_fulfilment).consequent;
    end
    %Возврат результата классификации в требуемом формате:
    switch type
        case 'number', decision=number_of_the_class;
        case 'name', decision =...
        = fis.output.mf(number_of_the_class).name;
        otherwise, error('Допустимые значения 3-го аргумента:
        %number или name')
    end
    if nargout == 2
        %Вектор степеней принадлежности каждому классу:
        mf_grades(1:number_of_decisions) = 0;
        for i = 1:number_of_rules
            index = fis.rule(i).consequent;
            %Номер класса в i-м правиле объединим через операцию
            %максимума степени принадлежности одному и тому же
            %классу по различным правилам:
            mf_grades(index) = max(mf_grades(index), d(i));
        end
    end
```

Функция fuzzy_classifier для получения промежуточных результатов нечеткого вывода вызывает функцию evalfis в формате `[a, b, c, d] = evalfis(X, fis)`. Затем находятся правила с максимальной степенью выполнения. Если таких правил несколько, тогда среди конкурирующих классов выбирается тот, сумма степеней принадлежности которому максимальна.

Проиллюстрируем применение программы fuzzy_classifier для классификации фишеровских ирисов. Задача состоит в отнесении ириса к одному из трех классов: 1 – Iris Setosa, 2 – Iris Versicolor; 3 – Iris Virginica. При классификации используются следующие признаки цветков: x_1 – длина чашелистика; x_2 – ширина чашелистика; x_3 – длина лепестка; x_4 – ширина лепестка. Исходные данные для классификации ирисов записаны в файле iris.dat из Fuzzy Logic Toolbox. Файл содержит 150 строк, каждая из которых описывает один ирис. Информация о цветке представлена пятеркой чисел – первые четыре числа соответствуют значениям признаков, а пятое – классу ириса. Двухмерные распределения ирисов показаны на рис. 4.5.

В FIS-редакторе создадим систему нечеткого вывода типа Сугено с четырьмя входами и одним выходом. Диапазоны изменения входных переменных установим такими же, как и для исходных данных: $x_1 \in [47, 79]$; $x_2 \in [20, 44]$; $x_3 \in [10, 69]$; $x_4 \in [1, 25]$. Для лингвистической оценки признаков цветков будем использовать

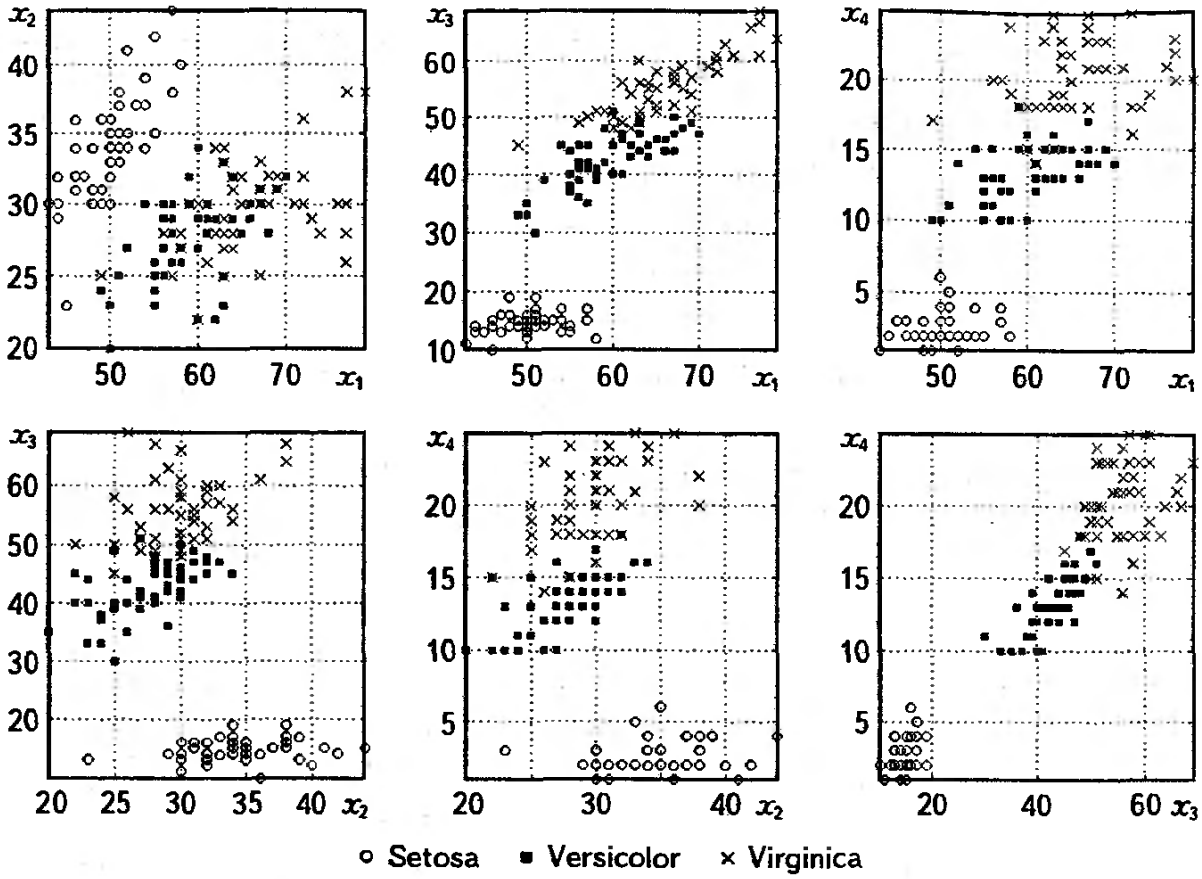


Рис. 4.5. 2D-распределения фишеровских ирисов

термы «низкий», «средний» и «высокий» с установленными по умолчанию треугольными функциями принадлежности. Взаимосвязь «входы – выход» опишем базой знаний из трех нечетких правил:

ЕСЛИ x_4 = «низкий», ТО y = «Iris Setosa»;

ЕСЛИ x_3 = «средний» И x_4 = «средний», ТО y = «Iris Versicolor»;

ЕСЛИ x_3 = «высокий» И x_4 = «высокий», ТО y = «Iris Verginica».

Логической операции И в посылках правил поставим в соответствие операцию минимума над функциями принадлежности. Для этого в FIS-редакторе выберем опцию **min** в меню **And method**. Применяя функцию fuzzy_classifier, обнаруживаем, что созданная нечеткая модель правильно классифицирует 130 из 150 ирисов. Для повышения безошибочности классификации нечеткую модель надо настроить.

Настройка нечеткого классификатора сводится к задачам оптимизации (2.14), (2.16) или (2.17). Для решения этих задач в среде MATLAB можно воспользоваться пакетом Optimization Toolbox [36]. На основе программ обучения нечетких моделей типа Мамдани (раздел 4.1) можно легко разработать сценарии настройки нечетких классификаторов, а также типовые целевые функции в соответствии с критериями (2.14), (2.16) и (2.17).

Для быстрой настройки весов правил нечеткого классификатора на основе соотношения (2.18) автором написан пакет FALEFC (**FA**st **LE**arning the **F**uzzy

Classifier). Время настройки весов правил с помощью пакета FALEFC на пор док меньше, чем при настройке функций принадлежности. Пакет FALEFC испол зует функции Fuzzy Logic Toolbox и Optimization Toolbox. Описание функци пакета приведено ниже.

fuzzy_classifier

Назначение Нечеткий вывод для задач классификации.

Синтаксис

```
decision = fuzzy_classifier(x, fis)
[decision, mf_grades] = fuzzy_classifier(x, fis, type
```

Описание Функция выполняет нечеткий вывод для задач классификаци Fuzzy_classifier вызывает функцию evalfis в формате

```
[a, b, c, d] = evalfis(x, fis),
```

что позволяет получить промежуточные результаты логическо вывода. Затем находятся правила с максимальной степенью вы полнения. Если таких правил несколько, тогда среди конкурир ющих классов выбирает тот, сумма степеней принадлежнос которому максимальна.

Функция fuzzy_classifier может иметь два или три входных арг мента:

1) `x` – входной вектор размером $1 \times n$, задающий призна классифицируемого объекта;
2) `fis` – система нечеткого вывода;
3) `type` – формат решения: 'number' – порядковый ном класса (значение по умолчанию) и 'name' – наименован класса.

Функция может возвращать два аргумента:

1) `decision` – результат классификации объекта `x`;
2) `mf_grades` – вектор степеней принадлежности объекта каждому классу (необязательный аргумент).

fast_w_learning

Назначение Типовой сценарий быстрой настройки весов правил нечетко классификатора.

Синтаксис

```
fast_w_learning
```

Описание Настраивает веса правил нечеткого классификатора, использ функцию нелинейной оптимизаци fmincon пакета Optimizati Toolbox. Высокая скорость настройки обеспечивается использ ванием RMG – матрицы степеней выполнения правил при ед ничных весовых коэффициентах, что позволяет выполнять н четкий вывод по простой формуле (2.18). Для настройки кон ретного классификатора в сценарии `fast_w_learning` нео ходимо заменить:

'fis_file_name.fis' на имя файла нечеткой системы;

'training_set_file_name' на имя файла обучающей выборки;

'testing_set_file_name' на имя файла тестовой выборки.

Критерий настройки задается переменной criterion. Допустимые значения: 1 – критерий I, соответствующий формуле (2.14); 2 – критерий II, соответствующий формуле (2.16); 3 – критерий III, соответствующий формуле (2.17).

Начальная точка оптимизации выбирается случайным образом. При многократной настройке нечеткого классификатора из разных начальных точек матрицы степеней выполнения правил при единичных весовых коэффициентах (RMG_tr и RMG_test) достаточно рассчитать один раз, а затем закомментировать строки 22 и 23 символами «%».

После настройки при помощи функции fast_cl_testing_with_rmg нечеткий классификатор тестируется на обучающей и тестовой выборках.

dp_for_cl_learning

Назначение Считывание выборки данных для настройки нечеткого классификатора.

Синтаксис

```
[INP, OUT_c, OUT_mu] = dp_for_fuzzy_cl_learning...
(fis, file_data_name)
```

Описание Считывает выборку данных и преобразовывает их к формату, требуемому программами настройки и тестирования нечеткого классификатора.

Входные аргументы функции dp_for_fuzzy_cl_learning:

1) file_data_name – имя файла данных. Формат файла file_data_name: в каждой строчке – одна пара «входы – выход»; значения входов – в первых столбцах; значение выходной переменной (номер класса) – в последнем столбце;

2) fis – система нечеткого вывода (нечеткий классификатор).

Функция возвращает три аргумента:

1) INP – матрица значений входных переменных;

2) OUT_c – вектор-столбец значений выходной переменной;

3) OUT_mu – матрица желаемых степеней принадлежностей (фаззифицированный вектор OUT_c), рассчитанных по формуле (2.15).

Пример

```
[INP, OUT_c, OUT_mu] = dp_for_fuzzy_cl_learning...
('iris_cl.fis', 'iris.dat')
%Считывание выборки данных для классификации ирисов.
```

fuzzy_classifier_rmg

Назначение Расчет матрицы степеней выполнения правил.

Синтаксис `RMG = rmg_fuzzy_cl(INP, fis)`

Описание Вычисляет `RMG` (Rule_Membership_Grades) – матрицу степеней выполнения правил при единичных весовых коэффициентах, необходимую для быстрой классификации. Размер матрицы: Количество_объектов × Количество_правил. Входные аргументы:

1) `INP` – объекты классификации. Размер матрицы `INP`: Количество_объектов × Количество_признаков;
2) `fis` – система нечеткого вывода.

rmg_fuzzy_cl

Назначение Быстрая классификация при новых значениях весов правил с использованием RMG.

Синтаксис `[decision, dmg] = fuzzy_classifier_rmg(w, RMG,... rule_order)`

Описание Выполняет быстрый нечеткий вывод для задач классификации. Предназначена для использования с программой настройки весов правил нечеткой базы знаний. Может также использоваться для быстрого тестирования нечеткого классификатора при новых весах правил. Высокая скорость классификации обеспечивается выполнением нечеткого вывода по формуле (2.18) с использованием матрицы степеней выполнения правил при единичных весовых коэффициентах (`RMG`).

Функция `fuzzy_classifier_rmg` вызывается с тремя входными аргументами:

1) `w` – вектор весов правил;
2) `RMG` – матрица степеней выполнения правил при `w = 1` для всей выборки данных. Размер матрицы: Количество_объектов × Количество_правил;
3) `rule_order` – вектор, координаты которого содержат количество правил для каждого типа решений (класса). Предполагается, что первая группа правил соответствует первому классу, вторая группа правил – второму классу, третья группа – третьему и т.д. Если `rule_order = [1 3 2]`, значит правило 1 базы знаний соответствует первому классу, правила 2–4 – второму классу, и правила 5–6 – третьему классу.

Функция `fuzzy_classifier_rmg` возвращает два выходных аргумента:

1) `decision` – вектор-столбец, содержащий номера классов решений. В качестве решения выбирается класс с максимальной степенью принадлежности. Для приграничных объектов, т.е. когда объект принадлежит нескольким классам с

максимальной степенью, в качестве решения среди конкурирующих классов выбирается тот, сумма степеней принадлежности которому максимальна;

2) dmg – (Decision Membership Grade) матрица размером Количество_объектов × Количество_решений, каждая строка которой содержит степени принадлежности объекта классам решений.

ob_fun_fast_w

Назначение Целевая функция для быстрой настройки весов нечетких правил, использующая RMG.

Синтаксис

```
delta = ob_fun_fast_w(w, rule_order, RMG, OUT_c,...
crit, OUT_mu, penalty)
```

Описание Возвращает отклонение между желаемым и действительным поведением нечеткого классификатора на обучающей выборке при новых значениях весов правил. Функция `fuzzy_classifier_rmg` вызывается с 4–7 входными аргументами:

1) `w` – вектор весов правил;

2) `rule_order` – вектор, координаты которого содержат количество правил для каждого типа решений (класса). Предполагается, что первая группа правил соответствует первому классу, вторая группа – второму, третья группа – третьему и т.д.;

3) `RMG` – матрица степеней выполнения правил (при `w = 1`) для всей выборки данных. Размер матрицы: Количество_объектов × Количество_правил;

4) `OUT_c` – вектор результатов эталонной классификации;

5) `crit` – критерий настройки: 1 – количество ошибок классификации; 2 – среднеквадратичная невязка между желаемыми и модельными степенями принадлежности; 3 – невязка между желаемыми и модельными степенями принадлежности по формуле (2.17). Значение по умолчанию – 2;

6) `OUT_mu` – вектор желаемых степеней принадлежности на выходе нечеткого классификатора (аргумент необходим при `crit = 2` или `crit = 3`);

7) `penalty` – штрафной коэффициент > 0 (значение по умолчанию равно 5).

Пример

```
delta = ob_fun_fast_w(w, rule_order, RMG, OUT_c, 1)
```

fast_cl_testing_with_rmg

Назначение Быстрое тестирование нечеткого классификатора при новых весах правил с использованием RMG.

Синтаксис

```
[decision, delta_m, delta_s, delta_p] =...
fast_cl_testing_with_rmg(w, rule_order, RMG,...
OUT_c, OUT_mu, penalty)
```

Описание

Быстрое тестирование нечеткого классификатора при новых весах правил с использованием RMG по трем критериям. Функция fast_cl_testing_with_rmg вызывается с 4–6 входными аргументами:

1) w – вектор весов правил;

2) rule_order – вектор, координаты которого содержат количество правил для каждого типа решений (класса). Предполагается, что первая группа правил соответствует первому классу, вторая группа – второму, третья группа – третьему и т.д.;

3) RMG – матрица степеней выполнения правил (при w = 1) для всей выборки данных. Размер матрицы: Количество_объектов × Количество_правил;

4) OUT_c – вектор результатов эталонной классификации;

5) OUT_mu – вектор желаемых степеней принадлежности на выходе нечеткого классификатора (необязательный аргумент);

6) penalty – штрафной коэффициент > 0 (значение по умолчанию равно 5).

Функция возвращает четыре аргумента:

1) decision – вектор-столбец, содержащий номера классов решений;

2) delta_m – процент ошибок классификации;

3) delta_s – среднеквадратичная невязка между желаемыми и модельными степенями принадлежности (формула (2.16));

4) delta_p – невязка между желаемыми и модельными степенями принадлежности по формуле (2.17).

Значения невязок также выводятся на экран.

rule_order_fuzzy_cl

Назначение

Определение количества правил в базе знаний для каждого класса.

Синтаксис

```
rule_order = rule_order_fuzzy_cl(fis)
```

Описание

Возвращает количество правил для каждого типа решений нечеткого классификатора fis. Предполагается, что первая группа правил соответствует первому решению, вторая группа – второму, третья группа – третьему. Если rule_order = [1 3 2], значит первое правило базы знаний соответствует первому классу, правила 2–4 – второму классу и правила 5–6 – третьему классу.

Проиллюстрируем использование пакета FALEFC для настройки весов правил нечеткого классификатора ирисов. Сформируем обучающую и тестовую вы-

борки. В обучающую выборку включим 120 ирисов с порядковыми номерами, некратными 5. В связи с небольшим объемом данных тестировать нечеткий классификатор будем на всей выборке. Для настройки весов правил нечеткого классификатора в типовом сценарии `fast_w_learning` укажем имена файлов системы нечеткого вывода `iris_cl.fis` и экспериментальных данных `iris.dat`. Сценарий настройки нечеткого классификатора ирисов приведен ниже.

```
%Настройка нечеткого классификатора ирисов
%Требуемые ресурсы: Fuzzy Logic Toolbox v.2.X Optimization
%Toolbox v.2.X
%FALEFC v.1.0
%$Revision: 1.3 $ $Date: 2004/04/03
%Загрузка исходной нечеткой модели:
fis = readfis('iris_c1.fis');
RULE_order = rule_order_fuzzy_cl(fis);
%Загрузка выборки данных:
[INP OUT_c OUT_mu] = dp_for_fuzzy_cl_learning(fis, 'iris.dat');
%Расчет RMG:
num_rule = length(fis.rule);
for i = 1:num_rule
    fis.rule(i).weight = 1;
end
RMG = rmg_fuzzy_cl(INP, fis);
%НАСТРАИВАЕМЫЕ ПАРАМЕТРЫ:
vlb_w(1:num_rule) = 0;                  %нижняя граница
w0(1:num_rule) = rand(1, num_rule); %начальная точка
vub_w(1:num_rule) = 1;                  %верхняя граница
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

%ПАРАМЕТРЫ ОПТИМИЗАЦИИ:
options = [];
options = optimset('Display', 'iter');
options.DiffMinChange = 0,1;
options.DiffMaxChange = 0,25;
options.LargeScale = 'off';
options.MaxIter = 20;
options.MaxFunEvals = 150;
%Формирование обучающей выборки:
tr_index = setdiff(1:150, 5:5:150);
%Оптимизация:
[wopt, delta] = fmincon(@ob_fun_fast_w, w0, [], [], [],...
[], vlb_w, vub_w, [], options, RULE_order, RMG(tr_index,...
:), OUT_c(tr_index, :), 3, OUT_mu(tr_index, :))
%Запись оптимальных весов в систему нечеткого вывода:
for i = 1:num_rule
    fis.rule(i).weight = wopt(i);
end
```

```
%Тестирование по трем критериям на обучающей выборке:
[decision, delta_m, delta_s, delta_p] =...
fast_cl_testing_with_rmg(wopt, RULE_order, RMG(tr_index, :),...
OUT_c (tr_index, :), OUT_mu(tr_index, :));
%Тестирование по трем критериям на всей выборке:
[decision, delta_m, delta_s, delta_p] =...
fast_cl_testing_with_rmg(wopt, RULE_order, RMG, OUT_c, OUT_mu)
```

В результате настройки получены такие весовые коэффициенты: 0,9 – для первого правила; 0,1 – для второго; 0,8 – для третьего. Настроенная нечеткая система правильно классифицирует 116 ирисов (96,67%) из обучающей выборки и 144 (96%) ирисов из всей выборки цветков.

4.4. НЕЧЕТКИЙ ВЫВОД ПРИ НЕЧЕТКИХ ИСХОДНЫХ ДАННЫХ

В разделе рассматривается как расширить Fuzzy Logic Toolbox для выполнения нечеткого вывода при нечетких исходных данных. Предполагается, что функции принадлежности нечетких исходных данных заданы гауссовой, колоколообразной или сигмоидной кривыми.

Для выполнения нечеткого вывода при нечетких данных необходимо уметь определять степени принадлежности входов к термам из базы знаний. Они рассчитываются по-разному при четких и нечетких входных значениях. При четких исходных данных степень принадлежности рассчитывается подстановкой текущего значения переменной в формулу функции принадлежности. При нечетких исходных данных необходимо определить степень принадлежности одного нечеткого множества $\tilde{A}$ – значения входной переменной, к другому нечеткому множеству $\tilde{B}$ – терму из базы знаний. Степень принадлежности рассчитывается как высота пересечения этих нечетких множеств (рис. 4.6).

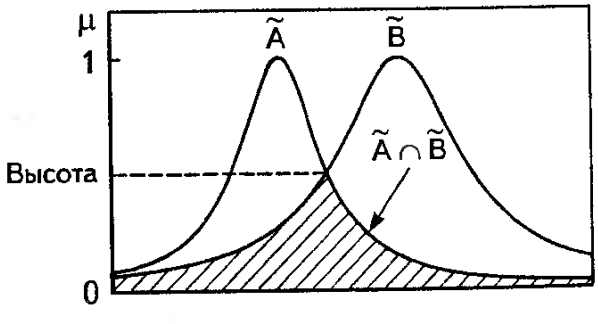


Рис. 4.6. Расчет степени принадлежности нечеткого множества нечеткому множеству

Функция fuzzy_input рассчитывает высоту пересечения двух нечетких множеств. Функции принадлежности нечетких множеств должны быть одного типа. Допустимые типы функций принадлежности: колоколообразная, гауссова и сигмоидная. Программу можно легко модифицировать под другие типы функций принадлежности. Код программы fuzzy_input приведен ниже.

```
function t = fuzzy_input(param1, param2, mftype)
%FUZZY_INPUT - рассчитывает степень принадлежности одного
%нечеткого множества к другому нечеткому множеству:
```

```
%param1 и param2 - параметры функций принадлежности
%нечетких множеств.
%Для 'gbellmf' коэффициенты крутизны функций принадлежности
%должны быть одинаковыми: param1(2) == param2(2).
%mftype - тип функции принадлежности. Допустимые типы
%функций принадлежностей: 'gaussmf', 'sigmf' и 'gbellmf'.
%Нечеткие множества должны быть заданы функциями
%принадлежности одного типа.
%Требуемые ресурсы: gaussmf.m gbellmf.m и sigmf.m из
%Fuzzy Logic Toolbox v2, Serhiy D. Shtovba,
%$Revision: 1.1 $ $Data: 2005/03/15

switch lower(mftype);
   case 'gaussmf'
       sigma1 = param1(1); c1 = param1(2);
       sigma2 = param2(1); c2 = param2(2);
       if c1 == c2 t = 1;
       else x1 = (c1*sigma2 + c2*sigma1)/(sigma2 + sigma1);
           y1 = gaussmf(x1, param1);
           if sigma1 == sigma2 y2 = y1;
           else x2 = (c1*sigma2 - c2*sigma1)/(sigma2 - sigma1);
               y2 = gaussmf(x2, param1);
           end
           t = max(y1, y2);
       end
   case 'sigmf'
       a1 = param1(1); c1 = param1(2);
       a2 = param2(1); c2 = param2(2);
       if ((a1 == a2) & (c1 == c2)) t = 1;
       elseif a1 == a2 t = 0;
       else x = (a2*c2 - a1*c1)/(a2 - a1);
          t = sigmf(x, param1);
       end
   case 'gbellmf'
          if param1(2) ~= param2(2) error ('Для gbellmf
          %коэффициенты крутизны должны быть одинаковыми:
          %param1(2) == param2(2)')
          else
              a1 = param1(1); c1 = param1(3);
              a2 = param2(1); c2 = param2(3);
              if c1 == c2 t = 1;
              else
                  x = (a2*c1 + a1*c2)/(a1 + a2);
                  y1 = gbellmf(x, param1);
                  if a1 == a2
                     y2 = 0;
                  else x = (a2*c1 - a1*c2)/(-a1 + a2);
```

```
                y2 = gbellmf(x, param1);
            end
            t = max(y1, y2);
        end
    end
otherwise
    error('Функция принадлежности должна быть gaussmf,
    %sigmf и gbellmf')
end
```

При нечетком выводе с помощью функции evalfis нужно каким-то образом указать, что входные переменные заданы нечеткими числами. Для этого нечеткое значение входной переменной кодируется следующим образом:

Fuzzy_value = FUZZY_LABEL–10*A–B,

где A – номер переменной в нечеткой системе; B – порядковый номер терма; FUZZY_LABEL – признак нечеткого аргумента, обозначенный отрицательным числом.

Если FUZZY_LABEL = –1000, тогда аргументы x ≥ –1010 рассматриваются как четкие, а аргументы x < –1010 – как нечеткие. Например, если FUZZY_LABEL = –1000, тогда значение x = –1012 соответствует входному нечеткому множеству с функцией принадлежности второго терма первой входной переменной. Таким образом, все нечеткие значения исходных данных должны быть описаны в нечеткой системе (fis-структуре). Эта fis-структура, а также константа FUZZY_LABEL должны быть доступны из m-функций, в которых запрограммированы формулы функций принадлежности. Это реализуется через глобальные переменные `FUZZY_LABEL` и `FISFI` (нечеткая система).

Запрограммированы три функции принадлежности, аргументами которых могут быть как четкие, так и нечеткие числа. Это функции:

- fsigmf – аналог сигмоидной функция принадлежности sigmf;
- fgaussmf – аналог гауссовой функции принадлежности gaussmf;
- fgbellmf – аналог колоколообразной функции принадлежности gbellmf.

Чтобы нечеткие системы могли работать с нечеткими исходными данными, необходимо в fis-файлах заменить имена функций принадлежностей входных переменных с `sigmf` на `fsigmf`, с `gaussmf` на `fgaussmf` и с `gbellmf` на `fgbellmf`. Если используются функции принадлежности других типов, то необходимо написать аналогичные fgaussmf.m функции и вставить новый блок `case` в файле `fuzzy_input.m`. Текст функции fgaussmf.m приведен ниже:

```
function y = fgaussmf(x, params)
%FGAUSSMF - гауссова функция принадлежности, по которой, в
%отличие от gaussmf, можно рассчитывать степень принадлежности
%при нечетком входном аргументе.
%Требуемые ресурсы: gaussmf.m из Fuzzy Logic Toolbox v2
%и fuzzy_input Serhiy Shtovba, %$Revision: 1.1 15-Mar-2005
```

```
global FISFI %нечеткая система
global FUZZY_LABEL %признак нечеткого аргумента, задается
%отрицательным числом <= -100. Если FUZZY_LABEL = -1000,
%тогда аргументы x >= -1010 рассматриваются как четкие, а
%аргументы x < -1010 - как нечеткие. Нечеткое значение
%кодируется так: FUZZY_LABEL-10*A-B, где A - номер переменной
%в FISFI, B - номер терма.
y = gaussmf(x, params);
if max(max(x < FUZZY_LABEL-10)) %Если есть нечеткие значения
    [ii jj] = size(x);
    for i = 1:ii
        for j = 1:jj
            if x(i,j)<(FUZZY_LABEL-10)
            %Если нечеткое значение на входе
                xx = x(i,j) - FUZZY_LABEL;
                var_index = floor(-.1*xx);
                term_index = -1*xx - var_index*10;
                mf_type = lower(FISFI.input(var_index)....
                mf(term_index).type);
                if   (strcmp(mf_type,   'gaussmf')   |...
                strcmp(mf_type, 'fgaussmf'))
                    %Степень принадлежности одного нечеткого
                    %множества другому:
                    params2 =  FISFI.input(var_index)....
                    mf(term_index).params;
                    y(i, j) = fuzzy_input(params, params2,...
                    'gaussmf');
                    else   error('Функция   принадлежности
                    %входного нечеткого числа должна быть
                    %gaussmf или fgaussmf')
                end
            end
        end
    end
end
```

Перед вызовом функции нечеткого вывода evalfis необходимо объявить глобальные переменные FUZZY_LABEL и FISFI и присвоить им соответствующие значения. Значение FUZZY_LABEL устанавливается такое, чтобы оно было меньше левой границы диапазонов изменения входных переменных нечеткой системы FISFI.

В качестве примера приведем сценарий выполнения нечеткого вывода при x_1 = Low, x_2 = Average и x_3 = Low для нечеткой системы 'sample_fuzzy.fis':

```
global FISFI
global FUZZY_LABEL
FISFI = readfis('sample_fuzzy');
```

```
FUZZY_LABEL = -1000;
evalfis([-1011 -1022 -1031], FISFI)
```

В результате получаем:

```
ans = 0,4006
```

Для входных значений x_1 = Low, x_2 = Average и x_3 = 0,2 наберем:

```
evalfis([-1011 -1022 0,2], FISFI).
```

В результате на экране получаем:

```
Warning: Some input values are outside of the specified input range.
> In C:\MATLAB7\toolbox\fuzzy\fuzzy\evalfis.m at line 73
ans = 0,5161
```

Для подавления предупреждения можно вместо функции evalfis использовать функцию evalfis_ww.

4.5. ПРОЕКТИРОВАНИЕ ИЕРАРХИЧЕСКИХ НЕЧЕТКИХ СИСТЕМ

В разделе рассматривается, как расширить Fuzzy Logic Toolbox, чтобы обеспечить проектирование нечетких иерархических систем. В иерархических системах выход одной базы знаний подается на вход для другой базы знаний (см. подраздел 1.6.8). Существует два способа создания иерархических нечетких систем. Первый способ состоит в выполнении нечеткого вывода для промежуточных переменных с последующей передачей четких значений этих переменных в нечеткие системы следующего уровня иерархии. При втором способе процедуры дефаззификации и фаззификации для промежуточных переменных не выполняются. Результат логического вывода в виде нечеткого множества напрямую передается в машину нечеткого вывода следующего уровня иерархии. Рассмотрим реализацию этих способов на основе Fuzzy Logic Toolbox.

4.5.1. ПЕРВЫЙ СПОСОБ

Для реализации первого способа необходимо вызвать функцию evalfis для каждой нечеткой базы знаний. Рассмотрим этот способ на примере реализации нечеткой экспертной системы прогнозирования конкурентоспособности марочного товара [40].

Рассматриваются марочные товары, т.е. товары, продающиеся под торговой маркой (брендом). Конкурентоспособность – одна из важных маркетинговых характеристик товара. Она выражает совокупную способность товара выдержать конкуренцию с другими товарами на определенном рынке, быть реализованным и принести прибыль. Критерием конкурентоспособности марочного товара назовем число $Q \in [0, 100]$. Чем больше значение этого критерия, тем больше шансов марочного товара быть выбранным покупателями, тем больше его сегмент рынка. На конкурентоспособность марочного товара влияет много производственных

психологических, социальных, политических и других факторов. Обозначим их через $x_1, x_2, ..., x_n$, тогда модель конкурентоспособности марочного товара будет представлять функциональное отображение вида:

$$\mathbf{X} = (x_1, x_2, \ldots, x_n) \to Q \in [0, 100],$$

где $\mathbf{X}$ – вектор влияющих факторов.

Иерархическая классификация влияющих факторов в виде дерева логического вывода показана на рис. 4.7. Элементы дерева интерпретируется так:

- корень дерева – конкурентоспособность марочного товара (Q);
- терминальные вершины – частные влияющие факторы ($x_1, \ldots, x_{10}$);
- нетерминальные вершины (двойные окружности) – свертки влияющих факторов;
- дуги графа, выходящие из нетерминальных вершин – укрупненные влияющие факторы (y_1, y_2, y_3).

Описание факторов приведено в табл. 4.2. Свертки f_Q, f_{y_1}, f_{y_2} и f_{y_3} осуществим посредством логического вывода по нечетким базам знаний.

Значения факторов будем выражать как отклонения (в процентах) от усредненных показателей по аналогичным товарам конкурирующих брендов на анализируемом рынке. Для моделирования укрупненных влияющих факторов используются экспертные нечеткие базы знаний типа Мамдани, приведенные в табл. 4.3–4.5.

Конкурентоспособность марочного товара будем моделировать с учетом трех типов сбыта, когда для потребителя показатели и цены, и качества, и имиджа, и сервиса являются: 1) плохими, 2) средними и 3) хорошими. Предполагается, что при каждом типе сбыта эластичность конкурентоспособности по факторам постоянна. Границы подобластей с постоянными эластичностями конкурентоспособности – нечеткие, что обусловлено плавным переходом одного типа сбыта в другой. В табл. 4.6 приведена нечеткая база знаний типа Сугено для моделирования конкурентоспособности марочного товара. Каждое правило этой базы знаний моделирует один тип сбыта. Коэффициенты в заключениях правил задают чувствительность конкурентоспособности по соответствующим факторам.

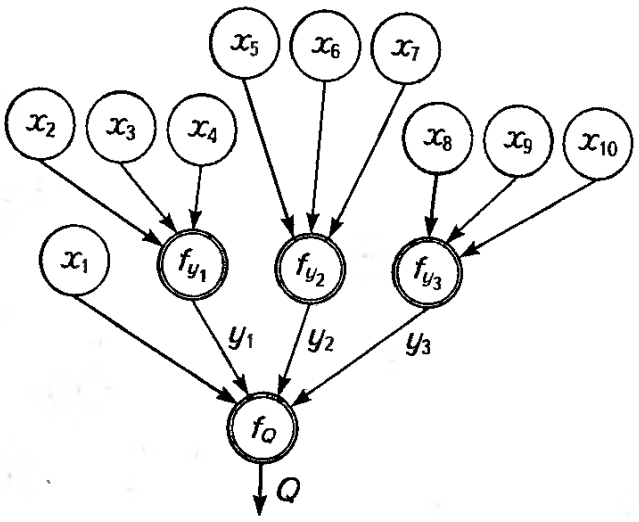


Рис. 4.7. Иерархическая классификация факторов, влияющих на конкурентоспособность

Графики функций принадлежности нечетких термов «низкий» (Н), «средний» (С) и «высокий» (В) приведены на рис. 4.8. Используется гауссова функция принадлежности.

Нечеткий вывод осуществляется алгоритмами Мамдани и Сугено. В ка-

Таблица 4

Факторы, влияющие на конкурентоспособность

Наименование фактора	Описание фактора
y_1 – качество	Совокупность потребительских свойств; способность удовлетворять ожидаемые потребности потребителя
y_2 – имидж	Целостная совокупность ассоциаций и впечатлений, представляющая торговую марку в сознании потребителя
y_3 – сервис	Множество услуг, скидок и льгот, предоставляемым реальным и потенциальным потребителям марочного товара
x_1 – цена	Розничная цена марочного товара на анализируемом рынке
x_2 – качество проектных решений	Потенциальное качество, заложенное в марочный товар. Определяется: для пищевых продуктов – рецептурой; для аппаратуры – схемотехническими решениями; для одежды – дизайном; для образовательных услуг – учебными планами
x_3 – качество производственных технологий	Объективные ограничения достижения потенциального качества. Для товаропроизводителей они обусловлены технологическим процессом, средствами труда (оборудование, инструменты) и предметами труда (комплектующие, сырье, ингредиенты). Для учебных заведений эти ограничения обусловлены лабораторной базой и учебным процессом (расписание, технологии передачи и контроля знаний и т.п.)
x_4 – качество персонала	Субъективные ограничения достижения потенциального качества, обусловленные квалификацией, дисциплинированностью и мотивированностью персонала
x_5 – ранг производителя	Мера доверия к производителю марочного товара, обычно определяемая государственными органами сертификации. Например, потребители больше доверяют товарам, произведенным концернами, заводами и фабриками согласно ГОСТу или ISO, чем частными предпринимателями, выпускающими продукцию по ТУ. Для вузов ранг определяется уровнем аккредитации и статусом (национальный университет, университет, институт или филиал)
x_6 – рекламное обеспечение	Информация, распространяющаяся в интересах производителя марочного товара. Состоит из рекламного обеспечения всего бренда и конкретного марочного товара. Определяется идентичностью бренд (имя, товарный знак, история и т.п.), а также объемом и качеством прямой и скрытой реклам
x_7 – уровень рекламаций	Информация, распространяющаяся не в интересах производителя марочного товара. Состоит из рекламаций на конкретный марочный товар и весь бренд. Определяется количеством и степенью претензий потребителей, уровнем распространения информации о рекламациях, а также контрпропагандой конкурентов
x_8 – удобство покупки	Легкость совершения покупки, определяемая географической и временной доступностью точек продажи, а также сервисным обслуживанием при приобретении товара (консультации, доставка и установка)
x_9 – сервис при эксплуатации	Удобства при эксплуатации марочного товара, определяемые возможностью модернизации товара, сроком гарантийного и послегарантийного обслуживания, географической и временной доступностью сервисных центров и точек сбыта расходных материалов
x_{10} – бонусы	Дополнительные льготы, скидки и услуги, которые могут использовать потребители марочного товара

Таблица 4.3

Нечеткая база знаний для моделирования качества марочного товара

x_2	x_3	x_4	y_1
Высокое	Высокое	Высокое	Высокое
Высокое	Высокое	Среднее	Высокое
Высокое	Среднее	Высокое	Высокое
Среднее	Высокое	Высокое	Высокое
Среднее	Высокое	Среднее	Высокое
Низкое	Низкое	Низкое	Низкое
Низкое	Низкое	Среднее	Низкое
Низкое	Среднее	Низкое	Низкое
Среднее	Низкое	Низкое	Низкое
Среднее	Низкое	Среднее	Низкое
Высокое	Низкое	Среднее	Среднее
Высокое	Среднее	Низкое	Среднее
Низкое	Высокое	Среднее	Среднее
Низкое	Среднее	Высокое	Среднее
Среднее	Высокое	Низкое	Среднее
Среднее	Низкое	Высокое	Среднее
Среднее	Среднее	Среднее	Среднее

Таблица 4.4

Нечеткая база знаний для моделирования имиджа марочного товара

x_5	x_6	x_7	y_2
Высокий	Высокое	Средний	Высокий
Высокий	Среднее	Низкий	Высокий
Любой	Высокое	Низкий	Высокий
Средний	Высокое	Средний	Высокий
Любой	Низкое	Высокий	Низкий
Низкий	Низкое	Средний	Низкий
Низкий	Среднее	Высокий	Низкий
Средний	Низкое	Средний	Низкий
Высокий	Низкое	Средний	Средний
Высокий	Среднее	Высокий	Средний
Низкий	Высокое	Средний	Средний
Низкий	Среднее	Низкий	Средний
Средний	Высокое	Высокий	Средний
Средний	Низкое	Низкий	Средний
Средний	Среднее	Средний	Средний

Таблица 4.5

Нечеткая база знаний для оценки уровня сервиса

x_8	x_9	x_{10}	y_3
Высокая	Высокий	Высокие	Высокий
Высокая	Высокий	Средние	Высокий
Высокая	Средний	Высокие	Высокий
Высокая	Средний	Средние	Высокий
Средняя	Высокий	Высокие	Высокий
Низкая	Низкий	Низкие	Низкий
Низкая	Низкий	Средние	Низкий
Низкая	Средний	Низкие	Низкий
Низкая	Средний	Средние	Низкий
Средняя	Низкий	Низкие	Низкий
Высокая	Низкий	Средние	Средний
Высокая	Средний	Низкие	Средний
Низкая	Высокий	Средние	Средний
Низкая	Средний	Высокие	Средний
Средняя	Высокий	Низкие	Средний
Средняя	Низкий	Высокие	Средний
Средняя	Средний	Средние	Средний

Таблица 4.6

Нечеткая база знаний для оценки конкурентоспособности

x_1	y_1	y_2	y_3	Q
Высокая	Низкое	Низкий	Низкий	$-0{,}08x_1+0{,}03y_1+0{,}25y_2+0{,}055y_3+14$
Средняя	Среднее	Средний	Средний	$-0{,}35x_1+0{,}4y_1+0{,}28y_2+0{,}05y_3+50$
Низкая	Высокое	Высокий	Высокий	$-0{,}06x_1+0{,}06y_1+0{,}06y_2+0{,}08y_3+80$

честве треугольной нормы выбрано умножение. При нечетком выводе Мамдани дефаззификация осуществляется по методу центра тяжести, а при выводе Сугено – по методу взвешенного среднего.

Нечеткая модель конкурентоспособности реализована четырьмя системами нечеткого вывода:

- q_y1.fis – нечеткая система моделирования качества марочного товара (y_1);
- q_y2.fis – нечеткая система моделирования имиджа марочного товара (y_2);
- q_y3.fis – нечеткая система моделирования сервиса, ассоциированного с марочным товаром (y_3);
- q_q.fis – нечеткая система прогнозирования конкурентоспособности марочного товара (Q).

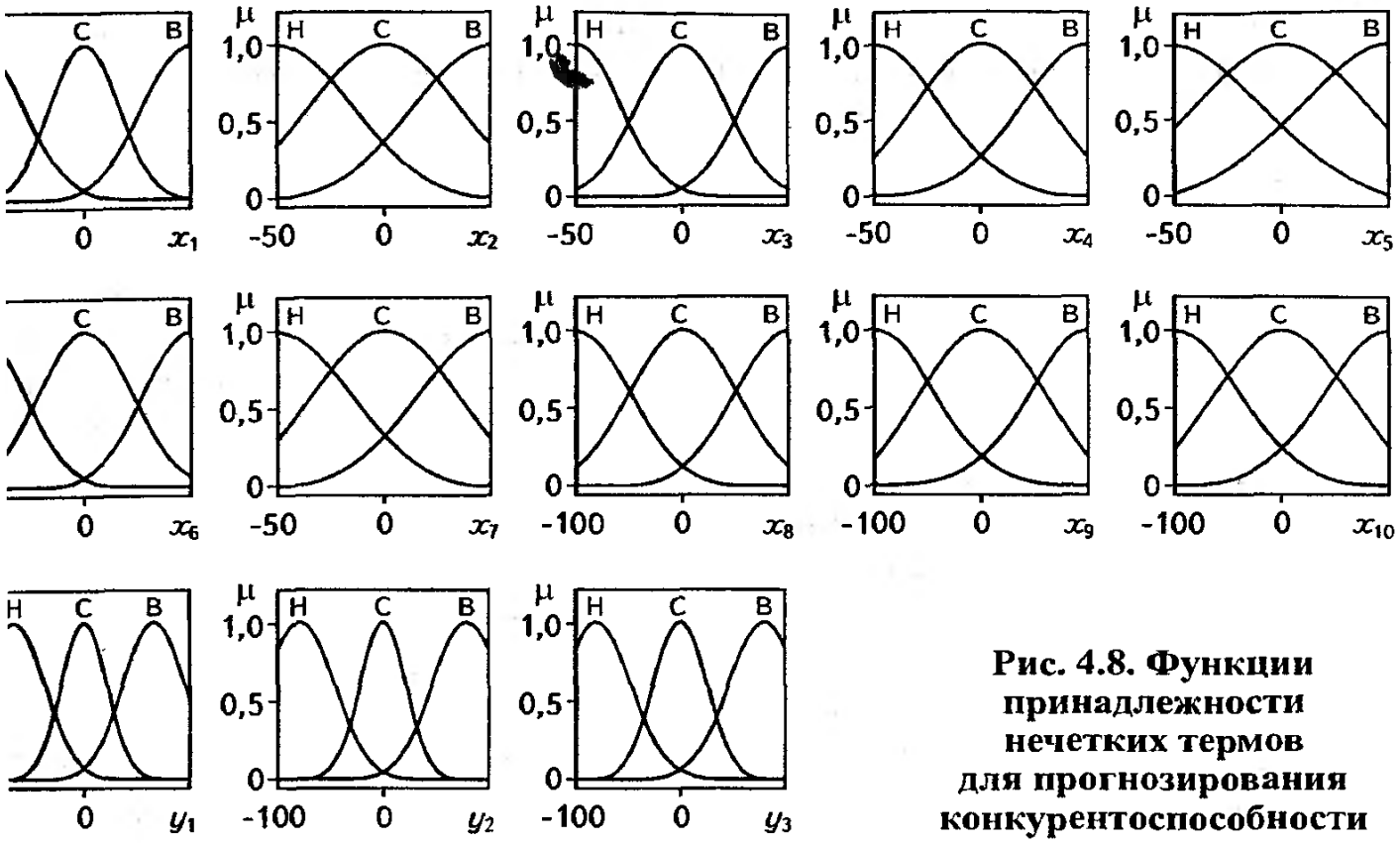


Рис. 4.8. Функции принадлежности нечетких термов для прогнозирования конкурентоспособности

Иерархический нечеткий вывод по дереву на рис. 4.7 осуществляется функ-
conc.m. Функция может возвращать два выходных аргумента: первый аргу-
– результат прогнозирования конкурентоспособности марочного товара;
й аргумент – значения укрупненных влияющих факторов y_1, y_2 и y_3. Функ-
ызывается с 10 входными аргументами, которые задают значения факторов
$_{10}$. Значения факторов $x_2 \div x_{10}$ можно задавать как числами, так и термами:
низкий; 'НС' – ниже среднего; 'С' – средний; 'ВС' – выше среднего; 'В' –
сий. Для логического вывода при нечетких входных данных используется
ция qgaussmf, модифицированная из гауссовой функции принадлежности
mf. Расчет степени принадлежности одного нечеткого множества к другому
кому множеству осуществляется функцией qual_inp_gauss. Логический вы-
роисходит через функцию evalfis_ww, которая аналогична функции evalfis,
выдает предупреждений при использовании нечетких входных данных.

ассмотрим использование нечеткой экспертной системы на следующих при-
. Показатели марочного товара A на некотором региональном рынке экспер-
енили так: x_1 = 10%; x_2 = «высокое»; x_3 = «среднее».

Показатели марочного товара A на региональном рынке г.Винница эксперты
или так (данные на сентябрь 2004 г.): x_1 = 10%; x_2 = «высокое»; x_3 = «среднее»;
среднее»; x_5 = «средний»; x_6 = –50%; x_7 = –40%; x_8 = –30%; x_9 = «средний» и
–80%.

Для расчета конкурентоспособности с помощью предлагаемой системы не-
имо набрать следующую команду:

```
P = conc(10, 'B', 'C', 'C', 'C', -50, -40, -30, 'C', -80)
```

В результате получаем, что конкурентоспособность этого марочного товара A является средней, с числовым значением: $Q_P = 51{,}75$. Выполнение нечетких выводов Мамдани для этого примера иллюстрирует рис. 4.9.

Показатели марочного товара $Б$ на том же рынке эксперты оценили так: $x_1 = 40\%$; x_2 = «высокое»; $x_3 = 25\%$; x_4 = «высокое»; x_5 = «высокий»; $x_6 = 70\%$; $x_7 = -20\%$; $x_8 = 80\%$; x_9 = «средний» и $x_{10} = -50\%$.

Наберем команду:

```
QN = conc(40, 'B', 25, 'B', 'B', 70, -20, 80, 'C', -50)
```

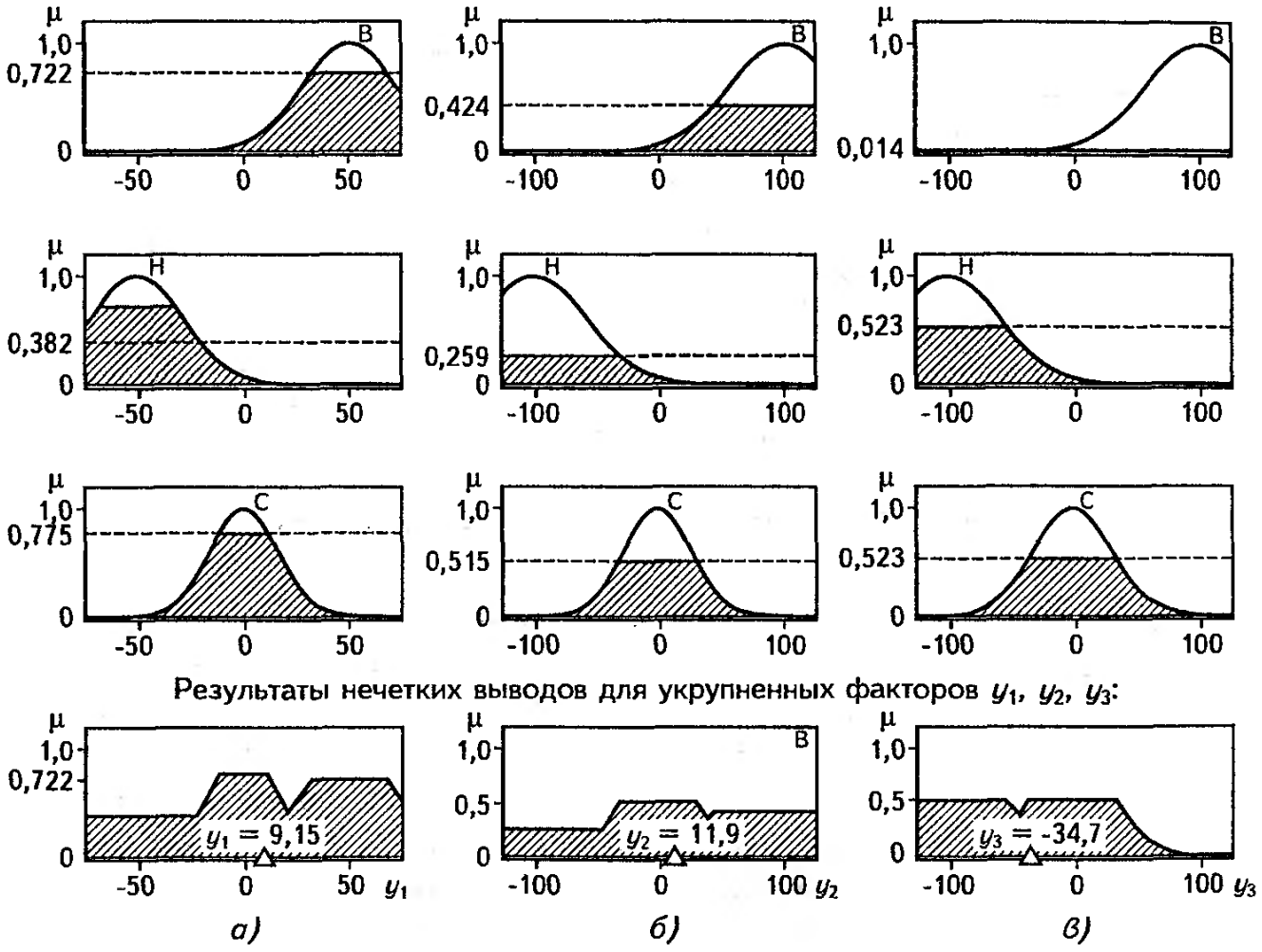


Рис. 4.9. Прогнозирование укрупненных факторов y_1, y_2 и y_3 для товара A

а – качество; *б* – имидж; *в* – сервис

В результате моделирования получаем, что конкурентоспособность товара $Б$: $Q_N = 62{,}41$. Конкурентоспособность товара A равна $Q_P = 51{,}75$. Следовательно, в рассматриваемом регионе рыночные сегменты этих товаров находятся в соотношении: $\beta_N : \beta_P = Q_N : Q_P = 62{,}41 : 51{,}75 = 1{,}21 : 1$.

Предположим, что для подъема конкурентоспособности товара A до уровня $Q_P^* = 63$ (т.е. большего, чем у конкурента) менеджер может изменять цену и уровень рекламы в таких диапазонах: $x_1 \in [-20, 20]$ и $x_6 \in [-55, 10]$. Применяя предлагаемую нечеткую экспертную систему, построим график зависимости конкурен-

пособности товара A от факторов x_1 и x_6 (рис. 4.10). Обеспечить желаемую курентоспособность $Q_P^* = 63$ можно снижением цены и увеличением уровня ламы. Возможные варианты заданного увеличения конкурентоспособности азаны на рис. 4.10 изолинией с маркером «63».

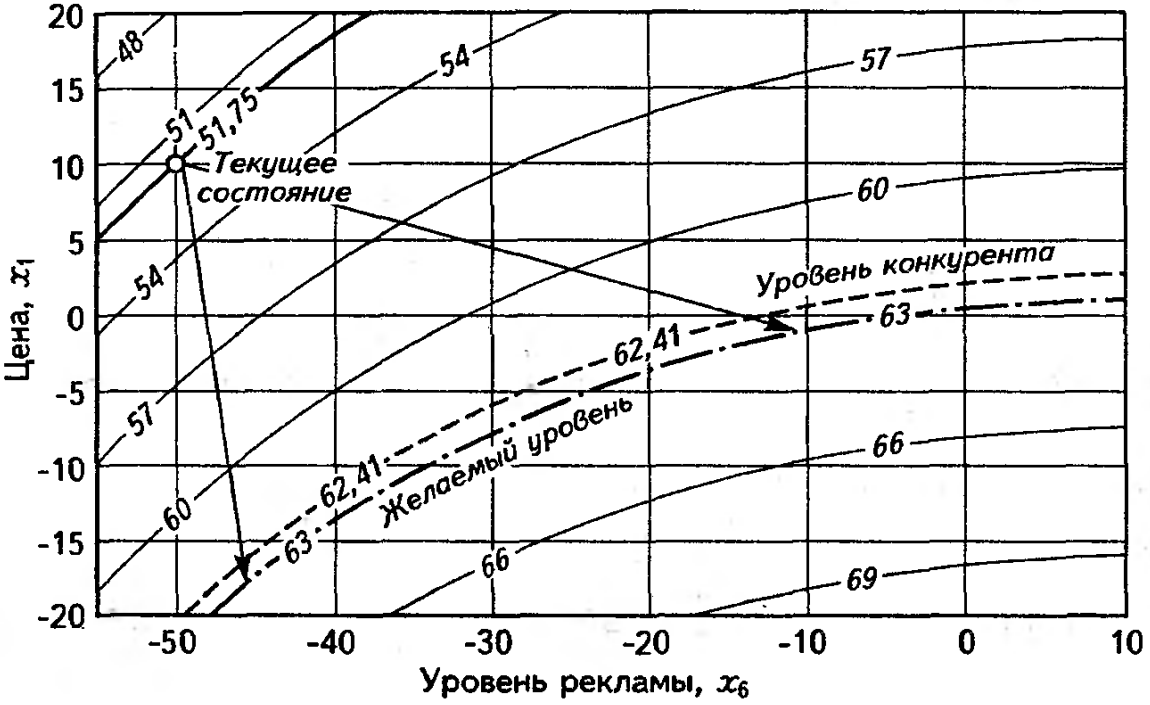


Рис. 4.10. Изолинии конкурентоспособности товара *A*

Модель конкурентоспособности марочного товара построена на основе толь- кспертных знаний, поэтому возможны несовпадения результатов нечеткого ода (теория) с экспериментальными данными. Для обеспечения достоверных льтатов необходимо провести параметрическую идентификацию нечеткой ели по экспериментальным данным маркетинговых исследований. В нечеткой ели настраивают параметры функций принадлежности термов а также коэф- ициенты в заключениях правил в базе знаний Сугено.

4.5.2. ВТОРОЙ СПОСОБ

Недостаток первого способа построения иерархических нечетких систем со- ит в том, что над промежуточными переменными последовательно выполня- я операции дефаззификации и фаззификации. Нечеткие результаты промежу- ых логических выводов дефаззифицируют, потом эти четкие значения пода- на вход нечетких систем следующего уровня иерархии и там они фаззифици- ться, т.е. снова становятся нечеткими. Следовательно, для промежуточных еменных надо задать функции принадлежности. Кроме того, необходимо спечить эквивалентность нечетких множества до и после операций дефаззи- ации и фаззификации. При втором способе процедуры дефаззификации и ификации для промежуточных переменных выполнять нет необходимости.

Результат логического вывода в виде нечеткого множества напрямую передается в машину нечеткого вывода следующего уровня иерархии. Поэтому для описания промежуточных переменных в иерархических нечетких базах знаний достаточно задать только терм-множества, без определения функций принадлежностей.

Рассмотрим, как расширить Fuzzy Logic Toolbox, чтобы выполнять логический вывод по иерархической нечеткой базе знаний по второму способу, т.е. без лишних дефаззификации и фаззификации промежуточных переменных. Для этого необходимо запрограммировать две процедуры. Первая процедура должна выдавать результат вывода по промежуточной базе знаний в виде нечеткого множества типа

$$\tilde{y} = \left(\frac{\mu_{d_1}(y)}{\tilde{d}_1}, \frac{\mu_{d_2}(y)}{\tilde{d}_2}, \ldots, \frac{\mu_{d_m}(y)}{\tilde{d}_m} \right),$$

где $\mu_{d_j}(y)$ – степень принадлежности результата нечеткому терму $\tilde{d}_j$, $j = \overline{1, m}$; m – количество термов.

Вторая процедура должна передавать найденные степени принадлежности $\mu_{d_j}(y)$, $j = \overline{1, m}$ в машину нечеткого вывода следующего уровня иерархии.

Для первой процедуры воспользуемся приемами, заложенными в нечеткий классификатор. В качестве промежуточной нечеткой системы будем использовать систему нечеткого вывода Сугено. Классам решений поставим в соответствие «функции принадлежности» выходной переменной. Параметры заключений правил (параметры «функций принадлежности» выходной переменной) могут быть произвольными, так как они не влияют на результирующее нечеткое множество. Для получения результата вывода в виде нечеткого множества выполним следующие команды:

```
%Нечеткий вывод:
[a, b, c, d] = evalfis(inputs,fis);
%inputs - входы, fis - нечеткая система
%Вытаскивание степеней принадлежности:
number_of_terms = length(fis.output(1).mf);
%Количество термов:
mf_grades = zeros(1, number_of_terms);
%Искомые степени принадлежности:
number_of_rules = length(fis.rule);
%Количество правил в базе знаний:
for j = 1:number_of_rules
    term_index = fis.rule(j).consequent;
    mf_grades(term_index) = max(mf_grades(term_index), d(j));
end
```

Для реализации второй процедуры предлагается специальная функция принадлежности treemf, которая возвращает степень принадлежности, равную ее первому параметру:

```
function mu = treemf(x, params)
mu = x; %Чтобы не определять размер
mu(:, :) = params(1).
```

Этот тип функции принадлежности следует установить для каждого терма ходной промежуточной переменной. Для передачи в машину нечеткого вывода ледующего уровня иерархии степеней принадлежности, найденных первой про-едурой, необходимо первый параметр функции принадлежности treemf устано-ить равным соответствующей координате вектора `mf_grades`. Параметры ункций принадлежности входных промежуточных переменных необходимо станавливать при каждом нечетком выводе.

Для логического вывода по иерархической базе знаний предлагается пакет uzzy_tree. Он включает функции treemf, prepare_tree и hier_evalfis. Для работы акета необходим Fuzzy Logic Toolbox. Для нечетких иерархических классифика-оров необходима также функция fuzzy_classifier из пакета FALEFC. Описание ункций пакета Fuzzy_tree приведено ниже.

treemf

азначение Функция принадлежности для промежуточных входных переменных в иерархических системах нечеткого вывода.

интаксис `mu = treemf(x, params)`

писание Возвращает степень принадлежности, равную первой координате входного аргумента `params`.

prepare_tree

азначение Преобразовывает дерево нечеткого вывода для функции hier_evalfis.

интаксис
```
[FIS_list, input_list] = prepare_tree...
(fis_file_names, tree_list)
```

писанне Преобразовывает дерево нечеткого вывода к формату, требуемому функцией hier_evalfis. Функцию следует один раз запустить для каждой иерархической базы знаний до вызова функции hier_evalfis. Функция устанавливает для всех термов нетерминальных входных переменных тип функции принадлежности 'treemf'.

Функция prepare_tree вызывается с двумя входными аргументами:

1) `fis_file_names` – {список fis-файлов}, составляющих иерархическую нечеткую базу знаний. Сначала нечеткий вывод осуществляется по первой нечеткой системе, затем – по второй и т.д.;

2) `tree_list` – матрица связей между базами знаний. Размер матрицы $(N–1)\times 2$, где N – количество баз знаний. Первая строка матрицы указывает, куда входит выходная переменная первой базы знаний, вторая строка указывает точку входа выходной переменной второй базы знаний и т.д. Формат: первый элемент строки – порядковый номер «принимаю-

щей» базы знаний из `fis_file_names`; второй элемен строки – порядковый номер входной переменной в «прини мающей» базе знаний. В базах знаний номера терминальны входных переменных должны следовать до нетерминальны (т.е. исходящих из других баз знаний).

Функция возвращает два аргумента:

1) `FIS_list` – массив нечетких систем. Последний элемен массива – нечеткая система верхнего уровня иерархии. Перва система не содержит нетерминальных входных переменных

2) `input_list` – матрица размером *N*×3. Показывает рас пределение терминальных входных переменных по нечетки системам. Каждая строка описывает одну нечеткую систему Первый столбец – общее количество входов системы; второ столбец – количество терминальных входов; третий столбец порядковый номер первой терминальной переменной.

hier_evalfis

Назначение Нечеткий вывод по иерархической базе знаний.

Синтаксис

```
[a, b, c, d] = hier_evalfis(x, FIS_list,...
input_list, tree_list, system_type)
```

Описание Осуществляет нечеткий вывод по иерархической базе знаний Перед вызовом функции hier_evalfis следует один раз запустит для каждой иерархической базы знаний функцию prepare_tree. Функция hier_evalfis имеет пять входных аргументов:

1) `x` – вектор значений входов (терминальных переменных)

2) `FIS_list` – массив нечетких систем. Последний элемен массива – нечеткая система верхнего уровня иерархии Первая система не содержит нетерминальных переменны Формируется функцией prepare_tree;

3) `input_list` – матрица размером *N*×3. Показывает рас пределение терминальных входных переменных по нечет ким системам. Каждая строка описывает одну нечеткую сис тему. Первый столбец – количество входов; второй столбец количество терминальных входов; третий столбец – порядко вый номер первой терминальной переменной, представлен ной вектором `x`. Формируется функцией prepare_tree;

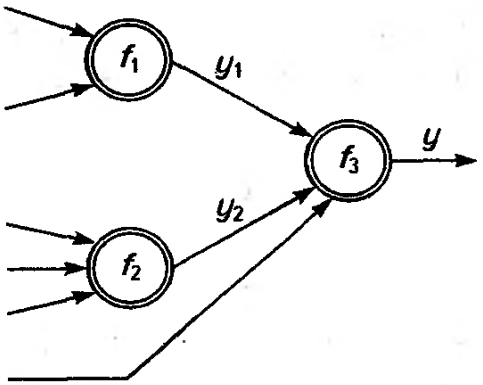
4) `tree_list` – матрица связей между базами знаний. Ра мер матрицы (*N*–1)×2, где *N* – количество баз знаний. Перва строка матрицы указывает, куда входит выходная перемен ная первой базы знаний, вторая строка указывает точку вход выходной переменной второй базы знаний и т.д. Форма первый элемент строки – номер «принимающей» базы зна ний; второй элемент строки – порядковый номер входной пе ременной в «принимающей» базе знаний. В базах знани

порядковые номера терминальных входных переменных должны быть меньше номеров нетерминальных (выходящих из других баз знаний);

5) `system_type` – тип нечеткой системы верхнего уровня иерархии: 1 – непрерывный выход, 2 – дискретный выход. Системы с непрерывным выходом должны быть системами Сугено нулевого порядка или системами Мамдани.

При `system_type == 1` функция hier_evalfis может возвращать до четырех аргументов: `a` – результат нечеткого вывода по иерархической базе знаний; `b`, `c` и `d` – промежуточные результаты вывода Мамдани по базе знаний верхнего уровня иерархии. Все аргументы аналогичны выходным переменным функции evalfis.

При `system_type == 2` функция hier_evalfis может возвращать до трех аргументов: `a` – результат нечеткого вывода по иерархической базе знаний в виде номера класса; `b` – результат нечеткого вывода по иерархической базе знаний в виде наименования класса; `c` – вектор степеней принадлежности каждому классу.



4.11. Пример иерархической нечеткой системы

Рассмотрим применение пакета Fuzzy_tree на примере иерархической системы (рис. 4.11). Нечеткие базы знаний приведены в табл. 4.7–4.9.

Таблица 4.7

Нечеткая база знаний о зависимости $y_1 = f_1(x_1, x_2)$

x_1	x_2	y_1
Низкий	Средний	Низкий
Низкий	Низкий	Средний
Высокий	Высокий	Высокий
Высокий	Средний	Средний

Таблица 4.8

Нечеткая база знаний о зависимости $y_2 = f_2(x_3, x_4, x_5)$

x_3	x_4	x_5	y_2
Низкий	Низкий	Низкий	Низкий
Средний	Низкий	Низкий	Низкий
Средний	Высокий	Низкий	Средний
Низкий	Средний	Высокий	Средний
Высокий	Высокий	Высокий	Высокий

Таблица 4.9

Нечеткая база знаний о зависимости $y = f_3(x_6, y_1, y_2)$

x_6	y_1	y_2	y
Низкий	Низкий	Низкий	Низкий
Низкий	Средний	Средний	Ниже среднего
Низкий	Высокий	Средний	Средний
Высокий	Средний	Высокий	Выше среднего
Высокий	Высокий	Высокий	Высокий

Нечеткие системы $y_1 = f_1(x_1, x_2)$, $y_2 = f_2(x_3, x_4, x_5)$ и $y = f_3(x_6, y_1, y_2)$ записаны в файлах hy1.fis, hy2.fis и hy.fis.

Необходимо найти значение выходной переменной y, если входные переменные принимают такие значения:

Для этого напишем следующую программу:

x_1	x_2	x_3	x_4	x_5	x_6
1	2	9	5	2	5
2	5	7	4	4	8
3	7	2	2	4	1

```
fis_file_names = {'hy1.fis' 'hy2.fis' 'hy.fis'};
%Список fis-файлов:
tree_list = [3 2; 3 3];
%Матрица связей между базами знаний:
[FIS_list, input_list] =...
prepare_tree(fis_file_names, tree_list)
X1 = [1 2 9 5 2 5]
%Значения входных (терминальных) переменных:
out1 = hier_evalfis(X1, FIS_list, input_list, tree_list, 1)
X2 = [2 5 7 4 4 8]
out2 = hier_evalfis(X2, FIS_list, input_list, tree_list, 1)
X3 = [3 7 2 2 4 1]
out3 = hier_evalfis(X3, FIS_list, input_list, tree_list, 1)
[out1 out2 out3]
```

В результате получаем такие значения выходной переменой y:

```
ans =
    4.9704   2.4302   3.0829
```

Заключение

Автор надеется, что в книге читатель нашел ответы на следующие вопросы:

- Что такое нечеткие множества, нечеткие отношения и нечеткий вывод?
- Что такое нечеткая система?
- Какие преимущества применения нечеткой логики в проектировании систем?
- Как идентифицировать нелинейные зависимости нечеткими базами знаний на основе экспертно-экспериментальной информации?
- Как получать нечеткие правила из экспериментальных данных?
- Как принимать решения в нечетких условиях на основе парных сравнений?
- Как обобщать аналитические модели на случай нечетких исходных данных?
- Как проектировать нечеткие системы с помощью MATLAB?
- Как обеспечить взаимодействие Fuzzy Logic Toolbox с Simulink, Optimization Toolbox и другими пакетами расширения вычислительной среды MATLAB?
- Как расширить Fuzzy Logic Toolbox под свои запросы?

Приведенные в книге теоретические сведения и практические примеры раскрывают наиболее разработанные в инженерном аспекте методы проектирования нечетких систем. Для углубленного изучения теории нечетких множеств и ее применения в Приложении приведен список интернет-ресурсов по нечетким системам.

Перспективным направлением развития нечеткой теории, по мнению автора, является гибридизация нечетких моделей с методами природных вычислений: нейронными сетями, ДНК-компьютингом, генетическими алгоритмами и эволюционным программированием, муравьиными алгоритмами и роевым интеллектом. Что касается приложений, то вскоре следует ожидать переключения внимания с нечеткого управления техническими объектами на принятие решений на основе нечеткого вывода в медицине, биологии, социологии, экономике, политике, спорте, религии и в других неинженерных областях.

Литература

1. **Беллман Р., Заде Л.** Принятие решений в расплывчатых условиях. В кн.: Вопросы анализа и процедуры принятия решений. – М.: Мир, 1976. – С. 172–215.

2. **Борисов А.Н., Крумберг О.А., Федоров И.П.** Принятие решений на основе нечетких моделей: примеры использования. – Рига: Зинатне, 1990. – 184 с.

3. **Дуда Р., Харт П.** Распознавание образов и анализ сцен. – М.: Мир, 1976. – 511 с.

4. **Заде Л.** Понятие лингвистической переменной и ее применение к принятию приближенных решений. – М.: Мир, 1976. – 167 с.

5. **Зараковский Г.М., Королев Б.А., Медведев В.И., Шлаен П.Я.** Введение в эргономику / Под ред. Зинченко В.П. – М.: Советское радио, 1974. – 352 с.

6. **Митюшкин Ю.И., Мокин Б.И., Ротштейн А.П.** Soft-Computing: идентификация закономерностей нечеткими базами знаний. – Винница: УНІВЕРСУМ-Вінниця, 2002. – 145 с.

7. **Орловский С.А.** Проблемы принятия решений при нечеткой исходной информации. – М.: Радио и связь, 1981. – 286 с.

8. **Панкевич О.Д.** Экспертная система диагностики трещин кирпичных конструкций // Строительные конструкции. – 2000. – Вып. 52. – С. 422–429.

9. **Панкевич О.Д., Маевская И.В.** Определение причин появления трещин кирпичных конструкций на основе нечетких баз знаний // Известия вузов: Строительство. – 2002. – №1–2. – С. 4–8.

10. **Панкевич О.Д., Штовба С.Д.** Диагностирование трещин строительных конструкций с помощью нечетких баз знаний. – Винница: УНІВЕРСУМ-Вінниця, 2005. – 108 с. (На укр. языке). www.vinnitsa.com/shtovba/doc/mono_@.rar.

11. **Ракитянская А.Б., Ротштейн А.П.** Прогнозирование результатов футбольных матчей с помощью нечетких моделей с генетической и нейронной настройками // Известия РАН. Теория и системы управления. – 2005. – №1. – С. 110–119.

12. **Ротштейн А.П.** Интеллектуальные технологии идентификации: нечеткая логика, генетические алгоритмы, нейронные сети. – Винница: УНІВЕРСУМ-Вінниця, 1999. – 320 с.

13. **Ротштейн А.П., Кательников Д.И.** Идентификация нелинейных зависимостей нечеткими базами знаний // Кибернетика и системный анализ. – 1998. – №5. – С. 53–61.

14. **Ротштейн А.П., Митюшкин Ю.И.** Извлечение нечетких правил из экспериментальных данных с помощью генетических алгоритмов // Кибернетика и системный анализ. – 2001. – №3. – С. 45–53.

15. **Ротштейн А.П., Штовба С.Д.** Влияние методов дефаззификации на скорость настройки нечеткой модели // Кибернетика и системный анализ. – 2002. – №5. – С. 169–176.

16. **Ротштейн А.П., Штовба С.Д.** Нечеткий многокритериальный анализ вариантов с применением парных сравнений // Известия РАН. Теория и системы управления. – 2001. – №3. – С. 150–154.

17. **Ротштейн А.П., Штовба С.Д.** Проектирование нечетких баз знаний: лабораторный практикум и курсовое проектирование: Учебное пособие. – Винница: Винницкий государственный технический университет, 1999. – 65 с. (На укр. языке).

18. **Ротштейн А.П., Штовба С.Д., Штовба Е.В.** Многокритериальный выбор бренд-проекта с помощью нечетких парных сравнений альтернатив // Управление проектами и программами. – 2006. – №2. – С. 138–146.

19. **Саати Т.Л.** Взаимодействие в иерархических системах // Техническая кибернетика. – 1979. – №1. – С. 68–84.

20. **Штовба С.Д.** Идентификация нелинейных зависимостей с помощью нечеткого логического вывода в системе MATLAB // Exponenta Pro: Математика в приложениях. – 2003. – №2. – С. 9–15.

21. **Штовба С.Д.** Классификация объектов на основе нечеткого вывода // Exponenta Pro: Математика в приложениях. – 2004. – №1. – С. 68–69.

22. **Штовба С.Д., Панкевич О.Д.** Проектирование нечетких классификаторов в системе MATLAB. Труды Всероссийской научной конференции «Проектирование научных и инженерных приложений в среде MATLAB». – М., 2004. – С. 1318–1335. http://matlab.exponenta.ru/conf2004/section_4_.zip.

23. **Цыпкин Я.З.** Основы информационной теории идентификации. – М.: Наука. – 1984. – 320 с.

24. **Babuska R.** Fuzzy Modeling for Control. Boston: Kluwer Academic Publishers.– 1998.

25. **Bellman R.E., Zadeh L.A.** Decision–Making in Fuzzy Environment // Management Science. – 1970. – Vol. 17. – №4. – P. 141–160.

26. **Bezdek J.C.** Pattern Recognition with Fuzzy Objective Function. – New York: Plenum Press., 1981.

27. **Fuzzy** Logic Toolbox. User's Guide, Version 2.1 The MathWorks, Inc., 2001.

28. **Holmblad L.P., Osregaard J.J.** Control of Cement Kiln by Fuzzy Logic. In «Approximate Reasoning in Decision Analysis» (Eds.: Gupta M.M. and Sanchez E.): Amsterdam, New York, Oxford. – 1982. – P. 389–400.

29. **Gustafson D.E., Kessel W.C.** Fuzzy Clustering with a Fuzzy Covariance Matrix. Proc. of IEEE CDC, San–Diego, USA. – P. 761–766.

30. **Jang J.-S. R.** ANFIS: Adaptive–Network–Based Fuzzy Inference System // IEEE Trans. Systems & Cybernetics. – 1993. – Vol. 23. – P. 665–685.

31. **Kosko B.** Fuzzy Systems as Universal Approximators // IEEE Trans. on Computers. – 1994. – Vol. 43. – №11. – P. 1329–1333.

32. **Ljung L.** System Identification, Theory for the User. Prentice–Hall. – 1987.

33. **Mamdani E.H., Assilian S.** An Experiment in Linguistic Synthesis with Fuzzy Logic Controller // Int. J. Man–Machine Studies. – 1975. – Vol. 7. – №1. – P. 1–13.

34. **Miller G.A.** The Magic Number Seven Plus or Minus Two: Some Limits on Our Capacity for Processing Information // Psychological Review. – 1956. – № 63. – P. 81–97.

35. **Nauck D., Klawonn F., Kruse R.** Foundations of Neuro–Fuzzy Systems. John Wiley & Sons. – 1997. – 305 p.

36. **Optimization** Toolbox. User's Guide, Version 2. The MathWorks, Inc., 1999.

37. **Rotshtein A.** Design and Tuning of Fuzzy Rule–Based System for Medical Diagnosis. In Fuzzy and Neuro–Fuzzy Systems in Medicine (Eds.: Teodorescu N.H., Kandel A., and Jain L.C.). USA, Boca–Raton: CRC–Press. – 1998. – P. 243–289.

38. **Rutkovska D., Pilinski M., Rutkowski L.** Sieci Neurronowe, Algoritmy Genetyczne i Systemy Rozmyte. Warszawa: PWN. – 1999. – 410 p.

39. **Shtovba S., Rotshtein A., Pankevich O.** Fuzzy Rule Based System for Diagnosis of Stone Construction Cracks of Buildings. In «Advances in Computational Intelligence and Learning, Methods and Applications» (Eds.: Zimmermann H–J., Tselentis G., van Someren M., Dounias G.). Kluwer Academic Publishers: Dordrecht, 2002. – 401–412 pp.

40. **Shtovba S., Shtovba O.** Prediction the Competitive Strength Index of Brand Product on Base of Fuzzy Logic. In Proc. of Workshop «Logic–Based Knowledge Representation», Dresden, Germany. – 2005. www.computational–logic.org/content/events/iccl–ss–2005/ talks/SerhiyShtovba.pdf.

41. **Takagi T., Sugeno M.** Fuzzy Identification of Systems and Its Applications to Modeling and Control // IEEE Trans. on Systems, Man, and Cybernetics. – 1985. – Vol. 15. – №1. – P. 116–132.

42. **Xei X.L., Beni G.A.** Validity Measure for Fuzzy Clustering // IEEE Trans. on Pattern Anal. and Machine Intell. 3 (8). – 1991. – P. 841–846.

43. **Yager R., Filev D.** Essentials of Fuzzy Modeling and Control. USA: John Wiley & Sons. – 1984. – 387 p.

44. **Zadeh L.** Fuzzy Sets // Information and Control. – 1965. – №8. – P. 338–353.

45. **Zadeh L.** Outline of a New Approach to the Analysis of Complex Systems and Decision Processes // IEEE Trans. Syst. Man Cybernet. – №3. – 1973. – P. 28–44.

46. **Zimmermann H.-J.** Fuzzy Set Theory and its Applications. 3rd ed. – Dordrecht: Kluwer Academic Publishers. – 1996. – 315 p.

Приложение

ИНТЕРНЕТ-РЕСУРСЫ ПО НЕЧЕТКИМ СИСТЕМАМ

http://matlab.exponenta.ru\fuzzylogic\index.asp.htm – раздел Fuzzy Logic Toolbox на сайте русскоязычных пользователей системы MATLAB, который ведет автор данной книги. На сайте работает форум пользователей системы MATLAB.

http://fuzzyset.narod.ru – русскоязычный сайт «Нечеткая логика, мягкие вычисления и вычислительный интеллект» Российской Ассоциация нечетких систем и мягких вычислений. Приемник сайта «Нечеткая логика, нечеткие системы и мягкие вычисления» (http://fuzzy.kstu.ru/) Казанского государственного технологического университета. Содержит множество ссылок, статей и других материалов по нечеткой логике, нечеткому моделированию, мягким вычислениям и интеллектуальным информационным системам, включая книгу Батыршина И.З. Основные операции нечеткой логики и их обобщения. – Казань: Отечество, 2001. – 101 с. На сайте публикуются анонсы конференций и семинаров.

http://zadeh.narod.ru – русскоязычный сайт, посвященный основателю нечеткой логики профессору Заде Л. Содержит статьи и доклады Заде Л., а также материалы о нем.

http://sedok.narod.ru/index.html – электронный вариант книги Недосекина А.О. Нечетко-множественный анализ риска фондовых инвестиций. – СПб: Изд-во Сезам, 2002. – 181 с.

http://www.plink.ru/tnm/index.htm – электронный вариант книги Алтунина А.Е., Семухина М.В. Модели и алгоритмы принятия решений в нечетких условиях. – Тюмень: Изд-во Тюменского государственного университета, 2000. – 352 с.

http://fizmatlit.narod.ru/webrary/dli-3/dli-3.htm – электронный вариант книги Круглова В.В., Дли М.И., Голунова Р.Ю. Нечеткая логика и искусственные нейронные сети. – М.: Физматлит, 2001. – 224 с.

http://www.vic.spb.ru/sci/sci.htm – русскоязычный сайт по нечеткой логике, нечеткому обучению и робототехнике. Содержит много интернет-ссылок на статьи и книги в этой области, а также на фирмы и организации, занимающиеся исследованиями и разработкой соответствующего программного обеспечения.

http://www.fuzzymod.narod.ru/ – русскоязычный сайт исследовательской группы по нечеткому моделированию. Содержит множество ссылок на различные интернет-ресурсы по теории нечетких множеств и ее приложениям, включая нечеткий анализ и обработку изображений, программные продукты, использующие нечеткую логику, журналы, посвященные нечеткой логике, распознаванию образов и анализу изображений.

http://www.mathworks.com/support/books/index.jsp?category=9 – MATLAB based books (Neural/Fuzzy) – аннотированный список книг по нечеткой логике и нейронных сетям, ориентированных на использование MATLAB.

http://www-bisc.cs.berkeley.edu/ – Berkeley Initiative in Soft Computing (BISC) – сай мягких вычислений Калифорнийского Университета в Беркли, созданный под рук водством основателя теории нечетких множеств профессора Заде. Содержит матери алы BISC-семинаров, проводимых в Беркли, информацию о наиболее важных собы тиях в области нечеткой логике, ссылки на наиболее полезные интернет-ресурсы п мягким вычислениям, информацию о рабочих группах BISC и многое другое. Суще ствует список рассылки новостей.

http://www.erudit.de/ – European Network for Fuzzy Logic and Uncertainty Modeling i Information Technology – сайт Европейской информационной сети по нечеткой логик и моделированию в условиях неопределенности. Содержит материалы семинаров симпозиумов, проведенных при участии этой сети, различные демонстрационны программы, например по нечеткому управлению перевернутым маятником, материа лы конкурсов, проведенных сетью. Сайт содержит банк данных (CITE) по нечетко логике и нейронным сетям с библиографической информацией, включая ключевы слова, о более чем 17000 англоязычных публикациях. В настоящее время информа ция на сайте не обновляется через завершение проекта. Основная часть сети ERUDI трансформировалась в новый проект – European Network on Intelligent Technologie for Smart Adaptive Systems (http://www.eunite.org/).

http://fuzzy.iau.dtu.dk/EUNITETRAIN – англоязычный образовательный сайт европей ской научной сети EUNITE. Библиотека сайта содержит много интересных книг, статей интернет-курсов, слайдов лекций, упражнений и программного обеспечения по нечетко логике и другим интеллектуальным технологиям. На сайте представлена информация научных конференциях, семинарах, школах, специальных выпусках журналов; есть сло вари терминов по нечеткой логике и смежным областям на семи европейских языках

http://nafips.org – North American Fuzzy Information Processing Society – сайт Севе ро-Американском ассоциации нечеткой обработки информации. Содержит информа цию о научных конференциях, проводимых ассоциацией, список исследовательски групп по нечеткой логике. Существует список рассылки новостей.

http://www.abo.fi/~rfuller/ifsa.html – International Fuzzy Systems Association – сай Международной ассоциации нечетких систем. Содержит ссылки на интернет-ресур сы по нечеткой логике, информацию о научных форумах, архив почты по нечетким множествам, справочник «Кто есть кто в нечетком мире».

http://www.faqs.org/faqs/fuzzy-logic/part1/ – FAQ: Fuzzy Logic and Fuzzy Expert Systems - страница часто задаваемых вопросов по нечеткой логике и нечетким экспертным систе мам, поддерживаемая институтом компьютерных наук университета Карнеги-Меллона

http://www.eusflat.org/index.htm – The European Society for Fuzzy Logic and Techno logy – сайт Европейской ассоциации по нечеткой логике. Содержит информацию научных исследованиях членов Ассоциации, о научных конференциях, проводимы при участии Ассоциации а также о образовательных курсах по нечеткой логике.

http://www.acse.shef.ac.uk/~jamei/research/fuzzy_bib.htm – библиография боле 2000 публикаций по нечеткой логике.

http://http.cs.berkeley.edu/People/Faculty/Homepages/zadeh.html – персональна страница основателя теории нечетких множеств профессора Заде Л.

http://www.dcsc.tudelft.nl/~babuska/ – англоязычная веб-страница профессора Ба бушки Р., включающая Fuzzy Identification Toolbox для MATLAB 6.5 и интерактивны задания в MATLAB по курсу «Нечеткое управление».

Содержание

С. Д. Штовба

Проектирование нечетких систем средствами MATLAB

Рассмотрены вопросы проектирования нечетких систем в пакете Fuzzy Logic Toolbox вычислительной среды MATLAB. Даны необходимые сведения в области теории нечетких множеств и нечеткой логики. Приведен теоретический материал по проектированию нечетких систем. Изложены теория нечеткой идентификации, методы нечеткой кластеризации и их применение для экстракции нечетких правил, а также метод принятия решений в нечетких условиях на основе слияния целей и ограничений. Рассмотрены авторские расширения пакета для проектирования нечетких классификаторов, построения иерархических нечетких систем, обучения нечетких баз знаний типа Мамдани, а также для логического вывода при нечетких исходных данных.

Для проектировщиков систем, будет полезна научным сотрудникам, аспирантам и студентам старших курсов, интересующимся применением теории нечетких множеств в управлении, идентификации, обработке сигналов, а также разработчикам интеллектуальных систем поддержки принятия решений в медицине, биологии, социологии, экономике, политике, спорте и в других областях.

Штовба Сергей Дмитриевич, доцент, кандидат технических наук, докторант кафедры компьютерных систем управления Винницкого национального технического университета, автор свыше 80 научных публикаций, член редколлегии журнала «Exponenta Pro. Математика в приложениях» и ведущий раздела Fuzzy Logic Toolbox на сайте www.matlab.ru.

Научные интересы автора связаны с применением интеллектуальных технологий (нечеткой логики, нейронных сетей, эволюционного программирования, генетических и муравьиных алгоритмов) для решения прикладных задач в следующих областях: оценка и обеспечение надежности человеко-машинных систем; управление качеством; техническая и медицинская диагностика; экспертные системы; системы поддержки принятия решений; интеллектуальная обработка данных и экстракция знаний.

Книги издательства «Горячая линия - Телеком»
можно заказать через почтовое агентство DESSY: 107113, г.Москва, а/я 10,
а также интернет-магазин: www.dessy.ru

ISBN 5-93517-359-X

Сайт издательства:
www.techbook.ru