Глава 5
ПАКЕТ FUZZY LOGIC TOOLBOX

5.1. Назначение и возможности пакета Fuzzy Logic Toolbox

Пакет Fuzzy Logic Toolbox (пакет нечеткой логики) — это совокупность прикладных программ, относящихся к теории *размытых* или *нечетких* множеств и позволяющих конструировать так называемые нечеткие экспертные и/или управляющие системы.

Основные возможности пакета:

- построение систем нечеткого вывода (экспертных систем, регуляторов, аппроксиматоров зависимостей);
- построение адаптивных нечетких систем (гибридных нейронных сетей);
- интерактивное динамическое моделирование в Simulink.

Пакет позволяет работу:

- в режиме графического интерфейса;
- в режиме командной строки;
- с использованием блоков и примеров пакета Simulink.

5.2. Графический интерфейс Fuzzy Logic Toolbox

5.2.1. Состав графического интерфейса. В состав программных средств Fuzzy Logic Toolbox входят следующие основные программы, позволяющие работать в режиме графического интерфейса:

- редактор нечеткой системы вывода Fuzzy Inference System Editor (FIS Editor или FIS-редактор) вместе со вспомогательными программами — редактором функций принадлежности (Membership Function Editor), редактором правил (Rule Editor), просмотрщиком правил (Rule Viewer) и просмотрщиком поверхности отклика (Surface Viewer);
- редактор гибридных систем (ANFIS Editor, ANFIS-редактор);
- программа нахождения центров кластеров (программа Clustering — кластеризация).

Набор данных программ предоставляет пользователю максимальные удобства для создания, редактирования и использования различных систем нечеткого вывода.

5.2.2. Построение нечеткой аппроксимирующей системы. Командой (функцией) **Fuzzy** из режима командной строки запускается основная интерфейсная программа пакета Fuzzy Logic — редактор нечеткой системы вывода (Fuzzy Inference System Editor,

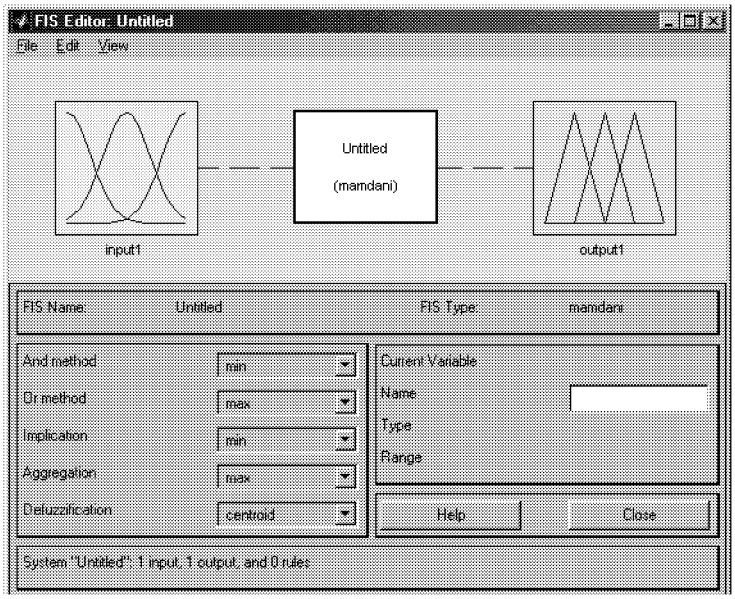

Рис. 5.1. Вид окна FIS Editor

FIS Editor, FIS-редактор). Вид открывающегося при этом окна приведен на рис. 5.1.

Главное меню редактора содержит позиции:

File — работа с файлами моделей (их создание, сохранение, считывание и печать);

Edit — операции редактирования (добавление и исключение входных и выходных переменных);

View — переход к дополнительному инструментарию.

Как говорят англичане, чтобы узнать вкус пудинга, надо его съесть, поэтому представляется целесообразным выяснение различных опций и возможностей данного редактора и связанных с ним других программ изучить на каком-либо конкретном примере.

Попробуем сконструировать нечеткую систему, отображающую зависимость между переменными x и y, заданную с помощью табл. 5.1 (легко видеть, что представленные в таблице данные отражают зависимость $y = x^2$).

Таблица 5.1. **Значения x и y**

x	1	0.6	0	0.4	1
y	-1	0.36	0	0.16	1

Требуемые действия отобразим следующими пунктами.

1. В позиции меню File выбираем опцию New Sugeno FIS (новая система типа Sugeno), при этом в блоке, отображаемом белым квадратом, в верхней части окна редактора появится надпись Untitled2 (sugeno).

2. Щелкнем левой кнопкой мыши по блоку, озаглавленному input1 (вход1). Затем в правой части редактора в поле, озаглавленном Name (Имя), вместо input1 введем обозначение нашего аргумента, т.е. x. Обратим внимание, что если теперь сделать где-нибудь (вне блоков редактора) однократный щелчок мыши, то имя отмеченного блока изменится на x; то же достигается нажатием после ввода клавиши Enter.

3. Дважды щелкнем по этому блоку. Перед нами откроется окно редактора функций принадлежности — Membership Function Editor (см. рис. 5.2). Войдем в позицию меню Edit данного редактора и выберем в нем опцию Add MFs (Add Membership Funcions — Добавить функций принадлежности). При этом появится диалоговое окно (рис. 5.3), позволяющее задать тип (MF type) и количество (Number of MFs) функций принадлежности (в данном случае все относится к входному сигналу, т.е. к переменной x). Выберем гауссовы функции принадлежности (gaussmf), а их количество зададим равным пяти — по числу значений аргумента в табл. 5.1. Подтвердим ввод информации нажатием кнопки OK, после чего произойдет возврат к окну редактора функций принадлежности.

4. В поле Range (Диапазон) установим диапазон изменения x от -1 до 1, т.е. диапазон, соответствующий табл. 5.1. Щелкнем затем левой кнопкой мыши где-нибудь в поле редактора (или нажмем клавишу ввода Enter). Обратим внимание, что после этого произойдет соответствующее изменение диапазона в поле Display Range (Диапазон дисплея).

5. Обратимся к графикам заданных нами функций принадлежности, изображенным в верхней части окна редактора функ-

ций принадлежности. Заметим, что для успешного решения поставленной задачи необходимо, чтобы ординаты максимумов этих

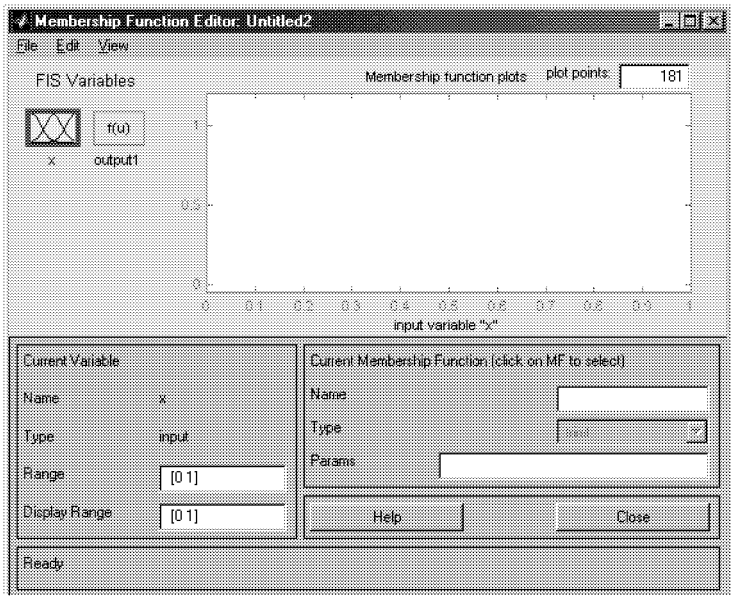

Рис. 5.2. Окно редактора функций принадлежности

функций совпадали с заданными значениями аргумента x. Для левой, центральной и правой функций такое условие выполнено, но две другие необходимо «подвинуть» вдоль оси абсцисс. «Пере-

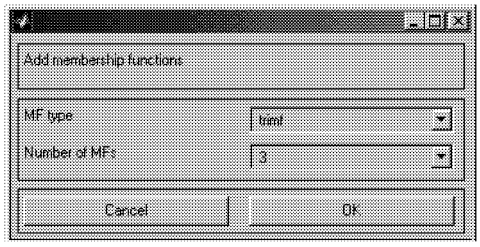

Рис. 5.3. Диалоговое окно задания типа и количества функций принадлежности

движка» делается весьма просто: подводим курсор к нужной кривой и щелкаем левой кнопкой мыши. Кривая выбирается, окрашиваясь в красный цвет, после чего с помощью курсора ее и можно

подвинуть в нужную сторону (более точную установку можно провести, изменяя числовые значения в поле Params (Параметры) — в данном случае каждой функции принадлежности соответствуют два параметра, при этом первый определяет размах кривой, а второй — положение ее центра). Для выбранной кривой, кроме этого, в поле Name можно изменять имя (завершая ввод каждого имени нажатием клавиши Enter). Проделаем требуемые перемещения кривых и зададим всем пяти кривым новые имена, например:

- самой левой — bn,
- следующей — n,
- центральной — z,
- следующей за ней справа — p,
- самой левой — bp.

Нажмем кнопку Close и выйдем из редактора функций принадлежности, возвратившись при этом в окно редактора нечеткой системы (FIS Editor).

6. Сделаем однократный щелчок левой кнопкой мыши по голубому квадрату (блоку), озаглавленному output1 (выход1). В окошке Name заменим имя output1 на y (как в пункте 2).

7. Дважды щелкнем по отмеченному блоку и перейдем к программе — редактору функций принадлежности. В позиции меню Edit выберем опцию Add MFs. Появляющееся диалоговое окно вида рис. 5.3 позволяет задать теперь в качестве функций принадлежности только линейные (linear) или постоянные (constant) — в зависимости от того, какой алгоритм Sugeno (1-го или 0-го порядка) мы выбираем. В рассматриваемой задаче необходимо выбрать постоянные функции принадлежности с общим числом 4 (по числу различных значений y в табл. 5.1). Подтвердим введенные данные нажатием кнопки OK, после чего произойдет возврат в окно редактора функций принадлежности.

8. Обратим внимание, что здесь диапазон (Range) изменения, устанавливаемый по умолчанию — [0, 1], менять не нужно. Изменим лишь имена функций принадлежности (их графики при использовании алгоритма Sugeno для выходных переменных не приводятся), например, задав их как соответствующие числовые значения y, т. е. 0, 0.16, 0.36, 1; одновременно эти же числовые значение введем в поле Params (рис. 5.4). Затем закроем окно нажатием кнопки Close и вернемся в окно FIS-редактора.

9. Дважды щелкнем левой кнопкой мыши по среднему (белому) блоку, при этом раскроется окно еще одной программы — редактора правил (Rule Editor). Введем соответствующие правила. При вводе каждого правила необходимо обозначить соот-

ветствие между каждой функцией принадлежности аргумента x и числовым значением y. Кривая, обозначенная нами bn, соответствует $x = -1$, т.е. $y = 1$. Выберем, поэтому в левом поле

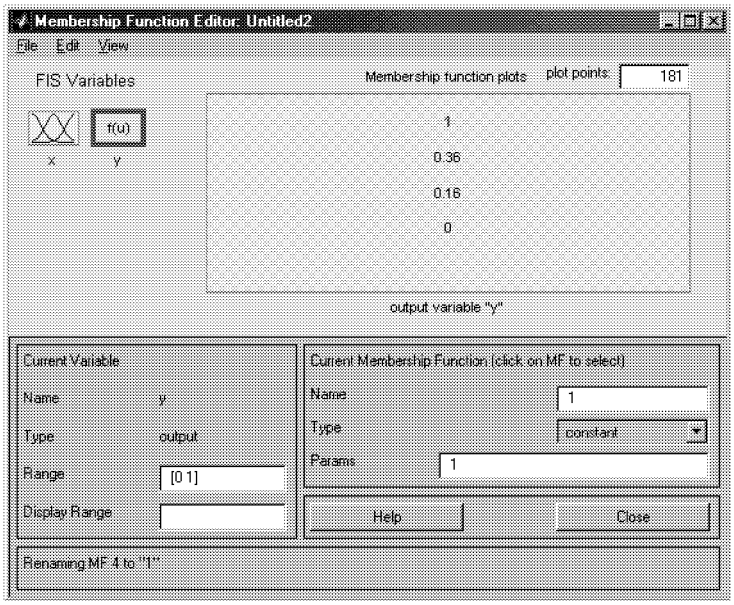

Рис. 5.4. Параметры функций принадлежности переменной y

(с заголовком x is) bn, а в правом 1 и нажмем кнопку Add rule (Добавить правило). Введенное правило появится в окне правил и будет представлять собой запись: 1. If (x is bn) then (y is 1) (1). Аналогично поступим для всех других значений x, в результате чего сформируется набор из 5 правил (см. рис. 5.5). Закроем окно редактора правил и возвратимся в окно FIS-редактора. Построение системы закончено и можно начать эксперименты по ее исследованию. Заметим, что большинство опций выбиралось нами по умолчанию.

10. Предварительно сохраним на диске (используя пункты меню File/Save to disk as...) созданную систему под каким-либо именем, например, Proba.

11. Выберем позицию меню View. Как видно из выпадающего при этом подменю, с помощью пунктов Edit membership functions и Edit rules можно совершить переход к двум выше рассмотренным программам — редакторам функций принадлежности и пра-

вил (то же можно сделать и нажатием клавиш Ctrl+2 или Ctrl+3), но сейчас нас будут интересовать два других пункта — View rules

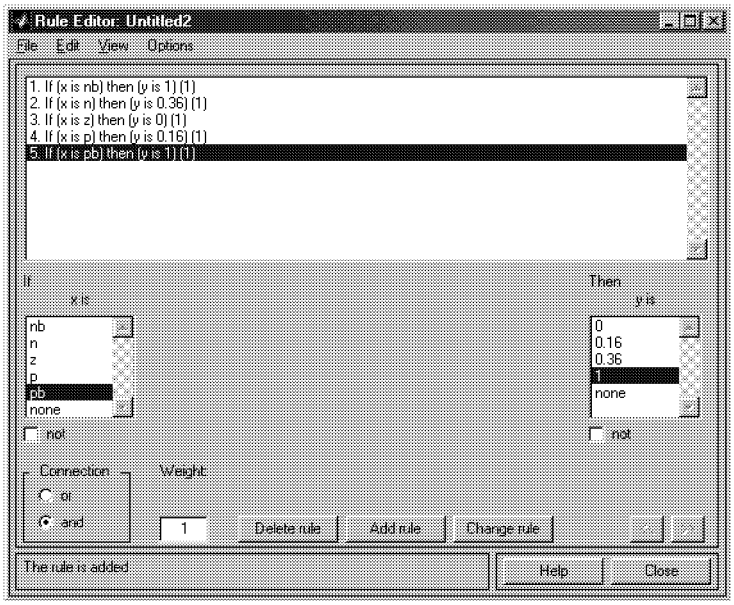

Рис. 5.5. Окно редактора правил

(Просмотр правил) и View surface (Просмотр поверхности). Выберем пункт View rules, при этом откроется окно (см. рис. 5.6) еще одной программы — просмотра правил (Rule Viewer).

12. В правой части окна в графической форме представлены функции принадлежности аргумента x, в левой — переменной выхода y с пояснением механизма принятия решения. Красная вертикальная черта, пересекающая графики в правой части окна, которую можно перемещать с помощью курсора, позволяет изменять значения переменной входа (это же можно делать, задавая числовые значения в поле Input (Вход)), при этом соответственно изменяются значения y в правой верхней части окна. Зададим, например, $x = 0.5$ в поле Input и нажмем затем клавишу ввода (Enter). Значение y сразу изменится и станет равным 0.202. Таким образом, с помощью построенной модели и окна просмотра правил можно решать задачу интерполяции, т.е. задачу, решение которой и требовалось найти. Изменение аргумента путем

перемещения красной вертикальной линии очень наглядно демонстрирует, как система определяет значения выхода.

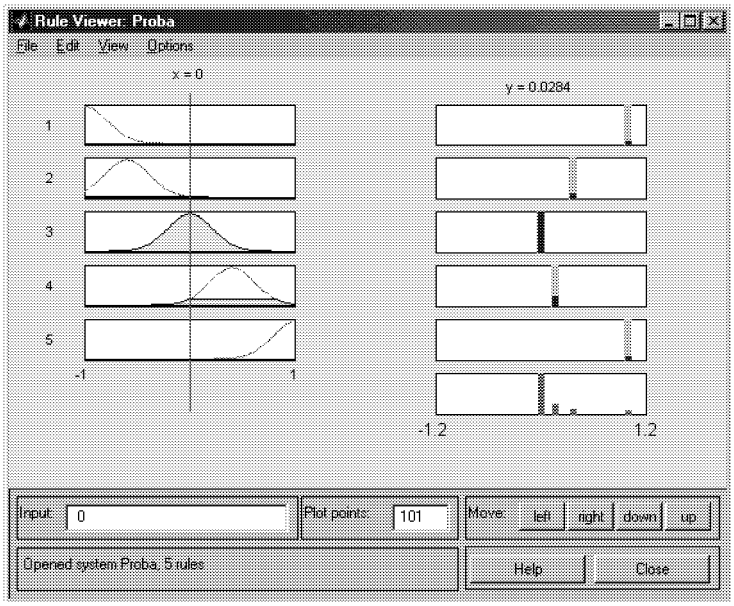

Рис. 5.6. Окно просмотра правил

13. Закроем окно просмотра правил и выбором пункта меню View/View surface перейдем к окну просмотра поверхности отклика (выхода), в нашем случае — к просмотру кривой $y(x)$ (см. рис. 5.7). Видно, что смоделированное системой по таблице данных (табл. 5.1) отображение не очень-то напоминает функцию x^2. Ну что ж, ничего удивительного в этом нет: число экспериментальных точек невелико, да и параметры функций принадлежности (для x) выбраны, скорее всего, неоптимальным образом. Ниже мы рассмотрим возможность улучшения качества подобной модели.

В заключение рассмотрения примера отметим, что с помощью вышеуказанных программ-редакторов на любом этапе проектирования нечеткой модели в нее можно внести необходимые коррективы, вплоть до задания какой-либо особенной пользовательской функции принадлежности. Из опций, устанавливаемых в FIS-редакторе по умолчанию при использовании алгоритма Sugeno, можно отметить:

• логический вывод организуется с помощью операции умножения (prod);

• композиция — с помощью операции логической суммы (вероятностного ИЛИ, probor);

• приведение к четкости — дискретным вариантом центроидного метода (взвешенным средним, wtaver).

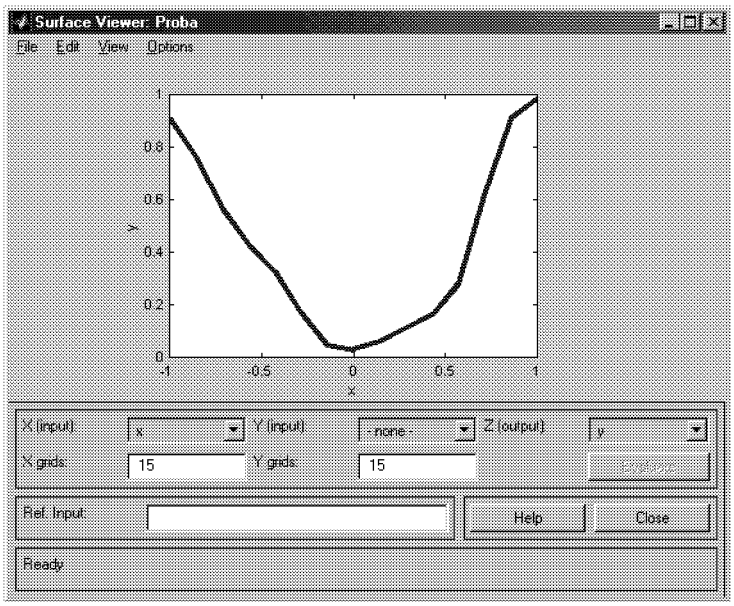

Рис. 5.7. Окно просмотра поверхности отклика

Используя соответствующие поля в левой нижней части окна FIS-редактора, данные опции можно, при желании, изменить.

5.2.3. Построение экспертной системы: сколько дать «на чай»? Рассмотрим теперь методику построения нечеткой экспертной системы, которая должна помочь пользователю с ответом на вопрос: сколько дать «на чай» официанту за обслуживание в ресторане? (Предположим, речь идет о местах, где такие чаевые принято давать, например, в ресторанах Парижа или Рио-де-Жанейро.)

Основываясь на каких-то устоявшихся обычаях и интуитивных представлениях, примем, что задача о чаевых может быть описана следующими предложениями.

1. Если обслуживание плохое или еда подгоревшая, то чаевые — малые.

2. Если обслуживание хорошее, то чаевые — средние.

3. Если обслуживание отличное или еда превосходная, то чаевые — щедрые.

Качество обслуживания и еды будем оценивать по 10-балльной системе (0 — наихудшая оценка, 10 — наилучшая).

Будем предполагать, далее, что малые чаевые составляют около 5% от стоимости обеда, средние — около 15% и щедрые — примерно 25%.

Заметим, что представленной информации, в принципе, достаточно для проектирования нечеткой экспертной системы. Такая система будет иметь 2 входа (которые условно можно назвать «сервис» и «еда»), один выход («чаевые»), три правила типа «если... то» (в соответствии с тремя приведенными предложениями) и по три значения (соответственно, 0 баллов, 5 баллов, 10 баллов и 5%, 15%, 25%) для центров функций принадлежности входов и выхода. Построим данную систему, используя алгоритм вывода Mamdani и, как в предыдущем примере, описывая требуемые действия по пунктам.

1. Командой **fuzzy** запускаем FIS-редактор. По умолчанию, исходный алгоритм вывода — типа Mamdani (о чем говорит надпись в центральном белом блоке) и здесь никаких изменений не требуется, но в системе должно быть два входа, поэтому через пункт меню Edit/Add input добавляем в систему этот второй вход (в окне редактора появляется второй желтый блок с именем input2). Делая далее однократный щелчок левой кнопкой мыши по блоку input1, меняем в поле имени его имя на «сервис», завершая ввод нового имени нажатием клавиши Enter. Аналогичным образом устанавливаем имя «еда» блоку input2 и «чаевые» — выходному блоку (справа вверху) output1. Присвоим сразу же и имя всей системе, например, «tip» (по-английски это и есть чаевые), выполнив это через пункт меню File/Save to workspace as... (Сохранить в рабочем пространстве как...). Вид окна редактора после указанных действий приведен на рис. 5.8.

2. Зададим теперь функции принадлежности переменных. Напомним еще раз, что программу-редактор функций принадлежности можно открыть тремя способами:

- через пункт меню View/Edit membership functions...,
- двойным щелчком левой кнопки мыши по иконке, отображающей соответствующую переменную,
- нажатием клавиш Ctrl+2.

Любым из приведенных способов перейдем к данной программе.

Задание и редактирование функций принадлежности начнем с переменной «сервис». Сначала в полях Range и Display Range

установим диапазон изменения и отображения этой переменной — от 0 до 10 (баллов), подтверждая ввод нажатием клавиши Enter. Затем через пункт меню Edit/Add MFs перейдем к диалоговому окну вида рис. 5.3 и зададим в нем функции принадлежности гауссова типа (gaussmf) с общим числом 3. Нажмем кнопку OK и возвратимся в окно редактора функций принадлежности. Не изменяя размах и положение заданных функций, заменим только их имена на «плохой», «хороший» и «отличный» (как в пункте 5 предыдущего примера).

Щелчком левой кнопки мыши по иконке «еда» войдем в окно редактирования функций принадлежности для этой переменной. Зададим сначала диапазон ее изменения от 0 до 10, а затем, по-

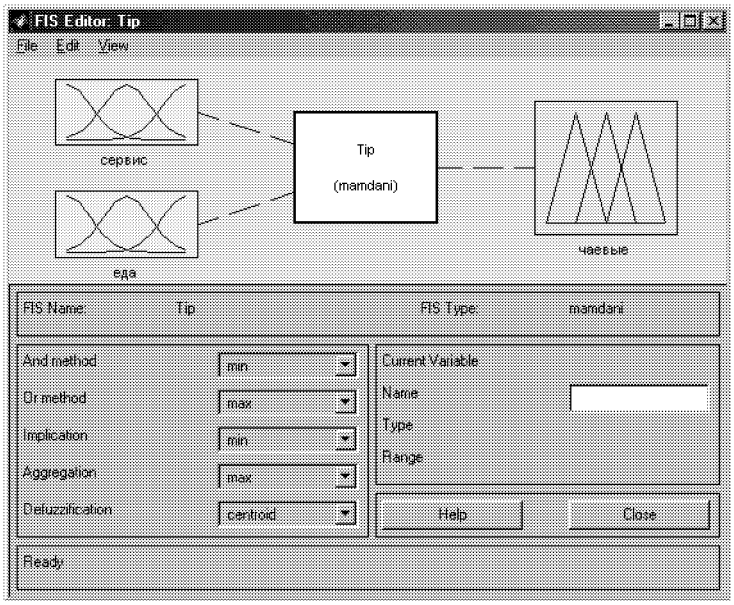

Рис. 5.8. Вид окна FIS-редактора после задания структуры системы

ступая как ранее, зададим две функции принадлежности трапецеидальной формы с параметрами, соответственно, [0 0 1 3] и [7 9 10 10] и именами «подгоревшая» и «превосходная».

Для выходной переменной «чаевые» укажем сначала диапазон изменения — от 0 до 30, потом зададим три функции принадлежности треугольной формы с именами «малые», «средние», «щедрые» так, как это представлено на рис. 5.9. Заметим, что можно,

разумеется, задать и какие-либо другие функции или выбрать их другие параметры.

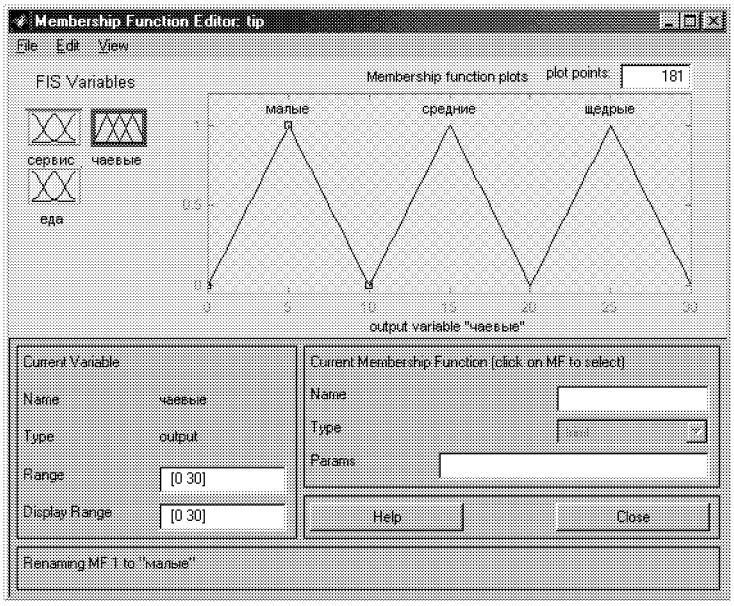

Рис. 5.9. Функции принадлежности переменной «чаевые»

3. Перейдем к конструированию правил. Для этого выберем пункт меню View/Edit rules... Далее ввод правил производится так же, как в п. 9 предыдущего примера, и в соответствии с предложениями, описывающими задачу. Заметим, что в первом и третьем правилах в качестве «связки» в предпосылках правила необходимо использовать не «И» (and), а «ИЛИ» (or); при вводе второго правила, где отсутствует переменная «еда», для нее выбирается опция none. Итоговый набор правил отображен рис. 5.10 и выглядит следующим образом:

1. If (сервис is плохой) or (еда is подгоревшая) then (чаевые is малые) (1)

2. If (сервис is хороший) then (чаевые is средние) (1)

3. If (сервис is отличный) или (еда is превосходная) then (чаевые is щедрые) (1)

Такая (подробная, verbose) запись представляется достаточно понятной; единица в скобках после каждого правила указывает его «вес» (Weight), т.е. значимость правила. Данный вес можно

менять, используя соответствующее поле в левой нижней части окна редактора правил. Правила представимы и в других формах: символической (symbolic) и индексной (indexed), при этом пере-

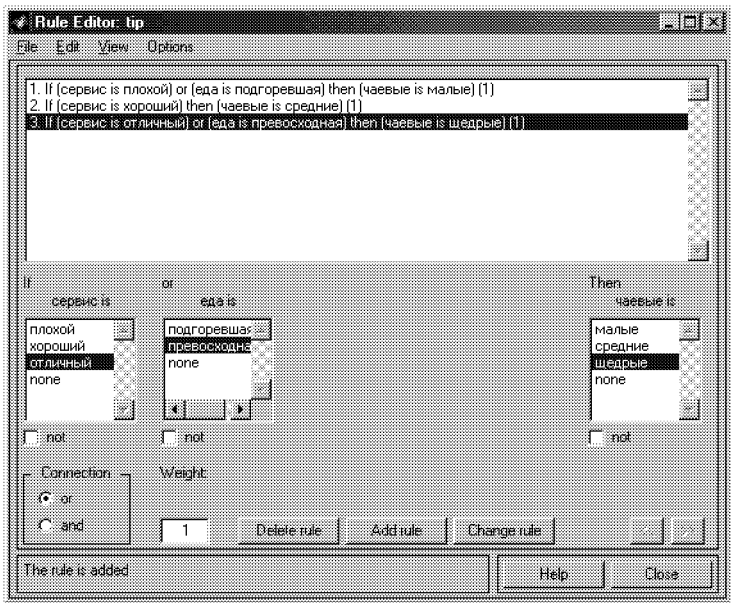

Рис. 5.10. Итоговый набор правил в задаче о чаевых

ход от одной формы к другой происходит через опции пункта меню редактора правил Options/Format. Вот как выглядят рассмотренные правила в символической форме:

1. (сервис==плохой)|(еда==подгоревшая)⇒
⇒(чаевые=малые) (1)
2. (сервис==хороший)⇒(чаевые=средние) (1)
3. (сервис==отличный)|(еда==превосходная)⇒
⇒(чаевые=щедрые) (1)

По-видимому, здесь тоже понятно все.

Наконец, самый сжатый формат представления правил — индексный — является тем форматом, который в действительности используется программой. В этом формате приведенные правила выглядят так:

1 1, 1 (1): 2
2 0, 2 (1): 2
3 2, 3 (1): 2

Здесь первая колонка относится к первой входной переменной (соответственно, первое, второе или третье возможное значение), вторая — ко второй, третья (после запятой) — к выходной переменной, цифра в скобках показывает вес правила и последняя цифра (после двоеточия) — на тип «связки» (1 для «И», 2 для «ИЛИ»).

На этом, собственно, конструирование экспертной системы закончено. Сохраним ее на диске под выбранным именем (tip).

4. Самое время теперь проверить систему в действии. Откроем (через пункт меню View/View rules...) окно просмотра правил и установим значения переменных: сервис=0 (т.е. никуда не годный), еда=10 (т.е. превосходная). Увидим ответ: чаевые=15 (т.е. средние). Ну что ж, с системой не поспоришь, надо платить (рис. 5.11). Можно проверить и другие варианты. В частности (может быть, не без удивления), выяснится, что нашей системой обслуживание ценится больше, чем качество еды: при наборе

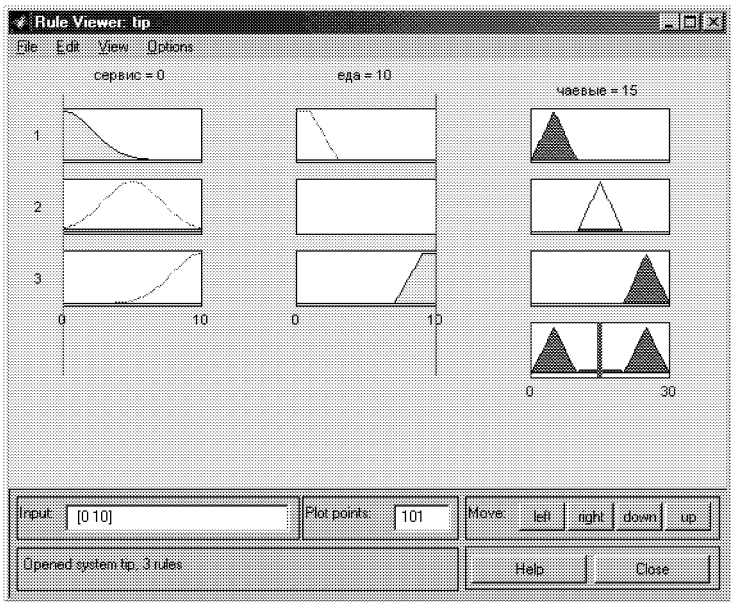

Рис. 5.11. Окно просмотра правил в задаче о чаевых

«сервис=10, еда=3» система советует определить размер чаевых в 23.9%, в то время как набору «сервис=3, еда=10» размер чаевых по рекомендации системы — 16.6% (от стоимости обеда). Впро-

чем, ничего удивительного здесь нет: это мы сами (не особенно подозревая об этом) заложили в систему соответствующие знания в виде совокупности приведенных правил.

Подтверждением отмеченной зависимости выходной переменной от входных может служить вид поверхности отклика, который представляется при выборе пункта меню View/View surface

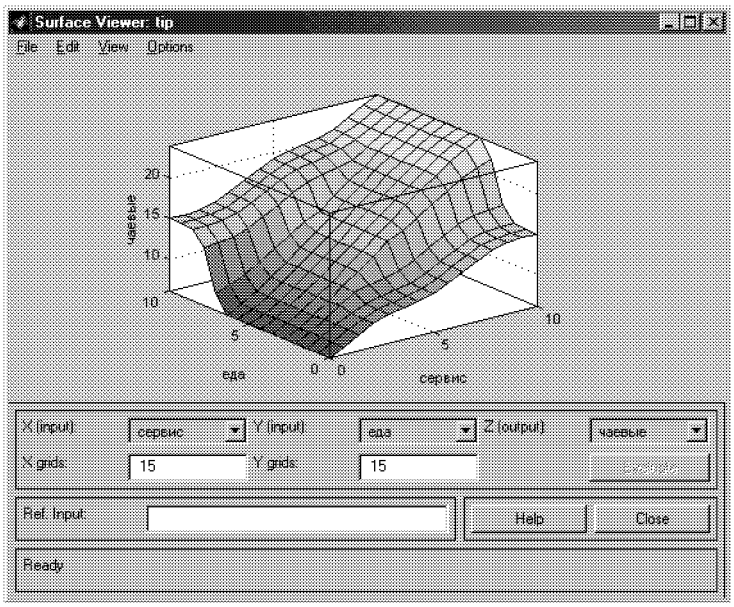

Рис. 5.12. Графический вид зависимости выходной переменной от входных

(рис. 5.12); обратите внимание, что с помощью мышки график можно поворачивать во все стороны.

В открывшемся окне, меняя имена переменных в полях ввода (X(input) и Y(input)), можно задать и просмотр одномерных зависимостей, например «чаевых» от «еды» (рис. 5.13).

5.2.4. Экспорт и импорт результатов. Когда вы сохраняете созданную вами нечеткую систему, используя пункты меню File/Save to disk или File/Save to disk as..., на диске создается текстовый (ASCII) файл достаточно простого формата с расширением .fis, который можно просматривать, при необходимости — редактировать вне системы MATLAB, а также использовать повторно при последующих сеансах работы с системой. Однако сохранение с

использованием пунктов File/Save to workspace или File/Save to workspace as... на самом деле только «легализует» созданную вами систему (под каким-либо именем) в среде MATLAB в течение те-

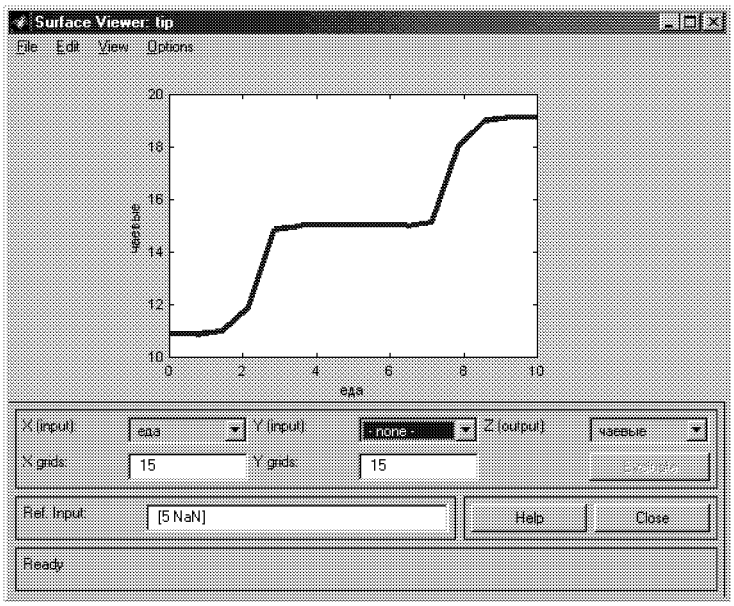

Рис. 5.13. Одномерная зависимость размера чаевых от качества еды

кущего сеанса работы и не допускает ее повторного использования в других сеансах.

5.2.5. Создание пользовательских функций принадлежности. Если по каким-либо причинам вас не устраивает ни одна из встроенных функций принадлежности, вы можете создать и использовать собственную подходящую функцию. Такая функция должна быть создана как M-файл со значениями от 0 до 1 и с числом аргументов не более 16. Приведем этапы создания данной функции под некоторым именем custmf.

1. Создается соответствующий M-файл с именем custmf.m.

2. Выбирается пункт Edit/Add custom MF (Редактирование/ Добавить пользовательскую функцию принадлежности) в меню редактора функций принадлежности.

3. В поле M-File function name появляющегося диалогового окна Add customized membership function вводится имя созданного M-файла (custmf).

4. В поле Parameter list данного окна вводятся необходимые числовые параметры.

5. Наконец, в поле MF name (Имя функции принадлежности) вводится какое-либо (уникальное) имя задаваемой функции (например, custmf).

6. Указанный ввод подтверждается нажатием кнопки OK (см. рис. 5.14).

Ниже приведен пример M-файла некоторой пользовательской функции принадлежности дискретного типа, имеющей имя testmf1

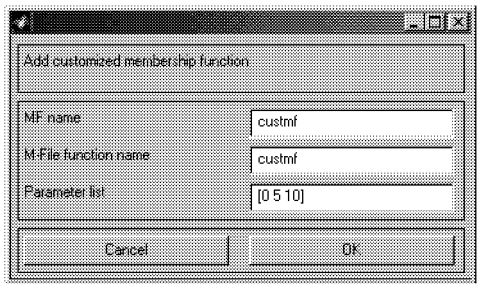

Рис. 5.14. Окно задания функции принадлежности пользователя

и зависящей от 8 числовых параметров (каждый — из диапазона $[0, 10]$):

```
function out = testmf1(x, params)
for i = 1:length(x)
if x(i)<params(1)
y(i) = params(1);
elseif x(i)<params(2)
y(i) = params(2);
elseif x(i)<params(3)
y(i) = params(3);
elseif x(i)<params(4)
y(i) = params(4);
elseif x(i)<params(5)
y(i) = params(5);
elseif x(i)<params(6)
y(i) = params(6);
elseif x(i)<params(7)
y(i) = params(7);
elseif x(i)<params(8)
y(i) = params(8);
else
```

```
y(i) = 0;
end
end
out =.1*y'
```

5.3. Графический интерфейс гибридных систем

Графический интерфейс гибридных (нечетких) нейронных систем вызывается функцией (из режима командной строки) **anfisedit**. Исполнение функции приводит к появлению окна редактора гибридных систем (ANFIS Editor, ANFIS-редактор), вид которого приведен на рис. 5.15.

С помощью данного редактора осуществляется создание или загрузка структуры гибридной системы, просмотр структуры,

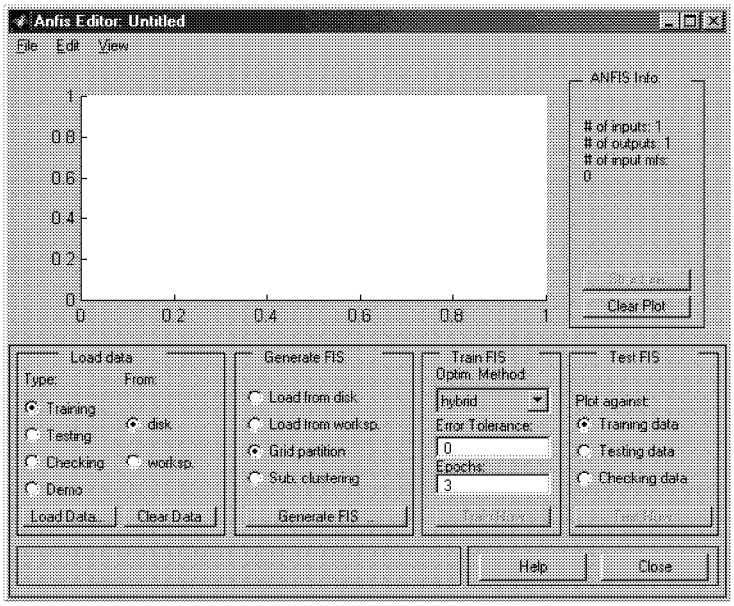

Рис. 5.15. Окно редактора гибридных систем

настройка ее параметров, проверка качества функционирования такой системы. Создание структуры и настройка параметров и проверка осуществляются по выборкам (наборам данных) — обучающей (Training data), проверочной (Checking data) и тестирующей (Testing data), которые предварительно должны быть представлены в виде текстовых фалов (с расширением .dat и раздели-

телями-табуляциями), первые колонки которых соответствуют входным переменным, а последняя (левая) — единственной выходной переменной; количество строк в таких файлах равно количеству образцов (примеров). Так, обучающая выборка, сформированная по табл. 5.1, представляется в виде

-1 1
-0.6 0.36
0.0 0.00
0.4 0.16
1 1

Строгих рекомендаций по объемам указанных выборок не существует, по-видимому, лучше всего исходить из принципа «чем больше, тем лучше». Обучающая и проверочная выборки непосредственно задействуются в процессе настройки параметров гибридной сети (проверочная — для выяснения ситуации: нет ли так называемого переобучения сети, при котором ошибка для обучающей последовательности стремится к нулю, а для проверочной — возрастает; впрочем, наличие проверочной выборки не является строго необходимым, оно лишь крайне желательно). Тестовая (или тестирующая выборка) применяется для проверки качества функционирования настроенной (обученной) сети.

Поясним пункты меню и опции редактора.

Пункты меню File и View, в общем идентичны аналогичным пунктам FIS-редактора, за тем исключением, что здесь работа может происходить только с алгоритмом нечеткого вывода Sugeno. Пункт меню Edit содержит единственный подпункт — Undo (Отменить выполненное действие).

Набор опций Load data (Загрузить данные) в нижней левой части окна редактора включает в себя:

- тип (Type) загружаемых данных (для обучения — Training, для тестирования — Testing, для проверки — Checking, демонстрационные — Demo);
- место, откуда должны загружаться данные (с диска — disk или из рабочей области MATLAB-workspace).

К данным опциям относятся две кнопки, нажатие на которых приводит к требуемым действиям — Load Data... (Загрузить данные) и Clear Data (очистить, т. е. стереть введенные данные).

Следующая группа опций (в середине нижней части окна ANFIS-редактора) объединена под именем Generate FIS (Создание нечеткой системы вывода). Данная группа включает в себя опции:

- загрузку структуры системы с диска (Load from disk);

• загрузку структуры системы из рабочей области MATLAB (Load from worksp.);

• разбиение (деление) областей определения входных переменных (аргументов) на подобласти — независимо для каждого аргумента (Grid partition);

• разбиение всей области определения аргументов (входных переменных) на подобласти — в комплексе для всех аргументов (Subtract clustering или Sub. clustering), а также кнопку Generate FIS, нажатие которой приводит к процессу создания гибридной системы с точностью до ряда параметров.

Следующая группа опций — Train FIS (Обучение нечеткой системы вывода) — позволяет определить метод «обучения» (Optim. Method) системы (т.е. метод настройки ее параметров) — гибридный (hybrid) или обратного распространения ошибки (backprop), установить уровень текущей суммарной (по всем образцам) ошибки обучения (Error Tolerance), при достижении которого процесс обучения заканчивается и количество циклов обучения (Epochs), т.е. количество «прогонов» всех образцов (или примеров) обучающей выборки; процесс обучения, таким образом заканчивается либо при достижении отмеченного уровня ошибки обучения, либо при проведении заданного количество циклов.

Кнопка Train Now (Начать обучение) процесс обучения, т.е. процесс настройки параметров гибридной сети.

В правом верхнем углу окна ANFIS-редактора выдается информация (ANFIS Info.) о проектируемой системе: о количестве входов, выходов, функций принадлежности входов; нажатие кнопки Structure (Структура) позволяет увидеть структуру сети. Кнопка Clear (Очистить) позволяет стереть все результаты.

Опции Test FIS в правом нижнем углу окна позволяют провести проверку и тестирование созданной и обученной системы с выводом результатов в виде графиков (соответствующие графики для обучающей выборки — Training data, тестирующей выборки — Testing data и проверочной выборки — Checking data. Кнопка Test Now позволяет запустить указанные процессы.

Работу с редактором рассмотрим на примере восстановления зависимости $y = x^2$ по данным табл. 5.1. Предположим, что эти данные сохранены в файле Proba.dat. Создание и проверку системы, как и раньше, проведем по этапам.

1. В окне ANFIS-редактора выберем тип загружаемых данных Training и нажмем кнопку Load data. В последующем стандартном окне диалога укажем местоположение и имя файла. Его открытие приводит к появлению в графической части окна редактора набора точек, соответствующих введенным данным (рис. 5.16).

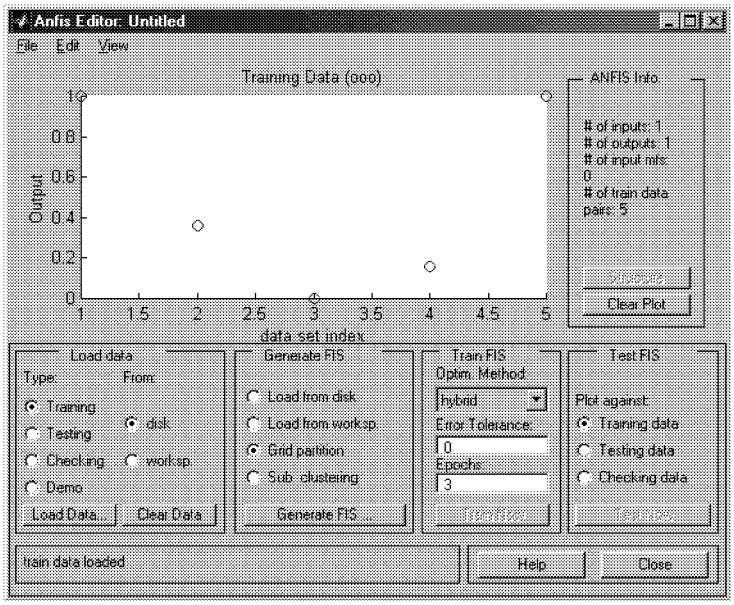

Рис. 5.16. Окно ANFIS-редактора после загрузки обучающей выборки

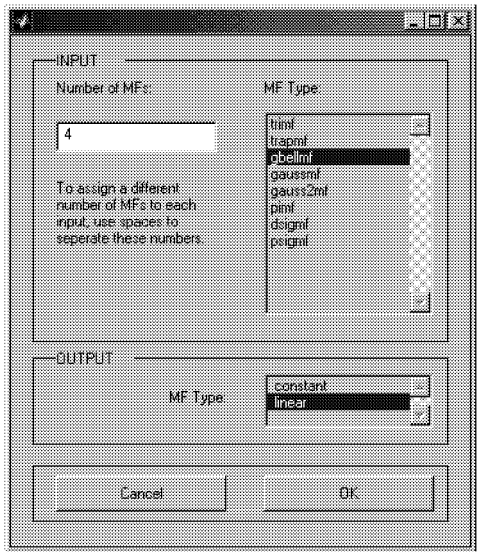

Рис. 5.17. Окно задания функций принадлежности

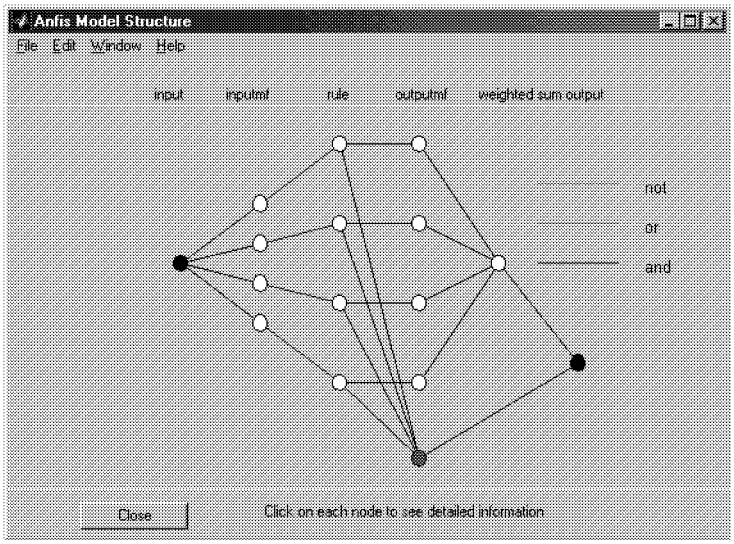

Рис. 5.18. Структура созданной гибридной сети

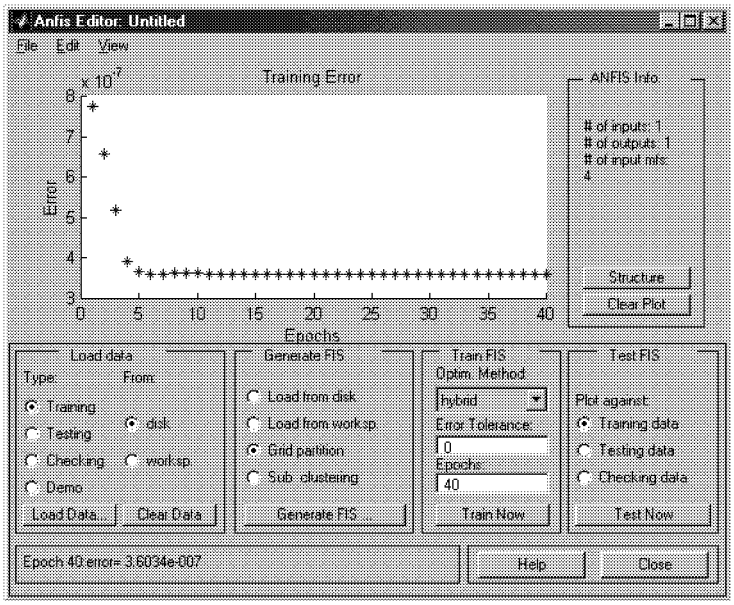

Рис. 5.19. Результат обучения сети

2. В группе опций Generate FIS по умолчанию активизирована опция Grid partition. Не будем ее изменять и нажмем кнопку Generate FIS, после чего появится диалоговое окно (рис. 5.17) для задания числа и типов функций принадлежности. Сохраним все установки по умолчанию, согласившись с ними нажатием кнопки OK. Произойдет возврат в основное окно ANFIS-редактора. Теперь структура гибридной сети создана, и ее графический вид можно просмотреть с помощью кнопки Structure (рис. 5.18).

3. Перейдем к опциям Train FIS. Не будем менять задаваемые по умолчанию метод настройки параметров (hybrid — гибридный) и уровень ошибки (0), но количество циклов обучения изменим на 40, после чего нажмем кнопку начала процесса обучения (Train Now). Получившийся результат в виде графика ошибки сети в зависимости от числа проведенных циклов обучения (из которого следует, что фактически обучение закончилось после пятого цикла) представлен на рис. 5.19.

4. Теперь нажатием кнопки Test Now можно начать процесс тестирования обученной сети, но, поскольку использовалась только одна — обучающая — выборка, ничего особенно интересного ожидать не приходится. Действительно, выход обученной системы практически совпадает с точками обучающей выборки (рис. 5.20).

5. Сохраним разработанную систему в файл на диске с именем Proba1 (с расширением .fis) и для исследования разработанной системы средствами FIS-редактора из командной строки MATLAB выполним команду **fuzzy**, а затем через пункты меню File/Open FIS from disk... откроем созданный файл. С созданной системой можно теперь выполнять все приемы редактирования (изменение имен переменных и т. п.) и исследования, которые были рассмотрены выше. Здесь нетрудно, кстати, убедиться, что качество аппроксимации данных существенно не улучшилось — слишком мало данных.

Что можно сказать про эффективность использования гибридных систем (и ANFIS-редактора)?

В данном случае используется только один алгоритм нечеткого вывода — Sugeno (нулевого или первого порядков), может быть только одна выходная переменная, всем правилам приписывается один и тот же единичный вес. Вообще говоря, возникают значительные проблемы при большом (более 5–6) количестве входных переменных. Это — ограничения и недостатки подхода.

Его несомненные достоинства: практически полная автоматизация процесса создания нечеткой (гибридной) системы, возможность просмотра сформированных правил и придания им содержа-

тельной (лингвистической) интерпретации, что позволяет, кстати говоря, рассматривать аппарат гибридных сетей как средство извлечения знаний из баз данных и существенно отличает данные сети от классических нейронных.

Рекомендуемая область применения: построение аппроксиматоров зависимостей по экспериментальным данным, построение

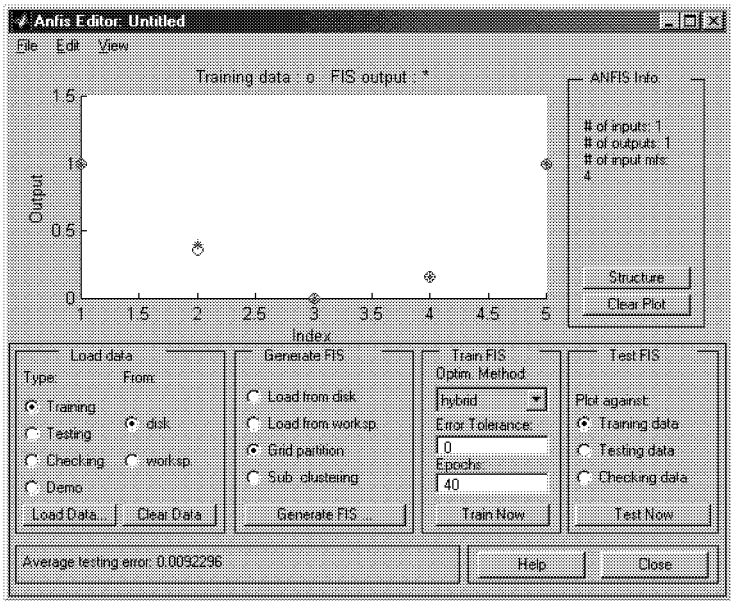

Рис. 5.20. Результат тестирования обученной системы

систем классификации (в случае бинарной или дискретной выходной переменной), изучение механизма явлений.

5.4. Графический интерфейс программы кластеризации

В пакет Fuzzy Logic Toolbox входит еще одна программа, позволяющая работу в режиме графического интерфейса, — программа Clustering (Кластеризация) выявления центров кластеров, т.е. точек в многомерном пространстве данных, около которых группируются (скапливаются) экспериментальные данные. Выявление подобных центров, надо сказать, является значимым этапом при предварительной обработке данных, поскольку позволяет сопоставить с этими центрами функции принадлежности переменных при последующем проектировании системы нечеткого вывода.

Запуск программы Clustering осуществляется командой (функцией) **findcluster**. В появляющемся окне программы имеется (вверху) главное меню, содержащее достаточно стандартный набор пунктов (File, Edit, Window, Help) и набор управляющих кнопок и опций (справа). К этим кнопкам относятся:

- кнопка загрузки фала данных Load Data,
- кнопка выбора алгоритма кластеризации — Method,
- четыре расположенные ниже кнопки опций алгоритма (их названия меняются в зависимости от выбранного алгоритма),
- кнопка начала итеративного процесса нахождения центров кластеров (кластеризации) — Start,
- кнопка сохранения результатов кластеризации (Save Center),
- кнопка очистки (стирания) графиков (Clear Plot),
- кнопка справочной информации (Info),
- кнопка завершения работы с программой (Close).

В программе используются два алгоритма выявления центров кластеров: Fuzzy c-means (который можно перевести как «Алгоритм нечетких центров») и Subtractive clustering («Вычитающая

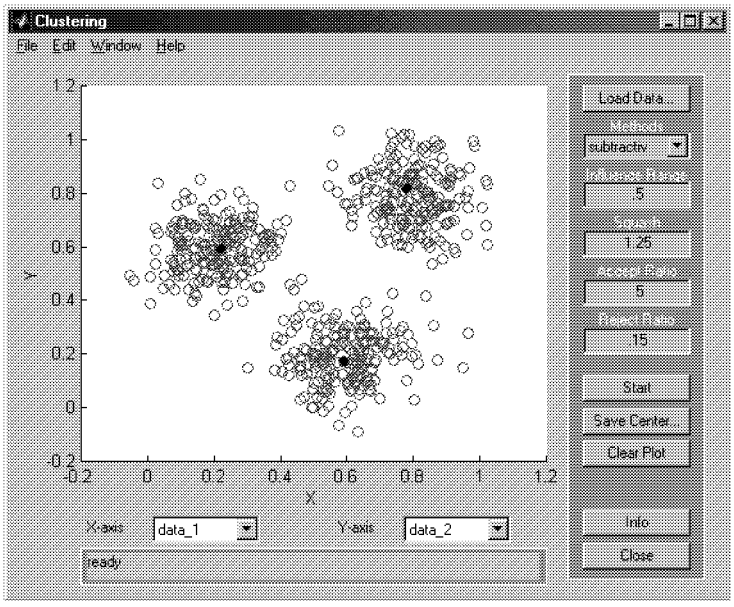

Рис. 5.21. Результат работы программы Clustering (центры кластеров окрашены в черный цвет)

кластеризация»). Если не вдаваться в их детальное теоретическое изложение, а ограничиться выявлением различий на уровне пользователя, то можно отметить, что алгоритм Fuzzy c-means, являясь, пожалуй, более точным (если понятие точности вообще здесь применимо), для своей работы требует задания таких опций, как число кластеров (кнопка Cluster num.) и число итераций (кнопка Max Iteration#). Ну, если число итераций еще можно задать как-то наугад, то ошибка в задании числа кластеров может привести к неприятным последствиям. Алгоритм Subtractive clustering менее точен, но и менее требователен к априорной информации; при работе с ним можно сохранить опции, заданные в программе по умолчанию. На рис. 5.21 приведен пример использования программы для фала данных clusterdemo.dat из директории Matlab/toolbox/fuzzy/fuzdemos/ при использовании алгоритма Subtractive clustering. Заметим, что выводится только двумерное поле рассеяния, но изменяя переменные в соответствующих полях (X-axis и Y-axis), можно «просмотреть» все многомерное пространство переменных.

5.5. Работа с Fuzzy Logic Toolbox в режиме командной строки

5.5.1. Возможности работы в режиме командной строки. Пакет Fuzzy Logic располагает большим набором функций, исполняемых из командной строки MATLAB и позволяющих, в принципе, не использовать при работе с системами нечеткого вывода рассмотренные программы графического интерфейса. Все функции делятся на группы:

1) вызова программ графического интерфейса;
2) задания функций принадлежности;
3) создания, редактирования, просмотра, открытия и сохранения систем нечеткого вывода;
4) дополнительные;
5) различные;
6) вызова диалоговых окон интерфейса;
7) блоков Simulink;
8) демонстрации возможностей пакета.

5.5.2. Функции вызова программ графического интерфейса. К этой группе относятся функции:

fuzzy — вызов FIS-редактора;
mfedit — вызов редактора функций принадлежности;
ruleedit — вызов редактора правил;
ruleview — вызов программы просмотра правил;

surfview — вызов программы просмотра поверхности отклика;
anfisedit — вызов ANFIS-редактора (только для систем, использующих алгоритм Sugeno и имеющих одну выходную переменную);
findcluster — вызов программы кластеризации.

Использование первых шести функций с аргументом (например, **fuzzy(a)**, где **a** — имя переменной рабочего пространства, присвоенное системе нечеткого вывода), открывает соответствующую программу с одновременной загрузкой в нее рассматриваемой системы.

Функция **findcluster(имя_файла)** открывает программу кластеризации с одновременной загрузкой указанного файла данных.

5.5.3. Задание функций принадлежности. В данную группу включены 11 функций (по числу функций принадлежности, используемых в пакете Fuzzy Logic).

1. Функция **dsigmf**.
Запись: **y = dsigmf(x,[a1 c1 a2 c2])**

Описание. Задается функция принадлежности, определяемая как разность двух сигмоидальных функций. Сигмоидальная функция, как известно, описывается выражением

$$f(x,\, a,\, c) = \frac{1}{1 + \exp\left(-a(x - c)\right)}$$

и зависит от двух числовых параметров a и c. Описываемая функция, как отмечено, является разностью двух сигмоидальных:

$$f_1(x,\, a_1,\, c_1) - f_2(x,\, a_2,\, c_2),$$

и зависит от четырех параметров a_1, c_1, a_2, c_2 (вектора параметров [a1 c1 a2 c2]).

Пример

```
» x = 0:0.1:10;
» y = dsigmf(x,[5 2 5 7]);
» plot(x,y)
» xlabel('dsigmf, P = [5 2 5 7]')
```

2. Функция **gauss2mf**.
Запись: **y = gauss2mf(x,[sig1 c1 sig2 c2])**

Описание. Задается функция принадлежности, являющаяся разностью двух гауссовых функций, определяемая соотношением

$$f(x,\, \sigma_1,\, c_1,\, \sigma_2,\, c_2) = \exp\left(-(x - c_1)^2/\sigma_1^2\right) - \exp\left(-(x - c_2)^2/\sigma_2^2\right)$$

и зависящую от четырех параметров $(\sigma_1, c_1, \sigma_2, c_2)$ или вектора параметров [sig1 c1 sig2 c2].

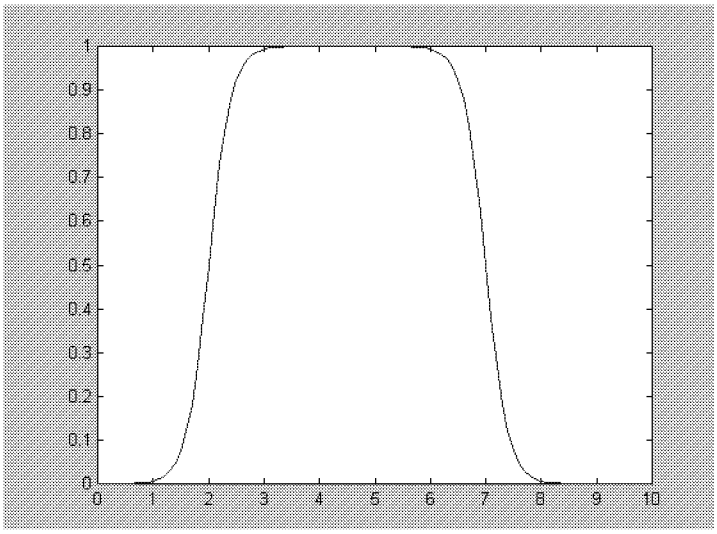

Рис. 5.22. Вид функции **dsigmf(x,[5 2 5 7])**

Пример

```
» x = (0:0.1:10)';
» y1 = gauss2mf(x, [2 4 1 8]);
» y2 = gauss2mf(x, [2 5 1 7]);
» y3 = gauss2mf(x, [2 6 1 6]);
» y4 = gauss2mf(x, [2 7 1 5]);
» y5 = gauss2mf(x, [2 8 1 4]);
» plot(x, [y1 y2 y3 y4 y5]);
» set(gcf, 'name', 'gauss2mf', 'numbertitle', 'off');
```

3. Функция **gaussmf**.

Запись: **y = gaussmf(x,[sig c])**

Описание. Задается функция принадлежности гауссова типа, зависящая от двух параметров ([**sig c**]).

Пример

```
» x = 0:0.1:10;
» y = gaussmf(x,[2 5]);
» plot(x,y)
» xlabel('gaussmf, P=[2 5]')
```

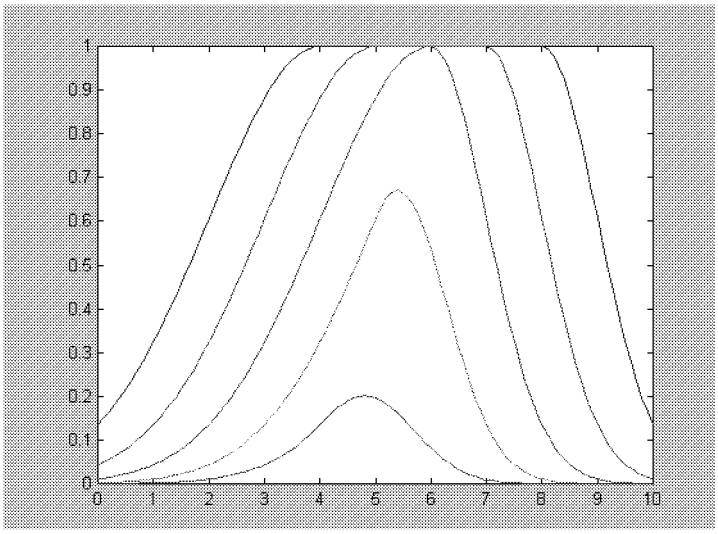

Рис. 5.25. Семейство кривых, определяемых функцией **gauss2mf**

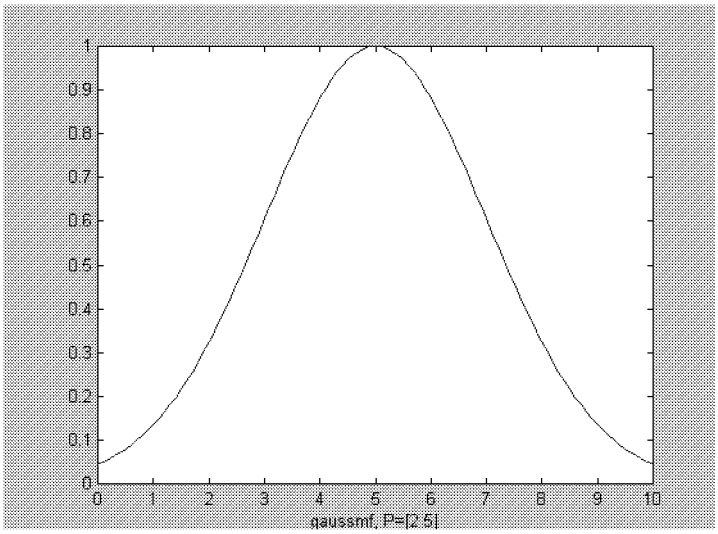

Рис. 5.24. Функция принадлежности типа гауссовой кривой

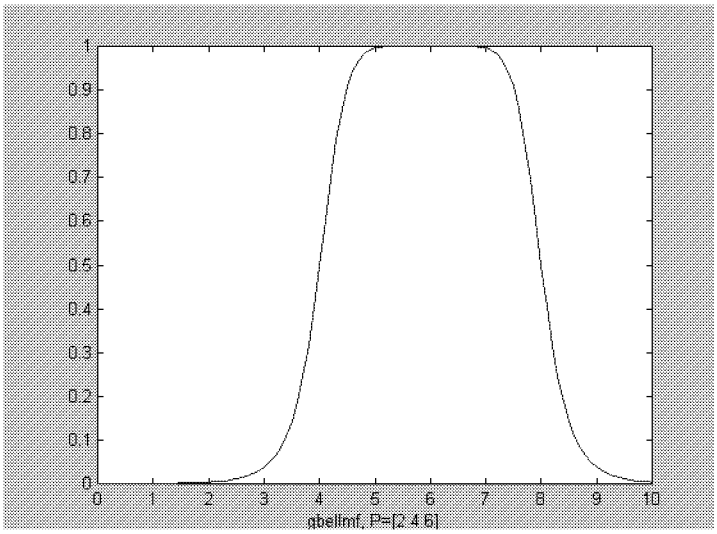

Рис. 5.26. График обобщенной колоколообразной функции

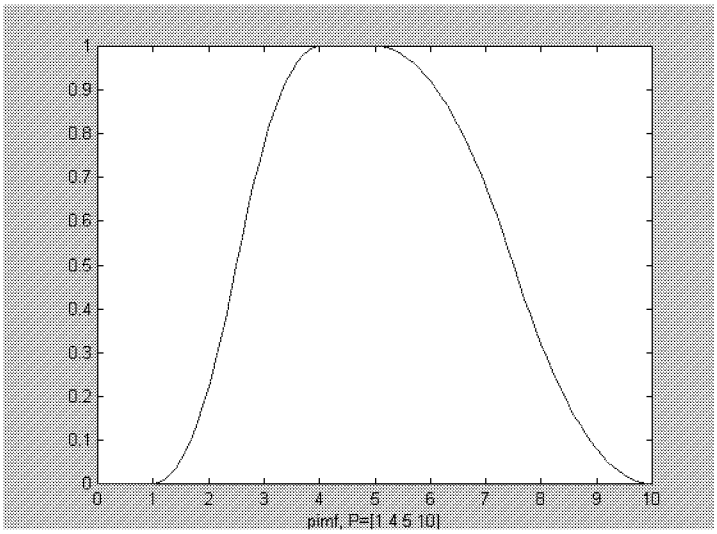

Рис. 5.26. График π-образной функции

4. Функция **gbellmf**.
Запись: **y = gbellmf(x,params)**

Описание. Задается функция принадлежности так называемого обобщенного колоколообразного типа с аналитическим описанием, зависящим от трех числовых параметров:

$$(x,\, a,\, b,\, c) = \frac{1}{1 + |(x - c)/a|^{2b}}.$$

Пример

```
» x = 0:0.1:10;
» y = gbellmf(x,[2 4 6]);
» plot(x,y)
» xlabel('gbellmf, P = [2 4 6]')
```

5. Функция **pimf**.
Запись: **y = pimf(x,[a b c d])**

Описание. Задается так называемую π-образная функция принадлежности, получившая свое название из-за своеобразной формы. Функция вычисляется с использованием сплайн аппроксимации по четырем точкам, задаваемым вектором параметров. Параметры a и d определяют основание кривой, а параметры b и c — положение плоской вершины.

Пример

```
» x = 0:0.1:10;
» y = pimf(x,[1 4 5 10]);
» plot(x,y)
» xlabel('pimf, P=[1 4 5 10]')
```

6. Функция **psigmf**.
Запись: **y = psigmf(x,[a1 c1 a2 c2])**

Описание. Задается функция принадлежности, определяемая как произведение двух сигмоидальных функций

$$f_1(x,\, a_1,\, c_1) - f_2(x,\, a_2,\, c_2),$$

и зависящая от четырех параметров a_1, c_1, a_2, c_2 (вектора параметров [a1 c1 a2 c2]).

Пример

```
» x = 0:0.1:10;
» y = psigmf(x,[2 3 -5 8]);
» plot(x,y)
» xlabel('psigmf, P = [2 3 -5 8]')
```

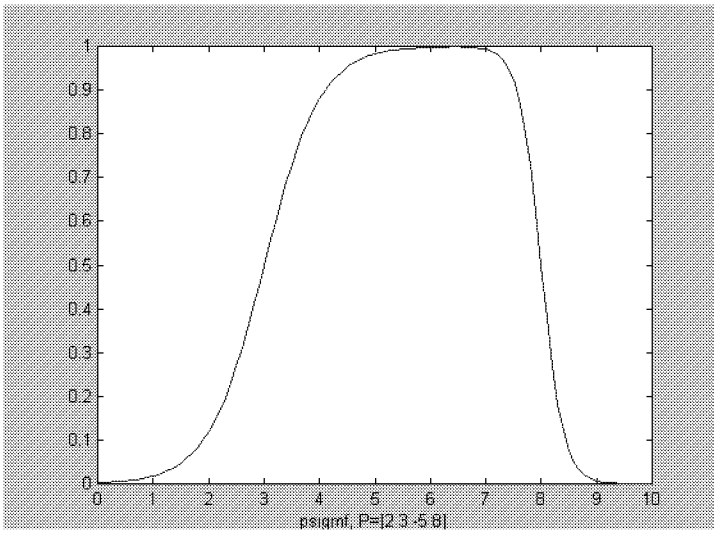

Рис. 5.27. Пример использования функции **psigmf**

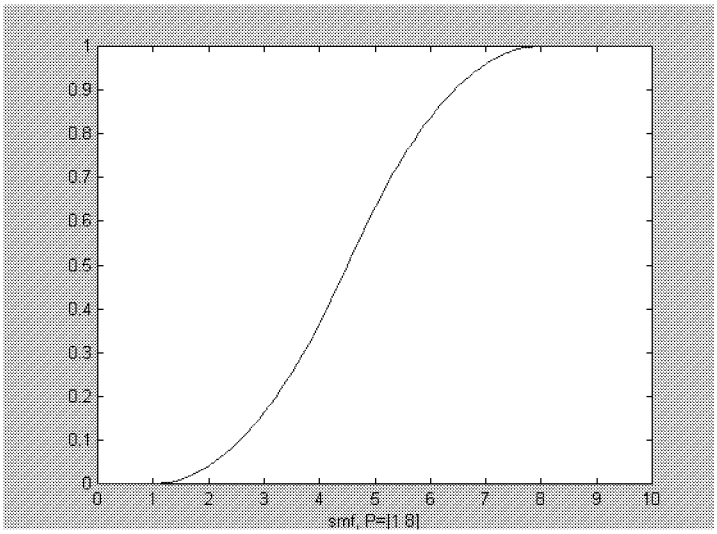

Рис. 5.28. График S-образной функции

7. Функция **smf**.
Запись: **y = smf(x,[a b])**

Описание. Задается зависящая от двух параметров так называемая S-образная функция принадлежности. Параметры a и b определяют диапазон значений аргумента, где функция возрастает.

Пример

```
» x = 0:0.1:10;
» y = smf(x,[1 8]);
» plot(x,y)
» xlabel('smf, P = [1 8]')
```

8. Функция **sigmf**.
Запись: **y = sigmf(x,[a c])**

Описание. Задается так называемая сигмоидальная функция принадлежности, определяемая выражением, приведенным выше и зависящим от двух параметров.

Пример

```
» x = 0:0.1:10;
» y = sigmf(x,[2 4]);
» plot(x,y)
» xlabel('sigmf, P = [2 4]')
```

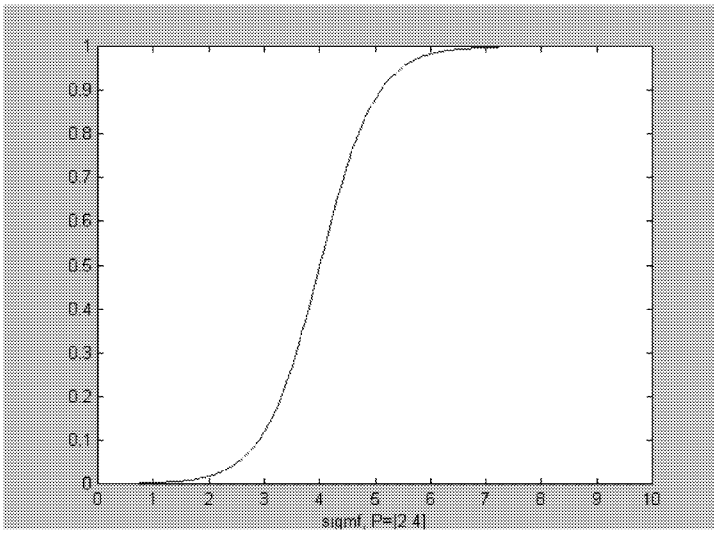

Рис. 5.29. График сигмоидальной функции

9. Функция **trapmf**.
Запись: **y = trapmf(x,[a b c d])**

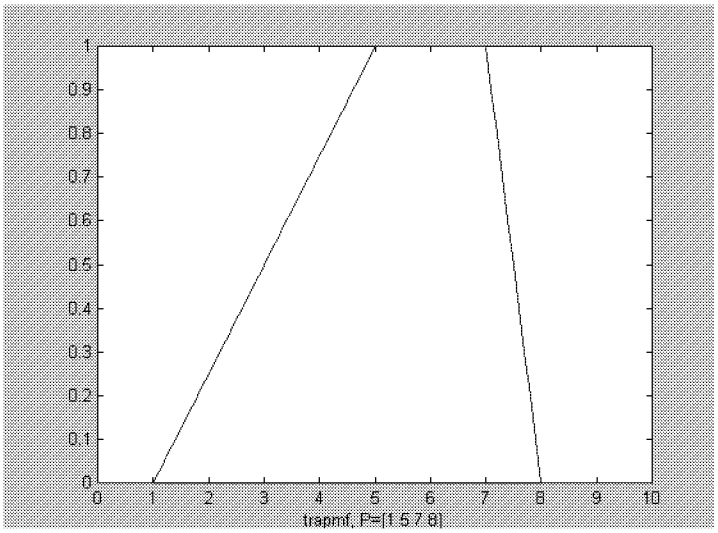

Рис. 5.30. Трапецеидальная функция принадлежности

Описание. Задается так называемая трапецеидальная функция принадлежности, зависящая от четырех параметров и определяемая формулой

$$f(x,\,a,\,b,\,c,\,d) = \left\{ \begin{array}{ll} 0, & x \leqslant a \\ \dfrac{x-a}{b-a}, & a \leqslant x \leqslant b \\ 1, & b \leqslant x \leqslant c \\ \dfrac{d-x}{d-c}, & c \leqslant x \leqslant d \\ 0, & d \leqslant x \end{array} \right\},$$

при этом параметры a и d определяют основание кривой, a, b и c — положение вершины.

Пример

```
» x = 0:0.1:10;
» y = trapmf(x,[1 5 7 8]);
» plot(x,y)
» xlabel('trapmf, P = [1 5 7 8]')
```

10. Функция **trimf**.
Запись: **y = trimf(x,params)** или **y = trimf(x,[a b c])**

О п и с а н и е. Задается зависящая от трех параметров функция принадлежности треугольной формы, при этом параметры a и c определяют основание треугольника, а параметр b — координату его вершины.

П р и м е р

```
» x = 0:0.1:10;
» y = trimf(x,[3 6 8]);
» plot(x,y)
» xlabel('trimf, P=[3 6 8]')
```

11. Функция **zmf**.
Запись: **y = zmf(x,[a b])**

О п и с а н и е. Задается зависящая от двух параметров функция принадлежности так называемой Z-образной формы. Параметры определяют диапазон убывания функции.

П р и м е р

```
» x = 0:0.1:10;
» y = zmf(x,[3 7]);
» plot(x,y)
» xlabel('zmf, P = [3 7]')
```

5.5.4. Функции сохранения, открытия и использования созданной системы. Чтение, использование и сохранение на диске созданной системы нечеткого вывода в режиме командной строки осуществляется функциями

readfis('имя_файла'),
evalfis(вектор_параметров, имя),
writefis(имя) или **writefis(имя, 'имя_файла')**

Здесь **имя_файла** — наименование файла с записанной системой (без указания расширения), **имя** — идентификатор, который определен системе в рабочей среде MATLAB, **вектор_параметров** — набор значений входов, для которых требуется рассчитать выход (возможна и матрица параметров, тогда результат расчетов — вектор в случае одной выходной переменной или матрица при нескольких таких переменных).

П р и м е р ы (применительно к ранее созданной и сохраненной на диске в виде файла с именем tip экспертной системе для задачи о чаевых):

```
» readfis('tip');
» evalfis([1 2],a)
```

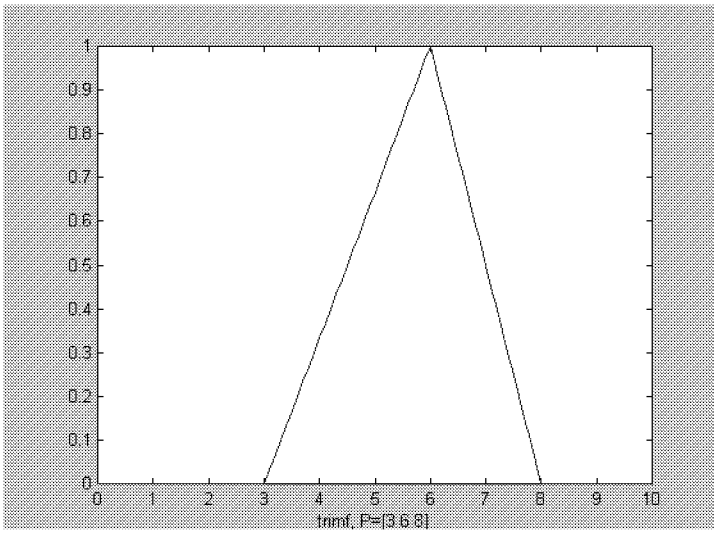

Рис. 5.31. График функции принадлежности треугольной формы

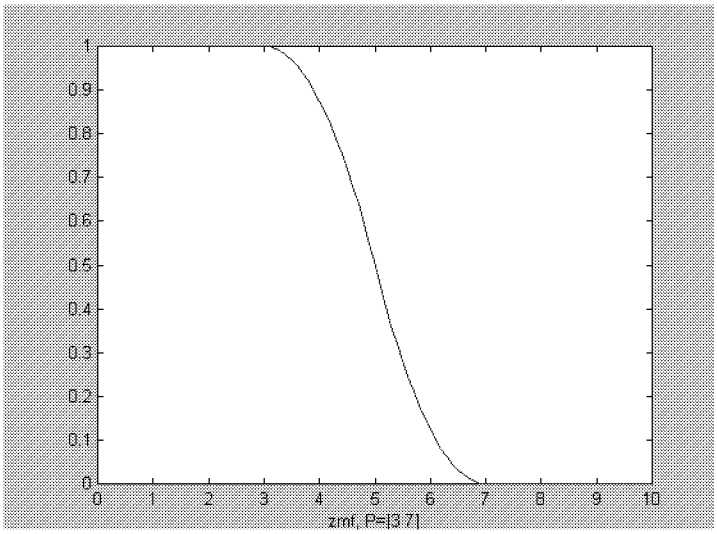

Рис. 5.32. Функция принадлежности Z-образной формы

```
ans =
  7.3949
» writefis(a)
ans =
tip
```

5.5.5. Функции использования графического окна. Следующие три функции позволяют использовать элементы графических изображений вне программ с графическим интерфейсом.

1. Команда (функция) **plotfis(a)** вызовет появление графического окна MATLAB с мнемоническим представлением разработанной системы (рис. 5.33).

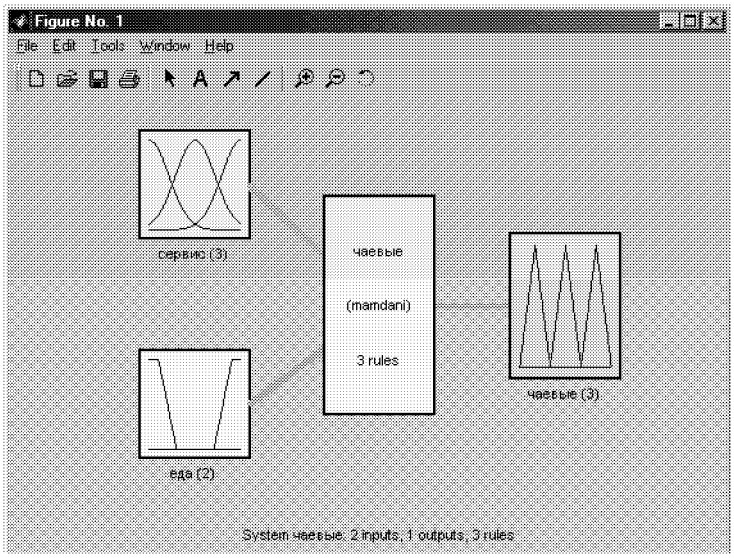

Рис. 5.33. Мнемоническое представление системы нечеткого вывода

2. Команда (функция) **plotmf** выполняет аналогичную операцию, но по отношению к графикам функций принадлежности. На рис. 5.34 приведен результат выполнения функции **plotmf(a,'input',1)**.

3. Наконец, команда (функция) **gensurf(a)** также делает то же самое, но применительно к поверхности отклика (рис. 5.35).

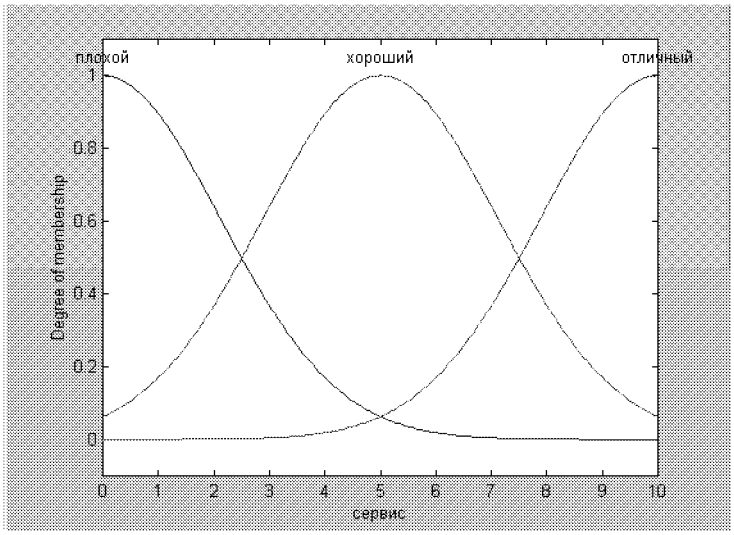

Рис. 5.34. Результат выполнения функции **plotmf(a,'input',1)**

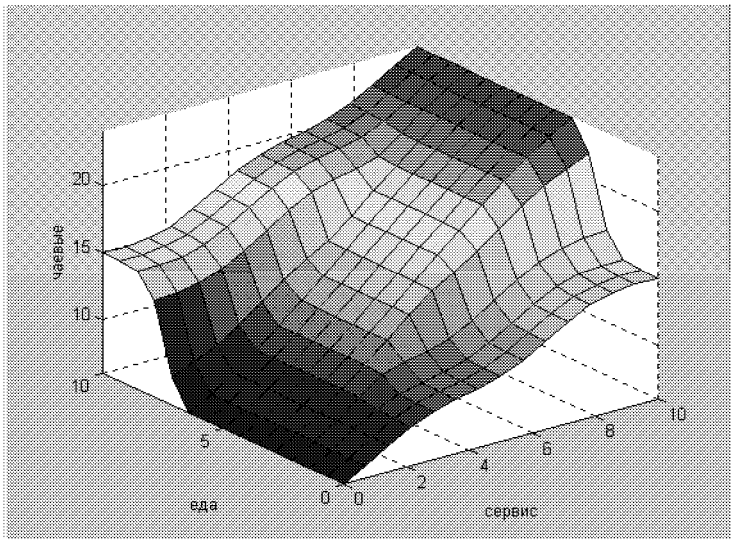

Рис. 5.35. Результат выполнения функция **gensurf(a)**

5.5.6. Функции создания, просмотра структуры и редактирования систем нечеткого вывода. Вообще, при желании можно совсем обойтись без программ графического интерфейса и, используя функции **newfis** (новая система), **addvar** (добавить переменную), **addmf** (добавить функцию принадлежности) и **addrule** (добавить правило), сконструировать систему нечеткого вывода целиком в режиме командной строки MATLAB. Процесс этот, надо сказать, значительно более трудоемкий, чем с применением указанных программ, поэтому, не вдаваясь особенно в детали, приведем лишь пример построения такой системы в задаче о чаевых:

```
a = newfis('tip');
a = addmf(a,'input',1,'сервис',[0 10]);
a = addmf(a,'input',1,'плохой','gaussmf',[1.5 0]);
a = addmf(a,'input',1,'хороший','gaussmf',[1.5 5]);
a = addmf(a,'input',1,'отличный','gaussmf',[1.5 10]);
a = addvar(a,'input','еда',[0 10]);
a = addmf(a,'input',2,'подгоревшая','trapmf',[-2 0 1 3]);
a = addmf(a,'input',2,'превосходная','trapmf',[7 9 10 12]);
a = addvar(a,'output','tip',[0 30]);
a = addmf(a,'output',1,'малые','trimf',[0 5 10]);
a = addmf(a,'output',1,'средние','trimf',[10 15 20]);
a = addmf(a,'output',1,'щедрые','trimf',[20 25 30]);
ruleList = [...
1 1 1 1 2
2 0 2 1 1
3 2 3 1 2];
a = addrule(a,ruleList);
```

При необходимости после просмотра элементов структуры (системы с идентификатором **a**) в режиме командной строки следует ввести команду вида **a.метка**, например,

```
» a.type
```

Получим ответ:

```
ans =
mamdani
```

Получение информации по всем элементам структуры обеспечивается функцией **getfis(a)**.

Результат ее выполнения:

```
Name = tip
  Type = mamdani
  NumInputs = 2
  InLabels =
    сервис
    еда
  NumOutputs = 1
  OutLabels =
```

```
    чаевые
  NumRules = 3
  AndMethod = min
  OrMethod = max
  ImpMethod = min
  AggMethod = max
  DefuzzMethod = centroid
ans =
tip
```

Функция **setfis** в определенном смысле противоположна функции **getfis** и позволяет изменять метки. Пример выполнения функции:

```
  » a = setfis(a,'name','чаевые')
  a =
name: 'чаевые'
type: 'mamdani'
andMethod: 'min'
orMethod: 'max'
defuzzMethod: 'centroid'
impMethod: 'min'
aggMethod: 'max'
input: [1×2 struct]
output: [1×1 struct]
rule: [1×3 struct]
```

Полный просмотр структуры системы нечеткого вывода осуществляется функцией **showfis(a)**.

Просмотр правил, включенных в систему нечеткого вывода, осуществляется с помощью функции **showrule**.

Запись:

showrule(имя)

showrule(имя,номера_правил)

showrule(fis,номера_правил,формат)

showrule(fis,номера_правил,формат,язык)

О п и с а н и е. В функции **имя** — это идентификатор рассматриваемой системы в среде MATLAB,

номера_правил — список правил, которые нужно показать, **формат** — строковая переменная, имеющая значения **'verbose'**, **'symbolic'** или **'indexed'**, **язык** — строковая переменная со значениями **'english'**, **'francais'** или **'deutsch'** (установки по умолчанию — **'verbose'** и **'english'**).

П р и м е р ы

```
» a = readfis('tip');
» showrule(a,1)
ans =
```

1. If (сервис is плохой) or (еда is подгоревшая) then (чаевые is малые) (1)
» showrule(a,[3 1],'symbolic')
ans =
3. (сервис==отличный)|(еда==превосходная)⇒ (чаевые=щедрые) (1)
1. (сервис==плохой)|(еда==подгоревшая)⇒ (чаевые=малые) (1)

Функция **rmmf** используется для удаления функции принадлежности из состава системы.

Запись:

имя = rmmf(имя,'varType',varIndex,'mf',mfIndex)

Описание. Параметры функции: **имя** — идентификатор системы в среде MATLAB, **varType** — строковая переменная со значениями **'input'** или **'output'**, **varIndex** — порядковый номер переменной (по списку переменных входа или выхода), **mf** — строковая переменная, задающая функцию принадлежности, **mfIndex** — порядковый данной функции.

Функция **rmvar** удаляет переменную из состава системы.

Запись:

[новое_имя,errorStr] = rmvar(имя,'varType',varIndex)
новое_имя = rmvar(имя,'varType',varIndex)

Описание. Здесь переменные **varType** и **varIndex** имеют тот же смысл, что и в предыдущей функции, переменная **errorStr** позволяет записать любое сообщение об ошибке, **новое_имя** — новое имя системы с исключенной переменной.

Функция приведения к четкости (дефазификации) **defuzz** позволяет по заданной функции принадлежности определить соответствующее четкое значение.

Запись: **out = defuzz(x,mf,type)**

Описание. Здесь **x** — обозначение числового аргумента, **mf** — тип функции принадлежности, **type** — задаваемый метод приведения к четкости — строковая переменная, имеющая возможные значения:

- **centroid;**
- **bisector;**
- **mom;**
- **som;**
- **lom.**

Смысл этих значений был рассмотрен ранее.

Пример

» x = -10:0.1:10;

```
» mf = trapmf(x,[-10 -8 -4 7]);
» xx = defuzz(x,mf,'centroid')
xx =
  -3.2857
```

Функция **evalmf** вычисляет значения заданной функции принадлежности.

Запись: **y = evalmf(x,mfParams,mfType)**

Описание. Здесь **x** — обозначение числового аргумента, **mfParams** — вектор необходимых числовых параметров, **mfType** — тип функции принадлежности.

Пример (см. рис. 5.36)

```
» x = 0:0.1:10;
» mfparams = [2 4 6];
» mftype = 'gbellmf';
» y = evalmf(x,mfparams,mftype);
» plot(x,y)
» xlabel('gbellmf, P = [2 4 6]')
```

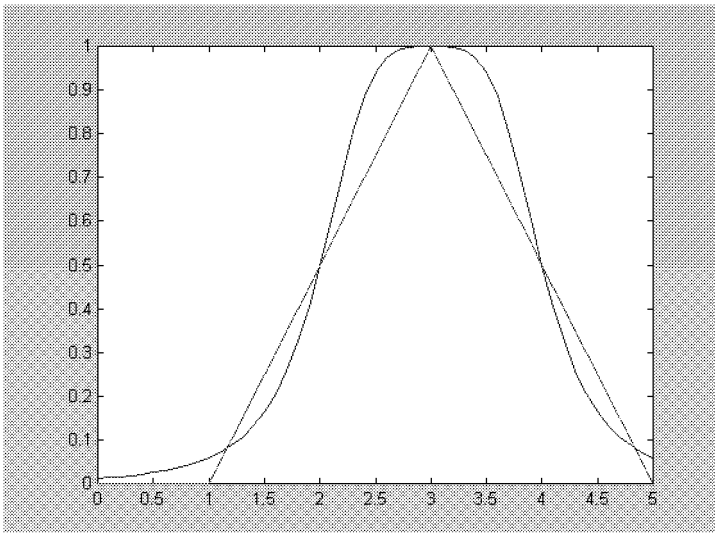

Рис. 5.36. Графическое представление результата выполнения команды **y = = evalmf(x,mfParams,mfType)**

Функция **mf2mf** позволяет заменять одну функцию принадлежности близкой ей по форме другой с некоторым «эквивалентным» набором числовых параметров.

Запись: **outParams = mf2mf(inParams,inType,outType)**

Описание. **inParams** — набор параметров исходной функции, **inType** — тип исходной функции принадлежности, **outType** — тип эквивалентной функции принадлежности, **outParams** — набор преобразованных параметров.

Пример

```
» x = 0:0.1:5;
» mfp1 = [1 2 3];
» mfp2 = mf2mf(mfp1,'gbellmf','trimf')
mfp2 =
   1    3    5
» plot(x,gbellmf(x,mfp1),x,trimf(x,mfp2))
```

Функцию **parsrule** можно отнести к числу сервисных.

Запись:

новое_имя = parsrule(имя,текст_правил)

новое_имя = parsrule(имя, текст_правил,формат)

новое_имя = parsrule(имя, текст_правил,формат,язык)

Описание. В данной функции идентификаторы **имя** и **новое_имя** относятся к исходной и модернизированной системам

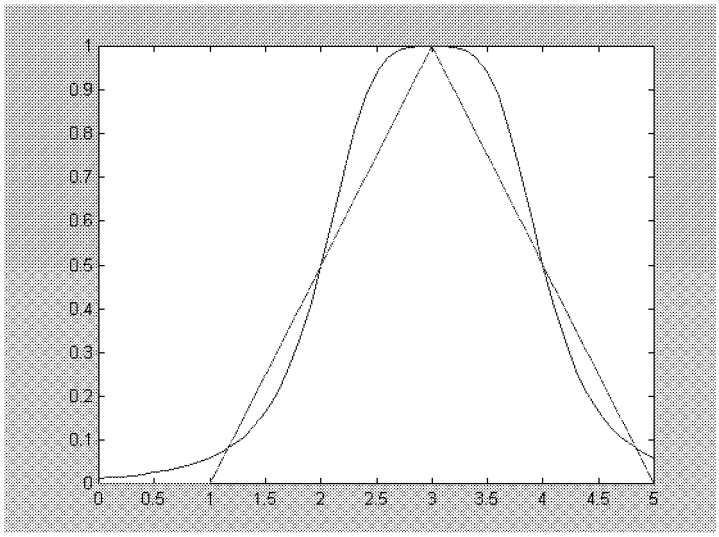

Рис. 5.37. Графики исходной (колоколообразной) и эквивалентной (треугольной) функций принадлежности

с нечетким выводом, строковые переменные **формат** и **язык** имеют тот же смысл, что и в рассмотренной ранее функции **showrule**,

текст правил — заданный в текстовой форме набор правил системы (на языке, определенной переменной **язык**; по умолчанию — английский, что допускает использование, в частности, русских имен для переменных, но требует английских связующих слов if, and, or, then, is). Функция выполняет грамматический разбор исходного текста и выдает набор нечетких правил в указанном формате (по умолчанию — подробный).

5.5.7. Функция создания и/или обучения гибридных сетей с архитектурой ANFIS. Функция **anfis** записывается в виде:

[имя,ошибка_о,шаг] = anfis(trnData)

[имя,ошибка_о,шаг] = anfis(trnData,имя)

[имя1,ошибка_о,шаг] = ...

anfis(trnData,имя,trnOpt,dispOpt)

[имя1,ошибка_о,шаг,имя2,ошибка_п] = ...

anfis(trnData,trnOpt,dispOpt,chkData)

[имя1,ошибка_о,шаг,имя2,ошибка_п] = ...

anfis(trnData,trnOpt,dispOpt,chkData,optMethod)

Функция предназначена для создания и/или обучения гибридных сетей с архитектурой ANFIS. Значения аргументов функции:

• **trnData** — матрица данных для обучения сети (обучающая выборка); последний столбец соответствует (единственной) выходной переменной, остальные столбцы — входным переменным;

• **имя** — идентификатор создаваемой гибридной сети; если структура системы нечеткого вывода с таким идентификатором уже создана, то она будет использована для настройки числовых параметров, в противном случае структура будет создана при выполнении функции с опциями, по умолчанию соответствующими выполнению функции **genfis1** (см. ниже);

• **trnOpt** — вектор опций обучения с элементами:

trnOpt(1): количество циклов обучения (по умолчанию 10),

trnOpt(2): целевой уровень ошибки обучения (по умолчанию 0),

trnOpt(3): начальный шаг алгоритма обучения (по умолчанию 0.01),

trnOpt(4): коэффициент уменьшения шага (по умолчанию 0.9),

trnOpt(5): коэффициент увеличения шага (по умолчанию 1.1);

• **dispOpt** — вектор опций вида выводимой информации (по умолчанию все элементы — единичные, что означает вывод всех видов возможной информации в процессе выполнения функции) с элементами:

dispOpt(1): **ANFIS**-информация, такая, как число входов, функции принадлежности и т. п.,

dispOpt(2): ошибка,

dispOpt(3): шаг обновления (корректировки) по каждому параметру,

dispOpt(4): конечные результаты;

• **chkData** — матрица проверочных данных (проверочная выборка), имеющая такой же набор столбцов, что и матрица данных для обучения сети, но, вообще говоря, другое число строк;

• **optMethod** — параметр, определяющий вид алгоритма обучения гибридной системы (1 — гибридный алгоритм, 0 — алгоритм обратного распространения ошибки; по умолчанию — 1).

Процесс обучения сети заканчивается, когда выполнено заданное число циклов обучения или ошибка обучения уменьшилась до заданного уровня. При отсутствии некоторых аргументов их значения принимаются по умолчанию.

Выходные параметры функции:

• **ошибка_о** — массив (вектор) значений ошибок для обучающей выборки;

• **ошибка_п** — массив (вектор) значений ошибок для проверочной выборки;

• **шаг** — массив значений шага в алгоритме обучения;

• **имя1** — идентификатор (имя) сети, образованной исходя из минимальной ошибки обучения;

• **имя2** — идентификатор (имя) сети, образованной исходя из минимальной ошибки для проверочной выборки.

Пример 1

```
» x = (0:0.1:10)';
» y = sin(2*x)./exp(x/5);
» trnData = [x y];
» a = anfis(trnData);
ANFIS info:
  Number of nodes: 12
  Number of linear parameters: 4
  Number of nonlinear parameters: 6
  Total number of parameters: 10
  Number of training data pairs: 101
  Number of checking data pairs: 0
  Number of fuzzy rules: 2
Start training ANFIS ...
  1    0.313942
  2    0.31388
  3    0.313817
  4    0.313753
  5    0.313689
Step size increases to 0.011000 after epoch 5.
```

```
    6    0.313624
    7    0.313552
    8    0.313479
    9    0.313406
  Step size increases to 0.012100 after epoch 9.
   10    0.313332
  Designated epoch number reached → ANFIS training
completed at epoch 10.
  » plot(x,y,x,evalfis(x,a),'-.');
  » legend('Обучающая выборка','Выход ANFIS');
```

Пример 2

```
  » x = (0:0.1:10)';
  » y = sin(2*x)./exp(x/5);
  » trnData = [x y];
  » numMFs = 5;
  » mfType = 'gbellmf';
  » epoch_n = 20;
  » in_fismat = genfis1(trnData,numMFs,mfType);
  » out_fismat = anfis(trnData,in_fismat,20);
  ANFIS info:
    Number of nodes: 24
    Number of linear parameters: 10
    Number of nonlinear parameters: 15
    Total number of parameters: 25
    Number of training data pairs: 101
    Number of checking data pairs: 0
    Number of fuzzy rules: 5
  Start training ANFIS ...
    1       0.0694086
    2       0.0680259
    3       0.066663
    4       0.0653198
    5       0.0639961
  Step size increases to 0.011000 after epoch 5.
    6       0.0626917
    7       0.0612787
    8       0.0598881
    9       0.0585193
  Step size increases to 0.012100 after epoch 9.
   10       0.0571712
   11       0.0557113
   12       0.0542741
   13       0.052858
  Step size increases to 0.013310 after epoch 13.
   14       0.0514612
   15       0.0499449
   16       0.0484475
   17       0.0469667
```

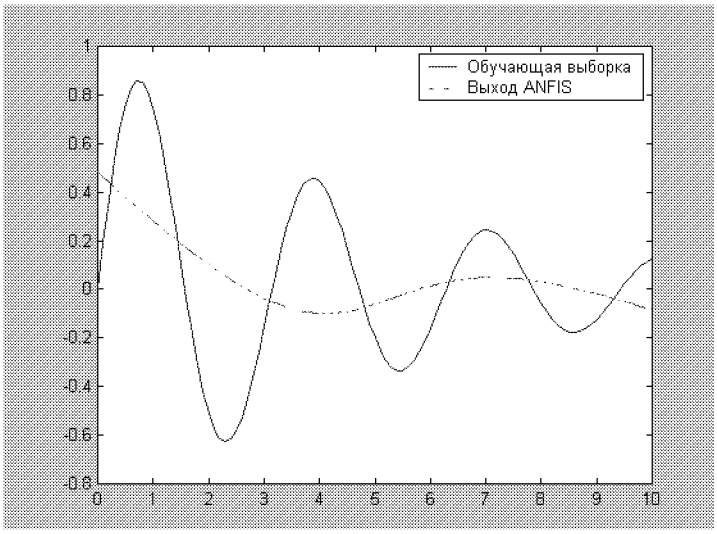

Рис. 5.38. Обучающая выборка и выход системы **a**

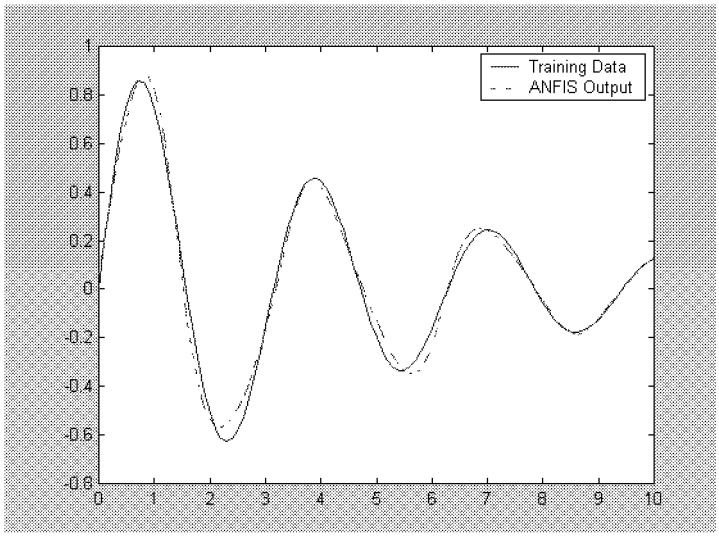

Рис. 5.39. Обучающая выборка и выход системы **fismat**

```
Step size increases to 0.014641 after epoch 17.
  18     0.0455003
  19     0.0439022
  20     0.0423184
Designated epoch number reached → ANFIS training
completed at epoch 20.
» plot(x,y,x,evalfis(x,out_fismat),'-.');
» legend('Training Data','ANFIS Output');
```

В примере 2 при проектировании гибридной системы заранее были заданы некоторые значения (количество и тип функций принадлежности, количество циклов обучения), в примере 1 для той же обучающей выборки все установки при проектировании системы выбраны по умолчанию. Отличие в качестве функционирования двух систем наглядно иллюстрируются рис. 5.38 и 5.39.

5.5.8. Функция кластеризации. Функция **fcm** осуществляет кластеризацию с использованием алгоритма Fuzzy c-means (см. выше). Она записывается в виде:

[center,U,obj_fcn] = fcm(data,cluster_n)

[center,U,obj_fcn] = fcm(data,cluster_n,options)

О п и с а н и е. Аргументы функции:

• **data** — матрица данных, каждый столбец которой соответствует одной из переменных, а каждая строка — одному из опытов;

• **cluster_n** — число задаваемых кластеров (должно быть больше 1);

• **options** — вектор опций функции с элементами:

options(1): показатель для матрицы **U** (по умолчанию 2.0),

options(2): максимальное число итераций при определении центров кластеров (по умолчанию 100),

options(3): минимальная сумма улучшения (по умолчанию 1e-5),

options(4): вывод информации в процессе итераций (по умолчанию 1).

Выходные параметры функции:

• **center** — матрица, элементы которой (по столбцам) являются координатами найденных центров кластеров, число строк равно числу центров;

• **U** — матрица значений принадлежности данных к выявленным кластерам;

• **obj_fcn** — значения целевой функции в процессе итераций.

Пример

```
» load clusterdemo.dat;
» data = clusterdemo;
» [center,U,obj_fcn] = fcm(data, 3);
Iteration count = 1, obj. fcn = 55.941585
Iteration count = 2, obj. fcn = 42.971277
Iteration count = 3, obj. fcn = 42.764109
Iteration count = 4, obj. fcn = 41.688335
Iteration count = 5, obj. fcn = 37.070571
Iteration count = 6, obj. fcn = 29.262573
Iteration count = 7, obj. fcn = 22.996848
Iteration count = 8, obj. fcn = 16.162985
Iteration count = 9, obj. fcn = 15.443897
Iteration count = 10, obj. fcn = 15.430759
Iteration count = 11, obj. fcn = 15.430533
Iteration count = 12, obj. fcn = 15.430528
» center
center =
   0.2051    0.5960    0.8090
   0.7939    0.7892    0.1968
   0.5963    0.1923    0.5057
```

5.5.9. Функция генерации FIS-структуры. Функция **genfis1** генерирует FIS-структуру по экспериментальным данным без проведения их кластеризации:

имя = genfis1(data)

имя = genfis1(data,numMFs,inmftype, outmftype)

Описание. Функцией генерируется структура системы нечеткого вывода типа Sugeno, являющая исходной для последующего формирования и обучения гибридной системы с помощью функции **anfis**. Аргументы функции:

- **data** — матрица данных (обучающая выборка), последний столбец которой соответствует выходной переменной, а остальные — входным переменным и число строк в которой равно числу наборов экспериментальных данных (образцов);
- **numMFs** — вектор, элементы которого определяют число функций принадлежности, задаваемых для каждого входа; если для всех входов задается одно и то число таких функций, данный аргумент задается как скаляр (один элемент);
- **inmftype** — строковый массив, элементы которого — типы функций принадлежности, задаваемые для входных переменных;
- **outmftype** — строковая переменная, определяющая тип функций принадлежности выходной переменной (возможны только значения **'linear'** или **'constant'**).

Число функций принадлежности выходной переменной равно числу нечетких правил, генерируемых **genfis1**, и зависит от элементов вектора **numMFs**. По умолчанию **numMFs = 2**, **inmftype = = 'gbellmf'**. Значения по умолчанию устанавливаются, если функция применяется без трех последних аргументов.

Применение функции ограничено размерностью задачи: так, при N входных переменных с минимальным разбиением области определения каждой переменной на две подобласти будет генерироваться 2^N правил, что при числе входов больше 5–6 делает анализ этих правил практически невозможным. Более экономной в этом плане является функция **genfis2** (см. ниже).

Пример

```
» data = [rand(10,1) 10*rand(10,1)-5 rand(10,1)];
» numMFs = [3 7];
» mfType = str2mat('pimf','trimf');
» fismat = genfis1(data,numMFs,mfType);
» [x,mf] = plotmf(fismat,'input',1);
» subplot(2,1,1), plot(x,mf);
» xlabel('input 1 (pimf)');
» [x,mf] = plotmf(fismat,'input',2);
» subplot(2,1,2), plot(x,mf);
» xlabel('input 2 (trimf)');
```

См. также пример 2 для функции **anfis**.

5.5.10. Функция генерации структуры нечеткого вывода. Функция **genfis2** генерирует структуру системы нечеткого вывода (типа Sugeno) с использованием алгоритма Subtractive clustering кластеризации данных:

имя = genfis2(Xin,Xout,radii)

имя = genfis2(Xin,Xout,radii,xBounds)

имя = genfis2(Xin,Xout,radii,xBounds,options)

Аргументами рассматриваемой функции являются:

• **Xin** — матрица (массив) входных данных обучающей выборки, столбцы которой ассоциированы с входными переменными, а каждая строка — с отдельным опытом;

• **Xout** — матрица выходных переменных обучающей выборки, столбцы которой представляют значения данных переменных, а число строк равно числу строк матрицы **Xin**;

• **radii** (радиусы) — вектор, определяющий «области влияния» центров кластеров по каждой входной переменной (в предположении, что область определения данных переменных — многомерный параллелепипед) с неотрицательными элементами, меньшими единицы; пусть, например, число входов — 3, тогда **radii =**

$=[0.5\ \ 0.4\ \ 0.3]$ означает, что по первой переменной «область влияния» составляет 0.5 от диапазона изменения этой переменной, а по второй и третьей — соответственно 0.4 и 0.3 (эти диапазоны определяются по элементам столбцов матрицы **Xin**). Если рассматриваемый аргумент задан как скаляр, то по всем переменным устанавливается один и тот же размер «области влияния» кластеров. Параметр имеет важное значение для проведения разбиения области определения входов на подобласти с последующим установлением числа нечетких правил;

• **XBounds** — матрица с 2 строками и N столбцами (N — число столбцов в матрице **Xin**, т. е. число входных переменных), устанавливающая границы областей определения входных переменных. Данные этой матрицы используются для преобразования (масштабирования) входов к диапазону $[-1,\ 1]$. Например, **Xbounds** $=[-10\ \ 0\ \ -1;\ \ 10\ \ 50\ \ 1]$ задает диапазон $[-10,\ 10]$ для первой переменной, $[0,\ 50]$ — для второй и $[-1,\ 1]$ — для третьей. Если рассматриваемый аргумент опущен, границы областей определяются как максимальные и минимальные элементы по столбцам матрицы **Xin**;

• **options** — вектор опций, устанавливающий параметры алгоритма кластеризации (подробней см. при описании функции **subclust**).

Функция **genfis2**, так же как **genfis1**, может применяться в комплекте с функцией **anfis** (для настройки параметров системы нечеткого вывода), но, в отличие от последней, **genfis2** возвращает уже полностью определенную, готовую к использованию систему, не требующую, вообще говоря, дополнительной оптимизации.

Функция **genfis2** производит более экономное, чем **genfis1** разбиение пространства входов на подобласти и может, поэтому, являться эффективным инструментов для выявления правил из набора экспериментальных данных.

Примеры

a = genfis2(Xin,Xout,0.5)

a = genfis2(Xin,Xout,[0.5 0.25 0.3])

5.5.11. Функция возврата центров кластеров. Функция **subclust** возвращает центры кластеров.

Запись: **[C,S] = subclust(X,radii,xBounds,options)**

Функцией определяются центры кластеров набора экспериментальных данных, отражаемых матрицей **X** (столбцы матрицы соответствуют переменным, строки — экспериментальным «точкам» или опытам) на основе реализации «вычитающего» алгоритма кластеризации (Subtractive clustering). В его основе лежит пред-

положение, что каждая экспериментальная точка может быть центром кластера, при этом вначале для каждой точки вычисляется

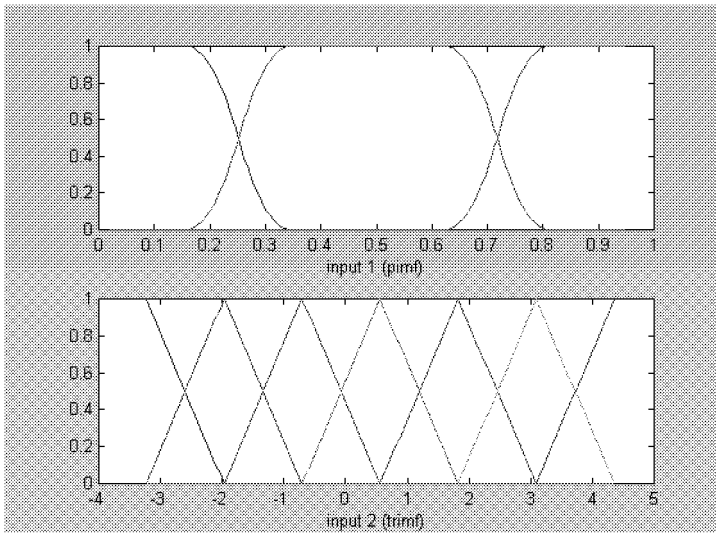

Рис. 5.40. Функции принадлежности, определенные **genfis1**

мера правдоподобия данного предположения («потенциал точки»), основанная на плотности точек в заданной окрестности рассматриваемой. Дальнейшие вычисления происходят итеративно:

- точка с наибольшим потенциалом объявляется центром первого кластера;
- из отмеченной окрестности этой точки удаляются все остальные точки;
- из оставшихся точек объявляется центр следующего кластера и т. д., пока не будут рассмотрены (исключены или объявлены центрами кластеров) все точки.

Размеры окрестностей задаются элементами вектора **radii**, который (как и аргумент **xBounds**) аналогичен рассмотренному при описании функции **Genfis2**.

Аргумент **options** является вектором со следующими элементами:

- **options(1) = quashFactor** (фактор подавления): используется для расширения «сферы влияния» выделенного центра кластера для «подавления» (уменьшения) потенциала точек, относящихся к данному кластеру; по умолчанию равен 1.25;

• **options(2) = acceptRatio** (коэффициент принятия): устанавливает, больше какого значения по отношению к потенциалу принятого центра кластера должен быть потенциал точки, чтобы эту точку можно было объявить центром нового кластера; по умолчанию 0.5;

• **options(3) = rejectRatio** (коэффициент отклонения): устанавливает, меньше какого значения по отношению к потенциалу принятого центра кластера должен быть потенциал точки, чтобы эту точку нельзя было объявить центром нового кластера; по умолчанию 0.15;

• **options(4) = verbose** (подробности): если этот элемент ненулевой, то выдается подробная информация и процессе нахождения центров кластеров; по умолчанию — 0.

Функция возвращает центры кластеров в виде матрицы **C**; каждая строка этой матрицы содержит координаты одного из найденных центров. Вектор **S** содержит элементы, определяющие диапазоны влияния центров кластеров по каждой переменной (для всех центров эти диапазоны одинаковы).

Пример

```
» X = load('clusterdemo.dat');
» [C,S] = subclust(X,0.5)
C =
   0.2220    0.5937    0.8113
   0.7797    0.8191    0.1801
   0.5914    0.1721    0.4872
S =
   0.1904    0.1988    0.2251
```

5.5.12. Различные другие функции. К этой группе относятся 10 функций:

1) **convertfis** — обеспечивает конвертацию FIS-матрицы пакета Fuzzy Logic версии 1.0 в FIS-структуру пакета Fuzzy Logic версии 2.0:

имя_новое = convertfis(имя старое)

2) **discfis** — преобразует непрерывные функции принадлежности системы нечеткого вывода в дискретные (задаваемые наборами точек).

3) **evalmmf** — вычисляет значения одновременной нескольких функций принадлежности. Является многомерным аналогом функции **evalmf**.

4) **fstrvcat** — объединяет заданные векторы и матрицы в одну матрицу:

Y = fstrvcat(y1,y2,y3,..)

Функция возвращает матрицу **Y**, представляющую собой объединение матриц-аргументов **y1,y2,y3,..**, при этом для выравнивания размеров итоговой матрицы она дополняется нулями. Элементы аргументов могут быть строкового типа, при этом их символы преобразуются в числовую форму.

5) **fuzarith** — выполняет алгебраические операции над нечеткими множествами:

C = fuzarith(X, A, B, operator)

Здесь **X** — вектор, выполняющий роль универсального множества, **A** и **B** — векторы, определяющие нечеткие подмножества-операнды, **C** — вектор, определяющий нечеткое подмножество —

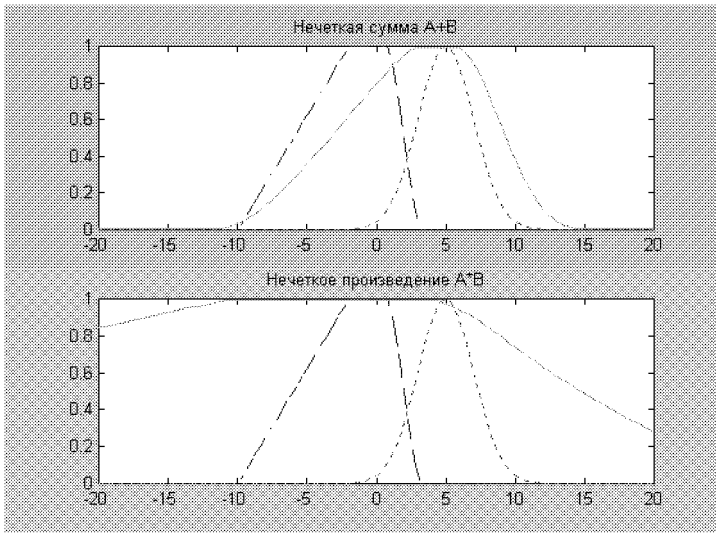

Рис. 5.41. Результаты выполнения функции **fuzarith**

результат операции, **operator** — строковая переменная, задающая вид операции и имеющая одно из возможных значений **'sum'** (сумма), **'sub'** (вычитание), **'prod'** (умножение), **'div'** (деление).

Пример

```
» point_n = 101; % Задание количества точек универсального множества
» x = linspace(min_x, max_x, point_n)';
» A = trapmf(x, [-10 -2 1 3]); % Трапецеидальная функция принадлежности подмножества A
» B = gaussmf(x, [2 5]); % Гауссова функция принадлежности подмножества B
```

```
» C1 = fuzarith(x, A, B, 'sum');
» subplot(2,1,1);
» plot(x, A, 'b--', x, B, 'm:', x, C1, 'c');
» title('Нечеткая сумма A+B');
» C2 = fuzarith(x, A, B, 'prod');
» subplot(2,1,2);
» plot(x, A, 'b--', x, B, 'm:', x, C2, 'c');
» title('Нечеткое произведение A*B');
```

6) **findrow** — возвращает номера аргументов функции **fstrvcat** по их именам.

7) **genparam** — возвращает параметры заданных функций принадлежности, определяя их по набору входных экспериментальных данных. Данные результаты могут быть использованы как начальные приближения при последующем применении команды **anfis**.

8) **mam2sug** — преобразует систему нечеткого вывода с алгоритмом Mamdani в систему, использующую алгоритм Sugeno:

имя_sug = mam2sug(имя_mam)

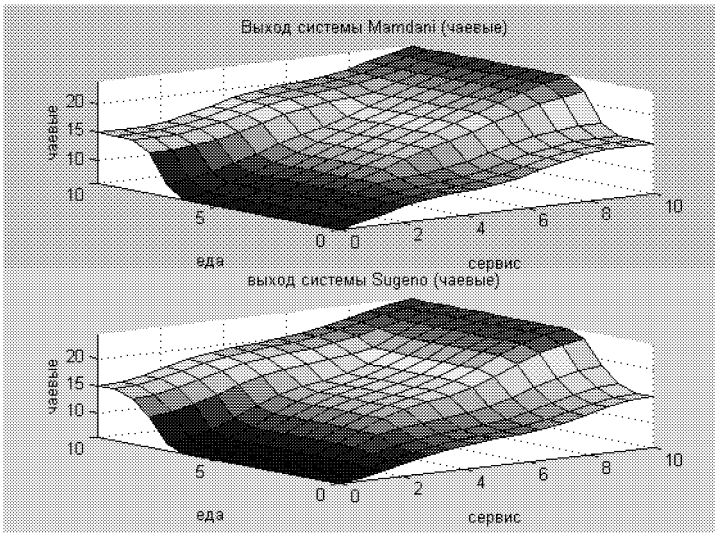

Рис. 5.42. Иллюстрация к выполнению функции **mam2sug**

Функция возвращает систему нечеткого вывода (с именем **имя_sug**), использующую алгоритм вывода Sugeno, по заданной

системе **имя_mam**, использующей алгоритм Mamdani. Возвращаемая система имеет постоянные функции принадлежности выходных переменных (допускается их число больше 1), определяемые как центроиды функций принадлежности следствий нечетких правил алгоритма Mamdani. Предпосылки правил при преобразовании не изменяются.

Пример

```
» a = readfis('tip');
» b = mam2sug(a);
» subplot(2,1,1); gensurf(a);
» title('Выход системы Mamdani (чаевые)');
» subplot(2,1,2); gensurf(b);
» title('выход системы Sugeno (чаевые)');
```

Замечание. Проверка, проведенная авторами, показала, что рассмотренная функция выполняется некорректно и приводит к неработающей системе вывода (попытки использования системы Sugeno вызывают сообщение об ошибке). Данный недостаток можно устранить, изменив в предпоследней строке файла mam2sug.m (в директории Matlab/toolbox/fuzzy/fuzzy/) **defuzzmethod** на **defuzzMethod**.

9) **probor** — выполняет операцию вероятностного ИЛИ над столбцами матрицы-аргумента:

Y = probor(X)

Если матрица-аргумент **X** имеет два столбца, **A** и **B**, то функция возвращает матрицу **Y = A+B − AB**; если аргумент содержит только один столбец, то **Y = X**.

10) **sugmax** — позволяет определить максимально возможные диапазоны изменения выходных переменных в заданной системе нечеткого вывода, использующей алгоритм Sugeno.

5.5.13. Функции вызова диалоговых окон интерфейса. Данную группу образуют 12 функций:

1) **cmfdlg** — вызывает диалоговое окно для задания функции принадлежности пользователя,

2) **cmthdlg** — вызывает диалоговое окно для задания пользовательского алгоритма нечеткого вывода,

3) **fisgui** — позволяет генерировать элементы графического интерфейса пользователя,

4) **gfmfdlg** — вызывает диалоговое окно для задания числа и типа функций принадлежности при проектировании системы типа Sugeno (с последующим применением функции **anfis**),

5) **mfdlg** — вызывает диалоговое окно для задания дополнительной функции принадлежности,

6) **mfdrag** — обеспечивает «перетаскивание» функции принадлежности с использованием мыши,

7) **popundo** — возвращает из стека результаты отмены предыдущего действия,

8) **pushundo** — помещает в стек данные по отменяемому действию,

9) **savedlg** — вызывает диалоговое окно, выдающее запрос на сохранение результатов,

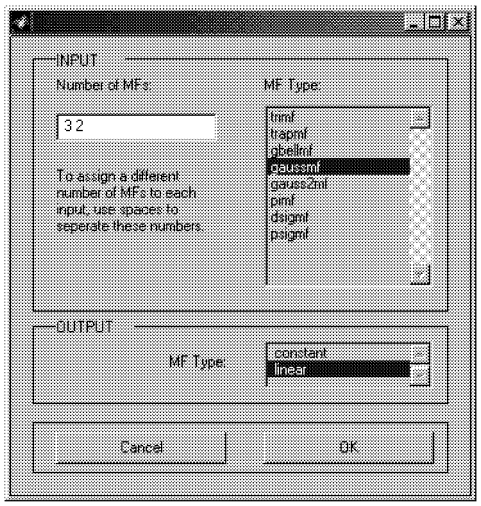

Рис. 5.43. Результат выполнения команды **gfmfdlg**

10) **statmsg** — помещает сообщения в поле состояния окна,

11) **updtfis** — обновляет элементы графического интерфейса пользователя,

12) **wsdlg** — вызывает диалоговое окно для выполнения операции сохранения в рабочем пространстве.

В качестве примера приведем выполнение функции **gfmfdlg**:

```
» a = readfis('tip');
» gfmfdlg('#init',a)
ans =
  '3 2'   'gaussmf'   'linear'
```

5.6. Работа Fuzzy Logic с блоками Simulink

Системы нечеткого вывода, созданные тем или иным образом с помощью пакета Fuzzy Logic Toolbox, допускают интеграцию с инструментами пакета Simulink, что позволяет выполнять моделирование систем в рамках последнего.

5.6.1. Пример: контроль уровня воды в баке. На рис. 5.44 изображен объект управления в виде бака с водой, к которому подходят две трубы: через одну трубу, снабженную краном, вода втекает в бак, через другую — вытекает.

Подачу воды в бак можно регулировать, больше или меньше открывая кран. Расход воды является неконтролируемым и зависит от диаметра выходной трубы (он фиксирован) и от текущего

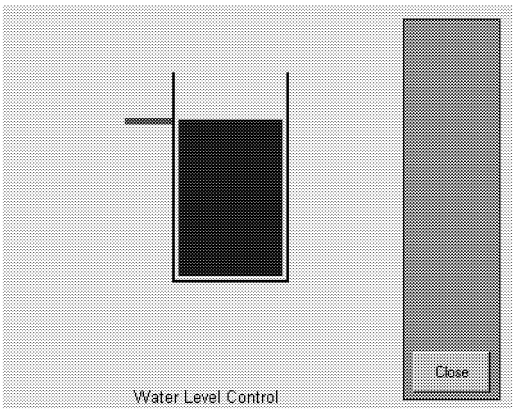

Рис. 5.44. Схематическое представление объекта управления (бака с водой)

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

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

Очевидно, в регулятор, обеспечивающий достижение цели управления, должна поступать информация о несоответствии (разности) требуемого и фактического уровней воды, при этом данный

регулятор должен вырабатывать управляющий сигнал на регулирующий элемент (кран).

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

1. If (level is okay) then (valve is no_change) (1)

2. If(level is low) then (valve is open_fast) (1)

3. If (level is high) then (valve is close_fast) (1)

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

5. If (level is okay and (rate is negative) then (valve is open_slow) (1),

что в переводе означает:

1. Если (уровень соответствует заданному), то (кран без изменения) (1)

2. Если (уровень низкий), то (кран быстро_открыть) (1)

3. Если (уровень высокий), то (кран быстро_закрыть) (1)

4. Если (уровень соответствует заданному) и (его прирост — положительный), то (кран надо медленно_закрывать) (1)

5. Если (уровень соответствует заданному) и (его прирост — отрицательный), то (кран надо медленно_открывать) (1)

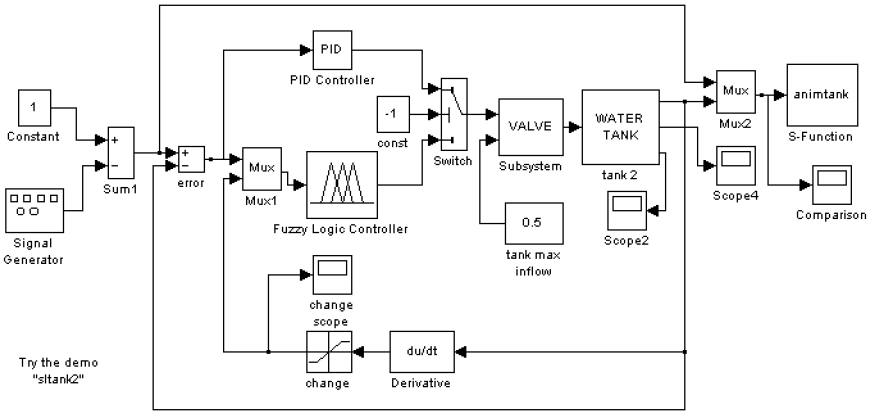

Рис. 5.45. Блок-диаграмма модели системы управления уровнем воды в баке с нечетким регулятором

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

Ее можно вызвать из командной строки командой **sltank**, что приведет к открытию окна Simulink с блок-диаграммой указанной модели (рис. 5.45).

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

Для изучения процесса функционирования системы необходимо нажать кнопку **Start** панели инструментов блок-диаграммы модели и дважды щелкнуть левой кнопкой мыши по блоку Comparison (Сравнение). В появившемся окне этого блока будут показаны изменяющиеся во времени сигналы заданного (последовательность импульсов прямоугольной формы) и фактического уровней воды (рис. 5.46). Как видно, переходный процесс в системе имеет апериодическую форму и заканчивается достаточно быстро, т. е. качество регулирования следует признать хорошим.

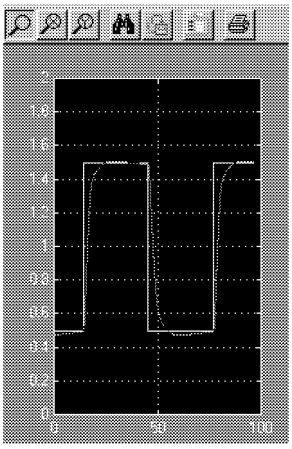

Рис. 5.46. Результаты моделирования системы управления с нечетким регулятором

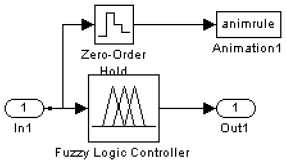

Рис. 5.47. Структура блока Fuzzy Logic Controller with Ruleviewer

Можно попытаться изменить характер функционирования системы, внося — например, с помощью рассмотренных программ с графическим интерфейсом (типа FIS-редактора и т.п.), определенные изменения в систему, задаваемую файлом tank.fis (изменяя правила, функции принадлежности, метод приведения к четкости и т. д.). Разумеется, все это лучше делать, остановив процесс моделирования.

Выполнив (в режиме командной строки) команду **sltankrule**, перейдем к блок диаграмме той же системы управления, но с нечетким регулятором с просмотрщиком правил (Fuzzy Logic Controller with Ruleviewer). Структура такого регулятора представляется в отдельном окне (рис. 5.47), если дважды щелкнуть левой кнопкой мыши по указанному блоку.

5.6.2. Построение нечеткой модели с использованием блоков Simulink. Для построения какой-то собственной моделирующей системы с использованием средств нечеткой логики и блоков Simulink рекомендуется просто скопировать блок Fuzzy Logic Controller из рассмотренной системы sltank (или какого-либо другого демонстрационного примера MATLAB) и поместить его в блок-диаграмму разрабатываемой системы. Из командной строки командой **fuzblock** можно также открыть библиотеку нечетких блоков пакета Simulink (в версии MATLAB 5.3, содержащей два блока: нечеткий регулятор — Fuzzy Logic Controller и нечеткий регулятор с просмотрщиком правил — Fuzzy Logic Controller with Ruleviewer, а в версии 5.2 — еще и несколько демонстрационных блоков), которые могут быть использованы точно так же, при необходимости — с дополнительным редактированием.

Отметим, что функционирование указанных блоков осуществляется с использованием системной S-функции **sffis.mex**. Запись этой функции такова:

output = sffis(t,x,u,flag,fismat),

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

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

5.6.3. Демонстрационные примеры работы с пакетом Fuzzy Logic Toolbox. Для ознакомления с пакетом Fuzzy Logic Toolbox можно использовать следующие функции (команды) в режиме командной строки:

defuzzdm — обзор методов приведения к четкости (дефазификации),

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

fuzdemos — демонстрация графического интерфейса пакета Fuzzy Logic,

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

juggler — демонстрация системы жонглирования мячом с использованием нечеткого регулятора,

invkine — демонстрация нечеткого управления движением робота-манипулятора,

irisfcm — демонстрация алгоритма кластеризации Fuzzy c-means,

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

slbb — демонстрация задачи «шар на качелях»,

slcp — демонстрация нечеткой системы управления перевернутым маятником,

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

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

sltbu — демонстрация функционирования нечеткой системы управления грузовиком.

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

СПИСОК ЛИТЕРАТУРЫ

1. Прикладные нечеткие системы / Под ред. Т. Тэрано, К.
М.: Мир, 1993. 368 с.
2. Змитрович А.И. Интеллектуальные информацио
Минск: НТООО "ТетраСистемс", 1997. 367 с.
3. Уоссермен Ф. Нейрокомпьютерная техника.—М.: Ми
4. Горбань А.Н., Россиев Д.А. Нейронные сети
компьютере.—Новосибирск: Наука, 1996. 275 с.
5. Круглов В.В., Борисов В.В. Искусственные нейро
и практика.—М.: Горячая линия—Телеком, 2001. 382 с.
6. Аверин А.Н., и др. Нечеткие множества в моделях у
ственного интелекта / Под ред. Д.А Поспелова.—М.: Нау
7. Кофман А., Алуха Х. Хил. Введение теории нечетки
вление предприятием.—Минск: Высшая школа, 1992. 2
8. Дли М.И. Локально-аппроксимационные модели сложн