

Министерство науки и высшего образования Российской Федерации

Амурский государственный университет

МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ
МОДЕЛИРОВАНИЕ: СПЕЦИАЛЬНЫЕ ГЛАВЫ

Практикум. Часть I.

Моделирование в условиях неопределенности.

Моделирование фрактальных структур.

Благовещенск
2019

ББК 22.193 я 73
М 34

*Рекомендовано
учебно-методическим советом университета*

Рецензент:

*И.Е. Еремин, профессор каф. информационных и управляющих систем
АмГУ, д-р техн. наук*

М 34 Математическое и компьютерное моделирование: специальные главы Практикум. Часть I. Моделирование в условиях неопределенности. Моделирование фрактальных структур. / сост. А.Г. Масловская, Л.И. Мороз – Благовещенск: Амурский гос. ун-т, 2019.

В практикуме рассматриваются примеры реализации математических моделей систем, функционирующих в условиях неопределенности (стохастическое моделирование методом Монте-Карло, нечеткое моделирование, моделирование на основе гибридных нейро-нечетких сетей). Отдельный раздел посвящен моделированию фрактальных объектов.

Излагаются краткие теоретические сведения о методах и средствах решения задач, освещаются практические аспекты реализации моделей с использованием инструментария пакета прикладных программ Matlab. Практикум содержит варианты индивидуальных заданий для лабораторных работ.

Учебное издание предназначено для студентов магистратуры, обучающихся по направлению подготовки 01.04.02 – «Прикладная математика и информатика», а также может быть использовано студентами других направлений подготовки и специальностей, применяющих методы и средства математического и компьютерного моделирования для решения прикладных задач.

ББК 22.193 я 73

ВВЕДЕНИЕ

В прикладной математической подготовке студентов, обучающихся по направлению «Прикладная математика и информатика», важное место занимает формирование устойчивых навыков и умений, позволяющих выполнять математическую формализацию изучаемого процесса или явления, осуществлять выбор методологии для построения вычислительной схемы решения прикладной задачи, реализовывать модель с использованием специализированного программного обеспечения, выполнять постановку и проведение вычислительного эксперимента, анализировать полученные результаты. Первая часть практикума «Математическое и компьютерное моделирование: специальные главы» посвящена методам и средствам реализации математических моделей систем, функционирующих в условиях неопределенности. Здесь детально рассмотрены подходы к реализации математических моделей стохастических процессов на основе метода Монте-Карло, нечеткое моделирование с использованием аппарата приложения Fuzzy Logic toolbox Matlab, моделирование с помощью гибридных сетей в системе ANFIS Matlab. Отдельный раздел посвящен реализации алгоритмов моделирования фрактальных структур.

Материал, содержащийся в данном издании, необходим для организации лабораторного практикума и самостоятельной работы студентов, изучающих методы и средства реализации моделей сложных систем. В практикуме приводятся базовые теоретические сведения по каждой из рассматриваемых тем, подробные алгоритмы и примеры решения практических задач с использованием инструментария ППП Matlab, контрольные вопросы для самопроверки, а также индивидуальные задания к лабораторным работам. Материал пособия ориентирован на студентов магистратуры, обучающихся по направлению подготовки 01.04.02 – «Прикладная математика и информатика», а также может быть использовано студентами других направлений подготовки и специальностей, применяющих методы и средства математического и компьютерного моделирования для решения прикладных задач.

1 МОДЕЛИРОВАНИЕ СТОХАСТИЧЕСКИХ ПРОЦЕССОВ МЕТОДОМ МОНТЕ-КАРЛО

Важным классом реализуемых на ЭВМ моделей, работающих в условиях неопределенности, являются стохастические модели, в которых реализация рассматриваемого процесса зависит от случайных параметров. Выходные значения для таких моделей при заданном наборе входных данных можно предсказать только в вероятностном смысле.

1.1 Базовые концепции метода Монте-Карло

Основу моделирования методом Монте-Карло составляет использование равномерно распределенных случайных чисел. Равномерно распределенные случайные числа заключены в интервале от 0 до 1 и выбираются случайным образом в соответствии с функцией распределения

$$F(x) = P\{X < x\} = x, \quad 0 < x < 1,$$

где $P\{X < x\}$ – вероятность того, что случайная величина X примет значение меньше x .

При равномерном распределении одинаково правдоподобно появление любых значений случайной величины в интервале $(0, 1)$. Основным методом получения случайных чисел является их генерация по модулю. Таким образом, мы имеем дело с «псевдослучайными числами». Пусть m, a, c, x_0 — целые числа, такие, что $m > x_0, a, c, x_0 > 0$. Псевдослучайное число x_i , из последовательности $\{x_i\}$ получается с помощью рекуррентного соотношения:

$$x_i = ax_{i-1} + c \pmod{m}.$$

Стохастические характеристики генерируемых чисел решающим образом зависят от выбора m, a, c . Неудачный выбор этих параметров приводит к ошибочным результатам, например, при моделировании методом Монте-Карло. Этим термином обозначается большая группа методов, в которых используются генераторы случайных чисел. Метод Монте-Карло используется для реше-

ния различных математических задач: интерполяции, вычисления интегралов, решения дифференциальных и интегральных уравнений, поиска экстремума.

Поскольку генерация случайных чисел осуществляется по формуле вида $\gamma_{k+1} = F(\gamma_k)$, а в коде программы можно записать лишь конечное число различных чисел, начиная с некоторого номера члены последовательности будут повторяться. Для численного моделирования часто требуется большое количество случайных чисел, поэтому период последовательности генерируемых случайных чисел, после которого последовательность начинает повторяться, должен быть достаточно большим. Он должен быть существенно больше требуемого для моделирования количества случайных чисел, иначе получаемые результаты будут искажены.

Программные оболочки, как правило, содержат генератор случайных чисел. Однако большинство статистических тестов показывает коррелированность между получаемыми случайными числами.

В Matlab для генерации псевдослучайных чисел, равномерно распределенных в интервале от 2^{-53} до $1-2^{-53}$ (примерно $(0,1)$), используют функцию `rand`. Теоретически функция `rand` позволяет без повторов сгенерировать 2^{1492} , `rand(n)` возвращает квадратную матрицу n -го порядка, элементами которой являются случайные числа. Для перехода от дробных чисел к числам из интервала $[a,b]$ достаточно воспользоваться преобразованием $a+(b-a)\cdot\text{rand}$. Функция `randn` позволяет сгенерировать случайные числа, распределенные по нормальному закону со следующими параметрами:

1) среднее значение случайной величины равно 0; $\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$;

2) дисперсия $\sigma^2 = 1$ или $\sigma^2 = \frac{1}{N} \left(\sum_{i=1}^N x_i^2 - \bar{x}^2 \right)$;

3) средняя квадратичная ошибка $\sigma = 1$.

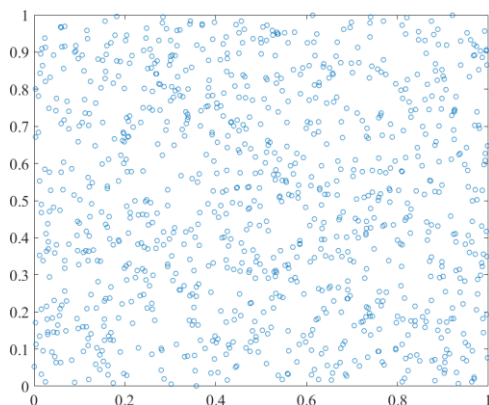
В математической статистике используют различные числовые характеристики случайных величин. Различают генеральные и выборочные характери-

стики. Наиболее важными характеристиками являются дисперсия и среднее значение.

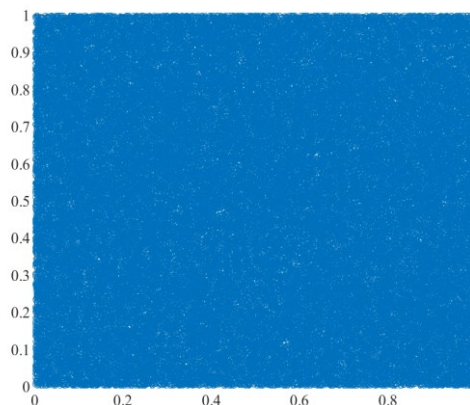
Существует быстрый тест, с помощью которого необходимо проверять каждый генератор. Качество генератора случайных чисел можно продемонстрировать, заполняя полностью d -мерную решетку (например, двух- и трехмерную, т. е. квадрат или куб). Хороший генератор должен заполнить все пространство такой ограниченной области (гиперкуба), рисунок 1.1.

Результаты подобного теста в Matlab:

```
>> n=100000;  
>> X=rand(1,n);  
>> Y=rand(1,n);  
>> plot(X,Y,'ko');
```



a (при $n=1000$)



б (при $n=100000$)

Рис. 1.1. Заполнение двумерной решетки генератором случайных чисел.

Следует отметить хорошее качество данного генератора. Для более объективной оценки качества подобных генераторов существует множество критериев.

Идея метода Монте-Карло чрезвычайно проста и состоит в следующем. Вместо того, чтобы описывать процесс в виде аналитического аппарата (набора уравнений), производится розыгрыш случайного явления с помощью специально организованной процедуры, включающей в себя случайность и дающей слу-

чайный результат. В результате статистического моделирования мы получаем в каждом случае новую, отличную от других реализация процесса. Одна реализация не несет никакой информации о всем процессе. Однако множество реализаций можно использовать как некий статический материал, который может быть обработан методами математической статистики.

Сущность метода. Требуется найти значение a некоторой изучаемой величины. Для этого выбирают такую случайную величину X , математическое ожидание которой равно a : $M(X) = a$. Проводят n испытаний, в результате которых получают n возможных значений X и вычисляют их среднее арифметическое: $\bar{X} = \sum X_i / n$ и принимают его в качестве оценки искомого числа a : $\bar{X} \approx a$. Применение метода Монте-Карло оправдано, когда организация процедуры розыгрыша проще, чем аналитический расчет.

Ошибка метода Монте-Карло пропорциональна $\sqrt{D/N}$, где D – некоторая постоянная, N – число случайных реализаций. Поэтому, чтобы уменьшить ошибку в 10 раз, необходимо увеличить число случайных реализаций в 100 раз. Поэтому чаще всего метод Монте-Карло применяется в тех случаях, когда требуется качественный результат задачи и требуемая точность не превосходит 5-10 %.

Отдельная реализация разыгрывается с помощью специальной процедуры – «бросание жребия». «Единичным жребием» называют любой опыт со случайным исходом, который отвечает на один из вопросов:

- Произошло или нет событие A ?
- Какое из событий A_1, A_2, \dots, A_n произошло?
- Какое значение приняла случайная величина X ?
- Какую совокупность значений приняла система случайных величин X_1, X_2, \dots, X_n ?

Любая реализация случайного явления методом Монте-Карло строится из цепочки единичных жребиев, сочетающихся с обычными расчетами. Ими учитывается влияние исхода жребия на дальнейший ход событий. Стандартным

механизмом организации единичного жребия является выброс случайного числа, все значения которого от 0 до 1 равновероятны (точнее, обладают одинаковой плотностью вероятности).

Произошло или нет событие A ? Чтобы ответить на этот вопрос, надо знать вероятность p наступления события A . Разыгрывается случайное число R , если оно меньше или равно p , что считают, что событие произошло, если больше – нет.

Какое из событий A_1, A_2, \dots, A_n произошло? Пусть данные события несовместны и образуют полную группу. Тогда сумма их вероятностей p_1, p_2, \dots, p_n равна 1. Разделим интервал $(0,1)$ на n участков длины p_1, p_2, \dots, p_n , на какой из участков попало R , то событие и произошло.

Какое значение приняла случайная величина X ?

Если случайная величина X дискретна, т.е. имеет значения X_1, X_2, \dots, X_n с вероятностями p_1, p_2, \dots, p_n , то очевидно, случай сводится к предыдущему.

Рассмотрим случай, когда случайная величина непрерывна и имеет заданную плотность вероятности $f(X)$.

Чтобы разыграть ее значение, необходимо перейти от плотности вероятности к функции распределения $F(X)$ по формуле:

$$F(X) = \int_{-\infty}^X f(x) dx,$$

затем найти от функции F обратную ей функцию ψ и разыграть случайное число R от 0 до 1 и взять от него эту обратную функцию: $X = \psi(R)$.

Графически: разыгрывается случайное число R и ищется для него такое значение X , при котором $F(X) = R$.

Какую совокупность значений приняла система случайных величин X_1, X_2, \dots, X_n ? Если случайные величины независимы, то достаточно n раз повторить указанную процедуру. Если они зависимы, то разыгрывать каждую последующую нужно на основе ее условного закона распределения, при условии, что все предыдущие приняли те значения, которые дал розыгрыш.

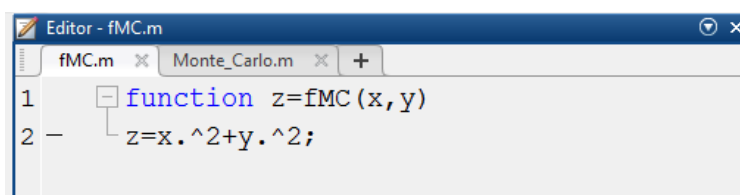
1.2 Примеры решения модельных задач на основе метода Монте-Карло: реализация в ППП Matlab

Пример 1. В качестве примера применения метода Монте-Карло рассмотрим вычисление интегралов (наиболее эффективно применение данного метода для вычисления многомерных интегралов, когда прямое интегрирование обычными численными методами невозможно). Основная идея метода заключается в том, чтобы вычислять подынтегральную функцию не в каждом узле интегрирования, поскольку число таких узлов может быть очень велико, а только в точках, являющихся репрезентативной случайной выборкой. Согласно теореме математического анализа, интеграл определяется средним значением подынтегральной функции на отрезке интегрирования. Таким образом, оценка интеграла методом «выборочного среднего»:

$$I = (b-a)\langle f \rangle = (b-a) \frac{1}{N} \sum_{i=1}^N f(x_i),$$

где x_i – случайные числа, равномерно распределенные на отрезке $[a,b]$; N – количество испытаний.

Предположим, что нам нужно вычислить интеграл $\int_0^1 \int_1^2 (x^2 + y^2) dx dy$ в области $\Omega = \{(x, y) : 0 \leq x \leq 1, 1 \leq y \leq 2\}$ и сравним результат с точным значением, равным $8/3$. Реализация в Matlab представлена на рисунках 1.2 и 1.3.



```
Editor - fMC.m
fMC.m  Monte_Carlo.m  +
1  function z=fMC(x, y)
2  z=x.^2+y.^2;
```

Рис. 1.2. Задание подынтегральной функции.

Результат работы программы:

```
>> Monte_Carlo
    2.6712
    0.0046
```

```

Editor - Monte_Carlo.m
fMC.m Monte_Carlo.m
1 - n=10000;
2 - a=0;b=1;c=1;d=2;
3 - X=a+(b-a)*rand(1,n);
4 - Y=c+(d-c)*rand(1,n);
5 - S=0;
6 - J=0;
7 - for i=1:n
8 -     S=S+fMC(X(i),Y(i));
9 - end
10 - J=S*(b-a)*(d-c)/n ; %значение интеграла
11 - disp(J)
12 - err=J-8/3; % ошибка вычисления
13 - disp(err)

```

Рис. 1.3. Вычисление двойного интеграла методом Монте-Карло.

Пример 2. Интересным случайным процессом является броуновское движение. В 1827 г. английский ботаник Броун наблюдал под микроскопом хаотическое движение мелких частиц в капле жидкости. Математическая теория этого процесса была построена гораздо позже А. Эйнштейном, а затем Н. Винером. Если броуновская частица является достаточно мелкой, то количество столкновений молекул воды с данной частицей не будет одним и тем же по всей площади поверхности частицы, что и приводит к ее хаотичному движению.

Рассмотрим двумерное броуновское движение. Пусть $\xi_1(t)$ и $\xi_2(t)$ – координаты частицы в момент времени t . Очевидно, что $\xi_1(t)$ и $\xi_2(t)$ являются случайными величинами. Можно допустить, что эти величины распределены по нормальному закону с нулевым математическим ожиданием и дисперсией $\sigma^2(t) = \sigma^2 t$, где константа σ^2 называется коэффициентом диффузии.

Разность координат частицы при каждом последующем шаге задается функцией нормального распределения $\text{rnorm}(n, a, \sigma)$, где n – число координат случайного вектора (в нашем случае – 2, поскольку мы рассматриваем двумерное движение), а a, σ – среднее и дисперсия нормального распределения. Реализация задачи в Matlab, показана на рисунке 1.4. Результат моделирование двумерного броуновского движения в пакете Matlab показан на рисунке 1.5.

```
Editor - Broun.m
fMC.m x Broun.m x +
1 - n=10000;
2 - X=rand(1,n);
3 - Y=rand(1,n);
4 - X(1)=0;
5 - Y(1)=0;
6 - for i=2:n
7 -     X(i)=X(i-1)+randn;
8 -     Y(i)=Y(i-1)+randn;
9 - end
10 - plot(X,Y)
11 - grid on
```

Рис. 1.4. Моделирование броуновского движения.

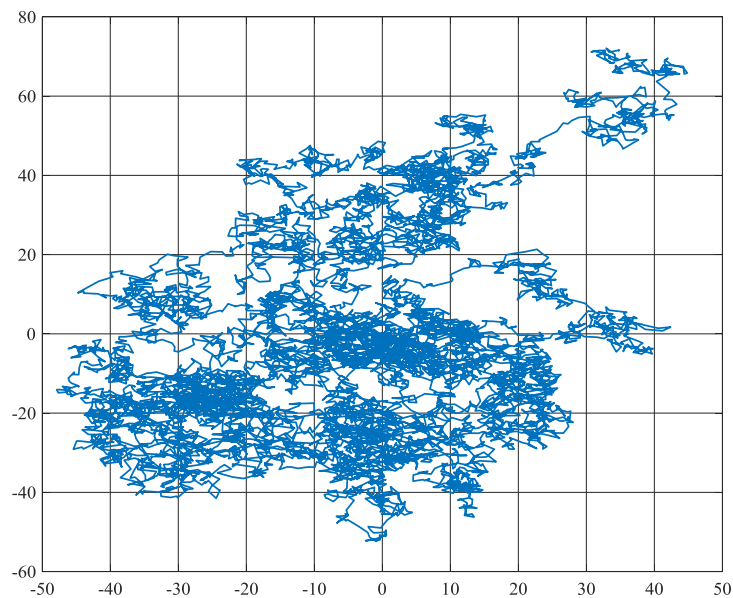


Рис. 1.5. Визуализация броуновского движения.

Контрольные вопросы

1. Какие модели называются стохастическими?
2. Дайте определение равномерному распределению случайной величины.
3. Опишите суть метода Монте-Карло.
4. Как вычислить дисперсию, среднеквадратичное отклонение случайных величин.
5. Что такое «выборочное среднее»?

1.3 Индивидуальные задания

Задание 1. Реализовать трехмерную модели броуновского движения, используя возможности эффекта анимации в Matlab.

Задание 2. Выполнить имитационное трехмерное моделирование случайного блуждания по закону, предусмотренному индивидуальным заданием.

Вариант 1. Частица совершает стохастическое движение из центра куба $-10 \leq x \leq 10$, $-10 \leq y \leq 10$, $-10 \leq z \leq 10$. При пересечении частицей плоскости $x=5$ генерируется еще 2 частицы, движущиеся по аналогичному закону. При достижении границы куба частица «отражается» от этой границы.

Вариант 2. N ($N \leq 100$) частиц генерируются одновременно и начинают случайное движение из центра куба: $-5 \leq x \leq 5$, $-5 \leq y \leq 5$, $-5 \leq z \leq 5$. При достижении любой грани куба частица «исчезает». Провести подсчет количества частиц с течением времени и вывести соответствующий график на экран.

Вариант 3. N ($N \leq 100$) частиц генерируются одновременно и начинают движение из центра куба: $-5 \leq x \leq 5$, $-5 \leq y \leq 5$, $-5 \leq z \leq 5$. При достижении любой грани куба частица «отражается» и продолжает случайное движение внутри куба. Провести подсчет количества частиц с течением времени и вывести соответствующий график на экран.

Вариант 4. N ($N \leq 100$) частиц генерируются одновременно и начинают движение из центра куба: $-5 \leq x \leq 5$, $-5 \leq y \leq 5$, $-5 \leq z \leq 5$. При случайном столкновении двух частиц (столкновением считать попадание в область, ограниченную малой окрестностью) – обе уничтожаются. Провести подсчет оставшихся частиц с течением времени.

Вариант 5. Частица совершает стохастическое движение из центра куба $-10 \leq x \leq 10$, $-10 \leq y \leq 10$, $-10 \leq z \leq 10$. При пересечении частицей граней куба генерируется еще две частицы, движущиеся по аналогичному закону. Провести подсчет количества частиц с течением времени и вывести соответствующий график на экран.

Вариант 6. $N=8$ частиц генерируются одновременно и начинают движение из вершин куба: $-5 \leq x \leq 5$, $-5 \leq y \leq 5$, $-5 \leq z \leq 5$. При случайном столкновении двух частиц (столкновением считать попадание в область, ограниченную малой окрестностью) – рождается новая частица, имеющая такие же координаты. Провести подсчет количества частиц с течением времени и вывести соответствующий график на экран.

Вариант 7. $N=1000$ частиц генерируются одновременно и начинают случайное движение из центра куба: $-5 \leq x \leq 5$, $-5 \leq y \leq 5$, $-5 \leq z \leq 5$. При достижении частицей грани $z = 5$, она «вылетает» из куба (удаляется), ее траектория не прорисовывается. Провести подсчет количества частиц с течением времени и вывести соответствующий график на экран.

Вариант 8. N ($N \leq 100$) частиц генерируются одновременно и начинают движение из вершин куба: $-5 \leq x \leq 5$, $-5 \leq y \leq 5$, $-5 \leq z \leq 5$. При достижении частицей центра куба (центром считать малую окрестность точки $(0,0,0)$), частица уничтожается. Провести подсчет «исчезнувших» частиц с течением времени и вывести соответствующий график на экран.

Вариант 9. Частица совершает стохастическое движение из центра куба $-10 \leq x \leq 10$, $-10 \leq y \leq 10$, $-10 \leq z \leq 10$. При пересечении частицей граней куба генерируется еще 2 частицы, движущиеся по аналогичному закону (при пересечении границы куба, частицу нужно «вернуть обратно»). Провести подсчет количества частиц с течением времени и вывести соответствующий график на экран.

Вариант 10. $N=100$ частиц генерируется одновременно и начинают движение из точки $x = 0$, $y = 0$, $z = 5$ куба: $-5 \leq x \leq 5$, $-5 \leq y \leq 5$, $-5 \leq z \leq 5$. Организовать случайное блуждание так, чтобы при попадании в нижнюю половину куба ($-5 \leq z \leq 0$) частица прекращала свое движение и ее положение было зафиксировано на плоскости $z = 0$.

2 НЕЧЕТКИЕ МОДЕЛИ

2.1 Основные концепции теории нечеткого моделирования

Фундаментальные основы нечеткой логики были заложены в трудах американского ученого Л. Заде в середине XX века, который предложил математически формализовать неточные, «размытые», качественные понятия из различных предметных областей и при этом, делать вполне определенные выводы из имеющейся информации. Так, в след за нечеткими множества (fuzzy – нечеткий, размытый), появилась нечеткая логика и нечеткое моделирование – новые научные направления, предоставляющие гибкие и эффективные инструменты исследования в самых различных приложениях.

Нечеткое моделирование востребовано при построении математического описания и анализе физических систем и бизнес-процессов, которые характеризуются неопределенностью и для которых построение детерминированных моделей и стохастических моделей не представляется возможным. Применение нечеткого моделирование расширяет свои границы, проникая в области, которые ранее считались исключительно гуманитарными. Процессы принятия решения и управления, сопровождающие социальные, экологические, экономические и физические системы, в последние годы все чаще строятся на использование математического аппарата нечеткого моделирования. Развитие нечеткого моделирования в некоторой степени стимулирует и развитие программно-аппаратных средств, используемых для реализации такого рода моделей.

Концепции теории нечеткого моделирования предоставляют возможность формализации нечетких данных, манипулировать этими данными, формулировать нечеткие выводы. Многие авторы отмечают, что именно нечетка логика в физическом аспекте лучше «моделирует» специфику человеческих умозаключений по сравнению с классическими системами, основанными на применении булевой алгебры и логике предикатов. Поэтому использование данного подхода позволяет строить математические модели, адекватно описывающие объекты и процессы реального мира.

Введем понятие нечеткого множества A , которое задается как некоторое множество упорядоченных кортежей:

$$\langle x, \mu_A(x) \rangle,$$

где x – это некоторый элемент универсума E ;

$\mu_A(x)$ – функция принадлежности, определяющая соответствие между некоторым элементов $x \in E$ и числом $\mu_A(x) \in \mathbb{R}$ и $\mu_A(x) \in [0,1]$:
 $\mu_A(x): E \rightarrow [0,1]$.

Таким образом, нечеткое множество – есть множество элементов, для которых нельзя сформулировать однозначный ответ на вопрос: «Принадлежит ли конкретный элемент заданному нечеткому множеству?».

Выражение $\mu_A(x) = 1$ для $x \in E$ говорит о том, что элемент x определенно принадлежит A , тогда как выражение $\mu_A(x) = 0$ означает, что элемент x определенно не принадлежит A .

Инициализировать функцию принадлежности можно явно, используя функциональный вид $(\mu_A(x) = \exp\left(-\left(\frac{x-1}{2}\right)^2\right))$, или с помощью дискретного представления – на основе перечисления набора значений $x \in \{x_i\}$ в фиорме:

$$A = \{\mu_A(x_1) / x_1 + \mu_A(x_2) / x_2 + \dots + \mu_A(x_n) / x_n\} \text{ или}$$

$$A = \left\{ \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \dots + \frac{\mu_A(x_n)}{x_n} \right\}.$$

Введем некоторые нечеткие аналоги частных нечетких множеств:

- пустое нечеткое множество \emptyset задается как множество, функция принадлежности для которого тождественно равна нулю $\mu_\emptyset = 0$ для всех элементов;

- универсальное множество E или U задается так, что все другие множества являются его подмножествами; принято, что функция принадлежности универсума $\mu_E = 1$ для всех элементов.

Для нечеткого подхода также требуется определить понятие нечетких чисел, которые вводятся как нечеткие множества на множестве \mathbb{R} всех действительных чисел x с функцией принадлежности $\mu_A(x) \in [0,1]$. Основными же в теории нечетких множеств являются лингвистические переменные, которые вводятся как словосочетания или слова некоторого формального или естественного языка и принимают определенные значения из так называемого терм-множества.

Для нечетких множеств характерно наличие определенного носителя $Supp(A)$, который есть подмножество элементов A , для которых $Supp(A) = \{x, \mu_A(x) > 0\}$.

Работая с нечеткими множествами на практике, исследователю потребуется задать для удобства функцию принадлежности в виде некоторой функциональной зависимости. Важность выделения классов функций принадлежности вызвана также возможностью их унифицированного представления в пакетах прикладных программ и программных средах (например, такая возможность реализована в ППП Matlab). Особое распространение получили следующие виды функций принадлежности (рисунок 2.1, в скобках указаны команды Matlab, позволяющие реализовать данные функции): треугольная (*trimf*), трапецеидальная (*trapmf*), гауссова (*gaussmf*), обобщенная колоколообразная (*gbellmf*), двойная гауссова (*gauss2mf*), сигмоидальная (*sigmf*), двойная сигмоидальная (*dsigmf*), произведение двух сигмоидальных функций (*psigmf*). Укажем математические представления некоторых из этих функций:

$$trimf(x, a, b, c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) - \text{треугольная (рисунок 2.1 а),}$$

$$trapmf(x, a, b, c, d) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{d-x}{c-b}\right), 0\right) - \text{трапецеидальная (рисунок 2.1}$$

$$\text{б), } gaussmf(x, \sigma, c) = \exp\left(-\left(\frac{x-c}{\sigma}\right)^2\right) - \text{гауссова (рисунок 2.1 в).}$$

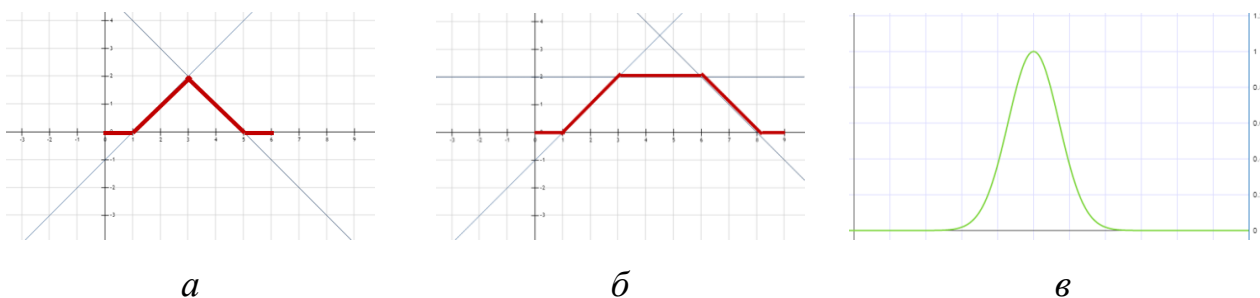


Рисунок 2.1 – Примеры графического представления функций принадлежности для нечетких множеств

Проиллюстрируем механизм вывода на примере вычисления значения некоторой функции $y = f(x)$, как показано на рисунке 2.2. Схожий процесс сопровождает и проведение операций с нечеткими числами. Применяемый в экспертных оценках и системах принятия решений аппарат нечетких выводов подразумевает создание базы знаний, которая формируется на основе оценок экспертов в конкретных предметных областях и представляется системой нечетких предикатных формул.

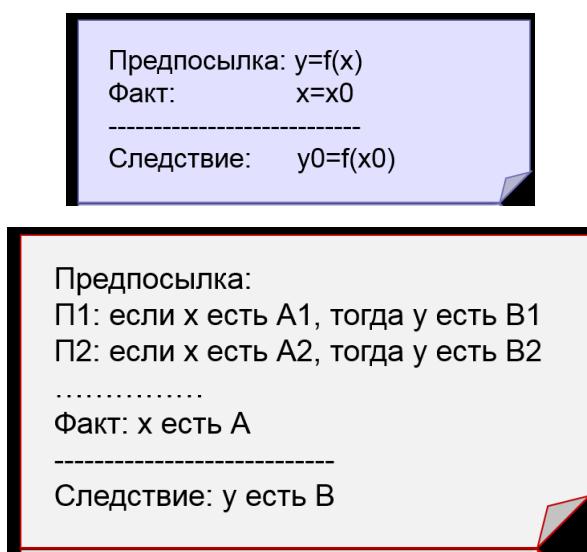


Рисунок 2.2 – Пример задания нечетких правил

На рисунке 2.3 представлена структура алгоритма нечеткого вывода. Кратко поясним каждый этап.

Этап 1. Введение нечеткости – фаззификация. Функции принадлежности, определенные на входных переменных, применяются к их фактическим

значениям для определения степени истинности каждой предпосылки каждого правила.



Рисунок 2.3 – Этапы нечеткого вывода

Этап 2. Логический вывод

Вычисленное значение истинности для предпосылок каждого правила применяется к заключениям каждого правила. Это приводит к одному нечеткому подмножеству, которое будет назначено каждой переменной вывода для каждого правила.

Правила логического вывода: операция \min (МИНИМУМ) и prod (УМНОЖЕНИЕ).

Этап 3. Агрегирование

Все нечеткие подмножества, назначенные к каждой переменной вывода (во всех правилах), объединяются вместе, чтобы сформировать одно нечеткое подмножество для каждой переменной вывода. При подобном объединении обычно используют операции \max (МАКСИМУМ) или sum (СУММА).

Этап 4. Дефаззификация – приведение к четкости

Используются специальные методы преобразования нечеткого набора выводов в четкое число.

2.2 Нечеткое моделирование с использованием инструментальных средств пакета Fuzzy Logic Toolbox Matlab: основные приемы работы

Пакет нечеткой логики Fuzzy Logic Toolbox – это пакет прикладных программ, относящихся к теории размытых или нечетких множеств и позволяю-

ших конструировать так называемые нечеткие экспертные и/или управляющие системы. Можно отметить основные возможности этого пакета:

- 1) построение систем нечеткого вывода (экспертных систем, регуляторов, аппроксиматоров зависимостей);
- 2) построение адаптивных нечетких систем (гибридных нейронных сетей);
- 3) интерактивное динамическое моделирование систем с нечеткой логикой в среде пакета блочного моделирования Simulink.

Пакет обеспечивает работу:

- 1) в режиме командной строки;
- 2) в окнах графического интерфейса;
- 3) в среде пакета расширения Simulink.

Пакет Fuzzy Logic Toolbox имеет обширную справку и документацию в двух файлах формата PDF (на английском языке), которые могут использоваться для получения исчерпывающей информации о составе, возможностях и функциях пакета.

Графический интерфейс пакета Fuzzy Logic Toolbox

Большинство приложений системы MATLAB имеет наиболее наглядную реализацию при использовании средств графического интерфейса пользователя GUI. В состав программных средств Fuzzy Logic Toolbox входят следующие основные программы, позволяющие работать в режиме GUI:

1. 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);
2. редактор гибридных систем (ANFIS Editor, ANFIS-редактор);
3. программа нахождения центров кластеров (программа Clustering – кластеризация).

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

Построение нечеткой аппроксимирующей системы. Командой (функцией) `fuzzy` из режима командной строки запускается основная интерфейсная программа пакета `Fuzzy Logic` – редактор нечеткой системы вывода. Вид открывающегося при этом окна приведен на рисунке 2.4. Строка меню редактора содержит следующие 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.

Приведем пример нечеткой системы, отображающей зависимость между переменными x и y , заданную с помощью таблицы (представленные в таблице данные отражают зависимость $y = x^2$):

x	-1	-0.6	0	0.4	1
y	1	0.25	0	0.18	1

1. Для создания модели необходимо выполнить следующие действия. В меню `File` выбираем команду `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 на блоке, озаглавленном `input1` (вход1). Затем в правой части редактора в поле `Name` (имя) вместо `input1` введем обозначение нашего аргумента, то есть x . Обратим внимание, что если теперь сделать где-нибудь (вне блоков редактора) однократный щелчок, то имя отмеченного блока изменится на x ; то же достигается нажатием клавиши `Enter` после ввода.

3. Дважды щелкнем на этом блоке. Перед нами откроется окно редактора функций принадлежности – `Membership Function Editor`, как показано на рисунке 2.5.

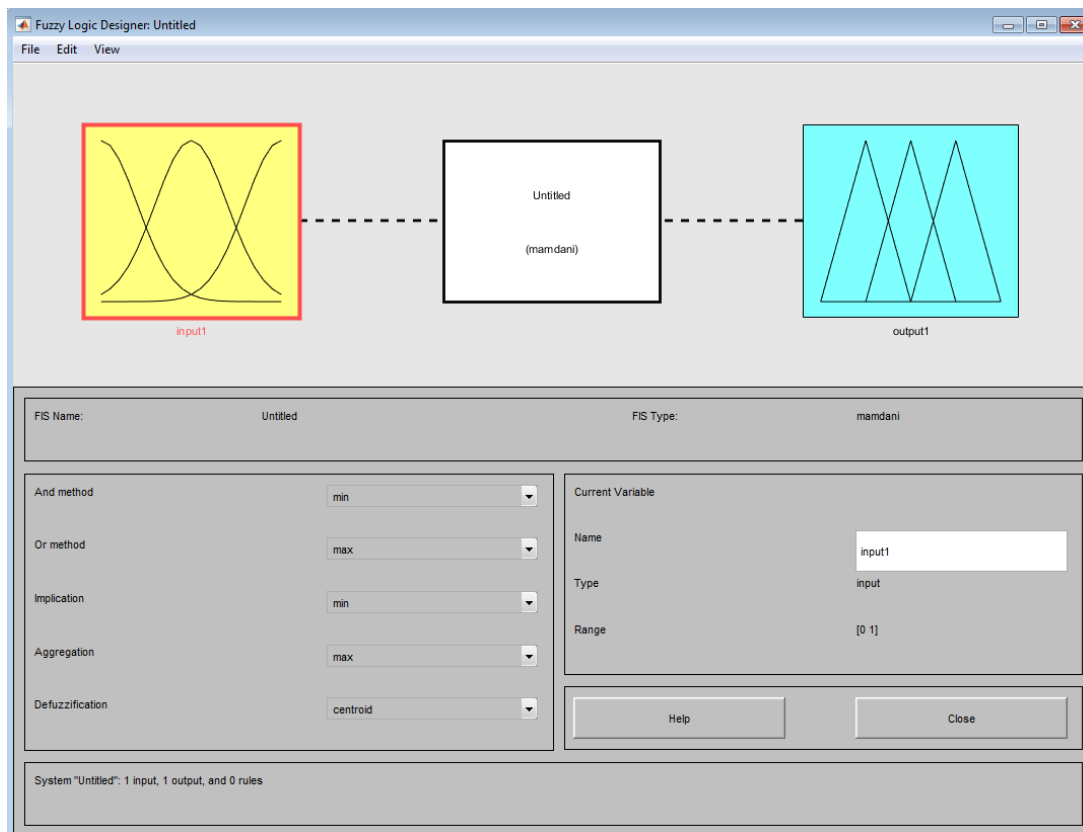


Рис. 2.4. Вид окна Fuzzy Logic Designer.

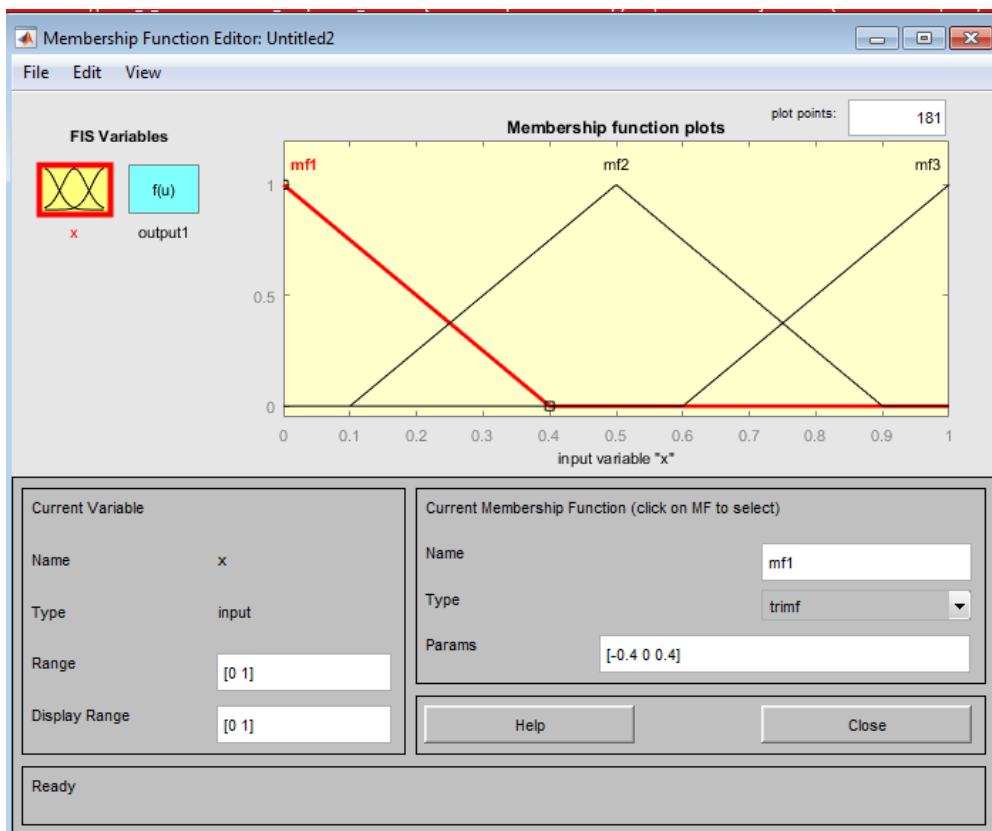


Рис. 2.5. Окно редактора функций принадлежности.

Обратимся к меню `Edit` данного редактора и выберем в нем сначала команду `Remove All MFs` (удалить все функции принадлежности) – для удаления установок по умолчанию, а затем команду – `Add MFs` (добавить функции принадлежности). При этом появится диалоговое окно, как представлено на рисунке 2.6, позволяющее задать тип (`MF type`) и количество (`Number of MFs`) функций принадлежности (в данном случае все относится к входному сигналу, то есть к переменной x). Выберем гауссовы функции принадлежности (`gaussmf`), а их количество зададим равным пяти – по числу значений аргумента функции $y(x)$. Подтвердим ввод информации нажатием кнопки `OK`, после чего произойдет возврат к окну редактора функций принадлежности.

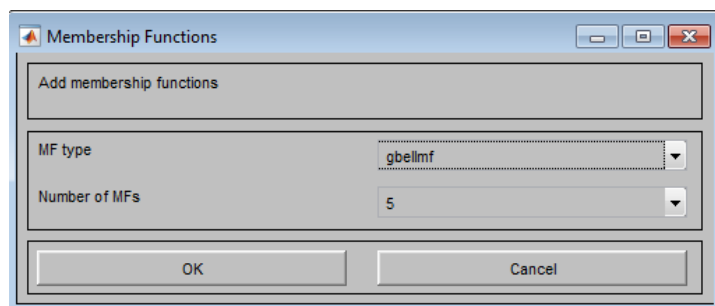


Рис. 2.6. Диалоговое окно задания типа и количества функций принадлежности.

4. В поле `Range` (диапазон) установим диапазон изменения x от -1 до 1 , то есть диапазон, соответствующий таблице. Щелкнем затем левой кнопкой мыши где-нибудь в поле редактора (или нажмем клавишу `Enter`). Обратим внимание, что после этого произойдет соответствующее изменение диапазона в поле `Display Range` (диапазон отображения).

5. Обратимся к графикам заданных нами функций принадлежности, изображенным в верхней части окна редактора функций принадлежности. Заметим, что для успешного решения поставленной задачи необходимо, чтобы ординаты максимумов этих функций совпадали с заданными значениями аргумента x . Для левой, центральной и правой функций такое условие выполнено, но две других необходимо «подвинуть» вдоль оси абсцисс. Это делается весьма просто: подводим указатель к нужной кривой и щелкаем левой кнопкой мыши.

Кривая выбирается, окрашиваясь в красный цвет, после чего с помощью мыши ее 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). Помимо этого, в поле Name можно изменить имя для выбранной кривой (завершая ввод каждого имени нажатием клавиши Enter). Проведем требуемые перемещения кривых и зададим всем пяти кривым новые имена, например, так: самой левой – left_b; следующей – left; центральной – central; следующей за ней справа – right; самой правой – right_b. Результат показан на рисунке 2.7.

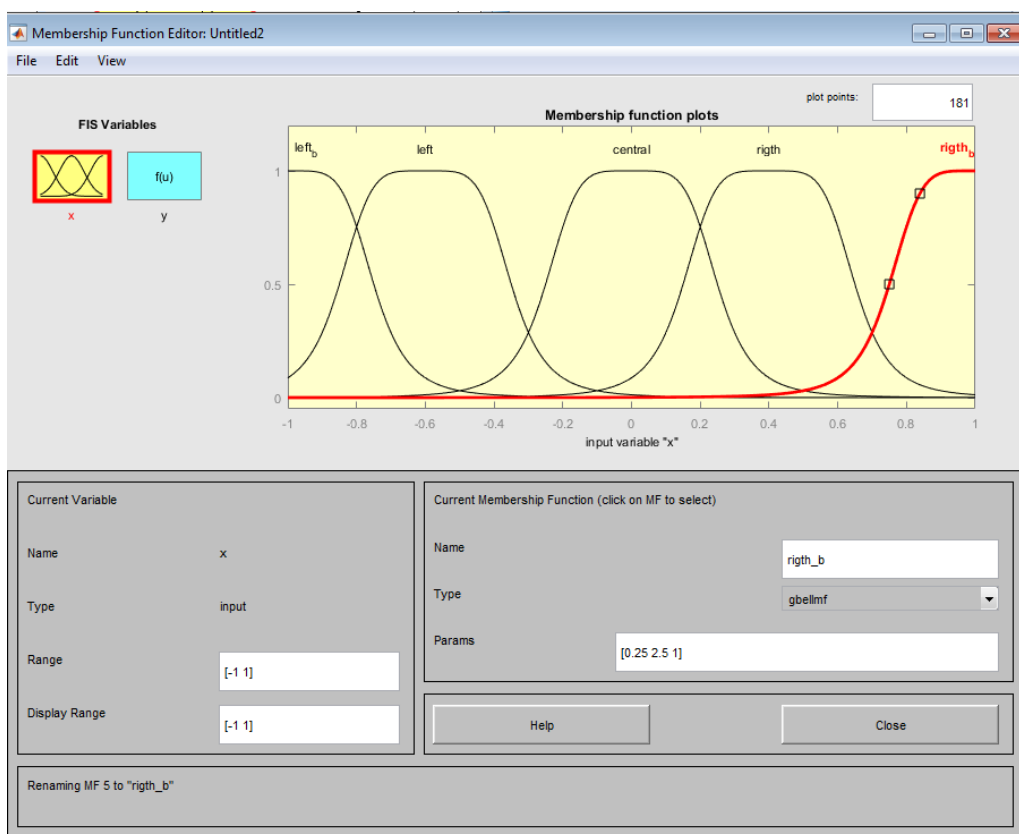


Рис. 2.7. Окно редактора: задание функций принадлежности для примера.

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

6. Кликнем на блоке голубого цвета `output1` (выход1). В поле `Name` заменим имя `output` на `y`, как в пункте 2.

7. Дважды кликнем на выделенном блоке и перейдем к редактору функций принадлежности. В окне `Membership Function Editor` вначале укажем на блок $f(u)$ с именем `y`. Далее в меню `Edit` последовательно выберем команды `Remove All MFs`, `Add MFs`. Появляющееся затем диалоговое окно вида рисунку 2.8 позволяет задать теперь в качестве функций принадлежности только линейные (`linear`) или постоянные (`constant`) – в зависимости от того, какой алгоритм Сугено (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`). Подтвердим введенные данные нажатием кнопки `OK`, после чего произойдет возврат в окно редактора функций принадлежности.

Диапазон изменения (`Range`), устанавливаемый по умолчанию, – $[0,1]$ – менять в данном случае не нужно. Изменим лишь имена функций принадлежности (их графики при использовании алгоритма Сугено для выходных переменных не приводятся), например, задав их как соответствующие числовые значения `y`, то есть 0, 0.18, 0.25, 1; одновременно эти же числовые значения введем в поле `Params`, изображенное на рисунке 2.8. Затем закроем окно нажатием кнопки `Close` и вернемся в окно редактора нечеткой системы (`Fuzzy Logic Designer`).

Дважды щелкнем на среднем (белом) блоке, при этом раскроется окно еще одной программы – редактора правил (`Rule Editor`). Введем соответствующие правила. При вводе каждого правила необходимо обозначить соответствие между каждой функцией принадлежности аргумента x и числовым значением `y`. Кривая, обозначенная нами `left_b`, соответствует $x = -1$, то есть `y = 1`. Выберем поэтому в левом поле (с заголовком `x is`) вариант `left_b`, а в правом 1 и нажмем кнопку `Add rule` (добавить правило). Введенное правило появится в окне правил и будет представлять собой такую запись:

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

Аналогично поступим для всех других значений x , в результате чего сформируется набор из пяти правил, рисунок 2.9.

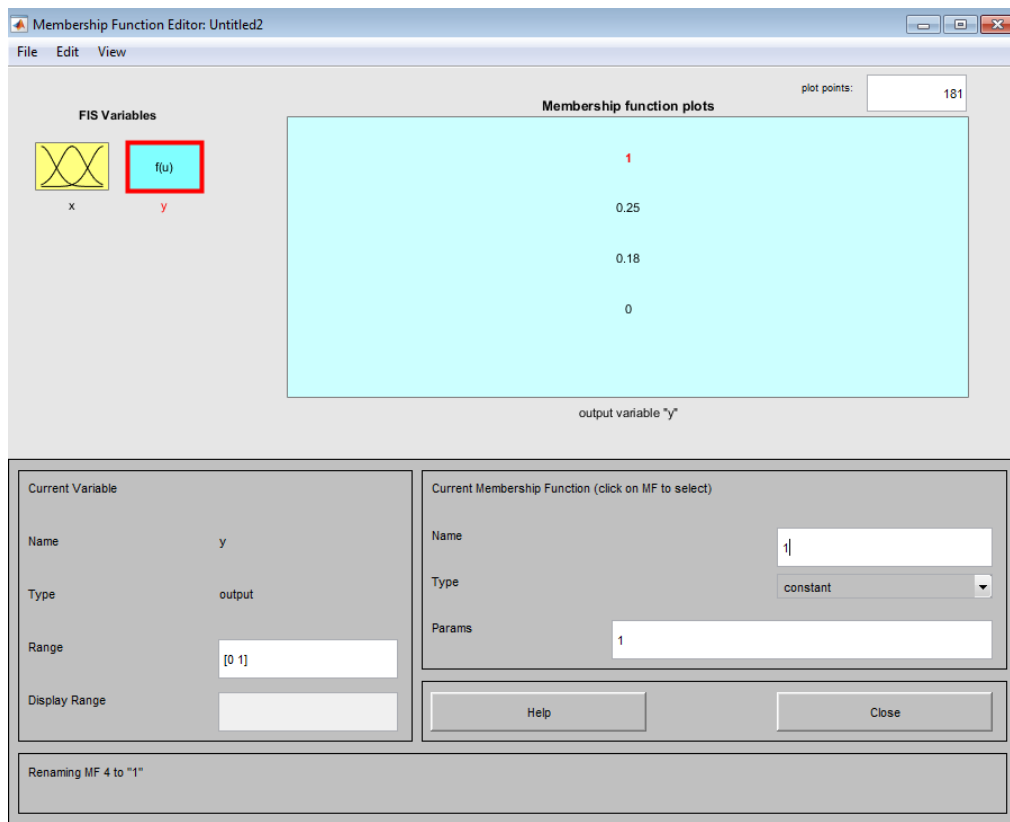


Рис. 2.8. Параметры функций принадлежности переменной y .

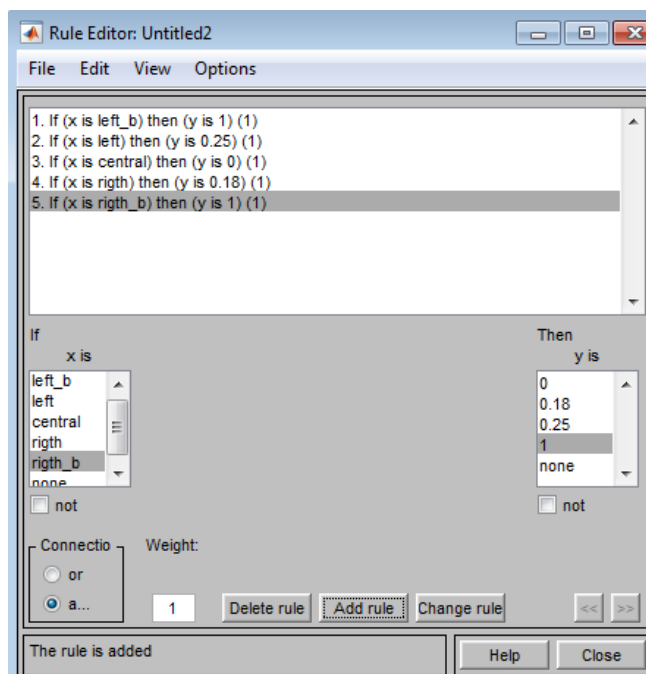


Рис. 2.9. Окно редактора правил.

10. Закроем окно редактора правил и вернемся в окно редактора нечеткой системы (Fuzzy Logic Designer). Построение нечеткой системы можно считать законченным, и можно начать эксперименты по анализу. Для большинства опций были сохранены значения по умолчанию. Предварительно сохраним на диске (используя команды меню File/Export/To file) созданную систему под каким-либо именем, например Fis1.

11. Обратимся к меню View. Используя пункт меню Rules, просмотрим окно просмотра правил – Rule Viewer, представленное на рисунке 2.10.

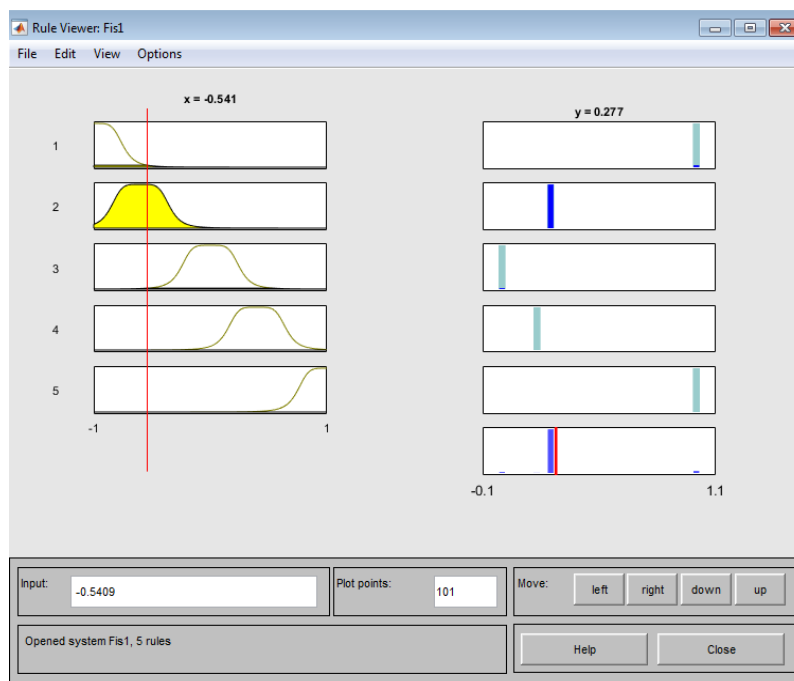


Рис. 2.10. Окно интерфейс для просмотра результата нечеткого вывода.

12. Слева в форме показаны функции принадлежности для аргумента x , в справа – для функции принадлежности переменной выхода y с пояснением механизма принятия решения. Красная вертикальная черта, пересекающая графики в левой части окна, которую можно перемещать с помощью мыши, 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, $x = 0.45$ в поле Input и нажмем затем клавишу Enter. Значение y сразу изменится и станет равным 0.186. Таким образом, с помощью построенной модели и окна просмотра пра-

вил можно решать задачу интерполяции, то есть задачу, решение которой и требовалось найти. Изменение аргумента путем перемещения красной вертикальной линии очень наглядно демонстрирует, как система определяет значения выхода.

13. Закроем окно просмотра правил и выбором команды меню View/Surface перейдем к окну просмотра поверхности отклика (выхода), в нашем случае – к просмотру кривой $y(x)$, рисунок 2.11. Видно, что построенное системой по таблице данных отображение лишь отдаленно соответствует функции x^2 – число экспериментальных точек невелико, да и параметры функций принадлежности (для x) выбраны, скорее всего, неоптимальным образом.

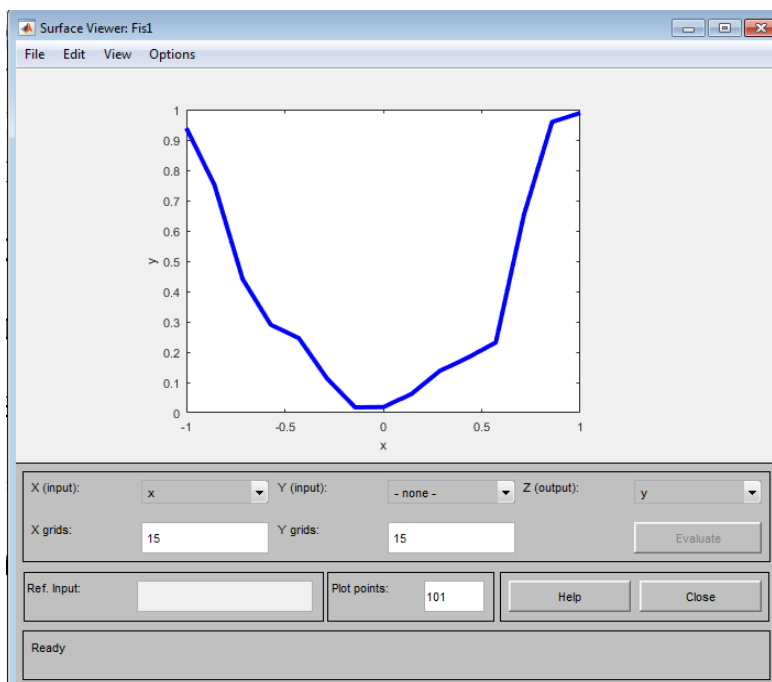


Рис. 2.11. Окно просмотра поверхности отклика.

14. 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 редакторе по умолчанию при использовании алгоритма Сугено, можно отметить следующие: логический вывод организуется с помощью операции умножения ($prod$); композиция организуется с помощью операции логической суммы (вероятностного ИЛИ, $probor$);

приведение к четкости организуется дискретным вариантом центроидного метода (взвешенным средним, *wtaver*). Используя соответствующие поля в левой нижней части окна редактора, данные опции можно при желании изменить.

2.3 Примеры создания нечетких экспертных систем в Fuzzy Logic Toolbox Matlab

Пример 1. Построение экспертной системы: сколько дать «на чай»?

Представим пример реализации методики построения нечеткой экспертной системы, которая отвечает за принятие решения об объеме поощрения официанта за обслуживание в ресторане. Принимая во внимание какие-либо субъективные суждения, устоявшиеся традиции или интуицию исследователя, допустим, что данная задача может быть формализована следующей совокупностью правил:

- если обслуживание плохое или блюда не устраивают клиента, то чаевые – маленькие;
- если обслуживание хорошее, то чаевые – средние;
- если обслуживание отличное или еда – превосходная, то чаевые – большие.

Качество обслуживания и еды пусть оцениваются по 10-бальной системе (0 – наихудшая оценка, 10 – наилучшая).

Будем предполагать далее, что маленькие чаевые составляют около 0 % от стоимости обеда, средние – около 10 % и большие – примерно 20 %. Представленной информации достаточно для проектирования простейшей нечеткой экспертной системы. Чтобы провести формализацию системы, определим:

- два входа (которые условно можно назвать «Service» («сервис») и «Food» («еда»)), один выход «tips» («чаевые»),
- три правила типа «если ..., то» (в соответствии с тремя приведенными предложениями) и по три значения (соответственно 0 баллов, 5 баллов, 10 баллов и 0 %, 10 %, 20 %) для центров функций принадлежности входов и выхода.

Построим данную систему, используя алгоритм вывода Mamdani и, как в предыдущем примере, описывая требуемые действия по шагам алгоритма.

1. Командой `fuzzy` запускаем редактор системы нечеткого моделирования. По умолчанию в системе используется алгоритм вывода типа Мамдани (это имя отражено в текстовом поле центрального блока белого цвета). Оставим без изменения. Для инициализации двух входов с помощью пункта меню `Edit /Add Variable/Input` добавляем в систему второй вход (в окне редактора появляется второй желтый блок с именем `input2`).

После это, активируя блок `input1`, изменим его имя на «Service», подтверждая ввод имени нажатием клавиши `Enter`. Аналогичным образом устанавливаем имя «Food» блоку `input2` и «tips» – выходному блоку (справа сверху) `output1`.

Сохраним экспертную систему под именем `Restaurant`, используя меню `File/Export/To Workspace` (сохранить в рабочем пространстве). Вид окна редактора после изменений представлен на рисунке 2.12.

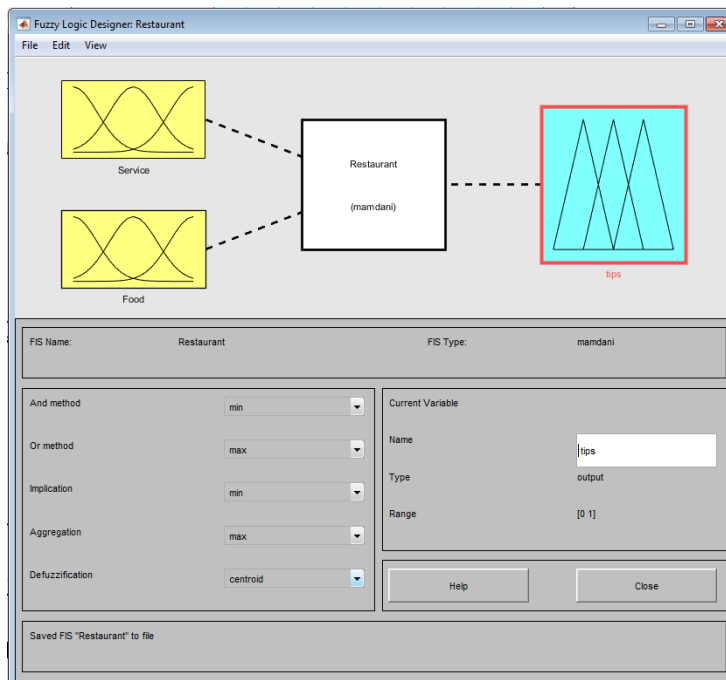


Рис. 2.12. Общий вид окна интерфейса редактора нечетких систем после задания структуры системы.

2. Далее инициализируем функции принадлежности входных и выходных переменных. Программу-редактор функций принадлежности можно вызвать через пункт меню `Edit/Membership functions`, двойным кликом мышью на значке, отображающую соответствующую переменную, нажатием комбинации клавиш `Ctrl+2`.

Задание и редактирование функций принадлежности начнем с переменной «Service». 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 пункты меню `Edit/Remove All MFs`, `Edit/Add MFs` перейдем к диалоговому окну задания функций принадлежности и определим три функции принадлежности гауссова типа (`gaussmf`). Нажмем кнопку `OK` и возвратимся в окно редактора функций принадлежности. Не изменяя размах и положение заданных функций, заменим только их имена на «poor», «good» и «excellent» аналогично шагу 5 алгоритма предыдущего примера.

Щелчком на значке «Food» войдем в окно редактирования функций принадлежности для этой переменной. Зададим сначала диапазон ее изменения от 0 до 10, а затем, поступая как ранее, зададим две функции принадлежности трапецидальной формы (`trapmf`) с параметрами соответственно `[0 0 1 3]` и `[7 9 10 10]` и именами «bad» и «delicious».

Для выходной переменной «tips» укажем сначала диапазон изменения (от 0 до 20), потом зададим три функции принадлежности треугольной формы с именами «little», «average» и «big» так, как это представлено на рисунке 2.13. Заметим, что можно, разумеется, задать и какие-либо другие функции или выбрать их другие параметры.

Перейдем к конструированию правил. Для этого выберем пункт меню `Edit/Rules`. Далее ввод правил проводится так же, как в пункте 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 «ИЛИ» (`or`); при вводе второго правила, где отсут-

ствует переменная «Food», для нее выбирается опция none. Итоговый набор правил отображен на рисунке 2.14.

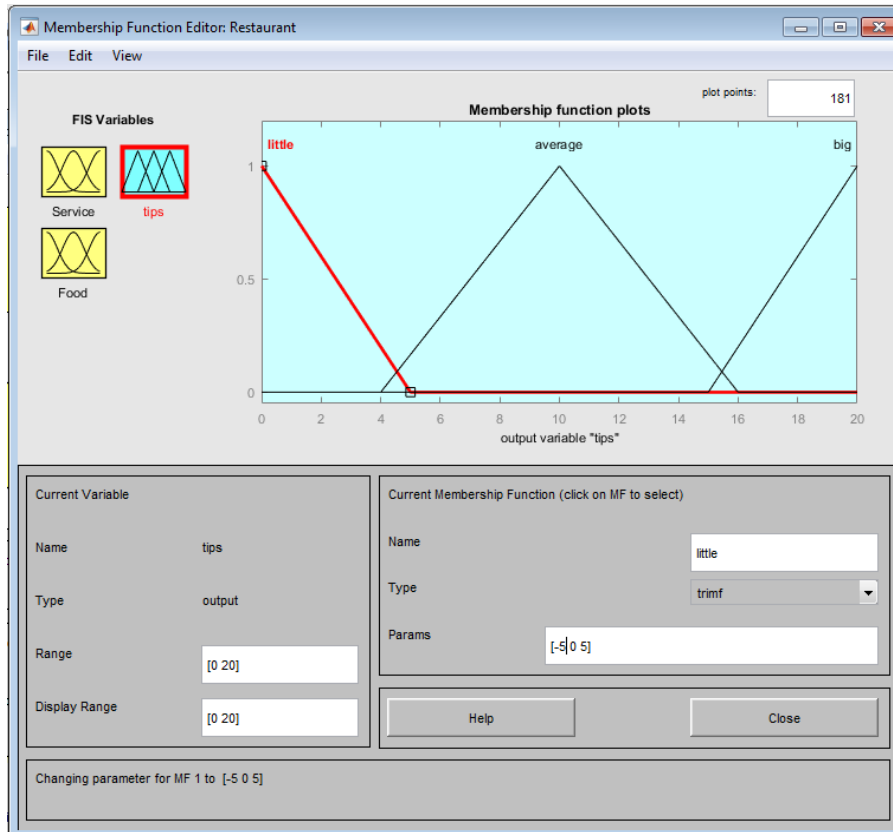


Рис. 2.13. Определение функции принадлежности переменной “tips”.

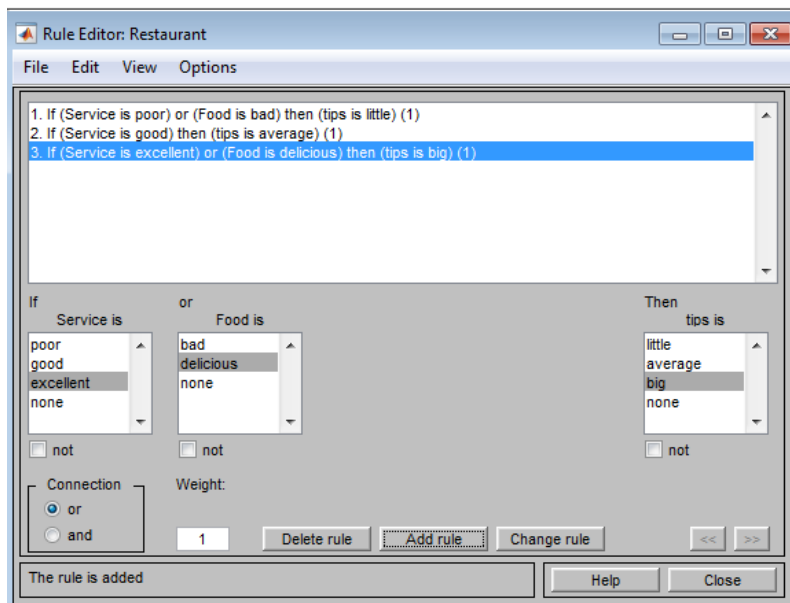


Рис. 2.14. Диалоговое окно для задания правил вывода.

Детальная запись представляется достаточно понятной; единица в скобках после каждого правила указывает его «вес» (*weight*), то есть 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 переход от одной формы к другой происходит с помощью меню *Options/Format* редактора правил.

Можно считать, что последний описанный шаг завершает конструирование модели экспертной системы. Сохраним файл под выбранным именем *Restaurant*.

Откроем (через пункт меню *View/Rules*) окно просмотра правил и установим значения переменных: «*Service*» = 0 (то есть сервис ужасный), «*Food*» более 9 (то есть еда замечательная). Увидим ответ: «*tips*» = 10 (средние), как показано на рисунке 2.15.

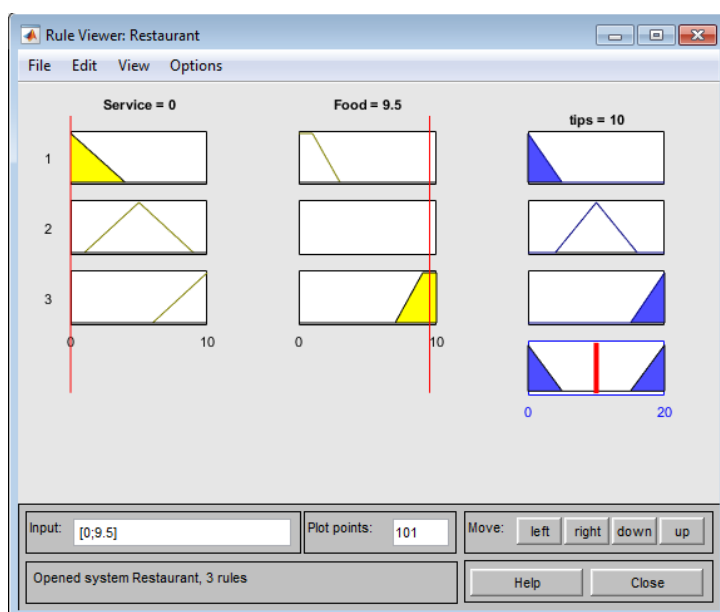


Рис. 2.15. Окно визуализации результатов логического вывода.

Продemonстрируем и другие варианты результатов моделирования при варьировании входных данных. Например, в данном примере обслуживание ценится больше, чем качество еды: при наборе «*Service*» = 10, «*Food*» = 3 система дает результат, соответствующий размеру чаевых 18.4 %, в то время как

набору «Service» = 0, «Food» = 10 чаевые средние – 10 %. Это отвечает, в целом, заданию исходных правил логического вывода.

В общем случае, требуется усредненное мнение нескольких экспертов или более углубленный анализ на основе методов получения экспертных оценок. Кроме того, для обеспечения адекватности результатов моделирования можно варьировать алгоритмы логического вывода, задание функций принадлежности (вида и числовых параметров), а также количество переменных.

На рисунке 2.16 показана поверхность, соответствующую формированию отклика системы на входные воздействия, визуализированная с помощью пункта меню View/Surface. Для просмотра особенностей графической визуализации можно воспользоваться функцией поворота графика (манипуляция мышью в графической области). В открывшемся окне, меняя имена переменных в полях ввода (X (input) и Y (input)) можно задать и просмотр одномерных зависимостей, например, «tips» от «Food»), как показано на рисунке 2.17.

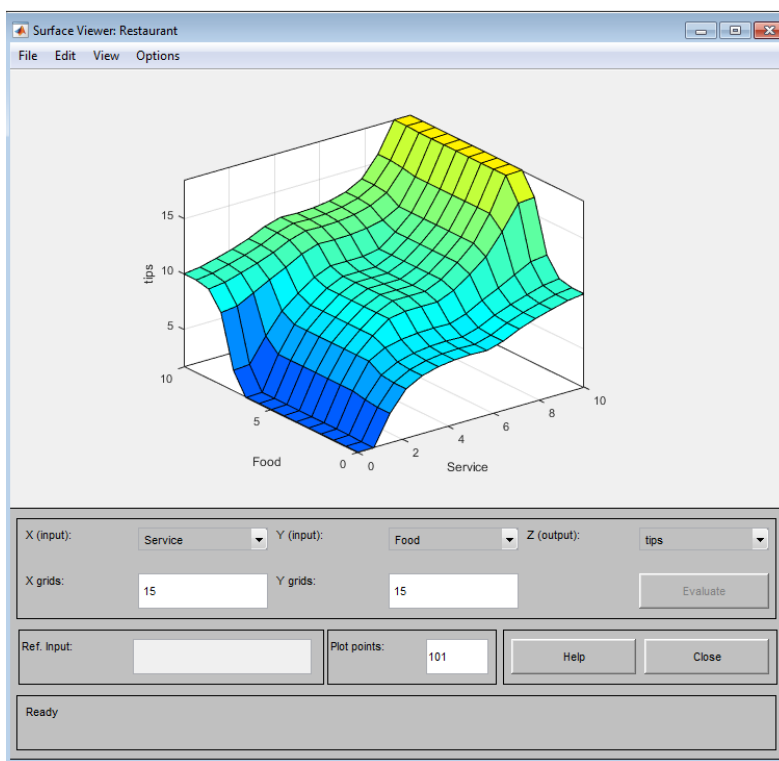


Рис. 2.16. Визуализация поверхности, задающей зависимость выходной переменной от входных.

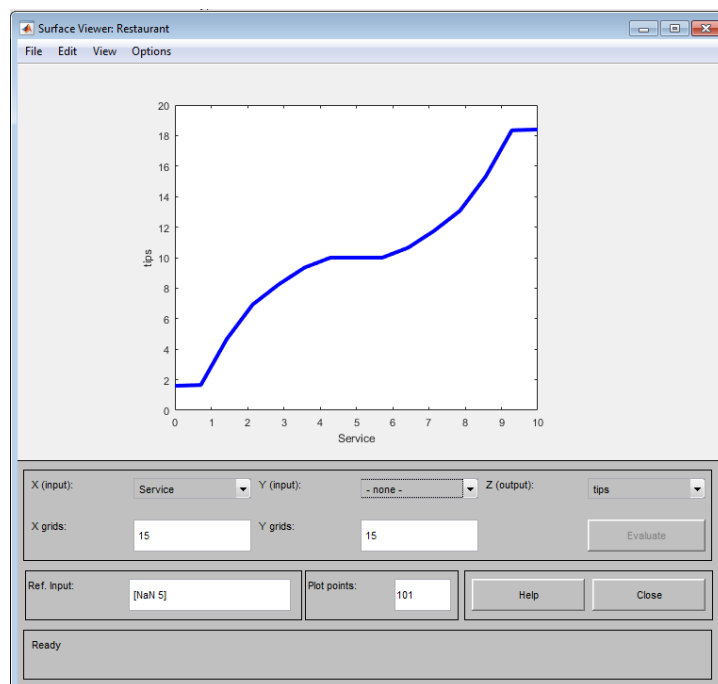


Рис. 2.17. Визуализация одномерной зависимости выходной переменной «tips» от входной переменной «Service».

При сохранении сконструированной нечеткой системы с использованием меню File/Export/To file на диске создается текстовый ASCII файл с расширением .fis. Его можно просматривать, при необходимости редактировать вне системы MATLAB, а также использовать повторно при последующих сеансах работы с системой. Сохранение с использованием пункта File/Export/To Workspace делает доступной реализацию модели созданной нечеткой системы в среде MATLAB в течение текущего сеанса работы и не допускает ее дальнейшего вызова.

Пример 2. Построить нечеткую базу знаний для задачи определения временных затрат для решения студентом задач некоторого учебного пособия (учитывать успеваемость студента, количество решаемых вариантов, состояние здоровья, наличие дополнительной литературы и т.д).

1. Выделим из предметной области две лингвистические переменные и зададим их значения – построим функции принадлежности:

(вход) x – «успеваемость студента». Пусть x принимает значения: {низкая, средняя, высокая}, которые можно оценить, используя шкалу $x \in [2,5]$.

(вход) y – «количество решаемых вариантов». Пусть y принимает значения: {мало, достаточно, много}, которые можно оценить, используя шкалу $y \in [1,11]$ (всего решается от 1 до 11 вариантов).

(выход) $time$ – «временные затраты». Пусть z принимает значения: {немного, много} времени, которые можно оценить, используя шкалу $z \in [1,9]$ (тратиться часов в неделю на решение задач).

Зададим функции принадлежности в пакете Fuzzy Logic Toolbox, как показано на рисунках 2.18-2.20.

2. Сформулируем на естественном языке правила вывода в виде импликаций или предложений с конструкцией «Если.., то», учитывая закономерности предметной области.

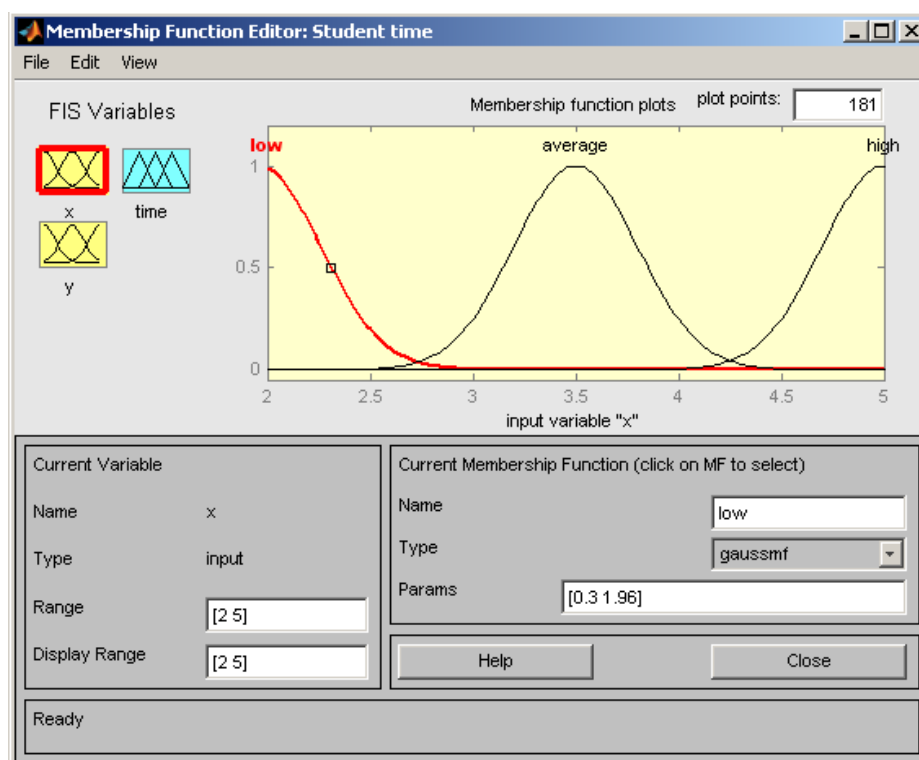


Рис. 2.18. Задание x в пакете Fuzzy Logic Toolbox.

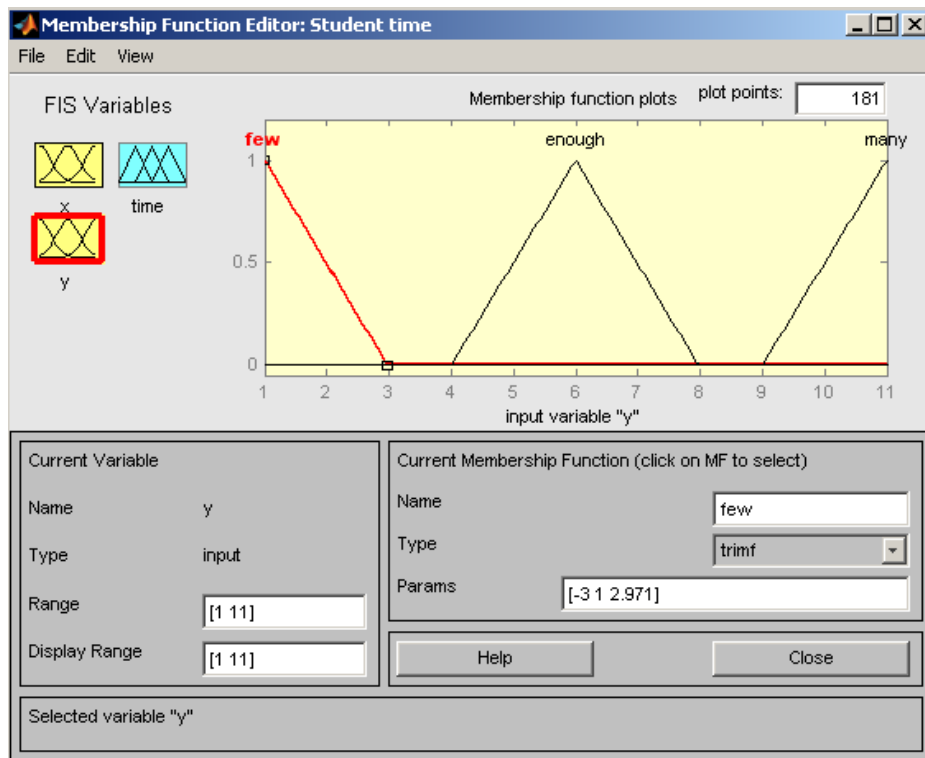


Рис. 2.19. Задание y в пакете Fuzzy Logic Toolbox.

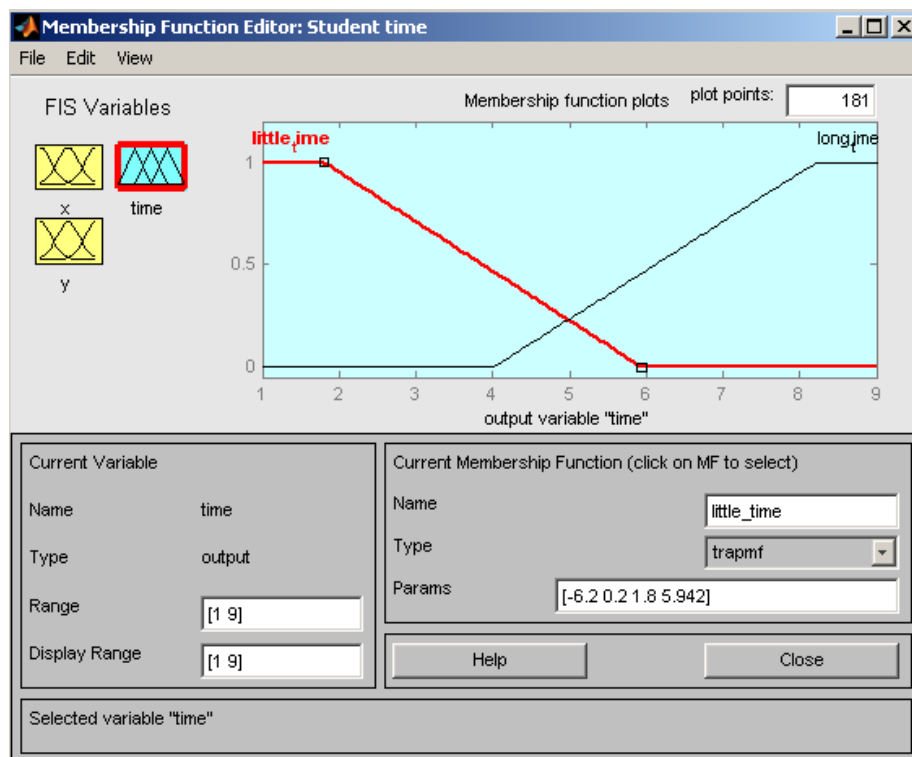


Рис. 2.20. Задание t в пакете Fuzzy Logic Toolbox.

«Если успеваемость студента высокая или хорошая и он решит малое количество вариантов, то ему требуется немного времени»;

«Если успеваемость студента высокая или хорошая и он решит много вариантов, то ему требуется достаточно много времени»;

«Если успеваемость студента низкая и он решит много вариантов, то ему требуется много времени»;

«Если успеваемость студента средняя и он решит большое количество вариантов, то ему требуется много времени».

Формализуем эти нечеткие правила в пакете Fuzzy Logic Toolbox, рисунок 2.21.

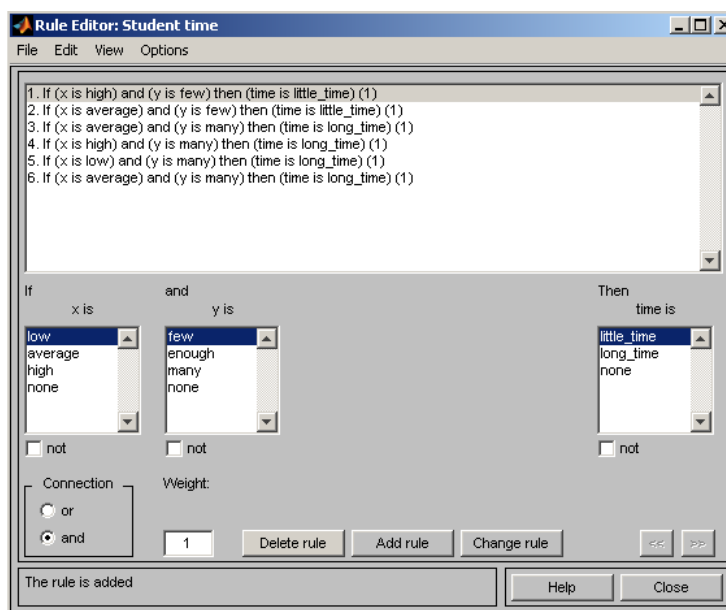


Рис. 2.21. Формализация нечетких правил в пакете Fuzzy Logic Toolbox.

Контрольные вопросы

1. Раскройте понятия нечеткое множество, «лингвистическая переменная». Приведите пример.
2. Сформулируйте определение функции принадлежности. Каков ее смысл? Приведите пример.
3. Какие операции над нечеткими множествами Вам известны?
4. Опишите общий процесс нечеткого вывода. Как строится композиция нечетких множеств?
5. Раскройте суть алгоритма Мамдани и алгоритма Сугено для нечеткого вывода.

6. Какие методы приведения к четкости Вам известны.
7. Опишите кратко возможности пакета Fuzzy Logic Toolbox.

2.4 Индивидуальные задания

Для предложенного варианта задания создать нечеткую модель в Fuzzy Logic Toolbox и провести анализ ее работы. Для реализации нечеткой модели в виде нечеткой экспертной системы необходимо выполнить следующую последовательность действий.

1. Выделить из предметной области лингвистические переменные (самостоятельно, не менее 2, задать их значения (построить функции принадлежности)).
2. Сформулировать на естественном языке правила вывода в виде предложений «Если..., то» (самостоятельно), учитывая закономерности предметной области. Формализовать нечеткие правила.
3. Выполнить все этапы логического вывода (фаззификацию, агрегирование и дефаззификацию) с помощью возможностей пакета Fuzzy Logic Toolbox.
4. провести анализ работы системы, придавая входным параметрам случайные значения (в отчете показать, как минимум 3 варианта реализаций).

Вариант 1. Построить нечеткую базу знаний для задачи определения качества закупки (учитывать соотношения цены, качества товара, объема закупок и т.д.).

Вариант 2. Построить нечеткую базу знаний распределения уровня нагрузок спортсмена (учитывать квалификацию спортсмена, уровень физического состояния, число потребляемых калорий и т.д.)

Вариант 3. Построить нечеткую базу знаний для задачи оценки управления транспортным средством (учитывать регулировку скорости с учетом передачи, погодные условия, интенсивности потока и т.д.).

Вариант 4. Построить нечеткую базу знаний для задачи оценки управления транспортным средством (учитывать управление рулем, газом, тормозом при въезде в гараж).

Вариант 5. Построить нечеткую базу знаний для задачи оценки подбора специй для блюда (учитывать соотношение количества и остроты специй, рецептуры, предпочтений, объема пищи и т.д.).

Вариант 6. Построить нечеткую базу знаний для задачи подачи электроэнергии в условиях экономии (учитывать время суток, тип помещений, количество людей, тип оборудования и т.д.).

Вариант 7. Построить нечеткую базу знаний для задачи расчета потребления бензина (учитывать тип совершаемых маневров, уровень подготовки водителя, состояние автомобиля, тип автомобиля и т.д.).

Вариант 8. Построить нечеткую базу знаний для задачи настройки аудиосистемы (мощность колонок, их количество, размер помещения, назначение установки и т.д.).

Вариант 9. Построить нечеткую базу знаний для задачи выбора дозы снотворного (количество препарата, действие препарата, восприимчивость к выбранному препарату, цель и т.д.).

Вариант 10. Построить нечеткую базу знаний для задачи определения количества линий в службе поддержки (учитывать количество обслуживаемых клиентов, среднюю частоту обращения в службу одного клиента, среднее время обслуживания одной заявки, квалификацию персонала и т.д.).

3 ПРОГНОЗИРОВАНИЕ ВРЕМЕННЫХ РЯДОВ С ПОМОЩЬЮ ГИБРИДНЫХ СЕТЕЙ С ИСПОЛЬЗОВАНИЕМ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ПАКЕТА MATLAB¹

3.1 Назначение и применение гибридных нейронных сетей

Определенный тип системы, основанный на моделях искусственного интеллекта, характеризуется определенными аспектами, что позволяет применять для решения разных классов задач специфические приемы. Например, нейронная сеть неплохо справляется с распознаванием и идентификацией изображений. Но такие системы достаточно сложны в плане аргументирования принятия решений. Нейронная сеть обучается достаточно медленно, в результате сеть формирует знания «автоматически», не предоставляя исследователю логику полученного вывода. Кроме того, нейронная сеть не позволяет ввести в нее дополнительную информацию в ходе обучения.

Для нечетких систем, наоборот, характерна доступность и аргументированность выводов. Однако, они не обладают способностью к обучению. В связи с чем, исторически возникла идея создания гибридных сетей, объединяющих достоинство нейронных сетей и нечеткого моделирования. Логические выводы в таких системах осуществляются с помощью нечеткой модели, а используемые функции принадлежности настраиваются с применением методов обучения нейронной сети. Таким образом, гибридные нейронные сети (ANFIS – Adaptive Neuro Fuzzy Inference System или адаптивная нечеткая нейронная система вывода) позиционируются как сети, которые способны приобретать новые знания и для исследователя являются логически понятными.

Входы, выходы и веса такой сети являются вещественными числами, принадлежащими отрезку $[0,1]$.

Примером подобной сети может служить система, имеющая следующую базу знаний (рисунок 3.1).

¹ Текст лабораторной работы составлен при участии Галаган К.Ю. в рамках выполнения магистерской диссертации по теме «Фрактальные закономерности и нейро-нечеткое прогнозирование финансовых временных рядов».

Правило 1: если x_1 есть L_1 и x_2 есть L_2 , и x_3 есть L_3 ,
тогда z есть H ;

Правило 2: если x_1 есть H_1 и x_2 есть H_2 , и x_3 есть H_3 ,
тогда z есть H ;

Правило 3: если x_1 есть S_1 и x_2 есть S_2 , и x_3 есть S_3 ,
тогда z есть S ,

где x_1, x_2, x_3 – входные переменные;

z – выход системы;

$L_1, L_2, L_3, H_1, H_2, H_3, H, M, S$ – некоторые нечеткие множества с функциями принадлежности сигмоидного типа (для упрощения записи последующих выкладок функции принадлежности в данном случае обозначены так же, как и соответствующие нечеткие множества).

Рис. 3.1. Пример описания структуры вывода гибридной сети.

$$L_j(t) = \frac{1}{1 + \exp(b_j(t - c_j))}, H_j(t) = \frac{1}{1 + \exp(-b_j(t - c_j))}, j = 1, 2, 3,$$

$$H(t) = \frac{1}{1 + \exp(b_4(t - c_4 + c_5))}, M(t) = \frac{1}{1 + \exp(-b_4(t - c_4))}, S(t) = \frac{1}{1 + \exp(b_4(t - c_4))}.$$

Определение значения выходной переменной проводится на основе следующего алгоритма.

Шаг I. Для каждого правила проводится оценка значения истинности предпосылок:

$$\begin{aligned} \alpha_1 &= L_1(a_1) \wedge L_2(a_2) \wedge L_3(a_3), \\ \alpha_2 &= H_1(a_1) \wedge H_2(a_2) \wedge L_3(a_3), \\ \alpha_3 &= H_1(a_1) \wedge H_2(a_2) \wedge H_3(a_3), \end{aligned} \tag{3.1}$$

где $\alpha_1, \alpha_2, \alpha_3$ задают для системы текущие значения входов;

Шаг II. Специфицируются частные значения выходов для каждого правила:

$$z_1 = B^{-1}(\alpha_1) = c_4 + c_5 + \frac{1}{b_4} \ln \frac{1-\alpha_1}{\alpha_1},$$

$$z_2 = B^{-1}(\alpha_2) = c_4 + \frac{1}{b_4} \ln \frac{1-\alpha_2}{\alpha_2},$$

$$z_3 = B^{-1}(\alpha_3) = c_4 + \frac{1}{b_4} \ln \frac{1-\alpha_2}{\alpha_2}.$$

Шаг III. Далее определяется значение общего выхода:

$$z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2 + \alpha_3 z_3}{\alpha_1 + \alpha_2 + \alpha_3}.$$

Указанный процесс продемонстрирован на рисунке 3.2.

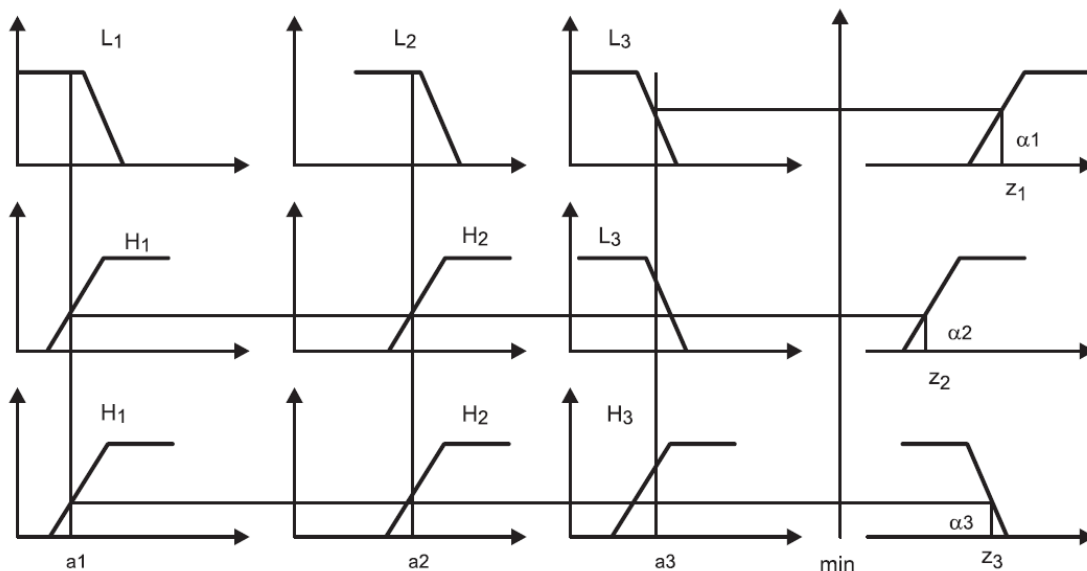


Рис. 3.2. Схема алгоритма вывода (абсциссы колонок графиков соответствуют: переменным x_1 , x_2 , x_3 , переменной вывода z соответственно).

Архитектура гибридной нейронной сети показана на рисунке 3.3, которая характеризуется следующим образом.

Слой 1 (layer_1): выходы узлов – есть значения функций принадлежности при заданных значениях входов.

Слой 2 (layer_2): выходы нейронов – степени истинности предпосылок каждого правила базы знаний системы.

Слой 3 (layer_3): нейроны проводят расчет значений:

$$\beta_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2 + \alpha_3}, \beta_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2 + \alpha_3}, \beta_3 = \frac{\alpha_3}{\alpha_1 + \alpha_2 + \alpha_3}.$$

Слой 4 (layer_4): нейроны ответственны за расчет:

$$\beta_1 z_1 = \beta_1 H^{-1}(a_1), \beta_2 z_2 = \beta_2 M^{-1}(a_2), \beta_3 z_3 = \beta_3 S^{-1}(a_3).$$

Слой 5 (layer_5): единственный нейрон вычисляет выход сети:

$$z_0 = \beta_1 z_1 + \beta_2 z_2 + \beta_3 z_3.$$

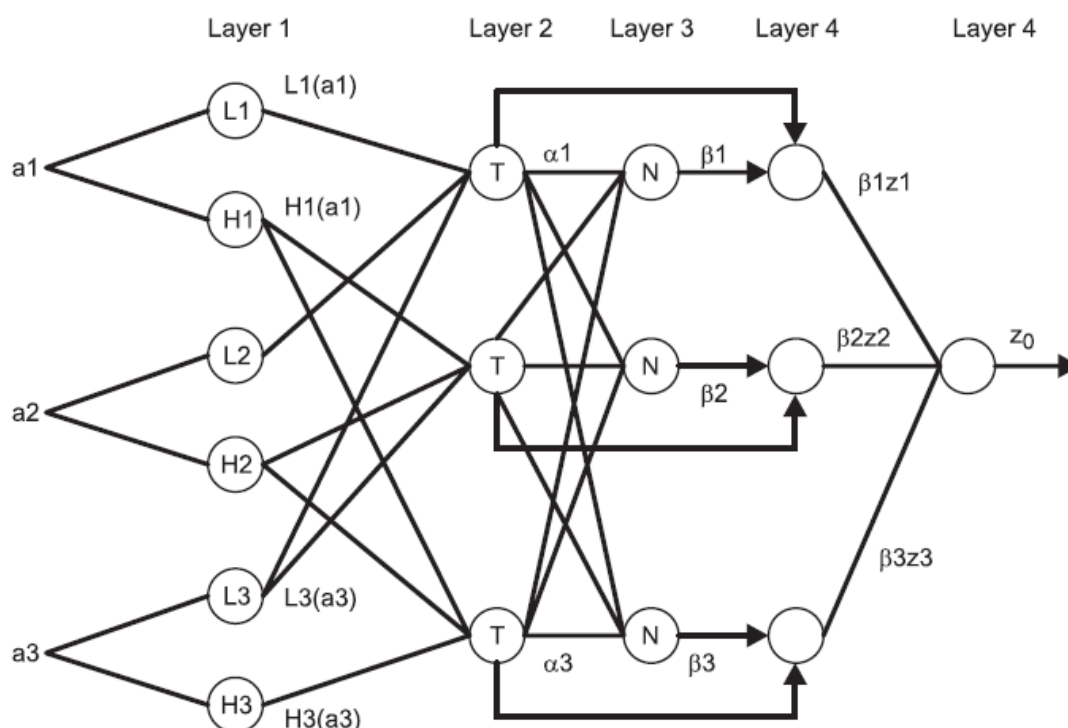


Рис. 3.3. Архитектура гибридной нейронной сети ANFIS.

Для корректировки параметров системы используется алгоритм обратного распространения ошибки или комбинированный метод гибридных сетей. Анализ исследований, представленных в литературных источниках, свидетельствует о том, что гибридные нейронные сети могут быть успешно использованы для прогнозирования временных рядов. Более того, в ряде случаев, они дают более адекватные результаты, чем традиционные модели прогнозирования.

3.2 Графический интерфейс приложения ANFIS

для моделирования гибридных нейронных систем

Графический интерфейс гибридных (нечетких) нейронных систем вызывается функцией `anfisedit` из режима командной строки. Использование функции приводит к появлению окна редактора гибридных систем (`ANFIS Editor`, ANFIS-редактор), вид которого представлен на рисунке 3.4.

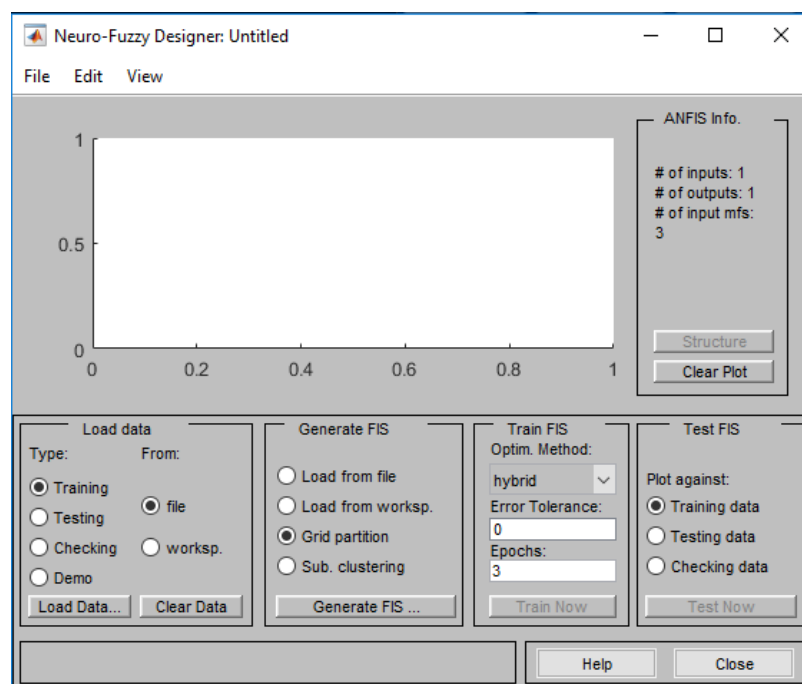


Рис. 3.4. Окно редактора гибридных систем.

С помощью данного редактора осуществляются создание или загрузка структуры гибридной системы, просмотр структуры, настройка ее параметров, проверка качества функционирования такой системы.

Создание структуры, настройка параметров и проверка осуществляются по выборкам (наборам данных) – обучающей (`Training data`), проверочной (`Checking data`) и тестирующей (`Testing data`). Предварительно они должны быть представлены в виде текстовых файлов (с расширением `dat` и разделителями-табуляциями), первые столбцы которых соответствуют входным переменным, а последний (правый) – единственной выходной переменной; количество строк в таких файлах равно количеству образцов (примеров).

Строгих рекомендаций по объемам указанных выборок не существует.

Обучающая и проверочная выборки непосредственно задействуются в процессе настройки параметров гибридной сети (проверочная – для выяснения ситуации, не происходит 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:

– Меню `File` – работа с файлами моделей (их создание, импорт, экспорт, печать и закрытие);

– Меню `View` – переход к дополнительному инструментарию.

– Меню `Edit` содержит команды:

1) `Undo` – отменить выполненное действие;

2) `FIS Properties` – вызов редактора нечеткой системы (`FIS Editor`);

3) `Membership Functions` – вызов редактора функций принадлежности (`Membership Function Editor`);

4) `Rules` – вызов редактора правил (`Rule Editor`);

5) `Anfis` – в данном случае команда не выполняет никаких действий.

Набор опций `Load data` (загрузка данных) 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 которые приводит к требуемым действиям – `Load Data` (загрузить данные) и `Clear Data` (стереть 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 следующие опции:

- загрузку структуры системы с диска (`from disk`);
- загрузку структуры системы из рабочей области MATLAB (`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 (`Sub.clustering`).

Кроме того, имеется также кнопка `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 (`backpropa`), а также 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 после проведения заданного количества циклов.

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

В правом верхнем углу окна ANFIS-редактора выдается информация (`ANFIS Info`) о проектируемой системе: количество 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.

3.3 Примеры построения гибридной нейро-нечеткой системы

Пример 1. Работу с редактором гибридных нейронных сетей рассмотрим на примере восстановления зависимости $y = x^2$ по данным таблицы:

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

Предположим, что эти данные подготовлены в файле «Pr.dat», представленные в следующем виде:

```
-1      1
-0.6   0.36
0.00   0.00
0.4    0.16
1      1
```

Создание и проверку системы 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, как представлено на рисунке 3.5.

2. В группе опций `Generate FIS` по умолчанию активизирован вариант `Grid partition`. Не будем изменять и нажмем кнопку `Generate FIS`, после чего появится диалоговое окно для задания числа и типов функций принадлежности, рисунок 3.6.

Сохраним все установки по умолчанию, согласившись с ними нажатием кнопки `ОК`. Произойдет возврат в основное окно ANFIS-редактора. Теперь

структура гибридной сети создана, и ее графический вид можно просмотреть с помощью кнопки Structure в соответствии с рисунком 3.7.

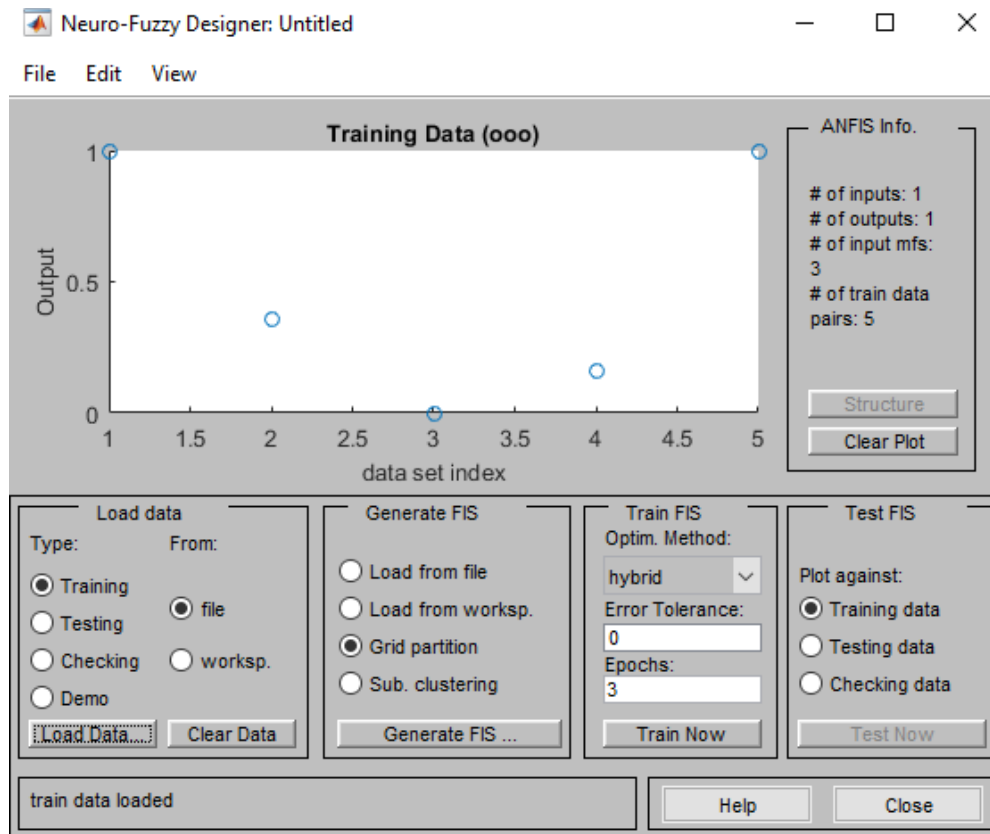


Рис. 3.5. Окно ANFIS-редактора после загрузки обучающей выборки.

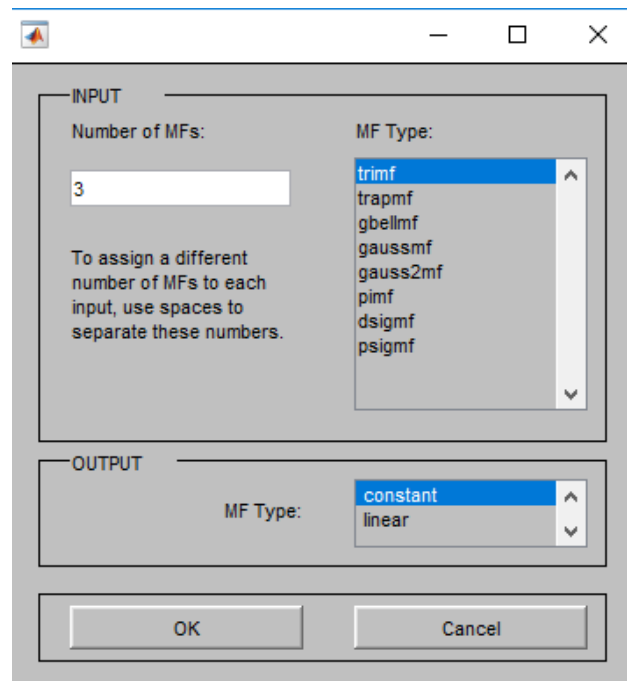


Рис. 3.6. Окно задания функций принадлежности.

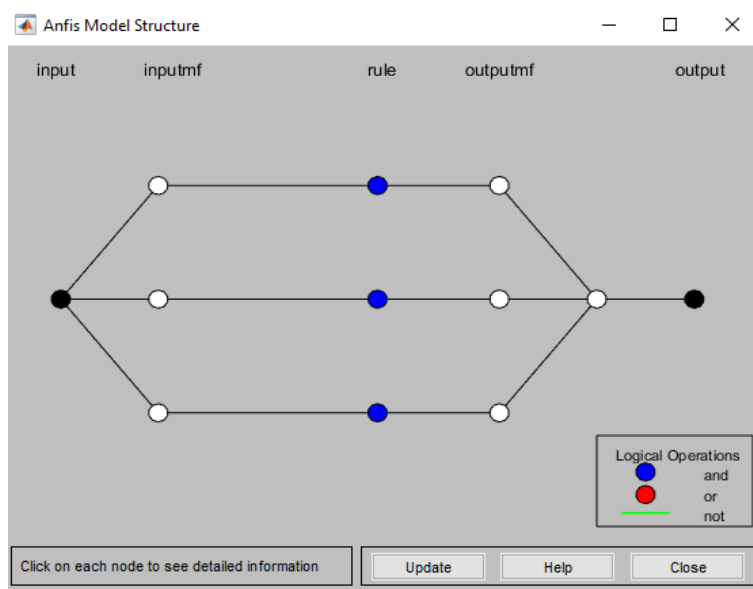


Рис. 3.7. Структура создания гибридной сети.

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 100, после чего нажмем кнопку запуска процесса обучения (`Train Now`). Получился результат в виде графика ошибки сети в зависимости от числа проведенных циклов обучения, из которого следует, что фактически обучение закончилось после 60 циклов, рисунок 3.8.

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 как проиллюстрировано на рисунке 3.9.

5. Сохраним разработанную систему в рабочую область с именем «Pr» (`File/Export/From Workspace`) и для выполнения прогноза разработанной системой средствами FIS-редактора из командной строки MATLAB выполним команду:

```
out=evalfis([5],anfis) % если входных переменных будет несколько, то они разделяются пробелами.
```

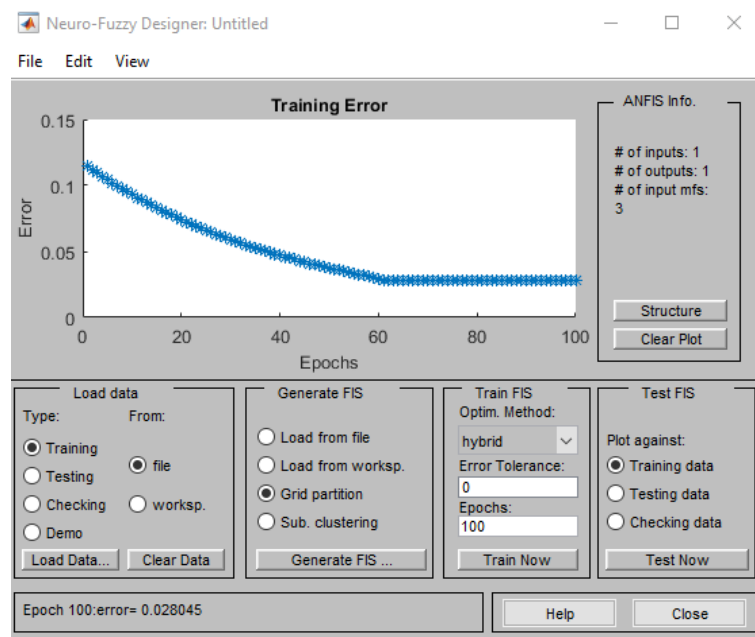


Рис. 3.8. Результат обучения сети.



Рис. 3.9. Результат тестирования обученной системы.

В данном случае используется 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 сфор-

мированных правил и придания им содержательной (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 систем классификации (в случае бинарной или дискретной выходной переменной), изучение механизма явлений.

Пример 2. Построим гибридную нейронную сеть, которая по данным о курсе валюты за четыре банковских дня предсказывает курс на пятый день (экстраполирует).

Нейросетевое прогнозирование в ANFIS-редакторе начинается с загрузки временного ряда. На рисунке 3.10 представлен исследуемый временной ряд (обменный курс валют евро/американский доллар), загруженный в ANFIS-редактор.

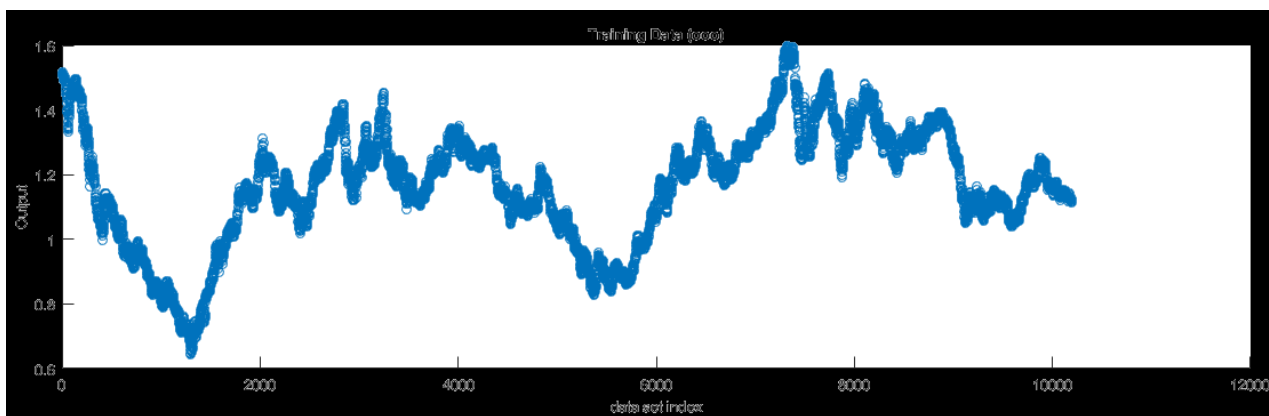


Рис. 3.10. Исследуемый временной ряд «EUR/USD».

На втором этапе происходит обучение нейронной сети. Динамика ошибки прогноза (error), то есть среднее отклонение фактических значений от прогнозных, с учетом циклов обучения (epochs) на исходных (тренировочных) данных представлена на рисунке 3.11.

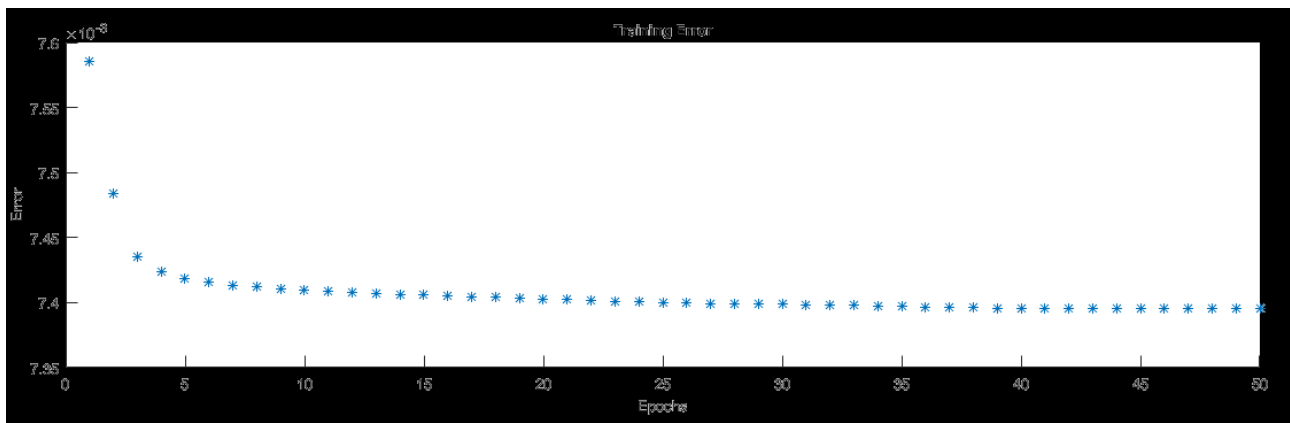


Рис. 3.11. Обучение нейронной сети для ряда «EUR/USD».

Среднее значение ошибки прогноза для обучаемых данных – 0.0073954 при 50 циклах обучения. Результаты тестирования гибридной сети для определения прогнозных значений финансового временного ряда «EUR/USD» на фиксированные даты, то есть прогноз осуществляется на один шаг, представлены на рисунке 3.12. Прогноз выполняется для определения значений в мае 2019 г.

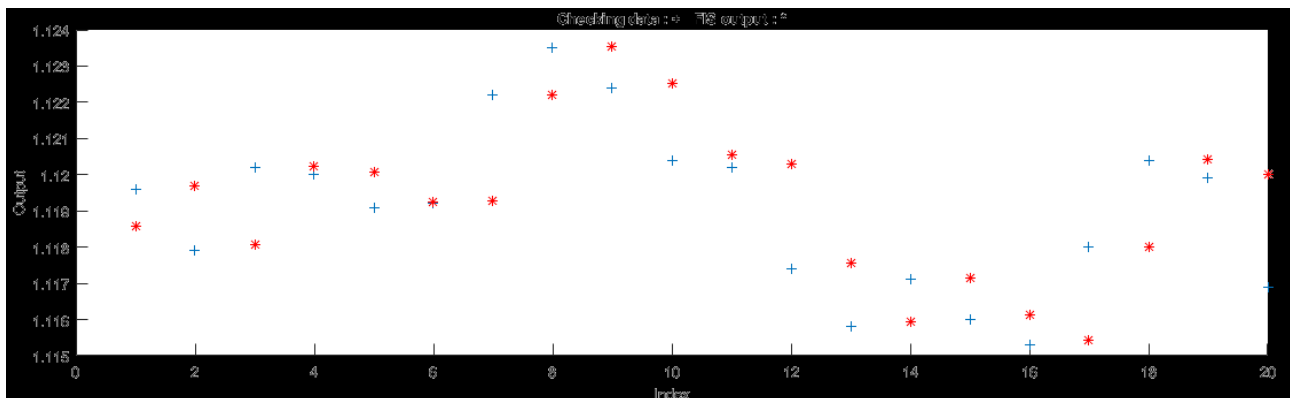


Рис. 3.12. Результаты тестирования гибридной сети для прогноза значений ряда «EUR/USD» на один шаг (+ – фактические значения, * – прогнозные)

Средняя ошибка прогноза для тестируемых данных составила 0.0017727.

Контрольные вопросы

1. Сформулируйте определение гибридной нейронной сети. Какие системы она в себе объединяет?
2. Чем являются входы, выходы и веса гибридных нейронных сетей?

3. Какие преимущества и недостатки имеют гибридные нейронные системы?
4. Для чего предназначен ANFIS-редактор?
5. Какие этапы можно выделять для построения нейронной сети?
6. Какие типы загружаемых данных (наборы данных) используются в ANFIS-редакторе? И какие из них являются обязательными для ввода?

3.4 Индивидуальные задания

Для реализации прогнозирования с помощью гибридной нейронной сети выполнить задания.

1. Сформировать обучающую, тестовую и проверочную выборки, воспользовавшись данными в соответствии с вариантом задания. Использовать данные² для текущего календарного года за любой выбранный и доступный период времени.

2. Построить нечеткую нейронную сеть и, экспериментируя с методами обучения, количествами функций принадлежности во входном слое, добиться наилучшего результата прогнозирования.

3. Сравнить полученные результаты и сделать выводы о качестве прогноза.

Вариант 1. Построить гибридную нейронную сеть, которая по данным о курсе доллара США по отношению к рублю за четыре банковских дня предсказывает курс на пятый день.

Вариант 2. Построить гибридную нейронную сеть, которая по данным о курсе евро по отношению к рублю за четыре банковских дня предсказывает курс на пятый день.

Вариант 3. Построить гибридную нейронную сеть, которая по данным о курсе китайского юаня по отношению к рублю за четыре банковских дня предсказывает курс на пятый день.

² Для работы воспользоваться данными открытого интернет-источника <http://m.ru.investing.com/>

Вариант 4. Построить гибридную нейронную сеть, которая по данным о курсе доллара США по отношению к китайскому юаню за четыре банковских дня предсказывает курс на пятый день.

Вариант 5. Построить гибридную нейронную сеть, которая по данным о курсе евро по отношению к доллару США за четыре банковских дня предсказывает курс на пятый день.

Вариант 6. Построить гибридную нейронную сеть, которая по данным о стоимости Bitcoin за четыре банковских дня предсказывает стоимость на пятый день.

Вариант 7. Построить гибридную нейронную сеть, которая по данным о стоимости золота за четыре банковских дня предсказывает стоимость на пятый день.

Вариант 8. Построить гибридную нейронную сеть, которая по данным о индексе Доу-Джонса за четыре банковских дня предсказывает стоимость на пятый день.

Вариант 9. Построить гибридную нейронную сеть, которая по данным о стоимости нефти марки «Brent» за четыре банковских дня предсказывает стоимость на пятый день.

Вариант 10. Построить гибридную нейронную сеть, которая по данным о стоимости платины за четыре банковских дня предсказывает стоимость на пятый день.

4 МОДЕЛИ ФРАКТАЛЬНЫХ СТРУКТУР

В книге «Фрактальная геометрия природы» Бенуа Мандельброт писал: «Почему геометрию часто называют холодной и сухой? Одна из причин заключается в ее неспособности описать форму облака, горы, дерева или берега моря. Облака – это не сферы, линии берега – это не окружность, и кора не является гладкой, и молния не распространяется по прямой. Природа демонстрирует нам не просто более высокую степень, а совсем другой уровень сложности. Число различных масштабов длин в структурах всегда бесконечно. Существование этих структур бросает нам вызов в виде трудной задачи изучения тех форм, которые Евклид отбросил как бесформенные, – задачи исследования морфологии аморфного».

Поэтому необходимы были новые геометрические понятия и методы для описания этих объектов. Одним из таких понятий и явилось понятие фрактала, введенное Б. Мандельбротом в 1975 г. Конструкции подобные фрактальным в той или иной форме появлялись много лет назад, но вся ценность понятия фрактала была осознана только в 70-х годах XX века. Понятие фрактала относится к некоторой статичной геометрической конфигурации, а для описания динамических явлений используется понятие хаоса.

Сейчас очевидно, что с помощью евклидовой геометрии сложно описывать природные объекты, т.к. в ней отсутствует некоторая нерегулярность, беспорядок. В таких случаях и применяется теория фракталов. Фракталы используются при создании изображений деревьев, горных ландшафтов, облаков; при анализе сигналов сложной формы; во многих областях в физики, химии, биологии.

4.1 Фракталы. Классификация фракталов

Понятия фрактал и фрактальная геометрия прочно вошли в обиход математиков и программистов. Слово фрактал образовано от латинского «fractus» и в переводе означает «состоящий из фрагментов». Оно было предложено Бенуа

Мандельбротом в 1975 году для обозначения нерегулярных, но самоподобных структур. В самом простом случае небольшая часть фрактала содержит информацию о всем фрактале.

Определение фрактала, данное Мандельбротом, звучит так: «Фракталом называется структура, состоящая из частей, которые в каком-то смысле подобны целому». В основном фракталы делят на геометрические, алгебраические и стохастические. Однако существуют и другие классификации.

Рукотворные и природные фракталы. К рукотворным относятся те фракталы, которые были придуманы учёными, они при любом масштабе обладают фрактальными свойствами. На природные фракталы накладывается ограничение на область существования – то есть максимальный и минимальный размер, при которых у объекта наблюдаются фрактальные свойства.

Классификация фракталом по степени определенности: детерминированные (алгебраические и геометрические) и недетерминированные (стохастические).

Геометрические фракталы. История фракталов началась с геометрических фракталов, которые исследовались математиками в XIX в. Геометрические фракталы являются также самыми наглядными, т.к. очевидна самоподобность.

В двумерном случае их получают с помощью некоторой ломаной (или поверхности в трехмерном случае), называемой генератором. За один шаг алгоритма каждый из отрезков, составляющих ломаную, заменяется на ломаную-генератор, в соответствующем масштабе. В результате бесконечного повторения этой процедуры, получается геометрический фрактал.

Рассмотрим один из таких фрактальных объектов – триадную кривую Коха. Эта кривая была описана в 1904 году шведским математиком Хельге фон Кохом, который, изучая работы Карла Вейерштрасса и Георга Кантора, натолкнулся на описание некоторых странных кривых с необычным поведением. Кривая Коха примечательна тем, что нигде не имеет касательной, т.е. нигде недифференцируема, хотя всюду непрерывна.

Построение кривой начинается с отрезка единичной длины (рис.1) – это 0-е поколение кривой Кох. Далее каждое звено (в нулевом поколении один отрезок) заменяется на образующий элемент, обозначенный на рис.1 через $n=1$. В результате такой замены получается следующее поколение кривой Кох. В 1-ом поколении – это кривая из четырех прямолинейных звеньев, каждое длиной по $1/3$. Для получения 3-го поколения проделываются те же действия – каждое звено заменяется на уменьшенный образующий элемент. Итак, для получения каждого последующего поколения, все звенья предыдущего поколения необходимо заменить уменьшенным образующим элементом. Кривая n -го поколения при любом конечном n называется предфракталом. На рисунке 4.1 представлены поколения кривой. При n стремящемся к бесконечности кривая Кох становится фрактальным объектом.

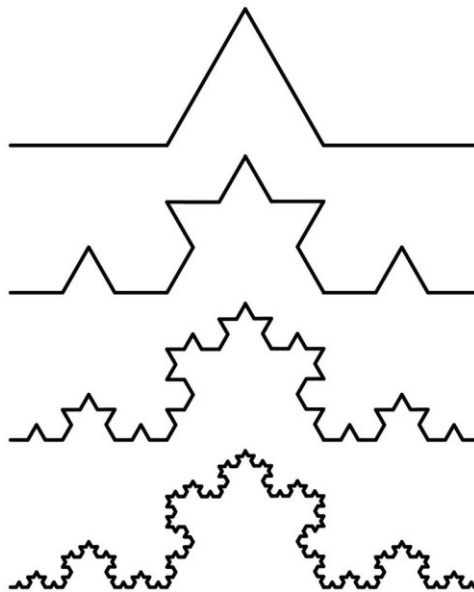


Рис. 4.1. Этапы построения кривой Коха.

Примерами таких фракталов являются: кривые дракона, Коха, Леви, Минковского, Пеано. К геометрическим фракталам также относят фракталы, получаемые похожими процедурами, например: множество Кантора, треугольник Серпинского, ковер Серпинского, кладбище Серпинского, губка Менгера, дерево Пифагора.

Алгебраические фракталы. Это самая крупная группа фракталов. Получают их с помощью нелинейных процессов в n -мерных пространствах. Если фазовым является двумерное пространство, то, окрашивая области притяжения различными цветами, можно получить цветовой фазовый портрет этой системы (итерационного процесса). Меняя алгоритм выбора цвета, можно получить сложные фрактальные картины с причудливыми многоцветными узорами. Неожиданностью для математиков стала возможность с помощью примитивных алгоритмов порождать очень сложные нетривиальные структуры. Самыми известными алгебраическими фракталами являются множества Мандельброта и Жюлиа, Бассейны Ньютона, биоморфы и т.д.

Фракталы при построении которых в итеративной системе случайным образом изменяются какие-либо параметры называются *стохастическими*. Термин «стохастичность» происходит от греческого слова, обозначающего «предположение». Стохастические фракталы приводят в модельным объектам, очень похожим на природные – несимметричным деревья, изрезанным береговым линиям и т.д. Двумерные стохастические фракталы используются при моделировании рельефа местности и поверхности моря.

4.2 Построение фракталов в ППП Matlab

Кривая Коха является типичным детерминированным фракталом. Процесс ее построения описан в п. 4.1.

Кривую Коха можно также построить с помощью двух инвариантных преобразований сжатия-отражения. Преобразование сжатия-отражения записывают в общем виде:

$$\begin{cases} x' = ax + by + c \\ y' = bx - ay + d \end{cases} \quad (4.1)$$

с коэффициентом сжатия $\sqrt{a^2 + b^2}$.

Первые фазы построения кривой Коха показаны на рисунке 4.2.

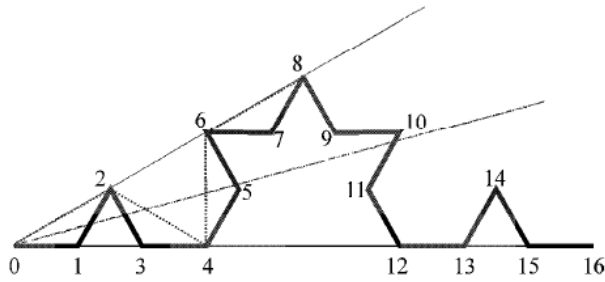


Рис. 4.2. Сжатие-растяжение во фрактале Коха.

Линия, соединяющая точки **0, 2, 4, 6, 8** имеет такую же форму, что и линия, соединяющая точки **0, 4, 8, 12, 16** первой фазы. Только теперь она отражается и сжимается. Показатель сжатия $\frac{1}{\sqrt{3}}$. Это преобразование записывается

формулой (4.1) с $c = d = 0$. Коэффициенты a, b равны координатам точки **8**:

$$a = \frac{1}{2}, b = \frac{1}{2\sqrt{3}}.$$

Итак, кривая Коха инвариантна относительно преобразования (1) с центром в точке $(0,0)$. Аналогично можно написать преобразование «отражение-сжатие» относительно точки $(1,0)$ Тогда преобразования примут вид:

$$T_1 : \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0.5 & \frac{1}{2\sqrt{3}} \\ \frac{1}{2\sqrt{3}} & -0.5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}, T_2 : \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0.5 & \frac{-1}{2\sqrt{3}} \\ \frac{-1}{2\sqrt{3}} & -0.5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2\sqrt{3}} \end{pmatrix}.$$

Однако более простым в реализации оказаться стохастический аналог этого фрактала. Для этого необходимо образовать последовательность точек P_0, P_1, \dots, P_N , каждая из которых получается из предыдущей с помощью применения к ней преобразования T_1 или T_2 , причем каждое из этих преобразований выбирается случайно, с вероятностью 0.5, т.е. $P_{n+1} = T_1(P_n) \vee T_2(P_n)$. На компьютере можно организовать этот процесс, используя генератор псевдослучайных чисел, например: если случайно выброшенное число из интервала $(0,1)$ меньше 0.5 применяем преобразование T_1 , иначе – T_2 .

Построение фрактала Коха с использованием стохастического подхода моделирования можно реализовать в Matlab следующим образом:

```

Editor - Koch_Fractal.m
Fractal_Julia.m x Koch_Fractal.m x +
1 % построение кривой Коха
2 - iter=50000; % количество итераций
3 - A=zeros(50000,2); % координаты точек кривой Коха
4 % задаем размеры фрактала
5 - x=0; y=0;
6 - for k=1:1:iter
7 -     p=rand(1); % случайное число в интервале (0; 1)
8 -     t=x;
9     % если p меньше 0.5, то используем формулы первого преобразования,
10    % иначе - второго
11 -     if p <= 0.5
12 -         x=0.5*x+y./(2*sqrt(3));
13 -         y=t./(2*sqrt(3))-0.5*y;
14 -     else
15 -         x=0.5*x-y./(2*sqrt(3))+0.5;
16 -         y=-t./(2*sqrt(3))-0.5*y+1/(2*sqrt(3));
17 -     end
18 -     A(k,1)=-x; % знак «-» переворачивает фрактал
19 -     A(k,2)=0.05-y; % 0.05 - смещение по оси y (для центрирования фрактала)
20 - end
21 - plot(A(:,1),A(:,2),'r.','LineWidth',1,'MarkerSize',1)|
22

```

Результат выполнения программы имеет вид, представленной на рисунке 3.3:

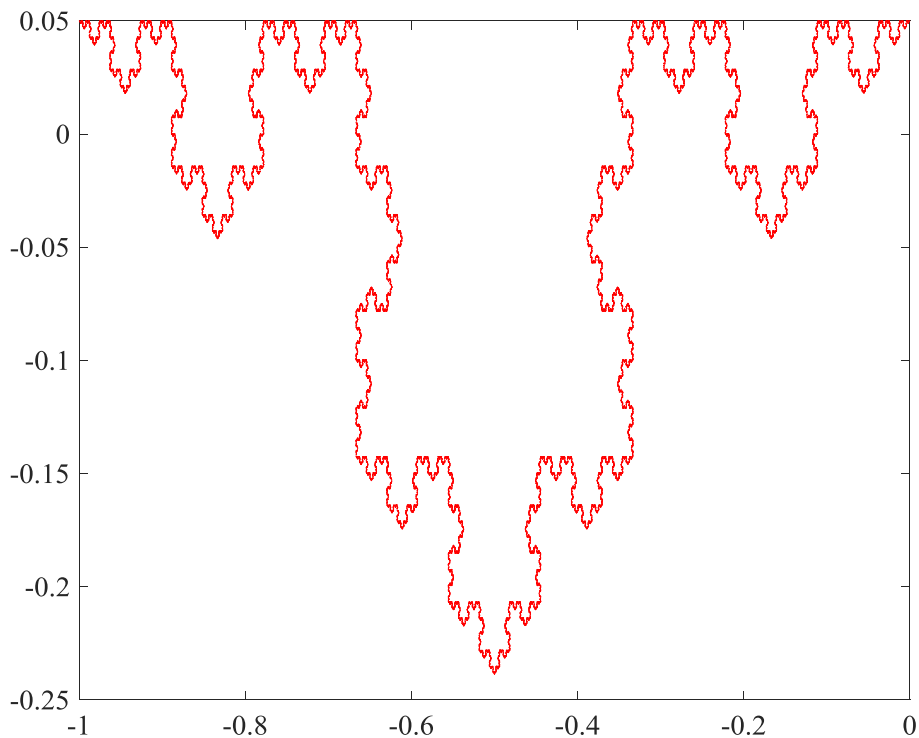


Рис. 3.3. Кривая Коха, построенная в ППП MATLAB.

В 1918 г. Гастон Жюлиа написал подробный «мемуары» о принципах построения новых математических монстров.

В работе Жюлиа рассматриваются итерации отображения вида:

$$x \rightarrow f(x, y), \quad y \rightarrow g(x, y),$$

которые сохраняют углы, т.е. конформные преобразования.

Наиболее изученным примером отображений такого вида является отображение

$$Z[i+1] = Z[i] \cdot Z[i] + C, \tag{4.2}$$

где $Z[i]$ и C – комплексные переменные.

Выражение (4.2) после разделения вещественной и мнимой части запишется в виде $z = x + iy, c = a + ib$:

$$\begin{aligned} x_{n+1} &= x_n^2 - y_n^2 + a, \\ y_{n+1} &= 2x_n y_n + b, \end{aligned} \quad n=0,1,2, \tag{4.3}$$

Оказывается, что это отображение дает множество фракталов, соответствующих множеству Жюлиа.

Фракталы Жюлиа можно разделить на два основных класса: связные и вполне связные. Во втором случае фрактал состоит из несчетного множества дискретных точек. Классический пример – точечное множество Кантора на отрезке $[0,1]$. Если же фрактал связный, он состоит из набора линий, иногда – из единственной замкнутой кривой, иногда – это петли внутри петель, иногда – дендрит.

Для фракталов Жюлиа тип зависит от значений параметров a и b . Мандельброт нашел множество параметров на плоскости (a, b) , для которых фрактал Жюлиа связный, для этого надо проверить орбиту, выходящую из начальной точки $x_0 = a, y_0 = b$. Если эта орбита уходит на бесконечность, то фрактал несвязный.

В качестве примера рассмотрим построение множества Жюлиа. Алгоритм его построения достаточно прост и основан на простом итеративном выраже-

нии (9.3). Итерации выполняются для каждой стартовой точки C прямоугольной или квадратной области – подмножестве комплексной плоскости. Итерационный процесс продолжается до тех пор, пока $Z[i]$ не выйдет за пределы окружности радиуса R , центр которой лежит в точке $(0,0)$, (это означает, что аттрактор динамической системы находится в бесконечности), или после достаточно большого числа итераций (например 200-500) $Z[i]$ сойдется к какой-нибудь точке окружности. В зависимости от количества итераций, в течении которых $Z[i]$ оставалась внутри окружности, можно установить цвет точки C (если $Z[i]$ остается внутри окружности в течение достаточно большого количества итераций, итерационный процесс прекращается и эта точка раstra окрашивается, например, в черный цвет).

Конечное значение z существенно зависит от значения параметра C . Для некоторых значений C значение z сходится к некоторому конечному числу, а для других z расходится (неограниченно расходится). Чтобы избежать численных проблем с расходимостью, примем ограничение: значения z должно находиться в пределах от -2 до 2 при каждой итерации. На рисунке фрактала ось X будет соответствовать действительной оси, а Y – мнимой.

В программе цвет точек задаётся по формуле:

```
img= sqrt (zRe.*zRe+zIm.*zIm)
```

Алгоритм построения множества Жюлиа (и фрактала Мандельброта) выглядит следующим образом:

1. Задание значения C
2. $z=0; n=0;$
3. $z=f(z);$
4. $n=n+1;$

Если не достигнуто максимальное значение итераций или значение z не превысило максимального значения, то следует повторять шаги 3 и 4.

Листинг программы, строящей множество Жюлиа представлен ниже.

```
Editor - Fractal_Julia.m
Fractal_Julia.m x +
1 - p=1000; % число пикселей
2 - s=4; %размер окна
3 - Img=zeros(p,p);
4 - for i=1:p
5 -     x0=0-s/2+i*s/p;
6 -     for j=1:p
7 -         y0=0-s/2+j*s/p;
8 -         x=x0;
9 -         y=y0;
10 -        z=0;
11 -        iter=1;
12 -        while iter<20
13 -            iter=iter+1;
14 -            x1=x^2-y^2-0.22;
15 -            y1=2*x*y-0.74;
16 -            x=x1;
17 -            y=y1;
18 -            z=x^2+y^2;
19 -            if z<4
20 -                Img(i,j)=sqrt(z);
21 -            else
22 -                iter=25;
23 -            end
24 -        end
25 -    end
26 - end
27 - imagesc(Img);
28 - colormap('jet');
29
```

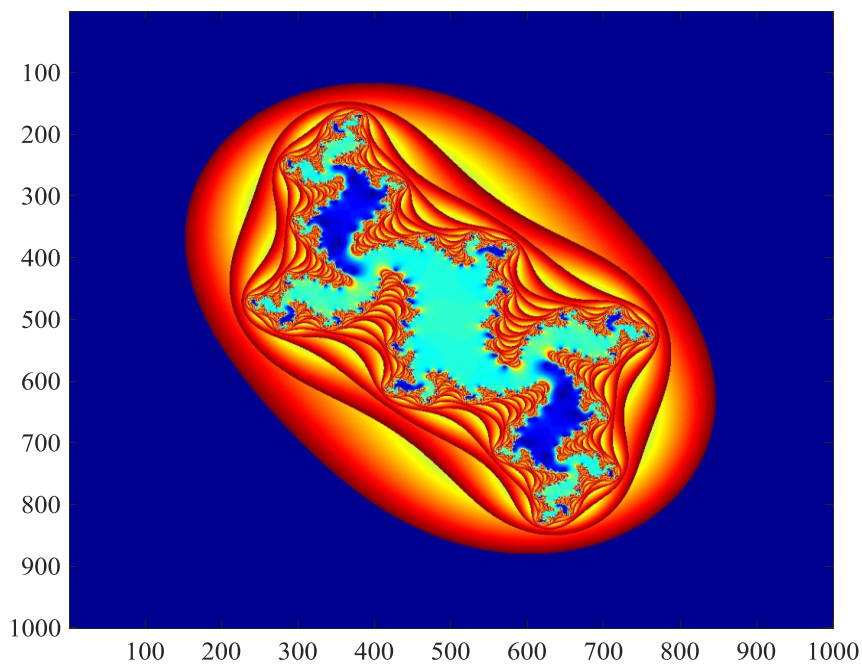


Рис. 4.3. Множество Жюлиа, построенное в ППП Matlab.

Контрольные вопросы

1. Дайте определение фрактала.
2. По каким признакам можно классифицировать фракталы ?
3. Приведите примеры каждого вида фракталов.
4. Перечислите основные этапы построения множества Жюлиа.
5. Назовите области применения фракталов.

4.3 Индивидуальные задания

Задание 1.

Вариант 1. Постройте алгебраический фрактал Монте-Карло, используя следующие преобразования:

$$T_1 : \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix},$$
$$T_2 : \begin{cases} x' = 1 + \frac{a(x-1)}{(x-1)^2 + y^2 + 1}, \\ x' = \frac{ay}{(x-1)^2 + y^2 + 1} \end{cases}. \text{ Параметр } a \text{ принять равным } 2.8.$$

Реализуйте стохастический вариант фрактала Монте-Карло.

Вариант 2. Постройте алгебраический фрактал типа дендрит, используя следующие преобразования:

$$T_1 : \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0,5 & 0,5 \\ 0,5 & -0,5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix},$$
$$T_2 : \begin{cases} x' = \frac{3x - y + 2}{5}, \\ x' = \frac{-x - 3y + 1}{5} \end{cases}.$$

Реализуйте стохастический вариант этого фрактала.

Вариант 3. Реализуйте алгебраический фрактал по следующей системе аффинных преобразований:

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_n \\ y_n \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}, \text{ принимая в качестве параметров следующие}$$

значения:

a	b	c	d	e	f	p
0	0	0	0.16	0	0	0.01
0.85	0.04	-0.04	0.85	0	1.6	0.85
0.2	-0.26	0.23	0.22	0	1.6	0.07
-0.15	0.28	0.26	0.24	0	0.44	0.07

Реализуйте стохастический вариант этого фрактала. В последнем столбце приведены вероятности p , в соответствии с которыми в методе случайных итераций выбирается то или иное преобразование.

Вариант 4. Реализуйте алгебраический фрактал по следующей системе аффинных преобразований:

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_n \\ y_n \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}, \text{ принимая в качестве параметров следующие}$$

значения:

a	b	c	d	e	f	p
0.1	0	0	0.16	0	0	0.01
0.85	0	0	0.85	0	1.6	0.85
-0.1667	-0.2887	0.2887	-0.1667	0	1.6	0.07
-0.1667	0.2887	-0.2887	-0.1667	0	1.6	0.07

Реализуйте стохастический вариант этого фрактала. В последнем столбце приведены вероятности p , в соответствии с которыми в методе случайных итераций выбирается то или иное преобразование.

Вариант 5. Реализуйте алгебраический фрактал, задаваемый двумя преобразованиями типа «поворот-сжатие»:

$$T_1 : \begin{cases} x' = ax - by \\ y' = bx + ay \end{cases}, \quad T_2 : \begin{cases} x' = cx - dy + 1 - c \\ y' = dx + cy - d \end{cases} \quad (T_1 - \text{поворот-сжатие относительно точки } (0,0) \text{ с показателем сжатия } \sqrt{a^2 + b^2}, T_2 - \text{поворот-сжатие относительно точки } (1,0) \text{ с показателем сжатия } \sqrt{c^2 + d^2}), \text{ приняв следующие значения параметров: } a = d = 0, b = c = 0.7. \text{ Реализуйте стохастический вариант этого фрактала.}$$

Вариант 6. Реализуйте алгебраический фрактал, задаваемый двумя преобразованиями типа «поворот-сжатие»:

$$T_1 : \begin{cases} x' = ax - by \\ y' = bx + ay \end{cases}, \quad T_2 : \begin{cases} x' = cx - dy + 1 - c \\ y' = dx + cy - d \end{cases} \quad (T_1 - \text{поворот-сжатие относительно точки } (0,0) \text{ с показателем сжатия } \sqrt{a^2 + b^2}, T_2 - \text{поворот-сжатие относительно точки } (1,0) \text{ с показателем сжатия } \sqrt{c^2 + d^2}), \text{ приняв следующие значения параметров: } a = b = 0.6, d = 0, c = 0.53. \text{ Реализуйте стохастический вариант этого фрактала.}$$

Вариант 7. Реализуйте алгебраический фрактал, задаваемый двумя преобразованиями типа «поворот-сжатие»:

$$T_1 : \begin{cases} x' = ax - by \\ y' = bx + ay \end{cases}, \quad T_2 : \begin{cases} x' = cx - dy + 1 - c \\ y' = dx + cy - d \end{cases} \quad (T_1 - \text{поворот-сжатие относительно точки } (0,0) \text{ с показателем сжатия } \sqrt{a^2 + b^2}, T_2 - \text{поворот-сжатие относительно точки } (1,0) \text{ с показателем сжатия } \sqrt{c^2 + d^2}), \text{ приняв следующие значения параметров: } a = 0, b = \frac{1}{\sqrt{2}}, c = 0.5, d = -0.5. \text{ Реализуйте стохастический вариант этого фрактала.}$$

Вариант 8. Реализуйте алгебраический фрактал, задаваемый двумя преобразованиями:

$$T_1 : \begin{cases} x' = ax - by \\ y' = bx + ay \end{cases}, \quad T_2 : \begin{cases} x' = ax + by + c \\ y' = bx - ay + d \end{cases}, \text{ приняв следующие значения параметров: } a = b = 0.4641, c = d = -0.1965.$$

метров: $a = b = 0.4641, c = d = -0.1965$.

Реализуйте стохастический вариант этого фрактала.

Вариант 9. Реализуйте алгебраический фрактал, задаваемый двумя преобразованиями:

$$T_1 : \begin{cases} x' = ax + by \\ y' = bx - ay \end{cases}, \quad T_2 : \begin{cases} x' = cx + dy \\ y' = cx - dy \end{cases}, \text{ приняв следующие значения параметров:}$$

ров: $a = 0.5, b = \frac{\sqrt{3}}{6}, c = \frac{2}{3}, d = 0$. Реализуйте стохастический вариант этого фрактала.

Вариант 10. Реализуйте алгебраический фрактал, задаваемый преобразованием:

$$T_1 : \begin{cases} x' = ax + by + e \\ y' = cx + dy + f \end{cases}, \text{ приняв следующие значения параметров:}$$

$a = d = 0.5, b = c = 0, e = 0.25, f = \frac{\sqrt{3}}{4}$. Реализуйте стохастический вариант этого фрактала.

Задание 2.

Вариант 1. Построить алгебраический фрактал из семейства фракталов Ньютона, задаваемого комплексным отображением:

$$z_{n+1} = \frac{(k-1)z_n^k + a}{k \cdot z_n^{k-1}}, \quad n=0, 1, 2, \dots$$

при $a = 1, k = 3$. Показать фрагмент границы фрактала, установить ее самоподобность. Пояснить основные шаги алгоритма.

Вариант 2. Построить алгебраический фрактал из семейства фракталов Ньютона, задаваемого комплексным отображением:

$$z_{n+1} = \frac{(k-1)z_n^k + a}{k \cdot z_n^{k-1}}, \quad n=0, 1, 2, \dots$$

при $a = 1, k = 4$. Показать фрагмент границы фрактала, установить ее самоподобность. Пояснить основные шаги алгоритма.

Вариант 3. Построить алгебраический фрактал из семейства фракталов Ньютона, задаваемого комплексным отображением:

$$z_{n+1} = \frac{(k-1)z_n^k + a}{k \cdot z_n^{k-1}}, n=0, 1, 2, \dots$$

при $a=1, k=5$. Показать фрагмент границы фрактала, установить ее самоподобность. Пояснить основные шаги алгоритма.

Вариант 4. Построить алгебраический фрактал «Пыль Фату», задаваемого комплексным отображением:

$$z_{n+1} = z_n^2 + c, \text{ при } c = 0.11037 - 0.67037i. \text{ Показать фрагмент границы}$$

фрактала, установить ее самоподобность. Пояснить основные шаги алгоритма.

Вариант 5. Построить алгебраический фрактал из семейства фракталов Жюлиа, задаваемого комплексным отображением:

$$z_{n+1} = z_n^2 + c, \text{ при } c = -0.3904 - 0.58679i. \text{ Показать фрагмент границы}$$

фрактала, установить ее самоподобность. Пояснить основные шаги алгоритма.

Вариант 6. Построить алгебраический фрактал «Дендрит», задаваемого комплексным отображением:

$$z_{n+1} = z_n^2 + c, \text{ при } c = i. \text{ Показать фрагмент границы фрактала, установить}$$

ее самоподобность. Пояснить основные шаги алгоритма.

Вариант 7. Построить алгебраический фрактал Жюлиа при $a=0.11, b=0.66$. Показать фрагмент границы фрактала, установить ее самоподобность. Пояснить основные шаги алгоритма. Чем отличается программная реализация построения фрактала Жюлиа от программной реализации фрактала Мандельброта?

Вариант 8. Построить алгебраический фрактал Жюлиа при $a=0, b=1$. Показать фрагмент границы фрактала, установить ее самоподобность. Пояснить основные шаги алгоритма. Чем отличается программная реализация построения фрактала Жюлиа от программной реализации фрактала Мандельброта?

Вариант 9. Построить алгебраический фрактал Жюлиа при $a=-0.75, b=0$. Показать фрагмент границы фрактала, установить ее самопо-

добнось. Пояснить основные шаги алгоритма. Чем отличается программная реализация построения фрактала Жюлиа от программной реализации фрактала Мандельброта?

Вариант 10. Построить алгебраический фрактал Мандельброта. Показать фрагмент границы фрактала, установить ее самоподобность. Пояснить основные шаги алгоритма. Чем отличается программная реализация построения фрактала Мандельброта от программной реализации фрактала Жюлиа?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Введение в математическое моделирование : учеб. пособие: рек. Мин. обр. РФ / В. Н. Ашихмин [и др.]; под ред. П. В. Трусова. – М. : Логос, 2007. – 439 с.
2. Ануфриев, И.Е. Matlab 7. / И.Е. Ануфриев, А.Б. Смирнов, Е.Н. Смирнова – СПб.: БХВ – Петербург, 2005. – 1104 с.
3. Божокин, С.В. Фракталы и мультифракталы / С.В. Божокин, Д.А. Паршин – Ижевск: НИЦ «Регулярная и хаотическая динамика», 2001. – 128 с.
4. Вентцель Е.С. Исследование операций: задачи, принципы, методология : учебное пособие / Е.С. Вентцель. – 5-е изд., стер. – М.: КНОРУС, 2013. – 192 с.
5. Дьяконов, В.П. MATLAB 6.5 SP1/7/7 SP1/7 SP2 + Simulink 5/6. Инструменты искусственного интеллекта и биоинформатики [Электронный ресурс]: монография / В.П. Дьяконов, В.В. Круглов. — Электрон. текстовые данные. – М. : СОЛОН-ПРЕСС, 2009. – 454 с. – 5-98003-255-X. – Режим доступа: <http://www.iprbookshop.ru/8683.html>.
6. Морозов, А.Д. Введение в теорию фракталов / А. Д. Морозов – Москва-Ижевск: Институт компьютерных исследований, 2002. – 160 с.
7. Пащенко, Ф.Ф. Введение в состоятельные методы моделирования систем: в 2 ч.: учеб. пособие: рек. УМО / Ф.Ф. Пащенко. – М. : Финансы и статистика, 2006 – 2007. Ч. 1: Математические основы моделирования систем. – 2006. – 328 с.
8. Петров, А.В. Моделирование процессов и систем. [Электронный ресурс] – Электрон. дан. – СПб.: Лань, 2015. – 288 с. – Режим доступа: <http://e.lanbook.com/book/68472>.
9. Самарский, А.А. Математическое моделирование: Идеи, методы, проблемы: моногр. / А.А. Самарский, А.П. Михайлов, – 2-е изд., испр. – М.: Москва: Физматлит, 2005. – 320 с.

СОДЕРЖАНИЕ

Введение	3
1 Моделирование стохастических процессов методом Монте-Карло	4
1.1 Базовые концепции метода Монте-Карло	4
1.2 Примеры решения модельных задач на основе метода Монте-Карло: реализация в ППП Matlab	9
1.3 Индивидуальные задания	12
2 Нечеткие модели	14
2.1 Основы теории нечеткого моделирования	14
2.2 Нечеткое моделирование с использованием инструментальных средств пакета Fuzzy Logic Toolbox Matlab: основные приемы работы	18
2.3 Примеры создания нечеткой экспертной системы в Fuzzy Logic Toolbox Matlab	28
2.4 Индивидуальные задания	38
3 Прогнозирование временных рядов с помощью гибридных нейро-нечетких сетей ANFIS	40
3.1 Назначение и применение гибридных нейронных сетей	40
3.2 Графический интерфейс приложения ANFIS для моделирования гибридных нейронных систем	44
3.3 Примеры построения гибридных нейро-нечетких систем	47
3.4 Индивидуальные задания	53
4 Модели фрактальных структур	55
4.1 Понятие фрактала. Классификация фракталов	55
4.2 Построение фракталов в ППП Matlab	58
4.3 Индивидуальные задания	64
Библиографический список	70

Составители

Анна Геннадьевна Масловская,

профессор кафедры математического анализа и моделирования АмГУ,

д-р физ.-мат. наук, доцент

Любовь Игоревна Мороз,

старший преподаватель

кафедры математического анализа и моделирования АмГУ

Математическое и компьютерное моделирование: специальные главы.

Практикум. Часть I. Моделирование в условиях неопределенности. Моделирование фрактальных структур.

Заказ 119.