Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
**«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)**

Институт инновационных технологий

Факультет Информационных технологий

Кафедра Информационных систем и программной инженерии

Р. И. МАКАРОВ

МОДЕЛИРОВАНИЕ

Курс лекций

по дисциплине «Моделирование» для магистрантов ВлГУ,

обучающихся по направлению 231000-Программная инженерия

Программа подготовки – “Разработка программно-информационных систем»

Владимир – 2013г.

Оглавление

Введение

Курс направлен сформировать представление магистрантов о моделировании как методе научного познания, ознакомить с использованием компьютера как средства познания и научно-исследовательской деятельности.

Целями освоения дисциплины являются ознакомление магистрантов с методами построения моделей сложных систем, с возможностями средств моделирования, оценкой качества моделей, проведением экспериментов для оценки эффективности сложных систем, применение моделей в задачах управления.

Процесс изучения дисциплины направлен на формирование и развитие основных

общекультурных компетенций:

– способность проявлять инициативу, в том числе в ситуациях риска, брать на себя всю полноту ответственности (ОК- 5);

– способность самостоятельно приобретать с помощью информационных технологий и использовать в практической деятельности новые знания и умения, в том числе в новых областях знаний, непосредственно не связанных со сферой деятельности (ОК- 6);

– способность к профессиональной эксплуатации современного оборудования и приборов (в соответствии с целями магистерской программы) (ОК- 7);

профессиональных компетенций:

–умение отбирать и разрабатывать методы исследования объектов профессиональной деятельности на основе общих тенденций развития программной инженерии (ПК-1);

–умение проводить анализ, синтез, оптимизацию решений с целью обеспечения качества объектов профессиональной деятельности (ПК-2);

–способность осуществлять сбор, анализ научно-технической информации, отечественного и зарубежного опыта по тематике исследования (ПК-7);

–уметь проводить разработку и исследования теоретических и экспериментальных моделей объектов профессиональной деятельности в различных областях (ПК-8);

–уметь проводить разработку и исследование методик анализа, синтеза, оптимизации и прогнозирования качества процессов функционирования информационных систем и технологий (ПК-9);

–умение осуществлять моделирование процессов и объектов на базе стандартных пакетов автоматизированного проектирования и исследований (ПК-10);

–способность проводить анализ результатов проведения экспериментов, осуществлять выбор оптимальных решений, подготавливать и составлять обзоры, отчеты и научные публикации (ПК-12);

Задачи дисциплины:

–повысить уровень компетенции магистрантов за счет вооружения соответствующими знаниями и практическими навыками использования методов моделирования сложных систем для решения специфических задач в области информационных систем и информационных технологий;

–ознакомить магистрантов с методами разработки математических и имитационных моделей;

–ознакомить магистрантов с технологией подготовки и проведения имитационного моделирования систем;

–изучить широкий круг вопросов, связанных с разработкой моделей предметных областей информационных систем.

В результате освоения дисциплины обучающийся должен демонстрировать следующие результаты образования:

1) Знать: основные математические схемы моделирования сложных систем; математический аппарат, описывающий взаимодействие информационных процессов и технологий на информационном, программном и техническом уровнях, знать технологию построения и использования моделей объектов.

2) Уметь: осуществлять математическую постановку исследуемых задач, уметь проводить оценку эффективности работы моделируемых объектов в области информационных систем и технологий;

3) Владеть математическим аппаратом для решения специфических задач в области информационных систем и технологий.

Общая трудоемкость дисциплины составляет 3 зачетных единицы, 108 часов.

Лекция 1 **Моделирование как метод познания**

1.1 Модель и ее свойства

Модели играют в жизни человека чрезвычайно важную роль — достаточно сказать, что в основе поведения человека, как системы разумной, лежит субъективная модель мира, создаваемая им на протяжении всей жизни на основе анализа личного и социального опыта.

Какими бывают модели? И какие средства формализации используются для представления знаний о системах?

Для начала обратимся к понятию модели и ее свойствам.

Модель — это совокупность логических, математических или иных объектов, связей и соотношений, отображающих с необходимой или предельно достижимой степенью подобия некоторый фрагмент реальности, подлежащий изучению, а также описание всех существенных свойств моделируемого объекта. Можно рассматривать различные аспекты подобия между моделью и фрагментов реального мира:

- физическое подобие, когда модель и объект имеют близкую физическую сущность;
- функциональное подобие, когда сходны их функции;
- динамическое подобие, проявляющееся в сходстве динамики изменения состояния объекта;
- топологическое подобие, проявляющееся в сходстве пространственной (в широком смысле, в том числе - организационной) структуры и иные.

Соответственно различают физические, функциональные, динамические, топологические и иные виды моделей. Кроме того, по принципу реализации выделяют натурные, полунатурные, имитационные и теоретические модели. В зависимости от обстоятельств (целей, условий) в практике используются разные модели.

Степень формализации моделей может варьироваться в широких пределах: от моделей, не подвергнутых процедурам формализации, до моделей строго формальных. Выбор формальных средств, используемых для представления моделей, не является произвольным и определяется двумя аспектами-компонентами модели [8]:

- *моделью интерпретации* или *интерфейсным компонентом* (характеризующим процесс двунаправленного взаимодействия с потребителем, в роли которого может выступать как человек, так и автоматизированная система, реализующая функции ввода и считывания данных);

- *сущностным компонентом* (характеризующим специфику моделируемого фрагмента реальности, закономерности его функционирования, структуры и т. п.).

Если взглянуть на любую модель с точки зрения, характерной для специалиста в области разработки программного обеспечения, знакомого с объектным подходом к программированию, то модель предстанет в виде совокупности инкапсулированных (помещенных одна в другую) моделей. При этом модель интерпретации (адаптации, интерфейса) представляет собой внешнюю оболочку модели, а сущностная модель фрагмента реальности (объекта, процесса явления и т. п.) заключена внутрь (см. рис. 1.1).

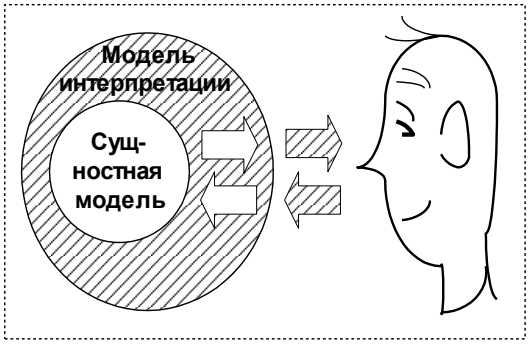


Рисунок 1.1 - Инкапсуляция моделей

В отличие от простых - одноуровневых моделей, сложные модели имеют несколько уровней вложенности, и на каждом уровне вложенности может существовать несколько разнородных моделей, однако, и для них изложенный выше подход остается справедливым (см. рис 1.2). Принцип матрешки широко используется при синтезе моделей самой различной семантики.

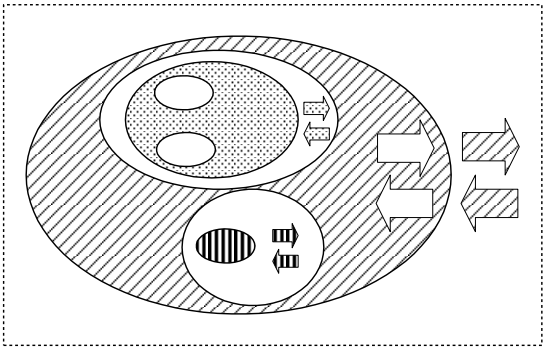


Рисунок 1.2 - Сложная модель, как иерархия модельных объектов.

Сущностная компонента модели является отражением некоторых сущностей, процессов и явлений реального мира и, в отличие от модели интерпретации, не может быть отображена с применением произвольно выбранных средств формализации предметной области. Для каждой предметной области существует некоторый диапазон приемлемых

средств формального выражения отношений и сущностей реального мира, отличающихся степенью детализации их выражения. Степень же детализации с одной стороны определяется спецификой задачи, а с другой - спецификой системы или процесса.

Перечислим наиболее значимые факторы, оказывающие влияние на выбор адекватной степени детализации модели:

* назначение модели и цель исследования (аналитическая, прогностическая модель и т. д., исследовательская (научная) модель, кибернетическая (управленческая) модель);

* избирательность исследования (выражению средствами модели подлежит система или процесс в целом или их отдельные аспекты);

* степенью полноты знаний о системе или процессах, подлежащих моделированию;

* динамические характеристики моделируемой системы/процесса;

* структура моделируемой системы;

* условия наблюдаемости (непрерывное, кусочно-непрерывное, дискретное);

* характеристика среды и параметры возмущающих воздействий;

* время, доступное для синтеза модели/производства вычислений;

* динамические и точностные характеристики системы сбора информации (точность результатов не может быть выше точности измерений);

* динамические и точностные характеристики системы управления (чаще всего, нет смысла анализировать динамические и статические параметры системы или процесса, если отсутствуют средства управления, обеспечивающие необходимую скорость и точность доведения управляющих воздействий)

* точностные характеристики методов, используемых для обработки данных;

* характеристики платформы, используемой для реализации модели (в случае применения специальных программных и технологических средств);

* точностные характеристики реализации методов, с учетом ограничений технологической платформы, используемой их реализации и иные.

Располагая знаниями высшего уровня (зная закон) исследователь менее всего стеснен в выборе средств моделирования. Однако в большинстве случаев такой свободы нет. Например, отсутствие достаточного объема знаний о системе не позволяет построить модель более высокой степени формализации, нежели вербальная или логико-лингвистическая модель типа сценария. Такая ситуация возникает тогда, когда причинно-следственные отношения не выявлены, структура системы и отношения между компонен-

тами установлены лишь частично и подлежат уточнению, что соответствует знаниям уровня гипотезы или теории в предложенной иерархии.

В то же время, даже располагая знанием закона, исследователь не всегда может выбрать произвольный способ формального представления системы, поскольку формальный аппарат, как правило, не универсален и привязан к конкретной предметной области и условиям наблюдений.

На практике ч*аще встречается ситуация, когда некоторая формальная система позволяет адекватно описывать феномены различного происхождения* — так обстоит дело со многими математическими формальными системами, полученными в результате развития естественнонаучных дисциплин. К числу формальных методов относятся аналитические, вероятностные и статистические, теоретико-множественные и логические, лингвистические и семиотические, а также графические и иные методы. Формальные модели, построенные с применением этих методов, получают названия, сходные с названиями использованных методов.

1.2 Моделирование, принципы моделируемости

Моделирование разработано с учетом принципа изоморфизма (многообразия): замены одного объекта на адекватную модель. Соотношение объекта и модели определяется степенью ее адекватного описания научными или иными средствами (вербально, графически, математически и т. п.) [9].

Моделирование является основополагающим методом исследования больших и сложных систем. В теории систем утверждается, что никаких других средств для качественного и эффективного описания больших и сложных систем кроме моделирования не существует. *В современной науке укоренилось представление, что «всякое познание является моделированием» (Н. Амосов).*

Принцип моделируемости заключается в том, что сложная система может быть представлена конечным множеством моделей, каждая из которых отражает определенную грань ее сущности.

Этот важный принцип дает возможность исследовать определенное свойство сложной системы при помощи одной или нескольких упрощенных (узкоориентированных) моделей. Модель, ориентированая на определенную группу свойств сложной системы, всегда проще самой системы. Создание полной модели для сложной системы всегда бесполезно, так как такая модель будет столь же сложной, как и система.

Принцип моделируемости включает несколько постулатов [6].

Постулат действий. Для изменения поведения системы требуется прирост воздействия, превосходящего некоторое пороговое значение.

Изменение поведения сложной системы может быть связано с энергетикой, с веществом и с информацией, которые, накапливаясь, проявляют свое влияние скачкообразно, путем качественного перехода. Одновременное энергетическое и информационное воздействие может привести к такому же результату, как энергетическое более высокого уровня.

Таким образом, порог есть функция трех переменных: количества определенного вещества, количества энергии и количества определенной информации.

Реакция системы на внешнее воздействие носит пороговый характер.

Постулат неопределенности. Существует область неопределенности, в пределах которой свойства системы могут быть описаны только вероятностными характеристиками. Повышение точности определения (измерения) какого-либо количественно описываемого свойства сложной системы сверх некоторого предела влечет за собой понижение возможной точности определения (измерения) другого свойства. Одновременно измерить значения двух (или более) параметров с точностью, превышающей определенный уровень, невозможно.

Максимальная точность определения (измерения) свойств системы зависит от присущей данной системе области неопределенности, внутри которой повышение точности определения (измерения) одного свойства влечет за собой снижение точности определения другого (других).

Постулат дополнительности. Сложные системы, находясь в различных средах (ситуациях), могут проявлять различные системные свойства, в том числе альтернативные (т.е. несовместимые ни в одной из ситуаций по отдельности). Наблюдатель воспринимает одни грани сущности в одних условиях и другие сущности в других.

Постулат многообразия моделей. Определение характеристик системы на всех уровнях производится с помощью множества моделей, которые в общем случае различаются используемыми математическими зависимостями и физическими закономерностями. Выбор моделей зависит от цели анализа и синтеза и особенностей исследуемой системы.

Постулат согласования уровней. Требования к системе, формируемые на любом уровне, выступают как условия (или ограничения) выбора частных моделей и предельных возможностей системы на нижележащих уровнях. В случае невозможности выполнения требований осуществляется корректировка условий.

Постулат внешнего дополнения. Проверка истинности результатов, получаемых на каждом уровне, производится с использованием исходных данных, моделей и методов вышележащих уровней. Данный постулат является основополагающим в общей теории систем, и его соблюдение является необходимым условием получения правильных решений на всех уровнях исследования системы.

Постулат достаточности. Последовательность уровней (этапов) определения требуемых характеристик в процессе совершенствования сложной системы выбирается по возрастанию затрат на улучшение системы, с проверкой достаточности принимаемых решений по заданным критериям эффективности. Постулат достаточности реализуется, как правило, при использовании критериев пригодности и разработке соответствующих моделей, с помощью которых принимаются конструктивные решения на каждом уровне выбора характеристик системы.

Постулат проверенного методического обеспечении. Для анализа и синтеза системы управления необходимо использовать хорошо отработанные и экспериментально проверенные модели и методики, обеспечивающие отдельные характеристики системы в заданные сроки и с требуемой точностью.

1.2. Основные методы формализации предметной области исследований. Поэтапный синтез моделей систем и процессов

Существует много методов формализации предметной области исследований. Рассмотрим некоторые из них и область применения в задачах моделирования.

Первичная вербальная модель представляет собой словесный портрет системы и проблемной ситуации, то есть представляет собой документ, аналогичный проекту технического (информационно-поискового и т.п.) задания, разрабатываемого организацией-заказчиком. Процесс синтеза первичной вербальной модели может осуществляться и при участии сторонних (приглашенных) специалистов. К этому шагу приходится прибегать в тех случаях, когда организация не располагает информацией, достаточной для принятия решения или выявления сущности противоречий.

Вербальная модель создается для сокращения неопределенности, компенсации неполноты знаний и формирования гипотезы или набора гипотез. Но первая и главная задача вербального моделирования — *создание вербального описания на материальном носителе. Вербальная модель - это не обязательно исключительно текстовый документ - она может содержать и количественные характеристики, элементы структуризации* (например, таблицы, схемы и графики) [8].

Важным этапом вербального моделирования является этап приведения (стандартизации) терминологии и сокращения избыточности описаний. Результатом выполнения этой процедуры является вербальная модель, построенная в едином стандартизованном тезаурусе, дальнейшее использование которой упрощает решение задач автоматизации процессов анализа и перевода модели на следующий уровень формального представления.

При решении задачи синтеза баз данных и систем информационного обеспечения деловых процессов, данных, полученных на этом этапе, зачастую оказывается достаточно для синтеза макета информационной системы.

Логико-лингвистические и семиотические модели характеризуется более высокой степенью формализации. При построении логико-лингвистических моделей широко используется символьный язык логики и формализм теории графов и алгоритмов. Наиболее распространенным способом формального представления логико-лингвистических моделей является граф. В математической интерпретации граф представляет собой формальную систему, описываемую, как G=(X,U), где X — множество вершин, U — множество ребер (дуг). Граф состоит из упорядоченных пар вершин, причем одна и та же пара может входить в множество U любое число раз, описывая различные виды отношений (рис. 1.3).

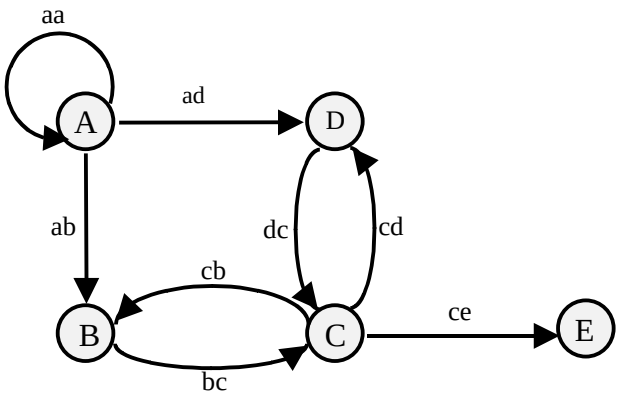


Рисунок 1.3 - Пример графа переходов.

Логико-смысловые модели позволяют формировать тематически связные описания различных аспектов проблемы (равно, как и проблемы в целом) и проводить структурный анализ проблемной области. В качестве примера применения логико-смыслового моделирования можно рассматривать гипертекстовые системы, получившие широкое распространение в глобальной телекоммуникационной сети Интернет.

Логико-смысловая модель представляется в виде связного неориентированного графа, в котором вершины соответствуют высказываниям, а ребра - семантическим связям между ними. Характеристики графа используются для исследования логико-смысловой сети.

Широкое применение логико-лингвистические модели нашли в отрасли разработки программного обеспечения, управления корпоративными информационными ресурсами и многих других отраслях, где требуется определенный уровень формализации, представляющий единство строгости, интуитивной понятности и высокой выразительной способности моделей [8].

Логические модели представляют собой следующий уровень формального представления (по сравнению с логико-лингвистическими). В таких моделях естественно-языковые высказывания замещаются на примитивные высказывания - литералы, между которыми устанавливаются отношения, предписываемые формальной логикой.

Различают логические модели, в которых рассматриваются различные схемы логических отношений: отношения логического следования, включения и иные, которыми замещаются отношения, характерные для традиционной формальной логики.

Наиболее широкое распространение логические модели получили в области построения систем искусственного интеллекта, где они используются в качестве основы для производства логического вывода из системы посылок, зафиксированных в базе знаний, в ответ на внешний запрос. Неполнота и нечеткость экспертных знаний сделали популярными такие направления логики, как многозначные логики, вероятностные логики и нечеткие логики (*Fuzzy Logic* — автор теории Л. Заде — 1960-е годы) [8].

Л. Заде предложил для представления таких знаний математическую теорию нечетких множеств и определил операции нечеткой логики.

Системы, использующие модели на базе нечеткой логики разрабатываются специально для решения плохо определенных задач и задач с использованием неполной и недостоверной информации. Внедрение аппарата нечетких логик в технологии создания экспертных систем привело к созданию нечетких экспертных систем (*Fuzzy Expert Systems*).

Нечеткая логика позволяет решать широкий класс задач, не поддающихся строгой формализации — методы нечеткой логики используются в системах управления сложными техническими комплексами, функционирующими в непредсказуемых условиях (летательными аппаратами, системами наведения высокоточного оружия и т. д.).

Статистические и теоретико-вероятностные методы составляют методологическую основу одноименного вида моделирования. *Статистическая или теоретико-вероятностная модель (стохастическая модель) — это модель, в которой обеспечивается учет влияния случайных факторов в процессе функционирования системы, основанная на применении статистической или теоретико-вероятностной методологии по отношению к повторяющимся феноменам*. Данная модель оперирует количественными крите-

риями при оценке повторяющихся явлений и позволяет учитывать их нелинейность, динамику, случайные возмущения за счет выдвижения на основе анализа результатов наблюдений гипотез о характере распределения некоторых случайных величин, сказывающихся на поведении системы.

По существу, теоретико-вероятностные и статистические модели отличаются уровнем неопределенности знаний о моделируемой системе, существующей на момент синтеза модели. В случае, когда представления о системе носят, скорее, теоретический характер и основываются исключительно на гипотезах о характере системы и возмущающих воздействий, не подкрепленных результатами наблюдений, теоретико-вероятностная модель является единственно возможной. Когда же на этапе синтеза модели уже существуют данные, полученные опытным путем, появляется возможность подкрепления гипотез за счет их статистической обработки.

Статистические модели применимы для изучения массовых явлений любой природы, включая и те, которые не относятся к категории вероятностно определенных (математическая статистика приспособлена и для решения детерминированных задач). При моделировании последних статистический процесс вводится в модель искусственно для получения статистических оценок численного решения (например, точности измерения параметров детерминированного процесса). Для обработки результатов наблюдений используются методы корреляционного, регрессионного, факторного, кластерного и иных видов анализа, оперирующих статистическими гипотезами. Особая роль здесь отводится *методу статистических испытаний* (*методу Монте-Карло*).

Аналитическое математическое моделирование — это вид моделирования, в ходе которого основная роль отводится аналитической математической модели. Важным достоинством аналитического моделирования является возможность получения на его основе фундаментальных результатов и инвариантных зависимостей, которые могут быть распространены как на различные случаи использования моделируемой системы в тех или иных ситуациях и распространены на случаи рассмотрения других систем данного класса.

Основным же недостатком аналитического моделирования является то, что его применение к сложным системам требует существенной идеализации описания системы. Это связано с разрастанием объемов вычислений даже при несущественном усложнении описаний. Такая идеализация может приводить к неполной адекватности получаемых результатов, к тому, что эти результаты могут использоваться лишь в качестве первого приближения.

Имитационная модель — это комплексное логико-математическое представление системы, реализованное в виде программы, предназначенной для решения на ЭВМ, включающее в себя модели различного типа, и рассматривающее аспект функционирования динамической системы во времени. Данный класс моделей применяется при невозможности строгого аналитического решения задачи или проведения натурного эксперимента. Имитационные модели служат для изучения поведения во времени сложной неоднородной динамической системы, относительно структуры которой существуют точные знания или детализированные гипотезы. Для каждого элемента или подсистемы моделируемой системы в памяти ЭВМ формируется блок данных, характеризующих ее текущее и предшествующие состояния, блок логических и вычислительных процедур, описывающих изменения критических параметров во времени, а также производятся вычисления этих параметров на основе заданных значений.

Комплекс подпрограмм или относительно автономных программных агентов функционирует под управлением программы-супервизора, осуществляющей диспетчеризацию вызовов, активизирующей и приостанавливающей на время выполнение тех или иных процедур в соответствии с планом машинного эксперимента, имитируя тем самым поведение системы. В результате машинного эксперимента формируются массивы данных о состоянии различных параметров системы в различные моменты времени с привязкой к системным событиям, имитируемым в ходе эксперимента.

При этом программа-супервизор управляет процессом имитации случайных возмущающих воздействий, от которых зависит функционирование системы в целом и ее элементов и подсистем.

Частным случаем имитационных моделей являются модели ситуационные. *Ситуационные модели - это модели, используемые при решении задач с неопределенностью, исходя из совокупности ситуаций*. В отличие от других моделей, основанных на заданном графе функционирования системы, для ситуационной модели такой граф неизвестен. Однако существует набор прецедентов ситуаций, обладающих малым прогностическим потенциалом.

В целом структура ситуационной модели определяется субъективными особенностями восприятия и свойственным аналитику способом разложения ситуации на составляющие. Это вызвано тем, что эксперт-аналитик, осуществляющий процедуру синтеза ситуационной модели, формулирует свои собственные критерии, соответствующие пребыванию системы в том или ином состоянии [8**]**.

Исследование альтернативных стратегий, как правило, проводится на моделях (увы, не всегда это возможно, да и не всякий руководитель в состоянии оценить преимущества моделирования перед непосредственным действием). Как правило, для решения задач многокритериального оценивания требуется использовать несколько разнородных моделей, отражающих различные аспекты поведения системы и ее элементов. Кроме того, здесь приходится сталкиваемся с проблемой изоляции процессов: с одной стороны - модель уже должна существовать (иначе невозможен синтез критериев), с другой - модель необходимо синтезировать. Но есть одно обстоятельство: в одном случае речь идет о модели системы и ситуации в целом, а в другом о характере ее изменения в ходе реализации альтернативной стратегии (по существу, модель должна быть кибернетической - то есть, учитывать свойства системы с точки зрения анализа управленческих стратегий). На этом этапе оценивается эффективность реализации некоторой альтернативы и производится выбор оптимальной (или близкой к оптимальной) альтернативы из множества допустимых.

На этапе собственно моделирования *модель используется* не в качестве объекта синтеза и анализа, а *как инструмент исследования*. То есть, модели полагаются адекватными и предполагается, что дальнейшие итерации по совершенствованию моделей нецелесообразны. Модели используются в качестве систем, замещающих заданные фрагменты реальности - на них проводятся вычислительные и логические операции, выражающие выявленные на предшествующих этапах отношения и зависимости, определяются значения критериев выбора, обеспечивающие возможность сопоставления альтернативных стратегий. Речь идет о вариации исходных параметров и логики, отображающей стратегию управления. В результате чего формируется блок исходных данных, включающих значения и оценки критериев выбора, рисков и т.п. данных, используемых на заключительном этапе.

На заключительном этапе исследований формулируются выводы из результатов моделирования и указания по реализации его результатов. На этом этапе знания, полученные в результате проведения всего цикла процедур системно-кибернетического исследования, должны приобрести точное и наглядное выражение. *Лицу, принимающему решение (осуществляющему выбор альтернативы), должны быть предоставлены аргументированные выводы и рекомендации в той форме и тех терминах, в которых он способен их воспринять*.

Тестовые задания

1. Понятие модели и ее свойства. Виды подобия между моделью и фрагментами реального мира.

2. Перечислите наиболее значимые факторы, оказывающие влияние на выбор адекватной степени детализации модели.

3. Принцип моделируемости сложных систем, постулаты моделируемости.

4. Методы формализации предметной области исследований.

5. Этапы построения моделей, собственно моделирование и вывод результатов исследований.

Лекция 2 **Моделирование систем и сетей массового обслуживания**

Достаточно часто при анализе вычислительных систем приходится решать т.н. задачи массового обслуживания, возникающие в следующей ситуации. [11] Пусть анализируется вычислительная система (ВС) обслуживания заявок, состоящая из некоторого количества ЭВМ. В ВС могут возникать, по крайней мере, две типичных ситуации:

- число заявок слишком велико для данной вычислительной мощности ВС, возникают очереди и за задержки в обслуживании приходится платить;

- в ВС поступает слишком мало заявок и теперь уже приходится учитывать потери, вызванные простоем ЭВМ ВС.

Ясно, что цель анализа в данном случае заключается в определении некоторого соотношения между потерями доходов по причине *очередей* и потерями по причине *простоя* ЭВМ в ВС, такого соотношения, при котором математическое ожидание суммарных потерь окажется минимальным.

Так вот, специальный раздел теории систем — *теория массового обслуживания,* позволяет:

- использовать методику определения средней длины очереди и среднего времени ожидания заявок в тех случаях, когда скорость поступления заявок и время их выполнения заданы;

- найти оптимальное соотношение между издержками по причине ожидания в очереди и издержками простоя ЭВМ в ВС;

- установить оптимальные стратегии обслуживания.

Обратим внимание на главную особенность такого подхода к задаче системного анализа - явную зависимость результатов анализа и получаемых рекомендаций от двух внешних факторов: частоты поступления и сложности заявок (а значит - времени их обслуживания).

Но это уже *связи* исследуемой системы с *внешним миром* и без учета этого факта не обойтись. Потребуется провести исследования потоков заявок по их численности и сложности, найти статистические показатели этих величин, выдвинуть и оценить достовер-

ность гипотез о законах их распределения. Лишь после этого можно анализировать как будет вести себя система при таких внешних воздействиях, как будут меняться ее показатели (значение суммарных издержек) при разных управляющих воздействиях или стратегиях управления.

Очень редко при этом используется сама система для проведения натурального эксперимент над ней. Чаще всего такой эксперимент связан с риском потерь заказчиков или неоправданными затратами на приобретение дополнительных ЭВМ для вычислительной системы, поэтому эксперимент ставится на модели.

2.1 Элементы теории массового обслуживания

Для современных вычислительных машин и систем характерна работа в режиме решения потока случайных по своим характеристикам задач, поступающих в общем случае в случайные моменты времени. Анализ и, самое главное, синтез подобных систем с учётом вероятностного фактора протекающих в них процессов возможны с использованием методов теории массового обслуживания.

Предмет теории массового обслуживания – системы и сети массового обслуживания. Под системой массового обслуживания (СМО) понимают динамическую систему, предназначенную для эффективного обслуживания случайного потока заявок (требований на обслуживание) при ограничениях на ресурсы системы. Обобщённая структурная схема приведена на рисунке 2.1.

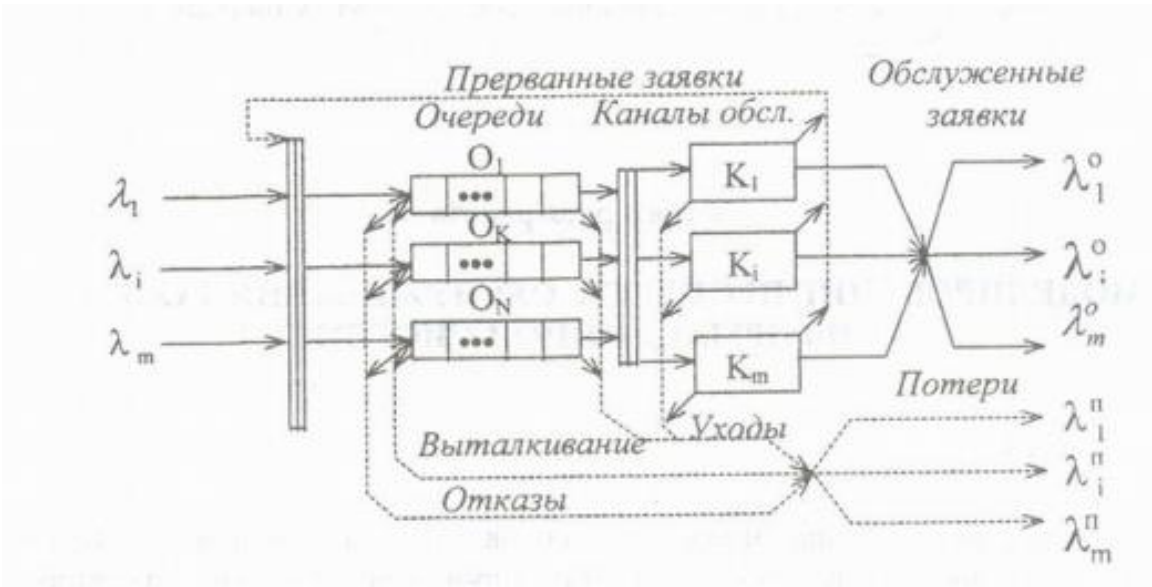


Рисунок 2.1 - Обобщённая схема системы массового обслуживания

Поступающие на вход СМО однородные (то есть требующие однородного обслуживания) заявки в зависимости от порождающей их причины делятся на типа, интенсивность потока заявок типа i ($i = \overline{1,M}$) обозначается как λi. Первопричина заявок, какова бы ни была её физическая природа, называется *источником заявок*, совокупность заявок всех типов – входящим потоком СМО.

Обслуживание заявок выполняется совокупностью m в общем случае разнотипных каналов. В произвольный момент времени канал может быть занят обслуживанием только одной заявки, в общем случае допускается прерывание начатого некоторым каналом процесса обслуживания.

Если в момент появления заявки на входе СМО хотя бы один канал свободен, её обслуживание может быть начато немедленно, без задержки. Однако, вполне вероятна ситуация, когда заявка застаёт СМО полностью загруженной, то есть когда все m каналов заняты обслуживанием. В этом случае начало обслуживания задерживается, заявка должна занять место в соответствующей очереди. Очередь может быть либо общей, либо раздельной; деление очереди выполняется обычно по приоритетному принципу. На число мест в очереди может быть наложено ограничение, это может быть сделано как для каждой очереди в отдельности, так и для всей совокупности очередей в целом. При этом возможны конфликтные ситуации, решением которых может быть либо отказ системы принять заявку, либо принятие заявки за счёт возможно, выталкивания из очереди другой, менее ценной заявки для системы в данный момент времени.

В зависимости от числа мест в очереди различают СМО с отказами и без отказов. В СМО с отказами число мест в очереди конечно.

В зависимости от допустимого времени пребывания заявки в системе различают СМО с «нетерпеливыми» и «терпеливыми» заявками. В СМО с «нетерпеливыми» заявками заявка может «уйти» из системы, если время пребывания её в СМО превысит некоторое допустимое значение, которое в общем случае может быть случайным или характеризоваться некоторым распределением. «Терпеливая» заявка, попав в СМО, непременно дождётся обслуживания.

Процесс продвижения заявки от входа к выходу СМО происходит в соответствии с некоторым законом управления процессами в СМО, который задаётся дисциплинами ожидания и обслуживания. Дисциплина ожидания определяет порядок приёма заявок в систему и размещения их в очереди, дисциплина обслуживания – порядок выбора заявок из очереди и назначения их на обслуживание.

Совокупность обслуженных и потерянных (полностью необслуженных либо недообслуженных) заявок образует выходящий поток СМО. В зависимости от структуры выходящего потока различают СМО без потерь («чистые» СМО) и СМО с потерями («смешанные» СМО).

2.2 Параметры и характеристики систем массового обслуживания

Параметры входящего потока. Процесс поступления заявок на обслуживание является в общем случае случайным и может рассматриваться как поток однородных событий, происходящих через случайные промежутки времени. Случайные временные интервалы между поступлениями заявок могут подчиняться различным законам распределения.

Наибольшее распространение в теории массового обслуживания получил простейший поток заявок, то есть поток, обладающий свойствами стационарности, ординарности и отсутствия последействия.

Если входящий поток представляет собой совокупность M потоков различных типов с интенсивностями λ_i, $i = \overline{1, M}$, то его можно характеризовать суммарной интенсивностью:

$$\lambda = \sum_{i=1}^{M} \lambda_i$$

К параметрам входящего потока причислим также допустимое время пребывания $\tau_{доп}$ заявки в системе массового обслуживания, рассматривая его как свойство заявки. По истечении времени пребывания попавшая в систему заявка покидает её, не дожидаясь назначения на обслуживание или даже прерывая начатое обслуживание.

Параметры структуры СМО. Каждая система массового обслуживания обладает определённой структурой, характеризующейся совокупностью параметров. По составу СМО можно разделить на СМО с одним каналом обслуживания (одноканальные СМО) и с *m* каналами обслуживания (многоканальные СМО). В свою очередь, многоканальные

СМО могут содержать одинаковые и различные по своей производительности каналы обслуживания.

Производительность канала обслуживания обратно пропорциональна длительности обслуживания. Эта случайная величина с функцией распределения $F(\tau)$, плотностью распределения $f(\tau)$ и математическим ожиданием $\tau_{об}$. Типы заявок различаются законами распределения. При этом принимается допущение о независимости длительности обслуживания для различных заявок одного типа, вполне корректное для большинства систем. Наряду с математическим ожиданием длительности обслуживания используется понятие интенсивности потока обслуживания – величины, характеризующей количество заявок, которое может быть обслужено в единицу времени постоянно загруженным каналом.

Важная компонента структуры СМО – очередь, параметром которой является число мест n. В приоритетных системах общая очередь может быть разделена на несколько очередей по числу различаемых приоритетов, для каждой из которых должно быть указано число мест n_i, $i = \overline{1, N}$. Ограничение на число мест может задаваться на всю совокупность очередей. При неограниченном числе мест в очереди (бесконечная очередь) отсутствуют потери заявок за счёт отказов и выталкиваний.

Закон управления процессами в СМО можно представить совокупностью двух дисциплин: дисциплины ожидания и дисциплины обслуживания. В бесприоритетных дисциплинах заявки какого-либо типа не имеют преимуществ перед заявками других типов при постановке в очередь или назначении на обслуживание. Если по каким-либо причинам заявки некоторых типов должны обслуживаться СМО быстрее, то они получают преимущество перед заявками других типов, называемое *приоритетом*. Приоритеты заявок характеризуются положительными числами 1, 2, 3, …, причём более высокому приоритету соответствует меньшее число. Дисциплины ожидания и обслуживания, учитывающие приоритеты, называются приоритетными.

Дисциплина ожидания определяет правила управления очередью, возникающей в том случае, когда каналы обслуживания не справляются с потоком заявок.

Дисциплина обслуживания определяет правила выбора заявки из очереди при назначении на обслуживание. При бесприоритетной дисциплине обслуживания заявки различных типов не имеют привилегий перед заявками других типов на досрочное обслуживание. В приоритетных дисциплинах обслуживания заявкам некоторых типов предо-

ставляется преимущественное право на обслуживание перед заявками других типов, называемое *приоритетом*. Различают относительные, абсолютные и смешанные приоритеты.

Относительные приоритеты учитываются только в момент назначения заявки на обслуживание. При освобождении канала обслуживания сравниваются приоритеты заявок, находящихся в очереди в состоянии ожидания, и обслуживание предоставляется заявке с наибольшим приоритетом.

Абсолютные приоритеты предполагают прерывание обслуживания низкоприоритетной заявки в момент поступления в СМО заявки с более высоким приоритетом, прерванная заявка ставится в начало либо общей очереди, либо очереди заявок соответствующего приоритета. Обслуживание прерванных заявок может производиться либо от начала (повторное обслуживание), либо от момента прерывания (дообслуживание).

Смешанные приоритеты предполагают сочетание рассмотренных видов приоритетов, причём для отдельных заявок может быть использовано бесприоритетное обслуживание.

Характеристики СМО могут быть разделены на две группы. Характеристики первой группы используются при рассмотрении СМО как элемента более сложной структуры, например сети массового обслуживания. Характеристики второй группы позволяют оценить способность конкретной СМО к выполнению возложенных на неё функций и называются *показателями эффективности.*

Важными характеристиками первой группы являются *характеристики выходящего потока заявок.* Выходящий поток заявок в общем случае распадается на поток обслуженных и поток потерянных заявок, каждый из которых характеризуется законом распределения длительности между соседними заявками. Если входящий поток содержит заявки M типов с интенсивностями λi потока заявок типов i, $i = \overline{1,M}$, то выходящий поток можно характеризовать суммарной интенсивностью потока обслуженных заявок и суммарной интенсивностью потока потерянных заявок соответственно:

$$\lambda_{об} = \sum_{i=1}^{M} \lambda_{об}\,,\ \lambda_{\text{ni}} = \sum_{i=1}^{M} \lambda_{ni}\,,$$

где $\lambda_{об}$ – интенсивность потока обслуженных заявок типа i; λ_{ni} – интенсивность потока потерянных заявок типа i. Очевидно, что $\lambda_{об} + \lambda_{ni} = \lambda$.

Под показателем эффективности понимается количественный показатель, частично характеризующий уровень выполнения системой массового обслуживания возложенных на неё функций. На основании показателей эффективности можно построить некоторый критерий эффективности, совокупно характеризующий эффективность СМО при ограничениях на её параметры. Эффективность СМО можно характеризовать большим числом различных показателей эффективности. Рассмотрим наиболее употребительные из них.

Вероятность обслуживания $p_{об}$ – это вероятность того, что произвольно выбранная из входящего потока с интенсивностью λ заявка будет обслужена, то есть окажется в потоке обслуженных заявок с интенсивностью $\lambda_{об}$:

$$p_{об} = \frac{\lambda_{об}}{\lambda}.$$

Иногда вероятность обслуживания называют относительной пропускной способностью.

Вероятность потери p_n – это вероятность того, что произвольно выбранная из входящего потока с интенсивностью λ заявка окажется в потоке потерянных заявок с интенсивностью λ_n:

$$p_n = \frac{\lambda_n}{\lambda} = \frac{\lambda - \lambda_{об}}{\lambda} = 1 - \frac{\lambda_{об}}{\lambda} = 1 - \mathrm{p}_{об}.$$

Среднее время ожидания $\overline{\mathrm{t}_{\mathrm{ож}}}$ заявки (среднее время пребывания заявки в очереди) – математическое ожидание времени ожидания; время ожидания $t_{ож}$ заявки – случайная величина, равная сумме длительностей интервалов времени, в течение которых заявка находится в очереди, начиная с момента появления заявки на входе СМО и кончая моментом, когда она последний раз покидает очередь по причине назначения на обслуживание, ухода из очереди (в случае нетерпеливых заявок) либо выталкивания низкоприоритетной заявки высокоприоритетной заявкой для некоторых приоритетных дисциплин ожидания.

Среднее время пребывания заявки в СМО $\overline{t_c}$ – математическое ожидание времени пребывания заявки в СМО, равное промежутку времени от момента поступления завки на вход СМО до момента появления её в выходном потоке и связанное с длительностью процессов ожидания *тож* и обслуживания *тоб*. Среднее время пребывания заявки в СМО равно сумме среднего времени ожидания (пребывания в очереди) и среднего времени обслуживания (пребывания в канале обслуживания):

$$\overline{t_c} = \overline{t_{ож}} - \overline{t_{об}}.$$

Критерием эффективности СМО является некоторая функция показателей эффективности, которая служит для совокупной оценки приспособленности СМО к выполнению возложенных на неё функций. Выбор того или иного критерия эффективности зависит от назначения СМО, условий её функционирования и так далее. Рассмотрим примеры критериев эффективности, применимых к вычислительным системам, работающим в режиме реального времени.

Пусть заявка обесценивается пропорционально её задержке в СМО, то есть время пребывания заявки в СМО (иногда учитывают лишь время ожидания в очереди). Тогда эффективность *E* СМО равна:

$$E = \sum_{i=1}^{M} e_{ci} \overline{t_{ci}},$$

где e_{ci} – штраф за единицу времени пребывания заявки *i*-го типа в СМО; $\overline{t_{ci}}$ – среднее время пребывания в СМО заявок.

Штрафу может подвергаться потеря заявки системой, иногда отдельно штрафуют потери за счёт отказов, выталкиваний и уходов. Возможен функционал вида

$$E' = \sum_{i=1}^{M} \left(e_{отк i} p_{откi} \lambda_i + e_{вi} p_{вi} \lambda + e_{уi} p_{уi} \lambda_i \right),$$

где $e_{откi}$ – штраф за отказ СМО принять заявку *i*-го типа; $e_{вi}$ – штраф за выталкивание из очереди заявки; $e_{уi}$ – штраф за уход из СМО нетерпеливой заявки; $p_{откi}$, $p_{вi}$, $p_{уi}$ – вероятно-

сти соответственно отказа, выталкивания и ухода заявки *i*-го типа; λ_i – интенсивность входящего потока заявок *i*-го типа.

Критерии эффективности рассмотренных типов могут, в свою очередь, объединяться в новые, более сложные критерии эффективности. Как характеристики, так и критерии эффективности СМО могут рассматриваться в одном из двух режимов работы СМО. Первый из режимов, называемый *переходным*, соответствует начальному этапу функционирования СМО. Начальными условиями для него являются отсутствие очереди и свободные от обслуживания каналы обслуживания. Режим, следующий за переходным, называется *установившимся*(следует отметить, что установившийся режим в СМО возможен не всегда); поведение системы в этом режиме характеризуется обычно средними значениями характеристик, отражающими устойчивые свойства СМО.

2.3 Моделирование вычислительных процессов и алгоритмов обслуживания вычислительных задач

Процесс обработки данных в информационной технологии преследует определенную цель – решение с помощью ЭВМ вычислительных задач, отображающих функциональные задачи той системы, в которой ведется управление. Для реализации этой цели должны существовать модели обработки данных, соответствующие алгоритмы управления и воплощенные в машинных программах [10].

Процесс обработки данных может быть разбит на ряд связанных между собой процедур: организация вычислительного процесса, преобразование данных и отображение данных.

Организация вычислительного процесса (ОВП)

Содержание процедуры процесса обработки данных представляет его концептуальный уровень, модели и методы, формализующие процессы обработки данных в ЭВМ – логический уровень, а средства аппаратной реализации процедур – физический уровень процесса.

Процедура ОВП имеет различную функциональную сложность в зависимости от класса и количества решаемых задач, режимов обработки данных, топологии системы об-

работки данных. При обработке данных на ЭВМ различают три основных режима: пакетный, разделения времени, реального времени.

При пакетном режиме обработки задания программы с соответствующими исходными данными накапливаются на дисковой памяти ЭВМ, образуя «пакет». Обработка заданий осуществляется в виде непрерывного потока. Такой режим позволяет максимально загрузить ЭВМ, но дает задержки в получении решения из-за того, что некоторое время задание простаивает в очереди.

Режим разделения времени реализуется путем выделения для выполнения заданий определенных интервалов времени (квантов). Предназначенные для обработки задания находятся в ОП ЭВМ одновременно. В режиме разделения времени возможна реализация диалоговых операций.

Режим реального времени используется при обработке данных в ЭВМ, предназначенных для управления физическими процессами. В таких системах ЭВМ должна обладать высокой скоростью реакции, чтобы успеть за короткий интервал времени обработать поступившие данные и использовать результаты для управления процессами. В режиме разделения времени используется вариант мультипрограммного режима.

Задания в виде программ и данных подвергаются процессу обработки, поступая из системы ввода, системы хранения и по каналам вычислительной сети. В этих условиях остро ставиться вопрос планирования и выполнения заданий в вычислительной системе.

Организация обслуживания вычислительных задач

При организации и планировании процесса обработки данных в ВС возможны различные методы организации и обслуживания очередей заданий. При этом преследуется цель получения лучших значений таких показателей, как производительность, загруженность ресурсов, малое время простоя, высокая пропускная способность, разумное время ожидания в очереди заданий.

При организации обслуживания вычислительных задач на логическом уровне создается *модель задачи обслуживания*, которая может иметь как прямой, так и оптимизационный характер. При постановке прямой задачи данными являются параметры ВС, а решением – показатели эффективности организации вычислительных процессов (ОВП).

При постановке оптимизационной задачи задаются требуемые показатели эффективности ОВП и требуется определить параметры ВС.

В ВС моменты появления заданий являются случайными и случайным является момент окончания вычислительной обработки. Поэтому при моделировании пользуются статистическими данными о среднем количестве поступающих заявок в единицу времени на обработку в ВС, а также о среднем времени решения одной задачи. Эти данные позволяют рассматривать процедуру организации ВП с помощью теории систем массового обслуживания [10]. ОВП можно представить схемой, приведенной на рис.2.2.

Такая схема может быть охарактеризована как система с дискретными состояниями и непрерывным временем. Под дискретным состоянием понимается то, что в любой момент времени система может находиться только в одном состоянии. Число состояний ограничено. Под непрерывным временем подразумевают, что границы перехода из одного состояния в другое не фиксированы. Состояние системы характеризуется числом заданий в очереди плюс число заданий, обрабатываемые ЭВМ. Очередь уменьшается, когда ЭВМ

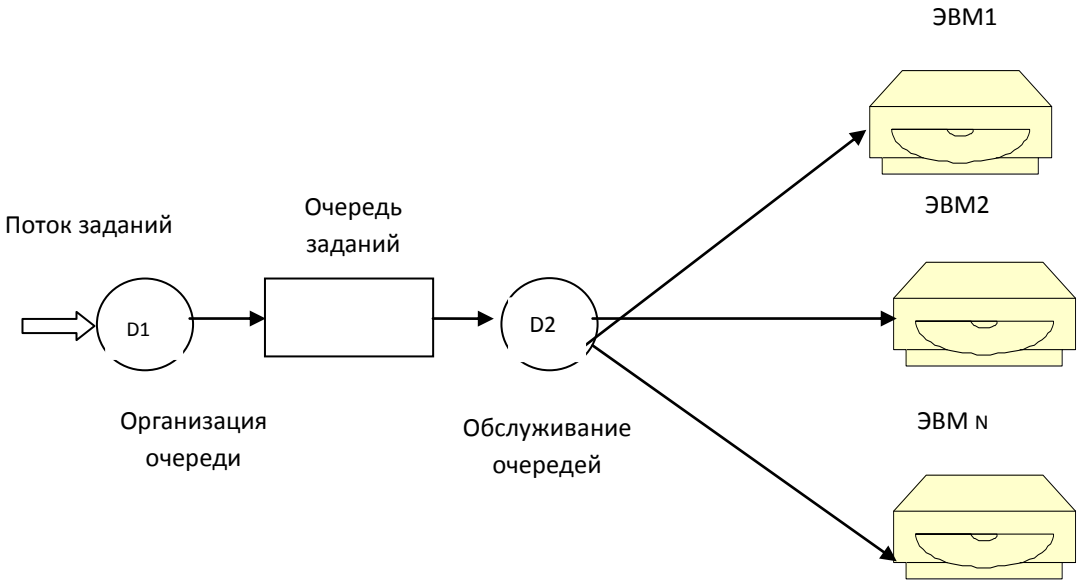


Рисунок 2.1 - Организация обслуживания заданий в многомашинной ВС

заканчивает обработку задания. Число заданий в системе растет благодаря потоку заданий. Поток заданий характеризуется интенсивностью λ – средним количеством заданий, поступающим в единицу времени. Среднее время обслуживания одного задания ЭВМ определяет интенсивность потока обслуживания μ:

$$\mu = 1/t_{обсл},$$

где $t_{обсл}$ – среднее время обработки одного задания.

Рассмотрим модель обслуживания вычислительных заданий (рис.2.2) введя следующие предположения:

- в системе протекают Марковские случайные процессы;

- потоки событий (появление заданий, окончание их обработки) являются простейшими;

- число заданий в очереди не ограничено, но конечно.

Случайный процесс, протекающий в системе, называется Марковским. Простейший поток событий характеризуется стационарностью (независимость параметров во времени), ординарностью (события в потоке появляются поодиночке) и «безпоследствием» (появляющиеся события не зависят друг от друга).

Обозначим состояния рассматриваемой системы:

S_0 – в системе нет заданий;

S_1 – в системе одно задание и оно обрабатывается на ЭВМ1;

S*n* – в системе n заданий и они обрабатываются на ЭВМ1, ЭВМ2,..ЭВМn ;

S_{n+1} – в системе (*n*+1) задание, n заданий обрабатываются на ЭВМ и одно задание стоит в очереди;

….

S*n*+*m* - в системе (*n*+*m*) заданий, *n* заданий обрабатываются на ЭВМ и *m* заданий стоят в очереди.

Рост числа заявок в системе происходит под воздействием их потока с интенсивностью λ, а уменьшение – под воздействием потока обслуживания с интенсивностью μ. Размеченный граф состояний системы приведен на рис.2.3.

Увеличение числа одновременно работающих машин приводит к росту интенсивности обслуживания от μ до *n*μ. Дальнейший рост числа заявок переводит систему в состояние *n*+1, *n*+2, ..n+m, а интенсивность потока обслуживания будет оставаться неизменной, равной nμ..

При исследовании такой вероятностной системы важно знать значение вероятностей состояний, с помощью которых можно вычислить показатели эффективности, такие, как количество заданий в системе, время ожидания обработки, пропускная способность и др.

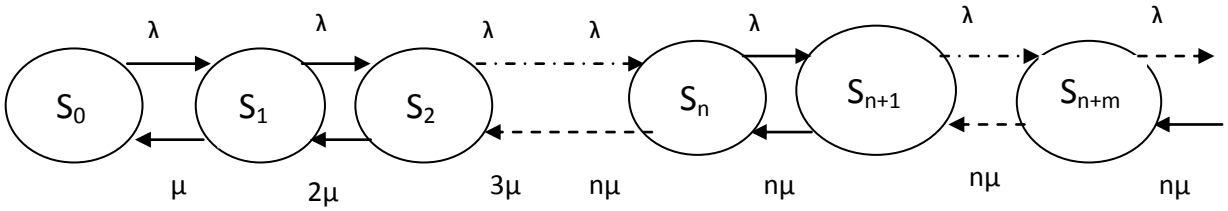


Рисунок 2.3 - Граф состояний многоканальной системы обслуживания с неограниченной очередью

Дискретная система в любой момент времени может находиться только в одном состоянии, поэтому

$$\sum_{i=1}^{k} P_i(t) = 1,$$

где *k*- число возможных состояний системы.

В процессе работы реальные вычислительные системы быстро достигают установившегося режима. Тогда вероятности состояний не будут зависеть от времени. Для вычисления финальных вероятностей используют систему дифференциальных уравнений Колмогорова, которые превращаются в систему алгебраических уравнений. На основе графа (рис. 2.3) может быть записана следующая система алгебраических уравнений [9]:

$$\lambda P_0 = \mu P_1;$$

$(1\mu + \lambda)P_1 = \lambda P_0 + 2\ \mu P_2;$

$(2\mu + \lambda)P_2 = \lambda P_1 + 3\ \mu P_3;$

……..

$(n\mu + \lambda)P_n = \lambda P_{n-1} + n\mu P_{n+1};$

$(n\mu + \lambda)P_n = \lambda P_{n-1} + n\mu P_{n+1};$

$(n\mu + \lambda)P_{n+1} = \lambda P_n + n\mu P_{n+2};$

…….

$(n\mu + \lambda)P_{n+j} = \lambda P_{n+j-1} + n\mu P_{n+j+1};\ j>=1.$

Финальные вероятности состояний системы в результате решения системы уравнений описываются следующими уравнениями:

$$P_0 = \left(1 + \sum_{i=1}^{n} \frac{\rho^i}{i!} + \frac{\rho^{n+1}}{n!(n-\rho)}\right)^{-1},$$

$$P_i = \frac{\rho^i}{i!} P_0, i = 1,2,..n,$$

$$P_{n+j} = \frac{\rho^{n+j}}{n^j n!} P_0,$$

где P_0 –вероятность состояния S_0, при котором в системе заявок нет;

$\rho=\lambda/\mu$ – параметр системы, показывающий, сколько в среднем заявок приходит в систему за время обслуживания заявки одной ЭВМ (одним каналом обслуживания);

P_i - вероятность состояния системы S_i, i=1, 2, ...n;

P_n – вероятность того, что все ЭВМ заняты обслуживанием заявок;

P_{n+j} – вероятность того, что все ЭВМ системы заняты обработкой заданий

и j заявок стоят в очереди.

Приведенные формулы имеют смысл тогда, когда очередь конечна,

т.е. $\rho/n < 1$ или $\lambda/n\mu < 1$.

Это выражение говорит о том, что в среднем число заданий, приходящих в вычислительную систему в единицу времени, должно быть, меньше числа обрабатываемых заданий в единицу времени всеми ЭВМ системы. При $\rho/n>1$ очередь растет до бесконечности и такая система не справится с потоком заданий. Тут появляются задания, ожидающие обработки вечно.

Основными показателями эффективности работы системы являются:

- среднее число занятых каналов (ЭВМ)

$$\bar{k} = \frac{\lambda}{\mu} = \rho,$$

- среднее число заданий в очереди

$$L_{оч} = \frac{\rho^{n+1} P_0}{n n! (1 - \rho / n)^2},$$

- среднее число заданий в системе

$$L_{сист} = L_{оч} + \bar{\kappa},$$

- среднее время пребывания задания в системе

$$W_{ист} = \frac{L_{сист}}{\lambda},$$

- среднее время пребывания задания в очереди

$$W_{оч} = \frac{L_{оч}}{\lambda}.$$

Для уменьшения времени пребывания задания в системе, а значит, и в очереди, требуется при заданной интенсивности потока заявок либо увеличивать число обслуживающих ЭВМ, либо уменьшить время обслуживания каждой ЭВМ, либо и то и другое вместе.

С помощью теории массового обслуживания можно получить аналитические выражения и при других дисциплинах обслуживания очереди и конфигурациях вычислительной системы.

При немарковских процессах в системе и не простейших потоках аналитические выражения получить трудно. В таких случаях моделирование проводят с помощью метода статистических испытаний (метод Монте – Карло), который позволяет создать алгоритмическую модель, включающую элементы случайности. Путем многократного запуска модели получают статистические данные, обработка которых дает значения финальных вероятностей состояний.

Тестовые задания

1. Общие сведения о теории массового обслуживания. Предмет теории массового обслуживания.
2. Параметры и характеристики систем массового обслуживания.
3. Моделирование систем массового обслуживания (СМО).
4. Показатели эффективности систем массового обслуживания.
5. Критерии эффективности систем массового обслуживания.
6. Моделирование вычислительных процессов и алгоритмов обслуживания вычислительных задач
7. Организация вычислительного процесса.
8. Граф состояний многоканальной системы обслуживания с неограниченной очередью.
9. Вычисление финальных вероятностей системы обслуживания. Система алгебраических уравнений Колмогорова.
10. Основными показателями эффективности работы системы массового обслуживания.

Лекция 3 **Введение в теорию нечетких множеств**

3.1 Общие понятия

В математике давно используется понятие множества – совокупности объектов, выделенных по некоторому признаку. Это понятие является базовым в современной математике и потому не определяется строго, формально. Так, если задано некоторое *базовое множество X* (конечное или бесконечное), то его *подмножеством* (*четким подмножеством*) *A* называется любое множество, содержащее в себе только элементы множества *X* (хотя, может быть, и не все его элементы).

Любое подмножество *A* множества *X* можно описать его *функцией принадлежности* $\mu_A : X \to \{0; 1\}$, значение которой для элемента $x \in X$ равно единице в том случае, если этот элемент принадлежит множеству *A*, и нулю – в противном случае.

Соответствие между подмножествами множества *X* и всевозможными функциями принадлежности $\mu : X \to \{0; 1\}$ является взаимно однозначным, то есть, определив некоторое подмножество, мы можем определить его функцию принадлежности, и обратно, задание функции $\mu_A : X \to \{0; 1\}$ задает и подмножество множества *X*.

В четком множестве любой элемент может или принадлежать ему, или не принадлежать, поэтому функция принадлежности принимает лишь два возможных значения ноль или единица.

В нечетком же множестве (точнее, в нечетком подмножестве базового множества *X*) любой элемент $x \in X$ может принадлежать множеству с некоторой степенью достоверности, принимающей значения от нуля (элемент достоверно не принадлежит множеству) до единицы (элемент достоверно принадлежит множеству). Соответственно и функция принадлежности нечеткого множества может принимать любое значение из отрезка [0; 1].

Мы определим понятие нечеткого множества через его функцию принадлежности. Пусть *X* – некоторое обыкновенное (четкое) множество. В дальнейшем мы будем рассматривать его нечеткие подмножества.

Определение 1**.** *Нечетким множеством* $\tilde{A}$ в *X* называется функция $\mu_A : X \to \{0; 1\}$, которая каждому из элементов множества *X* ставит в соответствие степень его принадлежности нечеткому множеству $\tilde{A}$.

Нечеткое множество $\tilde{A}$ называется *нормальным*, если $\sup_{x \in X} \mu_{\tilde{A}}(x) = 1$

В противном случае оно называется *субнормальным*. *Носителем supp* $\tilde{A}$ нечеткого множества $\tilde{A}$ называется обычное множество $supp\ \tilde{A} := \{x \in X : \mu_{\tilde{A}}(x) > 0\}$ *A*.

Пример 3.1. Пусть множество X – это множество всех действительных чисел. Множество A чисел, больших нуля, будет его четким подмножеством с функцией принадлежности

$$\mu_{A^{\sim}}(x) = \begin{cases} 1, & x>0 \\ 0, & x\le 0 \end{cases}$$

Мы можем определить нечеткое множество $A^{\sim}$ чисел, «много больших нуля», задав его функцию принадлежности $\mu_{A^{\sim}}$, например, следующим образом:

$$\mu_{A^{\sim}}(x) = \begin{cases} 1, & x>100 \\ 0, & x\le 0 \\ 0{,}5 - 0{,}5\cos(\pi x/100), & 0<x\le 100 \end{cases}$$

Действительно, числа, меньшие нуля, достоверно не являются много большими нуля, поэтому функция принадлежности в этих точка равна нулю.

(Далее нечеткие объекты (множества, отношения и т.д.) будут обычно обозначаться волной).

Числа, большие ста, в большинстве приложений (будем считать, что и в нашем случае), достоверно считаются много большими нуля, поэтому функция принадлежности для таких чисел равна единице. По поводу же чисел в интервале между 0 и 100 достоверно сказать, что они являются много большими нуля, нельзя, поэтому функция принадлежности на этом интервале принимает значения между нулем и единицей. В то же время понятно, что чем больше число, тем больше у нас оснований считать его много большим нуля. Поэтому на интервале от нуля до ста функция принадлежности монотонно возрастает.

Носителем нечеткого множества $A^{\sim}$ является интервал $(0; +\infty)$.

Функции принадлежности множеств A и $A^{\sim}$ приведены на рисунке 3.1.

Зачем же было введено понятие нечеткого множества? Для того же, для чего вводятся и другие математические объекты – чтобы описывать окружающий нас мир. В действительности большинство понятий, которые используют люди в повседневной жизни, являются нечеткими! Когда сапожник ждет *около трех минут* после нанесения клея перед склеиванием, когда хозяйка в соответствии с рецептом кладет в суп *две щепотки* соли, когда менеджер в коммерческой фирме выполняет указание руководства *существенно по-*

высить объемы продаж – все они выполняют нечеткие инструкции, сформулированные неформально с помощью разговорного языка.

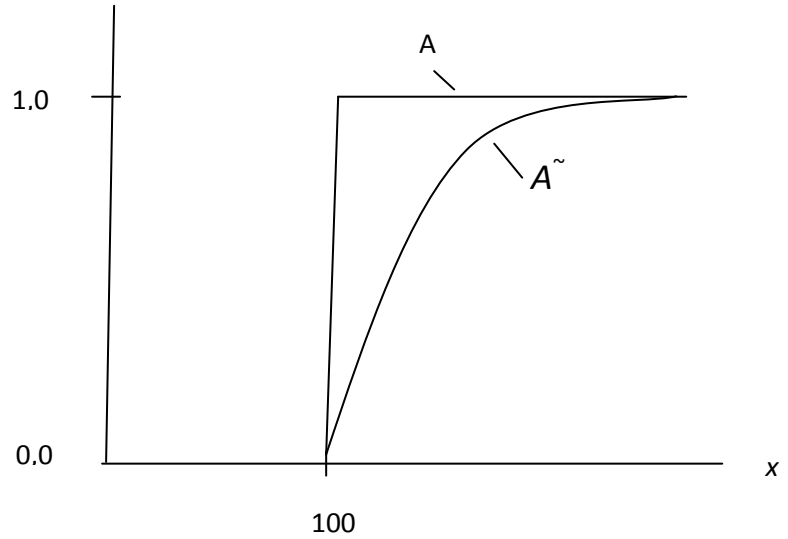


Рисунок 3.1 - Сравнение функций принадлежности четких и нечетких множеств

Даже формально четкие понятия, используемые в обыденной жизни, воспринимаются людьми как нечеткие. Например, в рецепте может быть четко указано «две чайные ложки соли», но хозяйка понимает, что блюдо не будет испорчено и если будет положено две с половиной ложки, не говоря уже о том, что чайные ложки, вообще говоря, бывают разной емкости.

Удобным способом математического описания неформальных понятий, подобных упомянутым выше, являются нечеткие множества.

Язык нечетких множеств имеет существенные преимущества перед языком теории вероятностей в том случае, когда оценки получаются из опроса экспертов. Известно, что люди в большинстве своем неправильно оценивают вероятности (особенно большие и малые). Потому требовать от экспертов – специалистов в конкретных предметных областях, а не математиков, оценок в форме распределения вероятности зачастую невозможно. Использование же полученных результатов для принятия решений можно квалифицировать как самонадеянное. Описание в форме нечетких множеств гораздо менее требовательно к квалификации экспертов и зачастую гораздо точнее отражает суть дела и имеющуюся у ЛПР информацию.

Конечно, за это удобство приходится платить. Предлагаемые теорией решения, основанные на нечеткой информации, и сами несут на себе печать нечеткости. Они могут рассматриваться лишь как рекомендации для ЛПР, требуя от него выбора одного из пред-

лагаемых вариантов. Тем не менее, даже этот факт можно рассматривать как достоинство теории – он показывает, как увеличение информированности ЛПР сказывается на достоверности и правильности принимаемых решений.

3.2 Операции над нечеткими множествами

Для того чтобы построить содержательную теорию нечетких множеств необходимо определить операции (такие как объединение, пересечение и т.п.) над нечеткими множествами, аналогичные операциям над обычными, четкими множествами. Сделать это позволяет аналогия между представлением четких и нечетких множеств в форме их функций принадлежности. Большинство операций над обычными множествами может быть сформулировано через операции над их функциями принадлежности. В то же время, функция принадлежности обычного множества является частным случаем функции принадлежности нечеткого множества, что позволяет *непосредственно* обобщать формулы для четких множеств на нечеткий случай. При этом при применении к четким множествам операция дает обычный результат.

Определение 2. Нечеткое множество $A^{\sim}$ в X *является подмножеством* нечеткого множества $B^{\sim}$ в X ($A^{\sim}$ принадлежит $B^{\sim}$, $A^{\sim} \in B^{\sim}$), если для всех $x \in X$ $\mu_{A^{\sim}}(x) \leq \mu_{B^{\sim}}(x)$. В теории множеств считается, что пустое множество 0 принадлежит любому множеству. Также по определению 2 и нечеткое пустое множество с функцией принадлежности $\mu_0(x) \equiv 0$ принадлежит любому нечеткому множеству.

Функцию принадлежности четкого множества $C = A \cap B$ – пересечения множеств A и B – можно записать в виде $\mu_C(x) = \min[\mu_A(x); \mu_B(x)]$.

Эту формулу можно использовать и для пересечения нечетких множеств $A^{\sim}$ и $B^{\sim}$, положив по определению (1):

$$\mu_{A^{\sim} \cap B^{\sim}}(x) := \min[\mu_{A^{\sim}}(x); \mu_{B^{\sim}}(x)] \text{ для всех } x \in X .$$

Введем также операции пересечения и объединения произвольного семейства нечетких множеств.

Определение 3. Пусть задано семейство нечетких множеств $A^{\sim}_{\Lambda}$, индексированных параметром $\lambda \in \Lambda$. Их *пересечением* будем называть нечеткое множество $B^{\sim}$ с функцией принадлежности

$$\mu_{B^{\sim}}(x) := \inf_{\lambda \in \Lambda} \mu_{A^{\sim}_{\Lambda}}(x)$$

Объединением же множеств $A^{\sim}_{\Lambda}$ $\Lambda \in \Lambda$ будем называть нечеткое множество $C^{\sim}$ с функцией принадлежности

$$\mu_{C^{\sim}}(x) := \sup_{\Lambda \in \Lambda} \mu_{A^{\sim}_{\Lambda}}(x)$$

Заметим, что носитель supp $(A^{\sim} \cap B^{\sim})$ пересечения нечетких множеств $A^{\sim}$ и $B^{\sim}$ равен пересечению supp $A^{\sim} \cap$ supp $B^{\sim}$ их носителей, а носитель объединения – объединению носителей.

Определение 4. *Дополнением* нечеткого множества $A^{\sim}$ называется нечеткое множество $A^{\sim'}$ с функцией принадлежности $\mu_{A^{\sim'}}(x) := 1 - \mu_{A^{\sim}}(x)$.

Здесь и далее дополнения нечетких множеств обозначаются штрихом.

Дополнение часто используется в теории принятия решений для построения отрицаний нечетких понятий. Например, для нечеткого множества «чисел много больших нуля» его дополнением будет множество чисел, «не являющихся много большими нуля».

Для обычных множеств пересечение множества и его дополнения пусто. Как показывает следующий пример, для нечетких множеств это уже не так.

Пример 3.2. Найдем пересечение нечеткого множества $A^{\sim}$ чисел, много больших нуля и его дополнения.

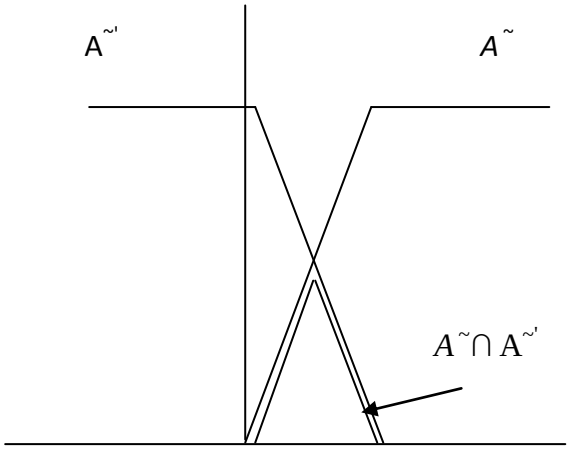


Рисунок 3.2 - Пересечение нечеткого множества и его дополнения

На рисунке 3.2 изображены функция принадлежности множества $A^{\sim}$, его дополнения $A^{'\sim}$, и их пересечения $A^{\sim} \cap A^{\sim'}$. Содержательно нечеткое множество $A^{\sim} \cap A^{\sim'}$ можно интерпретировать как нечеткое множество чисел, которые являются много большими нуля и одновременно не являющимися много большими нуля. Непустота этого множества

является следствием того, что понятие «быть много большим» описано нечетко и в некотором смысле множество $A^{\sim} \cap A^{\sim'}$ может рассматриваться как «нечеткая граница» между множеством $A^{\sim}$ и его дополнением.

Дополнение нечеткого множества является частным случаем операции разности двух нечетких множеств:

Определение 5. *Разностью* $A^{\sim} \setminus B^{\sim}$ нечетких множеств $A^{\sim}$ и $B^{\sim}$ называется нечеткое множество с функцией принадлежности

$$\mu_{A^{\sim} \setminus B^{\sim}}(x) := \max[\mu_{A^{\sim}}(x) - \mu_{B^{\sim}}(x); 0].$$

Таким образом, дополнение нечеткого множества $A^{\sim}$ в X – это разность $X \setminus A^{\sim}$.

Для нечетких множеств можно определить и операции, аналогов которых для четких множеств нет. Такова, скажем, операция умножения нечеткого множества на число.

Определение 6. *Произведением нечеткого множества* $A^{\sim}$ *на число* $a \in [0; 1]$ называется нечеткое множество $aA^{\sim}$ с функцией принадлежности

$$\mu_{aA^{\sim}}(x) = a\mu_{A^{\sim}}(x).$$

Определим еще одно понятие, которое оказывается очень полезным при анализе нечетких множеств.

Определение 7. *Множеством уровня* $a \in [1; 0]$ нечеткого множества $A^{\sim}$ называется обыкновенное множество $A^{\sim} := \{x \in X : \mu_{A^{\sim}}(x) \geq a\}$ – множество точек, степень принадлежности которых нечеткому множеству не меньше, чем a.

С использованием операции умножения на число произвольное нечеткое множество $A^{\sim}$ можно представить в виде объединения его взвешенных множеств уровня по формуле:

$$A^{\sim} = \cup\, aA_a^{\sim}, \qquad a \in [0,1]$$

Определение 8. *Выпуклой комбинацией* нечетких множеств $A_1^{\sim}, \ldots, A_n^{\sim}$ называется нечеткое множество $A^{\sim}$ с функцией принадлежности:

$$\mu_{A^{\sim}}(x) := \sum a_i \mu_{Ai^{\sim}}(x), \; i = 1, 2, \ldots n,$$

где $a_1, \ldots, a_n$ – произвольные неотрицательные числа, сумма которых равна единице.

С помощью выпуклой комбинации можно ввести понятие *выпуклого* семейства нечетких множеств – семейства, содержащего все выпуклые комбинации своих элементов.

Например, семейство всех нечетких множеств в X выпукло. Именно возможность рассматривать выпуклые семейства и выпуклые замыкания является основным техническим преимуществом нечетких множеств перед обыкновенными множествами.

Определение 9. *Декартовым произведением* нечетких множеств $A_1^{\sim}$ из X_1, …, $A_n^{\sim}$ из X_n называется нечеткое множество $A^{\sim}$ в $X = X_1 * \ldots * X_n$ с функцией принадлежности:

$$\mu_{A}\sim (x) := \min[\ \mu_{A1}\tilde{}\ (x_1);\ ...;\ \mu_{An}\tilde{}\ (x_n)].$$

Итак, основным способом определения операций над нечеткими множествами является обобщение соответствующих операций над обычными обществами.

3.3 Нечеткие отображения и задачи принятия решений

Продолжим обобщать понятия обычной, четкой математики на нечеткий случай. Определим понятия нечеткого отображения, образа и прообраза нечеткого множества при нечетком отображении и с их помощью решим две задачи принятия решений – задачу достижения нечеткой цели и задачу оптимизации при нечетких ограничениях.

Нечеткие отображения

Обычным, *четким отображением* (многозначным) f множества X во множество Y называется произвольное подмножество декартова произведения $X * Y$, то есть $f \in X * Y$. Множество X называется *областью определения* отображения, а Y – *областью значений*. Для фиксированного элемента $x^* \in X$ области определения отображения его *образом* при отображении f называется множество $f_j(x^*) := \{y \in Y : (x^*, y) \in f\}$.

Образом множества $A \in X$ при отображении f называется объединение образов всех элементов A, то есть множество

$$f(A) := \bigcup_{x \in f(A)} f(x) = \{y \in Y : x \in A, (x^*, y) \in f\}$$

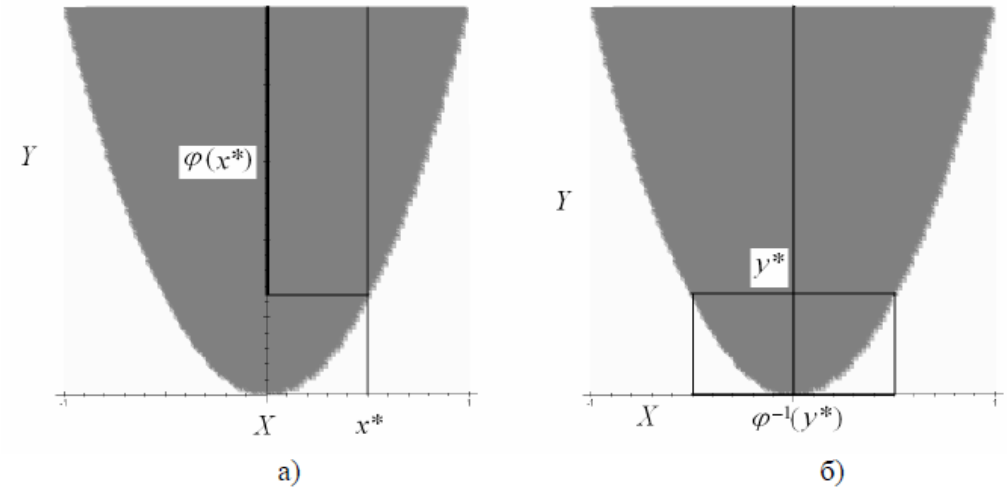


Рисунок 3.3 - Четкое многозначное отображение

Иногда на отображение накладывается дополнительное условие, что образ любого элемента должен состоять не более чем из одного элемента. В этом случае говорят об *однозначных* (*функциональных*) *отображениях*.

Для отображения f из X в Y *обратным отображением* f^{-1} называется такое отображение из Y в X, что $(y, x) \in f^{-1} \leftrightarrow (x, y) \in f$. Образ элемента $y^* \in Y$ при обратном отображении будем обозначать $f^{-1}(y^*)$. Очевидно, он является подмножеством множества X.

Пример 3.3. Рассмотрим четкое многозначное отображение f отрезка $X = [-1; 1]$ в отрезок $Y = [0; 1]$, которое каждой точке $x \in X$ ставит в соответствие множество точек $y \in Y$, больших, чем x^2. В этом случае $f = \{(x, y) \in X * Y : y > x^2\}$.

Этому отображению соответствует затененная область на рисунке 3.3. Образом точки $x^* = 0.5$ при таком отображении будет интервал (0.25;1] (см. рисунок 3.3 а), а образом точки $y^* = 0.25$ при обратном отображении – интервал (-0.5; 0.5) (см. рисунок 3.3 б).

Пример 3.4. В моделях принятия решений важную роль играет отображение, которое каждому возможному действию ЛПР ставит в соответствие реакцию системы на это действие. Пусть, например, действие ЛПР (менеджера по продажам) состоит в выборе суммы инвестиций в рекламу из множества X, а состояние системы (фирмы) описывается суммой ее продаж из множества Y. Однозначное отображение f из X в Y каждой сумме инвестиций ставит в соответствие сумму результирующих продаж, определяя, по сути, **модель системы (фирмы)**. Типичный вид этого отображения можно задать формулой $f(x) = s(1- \exp(-\Lambda x))$, где параметр s – это максимальная емкость рынка, а Λ – эластичность спроса. Пусть $s = 106$, $\Lambda = 10^{-5}$. Тогда вложение в рекламу суммы $x = 100000$ приводит к сумме продаж $y = 632121$, то есть точка $y = 632121$ является образом точки $x = 100000$ при отображении $f(x) = s(1- \exp(-\Lambda x))$.

Если четкое отображение – это подмножество декартова произведения $X * Y$ области определения и области значений, что же такое нечеткое отображение? Очевидно нечеткое подмножество $Y * X$. Тогда нечеткое отображение $f^{\sim}$ множества X во множество Y можно описать его функцией принадлежности $\mu_{f^{\sim}} : X * Y \to [0; 1]$. Функция принадлежности $\mu_{f^{\sim}}(x, y)$ определяет степень достоверности того, что точка y принадлежит образу точки x при нечетком отображении $f^{\sim}$. И как образом элемента $x^* \in X$ при четком отображении было четкое подмножество множества Y, так же образом $x^* \in X$ при нечетком отображении будет нечеткое подмножество множества Y с функцией принадлежности $\mu_{f^{\sim}}(x^*, y)$

Образом четкого множества при нечетком отображении будет объединение образов его элементов:

$$\mu_{f^{\sim}(A)}(y) := \sup_{x \in A} \mu_{f^{\sim}}(x, y)$$

Однако чтобы завершить обобщение понятия образа на нечеткий случай, необходимо определить образ нечеткого множества при нечетком же отображении. Понятно, что

образы элементов нечеткого множества должны объединяться с учетом степени принадлежности этих элементов нечеткому множеству. Запишем формулу для образа четкого множества *A*:

$$\mu_{\tilde{f}(A)}(y) := \sup_{x \in X} \min[\mu_A(x); \mu_{\tilde{f}}(x, y)]$$

Эта формула уже допускает непосредственное обобщение на нечеткий случай (Заменой четкого множества *A* на нечеткое множество $\tilde{A}$) , что позволяет дать следующее определение.

Заметим, что x^* фиксировано, и выражение $\mu_{\tilde{f}}(x^*, y)$действительно задает функцию принадлежности нечеткого подмножества множества *Y*.

Определение 10. *Образом* $\tilde{f}(\tilde{A})$ нечеткого множества $\tilde{A} \in X$ при нечетком отображении $\tilde{f}: X \to Y$ называется нечеткое подмножество множества *Y* с функцией принадлежности

$$\mu_{\tilde{f}(A)}(y) := \sup_{x \in X} \min[\mu_A(x); \mu_{\tilde{f}}(x, y)] \qquad (3.1)$$

В частности, если отображение $f: X \to Y$ четкое, то формулу (3.1) можно упростить, так как под знаком минимума остаются только образы точки *y* при обратном четком отображении f^{-1}. Действительно,

$$\mu_{f(\tilde{A})}(y) := \sup_{x \in X} \min[\mu_{\tilde{A}}(x); \mu_f(x, y)] = \sup_{x: f(x)=y} \mu_{\tilde{A}}(x) = \sup_{x \in f^{-1}(y)} \min \mu_{\tilde{A}}(x);$$

Пример 3.5. Пусть в условиях предыдущего примера реакция системы известна ЛПР лишь неточно, то есть отображение $\tilde{f}$ является нечетким. Скажем, для каждого выбора суммы инвестиций $x \in X$ его образом при отображении $\tilde{f}$ является нечеткое множество возможных продаж с функцией принадлежности $\mu_{\tilde{f}(x)}(y) = \max[0; 1 - (y - f(x))^2 / x^2]$, где функция «наиболее достоверных продаж» $f(x) = s(1 - exp(-\Lambda x))$ взята из предыдущего примера.

Функция принадлежности отображения $\tilde{j}$ изображена на рисунке 3.4, большему значению функции принадлежности соответствует более темный тон. На этом же рисунке изображена кривая $f(x) = s(1 - \exp(-\Lambda x))$, на которой функция принадлежности нечеткого отображения принимает максимальное значение, равное единице. На рисунке 3.5 изображены функции принадлежности образов точек $x_1 = 1800$, $x_2 = 2000$, $x_3 = 2200$ – нечеткая реакция системы на выбор соответствующих объемов инвестиций. Выбор конкретного объема инвестиций можно понимать как четкую инструкцию – «что делать». Пусть, однако,

ЛПР (менеджер) хочет узнать реакцию системы (фирмы) на выбор инвестиций в рекламу в размере «примерно 60000».

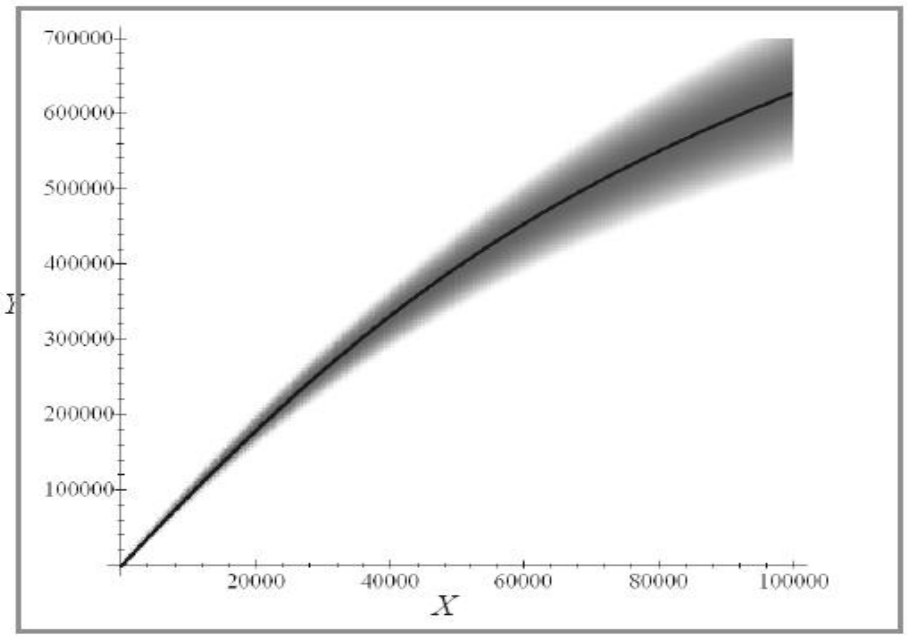


Рисунок 3.4 – Функция принадлежности нечеткого образа

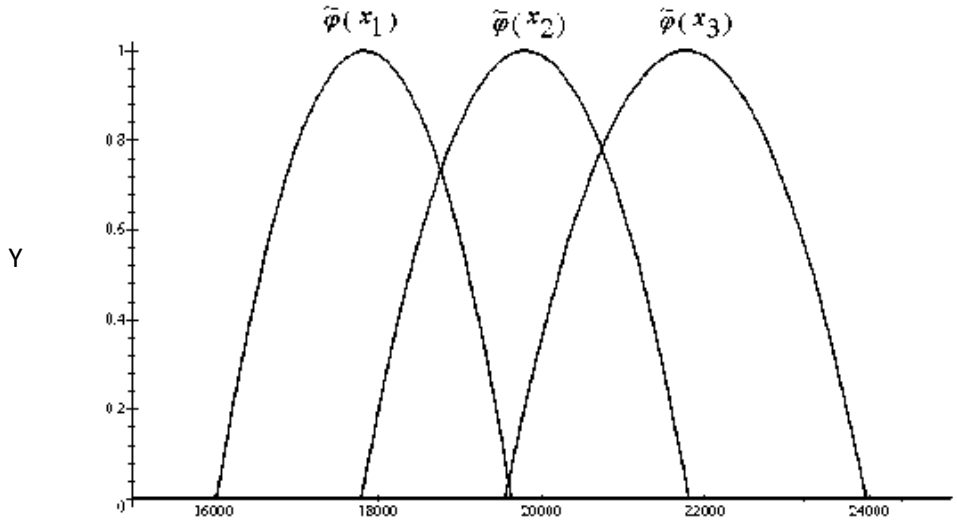


Рисунок 3.5 – Образы точек при нечетком отображении

Эта нечеткая инструкция описывается некоторым нечетким множеством $A^{\sim}$, и реакция системы будет образом $f^{\sim}(A^{\sim})$ этого нечеткого множества при нечетком отображении $f^{\sim}$. Образы нечетких множеств удобно строить следующим образом (см. рисунок 3.6). На нем, так же как и на рисунке 3.5, изображена функция принадлежности нечеткого отображения $f^{\sim}$. Горизонтальная ось соответствует множеству X и от нее вниз, «вверх ногами» строится функция принадлежности нечеткого множества $A^{\sim}$ – «примерно 60000».

Вертикальная ось соответствует множеству Y и от нее влево (повернутой на 90°) строится функция принадлежности образа $f^{\sim}(A^{\sim})$. Стрелки на рисунке 3.6 показывают направление «переноса» точек нечеткого множества – образ нечеткого множества является объединением образов точек этого множества с учетом их степени принадлежности.

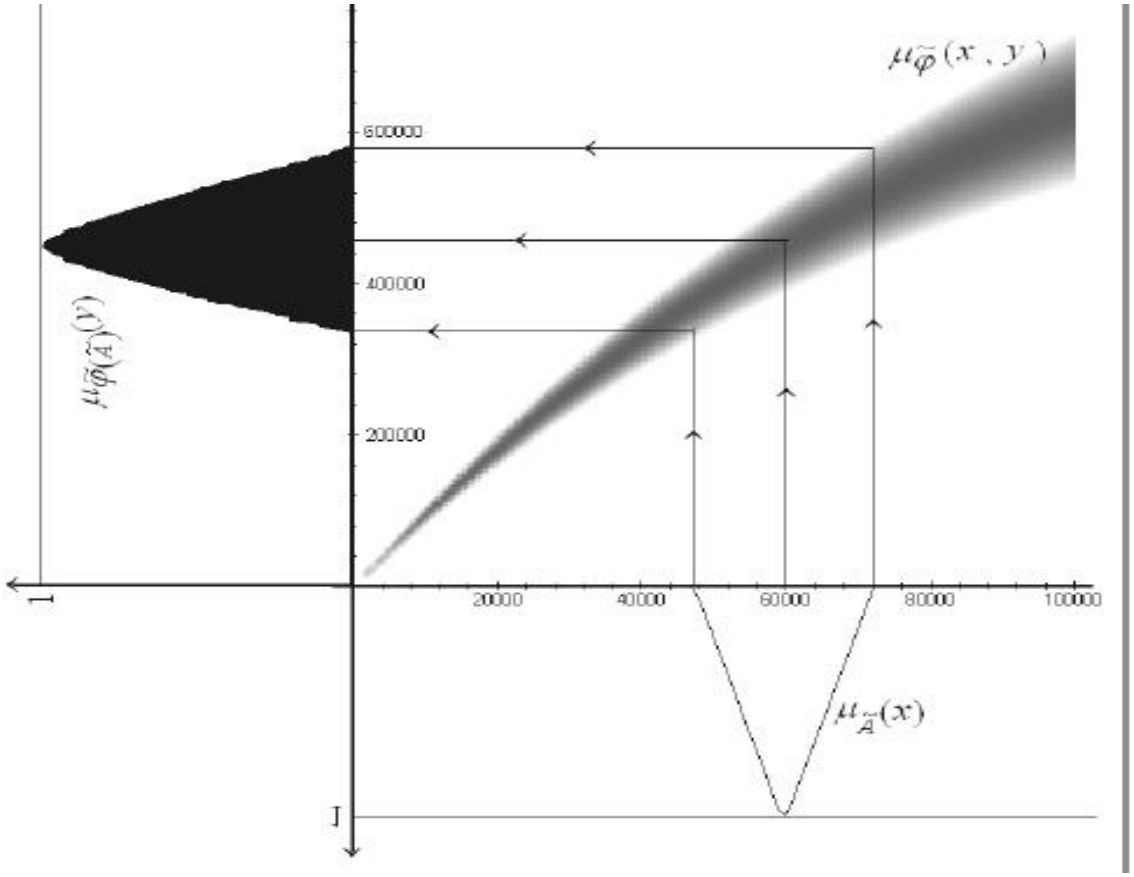


Рисунок 3.6 - Образ нечеткого множества при нечетком отображении

3.4 Прообраз нечеткого множества при нечетком отображении

Понятие образа нечеткого множества при нечетком отображении позволяет ЛПР вычислять нечеткую реакцию системы на нечеткие же управляющие воздействия. Тем не менее, для теории принятия решений гораздо важнее обратная задача – найти действия, которые приводят к желаемому результату.

Решить данную задачу позволяет понятие прообраза нечеткого множества, описанию которого посвящен настоящий раздел.

Определение 11. *Прообразом* $A^{\sim} \in X$ нечеткого множества $B^{\sim} \in Y$ при нечетком отображении $f^{\sim}: X \to Y$ называется объединение всех нечетких множеств $a^{\sim}$ таких, что их образ $f^{\sim}(a^{\sim})$ принадлежит нечеткому множеству $B^{\sim}$, то есть таких, что

$$\sup_{x \in X} \min[\mu_{a^{\sim}}(x); \mu_{f^{\sim}}(x, y)] \le \mu_{B^{\sim}}(y) \text{ для всех } y \in Y$$

Содержательно, прообраз нечеткого множества $B^{\sim} \in Y$ – это «максимальное» нечеткое множество $A^{\sim} \in X$, переходящее в $B^{\sim}$ при нечетком отображении $f^{\sim}: X \to Y$.

Таким образом, чтобы для некоторой нечеткой реакции системы определить то действие (возможно, нечеткое), которое приводит к данной реакции, необходимо найти прообраз нечеткого множества реакции.

Можно, однако, заметить, что далеко не любое нечеткое множество имеет непустой прообраз при нечетком отображении. Пустота прообраза «одноточечного» множества имеет простое содержательное объяснение. Это множество можно интерпретировать, как желание ЛПР получить единственный исход с положительной достоверностью, обеспечив нулевую достоверность остальных исходов. Но это невозможно, так как поведение системы нечетко и выбор любого действия приводит к нескольким возможным исходам. Вывод здесь прост – ЛПР должен ставить перед собой реальные цели, смягчая требования к нечеткому множеству результата.

Вычисление прообразов нечетких множеств имеет важное прикладное значение. В то же время, вычисление «по определению» весьма трудоемко. Следующий приводимый без доказательства результат позволяет дать более простую характеристику прообраза нечеткого множества $B^{\sim} \in Y$, пригодную для численной реализации.

Определим следующие четкие множества:

$N: \{(x, y) \in X * Y: \mu_{f^{\sim}}(x, y) > \mu_{B^{\sim}}(y)\}$ – множество пар элементов из области определения и области значений отображения, в которых значение функции принадлежности отображения строго превышает значение функции принадлежности множества $B^{\sim}$.

$N_x: \{y \in Y: (x, y) \in N$– «срез» множества N при фиксированном $x \in X$.

$X^0 := \{ x \in X: N_x \neq \emptyset\}$ – множество элементов области определения, для которых множество N_x не пусто.

Теорема 1. Функция принадлежности прообраза $A^{\sim} \in X$ нечеткого множества $B^{\sim} \in Y$ при нечетком отображении $f^{\sim}: X \to Y$ описывается выражением

$$\mu_A(x) = \begin{cases} \inf\limits_{y \in N_x} \mu_{B^{\sim}}(y), & x \in X^0 \\ 1, & x \in X \backslash X^0 \end{cases}$$

Проиллюстрируем нахождение прообраза нечеткого множества с помощью этой теоремы следующим примером.

Пример 3.6. Рассмотрим нечеткое отображение $f^{\sim}: X \to Y$ из примера 3.5. Пусть необходимо найти прообраз нечеткого множества $B^{\sim} \to Y$, функция принадлежности которого изображена на рисунке 3.7 слева (так же как и на рисунке 3.6, она изображена повер-

нутой влево на 90°). Также на рисунке для заданных $f^{\sim}$ и $B^{\sim}$ построено множество N – оно изображено черным цветом. Для двух точек, x_1 , x_2 на оси Y изображены множества N_{x1} , N_{x2} – «вертикальные срезы» множества N в этих точках. Из рисунка видно, что множество X^0 в данном примере совпадает с множеством X, так как для любой точки $x \in X$ множество N_x не пусто.

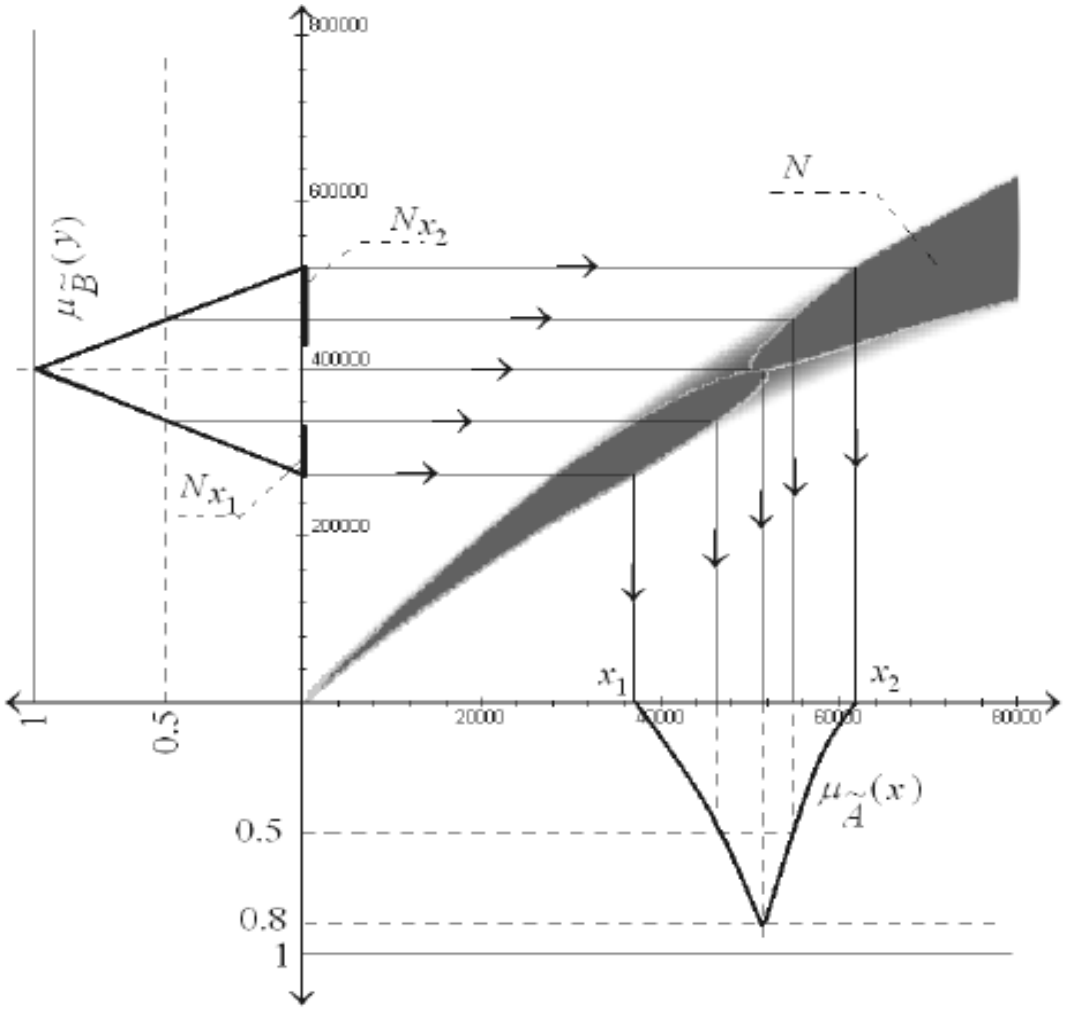


Рисунок 3.7 - Прообраз нечеткого множества при нечетком отображении

Следовательно, для вычисления значения функции принадлежности прообраза в заданной точке $x \in X$ необходимо найти минимальное значение $\mu_{B^{\sim}}(y)$ на множестве N_x . Так, например, $\mu_{A^{\sim}}(x_1) = \mu_{A^{\sim}}(x_2) = 0$, поскольку соответствующие множества N_{x1} и N_{x2} включают точки, в которых $\mu_{B^{\sim}}(.) = 0$. Стоит отметить, что хотя прообраз в данном примере не пуст, но его образ не совпадает с исходным множеством $B^{\sim}$ (Максимальное значение функции принадлежности прообраза равно примерно 0.8, а, значит, по формуле (3.1) , максимальное значение функции принадлежности образа не может превышать 0.8) , так как отображение $f^{\sim}$ «слишком нечеткое» для того, чтобы обеспечить острый пик функции принадлежности множества $B^{\sim}$.

Тестовые задания

1. Четкие и нечеткие множества, определения.
2. Операции над четкими и нечеткими множествами.
3. Нечеткие отображения и задачи принятия решений
4. Образ четкого и нечеткого множества.
5. Прообраз нечеткого множества при нечетком отображении

.

Лекция 4 **Моделирование задач принятия решений**

4.1 Задача достижения нечеткой цели

Обобщения на нечеткий случай понятий отображения, образа и прообраза множеств достаточны для того, чтобы сформулировать и решить задачу принятия решения в условиях нечеткой информации.

Задача формулируется так. Есть множество X возможных действий ЛПР и множество Y состояний управляемой системы. ЛПР в различной степени устраивают различные состояния системы – он стремится достичь своей цели, задаваемой нечетким подмножеством $\tilde{G} \in Y$. Для достижения своей цели ЛПР выбирает действия так, чтобы удовлетворить ограничениям на действия, задаваемым нечетким подмножеством $\tilde{C} \in X$. Состояние, в которое переходит система в зависимости от действия ЛПР, описывается нечетким отображением $\tilde{f}: X \to Y$.

Задача ЛПР состоит в том, чтобы определить действие (возможно, нечеткое), которое позволило бы ему одновременно достичь цели $\tilde{G}$ и удовлетворить ограничениям $\tilde{C}$. Предположим, что отображение $\tilde{f}$ – тождественное, и множество действий совпадает с множеством результатов. В этом случае и цель и ограничения являются подмножествами одного и того же множества X, а нечеткое множество $\tilde{D}$ действий, которые одновременно и достигают цели, и удовлетворяют ограничениям, равно пересечению нечетких множеств цели и ограничений, $\tilde{D} = \tilde{G} \cap \tilde{C}$. Тогда множество $\tilde{D}$ и является решением задачи достижения нечеткой цели.

Пример 4.1. Рассмотрим задачу, с которой периодически сталкивается каждый студент – задачу подготовки к экзамену. Пусть множество $X = \{1; 2; 3; 4; 5\}$ задает возможные уровни подготовки к экзамену (бо́льшее значение соответствует более интенсивной подготовке), а множество $Y = \{1; 2; 3; 4; 5\}$ описывает возможные исходы экзамена (оценки).

Пусть студента одинаково устраивает как оценка 4, так и 5 (наш студент не гонится за отличной оценкой), но категорически не устраивают оценки 1 или 2. Оценка 3 студента устраивает лишь частично.

Тогда цель студента можно описать нечетким множеством $\tilde{G}$, функция принадлежности которого приведена в следующей таблице.

y	1	2	3	4	5
$\mu_{\tilde{G}}(y)$	0	0	0,5	1	1

В то же время (в связи с недостатком времени или способностей) студент ограничен в возможностях подготовки к экзамену, и если возможность подготовки на уровне 1 и 2 не вызывает сомнений, то большие уровни подготовленности уже более сомнительны.

В следующей таблице приведена функция принадлежности нечеткого множества ограничений $C^{\sim}$.

x	1	2	3	4	5
$\mu_{C^{\sim}}(x)$	1	1	0,7	0,6	0,3

Предположим, что отображение, переводящее действия в результат, тождественное, то есть уровень подготовки однозначно определяет оценку на экзамене – уровень подготовки 1 приводит к оценке 1, уровень подготовки 2 – к оценке 2 и так далее.

Тогда задача выбора действия, достигающего цели с учетом ограничений, сводится к нахождению пересечения $D^{\sim} = G^{\sim} \cap C^{\sim}$ множеств цели и ограничений. Функция принадлежности множества $D^{\sim}$ приведена ниже.

X	1	2	3	4	5
$\mu_{D^{\sim}}(x)=min[\mu_{G^{\sim}}(x); \mu_{C^{\sim}}(x)]$	0	0	0,5	0,6	0,3

Таким образом, решение задачи достижения нечеткой цели само оказывается нечетким. Эта нечеткость является прямым следствием нечеткости в постановке задачи и может интерпретироваться как нечеткая рекомендация вида «готовиться примерно на 4».

Но ведь в реальности принимается единственное решение! И отдельного рассмотрения требует вопрос о конкретном действии, выбираемом на основе нечеткой рекомендации.

Часто в качестве четкого решения задачи достижения нечеткой цели предлагается выбор действия, имеющего максимальную степень принадлежности нечеткому решению – множеству $D^{\sim}$. Так, в рассмотренном примере четкая рекомендация состоит в том, чтобы готовиться «на четверку». Однако более осторожным и обоснованным представляется подход, в котором ЛПР предоставляется сама не-

четкая инструкция, как результат решения нечетко формализованной задачи и дальнейший выбор действия на основе этой рекомендации ЛПР осуществляет самостоятельно, основываясь, возможно, на информации, не нашедшей отражения в модели.

Однако каким образом искать решение задачи в том случае, когда отображение $f^{\sim}$ не является тождественным? В этом случае нечеткие подмножества ограничений $C^{\sim}$ и цели $G^{\sim}$ непосредственно не сравнимы, так как являются подмножествами разных пространств, X и Y соответственно. Однако мы можем отобразить множество цели $G^{\sim}$ во множество действий, найдя его прообраз $g^{\sim}$ при отображении $f^{\sim}$ – нечеткое множество действий, приводящих к заданной нечеткой цели без учета ограничений. Тогда, аналогично рассмотренному выше случаю, решением задачи будет пересечение $g^{\sim} \cap C^{\sim}$ прообраза цели с множеством ограничений.

Пример 4.2. Пусть в контексте предыдущего примера множества цели и ограничений прежние, а функция принадлежности отображения $f^{\sim}$ задается следующей таблицей.

Y	X				
	1	2	3	4	5
1	1	0,6	0	0	0
2	0	1	0,6	0	0
3	0	0	1	0,7	0,1
4	0	0	0	1	0,5
5	0	0	0	0	1

Из таблицы видно, что $f^{\sim}$ отражает риск получить оценку худшую, чем выбранный уровень подготовки. Для решения задачи необходимо по теореме 1 найти прообраз нечеткого множества цели при нечетком отображении. Поиск прообраза иллюстрируется следующей таблицей:

$\mu_{G^{\sim}}(y)$	$\mu_{g^{\sim}}(x)$	0	0	0	0,5	1
		X				
	Y	1	2	3	4	5
0	1	1	0,6	0	0	0
0	2	0	1	0,6	0	0
0,5	3	0	0	1	0,7	0,1
1	4	0	0	0	1	0,5
1	5	0	0	0	0	1

Цветным фоном выделены ячейки из множества N. Очевидно, действие $x = 5$ принадлежит множеству X^0, то есть, по теореме 1, $\mu_{g^\sim}(5) = 1$. Для вычисления принадлежности любого другого действия необходимо найти минимум *функции принадлежности цели* $G^\sim$ по затененным ячейкам столбца таблицы, соответствующего этому действию. Окончательное решение равно пересечению множества $g^\sim$ с множеством ограничений $C^\sim$ и имеет следующую функцию принадлежности:

X	1	2	3	4	5
$\mu_{D^\sim}(x)=min[\mu_{g^\sim}(x); \mu_{C^\sim}(x)]$	0	0	0	0,5	0,3

Мы видим, что риск получить худшую оценку ужесточает нечеткую рекомендацию – для достижения цели уже необходимо *«готовиться как минимум на* 4*»*. И даже при этом достоверность достижения цели при выборе уровня подготовки 4 оказывается ниже, чем в предыдущем примере.

4.2 Построение нечетких моделей

Ознакомимся с методами разработки математических моделей на основе использования математического аппарата нечетких систем в программе *MATLAB* 7.0.1

В *MATLAB* важная роль отводится специализированным группам программ, называемых *Toolboxes*. *Toolboxes* – это всесторонняя коллекция функций *MATLAB* (*M*-файлов), которые позволяют решать частные классы задач. Пакет прикладных программ *Fuzzy Logic* относится к теории нечетких (размытых) множеств. Обеспечивается поддержка современных методов нечеткой кластеризации и адаптивных нечетких нейронных сетей. Графические средства пакета позволяют интерактивно отслеживать особенности поведения системы.

Основные возможности пакета:

1) определение переменных, нечетких правил и функций принадлежности;

2) интерактивный просмотр нечеткого логического вывода;

3) современные методы: адаптивный нечеткий вывод с использованием нейронных сетей, нечеткая кластеризация;

4) интерактивное динамическое моделирование в *Simulink.*

Создание m-функции для моделирования. *M*-файлы являются обычными текстовыми файлами, которые создаются с помощью текстового редактора. Для операционной среды персонального компьютера система *MATLAB* поддерживает специальный встроенный

редактор/отладчик, хотя можно использовать и любой другой текстовый редактор с *ASCII*-кодами.

M-функции являются *M*-файлами, которые допускают наличие входных и выходных аргументов. Они работают с переменными в пределах *собственной рабочей области*, отличной от рабочей области системы *MATLAB.*

Построение нечётких моделей слабо формализовано. В каждом случае применяются свои параметры построения модели. В методе построения модели с использованием *FCM*-кластеризации можно управлять размерами выборки (количество наблюдений в выборке, количество входных переменных), используемой для построения модели, а также видом модели (Сугэно / Мамдани).

Одним из недостатков нечётких моделей является эффект «переобучения», когда модель даёт минимальную ошибку на элементах обучающего множества при большой ошибке на элементах тестирующего множества. Это означает, что модель «запомнила» обучающие наборы вместо того, чтобы научиться решать поставленную задачу. Для преодоления этого недостатка применяют разбиение исходной выборки на два подмножества: обучающую и проверочную. Обучающая используется для построения и настройки модели, проверочная – для определения пригодности модели.

Большие проблемы могут возникнуть при использовании большого числа входных переменных. При этом образуется огромное число внутренних элементов модели, сама модель становится относительно более медленной, затрудняется или становится вообще невозможным (при имеющихся технических мощностях) обучение модели.

Пакет *MATLAB Fuzzy Logic Toolbox* позволяет строить модели тремя различными способами:

1) *genfis*1 – построение нечёткой модели типа Сугено, которую далее необходимо обучить с использованием *anfis*-обучения;

2) *genfis*2 – генерация нечёткой модели с использованием алгоритма субтрактивной кластеризации;

3) *genfis*3 – генерация нечёткой модели с использованием нечёткой (*FCM*) кластеризации.

Выбор способа построения нечеткой модели, определение параметров (отбор информативных входных переменных, размер обучающей выборки, величина радиуса центра кластеризации) для построения качественной модели должен проводиться путём постановки вычислительного эксперимента со статистическими данными, используемыми для построения модели [13]. В качестве оценки адекватности модели можно использовать

значение коэффициента парной корреляции, который будет показывать, насколько достоверно нечёткая модель моделирует выходную переменную.

При помощи инструмента *Fuzzy Logic* можно посмотреть сформированную структуру модели и нечеткие правила. Можно посмотреть сформированные нечеткие переменные и функции принадлежности. При этом заданы минимальные и максимальные значения для *xi*, названия функций и переменных. Все функции принадлежности по умолчанию являются симметричными гауссовскими кривыми (*gaussmf*). Также можно просмотреть сформированный набор правил. Все правила представляются видом:
Если (Выражение1) *И* (Выражение2)…*И* (Выражение16) *то* (Значение 1)(Значение 2)…(Значение m).

Программная реализация системы для обеспечения взаимодействия множеств двух видов (четких и нечетких) вводится нечеткая система с фуззификатором (преобразование входных данных в нечеткое множество) на входе и дефуззификатором (преобразование нечетких множеств в конкретное значение) на выходе (рисунок 4.1) [2].

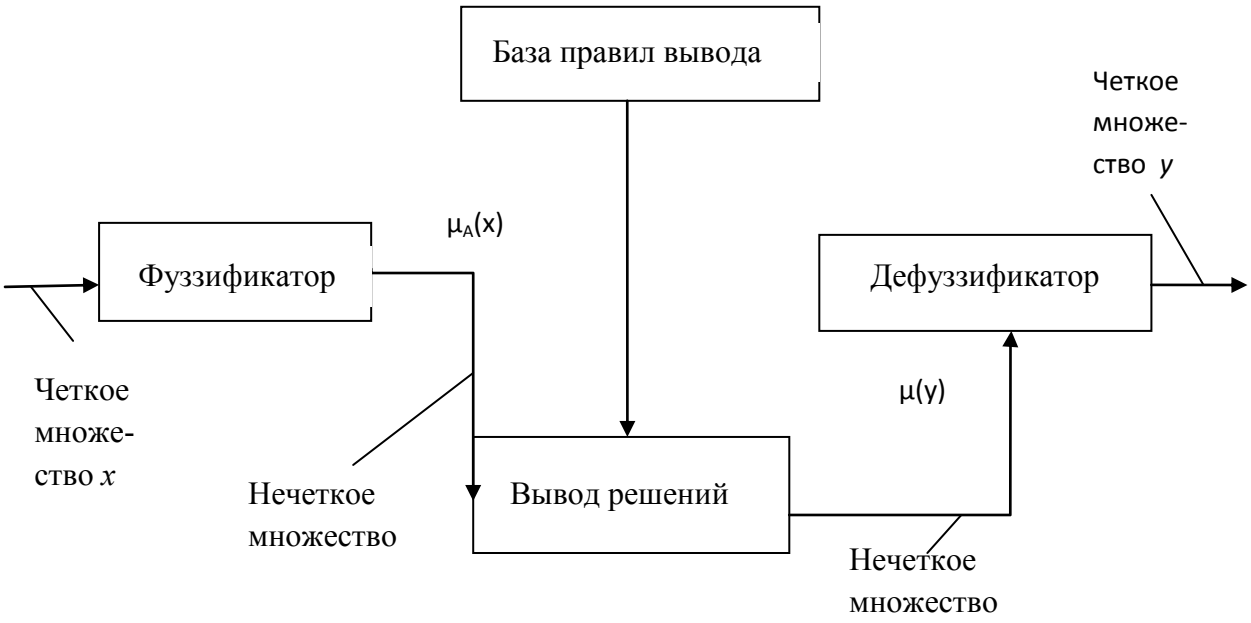


Рисунок 4.1 – Структура нечеткой системы с фуззификатором и дефукззификатором

Фуззификатор преобразует точное множество входных данных в нечеткое множество, определяемое с помощью значений функции принадлежности. Дефуззификатор решает обратную задачу, он формирует однозначное решение относительно значения выходной переменной на основе многих нечетких выводов, вырабатываемых исполнительным модулем нечеткой системы. Выходной сигнал исполнительного модуля имеет вид *M* нечетких множеств, определяющих диапазон изменения выходной переменной. Поскольк-

ку допускается применение множества нечетких правил, в системе предусматривается блок агрегирования, реализуемый в виде логического сумматора (оператора MAX), формирующего единственное нечеткое множество. Дефуззификатор преобразует это нечеткое множество в одно конкретное значение, принимаемое в качестве выходного сигнала системы.

Существует много методов перехода к точным значениям. В методе полной интерпретации точное значение выводимой переменной вычисляется как значение «центра тяжести» функции принадлежности для нечеткого значения. В методе максимума за точное значение выводимой переменной принимается максимальное значение функции принадлежности.

Для многоэкстремальных функций принадлежности чаще всего применяются следующие методы дефуззификации:

- Center Of Gravity (центр тяжести функции принадлежности);
- Mean Of Maximums (центр максимумов);
- First Maximum (максимум функции принадлежности с наименьшей абсциссой).

4.3 Алгоритм реализации нечеткого вывода на примере нечеткой экспертной системы

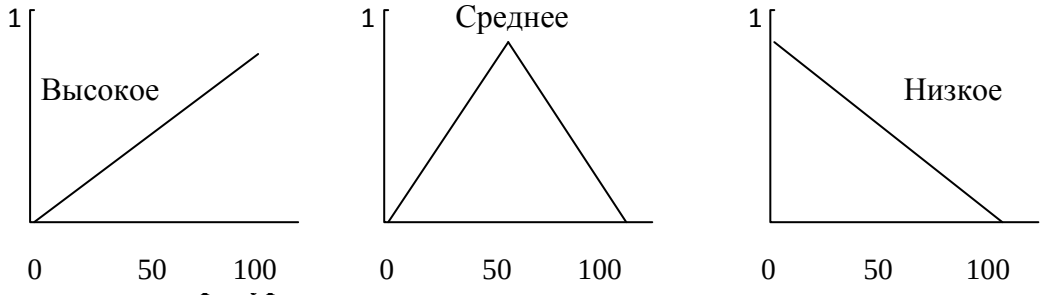
Имеется реактор, описываемый тремя параметрами: температурой, расходом и давлением рабочего вещества. Все показатели измеримы, множество возможных их значений известны. Предположим, что вышел из строя один датчик – измеряющий давление в реакторе. Но знать давление приблизительно желательно. Встает задача отыскания давления по показателям двух работающих датчиков - температуры и расхода рабочего вещества. Известно связи давления со значение двух других величин в виде следующих правил:

- *Если температура низкая, расход малый, то давление низкое;*
- *Если температура средняя, то давление среднее;*
- *Если температура высокая, расход большой, то давление высокое*.

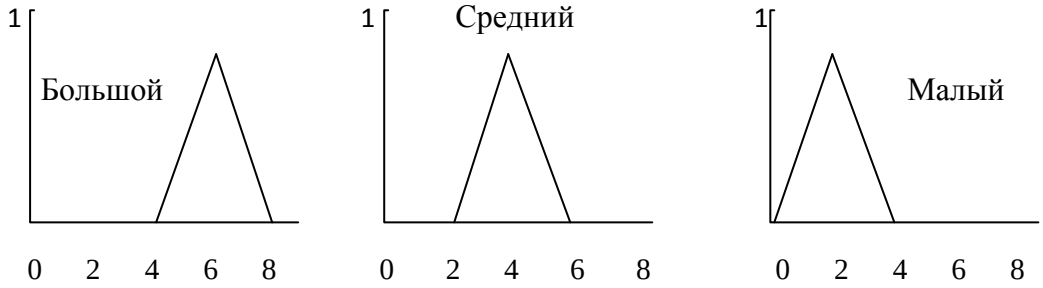
Температура, расход, давление – лингвистические переменные, опишем их функции принадлежности.

Температура – универсум (множество возможных значений) отрезок [0 – 150]. Начальное множество термов {высокая, средняя, низкая}. Функция принадлежности термов имеет вид:

Давление – универсум отрезок [0-100]. Начальное множество термов {высокое, среднее, низкое}. Функция принадлежности термов имеет вид:



Расход – универсум [0 - 8]. Начальное множество термов {большой, средний, малый}. Функция принадлежности термов имеет вид:



Пусть известны значения температуры 85, расхода 3,5. Произведем расчет значения давления. Сначала по значениям параметров температуры и расхода найдем *степени уверенности простейших подтверждения вида:* «Лингвистическая переменная А есть терм лингвистической переменной А». Этот этап называется фаззификацией – переходом от четких значений к степени уверенности по функциям принадлежности. Получаем следующие степени уверенности:

- Температура высокая 0,7;
- Температура средняя 1,0;
- Температура низкая 0,3;
- Расход большой 0;
- Расход средний 0,75;
- Расход малый 0,25.

Затем вычислим *степень уверенности посылок правил*:

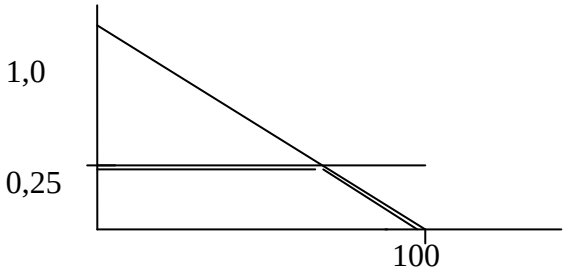
- Температура низкая и расход малый: min(Температура низкая, Расход малый) = min(0,3; 0,25)=0,25;
- Температура средняя: 1,0;
- Температура высокая и расход большой: max(Температура высокая, Расход большой) = max (0,7; 0)=0,7.

В левой части должны стоять либо конъюнкции либо дизъюнкции (при необходимости выполняется преобразование).

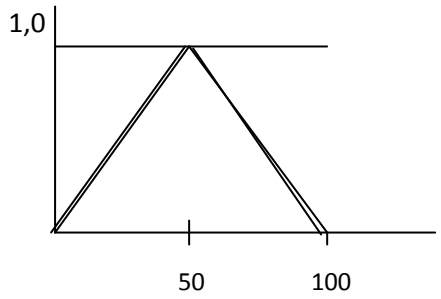
Каждое правило представляет собой нечеткую импликацию (следование). Степень уверенности посылки вычисляется, а степень уверенности заключения задается функцией принадлежности соответствующего терма. Используя способ построения нечеткой импликации получаем новую нечеткую переменную, соответствующую степени уверенности в значении выходных данных при применении к заданным входным соответствующего правила.

Используем определение нечеткой импликации как минимум левой и правой частей (определение Mamdani), имеем:

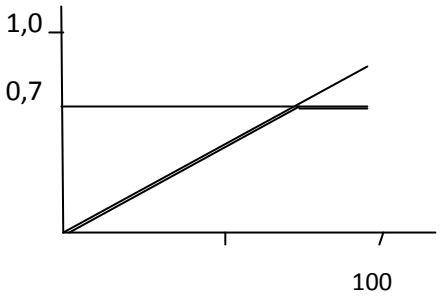
НЕЧЕТКАЯ ИМПЛИКАЦИЯ, СООТВЕТСТВУЮЩАЯ НЕЧЕТКОЙ ПЕРЕМЕННОЙ:
- ДАВЛЕНИЕ НИЗКОЕ (ПРАВИЛО 1)



- ДАВЛЕНИЕ СРЕДНЕЕ (ПРАВИЛО 2)



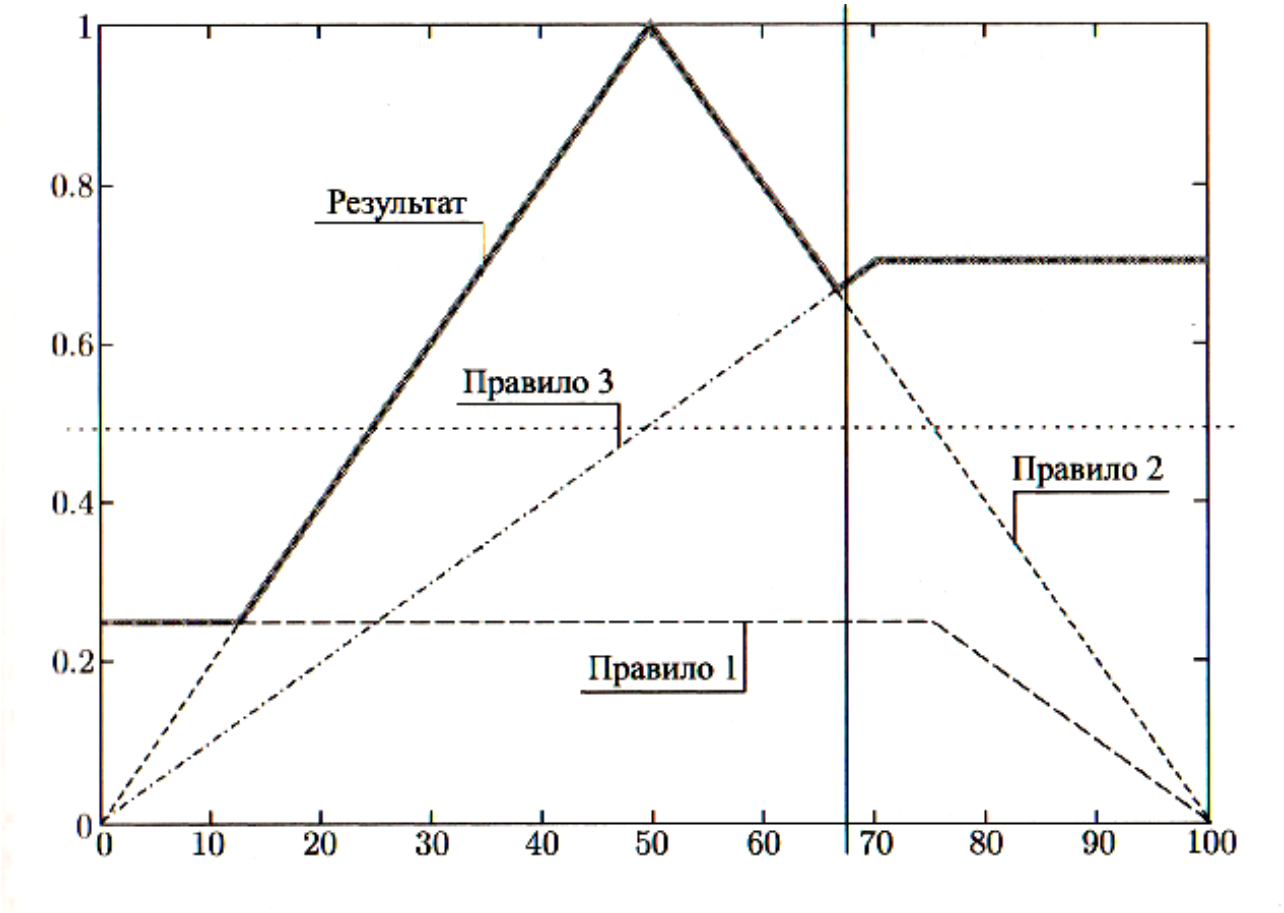
- ДАВЛЕНИЕ ВЫСОКОЕ (ПРАВИЛО 3)



Теперь необходимо объединить результаты применения всех трех правил. Этот этап называется аккумуляцией. Один из способов аккумуляции – построение максимума полученных функций принадлежности. Полученную функцию принадлежности можно считать результатом – это новый терм выходной переменной Давление. Его функция принадлежности говорит о степени уверенности в значении давления при заданных значениях входных параметров и использовании правил, определяющих соотношение входных и выходных переменных.

Для получения конкретного числового значения используется этап дефаззификации, т.е получения конкретного значения из универса по заданной на нем функции при-

надлежности. Используем метод дефаззификации – метод первого максимума. Применяя его к полученной функции принадлежности получаем значение давления, раного 50.



Таким образом, если мы знаем что температура равна 85, а расход рабочего вещества 3,5, то можно сделать вывод, что давление в реакторе равно примерно 50.

Тестовые задания

1. Формулирование задачи достижения нечеткой цели. Содержание задачи.

2. Решение нечеткой задачи подготовки студента к экзамену.

3. Поиск решения задачи, когда отображение $f^{\sim}$ не является тождественным, нечеткие подмножества ограничений $C^{\sim}$ и цели $G^{\sim}$ непосредственно не сравнимы, так как являются подмножествами разных пространств, X и Y соответственно.

4. Разработка математических моделей на основе использования математического аппарата нечетких систем в программе *MATLAB.*

5. Пакет прикладных программ *Fuzzy Logic*. Основные возможности пакета.

6. Эффект «переобучения» нечетких моделей. Способы преодоления этого недостатка.

7. Структура нечеткой системы с фуззификатором и дефукззификатором. Функции отдельных составляющих структуры.

8. Алгоритм реализации нечеткого вывода на примере нечеткой экспертной системы.

9. Алгоритм фаззификации – перехода от четких значений к степени уверенности по функциям принадлежности.

10. Вычисление степени уверенности посылок правил.

11. Построение нечеткой импликации. Оценка степени уверенности посылки.

12. Способ аккумуляции – построение максимума полученных функций принадлежности.

13. Получения конкретного числового значения из функции принадлежности - этап дефаззификации.

Лекция 5 **Основные положения теории искусственных нейронных сетей**

Под нейронными сетями подразумеваются вычислительные структуры, которые моделируют простые биологические процессы, обычно ассоциируемые с процессами человеческого мозга. Они представляют собой распределенные и параллельные системы, способные к адаптивному обучению путем анализа положительных и отрицательных воздействий. Элементарным преобразователем в данных сетях является искусственный нейрон или просто нейрон, названный так по аналогии с биологическим прототипом.

К настоящему времени предложено и изучено большое количество моделей нейроподобных элементов и нейронных сетей, ряд из которых будет рассмотрен ниже.

5.1 Структура и свойства искусственного нейрона

Формальный нейрон (см. рис.5.1.) представляет собой элемент выполняющий взвешенное суммирование значений координат входного вектора с последующим нелинейным преобразованием суммы. Коэффициенты суммирования называются *синаптическими весами*, а нелинейная функция - *функцией активации нейрона*. Такая схема в общих чертах отражает функционирование нейрона головного мозга: функция активации моделирует возбуждение нейрона, а весовые коэффициенты при входных координатах соответствуют обработке в синапсах нейрона. Выходной сигнал формируется на единственном выходе, который в нейрофизиологии называется *аксоном*. Нейроны группируются в нейронные слои и образуют нейронную сеть [14, 15].

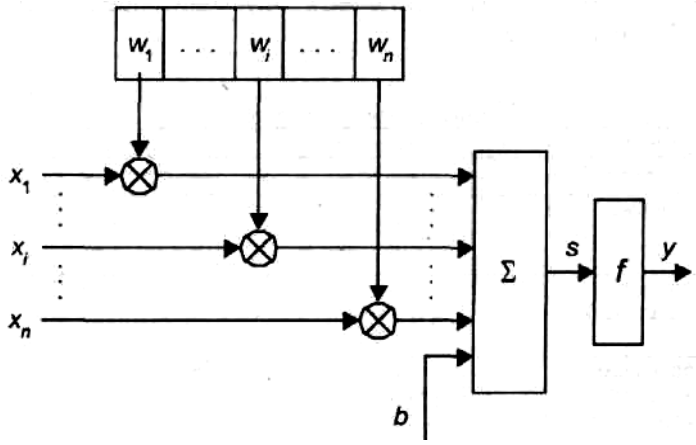


Рисунок 5.1 - Структура искусственного нейрона

Текущее состояние нейрона определяется, как взвешенная сумма его входов:

$$s = \sum_{i=1}^{n} x_i \cdot w_i + b \qquad (5.1)$$

Выход нейрона есть функция его состояния:

$$y = f(s) \qquad (5.2)$$

где w_i - вес (*weight*) синапса, $i = 1 \ldots n$; b - значение смещения (*bias*); s - результат суммирования (*sum*); x_i - компонент входного вектора (входной сигнал), $i = 1 \ldots n$; y - выходной сигнал нейрона; n - число входов нейрона; f - нелинейное преобразование (функция активации).

В общем случае входной сигнал, весовые коэффициенты и смещение могут принимать действительные значения, а во многих практических задачах - лишь некоторые фиксированные значения. Выход (y) определяется видом функции активации и может быть как действительным, так и целым.

Синаптические связи с положительными весами называют *возбуждающими,* с отрицательными весами - *тормозящими.*

Описанный вычислительный элемент можно считать упрощенной математической моделью биологических нейронов. Чтобы подчеркнуть различие нейронов биологических и искусственных, вторые иногда называют *нейроноподобными элементами* или *формальными нейронами.*

На входной сигнал (s) нелинейный преобразователь отвечает выходным сигналом $f(s)$, который представляет собой выход y нейрона.

Простейшим вариантом нейронной сети является *однослойный персептрон Розенблата.* Что же может такая простая схема? Оказывается, уже не так мало – произвести анализ входного вектора данных и распознать среди них вектор, который наилучшим образом соответствует карте синаптических весов. Фактически, это означает решение задачи распознавания. При определенной настройке синаптических весов однослойный пресептрон способен выполнять ряд логических функций, однако не все - для реализации функции исключающего “ИЛИ” - одного слоя принципиально недостаточно. В этом случае используют многослойные нейронные сети.

Многослойные нейронные сети способны с различной степенью точности аппроксимировать любую из существующих логических функций. Качество аппроксимации функций нейронной сетью, очевидно, будет зависеть от числа слоев и числа нейронов в каждом слое. Интуитивно можно предположить, что с увеличением числа слоев и числа нейронов качество нейронной сети улучшится, и это действительно так, но до некоторого предела, после чего увеличение объема становится неэффективным. Уровень порога связан с размерностями входного и выходного вектора, а также с внутренней структурой обрабатываемых данных.

Нелинейная функция *f* называется активационной и может иметь различный вид, как показано на рисунке 5.2. Одной из наиболее распространенных является нелинейная функция с насыщением, так называемая логистическая функция или сигмоид (т.е. функция S-образного вида):

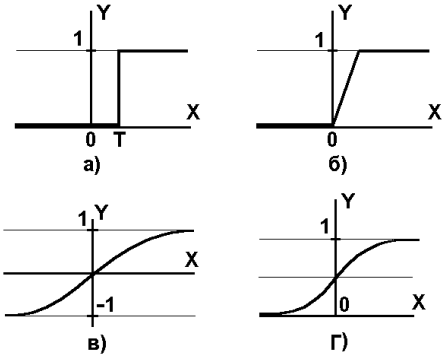


Рисунок 5. 2 – Виды активационных функций

а) функция единичного скачка; б) линейный порог (гистерезис); в) сигмоид – гиперболический тангенс; г) сигмоид – формула (5.3).

$$f(x) = \frac{1}{1+e^{-\alpha x}} \qquad (5.3)$$

При уменьшении α сигмоид становится более пологим, в пределе при α=0 вырождаясь в горизонтальную линию на уровне 0.5, при увеличении α сигмоид приближается по внешнему виду к функции единичного скачка с порогом T в точке x=0. Из выражения для сигмоида очевидно, что выходное значение нейрона лежит в диапазоне [0,1]. Одно из ценных свойств сигмоидной функции – простое выражение для ее производной, применение которого будет рассмотрено в дальнейшем.

$$f'(x) = \alpha \cdot f(x) \cdot (1 - f(x)) \qquad (5.4)$$

Следует отметить, что сигмоидная функция дифференцируема на всей оси абсцисс, что используется в некоторых алгоритмах обучения. Кроме того она обладает свойством усиливать слабые сигналы лучше, чем большие, и предотвращает насыщение от больших сигналов, так как они соответствуют областям аргументов, где сигмоид имеет пологий наклон.

5.2 Классификация нейронных сетей и их свойства

Нейронная сеть представляет собой совокупность нейроподобных элементов, определенным образом соединенных друг с другом и с внешней средой с помощью связей, определяемых весовыми коэффициентами. В зависимости от функций, выполняемых нейронами в сети, можно выделить три их типа [2]:

Таблица 5.1 - Функции активации нейронов [15]:

Название	Формула	Область значений
Линейная	$f(s)=k\,s$	$(-\infty, \infty)$
Полулинейная	$f(s)=\begin{cases} k\,s, & s>0, \\ 0, & s\le 0 \end{cases}$	$(0, \infty)$
Логистическая (сигмоидальная)	$f(s)=\dfrac{1}{1+e^{-as}}$	$(0, 1)$
Гиперболический тангенс (сигмоидальная)	$f(s)=\dfrac{e^{as}-e^{-as}}{e^{as}+e^{-as}}$	$(-1, 1)$
Экспоненциальная	$f(s)=e^{-as}$	$(0, \infty)$
Синусоидальная	$f(s)=\sin(s)$	$(-1, 1)$
Сигмоидальная (рациональная)	$f(s)=\dfrac{s}{a+\lvert s\rvert}$	$(-1, 1)$
Шаговая (линейная с насыщением)	$f(s)=\begin{cases} -1, & s\le -1, \\ s, & -1<s<1, \\ 1, & s\ge 1 \end{cases}$	$(-1, 1)$
Пороговая	$f(s)=\begin{cases} 0, & s<0, \\ 1, & s\ge 0 \end{cases}$	$(0, 1)$
Модульная	$f(s)=\lvert s\rvert$	$(0, \infty)$
Знаковая (сигнатурная)	$f(s)=\begin{cases} 1, & s>0, \\ -1, & s\le 0 \end{cases}$	$(-1, 1)$
Квадратичная	$f(s)=s^2$	$(0, \infty)$

1) *входные нейроны,* на которые подается вектор кодирующий входное воздействие или образ внешней среды; в них обычно не осуществляется вычислительных процедур, а информация передается с входа на выход путем изменения их активации;

2) *выходные нейроны,* выходные значения которых представляют выходы нейронной сети; преобразования в них осуществляются по выражениям (5.1) и (5.2);

3) *промежуточные нейроны,* составляющие основу нейронных сетей, преобразования в которых выполняются также по выражениям (5.1) и (5.2).

С точки зрения топологии можно выделить три основных типа нейронных сетей:

- полносвязные (рис. 5.3, а);
- многослойные или слоистые (рис. 5.3, б);
- слабосвязные (с локальными связями) (рис. 5.3, в).

В *полносвязных нейронных сетях* каждый нейрон передает свой выходной сигнал остальным нейронам, в том числе и самому себе. Все входные сигналы подаются всем нейронам. Выходными сигналами сети могут быть все или некоторые выходные сигналы нейронов после нескольких тактов функционирования сети.

В *многослойных нейронных сетях* нейроны объединяются в слои. Слой содержит совокупность нейронов с едиными входными сигналами. Число нейронов в слое может быть любым и не зависит от количества нейронов в других слоях. В общем случае сеть состоит из Q слоев, пронумерованных слева направо. Внешние входные сигналы подаются на входы нейронов входного слоя (его часто нумеруют как нулевой), а выходами сети являются выходные сигналы последнего слоя. Кроме входного и выходного слоев в многослойной нейронной сети есть один или несколько скрытых слоев. Связи от выходов нейронов некоторого слоя q к входам нейронов следующего слоя (q+1) называются последовательными.

В свою очередь, среди многослойных нейронных сетей выделяют следующие типы:

1) *Монотонные.* Это частный случай слоистых сетей с дополнительными условиями на связи и нейроны. Каждый слой кроме последнего (выходного) разбит на два блока: возбуждающий и тормозящий. Связи между блоками тоже разделяются на тормозящие и возбуждающие. Если от нейронов блока A к нейронам блока B ведут только возбуждающие связи, то это означает, что любой выходной сигнал блока является монотонной неубывающей функцией любого выходного сигнала блока A. Если же эти связи только тормозящие, то любой выходной сигнал блока B является невозрастающей функцией любого выходного сигнала блока A. Для нейронов монотонных сетей необходима монотонная зависимость выходного сигнала нейрона от параметров входных сигналов.

2) *Сети без обратных связей.* В таких сетях нейроны входного слоя получают входные сигналы, преобразуют их и передают нейронам первого скрытого слоя, и так далее вплоть до выходного, который выдает сигналы для интерпретатора и пользователя. Если не оговорено противное, то каждый выходной сигнал q-го слоя подастся на вход всех нейронов (q+1)-го слоя; однако возможен вариант соединения Q-го слоя с произвольным $(q+p)$-м слоем.

Среди многослойных сетей без обратных связей различают полносвязанные (выход каждого нейрона q-го слоя связан с входом каждого нейрона $(q+1)$-го слоя) и частично полносвязанные. Классическим вариантом слоистых сетей являются полносвязанные сети прямого распространения.

3) *Сети с обратными связями.* В сетях с обратными связями информация с последующих слоев передается на предыдущие. Среди них, в свою очередь, выделяют следующие:

• слоисто-циклические, отличающиеся тем, что слои замкнуты в кольцо: последний слой передает свои выходные сигналы первому; все слои равноправны и могут как получать входные сигналы, так и выдавать выходные;

• слоисто-полносвязанные состоят из слоев, каждый из которых представляет собой полносвязную сеть, а сигналы передаются как от слоя к слою, так и внутри слоя; в каждом слое цикл работы распадается на три части: прием сигналов с предыдущего слоя, обмен сигналами внутри слоя, выработка выходного сигнала и передача к последующему слою;

• полносвязанно-слоистые, по своей структуре аналогичные слоисто-полносвязанным, но функционирующим по-другому: в них не разделяются фазы обмена внутри слоя и передачи следующему, на каждом такте нейроны всех слоев принимают сигналы от нейронов как своего слоя, так и последующих.

В *слабосвязных нейронных сетях* нейроны располагаются в узлах прямоугольной или гексагональной решетки. Каждый нейрон связан с четырьмя (окрестность фон Неймана), шестью (окрестность Голея) или восемью (окрестность Мура) своими ближайшими соседями.

Известные нейронные сети можно разделить по типам структур нейронов на гомогенные (однородные) и гетерогенные. Гомогенные сети состоят из нейронов одного типа с единой функцией активации, а в гетерогенную сеть входят нейроны с различными функциями активации.

Существуют бинарные и аналоговые сети. Первые из них оперируют только двоичными сигналами, и выход каждого нейрона может принимать значение либо логического ноля (заторможенное состояние) либо логической единицы (возбужденное состояние).

Еще одна классификация делит нейронные сети на синхронные и асинхронные. В первом случае в каждый момент времени лишь один нейрон меняет свое состояние, во втором - состояние меняется сразу у целой группы нейронов, как правило, у всего слоя. Алгоритмически ход времени в нейронных сетях задается итерационным выполнением однотипных действий над нейронами. Далее будут рассматриваться только синхронные сети.

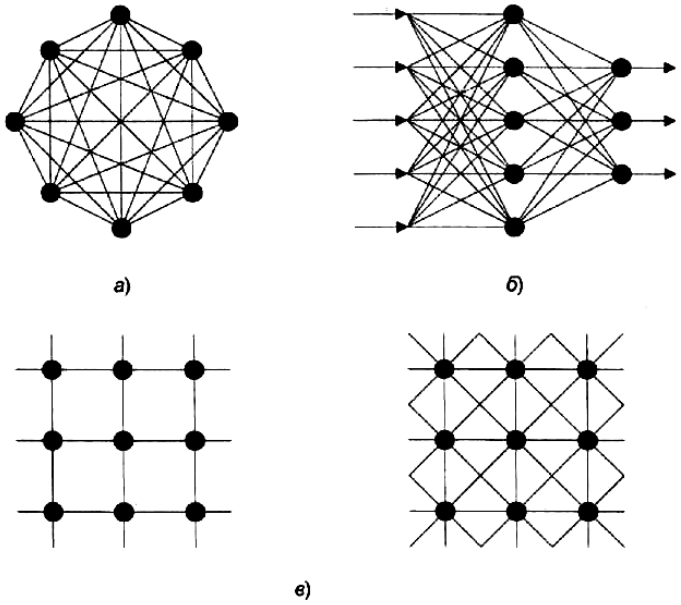


Рисунок 5.3 - Архитектуры нейронных сетей:
а- полносвязная сеть, б- многослойная сеть с последовательными связями,
в - слабосвязные сети

5.3 Алгоритмы обучения нейронных сетей

Процесс обучения нейронной сети может рассматриваться как настройка архитектуры и весов связей для эффективного выполнения поставленной задачи. Обычно итеративная настройка весов связей осуществляется в соответствии с обучающей выборкой. Свойство сети обучаться на примерах делает их более привлекательными по сравнению с системами, которые следуют системе правил функционирования, сформулированной экспертами [16].

Для организации процесса обучения, во-первых, надо иметь модель внешней среды, в которой функционирует нейронная сеть. Эта модель определяет парадигму обучения. Во-вторых, необходимо понять, как модифицировать весовые параметры сети.

Существуют три парадигмы обучения [16]: с *учителем, без учителя* (самообучение) и *смешанная.* В первом случае на каждый входной пример существует требуемый ответ. Веса настраиваются таким образом, чтобы выходы сети как можно более близкие к требуемым ответам. Более «жесткий» вариант обучения с учителем предполагает, что известна только критическая оценка правильности выхода нейронной сети, а не сами требуемые значения выхода.

Для обучения без учителя не нужно знания требуемых ответов на каждый пример обучающей выборки. В этом случае происходит распределение образцов по категориям

(кластерам) в соответствии с внутренней структурой данных или степенью корреляции между образцами.

При смешанном обучении весовые коэффициенты одной группы нейронов настраиваются посредством обучения с учителем, а другой группы - на основе самообучения.

В процессе обучения учитываются следующие свойства нейронных сетей [16]: емкость сети, сложность образцов и вычислительная сложность. Под емкостью сети понимается число запоминаемых образцов, с учетом сформированных функций и границ принятия решений. Сложность образцов определяет число обучающих примеров, необходимых для достижения способности сети к обобщению.

Известны четыре основных правила обучения, обусловленные связанными с ними архитектурами сетей (табл.5.2): *коррекция ошибки, правило Больцмана, правило Хебба* и *метод соревнования* [15].

Коррекция ошибки. Для каждого входного примера задан требуемый выход d, который может не совпадать с реальным y. Правило обучения при коррекции по ошибке состоит в использовании разницы (d - y) для изменения весов, с целью уменьшения ошибки рассогласования. Обучение производится только в случае ошибочного результата. Известны многочисленные модификации этого правила обучения.

Правило Больцмана. Правило Больцмана является стохастическим правилом обучения, обусловленным аналогией с термодинамическими принципами. В результате его выполнения осуществляется настройка весовых коэффициентов нейронов в соответствии с требуемым распределением вероятностей. Обучение правилу Больцмана может рассматриваться как отдельный случай коррекции по ошибке, в котором под ошибкой понимается расхождение корреляций состояний в двух режимах.

Правило Хебба. Правило Хебба является самым известным алгоритмом обучения нейронных сетей, суть которого заключается в следующем: если нейроны с обеих сторон синапса возбуждаются одновременно и регулярно, то сила синаптической связи возрастает. Важной особенностью является то, что изменение синаптического веса зависит только от активности связанных этим синапсом нейронов. Предложено большое количество разновидностей этого правила, различающихся особенностями модификации синаптических весов.

Метод соревнования. В отличие от правила Хебба, в котором множество выходных нейронов могут возбуждаться одновременно, здесь выходные нейроны соревнуются между собой. И выходной нейрон с максимальным значением взвешенной суммы является «победителем» («победитель забирает все»). Выходы же остальных выходных нейронов уста-

навливаются в неактивное состояние. При обучении модифицируются только веса нейрона-«победителя» в сторону увеличения близости к данному входному примеру. Это правило позволяет группировать входные данные на категории (кластеры) и представлять их отдельными выходными нейронами.

Таблица 5.2 - Известные алгоритмы обучения

Парадигма	Обучающее правило	Архитектура нейронной сети	Алгоритм обучения	Задачи
С учителем	Коррекция ошибки	Однослойный и многослойный персептрон	Алгоритмы обучения персептрона Обратное распространение	Классификация образов; Аппроксимация функций предсказание; управление
	Больцман	Рекуррентная	Алгоритм обучения Больцмана	Классификация образов
	Хебб	Многослойная прямого распространения	Линейный дискри-ми-нантный анализ	Анализ данных; классификация образов
	Соревнование	Соревнование	Векторное квантование	Категоризация внутри класса; сжатие данных
		Сеть ART	ART Map	Классификация образов
Без учителя	Коррекция ошибки	Многослойная прямого распространения	Проекция Саммона	Категоризация внутри класса; анализ данных
	Хебб	Прямого распространения или соревнование	Метод главных компонентов	Анализ данных; сжатие данных
		Сеть Хопфилда	Обучение ассоциативной памяти	Ассоциативная память
	Соревнование	Соревнование	Векторное квантование	Категоризация; сжатие данных
		SOM Кохонена	SOM Кохонена	Категоризация; анализ данных
		Сети ART	ART1, ART2	Категоризация
Смешанная	Коррекция ошибки и соревнование	Сеть RBFN	Алгоритм обучения RBFN	Классификация образов; аппроксимация функций; предсказание; управление

Нейронная сеть считается устойчивой, если после конечного числа итераций обучения ни один из примеров обучающей выборки не изменяет своей принадлежности в кластерах. Однако сеть не перестанет обучаться, если параметр скорости обучения не равен нулю. Но эта искусственная остановка обучения вызывает другую проблему, называемую пластичностью и связанную со способностью сети к адаптации к новым данным. Возникает дилемма стабильности-пластичности Гроссберга.

Список представленных в табл. 5.3 алгоритмов обучения нейронных сетей не является исчерпывающим. В последней колонке перечислены задачи, для которых может быть применены эти алгоритмы. Каждый алгоритм обучения ориентирован на сеть определенной архитектуры и предназначен для ограниченного класса задач.

На основе данной таблицы для решения поставленной задачи прогнозирования, были выбраны следующие виды сетей:

- сеть радиального основания (*RBFN*)
- многослойный перцептрон (*MPL*)
- однослойный перцептрон (*Leniar*)

Далее они будут рассмотрены более подробно.

5.4 Обратное распространение (*Neural Network with Back Propagation Training Algorithm*) [2]

1) Название. Нейронная сеть с обучением по методу обратного распространения ошибки.

Другие названия. Backprop. Back Propagation (*Neural*) *Network.*

Сеть обратного распространения (ошибки). *MultiLayer Perceptron with Back Propagation Training Algorithm* (многослойный персептрон с обучением по методу обратного распространения ошибки).

2) Авторы и история создания. Многослойные персептроны были предложены и исследованы в 1960-х годах Розенблаттом, Минским, Пейпертом и др. Лишь в середине 1980-х годов был предложен эффективный алгоритм обучения многослойных персептронов, основанный на вычислении градиента функции ошибки и названный обратным распространением ошибки.

3) Модель. Алгоритм обратного распространения - это итеративный градиентный алгоритм, который используется с целью минимизации среднеквадратического отклонения текущего выхода многослойного персептрона и требуемого выхода. Он используется для обучения многослойных нейронных сетей с последовательными связями. Нейроны в таких сетях делятся на группы с общим входным сигналом - слои. На каждый нейрон пер-

вого слоя подаются все элементы внешнего входного сигнала. Все выходы нейронов q-го слоя подаются на каждый нейрон слоя (q+1). Нейроны выполняют взвешенное суммирование входных сигналов. К сумме элементов входных сигналов, помноженных на соответствующие синаптические веса, прибавляется смещение нейрона. Над результатом суммирования выполняется нелинейное преобразование - функция активации (передаточная функция). Значение функции активации есть выход нейрона.

4) *Характеристики сети*. Тип входных сигналов - целые или действительные. Тип выходных сигналов - действительные из интервала, заданного передаточной функцией нейронов. Тип передаточной функции - сигмоидальная. В нейронных сетях применяются несколько вариантов сигмоидальных передаточных функций.

Функция Ферми (экспоненциальная сигмоида):

$$f(s) = \frac{1}{1+e^{-2as}}$$

где s- выход сумматора нейрона, a - некоторый параметр.

Рациональная сигмоида:

$$f(s) = \frac{1}{|s|+a}$$

Гиперболический тангенс:

$$f(s) = th\left(\frac{s}{a}\right) = \frac{e^{\frac{s}{a}} - e^{-\frac{s}{a}}}{e^{\frac{s}{a}} + e^{-\frac{s}{a}}}$$

Перечисленные функции относятся к однопараметрическим. Значение функции зависит от аргумента и одного параметра. Используются также и многопараметрические передаточные функции, например:

$$f(s) = p_1 \frac{s}{|s|+p_2} + p_3$$

Сигмоидальные функции являются монотонно возрастающими и имеют отличные от нуля производные на всей области определения. Эти характеристики обеспечивают правильное функционирование и обучение сети.

Наиболее эффективной передаточной функцией является рациональная сигмоида. Для вычисления гиперболического тангенса требуются большие вычислительные затраты.

Функционирование многослойной нейронной сети осуществляется в соответствии с формулами:

$$s_{i_q} = \sum_{i_{q-1}}^{N_{q-1}} w_{i_q i_{q-1}} w_{i_{q-1}} - b_{i_q}, \quad i_q = 1,\ldots, N_q, \quad m = 1,\ldots, L,$$

$$y_{i_q} = f(s_{i_q}), \quad i_q = 1,\ldots, N_q, \quad m = 1,\ldots, L,$$

где *s* - выход сумматора; *w* - вес связи; *y* - выход нейрона; *b* – смещение; *i* - номер нейрона; *N* - число нейронов в слое; *m* - номер слоя; *L* - число слоев; *f*- функция активации.

Обучение сети разбивается на следующие шаги:

ШАГ 1. Инициализация сети**.** Весовым коэффициентам и смещениям сети присваиваются малые случайные значения из установленных диапазонов.

ШАГ 2. Определение элемента обучающей выборки: (<Текущий вход>, <требуемый выход>). Текущие входы (x_1, ..., x_N), должны различаться для всех элементов обучающей выборки. При использовании многослойного персептрона в качестве классификатора требуемый выходной сигнал (d_1, ..., d_N) состоит из нулей за исключением одного единичного элемента, соответствующего классу, к которому принадлежит текущий входной сигнал.

ШАГ 3. Вычисление текущего выходного сигнала. Текущий выходной сигнал определяется в соответствии с традиционной схемой функционирования многослойной нейронной сети.

ШАГ 4. Настройка синаптических весов. Для настройки весовых коэффициентов используется рекурсивный алгоритм, который сначала применяется к выходным нейронам сети, а затем проходит сеть в обратном направлении до первого слоя. Синаптические веса настраиваются в соответствии с формулой:

$$w_{ij} = (t+1) = w_{ij}(t) + \eta \delta_j x_i,$$

где $w_{ij}(t)$ - вес от нейрона *i* или от элемента входного сигнала *i* к нейрону *j* в момент времени *t;* x_i, - выход нейрона *i* или *i*-й элемент входного сигнала, η - коэффициент скорости обучения; δ_j - значение ошибки для нейрона j.

Если нейрон с номером *j* принадлежит последнему слою, то:

$$\delta_j = y_j(1-y_j)(d_j - y_j),$$

где d_j, y_j - соответственно требуемый и текущий выход *j*-го нейрона.

Если *j*-й нейрон принадлежит одному из слоев с первого по предпоследний, то:

$$\delta_j = x_j(1-x_j)\sum_k \delta_k y_{jk},$$

где *j*-й нейрон принадлежит предыдущему слою, а индекс *k* пробегает все нейроны последующего слоя.

Смещения нейронов *b* настраиваются аналогичным образом.

5) *Области применения*. Распознавание образов, классификация, прогнозирование, синтез речи, контроль, адаптивное управление, построение экспертных систем.

6) *Недостатки*. Многокритериальная задача оптимизации в методе обратного распространения рассматривается как набор однокритериальных - на каждой итерации происходят изменения значений параметров сети, улучшающие работу лишь с одним примером обучающей выборки. Такой подход существенно уменьшает скорость обучения.

Классический метод обратного распространения относится к алгоритмам с линейной сходимостью. Для увеличения скорости сходимости необходимо использовать матрицы вторых производных функции ошибки.

7) *Преимущества.* Обратное распространение - первый эффективный алгоритм обучения многослойных нейронных сетей. Один из самых популярных алгоритмов обучения, с его помощью решены и решаются многочисленные практические задачи.

8) *Модификации*. Модификации алгоритма обратного распространения связаны с использованием различных функций ошибки, различных процедур определения направления и величины шага. Функции ошибки:

• интегральные функции ошибки по всей совокупности обучающих примеров;

• функции ошибки целых и дробных степеней; процедуры определения величины шага на каждой итерации:

• дихотомия;

• инерционные соотношения, например,

$$w_{ij} = (t+1) = w_{ij}(t) + \eta\delta_j x_i + \alpha\left(w_{ij}(t) - w_{ij}(t-1)\right),$$

где α - некоторое положительное число, меньше единицы;

• отжиг;.

Процедуры определения направления шага:

• с использованием матрицы производных второго порядка (метод Ньютона и др.);

• с использованием направлений на нескольких шагах (партан метод и др.).

5.5 Сеть радиального основания (*Radial Basis Function Network*) [14, 17]

1) *Название. Radial Basis Function Network* (*RBFN*).

Другое название. Сеть радиальных базисных функций.

2) *Авторы и история создания.* Под парадигмой *RBFN* понимается архитектура, предложенная *Moody* и *Darke*n в 1989 г. К классу *RBFN* относят также вероятностные и регрессионные нейронные сети.

3) Модель. В общем случае под термином *Radial Basis Function Network* понимается нейронная сеть, которая содержит слой скрытых нейронов с радиально симметричной активационной функцией, каждый из которых предназначен для хранения отдельного эталонного вектора (в виде вектора весов). Для построения *RBFN* необходимо выполнение следующих условий:

Во-первых, наличие эталонов, представленных в виде весовых векторов нейронов скрытого слоя.

Во-вторых, наличие способа измерения расстояния входного вектора от эталона. Обычно это стандартное евклидово расстояние.

В-третьих, специальная функция активации нейронов скрытого слоя, задающая выбранный способ измерения расстояния. Обычно используется функция Гаусса, существенно усиливающая малую разницу между входным и эталонным векторами.

Другими словами, выходной сигнал эталонного нейрона скрытого слоя y_i, - это функция (гауссиан) только от расстояния ρ_i, между входным и эталонным векторами:

$$\rho_i = \sqrt{\sum_{j=1}^{N} (x_j - cij)^2}, \qquad y_i = \exp\left[-\left(\frac{\rho_i - R}{\sigma_i}\right)^2\right], \qquad i=1,\ldots,L,$$

где $X = (x_1, \ldots x_N)$ - входной вектор; $C_i = (c_{i1}, \ldots, c_{iN})$ - весовой вектор i-го эталонного нейрона скрытого слоя; R, σ_i - параметры активационной функции; L - число эталонов.

Обучение слоя образцов-нейронов сети подразумевает предварительное проведение кластеризации для нахождения эталонных векторов и определенных эвристик для определения значений σ_i.

Нейроны скрытого слоя соединены по полносвязной схеме с нейронами выходного слоя, которые осуществляют взвешенное суммирование. Для нахождения значения весов от нейронов скрытого к выходному слою используется линейная регрессия.

В общем случае активационные функции нейронов скрытого слоя могут отражать законы распределения случайных величин (вероятностные нейронные сети) либо характеризовать различные аналитические зависимости между переменными (регрессионные нейронные сети).

Поверхность отклика радиального элемента представляет собой гауссову функцию (колоколообразной формы), с вершиной в центре и понижением к краям.

4) Области применения. Распознавание образов, классификация.

5) Недостатки. Заранее должно быть известно число эталонов, а также эвристики для построения активационных функций нейронов скрытого слоя.

6) Преимущества. Отсутствие этапа обучения в принятом смысле этого слова.

7) Модификации.

5.6 Сравнение сетей *RBF* с *MPL* [15, 17]

Элемент многослойного персептрона полностью задается значениями своих весов и порогов, которые в совокупности определяют уравнение разделяющей прямой и скорость изменения функции при отходе от этой линии. До действия сигмоидной функции активации уровень активации такого элемента определяется гиперплоскостью, поэтому в системе *ST Neural Networks* такие элементы называется линейными (хотя функция активации, как правило, нелинейна).

В отличие от них, радиальный элемент задается своим центром и "радиусом". Положение точки в *N*-мерном пространстве определяется *N* числовыми параметрами, т.е. их ровно столько же, сколько весов у линейного элемента, и поэтому координаты центра радиального элемента в пакете *ST Neural Networks* хранятся как "веса". Его радиус (отклонение) хранится как "порог".

Следует отчетливо понимать, что "веса" и "пороги" радиального элемента принципиально отличаются от весов и порогов линейного элемента, и если забыть об этом, термин может ввести в заблуждение. Радиальные веса на самом деле представляют точку, а радиальный порог - отклонение.

Сеть типа радиальной базисной функции (*RBF*) имеет промежуточный слой из радиальных элементов, каждый из которых воспроизводит гауссову поверхность отклика. Поскольку эти функции нелинейны, *для моделирования произвольной функции нет необходимости брать более одного промежуточного слоя*. Для моделирования любой функции необходимо лишь взять достаточное число радиальных элементов. Остается решить вопрос о том, как следует скомбинировать выходы скрытых радиальных элементов, чтобы получить из них выход сети. Оказывается, что достаточно взять их линейную комбинацию (т.е. взвешенную сумму гауссовых функций). Сеть *RBF* имеет выходной слой, состоящий из элементов с линейными функциями активации.

Сети *RBF* имеют ряд преимуществ перед сетями *MLP*. Во-первых, как уже сказано, они моделируют произвольную нелинейную функцию с помощью всего одного промежу-

точного слоя, и тем самым избавляют от необходимости решать вопрос о числе слоев. Во-вторых, параметры линейной комбинации в выходном слое можно полностью оптимизировать с помощью хорошо известных методов линейного моделирования, которые работают быстро и не испытывают трудностей с локальными минимумами, так мешающими при обучении *MLP*. Поэтому сеть *RBF* обучается очень быстро (на порядок быстрее *MLP*).

С другой стороны, до того, как применять линейную оптимизацию в выходном слое сети *RBF*, необходимо определить число радиальных элементов, положение их центров и величины отклонений. Соответствующие алгоритмы, хотя и работают быстрее алгоритмов обучения *MLP*, в меньшей степени пригодны для отыскания субоптимальных решений. В качестве компенсации, *Автоматический конструктор сети пакета ST Neural Networks* сможет выполнить за пользователя все необходимые действия по экспериментированию с сетью.

Другие отличия работы *RBF* от *MLP* связаны с различным представлением пространства модели: "групповым" в *RBF* и "плоскостным" в *MLP*.

Опыт показывает, что для правильного моделирования типичной функции сеть *RBF*, с ее более эксцентричной поверхностью отклика, требует несколько большего числа элементов. Конечно, можно специально придумать форму поверхности, которая будет хорошо представляться первым или, наоборот, вторым способом, но общий итог оказывается не в пользу *RBF*. Следовательно, *модель, основанная на RBF, будет работать медленнее и потребует больше памяти, чем соответствующий MLP* (однако она гораздо быстрее обучается, а в некоторых случаях это важнее).

С "групповым" подходом связано и неумение сетей *RBF* экстраполировать свои выводы за область известных данных. При удалении от обучающего множества значение функции отклика быстро спадает до нуля. Напротив, сеть *MLP* выдает более определенные решения при обработке сильно отклоняющихся данных. Достоинство это или недостаток - зависит от конкретной задачи, однако в целом склонность *MLP к некритическому экстраполированию результата считается его слабостью*. Экстраполяция на данные, лежащие далеко от обучающего множества, - вещь, как правило, опасная и необоснованная.

Сети *RBF* более чувствительны к "проклятию размерности" и испытывают *значительные трудности, когда число входов велико*. Мы обсудим этот вопрос ниже.

Как уже говорилось, обучение *RBF*-сети происходит в несколько этапов. Сначала определяются центры и отклонения для радиальных элементов; после этого оптимизируются параметры линейного выходного слоя.

Расположение центров должно соответствовать кластерам, реально присутствующим в исходных данных. Рассмотрим два наиболее часто используемых метода.

*Выборка из выборки***.** В качестве центров радиальных элементов берутся несколько случайно выбранных точек обучающего множества. В силу случайности выбора они "представляют" распределение обучающих данных в статистическом смысле. Однако, если число радиальных элементов невелико, такое представление может быть неудовлетворительным (*Haykin*, 1994).

*Алгоритм K-средних***.** Этот алгоритм (*Bishop*, 1995) стремится выбрать оптимальное множество точек, являющихся центроидами кластеров в обучающих данных. При *K* радиальных элементах их центры располагаются таким образом, чтобы:

- Каждая обучающая точка "относилась" к одному центру кластера и лежала к нему ближе, чем к любому другому центру;
- Каждый центр кластера был центроидом множества обучающих точек, относящихся к этому кластеру.

После того, как определено расположение центров, нужно найти отклонения. Величина отклонения (ее также называют сглаживающим фактором) определяет, насколько "острой" будет гауссова функция. Если эти функции выбраны слишком острыми, сеть не будет интерполировать данные между известными точками и потеряет способность к обобщению. Если же гауссовы функции взяты чересчур широкими, сеть не будет воспринимать мелкие детали. На самом деле сказанное - еще одна форма проявления дилеммы пере/недообучения. Как правило, отклонения выбираются таким образом, чтобы колпак каждой гауссовой функций захватывал "несколько" соседних центров. Для этого имеется несколько методов:

Явный. Отклонения задаются пользователем.

Изотропный. Отклонение берется одинаковым для всех элементов и определяется эвристически с учетом количества радиальных элементов и объема покрываемого пространства (*Haykin*, 1994).

*K ближайших соседей***.** Отклонение каждого элемента устанавливается (индивидуально) равным среднему расстоянию до его *K* ближайших соседей (*Bishop*, 1995). Тем самым отклонения будут меньше в тех частях пространства, где точки расположены густо, - здесь будут хорошо учитываться детали, - а там, где точек мало, отклонения будут большими (и будет производится интерполяция).

После того, как выбраны центры и отклонения, параметры выходного слоя оптимизируются с помощью стандартного метода линейной оптимизации - алгоритма псевдообратных матриц (сингулярного разложения) (*Haykin*, 1994; *Golub and Kahan*, 1965).

Могут быть построены различные гибридные разновидности радиальных базисных функций. Например, выходной слой может иметь нелинейные функции активации, и тогда для его обучения используется какой-либо из алгоритмов обучения многослойных персептронов, например метод обратного распространения. Можно также обучать радиальный (скрытый) слой с помощью алгоритма обучения сети Кохонена - это еще один способ разместить центры так, чтобы они отражали расположение данных.

Тестовые задания

1. Структура и свойства искусственного нейрона. Нейронная сеть.
2. Варианты нейронных сетей. Однослойные и многослойные сети.
3. Виды активационных функций, их характеристики.
4. Типы многослойных нейронных сетей и их характеристики.
5. Алгоритмы обучения нейронных сетей. Организация процесса обучения.
6. Сеть обратного распространения. Алгоритм обратного распространения. Характеристика сети.
7. Сеть радиального основания. Области применения.
8. Сравнение характеристик сетей *RBF* и *MPL*, область применения.

Лекция **6 Построение и обучение моделей на нейронных сетях**

Программу моделирования нейронной сети обычно называют программой-имитатором или нейропакетом, понимая под этим программную оболочку, эмулирующую для пользователя среду нейрокомпьютера на обычном компьютере.

В настоящее время на рынке программного обеспечения имеется множество самых разнообразных программ для моделирования нейронных сетей. Можно выделить несколько основных функций, которые реализованы во всех этих программах.

6.1 Создание нейронной сети

Для решения разных практических задач требуются различные модели нейронных сетей. Модель нейронной сети определяется моделями нейронов и структурой связей сети.

Программы-имитаторы в зависимости от структуры связей реализуют следующие группы нейронных сетей [17].

- *Многослойные нейронные сети.* Нейроны в таких сетях делятся на группы с общим входным сигналом - слои. Различают несколько типов связей между слоями с номерами q и $(q + p)$:
  - последовательные ($p = 1$);
  - прямые ($p > 1$);
  - обратные ($p < 0$).

  Связи между нейронами одного слоя называют латеральными (боковыми).
- *Полносвязные нейронные сети.* Каждый нейрон в полносвязных сетях связан со всеми остальными. На каждом такте функционирования сети на входы нейронов подается внешний входной сигнал и выходы нейронов предыдущего такта.
- *Нейронные сети с локальными связями.* Нейроны в таких сетях располагаются в узлах прямоугольной или гексагональной решетки. Каждый нейрон связан с небольшим числом (4, 6 или 8) своих топологических соседей.
- *Неструктурированные нейронные сети.* К этой группе относятся все модели нейронных сетей, которые нельзя отнести ни к одной из предыдущих групп.

Модели реализуемых программами-имитаторами нейронов чрезвычайно разнообразны. В простейшем случае нейроны первого порядка выполняет взвешенное суммирование компонентов входного вектора и нелинейное преобразование результата суммирования. Нейроны более высоких порядков осуществляют перемножение двумерных матриц и многомерных тензоров.

В моделях нейронов используются различные варианты нелинейных преобразований. Наиболее часто используются сигмоидальные, кусочно-линейные и жесткие пороговые функции активации. В сети все нейроны могут иметь как одинаковые (гомогенная сеть), так и различные функции активации (гетерогенная сеть).

Для построения нейронной сети, ориентированной на решение конкретной задачи, используются процедуры формирования нейронных сетей, которые обеспечивают ввод указанных характеристик моделей нейронов и структур нейронных сетей [18].

Каждая группа моделей нейронных сетей может быть использована для решения лишь некоторого ограниченного класса задач. Так, многослойные и полносвязные нейронные сети с сигмоидальными передаточными функциями используются для распознавания образов и адаптивного управления; нейронные сети с локальными связями - для обработки изображений и некоторых других частных задач. Для решения задач линейной алгебры используются многослойные сети с особыми передаточными функциями.

Лишь для небольшого числа моделей нейронных сетей существует строгое математическое обоснование возможности их применения для решения конкретных задач. В наибольшей степени теоретически проработаны двухслойные нейронные сети с сигмоидальными передаточными функциями.

6.2 Обучение нейронной сети

Для процесса обучения нейронных сетей выберем метод обучения «с учителем», который заключается в последовательном предъявлении примеров значений входных переменных (рисунок 6.1)[15,16].

При предъявлении примера на входе сеть выдает некоторый ответ, не обязательно верный. Известен и верный (желаемый) ответ. В данном случае желательно, чтобы на выходе соответствующего нейрона уровень сигнала был максимален.

После многократного предъявления примеров веса сети стабилизируются, причем сеть дает правильные ответы на все (или почти все) примеры из базы данных. В таком случае говорят, что сеть обучена. В программных реализациях можно видеть, что в процессе обучения величина ошибки (сумма квадратов ошибок по всем выходам) постепенно уменьшается. Когда величина ошибки достигает нуля или приемлемо малого уровня, обучение останавливают, и сеть готова к распознаванию.

Важно отметить, что вся информация, которую сеть приобретает о задаче, содержится в наборе примеров. Поэтому качество обучения сети зависит от количества примеров в обучающей выборке, а также от того, насколько полно эти примеры опи-

сывают задачу. Считается, что для полноценной тренировки требуется хотя бы несколько десятков (а лучше сотен) примеров.

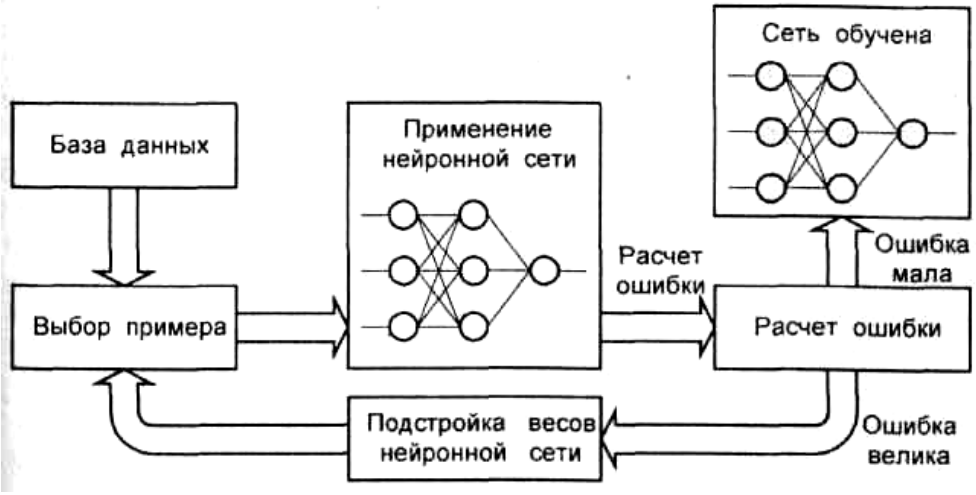


Рисунок 6.1 - Процесс обучения нейронной сети

При обучении нейронных сетей, как правило, используются следующие критерии останова:

- при достижении некоторого малого значения функции ошибки;
- в случае успешного решения всех примеров обучающей выборки (при неизменности выходных сигналов сети).

В нейроимитаторах предусмотрено наличие специальных процедур инициализации перед обучением сети, т. е. присваивания параметрам сети некоторых малых случайных значений.

Обучение представляет собой итерационную процедуру, которая при реализации на персональных компьютерах требует значительного времени. Скорость сходимости алгоритма обучения является одной из самых важных характеристик программ для моделирования нейронных сетей.

6.3 Тестирование обученной нейронной сети

Для проверки правильности обучения построенной нейронной сети в нейроимитаторах предусмотрены специальные средства ее тестирования. В сеть вводится некоторый сигнал, который, как правило, не совпадает ни с одним из входных сигналов примеров обучающей выборки. Далее анализируется получившийся выходной сигнал сети [15].

Тестирование обученной сети может проводиться либо на одиночных входных сигналах, либо на тестовой выборке, которая имеет структуру, аналогичную обучаю-

щей выборке, и также состоит из пар (<вход>, <требуемый выход>). Обычно, обучающая и тестовая выборки не пересекаются. Тестовая выборка строится индивидуально для каждой решаемой задачи.

Контрольное множество данных участвовует в кросс-проверке сетей при их обучении. При отсутствии данной проверки, нейронные сети будут слишком близко следовать обучающим данным («чрезмерная подгонка») и воспринимать не столько структуру данных, сколько содержащиеся в них помехи. Кроме того, нейронные сети, которые обучались без кросс-проверки, обладают малой способностью к обобщению данных (т.е. показывать хорошие результаты на совершенно новых данных).

Тестовое множество данных, которое вообще никак не будет участвовать в обучении сетей, будет использоваться только после того, как будут сформированы окончательные варианты обученных сетей.

6.4 Моделирование нейронных сетей автоматическим конструктором из нейропакета *STATISTICA Neural Networks*

Автоматический конструктор сети автоматически выбирает подходящий тип сети и вариант архитектуры (т.е. число скрытых элементов). В стандартном варианте большинство пользователей просто проводят серию экспериментов с сетями различных типов и архитектур, выделяя на эвристическом уровне наиболее многообещающие, после чего именно на них и сосредотачивается все внимание. Автоматический конструктор сети действует по такой же схеме, и даже более формально. Задачу определения архитектуры сети он рассматривает как задачу оптимизации и для исследования возможных вариантов архитектур использует довольно сложные методы поиска (в том числе линейный поиск и вариант метода «отжига») [17].

Действия автоматического конструктора очень напоминают действия человека в этой же ситуации. При этом работа алгоритма может занимать весьма длительное время, поскольку он перебирает множество различных вариантов архитектур. Однако времени у него уйдет не больше, чем у человека, выполняющего ту же серию экспериментов, а преимущество этого метода в том, что в процессе работы он не требует присутствия пользователя. Кроме того, этот алгоритм осуществляет перебор гораздо большего числа вариантов, чем способен выполнить человек, и как следствие, его результаты часто оказываются лучше.

Сначала, для моделирования нейронной сети нужно открыть файл, в котором имеются все необходимые входные и выходные переменные.

В программе *ST Neural Networks* все наблюдения из файла данных делятся на четыре группы (множества): обучающие, контрольные, тестовые и неучитываемые. Обучающее множество служит для обучения нейронной сети, контрольное – для независимой оценки хода обучения, тестовое – для окончательной оценки после завершения серии экспериментов. Неучитываемое множество не используется вовсе [17].

Аналогично, все переменные делятся на входные, выходные, входные/выходные (например, при анализе временных рядов) и неучитываемые (последние обычно являются «кандидатами на роль входных переменных», чья полезность для построения прогноза заранее неясна, и потому в процессе экспериментирования некоторые из них отключают).

Тип переменной или наблюдения обозначается цветом соответствующих ячеек в файле данных.

Таблица 6.1 -Типы переменных и наблюдений в нейропакете

STATISTICA Neural Networks

Наблюдения	
Обучающее	Черный заголовок строки и содержимое ячейки
Контрольное	Красный заголовок строки и содержимое ячейки
Тестовое	Синий заголовок строки и содержимое ячейки
Неучитываемое	Серый заголовок строки и содержимое ячейки
Переменные	
Входная	Черный заголовок столбца
Выходная	Синий заголовок столбца
Входная/Выходная	Зеленый заголовок столбца
Неучитываемая	Серый заголовок столбца

6.5 Выбор архитектуры нейронных сетей

Одна из наиболее трудных задач при построении нейронной сети – выбор архитектуры, а после того, как архитектура выбрана, - задание ее свободных параметров в большой степени зависит от сложности задачи, которую нужно решить, и поскольку заранее неизвестно, насколько задача сложна, необходим ряд экспериментов [15,16].

Самое трудное – выбрать число промежуточных слоев и число элементов в них.

Существуют следующие правила определения промежуточных слоев в многослойном персептроне *MPL*:

- При использовании *MPL* для большинства практических задач достаточно одного промежуточного слоя. Если даже при большом числе скрытых элементов не удается уменьшить ошибку до приемлемого уровня, можно попробовать сеть с двумя промежуточными слоями.
- Чем больше число скрытых элементов в сети, тем более сложную задачу

она может моделировать, но при этом потребуется более длительное обучение и возникнет опасность переобучения. При экспериментировании нужно следить за тем, *как меняются обучающая и контрольная ошибки*. Если при добавлении новых элементов уменьшаются обе ошибки, то, по-видимому, сеть пока слишком мала. Если же контрольная ошибка стала намного больше, чем ошибка обучения (и, в частности, если она увеличивается при дальнейшем обучении), то, вероятно, что данная сеть слишком большая.

- Общее число весов и пороговых коэффициентов в сети должно быть меньше, а лучше – намного меньше, чем число обучающих наблюдений. В идеале нужно *иметь наблюдений в десять или двадцать раз больше, чем имеется весов.* Если обучающих наблюдений недостаточно, следует ограничиться сетями небольших размеров, поскольку для моделирования сложной функции попросту недостает данных. *Если число наблюдений меньше, чем число входов, умноженное на число выходов, следует использовать только линейные модели.*

- Для оценки числа нейронов в скрытых слоях однородных нейронных сетях можно воспользоваться формулой (данная формула является следствием из теоремы Колмогорова-Арнольда-Хехт-Нильсена о представимости любой многомерной функции нескольких переменных с помощью нейронной сети фиксированной размерности) [15] для оценки необходимого числа синаптических весов W в многослойной сети с сигмоидальными передаточными функциями:

$$\frac{mN}{1+\log_2 N} \le W \le m\left(\frac{N}{n}+1\right)(n+m+1)+m$$

(6.1)

где n – размерность входного сигнала; m - размерность выходного сигнала; N – число элементов обучающей выборки.

Оценив необходимое число весов, можно рассчитать число нейронов в скрытых слоях. Например, для двуслойной сети это число составит:

$$L = \frac{W}{n+m}$$

(6.2)

Известны и другие формулы для оценки, например:

$$2(n+L+m) \le N \le 10(n+L+m), \qquad (6.3)$$

$$\frac{N}{10} - n - m \le L \le \frac{N}{2} - n - m. \qquad (6.4)$$

Точно так же можно рассчитать число нейронов в сетях с большим числом нейронов.

В исследованиях часто отказываются от нейронных сетей с рекуррентными связями, поскольку такие сети на практике показали, что они могут иметь неустойчивый характер и очень сложную динамику поведения. Рекуррентные сети представляют большой интерес для исследователей в области нейронных сетей, однако, при решении практических задач, наиболее *полезными являются структуры прямой передачи*

Сначала для решения поставленной задачи целесообразно рассматривать типичные нейронные сети *MPL* с прямой передачей сигнала и с сигмоидальными передаточными функциями. Входной слой служит для ввода значений входных переменных. Каждый из скрытых и выходных нейронов соединен со всеми элементами предыдущего слоя. Можно было бы рассмотреть сети, в которых нейроны связаны только с некоторыми из нейронов предыдущего слоя, однако, для большинства приложений *предпочтительны сети с полной системой связей.*

Пример 1. Создать нейронную сеть автоматическим конструктором на основе структуры регрессионных моделей. Для разработки модели используются режимные данные технологического процесса выработки листового стекла и показатели качества стекла (таблица 6.2). Для обучения нейронных сетей используются выборки из 200 обучающих, 82 контрольных и 83 тестовых наблюдений.

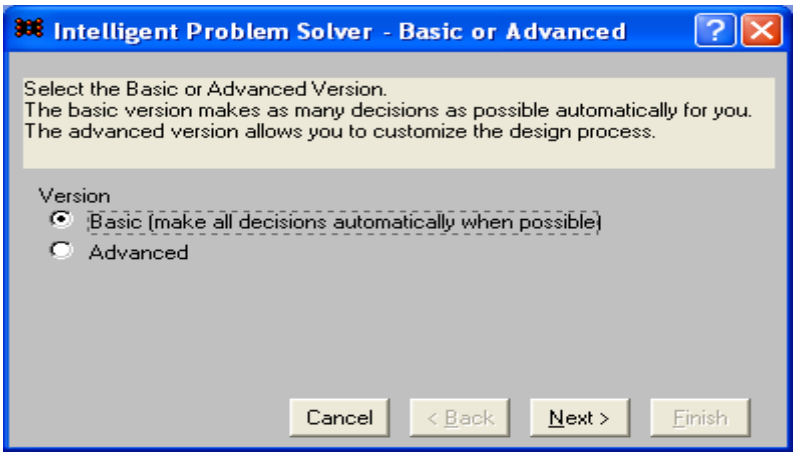
Таблица 6.2 – Данные для построения модели

Входные данные	
TCANAL	Температура канала стекловаренной печи
TIN1L	Температура олова в 1 пролете левая
TIN1R	Температура олова в 1 пролете правая
TIN12R	Температура олова в 12 пролете правая
TIN20R	Температура олова в 20 пролете правая
TOUTC	Температура выходоного конца ленты стекла
TOLSC	Толщина ленты стекла
K2	Качество 2 (дефекты возникающие во время формования ленты стекла)
Выходные данные	

R1DOL	Растр 1-й долянки
RAZN2	Разнотолщинность 2-й долянки
BLYM	Блюм эффект
IOMZLB	«Зебра» по левому борту ленты стекла

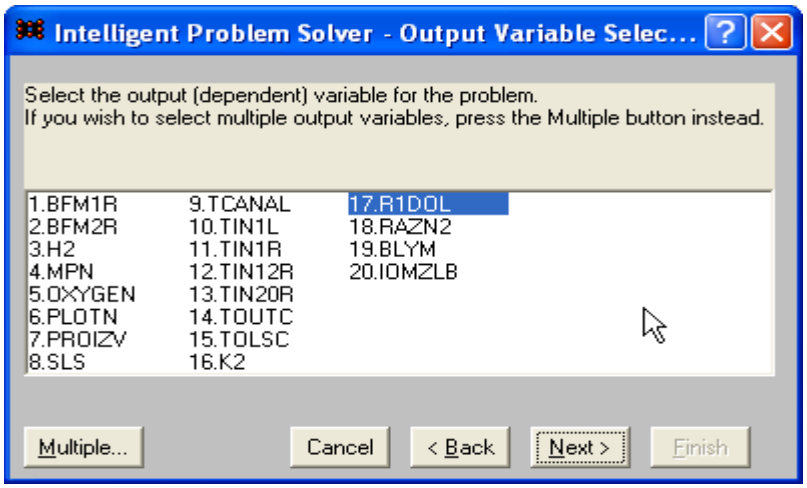
Рассмотрим построение нейронной сети моделирующей значения оптических искажений вырабатываемого стекла, оцениваемых величиной растра.

Создание нейронных сетей автоматическим конструктором начинается с открытия следующего диалогового окна:

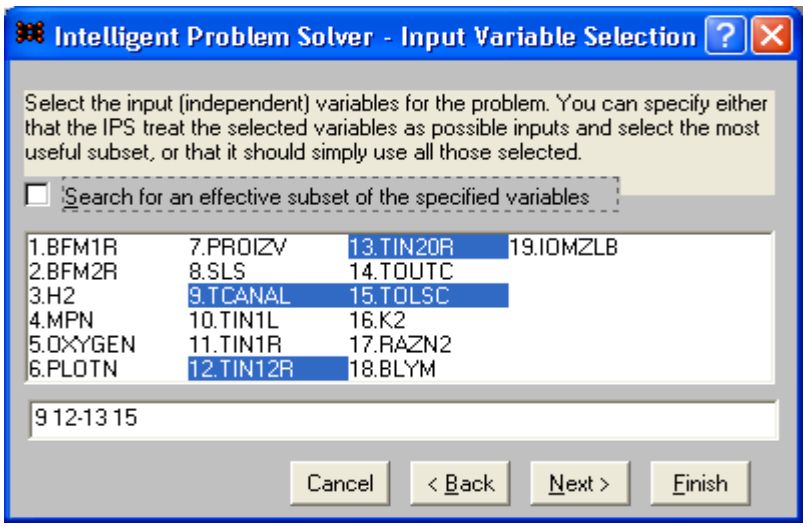


В исследованиях использован второй вариант (*Advanced*) построения нейронных сетей, поскольку он отличается от первого более расширенными настройками.

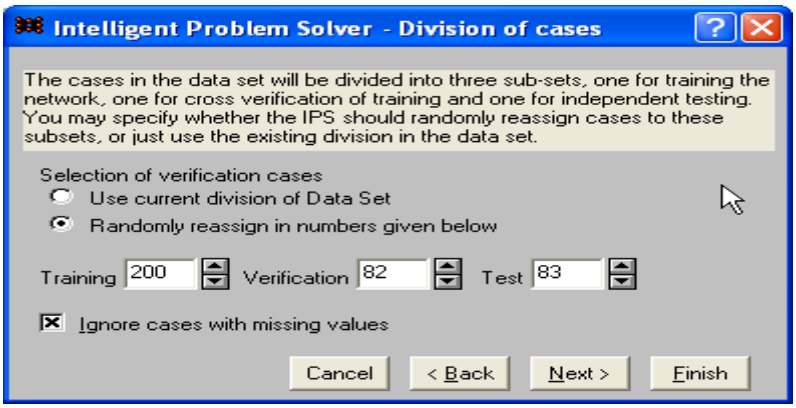
Далее нужно выбрать один из показателей качества стекла, который будет являться выходом нейронной сети. В данном случае – это растр 1- ой долянки 17.R1DOL.



В следующем окне необходимо выбрать режимные переменные, которые будут являться входными данными для нейронной сети (выбрано четыре переменных).

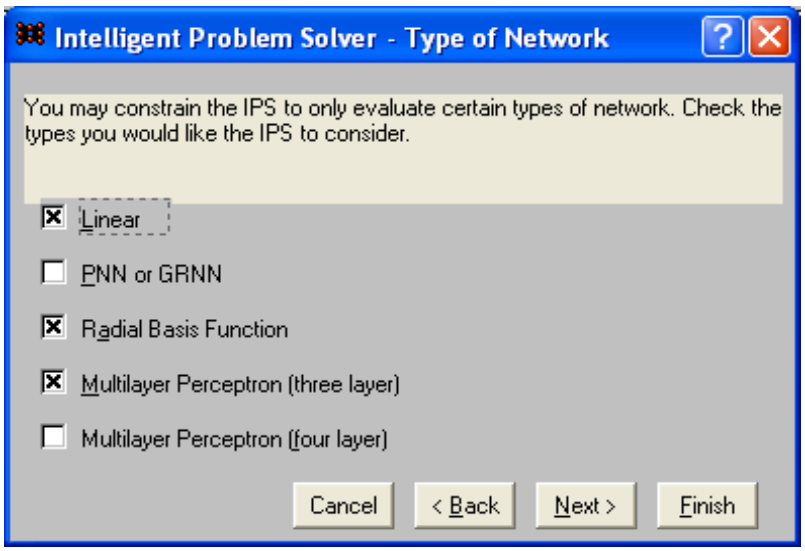


На следующем шаге автоматического конструктора нейронной сети, необходимо указать количество обучающих (*Training*), контрольных (*Verification*) и тестовых (*Test*) наблюдений.



Далее в следующем окне нужно выбрать типы сетей, которые будут анализироваться. Они упорядочены в порядке сложности обучения. Линейные (*Linear*) сети почти не требуют обучения и включены сюда потому, что дают хорошую точку отсчета для сравнения эффективности различных методов. Сети *PNN* и *GRNN* также довольно просты, радиальные базисные функции (*Radial Basis Functions*) устроены несколько сложнее, а трех- и четырехслойные многослойные персептроны (*Multiplayer Perceptrons*) – это сложные конструкции.

Для трехслойного *MPL* поиск сводится к выбору числа элементов в скрытом слое. По существу, это линейный поиск для функции, содержащей помехи (при разных прогонах обучения сети, даже с одним и тем же числом элементов, могут получаться немного различные результаты).



На следующем шаге определялась сложность нейронной сети.

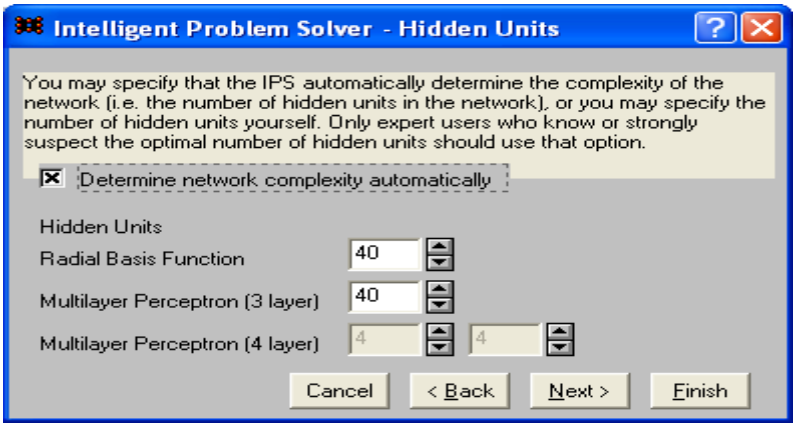
Сложность нейронной сети, в первую очередь, зависит от количества нейронов (L) в скрытых слоях, числа синаптических связей (W) в сети и числа элементов обучающей выборки (N). Из расчетов числа нейронов в промежуточном слое имеем следующее:

$$23 \le W \le 288 \quad \text{по формуле (6.1)},$$

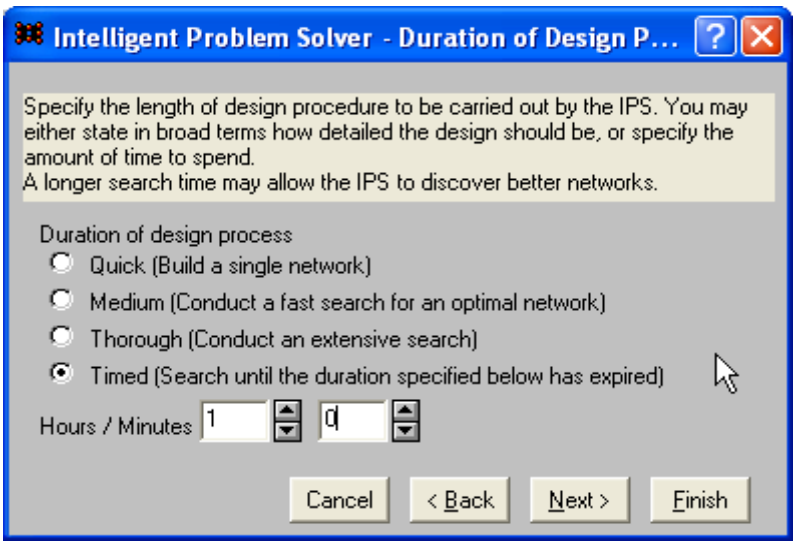
$$4 \le L \le 48 \quad \text{по формуле (6.2)},$$

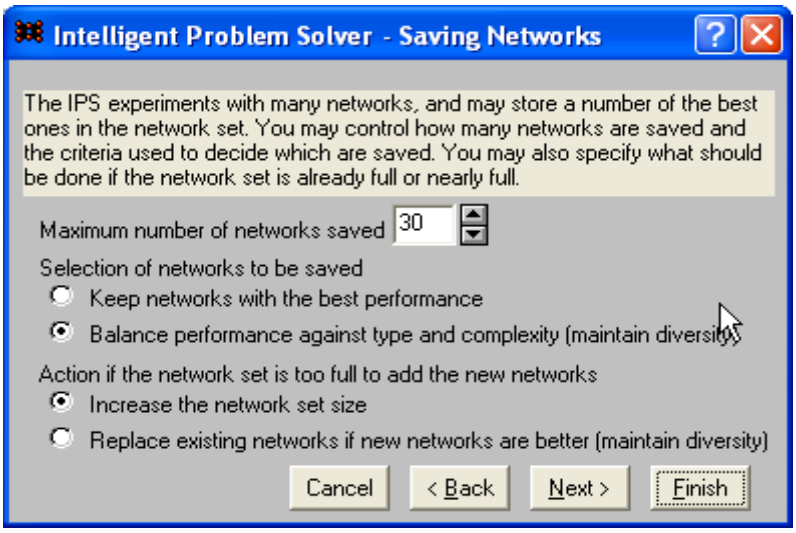
$$14 \le L \le 94 \quad \text{по формуле (6.4)}.$$

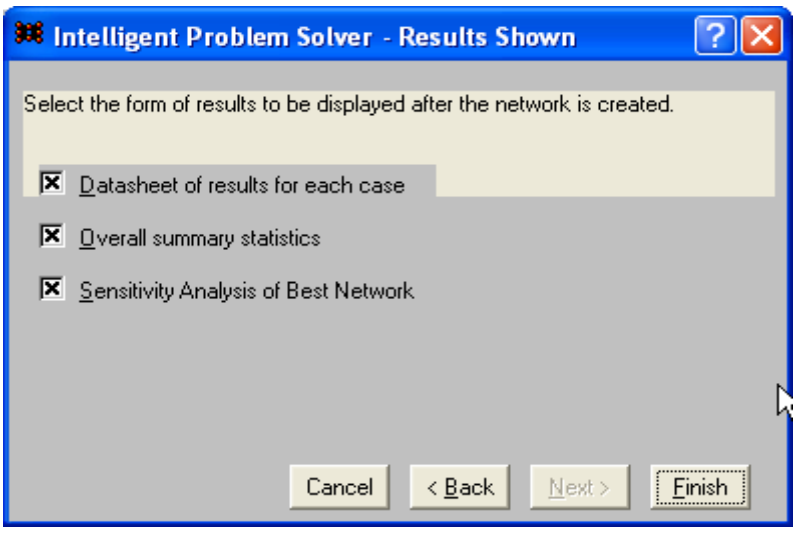
Таким образом, для быстроты анализа автоматическому конструктору сразу можно задать среднее значение числа нейронов в промежуточном слое.



На следующих этапах задается время в течение, которого будет идти поиск, максимальное число лучших сетей, которые будут сохранены и какие результаты будут выведены после выполнения поиска.

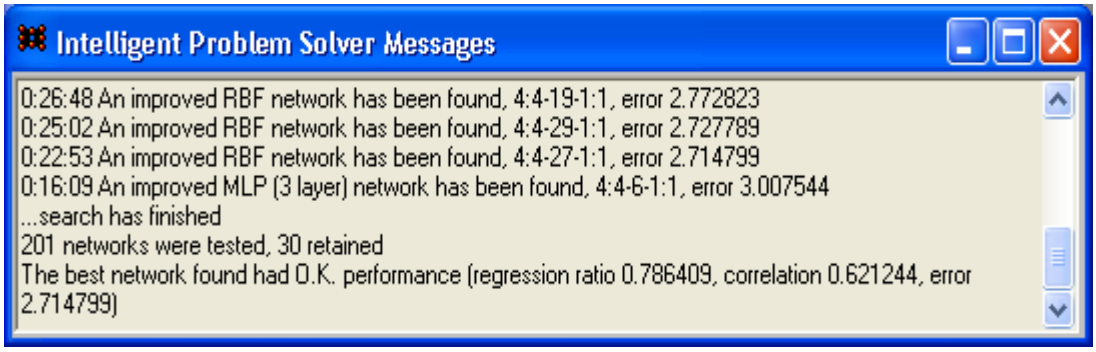






По окончании поиска лучшей нейронной сети для моделирования растра автоматический анализатор сообщил о том, что было протестировано 201 нейронная сеть с

различными архитектурами и самая лучшая из протестированных сетей обладает хорошими характеристиками прогнозирования.



Данные по десяти лучшим нейронным сетям из 201 протестированных приведены в следующей таблице:

Type	*Inputs*	*Hidden*	*Hidden(2)*	*TError*	*VError*	*TeError*	*TPerf*	*VPerf*	*TePerf*
RBF	4	24	-	2.86099	2.87544	3.69751	0.82190	0.83276	0.89846
RBF	4	19	-	2.99569	2.86022	3.81234	0.86060	0.82867	0.92920
RBF	4	28	-	2.81572	2.81671	3.63405	0.80890	0.81671	0.88560
RBF	4	22	-	2.87680	2.80449	3.79385	0.82645	0.81328	0.92648
RBF	4	19	-	3.01287	2.79976	3.84108	0.86554	0.81099	0.93580
RBF	4	19	-	2.97849	2.77282	3.84208	0.85566	0.80353	0.93652
RBF	4	26	-	2.82250	2.76111	3.61310	0.81085	0.80046	0.87848
RBF	4	29	-	2.82679	2.72778	3.6095	0.81208	0.79012	0.87791
RBF	4	27	-	2.83127	2.71479	3.60613	0.81337	0.78640	0.87643

где *Type* – вид нейронной сети;

Inputs – число входных переменных;

Hidden – число нейронов в первом промежуточном слое;

Hidden (2) – число нейронов во втором промежуточном слое;

TError – среднеквадратичная ошибка сети на обучающей выборке;

VError – среднеквадратичная ошибка сети на контрольной выборке;

TEEror – среднеквадратичная ошибка сети на тестовой выборке;

Отношение стандартного отклонения ошибки к стандартному отклонению данных:

TPerf – на обучающей выборке;

VPerf – на контрольной выборке;

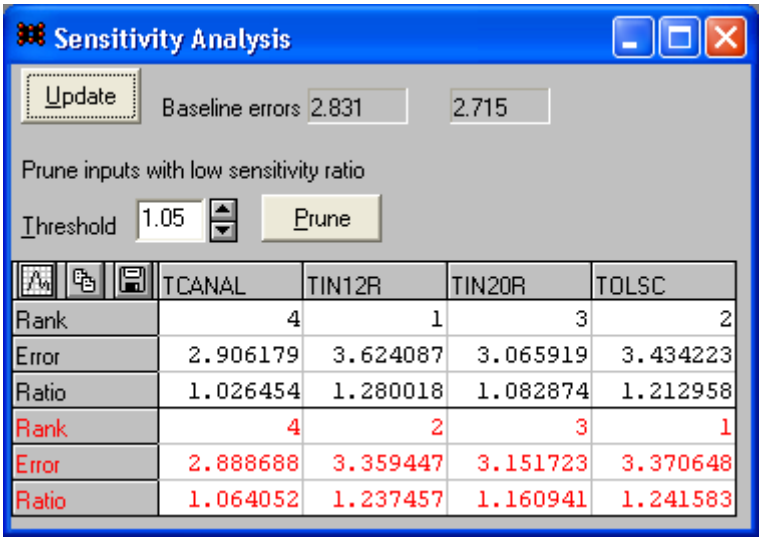
TEPerf – на тестовой выборке.

Из полученных результатов чувствительности нейронной сети к входным данным для обучающей и контрольной выборки видно, что нейронная сеть наиболее чувствительна к изменению значений режимных переменных Tin12 (температура олова) и TOLSC (толщина стекла).

Error – это значение ошибки сети в случае если рассматриваемая переменная не была бы в входных данных.

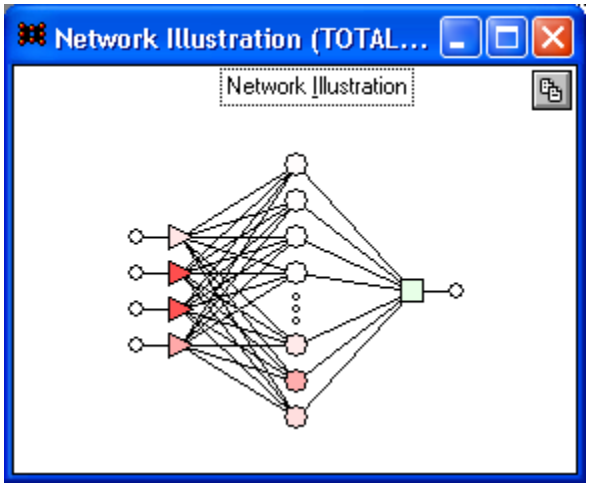
Ratio – это отношение значения *Error* к значению *Baseline Error*

Baseline Error - это значение ошибки сети в случае если все отобранные переменные будут являться входными данными.



	TCANAL	TIN12R	TIN20R	TOLSC
Rank	4	1	3	2
Error	2.906179	3.624087	3.065919	3.434223
Ratio	1.026454	1.280018	1.082874	1.212958
Rank	4	2	3	1
Error	2.888688	3.359447	3.151723	3.370648
Ratio	1.064052	1.237457	1.160941	1.241583

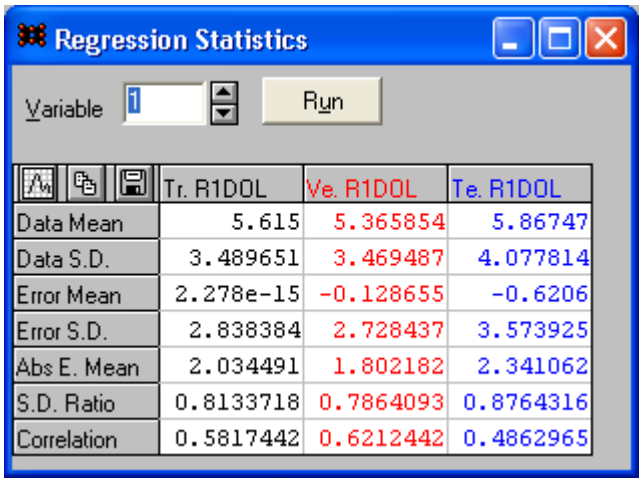
Ниже показан вид лучшей нейронной сети, прогнозирующей значения растра.



Оценка качества данной нейронной сети представлена в диалоговом окне (Регрессионная статистика).

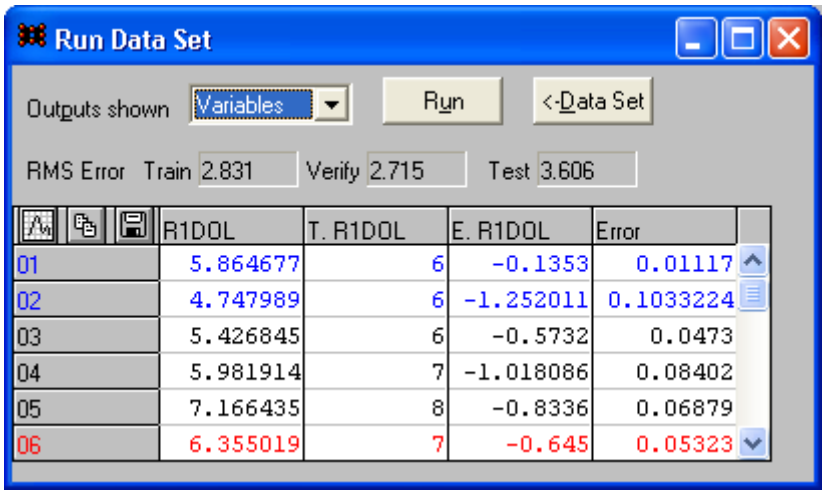
В окне *Regression Statistics* (статистики регрессии) подсчитываются среднее и стандартное отклонение для выходных переменных и ошибки сети, а также отношение стандартного отклонения ошибки к стандартному отклонению данных и коэффициент корреляции между спрогнозированным и фактическими значениями. Обычно значение средней ошибки (*Error Mean*) получается близким к нулю, а стандартное отклонение прогноза оказывается существенно меньше стандартного отклонения данных.

Если величина *S.D.Ratio* меньше 0,1, это означает хорошее качество регрессии.

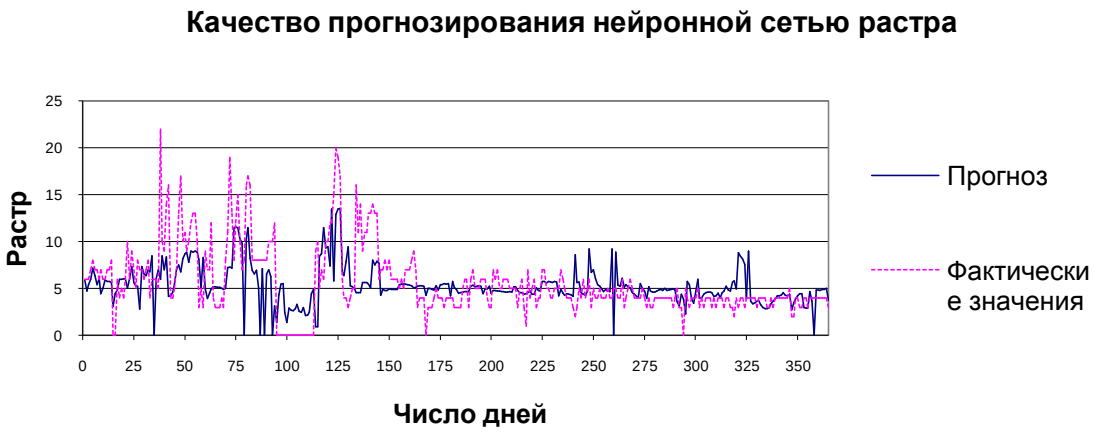


	Tr. R1DOL	Ve. R1DOL	Te. R1DOL
Data Mean	5.615	5.365854	5.86747
Data S.D.	3.489651	3.469487	4.077814
Error Mean	2.278e-15	-0.128655	-0.6206
Error S.D.	2.838384	2.728437	3.573925
Abs E. Mean	2.034491	1.802182	2.341062
S.D. Ratio	0.8133718	0.7864093	0.8764316
Correlation	0.5817442	0.6212442	0.4862965

В окне *Run Data Set* (прогнать набор данных) содержаться следующие значения (перечисленные слева направо): фактические выходы сети, целевые выходные значения, ошибки, т.е. разности между первым и вторым, и суммарная ошибка по каждому наблюдению. Над таблицей выдается окончательная среднеквадратичная ошибка (СКО) – *RMS error* сети на этом наборе данных.



	R1DOL	T. R1DOL	E. R1DOL	Error
01	5.864677	6	-0.1353	0.01117
02	4.747989	6	-1.252011	0.1033224
03	5.426845	6	-0.5732	0.0473
04	5.981914	7	-1.018086	0.08402
05	7.166435	8	-0.8336	0.06879
06	6.355019	7	-0.645	0.05323



Тестовые задания

1. Создание нейронной сети. Виды моделей.
2. Процесс обучения нейронной сети.
3. Тестирование обученной нейронной сети.
4. Моделирование нейронных сетей автоматическим конструктором.

Лекция 7 **Ситуационное моделирование или ситуационное управление**

7.1 Основы ситуационного моделирования и управления

Ситуационное моделирование или ситуационное управление - направление, предложенное Д.А. Поспеловым.

История возникновения. Метод ситуационного управления возник в связи с необходимостью моделирования процессов принятия решений в системах с активным элементом (человеком). В его основе лежат три основные предпосылки.

Первая предпосылка - это психология, которая начала изучать принципы и модели принятия решений человеком в оперативных ситуациях. Известны работы советских психологов в этой области - В.Н. Пушкина, Б.Ф.Ломова, В.П. Зинченко и др. [3]. В.Н. Пушкин сформулировал так называемую *модельную теорию мышления*. Он показал, что психологический механизм регулирования актов поведения человека тесно связан с построением в структурах мозга информационной модели объекта и внешнего мира, в рамках которого осуществляется процесс управления на основе восприятия человеком информации извне и уже имеющегося опыта и знаний. Основой построения модели являются понятийные представления об объектах и отношениях между ними, отражающие семантику выделенной сферы деятельности человека (предметной области). Модель объекта имеет многоуровневую структуру и определяет тот информационный контекст, на фоне которого протекают процессы управления. Чем богаче такая информационная модель объекта и выше способности манипулирования знаниями, тем выше качество принимаемых решений, многообразнее поведение человека. В.Н. Пушкин впервые выделил три важные особенности процесса принятия решений [14]: *наличие возможности классификации ситуаций в соответствии с типовыми решениями по управлению; принципиальная открытость больших систем; существенная ограниченность языка описания пространства состояний и решений объекта управления.*

Второй предпосылкой метода ситуационного управления стали представления, полученные в исследованиях по семиотике - науке о *знаковых системах.* Это работы Ю.А. Трейдера, Ю.Д. Апресяна. Была определена трехаспектная структура знака в любой знаковой системе: имя знака, отражающее его синтаксический аспект; содержание знака, выражающее его семантический аспект; назначение знака, определяющее его прагматический аспект (треугольник Фреге). В прикладной семиотике знаки, вариантами которых являются слова, предложения, тексты, стали рассматриваться как системы, замещающие реальные объекты, процессы, события внешнего мира [3]. Совокупности знаков с отношениями между ними, таким образом, стали моделирующими псевдофизическими аналогами

реальных систем функционирования и управления. Именно поэтому ситуационное управление называли еще и *семиотическим моделированием*, поскольку знаковый язык достаточен для описания и процессов функционирования объекта с требуемой степенью приближения.

Третья предпосылка связана с разработками в области *информационно-поисковых систем* и попытками создания формального языка описания и представления технических наук с целью автоматизации работ по реферированию научных публикаций и организации процессов поиска, хранения и представления информации. В рамках этих исследований Э.Ф. Скороходько был разработан и исследован язык, получивший затем название языка глжодов [3]. Свою реализацию этот язык нашел в информационно-поисковой системе БИТ, которая успешно и довольно долго эксплуатировалась в Институте кибернетики АН УССР.

На основе модельной теории мышления В.Н. Пушкина, языка *r-x*-кодов Э.Ф. Скороходько и семиотики Д.А. Поспелов, а затем Ю.И. Клыков в 1965 г. сформулировали новую кибернетическую концепцию управления большими системами в виде метода ситуационного управления.

Сущность метода

• За основу управления принято понятие *ситуация* как основной объект описания, анализа и принятия решений. Следовательно, необходимы соответствующие средства - описания, классификации, обучения и трансформации ситуаций в соответствии с принимаемыми решениями.

• Классификация ситуаций обосновывалась существованием, исходя из анализа структуры задач управления в больших системах, на каждом уровне управления множества ситуаций, число которых несоизмеримо велико по сравнению с множеством возможных решений по управлению. *Задача принятия решений трактовалась как задача поиска такого разбиения множества ситуаций на классы, при котором каждому классу соответствовало решение, наиболее целесообразное с позиции заданных критериев функционирования.* При наличии такого разбиения поиск решения в конкретной ситуации сводился к поиску класса и соотнесения ему решения по управлению. Однако такая постановка задачи справедлива для систем управления, в которых число потенциально возможных ситуаций (ПВС) существенно превышает (иногда на несколько порядков) число возможных решений по управлению. *Этот случай соответствует контекстно-независимому способу вывода решений, когда все множество ПВС разбивается на классы таким образом, чтобы каждому классу в соответствие ставилось решение по управлению*. Случай,

когда множества ситуаций и решений были либо соизмеримы по мощности, либо достаточно больше, чтобы этот факт можно было установить, был рассмотрен и разработан затем в работах Л.С. Загадской и ее школы [3].

За основу языка описания всего множества ситуаций были взяты идеи языков rx-кодов и синтагматических цепей. Роль множества объектов предметной области играли их знаковые эквиваленты в естественном языке, т.е. слова-имена, а в роли отношений выступали слова-имена, соответствующие реальным связям между объектами или процессами. В качестве грамматики языка ситуационного управления (ЯСУ) выступали правила порождения новых понятий и отношений, их преобразования и классификации (см. *Язык ситуационного управления).*

• *Важнейшая идея метода - формирование семиотической модели объекта путем обучения принятию решений.* При этом рассматривались два режима обучения: экспертом, хорошо знающим исследуемую предметную область, либо на основе анализа множества конкретных ситуаций и решений по управлению. Очевидно, что последний случай более длителен, не гарантирует полноту описания, требует наличия статистики ситуаций и принятых в них решений, что далеко не всегда возможно. Поэтому всеобщей практикой стало в основном использование первого подхода к обучению. Тем не менее наличие в ЯСУ средств обобщения и классификации ситуаций обеспечивает принципиальную возможность создания моделей, способных к усовершенствованию функций принятия решений в изменяющихся условиях работы объекта управления. Другими словами, создается возможность «выращивания» модели объекта для заданных условий функционирования.

Развитие ситуационного моделирования. В 1973 г. Л.С. Загадская (Болотова) [3] разработала еще один, новый тип систем ситуационного управления, рассматривавший класс систем управления, в котором мощности множеств возможных ситуаций и решений по управлению сопоставимы или неизвестны. Предлагалось все множество ситуаций разбивать на классы таким образом, чтобы каждому классу в соответствие ставилась *структура типового решения.* На следующем этапе решения эта структура доопределялась в процессе интерпретации и конкретизации решения и с учетом имеющихся ограничений на ресурсы. Таким образом, каждому типовому решению по управлению *Uj* в соответствие ставится его структура М., и, следовательно, кроме множества *U* - {£/,, *U*1,...*Un*\, строится множество структур типовых решений *M — {Mv M*2,...*Mn}.*

Затем для каждой структуры выявлялся необходимый контекст-пласт знаний, имеющий фреймовую структуру и включающий правила интерпретации ситуаций в пределах данной структуры и множество процедур для их трансформации и имитации. Была разра-

ботана также логико-семиотическая модель вывода решений на иерархии структур принятия решений [3].

Очевидно, что во втором случае существенно усложняется проблема построения модели предметной области (МПО). *Разработка МПО до сих пор является искусством, требует применения высочайшей квалификации системных аналитиков.* Необходимо ответить на ряд вопросов:

- Каким образом задаются границы выделенной предметной области?
- Каким образом формируется непротиворечивый язык описания всех множеств ситуаций и процессов для МПО со сложной, иерархической и распределенной структурой?
- Каким образом формируется система знаний о МПО, достаточная для достижения поставленных целей?
- Каким образом «проявляются» необходимые взаимодействия между участниками процессов управления и принятия решений, как они описываются?
- Каким образом принимаются решения в условиях неполноты, неопределенности и неоднозначности?

В результате исследования и разработки прикладных систем ситуационного управления была создана сквозная методология и технология проектирования систем ситуационного управления большими системами, включая необходимые инструментальные средства и системы на базе языков РЕФАЛ и ЛИСП [3].

Как следует из описания *языка ситуационного управления* и организации ситуационной модели управления, уже тогда, в 70-е гг. XX в., системы ситуационного управления (ССУ) имели все признаки современных экспертных систем (ЭС) по меньшей мере 2-го поколения, т.е. динамических ЭС. Это и наличие семиотической модели объекта управления и процессов его функционирования в виде системы правил продукционного типа, и естественно-языковый интерфейс с разработчиками и пользователями, и наличие встроенной логики времени, обеспечивающей работу ССУ в режиме реального времени и моделирования. Это и инструментальные программные средства реализации ССУ на базе языков ЛИСП и РЕФАЛ. Более того, отечественные специалисты создавали большие системы и даже внедряли их в практику в составе промышленных АСУ.

Примеры.

- Система ситуационного управления «Авиаремонт», выполненная Одесским отделением Института экономики АН УССР как часть АСУ «Авиаремонт» для ЦНИИАСУ (Рига).

• Система ситуационного диспетчерского управления взлетом и посадкой самолетов, разработанная для ВНИИРА (Ленинград).

• Система планирования сеансов спутниковой связи.

• Ряд систем специального назначения и др.

В начале 80-х гг. появились *экспертные системы*, и тут выяснилось, что по своей сути они вроде бы совпадают с ССУ, как у нас их и представляли. И термин этот показался более удачным, быстро вошел в моду. В результате уже к началу 90-х гг. XX в. почти все «ситуационщики» занимались ЭС. Таким образом, получилось, что ситуационное управление сыграло в нашей стране роль основы для большого числа специалистов по *искусственному интеллекту*.

7.2 Моделирование процессов принятия решений в системах с активным элементом (человеком).

Рассмотрим методику разработки ситуационной модели на основе анализа и формализации действий ЛПР на примере управления технологическим процессом варки-выработки листового стекла [19]. Отклонения режимных переменных рассматриваются как появление ситуаций (начальных) $s_н$ на объекте управления, а нахождение режимных переменных в пределах допусков как конечную ситуацию $s_к$. Управление можно представить как перевод объекта из начальной ситуации $s_н \in S_н$ в конечную ситуацию $s_к \in S_к$ под действием управляющих воздействий:

$$s_н \xrightarrow{u_1} s_1 \xrightarrow{u_2} s_2 \xrightarrow{u_3} .. \xrightarrow{u_n} s_к\ , \qquad (7.1)$$

где $S_н$, $S_к$ - множества начальных и конечных ситуаций;

s_i, i=1,... n - множества промежуточных ситуаций;

u_i, i=1,... n - управляющие воздействия по шагам управления.

В зависимости от числа последовательно выполняемых управляющих воздействий по переводу объекта из начальной ситуации в конечную, управление может быть многошаговым (7.1) или одношаговым:

$$s_н \xrightarrow{u_1} s_к\ . \qquad (7.2)$$

Наличие возможности классификации ситуаций в соответствии с типовыми решениями по управлению, позволяет использовать машинную процедуру формирования понятий *CLS*-9 для построения ситуационной модели управления [20]. Понятия представляются в виде деревьев классификаций, устанавливающих соответствие между дискретными

представлениями произвольных ситуаций и выбираемыми управляющими воздействиями. Действия ЛПР формализуются на основе высказываний экспертов о принимаемых управляющих решениях в тех или иных технологических ситуациях, возникающих на объекте управления.

Формирование понятий может рассматриваться как задача распознавания образов, когда по признакам (ситуации) определяется значение управляющего воздействия, т.е. из множества критериальных классов управляющего воздействия выбирается нужный класс.

Совокупность всевозможных признаков зависит от количества переменных, характеризующих признаки и числа возможных значений каждого признака. При трех значениях каждого признака $V_i = 3$, учитывающих знак отклонения переменной от задаваемого интервала:

“-1” - выход за пределы нормы в сторону уменьшения (меньше);

“0” - значение переменной в допустимых пределах (норма);

“+1” - выход за пределы нормы в сторону увеличения (больше)

и числе переменных ситуаций n, генеральная совокупность признаков будет иметь размерность 3^n. При этом каждая ситуация будет кодироваться n разрядным (троичным) числом.

Множество критериальных классов для каждого управляющего воздействия ограничивается тремя элементами:

$$K = \{ - , = , + \} , \qquad (7.3)$$

где “-” - отрицательное приращение управляющего воздействия (уменьшить);

“ = “ - нулевое приращение (не изменять);

“ + ” - положительное приращение (увеличить).

Процедура *CLS*-9 формирует деревья по каждому управляющему воздействию. От вершин деревьев, характеризующих признак, отходят V_i ветвей. Алгоритм предусматривает использование в качестве критерия того признака, который будет наиболее полезным при классификации относящихся к данной вершине объектов. Для описания алгоритма формирования понятий вводится следующая система обозначений:

$A = \{A_j\}$ - множество признаков, не рассматривавшихся ни в одной из вершин, расположенных выше исследуемой;

$K = \{ k \}$ - множество критериальных классов;

V_i - количество возможных значений признака i;

$n_{i,j,k}$ - количество относящихся к рассматриваемой вершине объектов, которые характеризуются значением j признака i и принадлежат к классу k;

n^*_{ij} - максимальное значение параметра $\{n_{ijk}\}$ для любого из k классов.

Процедура *CLS*-9 определяет все члены множества $\{n_{ijk}\}$ и для каждого признака находит значение величины

$$H_i = \sum_{j=1}^{V_i} n^*_{ij} \ . \qquad (7.4)$$

В качестве критерия выбирается тот признак, для которого значение H_i оказывается максимальным. Затем алгоритм формирует вершину, от которой вниз отходят ветви по числу V_i возможных значений выбранного признака i. Построенное дерево является “наилучшим из возможных” в смысле классификации всех объектов, описания которых были введены в память ЭВМ к моменту формирования дерева. Классификация всех объектов оказывается правильной.

Пример. Дана выборка для формирования понятий по выбору управляющих воздействий для процесса стекловарения в ванной печи средней производительности (таблица 7.1) [19].

Таблица 7.1- Выборка для разработки ситуационной модели управления процессом стекловарения

Ситуация	Признаки							Принимаемые решения по управлению		
№	$\Theta_{гс}$	$\Theta_{см}$	$Q_г$	P	$L_ш$	$L_п$	M	$\mu_г$	$\mu_в$	$\mu_ш$
1	+1	-1	0	0	0	0	0	=	-	=
2	+1	-1	0	+1	-1	0	0	=	+	-
3	+1	-1	0	-1	-1	-1	0	=	=	+
4	+1	-1	0	1	-1	+1	0	=	-	-
5	+1	-1	-1	+1	0	0	-1	+	=	-
5	+1	-1	+1	+1	-1	0	0	=	=	-
7	-1	-1	+1	+1	-1	0	0	=	+	-
8	-1	-1	-1	+1	-1	0	-1	+	+	-
9	+1	-1	-1	-1	-1	0	-1	+	=	+
10	+1	-1	+1	+1	-1	0	-1	=	=	-
11	+1	-1	-1	0	-1	-1	-1	+	=	=
12	+1	-1	+1	+1	-1	-1	0	=	=	-
13	+1	-1	-1	-1	-1	-1	-1	+	=	+
14	-1	-1	-1	-1	-1	-1	-1	+	+	+
15	+1	-1	+1	+1	-1	+1	-1	=	-	-
16	+1	-1	-1	-1	-1	+1	-1	+	-	=

Обозначения признаков: $\Theta_{гс}$ – температура газовой среды в ванной печи; $\Theta_{см}$ – температура стекломассы в ванной печи; $Q_г$ – расход газа на ванную стекловаренную печь; P – давление газового пространства в ванной печи; $L_ш$ – граница варочной шихты в ванной печи; $L_п$ – граница варочной пены в ванной печи; M- число работающих машин вертикального вытягивания стекла.

Обозначения управляющих воздействий: $\mu_г$ – расход природного газа на горелки; $\mu_в$ – расход воздуха на горение; $\mu_ш$ – положение шибера дымовой трубы (тяга дымовых газов).

Рассмотрим построение ситуационной модели $\mu_в$ – расхода воздуха на горение. Алгоритм построения модели состоит из следующих шагов:

1) построить таблицу частот n_{ij} распределения значений (-1,0.1) признаков по критериальным классам $\mu_в$ (-, =, +), (Таблица 7.2);

2) определить максимальное число попаданий n_{ij}^*для каждого значения признака в один из критериальных классов n_i;

3) для каждого признака рассчитать значение величины $H_i=\sum n_{ij}^*$;

4) в качестве критерия выбирать тот признак, для которого значение H_i оказывается максимальным. При одинаковых значениях H_i = 11, как в нашем случае, критерий выбирается экспертным путем, либо случайным образом. Таким признаком в конкретном случае выбран признак $L_п$.

Таблица 7.2 - Определение значения управляющего воздействия $\mu_в$

из множества критериальных классов по выборке n=16

Признаки		$\Theta_{гс}$			$\Theta_{см}$			$Q_г$			P			$L_ш$			$L_п$			M		
Значения		-1	0	+1	-1	0	+1	-1	0	+1	-1	0	+1	-1	0	+1	-1	0	+1	-1	0	+1
Критериальные классы $\mu_в$	-	0	0	4	4	0	0	1	2	1	1	1	2	3	1	0	0	1	3	2	2	0
	=	0	0	8	8	0	0	4	1	3	3	1	4	7	1	0	4	4	0	5	3	0
	+	3	0	1	4	0	0	2	1	1	1	0	3	4	0	0	1	3	0	2	2	0
n_{ij}^*		3	0	8	8	0	0	4	2	3	3	1	4	7	1	0	4	4	3	5	3	0
$H_i=\sum n_{ij}^*$		11			8			9			8			8			11			8		

5) построить дерево классификаций, в вершину записать выбранный признак $L_п$ с наибольшим критерием. От вершины вниз отходят ветви по числу значений выбранного признака с лева на право i=-1, 0, +1. При i=+1находится решение для трех ситуаций, записанных в строках 4, 15, 16 таблицы 7.1- выбирается критериальный класс $\mu_в$ = (-). В «лист» дерева записывается символом (-). Из таблицы 7.1 удаляются строки 4, 15, 16, в которых ситуации распознаны.

При значениях признака i=-1, i= 0 критериальные классы не различимы, нужна дополнительная информация для получения решения (рисунок 7.1);

6) для продвижения по ветви i=-1признака $L_п$ необходимо образовать узел для записи второго признака. Выбирается тот признак, который не использовался выше создаваемого узла и имеет наибольшее значение H (таблица 7.2). Таким признаком является $\Theta_{гс}$. Из созданного узла выходят три ветви со значениями признака $\Theta_{гс}$, равными i=-1, 0, +1;

7) найти решение при значении признака $\Theta_{гс}$, равном i=-1. Находится решение $\mu_в$= (+) для одной ситуаций, находящейся в строке 14 таблицы 7.1.

При i=0 классификация не определена, в «листе» дерева записывается символ (?).

При i=+1 находятся решения по таблице 4.1 как непустое пересечения данных, содержащихся в столбцах $L_п$ =(-1), $\Theta_{гс}$ =(+1) и $\mu_в$=(=). Классифицируются четыре ситуации $\mu_в$=(=), содержащиеся в строках 3, 11, 12, 13 таблицы 7.1;

8) найти решение для ветви i=0 признака $L_п$, т.е. вершины дерева. В качестве признака выбирается повторно $\Theta_{гс}$. При значении i=-1 находится решение $\mu_в$= (+) для двух ситуаций, содержащихся в строках 7, 8 таблицы 7.1.

При i=0 классификация не определена, в «листе» дерева записывается символ (?), а при i=+1 критериальные классы не различимы, нужна дополнительная информация для нахождения решения;

9) в качестве третьего признака выбирать тот, который не использовался выше по дереву, в тоже время имеет большее значение H (таблица 7.2). Таким признаком является Qг, который записывается в очередную вершину (рисунок 7.1). Из вершины Qг исходят три ветви со значениями, равными i=-1, 0, +1;

10) найти решение для ветви i=-1. По таблице 7.1 определяется решение $\mu_в$= (=) для одной ситуаций 9 (таблица 7.1).

При значении i=+1 находится решение $\mu_в$= (=) еще для двух ситуаций 5, 10 (таблица 7.1).

При i=0 классификация не однозначная, нужна дополнительная информация для нахождения решения;

11) для продвижения по ветви i=0 признака Qг создается узел, в который записывается один из четырех признаков с H=8 (таблица 4.2). Выбираем признак P. Из созданного узла выходят три ветви со значениями i=-1, 0, +1;

12) найти решения для значения i=-1признака P. Решение отсутствует, в «листе» записывается вопросительный знак.

При i=0 признака P по таблице 7.1 находится решение $\mu_в$= (-) для 1-й ситуаций (таблица 7.1).

При значении i=+1 признака P по таблице 7.1 находится решение $\mu_в$= (+) для 2-й ситуаций (таблица 7.1).

Дерево (рисунок 7.1) правильно классифицирует все 16 ситуаций. В «листьях» дерева записано множество критериальных классов для управляющего воздействия $\mu_в$, которое ограничено тремя элементами: { - ,= , + }.

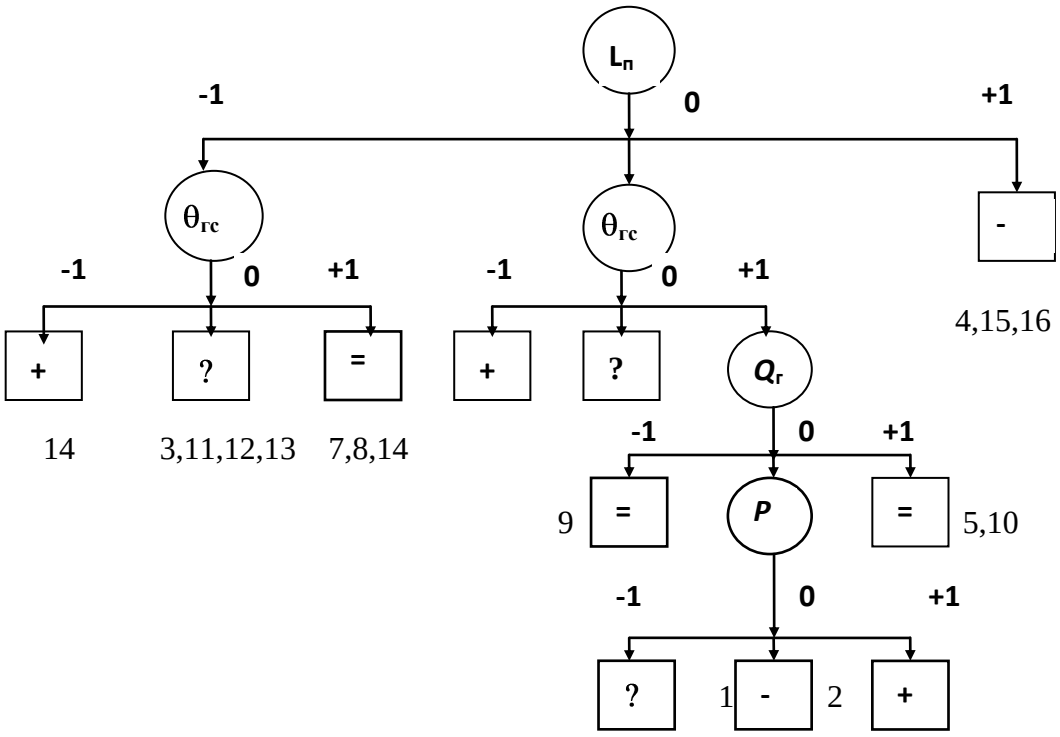


Рисунок 7.1 - Дерево классификации управляющего воздействия $\mu_в$

Рядом с листьями указаны номера классифицированных ситуаций. В трех «листьях» записана неопределенная классификация – «?». Это может быть следствием относительно малого количества ситуаций в выборке, либо может отражать тот факт, что некоторые ситуации не встречались в ходе технологического процесса, и такие опыты отсутствуют в выборке.

Приведенные на рисунке 7.1 дерево правильно классифицирует управляющие воздействия для всех ситуаций обучающей выборки (таблица 7.1). Процедура *CLS* позволяет выявить сложную логическую структуру принимаемых решений ЛПР по управлению объектом, оперируя со сравнительно небольшим количеством технологических ситуаций.

Сформированные понятия могут использоваться для классификации всех ситуаций генеральной совокупности в количестве 3^7. При этом правильность классификации новых ситуаций не гарантируется из-за отсутствия в исходной выборке примеров выбора управляющих воздействий по таким ситуациям, определяемым значениями признаков. Ошибочная классификация вновь встретившихся ситуаций требует перестройки дерева классификации управляющих воздействий. Любая, возникшая в технологическом процессе ситуация, для которой обнаруживается несоответствие, требует классификации ее экспертом и внесения в качестве дополнений к исходной выборке.

Следует ожидать, что достаточен некоторый объем выборки (заранее неопределенный, но малый по сравнению с полной совокупностью ситуаций), который приведет к формированию понятия, правильно классифицирующих всю совокупность ситуаций. Такой вывод согласуется с литературными [20] и экспериментальными [19] данными.

Задача принятия решений в конкретной ситуации трактуется как поиск решения по дереву классификации ситуаций. Так для ситуации, характеризующейся значениями признаков: температура газовой среды в ванной печи ниже нормы $\Theta_{гс}$=-1, температура стекломассы в норме $\Theta_{см}$=0, расход газа в норме $Q_г$=0, граница варочной шихты укоротилась $L_ш$=-1, граница варочной пены в норме $L_п$=0, число работающих машин вертикального вытягивания в норме M=0. По дереву классификации управляющего воздействия (рисунок 7.1) находим решение - увеличить расход воздуха $\mu_в$= (+). Такой ситуации в таблице 7.1 нет, что подтверждает возможность использования сформулированного понятия (дерева классификации ситуаций) для классификации ситуаций, отсутствующих в обучающей выборке.

Тестовые задания

1. Основные предпосылки, лежащие в основе ситуационного моделирования.

2. Сущность метода ситуационного моделирования и управления.

3. Трактовка задачи принятия решений.

4. Содержание машинной процедуры формирования понятий *CLS*-9 и применение ее для построения ситуационной модели управления.

5. Формирование дерева для управляющего воздействия. Выбор признаков и их значений.

6. Алгоритм построения дерева классификации управляющего воздействия.

Лекция 8 **Имитационное моделирование**

8.1 Аналитическое и имитационное моделирование

Существует два подхода к построению модели: аналитическое и имитационное моделирование.

Аналитическое моделирование основано на косвенном описании моделируемого объекта с помощью набора математических формул. При этом предполагается использование математической модели реального объекта в форме алгебраических, дифференциальных, интегральных, и других уравнений, связывающих выходные переменные с входными. Вводится система ограничений. Обычно предполагается наличие однозначной вычислительной процедуры получения точного решения уравнений.

Язык аналитического описания содержит следующие основные группы семантических элементов: критерий, неизвестные, данные, математические операции, ограничения. Наиболее существенно то, что аналитическая модель, вообще говоря, не является структурно подобной объекту моделирования. Под структурным подобием тут понимается однозначное соответствие элементов и связей модели элементам и связям моделируемого объекта. К аналитическим моделям относятся модели, построенные на основе аппарата математического программирования, корреляционного, регрессионного анализа.

Имитационное моделирование основано на прямом описании моделируемого объекта. Существенной характеристикой таких моделей является структурное подобие объекта и модели. Это значит, что каждому существенному с точки зрения решаемой задачи элементу объекта ставится в соответствие элемент модели. При этом описываются законы функционирования каждого элемента объекта и связи между ними. Работа с имитационной моделью заключается в проведении имитационного эксперимента. Процесс, протекающий в модели в ходе эксперимента, подобен процессу в реальном объекте. Поэтому исследование объекта по его имитационной модели сводится к изучению характеристик процесса, протекающего в ходе эксперимента.

Для формального представления реальной системы при имитационном моделировании обычно используется схема с дискретными событиями. При этом процесс функционирования системы во времени отождествляется с последовательностью событий, возникающих в системе в соответствии с закономерностями её функционирования. В формальное понятие события вкладывается конкретное смысловое содержание, определяемое целями моделирования.

При имитационном моделировании используемая математическая модель воспроизводит логику (алгоритм) функционирования исследуемой системы во времени при раз-

личных сочетаниях значений параметров системы и внешней среды. Это наблюдение поведения модели системы под влиянием входных воздействий.

Для компьютерного моделирования - математического моделирования с использованием средств вычислительной техники, характерна следующая последовательность действий:

1) определение цели моделирования;
2) разработка концептуальной модели - абстрактной модели, определяющей структуру моделируемой системы, свойства ее элементов и причинно-следственные связи, присущие системе и существенные для достижения цели моделирования;
3) формализация модели;
4) программная реализация модели;
5) планирование модельных экспериментов;
6) реализация плана эксперимента;
7) анализ и интерпретация результатов моделирования.

Когда в данной работы говорится об эксперименте, то речь идет об оценке эффективности системы с помощью имитационного моделирования, которое представляет собой наблюдение поведения модели системы под влиянием входных воздействий. При этом часть из них (а может быть, и все) носят случайный характер. В результате такого наблюдения исследователь получает набор экспериментальных данных, на основе которых могут быть оценены характеристики системы.

*Имитационная модел*ь. Данная разновидность моделей неразрывно связана с идеей машинного эксперимента. Собственно, имитационная модель — это модель комплексная, к которой не предъявляется строгих требований к применению моделей какого-то заданного типа. Идеология многомодельного исследования целиком основывается именно на этом типе моделей.

Имитационная модель — это комплексное логико-математическое представление системы, реализованное в виде программы, предназначенной для решения на ЭВМ, включающее в себя модели различного типа, и рассматривающее аспект функционирования динамической системы во времени. Данный класс моделей применяется при невозможности строгого аналитического решения задачи или проведения натурного эксперимента. Имитационные модели служат для изучения поведения во времени сложной неоднородной динамической системы, относительно структуры которой существуют точные знания или детализированные гипотезы. Для каждого элемента или подсистемы моделируемой

системы в памяти ЭВМ формируется блок данных, характеризующих ее текущее и предшествующие состояния, блок логических и вычислительных процедур, описывающих изменения критических параметров во времени, а также производятся вычисления этих параметров на основе заданных значений.

Комплекс подпрограмм или относительно автономных программных агентов функционирует под управлением программы-супервизора, осуществляющей диспетчеризацию вызовов, активизирующей и приостанавливающей на время выполнение тех или иных процедур в соответствии с планом машинного эксперимента, имитируя тем самым поведение системы. В результате машинного эксперимента формируются массивы данных о состоянии различных параметров системы в различные моменты времени с привязкой к системным событиям, имитируемым в ходе эксперимента.

При этом программа-супервизор управляет процессом имитации случайных возмущающих воздействий, от которых зависит функционирование системы в целом и ее элементов и подсистем. Широкое применение здесь находит метод Монте-Карло, ранее упоминавшийся нами.

Имитационная модель — это инструмент исследования, посредством которого могут осуществляться и манипуляции с масштабом времени функционирования модели. Различают имитационные модели, функционирующие как в натуральном, так и в замедленном или ускоренном масштабе времени. Это является крайне важным при анализе поведения систем, для наблюдения которых отсутствует возможность воспользоваться натуральным масштабом времени. К разряду таких систем могут быть отнесены экосистемы, популяции, системы, в которых протекают скоротечные физические процессы и иные.

Остановимся более подробно на понятии *статистический эксперимент*.

В его основе лежит метод статистических испытаний (метод Монте-Карло). Суть его состоит в том, что результат испытания ставится в зависимость от значения некоторой случайной величины, распределенной по заданному закону. Поэтому результат каждого отдельного испытания также носит случайный характер.

Проведя серию испытаний, получают множество частных значений наблюдаемой характеристики (т.е. выборку). Полученные статистические данные обрабатываются и представляются в виде соответствующих численных оценок интересующих исследователя величин (характеристик системы).

Теоретической основой метода статистических испытаний являются предельные теоремы теории вероятностей (теорема Чебышева, теорема Пуассона, теорема Бернулли).

Принципиальное значение предельных теорем состоит в том, что они гарантируют высокое качество статистических оценок при весьма большом числе испытаний.

Важно отметить, что метод статистических испытаний применим для исследования как *стохастических* , так и детерминированных *систем.*

Еще одной важной особенностью данного метода является то, что его реализация практически невозможна без использования компьютера.

Таким образом, статистический эксперимент является основой для оценки характеристик системы при имитационном моделировании. Он представляет собой наблюдение за поведением системы в течение некоторого промежутка времени.

8.2 Область применения имитационных моделей

При разработке имитационной модели остаются справедливыми принципы моделирования, рассмотренные ранее.

В качестве следствия из этого утверждения необходимо отметить два важных обстоятельства:

взаимосвязь между отдельными элементами системы. Описанными в модели, а также между некоторыми величинами (параметрами) может быть представлена в виде аналитических зависимостей;

модель можно считать реализуемой и имеющей практическую ценность, только в том случае, если в ней отражены лишь те свойства реальной системы, которые влияют на значение выбранного показателя эффективности.

Поскольку основой имитационного моделирования является метод статистических испытаний, то наибольший эффект от его применения достигается при исследовании сложных систем, на функционирование которых существенное влияние оказывают случайные факторы.

Применение имитационного моделирования целесообразно также в следующих случаях:

если не существует законченной постановки задачи на исследование и идет процесс познания объекта моделирования;

если характер протекающих в системе процессов не позволяет описать эти процессы в аналитической форме;

если необходимо наблюдать за поведением системы (или отдельных ее компонентов) в течение определенного периода, в том числе с изменением скорости протекания процессов;

при изучении новых ситуаций в системе либо при оценке функционирования ее в новых условиях;

если исследуемая система является элементом более сложной системы, другие элементы которой имеют реальное воплощение;

когда необходимо исследовать поведение системы при введении в нее новых компонентов;

при подготовке специалистов и освоении новой техники (в качестве тренажеров).

Приведенный список возможных областей применения имитационных моделей можно рассматривать и как перечень их достоинств, но, к сожалению, имитационные модели имеют и ряд недостатков. Первый, и весьма существенный, заключается в том, что разработка имитационных моделей, как правило, требует больших затрат времени и сил. Кроме того, любая имитационная модель сложной системы значительно менее объективна, чем аналитическая модель, поскольку она, прежде всего, отражает субъективные представления разработчика о моделируемой системе. И, наконец, результаты имитационного моделирования, как и при любом численном методе, всегда носят частный характер. Для получения обоснованных выводов необходимо проведение серии модельных экспериментов, а обработка результатов требует применения специальных статистических процедур.

8.3 Статистические теоретико-вероятностные модели

Статистические и теоретико-вероятностные методы составляют методологическую основу одноименного вида моделирования [8]. На этом уровне формализации модели речь о вскрытии закона, обеспечивающего устранение неопределенности при принятии решения, пока еще не идет, но существует некоторый массив наблюдений за данной системой или ее аналогом, позволяющих сделать некие выводы относительно прошлого/текущего/будущего состояния системы, основываясь на гипотезе об инвариантности ее поведения.

Как всегда, сформулируем определение. *Статистическая или теоретико-вероятностная модель (стохастическая модель) — это модель, в которой обеспечивается учет влияния случайных факторов в процессе функционирования системы, основанная на применении статистической или теоретико-вероятностной методологии по отношению к повторяющимся феноменам*. Данная модель оперирует количественными критериями при оценке повторяющихся явлений и позволяет учитывать их нелинейность, динамику, случайные возмущения за счет выдвижения на основе анализа результатов наблюдений гипотез о характере распределения некоторых случайных величин, сказывающихся на поведении системы.

По существу, теоретико-вероятностные и статистические модели отличаются уровнем неопределенности знаний о моделируемой системе, существующей на момент синтеза модели. В случае, когда представления о системе носят, скорее, теоретический характер и основываются исключительно на гипотезах о характере системы и возмущающих воздействий, не подкрепленных результатами наблюдений, теоретико-вероятностная модель является единственно возможной. Когда же на этапе синтеза модели уже существуют данные, полученные опытным путем, появляется возможность подкрепления гипотез за счет их статистической обработки. Это становится очевидным, если рассмотреть соотношение между методами математической статистики и теории вероятностей. Математическая статистика — это наука, изучающая методы вскрытия закономерностей, свойственных большим совокупностям однородных объектов или событий, на основании их выборочного обследования (либо большим массивам данных, полученных в результате наблюдения за одним и тем же объектом на протяжении достаточно протяженного интервала времени). Теория же вероятностей изучает количественные закономерности, которым следуют случайные явления, если эти явления определяются событиями известной вероятности. Соответственно, математическая статистика является связующим звеном между теорией вероятностей и явлениями реального мира, поскольку позволяет сформулировать оценки вероятности тех или иных событий на основе анализа статистических данных.

Можно утверждать, что статистические модели представляют собой особый вид математических моделей, использующих в качестве исходных данных не только актуальные данные о текущем состоянии объекта, но и данные, характеризующие состояние либо других объектов данного класса, либо этого объекта, но в иной момент времени. Статистические модели применимы для изучения массовых явлений любой природы, включая и те, которые не относятся к категории вероятностно определенных (математическая статистика приспособлена и для решения детерминированных задач). При моделировании последних статистический процесс вводится в модель искусственно для получения статистических оценок численного решения (например, точности измерения параметров детерминированного процесса).

Методы математической статистики и теории вероятности могут вводиться, в том числе, и в логические и логико-лингвистические модели, как это было указано в предыдущем подразделе. Например, могут рассматриваться методы интеграции статистических оценок в модели семантических отношений для придания различных весов дугам, связывающим отдельные вершины. Статистические оценки могут быть внедрены и в системы представления тезаурусов для разрешения ситуаций полисемии без обращения к процеду-

рам контекстного анализа. Иными словами, статистические методы могут составлять как основу модели, так и применяться для модификации моделей других типов.

Для обработки результатов наблюдений используются методы корреляционного, регрессионного, факторного, кластерного и иных видов анализа, оперирующих статистическими гипотезами. Особая роль здесь отводится *методу статистических испытаний* (*методу Монте-Карло*). Это метод численного решения математических задач, основанный на многократном теоретико-вероятностном и статистическом моделировании случайных величин или процессов с целью построения статистических оценок для искомых величин. Сущность метода состоит в реализации многократного моделирования случайного явления с помощью некоторой процедуры, дающей случайный результат. Для этого с применением ЭВМ создается некоторое множество реализаций случайных процессов, моделирующих возмущающие воздействия на исследуемый объект или процесс, после чего производится моделирование этого процесса или объекта в условиях, определяемых полученными случайными воздействиями. Результаты такого моделирования обрабатывают с использованием методов математической статистики. При этом могут варьироваться тип и параметры распределения случайной величины.

Реализация случайного процесса методом Монте-Карло представляет собой последовательность розыгрышей единичных жребиев, перемежающихся обычными расчетами, в ходе которых определяется результат возмущающего воздействия на объект или процесс, на исход операции.

Поскольку адекватность модели распределения случайных воздействий в общем случае установить трудно, задачей моделирования с применением метода Монте-Карло является обеспечение *робастности полученных решений (устойчивости к изменению параметров закона распределения случайных величин и начальных условий моделирования)*. Если результат моделирования не является робастным (существенно зависит от параметров закона распределения и параметров модели), то это свидетельствует о наличии высокого риска при принятии решения в данной реализации моделируемой системы.

Важную роль в статистических моделях играют гипотезы о характере процессов смены состояний в моделируемой системе. Так, например, весьма интересный случай представляет собой гипотеза о «*марковости*» процессов (получившая название в честь русского ученого А.А. Маркова — начало XX века). *Марковские процессы представляют собой случай процесса с детерминированными вероятностями, для которого ранняя предыстория смены состояний системы на некотором предшествующем интервале времени несущественна для установления вероятности наступления следующего события —*

основное значение придается ее текущему состоянию. Если существует уверенность в марковости процесса, это существенно меняет представления о системе (она может рассматриваться как «инерционная», в большой степени зависящая от текущего ее состояния и характера возмущающего воздействия). Принцип марковости был открыт при анализе текстов на естественных языках, где вероятность появления следующего символа может быть предсказана на основе статистического анализа текстовых массивов, на данном конкретном языке.

Статистическое моделирование тесно сопряжено с имитационным моделированием, в ходе которого модель объекта нередко «погружается в вероятностную (статистическую) среду», в которой проигрываются различные ситуации и режимы функционирования модели/объекта. Однако имитационные модели могут реализовываться и в детерминированных средах.

Методы статистического моделирования широко распространены в сфере стратегического планирования и управления. Широкому распространению методов статистического моделирования в сфере оперативного управления препятствует высокая трудоемкость процесса моделирования. В основном это связано с необходимостью глубокой математической проработки моделей и высокими требованиями, предъявляемыми к математическим познаниям пользователей.

8.4 Обобщенные модели

Наиболее известный в России подход к формальному описанию процессов функционирования сложных систем принадлежит Н.П. Бусленко [1]. Этот подход позволяет описывать поведение непрерывных и дискретных, детерминированных и *стохастических систем* , т.е. он является обобщенным и базируется на понятии агрегативной системы (англ. *aggregate system*), так называемой *А*-схемы.

Многое говорит за справедливость утверждения, что моделирующие системы должны иметь в своей основе единую формальную математическую схему, т.е. *А*-схему. Такая схема должна одновременно выполнять несколько функций:

- являться адекватным математическим описанием объекта моделирования;

- служить основой для построения алгоритмов и программ при машинной реализации модели;

- позволять в упрощенном варианте (для частных случаев) проводить аналитические исследования.

В качестве элементов *А*-схемы выступают агрегаты, а связь между агрегатами (внутри системы) и с внешней средой осуществляется с помощью оператора сопряжения

R. Вообще говоря, и сам агрегат может рассматриваться как *A*-схема, т.е. может разбиваться на элементы (агрегаты) следующего уровня.

Любой агрегат характеризуется следующими множествами:

- моментов времен *T*;
- входных сигналов *X*;
- выходных сигналов *Y*;
- состояний *Z* в каждый момент времени *t*;
- случайный оператор, описывающий процесс функционирования *V*;
- случайный оператор, описывающий скачки состояний в особые моменты времени t_δ *W*;
- оператор, описывающий момент выдачи выходного сигнала *G*.

Таким образом, под *агрегатом* понимается любой объект, определяемый упорядоченной совокупностью рассмотренных множеств *T, $Z^{(Y)}$, Y, X,* Z, *H* и случайных операторов *V, U, W, G*.

Последовательность входных сигналов, расположенных в порядке их поступления в *A*-схему, называется *входным сообщением* или *x*- сообщением. Последовательность выходных сигналов, упорядоченная относительно времени выдачи, называется *выходным сообщением* или *y*-сообщением.

Конструкция из отдельных агрегатов, составляющая модель сложной системы, называется агрегативной системой или *A*-схемой. Для ее описания необходимо иметь описание как отдельных агрегатов, так и связей между ними.

Вообще говоря, любая система, при моделировании которой важно рассматривать как возможность непрерывного изменения ее состояния, так и изменение ее состояния скачком, является агрегатом или агрегативной системой.

В этом отношении система массового обслуживания при моделировании ее по принципу Δt (изменение времени с постоянным шагом) может рассматриваться как агрегативная система. В этом случае непрерывными переменными состояния являются времена, оставшиеся до окончания обслуживания, до момента поступления заявки, до момента отказа и т.д. В качестве дискретно изменяющихся переменных состояния могут рассматриваться число заявок в накопителях, состояния каналов и т.д. Входные и выходные сигналы - это собственно информация о поступлении в систему или выходе из системы заявок. Соответственно это описывается всеми приведенными выше операторами.

К агрегативным системам следует также отнести и некоторые, вообще говоря, непрерывные системы, параметры которых могут изменяться скачком в зависимости от по-

ложения изображающей точки в фазовом пространстве или от входного сигнала. Сюда можно отнести системы с переменной структурой, многорежимные системы управления, системы, функционирующие в условиях различного рода отказов и т.д. В частности, сюда можно отнести, исходя из этих же соображений, и оптимальные по быстродействию системы управления.

8.5 Стохастическое программирование. Методы решения задач стохастического программирования

Стохастическим программированием называют метод решения оптимизационных задач, в которых целевая функция недоступна для вычисления в чистом виде, а может быть оценена или вычислена с погрешностью, например из эмпирических наблюдений или экспериментов. Стохастическое программирование может применяться для решения задач оценки параметров стохастических объектов, управления, динамической идентификации, однако, в отличие от прямых методов решения этих задач, стохастическое программирование не требует хранения всей накопленной информации об объекте, а предлагает способ адаптивной коррекции оценки по данным следующего наблюдения. Рассматриваются такие методы как Робинса–Монро, Киффера–Вольфовица, конечносходящиеся алгоритмы, методы случайного поиска. Кроме того, рассматриваются нейросетевые и генетические алгоритмы, как частные случаи алгоритмов стохастического программирования для решения специфических задач.

В *одних случаях* опыт, статистика и изучение процессов, определяющих изменение исходных данных и формирующих условия, в которых реализуется план, проект или система управления, позволяют устанавливать те или иные *вероятностные характеристики параметров целевой функции и ограничений задачи. В других случаях* нет оснований, для каких бы то ни было *суждений о статистических особенностях явлений*, способных изменить предполагаемые значения параметров условий задачи. Ситуации первого типа называются ситуациями, связанными с риском, а ситуации второго типа - неопределенными. И те, и другие являются предметом исследования стохастического программирования—раздела математического программирования, изучающего *теорию и методы решения условных экстремальных задач при неполной информации о параметрах условий задачи.*

Постановки задач стохастического программирования существенным образом зависят от целевых установок и информационной структуры задачи.

В приложениях стохастическое программирование используется для решения задач двух типов. В задачах *первого типа* прогнозируются статистические характеристики

поведения множества идентичных экстремальных систем. Соответствующий раздел стохастического программирования называется *пассивным стохастическим программированием* . Модели *второго типа* предназначены для построения методов и алгоритмов планирования и управления в условиях неполной информации. Соответствующий раздел стохастического программирования называется *активным стохастическим программированием* , подчеркивая этим действенную целевую направленность моделей. Подходы к постановке и анализу стохастических экстремальных задач существенно различаются в зависимости от того, получена ли информация о параметрах условий задачи (пли об их статистических характеристиках) в один прием или по частям (в два или более этапов).

При построении *стохастической модели* важно также знать, необходимо ли единственное решение, не подлежащее корректировке, или можно по мере накопления информации один или несколько раз подправлять решение. Другими словами, речь идет о том, какая задача рассматривается: статическая или динамическая. В соответствии с этим в *стохастическом программировании исследуются одноэтапные, двухэтапные и многоэтапные задачи.*

Статические, или одноэтапные, задачи стохастического программирования представляют собой естественные стохастические аналоги детерминированных экстремальных задач, в которых динамика поступления исходной информации не играет роли, а решение принимается один раз и не корректируется. Одноэтапные стохастические задачи, как те, что порождены детерминированными моделями стохастического программирования, так и те, что имеют смысл только при случайных параметрах условий, различаются характером ограничений и выбором целевой функции.

Разработка предварительного плана и компенсация невязок - два этапа решения одной задачи. В соответствии с этим задачи рассматриваемого типа называют *двухэтапными задачами стохастического программирования* . Естественным обобщением двухэтапных задач являются многоэтапные (динамические) задачи стохастического программирования. Часто в процессе управления представляется возможность последовательно наблюдать ряд реализаций параметров условий и соответствующим образом корректировать план. Естественно, что как предварительный план, так и последовательные корректировки должны, помимо содержательных ограничений, учитывать априорные статистические характеристики случайных параметров условий на каждом этапе. К анализу многоэтапных задач стохастического программирования сводятся формальные исследования численных методов планирования производства и развития экономической системы. Роль стохастических моделей и методов в исследовании закономерностей пове-

дения экономических систем и в разработке количественных методов планирования экономики и управления производством имеет два аспекта — методологический и вычислительный. И тот и другой связаны с одной из важнейших категорий современной математической логики — с понятием сложности, точнее, *с понятиями «сложность алгоритма», «сложность вычислений» и «сложность развития».*

Роль вычислительного аспекта проблемы определяется тем, что планирование, управление и проектирование происходят, как правило, в условиях неполной информации. Рыночная конъюнктура, спрос на продукцию, изменения в состоянии оборудования не могут быть точно предсказаны. В условиях конкурентной экономики дополнительно возникает направленная дезинформация.

Учет случайных факторов и неопределенности в планировании и управлении — важная задача стохастического программирования . Однако этим не исчерпывается роль стохастических методов в экономическом анализе. *Принципы стохастического программирования дают основание для сопоставления затрат на накопление и хранение информации с достигаемым экономическим эффектом, позволяют аргументировать рациональное разделение задач между человеком и вычислительной машиной и служат теоретическим фундаментом для алгоритмизации управления сложными системами.*

Принципы стохастического программирования позволяют сблизить точные, но узко направленные формальные математические методы с широкими, но нечеткими содержательными эвристическими методами анализа. И здесь, таким образом, мы переходим *к методологической роли стохастического программирования в исследовании сложных систем*. В связи с оценками сложности алгоритмов и вычислений представляет смысл условно разделить *задачи планирования*, управления и проектирования на *задачи вычислительного и не вычислительного характера*. Многие задачи управления, должны быть отнесены к классу задач не вычислительного характера. Т.о. необходимо согласование сложности управляемого объекта и управляющего устройства за счет рационального упрощения объекта (разумной переформулировки задачи).

Основные классы задач, для решения которых создается вычислительный комплекс, непосредственно или методами стохастического расширения формулируются как модели стохастического программирования .

Вообще говоря, все модели выбора решения, сформулированные в терминах математического программирования, могут быть (а в практических задачах, отвечающих управлению сложными системами и процессами, должны быть) сформулированы как модели стохастического программирования .

Соответствие формально построенных стохастических моделей содержательным постановкам - решающее условие успешного управления в условиях неполной информации. Вряд ли могут быть приведены универсальные рекомендации по выбору информационной структуры модели и статистических характеристик, используемых для формирования целевого функционала задачи и области его определения.

Анализ опыта решения практических экстремальных задач методами математического программирования свидетельствует о серьезных успехах этого подхода (и о внедрении данных методов в практику планирования, управления и проектирования) в задачах относительно простой структуры, главным образом одно экстремальных, при не слишком большой размерности задачи, когда число переменных и ограничений (в моделях достаточно общего вида) не превышает сотен или тысяч. Однако методы детерминированного математического программирования не прививаются в системах большой сложности, отвечающих многоэкстремальным задачам или задачам большой размерности.

До сих пор нет достаточно конструктивного метода решения общей (даже линейной) двухэтапной задачи стохастического программирования . Стандартные методы выпуклого программирования в общем случае неприменимы для вычисления предварительного плана — решения выпуклой задачи первого этапа. Основная трудность в том, что целевая функция и область определения планов первого этапа заданы, вообще говоря, неявно. В случаях, когда область K имеет относительно простую структуру или задача оказывается с простой рекурсией, эффективным, хотя и трудоемким методом вычисления предварительного плана, оказывается метод стохастических градиентов [21], представляющий собой итеративный метод типа стохастической аппроксимации.

Все это подсказывает путь алгоритмизации решения сложных задач в автоматизированных системах управления - замену трудоемких процедур, отвечающих обоснованным (точным или приближенным) методам решения детерминированных экстремальных задач, относительно простыми «законами управления» - решающими правилами или решающими распределениями стохастического расширения соответствующих задач.

Платой за упрощение задачи и за переход от громоздких алгоритмов к относительно простым решающим механизмам служат трудоемкая предварительная работа по построению «законов управления» и некоторая потеря эффективности решения задачи в каждом отдельном случае.

В литературе по стохастическому программированию описаны многочисленные модели выбора решений, сформулированные в терминах стохастического программирования . Разнообразные задачи управления запасами - классические примеры стохастиче-

ских моделей. Синтез систем массового обслуживания, удовлетворяющих заданным требованиям и оптимизирующих пропускную способность системы или определяемый ею доход, сводится к решению экстремальных стохастических задач.

Тестовые задания

1. Имитационная модель и имитационное моделирование.

2. Последовательность действий при компьютерном моделировании. Моделирующие программы, их состав.

3. Область применения имитационных моделей.

4. Статистические теоретико-вероятностные модели.

5. Обобщенные модели. Агрегативные системы *A*-схемы.

6. Стохастическое программирование. Методы решения задач стохастического программирования.

Лекция 9 **Имитационное моделирование непрерывных процессов**

9.1 Описание моделирующей системы

Математическое моделирование с использованием компьютерной техники играет важную роль в совершенствовании управления технологическими процессами. Ниже описывается методика использования вычислительного эксперимента и имитационного моделирования для оценки эффективности управления технологическим процессом производства листового стекла и выработки предложений по совершенствованию алгоритмов управления с целью дальнейшего улучшения качества вырабатываемой продукции [19].

Предлагаемая методика состоит из следующих этапов:

1) разработка регрессионных моделей, описывающих зависимость показателей качества вырабатываемой продукции от режима работы технологического оборудования;

2) формализация задачи управления в пространстве режимных переменных технологического оборудования; выбор критериев управления и ограничений;

3) имитационное моделирование системы управления с выбранным алгоритмом управления; оценка достигаемого улучшения качества продукции;

4) повторение этапов 2 и 3 при других формулировках задач управления и алгоритмов их решения; выбор рационального варианта постановки и решения задачи управления качеством продукции;

5) выработка предложений по совершенствованию алгоритмов управления технологическим процессом для дальнейшего улучшения качества вырабатываемой продукции.

Достоверность результатов моделирования во многом зависит от точности регрессионных моделей, используемых при моделировании. Для повышения точности параметры моделей уточняются с помощью алгоритма адаптации по фактическим данным режимных переменных и показателям качества продукции.

Используемые при моделировании регрессионные уравнения имеют линейную структуру, которые формально можно записать в следующем виде:

$$y(\mathrm{t}) = b_0(t) + b_1(t) \cdot x_1(t-\tau_1) + \ldots + b_k(t) \cdot x_k(t-\tau_k), \qquad (9.1)$$

где $b_i(t)$, $i=0,1,..,k$ - параметры модели; t - текущее время; k - количество факторов в модели; x_i - входные переменные; τ_i - запаздывание по входным каналам; $y(t)$ - выходная переменная.

Точность регрессионного уравнения оценивалась величиной абсолютной погрешностью $\Delta y(t)$, вычисляемой по формуле:

$$\Delta y(t) = y_{\phi}(t) - y(t), \qquad (9.2)$$

где $y_{\phi}(t)$, $y(t)$ – фактическое и рассчитанное по регрессионному уравнению значение показателя качества продукции в момент времени t.

При превышении ошибки допустимой величины происходит инициирование алгоритма адаптации (Рис. 9.1).

Алгоритм адаптации, используя информацию о входных и выходных переменных, значения текущих параметров модели и ее ошибки, проводит коррекцию коэффициентов модели по формуле :

$$b_i(t) = b_i(t\text{-}1) + (y(t) - \sum_{i=0}^{k} b_i(t\text{-}1)\, x_i(t\text{-}\tau_i)) / (\gamma + \sum_{i=0}^{k} x_i^{\,2}(t\text{-}\tau_i))\, x_i(t\text{-}\tau_i)\,, \qquad (9.3)$$

где τ_i - запаздывание по каналу; γ - параметр алгоритма адаптации.

Сходимость алгоритма обеспечивается выбором значения параметра γ и начальных значений коэффициентов регрессии. Значение γ подбирается экспериментально. При отсутствии помех в измерении входных переменных принимается γ=0. Для модели с неизменяемыми значениями параметров γ=∞. В качестве начальных значений коэффициентов адаптивной модели выбираются значения коэффициентов из уравнения регрессии.

С помощью машинного эксперимента, реализованного на контрольной выборке, уточнялись статистические данные о работоспособности регрессионных моделей (дисперсия погрешности модели, периодичность коррекции, диапазон изменения коэффициентов и др.).

После создания адекватных моделей разрабатывалась имитационная модель управления производством. Инерционность объекта управления, низкочастотный спектр возмущающих воздействий позволяют выбрать дискретность управления процессом: один раз в сутки корректируется режим работы технологического оборудования. Между интервалами управления выбранный режим стабилизируется.

Анализ случайных процессов изменения выходных переменных, инерционность каналов управления по режимным переменным, спектральные характеристики возмуща-

ющих воздействий позволяют отнести рассматриваемую задачу к числу задач статического планирования и управления.

Реализуемость статического подхода к решению задач управления процессом подтверждается тем, что действующие возмущающие воздействия компенсируются при помощи режимных переменных значительно быстрее, чем успевают изменяться значения самих возмущений.

Полученные регрессионные уравнения представляют собой приближенную модель объекта, позволяющие рассчитать выходные показатели технологического процесса с допустимой небольшой ошибкой. Аппроксимация динамических моделей каналов режимных переменных характеристиками звена чистого запаздывания позволяет использовать для математического описания статического режима работы технологической линии систему алгебраических уравнений. В этом случае выходные показатели могут рассчитываться по входным переменным с учетом эквивалентных времен запаздывания.

При решении задачи управления с использованием регрессионных уравнений может оказаться, что технологический режим, выбранный в результате решения задачи с приближенной моделью процесса, в действительности не будет обеспечивать выработку продукции заданного качества.

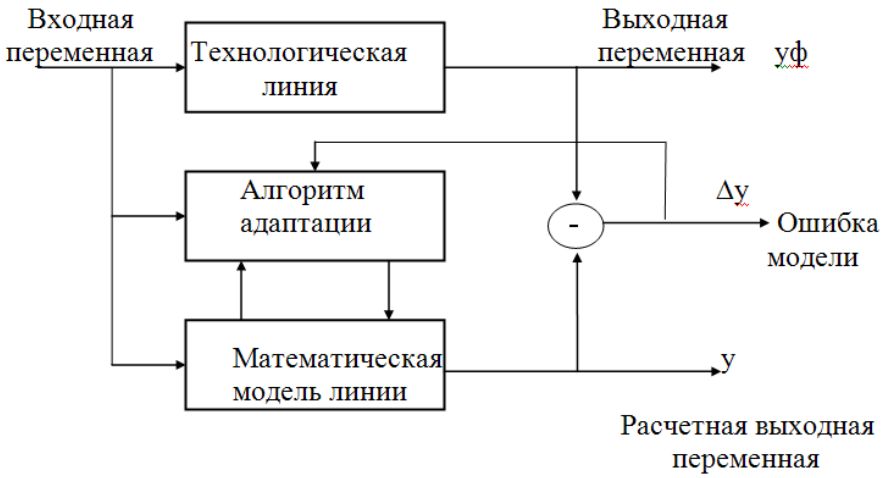


Рисунок 9.1- Схема коррекции параметров регрессионного уравнения

Организация управления процессом с заданной надежностью (вероятностью) возможна двумя способами:

1) разработкой адаптивных математических моделей;

2) разработкой алгоритмов управления, учитывающих вероятностный характер полученного математического описания и использующих некоторую дополнительную информацию о процессе.

При моделировании использован первый подход - разработка моделей с заданной точностью и использование детерминированных алгоритмов для решения статической задачи управления (планирования) процессом. Этот подход отличается простотой и ясностью алгоритмов управления. При адекватных моделях он позволяет получать решения, близкие к результатам, полученным с использованием сложных алгоритмов, используемых при решении стохастических задач.

Использование машинных имитационных моделей позволяет получать информацию, дополняющую результаты реальных испытаний системы управления и оценивать эффективность функционирующей системы. При создании имитационной модели ориентировались на использование экспериментальных данных по входным и выходным переменным, собираемых с объекта. Экспериментальные данные позволяют оценить точность соответствия поведения модели поведению реального объекта и при необходимости уточнить модель, оценить эффективность исследуемых алгоритмов управления по сравнению с ручным ведением процесса при одинаковых исходных данных.

Системы автоматизированного управления крупными производствами относятся к числу больших и сложных систем, что усложняет их исследование. Возникшая проблема решается путем сведения задачи большой размерности к последовательному решению нескольких задач малой размерности за счет ее декомпозиции.

Рассмотрим использование имитационного моделирования для оценки эффективности управления технологическим оборудованием на примере производства листового стекла флоат-способом в ОАО «Эй Джи Си Борский стекольный завод» и выработки предложений по коррекции режима работы оборудования, с целью повышения качества вырабатываемого стекла.

Спроектированная СППР предназначена для использования на среднем уровне управления производством листового стекла с непрерывным характером протекающих процессов (Рис.9.2).

Моделирующий программный комплекс позволяет выполнять:

- статистический анализ производства и его прогноз на задаваемый временной интервал;

- проводить многовариантные вычисления алгоритмов управления, оценивать ожидаемые результаты выбираемых управляющих решений.

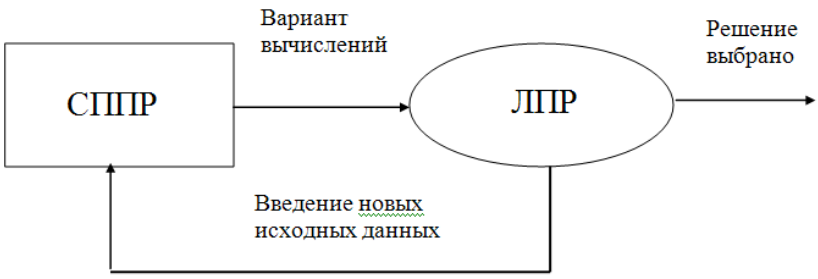


Рисунок 9.2 – Схема формирования выработки решений по управлению

Результаты решений выдаются в графической форме, а также в виде таблиц на экране монитора, либо на принтере. Для поддержки работоспособности системы в реальном масштабе времени требуется периодически вводить показатели работы производства в моделирующую систему. Система работает с ранее занесенными в компьютер данными.

ПК функционирует в *IBM* совместимых компьютерах в операционной среде *Widows*. ПК *состоит* из двух компонент для работы в режиме диалога (Рис.9.3.) и режиме конфигурирования (Рис.9.4) системы.

Оперативная помощь пользователям вызывается в режиме конфигурирования в пункте меню «Помощь». Все файлы, необходимые для работы системы, создаются средствами самого ПК.

Инсталляция ПК проводится в режиме конфигурирования. Для настройки выбирается пункт меню “Файл ”. Выбрав пункт “Параметры ” из меню следующего уровня устанавливаются параметры настройки.

Указывается путь к данным, хранящимся в БД (например *AWS*aws*). Задание пути к файлам и расширения имен файлов позволяет указывать при обращении к элементам БД только имя файла дескриптора элемента.

Конфигурирование базы данных включает в себя создание новых элементов БД, коррекцию, просмотр и удаление существующих элементов. При работе в режиме конфигурирования в пункте меню “Объект ” администратор базы данных может просмотреть имена, типы, имена файлов дескрипторов существующих элементов БД, корректировать информацию, определяющую сущность элементов БД, удалять файлы дескрипторов с диска, а также выполнять функции, определенные для нормальной работы комплекса, без запуска компоненты «Режим диалога».

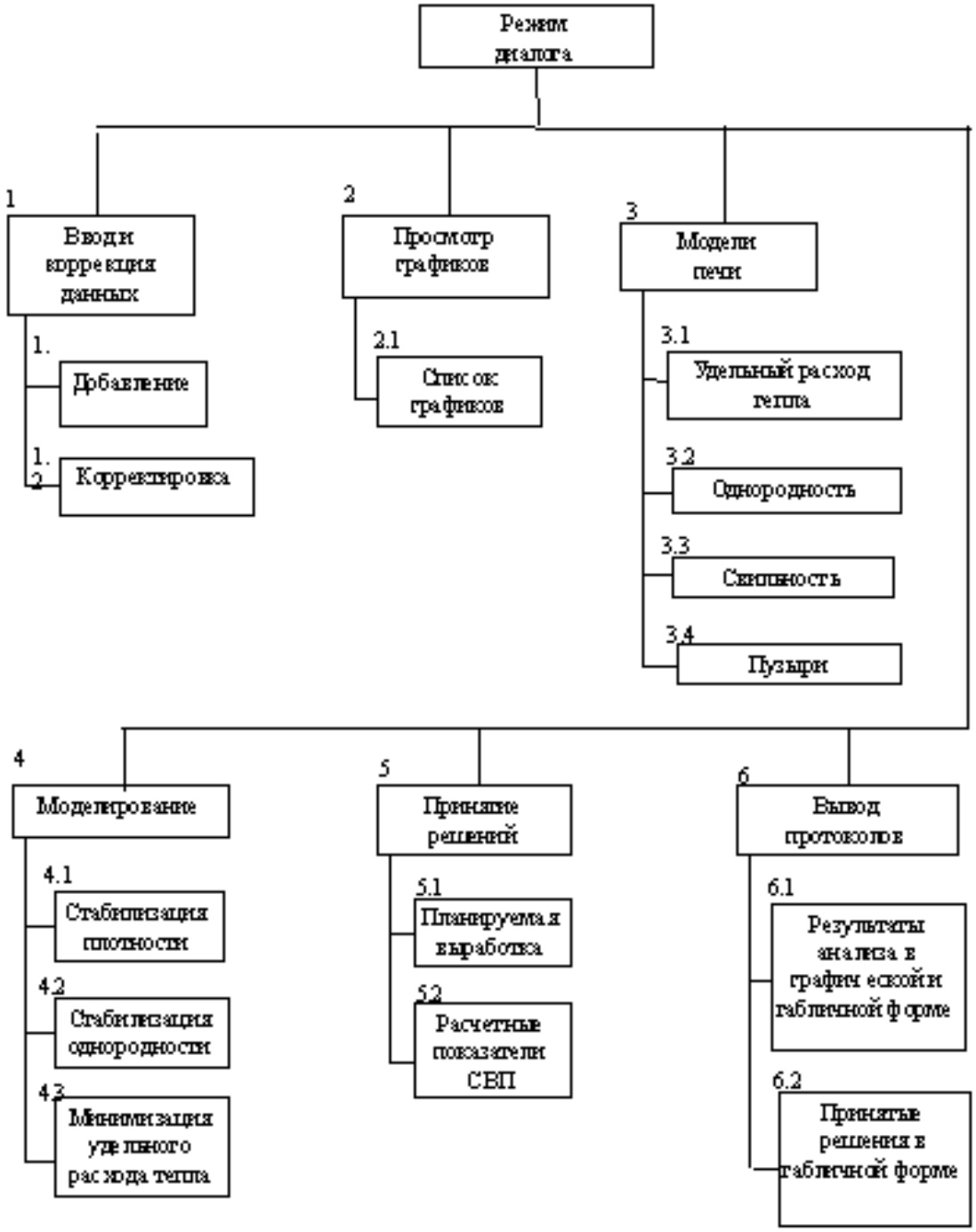


Рисунок 9.3 - Меню пользователя

Для просмотра списка элементов БД вызывается каталог объектов с помощью пункта меню “Просмотр ”. Здесь можно просмотреть списки элементов выбранного типа.

Система поддержки принятия решений конфигурируется под решаемую задачу в режиме Конфигуратор (Рис.9.4). Меню пользователя создается с помощью иконок, расположенных с левой стороны окна, в котором графически отображается иерархическая структура меню пользователя. Иконки имеют следующие названия, соответствующие вы-

полняемым действиям: «Новый пункт меню», «Удаление текущего элемента», «Новое подменю», операции перемещения вверх и вниз элементов списка меню.

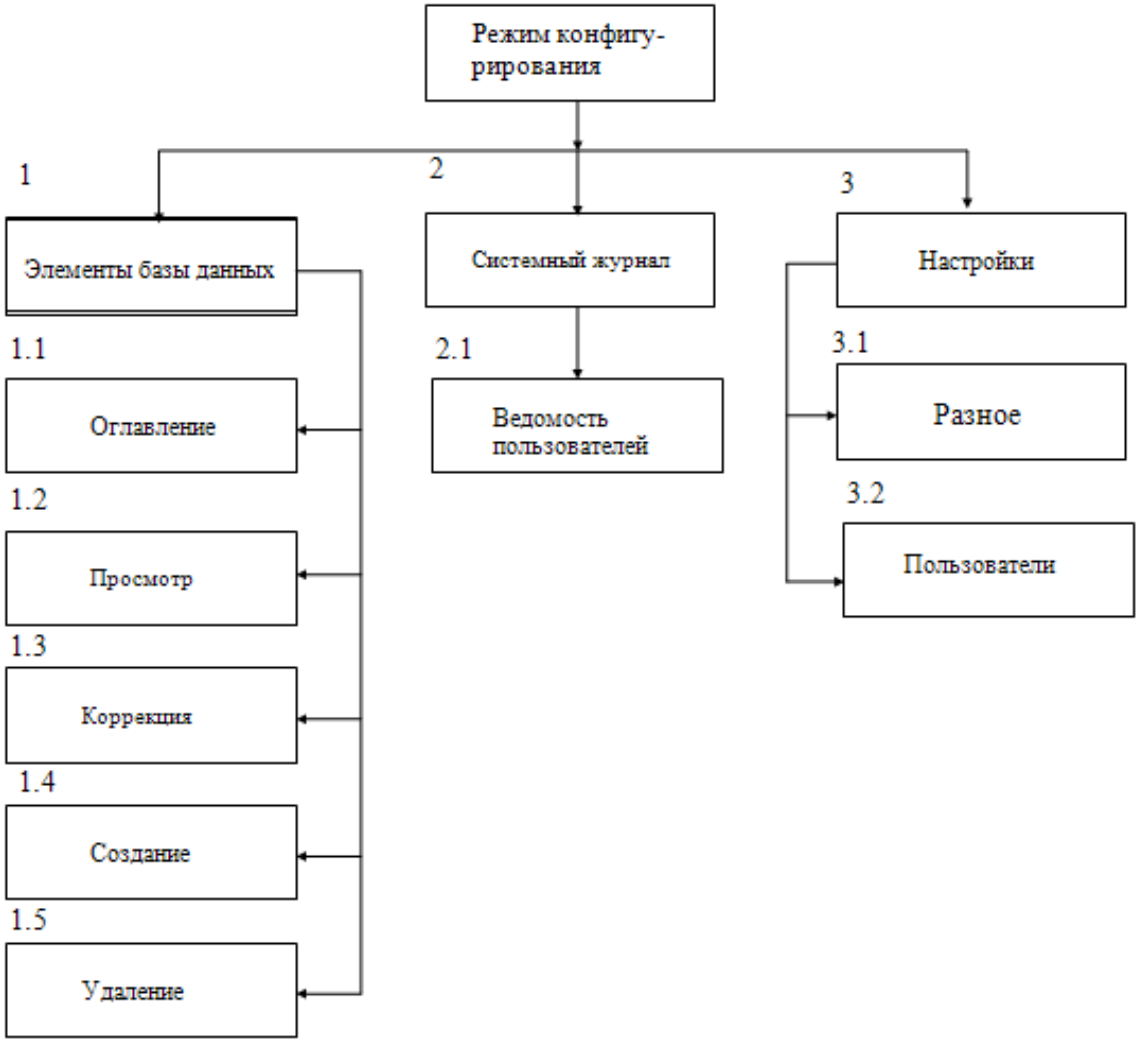


Рисунок 9.4 - Меню режима конфигурирования системы

Содержание пунктов и подпунктов меню заполняется созданными элементами из БД. Перечень элементов отображается в правом окне меню конфигуратора. Манипуляция элементами проводится с помощью кнопок, расположенных с правой стороны названного окна. Кнопки имеют названия, соответствующие выполняемым операциям: «Создать объект», «Подключить объект», Отключить объект». Перемещение объектов в списке осуществляется кнопками, помеченными треугольниками.

Конфигуратор позволяет разрабатывать меню пользователя иерархической структуры. Каждый пункт меню может состоять из подпунктов, а подпункты - в свою очередь из других подпунктов и т.д. Иерархичность структуры меню обеспечивается за счет включения в список объектов в качестве элементов других списков и т.д.

Программное обеспечение моделирующей системы требует распределения оперативной памяти компьютера под функциональные программы системы и базу данных. Память, занимаемая функциональными программами, относительно стабильна, а выделяемая под локальную базу данных - зависит от количества элементов и их размерностей.

В базе данных моделирующей системы содержатся данные регулярного типа для построения моделей. Точность моделей обеспечивается за счет периодической корректировки ее параметров (коэффициентов) с использованием одношагового алгоритма адаптации.

Все модели создают класс элементов БД типа “Формула”. Формула записывается с помощью оператора присваивания. Идентификатор рассчитываемой переменной записывается с левой стороны от оператора присваивания “:=“, а само вычисляемое выражение - с правой стороны. В выражении используются вычислительные функции, а также логические действия, такие как сдвиг данных (*shift*), выбор ближайшего по времени значения переменной (*near*) и др. Алгоритм вычисления значений переменной формирует файл с расчетными данными, образуя переменную регулярного типа.

Адаптация коэффициентов линейных моделей, описывающих зависимость выходных переменных объекта управления от входных переменных, проводится для повышения точности модели. Пусть процесс описывается линейным уравнением в дискретные моменты времени n:

$$y(n)=\Sigma b_i(n)*x_i(n-k_i),\ i=1,2,...m, \qquad (9.4)$$

где $y(n)$ - выходная переменная объекта управления на n-ом такте;

$x_i\ (n-\ k_i)$ - значение i-ой входной переменной в n-ом такте с временным сдвигом на k дискрет;

$b_i(n)$ - изменяющиеся (неизвестные) параметры модели;

m - число входных переменных.

Тогда параметры модели будут уточняться с использованием оптимального одношагового алгоритма адаптации (9.3).

Для повышения эффективности алгоритма адаптации в нем используются стандартизированные значения входных переменных. Стандартизация входных переменных проводится с использованием прошлых и текущих данных. Временной интервал стандартизации T задается установкой одноименного параметра при создании элемента “Адаптация”. Для выполнения процедуры стандартизации необходимо исключить равенство нулю дисперсии входных переменных на интервале времени стандартизации T. Здесь же задает-

ся допустимая абсолютная погрешность адаптивной модели Δy_i в окне «Допуск», а также «Параметр адаптации» γ.

При моделировании используются ретроспективные данные о протекавшем процессе, который управлялся вручную или с использованием иного алгоритма управления.

9.2 Имитационное моделирование технологического процесса стекловарения в производстве листового стекла флоат-способом

Опишем постановку вычислительного эксперимента по оценке эффективности алгоритма стабилизации суточного изменения плотности вырабатываемого стекла. Суточное изменение плотности не должно превышать 0,0005 г/см^3 [19].

При испытании алгоритма управления устанавливался диапазон изменения температуры стекломассы Θ_1 - Θ_5 таким, каким он был при ручном управлении:

$$\begin{aligned} 946\ ^0\text{C} \le \Theta_1 \le 1127\ ^0C\ , \\ 968\ ^0\text{C} \le \Theta_2 \le 1135\ ^0C\ , \\ 986\ ^0\text{C} \le \Theta_3 \le 1143\ ^0C\ , \\ 994\ ^0\text{C} \le \Theta_4 \le 1152\ ^0C. \end{aligned} \qquad (9.5)$$

Одновременно задавались требования к форме температурной кривой по длине ванной печи системой ограничений:

$$\begin{aligned} \Theta_2 - \Theta_1 \ge 10\ ^0C, \\ \Theta_3 - \Theta_2 \ge 10\ ^0C, \\ \Theta_4 - \Theta_3 \ge 10\ ^0C, \\ \Theta_5 - \Theta_4 \ge 10\ ^0C. \end{aligned} \qquad (9.6)$$

Для обеспечения плавности регулирования теплового режима работы печи, ограничивалась величина суточной коррекции температуры стекломассы:

$$\begin{aligned} |\Delta\Theta_1| \le 15\ ^0C, \\ |\Delta\Theta_2| \le 15\ ^0C, \\ |\Delta\Theta_3| \le 15\ ^0C, \\ |\Delta\Theta_4| \le 15\ ^0C. \end{aligned} \qquad (9.7)$$

Задача управления стекловаренной печью сформулирована как одношаговая задача принятия решений по коррекции теплового режима работы печи с периодичностью один раз в сутки.

Критерий управления записывался с помощью штрафной функции [19]. Штраф накладывается за нарушение системы ограничений (9.5) – (9.7), а также превышение суточного изменения плотности вырабатываемого стекла величины 0,0005 г/см3. Поиск оп-

тимального режима по температурам Θ_1 - Θ_4 проводился из условия обеспечения минимального значения штрафной функции. Оптимизация проводилась с использованием модифицированного варианта метода покоординатного спуска с удвоением шага вдоль режимных переменных Θ_1 - Θ_4.

Сравнительные графики результатов моделирования с фактическим режимом работы печи по температуре стекломассы приведены на рис.9.5.

Как видно из рисунка, ручной режим ведения процесса отличается нестабильностью и низкой точностью. Статистические оценки режимов приведены в табл.9.1.

Таблица 9.1 - Сравнение температурного режима управления плотностью стекла с ручным ведением процесса варки

Оценка режима	Ручное ведение процесса				Управление плотностью стекла			
Диапазон изменения, ^0C	946-1127	968-1125	986-1143	994-1152	-	-	-	-
Среднее значение, ^0C	1052,5	1070,8	1082	1093,5	1040	1067	1088	1100
Среднеквадратичное отклонение, ^0C	34,2	31,3	33	33,8	0	0	0	0
Температура	Θ_1	Θ_2	Θ_3	Θ_4	Θ_1	Θ_2	Θ_3	Θ_4

Как видно из графика (рис. 9.5), а также из табличных данных (табл. 9.1) алгоритм управления плотностью стабилизирует тепловой режим работы ванной печи. Расчетные температуры близки к среднеарифметическим значениям фактических температур.

Стабилизация теплового режима работы ванной печи улучшает качество вырабатываемого стекла. Результаты имитационного моделирования алгоритма управления плотностью стекла и достигаемые при этом показатели качества сведены в табл. 9.2.

Таблица 9.2 - Статистические данные результатов моделирования алгоритма управления плотностью вырабатываемого стекла

Дата	Плотность стекла, г/см3		Дефекты 1 и 2-го класса, шт./10 м2		Оптические искажения, угл. град.	
	Средн.	С.К.О.	Средн.	С.К.О.	Средн.	С.К.О.
31.01.03	2, 4899	0,00037	1,07	0,55	63,8	4,6
28.02.03	2, 4942	0,0023	0,84	0,31	63,8	7,2
31.03.03	2, 4917	0,0025	1,78	0,94	59,3	2,7
30.04.03	2, 4897	0,00035	0,88	0,92	65	4,8
31.05.03	2.4893	0,00043	0,75	0,295	70	3,7

30.06.03	2, 4902	0,00032	1,02	0,56	70	2,3
31.07.03	2, 4911	0,00064	0,63	0,34	68,5	4,3
31.08.03	2, 4949	0,0025	0,52	0,13	62	4,85
30.09.03	2, 4942	0,0037	1,15	0,67	61,8	1,7
31.10.03	2, 4919	0,0005	1,1	0,41	62,6	6,5
30.11.03	2, 4924	0,00051	0,66	0,9	65,4	4,8
31.12.03	2, 4929	0,0005	0,77	0,95	64,3	5,1
Среднее арифметическое	2, 4919	0,00244	0,93	0,71	64,7	5,57

Обозначения: Средн. – среднее арифметическое значение параметра;
С.К.О. – среднеквадратичное отклонение параметра.

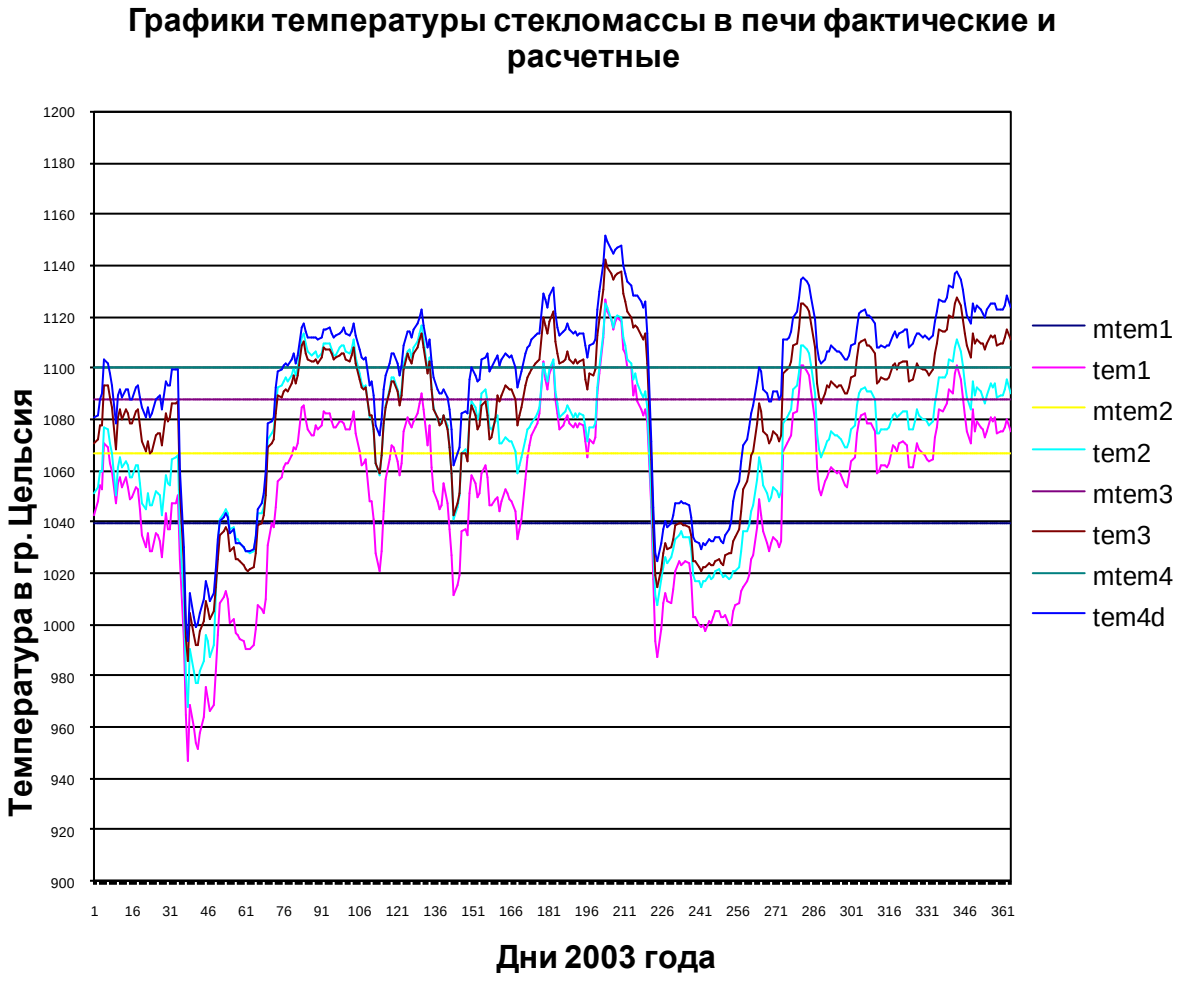


Рисунок 9.5 - Сравнение теплового режима работы ванной печи с результатами моделирования: *tem*1–*tem*4 – температуры $\Theta 1, \Theta 2, \Theta 3, \Theta 4$ при ручном ведении процесса; *mtem*1-*mtem*4 – соответственно при управлении

Для сопоставления алгоритма управления плотностью стекла с ручным ведением процесса результаты моделирования и показатели ручного управления сведены в табл. 9.3.

Таблица 9.3 - Сравнительные показатели управления печью по плотности вырабатываемого стекла с ручным ведением процесса варки

Показатели	Плотность стекла, г/см3		Оптические искажения, угл. град.		Дефекты 1 и 2-го класса, шт./ 10 м2		Удел. расход газа, м3/кг
Значение	Средн.	С.К.О.	Средн.	С.К.О.	Средн.	С.К.О.	Средн.
Управление плотностью	2, 4919	0,00244	64,7	5,57	0,93	0,71	0,203
Ручное управление	2,4933	0,00346	63,8	5,9	1,2	2,3	0,203

Как видно из табл. 9.3, алгоритм управления позволяет стабилизировать плотность вырабатываемого стекла, что приводит к уменьшению содержания дефектов 1 и 2-го классов в стекле. Среднее квадратичное отклонение плотности вырабатываемого стекла при одной и той же шихте, загружаемой в печь, можно уменьшить в 1,4 раза. При этом одновременно уменьшается среднее квадратичное отклонение содержания дефектов 1 и 2-го классов в вырабатываемом стекле в 3 раза.

Оптические искажения вырабатываемого стекла остаются на прежнем уровне, отвечающем требованиям, предъявляемым к стеклу высшей категории марки М0.

Удельный расход газа на стекловарение остается прежним, как и при ручном управлении, т.е. исследуемый алгоритм управления позволяет улучшить качество вырабатываемого стекла без увеличения затрат топлива на ведение технологического процесса варки.

Таким образом, имитационное моделирование показало возможность дальнейшего улучшения качества вырабатываемого стекла при переходе от ручного ведения процесса к управлению по алгоритму стабилизации плотности за счет направленной коррекции теплового режима работы ванной печи по температуре стекломассы.

Тестовые задания

1. Особенности методики использования вычислительного эксперимента и имитационного моделирования для оценки эффективности управления технологическими процессами и выработки предложений по совершенствованию алгоритмов управления.

2. Повышение достоверности результатов моделирования. Методы повышения точности разрабатываемых моделей.

Заключение

Процесс разработки информационных систем, реализующих новую информационную технологию, непрерывно совершенствуется. В центре внимания разработчиков оказываются все более сложные системы, что затрудняет, а в ряде случаев исключает, использование физических моделей, что повышает значимость математических моделей и моделирования при создании и совершенствовании систем. Актуальность математических моделей и компьютерного моделирования непрерывно возрастает из-за их гибкости, адекватности реальным процессам, наличием большого количества разработанных программных средств моделирования и относительно невысокой их стоимостью. Особенно эффективно применение моделирования на разных стадиях проектирования информационных систем: выполнении предпроектных исследований, эскизном проектировании и техническом проектировании создаваемых систем.

Современные ЭВМ, и прежде всего персональные компьютеры, позволяют существенно увеличить сложность разрабатываемых моделей и постановку вычислительного эксперимента на разработанных моделях. Позволяют строить комбинированные анлитико-имитационные модели, широко использовать модели на нечетких множествах и нейронных сетях, что позволяет адекватно описывать исследуемые процессы и явления. Поэтому в лекциях значительное внимание уделено изучению различных типов моделей и моделированию.

В курсе лекций сделана попытка системного изложения основ моделирования систем. Реализация возможностей машинного моделирования информационных систем излагается автором в методических указаниях к лабораторным занятиям по дисциплине «Моделирование». На лабораторных занятиях магистранты знакомятся со средствами разработки моделей и моделированием на современных ПЭВМ с использование стандартных пакетов программ, таких как *MATLAB, STATISTICA Neural Networks*, а также программ, разработанных на кафедре: *SMO*32, *Operator*2.4*Final.*

Курс «Моделирование» направлен сформировать представление магистрантов о моделировании как методе научного познания, ознакомить с использованием компьютера как средства познания и научно-исследовательской деятельности. Целями освоения дисциплины являются ознакомление магистрантов с методами построения моделей сложных систем, с возможностями средств моделирования, оценкой качества моделей, проведением экспериментов для оценки эффективности сложных систем, применение моделей в задачах управления.

Список литературы

1. Советов Б.Я., Яковлев С.Ф. Моделирование систем: Учеб. для вузов – 3-е изд., перераб. и доп. –М.: Высш. шк., 2001.-343с. *ISBN* 5-06-003860-2

2. Осовский С. Нейронные сети для обработки информации / Пер. с польского И.Д. Рудинского. –М.: Финансы и статистика, 2002. -344с.

3. Теория систем и системный анализ в управлении организациями: Справочник: Учеб. пособие / Под ред. В.Н. Волковой и А.А. Емельянова. - М.: Финансы и статистика, 2006. - 848 с:. *ISBN* 5-279-02933-5.

4. Информационные технологии в управлении качеством автомобильного стекла: учеб. пособие / Р.И. Макаров [и др.]; Владим. гос. ун-т.-Владимир: Изд-во Владим. гос. ун-та, 2010.-276с. *ISBN* 978-5-9984-0038-4.

5. Бильфельд Н.В., Иванова Е.В. Программирование в Matlab: учебно-метод пособие / Н.В. Бильфельд, Е.В. Иванова: Березниковский филиал ПНИПУ. – Пермь: Изд-во Перм. Нац. Исслед. Политехн. Ун-та, 2011.-235с. *ISBN* 978-5-398-00672-8

6. Малинин А.С., Мухин В.И. Исследование системы управления.

М.: ГУВШЭ, 2002. -380с.

7. Короткое Э.М. Исследование систем управления - М.: 000 Издательско-Консалтинговое Предприятие «ДеКА», 2004. - 336 с.

8. Курносов Ю.В., Конотопов П.Ю. АНАЛИТИКА: методология, технология и организация информационно-аналитической работы. — Москва: Издательство «Русаки», 2004 г. — 550 с. *ISBN* 5-93347-151-8

9. Алексеева М. Б., Балан С. Н. Основы теории систем и системного анализа: Учеб. пособие. - СПб.: СПбГИЭУ, 2002. - 88 с. *ISBN* 5-88996-229-X

10 . Методология проектирования информационных систем: учеб. пособие/Р.И. Макаров, Е.Р. Хорошева; Владимирский гос. университет, Владимир, 2008.-334с.

11. Основы теории систем и системного анализа/ конспект лекций/ Институт делового администрирования / частное учебное заведение. Кривой Рог, 1996. -77с.

12. Орловский С.А. Проблемы принятия решений при н6ечеткой исходной информации. М.: Наука, 1981.

13. Медведев, В.С. Нейронные сети. *MATLAB* 6/ В.С. Медведев, В.Г. Потемкин; под общ. ред. В.Г. Потемкина. – М.: ДИАЛОГ-МИФИ, 2002. – 496 с. – *ISBN* 5–86404–163–7.

14. Пушкин В.Н. Оперативное мышление в больших системах / В.Н. Пушкин. - М.: Энергия, 1965.

15. Круглов В.В., Борисов В.В. Искусственные нейронные сети. Теория и практика – 2-е изд. – М.: Горячая линия – Телеком, 2002. – 382 с.

16. Круглов В.В., Борисов В.В., Харитонов Е.В. Нейронные сети: конфигурации, обучение, применение. Смоленск: Изд-во Моск. энерг. ин-та, 1998.

17. Нейронные сети. STATISTICA Neural Networks: Пер. с англ. – М.: Горячая линия – Телеком. 2001. – 182 с.

18. Барский А.Б. Нейронные сети: распознавание, управление, принятие решений. – М.: Финансы и статистика, 2004. – 176с. *ISBN* 5-279-02757-X.

19. Макаров Р.И., Хорошева Е.Р., Лукашин С.А. Автоматизация производства листового стекла (флоат-способ)/Под ред. Р.И. Макарова; Владим. гос. ун-т. Вдалимир, 2000. -248с. *ISBN* 5-89368-206-8.

20. Хант Э., Марин Дж., Стоун Ф. Моделирование процесса формирования понятий на вычислительной машине (пер. с англ.). - М.: Мир, 1970. - 301 с.

21. Ермольев Ю. М. Методы стохастического программирования. 1976. -240 с.