ПРИОРИТЕТНЫЙ НАЦИОНАЛЬНЫЙ ПРОЕКТ «ОБРАЗОВАНИЕ»
РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

К.А. ПУПКОВ

МЕХАТРОНИКА

Учебное пособие

Москва

2008

Инновационная образовательная программа
Российского университета дружбы народов

«Создание комплекса инновационных образовательных программ и формирование инновационной образовательной среды, позволяющих эффективно реализовывать государственные интересы РФ через систему экспорта образовательных услуг»

Экспертное заключение –

кандидат технических наук, доцент *Ю.С. Васечкин*

Пупков К.А.

Мехатроника: Учеб. пособие. – М.: РУДН, 2008. – 132 с.

В учебном пособии представлены результаты разработки методического обеспечения процессов проектирования, моделирования, исследования и реализации интеллектуальных систем, основанных на комплексировании принципов робастного, нейро-нечеткого и адаптивного управления. Впервые рассмотрены задачи реализации интеллектуальных систем (ИС) в мехатронике как области науки и техники создания интеллектуальных машин и комплексов.

Приведены результаты методического обеспечения проектирования вычислительной среды ИС, а также методического обеспечения проектирования и реализации робастного, нейро-нечеткого и адаптивного управления.

Для студентов, обучающихся по направлению «Автоматизация и управление», и инженеров, обучающихся по специальности «Управление и информатика в технических системах».

Учебное пособие выполнено в рамках инновационной образовательной программы Российского университета дружбы народов, направление «Комплекс экспортоориентированных инновационных образовательных программ по приоритетным направлениям науки и технологий», и входит в состав учебно-методического комплекса, включающего описание курса, программу и электронный учебник.

Содержание

Введение

Данное учебное пособие посвящено проблемам разработки методического обеспечения процессов проектирования, моделирования, исследования и реализации интеллектуальных систем, основанных на комплексировании принципов робастного, нейро-нечеткого и адаптивного управления. Методическое обеспечение имеет огромное значение, так как по сути определяет технологическую последовательность реализации ИМС от концептуальной модели до опытного образца. Имеет значение техническая и программная среда, в которой реализуется ИМС. В современных условиях и на перспективу создания ИМС будет ориентировано на использование достижений мехатроники как области науки об интеллектуальных машинах. Поэтому первый раздел пособия посвящен проблемам интеллектуальных систем в мехатронике. Здесь показаны этапы развития мехатроники от макро- до наномехатроники на основе использования интеллектуальных систем.

Существенным компонентом ИМС является вычислительная среда, на которой реализуются алгоритмы обработки информации и управления, базы знаний и алгоритмы принятия решений. Здесь необходимо обеспечить высокую точность и надежность работы вычислительной среды. Поэтому во втором разделе показано методическое обеспечение процессов проектирования и реализации вычислительной части ИМС как на аппаратном, так и на уровне операционной системы и программного обеспечения.

Реализация ИМС высокой точности и надежности требует разработки методического обеспечения процессов проектирования робастного управления. Это связано с тем, что априорные сведения о динамике объекта управления и характеристиках воздействий окружающей среды не в полной мере отражают реальные свойства. Применение в этом случае классических методов управления может приводить к значительным погрешностям. Поэтому здесь рассмотрено именно робастное управление, которое позволяет при синтезе закона управления учитывать неполноту знаний динамики объекта и

характеристику окружающей среды. Этим проблемам посвящен третий раздел пособия.

Методические аспекты применения ИНС в интеллектуальных системах управления приведены в четвертом разделе. Именно применение ИНС позволяет повысить точность знания моделей на основе нейросетевой идентификации.

Пятый раздел охватывает методическое обеспечение процессов проектирования и реализации робастно-адаптивного управления в ИМС высокой точности и надежности. Использование этих материалов позволит повысить уровень проектирования интеллектуальных систем.

В целом учебное пособие охватывает классические и современные принципы, теорию и технологию построения мехатронных устройств и систем на основе мехатроники как науки о построении интеллектуальных машин. Наиболее развитыми мехатронными системами являются коалиции взаимосвязанных роботов, реализующих тот или иной технологический производственный процесс. Каждый робот в коалиции должен быть обучен тем или иным действиям и стремиться к самоорганизации на основе полученной информации о собственном состоянии и воздействиях окружающей среды.

Для этой цели используются искусственные нейронные сети. Существенным является системная организация роботов и формирование цели и принятие решения к действию в самой системе.

Значительное место в мехатронике помимо алгоритмического обеспечения имеет системное и прикладное программное обеспечение, реализованное в параллельной вычислительной среде. Исполнительные устройства должны быть реализованы на базе устройств микромеханики.

Учебное пособие будет полезно аспирантам и научным работникам, специализирующимся в области мехатроники и мехатронных систем и интеллектуализации процессов обработки информации в системах такого типа.

1. Интеллектуальные системы в мехатронике

Успехи в области мехатроники, микро (нано) процессорной техники и информационных технологий приводят к необходимости разработки и создания нового типа систем обработки информации и управления – интеллектуальных. Этот тип систем особенно важен в мехатронике, поскольку проектирование механических систем и их систем управления должно осуществляться как единое целое – интегрированные системы. При этом надо учитывать, что одни проблемы могут быть решены легко и просто в физических и механических образцах, другие в «информационных» образцах – в микроконтроллерах, объединенных информационным процессом, реализованным в том числе в программном обеспечении.

Современная мехатроника появилась не на пустом месте, ей предшествовал огромный опыт комплексирования механических устройств, радио- и оптоэлектроники, аналоговых и цифровых вычислительных средств в сложных системах различного назначения. Компоненты этих систем представляют собой отдельные блоки, которые соединялись с помощью различных преобразователей (ЦАП, АЦП) и других сопрягающих устройств. Однако, как и в нынешней мехатронике, все эти блоки в системе объединялись информационным процессом, включающим получение и обработку измерительной информации, ее распознавание, прогнозирование, выработку управления, исполнение управления действием и контроль результатов действия.

С современных позиций такие системы можно отнести к **макромехатронике**.

Усложнение решаемых задач, обеспечение высокой точности и надежности работы систем, улучшение качества выпускаемой продукции, безопасность систем потребовало новых подходов к их построению, обеспечивающих гибкую обработку информации в условиях ее неполноты и противоречивости,

принятия решения, синтеза и коррекции цели, сложного воздействия окружающей среды.

Такими являются интеллектуальные системы. Поскольку единственной реальной интеллектуальной системой является человек, то естественно в этом случае было бы обратиться к исследованиям в области физиологии и нейрофизиологии.

В 1935 году П.К. Анохин применил разработанный им системный подход к изучению и пониманию функций живого организма, опираясь на предложенную им же оригинальную теорию функциональной системы [1]. Используя тончайшие методы аналитического исследования нервной системы, П.К. Анохин находит место любому микрофизиологическому процессу в архитектуре целостной приспособительной реакции организма. В этой работе функциональная система впервые была определена как замкнутое физиологическое образование с наличием обратной информационной связи о результатах действия. Каждая функциональная система (**от уровня клетки до макроуровня**), обеспечивающая приспособительный эффект, имеет многочисленные каналы, по которым информация с периферии достигает соответствующих нервных центров. Полезный приспособительный эффект является определяющим в любой функциональной системе, поскольку способствует достижению цели, которая выступает в том числе как системообразующий фактор. Отличительная черта любого, даже самого маленького результата, способствующего достижению цели, – то , что он непременно получается на основе принципа самоорганизации и независимо от уровня в иерархии и сложности обладает одними и теми же узловыми механизмами, такими, как афферентный синтез цели, принятие решения к действию; эфферентная программа действия, акцептор действия, предсказывающий параметры результата и, наконец, сличение параметров полученного результата с параметрами, предсказанными и спрогнозированными акцептором действия.

Афферентный синтез является исходным для построения любой целенаправленной деятельности.

Четыре ведущих его компонента – исходная доминирующая мотивация, обстановочная и пусковая афферентация и, наконец, память позволяют представить механизм афферентного синтеза следующим образом: на основе исходной доминирующей мотивации, возникающей в результате той или иной внутренней потребности организма и памяти, организм, стимулируемый различными пусковыми сигналами, активно оценивает раздражители внешней среды, вырабатывает цель и принимает соответствующее решение к действию. В соответствии с целью все компоненты системы взаимодействуют так, чтобы выполнялась эфферентная программа действия. Аппарат акцептора результатов действия, формирующийся на основе определенной потребности, памяти, обстановки и специальных сигналов, заключает в себе все свойства будущего результата и поэтому служит для сопоставления предсказанного и реально полученного результатов. Обратная афферентация о параметрах результата есть не что иное, как обратная связь. Поэтому функциональная система Анохина является фундаментальным результатом, наиболее полно и просто объясняющим характер происходящих процессов. Заметим, что Норберт Винер в 1960 году признал, что рассмотренный в плоскости физиологической кибернетики этот результат намного опередил рождение кибернетического направления в целом.

Именно основываясь на фундаментальных результатах П.К. Анохина [1] и научных исследованиях К.А. Пупкова [2] в 1989 году была предложена структура и дано определение интеллектуальной системы (рис. 1.1).

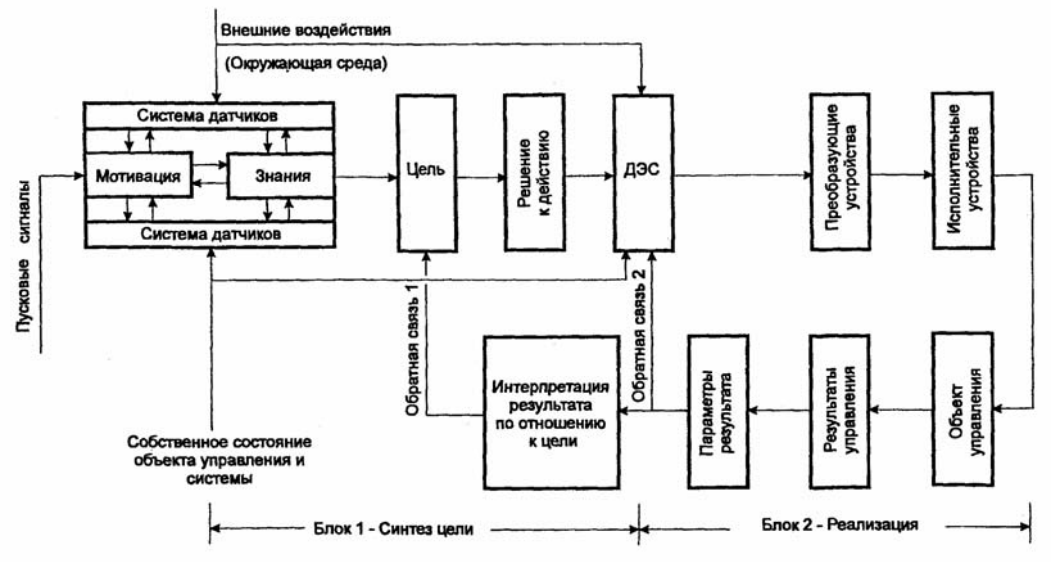

Рис. 1.1. Структурная схема интеллектуальной системы

Под интеллектуальной системой будем понимать объединенную информационным процессом совокупность технических средств и программного обеспечения, работающую во взаимосвязи с человеком (коллективом людей) или автономно, способную на основе сведений и знаний при наличии мотивации синтезировать цель, принимать решение к действию и находить рациональные способы достижения цели.

Такая структура системы может быть использована при построении интеллектуальных систем, применяемых во всех сферах человеческой деятельности. На рис. 1.1. видно, что интеллектуальная система состоит из двух блоков, в первом из которых синтезируется цель, во втором – процесс достижения цели.

В первом блоке в качестве исходного компонента выступает мотивация (потребность в чем-либо), которая сочетается с информацией, получаемой с помощью системы датчиков, о состоянии окружающей среды и собственном состоянии системы. При синтезе цели знания используются активно, т.е. на основе знаний, хранящихся в памяти системы, окружающая среда и собственно система стимулируются пусковыми сигналами, происходит активная оценка раздражителей внешней среды. Далее информация поступает в динамическую экспертную систему (ДЭС), в которой моделируются и реализуются алгоритмы функционирования эфферентных возбуждений (управления), акцептора

действия, содержащего в себе все свойства будущего результата и служащего для сопоставления предсказанного и реально полученного результатов.

В ДЭС решается задача комплексирования и самоорганизации робастных, нейро-нечетких и адаптивных алгоритмов управления по мере накопления и повышения достоверности информации о воздействии окружающей среды и изменчивости собственного состояния системы. Существенным для ДЭС является наличие базы знаний. Управление, выработанное в ДЭС, реализуется с помощью исполнительных устройств, изменяющих состояние объекта управления, а информация о параметрах результата управления поступает по обратной связи *2* в ДЭС, где сопоставляются параметры предсказанного и реального результатов. Если при выработанном управлении цель достигается, т.е. разность между параметрами результатов удовлетворяет требованиям, то управление подкрепляется, если же нет, – корректируется. Если окажется, что синтезированная цель не является достижимой, то параметры результата интерпретируются по отношению к цели, и производится коррекция цели (обратная связь *1*).

Современная мехатроника – наука об интеллектуальных машинах (The Science of Intelligent Machines).

В своем докладе, посвященном перспективным направлениям в интеллектуальных мехатронных системах («Future Trends in Intelligent Mechatronics Systems» , The 7^{th} Mechatronics Forum, Sept. 2000. Atlanta, USA) основатель мехатроники С. Яскава отмечает, что концепция «Мехатроника» – удачное слияние механизмов и микроэлектроники была основана в конце 60-х годов. Эта концепция увеличила производительность в таких отраслях, как автомобилестроение, компьютеры, средства связи и дала возможность глобальному развитию. Это привело к эффективности массового производства. Оно было сконцентрировано на получение материальной выгоды. Теперь мы должны взять в свое распоряжение концепцию окружающей среды – полный жизненный цикл и «стряхнуть пыль», – наше дело. Поскольку мы движемся из эры «закрытого сбалансированного общества» к «открытому

несбалансированному обществу», управление и глобальная стандартизация необходимы.

Используя Интернет и слияние мехатроники с информационными сетями, можно покончить с географическими трудностями глобальной кооперации. Речь идет о прямом использовании парадигмы интеллектуальных систем в мехатронике. Причем это использование должно быть не только на глобальном уровне построения интеллектуальных мехатронных систем, но и на уровне построения интеллектуальных машин, аппаратов, роботов, приборов, а также на уровне отдельных компонентов, включая построение сенсоров, высокоточных измерительных устройств линейного движения, сканирующих оптических микроскопов, самокалибрующейся техники и т.п., поскольку в интеллектуальных системах самый маленький результат, способствующий достижению цели, непременно получается на основе принципа самоорганизации и независимо от уровня и сложности обладает одними и теми же узловыми механизмами. Заметим, что классический мехатронный модуль может быть «кирпичиком» для построения интеллектуальных машин.

Интеллектуализация мехатроники, внедрение интеллектуальных систем в мехатронику, мехатронные системы и робототехнические системы [2] поставило ряд новых теоретических и практических проблем. Так, с точки зрения интеллектуализации обучение и адаптация в системах приобретает все большее значение. Здесь процессы интеллектуализации можно описывать в терминах нейронных сетей, фаззилогики, эволюционных алгоритмов. Важнейшим направлением будет нахождение алгоритмов «гибкой» логики при принятии решения и выработки управления. В этом случае эффективным окажется комплексирование робастных, нейро-нечетких и адаптивных алгоритмов в базе знаний интеллектуальных систем.

Здесь важно, чтобы математическое и программное обеспечение этих алгоритмов, реализованных в системе, позволяло бы эксплуатировать ее в реальном времени. Отсюда – распределенные вычислительные сети, распараллеливание алгоритмов, параллельные языки программирования.

Поскольку гибкая обработка информации и управления выходит за пределы традиционных подходов, то можно указать некоторые области исследования новых функций, а именно: распознавание и понимание разного рода информации типа рисунков, звуков речи и символьной информации, присущие естественным языкам; вывод и решение задач с помощью баз знаний, которые допускают прямую обработку информации и обладают способностью к обучению и самоорганизации; интерфейс и моделирование взаимодействия человека с реальным миром; управление и автоматическое управление в интеллектуальных системах, функционирующих в реальном времени.

Здесь можно указать два направления развития интеллектуальных систем: автоматические интеллектуальные системы, адаптированные к реальной окружающей среде, и диалоговые системы, в которых интегрируются функции автоматических систем и человека в их взаимодействии.

В первом направлении системы должны быть способны автономно понимать и контролировать среду путем активного и адаптивного взаимодействия с реальным миром, а также взять на себя часть деятельности человека в этом мире. Таким системам необходимо справляться с неполнотой, неопределенностью и изменчивостью информации, характерными для реального мира. К новым функциям таких систем можно отнести понимание воздействий окружающей среды, моделирование реального мира, планирование последовательности действий, оптимальное управление с целью достижения желаемого результата, элементы адаптации и самоорганизации.

Второе направление означает «объединение» системы с человеком. Это должны быть гибкие системы, поддерживающие и повышающие интеллектуальную деятельность людей в таких областях, как решение задач и получение информации за счет расширения каналов связи между людьми и системами. Чтобы помочь людям в решении задач и получении новой информации, потребуется воспринимать и интегрировать различную информацию. Здесь новые функции в системе: вопрос и ответ, высказанные на естественном языке; понимание намерений на базе различной информации,

поступающей от людей; реализация интеллектуальной поддержки для нахождения и представления полезной информации в огромном количестве данных, хранящихся в базах данных; интеллектуальное моделирование для создания новых информационных данных и прогнозирование изменений в реальном мире; методы интеграции для обеспечения взаимодействия человека и системы; вычислительная модель реального мира и т.д.

Огромное значение в мехатронике приобретает технология и бизнес. Ясно, что системы мехатроники и их производство являются комбинацией компьютеров, механизмов, исполнительных устройств, чувствительных элементов и функций управления, которые организуются при проектировании так, чтобы достигнуть высокого качества и получить выгоду от уровня интеграции систем. Модель успешного бизнеса для мехатронных решений нуждается в том, чтобы компании и партнеры взаимодействовали в определении функций и проектировании мехатронных систем, технологии их производства, материалах и самом производстве. Здесь надо рассчитывать на использование CALS-технологий. Технология и быстрые темпы ее развития являются наиболее важным фактором, влияющим на «глобальность» и на практические аспекты бизнеса и инженерии. Современная экономика и общество в целом управляются технологией (основываются на технологии). Начало XXI века может стать для инженеров настоящей эпохой Ренессанса. Естественно, это должно учитываться при подготовке учебных планов в университетах и при повышении квалификации.

Исследования и разработки по данному проекту выполнялись по четырем этапам. На первом этапе была проведена разработка и исследование принципов построения и типовых структур интеллектуальных систем, основанных на комплексировании принципов робастного, нейро-нечеткого и адаптивного управления. Полученные на данном этапе результаты являются теоретическими и носят фундаментальный характер.

Синтезированы законы управления в условиях неточного знания моделей окружающей среды и объекта управления. Определены принципы и пути

комплексирования робастного, нейро-нечеткого и адаптивного управления в интеллектуальных системах высокой точности и надежности. Полученные законы управления математически строго обоснованы и достаточно точно отражают физику протекающих в системе процессов. Полученные результаты в значительной степени являются новыми.

Второй этап проекта был посвящен разработке алгоритмического обеспечения процессов проектирования и исследования интеллектуальных систем, основанных на комплексировании принципов робастного, нейро-нечеткого и адаптивного управления. На этом этапе была проведена алгоритмизация разработанных в первой части проекта законов управления с целью обеспечения их эффективной реализации в вычислительной части ИС и соответственно разработки программного обеспечения. Разработаны требования к вычислительной части ИС и составу программного обеспечения. Данные результаты являются новыми.

На третьем этапе разработки проекта обеспечивалось создание программного обеспечения процессов проектирования, моделирования, исследования и реализации ИС, основанных на комплексировании принципов робастного, нейро-нечеткого и адаптивного управления. Использование программного обеспечения позволило провести математическое моделирование процессов, происходящих в ИС высокой точности и надежности. Результаты обработки и оценки экспериментальных данных показали высокую степень соответствия этих результатов теоретически полученным законам управления. Погрешность составляет порядка 7 – 8%.

На четвертом этапе разработано методическое обеспечение, которое позволяет эффективно использовать фундаментальные результаты в практике проектирования, моделирования и исследования интеллектуальных систем высокой точности и надежности.

2. Разработка методического обеспечения процессов проектирования и реализации вычислительных сред ИС

Вычислительная среда – объединенная единым информационным процессом совокупность программного и аппаратного обеспечения, задачей которой является выполнение программного кода.

Для интеллектуальных систем данное определение дополняется условиями, главным образом, скорости выполнения кода и возможной его динамической модификации (часть этой задачи возложена на главный интеллектуальный модуль, решающий задачу: как и что модифицировать, задачу же быстрого преобразования кода языка в машинный и его последующее выполнение возлагается на вычислительную среду).

Поскольку, как было описано выше, сама вычислительная среда состоит из двух подсистем, связанных между собой (важно предусмотреть, чтобы связь была как можно менее жесткой, чтобы не получить зависимость от конкретной аппаратной реализации, например, путем ввода абстрактного уровня аппаратуры), то проектирование и реализация данных подсистем сводится также на два основных этапа – программный и аппаратный.

Для получения необходимых динамических характеристик вычислительной системы необходимо выбрать базисную технологию обработки информации. В качестве такого базиса для вычислительной среды интеллектуальной системы выберем параллельные технологии, трудные в исполнении, но способные придать необходимые характеристики системе. В соответствии с аппаратной частью программная, в свою очередь, также должна опираться на параллельные технологии, дабы использовать всю мощь предоставленных ресурсов.

Оптимизация алгоритмов программной части по использованию максимального процента ресурсов является основной задачей при проектировании и реализации вычислительной среды.

2.1. Аппаратная часть вычислительной среды.

Параллельные системы как основа вычислительной среды

Параллельная обработка данных, воплощая идею одновременного выполнения нескольких действий, имеет две разновидности: конвейерность и собственно параллельность. Оба вида параллельной обработки интуитивно понятны, поэтому сделаем лишь небольшие пояснения.

Параллельная обработка. Если некое устройство выполняет одну операцию за единицу времени, то тысячу операций оно выполнит за тысячу единиц. Если предположить, что есть пять таких же независимых устройств, способных работать одновременно, то ту же тысячу операций система из пяти устройств может выполнить уже не за тысячу, а за двести единиц времени. Аналогично система из N устройств ту же работу выполнит за $1000/N$ единиц времени.

Конвейерная обработка. Идея конвейерной обработки заключается в выделении отдельных этапов выполнения общей операции, причем каждый этап, выполнив свою работу, передавал бы результат следующему, одновременно принимая новую порцию входных данных. Получаем очевидный выигрыш в скорости обработки за счет совмещения прежде разнесенных во времени операций.

Под *многопроцессорной вычислительной системой (МВС)* будем понимать комплекс вычислительных средств, связанных физически и программно, в котором одновременно (в один и тот же момент времени) может выполняться несколько арифметических или логических операций по преобразованию данных.

Известно, что повышение производительности вычислительной системы при последовательном выполнении операций зависит от возрастания тактовой частоты ее элементов. Между тактовой частотой работы машины и ее размерами существует определенная зависимость:

$$V_{пр} = a(c/\gamma)^3 (1/\nu^3), \qquad (2.1)$$

где $V_{пр}$ – предельный физический объем вычислительной системы, a – некоторый постоянный коэффициент, учитывающий геометрию машины, c – скорость света, γ – константа, v – тактовая частота машины.

Из (2.1) виден теоретический предел повышения производительности при последовательном выполнении операций.

Рассмотрим концепцию параллельного выполнения операций, при которой производительность определяется как количеством элементов, так и частотой их работы. Увеличение производительности в этом случае достигается за счет увеличения количества параллельно работающих элементов при одновременном уменьшении тактовой частоты.

$$P = bnv, \tag{2.2}$$

где P – производительность машины, n – число элементов, v – тактовая частота, b – некоторый постоянный коэффициент.

Предельно допустимое число элементов ($n_{пр}$) определяется соотношением

$$n_{пр} = b(c/\gamma)^3(1/v^3). \tag{2.3}$$

Из соотношений (2.2) и (2.3) видно, что с уменьшением тактовой частоты допускается сколь угодно высокая производительность вычислений.

Распределенная обработка информации базируется на модели коллектива вычислителей, в основу которой положены следующие аксиомы:

- **аксиома параллельности задачи**: всякая сложная задача может быть представлена в виде связанных между собой простых задач, которые при необходимости обмениваются информацией;
- **аксиома параллельности алгоритмов**: для задачи любой сложности может быть предложен параллельный алгоритм, допускающий ее эффективное решение;
- **аксиома параллельности модели коллектива вычислителей**: модель может обеспечить при параллельной работе сколь угодно высокую производительность;

- **аксиома реконфигурации логической структуры**: структура изменяется как при переходе от задачи к задаче, так и в процессе решения задачи.

2.2. Информационные модели

Для решения задачи на параллельном компьютере необходимо распределить вычисления между процессорами системы так, чтобы каждый процессор был занят решением части задачи. Необходимо обеспечить минимальный объем данных, пересылаемых между процессорами, так как коммуникации – более медленные операции, чем вычисления.

Среда параллельного программирования должна обеспечивать адекватное управление распределением и коммутациями данных.

Основные модели параллельного программирования:

- процесс/канал (Process/Chanel),
- обмен сообщениями (Message Passing),
- параллелизм данных (Data Parallel),
- общая память (Common Memory).

Типы параллелизма:

- естественный,
- параллелизм множества объектов,
- параллелизм независимых ветвей,
- параллелизм смежных операций.

Поскольку в параллельных системах основным является использование самого принципа параллельности, поэтому в вычислительной системе должны присутствовать все типы параллелизма для обеспечения наиболее полного распараллеливания процессов.

В любой параллельной системе главные вычислительные модули должны некоторым образом обмениваться информацией друг с другом. Существует две

основные информационные модели – мультипроцессорная и мультикомпьютерная.

В мультипроцессорных системах, как правило, выделяют три вида, в зависимости от механизма реализации памяти совместного использования:

- UMA (Uniform Memory Access) – с однородным доступом к памяти;
- NUMA (Non Uniform Memory Access) – с неоднородным доступом к памяти;
- COMA (Cache Only Memory Access) – с доступом только к кэш-памяти.

У каждой из данных технологий информационных моделей присутствуют свои плюсы и минусы, например, сложность программирования, сложность масштабирования системы и т.д. Поэтому имеет смысл использовать так называемые гибридные системы, основным требованием которых является возможность масштабирования. Здесь существуют три основных подхода. Первый – совместно используемая память на уровне аппаратного обеспечения: отслеживание за адресным пространством возложено на операционную систему (ОС). Второй – распределенная совместно используемая память (Distributed Shared Memory): отслеживание также возложено на ОС, но память распределена и представляет собой единое виртуальное пространство. Третий подход – реализация общей памяти на уровне программного обеспечения. Абстракция разделенной памяти создается языком программирования (ЯП), и эта абстракция используется компилятором. Никакого специального аппаратного или программного обеспечения не требуется.

В качестве информационной модели вычислительной среды интеллектуальной системы наиболее подходит смешанный тип как с аппаратным контролем разделения памяти, так и с программным.

Как уже было сказано выше, одна из основных задач – обеспечение максимально простой и ликвидной процедуры масштабирования системы. Здесь можно разделить систему на основные ядра и дополнительные. Основные ядра представляют собой мультипроцессор и объединяются в гиперкуб.

Дополнительные ядра могут быть подключены к основным и представляют собой структуру мультикомпьютера.

Основные ядра расширения являются более глобальным решением повышения производительности, но также более сложны в производстве и, соответственно, более дорогие. Дополнительные же ядра вследствие своей физической структуры повышают производительность не так сильно, как основные, но более просты в производстве и, соответственно, более дешевы.

Основные ядра используют аппаратный уровень в процессе управления памятью, дополнительные – программный.

2.3. Программная часть

Операционные системы реального времени

Любая ОС обязана обеспечить полный цикл жизни программного обеспечения: создание текста программы, ее компиляция, компоновка, отладка, исполнение, сопровождение. Задачи реального времени предъявляют свои требования к вычислительно-управляющим системам, в том числе к ОС, в которых реализовано программное обеспечение реального времени. Эти требования изложены в стандарте POSIX 1003.4 рабочего комитета IEEE. Стандарт определяет ОС как систему реального времени, если она обеспечивает требуемый уровень сервиса за вполне определенное, ограниченное время, то есть ОС реального времени должна быть предсказуемой [4 – 10].

Становится очевидным то, что задачи реального времени необходимо реализовывать в рамках специфической системной программной среды. В соответствии с [8] системы реального времени можно разделить на 4 класса:

1-й класс: программирование на уровне микропроцессоров. При этом программы для программируемых микропроцессоров, встраиваемых в различные устройства, очень небольшие и обычно написаны на языке низкого уровня типа ассемблера или PLM. Внутрисхемные эмуляторы пригодны для

отладки, но высокоуровневые средства разработки и отладки программ не применимы. Операционная среда обычно недоступна;

2-й класс: минимальное ядро системы реального времени. На более высоком уровне находятся системы реального времени, обеспечивающие минимальную среду исполнения. Предусмотрены лишь основные функции, а управление памятью и диспетчер часто недоступны. Ядро представляет собой набор программ, выполняющих типичные, необходимые для встроенных систем низкого уровня функции, такие, как операции с плавающей запятой и минимальный сервис ввода/вывода. Прикладная программа разрабатывается в инструментальной среде, а выполняется, как правило, на встроенных системах;

3-й класс: ядро системы реального времени и инструментальная среда. Этот класс систем обладает многими чертами ОС с полным сервисом. Разработка ведется в инструментальной среде, а исполнение – на целевых системах. Этот тип систем обеспечивает гораздо более высокий уровень сервиса для разработчика прикладной программы. Сюда включены такие средства, как дистанционный символьный отладчик, протокол ошибок и другие средства CASE. Часто доступно параллельное выполнение программ;

4-й класс: ОС с полным сервисом. Такие ОС могут быть применены для любых приложений реального времени. Разработка и исполнение прикладных программ ведутся в рамках одной и той же системы.

Системы 2-го и 3-го классов принято называть системами «жесткого» реального времени, а 4-го класса – «мягкого». Очевидно, это можно объяснить тем, что в первом случае к системе предъявляются более жесткие требования по времени реакции и необходимому объему памяти, чем во втором. Как мы видим, среда разработки и среда исполнения в системах реального времени могут быть разделены, а требования, предъявляемые к ним, весьма различны. Рассмотрим их более подробно.

Среда исполнения. Требования, предъявляемые к среде исполнения систем реального времени, следующие:

- небольшая память системы – для возможности ее встраивания;

- система должна быть полностью резидентна в памяти, чтобы избежать замещения страниц памяти или подкачки;
- система должна быть многозадачной – для обеспечения максимально эффективного использования всех ресурсов системы;
- ядро с приоритетом на обслуживание прерывания. Приоритет на прерывание означает, что готовый к запуску процесс, обладающий некоторым приоритетом, обязательно имеет преимущество по отношению к процессу с более низким приоритетом, быстро заменяет последний и поступает на выполнение. Ядро заканчивает любую сервисную работу, как только поступает задача с высшим приоритетом. Это гарантирует предсказуемость системы;
- диспетчер с приоритетом – дает возможность разработчику прикладной программы присвоить каждому загрузочному модулю приоритет, неподвластный системе. Присвоение приоритетов используется для определения очередности запуска программ, готовых к исполнению. Альтернативным такому типу диспетчеризации является диспетчеризация типа «карусель», при которой каждой готовой к продолжению программе дается равный шанс запуска. При использовании этого метода нет контроля за тем, какая программа и когда будет выполняться. В среде реального времени это недопустимо. Диспетчеризация, в основу которой положен принцип присвоения приоритета, и наличие ядра с приоритетом на прерывание позволяют разработчику прикладной программы полностью контролировать систему. Если наступает событие с высшим приоритетом, система прекращает обработку задачи с низшим приоритетом и отвечает на вновь поступивший запрос.

Сочетание описанных выше свойств создает мощную и эффективную среду исполнения в реальном времени.

Кроме свойств среды исполнения, необходимо рассмотреть также сервис, предоставляемый ядром ОС реального времени. Основой любой среды

исполнения в реальном времени является ядро или диспетчер. Ядро управляет аппаратными средствами целевого компьютера: центральным процессором, памятью и устройствами ввода/вывода; контролирует работу всех других систем и программных средств прикладного характера. В системе реального времени диспетчер занимает место между аппаратными средствами целевого компьютера и прикладным программным обеспечением. Он обеспечивает специальный сервис, необходимый для работы приложений реального времени. Предоставляемый ядром сервис дает прикладным программам доступ к таким ресурсам системы, как, например, память или устройства ввода/вывода.

В связи с особенностями сегодняшнего рынка программных технологий существует три варианта определения программной платформы для разрабатываемой системы:

- Разработать ОСРВ самостоятельно.
- Приспособить под свои нужды свободно распространяемую ОС.
- Использовать готовую коммерческую ОС.

2.4. Языки параллельного программирования

Под параллельным программированием будем понимать создание объектов – параллельных программ, каждая из которых есть организованная совокупность модулей с возможностями одновременного выполнения и взаимодействия.

Язык программирования – это формальный язык связи человека с компьютером, предназначенный для описания данных и алгоритмов их обработки.

Основные требования к ЯП:

- Ясность, простота и согласованность понятий языка с простыми и регулярными правилами их конструирования;
- Ясность структуры программы, написанной на ЯП, и возможность ее модификации;

- Естественность в приложениях (наличие подходящих для реализации требуемых задач структур данных, операций, управляющих структур и естественный синтаксис языка);
- Легкость расширения за счет моделирования простыми средствами сложных структур, имеющихся в конкретных задачах;
- Мощное внешнее обеспечение (средства тестирования, отладки, редактирования, хранения);
- Эффективность создания, трансляции, тестирования, выполнения и использования программ.

Помимо основных требований, к ЯПП предъявляются дополнительные требования, а именно ЯПП должен:

- Иметь средства максимального выражения в программе присущего данной задаче естественного параллелизма;
- Быть независимым от структуры конкретного компьютера, в частности от числа процессоров, доступных для программ, от времени выполнения отдельных ветвей программы и т.д.;
- Обладать простотой диспетчеризации параллельных программ, записанных на нем;
- Обеспечить простоту записи (преобразования) программ на ЯПП по заданным последовательным алгоритмам.

При проектировании компьютеров с параллельной архитектурой необходимо учитывать особенности ПО. Существует несколько подходов к созданию ПО:

1. Создание специальных библиотек для параллельной обработки задач, хорошо поддающихся распараллеливанию, например, обработка матриц, решение СЛАУ и т.д. Недостаток – только часть кода будет исполняться параллельно.
2. Создание библиотек, содержащих примитивы коммуникации и управления. Здесь необходимо вручную задавать сценарий исполнения программы и управлять им, применяя дополнительные примитивы.

3. Дополнение последовательного языка программирования специальными конструкциями, порождающими параллельные процессы, позволяющие одновременно обрабатывать тела циклов, компоненты векторов, строки или столбцы матрицы и т.д. На сегодня – это самый распространенный подход.
4. Разработка специального нового языка параллельного программирования (ЯПП). Среди данного вида известны императивные (их команды изменяют переменные состояния), функциональные, логические и объектно-ориентированные.

Исходя из средств задания взаимодействий между параллельными процессами можно выделить две группы языков:

- Взаимодействие фрагментов (процессов) с помощью доступа к общим переменным;
- Взаимодействие посредством передачи межпроцессорных сообщений.

Также ЯПП разделяются по числу участвующих во взаимодействии процессов. Здесь выделяют языки, обеспечивающие задание:

- Индивидуальных (парных) взаимодействий между фрагментами: механизм подчиненных процессов ОС/360, язык взаимодействующих процессов Хоара, OCCAM, язык граф-схем;
- Между фрагментами (процессами): OBC-Cobol (Fortran), параллельный Cobol, языковые средства компьютера ILLIAC-IV.

Важной задачей, решаемой ЯПП, является задача распределения инструкций по процессорам (машинам). Для ее решения разрабатываются методы распознавания и выделения независимых фрагментов программ. Основные вопросы, возникающие в данном процессе, следующие:

- Кто и когда осуществляет разбиение исходной программы и распределение по процессорам;
- Какие критерии используются при разбиении и распределении;
- Какое время сохраняется разбиение;
- Каковы максимально неделимые участки.

В настоящее время существует достаточно много ЯПП, однако трудно выбрать наиболее приемлемый для использования. Ниже приведены некоторые модельные языки, идеи которых используются в большинстве современных ЯПП.

Р- и К-языки

Программа, записанная на Р-языке, представляет собой матрицу, элементами которой являются операторы. Каждый столбец матрицы включает независимые операторы, которые можно выполнять параллельно. Процесс выполнение программы – последовательное прохождение всех столбцов. Здесь остается явное указание порядка выполнения операций, но вводится двухкоординатная система записи, где одна координата указывает последовательность выполнения операций во времени, другая – распределение операций между ветвями вычислений. Не позволяет сохранить природную асинхронность задач и требует учитывать структуру конкретной вычислительной системы.

Суть работы К-языка заключается в задании множества элементарных операторов и множества порождающих правил построения алгоритмов, образующих в совокупности некоторое исчисление. Запись вывода в этом исчислении и является К-программой. Здесь имеются три типа основных порождающих правил: суперпозиция, дизъюнкция и рекурренция. Порядок выполнения операторов безразличен, что и порождает параллелизм в их выполнении. Правило дизъюнкции задает ветвление, а правило рекурренции – цикличность.

Язык OCCAM

Создан для многотранспьютерных систем. Теоретической основой данного языка является язык CSP (Communicating Sequential Processes), разработанный Т. Хоаром. Основным понятием языка является процесс (примитивный, составной, именованный), напоминающий процедуры и подпрограммы обычных ЯП.

Процессы взаимодействуют с помощью каналов и могут выполняться как последовательно, так и параллельно. Так как в системе команд транспьютера имеются команды, предназначенные для компиляции примитивных процессов считывания из канала и записи в канал, то эти процессы хорошо согласуются с транспьютерной архитектурой. Язык OCCAM постоянно развивается в связи с совершенствованием транспьютеров и приближается к языку высокого уровня с естественным расширением класса решаемых задач.

Язык Erlang

Язык программирования для параллельных и распределенных систем. Разработан в Ericsson Computer Science Laboratories.

Многие примитивы данного языка специально разработаны для решения проблем, наиболее часто встречаемых при программировании параллельных RTS (Real Time System – систем реального времени). Модель параллелизма основана на понятии процесса и задается в явном виде. Процесс обмена информацией между процессами – передача сообщений. Язык имеет встроенные механизмы распределенного программирования, что позволяет достаточно просто разрабатывать программы, которые могут быть запущены как на одном, так и на совокупности компьютеров.

Язык программирования mpC

ЯП mpC – расширение языка C, разработанное для программирования параллельных вычислений в компьютерных сетях. Основной целью параллельных вычислений является ускорение решения задачи. Поэтому главное внимание в данном языке уделяется средствам, позволяющим максимально облегчить разработку эффективных программ для решения задач в компьютерных сетях.

2.5. Задача планирования в мультисистемах

Самой главной задачей в работе мультисистем, является распределение задач по процессорам, или задачей планирования ресурсов [11], [12].

Все задачи (процессы) в системе конкурируют между собой из-за процессорного времени. Помимо процессов пользователя имеются также и системные процессы, которым также необходимо процессорное время.

Существует несколько основных подходов к решению задачи диспетчирования – статический и динамический. Статический нацелен на конкретную МВС и сильно зависит от числа задач, динамический же использует принцип загрузки машин (процессоров) по мере их освобождения.

На основании данных подходов строятся алгоритмы планирования ресурсов. Из них выделяются следующие алгоритмы.

Алгоритм планирования по наивысшему приоритету (highest priority first – HPF). Здесь процессор предоставляется процессу, имеющему наивысший приоритет. Здесь приоритет составляется исходя из множества правил вычисления приоритета, однако все же есть вероятность большого простоя процессов с низким приоритетом, что недопустимо в системах реального времени.

Алгоритм простого круговорота (round robin – RR). Информация о приоритетах не используется. Каждому процессу выделяется квант времени на его выполнение. Регулируя величиной кванта, можно достигнуть приемлемой производительности. На базе данного алгоритма существует несколько модификаций, позволяющих еще сильнее улучшить производительность путем ввода различных критериев обработки процессов.

Алгоритм очереди с обратной связью (feedback – FB). Использует *n* очередей, каждая из которых обслуживается в порядке поступления. Процессы, требующее незначительного времени обслуживания, обеспечиваются в этом случае лучше, чем при круговороте. Однако большое число очередей может увеличить накладные расходы времени.

Алгоритм многоуровневого планирования. В основе данного алгоритма лежит следующий принцип – операции, которые встречаются часто, должны требовать меньше времени, чем те, которые встречаются редко. В связи с чем операции разбиваются на уровни.

Также, наряду с вышеописанными «техническими» алгоритмами, существует вариант использования генетического алгоритма – алгоритма случайного поиска, основанного на принципах эволюции и генетики.

В отличие от статических и детерминированных алгоритмов генетический алгоритм очень гибко приспосабливается к любому набору входных данных независимо от характеристик набора и выбирает близкое к оптимальному решению в большем числе случаев. Однако этот алгоритм эффективен по времени только в тех случаях, когда время исполнения планируемых задач намного меньше времени самого алгоритма. Также данный алгоритм достаточно труден в реализации, однако существуют модификации данного алгоритма, которые эффективно устраняют какой-либо из недостатков.

3. Разработка методического обеспечения процессов проектирования, моделирования, исследования и реализации робастного управления в интеллектуальных системах управления

Методическое обеспечение реализации, проектирования и исследования ИСУ использует тот факт, что ИСУ как система, создаваемая для достижения цели, должна обладать теми же механизмами, как и такая же функциональная схема приспособительных реакций. Достижения современной науки в таких областях, как теория управления, микроэлектроника, теория информации, нейрофизиология, информационные технологии являются технической базой, обеспечивающей реализацию ИСУ.

Основной акцент в методике проектирования процессов реализации робастных методов управления в ИСУ делается на принятии оперативных решений в реальном масштабе времени на этапах, когда информации об окружающей среде и действующих возмущениях не достаточно для оценки ключевых параметров управления.

Одним из основных вопросов в разработке методики ИСУ является способ решения задачи комплексирования алгоритмов управления. Одним из способов решения данного вопроса может быть синтез расчетно-логической ДЭС, где база знаний сочетает описание в виде строгих математических формул с информацией экспертов, а также соответственно – математические методы поиска решения с нестрогими эвристическими методами, причем вес того или другого компонента определяется возможностью адекватного описания предметной области и способом поиска решения. Составной частью решения задачи комплексирования ИСУ будет определение области оптимального применения конкретного, например робастного на основе H_∞-теории оптимизации, метода управления, на множестве заранее известных и заложенных в базу данных алгоритмов ИСУ.

3.1. Методика синтеза структурной схемы ИСУ

Исследования в области синтеза интеллектуальных систем привели к необходимости разработки методики построения соответствующей структурной схемы. Методика построения базируется на применении новых информационных технологий (включая динамические экспертные системы, базы знаний, параллельные алгоритмы и языки программирования, робастные и адаптивные методы управления), а также новых технических средств, позволяющих создавать интеллектуальные системы [13].

Расчет ИСУ требует специфического подхода, который должен обеспечить практическую работоспособность ИСУ, в том числе и в начальный период ее работы, проходящий в условиях сильной неопределенности.

Функциональная система Анохина–Пупкова (см. рис. 1.1) является фундаментальным результатом, наиболее полно и просто объясняющим характер происходящих процессов, понимание которых позволяет приступить к синтезу ИСУ. Для выбранной цели все компоненты системы взаимодействуют так, чтобы выполнялась эфферентная (обратная афферентной) программа действия, а это аналог обратной связи, положительные стороны которой

известны из основ теории автоматического управления. Данная структура инвариантна к объекту управления и носит универсальный характер.

Оценка полноты базы знаний

Структура и организация базы знаний тесно связана с языком и формой представления модели процесса управления. Для обеспечения интерактивного режима работы с операторами аналогичным образом должен обеспечиваться ввод в базу поступающих от них команд, запросов и сообщений. В частности, для этого база управляющей системы должна хранить предысторию временных состояний наблюдаемых объектов. Таким образом, можно считать, что понятие состояния отображается в интеллектуальной управляющей системе в виде определенной динамической информационной модели [14].

Это приводит к понятию расчетно-логической ДЭС, где база знаний сочетает описание в виде строгих математических формул с информацией экспертов, а также соответственно – математические методы поиска решения с нестрогими эвристическими методами, причем вес того или другого компонента определяется возможностью адекватного описания предметной области и способом поиска решения.

Основные компоненты базы знаний:

- сведения о структуре и постоянных свойствах объектов и среды, с которыми взаимодействует система;
- совокупность логико-динамических моделей, описывающих поведение во времени с учетом допустимых управляющих воздействий;
- совокупность правил, законов и алгоритмов генерации реакции системы на поступающую входную информацию.

Обрабатывающее исполнительное ядро управляющей системы в обсуждаемой схеме можно рассматривать как интерпретатор модели процесса управления. Эта модель руководит работой исполнительного ядра и задает

логику выполняемых им действий. В таком виде программное обеспечение управляющей системы может быть построено как инвариантное, т.е. настраиваемое на задачу путем изменения описания модели требуемого процесса.

Формулу оценки количества информации при условии, что событие x является нечёткой лингвистической переменной, можно представить с помощью функции принадлежности $\mu(x)$. Более узкая кривая точнее определяет количественную характеристику нечёткой лингвистической переменной, что, в свою очередь, приводит к уменьшению площади под кривой, тогда (если принять во внимание, что нечёткие переменные приведены к универсальному интервалу [0..1]) получим:

$$I(x) = \left(\int_0^1 \mu(x) dx \right)^{-1} . \qquad (3.1)$$

Теперь величина $I(x)$ характеризует полноту базы данных. Однако эта величина не может учесть описание аномальных выбросов, поэтому в ДЭС предусмотрена коррекция лингвистических переменных в рамках универсального интервала [0..1]).

Из (3.1) видно, что уменьшение площади кривой приводит к увеличению среднего количества информации в базе знаний. Таким образом, максимальное значение $I(x)$ стремится к бесконечности, однако на практике из–за ограниченной точности измерительных приборов $I(x)$ будет ограничено сверху. Положим, что точность определения рассматриваемых в рамках работы лингвистических переменных будет отражена в интервале [0..1] диапазоном 0,05, тогда максимальное значение $I(x)$ = 1/(14×0,05 + 9×0,05) = 0,87. Здесь 14 – количество лингвистических переменных в базе знаний о дорожном полотне, а 9 – аналогичное количество, характеризующее поведение водителя (применительно к синтезу ИСУ повышения управляемости колёсного транспортного средства при торможении).

Минимальное значение $I(x)$ можно рассчитать аналогично, допустим, что все функции принадлежности имеют значение $I(x)$ = 1, тогда $I(x)$ = 1/(14×1 + +9×1) = 0,043. Имея такие оценочные границы количества информации, можно определить процентное соотношение полноты базы знаний.

3.2. Пример структурной схемы ИСУ повышения управляемости КТС с АБС

Основной задачей ИСУ КТС является повышение безопасности, улучшение показателей экономической эффективности использования КТС, снижение сложности управления, снижение вредных воздействий среды. В самом общем случае структура представления знаний ИСУ состоит из следующих основных подсистем: базы целей, предметных и процедурных знаний, включающих данные об обстановке на дороге, процедуры назначения весовых коэффициентов для определения риска, а также данные, обработанные нейронной сетью, по определению типа поверхности (рис. 3.1). Подразумевается далее, что реализован алгоритм оценки типа дорожного полотна на основе нейронной сети, что является дополнительным источником информации для ИСУ.

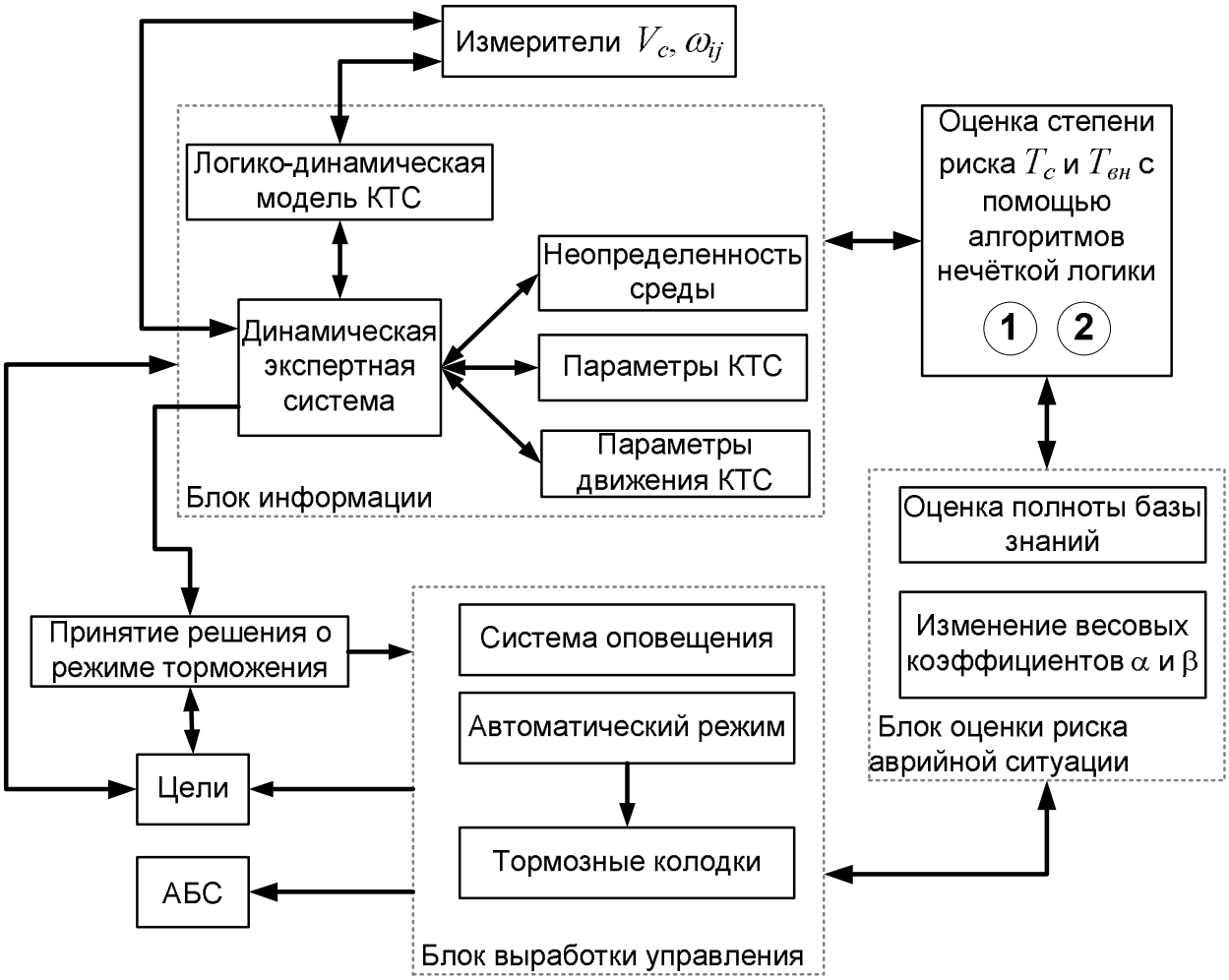

Рис. 3.1. Функционально-структурная схема ИСУ торможения КТС

Знания модели объекта управления, его динамических свойств и робастного закона управления [15] в совокупности с теоретическими положениями построения ИСУ создало предпосылки к построению интеллектуальных систем управления торможением колесного транспортного средства с АБС.

В концепции ИСУ предполагается её взаимодействие с окружающей средой, наличие мотивации, использование знаний для синтеза цели, оценки, принятия решений и выработки управления, контроль реальных результатов управления и сопоставление их с прогнозированными динамической экспертной системой результатами действия с корректировкой цели в случае необходимости. Для интеллектуальной системы управления торможением КТС с АБС характерна оценка взаимодействия пары «колесо – дорога», в которую

входят прогнозирование характера изменений текущих условий при наступлении возможной аварийной дорожной ситуации и выбор цели управления в соответствии с текущими и прогнозируемыми дорожными условиями.

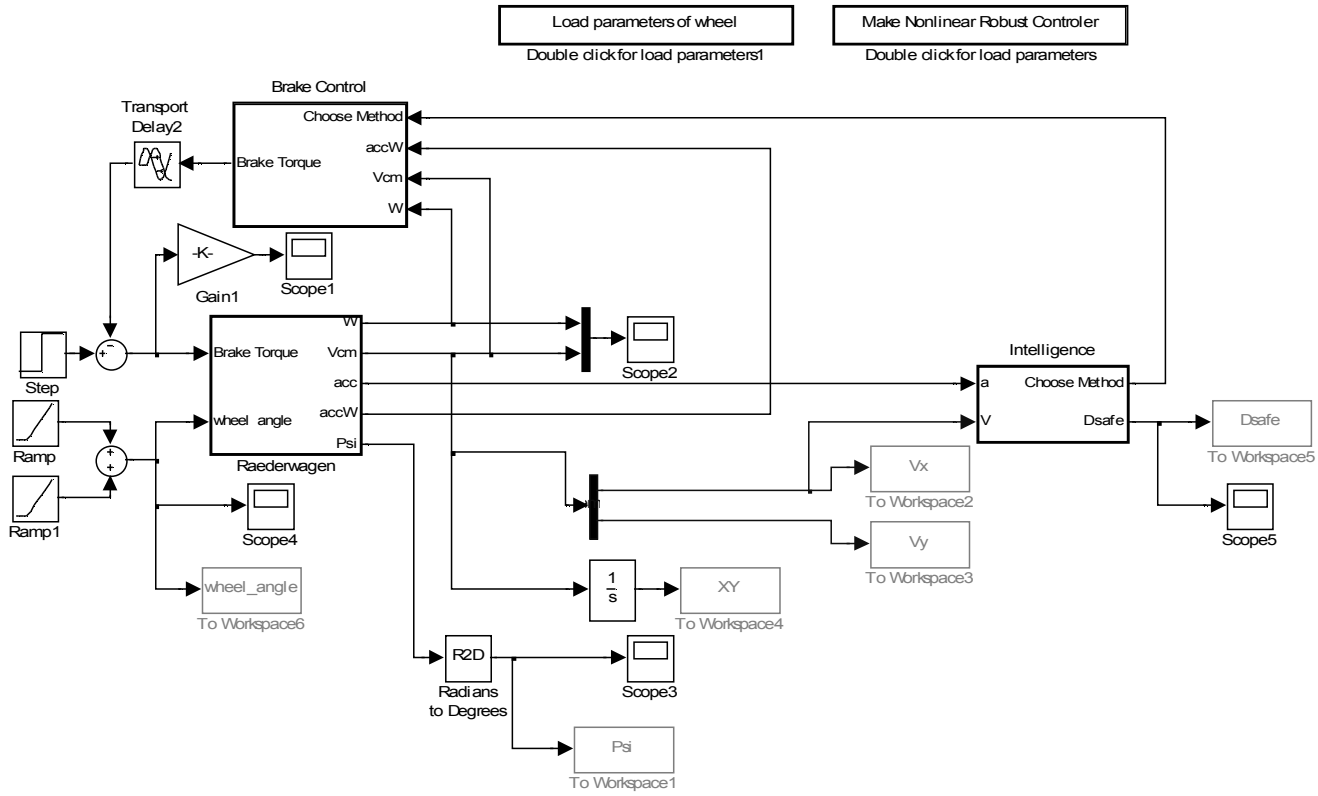

Рис. 3.2. К методике построения структурной схемы ИСУ

Применение четырехканальной схемы позволяет управлять торможением каждого колеса в отдельности, что способствует повышению управляемости КТС, однако имеющиеся разработанные системы повышения активной безопасности КТС при торможении имеют программный закон управления, часто угловой скоростью колеса, и в лучшем случае меняют настройки параметров контроллера управления. Одним из способов достижения заданного качества работы ИСУ повышения управляемости при торможении КТС является учет влияния свойств поверхности, на которой происходит торможение. Наиболее опасная ситуация – микст (торможение на поверхностях

с различными по бортам КТС свойствами сцепления колес), поэтому высокая точность его оценки – важная задача ИСУ, напрямую связанная с эффективностью торможения, а, следовательно, и с ослаблением развития (вплоть до предотвращения) аварийной ситуации.

3.3. Методика синтеза целей ИСУ

Разработка подхода к синтезу целей

Интеллектуальная система синтезирует цели, принимает решение к действию, обеспечивает действие для достижения цели, прогнозирует значения параметров результата действия и сопоставляет их с реальными текущими достигнутыми результатами, образуя обратную связь. Основываясь на таком описании функционирования ИСУ, можно проиллюстрировать методику синтеза на примере схемы рис. 3.3.

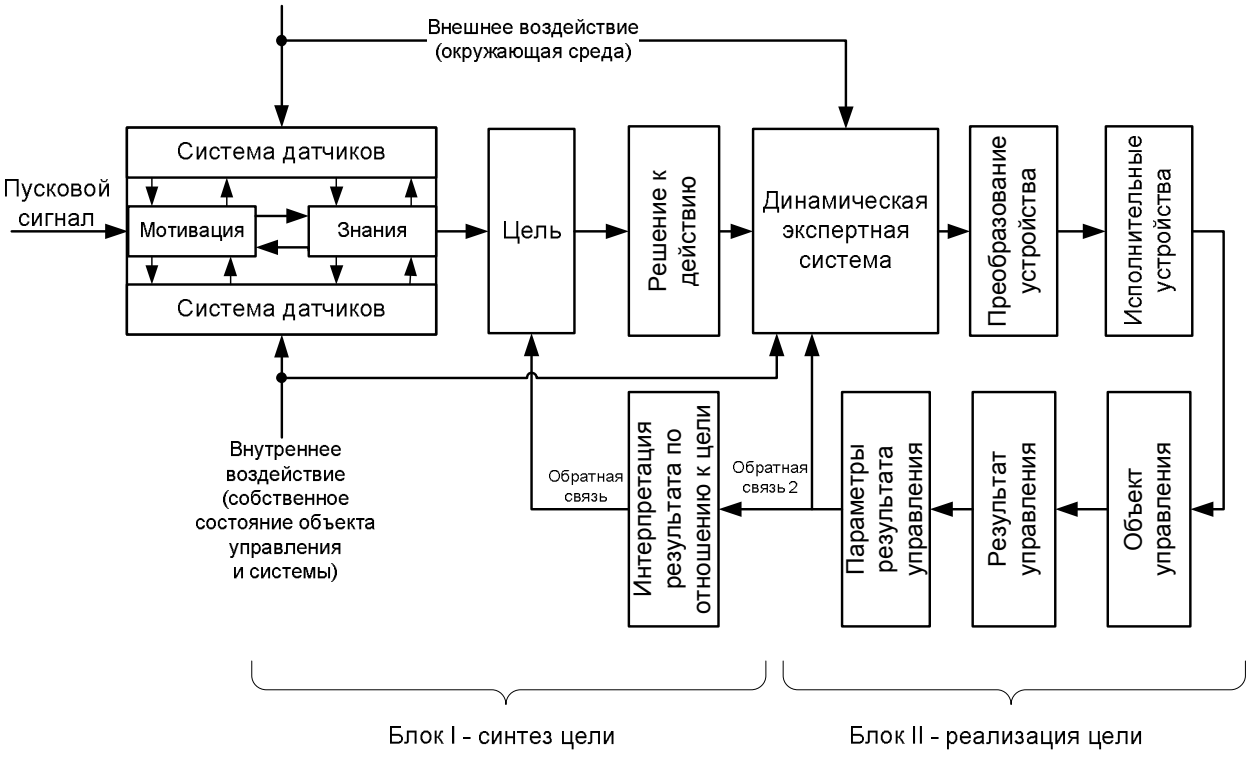

Рис. 3.3. Схема синтеза и реализации цели управления в ИСУ

В первом блоке на рис. 3.3 на основе активного оценивания информации, полученной от системы датчиков, при наличии мотивации и знаний

синтезируется цель и принимается решение к действию. Изменчивость окружающей среды и собственного состояния системы может приводить к потребности (мотивации) в определённых действиях (например, снижение скорости на опасных участках дороги для упреждения развития аварийной ситуации, которая могла бы возникнуть при торможении), а при наличии знаний может быть синтезирована цель. Продолжая активно оценивать информацию об окружающей среде и собственном состоянии системы, в том числе объекта управления, ИСУ при сопоставлении вариантов достижения цели можно принять решение к действию.

Модернизированный алгоритм развёртки дерева целей с учётом возможностей корректировки и оценки полноты базы знаний имеет вид, представленный на рис. 3.4.

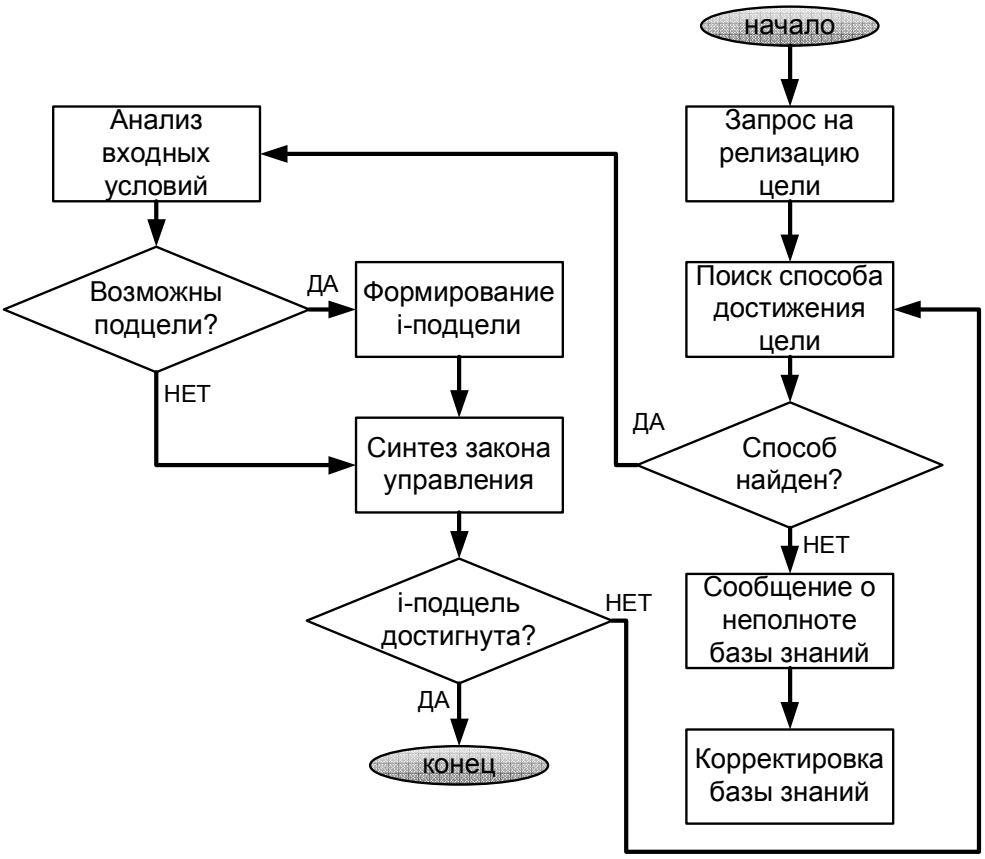

Рис. 3.4. Алгоритм реализации дерева целей

Синтез целей можно проводить на основе метода суждений. Этот метод является аналогом логического программирования и метода резолюций.

Правила такие же, как и в логическом программировании, но цель метода суждения состоит из подцелей, которые могут иметь ограничения.

Когда цели ИСУ сформированы, то возникает задача принятия решения, реализация которого обеспечивала бы оптимальное достижение цели. Функционирование ИСУ происходит в тесной связи с внешней средой, вносящей неопределённость, причем как сигнальную, так и параметрическую. Таким образом, возникает задача выбора решения, происходящее в динамической экспертной системе в условиях неопределённости.

Пример синтеза целей ИСУ повышения управляемости КТС с АБС

Для ИСУ торможением КТС с АБС можно предложить концепцию синтеза целей (рис. 3.5), которая состоит из определения подцелей ГЦ – повышения управляемости и устойчивости при торможении – и достигается выбором оптимального в данных условиях метода и синтеза закона управления, а также из задачи повышения полноты базы знаний (3.1). Также учтены и меры по упреждению развития аварийной ситуации.

Поддержание траектории движения при торможении связано с выполнением определённых подцелей нижнего уровня, таких как определение нарушения связи колеса в контакте с дорогой, повышения управляемости при торможении в условиях микста (наиболее опасная ситуация в процессе торможения) и повышение безопасности при низком сцеплении.

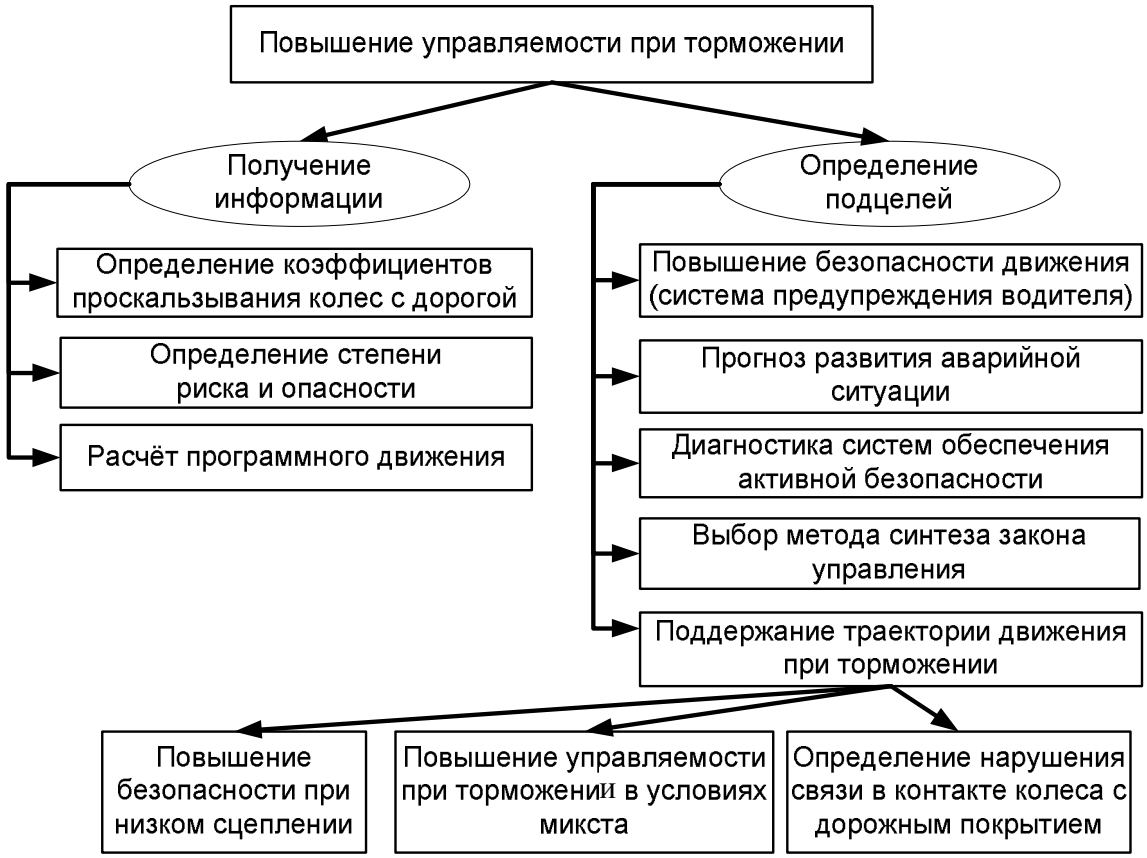

Рис. 3.5. Синтез целей ИСУ торможением КТС с АБС

Идея синтеза базируется на разделении задач – получения и обработки информации (определении подцелей). Нижний уровень целей связан непосредственной реализаций синтезированного закона управления. Декомпозиции ГЦ помогает формализовать постановку задач синтеза, а увеличение глубины декомпозиции позволяет использовать разработанные и известные методы синтеза.

3.4. Методика интеграции принципов робастного управления на основе H_∞-теории оптимизации в алгоритмическое обеспечение ИСУ

Постановка задачи синтеза робастного регулятора

Синтезируемая система представляется структурной схемой, изображённой на рис. 3.6 [13] .

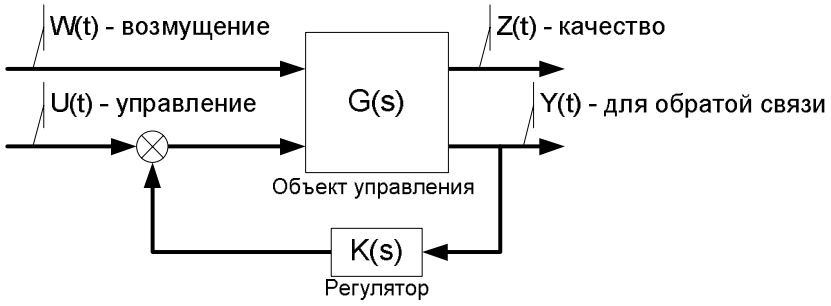

Рис. 3.6. Структурная схема синтезируемой системы

$G(s)=\begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix}=\begin{bmatrix} G_z^w(s) & G_z^u(s) \\ G_y^w(s) & G_y^u(s) \end{bmatrix}$ – многомерная передаточная функция (МПФ) объекта оптимизации от вектора $[w(t)\ \ u(t)]^T$ до вектора $\left[z^T(t)\ \ y^T(t)\right]^T$. Например, $G_{11}(s)\equiv G_z^w(s)$ – МПФ объекта от возмущения $w(t)$ до контролируемой переменной $z(t)$. Передаточная функция от возмущения $w(t)$ к контролируемой переменной $z(t)$ (см. рис. 3.7) системы, замкнутой регулятором $K(s)$.

Задачей H_∞-оптимизации является синтез такого регулятора K, который бы минимизировал H_∞-норму МПФ $T_z^w(s)$ от $w(t)$ до $z(t)$ замкнутой через него системы:

$$\left\|T_z^w(s)\right\|_{H_\infty} \overset{def}{=} \sup_{c>0}\sup_{\omega}\sqrt{\lambda_{\max} T_z^{w^T}(c-j\omega)T_z^w(c+j\omega)}.$$

Здесь $s=c+j\omega$ – комплексная переменная; $\lambda_{\max}$ – максимальное собственное значение квадратной матрицы: $T_z^{w^T}(c-j\omega)T_z^w(c+j\omega)$.

Показатель качества управления: $J(K)=\left\|T_z^w(s)\right\|_\infty$, оптимальное значение которого $J(K_{opt})=\inf_K\left\|T_z^w(s)\right\|_\infty=\gamma_{opt}$.

В этом случае регулятор обеспечит минимальное влияние возмущений, в том числе и самого плохого, поэтому данная задача по своей сути является

минимаксной. Погружение задачи в ограничения, накладываемые H_∞-теорией, рассматриваются в [13].

3.5. Классический подход к построению оптимального робастного регулятора

Для построения регулятора по классическому методу надо рассмотреть вопросы, связанные со стабилизацией исходной системы. Рассматриваемая система в этом методе не отличается от ранее описанных и соответствует описанию в пространстве состояний с передаточной функцией:

$$G(s) = \left[\begin{array}{c|c} \frac{G_z^w(s)}{G_y^w(s)} & \frac{G_z^u(s)}{G_y^u(s)} \end{array}\right] \Leftrightarrow \left[\begin{array}{ccc} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right].$$

Это описание эквивалентно представлению

$$G(s) = \left[\begin{array}{c|c} \frac{G_{11}(s)}{G_{21}(s)} & \frac{G_{12}(s)}{G_{22}(s)} \end{array}\right].$$

Замкнутая система определяется следующими соотношениями:

$$\begin{bmatrix} Z \\ Y \\ E \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \Phi_{12} & \Phi_{13} \\ \Phi_{21} & \Phi_{22} & \Phi_{23} \\ \Phi_{31} & \Phi_{32} & \Phi_{33} \end{bmatrix} \begin{bmatrix} W \\ U \\ U_{noise} \end{bmatrix}.$$

Система называется внутренне устойчивой, если все девять функций $\Phi_{ij}(s)$ устойчивы. Данное определение в практических целях использовать неудобно. Воспользуемся условием $G_{11}(s), G_{12}(s), G_{21}(s) \in \Re H^\infty$. Тогда условия внутренней устойчивости определяются более просто. Если регулятор стабилизирует $G_{22}(s)$, то он стабилизирует и всю $G(s)$.

Применение классического метода дает представление регулятора в следующем виде:

$$K(s) = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & 0 \end{bmatrix}.$$

Здесь $K(s) = X \cdot Y^{-1}$ – центральный регулятор, $G(s) = M \cdot N^{-1}$ – расширенное представление объекта управления.

Для реализации регулятора необходимо решить следующие уравнения, перекликающиеся с методом модального управления.

$$\dot{\tilde{x}} = A\tilde{x} + B_2 + u_2 - H\tilde{y}$$

$$u_2 = F\tilde{x} + \tilde{u}_2$$

$$\tilde{y}_2 = y_2 - C_2\tilde{x} + D_{22}u_2 .$$

Применение методов модального управления и понятия наблюдающих устройств позволяет сформировать указанную систему уравнений. Возникает необходимость задания матриц *F, H*. Разработчик обладает определенной свободой при выборе указанных матриц обратной связи, выражающейся в назначении желаемых корней характеристического многочлена.

Методика, описанная в [13], дает возможность программной реализации поиска матриц обратной связи с заданными желаемыми вещественными корнями характеристического многочлена.

В классическом подходе к построению робастного регулятора выделяем две системы для поиска матриц *F, H*. Матрица *F* является стабилизирующей матрицей обратной связи для исходной системы:

$$G(s) = \begin{bmatrix} A & B \\ C & D \end{bmatrix}.$$

Постановка *F* в обратную связь дает соответствующие желаемые корни уравнения. Матрица *H* является стабилизирующей матрицей обратной связи для транспонированной системы (обозначение «с волной»).

$$\tilde{G}(s) = \begin{bmatrix} A_c & B_c \\ C_c & D_c \end{bmatrix} = \begin{bmatrix} A^T & C^T \\ B^T & D^T \end{bmatrix}.$$

Зная матрицы *F, H,* по известному алгоритму находим компоненты двойной взаимно простой факторизации.

$$M = \begin{bmatrix} A + BF & B \\ C + DF & D \end{bmatrix} \qquad N = \begin{bmatrix} A + BF & B \\ F & I_{n\times n} \end{bmatrix}$$

$$\tilde{M} = \begin{bmatrix} (A_c + B_c H)^T & (C_c + D_c H)^T \\ {B_c}^T & D^T \end{bmatrix} \quad \tilde{N} = \begin{bmatrix} (A_c + B_c H)^T & H^T \\ {B_c}^T & I_{n\times n} \end{bmatrix}$$

$$X = \begin{bmatrix} A+BF & -H^T \\ F & O_{n\times n} \end{bmatrix} \qquad Y = \begin{bmatrix} A+BF & -H^T \\ C+DF & I_{n\times n} \end{bmatrix}$$

$$\widetilde{Y} = \begin{bmatrix} (A_c + B_c H)^T & -(C_c + D_c H)^T \\ F & I_{n\times n} \end{bmatrix} \quad \widetilde{X} = \begin{bmatrix} (A_c + B_c H)^T & -H^T \\ F & O_{n\times n} \end{bmatrix}$$

Выбор стабилизирующего регулятора связан с проведением двойной взаимно простой факторизации (ДВПФ) объекта $G_{22}(s)$. При этом если обозначить передаточную функцию числителя и знаменателя объекта как M и N соответственно, а передаточную функцию числителя и знаменателя регулятора как K_a, K_b, то условие ДВПФ следующее:

$$\begin{bmatrix} \widetilde{K}_b & -\widetilde{K}_a \\ -\widetilde{M} & \widetilde{N} \end{bmatrix} \begin{bmatrix} N & K_a \\ M & K_b \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}.$$

Структурная схема с учетом введенной параметризации принимает вид:

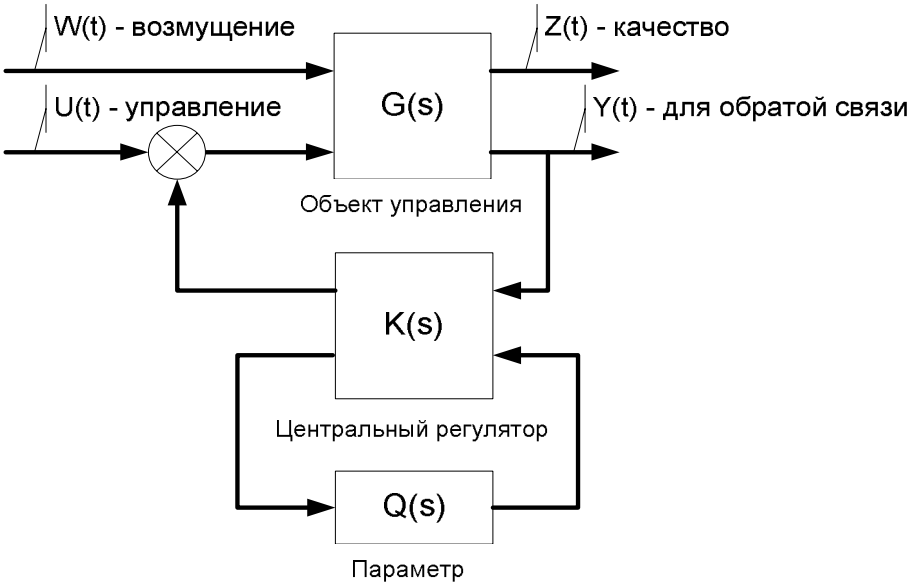

Рис. 3.7. Параметризация стабилизирующих регуляторов

Одно из достоинств параметризации класса регуляторов – параметр $Q(s)$ входит в выражение ПФ замкнутой системы линейно. Это позволяет проводить более простой поиск оптимальных (по конкретному критерию оптимальности) регуляторов в указанном классе. Использование центрального регулятора

приводит к получению робастной системы, удовлетворяющей разработчика по качеству, принятому в пространстве H_∞. Параметризация позволяет влиять на качество переходного процесса в привычном для специалиста виде – «коробочка» Солодовникова, время переходного процесса, перерегулирование.

Достоинство классического метода построения регулятора – в свободном выборе желаемых характеристик переходных процессов. Недостаток – высокий порядок регулятора.

Методика синтеза оптимального H_2-регулятора

Оптимальный регулятор строится за конечное число операций.

Алгоритм построения имеет линейную структуру и состоит из следующих этапов:

1) вводим найденные в результате линеаризации матрицы представления объекта в пространстве состояний (A, B_1, B_2, C_1, C_2);

2) решаем CARE: $A^T X_2 + X_2 A - X_2 B_2 B_2^T X_2 + C_1^T C_1 = 0$;

3) решаем FARE: $A Y_2 + Y_2 A^T - Y_2 C_2^T C_2 Y_2 + B_1 B_1^T = 0$;

4) ABCD-представление искомого регулятора:

$$K_2(s) \overset{\Delta}{=} \begin{bmatrix} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{bmatrix} = \begin{bmatrix} A + B_2 F_2 + L_2 C_2 & -L_2 \\ F_2 & 0 \end{bmatrix}, \qquad (3.2)$$

где $F_2 = -B_2^T X_2; L_2 = -Y_2 C_2^T$.

Методика построения оптимального H_∞-регулятора

Поскольку H_∞-норма передаточной функции T_z^w есть корень квадратный из энергии выхода при подаче на вход сигнала с единичной энергией, то минимизация $\left\| T_z^w \right\|_\infty$ означает минимизацию энергии ошибки для наихудшего случая (из некоторого класса входного возмущения).

Алгоритм синтеза оптимального H_∞-регулятора намного сложнее рассмотренного ранее: в отличие от H_2-варианта, H_∞-регулятор (как и H_∞-норма) не может быть определён за конечное число операций и потому требует специальной итерационной процедуры (рис. 3.8).

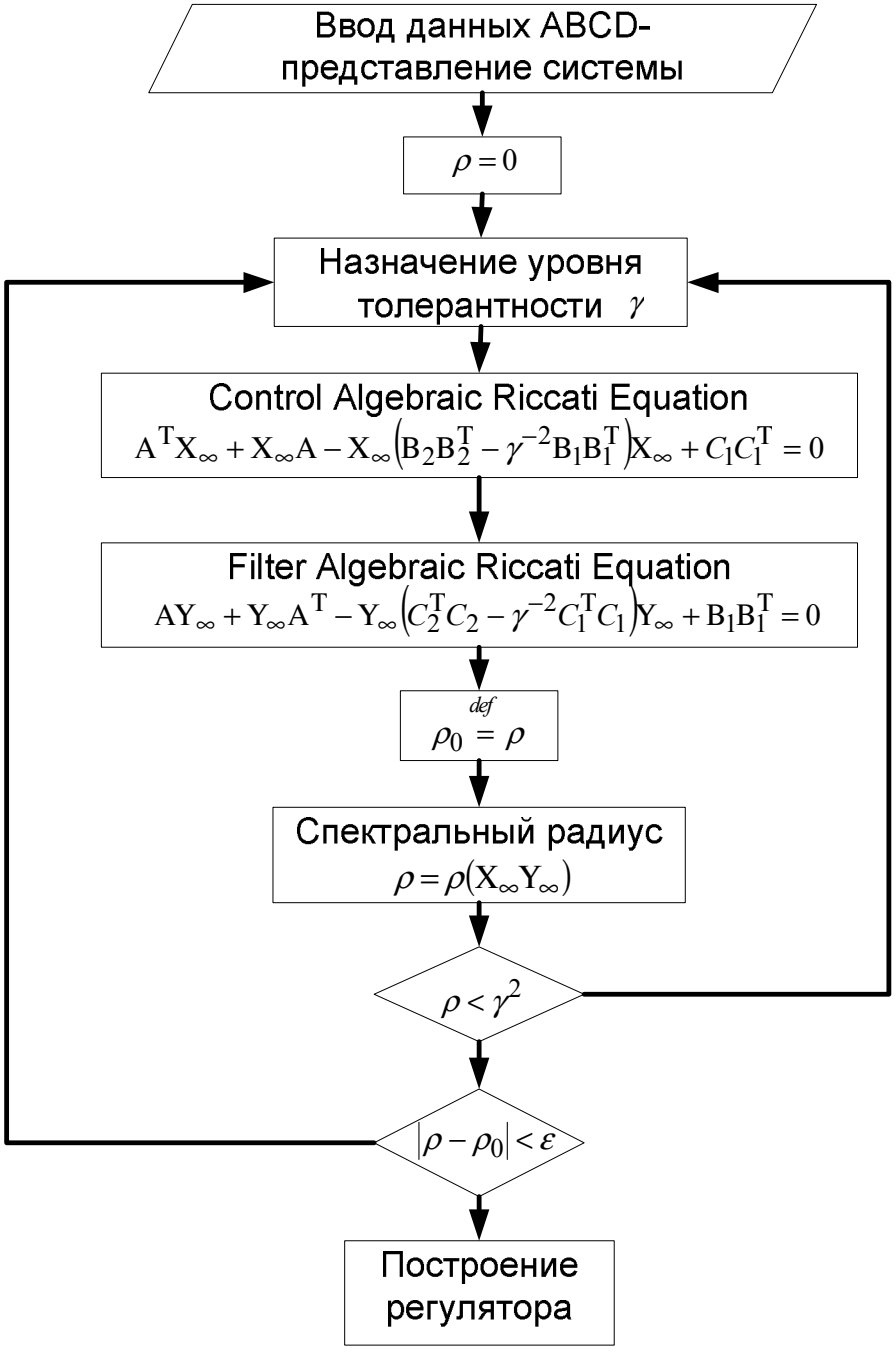

Рис. 3.8. Алгоритм синтеза H_∞-регулятора

Искомый H_∞-регулятор, у которого $\left\| T_z^w \right\|_\infty < \gamma$,

$$K_\infty(s) \overset{\Delta}{=} \begin{bmatrix} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{bmatrix} = \begin{bmatrix} A + \gamma^{-2} B_1 B_1^T X_\infty + B_2 F_\infty + Z_\infty L_\infty C_2 & -Z_\infty L_\infty \\ F_\infty & 0 \end{bmatrix} \quad (3.3)$$

Предложенный алгоритм дает возможность только приближаться к оптимальному регулятору, т.е. построить лишь субоптимальный регулятор. Построение таким способом H_∞-регулятора намного более трудоёмкое ещё и потому, что необходимо решать два уравнения Риккати в каждом цикле выбора параметра γ, а для H_2-варианта уравнения решаются только один раз. Траектории КТС с робастным регулятором отличаются друг от друга не сильно в силу малости периода рассматриваемого времени L и быстрой скорости сходимости итерационной процедуры.

3.6. Пример синтеза робастного регулятора в рамках классического подхода

Для анализа эффективности регулятора рассмотрим линейную систему с передаточной функцией вида:

$$W_o = \frac{K(T_1 s + 1)}{{T_2}^2 s^2 + 2T_2 \xi s + 1}.$$

Подбор параметров осуществим из условия устойчивости системы W_o.

В пространстве состояний матрицы системы *G(s)*:

$$A = \begin{bmatrix} -2 & -100 \\ 1 & 0 \end{bmatrix}; B_i = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; C_i = \begin{bmatrix} 10 & 100 \\ 10 & 100 \end{bmatrix}; D_{ij} = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}, \forall i, j = 1..2$$

Корневой годограф исходной системы (рис 3.9 и 3.10). Выбираем желаемые корни -10 и -20. Для таких корней строим центральный регулятор.

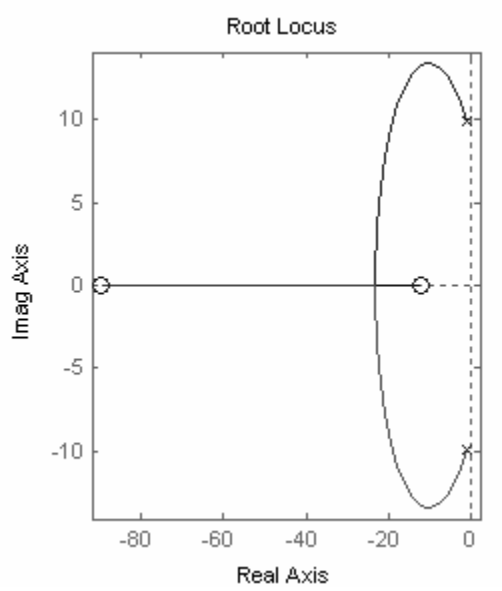

Рис. 3.9. Корневой годограф исходной системы

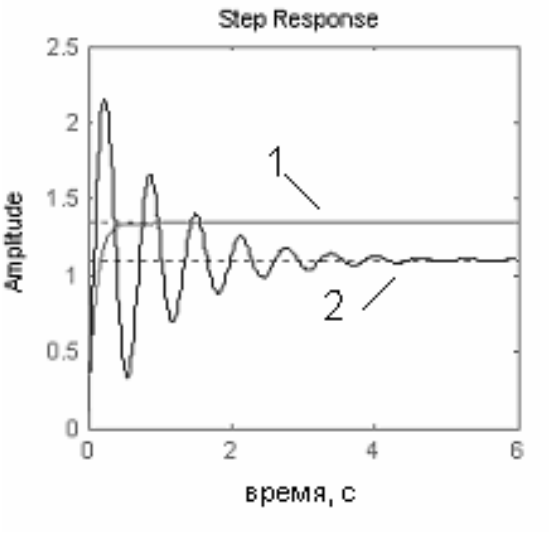

Рис. 3.10. Сравнение переходных процессов: 1– с регулятором, 2 – без регулятора

Выясним влияние параметров передаточной функции, позволяющей параметризовать центральный регулятор. Выбираем простейшее звено:

$$Q(s) = \frac{K_q}{T_q s + 1}.$$

Исследуем поведение системы с параметризацией указанной передаточной функцией на области изменения параметров $T_q = K_q = 0{,}5\ldots3$ (рис. 3.11 и 3.12).

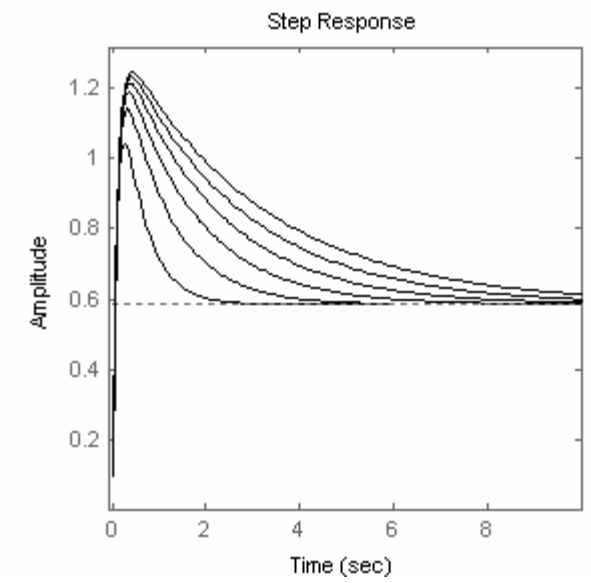

Рис. 3.11. T_q = var, K_q = fix

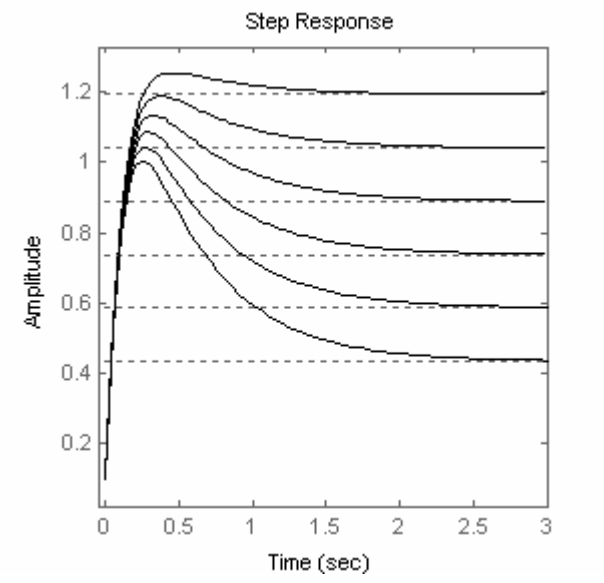

Рис. 3.12. T_q = fix , K_q = var

Анализируя влияния параметров *Q(s)*, приходим к выводу – изменение T_q приводит к пропорциональному увеличению времени переходного процесса системы с регулятором, а увеличение K_q уменьшает установившееся значение переходного процесса.

Достоинство классического метода построения регулятора в свободном выборе желаемых характеристик переходных процессов. Недостаток – высокий порядок регулятора. Порядок центрального регулятора равен 4, тогда как порядок системы равен 2, а порядок параметризированного регулятора = 10. Это объясняет выбор наиболее простой передаточной функции *Q(s)*.

3.7. Пример синтеза оптимального H_2- и H_∞-регулятора

Для синтеза регулятора выбираем линейную систему с передаточной функцией вида:

$$W_o = \frac{K(T_1 s + 1)}{T_2{}^2 s^2 + 2T_2 \xi s + 1}$$

Подбор параметров осуществим из условия устойчивости системы W_o

В пространстве состояний матрицы системы *G(s)*:

$$A = \begin{bmatrix} -2 & -100 \\ 1 & 0 \end{bmatrix}; B_i = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; C_i = \begin{bmatrix} 10 & 100 \\ 10 & 100 \end{bmatrix}; D_{ij} = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}, \forall i, j = 1..2$$

По алгоритмам, описанным выше, строим H_2- и H_∞-регулятор. Возмущающее воздействие – случайное, ограниченное по норме [-1…1].

Для оценки качества по величине бесконечной нормы в пространстве Харди построим кривые Найквиста систем с различными регуляторами (рис. 3.13).

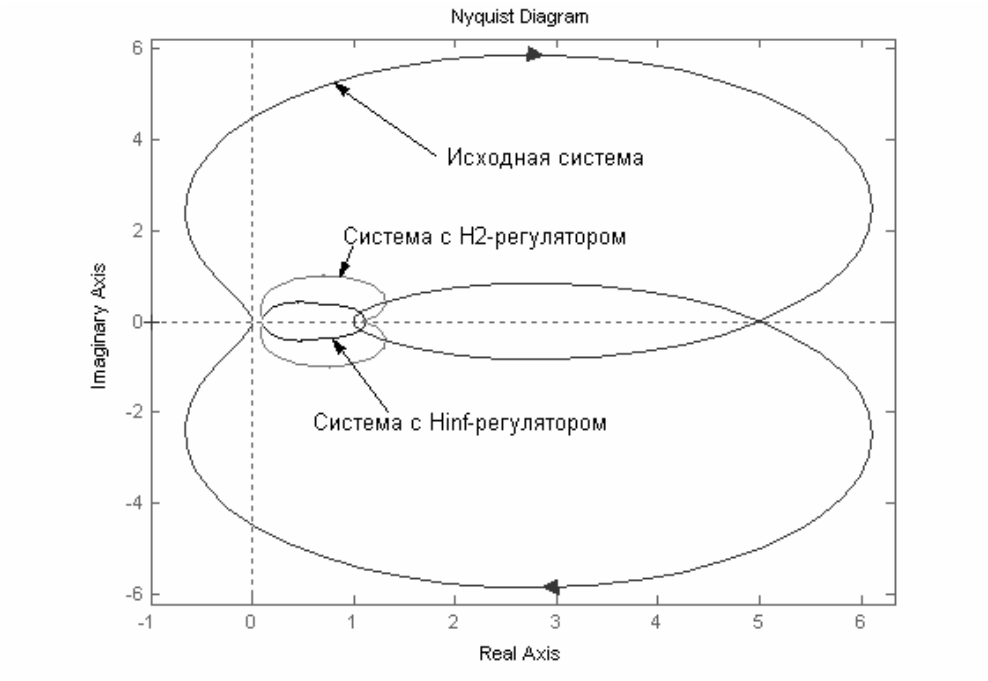

Рис. 3.13. Кривая Найквиста для систем с различными регуляторами

В случае построения H_∞-регулятора можно выделить набор субоптимальных регуляторов с уровнем толерантности γ. Для построения оптимального регулятора можно с помощью моделирования определить γ_{opt} и для уровня толерантности $\gamma = \gamma_{opt}$ построить оптимальный H_∞-регулятор.

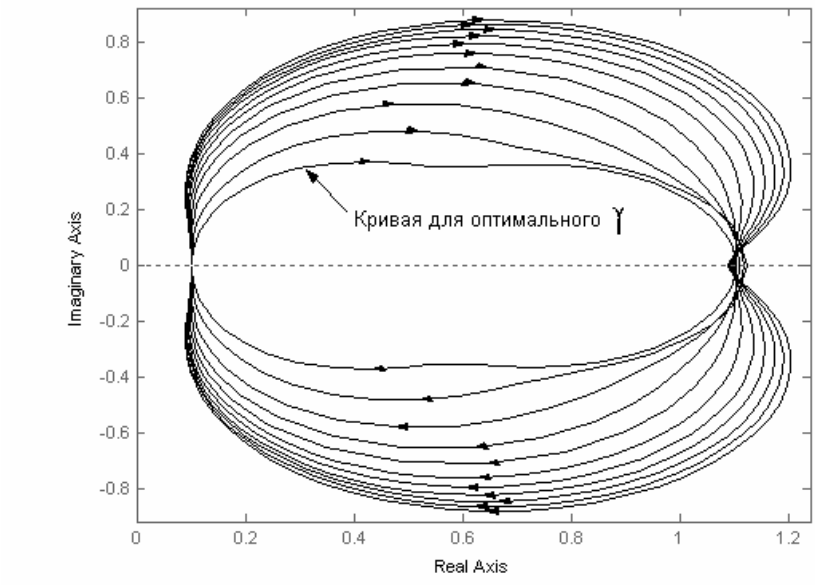

Рис. 3.14. Кривая Найквиста для замкнутой системы с H_∞-регулятором для различных уровней толерантности

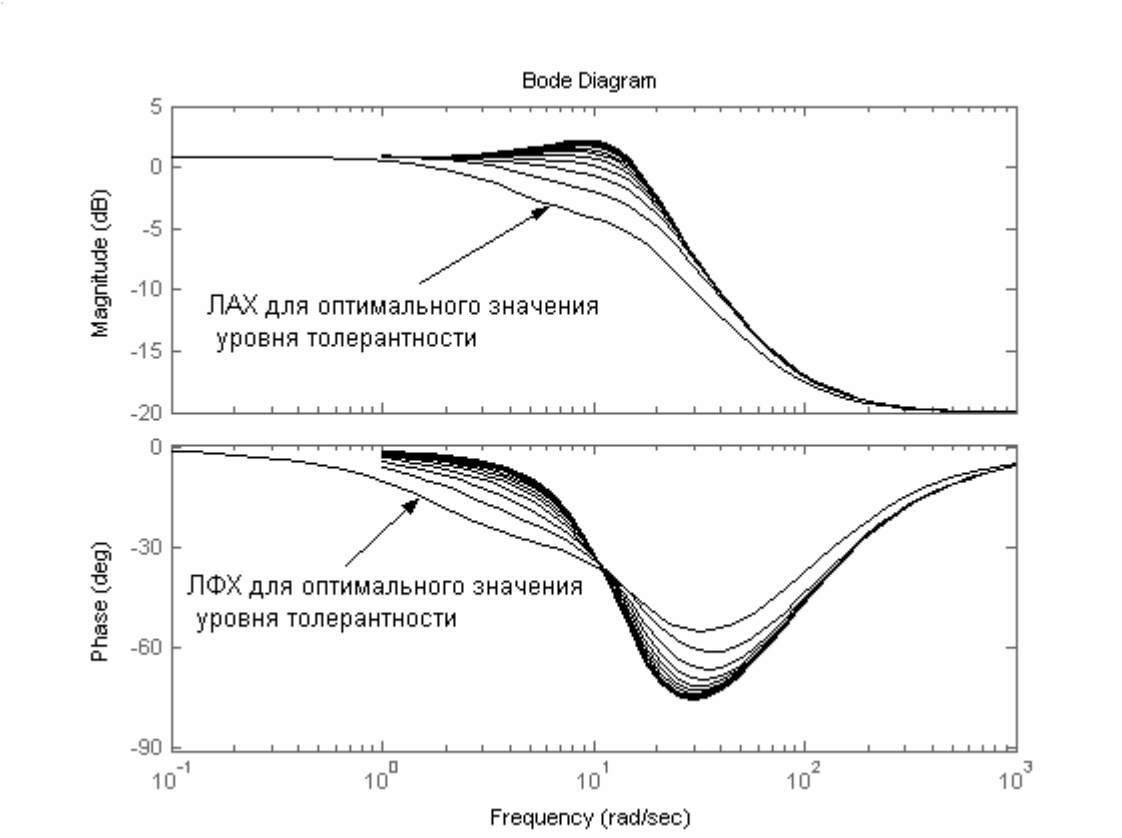

Рис. 3.15. ЛАХ для замкнутой системы с H_∞-регулятором для различных уровней толерантности

На рис. 3.14, 3.15 построены соответственно кривые Найквиста и ЛАХ для различных уровней толерантности $\gamma \geq \gamma_{opt}$. Оптимальному значению уровня толерантности соответствует ЛАХ с наилучшими свойствами подавления шумов.

Параметры робастного регулятора, оптимального в смысле $\inf_K \left\| F_L(G, K) \right\|_\infty = \gamma_{opt}$:

$$\widehat{A} = \begin{bmatrix} -171{,}8 & -1722{,}8 \\ -5{,}1 & -61{,}1 \end{bmatrix}; \widehat{B} = \begin{bmatrix} 7{,}9 & 7{,}9 \\ 0{,}3 & 0{,}3 \end{bmatrix};$$

$$\widehat{C} = \begin{bmatrix} -23{,}1 & -82{,}6 \end{bmatrix}; \widehat{D} = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

3.8. Динамическая экспертная система как основная часть ИСУ

Методическое обоснование синтеза структурной схемы базируется на понятии афферентного синтеза цели (базы построения целенаправленной деятельности), который основан на исходной доминирующей мотивации, обстановочной и пусковой афферентации и памяти [22]. Механизм афферентного синтеза можно описать так: на основе исходной доминирующей мотивации, возникающей в результате внутренней потребности и памяти, ИСУ по генерируемым вычислительной сетью сигналам активно оценивает возмущения внешней среды, вырабатывает цель и принимает соответствующее решение к действию.

Динамическая экспертная система (выбирая цель управления автоматически, исключает человека из участия в управлении) с использованием базы знаний производит экспертную оценку, на основании которой принимается решение о действии и прогнозируются результаты действия (акцептор действия).

На основании сведений об окружающей среде и собственном состоянии системы при наличии памяти и мотивации синтезируется цель, которая наряду с другими данными воспринимается динамической экспертной системой. ДЭС оценивает полноту информации, которая доступна ИСУ, и выбирает наиболее подходящий метод синтеза управления (например, в условиях, когда имеется лишь предположение об ограничениях на норму возмущающего сигнала, ДЭС выбирает робастный метод управления, основанный на H_∞-теории, для синтеза регулятора). ДЭС работает в реальном времени (в темпе с управляемым процессом) и в ускоренном масштабе времени (что позволяет прогнозировать поведение системы и использовать результат прогноза при принятии решения в реальном времени).

В соответствии с принятым решением вырабатывается управление, т.е. синтезируется тот или иной алгоритм или закон управления, который реализуется с помощью различных исполнительных органов и воздействует непосредственно на объект управления. Результаты этого воздействия

сравниваются с прогнозируемыми (механизм обратной связи, акцептор действия). При несоответствии результатов на базе новой экспертной оценки принимается решение, вырабатывается и реализуется управление, устраняющее это несоответствие. При соответствии результатов подкрепляется предшествующее управление. Если соответствие недостижимо, то уточняется цель.

3.9. Методика построения модели ДЭС

В процессе работы ДЭС последовательность действий может быть следующей: анализ текущей ситуации, оценка возможных событий и решений или исходов. Таким образом, имеется цепочка отображений вида: ситуация → событие → решение. Эта цепочка не является законченной, так как принятое решение может создать свои ситуацию, предшествующую новому событию, приводящему к новому решению, и т.д. Отсюда очевидно наличие причинно-следственной связи «ситуация → решение». В связи с этим возникает необходимость замкнуть цепочку отображений при учете того, что не соблюдается оптимальность принятых способов достижения целей.

Существуют различные способы организации работы ДЭС посредством применения одного из видов систем аналитического типа. Рассмотрим их достоинства и недостатки;

1) предметно-ориентированные аналитические системы. Они отличаются высокой степенью автоматизации и довольно дёшевы. К отрицательным сторонам можно отнести использование специфического интерфейса, часто такие системы не допускают гибкой перестройки.

2) статистические пакеты. Достоинство таких программных пакетов в том, что они выполняют одни и те же операции, что позволяет использовать более быстрые, по сравнению с универсальными, специальные вычислительные средства. Методика, применяющаяся в таких программах, давно апробирована. Однако статистические пакеты могут требовать программирования на внутреннем языке, их рыночная стоимость достаточно высока. Основной

недостаток в том, что они ограничены запрограммированной областью применения;

3) нейронные сети. Отличаются, пожалуй, самой высокой эффективностью и универсальностью, хотя и обладают известной сложностью. К недостаткам можно отнести не самую прозрачную модель, а также значительную стоимость;

4) деревья решений. Правила, используемые для принятия решений по такому методу, просты и наглядны. В связи с универсальностью построения деревьев возникает и широкая область их применения. Метод обладает достаточным быстродействием. Отрицательной стороной метода является проблема выбора нужного признака, позволяющего построить эффективные правила;

5) генетические алгоритмы. Несомненно, являются мощным средством оптимизации, однако требуют умения эффективно определить постанову задачи, правила отбора хромосом, способов мутации.

В каждом конкретном случае разработчик ИСУ выбирает наиболее эффективный метод построения ДЭС. Для ИСУ торможения КТС больше подходит предметно-ориентированная аналитическая система или деревья решений, так как требуется достигнуть цель в жёстких временных рамках, а детальная проработка объекта управления позволит избежать применения мощных, но иных рассмотренных расточительных по времени алгоритмов.

Преимущества использования деревьев решений [18] :

- быстрый процесс обучения;
- генерация правил в областях, где эксперту трудно формализовать свои знания;
- извлечение правил на естественном языке;
- интуитивно понятная классификационная модель;
- высокая точность прогноза, сопоставимая с другими методами (статистика, нейронные сети);
- построение непараметрических моделей.

Наиболее перспективным видится объединение предметно-ориентированных имеющихся систем повышения безопасности движения КТС под управлением ИСУ с ДЭС, работающей по алгоритму деревьев решений, что позволит обеспечить повышение точности и надёжности работы системы.

3.10. Методика разработки ПО ДЭС ИСУ повышения КТС с АБС

Методические основы создания ПО ИСУ могут быть представлены следующими этапами [23]:

- изучение предметной области;
- разработка иерархической структуры целей;
- определение критериев достижения целей и условий перехода от одной цели к другой;
- определение структуры и содержания базы знаний и базы данных;
- создание концептуальной программной модели ИСУ;
- отладка.

Содержание перечисленных основных этапов подробнее представлено в табл. 3.1. Там же приводится пример создания ПО ИСУ повышения управляемости торможения колёсного транспортного средства (КТС) с антиблокировочной системой (АБС).

Таблица 3.1. Методические основы создания ПО ИСУ

№ п/п	Этап создания	Описание этапа
1	Изучение предметной области	Создание модели объекта управления, его динамических характеристик, модели внешней среды и их взаимодействия. Моделирование и анализ полученных результатов
2	Разработка иерархической структуры целей	На основе моделирования и изучения предметной области выделяется главная цель (ГЦ), выделяются подцели. Строится так называемое дерево целей

Окончание табл. 3.2

3	Определение критериев достижения целей и условий перехода от одной цели к другой	Составленное на предыдущем этапе иерархическое дерево целей проверяется на логическую непротиворечивость и полноту описания. Критерии достижения и условия перехода формулируются в качестве подмножеств фазовых координат
4	Определение структуры и содержания базы знаний и базы данных	На этом этапе формируется структура базы данных, достаточная для описания характеристик объекта управления. База данных и критерии достижения целей позволяют наполнить базу знаний ИСУ. Первоначальное наполнение строится на мнении экспертов предметной области
5	Создание концептуальной программной модели ИСУ	Консолидация результатов предыдущих этапов приводит к созданию верхнего уровня ПО ИСУ. Концептуальная программная модель позволяет отработать основные алгоритмы управления, выявить неучтённые критерии переходов между целями
6	Разработка ПО нижнего уровня (ПО отдельных блоков функциональной схемы ИСУ)	Формируется техническое задание для создания ПО нижнего уровня исходя из образа системы, полученного на этапе концептуального программирования
7	Отладка	На этом этапе производится коррекция ошибок, внесенных в ПО на предыдущих этапах. Особое внимание уделяется взаимосвязи отдельных блоков как в горизонтальном, так и в вертикальном направлении. Этот этап характеризуется высокой трудоёмкостью

3.11. Пример использования методики синтеза модели ДЭС для ИСУ КТС с АБС

Программная реализация метода деревьев решений может быть осуществлена в среде Matlab, в toolbox Stateflow. Пусть задано некоторое множество *T*, содержащее цели (объекты), каждая из которых характеризуется *m* атрибутами (свойствами), причем один из них указывает на принадлежность объекта к определенному классу, другой определяет условия реализации цели, условия перехода к другой цели и т.д. Идея построения деревьев решений из множества *T*, впервые высказанная Хантом, приводится по Р. Куинлену (R. Quinlan).

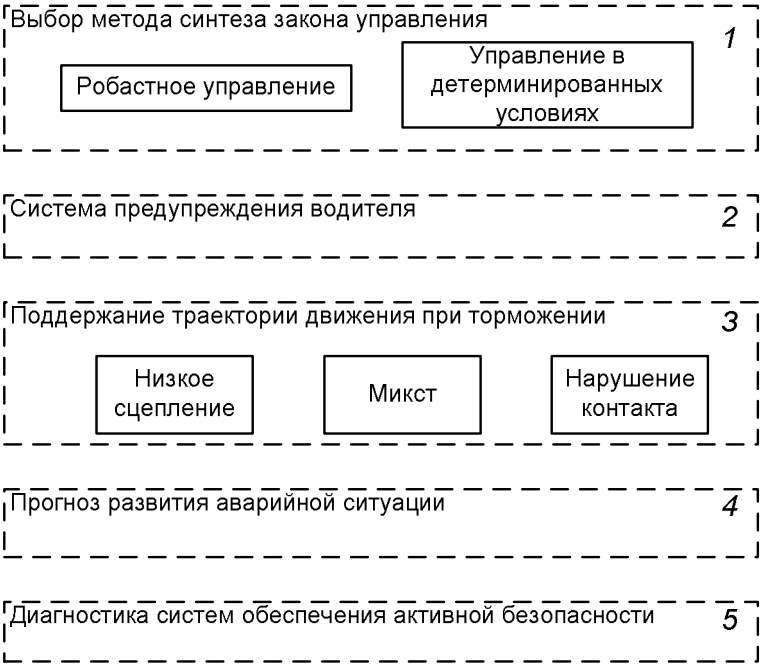

Рис. 3.16. Представление иерархии дерева целей в ДЭС

В терминах структуры Stateflow пунктирные блоки выполняются квазипараллельно, т.е. в такт с разрядностью бортового процессора. Последовательность их выполнения обозначается цифрой в правом верхнем углу. Таким образом, иерархическое дерево целей (см. рис. 3.5) представляется в ДЭС в виде рис. 3.16.

Блоки из сплошных линий выполняются в условиях ограничивающего блока и каждый конкретный момент времени может выполняться только один сплошной блок. Сплошные блоки реализуют последовательную схему выполнения [16].

Для построения дерева на каждом внутреннем узле необходимо найти такое условие (проверку), которое бы разбивало множество, ассоциированное с этим узлом, на подмножества. В качестве такой проверки должен быть выбран один из атрибутов. Общее правило для выбора атрибута можно сформулировать следующим образом: выбранный атрибут должен разбить множество так, чтобы получаемые в итоге подмножества состояли из объектов, принадлежащих к одному классу, или были максимально приближены к этому, т.е. количество объектов из других классов («примесей») в каждом из этих множеств было как можно меньше.

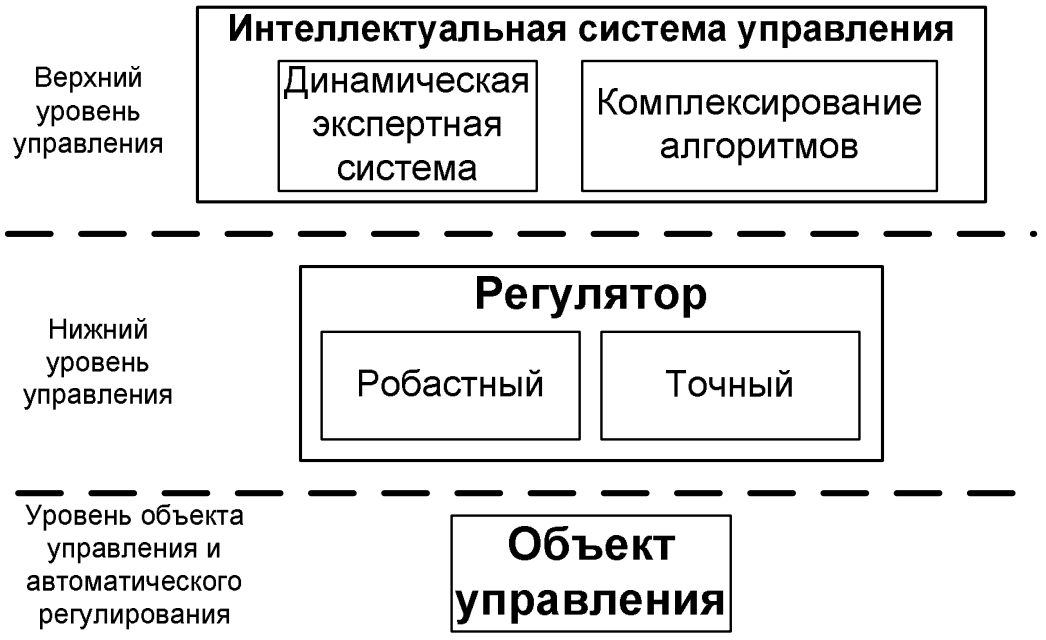

Рис. 3.17. К методике комплексирования алгоритмов ИСУ робастным регулятором

Каждый блок предоставления целей в ДЭС обозначает определённое состояние ИСУ, в котором происходит целенаправленная деятельность на достижение конкретной цели. В математическое описание ДЭС введём:

переменную Method – выбор метода синтеза управления, вектор Aim, имеющий размерность равную 5. Вектор Aim и переменная Method являются выходной величиной и, следовательно, результатом автоматического целеполагания. На ряду с этим, входными величинами будут: результат диагностики системы обеспечения активной безопасности (Check), мера полноты базы знаний (Info_fullness), оценка риска (risk) и опасности (danger) аварийной ситуации, оценка свойств поверхности торможения (Info_wheel_drive).

На основе таких входных данных формируем закон перехода от одного состояния к другому. На поле входных параметров вводим атрибуты – граничные значения, отвечающие за изменение состояния. Для наглядного изображения решающей функции приходится выбрать отдельные характерные сечения, которые отражены в табл. 3.2.

Таблица 3.3. Сечения решающей функции

Переменная состояния	Условие входа в состояние	Условие выхода из состояния
Method – выбор метода синтеза закона управления.		
Робастный	[Info_fullness<0.5]	[Info_fullness>0.8]
Детерминированный	[Info_fullness>0.8]	[Info_fullness<0.5]
Aim(1) – система предупреждения водителя	[danger>0.7]	[danger<=0.7]
Aim(2) – прогноз развития аварийной ситуации	Переход к робастному управлению, при [Info_wheel_drive<0.6]	[Info_wheel_drive>=0.6]
Aim(3) – нарушение контакта	Постоянное отслеживание, при условии, что не выполняются другие цели	[(Info_fullness<0.5) или (danger>0.7)]

Окончание табл. 3.2

Aim(4) – микст	[(danger>0.6) и (Info_wheel_drive<0.6)].	[(danger<0.3) или (risk<0.1)]
Aim(5) – низкое сцепление	[((danger>0.4) или (risk>0.6)) и (Info_wheel_drive>=0.6)]	[danger<0.4]
Диагностика системы	Проводится постоянно, при определении неисправности включается система предупреждения водителя	

Модель работы ДЭС можно получить в Stateflow среды Matlab (рис. 3.18).

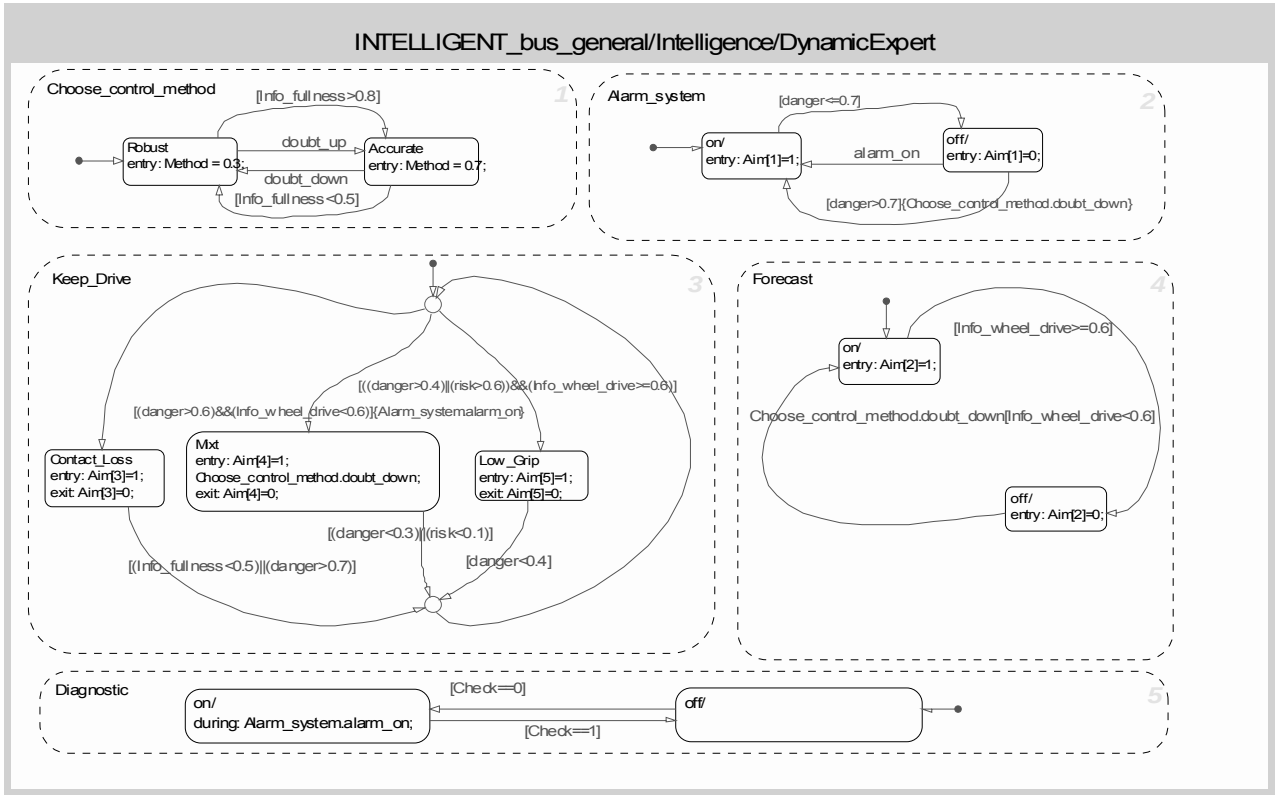

Рис. 3.18. Общий вид модели ДЭС в Stateflow Matlab

В данную структуру добавлены внутренние переменные: включение системы предупреждения водителя (Alarm_on), переход между робастным и детерминированным (в смысле действующих возмущений) методом управления (doubt_up, doubt_down). Эти переменные необходимы для программной реализации решающей функции в ИСУ.

Модель ДЭС реализует систему принятия решений на основе объединения предметно-ориентированных систем повышения безопасности движения КТС. Принятым решением для достижения цели из множества целей являются значения реакций ИСУ в конкретной ситуации на входные воздействия при их отнесении к одному из элементов множества целевых заданий.

4. Методические аспекты применения искусственных нейронных сетей в интеллектуальных системах управления высокой точности и надежности

В последнее время интеллектуальное управление становится широко распространенным средством для многих технических и промышленных приложений [24]. Такие системы управления обладают способностью адаптации к возмущениям, изменениям внешней среды и условиям работы.

В настоящее время исследования в области экспертных систем, традиционно считавшихся основным инструментом интеллектуальных систем, сокращаются, а применение нейросетевых технологий стабильно нарастает.

Искусственные нейронные сети, благодаря своим способностям к самоорганизации и обучению, рассматриваются как перспективные средства для разработки интеллектуальных систем высокой точности и надежности.

Нейросетевые модели эффективно используются в схемах идентификации и управления систем высокой точности и надежности. Актуальными являются разработка и исследование методов и алгоритмов оптимизации параметров нейросетевых моделей и нейроконтроллеров, пригодных для решения задач комплексирования в интеллектуальных системах.

4.1. Синтез систем управления на основе ИНС

Во многих реальных системах имеются нелинейные характеристики, сложные для моделирования динамические элементы, неконтролируемые шумы и помехи, множество обратных связей и другие факторы, затрудняющие реализацию стратегий управления. За последнее время новые стратегии

управления в основном развивались на базе современной и классической теорий управления. Как современная (в частности, технологии адаптивного и оптимального управления), так и классическая теория управления в значительной степени базировались на идее линеаризации систем.

Для практического применения данного подхода необходима, прежде всего, разработка математических моделей. Однако математическое моделирование, реализуемое на основе предположения о линейности системы, может не отражать ее действительных физических свойств. Даже если удается построить сложные математические модели, точно отражающие физические соотношения между входами и выходами системы, они могут оказаться бесполезными для разработки системы управления.

В последнее время теория ИНС быстро развивается и применяется в разных областях. Нейросетевым управлением называется применение ИНС для выработки управляющих сигналов.

4.2. Постановка задачи управления динамическими объектами на основе нейросетевой модели

Синтез системы управления подразумевает создание такого устройства, в результате работы которого система ведет себя «должным образом», т.е. достигается заранее определенная цель управления. Однако на поведение системы в целом (и, следовательно, на определение цели управления) существенное влияние оказывают динамические свойства исполнительных органов и объекта управления, ошибки измерительного оборудования, наличие вычислительных мощностей и т.д.

Традиционно выделяются два основных типа систем управления:

- системы стабилизации, предназначенные для поддержания заданного значения системной переменной (выхода системы) на определенном уровне независимо от действующих на систему возмущений;
- следящие системы (системы серворегулирования), предназначенные для поддержания (выхода) системы на определенной траектории.

На этапе определение метода синтеза системы управления рассматриваются следующие задачи:

- замкнутая система, состоящая из регулятора и объекта регулирования, должна соответствовать заранее определенной модели (например, заданной передаточной функции). Для решения задачи в данной постановке используется ряд методов, таких, как модальное управление и управление по эталонной модели;
- желаемое поведение системы формулируется в виде (квадратичного) критерия, который минимизируется в результате работы регулятора. К методам решения задачи управления в данной постановке относятся, например, управление по минимуму дисперсии, управление с прогнозированием и оптимальное управление.

Стандартным требованием к замкнутым системам является обеспечение устойчивости [24]. К сожалению, анализ устойчивости нелинейных систем чрезвычайно сложен. Это объясняется тем, что нелинейная система, будучи устойчивой в одних режимах, может быть неустойчива в других.

В некоторых работах предлагается использовать нейросетевые методы синтеза адаптивных регуляторов для нестационарных нелинейных систем. Несмотря на теоретическую обоснованность данных методов, их практическое применение ограничено узким классом систем с «медленной» динамикой.

Существуют два принципиально отличных подхода к построению нейросетевых систем управления [24]:

- прямые методы синтеза – регулятор реализуется непосредственно на нейросети. Применение метода не вызывает трудностей, однако необходимость постоянной настройки нейросети приводит к ряду проблем;
- косвенные методы синтеза – нейросеть используется в качестве модели объекта управления, а синтез регулятора осуществляется традиционным методом. В данном случае процедура разработки системы управления включает следующие основные этапы:

1) проведение эксперимента с целью получения множества данных, представляющих объект регулирования во всем рабочем диапазоне;
2) построение нейросетевой модели объекта посредством обучения нейросети на множестве экспериментальных данных;
3) синтез регулятора с использованием полученной нейросетевой модели (возможно, в режиме реального времени).

В данной главе рассматривается первый подход, основанный на применении нейросетевых структур в качестве регулятора.

Основные этапы процесса синтеза систем автоматического управления [24] на основе нейросетевых методов представлены на рис. 4.1.

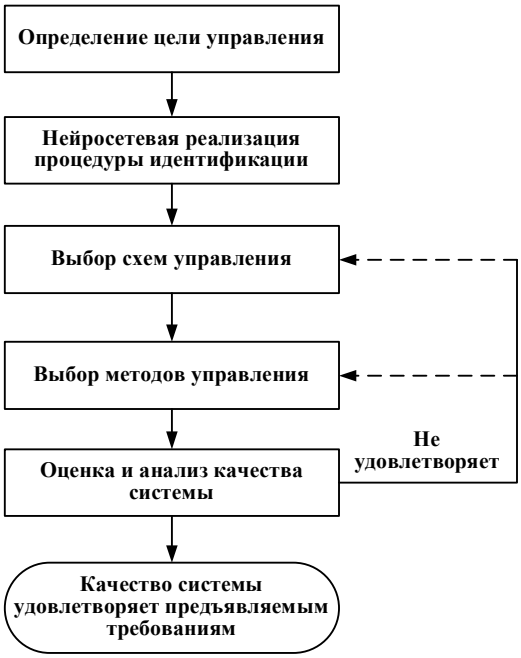

Рис. 4.1. Основные этапы процесса синтеза систем автоматического управления на основе ИНС

В данной главе рассматриваются следующие схемы управления:

- нейросетевое управление на основе инверсной модели объекта,
- нейросетевое оптимальное управление.

Для удобства, запишем уравнение некоторой системы в форме:

$$y(t+1) = g\left(y(t),\ldots,y(t-n+1),\ u(t),\ldots,u(t-m)\right) \qquad (4.1)$$

где u – вектор входов, y – вектор выходов, t – дискретное целочисленное время, n и m – неотрицательные числа, $g(\bullet)$ – некоторая функция.

Для иллюстрации различных концепций построения нейросетевых систем управления выбрана система второго порядка с гладкими нелинейностями.

Методы обучения ИНС при разработке нейросетевых регуляторов

Методы обучения ИНС при решении задач нейросетевого управления могут быть разделены на два вида:

1. Методы обучения в режиме офлайн (offline).

2. Методы обучения в режиме реального времени (онлайн – online).

К методам обучения в режиме реального времени относятся:

- метод обучения ИНС на основе ошибки выхода,
- метод обучения ИНС на основе прогнозируемой ошибки выхода.

По существу эти методы являются аналогами методов, применяемых для настройки коэффициентов ИНС в рамках процедуры нейросетевой идентификации. К ним относятся обобщенный метод обучения инверсной нейросетевой модели. Отличием является специфика выбора регрессионного вектора, выхода нейросетевой модели и критерия оптимизации.

4.3. Алгоритмы обучения ИНС регуляторов в режиме реального времени

Алгоритм обучения нейросетевого управления на основе ошибки выхода. ИНС оптимизируется в соответствии со следующим критерием:

$$J(\boldsymbol{\theta}) = \sum_t \big(r(t) - y(t)\big)^2 + \rho\xi\big(u(t)\big), \qquad (4.2)$$

где $\xi(u(t))$ – функция от управляющего сигнала $u(t)$; $\rho \in [0,1]$ – штрафная константа; *y(t)* – выход управляемого объекта; *r(t)* – эталонный сигнал.

Алгоритм может быть представлен в следующем виде

Шаг 1. Прочитать *y(t).*

Шаг 2. Построить вектор регрессора $\varphi(t, \boldsymbol{\theta})$.

Шаг 3. Обучить сеть по критерию (5.2).

Шаг 4. Генерировать управляющий сигнал $u(t)$.

Шаг 5. Применить $u(t)$ к объекту управления.

Шаг 6. Обработать данные и перейти к шагу 1.

К этому подходу относятся специализированный метод обучения инверсной нейросетевой модели, метод обучения нейросетевой модели для синтеза оптимального управления и т.д. Для специализированного метода обучения инверсной нейросетевой модели значение штрафной константы равно нулю (ρ = 0). В методе обучения нейросетевой модели для синтеза оптимального управления $\xi(u(t)) = \|u(t-1)\|^2$.

Замечание. В этом методе предполагается, что нейросетевая модель объекта получена в результате реализации процедуры идентификации.

Алгоритм обучения нейросетевого управления на основе прогнозируемой ошибки выхода. В этом случае критерий оптимизации сети представлен в следующем виде:

$$J(u(t),t) = \sum_{i=N_1}^{N_2} \left(r(t+i) - \hat{y}(t+i|\theta)\right)^2 + \rho\xi(u(t)), \qquad (4.3)$$

где $\xi(u(t))$ – функция от управляющего сигнала $u(t)$; $\rho \in [0,1]$ – штрафная константа; $\hat{y}(t|\theta)$ – выход нейросетевой прогнозирующей модели; *r(t)* – эталонный сигнал; N_1 и N_2 – горизонт прогнозирования.

Алгоритм реализован следующим образом:

Шаг 1. Прочитать *y(t)*.

Шаг 2. Построить вектор регрессора $\varphi(t)$.

Шаг 3. Найти оптимальный управляющий сигнал, минимизирующий критерий (3).

Шаг 4. Применить оптимальный управляющий сигнал $u(t)$ к объекту управления.

Шаг 5. Обработать данные и перейти к шагу 1.

К этому подходу относятся метод обучения нейросетевой модели для синтеза оптимального управления. В этом случае $\xi(u(t)) = \sum_{i=1}^{N_u} [\Delta u(t+i-1)]^2$, где N_u – горизонт управления; $\Delta u(t+i-1) = u(t+i) - u(t+i-1)$.

Замечание. В этом методе предполагается, что нейросетевой идентификатор получен процедурой, рассмотренной во второй главе.

Синтез нейросетевого управления на основе инверсной модели объекта

В данном случае основным принципом синтеза нейросетевого регулятора является нахождение инверсной нейросетевой модели динамической системы.

Нейроконтроллер на основе инверсной модели имеет вид:

$$\hat{u}(t) = \hat{g}^{-1}\left(y(t+1), y(t), \ldots, y(t-n+1),\ u(t-1), \ldots, u(t-m)\right). \qquad (4.4)$$

Данная модель обучается на множестве примеров (экспериментальных данных), а в режиме непосредственного функционирования использует следующий вектор входов

$$\varphi(t) = \left[r(t+1), y(t), \ldots, y(t-n+1),\ u(t-1), \ldots, u(t-m)\right]^{\mathrm{T}}, \qquad (4.5)$$

где $r(t+1)$ – желаемый входной сигнал системы (уставка) для момента времени $(t+1)$.

Структурная схема нейросетевой системы управления с инверсной моделью представлена на рис. 4.2. Инверсные модели динамической системы могут быть получены на основе обобщенного метода обучения в режиме офлайн (англ. offline) или метода обучения в режиме реального времени (англ. online – онлайн).

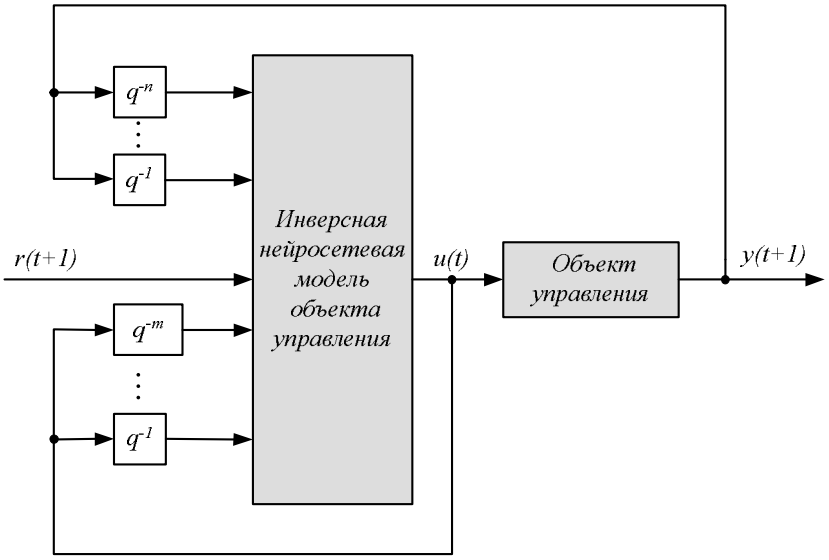

Рис. 4.2. Структурная схема нейросетевой системы управления с инверсной моделью

4.4. Обобщенный метод обучения инверсной нейросетевой модели

Наиболее простой способ построения инверсной нейросетевой модели динамической системы по существу является аналогом процедуры нейросетевой идентификации.

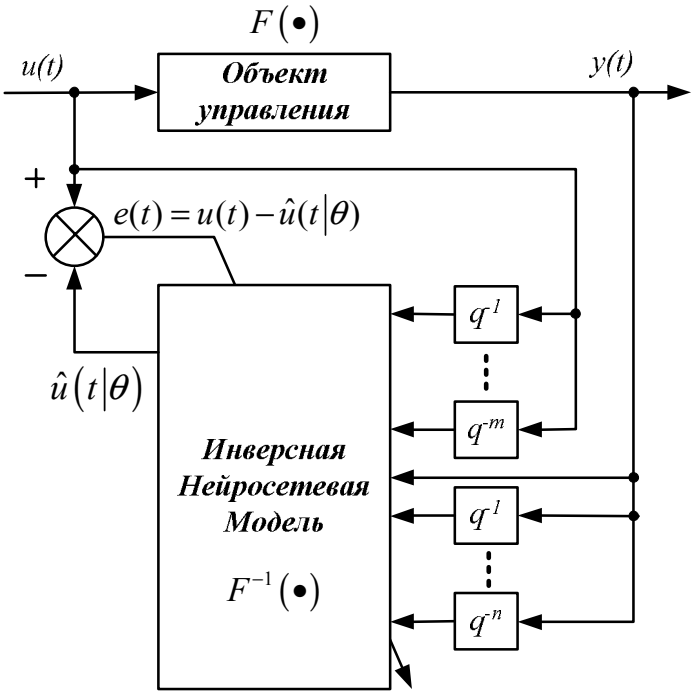

Рис. 4.3. Схема реализации метода обучения инверсной нейросетевой модели объекта

Единственным отличием является специфика выбора регрессионного вектора и выхода нейросетевой модели. В данном случае они определяются соотношениями (4.4) и (4.5). При обучении нейросети минимизируется следующий критерий:

$$J(\theta, Z^N) = \frac{1}{2N}\sum_{t=1}^{N}\left(u(t) - \hat{u}(t|\theta)\right)^2 . \qquad (4.6)$$

В данном случае процесс настройки коэффициентов нейросети производится в режиме офлайн, а процедура настройки весовых коэффициентов нейросети носит название **обобщенного обучения**. Основным достоинством данного метода является независимость от модели реальной динамической системы, т.е. нейросетевой регулятор строится непосредственно на основе экспериментальных данных. На рис. 4.3 представлен принцип обучения инверсной нейросетевой модели управляемого объекта.

Для пояснения принципа работы алгоритма предположим, что известно желаемое значение входа системы (уставка) на последующем шаге (см. рис. 4.2). Передаточная функция замкнутой системы может быть представлена как

$$H(q^{-1}) = q^{-1}. \qquad (4.7)$$

Таким образом, регулятор линеаризует исходную систему, выходной сигнал в точности повторяет уставку, за исключением смещения на один шаг назад.

В случае если временная задержка (запаздывание) в системе превышает единичную, использование принципа управления на основе инверсной модели затрудняется. Передаточная функция замкнутой системы определяется как $H(q) = q^{-d}$, где d – запаздывание.

Предположим, что система может быть представлена уравнением следующего вида:

$$y(t+d) = g\left(y(t+d-1),\ldots,y(t+d-n),\ u(t),\ldots,u(t-m)\right). \qquad (4.8)$$

Нейросетевой регулятор представляет собой инверсную модель системы:

$$\hat{u}(t) = \hat{g}^{-1}\begin{pmatrix} y(t+d), y(t+d-1),\ldots,y(t),\ldots,y(t+d-n), \\ u(t-1),\ldots,u(t-m+1) \end{pmatrix}. \qquad (4.9)$$

По аналогии $y(t+d)$ заменяется на желаемое значение выхода системы в момент времени $(t+d)$. Таким образом, значения необходимо доопределить и подставить в уравнение (4.9). Доопределение значений выходов системы может быть осуществлено с использованием (нейросетевой) прогнозирующей модели.

Другой способ – непосредственное включение прогнозирующей модели в нейросетевой регулятор. Предположим, что запаздывание $d=2$. Тогда необходимо доопределить лишь одно значение $y(t+1)$:

$$y(t+1)=\hat{y}(t+1)=\hat{g}_1(y(t),\ldots,y(t+1-n),\ u(t-1),\ldots,u(t-m-1)). \qquad (4.10)$$

Инверсная модель системы может быть получена посредством объединения (4.9) и (4.10):

$$\hat{u}(t)=\hat{g}^{-1}(y(t+2),y(t),\ldots,y(t-n+2),\ y(t-n+1),\ u(t-1),u(t-m),u(t-m-1)). \qquad (4.11)$$

Аналогично можно получить инверсный нейросетевой регулятор для случая $d>2$.

Несмотря на простоту реализации, практическое применение нейросетевых регуляторов на основе инверсных моделей существенно ограничено. Это связано с тем, что использование обратной связи для достижения «быстрой» реакции системы на изменение входного сигнала (уставки) обычно приводит к значительной чувствительности системы к высокочастотным возмущениям. Кроме того, управляющее воздействие является чрезвычайно активным, т.е. могут возникнуть проблемы с его реализацией исполнительными устройствами.

Другая проблема связана с тем, что соотношение

$$g(y(t),\ldots,y(t-n+1),u_1(t),\ldots,u(t-m))\neq$$
$$\neq g(y(t),\ldots,y(t-n+1),u_2(t),\ldots,u(t-m)). \qquad (4.12)$$

не всегда выполняется для $u_1(t)\neq u_2(t)$, что в большинстве случаев приводит к неадекватной инверсной модели системы.

Синтез нейросетевой системы управления основе обобщенного метода обучения инверсной нейросетевой модели. Для построения

регулятора на основе ИНС использована ANNARX модель объекта управления, полученная на этапе реализации процедуры идентификации [24].

На полученном множеством данных (рис. 4.5) обучена нейросетевая структура, содержащая 5 нейронов с тангенциальными функциями активации в скрытом слое и один линейный нейрон в выходном слое (рис. 4.4). В качестве регулятора использована инверсная нейросетевая модель системы, обученная методом Левенберга-Марквардта [24,26]. Результаты моделирования системы в режиме нормального функционирования представлены на рис. 4.5.

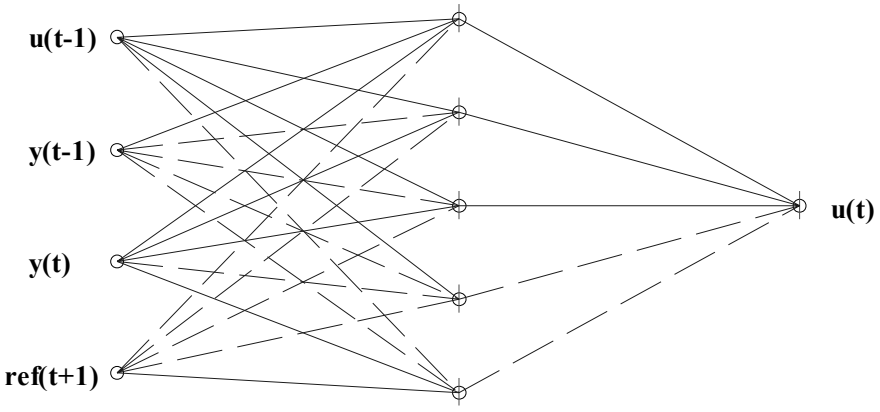

Рис. 4.4. Нейросетевое управление

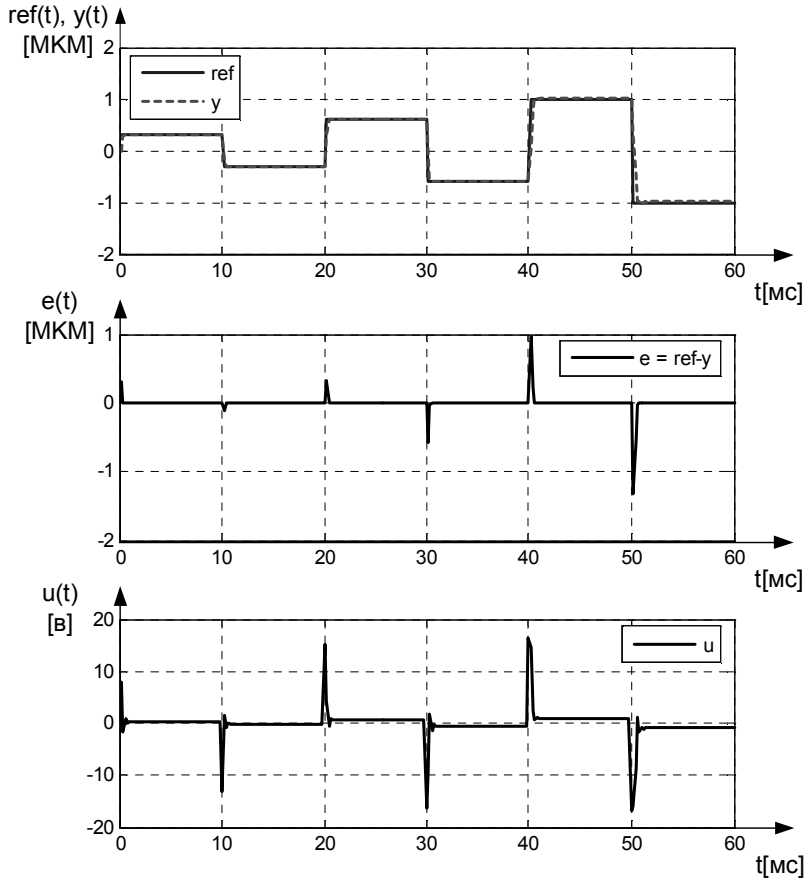

Рис. 4.5. Результаты моделирования нейросетевого регулятора на основе инверсной модели (обобщенный метод обучения): *u(t)* – управляющий сигнал, *y(t)* – выходной сигнал системы, *ref(t)* – уставка, *t*[мс] – время

Полученная система с достаточной точностью отслеживает сигнал уставки во всем рабочем диапазоне, обеспечивая при этом приемлемое время переходных процессов. Однако скорость изменения управляющего воздействия достаточно высока, что в ряде случаев может оказаться неприемлемым.

4.5. Обучение нейросетевой модели в режиме реального времени (специализированный метод обучения)

При использовании обобщенного метода обучения инверсной нейросетевой модели используется критерий (4.6), в соответствии с которым выход нейросети максимально приближается к некоторой последовательности управляющих воздействий. В реальности основной задачей регулирования

является максимальное приближение выхода объекта к желаемому значению, определяемому уставкой. Таким образом, более естественно выглядит критерий типа

$$J(\boldsymbol{\theta}, Z^N) = \frac{1}{2N}\sum_{t=1}^{N}\big(r(t) - y(t)\big)^2. \tag{4.13}$$

Использование критерия (4.13) в обобщенном методе обучения не представляется возможным, так как выход системы $y(t)$ непосредственно зависит от выходного сигнала инверсной модели $u(t-1)$.

Модификация критерия (4.13) и рекуррентная схема обучения ИНС позволяют построить алгоритм обучения инверсной нейросетевой модели в режиме реального времени.

Инверсная нейросетевая модель обучается в соответствии со следующей модификацией критерия типа (4.13):

$$J_t(\boldsymbol{\theta}, Z^t) = J_{t-1}(\boldsymbol{\theta}, Z^{t-1}) + \big(r(t) - y(t)\big)^2. \tag{4.14}$$

Рекуррентный градиентный метод. Рассмотрим рекуррентный градиентный метод обучения нейросетевой модели в соответствии с критерием (4.14). Предположим, что составляющая $J_{t-1}(\boldsymbol{\theta}, Z^{t-1})$ минимизирована, а весовые коэффициенты нейросетевой модели в момент времени t настраиваются в соответствии с выражением

$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) - \eta_t \frac{de^2(t)}{d\boldsymbol{\theta}}, \tag{4.15}$$

где $e(t) = r(t) - y(t)$ и

$$\frac{de^2(t)}{d\boldsymbol{\theta}} = -\frac{dy(t)}{d\boldsymbol{\theta}} e(t). \tag{4.16}$$

Градиент вычисляется по формуле

$$\begin{aligned} \frac{dy(t)}{d\boldsymbol{\theta}} &= \frac{\partial y(t)}{\partial u(t-1)} \frac{du(t-1)}{d\boldsymbol{\theta}} \\ &= \frac{\partial y(t)}{\partial u(t-1)}\left[\frac{\partial u(t-1)}{\partial \boldsymbol{\theta}} + \sum_{i=1}^{n} \frac{\partial u(t-1)}{\partial y(t-i)} \frac{dy(t-i)}{d\boldsymbol{\theta}} + \sum_{i=2}^{m} \frac{\partial u(t-1)}{\partial u(t-i)} \frac{du(t-i)}{d\boldsymbol{\theta}}\right]. \end{aligned} \tag{4.17}$$

В формулу (4.17) входит якобиан системы $\frac{\partial y(t)}{\partial u(t-1)}$, который может быть получен с использованием прогнозирующей модели системы (нейросетевого идентификатора):

$$\frac{\partial y(t)}{\partial u(t-1)} \approx \frac{\partial \hat{y}(t)}{\partial u(t-1)}. \tag{4.18}$$

На практике для простоты реализации метода обучения нейросетевой модели объекта в режиме реального времени коррекция весовых коэффициентов может быть выполнена методом обратного распространения «*виртуальной*» ошибки, $e_u(t)$ (рис. 4.6):

$$e_u(t) = \frac{\partial \hat{y}(t)}{\partial u(t-1)} e(t). \tag{4.19}$$

Обратное распространение «виртуальной» ошибки [24], $e_u(t)$, через нейросетевой регулятор представлено следующим образом (используется двухслойная ИНС с f_j – активационной функцией нейрона скрытого слоя *j*, $F_k(\bullet)$ – активационной функцией выхода *k*):

Градиент для скрытого выходного слоя:

$$\begin{aligned} \mathbf{g}(W_{kj}) &= \frac{de^2(t)}{dW_{kj}} = \sum_{t=1}^{N} h_j(t) F_k' \left(\sum_{j=0}^{n_h} W_{kj} h_j(t) \right) e_u(t) \\ &= \sum_{t=1}^{N} h_j(t) \delta_k^{(W)}(t), \end{aligned} \tag{4.20}$$

где «ошибка» вводится как

$$\delta_k^{(W)}(t) = F_k' \left(\sum_{j=0}^{n_h} W_{kj} h_j(t, \boldsymbol{\theta}) \right) e_u(t). \tag{4.21}$$

Градиент для входного - скрытого слоя определяется как

$$\begin{aligned} \mathbf{g}(w_{jl}) &= \frac{de^2(t)}{dw_{jl}} = \sum_{t=1}^{N} \varphi_l(t) f_j' \left(\sum_{l=0}^{n_\varphi} w_{jl} \varphi_l(t) \right) \sum_{k=1}^{n_y} W_{kj}(t) \delta_k^{(W)}(t) \\ &= \sum_{t=1}^{N} \varphi_l(t) \delta_j^{(W)}(t), \end{aligned} \tag{4.22}$$

где n_y – число выходов,

$$\delta_j^{(w)}(t) = f_j'\left(\sum_{l=0}^{n_\varphi} w_{jl}\varphi_l(t)\right)\sum_{k=1}^{n_y} W_{kj}(t)\delta_k^{(W)}(t)\,. \qquad (4.23)$$

Изменение весовых коэффициентов сети градиентным методом выполняется следующим образом:

- изменение весовых коэффициентов скрытого выходного слоя

$$W_{kj}(t+1) = W_{kj}(t) + \eta(t)\mathbf{g}(W_{kj})\,, \qquad (4.24)$$

- изменение весовых коэффициентов входного скрытого слоя

$$w_{jl}(t+1) = w_{jl}(t) + \eta(t)\mathbf{g}(w_{jl})\,. \qquad (4.25)$$

Процедура реализации специализированного алгоритма обучения инверсной нейросетевой модели в режиме реального времени на основе рекуррентного градиентного метода может быть представлена в следующем виде:

Шаг 1. Прочитать *y(t)*.

Шаг 2. Вычислить ошибку $e(t) = r(t) - y(t)$.

Шаг 3. Вычислить якобиан модели идентификатора $\dfrac{\partial \hat{y}(t)}{\partial u(t-1)}$.

Шаг 4. Вычислить виртуальную ошибку $e_u(t) = \dfrac{\partial \hat{y}(t)}{\partial u(t-1)} e(t)$.

Шаг 5. Вычислить градиенты для слоев по формулам (4.20) – (4.23).

Шаг 6. Модифицировать весовые коэффициенты сети по формулам (4.24) и (4.25).

Шаг 7. Генерировать новый управляющий сигнал и применить его к объекту управления.

Шаг 8. Если условия снова не выполнены, перейти к шагу 1.

Рекуррентный метод Гаусса-Ньютона 1. Этот метод реализован на основе метода псевдолинейной регрессии. Из уравнения (4.17) имеем:

$$\boldsymbol{\psi}(t) = \frac{dy(t)}{d\boldsymbol{\theta}} \approx \frac{\partial \hat{y}(t)}{\partial u(t-1)} \frac{\partial u(t-1)}{\partial \boldsymbol{\theta}} = \frac{\partial \hat{y}(t)}{\partial u(t-1)} \boldsymbol{\phi}(t)\,, \qquad (4.26)$$

где $\phi(t)$ – частная производная сигнала управления по весовым коэффициентам нейросетевого управления. Можно непосредственно применять рекуррентный метод Гаусса-Ньютона.

Рекуррентный метод Гаусса-Ньютона 2. В этом методе используется уравнение (4.17). Ошибка прогнозированная вычисляется по форме:

$$\begin{aligned}\boldsymbol{\psi}(t) &= \frac{\partial y(t)}{\partial u(t-1)}\boldsymbol{\psi}_u(t) = \frac{\partial y(t)}{\partial u(t-1)}\frac{du(t-1)}{d\boldsymbol{\theta}} \\ &\approx \frac{\partial \hat{y}(t)}{\partial u(t-1)}\left[\phi(t) + \sum_{i=1}^{n}\frac{\partial u(t-1)}{\partial y(t-i)}\boldsymbol{\psi}(t-i) + \sum_{i=2}^{m}\frac{\partial u(t-1)}{\partial u(t-i)}\boldsymbol{\psi}_u(t-i)\right].\end{aligned} \tag{4.27}$$

Вычисление якобиана нейросетевой модели идентификатора $\frac{\partial \hat{y}(t)}{\partial u(t-1)}$ и частной производной $\phi(t)$ по весовым коэффициентам нейронной сети рассмотрено в работе [25]. Для вычисления остальных компонент в формуле (4.27), могут быть применены следующие формулы:

$$\sum_{i=1}^{n}\frac{\partial u(t-1)}{\partial y(t-i)} = \sum_{i=2}^{n+1}\frac{\partial u(t-1)}{\partial \varphi(i,t-1)}, \tag{4.28}$$

$$\sum_{i=2}^{m}\frac{\partial u(t-1)}{\partial u(t-i)} = \sum_{i=n+2}^{n+m}\frac{\partial u(t-1)}{\partial \varphi(i,t-1)}, \tag{4.29}$$

где $\varphi(t-1) = [r(t), y(t-1),\ldots, y(t-n),\ u(t-2),\ldots, u(t-m-1)]^{\mathrm{T}}$.

Частная производная $\frac{\partial u(t)}{\partial \varphi(t)}$ – якобиан нейросетевой модели управления.

Алгоритм рекуррентного метода Гаусса-Ньютона может быть представлен следующим образом:

Шаг 1. Прочитать *y(t)*.

Шаг 2. Вычислить ошибку $e(t) = r(t) - y(t)$.

Шаг 3. Вычислить якобиан нейросетевой модели идентификатора $\frac{\partial \hat{y}(t)}{\partial u(t-1)}$.

Шаг 4. Вычислить частную производную $\phi(t)$.

Шаг 5. Вычислить частную производную $\boldsymbol{\psi}(t)$ по формулам (4.26) или (4.27).

Шаг 6. Модифицировать весовые коэффициенты сети по формулам (4.24)

или (4.25).

Шаг 7. Генерировать новый управляющий сигнал и применить его к объекту управления.

Шаг 8. Если условия снова не выполнены, перейти к шагу 1.

4.6. Обучение нейросетевой модели в режиме реального времени с использованием эталонной модели

В ряде случаев синтез системы управления подразумевает получение некоторой заранее определенной (эталонной) характеристики замкнутой системы, например:

$$y_m(t) = \frac{B_m(q^{-1})}{A_m(q^{-1})} r(t) . \qquad (4.30)$$

В подобных случаях в критерии обучения нейросетевой модели может быть использована ошибка типа $e(t) = y_m(t) - y(t)$. Таким образом, прослеживается аналогия с адаптивными системами управления по эталонной модели.

Схема обучения нейросетевой модели объекта в режиме реального времени с использованием эталонной модели представлена на рис. 4.6.

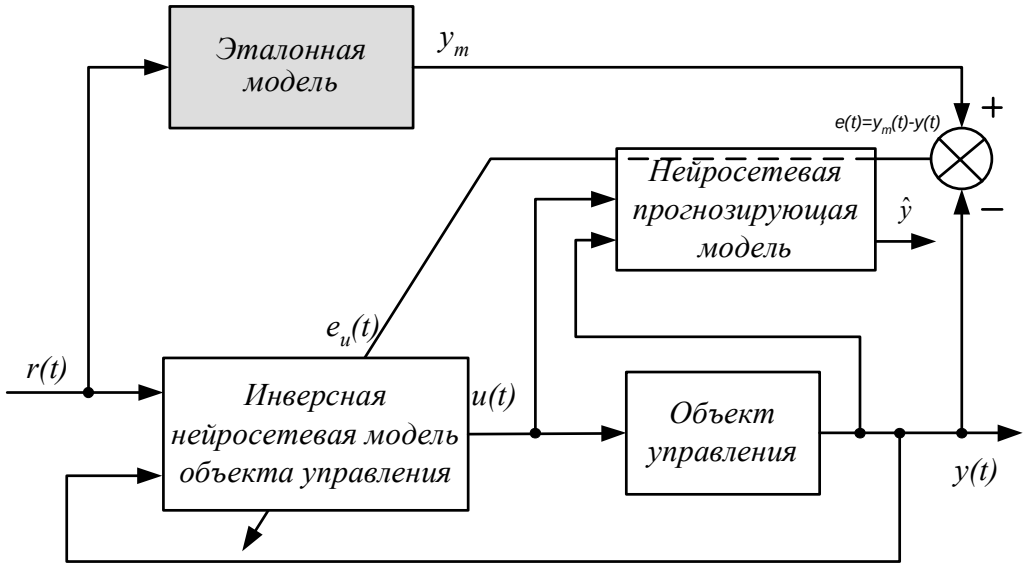

Рис. 4.6. Схема реализации специализированного алгоритма обучения инверсной нейросетевой модели в режиме реального времени

Синтез нейросетевой системы управления на основе метода обучения инверсной нейросетевой модели в режиме реального времени (специализированный метод обучения). На основе структуры инверсной нейросетевой модели (см. рис. 4.4.) мы реализуем процесс обучения с использованием специализированного метода на основе рекуррентного алгоритма Гаусса-Ньютона. Результаты тестирования представлены на рис. 4.7.

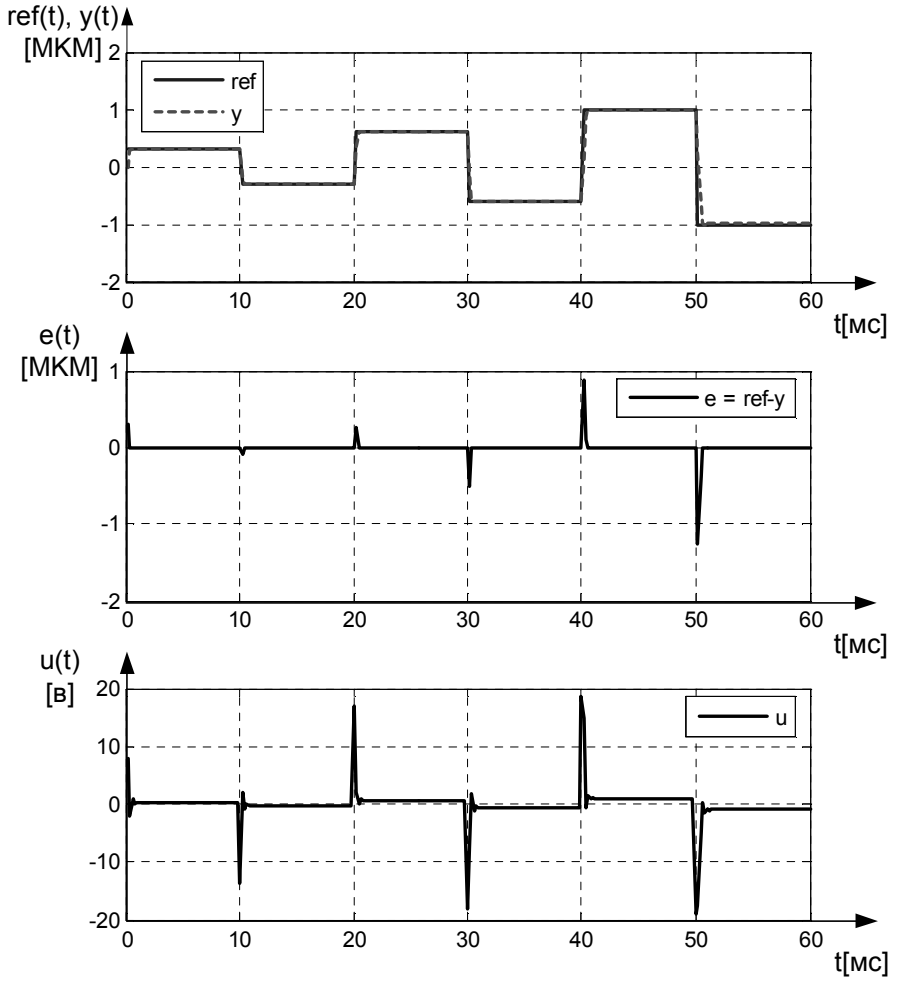

Рис. 4.7. Результаты тестирования нейросетевого регулятора на основе инверсной модели и специального метода, *u(t)* – управляющий сигнал, *y(t)* – выходной сигнал системы, *ref(t)* – уставка, *t*[мс] – время

Полученная система с высокой точностью отслеживает сигнал уставки во всем рабочем диапазоне, время переходных процессов соответствует требованиям, предъявляемым к системе. Скорость изменения управляющего

воздействия достаточно высока, что в ряде случаев может оказаться неприемлемым.

4.7. Методические рекомендации по реализации и применению метода синтеза управления на основе инверсных нейросетевых моделей

Значительное число весовых коэффициентов нейросетевой модели и медленная сходимость градиентного метода оптимизации затрудняют использование принципа регулирования на основе инверсных нейросетевых моделей, обучаемых в режиме реального времени. Вычислительные (временные) затраты на обучение нейросетевой модели могут быть уменьшены путем применения обобщенного метода обучения на предварительном этапе инициализации настраиваемых параметров нейросетевой модели. Кроме того, для увеличения скорости сходимости может быть использован рекуррентный метод Гаусса-Ньютона.

Существенным преимуществом метода обучения в режиме реального времени является отсутствие проблем, связанных с выбором оптимальной структуры нейросетевой модели и «переобучением» нейросети. В большинстве случаев реальная система выполняет ограниченный круг задач. Типичным примером являются робототехнические системы, рабочий орган которых движется по заранее определенной траектории. В таком случае нейросетевой регулятор может настраиваться именно на заданную траекторию.

Другим преимуществом онлайнового подхода является возможность создания регуляторов для нестационарных систем.

Таким образом, обобщенная процедура реализации метода управления на основе инверсных нейросетевых моделей представлена следующим образом:

Шаг 1. Проведение эксперимента (получение множества экспериментальных данных).

Шаг 2. Построение прогнозирующей нейросетевой модели системы посредством реализации процедуры идентификации.

Шаг 3. Инициализация нейросетевой инверсной модели системы обобщенным методом обучения.

Шаг 4. Предварительная настройка параметров инверсной нейросетевой модели методом обучения в режиме реального времени с использованием прогнозирующей модели.

Шаг 5. Окончательная настройка параметров инверсной нейросетевой модели методом обучения в режиме реального времени.

Метод управления на основе инверсных нейросетевых моделей обладает следующими преимуществами и недостатками:

Преимущества:

- интуитивная понятность и простота реализации;
- возможность построения регуляторов для нестационарных систем (в случае использования обучения нейросетевой модели в режиме реального времени).

Недостатки:

- неустойчивость инверсной модели, в большинстве случаев приводящая к неустойчивости замкнутой системы;
- высокая чувствительность к возмущениям (в большинстве случаев).

Нейросетевое оптимальное управление

Основной идеей оптимального управления является синтез такого критерия оптимизации параметров регулятора, в котором помимо точного отслеживания заданной траектории (уставки) предусмотрено «штрафование» по амплитуде управляющего воздействия [24]. Принцип оптимального управления может быть реализован на основе метода обучения ИНС, при этом $\xi(u(t)) = \|u(t-1)\|^2$, $\rho \geq 0$ и критерий (4.3) примет следующий вид:

$$J(\mathbf{\theta}) = \frac{1}{2N}\sum_{t=1}^{N}[r(t) - y(t)]^2 + \rho[u(t-1)]^2,\ \rho \geq 0. \qquad (4.31)$$

При $\rho = 0$ критерий вырождается до вида (4.13) и обученная нейросеть представляет собой инверсную модель системы. При увеличении параметра ρ

регулятор уже не является инверсной моделью системы, но сигнал управления становится более гладким с меньшей амплитудой.

При использовании специализированной схемы обучения регулятор настраивается в режиме реального времени. В этом случае могут быть использованы алгоритм обратного распространения (ошибки) или рекуррентный метод Гаусса-Ньютона, причем использование последнего предпочтительнее по причине высокой скорости сходимости. Однако критерий оптимизации при введении штрафа на управляющее воздействие отличен от критерия наименьших квадратов. Поэтому традиционная схема оптимизации претерпевает некоторые изменения.

Перепишем критерий оптимизации следующим образом:

$$J_t\left(\boldsymbol{\theta}, Z^t\right) = J_{t-1}\left(\boldsymbol{\theta}, Z^{t-1}\right) + \left(\left(r(t) - y(t)\right)^2 + \rho\left(u(t-1)\right)^2\right). \tag{4.32}$$

Предполагается, что составляющая J_{t-1} минимизирована. Для вычисления градиента может быть использовано следующее выражение:

$$\mathbf{g}(\boldsymbol{\theta}) \approx \frac{du(t-1)}{d\boldsymbol{\theta}}\left(-\frac{\partial \hat{y}(t)}{\partial u(t-1)}e(t) + \rho u(t-1)\right). \tag{4.33}$$

Введя обозначения

$$\begin{aligned} e_u(t) &\equiv \frac{\partial \hat{y}(t)}{\partial u(t-1)}e(t), \\ \Psi_u(t) &\equiv \frac{du(t-1)}{d\boldsymbol{\theta}}, \\ \Psi(t) &= \frac{\partial \hat{y}(t)}{\partial u(t-1)}\Psi_u(t), \end{aligned} \tag{4.34}$$

можно представить (4.33) как

$$\mathbf{g}(\boldsymbol{\theta}) \approx \Psi_u(t)\left(-e_u(t) + \rho u(t-1)\right). \tag{4.35}$$

Из (4.34) можем прямо применять рекуррентный метод Гаусса-Ньютона, например:

$$\mathbf{P}(t) = \frac{1}{\lambda}\left(\mathbf{P}(t-1) - \frac{\mathbf{P}(t-1)\boldsymbol{\psi}(t)\boldsymbol{\psi}^{\mathrm{T}}(t)\mathbf{P}(t-1)}{\lambda + \boldsymbol{\psi}^{\mathrm{T}}(t)\mathbf{P}(t-1)\boldsymbol{\psi}(t)}\right), \tag{4.36}$$

$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) + \mathbf{P}(t)\boldsymbol{\psi}_u\left(e_u(t) - \rho u(t-1)\right). \tag{4.37}$$

В некоторых случаях бывает целесообразно оптимизировать регулятор для управления по некоторой заранее заданной траектории. Этого можно достигнуть путем многократного обучения по рассматриваемой траектории. Можно утверждать, что при этом минимизируется критерий

$$J(\theta) = E\left\{\sum_{t=1}^{N}\left[r(t) - y(t)\right]^2 + \rho\left[u(t)\right]^2\right\}, \qquad (4.38)$$

где N – число дискрета на выбранной траектории.

Синтез нейросетевой оптимальной системы управления.

Нейросетевой оптимальной регулятор, содержащий 5 нейронов с тангенциальными функциями активации в скрытом слое и один линейный нейрон в выходном слое (см. рис. 4.4.), обучен методом рекуррентного алгоритма Гаусса-Ньютона в соответствии с критерием (4.31), здесь штрафной коэффициент $\rho = 6e-4$. Результаты тестирования представлены на рис. 4.8.

Полученная система с достаточной точностью отслеживает сигнал уставки во всем рабочем диапазоне, время переходных процессов также удовлетворяет требованиям, предъявляемым к системе. Скорость изменения управляющего воздействия меньше, чем при использовании регулятора на основе инверсной нейросетевой модели, что является положительным фактором с точки зрения надежности системы.

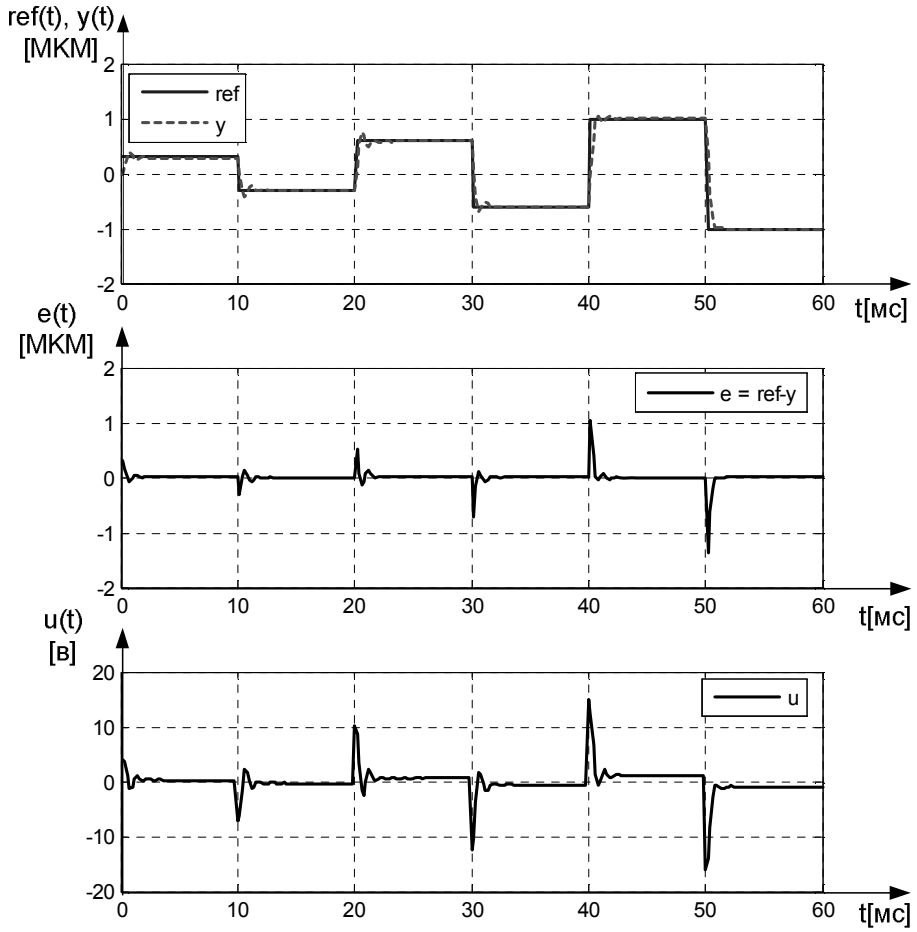

Рис. 4.8. Результаты тестирования нейросетевого оптимального регулятора: *u(t)* – управляющий сигнал, *y(t)* – выходной сигнал системы, *ref(t)* – уставка, *t*[мс] – время

Оптимальное управление на основе нейросетевых моделей обладает следующими преимуществами и недостатками:

Преимущества:

- простота реализации;
- возможность использования для построения систем управления широкого класса систем;
- выработка управляющего сигнала сравнительно небольшой амплитуды с незначительными осцилляциями;
- возможность эффективного использования для создания систем программного управления.

Недостатки:

- необходимость настройки параметров регулятора в режиме реального времени (требует существенных вычислительных затрат);
- необходимость перенастройки параметров при изменении коэффициента штрафа на управление.

4.8. Вычислительные процедуры методов оценки параметров нейросетевых моделей

Якобиан нейросетевой модели

Якобиан, или мгновенная матрица усиления, является производной выхода ИНС по входам для заданной пары «вход-выход».

Для двухслойной ИНС с линейными активационными функциями нейронов выходного слоя и тангенциальными активационными функциями нейронов скрытого слоя частная производная по произвольному входу определяется как

$$\frac{\partial \hat{y}_k\left(t|\boldsymbol{\theta}\right)}{\partial \varphi_l\left(t,\boldsymbol{\theta}\right)} = \sum_{j=1}^{n_h} W_{kj} w_{jl} \left[1 - \tanh^2 \left(\sum_{l=1}^{n_\varphi} w_{ji} \varphi\left(t,\boldsymbol{\theta}\right) + w_{j0} \right) \right] =$$

$$= \sum_{j=1}^{n_h} W_{kj} w_{jl} \left[1 - h_j^2\left(t,\boldsymbol{\theta}\right) \right]. \tag{4.39}$$

Метод обратного распространения ошибки

В случае, когда нейронная сеть содержит более одного скрытого слоя с нелинейными функциями активации, выражения для определения значений $\phi(t)$, соответствующих элементам матрицы частных производных, очевидно, становятся более сложными. Алгоритм определения градиента минимизируемого критерия для сети с произвольным числом скрытых слоев и произвольным видом активационных функций, использующий особенности структуры ИНС, носит название метода обратного распространения (ошибки), или обобщенного дельта-правила. Так как алгоритм рассчитан на обучение ИНС прямого действия, то он может быть непосредственно применен только к

модельным структурам типа ANNARX. Тем не менее метод может быть модифицирован и для получения частных производных $\phi(t)$.

Градиент критерия наименьших квадратов (рассматривается общий случай ИНС с несколькими выходами) может быть представлен следующим образом:

$$\begin{aligned}\mathbf{g}(\boldsymbol{\theta}) = V_N'(\boldsymbol{\theta}, Z^N) &= \frac{1}{N}\sum_{t=1}^{N}\frac{\partial \varepsilon^{\mathrm{T}}(t,\boldsymbol{\theta})}{\partial \varepsilon}\varepsilon(t,\boldsymbol{\theta}) \\ &= -\frac{1}{N}\sum_{t=1}^{N}\frac{\partial \hat{y}^{\mathrm{T}}(t|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}(y(t)-\hat{y}(t|\boldsymbol{\theta})).\end{aligned} \tag{4.40}$$

Прогнозируемое ИНС выходное значение вычисляется в соответствии со следующим выражением (для выхода k):

$$\hat{y}_k(t|\theta) = F_k\left(\sum_{j=0}^{n_h} W_{kj}h_j(t)\right) = F_k\left(\sum_{j=1}^{n_h} W_{kj}f_j\left(\sum_{l=0}^{n_\varphi} w_{jl}\varphi_l(t)\right) + W_{k0}\right), \tag{4.41}$$

где f_j – активационная функция нейрона скрытого слоя j, а $F_k(\bullet)$– активационная функция выхода k. Для простоты нейронные смещения представлены как добавочные весовые коэффициенты, т.е. $h_0(t) = \varphi_0(t) = 1$.

Частные производные выходов ИНС по весовым коэффициентам определяются следующими соотношениями:

$$\phi_{jk}^{(W)} = \frac{\partial \hat{y}_k(t|\boldsymbol{\theta})}{\partial W_{kj}} = h_j(t)F_k'\left(\sum_{j=0}^{n_h} W_{kj}h_j(t)\right), \tag{4.42}$$

$$\phi_{kjl}^{(w)} = \frac{\partial \hat{y}_k(t|\boldsymbol{\theta})}{\partial w_{jl}} = \varphi_l(t)f_j'\left(\sum_{j=0}^{n_\varphi} w_{jl}\varphi_l(t)\right)W_{kj}F_k'\left(\sum_{j=0}^{n_h} W_{kj}h_j(t)\right), \tag{4.43}$$

что приводит к следующему выражению для определения градиента скрытого выходного слоя:

$$\begin{aligned}\mathbf{g}(W_{kj}) &= \sum_{t=1}^{N} h_j(t)F_k'\left(\sum_{j=0}^{n_h} W_{kj}h_j(t)\right)\left(y_k(t)-\hat{y}_k(t|\boldsymbol{\theta})\right) \\ &= \sum_{t=1}^{N} h_j(t)\delta_k^{(W)}(t),\end{aligned} \tag{4.44}$$

где «ошибка» или «дельта» вводится как

$$\delta_k^{(W)}(t) = F_k'\left(\sum_{j=0}^{n_h} W_{kj} h_j(t,\boldsymbol{\theta})\right)\left(y_k(t) - \hat{y}_k(t|\boldsymbol{\theta})\right). \qquad (4.45)$$

Градиент для входного-скрытого слоя определяется как

$$\mathbf{g}(w_{jl}) = \sum_{t=1}^{N} \varphi_l(t) f_j'\left(\sum_{l=0}^{n_\varphi} w_{jl}\varphi_l(t)\right)\sum_{k=1}^{n_y} W_{kj}(t)\delta_k^{(W)}(t) =$$

$$= \sum_{t=1}^{N} \varphi_l(t)\delta_k^{(W)}(t), \qquad (4.46)$$

где

$$\delta_j^{(w)}(t) = f_j'\left(\sum_{l=0}^{n_\varphi} w_{jl}\varphi_l(t)\right)\sum_{k=1}^{n_y} W_{kj}(t)\delta_k^{(W)}(t). \qquad (4.47)$$

и n_φ – число выхода, n_h – число нейронов в скрытом слое, n_y – число выхода ИНС.

Аналогичным образом алгоритм обобщается на случай произвольного числа скрытых слоев ИНС. Процедура заключается в распространении по ИНС значения «дельта» (4.46) слой за слоем в обратном направлении.

При необходимости частные производные $\phi(t)$ могут быть получены из выражений (4.46) и (4.47). Из выражения (4.45) удаляется значение ошибки прогнозирования (невязки), затем отдельно для каждого выхода применяется метод обратного распространения ошибки.

5. Методика синтеза алгоритмов оценивания моделей динамических объектов для организации робастно-адаптивного управления в интеллектуальных системах, основанных на комплексировании принципов робастного, нейро-нечеткого и адаптивного управления

В разделе рассматриваются проблемы идентификации математических моделей нелинейных, неопределенных объектов управления с применением нечетких интеллектуальных технологий.

Возможны два подхода к решению проблемы идентификации. Первый подход основывается на использовании обучающих выборок, составленных из наборов входных-выходных данных. Второй подход основывается на использовании априорной информации об объекте управления в виде их моделей, заданных в аналитической форме.

Основная идея первого подхода состоит в том, чтобы сформировать адекватную модель динамической системы в классе нечетких моделей, используя при этом наборы входных-выходных данных. Таким образом, возникает задача разработки и исследования алгоритмов синтеза нечетких моделей систем на основе обработки полученных в результате эксперимента наборов данных.

Второй поход к решению проблемы идентификации основан на использовании модели объекта в аналитической форме, что позволяет выявить особые свойства этих объектов управления. В работе, в частности, рассматривается класс нелинейных объектов управления с секторальными ограничениями на выходные переменные. Эта особенность динамических свойств рассматриваемого класса нелинейных объектов позволяет решать проблему идентификации в классе нечетких моделей Такаги -Сугено.

Полученные в результате решения задачи идентификации оценки нечетких моделей позволяют синтезировать робастные и робастно-адаптивные алгоритмы управления нелинейными, неопределенными объектами.

5.1. Общая проблема аппроксимации функций на основе использования экспериментальных наборов данных

Рассмотрим проблему приближения или аппроксимации функций, смысл которой состоит в синтезе такой системы функций, которая обеспечивала бы приближение к другой функции на некотором ограниченном наборе входных - выходных данных.

Представим аппроксимируемую функцию в форме некоторого отображения

$$g : \bar{X} \to \bar{Y},$$

где $\bar{X} \in R$ – множество входных значений, а $\bar{Y} \in R$ – множество выходных значений.

Введем в рассмотрение нечеткую систему

$$f : X \to Y,$$

где $X \subset \bar{X}$ и $Y \subset \bar{Y}$ - являются некоторыми подмножествами из всего допустимого диапазона значений входных и выходных переменных функции.

С учетом параметризации функции g и введением дополнительного вектора параметров θ представим функцию g в виде

$$g(x) = f(x, \theta) + e(x) \qquad (5.1)$$

где $x = [x_1, x_2, ..., x_n]^T \in X$ и ошибка приближения $e(x)$ – малая величина.

Обозначим через $x(k)$ – значение входного вектора x в дискретный момент времени k, а через $x_j(k)$ – значение *j*-й компоненты этого входного вектора в дискретный момент времени k.

Предположим, что существует такой вектор параметров θ нечеткой системы $f(x, \theta)$, при котором обеспечивается аппроксимация функции g на ограниченном множестве пар данных входа–выхода. Пусть *i*-я пара данных

входа-выхода для функции g обозначена парой (x^i, y^i), где $x^i \in X$, $y^i \in Y$ и $y^i = g(x^i)$. Вектор $x^i = \left[x_1^i, x_2^i, ..., x_n^i\right]^T$ является *i*-ым вектором входных данных. Тогда обозначим через x_j^i – *j*-ю компоненту *i*-го вектора входных данных. Это множество пар входных-выходных данных называется множеством обучающих данных, определенных в виде

$$G = \left\{(x^1, y^1), (x^2, y^2), ..., (x^M, y^M)\right\} \subset X * Y \qquad (5.2)$$

где M – число пар входных - выходных данных.

Пусть $d^{(i)}$ – обозначает *i*–ю пару входных-выходных данных, т. е. $\mathrm{d}^{(\mathrm{i})} = (x^i, y^i)$.

На рис. 5.1 изображено графическое представление процесса отображения функций.

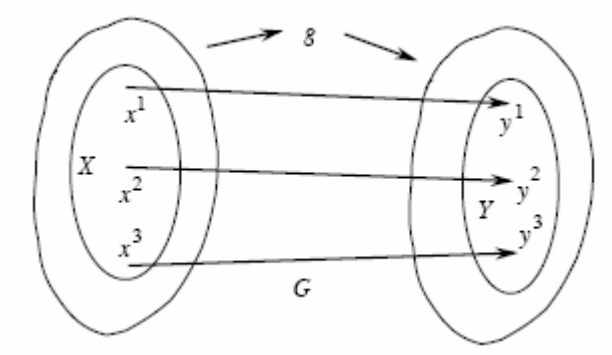

Рис. 5.1. Отображение функций для трех известных пар входных - выходных данных

Необходимо отметить, что при решении задачи аппроксимации функций возникают две следующие сложные проблемы:

- проблема определения такой системы аппроксимирующих функций f, чтобы обеспечивалось хорошее приближение к функции g, при использовании ограниченного, малого набора обучающих входных – выходных данных;
- проблема оценивание качества процесса аппроксимации функции g - системой функций f.

Рассмотрим иллюстративный пример процесса аппроксимации функций. Предположим , что

$$n = 2, X \subset R^2, Y = [0,10] \text{ и } g: X \to Y \text{ где } M = 3.$$

Набор обучающих пар данных :

$$G = \left\{ \left(\begin{bmatrix} 0 \\ 2 \end{bmatrix}, 1 \right), \left(\begin{bmatrix} 2 \\ 4 \end{bmatrix}, 5 \right), \left(\begin{bmatrix} 3 \\ 6 \end{bmatrix}, 6 \right) \right\} \tag{5.3}$$

Геометрический смысл множества пар данных G представлен на рис. 5.2.

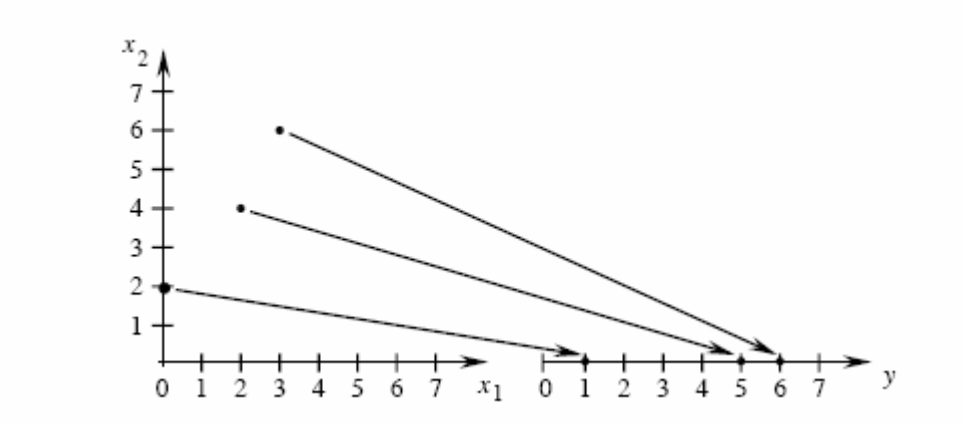

Рис.5.2. Множество обучающих пар данных G, сгенерированных функцией g.

Проблема приближения функций связана как с определением системы функций $f(x,\theta)$, так и с заданием вектора параметров θ так, чтобы погрешность аппроксимации функции g функцией $f(x,\theta)$ была минимальна на множестве G.

Для оценивания степени точности аппроксимации нечеткой системой $f(x,\theta)$ функции $g(x)$ воспользуемся следующим неравенством:

$$\sup_{x \in X} \{ |g(x) - f(x)| \} \tag{5.4}$$

Выражение (5.4) определяет верхнюю границу ошибки приближения. Из этого выражения следует, что для того, чтобы получить оценку точности аппроксимации функции $g(x)$, необходимо иметь полное знание о ней, т. е. знать ее значения для всего диапазона входных данных, в то время как мы располагаем лишь частичным знанием о функции $g(x)$, поскольку располагаем лишь набором данных из заданного органичного множества G.

Поэтому оценку точности приближения функции $g(x)$ нечеткими функциями $f(x,\theta)$ можно осуществлять по их значениям в некоторых дискретных точках $x \in X$, которые выбраны из располагаемого доступного набора данных входа–

выхода. Этот набор данных входа – выхода Γ назовем «тестовым набором» и обозначим

$$\Gamma = \{(x^1, y^1), (x^2, y^2), ..., (x^{M_\Gamma}, y^{M_\Gamma})\} \subset X * Y \qquad (5.5)$$

Здесь M_Γ обозначает число известных тестовых пар данных входа–выхода.

Заметим, что оценка ошибки аппроксимации функции $g(x)$ – нечеткой системой функций $f(x,\theta)$ может быть получена как на основе использования данных из тестового множества Γ, так и на основе сравнения значений $g(x)$ и $f(x,\theta)$ для всех $x \in X$. Однако при использовании известной информации из ограниченного набора можно получить лишь единственную оценку.

Таким образом, значение ошибки аппроксимации функции $g(x)$ нечеткой системой функций $f(x,\theta)$ для каждого момента времени определим в виде

$$\sum_{x^i, y^i \in \Gamma} (g(x^i) - f(x^i, \theta))^2$$

или

$$\sup_{(x^i, y^i) \in \Gamma} \{\left| g(x^i) - f(x^i, \theta) \right|\}. \qquad (5.6)$$

Тип аппроксимирующей функции и ее структура могут существенно влиять на точность аппроксимации.

В частности, при решении ряда прикладных задач, основанных на использовании процедур аппроксимации функций, более эффективными, с точки зрения точности аппроксимации, являются функциональные нечеткие системы Такаги – Сугено, которые обеспечивают лучшую аппроксимацию, по сравнению с нечеткими системами Мамдани.

Пусть система функций $f(x,\theta)$ интерпретируется как некоторый аппроксиматор, который параметризуется вектором θ. Выбор параметрического вектора θ зависит как от числа используемых функций принадлежности, так и от числа используемых правил нечеткого логического вывода.

5.2. Идентификация нелинейных объектов управления с использованием нечетких моделей

Решение проблемы идентификации и управления сложными нелинейными объектами с использованием интеллектуальных нечетких технологий [27–29] является более эффективным по сравнению с методами идентификации и управления, основанными на использовании традиционных, классических моделей этих объектов в аналитической форме как вследствие сложности получения этих моделей, так и вследствие сложности их использования для решения задач синтеза управления.

На рис. 5.3 представлена блок-схема, отражающая структуру процедуры синтеза нечетких регуляторов на основе использования нечетких моделей объекта управления. Из этого рисунка следует, что при решении задачи синтеза регуляторов для нелинейных объектов управления с нелинейной динамикой предварительно необходимо идентифицировать его нечеткую нелинейную модель.

Таким образом, конструирование нечеткой модели объекта управления является важной основной компонентой этой схемы.

Данная схема синтеза управления предполагает два возможных подхода к построению нечетких моделей:

- идентификация нечетких моделей на основе использования наборов входных - выходных данных;
- конструирование нечетких моделей на основе использования полученных из физических законов аналитических зависимостей, система которых представляет формальную модель нелинейного объекта управления.

В представленном выше материале рассматривались общие проблемы аппроксимации функций, в том числе проблема использования в качестве аппроксимирующих систем нечетких функций. При этом идея аппроксимации функций иллюстрировалась на примерах, отражающих первый подход к проблеме идентификации моделей нелинейных объектов в классе нечетких моделей.

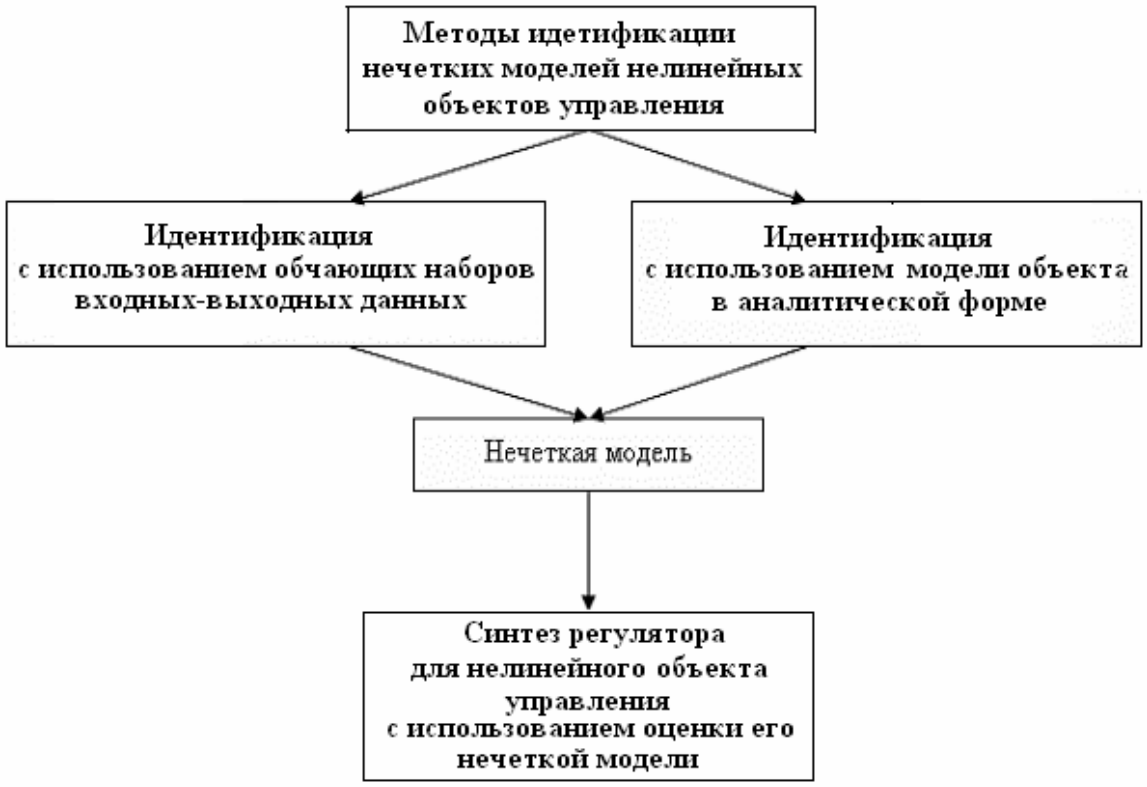

Рис. 5.3. Синтез регуляторов с использованием оценки нечеткой модели

Процедуры, реализующие как первый, так и второй подходы, предполагают выполнение двух следующих основных этапов:

- -оценивание или задание структуры нечеткой модели идентифицируемого нелинейного объекта;
- -решение задачи параметрической идентификации нечеткой модели нелинейного объекта.

Однако, в зависимости от используемого подхода, структуры моделей нелинейных объектов управления, а также алгоритмы параметрического оценивания этих моделей являются различными.

В этой связи в последующем материале продолжим рассмотрение проблемы идентификации нелинейных динамических систем в классе нечетких моделей на основе использования как первого, так и второго подходов более подробно.

Использование наборов входных-выходных данных для идентификации объектов управления

Идентификация динамических систем представляет собой процесс получения оценок математических моделей динамических систем на основе использования и обработки экспериментальных данных, полученных в ходе исследования этой системы.

Предположим, что $g(x)$ представляет модель некоторой физической системы, которую необходимо идентифицировать. При этом обучающее множество G определяется экспериментальными наборами входных–выходных данных.

Постановку задачи проиллюстрируем на примере. Пусть задана структура модели некоторой динамической системы. При этом структура модели соответствует линейной модели физической системы и может быть представлена в виде

$$y(k) = \sum_{i=1}^{\bar{q}} \theta_{a_i} y(k-i) + \sum_{i=0}^{\bar{p}} \theta_{b_i} u(k-i), \qquad (5.7)$$

где $u(k)$ и $y(k)-$ значения входных и выходных переменных объекта управления в дискретные моменты времени $k \geq 0$.

Пусть известна полученная экспериментальным путем обучающая выборка, состоящая из наборов пар входных-выходных данных [u_i, y_i.]

Таким образом, проблема параметрического оценивания модели объекта управления состоит в нахождении наилучшей, в определенном смысле, оценки вектора параметров θ модели системы с заданной структурой (5.7).

Для решения этой проблемы можно использовать различные вычислительные алгоритмы и процедуры. Рассмотрим эту проблему более подробно.

Алгоритмы параметрического оценивания на основе реализации не рекурсивных процедур метода наименьших квадратов

Рассмотрим один из подходов к решению задач параметрического

оценивания моделей на основе использования не рекурсивных процедур метода наименьших квадратов для обучения нечетких идентификационных моделей.

Пусть $g(x)$ обозначает модель физической системы, которую нужно идентифицировать. Тогда обучающее множество G определяется располагаемым набором экспериментальных данных входа-выхода, которые сгенерированы этой системой.

Пусть структура идентифицируемой модели представлена в виде (5.7).

Введем в рассмотрение аппроксимирующую функцию общего вида $f(x,\theta)$. Ее структуру зададим в форме

$$f(x,\theta) = \theta^T x(k) \tag{5.8}$$

где

$x(k) = [y(k-1),..,y(k-\bar{q}),u(k-1),..,u(k-\bar{p})]^T$ – вектор регрессии размерности $N \times 1$, $N = \bar{p} + \bar{q} + 1$;

и

$\theta = [\theta_{a1},..,\theta_{a\bar{q}},\theta_{b0},...,\theta_{b\bar{p}}]$ – вектор параметров аппроксимирующей функции размерности $N \times 1$, $N = \bar{p} + \bar{q} + 1$.

Таким образом, процесс идентификации будет состоять в определении, настройке компонент вектора θ с использованием информации от G для того, чтобы $f(x,\theta) \approx g(x)$ для всех $x \in X$.

Для реализации этой процедуры необходимо задать начальные приближения. Для этого выделим из множества G набор выходных данных в виде

$$Y(M) = [y^1, y^2, ..., y^M]^T$$

Здесь $Y(M)$ – вектор выходных данных размерности $M \times 1$, где $y^i \in G$, $i = 1, 2, \ldots, M$.

Обозначим

$$\Phi(M) = \begin{bmatrix} (x^1)^T \\ (x^2)^T \\ \cdot \\ \cdot \\ \cdot \\ (x^M)^T \end{bmatrix},$$

где $\Phi(M)$ – матрица размерности $M \times N$, которая состоит из векторов входных данных $x^i \in G$, $i = 1, 2, \ldots, M$.

Определим ошибку приближения в виде

$$e_i = y^i - (x^i)^T \theta,$$

где e_i - ошибка приближения для i-й пары данных $(x^i, y^i) \in G$.

Определим вектор ошибок аппроксимации в виде

$$E(M) = [e_1, e_2, ..., e_M]^T$$

такой, что

$$E = Y - \Phi\theta$$

Выберем функцию Ляпунова в форме

$$V(\theta) = \frac{1}{2} E^T E \tag{5.9}$$

Полученное выражение позволяет использовать его для оценки степени эффективности приближения на всем наборе данных для заданного θ.

Определим вектор θ такой, чтобы минимизировать $V(\theta)$. Поскольку $V(\theta)$ – выпуклое множество по θ, то локальный минимум является и глобальным минимумом.

Из (5.9) имеем

$$2V = E^T E = Y^T Y - Y^T \Phi\theta - \theta^T \Phi^T Y + \theta^T \Phi^T \Phi\theta$$

Предположим, что матрица $\Phi^T\Phi$ имеет обратную матрицу. Тогда функция $V(\theta)$ может быть представлена в виде

$$2V = Y^T Y - Y^T \Phi\theta - \theta^T \Phi^T Y + \theta^T \Phi^T \Phi\theta + Y^T \Phi(\Phi^T\Phi)^{-1}\Phi^T Y - Y^T \Phi(\Phi^T\Phi)^{-1}\Phi^T Y$$

или

$$2V = Y^T (I - \Phi(\Phi^T\Phi)^{-1}\Phi^T)Y - (\theta - (\Phi^T\Phi)^{-1}\Phi^T Y)^T \Phi^T\Phi(\theta - (\Phi^T\Phi)^{-1}\Phi^T Y) \tag{5.10}$$

Можно видеть, что в уравнении (5.10) первый член не зависит от θ. Поэтому условия минимума функции $V(\theta)$ определяются только другими членами выражения. Следовательно, если выбрать θ так, чтобы второй член в выражении (5.10) был нулевым, то получим условие для определения наименьшего значения функции Ляпунова $V(\theta)$.

Определим значение $\hat{\theta}$, при котором функция $V(\theta)$ получает наименьшее значение, и представим его в виде

$$\hat{\theta} = (\Phi^T \Phi)^{-1} \Phi^T Y . \qquad (5.11)$$

Из уравнения (5.10) следует, что если $\hat{\theta}$ определяется из выражения (5.11), то второй член в выражении (5.10) становится равным нулю. Кроме того, очевидно, что можно непосредственно вычислить наилучшую оценку вектора параметров $\hat{\theta}$ с использованием процедуры метода наименьших квадратов, путем обработки наборов данных Ф и Y.

Если обучающий набор данных получен от нелинейного динамического объекта с известными $\bar{q}$ и $\bar{p}$, то для достижения высокой точности аппроксимации необходим набор данных достаточно большой размерности M. Приведем простой пример для иллюстрации этого алгоритма.

Предположим, что структура модели идентификации представляется в виде

$$y = x_1\theta_1 + x_2\theta_2$$

Обучающий набор данных для решения задачи аппроксимации функции $g(x)$

$$\left\{\left(\begin{bmatrix}1\\1\end{bmatrix},1\right),\left(\begin{bmatrix}2\\1\end{bmatrix},1\right),\left(\begin{bmatrix}3\\1\end{bmatrix},3\right)\right\}. \qquad (5.12)$$

Используем равенство (3.11), где $\Phi = \begin{bmatrix}1 & 1\\ 2 & 1\\ 3 & 1\end{bmatrix}$ и $Y = \begin{bmatrix}1\\1\\3\end{bmatrix}$.

Получим значение оценки $\hat{\theta}$ по формуле

$$\hat{\theta} = \left(\Phi^T\Phi\right)^{-1}\Phi^T Y = \left(\begin{bmatrix}14 & 6\\ 6 & 3\end{bmatrix}\right)^{-1}\begin{bmatrix}12\\5\end{bmatrix} = \begin{bmatrix}1\\ -\frac{1}{3}\end{bmatrix}.$$

Отсюда следует искомое решение.

Модель идентификации может быть описана в виде

$$y = x_1 - \frac{1}{3}x_2 .$$

Выражение (5.12) отражает наилучший результат аппроксимации функции с использованием набора данных и его обработкой с использованием не рекурсивной процедуры метода наименьших квадратов.

Не рекурсивный алгоритм метода наименьших квадратов успешно применяется для решения широкого класса прикладных задач, связанных с решением задач идентификации моделей динамических систем. Однако этот метод имеет ряд недостатков, один из которых состоит в том, что для его реализации необходимо выполнение условия существования обратной, инверсной матрицы $\left(\Phi^T\Phi\right)^{-1}$.

Кроме того, возникают чисто вычислительные проблемы, связанные с вычислением этой обратной матрицы. Поэтому рассмотрим, также, другой подход, который свободен от указанных выше недостатков. Этот подход основан на реализации рекурсивных процедур метода наименьших квадратов.

Алгоритмы параметрического оценивания на основе рекурсивных процедур метода наименьших квадратов

Рекурсивный метод наименьших квадратов позволяет избежать трудностей, связанных с вычислением обратной матрицы $\left(\Phi^T\Phi\right)^{-1}$. Обратим внимание на то, что размерность G увеличивается с каждым дискретным шагом времени.

Поскольку вычислительный алгоритм является рекурсивным, то номер шага k определяет длину выборки M. Поэтому можно обозначить $k = M$.

Тогда выражение для обратной матрицы $\left(\Phi^T\Phi\right)^{-1}$ можно представить в виде

$$P(k) = (\Phi^T\Phi)^{-1} = \left(\sum_{i=1}^{k} x^i (x^i)^T\right)^{-1}. \qquad (5.13)$$

Если $\Phi^T\Phi$ является не особой для всех k, то получим

$$P^{-1}(k) = (\Phi^T\Phi) = \sum_{i=1}^{k} x^i (x^i)^T.$$

или

$$P^{-1}(k) = \sum_{i=1}^{k-1} x^i (x^i)^T + x^k (x^k)^T .$$

Отсюда

$$P^{-1}(k) = P^{-1}(k-1) + x^k (x^k)^T . \tag{5.14}$$

Вектор $\hat{\theta}(k-1)$ является среднеквадратической оценкой, полученной из обработки k-1 пар данных.

Из (5.11) имеем

$$\hat{\theta}(k) = (\Phi^T \Phi)^{-1} \Phi^T Y = \left(\sum_{i=1}^{k} x^i (x^i)^T \right)^{-1} \left(\sum_{i=1}^{k} x^i y^i \right) = P(k) \left(\sum_{i=1}^{k} x^i y^i \right)$$

$$= P(k) \left(\sum_{i=1}^{k-1} x^i y^i + x^k y^k \right). \tag{5.15}$$

Откуда следует

$$\hat{\theta}(k-1) = P(k-1) \sum_{i=1}^{k-1} x^i y^i$$

и

$$P^{-1}(k-1) \hat{\theta}(k-1) = \sum_{i=1}^{k-1} x^i y^i . \tag{5.16}$$

Производя замену выражения для $P^{-1}(k-1)$ из (5.14) на выражение из уравнения (5.16), получим

$$(P^{-1}(k) - x^k (x^k)^T) \hat{\theta}(k-1) = \sum_{i=1}^{k-1} x^i y^i .$$

С учетом уравнения (5.15), имеем

$$\hat{\theta}(k) = P(k)(P^{-1}(k) - x^k (x^k)^T) \hat{\theta}(k-1) + P(k) x^k y^k$$

$$= \hat{\theta}(k-1) - P(k) x^k (x^k)^T) \hat{\theta}(k-1) + P(k) x^k y^k$$

$$= \hat{\theta}(k-1) + P(k) x^k (y^k - (x^k)^T) \hat{\theta}(k-1)) . \tag{5.17}$$

Получим выражение для оценки параметров $\hat{\theta}(k)$ на шаге k, как функцию оценки $\hat{\theta}(k-1)$, полученной на предыдущем шаге, а также как функцию последней пары данных (x^k, y^k).

Заметим, что $(y^k - (x^k)^T \hat{\theta}(k-1))$ является ошибкой прогноза значения $\boldsymbol{y^k}$ при использовании $\hat{\theta}(k-1)$.

Для вычисления нового значения вектора оценки $\hat{\theta}(k)$ в уравнении (5.17) нужно знать матрицу $P(k)$. Для вычисления матрицы $P(k)$ можно использовать следующий рекуррентный алгоритм

$$P(k) = (\Phi(k)^T \Phi(k))^{-1}$$

$$= (\Phi^T(k-1)\Phi(k-1) + x^k (x^k)^T)^{-1}$$

$$= (P^{-1}(k-1) + x^k (x^k)^T)^{-1},$$

откуда

$$P^{-1}(k) = P^{-1}(k-1) + x^k (x^k)^T.$$

Как видно, в этом случае необходимо вычислять обратную матрицу на каждом шаге. Очевидно, что это нежелательно при реализации алгоритма в реальном времени. Поэтому воспользуемся свойством следующего преобразования

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}.$$

Используем указанное выше свойство.

Здесь $A = P^{-1}(k-1)$, $B = x^k$, $C = I$, и $D = (x^k)^T$, тогда получим матрицу $P(k)$ в виде

$$P(k) = P(k-1) - P(k-1)x^k (I + (x^k)^T P(k-1)x^k)^{-1} (x^k)^T P(k-1). \qquad (5.18)$$

Тогда выражение для оценки $\hat{\theta}(k)$

$$\hat{\theta}(k) = \hat{\theta}(k-1) + P(k)x^k (y^k - (x^k)^T \hat{\theta}(k-1)). \qquad (5.19)$$

Уравнения (5.18) и (5.19) представляют алгоритмы вычислительной процедуры рекурсивного метода наименьших квадратов (РМНК).

Для инициализации РМНК-алгоритма необходимо задать начальные приближения для вектора оценки $\hat{\theta}(0)$ и матрицы $P(0)$. На практике обычно выбирают в качестве начального приближения значения $\hat{\theta}(0) = 0$ и $P(0) = P_0$, где $P_0 = \alpha I$. Здесь I – единичная матрица, α – большая положительная константа.

Алгоритмы параметрического оценивания нечетких моделей Мамдани

Нечеткая модель Мамдани может быть задана в виде

$$y = f(x,\theta) = \frac{\sum_{i=1}^{R} b_i \mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)}, \tag{5.20}$$

где $\mu_i(x)$ – функции принадлежности *i-го* правила;

R – число правил вывода и $b_i, i = 1,2,...,R$ – координаты центров функций принадлежности выходных переменных.

Представим нечеткую модель системы в виде

$$f(x,\theta) = \frac{\sum_{i=1}^{R} b_1 \mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} + \frac{\sum_{i=1}^{R} b_2 \mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} + ... + \frac{\sum_{i=1}^{R} b_R \mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)}. \tag{5.21}$$

Обозначим

$$\xi_i = \frac{\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)}.$$

Тогда

$$f(x,\theta) = b_1\xi_1 + b_2\xi_2 + ... + b_R\xi_R.$$

Определим векторы

$$\xi(x) = [\xi_1, \xi_2, ..., \xi_R]^T \qquad \text{и} \qquad \theta = [b_1, b_2, ..., b_R]^T.$$

Тогда

$$y = f(x,\theta) = \theta^T \xi(x). \tag{5.22}$$

где вектор $\xi(x)$ может рассматриваться как регрессионный вектор x в алгоритме наименьших квадратов. Тогда обучающие данные x^i соответствуют значениям обучающей выборки $\xi(x^i)$, а использование алгоритмов наименьших квадратов позволяет найти наилучшие оценки значений центров функций принадлежности выходных переменных b_i.

Для этого следует подставить в формулу для Ф вместо x^i вектор $\xi(x^i)$.

Реализация процедур метода наименьших квадратов позволяет решать задачу параметрического оценивания как в оперативном, так и в неоперативном режимах.

Алгоритмы параметрического оценивания нечетких моделей Сугено

Нечеткие модели Сугено описываются в виде

$$y = f(x,\theta) = \frac{\sum_{i=1}^{R} g_i(x)\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)}$$

где $g(x)$ – весовые функции; $g_i(x) = a_{i,0} + a_{i,1}x + ... + a_{i,n}x$.

Получим

$$y = \frac{\sum_{i=1}^{R} a_{i,0}\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} + \frac{\sum_{i=1}^{R} a_{i,1}x_1\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} + ... + \frac{\sum_{i=1}^{R} a_{i,n}x_n\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)}. \quad (5.23)$$

Обозначим

$$\xi(x) = [\xi_1(x), \xi_2(x), ..., \xi_R(x), x_1\xi_1(x), x_1\xi_2(x), ..., x_1\xi_R(x), ..., x_n\xi_1(x), x_n\xi_2(x)..., x_n\xi_R(x)]^T$$

и $\theta = [a_{1,0}, a_{2,0}, ..., a_{R,0}, a_{1,1}, a_{2,1}, ..., a_{R,1}, ..., a_{1,n}, a_{2,n}, ..., a_{R,n}]^T$.

Тогда

$$f(x,\theta) = \theta^T \xi(x).$$

Используем метод наименьших квадратов для параметрической настройки системы аппроксимирующих функций $f(x,\theta)$. Заменим в формуле для Ф компоненты x^i на компоненты вектора $\xi(x^i)$. Для иллюстрации процедуры обучения нечетких систем с использованием алгоритмов наименьших квадратов рассмотрим следующий пример.

Используем для обучения множество данных G (5.3) для параметрической настройки нечетких систем $f(x,\theta)$. Выберем $R = 2$ и $n = 2$. Тогда имеем $\theta = [b_1, b_2]^T$ и

$$\xi_i = \frac{\prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{R}\prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)}. \qquad (5.24)$$

С учетом (5.3) выберем функции принадлежности входных переменных со следующими параметрами:

- координаты центров функций принадлежности c_j^i:

$c_1^1 = x_1^1 = 0$, $c_2^1 = x_2^1 = 2$, $c_1^2 = x_1^2 = 2$, $c_2^2 = x_2^2 = 4$;

- основание функций принадлежности $\sigma_j^i = 2$, для всех $i = 1,2, j = 1,2$.

Заменим в формуле для Φ компоненты x^i на компоненты ξ_i.

Тогда для $\xi = [\xi_1, \xi_2]$ и $M = 3$ получим:

$$\Phi = \begin{bmatrix} \xi^T(x^1) \\ \xi^T(x^2) \\ \xi^T(x^3) \end{bmatrix} = \begin{bmatrix} 0.8634 & 0.1366 \\ 0.5234 & 0.4766 \\ 0.2173 & 0.7827 \end{bmatrix}.$$

Откуда $Y = \left[y^1, y^2, y^3\right] = [1,5,6]$.

Использование не рекурсивной процедуры алгоритма наименьших квадратов позволяет получить

$$\hat{\theta} = [b_1, b_2]^T = [0.3646, 8.1779]^T.$$

Проверим нечеткую систему на использовавшейся обучающей выборке. Получим следующие значения аппроксимирующих функций в точках $\{x^1, x^2, x^3\}$:

$$f(x^1, \hat{\theta}) = 1.432;$$
$$f(x^2, \hat{\theta}) = 4.0883;$$
$$f(x^3, \hat{\theta}) = 6.4798.$$

Видно, что значение ошибки приближения является большим, потому что ошибка приближения зависит от выбора параметров функции принадлежности и числа используемых функций принадлежности.

Рассмотрим другой метод, который относится к группе градиентных методов. Эти градиентные методы обладают тем достоинством, что они не

зависят от выбора тех или иных параметров функций принадлежности, которые должны настраиваться в процессе обучения нечетких систем.

5.3. Использование градиентных методов для параметрического оценивания нечетких моделей динамических систем

Рассмотрим проблемы, связанные с применением градиентных алгоритмов для решения задачи параметрического оценивания нечетких моделей, цель которой состоит в определении таких значений вектора параметров θ нечетких моделей, для которых аппроксимирующая нечеткая система обеспечивает наилучшее приближение, т. е. ошибка приближения между $f(x,\theta)$ и $g(x)$ принимает наименьшее значение.

В отличие от метода наименьших квадратов, градиентные алгоритмы обеспечивают настройку всех параметров нечеткой системы. В частности, помимо центров функций принадлежности выходных переменных c_j^i, как это имело место при использовании метода наименьших квадратов, градиентные методы обеспечивают настройку центров функций принадлежности выходных переменных, а также показателей нечеткости функций принадлежности входных переменных.

Градиентные алгоритмы параметрического оценивания нечетких моделей Мамдани

В нечетких моделях Мамдани в качестве функций принадлежности могут использоваться функции гауссовского типа с параметрами:

- c_j^i – центры функций принадлежности;
- σ_j^i – показатели нечеткости типа входных переменных;
- b_i – центры функций принадлежности выходных переменных.

Кроме того, в качестве условий минимума нечетких переменных (нечеткого «И») используются произведения их нечетких значений, правила вывода, а в

качестве метода дефаззификации – метод «центра среднего». Нечеткая система может быть задана в виде

$$y = f(x,\theta) = \frac{\sum_{i=1}^{R} b_i \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{R} \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)}.$$

Предположим, что $(x^m, y^m) \in G$ – $m^{ая}$ пара из обучающего набора данных. Ошибку приближения представим в виде

$$e_m = \frac{1}{2}\left[f(x^m,\theta) - y^m \right]^2 \qquad (5.25)$$

Использование градиентного метода должно позволить найти такой вектор параметров θ, который бы обеспечил минимум ошибки e_m. При этом параметрами нечетких систем являются b_i, c_j^i, и σ_j^i, $i = 1,2,...,R$, $j = 1,2,...,n$.

Обозначим $\theta(k)$ значения вектора параметров в момент времени k.

Рассмотрим закон обновления центров функций принадлежности выходных переменных b_i. Для достижения минимума ошибки приближения e_m используем закон обновления параметров b_i в виде

$$b_i(k+1) = b_i(k) - \lambda_1 \frac{\partial e_m}{\partial b_i}\Big|_k ,$$

где $i = 1,2,...,R$ и $k \geq 0$ – индекс шага обновления параметров. Параметр λ_1 характеризует скорость изменения (сходимости) настроек параметров, т. е. если параметр λ_1 выбран малым, то скорость корректировки параметров b_i - мала, а в противном случае, то есть если параметр λ_1 выбран большим, то скорость настройки параметров b_i является высокой.

Из (5.25) найдем выражение для частной производной от ошибки аппроксимации $\frac{\partial e_m}{\partial b_i}$ в виде

$$\frac{\partial e_m}{\partial b_i} = \left(f(x^m,\theta) - y^m \right) \frac{\partial f(x^m,\theta)}{\partial b_i}$$

или

$$\frac{\partial e_m}{\partial b_i} = \left(f(x^m, \theta) - y^m \right) \frac{\prod_{j=1}^{n} \exp\left(-\frac{1}{2} \left(\frac{x_j^m - c_j^i}{\sigma_j^i} \right)^2 \right)}{\sum_{i=1}^{R} \prod_{j=1}^{n} \exp\left(-\frac{1}{2} \left(\frac{x_j^m - c_j^i}{\sigma_j^i} \right)^2 \right)}.$$

Обозначим

$$\mu_i\left(x^m, k\right) = \prod_{j=1}^{n} \exp\left(-\frac{1}{2} \left(\frac{x_j^m - c_j^i}{\sigma_j^i} \right)^2 \right)$$

и

$$\varepsilon_m(k) = f(x^m, \theta(k)) - y^m$$

В результате получим рекуррентный алгоритм настройки параметров в форме

$$b_i(k+1) = b_i(k) - \lambda_1 \varepsilon_m(k) \frac{\mu_i\left(x^m, k\right)}{\sum_{i=1}^{R} \mu_i\left(x^m, k\right)}. \qquad (5.26)$$

Рассмотрим теперь закон обновления центров функции принадлежности входных переменных c_j^i. Будем осуществлять обучение (настройку) c_j^i в соответствии с алгоритмом

$$c_j^i(k+1) = c_j^i(k) - \lambda_2 \frac{\partial e_m}{\partial c_j^i}\Big|_k,$$

где $\lambda_2 \geq 0$ – параметр скорости настройки. Здесь $i = 1, 2, ..., R$, $j = 1, 2, ..., n$ и $k \geq 0$.

Из уравнения (5.22) найдем выражение для частной производной $\frac{\partial e_m}{\partial c_j^i}$ в момент времени $k \geq 0$. Получим:

$$\frac{\partial e_m}{\partial c_j^i} = \varepsilon_m(k) \frac{\partial f(x^m, \theta)}{\partial \mu_i\left(x^m, k\right)} \frac{\partial \mu_i\left(x^m, k\right)}{\partial c_j^i},$$

где

$$\frac{\partial f(x^m, \theta)}{\partial \mu_i\left(x^m, k\right)} = \frac{\left(\sum_{i=1}^{R} \mu_i\left(x^m, k\right) \right) b_i(k) - \left(\sum_{i=1}^{R} b_i(k) \mu_i\left(x^m, k\right) \right)}{\left(\sum_{i=1}^{R} \mu_i\left(x^m, k\right) \right)^2},$$

так, что

$$\frac{\partial f(x^m,\theta)}{\partial \mu_i\left(x^m,k\right)} = \frac{b_i(k) - \partial f(x^m,\theta)}{\sum_{i=1}^{R} \mu_i\left(x^m,k\right)}.$$

Кроме того,

$$\frac{\partial \mu_i\left(x^m,k\right)}{\partial c_j^i} = \mu_i\left(x^m,k\right)\left(\frac{x_j^m - c_j^i}{(\sigma_j^i)^2}\right).$$

В результате получим

$$c_j^i(k+1) = c_j^i(k) - \lambda_2 \varepsilon_m(k)\left(\frac{b_i(k) - \partial f(x^m,\theta)}{\sum_{i=1}^{R} \mu_i\left(x^m,k\right)}\right)\mu_i\left(x^m,k\right)\left(\frac{x_j^m - c_j^i}{(\sigma_j^i)^2}\right). \qquad (5.27)$$

Рассмотрим теперь градиентный алгоритм рекуррентной настройки показателя нечеткости σ_j^i функций принадлежности входных переменных.

Пусть алгоритм обучения параметров σ_j^i имеет вид

$$\sigma_j^i(k+1) = \sigma_j^i(k) - \lambda_3 \frac{\partial e_m}{\partial b_i}\Big|_k,$$

где $\lambda_2 \geq 0$ – параметр, характеризующий скорость обучения. Здесь $i = 1,2,...,R$, $j = 1,2,...,n$ и $k \geq 0$.

Из (5.25) найдем выражение для частной производной $\frac{\partial e_m}{\partial \sigma_j^i}$ в дискретный момент времени $k \geq 0$

$$\frac{\partial e_m}{\partial \sigma_j^i} = \varepsilon_m(k)\frac{\partial f(x^m,\theta(k))}{\partial \mu_i\left(x^m,k\right)}\frac{\partial \mu_i\left(x^m,k\right)}{\partial \sigma_j^i},$$

где

$$\frac{\partial \mu_i\left(x^m,k\right)}{\partial \sigma_j^i} = \mu_i\left(x^m,k\right)\frac{\left(x_j^m - c_j^i(k)\right)^2}{(\sigma_j^i)^3}.$$

В результате получим

$$\sigma_j^i(k+1) = \sigma_j^i(k) - \lambda_3 \varepsilon_m(k)\left(\frac{b_i(k) - \partial f(x^m,\theta)}{\sum_{i=1}^{R} \mu_i\left(x^m,k\right)}\right)\mu_i\left(x^m,k\right)\frac{\left(x_j^m - c_j^i(k)\right)^2}{(\sigma_j^i)^3} \qquad (5.28)$$

Таким образом, для обновления вектора параметров нечетких систем θ могут использоваться уравнения (5.26), (5.27), (5.28). Следует отметить, что в

уравнениях (5.26), (5.27), (5.28) функции принадлежности являются гауссовыми.

Выполняя аналогичные действия, можно найти алгоритмы настройки векторов параметров θ и для других типов функций принадлежности.

Вычислительные процедуры градиентного метода могут быть реализованы как в оперативном (online) режиме, так и в неоперативном (offline) режиме.

Градиентные алгоритмы параметрического оценивания нечетких моделей Сугено

Нечеткая система Сугено может быть представлена в виде

$$f(x,\theta(k)) = \frac{\sum_{i=1}^{R} g_i(x,k)\mu_i(x,k)}{\sum_{i=1}^{R} \mu_i(x,k)}, \qquad (5.29)$$

где $\mu_i(x,k)$ – функции принадлежности, определенные в (5.26);

$x = [x_1, x_2, ..., x_n]^T$ – вектор входных переменных;

функция $g_i(x,k) = a_{i,0}(k) + a_{i,1}(k)x_1 + a_{i,2}(k)x_2 + ... + a_{i,n}(k)x_n$.

Рассмотрим алгоритм обновления параметров функции $g(x,k)$. Этот алгоритм должен обеспечить настройку параметров a_{ij} функции $g_i(x,k)$, а также параметров c_j^i и σ_j^i функций принадлежности.

Для настройки параметров a_{ij} используем следующий алгоритм:

$$a_{i,j}(k+1) = a_{i,j}(k) - \lambda_4 \frac{\partial e_m}{\partial a_{i,j}}\Big|_k, \qquad (5.30)$$

где $\lambda_4 \geq 0$ – параметр, влияющий на скорость настройки. Здесь $i = 1,2,...,R$, $j = 0,1,2,...,n$ и $k \geq 0$.

Из (5.25) найдем выражение для частной производной $\frac{\partial e_m}{\partial a_{i,j}}$ в дискретный момент времени $k \geq 0$

$$\frac{\partial e_m}{\partial a_{i,j}} = \varepsilon_m(k) \frac{\partial f(x^m,\theta(k))}{\partial g_i(x^m,k)} \frac{\partial g_i(x^m,k)}{\partial a_{i,j}(k)} \qquad (5.31)$$

где

$$\frac{\partial f(x^m,\theta(k))}{\partial g_i(x^m,k)} = \frac{\mu_i(x^m,k)}{\sum_{i=1}^{R}\mu_i(x^m,k)}$$

для всех $i = 1, 2, \ldots, R$.

Далее

$$\frac{\partial g_i(x^m,k)}{\partial a_{i,0}(k)} = 1$$

и

$$\frac{\partial g_i(x\ ,k)}{\partial a_{i,j}(k)} = x_j$$

для всех $i = 1, 2, \ldots, R$ и $j = 1,2,...,n$.

Тогда получим

$$a_{i,j}(k+1) = \begin{cases} a_{i,j}(k) - \lambda_4 \varepsilon_m(k) \dfrac{\mu_i(x^m,k)}{\sum_{i=1}^{R}\mu_i(x^m,k)} & \textbf{\textit{if j = 0}} \\ a_{i,j}(k) - \lambda_4 \varepsilon_m(k) \dfrac{\mu_i(x^m,k)}{\sum_{i=1}^{R}\mu_i(x^m,k)} x_j & \textbf{\textit{if j = 1...n}} \end{cases} \qquad (5.32)$$

Рассмотрим теперь алгоритмы настройки параметров функций принадлежности входных переменных. Алгоритмы настройки центров функций принадлежности c_j^i аналогичны тем, которые использовались для обучения нечетких систем Мамдани.

Таким образом, алгоритм настройки центров c_j^i функций принадлежности с использованием обучающего набора данных имеет вид

$$c_j^i(k+1) = c_j^i(k) - \lambda_2 \varepsilon_m(k) \left(\frac{b_i(k) - \partial f(x^m,\theta)}{\sum_{i=1}^{R}\mu_i\left(x^m,k\right)} \right) \mu_i\left(x^m,k\right) \left(\frac{x_j^m - c_j^i}{(\sigma_j^i)^2} \right) \qquad (5.33)$$

Рассмотрим теперь алгоритмы настройки показателей нечеткости σ_j^i параметров функций принадлежности. Алгоритмы настройки показателей

нечеткости σ_j^i функций принадлежности нечетких систем Сугено аналогичны тем, которые использовались для обучения нечетких систем Мамдани.

Таким образом, можно записать выражение для алгоритма настройки показателей нечеткости функций принадлежности для входных переменных σ_j^i в виде

$$\sigma_j^i(k+1) = \sigma_j^i(k) - \lambda_3 \varepsilon_m(k) \left(\frac{b_i(k) - \partial f(x^m, \theta)}{\sum_{i=1}^{R} \mu_i\left(x^m, k\right)} \right) \mu_i\left(x^m, k\right) \frac{\left(x_j^m - c_j^i(k)\right)^2}{(\sigma_j^i)^3} \qquad (5.34)$$

В заключение отметим, что функциональные нечеткие системы Сугено могут обучаться в оперативном (online) или в неоперативном (offline) режимах также, как и рекурсивные алгоритмы метода наименьших квадратов.

Для иллюстрации этих алгоритмов используем набор обучающих данных, который использовался для обучения нечетких моделей на основе процедур метода наименьших квадратов.

Пусть нечеткая система Такаги-Сугено использует два правила ($R = 2$) и две входных переменных ($n = 2$). Пусть обучающая выборка, которую можно использовать для обучения нечеткой системы состоит из следующих данных:

$$G = \left\{ \left(\begin{bmatrix} 0 \\ 2 \end{bmatrix}, 1 \right), \left(\begin{bmatrix} 2 \\ 4 \end{bmatrix}, 5 \right), \left(\begin{bmatrix} 3 \\ 6 \end{bmatrix}, 6 \right) \right\}.$$

Пусть рекурсивная процедура, используемая для обучения нечеткой системы, выполняется за 40 циклов, т. е. G = 40. Это позволяет получить малое значение ошибки между выходами нечеткой аппроксимирующей системы и выходами объекта управления.

С учетом (5.31), (5.32), (5.33) получим выражение для обновления значений параметров функций принадлежности $c_j^i(k)$, $a_{i,j}(k)$, $\sigma_j^i(k)$ для всех $i = 1, 2..., R$, $j = 1, 2, .., n$.

При начальной инициализации алгоритма выберем $\lambda_4 = 0.01$, $\lambda_2 = 1$, $\lambda_3 = 1$, $a_{i,j} = 1$ для всех I,j и $c_1^1(0) = 1.5$, $c_2^1(0) = 3$, $c_1^2(0) = 3$, $c_2^2(0) = 5$.

По завершению работы вычислительной рекурсивной процедуры оказывается, что ошибка аппроксимации меньше, чем 0.125. Это соответствует следующим значениям параметров:

$$a_{1,0}(119) = 0.874\,,\ a_{1,1}(119) = 0.998,\ a_{1,2}(119) = 0.7309\,;$$
$$a_{2,0}(119) = 0{,}7642\,,\ a_{2,1}(119) = 0{,}3426\,,\ a_{2,2}(119) = 0{,}7642\,.$$

Центры функций принадлежности входных переменных

$$c_1^1(119) = 2.1982\,,\ c_1^2(119) = 2.6379\,;$$
$$c_2^1(119) = 4.2833\,,\ c_2^2(119) = 4.7439\,.$$

Значения показателей нечеткости

$$\sigma_1^1(119) = 0.7654, \sigma_1^2(119) = 2.6423\,;$$
$$\sigma_2^1(119) = 1.2713, \sigma_2^2(119) = 2.6636\,.$$

Результаты моделирования показывают, что полученные настройки результаты отличаются более высокой точностью, чем результаты, полученные с использованием не рекурсивных алгоритмов. Однако необходимо отметить, что в обоих использовавшихся выше методах структура нечетких систем должна быть предварительно задана.

Выбор и задание структуры нечетких моделей оказывает существенное влияние на качество и точность аппроксимации моделей динамических объектов.

Заключение

1. Результаты разработки и исследования принципов построения и типовых структур ИС, основанных на комплексировании принципов робастного, нейро-нечеткого и адаптивного управления показали возможность создания эффективных систем высокой точности и надежности, использующие новейшие способы их построения.

2. Разработанное алгоритмическое обеспечение процессов проектирования и исследования ИС, основанных на комплексировании принципов робастного, нейро-нечеткого и адаптивного управления, позволило определить структуру и выработать требования к программному обеспечению вычислительной части ИС.

3. Разработанное программное обеспечение, отражающее динамическое проектирование и моделирование комплексированных интеллектуальных систем с синтезированными робастными регуляторами позволяет создавать более эффективные системы.

4. Разработанное методическое обеспечение процессов проектирования, моделирования, исследования и реализации интеллектуальных систем, основанных на комплексировании принципов робастного, нейро-нечеткого и адаптивного управления может быть использовано при разработке систем нового поколения.

5. Приведены результаты разработки методического обеспечения робастного управления, позволяющие синтезировать закон управления и параметры настройки регулятора.

6. Дан анализ основных принципов идентификации нелинейных объектов с использованием нечетких моделей обеих типов, а также проведен сравнительный анализ вычислительных процедур, лежащих в основе процессов обучения и идентификации этих моделей.

7. Результаты исследований показали, что вычислительные алгоритмы, лежащие в основе процедур обучения на основе наборов входных–выходных данных, отличаются значительной вычислительной сложностью, что

затрудняет их реализацию в адаптивных системах управления, которые представляют собой системы реального времени.

8. Показано, что наиболее эффективным для решения прикладных задач идентификации и организации адаптивного управления является подход, основанный на использовании нечетких моделей типа Такаги-Сугено.

9. Разработаны и обоснованы методические аспекты применения интеллектуальных нейросетей в интеллектуальных системах управления высокой точности и надежности.

Обозначения и сокращения

ИС – интеллектуальная система

ПО – программное обеспечение

ДЭС – динамическая экспертная система

ГЦ – главная цель

КТС – колесное транспортное средство

АБС – антиблокировочная система

ИНС – искусственная нейронная сеть

ОС – операционная система

САИ – система анализа изображений

ОУ – объект управления

БЗ – база знаний

ИМС – интеллектуальная мехатронная система

Список использованной литературы

1. *Анохин П.К.* Проблемы центра и периферии в физиологии нервной деятельности. – Н.Новгород. 1935.

2. *Пупков К.А., Коньков В.Г.* Интеллектуальные системы. – М.: Изд-во МГТУ им. Н.Э. Баумана. 2003. – 348 с.

3. *Афонин В.Л., Макушкин В.А.* Интеллектуальные робототехнические системы. – М.: Интернет-Университет информационных технологий, 2005. – 208 с.

4. IEEE Standards Project P1003.4 POSIX Part 1: System Application Program Interface (API) - Amendment 1: Realtime Extension. Draft 13.-IEEE, 1992.

5. IEEE Standards Project P1003.4a Thread Extension for Portable Operating Systems. Draft 6.-IEEE, 1992.

6. EEE Standards Project P1003.4b POSIX Part 1: Realtime System API Extension. Draft 6.- IEEE, 1993.

7. IEEE Standards Project P1003.13 Part 1: POSIX Realtime Application Support (AEP). Draft 5.- IEEE, 1992.

8. Операционные системы реального времени (Жданов А.А., ЗАО "РТСофт", PCWeek, 8/1999).

9. LynxOS – операционная система реального времени в стандарте POSIX (Золотарев С.В., Калядин А.Ю., ЗАО "РТСофт").

10. Замечания о выборе операционных систем при построении систем реального времени (А. Жданов, А. Латыев., ЗАО "РТСофт", PCWeek, 1/2001).

11. Материалы сайта Научно-исследовательского института системных исследований (www.niisi.ru) – ОСРВ С2000.

12. Архитектура компьютеров (Буза М.К., ООО «Новое издание», 2006 – 559с.)

13. Методы классической и современной теории автоматического управления. Т.3/ Под ред. Н.Д. Егупова – М.: МГТУ, 2000.

14. Прикладные интеллектуальные системы, основанные на мягких вычислениях/ Под ред. Н. Г. Ярушкиной–Ульяновский государственный технический университет, 2004. 140 с.

15. *Андриков Д.А., Коньков В.Г., Кулаков Б.Б., Пупков К.А.* Интеллектуальная система управления КТС с АБС//Вестник РУДН. –2007. – № 1. – С. 36–47.

16. *Поршнев С.В.* Matlab 7: Основы работы и программирования. – М: Бином, 2006, 319 с.

17. *Куприянов В.В., Фомичёва О.Е.* Интеллектуализация технологий автоматизированных систем. – М.:МГГУ, 1994. – 101 с.

18. *Бериша А.М., Вагин В.Н., Куликов А.В., Фомина М.В.* Методы обнаружения знаний в замкнутых базах данных//Известия РАН. Теория и системы управления. – 2005. – № 6. – С. 143–158.

19. *J. Albus*, The NIST Real-Time Control System (RCS): An Application Survey. In Proceedings of the 1995 AAAI Spring Symposium Series, 1995.

20. *Ernst G.W., He X.* Methods for an expert system to access an external database//In: Sterling L.S. (ed.) Intelligent systems: Concepts and applications. – New York: Plenum Press, 1993. – P. 39–65.

21. *A. Lacaze, K. Murphy, M. DelGiorno.* Autonomous Mobility for the DEMO III Experimental Unmanned Vehicle. In Proceedings of the AUVSI, 2002.

22. *J. Albus and A. Meystel.* Engineering of Mind. John Wiley & Sons, Inc., 2001.

23. *A. Barbera, E. Messina, Hui-Min Huang, C. Schlenoff, S. Balakirsky* Software Engineering for Intelligent Control Systems//KI, vol. 18, Issue: 3, 2004, p. 22 – 26.

24. *Пупков К.А.* Методы робастного, нейро-нечеткого и адаптивного управления: Учебник. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. – 744 с.

25. *Гаврилов А.И.* Перспективы применения нейросетевых технологий в системах автоматического управления // Вестник МГТУ им. Баумана. Приборостроение. – 1998. – № 1. – С. 119 –126.

26. *Гилл Ф., Мюррей У., Райт М.* Практическая оптимизация: Пер. с англ. – М.: Мир, 1985. – 509 с.

27. *Заде Л.А.* Понятие лингвистической переменной и его применение к принятию приближенных решений. – М.: Мир, 1976

28. *Кофман А.* Введение в теорию нечетких множеств.– М.: Радио и связь, 1982.

29. Прикладные нечеткие системы/ Под ред. Т. Тэрано, К. Асаи, М. Сугено –М.: Мир, 1993.

Цель и задачи курса

Целью курса является изучение научных основ мехатроники как науки об интеллектуальных машинах.

Задачами курса являются:

- освоение и умение эксплуатировать модели процессов получения и обработки информации, алгоритмов принятия решений в системах и алгоритмов выработки управления;
- изучение и освоение технических средств, входящих в состав мехатронных систем, включающих элементы микромехатроники, микро и наноэлектроники;
- получение навыков проектирования и оптимизации структуры мехатронных систем, навыков системного программирования и формирования базы знаний в инамической экспертной системы интеллектуальных систем (ИС).

Областью знаний, в которой используется мехатроника, как наука об интеллектуальных механизмах, являются интеллектуальные системы и системные исследования в задачах управления.

Курс предназначен для обучения в бакалавиате.

Данная дисциплина предназначена для подготовки бакалавров по направлению «Автоматизация и управление» и специалистов по специальности «Управление и информатика в технических системах». Курс является обязательным.

Курс является теоретическим, но предполагает получение практических навыков по разработке и эксплуатации мехатронных систем.

Инновационность курса по:

- содержанию

В курсе используются последние научные достижения по комплексированию робастного, нейро-нечеткого и адаптивных алгоритмов в

интеллектуальных системах высокой точности и надежности; достижения в области разработки микроконтроллеров для управления процессами в мехатронных системах.

- методике преподавания

Впервые в мире используется трехуровневое моделирование мехатронных систем: математическое, натурно-математическое и натурное с выходом управляющего устройства на объект через глобальную сеть Интернет в реальном времени.

- литературе

Используются учебники, разработанные автором и монографии и статьи в научных журналах по интеллектуальным мехатронным системам, а именно Пупков К.А., Коньков В.Г. Интеллектуальные системы. М.: МГТУ им. Н.Э. Баумана, 2003, Афонин В.А., Макушкин В.А. Интеллектуальные робототехнические системы. М.: 2005.

- организации учебного процесса

Введен курсовой проект, ориентированный на использование новейших разработок в области наноэлектроники, нанотехнологий и информационных технологий.

Структура курса

Темы лекций:

Лекция 1. – 2 часа

Мехатроника – наука об интеллектуальных машинах и системах машин. Предмет исследования мехатроники. Истоки и исторические аспекты мехатроники и ее роль в современных технологиях.

Самостоятельная работа студента: 3 часа

Лекция 2. – 2 часа

Мехатронные системы и их структуры. Определение мехатронной системы. Состав технических средств и программного обеспечения.

Самостоятельная работа студента: 3 часа

Лекция 3. – 2 часа

Интеллектуализация процессов обработки информации в мехатронных системах. Понятие информационного процесса в системах. Определение интеллектуальной системы.

Самостоятельная работа студента: 3 часа

Лекция 4. – 2 часа

Динамические экспертные системы. Модели окружающей среды и собственного состояния мехатронных систем. Алгоритмы формирования цели в мехатронных системах.

Самостоятельная работа студента: 3 часа

Лекция 5. – 2 часа

Системы представления знаний в мехатронных системах. Нечеткая логика. Продукционное программирование. Семантические сети. Сети Петри. Исчисления предикатов. «Гибкая» логика. Генетические алгоритмы.

Самостоятельная работа студента: 4 часа

Лекция 6. – 2 часа

Методы принятия решений в мехатронных системах. Поиск решений в продукционных системах. Поиск решений в «гибкой» логике. Статистические методы принятия решений.

Самостоятельная работа студента: 3 часа

Лекция 7. – 2 часа

Методы распознавания и классификации информационных признаков. Алгоритмы распознавания и классификации информации. Программное обеспечение процессов распознавания.

Самостоятельная работа студента: 4 часа

Лекция 8. – 2 часа

Нечеткая логика в задачах управления. Фазификация. Дефазификация.

Самостоятельная работа студента: 3 часа

Лекция 9. – 2 часа

Искусственные нейронные сети. Принципы формирования. Идентификация в мехатронных системах. Обучение и самообучение нейронных сетей.

Самостоятельная работа студента: 4 часа

Лекция 10. – 2 часа

Робастные алгоритмы управления в мехатронных системах. H^{∞} - теория управления. Классическое робастное управление.

Самостоятельная работа студента: 3 часа

Лекция 11. – 2 часа

Комплексирование робастных, нейро-нечетких и адаптивных алгоритмов в мехатронных системах. Самоорганизация алгоритмов и формирование признаков перестройки алгоритмов.

Самостоятельная работа студента: 3 часа

Лекция 12. – 2 часа

Технические средства мехатронных систем. Микросенсорные датчики и их интеллектуализация. Микроконтроллеры. Исполнительные устройства мехатроники. Актуаторы.

Самостоятельная работа студента: 3 часа

Лекция 13. – 2 часа

Системы программного обеспечения. Программная среда G2.

Самостоятельная работа студента: 3 часа

Лекция 14. – 2 часа

Автономные мехатронные системы. Устойчивость, качество и точность мехатронных систем.

Самостоятельная работа студента: 3 часа

Лекция 15. – 2 часа

Мехатронные системы, работающие во взаимосвязи с человеком. Идентификация и оценка динамических характеристик человека – оператора. Синтез оптимальных систем «человек-машина».

Самостоятельная работа студента: 3 часа

Лекция 16. – 2 часа

Модульный принцип построения мехатронных систем. Надежность мехатронных систем. Модификация мехатронных систем.

Самостоятельная работа студента: 3 часа

Лекция 17. – 2 часа

Интеллектуализация мехатронной системы робота. Система управления движением элементов робота. Техническое зрение робота.

Самостоятельная работа студента: 3 часа

Лекция 18. – 2 часа

Практическое применение программных систем обработки информации и управления в мехатронных системах. Возможности программных систем Matlab и Simulink.

Самостоятельная работа студента: 3 часа

Темы практических занятий:

Практическое занятие 1. – 3 часа

Математическое моделирование мехатронных систем в среде MatLab. Формирование модели следящей системы. Исследование устойчивости, качества и точности работы системы.

Практическое занятие 2. – 3 часа

Исследование функциональных свойств микроконтроллера на стенде «КОНТАР». Тестирование контроллера и анализ погрешностей. Способы программной коррекции и настройки микроконтроллера.

Практическое занятие 3. – 3 часа

Моделирование систем распознавания изображений. Анализ и оптимизация алгоритмов обработки информации в интеллектуальных системах. Распознавание по методу аналогий.

Практическое занятие 4. – 3 часа

Освоение и исследование динамических экспертных систем. Оценка эффективности работы динамических экспертных систем.

Практическое занятие 5. – 3 часа

Натурно-математическое моделирование и натурные испытания мехатронных систем.

Практическое занятие 6. – 3 часа

Практическое применение программной среды G2. Формирование базы знаний мехатронных систем. Исследование эффективности алгоритмов принятия решений.

Практическое занятие 7. – 3 часа

Исследование динамических характеристик робота-манипулятора «Kawasaki». Исследование характеристики точности позиционирования. Исследование динамических погрешностей робота-манипулятора.

Практическое занятие 8. – 3 часа

Исследование датчиков информации мехатронных систем. Оптические, тепловые, моментные датчики.

Практическое занятие 9. – 3 часа

Исследование динамических характеристик исполнительных устройств мехатронных систем. Электрические , магнитные, пневматические и гидравлические устройства.

Практическое занятие 10. – 3 часа

Исследование динамических свойств человека – оператора по зрительному каналу, по воздействию вибрации.

Курсовая работа – 6 часов

Темы курсовых работ:

1. Исследование и разработка структуры мехатронной системы.
2. Исследование и оптимизация алгоритма и программы вычислений управления для обеспечения желаемого качества и быстродействия системы.
3. Разработка и исследование алгоритма формирования цели в ИМС.
4. Разработка динамической экспертной системы на основе инструментальной программной среды G2.
5. Синтез алгоритмов нечеткой логики в ИМС.
6. Оптимизация программ обработки информации и управления с помощью генетических алгоритмов.
7. Поиск алгоритмов «гибкой» логики
8. Принятие решений в мехатронных системах на базе программной среды G2.
9. Анализ и оценка алгоритмов распознавания образов.
10. Исследование и оценка эффективности алгоритмов обучения ИНС.
11. Исследования и оценка эффективности робастного управления в ИНС.
12. Исследование возможностей самоорганизации алгоритмов робастного, нейро-нечеткого и адаптивного управления в ИНС.
13. Исследование устойчивости и качества работы ИНС.
14. Исследование технического зрения робота-манипулятора.
15. Синтез оптимальной системы «человек – машина - среда».

Описание системы контроля знаний

В курсе «Мехатроника» предусматриваются цикл лекций, практические занятия (лабораторные работы) и курсовая работа.

В систему контроля знаний входит: контроль посещения лекций, контроль выполнения лабораторных работ, контроль поэтапного выполнения

курсовой работы. Особо ценится своевременное выполнение лабораторных работ, качество выполнения курсовой работы, и итоговое испытание.

Промежуточная аттестация студентов проводится в конце каждого месяца, и результаты размещаются на учебном портале.

Правила выполнения письменных работ (курсовых, лабораторных):

Список тем курсовых работ предлагается студентам в начале учебного семестра. Студент вправе выбрать тему из данного списка или предложить свою (согласовав с преподавателем). Курсовая работа выполняется и сдается в срок, указанный в календарном плане.

Требования к оформлению работ: полуторный интервал, кегль — 13, цитирование и сноски в соответствии с принятыми стандартами, выверенность грамматики, орфографии, синтаксиса.

Курсовая работа должна содержать обзорную часть проблемы, иметь четкую постановку задачи, содержать теоретические исследования проблемы и материалы математического моделирования.

Текст отчета о лабораторной работе должен содержать краткую теоретическую и развернутую практическую части, с подробными комментариями ко всем этапам моделирования, объем не менее 4—6 страниц.

Балльная структура оценки:

Посещение лекций: 0- 20 баллов

Практические занятия (лабораторные работы): 0-35 баллов

Курсовая работа: 0 - 25 баллов

Итоговое испытание: 0 - 20 баллов

Всего - 100 баллов

Шкала оценок:

Баллы за семестр	Автоматическая оценка
91-100	5
76-90	4
56-75	3
35-55	-
<35	-

Студенты, получившие положительные оценки по результатам работы в семестре, но претендующие на получение более высокой оценки, могут участвовать в сдаче экзаменов в период сессии. Количество баллов за экзамен от 0 до 25 баллов.

Студенты, набравшие в течение семестра 35-55 баллов, обязаны пройти итоговую семестровую аттестацию в установленном порядке.

Студенты, не выполнившие программу изучаемой дисциплины и не набравшие 35 баллов, не допускаются до прохождения итоговой семестровой аттестации.

Академическая этика

В содержание курса включаются оригинальные научные и практические разработки автора УМК. Привлеченные материалы в содержание курса будут иметь ссылку на соответствующие источники.

Программа курса УМК «Мехатроника»

направления «Автоматизация и управление»

Лекции – 36 часов – 1 зачетная единица (кредит)

Раздел I. Введение

Мехатроника – наука об интеллектуальных машинах и системах машин. Предмет исследования мехатроники. Истоки и исторические аспекты мехатроники и ее роль в современных технологиях. Математические модели мехатроники. Понятие мехатронной системы. Модели мехатронных систем. Техническая среда мехатроники. Информационные технологии в мехатронике. Основные этапы в развитии мехатроники.

Раздел II. Типы мехатронных систем и виды их моделей

Интегрированные модели мехатронных систем автоматического регулирования. Интегрированные на основе микромеханики и микроэлектроники мехатронные системы управления. Мехатронные системы управления технологическими процессами. Мехатронные системы управления в робототехнике с элементами очувствления и технического зрения. Мехатронные системы управления системами искусственных спутников Земли и других космических аппаратов в глобальных системах связи и задачах дистанционного зондирования Земли и др. Перспективы применения наноэлектроники в мехатронных системах различного назначения.

Модели мехатронных систем. Математические, натурно-математические и физические модели мехатронных систем. Структуры мехатронных систем. Модели систем представления знаний. Нечеткая логика, «гибкая» логика, системы продукций, семантические сети, логико-лингвистические модели, теоретиковероятностные модели в мехатронных системах.

Модели и методы поиска решений в мехатронных системах. Продукционное программирование и поиск решений. Поиск решений на основе «гибкой» логики. Поиск решений на основе последовательного анализа. Логико-лингвистические модели поиска решений. Модели поиска решений в

пространстве состояний. Модели поиска решений на основе искусственных нейронных сетей.

Модели распознавания образов в мехатронных системах. Модели признаков в распознавании образов. Использование экспериментальных данных для обучения систем распознавания образов. Метод эталонных моделей. Цифровые модели систем распознавания образов.

Модели динамических экспертных систем (ДЭС). Модели синтеза цели в ДЭС. Модели базы знаний. Состав базы знаний. Модель интеллектуальной мехатронной системы. Модели внешних и внутренних воздействий в интеллектуальных и мехатронных системах.

Модели процессов обработки информации и управления в интеллектуальных мехатронных системах (ИМС). Модели обратных связей по параметрам результата действия и по цели в ИМС. Понимание естественного языка и модели речевого общения в ИМС.

Модели робастного, нейро-нечеткого и адаптивного управления и их комплексирование в ИМС.

Раздел III. Системы программного обеспечения в ИМС.

Понятие о программных системах CAD и CAM автоматизированного проектирования и CALS – технологиях при проектировании и эксплуатации ИМС.

Применение языков параллельного программирования (OCCAM-2, параллельный C++ и др.) для обеспечения работы ИМС в реальном времени. Языки моделирования Matlab в системе Simulink. Программная система G2.

Раздел IV. Элементы и устройства мехатроники и их применение в ИМС.

Оптические, тепловые, электромагнитные и механические датчики информации в ИМС. Устройства преобразования аналоговой информации в цифровую и обратное преобразование. Оцифровка изображений и устройства ввода в ЭВМ. Вычислительные среды: последовательные, параллельные, логические, сетевые. Применений микро и нанопроцессоров, сигнальных процессоров, оптических и мульти транспьютерных вычислительных сред.

Исполнительные устройства, актуаторы, интегрированные модули микромеханики и микроэлектроники.

Интеллектуальная мехатронная система робота – станка. Интеллектуальная мехатронная система управления технологическим процессом. Человеко-машинная система управления летательным аппаратом.

Раздел V. Заключение

Список обязательной и дополнительной литературы

Пупков К.А., Коньков В.Г. Интеллектуальные системы. М.: МГТУ им. Н.Э. Баумана, 2003.

Афонин В.А., Макушкин В.А. Интеллектуальные робототехнические системы. Курс лекций. Учебное пособие. Интернет – Университет информационных технологий. М.: 2005.

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
КАФЕДРА КИБЕРНЕТИКИ И МЕХАТРОНИКИ

КАЛЕНДАРНЫЙ ПЛАН

учебных занятий по дисциплине Мехатроника
Индекс специальности 550200 группы ИУБ-401, 402, 403 курс 4, семестр 7 200 _/200_ уч. года

д.т.н., проф. Пупков К.А.
(звание, степень, фамилия и инициалы ведущего дисциплину)

Число недель 18
Лекций 36 час
Практ. занятий 30 час
Курс. проект 6 час
Экзамены час
Зачеты час
Консульт. час
Всего 72 часа

Недели	Лекции	Число часов	Лабораторные / практические занятия, срок выполнения	Число часов
1 неделя	Мехатроника – наука об интеллектуальных машинах и системах машин. Предмет исследования мехатроники. Истоки и исторические аспекты мехатроники и ее роль в современных технологиях.	2	Математическое моделирование мехатронных систем в среде MatLab. Формирование модели следящей системы. Исследование устойчивости, качества и точности работы системы.	3
2 неделя	Мехатронные системы и их структуры. Определение мехатронной системы. Состав технических средств и программного обеспечения.	2		
3 неделя	Интеллектуализация процессов обработки информации в мехатронных системах. Понятие информационного процесса в системах. Определение интеллектуальной системы.	2	Исследование функциональных свойств микроконтроллера на стенде «КОНТАР». Тестирование контроллера и анализ погрешностей. Способы программной коррекции и настройки микроконтроллера.	3
4 неделя	Динамические экспертные системы. Модели окружающей среды и собственного состояния мехатронных систем. Алгоритмы формирования цели в мехатронных системах.	2		
5 неделя	Системы представления знаний в мехатронных системах. Нечеткая логика. Продукционное программирование. Семантические сети. Сети Петри. Исчисления предикатов. «Гибкая» логика. Генетические алгоритмы.	2	Моделирование систем распознавания изображений. Анализ и оптимизация алгоритмов обработки информации в интеллектуальных системах. Распознавание по методу аналогий.	3

6 неделя	Методы принятия решений в мехатронных системах. Поиск решений в продукционных системах. Поиск решений в «гибкой» логике. Статистические методы принятия решений.	2		
7 неделя	Методы распознавания и классификации информационных признаков. Алгоритмы распознавания и классификации информации. Программное обеспечение процессов распознавания.	2	Освоение и исследование динамических экспертных систем. Оценка эффективности работы динамических экспертных систем.	3
8 неделя	Нечеткая логика в задачах управления. Фазификация. Дефазификация.	2		
9 неделя	Искусственные нейронные сети. Принципы формирования. Идентификация в мехатронных системах. Обучение и самообучение нейронных сетей.	2	Натурно-математическое моделирование и натурные испытания мехатронных систем.	3
10 неделя	Робастные алгоритмы управления в мехатронных системах. H^{∞} - теория управления. Классическое робастное управление.	2		
11 неделя	Комплексирование робастных, нейро-нечетких и адаптивных алгоритмов в мехатронных системах. Самоорганизация алгоритмов и формирование признаков перестройки алгоритмов.	2	Практическое применение программной среды G2. Формирование базы знаний мехатронных систем. Исследование эффективности алгоритмов принятия решений.	3
12 неделя	Технические средства мехатронных систем. Микросенсорные датчики и их интеллектуализация. Микроконтроллеры. Исполнительные устройства мехатроники. Актуаторы.	2		
13 неделя	Системы программного обеспечения. Программная среда G2.	2	Исследование динамических характеристик робота-манипулятора «Kawasaki». Исследование характеристики точности позиционирования. Исследование динамических погрешностей робота-манипулятора.	3
14 неделя	Автономные мехатронные системы. Устойчивость, качество и точность мехатронных систем.	2		

15 неделя	Мехатронные системы, работающие во взаимосвязи с человеком. Идентификация и оценка динамических характеристик человека – оператора. Синтез оптимальных систем «человек-машина».	2	Исследование датчиков информации мехатронных систем. Оптические, тепловые, моментные датчики.	3
16 неделя	Модульный принцип построения мехатронных систем. Надежность мехатронных систем. Модификация мехатронных систем.	2		
17 неделя	Интеллектуализация мехатронной системы робота. Система управления движением элементов робота. Техническое зрение робота.	2	Исследование динамических характеристик исполнительных устройств мехатронных систем. Электрические , магнитные, пневматические и гидравлические устройства.	3
18 неделя	Практическое применение программных систем обработки информации и управления в мехатронных системах. Возможности программных систем Matlab и Simulink.	2	Исследование динамических свойств человека – оператора по зрительному каналу, по воздействию вибрации.	3

Ведущий дисциплину: _____/ К.А. Пупков

Декан инженерного факультета: _____ / Н.К. Пономарев

Дата