Министерство образования и науки Российской Федерации

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

О. В. Веселов П. С. Сабуров

МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ДИАГНОСТИКЕ

Учебное пособие

Допущено Учебно-методическим объединением вузов по образованию в области автоматизированного машиностроения (УМО АМ) в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению подготовки «Автоматизация технологических процессов и производств»

Владимир 2015

УДК 004.89:656.13
ББК 32.813:39.3
В38

Рецензенты:

Кандидат технических наук, доцент
зав. кафедрой электротехники, электроники и автоматики
Московского государственного технологического университета «Станкин»
В. В. Филатов

Доктор технических наук, профессор
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых
М. Ю. Монахов

Печатается по решению редакционно-издательского совета ВлГУ

В38 **Веселов, О. В.** Методы искусственного интеллекта в диагностике : учеб. пособие / О. В. Веселов, П. С. Сабуров ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир : Изд-во ВлГУ, 2015. – 251 с.
ISBN 978-5-9984-0579-2

Изложены вопросы применения системы моделирования MATLAB для построения алгоритмов диагностирования технических систем на основе методов искусственного интеллекта с использованием пакетов «NNTool», «FuzzyLogic», «Simulink», «Stateflow».

Предназначено для подготовки магистрантов по направлению 15.04.06 – Мехатроника и робототехника. Может быть полезно при изучении данного курса студентам различных специальностей, изучающим методы искусственного интеллекта, а также инженерам, магистрантам и аспирантам, использующим в своих разработках нейро-нечеткие алгоритмы диагностирования технического состояния электромеханических систем.

Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС 3-го поколения.

Табл. 4. Ил. 103. Библиогр.: 21 назв.

УДК 004.89:656.13
ББК 32.813:39.3

ISBN 978-5-9984-0579-2

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Что такое искусственные нейронные сети и нечеткая логика? Что они могут делать? Как работают и как их использовать? Эти и другие подобные вопросы задают специалисты из разных областей. Найти вразумительный ответ нелегко.

Университетских курсов мало, а соответствующая литература слишком обширна и специализированна. В большинстве своем она написана техническим языком, многие тексты предполагают свободное владение разделами высшей математики.

Не все обучаемые обладают в необходимом объеме знаниями для использования их в решении различных задач. Кроме того, уровень знаний обучаемых может различаться, что вызывает дополнительные сложности в усвоении материала. Поэтому весь материал, приведённый в пособии, детально описывает последовательность действий по изучению тех или иных разделов.

Одним из мощных инструментов в проведении исследований, связанных с искусственным интеллектом, позволяющим решать разнообразные задачи в различных отраслях знаний, является MATLAB.

Настоящее пособие призвано помочь решить две задачи: научиться пользоваться средствами программы MATLAB и освоить материал по нейронным сетям и нечеткой логике на основе использования прикладных пакетов «Simulink», «NNTool» и «Fuzzylogic». В нем рассматриваются вопросы диагностики электромеханических систем.

На конкретном примере шаг за шагом показаны этапы освоения материала. Изучение начинается с создания модели ЭМС в пакете «Simulink», изучения ее свойств, получения данных, необходимых для проведения дальнейших исследований. Затем выбирается и анализируется нейронная сеть средствами пакета «NNTool», обучается, проходит тренировку и используется в оценке состояния ЭМС. Аналогично строится нечеткий контроллер принятия решений о состоянии объекта на основе информации, полученной на выходах нейронной сети.

Пособие имеет практическую направленность. Никакой другой метод не позволит добиться столь же глубокого понимания.

Текст не обязательно читать от начала до конца, для понимания достаточно знакомства с содержанием выбранной темы.

Все важные понятия формулируются доступным языком. Математические выкладки используются, если они делают изложение более ясным.

Пособие рассчитано на впервые изучающих реализацию методов искусственного интеллекта средствами MATLAB, имеющих знания в области моделирования, работы с программными системами и ранее изучавших сходные курсы.

Читатели, пожелавшие продолжить более углубленное теоретическое изучение, могут воспользоваться литературой, приведенной в библиографическом списке в конце пособия.

ВВЕДЕНИЕ

Основная цель технической диагностики состоит в организации эффективных процессов оценки технического состояния объектов различной сложности.

Одним из факторов, существенно влияющих на эффективность процесса оценки технического состояния, является качество алгоритмов.

Необходимость увеличения производительности труда на операциях диагноза, сокращения времени обнаружения, поиска и устранения неисправностей, уменьшения объемов и сложности средств диагноза вызывает интерес к разработке методов построения алгоритмов диагноза, требующих минимальных затрат на их реализацию. Построение таких алгоритмов во многих случаях сопряжено с большими вычислительными затратами, и поэтому зачастую удовлетворяется упрощенными алгоритмами, затраты на реализацию которых малы, но не обязательно минимальны.

Интуитивные методы построения алгоритмов диагноза не могут гарантировать получения объективного заключения о действительном техническом состоянии объекта. Отсюда следует необходимость разработки формальных методов построения алгоритмов диагноза технического состояния объектов. Это особенно важно для сложных объектов, насчитывающих десятки, сотни и тысячи функционально и конструктивно взаимосвязанных компонент и зачастую требующих многих часов для обнаружения и поиска неисправностей интуитивными способами. Применение формальных методов, кроме того, позволяет автоматизировать процессы построения алгоритмов диагноза при помощи вычислительных средств.

Эффективность процессов диагноза определяется не только качеством алгоритмов, но и в неменьшей степени качеством средств диагноза. Последние могут быть аппаратными или программными, внешними или встроенными, ручными, автоматизированными или автоматическими, специализированными или универсальными.

Сбор данных о состоянии объекта диагностики требует применения надежно работающих внешних и встроенных аппаратурных средств диагноза, обеспечивающих высокую точность измерений и автоматическую обработку и документирование данных. При этом будет гарантирована достоверность результатов диагноза, сведено к минимуму влияние субъективных факторов и упрощена статистическая обработка результатов.

В настоящее время в большинстве случаев проектирование сложных объектов ведется без должного учета того, как они будут проверяться и налаживаться в условиях производства или ремонта, как будут организованы проверка работоспособности, правильности функционирования и поиск неисправностей в условиях их эксплуатации или хранения. Недооценка важности своевременной (на этапе проектирования объектов) и глубокой проработки вопросов организации эффективных процедур диагноза, в том числе автоматизации поиска неисправностей сложных объектов, ведет к непроизводительным материальным затратам, затратам времени и квалифицированной рабочей силы при наладке, профилактике и ремонте.

Среди объективных причин такого положения следует назвать недостаточное развитие теории и методов технической диагностики, слабую проработку принципов построения технических средств диагноза, а также отсутствие налаженного производства таких средств. Существенным является также психологический фактор, состоящий в том, что почти все разработчики считают творческим, созидательным делом непосредственно разработку объектов (изделий, устройств, агрегатов, систем), выполняющих заданные им функции, и не придают должного значения вопросам организации наладки, профилактики и ремонта проектируемых объектов.

Многие из указанных недостатков будут исключены, если задачи диагноза решать на этапе проектирования объектов. Иначе говоря, разработку систем и средств диагноза следует считать такой же обязательной и важной частью проекта нового объекта, как и разработку самого объекта или других его систем и средств управления.

Появление компьютеров радикально изменило подход к построению как систем диагноза, так и методов и алгоритмов диагностирования. Стало возможным хранить данные о результатах диагностиро-

вания на любом этапе жизненного цикла оборудования, создавать сложные модели, упрощающие процедуры диагностирования, выполнять более широкий круг задач, охватывающий и прогностику, и генезис.

Установление диагноза можно рассматривать как специфический процесс управления, целью которого является определение технического состояния объекта. Это хорошо согласуется с современным пониманием управления как процесса осуществления целенаправленных управляющих воздействий на управляемый объект, а кроме того, четко определяет предмет исследований и задачи технической диагностики с позиций общей теории управления.

Еще одной стороной описываемого процесса является получение информации. Обычно предполагается два подхода. В первом случае используются специально устанавливаемые датчики, ориентированные только лишь на выполнение процедур диагностирования. Во втором используются датчики, являющиеся неотъемлемой частью диагностируемого объекта, и других датчиков в системе не предусмотрено. При этом не обязательно, что эти датчики дадут тот объем информации, который закладывается в методе диагностирования. В этом случае приходится использовать методы восстановления необходимой информации, прибегая к помощи теории идентификации.

Появившиеся в последнее время информационные системы, использующие в своем составе компьютеры, накладывают отпечаток на решение задач по обеспечению качества. Становится актуальным переход от обслуживания оборудования по регламенту к обслуживанию по состоянию. В последнем случае имеется возможность отслеживать состояние объекта непрерывно, накапливая данные о состоянии объекта, что, в свою очередь, дает возможность составлять прогноз о состоянии на ближайшую и отдаленную перспективу.

Повышение качества технических систем, решаемое на основе методов компьютерной диагностики в совокупности с развитием методов искусственного интеллекта, связано с разработкой новых подходов, учитывающих особенности современного оборудования и технологии (рис. 1). Поэтому изучение и использование нейронных сетей, нечеткой логики как средства искусственного интеллекта при решении задач диагностики является весьма эффективными.

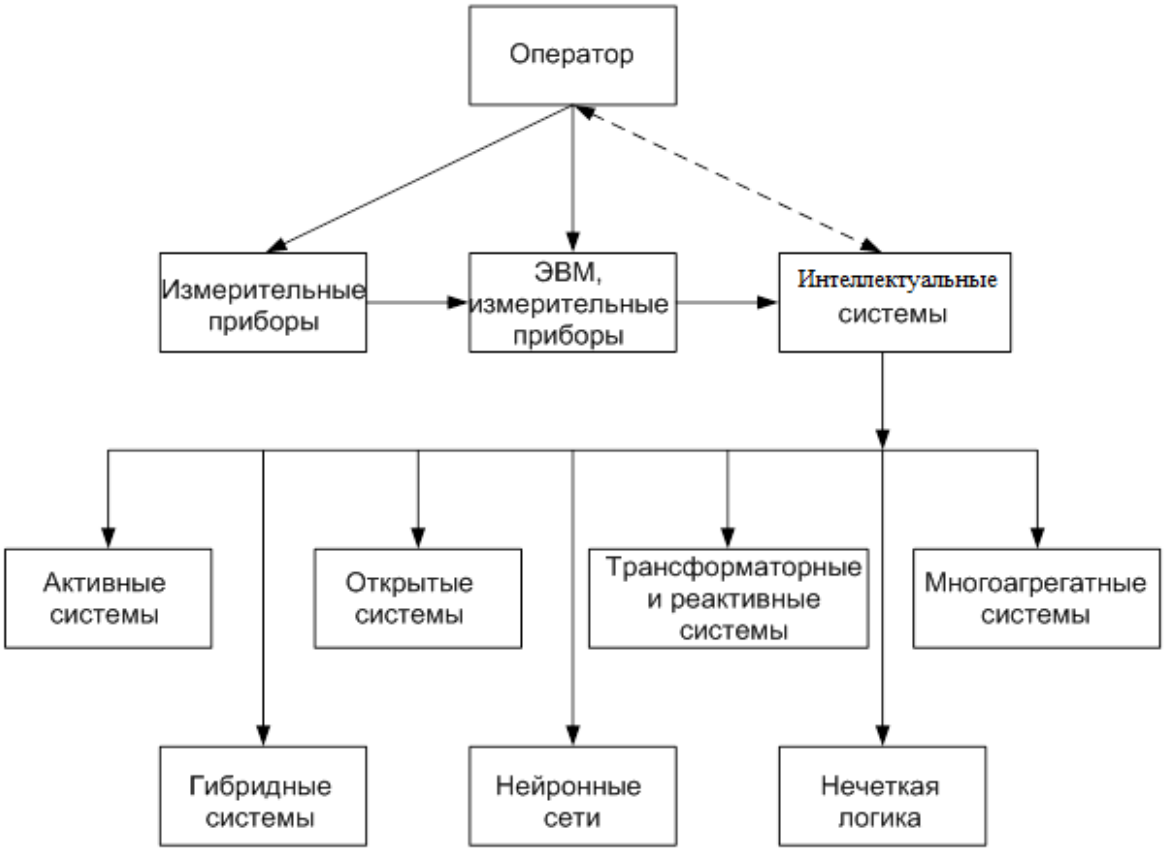


Рис. 1. Эволюция интеллектуальных систем

Любое оборудование, с помощью которого осуществляется технологический процесс, недолговечно и небезотказно. Поэтому необходимо предопределять неисправности своевременно, для того чтобы можно было продлить жизненный цикл техники или заменить ее без поломок и аварий, которые нередко могут привести к большим экономическим затратам и потерям. С этой задачей легко справиться при непрерывном или систематическом (тестовом) контроле и/или диагностировании электромеханических систем, обеспечивая надежность и качество работы. Под надежностью понимается способность системы выполнять заданные функции, сохраняя во времени значения установленных эксплуатационных показателей, определенных в заданных пределах при заданных условиях эксплуатации.

При диагностировании электромеханических систем информация с различных контрольных точек поступает в устройство диагностики, где происходит анализ полученной информации и оценивается техническое состояние оборудования. Чем больше контрольных то-

чек, тем более достоверную и объективную информацию можно получить о процессах, происходящих внутри объекта. Но не всегда вся информация доступна. Чаще всего требуется наблюдение изменения отдельных параметров, по которым определяется неисправность объекта или готовность к дальнейшей работе. Например, для электропривода (ЭП) наиболее распространенными параметрами диагностирования, доступными для непосредственного измерения, являются скорость, положение и ток, по которым можно судить о состоянии данной ЭМС.

Наибольшей популярностью при решении задач диагностики пользуются интеллектуальные системы (ИС). Важными особенностями ИС являются способность анализа и объяснения своих действий и знаний, приобретение новых знаний и изменение в соответствии с ними своего поведения, обеспечение взаимосвязи между оператором, ЭВМ и объектом. На смену многочисленным измерительным приборам пришли новые компьютерные технологии, следствием развития которых является появление нечеткой логики и нейронных сетей (НС), а также их сочетание.

Одним из вариантов может быть система построения на базе ПЭВМ, включающая в себя монитор, панель оператора и печатающее устройство. Программное обеспечение (рис. 2), построенное на основе операционной системы MS-DOS и языка программирования C++, включает в себя следующие основные компоненты: блок ввода данных, блок оказания помощи при вводе, блок управления занесением информации, баз данных, знаний и готовых решений, блок принятия решений, блок пополнения баз.

Блок ввода данных обеспечивает взаимодействие пользователя системы проектирования в процессе ввода, редактирования вводимых данных, распределения полей для вводимых данных и т.п. Информация представляется на экране монитора в виде панелей с описанием последовательности ввода как поэлементно, так и в целом по структуре записи. Каждое меню экрана соответствует описанию определенной подсистемы искусственного интеллекта (ИИ) и сопровождается окном, в котором выводится информация о том, как лучше описать вводимый параметр, его размерность, степень важности, сообщается об отсутствии параметров, без которых система не обеспечит решение поставленной задачи.

Рис. 2. Интеллектуальная система, построенная на основе ЭВМ

По желанию пользователя на экране может быть выведена последовательность окон, в которых будут предложены варианты описания отдельных подсистем. Иногда возникает необходимость вводить данные, не представленные в меню пользователя. Такие параметры заносятся в раздел особых условий и впоследствии – в базу данных. Кроме того, возможно словесное описание отдельных требований с использованием ключевых слов.

После ввода исходной информации система переходит к обработке. Процесс может выполняться в автоматическом режиме или в диалоге с пользователем. Это определяется в панели задания режимов работы. Блок управления базами первоначально анализирует базу готовых решений. Если подходящего решения нет, то происходит объединение баз данных и знаний и начинается процесс отыскания нужного решения. В противном случае пользователю дается сообщение о том, что существует готовое решение, но отличающееся в части отдельных параметров. Если пользователь заинтересован в готовом решении, начинается диалог и выводятся характеристики ИИ. В процессе эксплуатации системы возможно пополнение как базы данных, так и базы знаний. Что касается базы готовых решений, то она пополняется автоматически каждый раз, как только происходит обращение к системе и ее работа не прервана в процессе проектирования. Для пополнения знаний и данных используется диалоговый блок, доступ к которому имеют только специалисты, кроме тех случаев, когда система сама на основе методов обучения и самообучения производит пополнение баз. Вводимая информация интерпретируется соответствующим образом, и происходит ее занесение в списки с одновременным указанием параметров связи. И, наконец, последние два программных блока позволяют выводить информацию в необходимом виде на видеотерминал и на печатающее устройство и получать решение о структуре и параметрах интеллектуального интерфейса.

1. АРХИТЕКТУРА ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ ДИАГНОСТИКИ

Архитектура системы диагностики (СД) неисправностей ЭМС представляет собой две подсистемы: подсистему приема и обработки информации и подсистему интерпретации полученной информации о состоянии объекта диагностирования (рисунок). Первая осуществляет прием данных по состоянию объекта и их последующую обработку (распределение данных, оценку переменных и их отображение). Вторая подсистема производит распознавание с помощью искусственной нейронной сети (ИНС) неисправностей ЭМС, а нечеткий контроллер дает рекомендации по реализации дальнейших действий (формирует выводы и предлагает вариант решения).

Архитектура интеллектуальной системы

В настоящее время ИС используются при решении задач различных типов: принятие решений в условиях неопределенности (неполноты), интерпретация символов и сигналов, предсказание, диагностика, конструирование, планирование, управление, контроль и др.

Основные этапы проектирования модели НС для оценки технического состояния ЭМС:

1) определение логических условий, выделяющих анализируемую выборку в зарегистрированных переходных процессах;

2) задание символьных нечетких переменных для векторов входа и выхода моделируемого объекта;

3) формирование базы «нечетких» правил, описывающей все возможные эксплуатационные режимы работы ЭМС;

4) создание на основании зарегистрированных выборок реальных переходных процессов исправной работы тренировочных, тестирующих и проверочных данных для обучения;

5) обучение адаптивной нейро-нечеткой сети.

Такое представление позволяет сформировать свою систему диагностики на базе нейронной сети, определив все этапы проектирования. То есть создать базу логических условий и правил для системы, спроектировать ее и обучить.

Любая система диагностики опирается на исследование объекта диагностики (ОД), получение и преобразование сигналов, поступающих от объекта (интерфейс), нахождение и устранение неисправностей и оформление предположений о дальнейшей работе объекта.

На вход ОД подается сигнал, с помощью которого на выходе выдается достоверная информация о работе всей системы в целом. Полученный сигнал через интерфейс преобразуется и поступает на вход нейросети, где согласно алгоритму работы НС исследуется, и выводы о состоянии системы отображаются на экран.

Контрольные вопросы

1. Что представляет собой архитектура системы диагностики неисправностей электромеханической системы?

2. В чем состоит назначение подсистемы приема и обработки информации?

3. Для решения каких задач интеллектуальные системы используются в настоящее время?

4. Назовите основные этапы проектирования модели нейронной сети для оценки технического состояния ЭМС.

2. ПОСТРОЕНИЕ МОДЕЛИ СИСТЕМЫ

В качестве объекта будем рассматривать привод постоянного тока с высокомоментным двигателем.

Двигатель постоянного тока (ДПТ) наиболее часто используется при построении систем управления технологическими машинами за счет простоты управления и хороших динамических характеристик.

ДПТ классифицируют по виду магнитной системы статора:

- с возбуждением от постоянных магнитов;
- с электромагнитным возбуждением.

Наибольший интерес представляют ДПТ с возбуждением от постоянных магнитов. Однако в последнее время интерес представляют обращенные электрические машины, в которых ротор выполнен как постоянный магнит, а статор представлен в виде обмотки. В этом случае отсутствует коллекторный аппарат, снижающий надежность, а характеристики остаются такими же, как и у машин с возбуждением от постоянных магнитов.

Выполнение процедур диагностики предполагает детальное описание процессов происходящих в ДПТ, что позволяет оценивать техническое состояние с использованием алгоритмов, построенных на их основе.

Непосредственно измеряемыми переменными в ДПТ являются ток, напряжение и скорость вращения ротора. Остальные параметры могут определяться на основе измеряемых с использование методов идентификации.

Отметим некоторые упрощенные формулы, используемые при описании процедуры управления ДПТ: крутящий момент, развиваемый двигателем, пропорционален току в обмотке якоря (ротора): $\mathrm{M} = k_m I$, где I – ток в обмотке якоря, k_m – коэффициент крутящего момента двигателя; ток в обмотке ротора определяется как $I = (U - E)/R$, где U – напряжение, приложенное к обмотке ротора, R – сопротивление обмотки ротора. ПротивоЭДС в обмотках якоря пропорциональна угловой частоте вращения ротора $E = k_e\omega$, где k_e – коэффициент электродвижущей силы (ЭДС) двигателя, ω – угловая скорость вращения ротора, и тогда $\mathrm{M} = k_m(U - E)/R$.

Следовательно, величиной крутящего момента можно управлять, меняя напряжение на якоре ДПТ. Такой способ применяют для относительно маломощных двигателей.

Для управления более мощными двигателями используют системы с силовым преобразователем с ШИМ-регулированием, когда изменяется величина напряжения в зависимости от длительности импульса, приложенного к якорю двигателя, и регуляторов, формирующих управляющее воздействие.

Двигатель постоянного тока с независимым возбуждением (без учета поперечной реакции якоря и влияния действия вихревых токов) при одномассовой расчетной схеме описывается следующей системой дифференциальных уравнений:

$$\left.\begin{aligned} u_{я} &= i_{я}R_{я} + L_{я}\frac{di_{я}}{dt} + e \\ u_{в} &= i_{в}R_{в} + L_{в}\frac{di_{в}}{dt} \\ e &= C_E \Phi \omega \\ \mathrm{M} &= C_{\mathrm{M}} \Phi i_{я} \\ \mathrm{M} - \mathrm{M}_C &= J\frac{d\omega}{dt} \end{aligned}\right\},$$

где $u_{я}$ – напряжение на якорной обмотке двигателя;

e – ЭДС якоря;

$i_{я}$ – ток якоря;

$R_{Я}$ – сопротивление якорной цепи, Ом;

$L_{Я}$ – индуктивность якорной цепи, Гн;

$R_{В}$ – сопротивление обмотки возбуждения, Ом;

$L_{В}$ – индуктивность обмотки возбуждения, Гн;

$C_{М}$ – конструктивный коэффициент двигателя;

$C_{Е}$ – коэффициент ЭДС двигателя;

ω – скорость вращения вала двигателя C^{-1};

Ф – полезный поток одного полюса, Вб;

J – момент инерции суммарный, приведенный к валу двигателя, кг·м^2;

М – электромагнитный момент двигателя, Н·м;

M_C – момент нагрузки, приведенный к валу двигателя, Н·м.

Модель двигателя постоянного тока с независимым возбуждением (НВ), полученная на основании уравнений, показана на рис. 2.1.

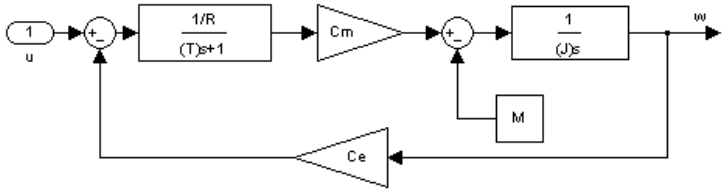


Рис. 2.1. Структурная схема ДПТ с НВ

Далее создадим структурную схему для моделирования привода (рис. 2.2) на основе реальной схемы привода ПРВП 02.

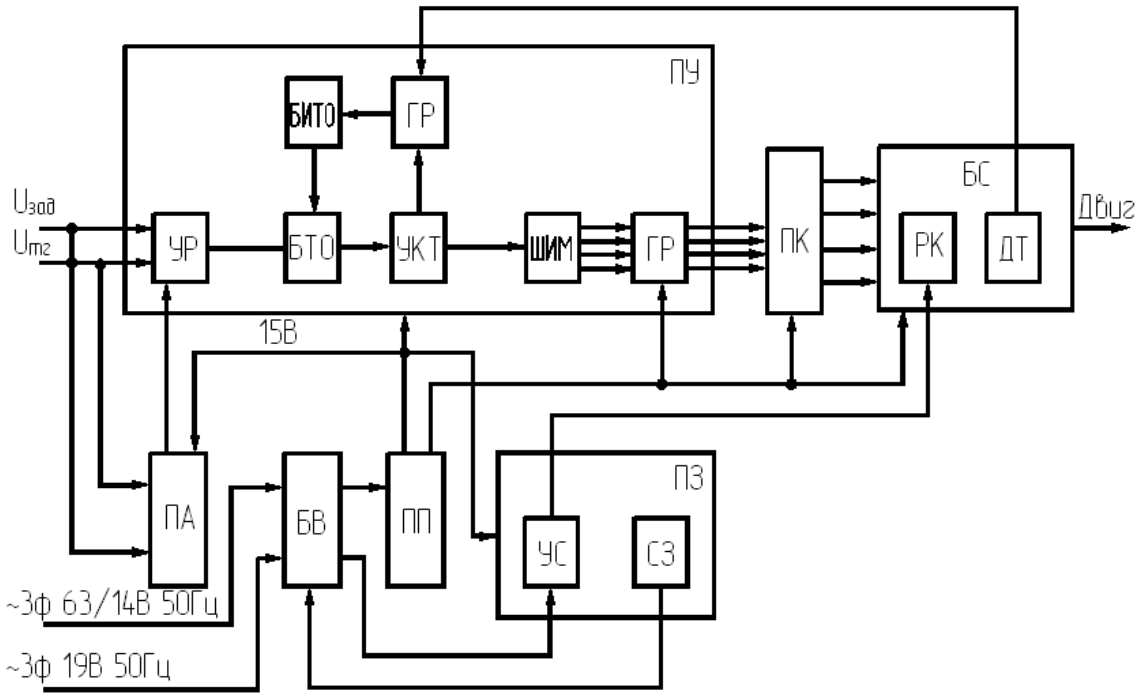


Рис. 2.2. Структурная схема привода ПРВП 02

Регулятор скорости, используемый в приводе ПРВП 02, имеет электрическую принципиальную схему, приведенную на рис. 2.3.

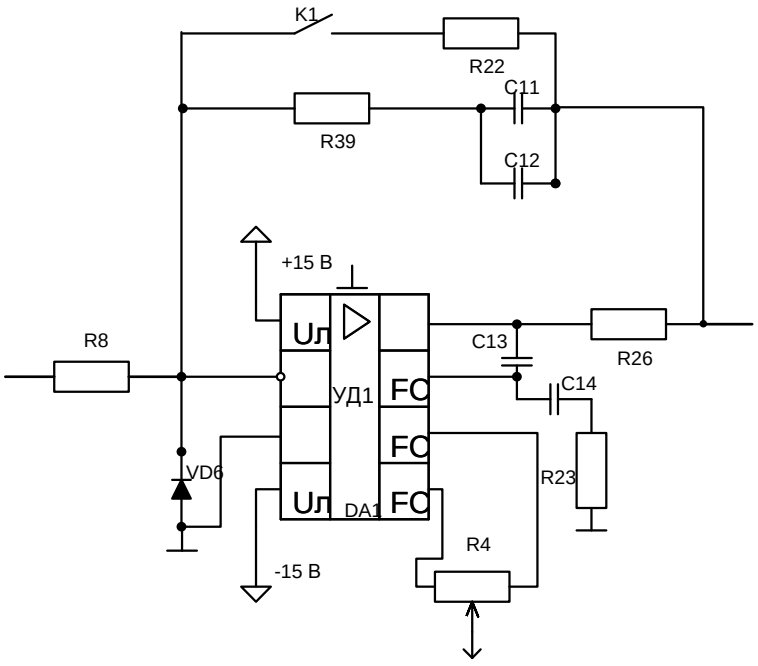


Рис. 2.3. Электрическая принципиальная
схема регулятора скорости

Найдем параметры регулятора скорости:

$$\mathrm{K} = \frac{R_{\text{вых}}}{R_{\text{вх}}} = \frac{R_{39}}{R_{26} + R_{23}} = \frac{75}{43} = 1{,}744;$$

$$T = RC = R_{39}\left(C_{11} + C_{12}\right) = 75 \cdot 10^{3} \cdot 0{,}2 \cdot 10^{-6} = 0{,}015\ \text{с};$$

$$W_{pc}(p) = \frac{(R_{39}(C_{11} + C_{12}))p + 1}{(R_{8}(C_{11} + C_{12}))p} = \frac{0{,}02\,p + 1}{0{,}00102\,p},$$

где К – коэффициент усиления; T – постоянная времени; Wpc – передаточная функция.

Регулятор тока имеет электрическую принципиальную схему, изображенную на рис. 2.4.

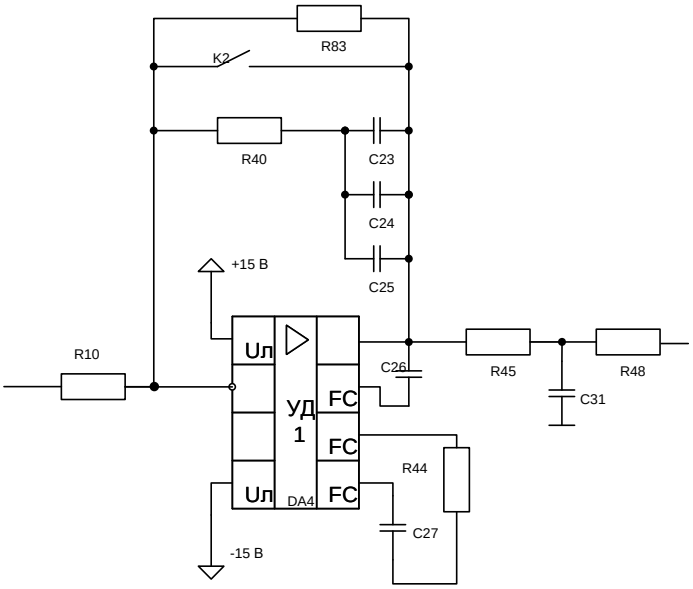


Рис. 2.4. Регулятор тока

Найдем параметры регулятора тока:

$$W_{\text{рт}}(p) = \frac{(R_{40}(C_{23} + C_{24} + C_{25}))p + 1}{(R_{31}(C_{23} + C_{24} + C_{25}))p} = \frac{0{,}0045\,p + 1}{0{,}003\,p};$$

$$K = \frac{R_{\text{вых}}}{R_{\text{вх}}};$$

$$R_{\text{вых}} = \frac{R_{83} \cdot R_{40}}{R_{83} + R_{40}} = \frac{100 \cdot 15}{100 + 15} = 13{,}044\ \text{кОм};$$

$$K = \frac{R_{\text{вых}}}{R_{\text{вх}}} = \frac{13{,}044}{10} = 1{,}304;$$

$$T = RC = R_{40}\left(C_{23} + C_{24} + C_{25}\right) = 15 \cdot 10^{3} \cdot 0{,}3 \cdot 10^{-6} = 0{,}0045\ \text{с}.$$

Рассчитаем коэффициент обратной связи по току

$$K_{\text{ток}} = \frac{2000}{10} = 200\,.$$

Рассчитаем коэффициент обратной связи по скорости

$$K_{\text{ос}} = \frac{2000}{94} = 0{,}043.$$

Рассчитаем RC-фильтр, стоящий на входе,

$$W_{\text{ф1}}(p) = \frac{1}{(R_3C_5)p+1} = \frac{1}{0{,}0051\,p+1}.$$

Рассчитаем RC-фильтры, стоящие в цепи обратной связи по току,

$$W_{\text{ф2}}(p) = \frac{1}{(R_7C_9)p+1} = \frac{1}{0{,}000136\,p+1}.$$

Меняя значения элементов схемы, создавая отклонения от паспортных, соберем экспериментальные данные (рис. 2.5).

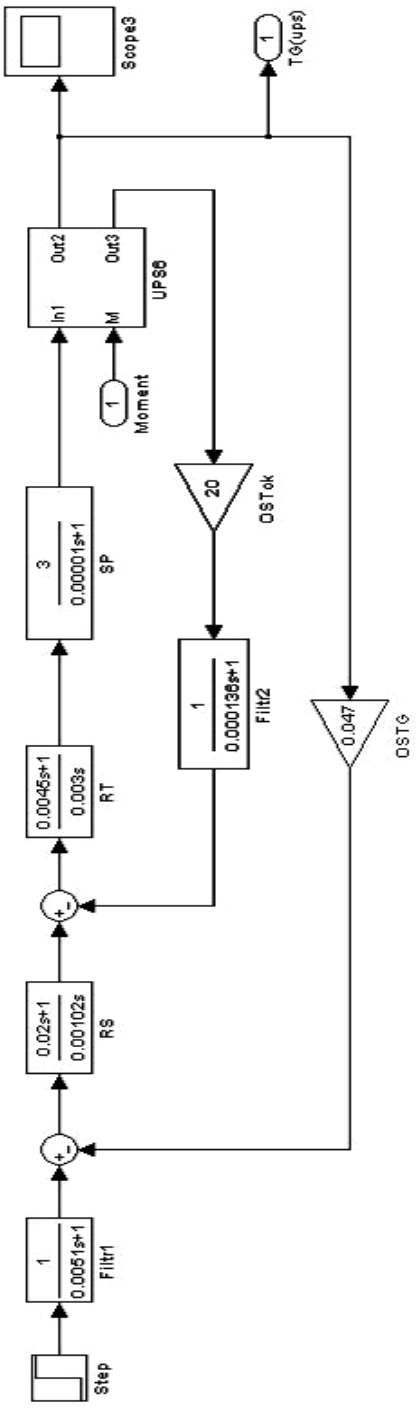


Рис. 2.5. Схема для моделирования привода

Для этого воспользуемся программой «Simulink».

Для запуска программы необходимо предварительно запустить пакет MATLAB. Основное окно пакета MATLAB показано на рис. 2.6. Там же показана подсказка, появляющаяся в окне при наведении указателя мыши на ярлык «Simulink» в панели инструментов.

После открытия основного окна программы MATLAB нужно запустить программу «Simulink». Это можно сделать одним из трех способов:

- нажать кнопку«Simulink» на панели инструментов командного окна MATLAB;

• в командной строке главного окна MATLAB напечатать «Simulink» и нажать клавишу «Enter» на клавиатуре;

• выполнить команду «Open…» в меню «File» и открыть файл модели (*mdl* – файл).

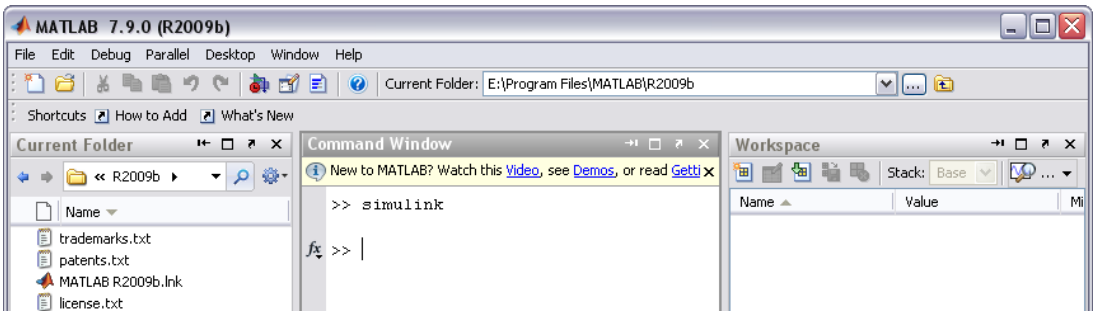


Рис. 2.6. Основное окно программы MATLAB

Последний вариант удобно использовать для запуска уже готовой и отлаженной модели, когда требуется лишь провести расчеты и не нужно добавлять новые блоки в модель. Использование первого и второго способов приводит к открытию окна обозревателя разделов библиотеки «Simulink» (рис. 2.7).

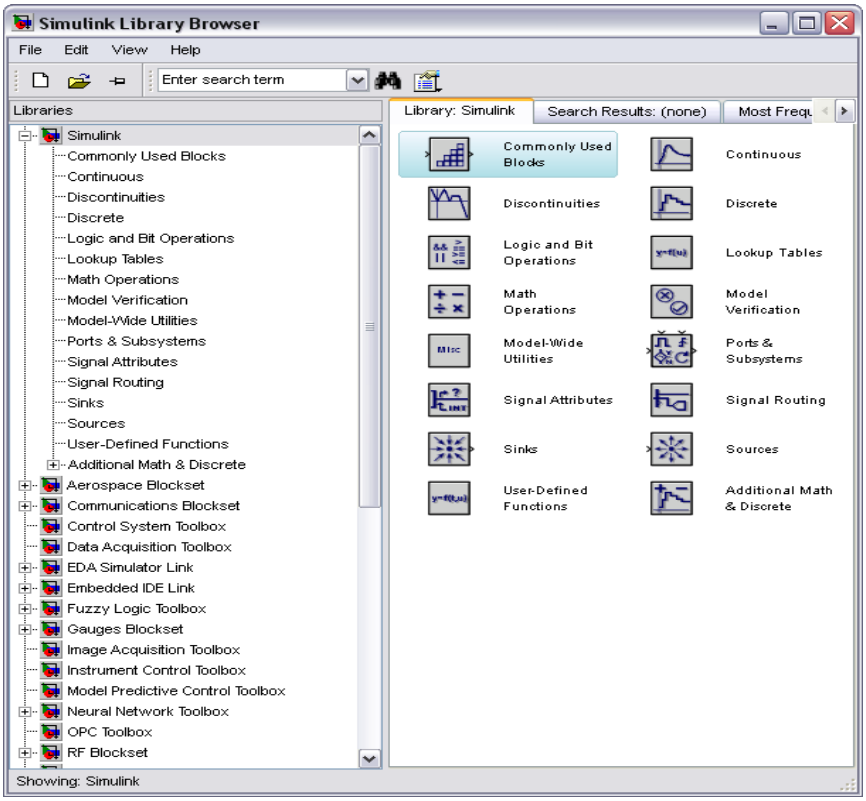


Рис. 2.7. Окно обозревателя разделов библиотеки «Simulink»

Контрольные вопросы

1. Как классифицируют двигатели постоянного тока по виду магнитной системы статора?

2. Какие переменные являются непосредственно измеряемыми в ДПТ?

3. Перечислите способы открытия окна обозревателя разделов библиотеки «Simulink».

4. Какие формулы могут использоваться при описании процедуры управления ДПТ?

3. ОБОЗРЕВАТЕЛЬ РАЗДЕЛОВ БИБЛИОТЕКИ «SIMULINK»

3.1. Общие сведения

Программа «Simulink» является приложением к пакету MATLAB. При моделировании с использованием «Simulink» реализуется принцип визуального программирования, в соответствии с которым пользователь на экране из библиотеки стандартных блоков создает модель устройства и осуществляет расчеты. При этом в отличие от классических способов моделирования пользователю не нужно досконально изучать язык программирования и численные методы математики, а достаточно общих знаний, требующихся при работе на компьютере и, естественно, знаний той предметной области, в которой он работает.

«Simulink» является достаточно самостоятельным инструментом MATLAB, и при работе с ним совсем не требуется знать сам MATLAB и остальные его приложения. Доступ к функциям MATLAB и другим его инструментам остается открытым, и их можно использовать в «Simulink». Часть входящих в состав пакетов имеет инструменты, встраиваемые в «Simulink» (например, «LTI-Viewer» приложения «Control System Toolbox» – пакета для разработки систем управления). Имеются также дополнительные библиотеки блоков для разных областей применения (например, «Power System Blockset» – моделирование электротехнических устройств, «Digital Signal Processing Blockset» – набор блоков для разработки цифровых устройств и т.д).

При работе с «Simulink» пользователь имеет возможность модернизировать библиотечные блоки, создавать свои собственные, а также составлять новые библиотеки блоков.

При моделировании пользователь может выбирать метод решения дифференциальных уравнений, а также способ изменения модельного времени (с фиксированным или переменным шагом). В ходе моделирования имеется возможность следить за процессами, происходящими в системе. Для этого используются специальные устройства наблюдения, входящие в состав библиотеки «Simulink». Результаты моделирования могут быть представлены в виде графиков или таблиц.

Преимущество «Simulink» заключается также в том, что он позволяет пополнять библиотеки блоков с помощью подпрограмм, написанных как на языке MATLAB, так и на языках*C++*, *Fortran* и *Ada*.

Система меню обозревателя библиотек программы «Simulink» состоит из следующих элементов (таблица):

- «File» (Файл) – работа с файлами библиотек;
- «Edit» (Редактирование) – добавление блоков и их поиск;
- «View» (Вид) – управление показом элементов интерфейса;
- «Help» (Справка) – вызов справочной системы.

НАБОР КОМАНД МЕНЮ ОБОЗРЕВАТЕЛЯ

<table>
<tr><th>Команда</th><th colspan="2">Назначение</th></tr>
<tr><td colspan="3">Меню File (Файл)</td></tr>
<tr><td rowspan="3">New</td><td colspan="2">Открыть окно новой блок-диаграммы</td></tr>
<tr><td rowspan="2">Model (Ctrl-N)
Library</td><td>Открыть окно для создания «Simulink» модели</td></tr>
<tr><td>Открыть окно для создания новой библиотеки «Simulink»</td></tr>
<tr><td>Open … (Ctrl – O)</td><td colspan="2">Открыть существующий mdl-файл. При выборе данного пункта открывается окно диалога, с помощью которого можно отыскать и открыть требуемый файл модели</td></tr>
<tr><td>Close (Ctrl – W)</td><td colspan="2">Закрыть окно модели (и соответствующий mdl-файл). В том случае, если модель изменялась, то перед закрытием окна MATLAB запросит подтверждение на закрытие файла</td></tr>
<tr><td>Preferences…</td><td colspan="2">Настройка «Simulink». Задает параметры создаваемых моделей</td></tr>
<tr><td></td><td colspan="2"></td></tr>
</table>

Окончание

Команда	Назначение
Меню Edit (Редактирование)	
Add to the current model	Добавить выделенный блок в текущую модель
Find block...	Найти блок с заданным именем. Команда выводит окно с запросом имени блока
Find next block...	Найти следующий блок с заданным именем. Эту же операцию выполняет и команда Find next в окне задания слова для поиска
Меню View (Вид)	
Toolbar	Вывод/скрытие панели инструментов
Status bar	Вывод/скрытие строки состояния
Description	Вывод/скрытие окна сообщений
Stay on top	Установка статуса окна обозревателя библиотек «поверх всех окон»
Collapse entire Browser	Закрытие текущего раздела библиотеки
Expand entire Browser	Раскрытие текущего раздела библиотеки
Help (Справка)	
Help on the selected block	Справка по выделенному блоку
«Simulink» help	Вывод окна справочной системы «Simulink»
Tip of the day	Полезные советы каждый день

3.2. Запуск «Simulink»

Для запуска программы необходимо предварительно запустить пакет MATLAB. Основное окно пакета MATLAB показано на рис. 3.1. Там же показана подсказка, появляющаяся в окне при наведении указателя мыши на ярлык «Simulink» в панели инструментов.

После открытия основного окна программы MATLAB нужно запустить программу «Simulink». Это можно сделать одним из трех способов:

- нажать кнопку(«Simulink») на панели инструментов командного окна MATLAB;

• в командной строке главного окна MATLAB напечатать «Simulink» и нажать клавишу «Enter» на клавиатуре;

• выполнить команду «Open…» в меню «File» и открыть файл модели (*mdl* – файл).

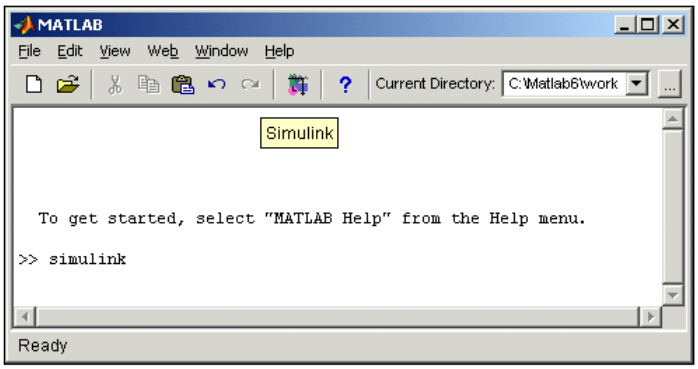


Рис. 3.1. Основное окно программы MATLAB

Последний вариант удобно использовать для запуска уже готовой и отлаженной модели, когда требуется лишь провести расчеты и не нужно добавлять новые блоки в модель. Использование первого и второго способов приводит к открытию окна обозревателя разделов библиотеки «Simulink» (рис. 3.2).

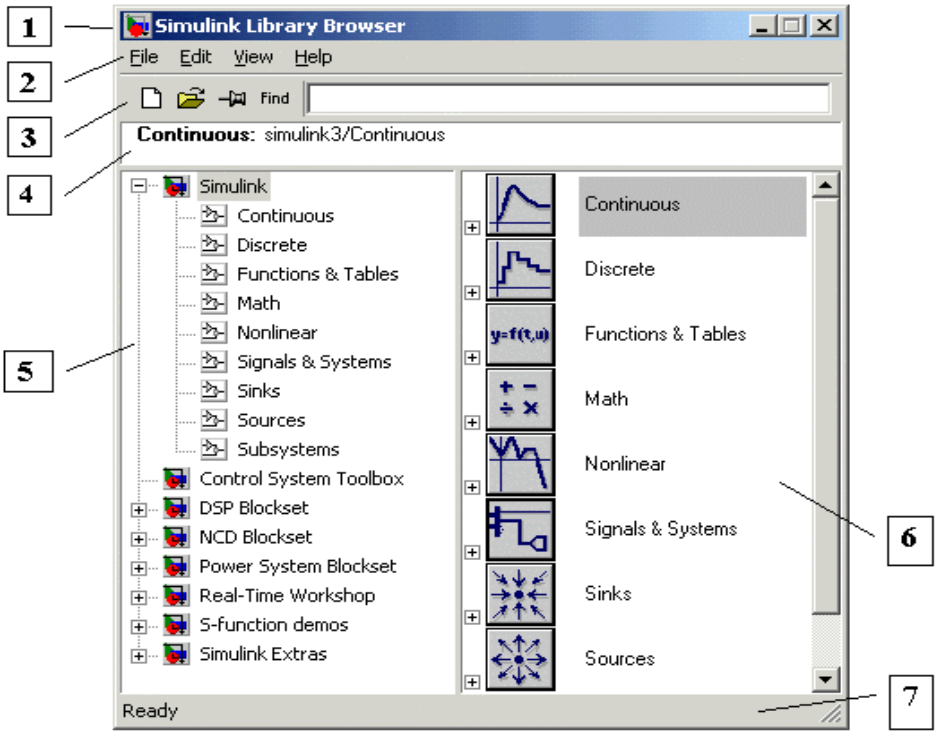


Рис. 3.2. Окно обозревателя разделов библиотеки «Simulink»

3.3. Окно обозревателя

Окно обозревателя библиотеки блоков содержит следующие элементы (см. рис. 3.2):

1. Заголовок с названием окна – «Simulink Library Browser».

2. Меню с командами «File», «Edit», «View», «Help».

3. Панель инструментов с ярлыками наиболее часто используемых команд.

4. Окно комментария для вывода поясняющего сообщения о выбранном блоке.

5. Список разделов библиотеки, реализованный в виде дерева.

6. Окно содержимого раздела библиотеки (список вложенных разделов библиотеки или блоков).

7. Строка состояния, содержащая подсказку по выполняемому действию.

На рис. 3.2 выделена основная библиотека «Simulink» (в левой части окна) и показаны ее разделы (в правой части окна).

Библиотека «Simulink» содержит следующие основные разделы:

1. «Continuous» – линейные блоки.
2. «Discrete» – дискретные блоки.
3. «Functions & Tables» – функции и таблицы.
4. «Math» – блоки математических операций.
5. «Nonlinear» – нелинейные блоки.
6. «Signals & Systems» – сигналы и системы.
7. «Sinks» – регистрирующие устройства.
8. «Sources» – источники сигналов и воздействий.
9. «Subsystems» – блоки подсистем.

Список разделов библиотеки «Simulink» представлен в виде дерева, и правила работы с ним являются общими для списков такого вида.

Пиктограмма свернутого узла дерева содержит символ «+», а пиктограмма развернутого – символ «–».

Для того чтобы развернуть или свернуть узел дерева, достаточно щелкнуть на его пиктограмме левой клавишей мыши (ЛКМ).

При выборе соответствующего раздела библиотеки в правой части окна отображается его содержимое (рис. 3.3).

Для работы с окном используются команды, собранные в меню.

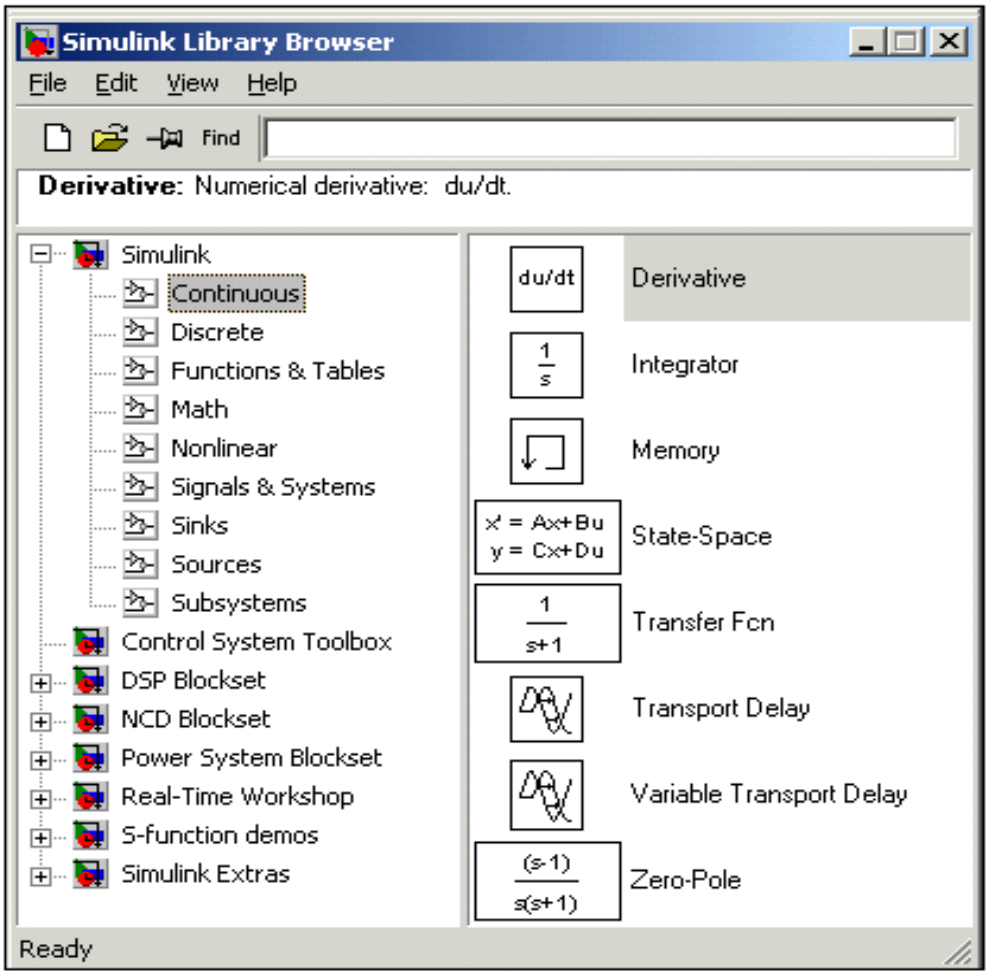


Рис. 3.3. Окно обозревателя с набором блоков раздела библиотеки

Меню обозревателя библиотек содержит следующие пункты:

- File (Файл) – работа с файлами библиотек;
- Edit (Редактирование) – добавление блоков и их поиск (по названию);
- View (Вид) – управление показом элементов интерфейса;
- Help (Справка) – вывод окна справки по обозревателю библиотек.

Для работы с обозревателем можно также использовать кнопки на панели инструментов (рис. 3.4).

Рис. 3.4. Панель инструментов обозревателя разделов библиотек

Кнопки панели инструментов имеют следующее назначение:

1 – создать новую *S*-модель (открыть новое окно модели);

2 – открыть одну из существующих *S*-моделей;

3 – изменить свойства окна обозревателя. Данная кнопка позволяет установить режим отображения окна обозревателя «поверх всех окон». Повторное нажатие отменяет такой режим;

4 – поиск блока по названию (по первым символам названия). После того как блок будет найден, в окне обозревателя откроется соответствующий раздел библиотеки, а блок будет выделен. Если же блок с таким названием отсутствует, то в окне комментария будет выведено сообщение Not found <имя блока> (Блок не найден).

3.4. Создание модели

Для создания модели в среде «Simulink» необходимо последовательно выполнить ряд действий:

1. Создать новый файл модели с помощью команды «File/New/Model» или используя кнопку на панели инструментов (***здесь и далее с помощью символа "/" указаны пункты меню программы, которые необходимо последовательно выбрать для выполнения указанного действия***).

Вновь созданное окно модели показано на рис. 3.5.

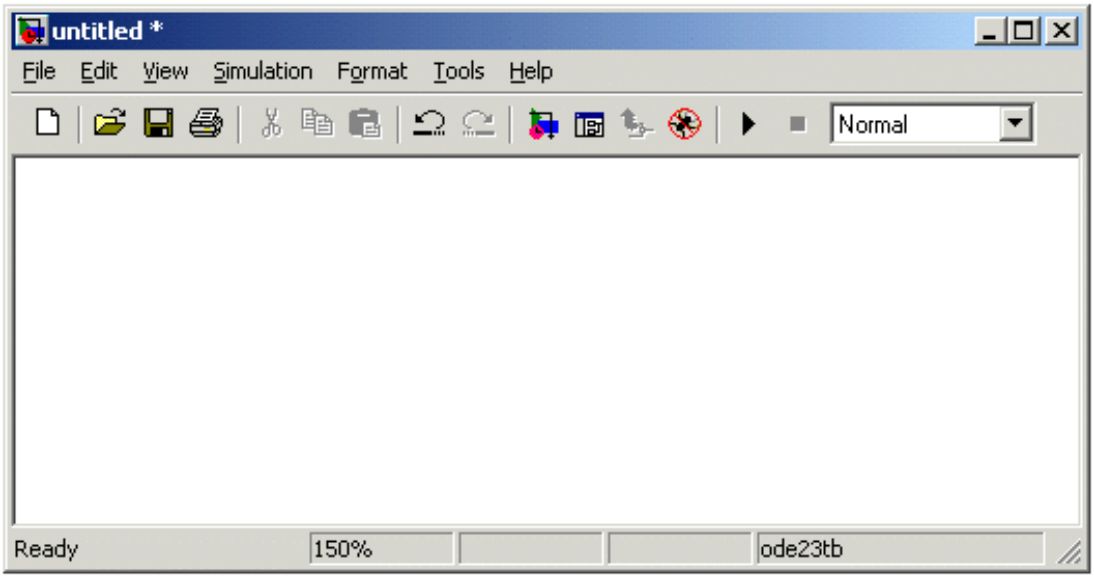


Рис. 3.5. Пустое окно модели

2. Расположить блоки в окне модели. Для этого необходимо открыть соответствующий раздел библиотеки (например, «Sources» – источники). Далее, указав курсором на требуемый блок и нажав на левую клавишу мыши, «перетащить» блок в созданное окно. Клавишу мыши нужно держать нажатой. На рис. 3.6 показано окно модели, содержащее блоки.

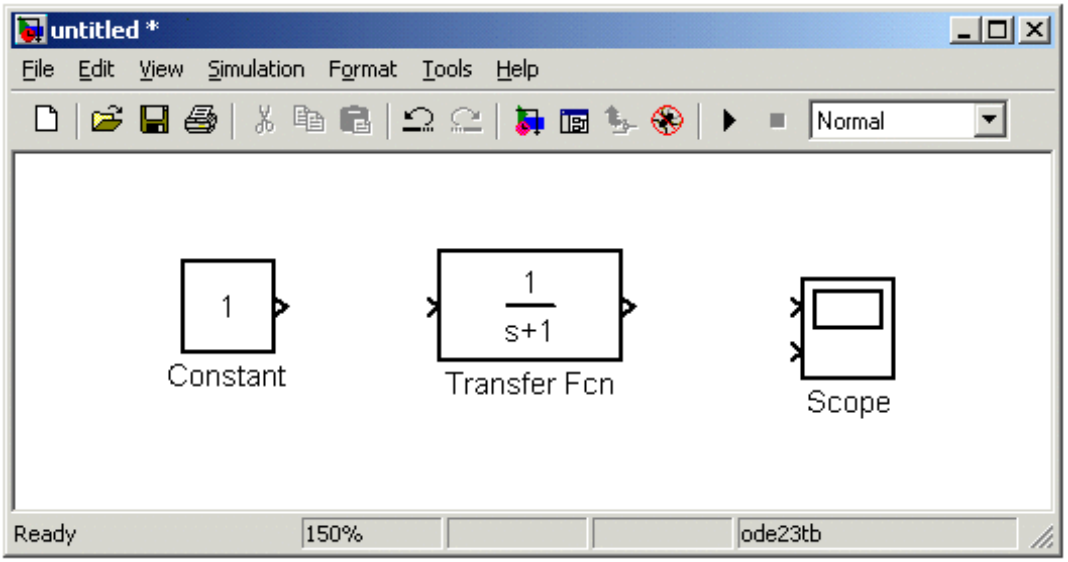


Рис. 3.6. Окно модели, содержащее блоки

Для удаления блока необходимо выбрать блок (указать курсором на его изображение и нажать левую клавишу «мыши»), а затем нажать клавишу «Delete» на клавиатуре.

Для изменения размеров блока требуется выбрать блок, установить курсор в один из углов блока и, нажав левую клавишу «мыши», изменить размер блока (курсор при этом превратится в двухстороннюю стрелку).

3. Далее, если это требуется, нужно изменить параметры блока, установленные программой «по умолчанию». Для этого необходимо дважды щелкнуть левой клавишей «мыши», указав курсором на изображение блока. Откроется окно редактирования параметров данного блока. При задании численных параметров следует иметь в виду, что в качестве десятичного разделителя должна использоваться точка, а не запятая. После внесения изменений нужно закрыть окно кнопкой «OK». На рис. 3.7 в качестве примера показаны блок, моделирующий передаточную функцию, и окно редактирования параметров данного блока.

4. После установки на схеме всех блоков из требуемых библиотек нужно выполнить соединение элементов схемы. Для соединения блоков необходимо указать курсором на «выход» блока, а затем нажать и, не отпуская левую клавишу «мыши», провести линию к входу другого блока. После чего отпустить клавишу. В случае правильного соединения изображение стрелки на входе блока изменяет цвет. Для создания точки разветвления в соединительной линии нужно подвести курсор к предполагаемому узлу и, нажав правую клавишу «мыши», протянуть линию. Для удаления линии требуется выбрать линию (так же, как это выполняется для блока), а затем нажать клавишу «Delete» на клавиатуре. Схема модели, в которой выполнены соединения между блоками, показана на рис. 3.8.

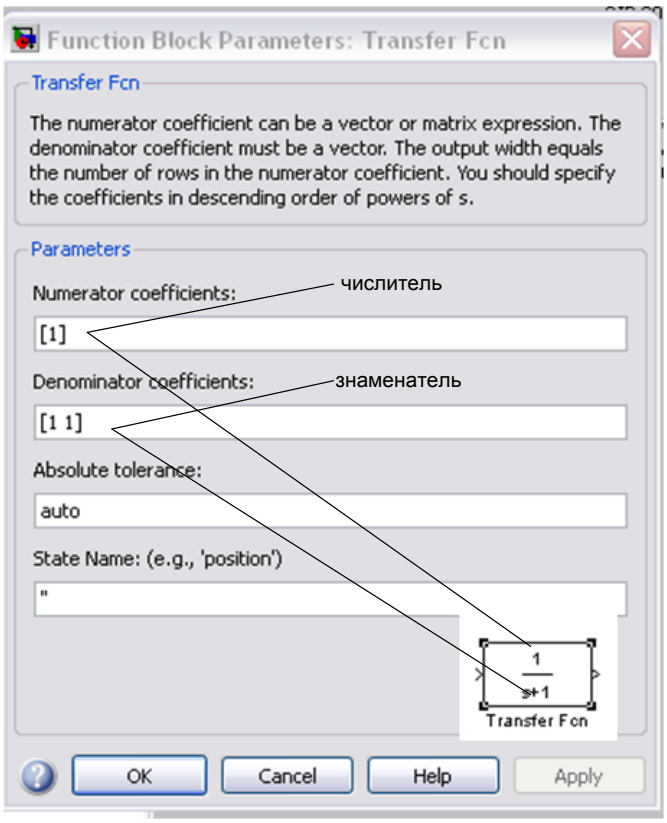


Рис. 3.7. Блок, моделирующий передаточную функцию, и окно редактирования параметров блока

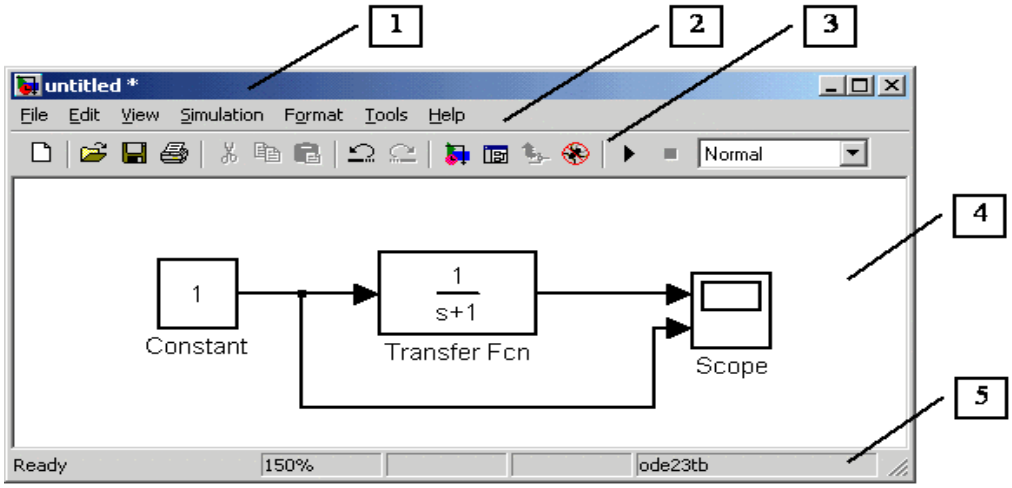


Рис. 3.8. Схема модели

5. После составления расчетной схемы необходимо сохранить ее в виде файла на диске, выбрав пункт меню «File/Save As...» в окне схемы и указав папку и имя файла. Следует иметь в виду, что имя файла не должно превышать 32 символов, должно начинаться с буквы и не может содержать символы кириллицы и спецсимволы. Это же требование относится и к пути файла (к тем папкам, в которых сохраняется файл). При последующем редактировании схемы можно пользоваться пунктом меню «Fille/Save». При повторных запусках программы «Simulink» загрузка схемы осуществляется с помощью меню «File/Open...» в окне обозревателя библиотеки или из основного окна MATLAB.

3.5. Окно модели

Окно модели содержит следующие элементы (см. рис. 3.8):

1 – заголовок, с названием окна. Вновь созданному окну присваивается имя «Untitled» с соответствующим номером;

2 – меню с командами «File», «Edit», «View» и т.д.;

3 – панель инструментов;

4 – окно для создания схемы модели;

5 – строку состояния, содержащую информацию о текущем состоянии модели.

Меню окна содержит команды для редактирования модели, ее настройки и управления процессом расчета, работы файлами и т.п.:

1. «File» (Файл) – работа с файлами моделей.
2. «Edit» (Редактирование) – изменение модели и поиск блоков.
3. «View» (Вид) – управление показом элементов интерфейса.
4. «Simulation» (Моделирование) – задание настроек для моделирования и управления процессом расчета.
5. «Format» (Форматирование) – изменение внешнего вида блоков и модели в целом.
6. «Tools» (Инструментальные средства) – применение специальных средств для работы с моделью (отладчик, линейный анализ и т.п.).
7. «Help» (Справка) – вывод окон справочной системы.

Для работы с моделью можно также использовать кнопки на панели инструментов (рис. 3.9).

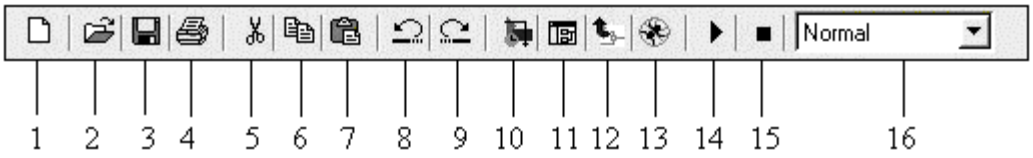


Рис. 3.9. Панель инструментов окна модели

Кнопки панели инструментов имеют следующее назначение:

1) «New Model» – открыть новое (пустое) окно модели;

2) «Open Model» – открыть существующий mdl-файл;

3) «Save Model» – сохранить mdl-файл на диске;

4) «Print Model» – вывод на печать блок-диаграммы модели;

5) «Cut» – вырезать выделенную часть модели в буфер промежуточного хранения;

6) «Copy» – скопировать выделенную часть модели в буфер промежуточного хранения;

7) «Paste» – вставить в окно модели содержимое буфера промежуточного хранения;

8) «Undo» – отменить предыдущую операцию редактирования;

9) «Redo» – восстановить результат отмененной операции редактирования;

10) «Library Browser» – открыть окно обозревателя библиотек;

11) «Toggle Model Browser» – открыть окно обозревателя модели;

12) «Go to parent system» – переход из подсистемы в систему высшего уровня иерархии («родительскую систему»). Команда доступна, только если открыта подсистема;

13) «Debug» – запуск отладчика модели;

14) «Start/Pause/Continue Simulation» – запуск модели на исполнение (команда Start); после запуска модели на изображении кнопки выводится символ, и ей соответствует уже команда «Pause» (Приостановить моделирование); для возобновления моделирования следует щелкнуть по той же кнопке, поскольку в режиме паузы ей соответствует команда «Continue» (Продолжить);

15) «Stop» – закончить моделирование. Кнопка становится доступной после начала моделирования, а также после выполнения команды Pause;

16) «Normal/Accelerator» – обычный/ускоренный режим расчета. Инструмент доступен, если установлено приложение «Simulink Performance Tool».

В нижней части окна модели находится строка состояния, в которой отображаются краткие комментарии к кнопкам панели инструментов, а также к пунктам меню, когда указатель мыши находится над соответствующим элементом интерфейса. Это же текстовое поле используется и для индикации состояния «Simulink»: «Ready» (Готов) или «Running» (Выполнение). В строке состояния отображаются также:

- масштаб отображения блок-диаграммы (в процентах, исходное значение равно 100 %);
- индикатор степени завершенности сеанса моделирования (появляется после запуска модели);
- текущее значения модельного времени (выводится также только после запуска модели);
- используемый алгоритм расчета состояний модели (метод решения).

3.6. Основные приемы подготовки и редактирования модели. Работа с объектами

Добавление текстовых надписей

Для повышения наглядности модели удобно использовать текстовые надписи. Для создания надписи нужно указать мышью место надписи и дважды щелкнуть левой клавишей мыши. После этого появится прямоугольная рамка с курсором ввода. Аналогичным образом можно изменить и подписи к блоками моделей. На рис. 3.10 показаны текстовая надпись и изменение надписи в блоке передаточной функции. Следует иметь в виду, что рассматриваемая версия программы («Simulink» 4) не адаптирована к использованию кириллических шрифтов и применение их может иметь самые разные последствия: отображение надписей в нечитаемом виде, обрезание надписей, сообщения об ошибках, а также невозможность открыть модель после ее сохранения. Поэтому применение надписей на русском языке для текущей версии «Simulink» крайне нежелательно.

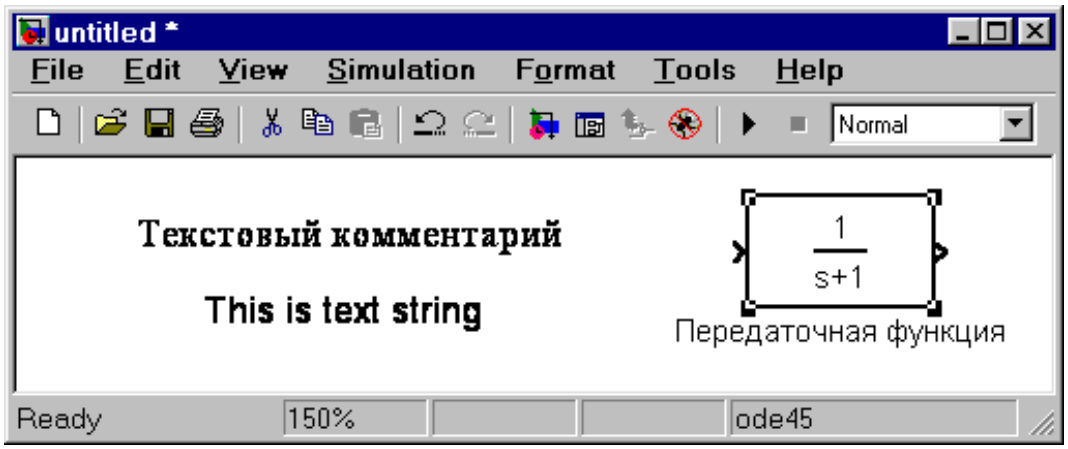


Рис. 3.10. Текстовая надпись и изменение надписи в «Transfer Function»

Выделение объектов

Для выполнения какого-либо действия с элементом модели (блоком, соединительной линией, надписью) этот элемент необходимо сначала выделить.

Выделение объектов проще всего осуществляется мышью. Для этого необходимо установить курсор мыши на нужном объекте и щелкнуть левой клавишей мыши. Произойдет выделение объекта. Об этом будут свидетельствовать маркеры по углам объекта (см. рис. 3.10). Можно также выделить несколько объектов. Для этого надо установить курсор мыши вблизи группы объектов, нажать левую клавишу мыши и, не отпуская ее, начать перемещать мышь. Появится пунктирная рамка, размеры которой будут изменяться при перемещении мыши. Все охваченные рамкой объекты становятся выделенными. Выделить все объекты также можно, используя команду «Edit/Select All». После выделения объекта его можно копировать или перемещать в буфер промежуточного хранения, извлекать из буфера, а также удалять, используя стандартные приемы работы в *Windows*-программах.

Копирование и перемещение объектов в буфер промежуточного хранения

Для копирования объекта в буфер его необходимо предварительно выделить, а затем выполнить команду «Edit/Copy» или воспользоваться инструментом на панели инструментов.

Для вырезания объекта в буфер его необходимо предварительно выделить, а затем выполнить команду «Edit/Cut» или воспользоваться инструментом на панели инструментов. При выполнении данных

операций следует иметь в виду, что объекты помещаются в собственный буфер MATLAB и недоступны из других приложений. Использование команды «Edit/Copy model to Clipboard» позволяет поместить графическое изображение модели в буфер *Windows* и соответственно делает его доступным для остальных программ.

Копирование можно выполнить и таким образом: нажать правую клавишу мыши и, не отпуская ее, переместить объект. При этом будет создана копия объекта, которую можно переместить в необходимое место.

Вставка объектов из буфера промежуточного хранения

Для вставки объекта из буфера необходимо предварительно указать место вставки, щелкнув левой клавишей мыши в предполагаемом месте вставки, а затем выполнить команду «Edit/Paste» или воспользоваться инструментом на панели инструментов.

Удаление объектов

Для удаления объекта его необходимо предварительно выделить, а затем выполнить команду «Edit/Clear» или воспользоваться клавишей «Delete» на клавиатуре. Следует учесть, что команда «Clear» удаляет блок без помещения его в буфер обмена. Однако эту операцию можно отменить командой меню «File/Undo».

Соединение блоков

Для соединения блоков необходимо сначала установить курсор мыши на выходной порт одного из блоков. Курсор при этом превратится в большой крест из тонких линий (рис. 3.11). Держа нажатой левую кнопку мыши, нужно переместить курсор ко входному порту нужного блока. Курсор мыши примет вид креста из тонких сдвоенных линий (рис. 3.12). После создания линии необходимо отпустить левую клавишу мыши. Свидетельством того, что соединение создано, будет жирная стрелка у входного порта блока. Выделение линии производится точно так же, как и выделение блока – одинарным щелчком левой клавиши мыши. Черные маркеры, расположенные в узлах соединительной линии, будут говорить о том, что линия выделена.

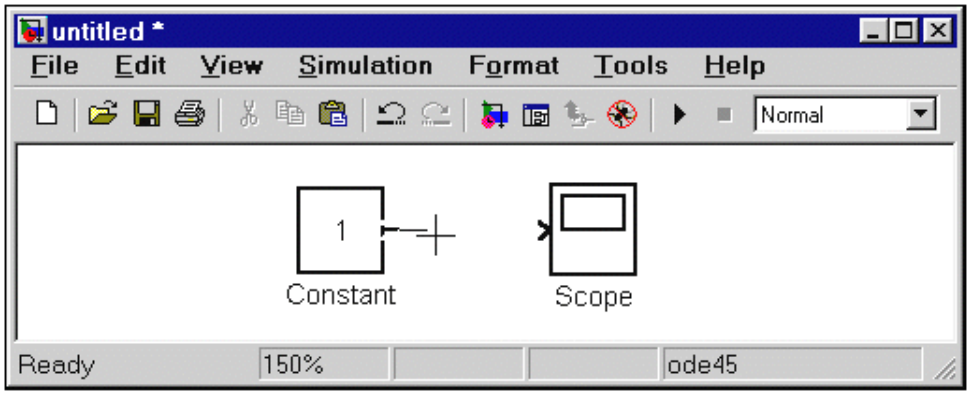


Рис. 3.11. Начало создания соединения

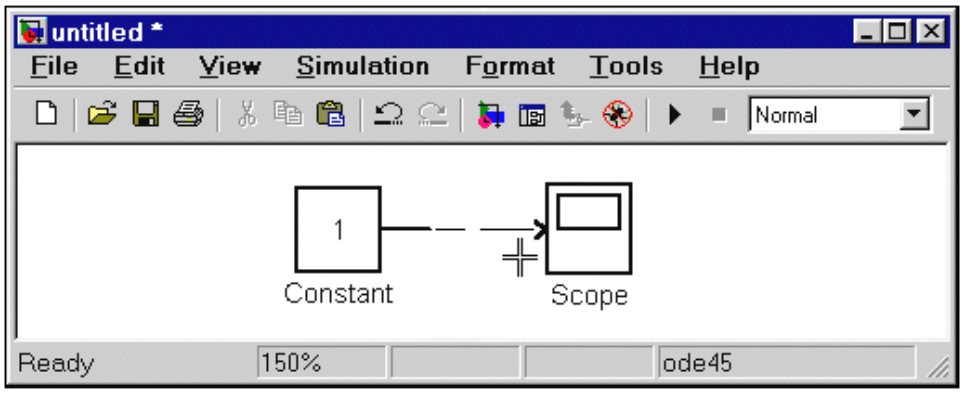


Рис. 3.12. Завершение создания соединения

Создание петли линии соединения выполняется так же, как перемещение блока. Линия соединения выделяется, и затем нужная часть линии перемещается. Рис. 3.13 поясняет этот процесс.

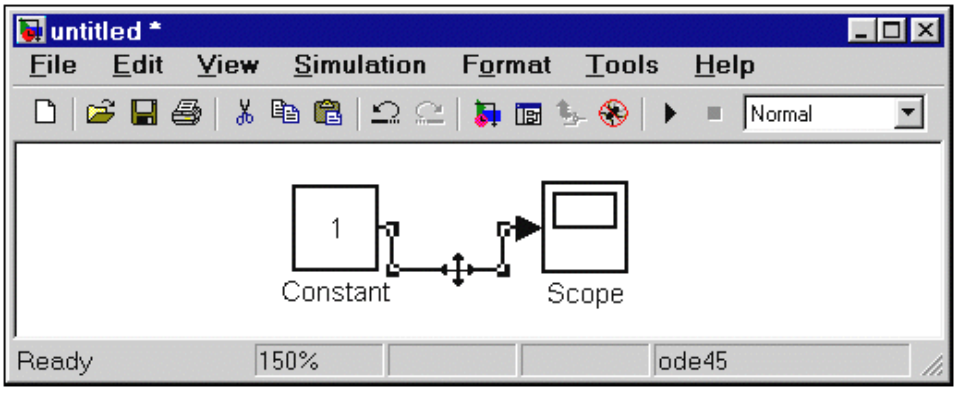


Рис. 3.13. Создание петли в соединительной линии

Удаление соединений выполняется так же, как и любых других объектов (см. *Удаление объектов*).

Изменение размеров блоков

Для изменения размера блока он выделяется, после чего курсор мыши надо установить на один из маркеров по углам блока. После превращения курсора в двустороннюю стрелку необходимо нажать левую клавишу мыши и растянуть (или сжать) изображения блока. На рис. 3.14 показан этот процесс. Размеры надписей блока при этом не изменяются.

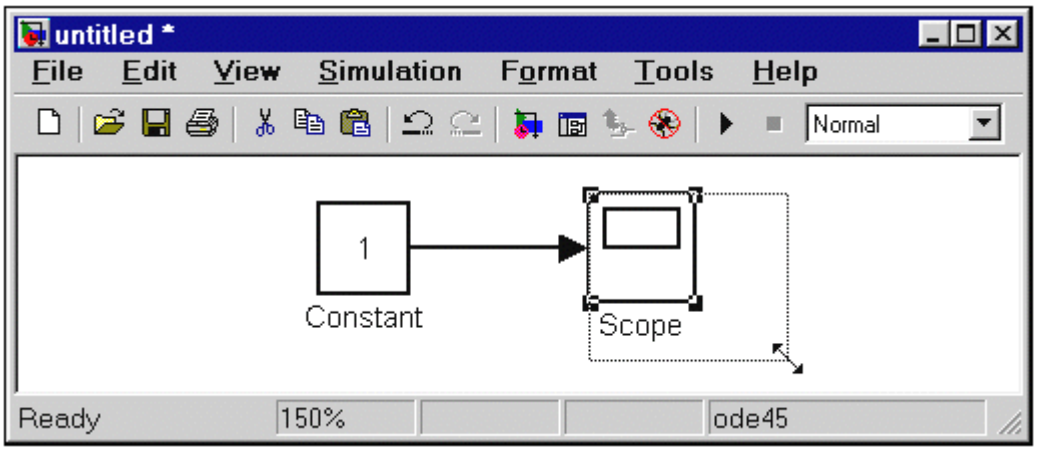


Рис. 3.14. Изменение размера блока

Перемещение блоков

Любой блок модели можно переместить, выделив его и передвинув, держа нажатой левую клавишу мыши. Если к входам и выходам блока подведены соединительные линии, то они не разрываются, а лишь сокращаются или увеличиваются в длине. В соединение можно также вставить блок, имеющий один вход и один выход. Для этого его нужно расположить в требуемом месте соединительной линии.

Использование команд «Undo» и «Redo»

В процессе освоения программы пользователь может совершать действия, кажущиеся ему необратимыми (например, случайное удаление части модели, копирование и т.д.). В этом случае следует воспользоваться командой «Undo» – отмена последней операции. Команду можно вызвать с помощью кнопки в панели инструментов окна модели или из меню «Edit». Для восстановления отмененной операции служит команда «Redo» (инструмент).

Форматирование объектов

В меню «Format» (так же, как и в контекстном меню, вызываемом нажатием правой клавиши мыши на объекте) находится набор

команд форматирования блоков. Команды форматирования разделяются на несколько групп:

1. Изменение отображения надписей:
   - «Font» – форматирование шрифта надписей и текстовых блоков;
   - «Text alignment» – выравнивание текста в текстовых надписях;
   - «Flip name» – перемещение подписи блока;
   - «Show/Hide name» – отображение или скрытие подписи блока.
2. Изменение цветов отображения блоков:
   - «Foreground color» – выбор цвета линий для выделенных блоков;
   - «Background color» – выбор цвета фона выделенных блоков;
   - «Screen color» – выбор цвета фона для всего окна модели.
3. Изменение положения блока и его вида:
   - «Flip block» – зеркальное отображение относительно вертикальной оси симметрии;
   - «Rotate block» – поворот блока на 90° по часовой стрелке;
   - «Show drop shadow» – показ тени от блока;
   - «Show port labels» – показ меток портов.
4. Прочие установки:
   - «Library link display» – показ связей с библиотеками;
   - «Sample time colors» – выбор цвета блока индикации времени;
   - «Wide nonscalar lines» – увеличение/уменьшение ширины нескалярных линий;
   - «Signal dimensions» – показ размерности сигналов;
   - «Port data types» – показ данных о типе портов;
   - «Storage class» – класс памяти. Параметр, устанавливаемый при работе Real-Time Workshop;
   - «Execution order» – вывод порядкового номера блока в последовательности исполнения.

3.7. Установка параметров расчета и его выполнение

Перед выполнением расчетов необходимо предварительно задать параметры расчета. Задание параметров расчета выполняется в панели управления меню «Simulation/Parameters». Вид панели управления приведен на рис. 3.15 и более поздних версий – на рис. 3.16.

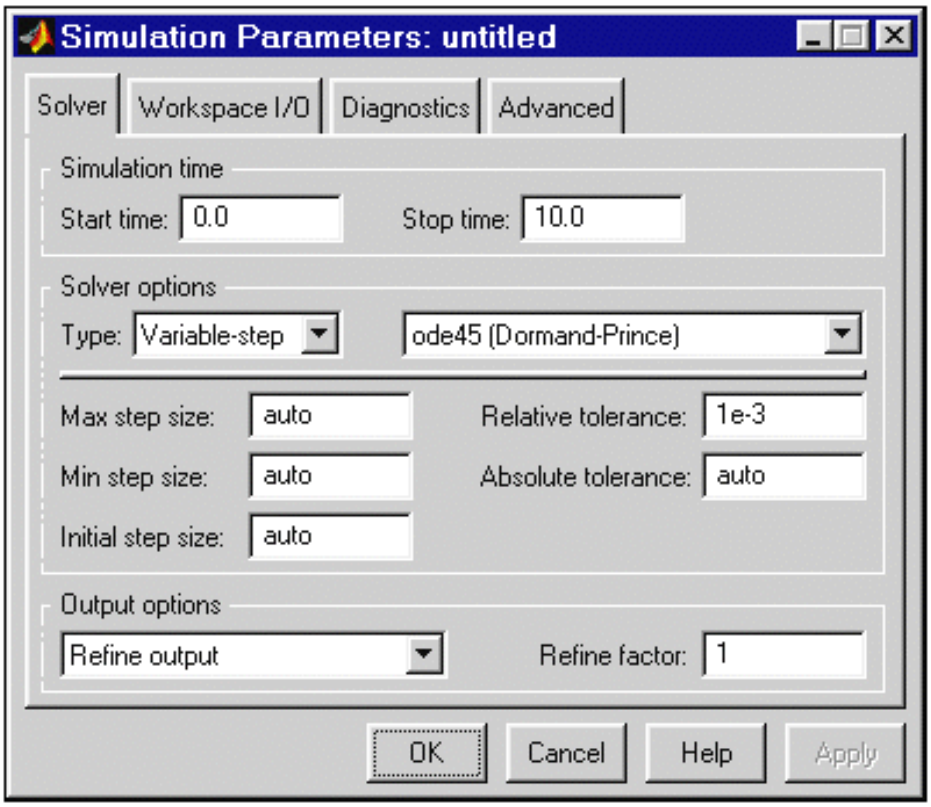


Рис. 3.15. Панель управления

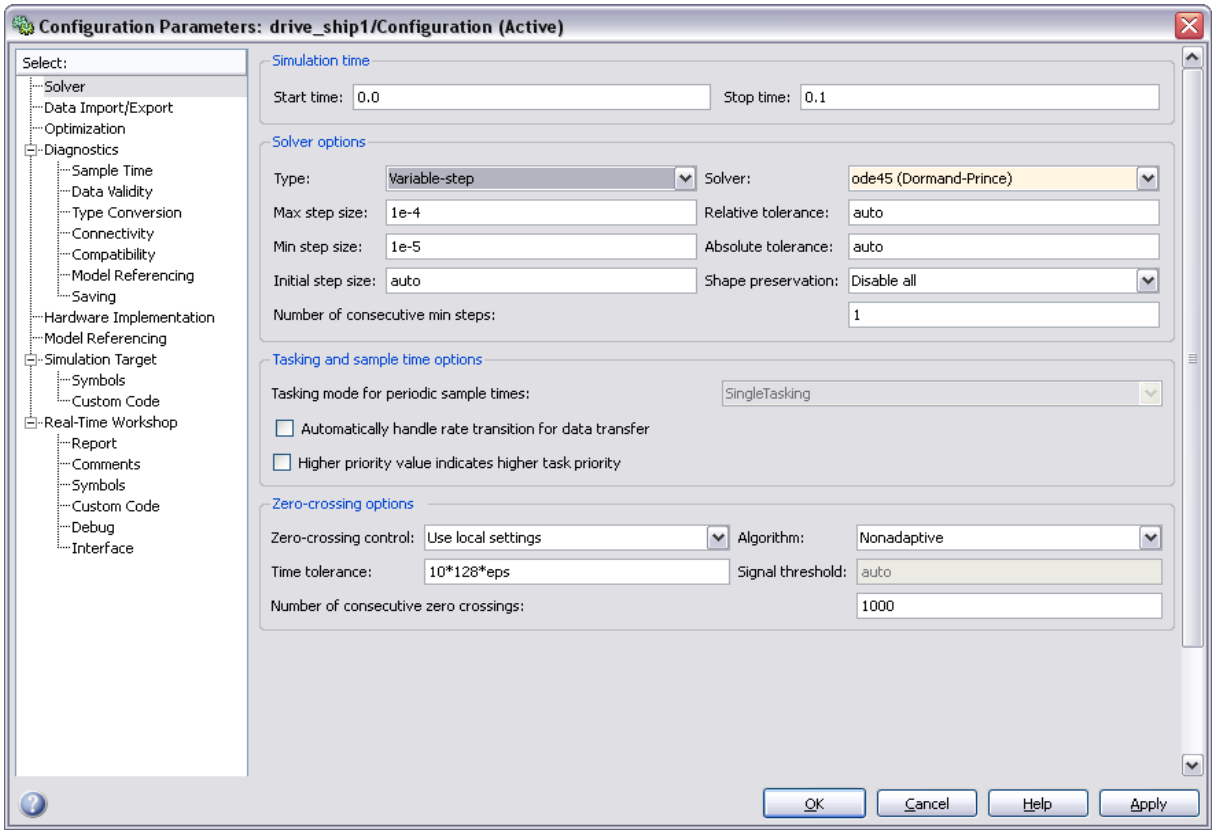


Рис. 3.16. Панель управления

Окно настройки параметров расчета имеет 4 вкладки:

• «Solver» (Расчет) – установка параметров расчета модели;

• «Workspace I/O» (Ввод/вывод данных в рабочую область) – установка параметров обмена данными с рабочей областью MATLAB;

• «Diagnostics» (Диагностика) – выбор параметров диагностического режима;

• «Advanced» (Дополнительно) – установка дополнительных параметров.

Установка параметров расчета модели выполняется с помощью элементов управления, размещенных на вкладке «Solver». Эти элементы разделены на три группы (см. рис. 3.15): «Simulation time» (Интервал моделирования, или, иными словами, время расчета), «Solver options» (Параметры расчета), «Output options» (Параметры вывода).

Установка параметров расчета модели

«Simulation time» (Интервал моделирования, или время расчета). Время расчета задается указанием начального («Start time») и конечного («Stop time») значений времени расчета. Начальное время, как правило, задается равным нулю. Величина конечного времени задается пользователем исходя из условий решаемой задачи.

«Solver options» (Параметры расчета). При выборе параметров расчета необходимо указать способ моделирования («Type») и метод расчета нового состояния системы. Для параметра «Type» доступны два варианта – с фиксированным («Fixed-step») или переменным («Variable-step») шагом. Как правило, «Variable-step» используется для моделирования непрерывных систем, а «Fixed-step» – для дискретных.

Список методов расчета нового состояния системы содержит несколько вариантов. Первый вариант «discrete» используется для расчета дискретных систем. Остальные методы используются для расчета непрерывных систем. Эти методы различны для переменного «Variable-step» и для фиксированного «Fixed-step» шага времени, но, по сути, представляют собой процедуры решения систем дифференциальных уравнений. Подробное описание каждого из методов расчета состояний системы приведено во встроенной справочной системе MATLAB.

Ниже двух раскрывающихся списков «Type» находится область, содержимое которой меняется в зависимости от выбранного способа изменения модельного времени. При выборе «Fixed-step» в данной области появляется текстовое поле «Fixed-step size» (величина фиксированного шага), позволяющее указывать величину шага моделирования (рис. 3.17). Величина шага моделирования по умолчанию устанавливается системой автоматически «auto». Требуемая величина шага может быть введена вместо значения *auto* либо в форме числа, либо в виде вычисляемого выражения (то же самое относится и ко всем параметрам, устанавливаемым системой автоматически).

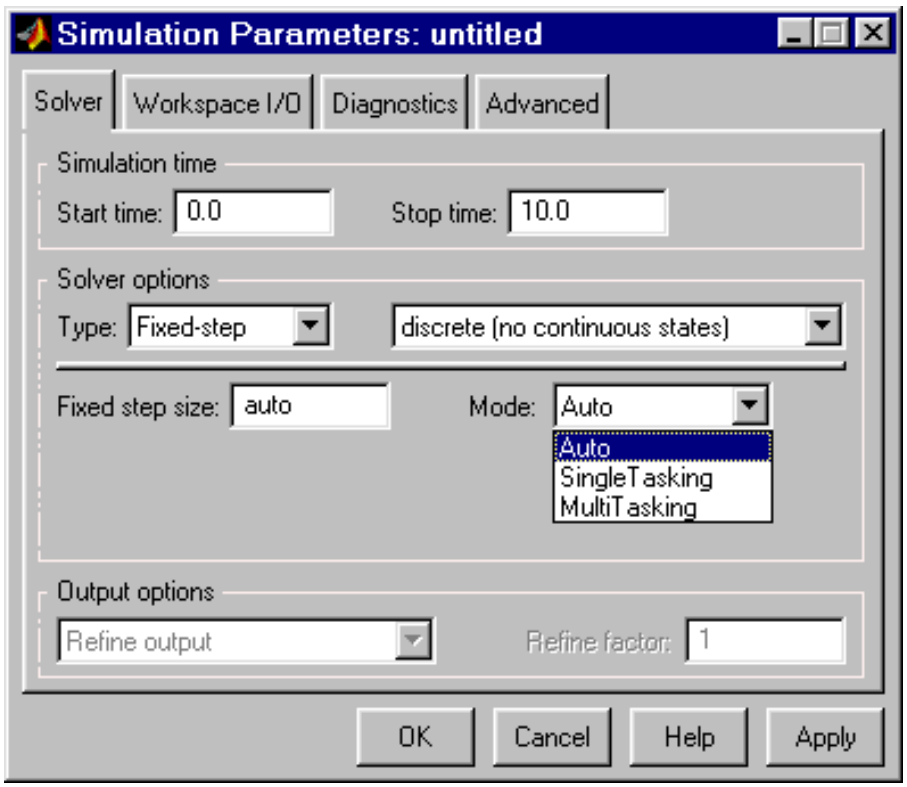


Рис. 3.17. Вкладка «Solver» при выборе фиксированного шага расчета

При выборе «Fixed-step» необходимо также задать режим расчета «Mode». Для параметра «Mode» доступны три варианта:

• «MultiTasking» (Многозадачный) – необходимо использовать, если в модели присутствуют параллельно работающие подсистемы, и результат работы модели зависит от временных параметров этих подсистем. Режим позволяет выявить несоответствие скорости и дискретности сигналов, пересылаемых блоками друг другу;

• «SingleTasking» (Однозадачный) – используется для тех моделей, в которых недостаточно строгая синхронизация работы отдельных составляющих не влияет на конечный результат моделирования;

• «Auto» (Автоматический выбор режима) – позволяет «Simulink» автоматически устанавливать режим «MultiTasking» для тех моделей, в которых используются блоки с различными скоростями передачи сигналов и режим «SingleTasking» для моделей, в которых содержатся блоки, оперирующие одинаковыми скоростями.

При выборе «Variable-step» в области появляются поля для установки трех параметров:

• «Max step size» – максимальный шаг расчета. По умолчанию он устанавливается автоматически «auto», и его значение в этом случае равно «SfopTime–StartTime»/50. Довольно часто это значение оказывается слишком большим, и наблюдаемые графики представляют собой ломаные (а не плавные) линии. В этом случае величину максимального шага расчета необходимо задавать явным образом;

• «Min step size» – минимальный шаг расчета;

• «Initial step size» – начальное значение шага моделирования.

При моделировании непрерывных систем с использованием переменного шага необходимо указать точность вычислений: относительную «Relative tolerance» и абсолютную «Absolute tolerance». По умолчанию они равны соответственно 10^{-3} и «auto».

«Output options» (Параметры вывода). В нижней части вкладки «Solver» задаются настройки параметров вывода выходных сигналов моделируемой системы «Output options». Для данного параметра возможен выбор одного из трех вариантов:

«Refine output» (скорректированный вывод) – позволяет изменять дискретность регистрации модельного времени и тех сигналов, которые сохраняются в рабочей области MATLAB с помощью блока «To Workspace». Установка величины дискретности выполняется в строке редактирования «Refine factor», расположенной справа. По умолчанию значение «Refine factor» равно 1, это означает, что регистрация производится с шагом $\Delta t = 1$ (т. е. для каждого значения модельного времени). Если задать «Refine factor» равным 2, это означает, что будет регистрироваться каждое второе значение сигналов, 3 – каждое третье и т. д. Параметр «Refine factor» может принимать только целые положительные значения.

«Produce additional output» (дополнительный вывод) – обеспечивает дополнительную регистрацию параметров модели в заданные моменты времени; их значения вводятся в строке редактирования (в этом случае она называется «Output times») в виде списка, заключенного в квадратные скобки. При использовании этого варианта базовый шаг регистрации (Δt) равен 1. Значения времени в списке «Output times» могут быть дробными числами и иметь любую точность.

«Produce specified output only» (формировать только заданный вывод) – устанавливает вывод параметров модели только в заданные моменты времени, которые указываются в поле «Output times» (моменты времени вывода).

Установка параметров обмена с рабочей областью

Элементы, позволяющие управлять вводом и выводом в рабочую область MATLAB промежуточных данных и результатов моделирования, расположены на вкладке «Workspace I/O» (рис. 3.18).

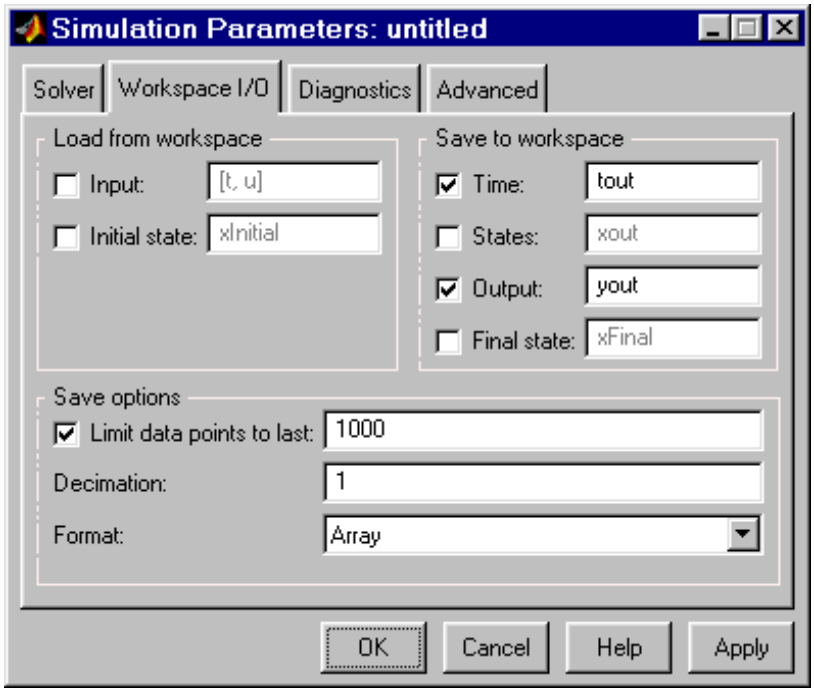


Рис. 3.18. Вкладка Workspace I/O диалогового окна установки параметров моделирования

Элементы вкладки разделены на три поля:

• «Load from workspace» (загрузить из рабочей области). Если флажок «Input» (входные данные) установлен, то в расположенном справа текстовом поле можно ввести формат данных, которые будут

считываться из рабочей области MATLAB. Установка флажка «Initial State» (начальное состояние) позволяет ввести в связанном с ним текстовом поле имя переменной, содержащей параметры начального состояния модели. Данные, указанные в полях «Input и Initial State», передаются в исполняемую модель посредством одного или более блоков «In» (из раздела библиотеки «Sources»);

• «Save to workspace» (записать в рабочую область) – позволяет установить режим вывода значений сигналов в рабочую область MATLAB и задать их имена;

• «Save options» (параметры записи) – задает количество строк при передаче переменных в рабочую область. Если флажок «Limit rows to last» установлен, то в поле ввода можно указать количество передаваемых строк (отсчет строк производится от момента завершения расчета). Если флажок не установлен, то передаются все данные. Параметр «Decimation» (исключение) задает шаг записи переменных в рабочую область (аналогично параметру «Refine factor» вкладки «Solver»). Параметр «Format» (формат данных) задает формат передаваемых в рабочую область данных. Доступные форматы «Array» (массив), «Structure» (структура), «Structure With Time» (структура с дополнительным полем – «время»).

Установка параметров диагностирования модели

Вкладка «Diagnostics» (рис. 3.19) позволяет изменять перечень диагностических сообщений, выводимых «Simulink» в командном окне MATLAB, а также устанавливать дополнительные параметры диагностики модели.

Сообщения об ошибках или проблемных ситуациях, обнаруженных «Simulink» в ходе моделирования и требующих вмешательства разработчика, выводятся в командном окне MATLAB. Исходный перечень таких ситуаций и вид реакции на них приведен в списке на вкладке «Diagnostics». Разработчик может указать вид реакции на каждое из них, используя группу переключателей в поле «Action» (они становятся доступны, если в списке выбрано одно из событий):

• «None» – игнорировать;

• «Warning» – выдать предупреждение и продолжить моделирование;

• «Error» – выдать сообщение об ошибке и остановить сеанс моделирования.

Выбранный вид реакции отображается в списке рядом с наименованием события.

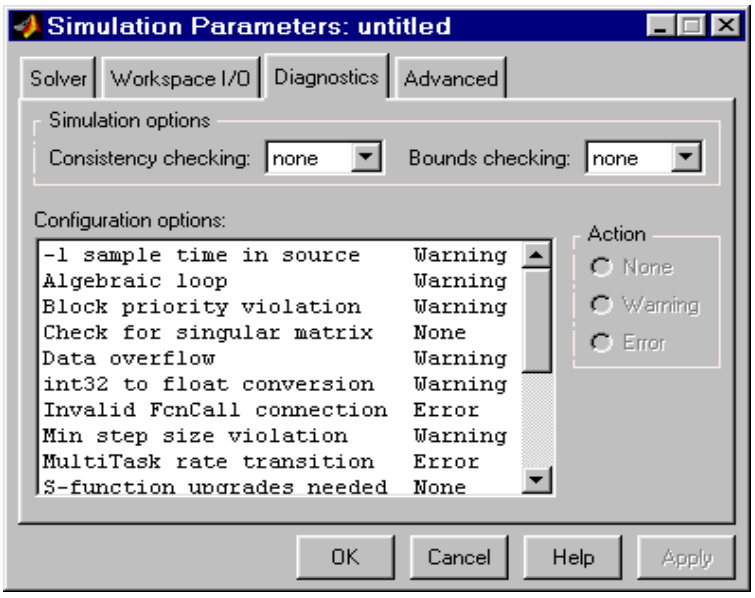


Рис. 3.19. Вкладка «Diagnostics» окна установки параметров моделирования

Выполнение расчета

Запуск расчета выполняется с помощью выбора пункта меню «Simulation/Start» или инструмента на панели инструментов. Процесс расчета можно завершить досрочно, выбрав пункт меню «Simulation/ Stop» или соответствующую кнопку на панели инструментов. Расчет также можно приостановить с помощью комбинации«Simulation/ Pause» и затем продолжить «Simulation/Continue».

3.8. Завершение работы

Для завершения работы необходимо сохранить модель в файле, закрыть окно модели, окно обозревателя библиотек, а также основное окно пакета MATLAB.

Дополнительные материалы по работе с пакетом «Simulink» изложены в прил. 1.

Контрольные вопросы

1. Какие элементы содержит окно обозревателя библиотеки «Simulink»?
2. Перечислите основные разделы библиотеки «Simulink».
3. Опишите процедуру поиска блока по названию.
4. Перечислите способы добавления и удаления блоков из модели.
5. Какие кнопки для работы с моделью находятся на панели инструментов?

4. РАЗРАБОТКА НЕЙРОКОНТРОЛЛЕРА

4.1. Общая характеристика нейронных сетей

Хотя один нейрон и способен выполнять простейшие процедуры распознавания, сила нейронных вычислений проистекает от соединений нейронов в сети.

Искусственные нейронные сети (ИНС) различаются своей архитектурой: структурой связи между нейронами, числом слоев, функцией активации нейронов, алгоритмом обучения. С этой точки зрения среди известных ИНС можно выделить статические, динамические сети и *fuzzy*-структуры; однослойные и многослойные сети. Различия вычислительных процессов в сетях часто обусловлены способом взаимосвязи нейронов, поэтому выделяют следующие виды сетей:

- прямого распространения – сигнал проходит по сети от входа к выходу в одном направлении;
- с обратными связями;
- с боковыми обратными связями;
- гибридные.

В целом, по структуре связей ИНС могут быть сгруппированы в два класса: сети прямого распространения – без обратных связей в структуре и рекуррентные сети – с обратными связями. В первом классе наиболее известными и чаще используемыми являются многослойные нейронные сети, где искусственные нейроны расположены слоями. Связь между слоями однонаправленная, и в общем случае выход каждого нейрона связан со всеми входами нейронов последующего слоя. Такие сети являются статическими, т.к. не имеют в своей структуре ни обратных связей, ни динамических элементов, а выход зависит от заданного множества на входе и не зависит от предыдущих состояний сети. Сети второго класса являются динамическими, т.к. из-за обратных связей состояние сети в каждый момент времени зависит от предшествующего состояния. На рис. 4.1 приведена классификация нейронных сетей.

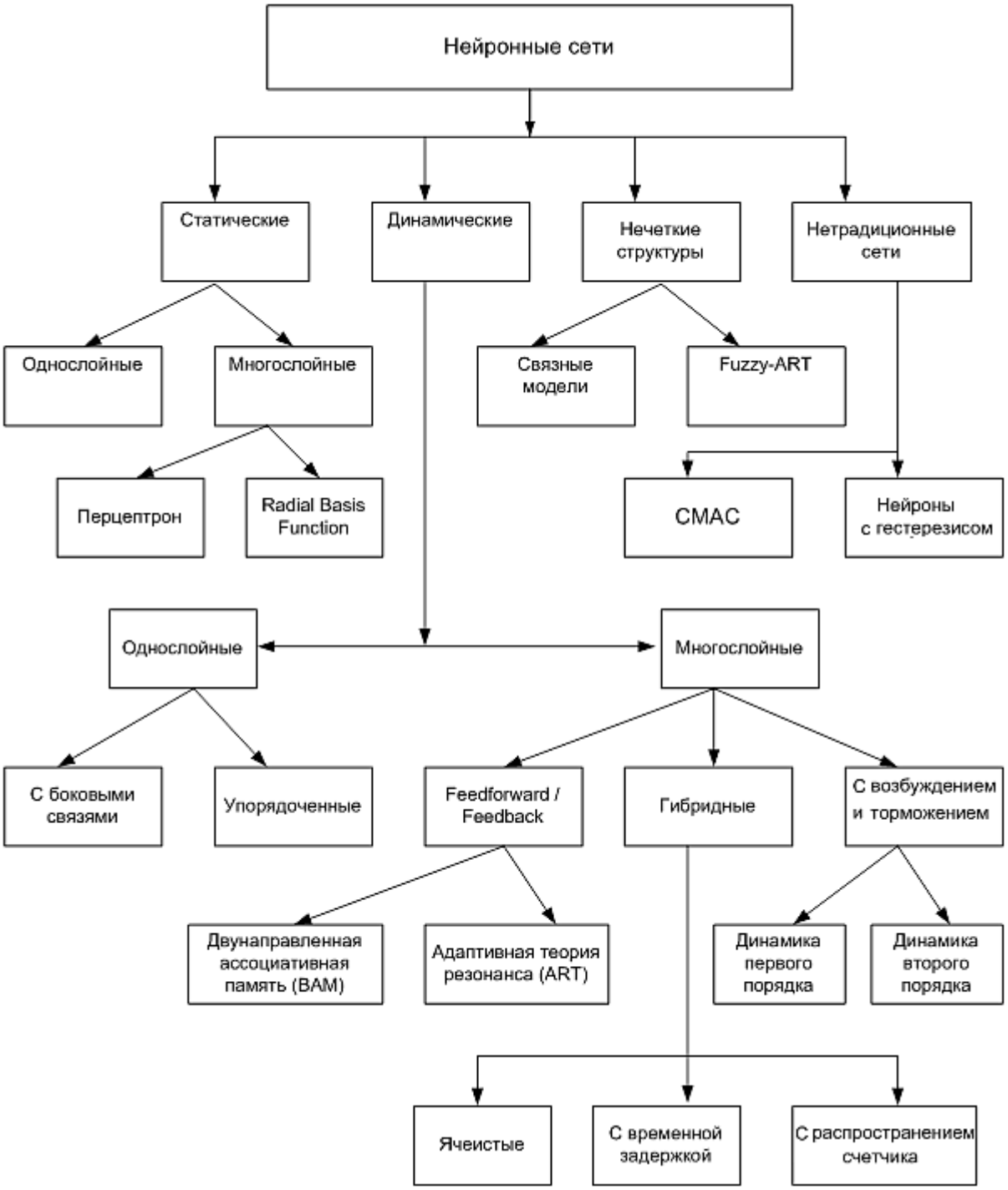


Рис. 4.1. Классификация нейронных сетей

Простейшая сеть состоит из группы нейронов, образующих слой, как показано в правой части рис. 4.2. Отметим, что вершины-круги слева служат лишь для распределения входных сигналов. Они не выполняют каких-либо вычислений и поэтому не будут считаться слоем. По этой причине они обозначены кругами, чтобы отличать их от вычисляющих нейронов, обозначенных квадратами. Каждый эле-

мент из множества входов X отдельным весом соединен с каждым искусственным нейроном. А каждый нейрон выдает взвешенную сумму входов в сеть. В искусственных и биологических сетях многие соединения могут отсутствовать, все соединения показаны в целях общности. Могут иметь место также соединения между выходами и входами элементов в слое.

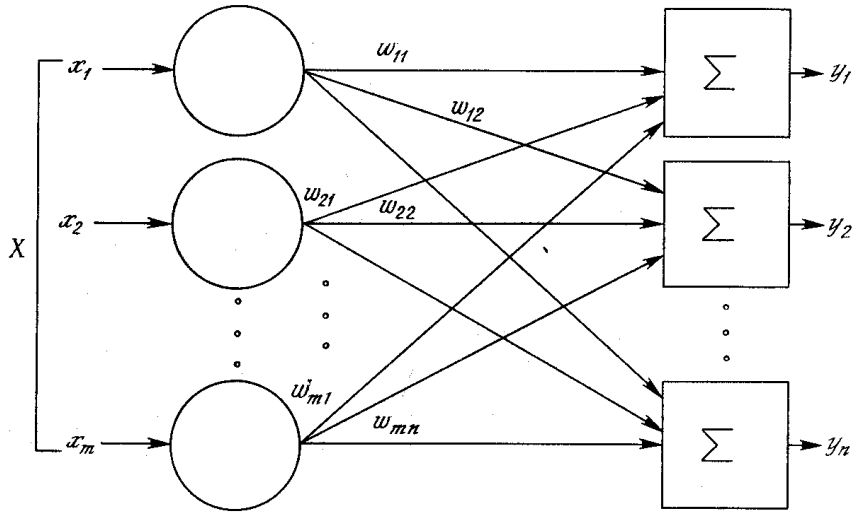


Рис. 4.2. Однослойная нейросеть

Удобно считать веса элементами матрицы *W*. Матрица имеет *m* строк и *n* столбцов, где *m* – число входов, а *n* – число нейронов. Например, w_{22} – это вес, связывающий второй вход со вторым нейроном. Таким образом, вычисление выходного вектора *N*, компонентами которого являются выходы *OUT* нейронов, сводится к матричному умножению *N* = *XW*, где *N* и *X* – вектор-строки.

Как правило, передаточные функции всех нейронов в сети фиксированы, а веса являются параметрами сети и могут изменяться. Некоторые входы нейронов помечены как внешние входы сети, а некоторые выходы – как внешние выходы сети.

При подаче любых чисел на входы сети получается какой-то набор чисел на выходах сети. Таким образом, работа нейросети состоит в преобразовании входного вектора в выходной вектор, причем это преобразование задается весами сети. Практически любую задачу можно свести к задаче, решаемой нейросетью.

Более крупные и сложные нейронные сети обладают, как правило, и большими вычислительными возможностями. Хотя созданы сети всех конфигураций, какие только можно себе представить, послойная организация нейронов копирует слоистые структуры определен-

ных отделов мозга. Оказалось, что такие многослойные сети обладают бóльшими возможностями, чем однослойные, и в последние годы были разработаны алгоритмы для их обучения.

Нейроны расположены в несколько слоев. Нейроны первого слоя получают входные сигналы, преобразуют их и через точки ветвления передают нейронам второго слоя. Далее срабатывает второй слой и так далее до *к*-го слоя, который выдает выходные сигналы для интерпретатора и пользователя. Если не оговорено противное, то каждый выходной сигнал *i*-го слоя подается на вход всех нейронов *i*+1-го слоя.

Число нейронов в каждом слое может быть любым и никак заранее не связано с количеством нейронов в других слоях. Теоретически число слоев может быть произвольным, однако фактически оно ограничено ресурсами компьютера или специализированной микросхемы, на которых обычно реализуется НС. Чем сложнее НС, тем масштабнее задачи, подвластные ей (рис. 4.3; 4.4).

Рис. 4.3. Процесс распространения информации в НС

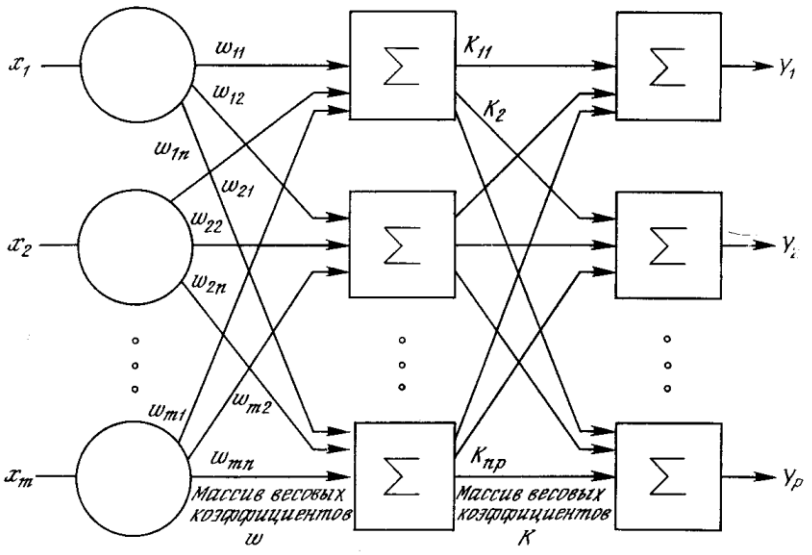


Рис. 4.4. Двуслойная нейронная сеть

Стандартный способ подачи входных сигналов: все нейроны первого слоя получают каждый входной сигнал.

Многослойные сети не могут привести к увеличению вычислительной мощности по сравнению с однослойной сетью лишь в том случае, если активационная функция между слоями будет нелинейной. Вычисление выхода слоя заключается в умножении входного вектора на первую весовую матрицу с последующим умножением (если отсутствует нелинейная активационная функция) результирующего вектора на вторую весовую матрицу: (XW1)W2.

Так как умножение матриц ассоциативно, то: X(W1W2).

Это показывает, что двухслойная линейная сеть эквивалентна одному слою с весовой матрицей, равной произведению двух весовых матриц. Следовательно, любая многослойная линейная сеть может быть заменена эквивалентной однослойной сетью. Однослойные сети весьма ограниченны по своим вычислительным возможностям. Таким образом, для расширения возможностей сетей по сравнению с однослойной сетью необходима нелинейная активационная функция.

Сети более общего вида, имеющие соединения от выходов к входам, называются сетями с обратными связями. У сетей без обратных связей нет памяти, их выход полностью определяется текущими входами и значениями весов. В некоторых конфигурациях сетей с обратными связями предыдущие значения выходов возвращаются на входы; выход, следовательно, определяется как текущим входом, так и предыдущими выходами. По этой причине сети с обратными связя-

ми могут обладать свойствами, сходными с кратковременной человеческой памятью, сетевые выходы частично зависят от предыдущих входов.

Особое распространение получили трехслойные сети, в которых каждый слой имеет свое наименование: первый – входной, второй – скрытый, третий – выходной.

В дальнейшем будут подробно рассмотрены некоторые виды как статических, так и динамических сетей.

Выбор структуры НС осуществляется в соответствии с особенностями и сложностью задачи. Для решения некоторых отдельных типов задач уже существуют оптимальные на сегодняшний день конфигурации. Если же задача не может быть сведена ни к одному из известных типов, разработчику приходится решать сложную проблему синтеза новой конфигурации. При этом он руководствуется несколькими основополагающими принципами: возможности сети возрастают с увеличением числа ячеек сети, плотности связей между ними и числом выделенных слоев. Введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о динамической устойчивости сети; сложность алгоритмов функционирования сети (в том числе, например, введение нескольких типов синапсов – возбуждающих, тормозящих и др.) также способствует усилению мощи НС. Вопрос о необходимых и достаточных свойствах сети для решения того или иного рода задач представляет собой целое направление нейрокомпьютерной науки. Так как проблема синтеза НС сильно зависит от решаемой задачи, дать общие подробные рекомендации затруднительно. В большинстве случаев оптимальный вариант получается на основе интуитивного подбора.

Очевидно, что процесс функционирования НС, то есть сущность действий, которые она способна выполнять, зависит от величин синаптических связей, поэтому, задавшись определенной структурой НС, отвечающей какой-либо задаче, разработчик сети должен найти оптимальные значения всех переменных весовых коэффициентов (некоторые синаптические связи могут быть постоянными). Этот этап называется обучением НС, и от того, насколько качественно он будет выполнен, зависит способность сети решать поставленные перед ней проблемы во время эксплуатации. На этапе обучения кроме параметра качества подбора весов важную роль играет время обучения. Как правило, эти два параметра связаны обратной зависимостью и их приходится выбирать на основе компромисса.

Оценка состояния систем управления предполагает использование специальных устройств и алгоритмов, предназначенных для определения неисправностей. Исследование влияния нейросети на диагностирование электропривода было проведено на базе ИНС с латеральным торможением – метод Гроссберга, сеть встречного распространения. Основной принцип работы нейронной сети заключается в настройке параметров нейрона таким образом, чтобы поведение сети соответствовало некоторому желаемому поведению. Регулируя веса или параметры смещения, можно обучить сеть выполнять конкретную работу, возможно, так же, что сеть сама будет корректировать свои параметры, чтобы достичь требуемого результата. Процесс обучения – это процесс подгонки параметров нейронной сети к той модели процесса или явления, которая реализуется нейронной сетью. При помощи регулирования параметров обучения исследуемая система способна самонастраиваться (самообучаться) на оптимальные режимы в процессе эксплуатации (или на начальном этапе обучения).

Далее приведена таблица основных алгоритмов настройки НС (табл. 4.1).

Таблица 4.1

ОСНОВНЫЕ АЛГОРИТМЫ НЕЙРОННЫХ СЕТЕЙ

Английское название	**Русское название**
Simulated annealing	Обучение обжигом
ART-1 network	Искусственный резонанс-1
BAM network	Двунаправленная автоассоциативная память
Boltzman machine	Больцманово обучение
Back Propagation	Обратное распространение
Counter Propagation	Сеть встречного распространения
Feed-Forward MAXNET	Сеть поиска максимума с прямыми связями
Gaussian Classifier	Гауссов классификатор
Genetic Algorithm	Генетический алгоритм
Hamming Network	Сеть Хэмминга
Hopfield Network	Сеть Хопфилда
INSTAR Network	Входная звезда
Kohonen Network	Сеть Кохонена
MAXNET Network	Сеть поиска максимума
OUTSTAR Network	Выходная звезда
Delta-Bar-Delta Network	Delta-Bar-Delta сеть
Extended DBD Network	Расширенная DBD сеть
Radial Basis Function Network	Сеть радиального основания
Single Layer Perceptron	Однослойный персептрон

В изученной нейросети между нейронами одного слоя имеются постоянные тормозящие связи (латеральное торможение). Согласно заложенному алгоритму система относит образ регулируемого параметра к одному из классов в зависимости от того, на какой из запомненных образов он больше всего похож. Если входящий образ не соответствует ни одному из запомненных, создается новый класс путем его запоминания. Если найден класс с определенным допуском, соответствующий входному, то он модифицирует его так, чтобы стать еще больше похожим на входящий. При этом активируются нейроны сети, образуя определенные связи между слоями. Главная особенность систем на базе НС – способность к обучению во время проектирования или в процессе управления реальным объектом. По этому методу система способна обучаться, определяя порядок поиска соответствия входного образа своему прототипу. А решение прекратить поиск зависит от критерия сходства k ($0<k<1$), который контролирует степень подобия и определяется порогом распознавания ρ. При фиксированном пороге сеть автоматически масштабирует свою чувствительность к образам различной сложности. Если входной образ сложный, то система пренебрегает небольшим различиями между входными сигналами и прототипом, а если входной образ содержит мало активных нейронов, то такое же несовпадение может привести к генерации сигнала сброса. Изменение порога чувствительности (например, по результатам работы системы) позволяет регулировать параметры сходства и детально анализировать входной образ.

Таким образом, задача диагностики электромеханического устройства на основе нейросетей сводится к исследованию изменений какого-либо параметра устройства, занесению в базу знаний информации об исправности и неисправности устройства по исследуемому параметру, сопоставление заложенной и полученной информации и формирование предложений по дальнейшей эксплуатации устройства.

Как было сказано выше, одним из свойств НС является способность к обучению, то есть эксперт-учитель, формируя нейросеть, за-

кладывает в ее БЗ свои знания, выводы об исправности или неисправности той или иной системы, и НС сама представляет заключения о состоянии системы. Этим значительно упрощается диагностика.

Нейросети также обладают высокой вычислительной мощностью и высокой отказоустойчивостью.

При исследовании влияния свойств нейросети на качество диагностики на вход нейронной сети подается вектор (матрица из двух строк) с исправным и неисправным сигналами с ЭМС. Результатом обучения является отнесение неисправных точек (отклонение от нормы) ко второму классу (2 – неисправность при изменении параметров). Проводится процесс обучения и сеть тестируется для просмотра результатов обучения.

Допустим, правильный результат отнесения неисправности к классам имеет вид:

Y = правильный результат

Columns 1 through 9

111111111

000000000

Columns 10 through 18

111111111

000000000

Columns 19 through 27

111110110

000001001

Columns 28 through 36

111000111

000111000

Columns 37 through 45

111111000

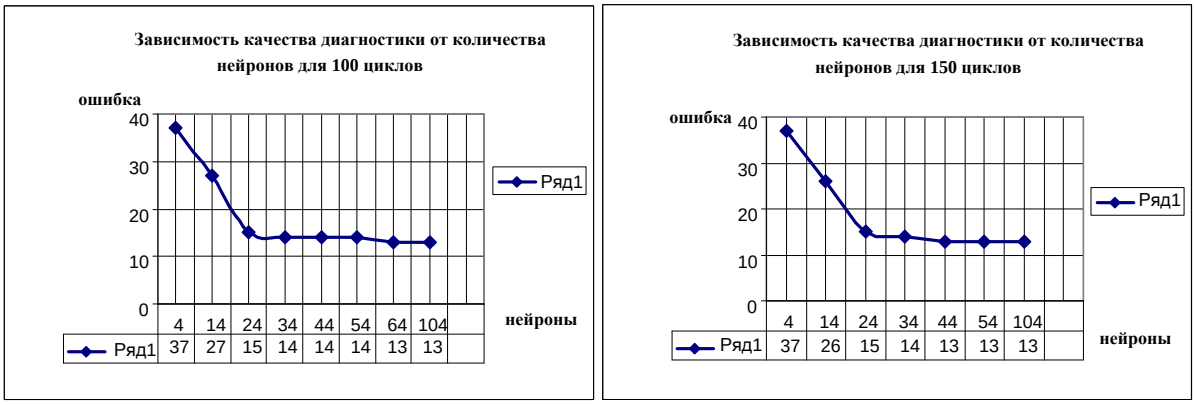
000000111

Первый класс – с 1-й по 23, 25, 26, 28-30, 34 – 42-ю точки.

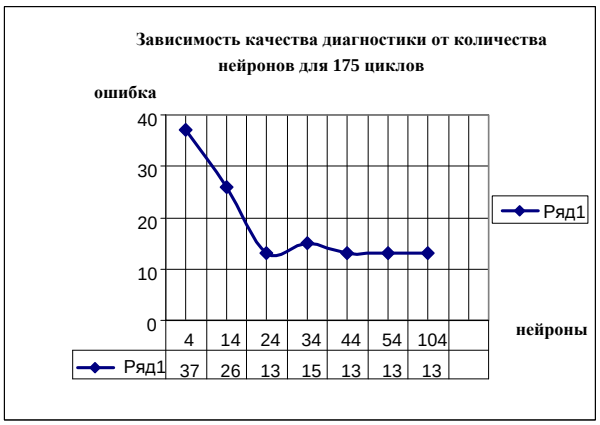
Второй класс – 24, 27, 31-33, 43 – 45-я точки.

Обучая и тестируя НС, изменяя параметры процесса обучения, добиваемся требуемого результата.

В результате исследований получаем, что с увеличением числа нейронов качество распознавания увеличивается (рис. 4.5).



а) *б)*



в)

Рис. 4.5. График зависимости качества диагностики от количества нейронов: *а* – 100 циклов обучения, *б* – 150 циклов обучения, *в* – 175 циклов обучения

Также при увеличении числа нейронов процесс обучения проходит за меньшее количество установившихся циклов, то есть ошибка обучения снижается до минимальной за меньшее количество циклов и колеблется в зависимости от самого процесса без резких скачков (рис. 4.6).

Рис. 4.6. График зависимости качества обучения от количества нейронов

Таким образом, изменяя параметры обучения, можно добиться требуемого качества распознавания, выбрав оптимальные характеристики процесса обучения (коэффициент обучения, количество циклов обучения, функция обучения), алгоритма настройки НС и ее архитектуры. Исследование и применение ИС необходимо и актуально. При решении различных задач, требующих точных моментальных выводов, нейросети показывают высокую вычислительную мощность и высокую отказоустойчивость, способность обучаться, самообучаться, адаптироваться к новым знаниям.

4.2. Моделирование системы с помощью MATLAB

Основной проблемой технической диагностики, основанной на нейросетях, является автоматизированный сбор, обработка измеренных характеристик объекта и выдача заключения о его состоянии.

Для решения проблем технической диагностики используются разнообразные компьютерные программные комплексы, основанные на различных математических методах. В данной работе рассматриваются вопросы диагностики привода с помощью системы MATLAB с пакетом расширения «Neural Network Toolbox».

При определении входных данных, необходимых для работы автоматической системы диагностики, основанной на нейронных сетях, анализируют переходные характеристики привода при различных неисправностях в нем (см. «Построение модели системы»). Набор входных данных, полученный при исследовании привода (вектор состояния привода), отражает минимум, по которому можно определить его состояние. Данные представляют собой последовательность числовых значений-координат переходной характеристики (в векторной форме), снятых через определенные, равные промежутки времени (рис. 4.7).

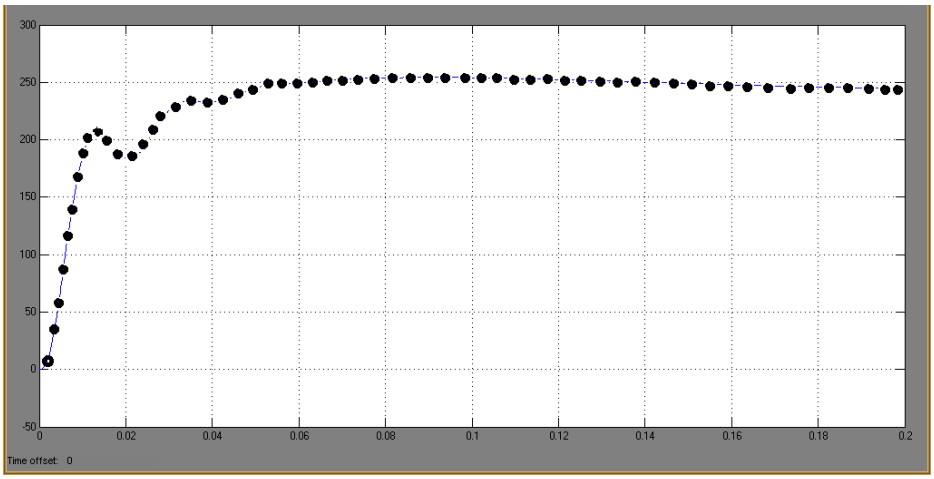


Рис. 4.7. Измеряемые данные

Набор входных данных (вектор состояния привода), полученный из координат переходной характеристики, вводится в компьютер для автоматической постановки диагноза.

При постановке задачи для обучения нейросетей исходят из того, что диагностическая система должна выбирать один из предполагаемых диагнозов из заданного набора на основе параметров привода, по которым производится оценка его состояния.

4.3. Сбор данных для нейронной сети

Набор данных представляет собой набор *наблюдений*, для которых указаны значения входных и выходных *переменных*. Первый вопрос, который нужно решить, – какие переменные использовать и сколько (и каких) наблюдений собрать. Предположим, что только пять переменных оказывают существенное влияние на работу привода:

1) момент нагрузки (M);
2) сопротивление якоря двигателя (R);
3) коэффициент в блоке широтно-импульсного модулятора (T_{shim});
4) коэффициент в блоке контура тока (K_t);
5) коэффициент в блоке контура скорости (K_s).

Нейронные сети могут работать с числовыми данными, лежащими в определенном ограниченном диапазоне. Это создает проблемы в случаях, когда данные имеют нестандартный масштаб, когда в них имеются пропущенные значения и когда данные являются нечисловыми. В этой работе была, например, проблема с начальным значением обучающей выборки, которое во всех случаях было равно нулю. При поиске весов для нейронов нейросеть умножает обучающий пример на случайный весовой коэффициент, находит произведение и сравнивает с заданным результатом. В случае с нулем она не может подобрать правильный вес нейронам, так как при этом в любом случае произведение оказывалось равным нулю. Проблема «эффекта нуля» была решена введением блоков, увеличивающих все значения входных векторов на равные величины, то есть график смещался вверх на определенную величину по оси ординат (рис. 4.8).

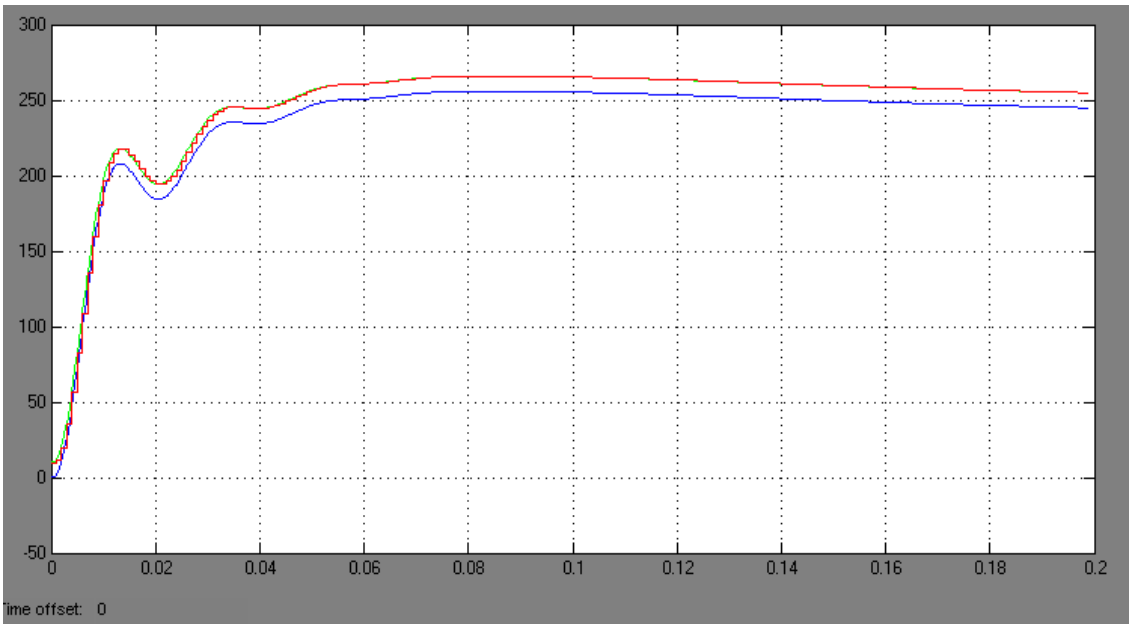


Рис. 4.8. Переходная характеристика модели в процессе сбора данных

Вопрос о том, сколько наблюдений нужно иметь для обучения сети, часто оказывается непростым. Известен ряд эвристических правил, увязывающих число необходимых наблюдений с размерами сети (простейшее из них гласит, что число наблюдений должно быть в десять раз больше числа связей в сети). На самом деле это число зависит также от сложности (заранее неизвестной) того отображения, которое нейронная сеть стремится воспроизвести. С ростом количества переменных количество требуемых наблюдений растет нелинейно, так что уже при довольно небольшом (например, пятьдесят) числе переменных может потребоваться огромное число наблюдений.

Во многих реальных задачах приходится иметь дело с не вполне достоверными данными. Значения некоторых переменных могут быть искажены шумом или частично отсутствовать. Пакет «ST Neural Networks» имеет специальные средства работы с пропущенными значениями (они могут быть заменены на среднее значение этой переменной или на другие ее статистики). Кроме того, нейронные сети в целом устойчивы к шумам. Однако у этой устойчивости есть предел. Например, выбросы, т.е. значения, лежащие очень далеко от области нормальных значений некоторой переменной, могут исказить результат обучения. В таких случаях лучше всего постараться обнаружить и удалить эти выбросы (либо удалив соответствующие наблюдения, либо преобразовав выбросы в пропущенные значения).

В рассматриваемом случае для формирования обучающего множества было взято 200 координатных точек для каждого графика. Из графика на рис. 4.8 видно преобразование графика, т.е. сначала график смещается вверх по оси ординат на определенную величину без изменения формы кривой, а затем он из непрерывного преобразуется в ступенчатый с помощью блока дискретизации.

После того как определено число слоев и число элементов в каждом из них, нужно найти значения для весов и порогов сети, которые бы минимизировали ошибку прогноза, выдаваемого сетью. Именно для этого служат *алгоритмы обучения*. С использованием собранных исторических данных веса пороговые значения автоматически корректируются с целью минимизировать эту ошибку. По сути, этот процесс представляет собой подгонку модели, которая реализуется сетью, к имеющимся обучающим данным. Ошибка для конкретной конфигурации сети определяется путем прогона через сеть всех имеющихся наблюдений и сравнения реально выдаваемых выходных значений с желаемыми (целевыми) значениями. Все такие разности суммируются в так называемую *функцию ошибок*, значение которой и есть ошибка сети. В качестве функции ошибок чаще всего берется сумма квадратов ошибок, т.е. когда все ошибки выходных элементов для всех наблюдений возводятся в квадрат и затем суммируются. При работе с пакетом «ST Neural Networks» пользователю выдается так называемая среднеквадратичная ошибка (RMS). Описанная выше величина нормируется на число наблюдений и переменных, после чего из нее извлекается квадратный корень – это очень хорошая мера ошибки, усредненная по всему обучающему множеству и по всем выходным элементам.

Общая схема обучения нейросети включает несколько этапов.

Создается обучающая выборка (в данном случае из 420 элементов). Для этого в MATLAB создаем модель для сбора данных, как показано на рис. 4.9.

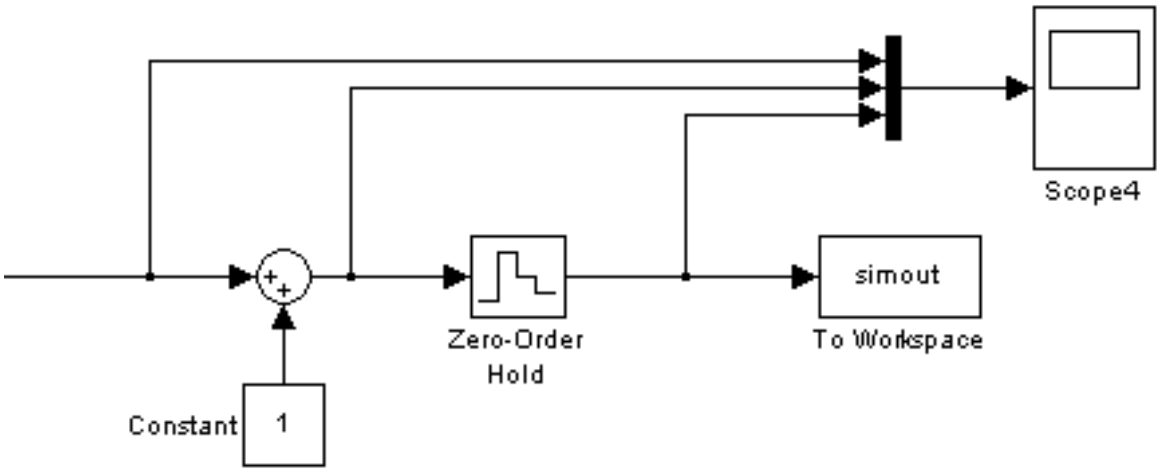


Рис. 4.9. Фрагмент модели для сбора обучающей информации

На этом рисунке изображены блок «Constant» и блок суммирования, которые прибавляют к исходному сигналу единицу для исключения «эффекта нуля». Блок «Zero-Order Hold» преобразует непрерывный сигнал в ступенчатый. Блок «To Workspace» передает координаты «ступеней» в рабочую область MATLAB и создает таблицу (вектор) данных из 200 элементов.

После записи первого вектора данных меняют переменную с помощью маски и снова записывают данные в рабочую область MATLAB. Результаты моделирования объединяются в одну матрицу с числом столбцов, равным числу элементов в обучающей выборке. Эта матрица называется матрицей входов. В данной работе фиксировались 20 векторов для каждого из 21 диагноза, которые мы собираемся ставить. Получилась матрица размером 420×200, т.е. 420 примеров для обучения по 200 элементов в каждом.

Для ускорения процесса сбора данных можно использовать модель, приведенную на рис. 4.10.

На данном рисунке изображены 20 параллельно работающих приводов (по числу обучающих векторов для каждой неисправности). В каждом блоке привода с помощью маски меняем значения только той переменной, которую исследуем (рис. 4.11). Далее запускаем моделирование и в рабочей области MATLAB получаем матрицу значений для одного из 21 состояния привода (рис. 4.12). Повторяем для другого состояния привода и так далее для 21 состояния в

соответствии с прил. 1. Получается 21 матрица размером 200 × 20, которые мы объединяем в одну общую с размерами 420 × 200. Эта матрица и есть матрица входов.

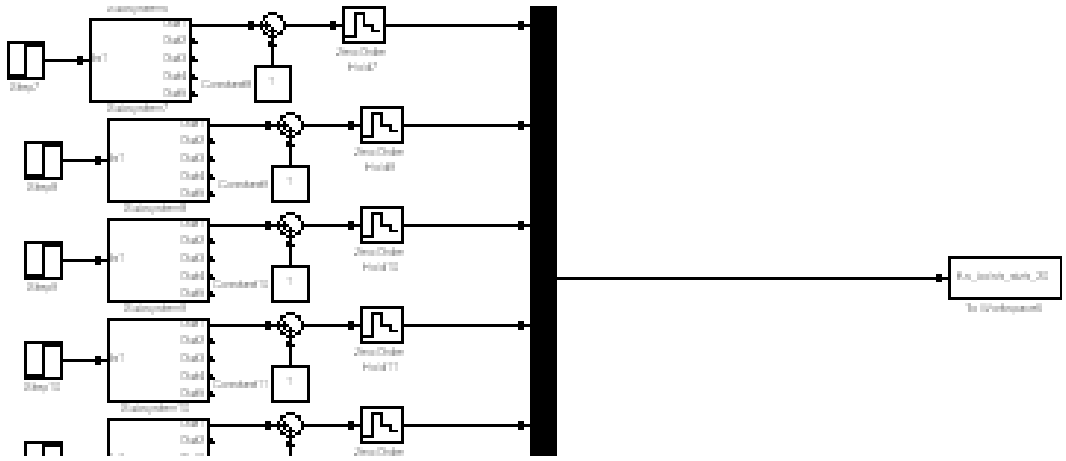


Рис. 4.10. Фрагмент модели для ускорения процесса сбора данных

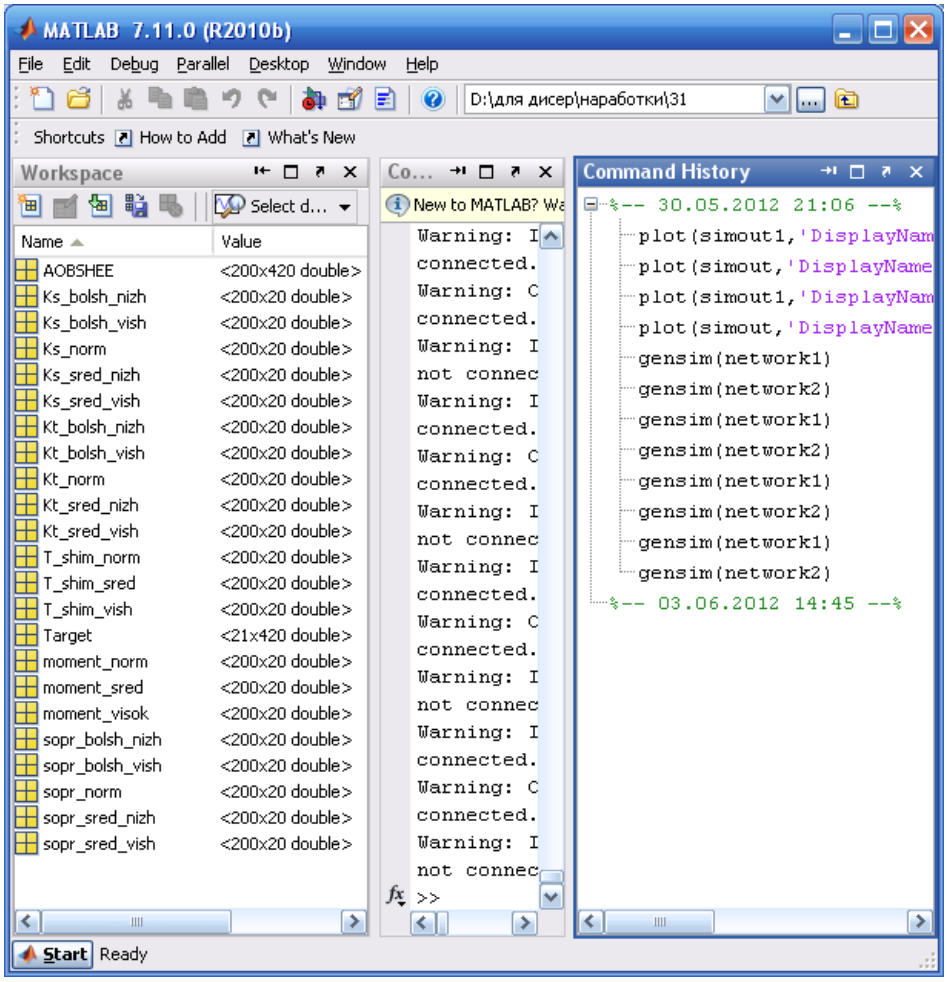


Рис. 4.11. Входные данные в рабочей области MATLAB

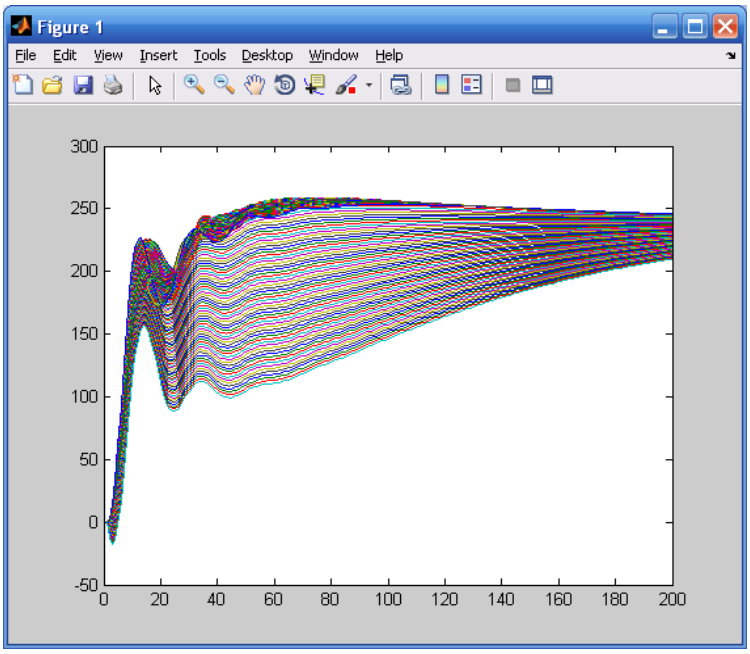


Рис. 4.12. Данные в виде графиков

Далее создается матрица целей в виде нулей и единиц. Каждому верно распознанному состоянию будет соответствовать единица на выходе нейросети. Все остальные выходы при этом принимают значение 0. Если нейросеть распознала первое состояние привода, то на её первом выходе должна получиться единица, на остальных выходах – нули. Если она распознала второе состояние, единица получается на втором выходе, а на остальных – нули. Число столбцов должно совпадать с числом столбцов матрицы входов. То есть каждому столбцу матрицы входов будет соответствовать столбец матрицы целей. В данной работе получилась матрица размером 420×21 (рис. 4.13; 4.14).

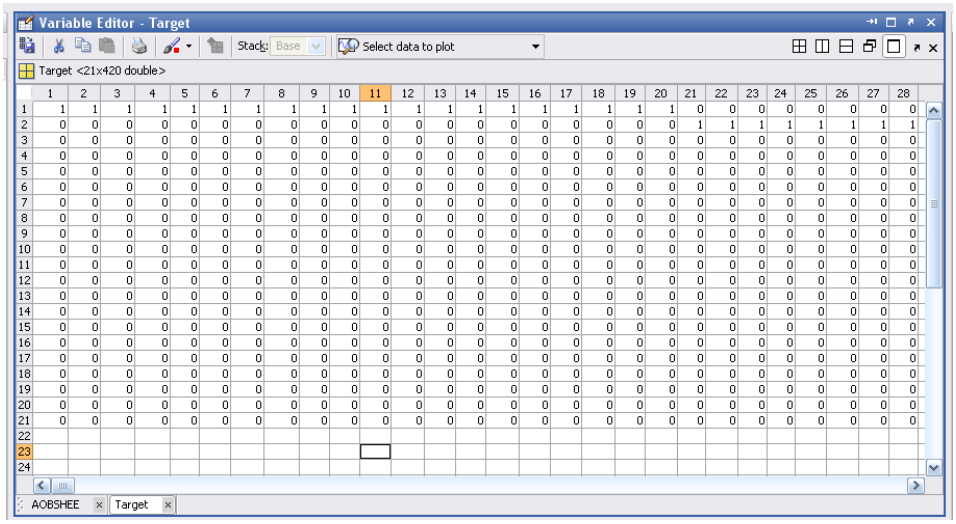


Рис. 4.13. Матрица целей в рабочей области MATLAB

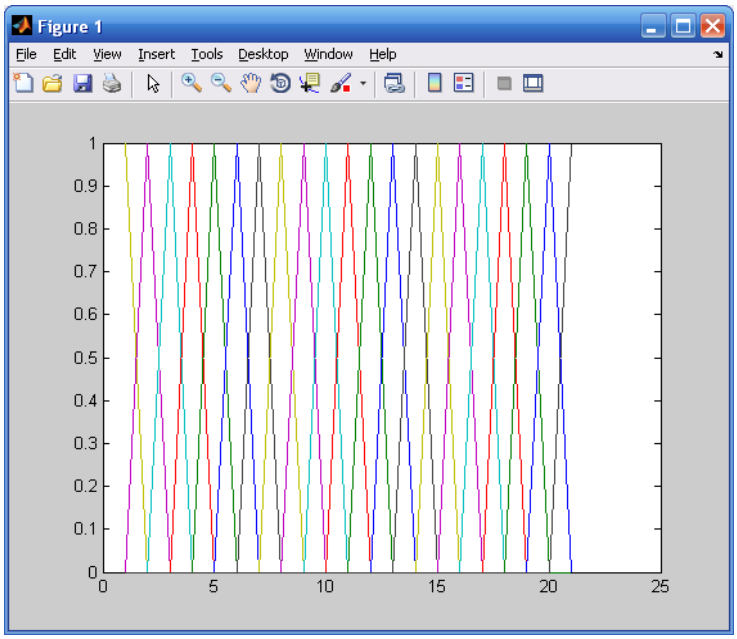


Рис. 4.14. Цели в виде графиков

4.4. Выбор нейросети для конкретной задачи и проверки ее работы

После создания матрицы входов и матрицы целей можно приступать к созданию самой нейросети. В главном окне MATLAB в левом нижнем углу нужно нажать кнопку «Start» и выбрать «Toolbox – Neural Network Tool» (рис. 4.15).

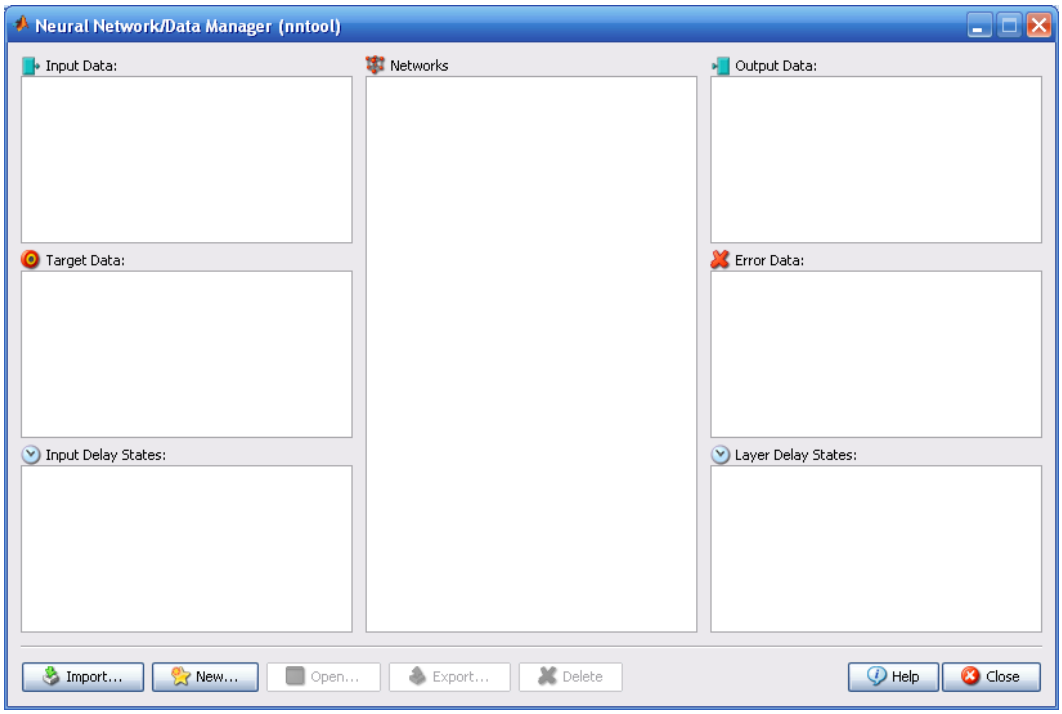


Рис. 4.15. Окно создания нейросетей

В открывшемся окне выбираем «Import» и загружаем в появившемся окне матрицу входов и матрицу целей. После того как матрицы загружены, в окне «Neural Network Tool»нажимаем кнопку «New» и открываем окно создания сети («Create Networkor Data»). В разделе «Network Properties» выбираем нужный нам тип сети, настраиваем количество слоев и вид функции активации. Так как в настоящее время создано очень много нейросетей с различными алгоритмами, подбирать сеть под нашу задачу будем методом их перебора, пока не найдем сеть, удовлетворяющую нашим запросам. Выбор осуществляем из сетей, предназначенных для классификации образов.

1. Создаем сеть **Feed-forward distributed time delay**. Загружаем для нее данные для обучения, полученные на предыдущем этапе работы (рис. 4.16).

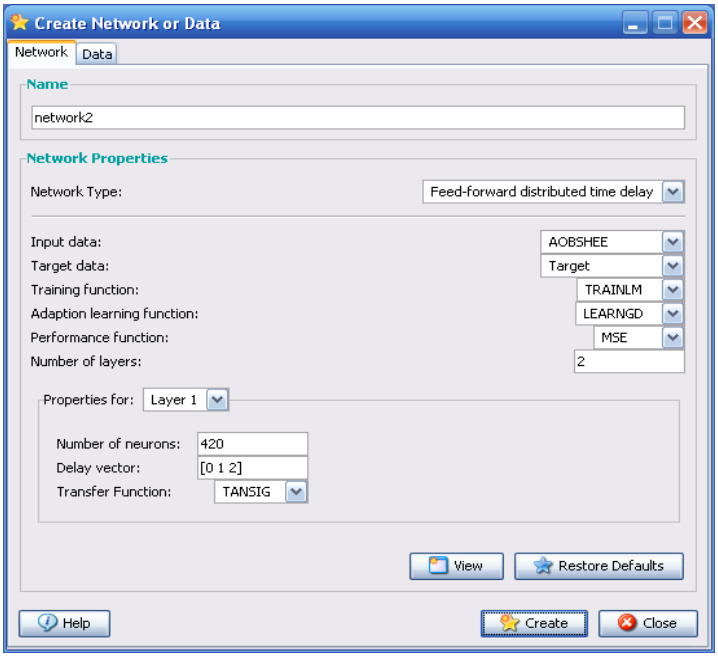


Рис. 4.16. Окно настройки создаваемой нейросети

• Нажимаем кнопку «View», чтобы посмотреть структуру создаваемой сети (рис. 4.17).

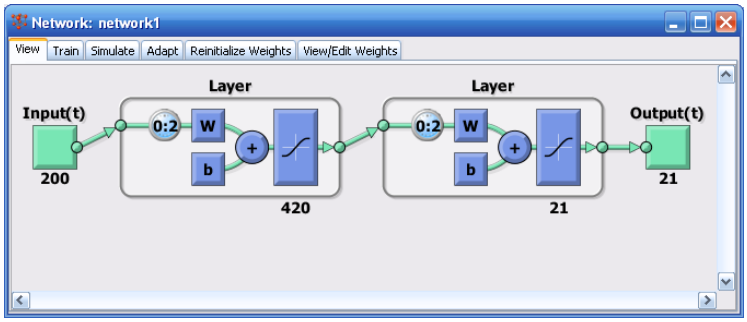


Рис. 4.17. Структура создаваемой сети

• Нажимаем кнопку «Create», чтобы создать сеть. Полученную сеть экспортируем в рабочую область MATLAB, а затем в «Simulink».

Для проверки работоспособности нейросети необходимо создать модель.

Эта модель должна включать в себя:

- электропривод;
- блоки предварительной обработки сигнала;
- собственно блок нейросети;
- блоки обработки данных, получаемых с выхода нейросети;
- блоки вывода результатов в форме, удобной для дальнейшего использования человеком или ЭВМ.

• Запускаем моделирование, чтобы проверить работоспособность сети. Изменяя входные данные и наблюдая за выходом нейронной сети, сделаем вывод о ее пригодности для решения данной задачи (рис. 4.18).

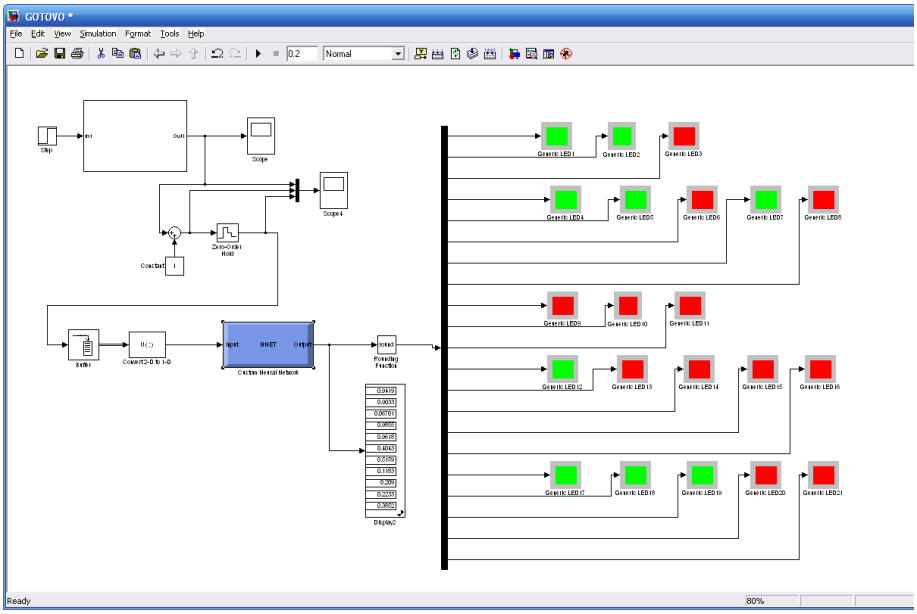


Рис. 4.18. Проверка работоспособности сети

Данная нейросеть не может быть применена в нашем случае, так как целевые значения не соответствуют входным данным.

2. Создаем сеть **Generalized regression**. Загружаем для нее данные для обучения (рис. 4.19).

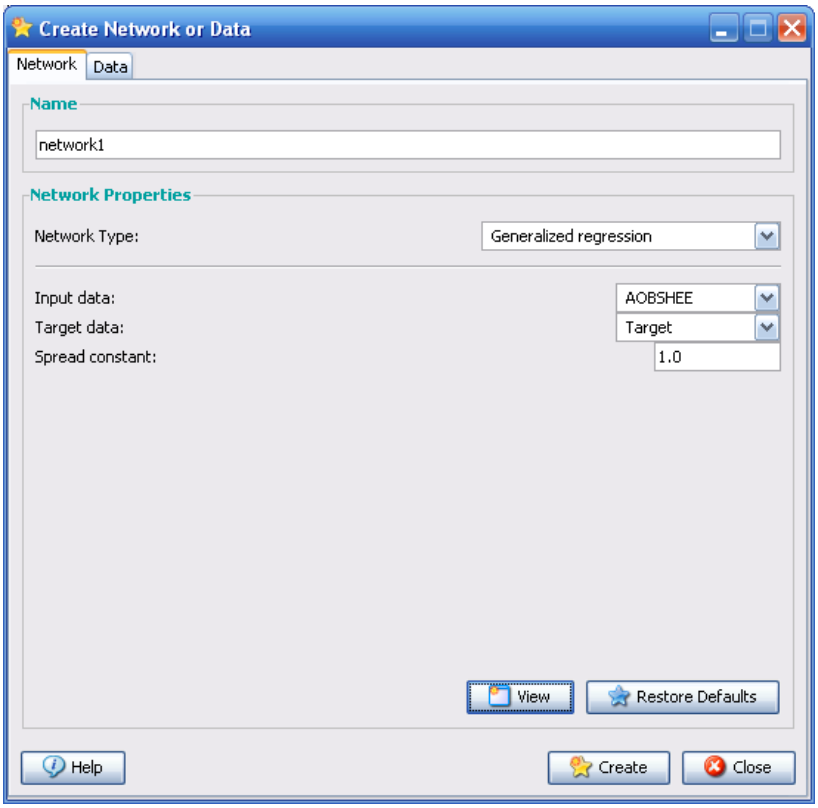


Рис. 4.19. Окно настройки создаваемой нейросети

• Нажимаем кнопку «View», чтобы посмотреть структуру создаваемой сети (рис. 4.20).

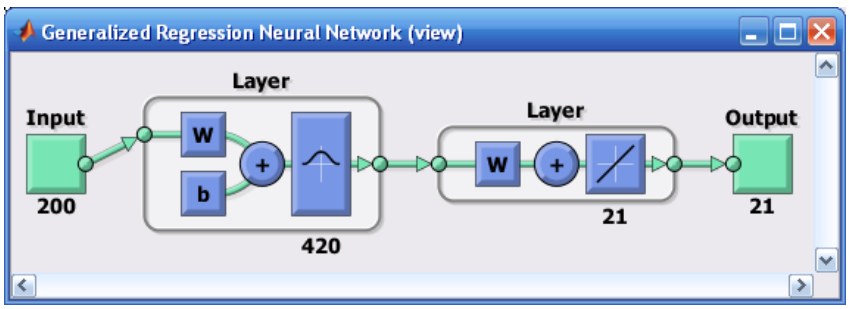


Рис. 4.20. Структура создаваемой сети

• Нажимаем кнопку «Create», чтобы создать сеть. Полученную сеть экспортируем в рабочую область MATLAB, а затем в «Simulink».

• Запускаем моделирование, чтобы проверить работоспособность сети. Изменяя входные данные и наблюдая за выходным сигналом нейронной сети, делаем вывод о ее пригодности для решения данной задачи (рис. 4.21).

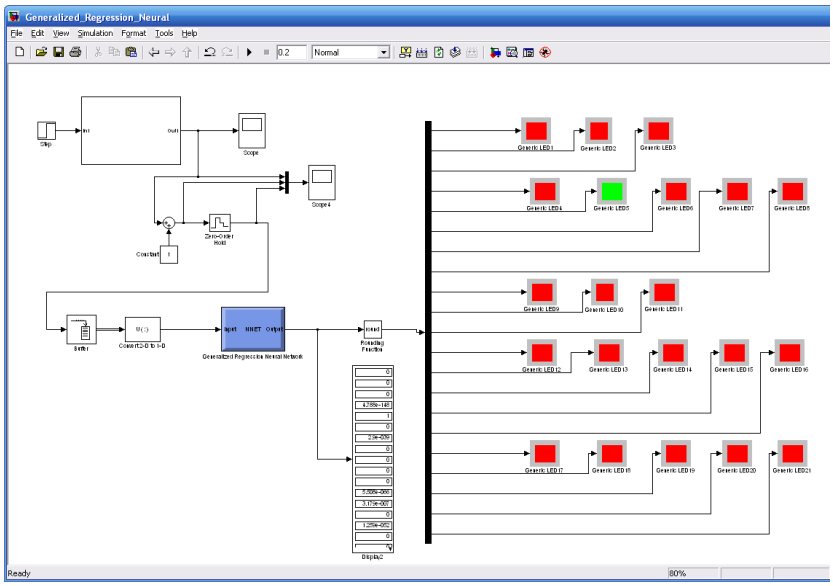


Рис. 4.21. Проверка работоспособности сети

Данная нейросеть может быть применена в нашем случае, так как сеть правильно воспроизводит целевые значения в соответствии с входными данными. И все-таки сеть работает с некоторыми отклонениями, которые можно исключить введением после нейросети блока округления выходных данных.

3. Создаем сеть **Layer Recurrent.** Загружаем для нее данные для обучения (рис. 4.22).

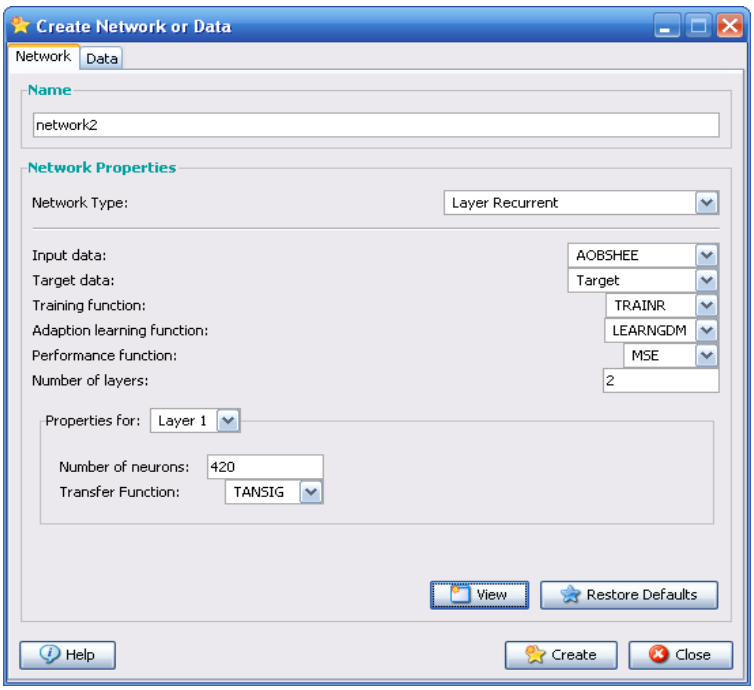


Рис. 4.22. Окно настройки создаваемой нейросети

• Нажимаем кнопку «View», чтобы посмотреть структуру создаваемой нейронной сети (рис. 4.23).

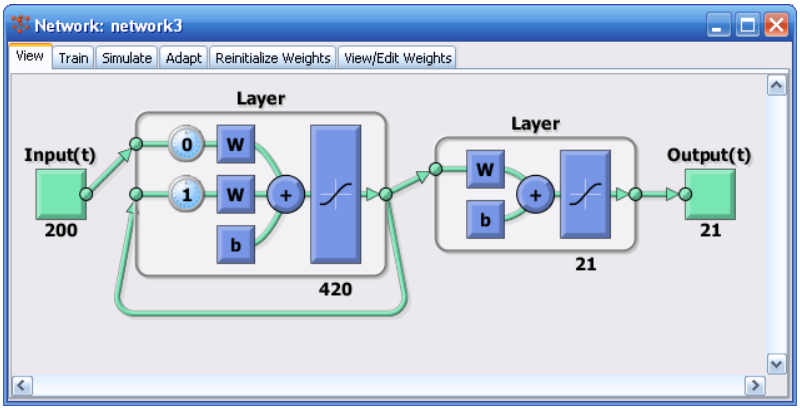


Рис. 4.23. Структура создаваемой сети

• Нажимаем кнопку «Create», чтобы создать сеть. Полученную сеть открываем, во вкладке «Train» настраиваем параметры обучения и нажимаем«Train Network» для обучения сети (рис. 4.24).

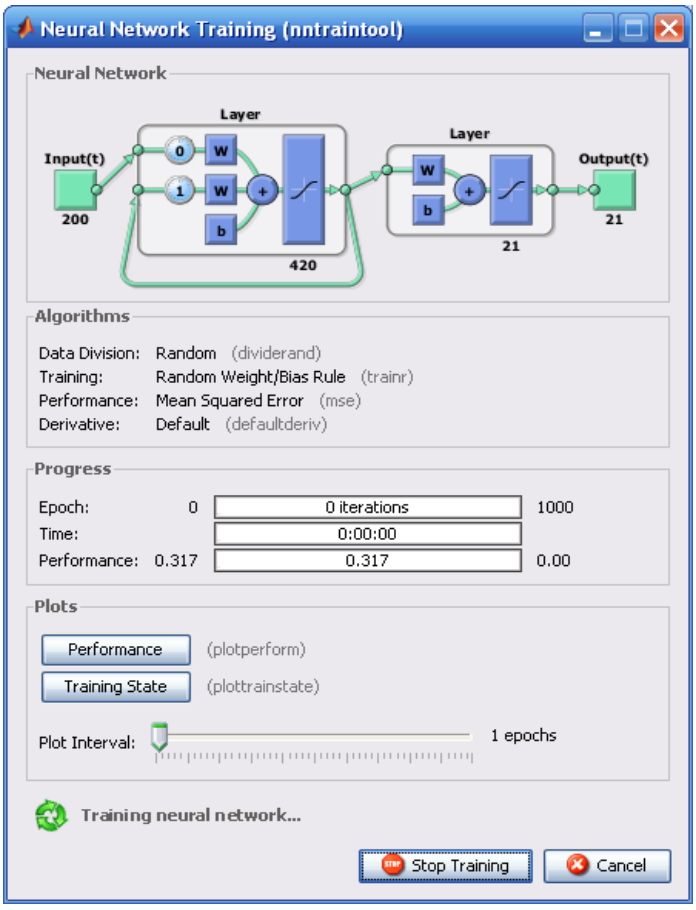


Рис. 4.24. Окно тренировки нейросети

Данная сеть непригодна для дальнейшего использования, так как обучается очень долго и ошибка обучения не достигает максимального значения.

4. Создаем сеть **Linear layer (design)**. Загружаем для нее данные для обучения (рис. 4.25).

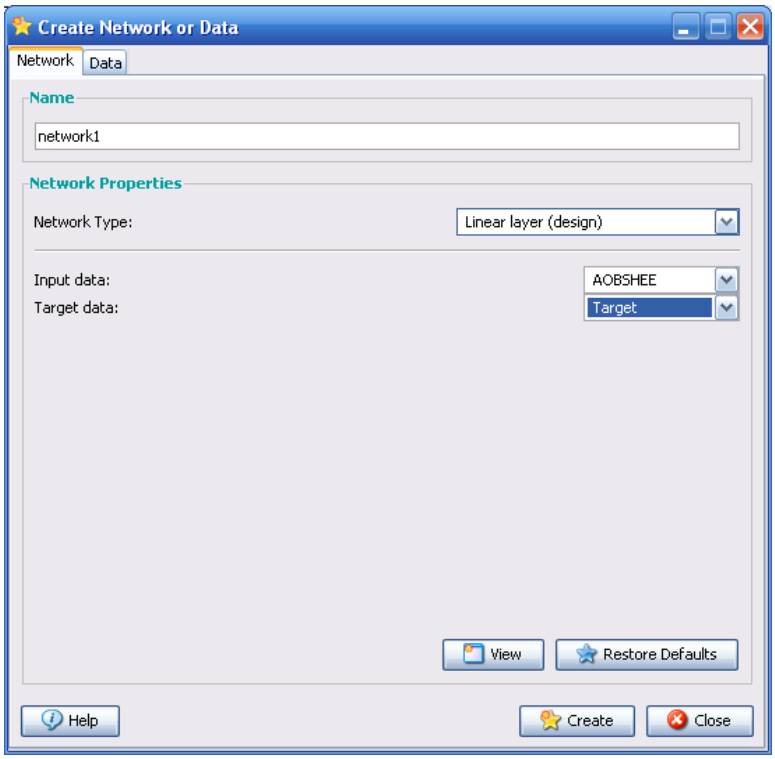


Рис. 4.25. Окно настройки создаваемой нейросети

- Нажимаем кнопку «View», чтобы посмотреть структуру создаваемой нейронной сети (рис. 4.26).

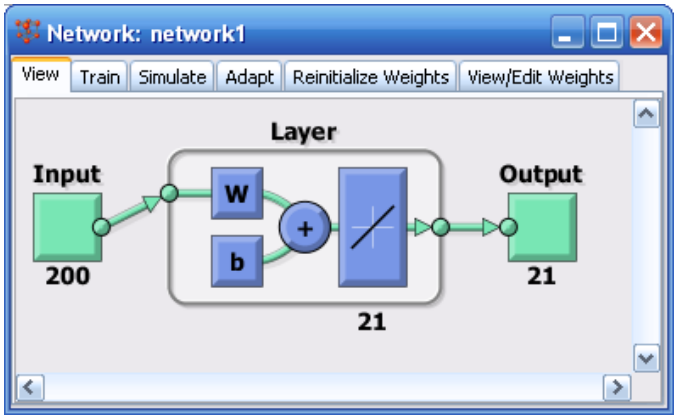


Рис. 4.26. Структура создаваемой сети

- Нажимаем кнопку «Create», чтобы создать сеть. Полученную сеть экспортируем в рабочую область MATLAB, а затем в «Simulink».

• Запускаем моделирование, чтобы проверить работоспособность сети. Изменяя входные данные и наблюдая за выходом с нейронной сети, делаем вывод о ее пригодности для решения данной задачи (рис. 4.27).

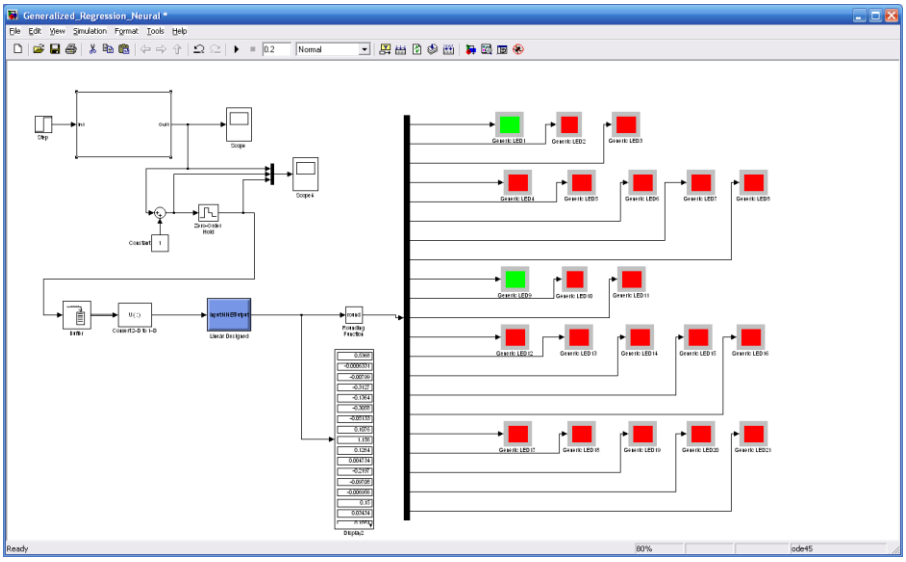


Рис. 4.27. Проверка работоспособности сети

Данная нейросеть не может быть применена в нашем случае, так как целевые значения не соответствуют входным данным.

5. Создаем сеть **Linear layer (train)**. Загружаем для нее данные для обучения (рис. 4.28).

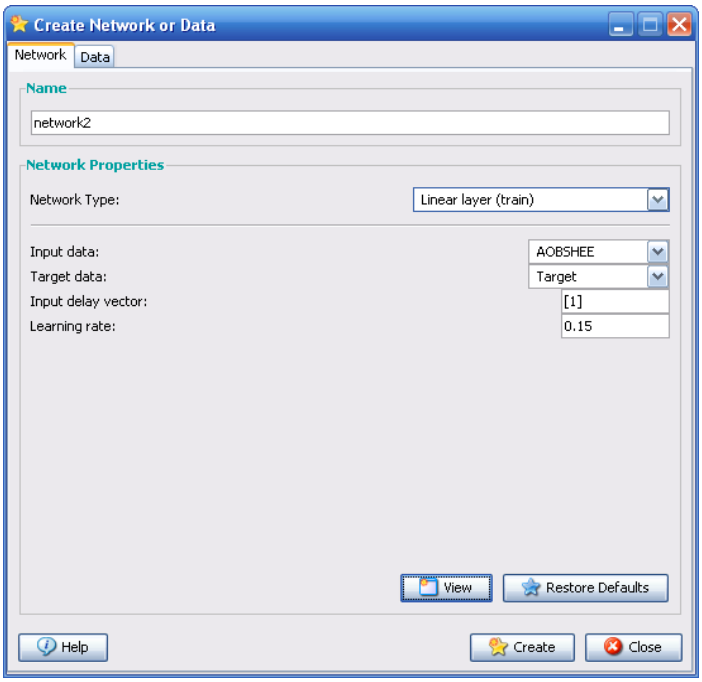


Рис. 4.28. Окно настройки создаваемой нейросети

• Нажимаем кнопку «View», чтобы посмотреть структуру создаваемой нейронной сети (рис. 4.29).

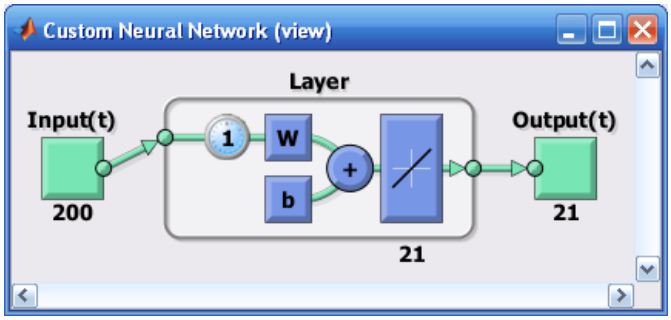


Рис. 4.29. Структура создаваемой сети

• Нажимаем кнопку «Create», чтобы создать сеть. Полученную сеть открываем, во вкладке «Train» настраиваем параметры обучения, нажимаем «Train Network» для обучения сети (рис. 4.30).

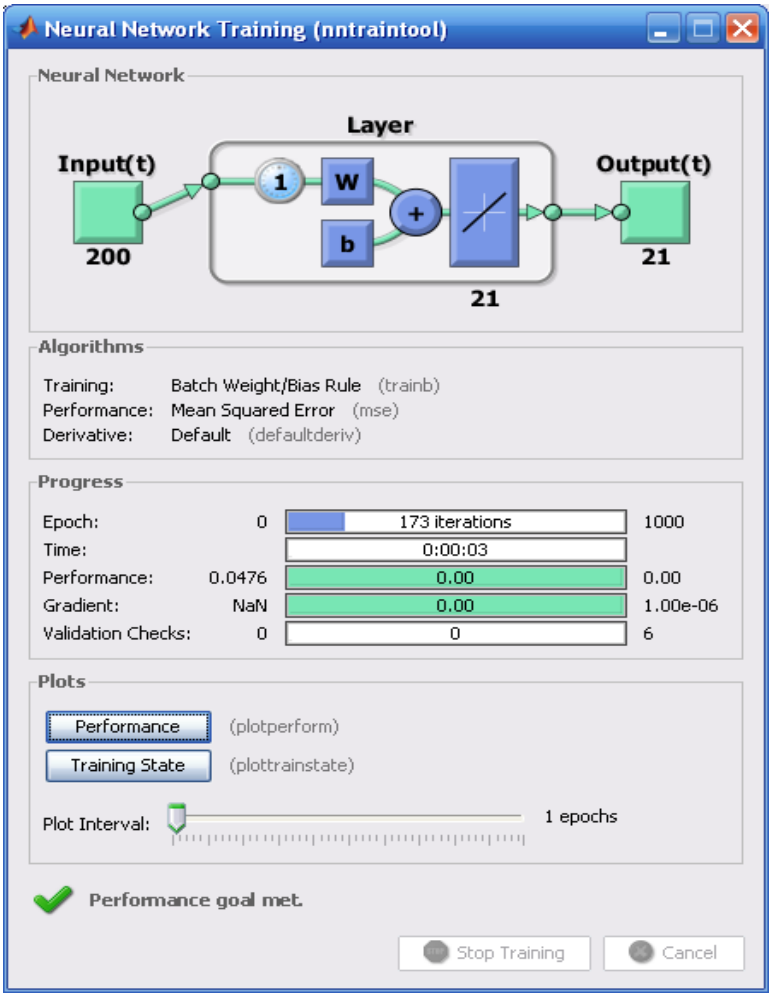


Рис. 4.30. Окно тренировки создаваемой нейросети

Обучение нейронной сети заканчивается через 173 прохода (рис. 4.31).

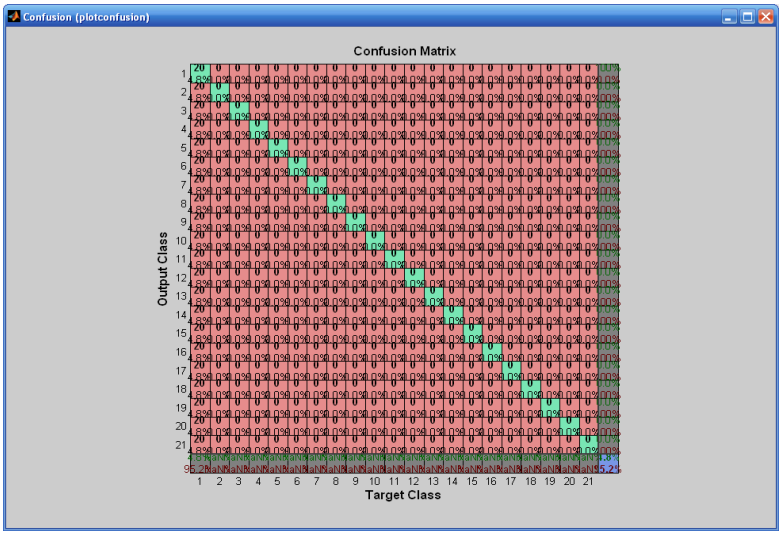


Рис. 4.31. График правильно распознанных сигналов

Но если проанализировать график ошибок, можно увидеть, что сеть научилась правильно распознавать только 4,8 % входных данных, что недопустимо. То есть данная сеть нам не подходит.

6. Создаем сеть **NARX**. Загружаем для нее данные для обучения (рис. 4.32).

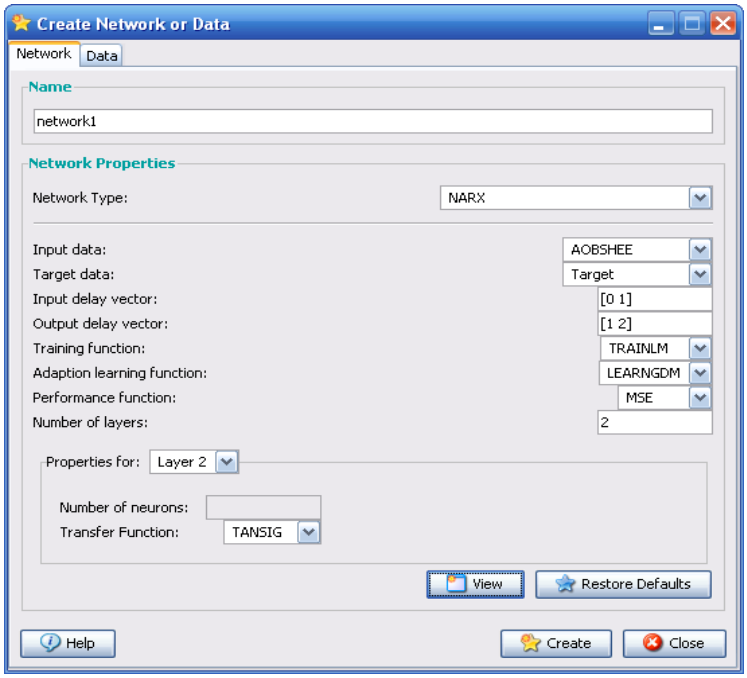


Рис. 4.32. Окно настройки создаваемой нейросети

- Нажимаем кнопку «View», чтобы посмотреть структуру создаваемой нейронной сети (рис. 4.33).

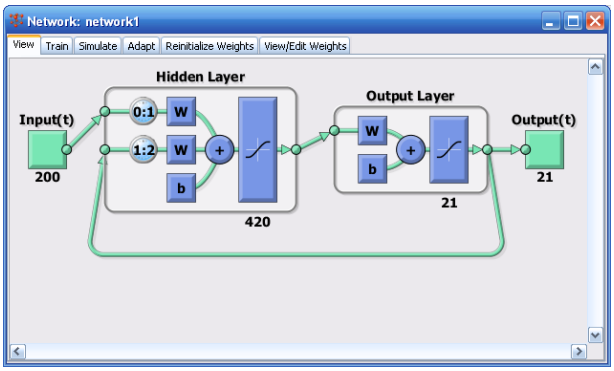


Рис. 4.33. Структура создаваемой сети

• Нажимаем кнопку «Create», чтобы создать сеть. Полученную сеть экспортируем в рабочую область MATLAB, а затем в «Simulink».

• Запускаем моделирование, чтобы проверить работоспособность сети. Изменяя входные данные и наблюдая за выходом нейронной сети, делаем вывод о ее пригодности для решения данной задачи (рис. 4.34).

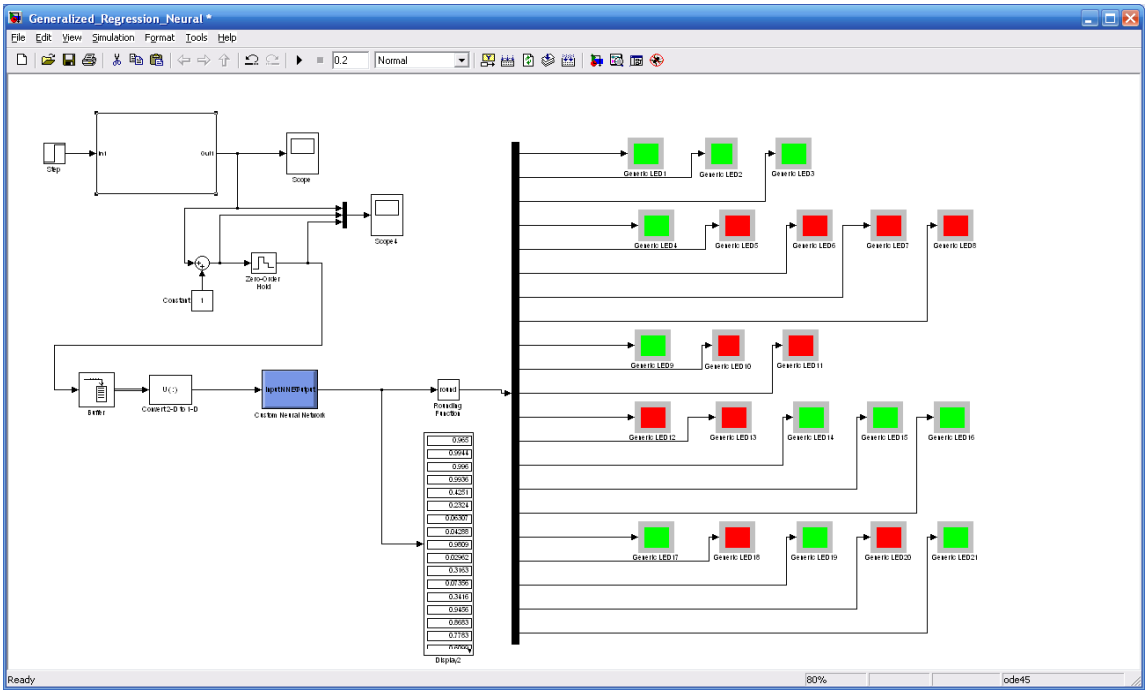


Рис. 4.34. Проверка работоспособности сети

Данная нейросеть не может быть применена в нашем случае, так как целевые значения не соответствуют входным данным.

7. Создаем сеть **Perceptron.** Загружаем для нее данные для обучения (рис. 4.35).

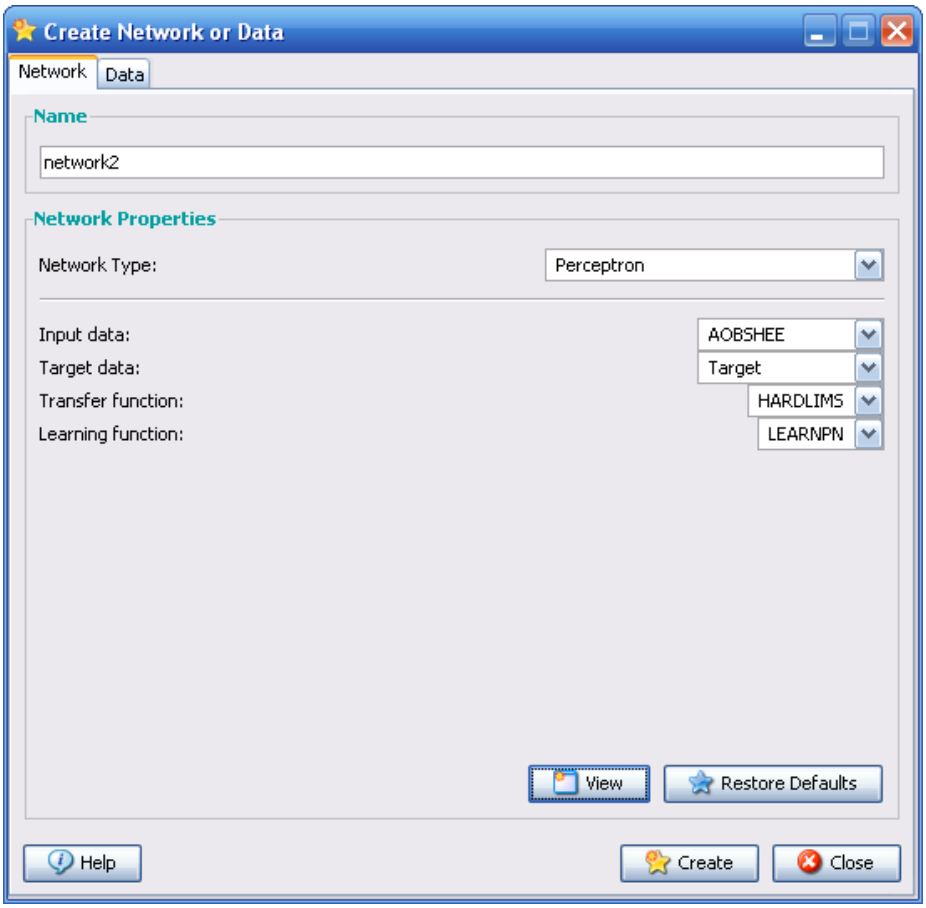


Рис. 4.35. Окно настройки создаваемой нейросети

• Нажимаем кнопку «View», чтобы посмотреть структуру создаваемой нейронной сети (рис. 4.36).

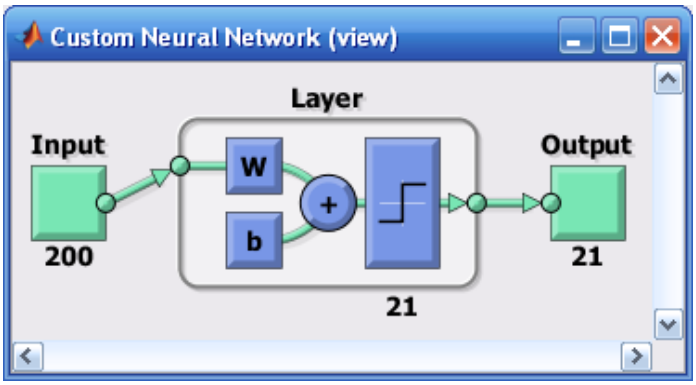


Рис. 4.36. Структура создаваемой сети

• Нажимаем кнопку «Create», чтобы создать сеть. Полученную сеть открываем, во вкладке «Train» настраиваем параметры обучения и нажимаем «Train Network» для обучения сети (рис. 4.37).

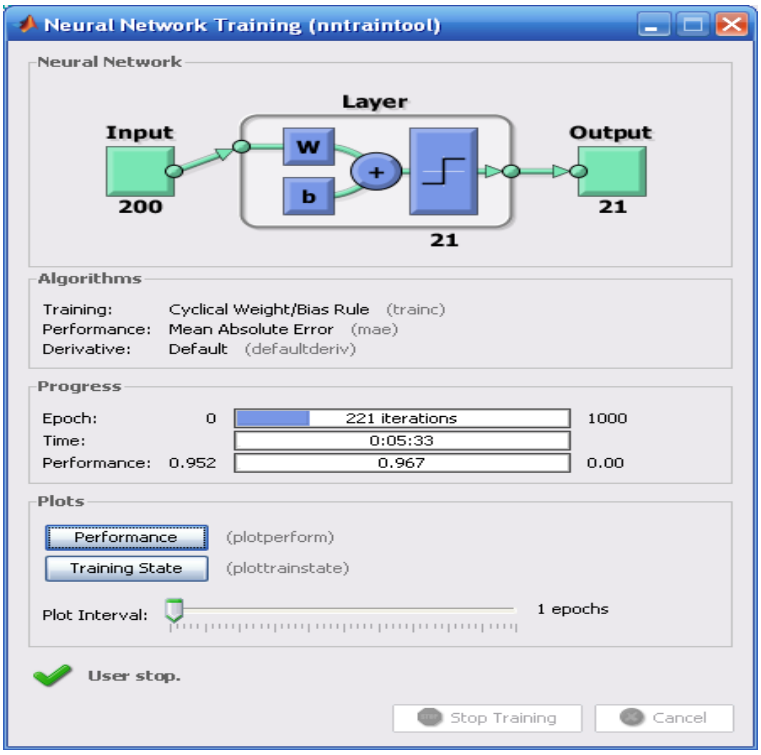


Рис. 4.37. Окно тренировки создаваемой нейросети

Данная сеть непригодна для дальнейшего использования, так как обучается очень долго и ошибка обучения не достигает максимального значения.

8. Создаем сеть **Probabilistic.** Загружаем для нее данные для обучения (рис. 4.38).

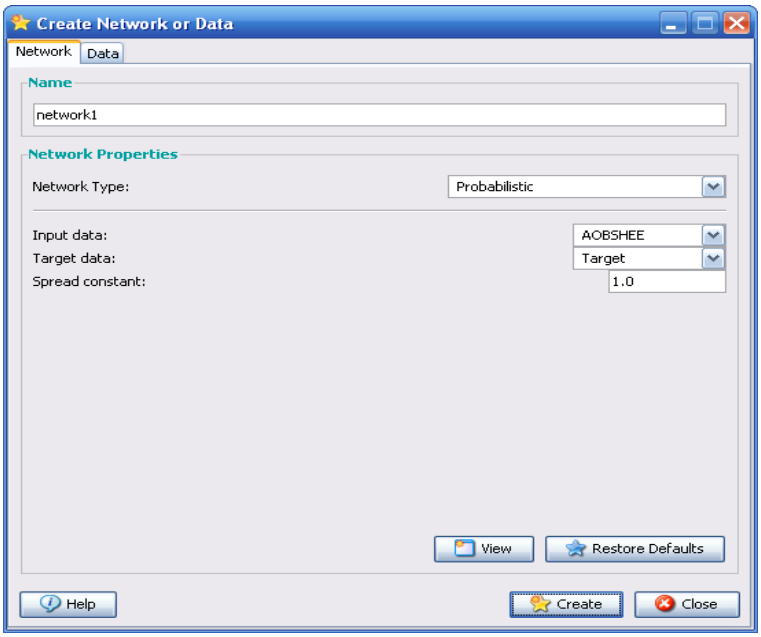


Рис. 4.38. Окно настройки создаваемой нейросети

- Нажав кнопку «View», можно посмотреть структуру создаваемой нейронной сети (рис. 4.39).

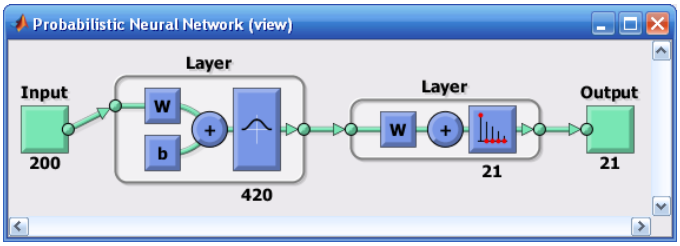


Рис. 4.39. Структура создаваемой сети

После нажатия кнопки «Create» автоматически создается сеть и загружается в окно «Neural Network Tool».

Созданную нейросеть экспортируем в рабочую область MATLAB, а оттуда с помощью команды *gensim* – в виде блока в окно «„Simulink" Tools» (рис. 4.40).

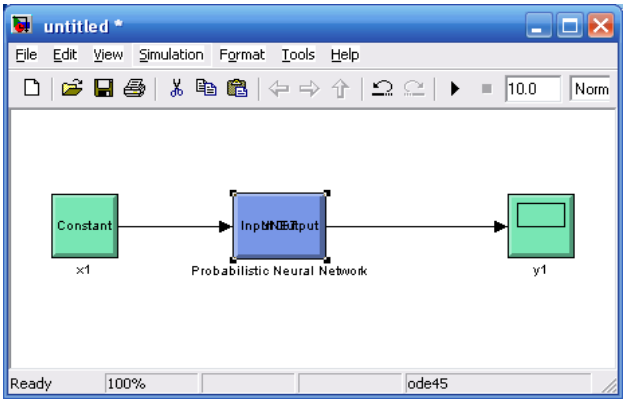


Рис. 4.40. Экспорт нейросети в «Simulink»

Затем проверяем работоспособность созданной сети (рис. 4.41).

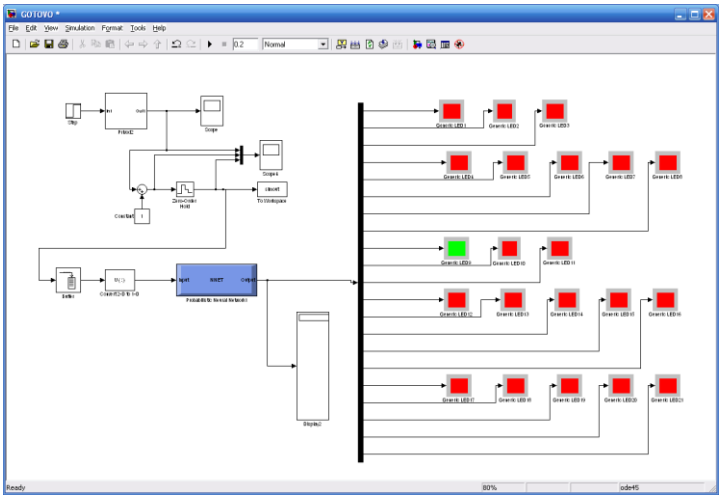
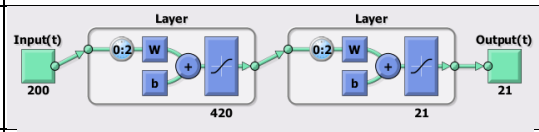
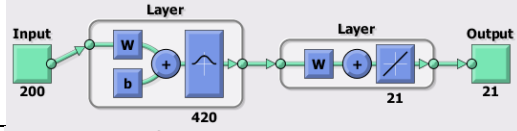
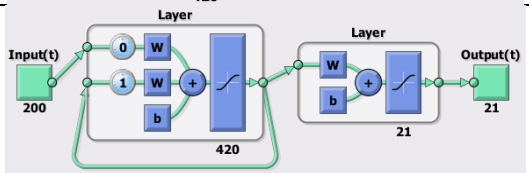
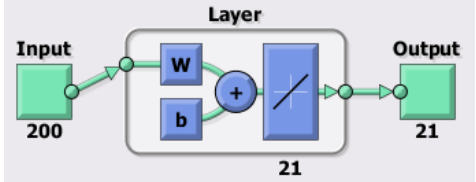
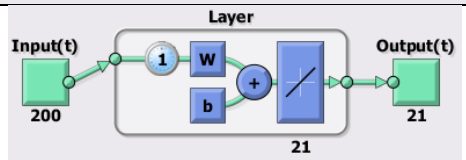
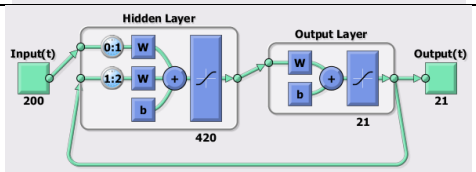
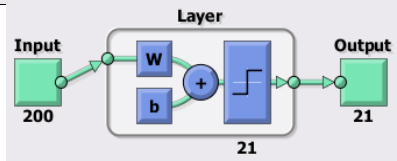
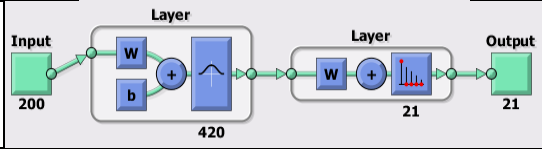


Рис. 4.41. Система в работе в «Simulink»

Результаты работы по поиску наиболее подходящей сети представлены в табл. 4.2.

Таблица 4.2

АРХИТЕКТУРЫ НЕЙРОННЫХ СЕТЕЙ

Тип нейронной сети	**Архитектура сети**	**Число эпох обучения**	**Точность обучения**	**Время обучения**
Feed-forward distributed time delay		-	-	3 с
Generalize-dRegression		-	-	3 с
Layerrecurrent		6	0,317	∞
Linear layer (design)		-	-	3 с
Linear layer (train)		173	1×10^{-6}	20 мин
NARX		-	-	3 с
Perceptron		221	0,967	5:33 мин
Probabilistic		-	-	3 с

После перебора всех типов нейросетей была найдена сеть, наиболее точно воспроизводящая заданные зависимости. Ей оказалась вероятностная сеть **Probabilistic Neural Network** – одна из разновидностей радиально базисных сетей.

Вероятностная нейронная сеть (*PNN* – «Probabilistic Neural Network») представляет собой параллельную реализацию статистических методов Байеса. В *PNN* образцы классифицируются на основе оценок их близости к соседним образцам. При этом используется ряд критериев статистических методов, на основе которых принимается решение о том, к какому классу отнести еще не классифицированный образец. Формальным правилом при классификации является то, что класс с наиболее плотным распределением в области неизвестного образца, а также с более высокой априорной вероятностью и с более высокой ценой ошибки классификации будет иметь преимущество по сравнению с другими классами.

«Probabilistic Neural Network» – нейронная сеть, которая формирует на выходе оценку вероятности принадлежности объекта к определенному классу. В процессе обучения она приобретает способность оценивать функцию распределения, а ее выход рассматривается как ожидаемое значение модели в определенной точке пространства входов.

Нейросеть работает по следующему циклу.

1. Вектор входных сигналов распространяется по связям между нейронами (прямое функционирование).
2. Измеряются сигналы, выданные выходными нейронами.
3. Производится анализ выданных сигналов и вычисляется оценка (разница между выданным сетью ответом и требуемым ответом, имеющимся в примере). Чем меньше значение разности, тем лучше распознан пример, тем ближе к требуемому выданный сетью ответ. Разница, равная нулю, означает, что требуемое соответствие вычисленного и известного (целевого) ответов достигнуто и больше ничего не предпринимается.
4. В противном случае на основании числового значения разности вычисляются поправочные коэффициенты для каждого синаптического веса матрицы связей, после чего производится подстройка синаптических весов (обратное функционирование). В коррекции весов синапсов и заключается обучение.
5. Осуществляется переход к следующему примеру, и все вышеперечисленные операции повторяются. Проход по всем примерам обучающей выборки с первого до последнего считается одним циклом обучения.

Для каждого примера в MATLAB вычисляется ошибка обучения. Вычисляется также суммарная ошибка всех примеров обучающей выборки. Если после прохождения нескольких циклов она равна нулю или малому значению, заданному при создании сети, обучение считается законченным, в противном случае циклы повторяются. Число циклов обучения (эпох), так же, как и время полного обучения, зависит от многих факторов: размера обучающей выборки, числа входных параметров, типа и параметров нейросети и даже от случайного расклада весов синапсов при инициализации сети.

В соответствии с этим правилом для двух классов A и B считают, что элемент *x* принадлежит классу A, если

$$hA\ cA\ fA(x) > hB\ cB\ fB(x),$$

где h – априорная вероятность; c – цена ошибки классификации; $f(x)$ – функция плотности вероятностей. Оценки стоимости ошибки классификации и априорной вероятности предполагают хорошее знание решаемой задачи и в базовой модели сети не используются (считаются одинаковыми).

В общем случае под радиальной базисной нейронной сетью («Radial Basis Function Network», сеть «RBF») понимается двухслойная сеть без обратных связей, которая содержит скрытый слой радиально симметричных нейронов.

Радиальные базисные нейронные сети состоят из бóльшего количества нейронов, чем стандартные сети с прямой передачей сигналов и обучением методом обратного распространения ошибки, но на их создание требуется значительно меньше времени. Эти сети наиболее эффективны, когда доступно большое количество обучающих векторов.

4.5. Использование «Simulink» при построении нейронных сетей

Пакет Neural Network Toolbox содержит ряд блоков, которые либо могут быть непосредственно использованы для построения нейронных сетей в среде «Simulink», либо применяться вместе с рассмотренной выше функцией **gensim.**

Для вызова отмеченного набора блоков в командной строке необходимо набрать команду **neural,** после выполнения которой появляется окно вида (рис. 4.42). Каждый из представленных блоков в свою очередь является набором (библиотекой) некоторых блоков. Рассмотрим их.

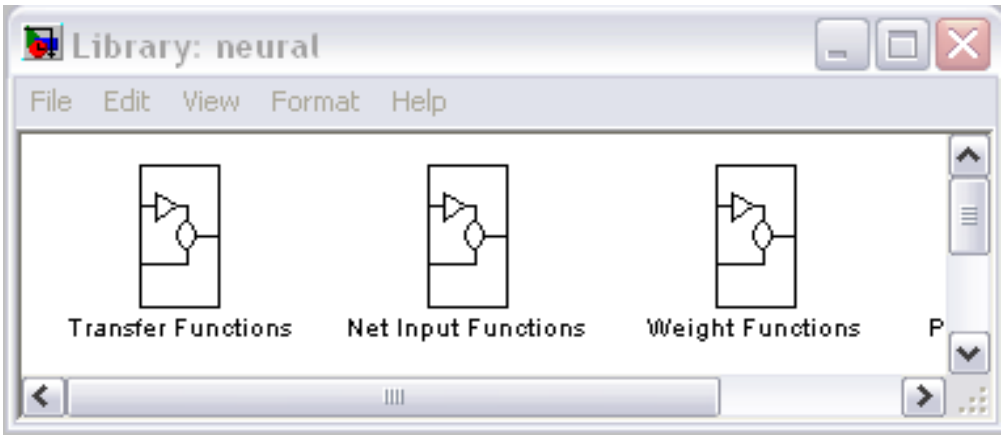


Рис. 4.42. Основные нейросетевые блоки Simulink

Блоки функций активации *(Transfer Functions).* Двойной щелчок левой кнопки мыши на блоке Transfer Functions приводит к появлению библиотеки функций активации (рис. 4. 43). Каждый из этих блоков данной библиотеки преобразует подаваемый на него вектор в соответствующий вектор той же размерности.

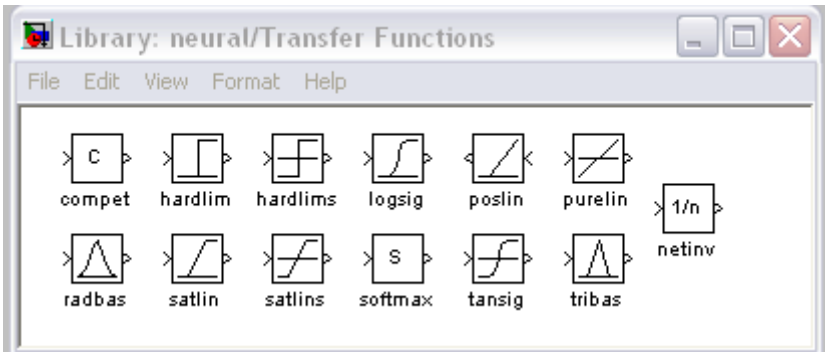


Рис. 4.43. Библиотека функций активации

Блоки преобразования входов сети. Проводя аналогичную рассмотренной операцию, но с блоком Net Input Functions, придем к библиотеке блоков вида рис. 4.44.

Блоки данной библиотеки реализуют рассмотренные выше функции преобразования входов сети.

Блоки весовых коэффициентов. Точно так же (но щелкая левой кнопкой мыши по иконке с надписью Weight Functions) придем к библиотеке блоков (рис. 4.45), реализующих некоторые функции весов и расстояний.

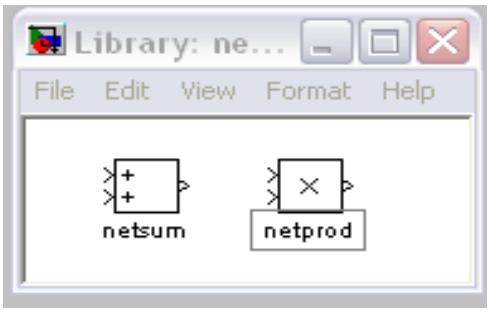


РИС. 4.44. Библиотека блоков преобразований сигналов

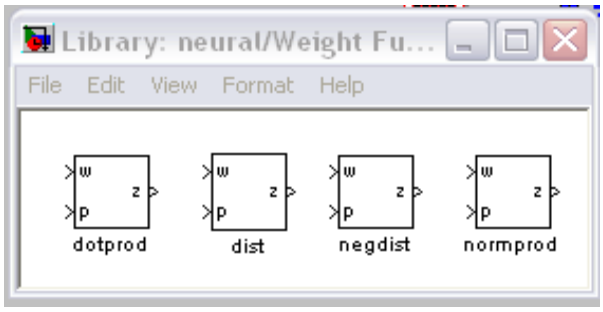


Рис. 4.45. Библиотека блоков весовых коэффициентов

Отметим, что при задании конкретных числовых значений при операции со всеми приведенными блоками ввиду особенностей «Simulink» векторы необходимо представлять как столбцы, а не как строки (как это было до сих пор).

4.6. Формирование нейросетевых моделей

Основной функцией для формирования нейросетевых моделей в «Simulink» является функция **gensim,** записываемая в форме **gensim(net,st),** где **net** – имя созданной НС, **st** – интервал дискретизации (если НС не имеет задержек, ассоциированных с ее входами или слоями, значение данного аргумента устанавливается равным -1).

В качестве примера использования средств «Simulink» рассмотрим следующий.

Пусть входной и целевой векторы имеют вид

p = [1 2 34 5];

t = [l 3 5 7 9].

Создадим линейную НС и протестируем ее:

»p = [1 2 3 4 5];

»t = [1 3 5 7 9];

» net = newlind(p,t);

» y = sim(net,p)

У = 1.0000 3.0000 5.0000 7.0000 9.0000

Затем запустим «Simulink» командой

» gensim(net,-l)

Это приведет к открытию блок-диаграммы (рис. 4.46).

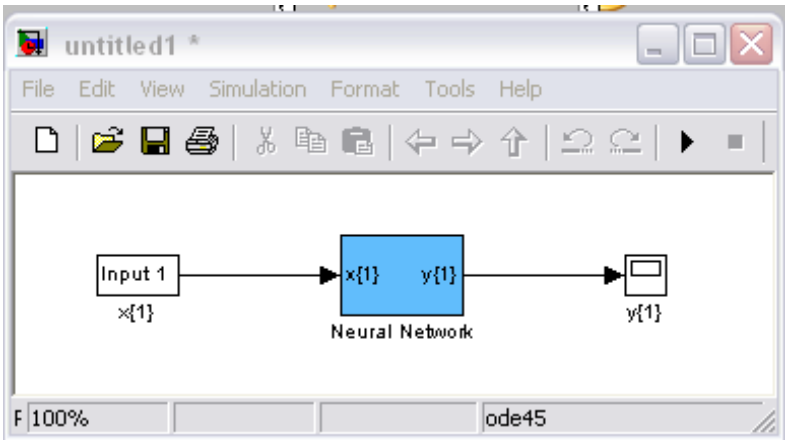


Рис. 4.46. Созданная нейросетевая модель «Simulink»

Для проведения тестирования модели щелкнем дважды по левой иконке (с надписью Input 1, т.е. Вход 1), что приведет к открытию диалогового окна (рис. 4.47).

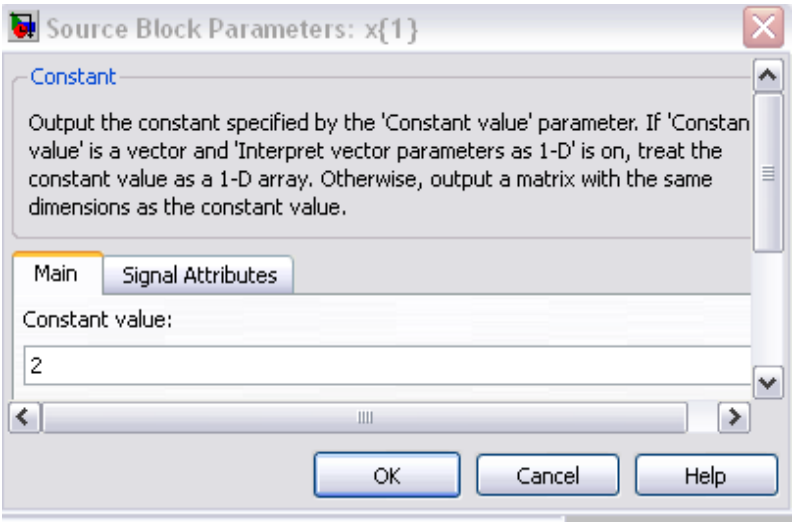


Рис. 4.47. Диалоговое окно задания входа НС

В данном случае блок Input 1 является стандартным блоком задания константы **(Constant).** Изменим значение по умолчанию на 2 и нажмем кнопку ОК. Затем нажмем кнопку **Start** в меню моделирования. Расчет нового значения сетью производится практически мгновенно. Для его вывода необходимо дважды щелкнуть мышью по правой иконке (блоку y(1)). Результат вычислений отображается на рис. 4.48 – он равен 3, как и требуется.

Отметим, что дважды щелкая левой кнопкой мыши по блоку Neural Network, затем – по блоку Layer 1, можно получить детальную графическую информацию о структуре сети (рис. 4.49).

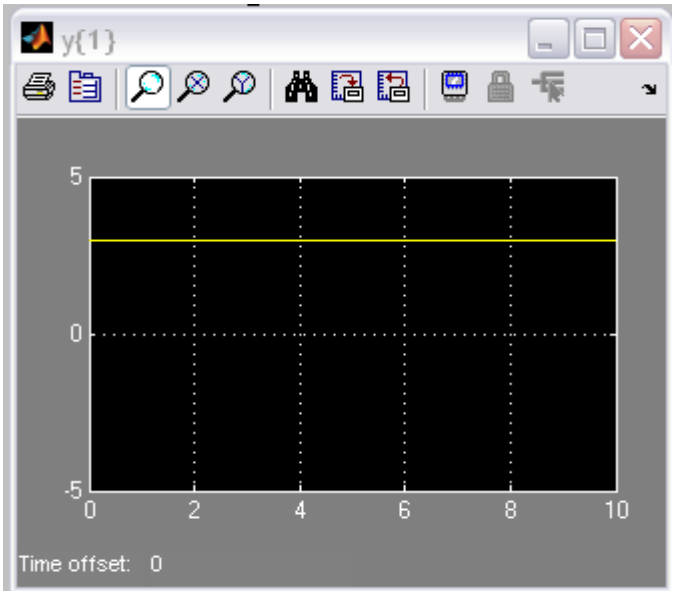


Рис. 4.48. Окно с выходом НС

С созданной сетью можно проводить различные эксперименты, возможные в среде «Simulink», с помощью команды **gensim** осуществляется интеграция созданных нейросетей в блок-диаграммы этого пакета с использованием имеющихся при этом инструментов моделирования различных систем (например, встраивание нейросетевого регулятора в систему управления, моделирование последней и т.п.).

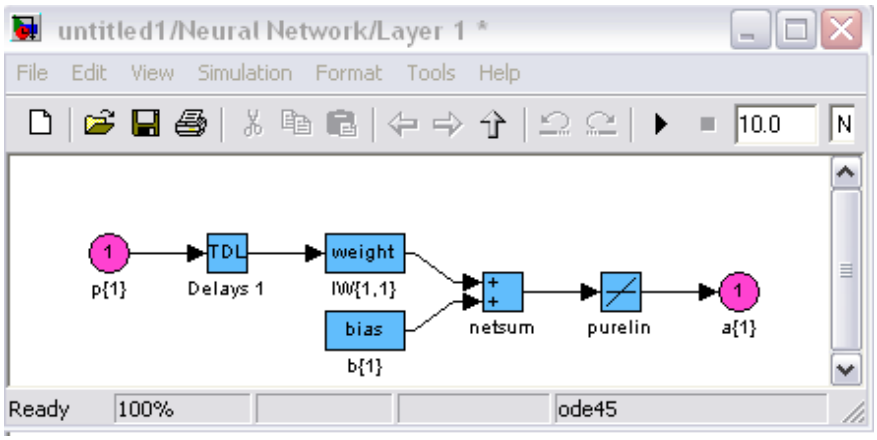


Рис. 4.49. Структура созданной НС

4.7. Построение нейрорегуляторов для задач управления

Для создания и обучения нейронной сети, работающей параллельно объекту управления, в первую очередь создадим модель этого объекта. В нашем случае нейронная сеть будет повторять параметры ПИД-регулятора. На рис. 4.50 представлена модель передаточной функции в цепи с ПИД-регулятором с обратной связью, параллельно которому представлена точно такая же модель, но место «ПИД» оста-

ется пустым. На входной и выходной сигналы устанавливаем 2 блока «ToWorkspace» и переименовываем их как *p* и *t*, где *p* – последовательность входов (задание), а *t* – цель.

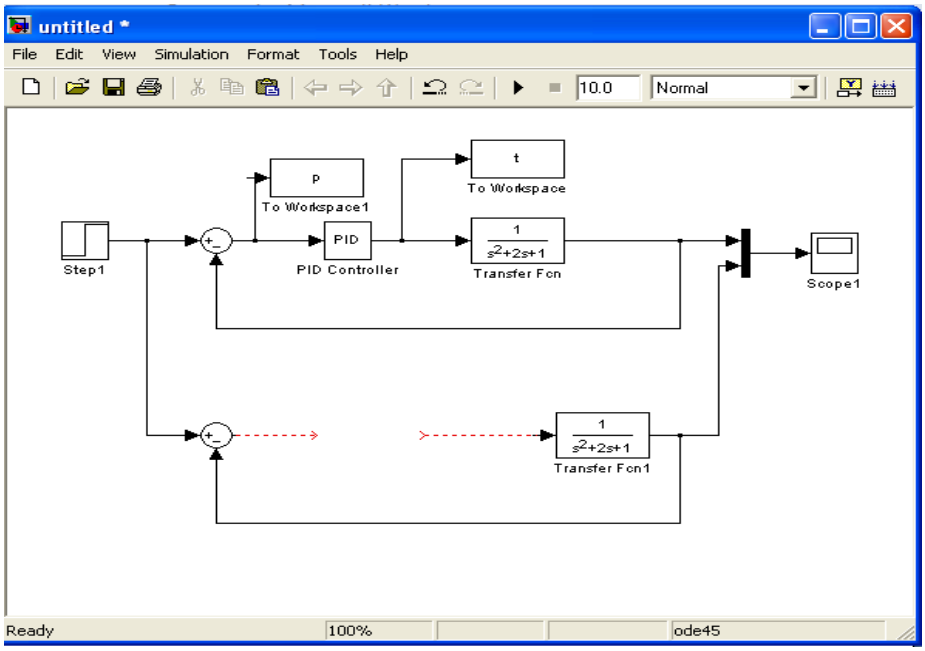


Рис. 4.50. Модель передаточной функции

Затем нажимаем «Startsimulation», и наши значения массива задания *p* и цели *t* формируются в «Workspace».

Моделирование нейронной сети в системе «Simulink»

Формирование М-файла для моделирования нейронной сети имеет *синтаксис:*

[*Y,Pf,Af,E,perf*] = *sim*(*net,P,Pi,Ai,T*);

[*Y,Pf,Af,E,perf*] = *sim*(*net,*{*Q TS*}*Pi,Ai,T*);

[*Y,Pf,Af,E,perf*] = *sim*(*net,Q,Pi,Ai,T*).

Описание: функция [*Y,Pf,Af,E,perf*] = *sim*(*net,P,Pi,Ai,T*) выполняет моделирование нейронной сети и имеет следующие входные и выходные аргументы.

Входные аргументы:

• *net* – имя нейронной сети;

• *P* – массив входов;

• *Pi* – начальные условия на ЛЗ входов, по умолчанию нулевой вектор;

• *Ai* – начальные условия на ЛЗ слоев, по умолчанию нулевой вектор;

• *T* – вектор целей, по умолчанию нулевой вектор.

Выходные аргументы:

• *Y* – массив выходов;

• *Pf* – состояния на ЛЗ входов после моделирования;

• *Af* – состояния на ЛЗ слоев после моделирования;

• *E* – массив ошибок;

• *perf* – значение функционала качества.

Заметим, что аргументы *Pi*, *Ai*, *Pf* и *Af* являются не обязательными и применяются в случае динамических сетей с ЛЗ.

Сформируем М-файл для S-модели сети Элмана, реализующей следующие соотношения между входом и целью:

P = rot90(p);

T = rot90(t);

Создадим двуслойную сеть Элмана с массивом входов с диапазоном значений [*minmax*(*P*)], имеющую 70 нейронов с функцией активации *tansig* в слое 1 и 1 нейрон с функцией активации *purelin* в слое 2 с прямой передачей сигнала. При этом в качестве обучающего алгоритма выбран алгоритм «Levenberg-Marquardt» (*trainlm*). Этот алгоритм обеспечивает быстрое обучение, но требует много ресурсов. В случае, если для реализации этого алгоритма не хватит оперативной памяти, можно использовать другие алгоритмы (*trainbfg*, *trainrp*, *trainscg*, *traincgb*, *traincgf*, *traincgp*, *trainoss*, *traingdx*). По умолчанию используется *trainlm*. Указанная сеть формируется с помощью процедуры:

net=newff([minmax(P)],[70 1],{'tansig','purelin'},'trainlm');

%двухслойная сеть с прямой передачей сигнала;

%первый слой – 70 нейронов с функцией активации *tansig*;

%второй слой – 1 нейрон с функцией активации *purelin*;

%диапазон изменения входа задается командой *minmax*(*P*);

%функция обучения *trainlm*.

Первый аргумент – матрица M × 2 минимальных и максимальных значений компонент входных векторов – вычисляется с помощью процедуры *minmax*.

Результатом выполнения процедуры *newff* является объект – нейронная сеть *net* заданной конфигурации. Сеть можно сохранить на

диске в виде *mat*. файла с помощью команды «Save»и загрузить снова с помощью команды «Load».

Обработка тестового массива:

1. Создаем сеть: *Y = sim (net, P).*

2. Задаем максимальное количество циклов обучения. После того как будет выполнено это количество циклов, обучение будет завершено:

net.trainParam.epochs=500 (задаем количество эпох обучения).

3. Задаем интервал вывода информации, измеренный в циклах:

net.trainParam.show=100.

4. Выполняем процедуру обучения:

net=train(net,P,T); % тренируем сеть.

5. Для проверки качества обучения промоделируем спроектированную сеть:*Y = sim(net,P).*

6. Строим графики исследуемой функции:

figure(1); plot(P,T,'y'); hold on; grid on; plot(P,Y,'r').

7. Формируем S-модель нейронной сети с именем *net* для ее запуска в среде системы «Simulink»:

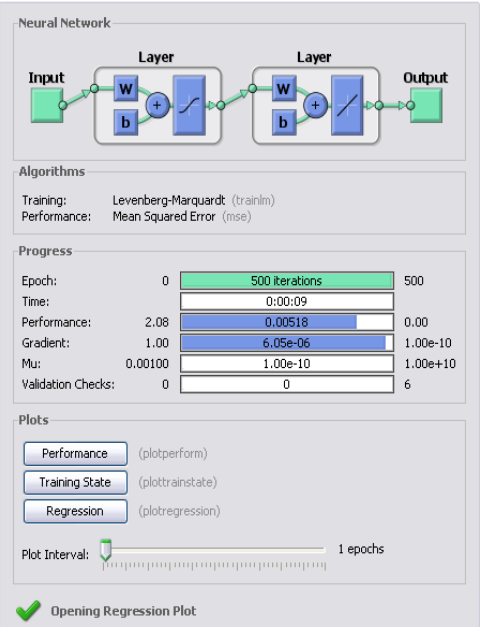


Рис. 4.51. Окно обучения сети

gensim(net) (моделируем сеть).

8. Смотрим характеристики сети: *disp*(*net*).

Затем нажимаем «Runmfail.m», в результате откроется окно системы «Simulink» с S-моделью нейронной сети, которая включает блок входа, блок нейронной сети и блок осциллографа.

Перед открытием окна с S-моделью открывается окно обучения сети «Neural Network Training» (рис. 4.51), в котором виден отсчет эпох обучения.

При нажатии кнопки «Performance» открывается график обученной нами сети (рис. 4.52). Далее «Training State» (рис. 4.53) и «Regression» (рис. 4.54).

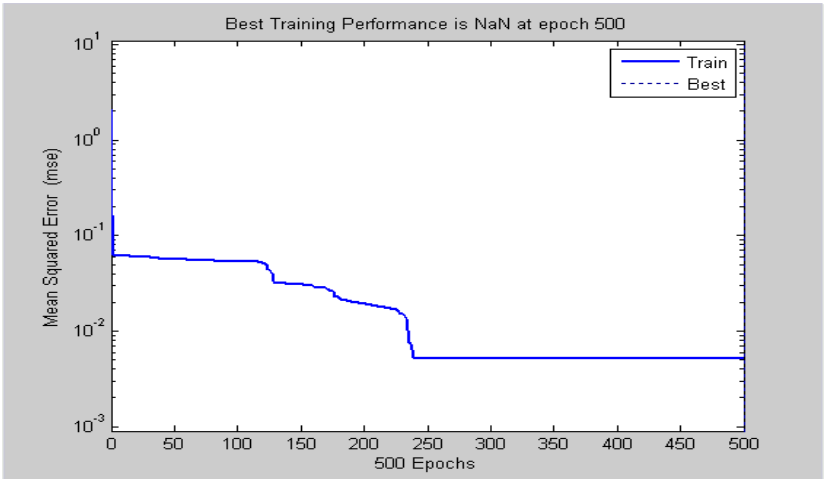


Рис. 4.52. График обученной сети

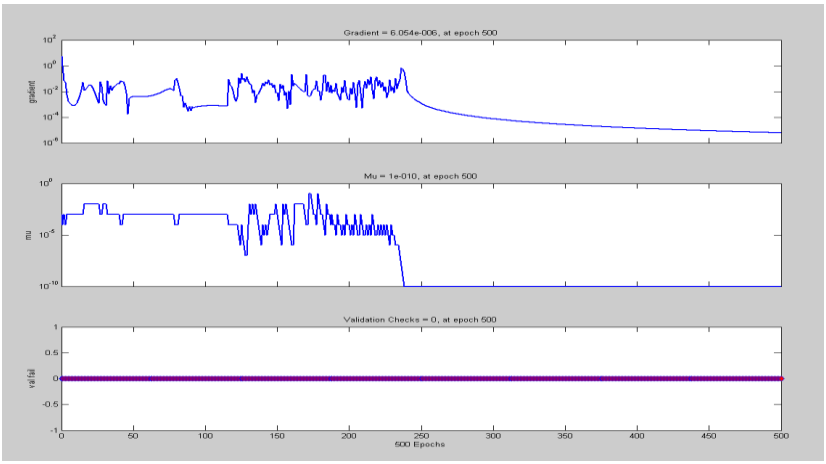


Рис. 4.53. «Training State»

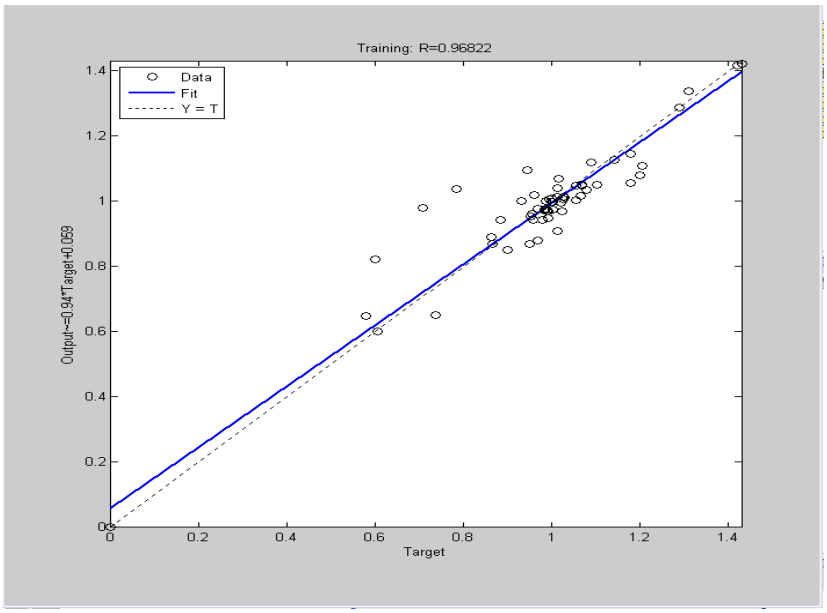


Рис. 4.54. «Regression»

В результате выполнения функции *gensim(net)* открывается окно, на котором представлена нейронная сеть (рис. 4.55).

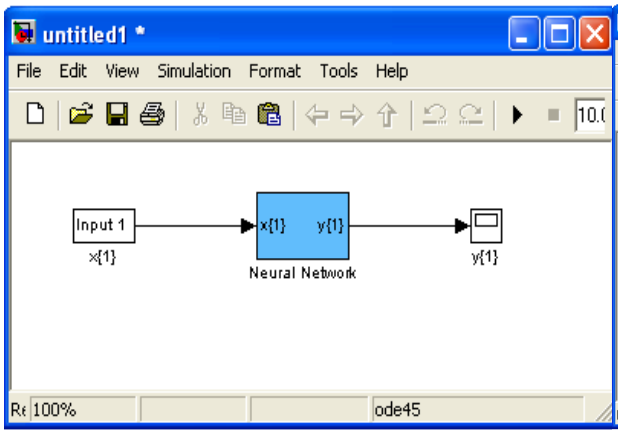


Рис. 4.55. Нейронная сеть

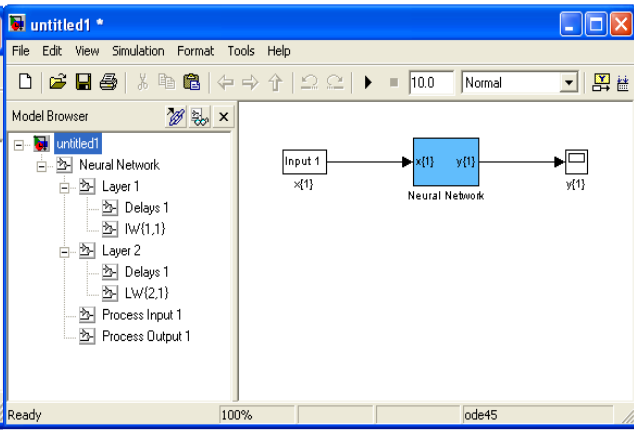


Рис. 4.56. Содержание сети

Нажатие кнопки «View/Model Browser Option/Model Browser» переводит окно системы «Simulink» в другое состояние (рис. 4.56).

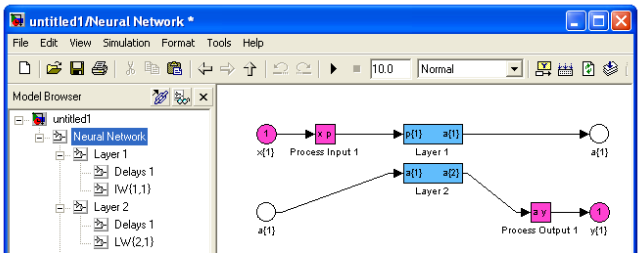
Окно «Simulink» дополняется средством просмотра модели, расположенным слева от схемы сети. Включение кнопки «Show library links» и соседней с ней «Browse masked subsystem» позволяет раскрыть описание структуры нейронной сети и просматривать ее элементы простым выбором того или иного элемента структуры. Рис. 4.57 (*а – д*) поясняет эту возможность.

Следующее, что нам нужно сделать, это скопировать блок нейронной сети (см. рис. 4.55) и вставить его в оставленное пустое место нашей исходной модели, представленной на рис. 4.50.

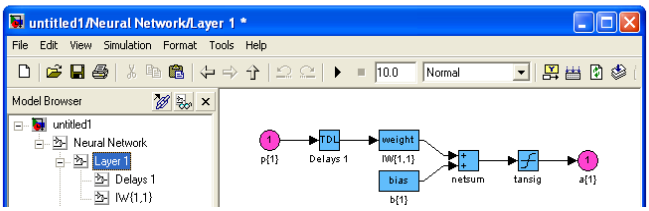
Итак, на рис. 4.58 представлена нейронная сеть, работающая параллельно объекту управления.

Запускаем нашу модель нажатием кнопки «Start simulation» и двойным щелчком левой клавиши мыши открываем осциллограф, на котором показаны графики объекта и нейронной сети (рис. 4.59).

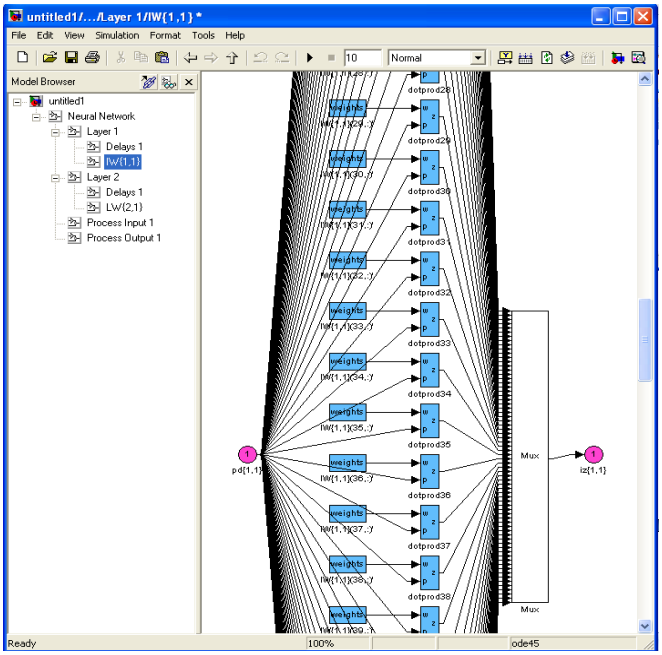
а)



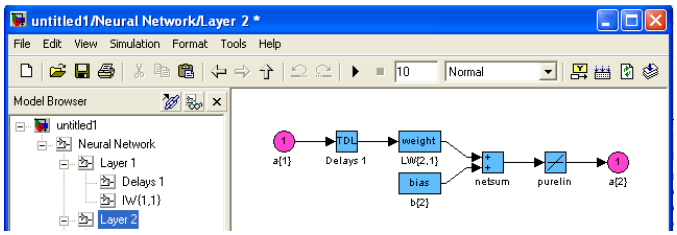
б)



в)



г)



д)

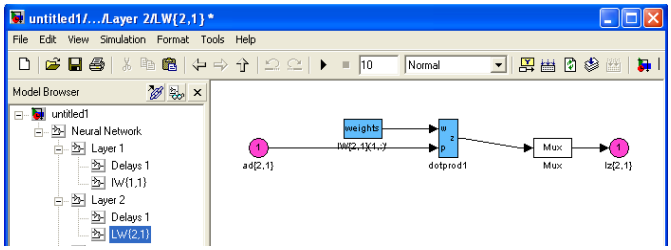


Рис. 4.57. Структура нейронной сети

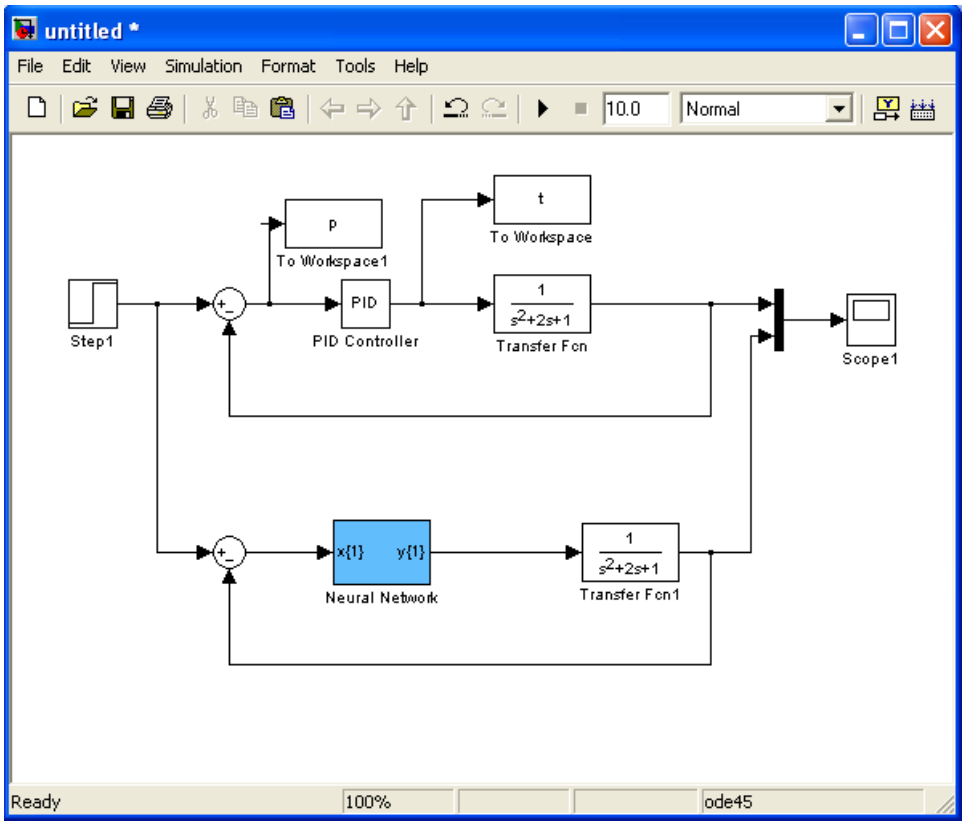


Рис. 4.58. Нейронная сеть, работающая параллельно процессу

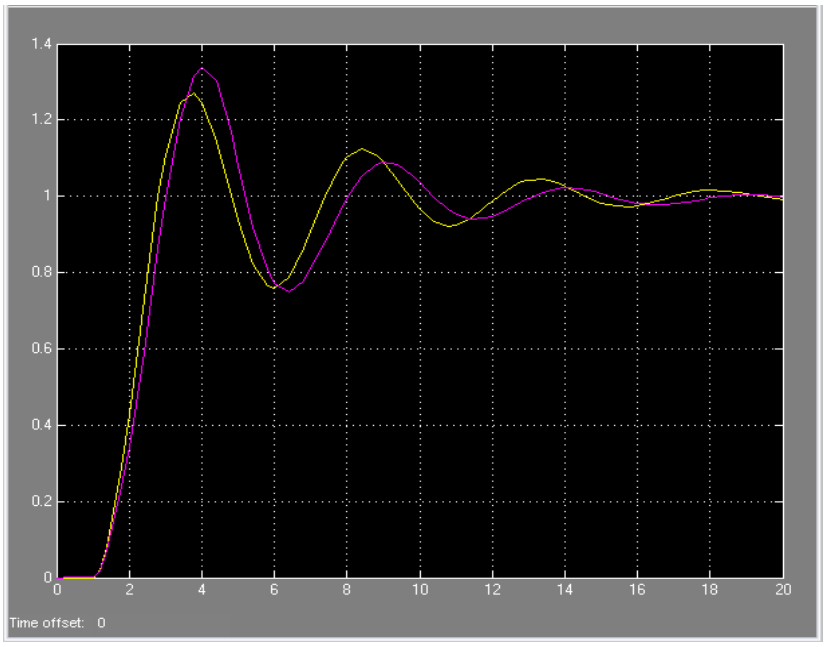


Рис. 4.59. Графики объекта и нейронной сети

Информация по использованию нейросетей изложена также в прил. 2, 3.

Контрольные задания*

1. **Нейронные сети – это:**

a) сети с обратными связями;

b) большой класс разнообразных систем, архитектура которых имитирует построение нервной ткани из нейронов;

c) сети с прямым распространением;

d) многослойные сети.

2. **В основу нейросети входит:**

a) однотипные элементы (ячейки);

b) группа синапсов;

c) аксоны;

d) синоптическая связь.

3. **Архитектура нейросети определяет:**

a) какие нейроны будут использоваться (число входов, активационные функции);

b) каким образом следует соединить их между собой (топология сети);

c) что взять в качестве входов и выходов сети;

d) всё вышеперечисленное.

4. **С точки зрения архитектуры можно выделить следующие типы нейронных сетей:**

a) полносвязные ИНС;

b) слабосвязные ИНС;

c) многослойные ИНС;

d) все вышеперечисленные.

5. **Среди многослойных нейронных сетей можно выделить:**

a) монотонные;

b) сети без обратных связей;

c) сети с обратными связями (рекуррентные сети);

d) все вышеперечисленные.

---

* Из предложенных ответов выберите правильный.

5. НЕЧЕТКИЕ МНОЖЕСТВА В СИСТЕМАХ ДИАГНОСТИКИ

5.1. Основные положения теории нечетких множеств

Наиболее важным применением теории нечетких множеств являются контроллеры нечеткой логики. Их функционирование несколько отличается от работы обычных контроллеров; для описания системы вместо дифференционных уравнений используются знания экспертов. Эти знания могут быть выражены с помощью лингвистических переменных, которые описаны нечеткими множествами.

5.2. Общая структура нечеткого контроллера

Общая структура микроконтроллера, использующего нечеткую логику, содержит:

- блок фаззификации;
- базу знаний;
- блок решений;
- блок дефаззификации.

Блок фаззификации преобразует четкие величины, измеренные на выходе объекта диагностирования, в нечеткие величины, которые описаны лингвистическими переменными в базе знаний.

Блок решений использует нечеткие условные (*if - then*) правила, заложенные в базу знаний, для преобразования нечетких входных данных в необходимые управляющие влияния, которые также носят нечеткий характер.

Блок дефаззификации превращает нечеткие данные с выхода блока решений в четкую величину, которая используется для диагностики объекта.

В качестве примера известных микроконтроллеров, использующих нечеткую логику, можно назвать 68HC11, 68HC12 фирмы *Motorola, MCS*-96 фирмы *Intel*, а также некоторые другие.

5.3. Создание модели нечеткого контроллера в MATLAB

Назначение и возможности пакета «Fuzzy Logic Toolbox»

Пакет нечеткой логики – это пакет прикладных программ, относящихся к теории размытых или нечетких множеств и позволяющих конструировать так называемые нечеткие экспертные и/или управляющие системы. Основные возможности пакета:

– построение систем нечеткого вывода (экспертных систем, регуляторов, аппроксиматоров зависимостей);

– построение адаптивных нечетких систем (гибридных нейронных сетей);

– интерактивное динамическое моделирование в «Simulink»;

– работа:

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

Построение нечеткой системы

Командой (функцией) «Fuzzy» из режима командной строки запускается основная интерфейсная программа пакета «Fuzzy Logic» – редактор нечеткой системы вывода. Вид открывающегося при этом окна приведен на рис. 5.1.

Строка меню редактора содержит следующие позиции:

• «File» – работа с файлами моделей (их создание, сохранение, считывание и печать);

• «Edit» – операции редактирования (добавление и исключение входных и выходных переменных);

• «View» – переход к дополнительному инструментарию.

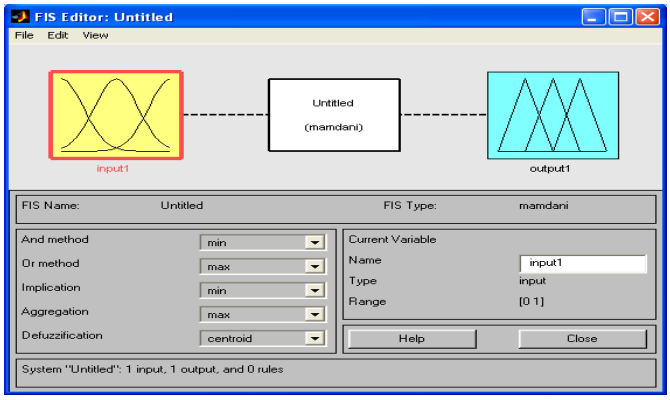


Рис. 5.1. Вид окна «FIS Editor»

Создается нечеткая система, отображающая зависимость между входными и выходными переменными. В качестве входных переменных – отклонения скорости и тока, полученные при разности выходных сигналов модели электродвигателя, работающего при номинальных значениях параметров, и реального двигателя.

Нечеткие выводы

Используемый в различного рода экспертных и управляющих системах механизм нечетких выводов в своей основе имеет базу знаний, формируемую специалистами предметной области в виде совокупности нечетких предикатных правил вида:

$П_1$: если *x* есть A_1, то *y* есть B_1,

$П_2$: если *x* есть A_2, то *y* есть B_2,

…

$П_n$: если *x* есть A_n, то *y* есть B_n,

где *x* – входная переменная (имя для известных значений данных), *y* – переменная вывода (имя для значения данных, которое будет вычислено); A и B – функции принадлежности, определенные соответственно на *x* и *y*.

Пример подобного правила:

если *x* – низко, то *y* – высоко.

Приведем более детальное пояснение. Знание эксперта $A \rightarrow B$ отражает нечеткое причинное отношение предпосылки и заключения, поэтому его можно назвать нечетким отношением и обозначить через R:

$R = A \rightarrow B$, где «$\rightarrow$» называют нечеткой импликацией.

Отношение R можно рассматривать как нечеткое подмножество прямого произведения XY полного множества предпосылок X и заключений Y. Таким образом, процесс получения (нечеткого) результата вывода B' с использованием данного наблюдения A' и знания $A \rightarrow B$ можно представить в виде формулы

$B' = A' \bullet R = A' \bullet (A \rightarrow B)$,

где «•» – введенная выше операция свертки.

Как операцию композиции, так и операцию импликации в алгебре нечетких множеств можно реализовывать по-разному (при этом, естественно, будет разниться и итоговый получаемый результат), но в любом случае общий логический вывод осуществляется за следующие четыре этапа.

1. *Нечеткость* (введение нечеткости, фаззификация, «fuzzification»). Функции принадлежности, определенные на входных переменных, применяются к их фактическим значениям для определения степени истинности каждой предпосылки каждого правила.

2. *Логический вывод*. Вычисленное значение истинности для предпосылок каждого правила применяется к заключениям каждого правила. Это приводит к одному нечеткому подмножеству, которое будет назначено каждой переменной вывода для каждого правила. В качестве правил логического вывода обычно используются только операции «min» (минимум) или «prod» (умножение). В логическом

выводе min функция принадлежности вывода «отсекается» по высоте, соответствующей вычисленной степени истинности предпосылки правила (нечеткая логика «И»). В логическом выводе prod функция принадлежности вывода масштабируется при помощи вычисленной степени истинности предпосылки правила.

3. *Композиция*. Нечеткие подмножества, назначенные для каждой переменной вывода (во всех правилах), объединяются вместе, чтобы сформировать одно нечеткое подмножество для каждой переменной вывода. При подобном объединении обычно используются операции «max» (максимум) или «sum» (сумма). При композиции max комбинированный вывод нечеткого подмножества конструируется как поточечный максимум по всем нечетким подмножествам (нечеткая логика «ИЛИ»). При композиции sum комбинированный вывод нечеткого подмножества конструируется как поточечная сумма по всем нечетким подмножествам, назначенным переменной выхода правилами логического вывода.

4. *Заключение* (дополнительно) – приведение к четкости (дефаззификация, «defuzzification»), которое используется, когда полезно преобразовать нечеткий набор выводов в четкое число.

Рассмотрим следующие наиболее употребительные модификации алгоритма нечеткого вывода, полагая для простоты, что базу знаний организуют два нечетких правила вида:

$П_1$: если x есть A_1, и y есть B_1, тогда z есть C_1,

$П_2$: если x есть A_2,и y есть B_2, тогда z есть C_2,

где x и y – имена входных переменных, z – имя переменной вывода, A_1, A_2, B_1, B_2, C_1,C_2 – некоторые заданные функции принадлежности, при этом четкое значение z_0 необходимо определить на основе приведенной информации и четких значений x_0и y_0.

Алгоритм Мамдани (Mamdani)

Данный алгоритм математически может быть описан следующим образом.

1. Нечеткость: находятся степени истинности для предпосылок каждого правила: $A_1(x_0)$, $A_2(x_0)$, $B_1(y_0)$, $B_2(y_0)$.

2. Нечеткий вывод: находятся уровни «отсечения» для предпосылок каждого из правил (с использованием операции «min»):

α_1,= $A_1(x_0) \wedge B_1(y_0)$,

$\alpha_2 = A_2(x_0) \wedge B_2(y_0)$,

где через «∧», как и раньше, обозначена операция логического минимума («min»), затем находятся усеченные функции принадлежности:

$C'_1(z) = (\alpha_1 \wedge C_1(z))$,

$C'_2(z) = (\alpha_2 \wedge C_2(z))$.

3. Композиция: с использованием операции «max» (обозначаемой как «у») производится объединение найденных усеченных функций, что приводит к получению итогового нечеткого подмножества для переменной вывода с функцией принадлежности:

$\mu_\Sigma(z) = C(z) = C'_1(z) \vee C'_2(z) = (\alpha_1 \wedge C_1(z)) \vee (\alpha_2 \wedge C_2(z))$.

4. Наконец, приведение к четкости (для нахождения z_0) проводится, например, центроидным методом (как центр тяжести для кривой $\mu_\Sigma(z)$):

$$z_0 = \frac{\int_\Omega z\mu_\Sigma(z)dz}{\int_\Omega \mu_\Sigma(z)dz}.$$

Алгоритм иллюстрируется на рис. 5.2.

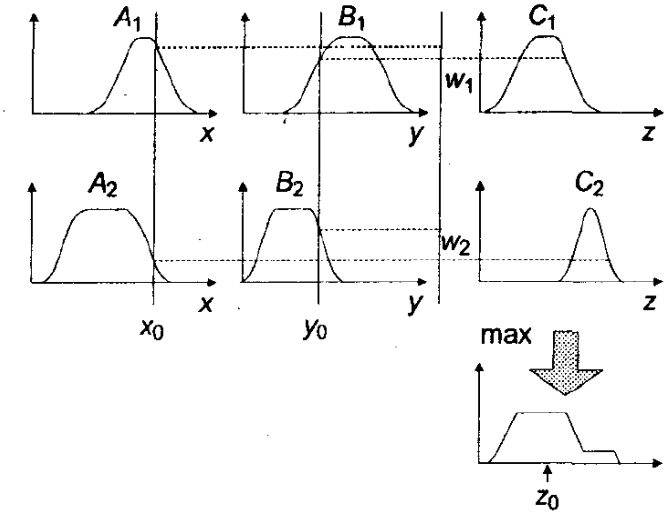


Рис. 5.2. Иллюстрация к алгоритму Мамдани

Алгоритм Сугэно (Sugeno)

Сугэно (Sugeno) и Такаги (Takagi) использовали набор правил в следующей форме (как и раньше, приводим пример двух правил):

$П_1$: если x есть A_1 и y есть B_1, тогда $z_1 = a_1x + b_1y$,

$П_2$: если x есть A_2 и y есть B_2, тогда $z_2 = a_2x + b_2y$.

Представление алгоритма

1. Первый этап – как в алгоритме Мамдани.

2. На втором этапе находятся $\alpha_1 = A_1(x_0) \wedge B_1(y_0)$, $\alpha_2 = A_2(x_0) \wedge B_2(y_0)$ и индивидуальные выводы правил:

$z_1^{\cdot} = a_1 x_0 + b_1 y_0$,

$z_2^{\cdot} = a_2 x_0 + b_2 y_0$.

3. На третьем этапе определяется четкое значение переменной вывода:

$$z_0 = \frac{\alpha_1 \overset{\bullet}{z}_1 + \alpha_2 \overset{\bullet}{z}_2}{\alpha_1 + \alpha_2}.$$

Алгоритм иллюстрируется на рис. 5.3.

Приведенное представление относится к алгоритму Сугэно 1-го порядка. Если правила записаны в форме:

$П_1$: если x есть A_1 и y есть B_1, то $z_1 = c_1$,

$П_2$: если x есть A_2 и y есть B_2, то $z_2 = c_2$,

то говорят, что задан алгоритм Сугэно 0-го порядка (рис. 5.4).

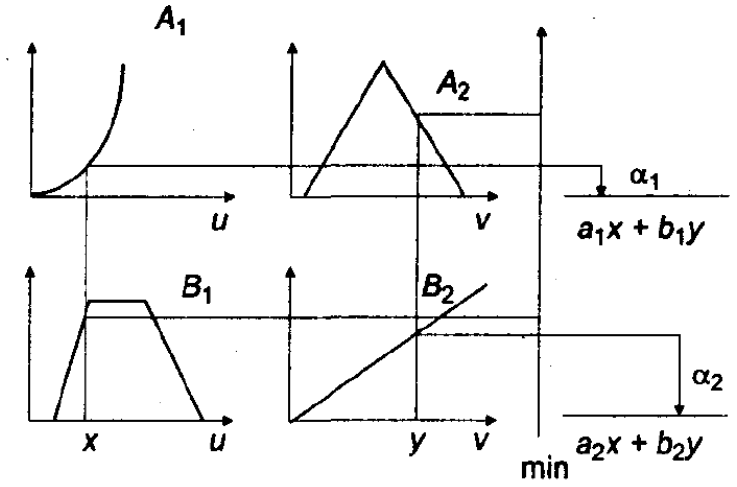


Рис. 5.3. Иллюстрация к алгоритму Сугэно

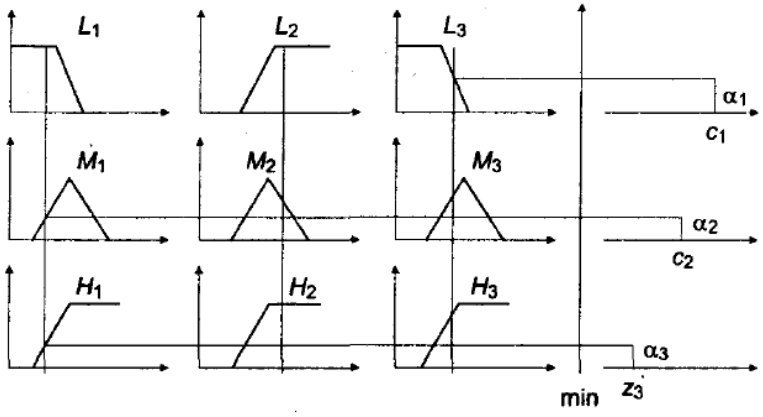


Рис. 5.4. Алгоритм Сугэно 0-го порядка

Для данной задачи будем использовать алгоритм Сугэно.

Требуемые действия:

1. В меню «File» выбираем команду «NewSugenoFIS» (Новая система типа «Sugeno»), при этом в блоке, отображаемом белым квадратом в верхней части окна редактора, появится надпись «Untitled (sugeno)».

2. Щелкнем на блоке, озаглавленном «inputl». Затем в правой части редактора в поле «Name» (Имя) вместо лингвистической переменной «inputl» введем обозначение нашего аргумента, т.е. «отклонение скорости». Лингвистической переменной (linguistic variable) называется переменная, значениями которой могут быть слова или словосочетания некоторого естественного или искусственного языка.

Обратим внимание, что если теперь сделать где-нибудь (вне блоков редактора) однократный щелчок, то имя отмеченного блока изменится на отклонение скорости; то же достигается нажатием клавиши Enter после ввода. Добавим второй вход (открываем меню «Edit» данного редактора и выбираем в нем команду «Add Variable → Input»). Аналогично обозначаем вход 2 как «отклонение тока».

3. В нечеткой логике для задания функций принадлежности используются типовые формы функций принадлежности (рис. 5.5):

– треугольная (trimf);
– трапецеидальная (trapmf);
– гауссова (gaussmf);
– двойная гауссова (gauss2mf);
– обобщенная колоколообразная (gbellmf);
– сигмоидальная (sigmf);
– двойная сигмоидальная (dsigmf);
– произведение двух сигмоидальных функций (psigmf);
– Z-функция;
– S-функция;
– Pi-функция.

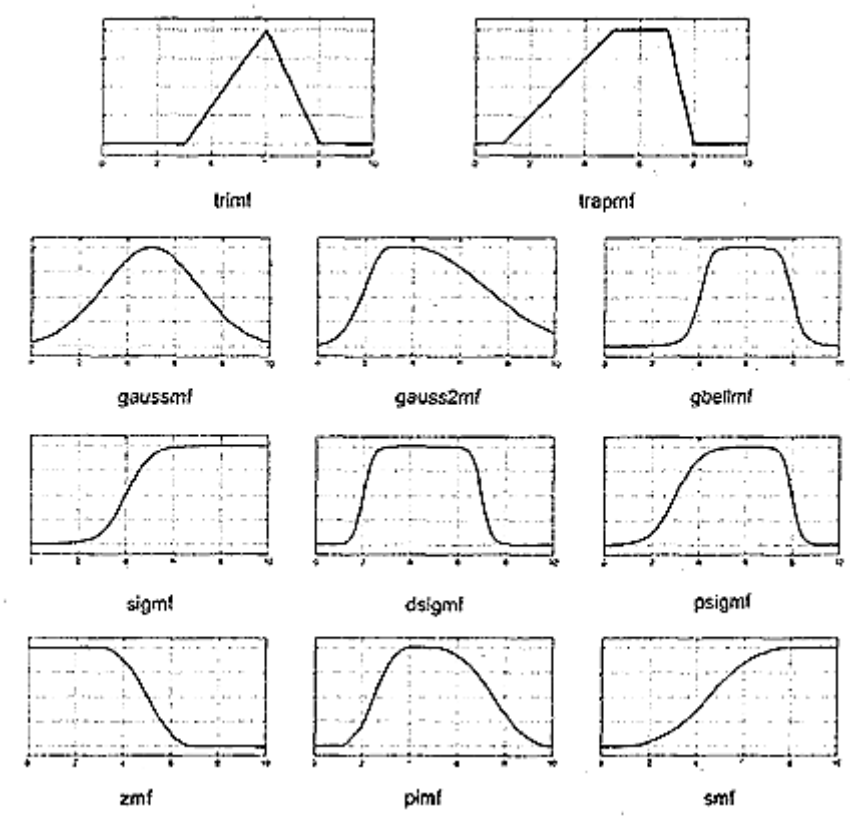


Рис. 5.5. Типовые функции принадлежности

Конкретный вид данных функций определяется значениями параметров, входящих в их аналитические представления.

Дважды щелкнем на этом блоке. Перед нами откроется окно редактора функций принадлежности – «Membership Function Pditor» (рис. 5.6). Откроем меню «Edit» данного редактора и выберем в нем команду «Add MFs» («Add Membership Functions» – добавить функции принадлежности). При этом появится диалоговое окно (рис. 5.7), позволяющее задать тип («MF type») и количество («Number of MFs») функций принадлежности (в данном случае все относится к входному сигналу 1, т.е. к переменной «отклонение скорости»). Выберем трапецеидальные функции принадлежности (*trapmf*), так как они наиболее полно описывают все точки, принадлежащие кривой отклонений входных сигналов нечеткой системы, а их количество зададим равным девяти – по числу выбранных значений интервалов отклонений скорости. Подтвердим ввод информации нажатием кнопки «ОК», после чего произойдет возврат к окну редактора функций принадлежности. Аналогично зададим функции принадлежности для входа 2, число которых соответствует заданной градации отклонений тока, т.е. пяти.

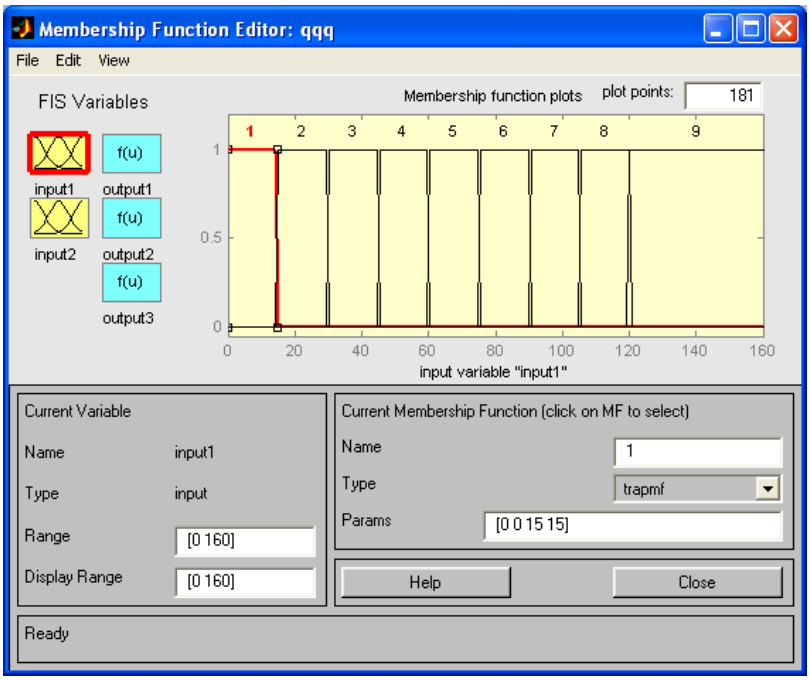


Рис. 5.6. Окно редактора функций принадлежности

4. В поле «Range» (Диапазон) установим диапазон изменения отклонений скорости от 0 до 160, так как при большем отклонении данного параметра можно сделать вывод о неисправности двигателя. Щелкнем затем левой кнопкой мыши где-нибудь в поле редактора (или нажмем клавишу «Enter»). Обратим внимание, что после этого произойдет соответствующее изменение диапазона в поле «Display Range» (Диапазон отображения).

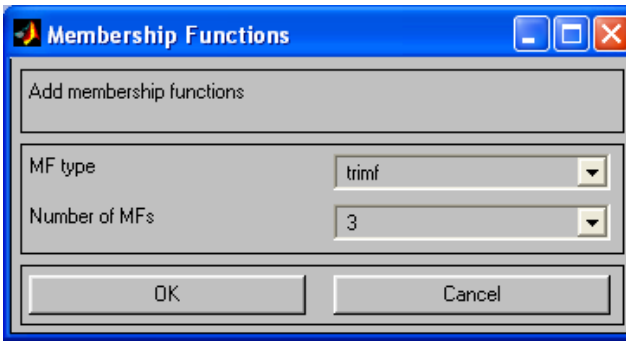


Рис. 5.7. Диалоговое окно задания типа и количества функций принадлежности входной переменной

5. Редактируем функции принадлежности, задавая в поле «Params» (Параметры) числовые значения (в данном случае каждой функции принадлежности соответствуют четыре параметра, при этом первый и четвертый определяют размах кривой, второй и третий – положение ее основания). В поле «Name» изменяем имя для выбранной кривой (завершая ввод каждого имени нажатием клавиши «Enter»).

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

6. Добавим два выхода (открываем меню «Edit» данного редактора и выбираем в нем команду «Add Variable→Output»). Сделаем однократный щелчок на голубом квадрате (блоке), озаглавленном «outputl» (Выход1). В поле «Name» заменяем имя «outputl» на «перегрев». Подобным образом заменим имя «output2» на «увеличение момента инерции», имя «output3» – на «превышение номинальной нагрузки».

7. Дважды щелкнем на выделенном блоке и перейдем к редактору функций принадлежности. В меню «Edit» выберем команду «Add MFs». Появляющееся затем диалоговое окно вида (рис. 5.8) позволяет теперь задать в качестве функций принадлежности только линейные (*linear*) или постоянные (*constant*) – в зависимости от того, какой алгоритм Сугэно (1-го или 0-го порядка) мы выбираем. В рассматриваемой задаче необходимо выбрать постоянные функции принадлежности. Подтвердим введенные данные нажатием кнопки «ОК», после чего произойдет возврат в окно редактора функций принадлежности.

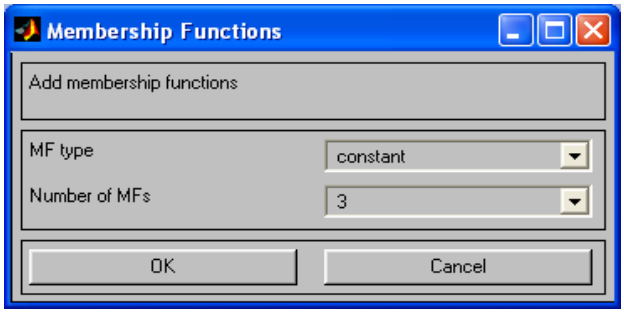


Рис. 5.8. Диалоговое окно задания типа и количества функций принадлежности выходной переменной

8. Обратим внимание, что диапазон изменения («Range»), устанавливаемый по умолчанию – [0,1], – менять в данном случае не нужно. Изменяем лишь имена функций принадлежности (их графики при использовании алгоритма Сугэно для выходных переменных не приводятся), например, задав их как соответствующие числовые значения выходных переменных, одновременно эти же числовые значения введем в поле «Params» (рис. 5.9). Затем закроем окно нажатием кнопки «Close» и вернемся в окно FIS-редактора.

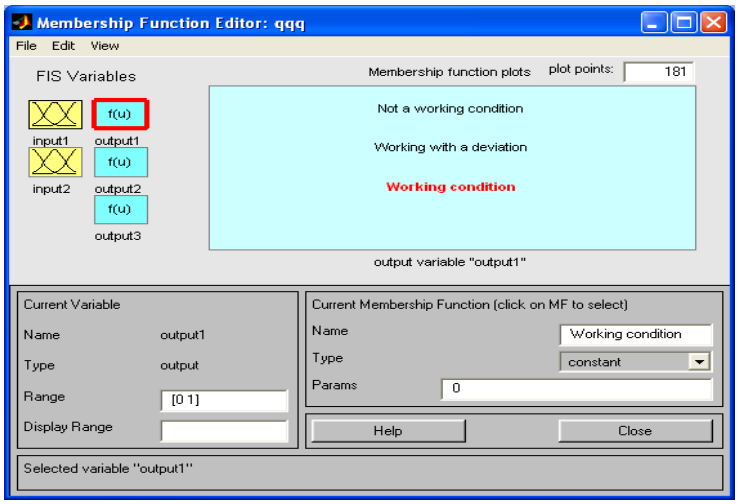


Рис. 5.9. Параметры функций принадлежности выходной переменной

Для установки соответствия между функциями принадлежности входных и выходных сигналов создаем систему правил, присваивая каждой функции принадлежности в MATLAB соответствующее лингвистическое значение переменной, несущее заданный физический смысл (табл. 5.1, 5.2).

Таблица 5.1

ТАБЛИЦА СООТВЕТСТВИЯ

Функция принадлежности	**Лингвистическое значение**
Отклонение скорости	
a1	очень маленькое
a2	маленькое
a3	много ниже среднего
a4	ниже среднего
a5	среднее
a6	выше среднего
a7	много выше среднего
a8	большое
a9	очень большое
Отклонение тока	
b1	маленькое
b2	ниже среднего
b3	среднее
b4	выше среднего
b5	большое

Таблица 5.2

СИСТЕМА ПРАВИЛ

Откл. тока / **Откл. ск-ти**	**Маленькое**	**Ниже среднего**	**Среднее**	**Выше среднего**	**Большое**
Очень маленькое	увеличение момента нагрузки в допустимых пределах	увеличение момента инерции в допустимых пределах	увеличение сопротивления якорной цепи	недопустимое увеличение сопротивления якорной цепи	недопустимое увеличение сопротивления якорной цепи
Маленькое	увеличение момента нагрузки в допустимых пределах	увеличение момента инерции в допустимых пределах	увеличение сопротивления якорной цепи	–	недопустимое увеличение сопротивления якорной цепи
Много меньше среднего	увеличение момента нагрузки в допустимых пределах	увеличение момента инерции в допустимых пределах	увеличение сопротивления якорной цепи	–	недопустимое увеличение сопротивления якорной цепи
Меньше среднего	увеличение момента нагрузки в допустимых пределах	увеличение момента инерции в допустимых пределах	увеличение сопротивления якорной цепи	–	недопустимое увеличение сопротивления якорной цепи
Среднее	увеличение сопротивления якорной цепи	увеличение момента нагрузки в допустимых пределах	увеличение момента инерции в допустимых пределах	–	недопустимое увеличение сопротивления якорной цепи
Больше среднего	увеличение сопротивления якорной цепи	превышение допустимого момента нагрузки	превышение допустимого момента инерции	недопустимое увеличение сопротивления якорной цепи	недопустимое увеличение сопротивления якорной цепи
Много больше среднего	недопустимое увеличение сопротивления якорной цепи	превышение допустимого момента нагрузки	превышение допустимого момента инерции	недопустимое увеличение сопротивления якорной цепи	–
Большое	недопустимое увеличение сопротивления якорной цепи	–	превышение допустимого момента нагрузки	превышение допустимого момента инерции	–
Очень большое	недопустимое увеличение сопротивления якорной цепи	недопустимое увеличение сопротивления якорной цепи	превышение допустимого момента нагрузки	превышение допустимого момента инерции	–

9. Дважды щелкнем на среднем (белом) блоке, при этом раскроется окно еще одной программы – редактора правил («Rule Editor»). Введем соответствующие правила. При вводе каждого правила необходимо обозначить соответствие между каждой функцией принадлежности входных переменных и числовым значением выходных переменных. Выберем в левых полях (с заголовками «отклонение скорости» и «отклонение тока») вариант, соответствующий состоянию электродвигателя, описанному в полях выходов, и нажмем кнопку «Addrule» («Добавить правило»). Введенное правило появится в окне правил. Аналогично поступим для всех других значений входных переменных, в результате чего сформируется набор правил (рис. 5.10). Закроем окно редактора правил и возвратимся в окно FIS-редактора. Построение системы закончено, и можно начать эксперименты по ее исследованию. Заметим, что для большинства опций были сохранены значения по умолчанию.

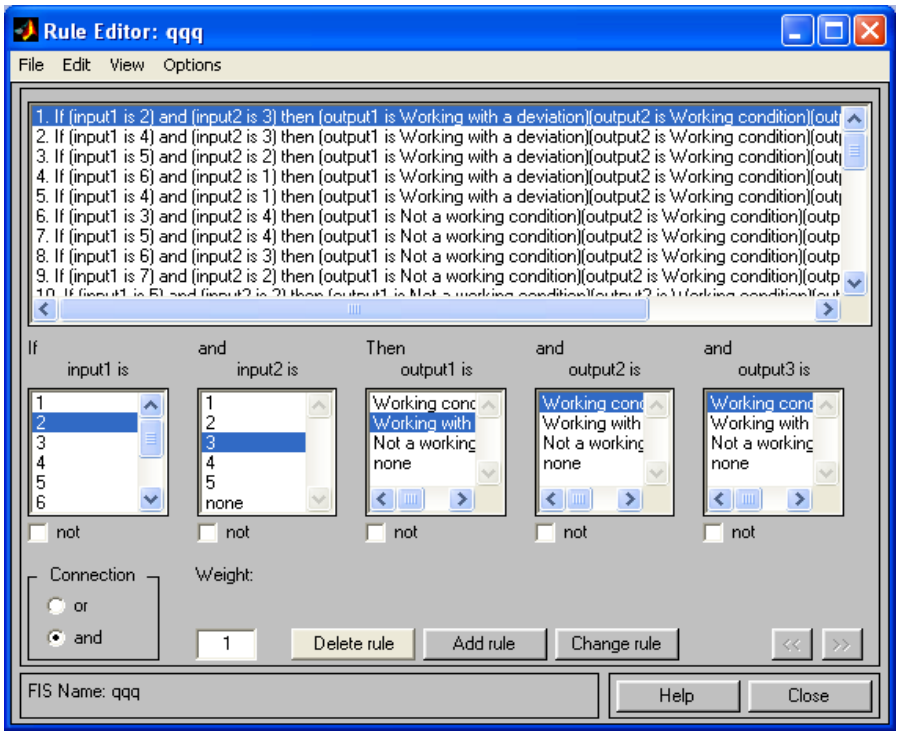


Рис. 5.10. Окно редактора правил

10. Предварительно сохраним на диске (используя команды меню «File→Savetodiskas..») созданную систему под каким-либо именем.

Раскроем меню «View». С помощью его команд «Edit membership functions» и «Edit rules» можно совершить переход к двум рас-

смотренным ранее программам – редакторам функций принадлежности и правил (то же можно сделать нажатием клавиш Ctrl+2 и Ctrl+3), но сейчас нас будут интересовать две другие команды – «View rules» (Просмотр правил) и «View surface» (Просмотр поверхности). Выберем команду «View rules», при этом откроется окно (рис. 5.11) еще одной программы – Просмотра правил (Rule Viewer).

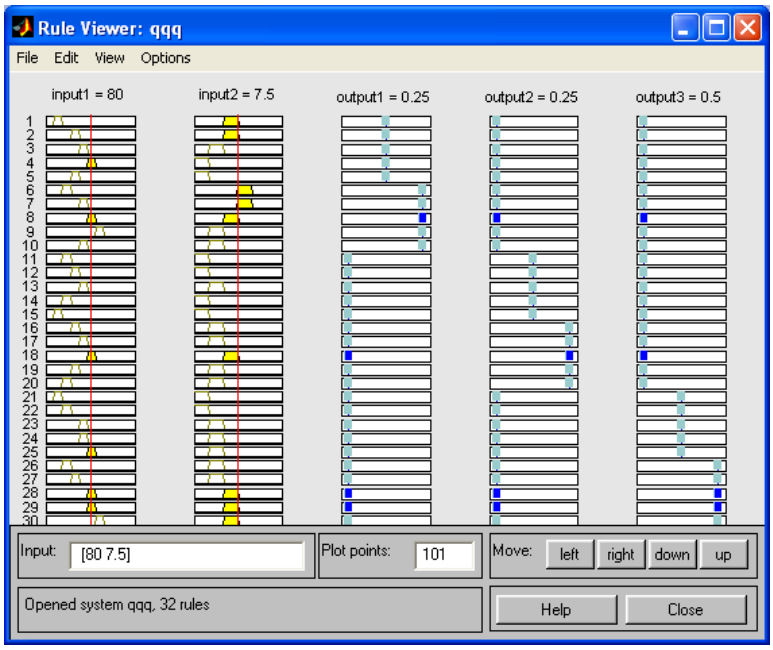


Рис. 5.11. Окно Просмотра правил

11. В левой части окна в графической форме представлены функции принадлежности входных переменных «отклонение скорости» и «отклонение тока», в правой – функции принадлежности выходных переменных с пояснением механизма принятия решения. Красная вертикальная черта, пересекающая графики в левой части окна, которую можно перемещать с помощью мыши, позволяет изменять значения переменной входа (это же можно делать, задавая числовые значения в поле «Input» (Вход)), при этом соответственно изменяются значения выходных параметров правой верхней части окна. Таким образом, с помощью построенной модели и окна просмотра правил можно просмотреть все возможные варианты комбинаций входных параметров «отклонение скорости» и «отклонение тока» и соответствующие им выходные переменные. Изменение аргумента путем перемещения красной вертикальной линии очень наглядно демонстрирует, как система определяет значения выхода.

12. Закроем окно просмотра правил и выбором команды меню «View→View surface» перейдем к окну просмотра поверхности отклика (выхода) (рис. 5.12). Видно, что смоделированное системой отображение показывает зависимость выхода от конкретных значений входных величин. В поле Output можно выбрать выход, для которого необходимо просмотреть зависимость от входных параметров.

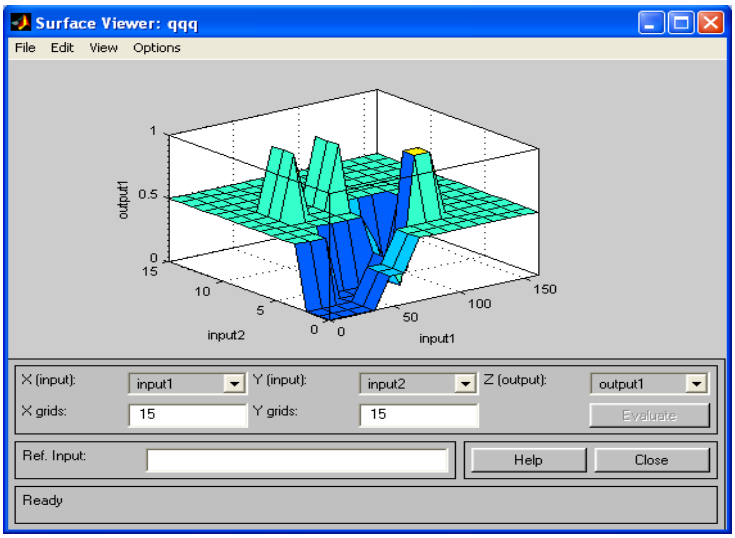


Рис. 5.12. Окно Просмотра поверхности отклика

С помощью вышеуказанных программ-редакторов на любом этапе проектирования нечеткой модели в нее можно внести необходимые коррективы вплоть до задания какой-либо особенной пользовательской функции принадлежности. Из опций, устанавливаемых в FIS-редакторе по умолчанию при использовании алгоритма Сугэно, можно отметить следующие:

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

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

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

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

Работа Fuzzy Logic с«Simulink»

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

Блоки для пакета «Simulink»

FUZBLOCK

Библиотека блоков для внедрения нечетких контроллеров в «Simulink»-модули.

Синтаксис: ***fuzblock.***

Библиотека содержит следующие блоки:

- «Fuzzy Logic Controller» – нечеткий контроллер;
- «Fuzzy Logic Controller with Ruleviewer» – нечеткий контроллер с выводом окна RuleViewer во время моделирования в пакете «Simulink»;
- «Membership Functions» – библиотека функций принадлежностей.

Для включения системы нечеткого логического вывода в «Simulink»-модуль необходимо выбрать блок «Fuzzy Logic Controller» или «Fuzzy Logic Controller with Ruleviewer», затем сделать двойной щелчок по этому блоку и в появившемся диалоговом окне ввести имя файла или наименование переменной в рабочей области, соответствующее системе нечеткого логического вывода. Если система нечеткого логического вывода имеет несколько входов, тогда в «Simulink»-модуле эти входы необходимо мультиплексировать вместе до ввода в нечеткий контроллер. Аналогично, если система нечеткого логического вывода имеет несколько выходов, тогда выходные сигналы блока будут представлены одной мультиплексной линией.

Для построения нетиповых нечетких контроллеров, т. е. отличных от блоков «Fuzzy Logic Controller» и «Fuzzy Logic Controller with Ruleviewer», можно использовать блоки, входящие в библиотеку функций принадлежности («Membership Functions»):

- «Diff. Sigmoidal MF» – разница двух сигмоидных функций принадлежности;
- «Gaussian MF» – гауссовская функция принадлежности;

- «Gaussian 2MF» – комбинация двух гауссовских функций принадлежности;
- «Generalized Bell MF» – обобщенная колоколообразная функция принадлежности;
- «Pi-shared MF» – пи-подобная функция принадлежности;
- «Probabilistic OR» – вероятностная реализация логической операции «ИЛИ»;
- «Probabilistic Rule Agg» – вероятностная реализация операции объединения правил;
- «Prod. Sigmoidal MF» – произведение двух сигмоидных функций принадлежности;
- «S-shaped MF» – *S*-подобная функция принадлежности;
- «Sigmoidal MF» – сигмоидная функция принадлежности;
- «Trapezoidal MF» – трапециевидная функция принадлежности;
- «Triangular MF» – треугольная функция принадлежности;
- «Z-shaped MF» – *Z*-подобная функция принадлежности.

SFFIS

Оптимизированная под «Simulink» функция нечеткого логического вывода.

Синтаксис: ***output = sffis (t, x, u, flag, fis).***

Функция «sffis» – это *MEX*-файл, специально оптимизированный для использования в пакете «Simulink». Функция «sffis» выполняет нечеткий логический вывод аналогично функции «evalfis». Функция «sffis» имеет 5 входных аргументов:

- *t*, *x*, и «flag» – стандартные аргументы *S*-функций пакета «Simulink»;
- *u* – вектор значений входных переменных, для которых необходимо осуществить нечеткий логический вывод;
- «fis» – система нечеткого логического вывода.

Функция «sffis» возвращает единственный аргумент *output*, содержащий результат нечеткого логического вывода.

В данной случае была разработана модель электродвигателя, включаемая параллельно реальному двигателю. Сигналы с их выходов сравниваются и подаются на вход нечеткого контроллера. На рис. 5.13 изображена модель системы диагностики.

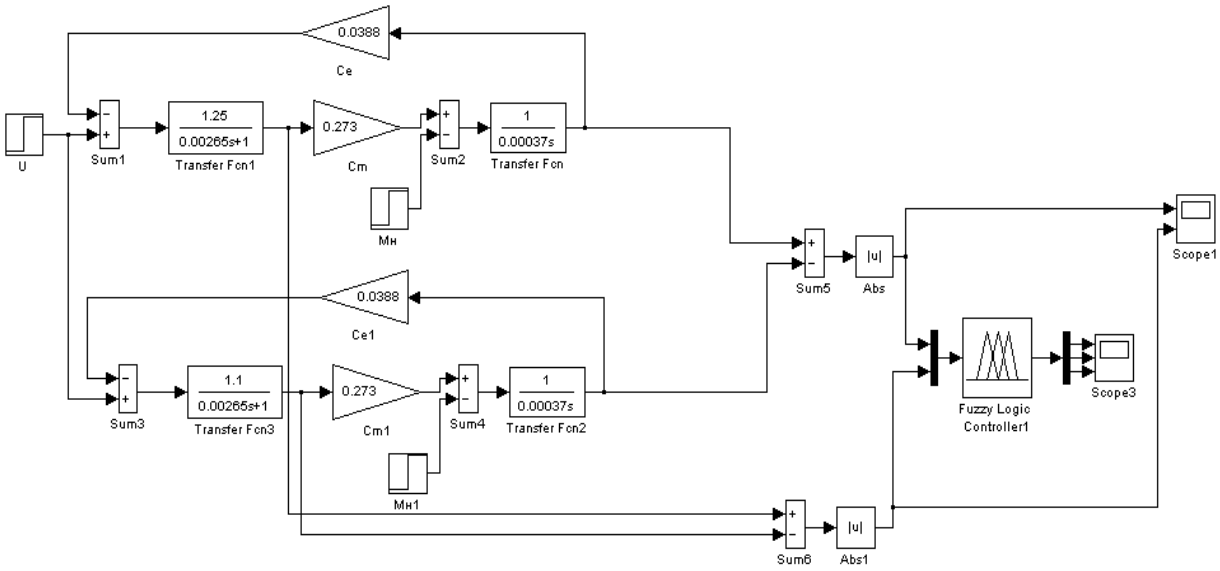


Рис. 5.13. Модель системы диагностики

Состояние электродвигателя оценивается по величине отклонения выходных сигналов. В зависимости от значений скорости и тока нечеткий контроллер на определенном выходе формирует сигнал, соответствующий правилам, прописанным в окне редактора правил.

5.4. Результаты моделирования нечеткой системы диагностики

На рис. 5.14 представлены результаты работы нечеткой системы диагностики электродвигателя, работающего при номинальных значениях. Из графиков видно, что на всех выходах нечеткой системы отсутствуют отклонения, что свидетельствует о ее исправности. Моделируем различные режимы работы двигателя. Чтобы получить выходные характеристики при перегреве двигателя, меняем сопротивление якорной цепи. Результаты представлены на рис. 5.15 и 5.16. На рис. 5.17, 5.18 и рис. 5.19, 5.20 получены выходные сигналы нечеткой системы при изменении момента инерции и при увеличении нагрузки на валу двигателя соответственно.

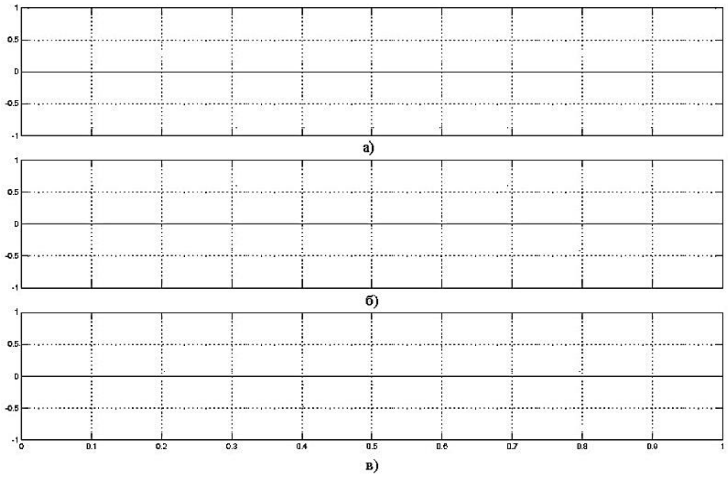


Рис. 5.14. Результаты моделирования системы диагностики при номинальных параметрах: *а* – перегрев двигателя; *б* – изменение момента инерции, *в* – увеличение момента на валу двигателя

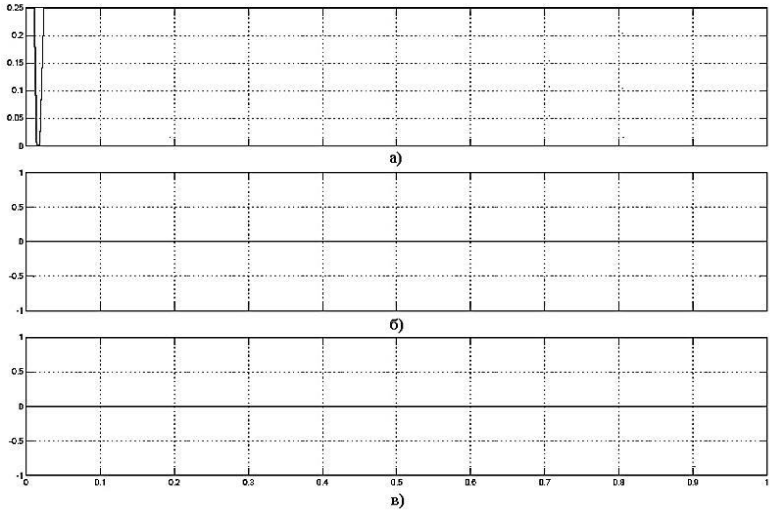


Рис. 5.15. Результаты моделирования при изменении температуры двигателя

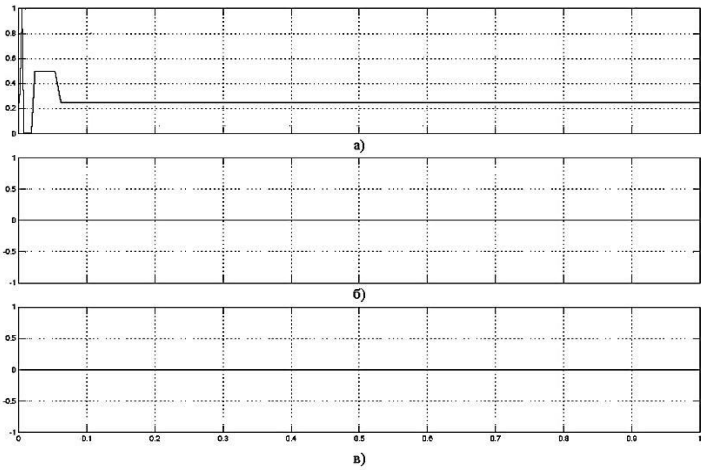


Рис. 5.16. Результаты моделирования при дальнейшем нагреве двигателя

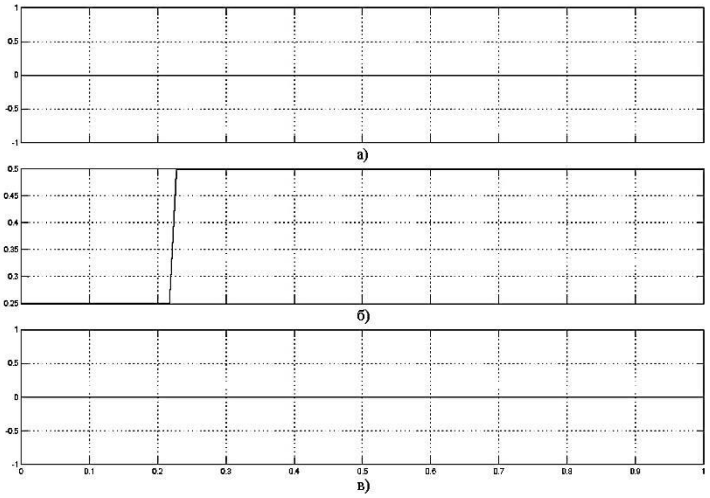


Рис. 5.17. Результаты моделирования
при изменении момента инерции

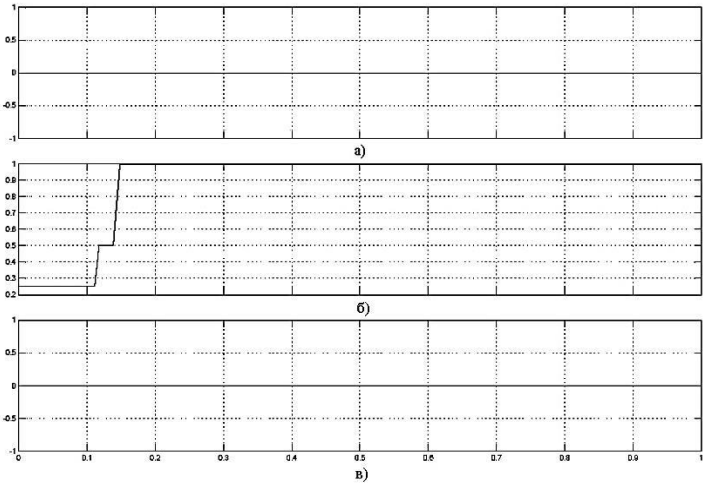


Рис. 5.18. Результаты моделирования
при дальнейшем изменении момента инерции

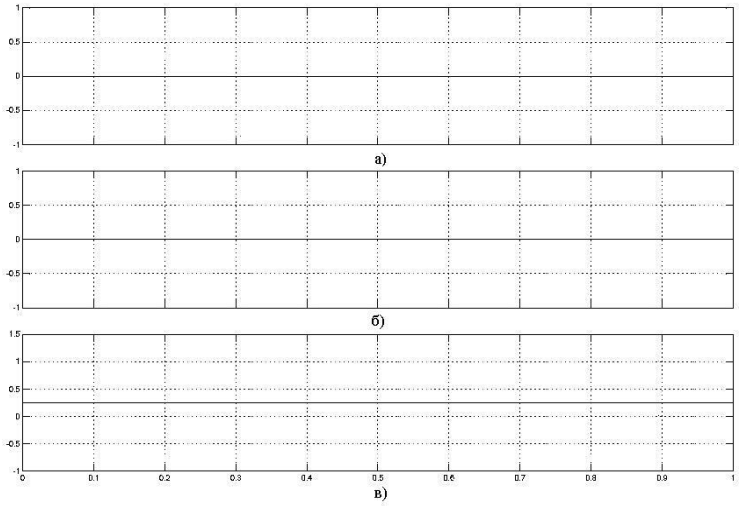


Рис. 5.19. Результаты моделирования
при увеличении нагрузки на валу двигателя

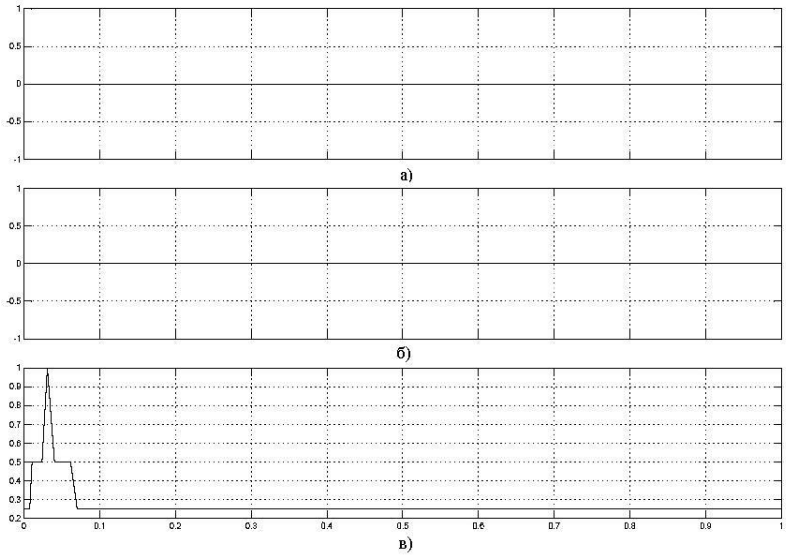


Рис. 5.20. Результаты моделирования
при дальнейшем увеличении нагрузки
на валу двигателя

В процессе работы двигателя при изменении каких-либо параметров на выходе нечеткой системы диагностики, соответствующей конкретной неисправности, формируется сигнал, при оценке которого можно делать выводы о состоянии двигателя.

Преимущества нечетких систем

Преимущества *fuzzy*-систем по сравнению с другими:

- возможность оперировать нечеткими входными данными: например, непрерывно изменяющимися во времени значениями (динамические задачи), значениями, которые невозможно задать однозначно (результаты статистических опросов, рекламные компании и т.д.);
- возможность нечеткой формализации критериев оценки и сравнения: оперирование критериями «большинство», «возможно», «преимущественно» и т.д.;
- возможность проведения качественных оценок как входных данных, так и выходных результатов: вы оперируете не только значениями данных, но и их степенью достоверности (не путать с вероятностью!) и ее распределением;
- возможность проведения быстрого моделирования сложных динамических систем и их сравнительный анализ с заданной степенью точности: оперируя принципами поведения системы, описанны-

ми *fuzzy*-методами, вы, во-первых, не тратите много времени на выяснение точных значений переменных и составление описывающих уравнений, во-вторых, можете оценить разные варианты выходных значений.

Рассмотренный вариант построения систем диагностирования, основанный на применении нечеткой логики, показал значительные преимущества данных систем. Анализ полученных результатов моделирования позволяет сделать выводы о высокой точности определения состояния электромеханической системы с помощью данного метода. При использовании нечеткой логики снижается время идентификации состояния, что позволяет своевременно реагировать на изменение каких-либо параметров и вовремя предупредить неисправность.

Материалы по использованию функций нечеткой логики приведены в прил. 4, 5.

Контрольные задания*

1. К основным понятиям теории нечетких множеств относятся:

a) нечеткое число;

b) нечеткая переменная;

c) лингвистическое число;

d) лингвистическая переменная.

2. Семантические правила М, определяют:

a) новые термы с использованием квантификаторов;

b) функции принадлежности новых значений, определенных в G;

c) соответствие между лингвистическим значением и нечетким множеством;

d) наименование лингвистической переменной.

3. К операциям над нечеткими множествами относят:

a) геометрические операции над нечеткими множествами;

b) теоретико-множественные операции над нечеткими множествами;

c) арифметические операции над нечеткими числами;

d) логические операции.

---

* Из предложенных ответов выберите правильный. Проверьте себя по прил. 6.

4. Отличием нечеткой арифметики от традиционной является:

a) отсутствие операции вычитания;
b) отсутствие операции сложения;
c) выполнение операций над нечеткими числами;
d) нет правильного ответа.

5. Правила выполнения нечетких логических операций определяются с помощью:

a) логических правил;
b) принципа обобщения Заде;
c) арифметических правил.

6. ПАКЕТ ДЛЯ СОЗДАНИЯ АЛГОРИТМОВ «STATEFLOW»

Общие положения

Цифровые системы, осуществляющие управление в реальном масштабе времени и использующие ЭВМ для формирования закона управления, являются весьма сложными для анализа гибридными объектами. В настоящее время такие системы, как правило, выполняются на базе многозадачных управляющих контроллеров и имеют сетевую организацию. Различные части системы имеют разную природу (непрерывный объект и дискретная управляющая часть), а система в целом описывается сложной комбинацией дифференциальных уравнений, алгебраических уравнений и неравенств и логических условий. Поскольку для преобразований аналог-код и код-аналог и других вычислений, а также для передачи информации по сети требуется определенное время, при реализации цифрового управления возникает временная задержка. Это приводит к снижению качества управления, иногда до недопустимо низкого уровня. Чтобы избежать негативного влияния такого запаздывания и оптимально использовать доступные системе управления вычислительные ресурсы, проектирование алгоритмов управления и программного обеспечения должно вестись с учетом данного фактора.

В теории цифрового управления интервалы осуществления выборки обычно принимаются одинаковыми, а задержка управления считается несущественной или постоянной. Однако на практике это

имеет место лишь в редких случаях. В контроллере задачи накладываются друг на друга и блокируются в ожидании общих ресурсов. Время выполнения самих задач может изменяться. Передача данных по сети происходит с задержками, величина и стабильность которых зависит как от протокола связи, так и от загруженности каналов. В этих условиях аналитический анализ поведения цифровых систем управления затруднен, наиболее естественным представляется путь имитационного моделирования.

Разработка имитационной модели гибридной системы, характеризующейся переменным значением шага квантования, может вестись различными способами. Среди готовых инструментов моделирования можно выделить построенный на базе пакета MATLAB симулятор «TrueTime». Желающие оставаться в рамках стандартного набора MATLAB могут построить «Simulink»-модель, используя в качестве примера модель дискретной системы с переменным тактом квантования.

Еще один вариант, представляющийся наиболее естественным, заключается в применении наряду с «Simulink» программы «Stateflow». Специально разработанный для моделирования дискретных управляющих устройств и систем «Stateflow» дополняет и расширяет возможности «Simulink». В «Simulink» достаточно просто моделируются непрерывные объекты. Используя язык диаграмм состояний и переходов, можно получить в «Stateflow» описание управляющей части моделируемой системы. При этом алгоритм управления может иметь сколь угодно высокую сложность, временные задержки могут быть как детерминированными, так и носить случайный характер. Дополнительная возможность следить в режиме реального времени за процессом выполнения «Stateflow»-диаграммы путем включения режима анимации делает процесс моделирования наглядным.

Описание работы «Stateflow»

Действия – то, что происходит при выполнении «Stateflow»-диаграммы. Действия могут быть выполнены при переходе из одного состояния в другое или вследствие приобретения состоянием статуса активного.

Переходы могут иметь действия условий (*conditionactions*) и действия переходов (*transitionactions*).

Например, как изображено на рис. 6.1.

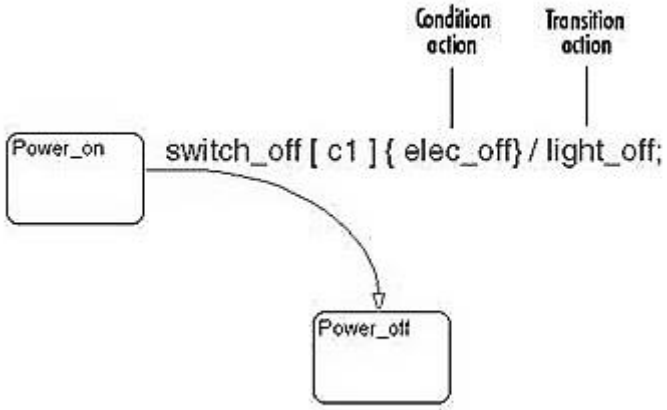


Рис. 6.1. Условия переходов

Состояния могут иметь действия при входе в состояние (*entry*), действие во время активности состояния (*during*), при выходе из состояния (*exit*), действие при наступлении события с именем *event_name*.

Например, как изображено на рис. 6.2.

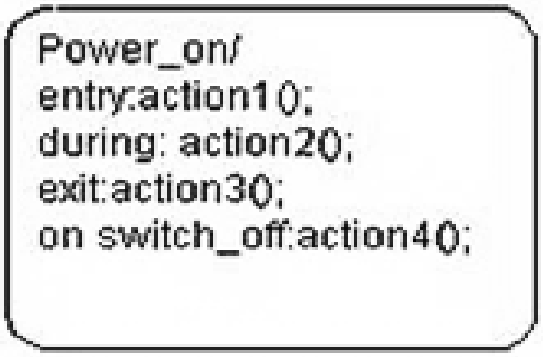


Рис. 6.2. Действия для состояний

Если Вы после имени состояния и символа / (*backslash*) указываете действия, не используя ключевое слово *entry*, действия интерпретируются как действия при входе в состояние. То, какое именно действие будет выполнено, определяется инструкциями на языке действий (*Action Language*). Действие может изменить значение некоторой переменной величины, вызвать функцию или событие.

«ChartInstance» – экземпляр диаграммы

Экземпляр диаграммы – ссылка из «Stateflow»-модели на диаграмму, которая находится в библиотеке «Simulink». Диаграмма в библиотеке может иметь много экземпляров. Обновление диаграммы в библиотеке автоматически обновляет все экземпляры диаграммы.

«Condition» – условие

Условие – это заключенное в квадратные скобки логическое выражение, определяющее, произойдет переход или нет. Например, [*speed>threshold*] – это условие.

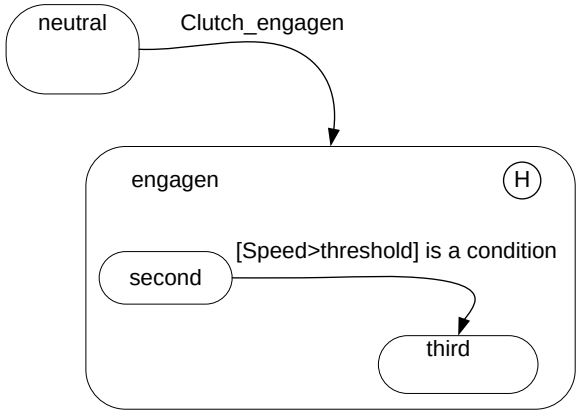


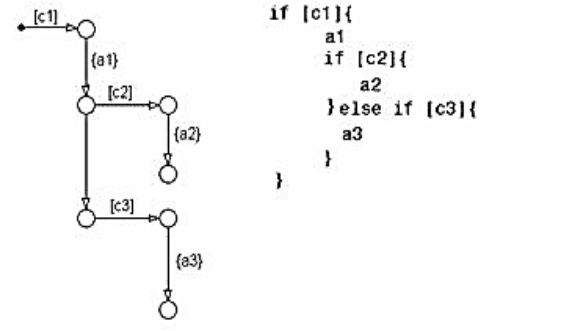
Рис. 6.3. Условие срабатывания перехода

Правила описания условий определяет язык действий (*Action Language*).

«ConnectiveJunction» – подключаемое соединение

Подключаемые соединения (точки принятия решений) – графические объекты, упрощающие вид «Stateflow»-диаграмм и позволяющие получать более эффективные коды. Подключаемое соединение – это альтернативный путь получения систем с желаемым поведением.

Пример, изображенный на рис. 6.4, показывает, как подключаемое соединение (в виде маленьких кружков) используется для изображения *if*-структуры.




```
if [c1]{
     a1
     if [c2]{
        a2
     }else if [c3]{
       a3
     }
}
```

Название	Кнопка	Описание
Подключаемое соединение		Подключаемое соединение используется в ситуациях, когда переходы соединяют одно состояние с двумя или более состояниями, но при разных условиях.

Рис. 6.4. Изображение if-структуры

«Data» – данные

Объекты данные – это числовые значения, используемые в «Stateflow»-диаграмме.

«DataDictionary» – словарь данных

Словарь данных – это база, в которой хранится информация «Stateflow»-диаграммы. Когда Вы создаете объекты «Stateflow»-диаграммы, информация о них записывается в словаре данных в момент сохранения диаграммы.

«Decomposition» – декомпозиция

К состоянию можно применить декомпозицию, когда оно состоит из нескольких подсостояний. К диаграмме, состоящей хотя бы из одного состояния, тоже можно применить декомпозицию. Для представления иерархии состояний существует несколько правил, в соответствие с которыми состояния группируются. Суперсостояние имеет параллельную (И) или последовательную (ИЛИ) декомпозицию. Все подсостояния, находящиеся на одном уровне иерархии, должны иметь одинаковую декомпозицию.

Параллельная (И) декомпозиция

Ее можно распознать по пунктирным границам состояний. Это представление используется, если все состояния на данном уровне иерархии активны одновременно. Активность параллельных состояний независима.

Последовательная (ИЛИ) декомпозиция

Она отображается состояниями с непрерывными границами. Эта декомпозиция используется в описании систем, режимы которых последовательны. На данном уровне иерархии в данное время может быть активным только одно состояние.

«Default Transition» – безусловные переходы

Безусловные переходы в основном используются для определения активности последовательных (ИЛИ) состояний, когда есть неопределенность между двумя или несколькими последовательными состояниями. Чтобы создать безусловный переход, выберите кнопку безусловного перехода на панели инструментов и переместите его к объекту. Безусловные переходы – это переходы, не имеющие объекта-источника (рис. 6.5).

Название	Кнопка	Описание
Безусловный переход		Используйте безусловный переход на каждом уровне иерархии для указания на состояние, которое должно стать активным по умолчанию.

Рис. 6.5. Безусловный переход

«Events» – события

События управляют выполнением диаграммы. Все события, управляющие «Stateflow»-диаграммой, должны быть определены. Когда событие происходит, это изменяет статус состояния. Поступление события может вынудить переход состояться и/или действие совершиться. События передаются сверху вниз, начиная с родительского уровня иерархии. События добавляются, удаляются и редактируются в проводнике («Stateflow Explorer»).

«Finite State Machine» – конечный автомат

Конечный автомат – форма представления управляемой событиями системы. Конечный автомат также используется для описания реактивных систем. В реактивных системах или в управляемых событиями системах переход системы из одного режима или состояния в другой режим или состояние происходит, если условие, определяющее изменение, истинно.

«Flow Graph» – граф потока

Граф потока – это набор путей потока. Он начинается с сегмента перехода, который, в свою очередь, начинается с состояния или с сегмента безусловного перехода.

«Flow Path» – путь потока

Путь потока – это упорядоченная последовательность сегментов перехода и соединений, где каждый последующий сегмент начинается с соединения, которое закончило предыдущий сегмент.

«Flow Subgraph» – подграф потока

Подграф – набор путей, которые начинаются с одного и того же сегмента перехода.

«GraphicalFunction» – графическая функция

Графическая функция – функция, логика которой определена графом потока.

«Hierarchy» – иерархия

Иерархия позволяет создать сложные системы, помещая состояния в другие состояния более высокого уровня. Иерархическая модель обычно уменьшает число переходов и создает ясные, более наглядные диаграммы.

«History Junction» – соединение с памятью

Соединение с памятью позволяет определить подсостояние, к которому будет произведен переход, основываясь на информации о предыстории. Если суперсостояние имеет соединение с памятью, переход осуществляется к тому подсостоянию, которое было активно последним. Соединение с памятью применяется к тому уровню иерархии, на котором оно имеется (рис. 6.6).

Название	Кнопка	Описание
Соединение с памятью		Соединение с памятью показывает (при попадании на этот уровень иерархии), что состояние, которое было активным, вновь станет активным.

Рис. 6.6. Пиктограмма соединения с памятью

«Inner Transitions» – внутренние переходы

Внутренний переход – переход, который не выводит из текущего состояния. Внутренние переходы наиболее эффективны, когда определены для суперсостояний с *XOR* (исключающее «ИЛИ») декомпозицией. Использование внутренних переходов может упростить диаграмму «Stateflow».

«Library Link» – библиотечная ссылка

Библиотечная ссылка – ссылка на диаграмму, которая хранится в библиотечной модели в библиотеке блоков «Simulink».

«Library Model» – библиотечная модель

Библиотечная модель – модель «Stateflow», которая хранится в библиотеке «Simulink». Вы можете включать диаграммы из библиотеки в Вашу модель, копируя их, при этом «Stateflow» физически не включает диаграмму в Вашу модель, а создает ссылку на диаграмму из библиотеки. Можно создавать множество ссылок на одну диаграмму. Каждая ссылка будет экземпляром диаграммы. Когда Вы включаете диаграмму из библиотеки в Вашу модель, Вы также включаете ее машину состояний. Таким образом, «Stateflow»-модель, которая включает ссылки на библиотеки диаграмм, имеет множество машин состояний. Когда «Stateflow» создает модель, которая включает диаграммы из библиотеки моделей, то вставляются все диаграммы из нее, даже если имеются ссылки только на некоторые из моделей. Однако, когда «Stateflow» производит автономный или *RTW* код, то

включаются только те диаграммы, на которые есть ссылки. Модель, которая включает ссылки на библиотечную модель, может моделироваться только, если все диаграммы в библиотечной модели свободны от ошибок.

«Machine» – машина

Машина – собрание всех «Stateflow»-блоков, определенных «Simulink»-моделью за исключением экземпляров диаграммы (библиотечных ссылок). Если модель включает библиотечные ссылки, то она также включает машины состояний, определенные моделями, из которых ссылки исходят.

«Notation» – нотация

Нотация определяет набор объектов и правил, которые устанавливают отношениями между этими объектами. Нотация «Stateflow» описывает язык, который сообщает модели информацию, заключенную в диаграмме «Stateflow». Нотация состоит:

- из набора графических объектов;
- набора неграфических (текстовых) объектов;
- определенных отношений между этими объектами.

«Parallelism» – параллелизм

Система с параллелизмом может иметь два или больше состояний, которые могут быть активными одновременно. Активность параллельных состояний независима. Параллелизм представлен параллельной (И) декомпозицей состояния.

«Real-Time Workshop (RTW)» – мастерская реального времени

Мастерская реального времени – автоматический генератор кода языка*C* для «Simulink». Она производит код языка*C* непосредственно по модели «Simulink»-диаграммы и автоматически строит программы, которые могут исполняться в реальном времени в различных режимах.

«RTWTarget-RTW» код

RTW код – исполняемый код, построенный из кода, произведенного *RTW*.

«S-Function» (S-функция)

При использовании «Simulink» вместе с «Stateflow» для моделирования «Stateflow» производит S-функцию (*MEX-file*) для каждой «Stateflow»-машины, чтобы поддержать создание модели. Этот код используется для моделирования и называется S-функцией.

«Semantics» – семантика

Семантика описывает, как нотация интерпретируется и реализуется. Законченная диаграмма «Stateflow» сообщает, как система будет себя вести. Диаграмма «Stateflow» содержит действия, связанные с переходами и состояниями. Семантика описывает, в какой последовательности эти действия выполняются в ходе выполнения диаграммы «Stateflow».

«Simulink»

«Simulink» – программа для моделирования, создания и анализа динамических систем. «Simulink» позволяет моделировать линейные и нелинейные системы, непрерывные или дискретные. В «Simulink» системы представляются как блок-диаграммы. «Stateflow» – часть этого представления. Блок «Stateflow» – замаскированная «Simulink»-модель. «Stateflow» строит *S*-функцию, которая соответствует каждой «Stateflow»-машине. Контролируемое поведение, которое модель «Stateflow» дополняет алгоритмическим поведением, моделируется в блок-диаграммы «Simulink». Включая диаграмму «Stateflow» в модель «Simulink», можно добавлять событийное поведение в созданную в «Simulink» модель. Вы создаете модели, которые представляют данные и управляют потоком, дополняя «Stateflow»-блоками стандартные наборы «Simulink»-блоков. Эти смешанные модели создаются, используя «Simulink».

«State» – состояние

Состояниями описывается поведение реактивной системы. Реактивная система имеет много возможных состояний. Состояния в диаграмме «Stateflow» представляют их. Активность или неактивность состояний динамически изменяется, основываясь на переходах между событиями и условиях. Каждое состояние имеет иерархию. Если диаграмма состоит из одного состояния, то предок этого состояния – са-

ма диаграмма. Состояние также может иметь память, которая применяется к его уровню в иерархии диаграммы. Состояния могут иметь действия, которые выполняются в последовательности, базирующейся на типах действий. Типы действий: при входе, в течение, при выходе, при наступлении события с именем event_name (рис. 6.7).

Название	Кнопка	Описание
Состояние		Используйте состояния, чтобы описать режимы работы системы.

Рис. 6.7. Пиктограмма состояния

Блок «Stateflow»

Блок «Stateflow» – замаскированная «Simulink»-модель, эквивалентная пустой диаграмме без имени. Используйте «Stateflow»-блок, чтобы включить «Stateflow»-диаграмму в модель «Simulink». Управляющее поведение, которым модели «Stateflow» дополняют алгоритмическое поведение, формируется в блочных диаграммах. Включая блоки «Stateflow» в модели «Simulink», можно добавить комплексное поведение, управляемое событиями, в имитацию «Simulink». Вы создаете модели, которые представляют как данные, так и поток управления посредством объединения блоков «Stateflow» со стандартными библиотеками моделей «Simulink». Эти комбинированные модели сымитированы с использованием «Simulink».

«Stateflow Debugger» – отладчик «Stateflow»

Используйте отладчик, чтобы отладить и анимировать ваши «Stateflow»-диаграммы. Отладчик может также использоваться, чтобы выполнить динамическую проверку.

«Stateflow»-диаграмма

Используя «Stateflow», Вы создаете «Stateflow»-диаграммы. «Stateflow»-диаграмма является графическим представлением конечного автомата, где состояния и переходы формируют основные строительные блоки системы.

«Stateflow Explorer» – «Stateflow»-проводник

Используйте проводник, чтобы добавлять, удалять и изменять данные, события и целевые объекты.

«Stateflow Finder» – «Stateflow»-поиск

Используйте «Stateflow»-поиск, чтобы отобразить список объектов, базируясь на заданных критериях. Вы можете иметь непосредственный доступ к диалоговому меню свойств любого объекта на дисплее результатов поиска.

«Subchart» – поддиаграмма

Поддиаграмма – это диаграмма, содержащаяся в другой диаграмме.

«Superstate» – подсостояние

Состояние называется подсостоянием, если оно содержится в другом состоянии (суперсостоянии – *substate*) (рис. 6.8).

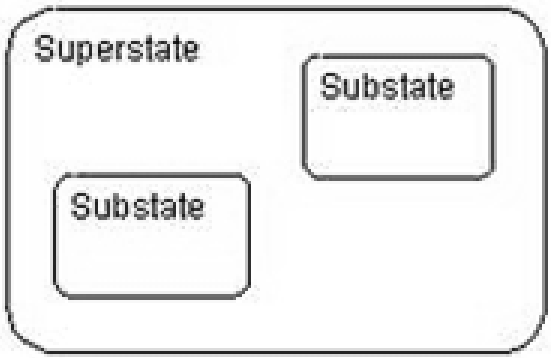


Рис. 6.8. Фрагмент подсостояния

«Superstate» – суперсостояние

Состояние является суперсостоянием, если оно содержит другие состояния (подсостояния) (рис. 6.9).

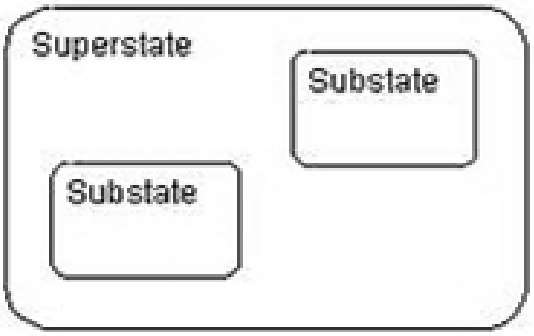


Рис. 6.9. Фрагмент суперсостояния

«Supertransition» – суперпереход

Суперпереход – это переход между объектами, расположенными в разных поддиаграммах.

«Target» – код

Код является выполняемой программой, созданной из кода, сгенерированного «Stateflow» или «Real-TimeWorkshop» (мастерской реального времени).

«TopdownProcessing» – обработка сверху вниз

Обработка сверху вниз имеет отношение к пути, по которому «Stateflow» обрабатывает состояния и события. А именно, «Stateflow» обрабатывает суперсостояния перед состояниями. «Stateflow» обрабатывает состояние, только если сначала активизировано его суперсостояние.

«Transition» – переход

Переход описывает направление, по которому система перемещается из одного состояния в другое. Концы перехода могут подключаться к источнику (начало перехода) и пункту назначения (конец перехода). Чаще всего переходы происходят после того, как случаются какие-то события.

«TransitionPath» – маршрут перехода

Маршрут перехода – это путь потока, который начинается и кончается в состоянии.

«Transition Segment» – сегмент перехода

Сегмент перехода – единичная ориентированная ветвь на диаграмме. Сегменты перехода иногда называют переходами.

«Simulink»\«Stateflow»

Моделирование в «Simulink»\«Stateflow» цифровой системы управления.

Рассмотрим в качестве примера модель цифровой системы управления объектом «перевернутый маятник». Передаточная функция объекта имеет вид

$$W(s)=\frac{1}{s^2-4}.$$

В качестве регулятора выберем звено с передаточной функцией $W(s)=s+5$.

Модель непрерывной системы приведена на рис. 6.10, а переходный процесс в системе стабилизации маятника в вертикальном положении показан на рис. 6.11.

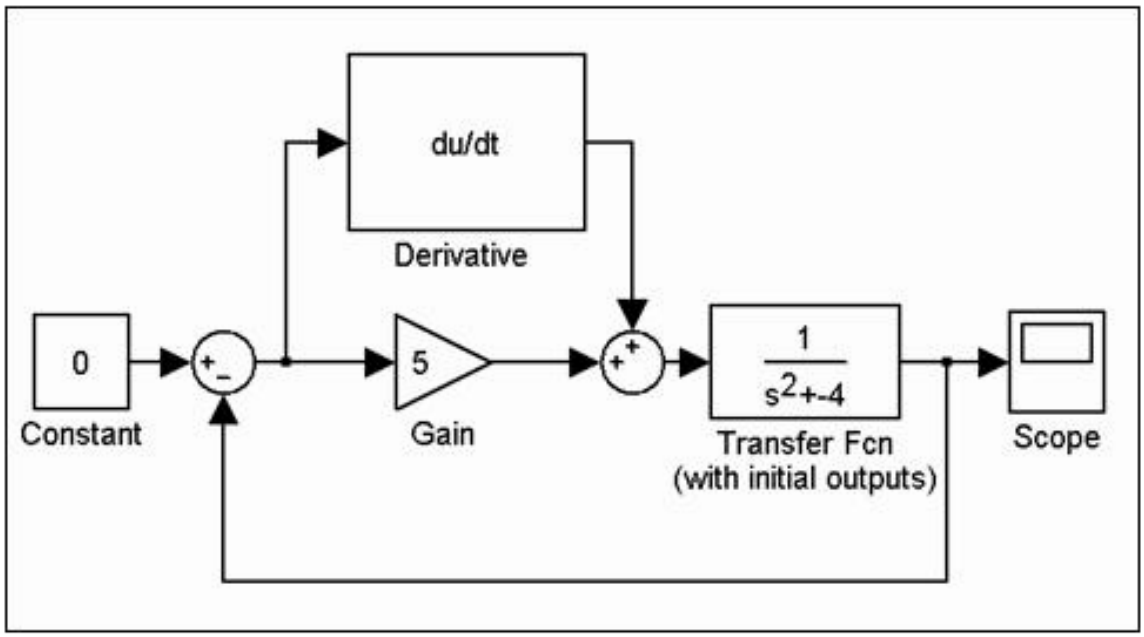


Рис. 6.10. Модель непрерывной системы стабилизации маятника

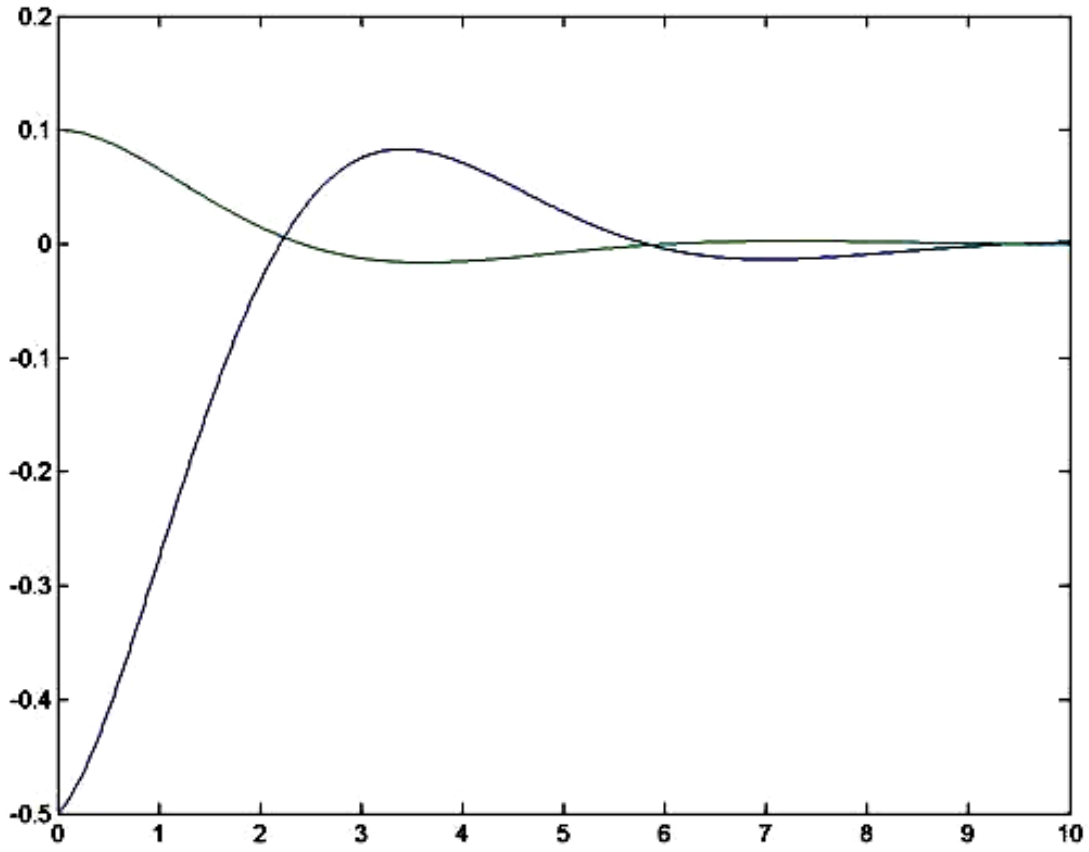


Рис. 6.11. Переходный процесс в непрерывной системе стабилизации маятника

Рассмотрим теперь цифровую систему управления, реализуя регулятор средствами «Stateflow». Модель системы управления примет следующий вид (рис. 6.12).

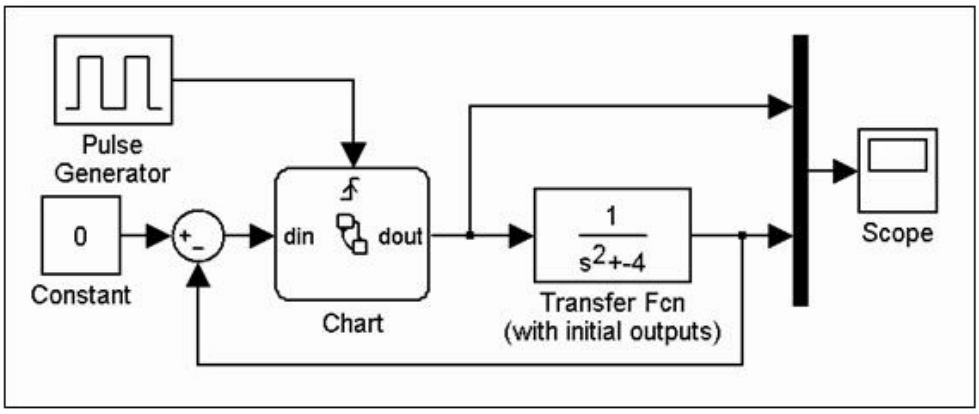


Рис. 6.12. Модель цифровой системы стабилизации маятника

Цифровой вариант регулятора будет иметь передаточную функцию

$$W(s) = 5 + \left(1 - z^{-1}\right) lh,$$

где h – шаг квантования.

«Stateflow»-модель цифрового регулятора представлена на рис. 6.13.

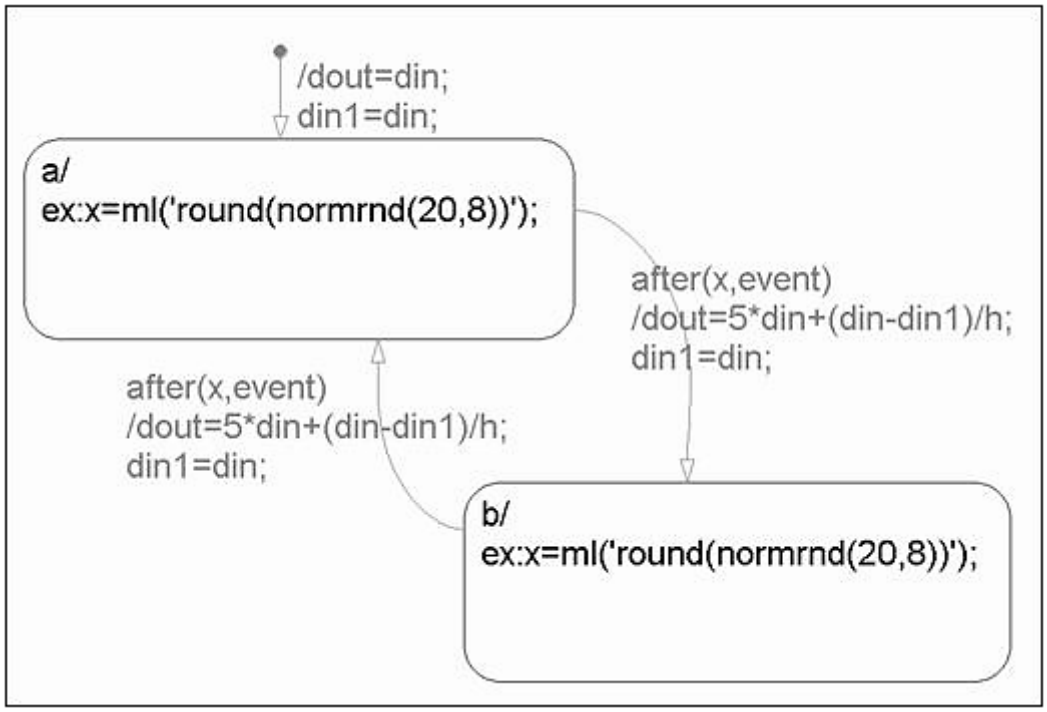


Рис. 6.13. «Stateflow»-модель цифрового регулятора

Эта модель реализует переменную временную задержку в канале управления, изменяющуюся случайным образом в соответствии с нормальным законом с параметрами 0,2 с и 0,08 с. Переход из одного состояния в другое сопровождается вычислением нового значения сигнала управления, который действует на объект в течение следующего такта. Переходные процессы в системе представлены на рис. 6.14.

Для сравнения на рис. 6.15 представлены переходные процессы в системе с постоянным шагом квантования 0,2 с.

Результат работы системы может оцениваться по интегральному среднеквадратичному критерию качества

$$I = \int_0^T \varepsilon^2 dt,$$

где ε – ошибка системы; T – время окончания процесса регулирования.

Исследование рассмотренных выше моделей показало, что наилучшим качеством обладает непрерывная система, для которой I = 0,009985. У цифровой системы с постоянным шагом квантования 0,2 с I = 0,01032. В случае цифровой системы с непостоянным шагом квантования среднее значение критерия равнялось 0,01061, хотя в некоторых редких случаях оно было меньше, чем у непрерывной системы.

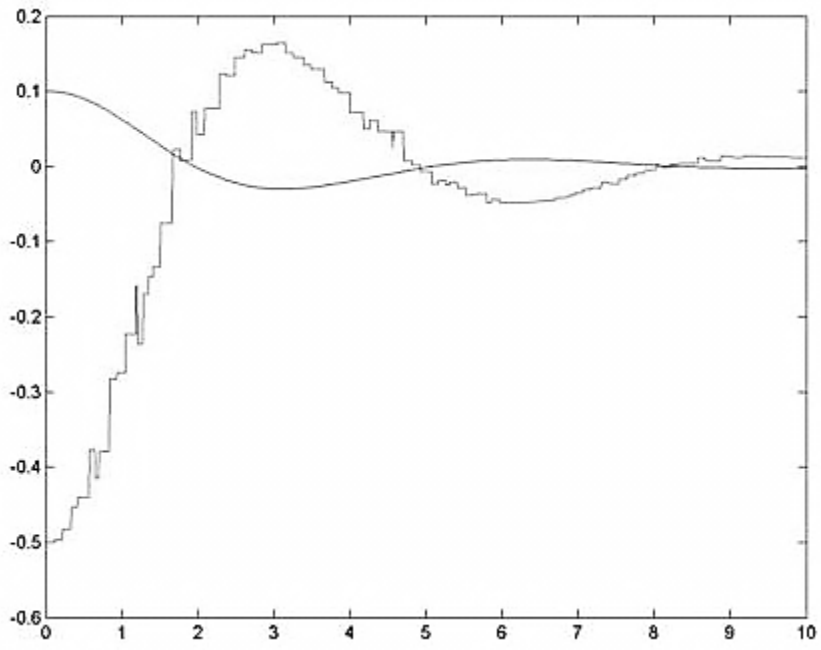


Рис. 6.14. Переходный процесс в дискретной системе
с переменной временной задержкой

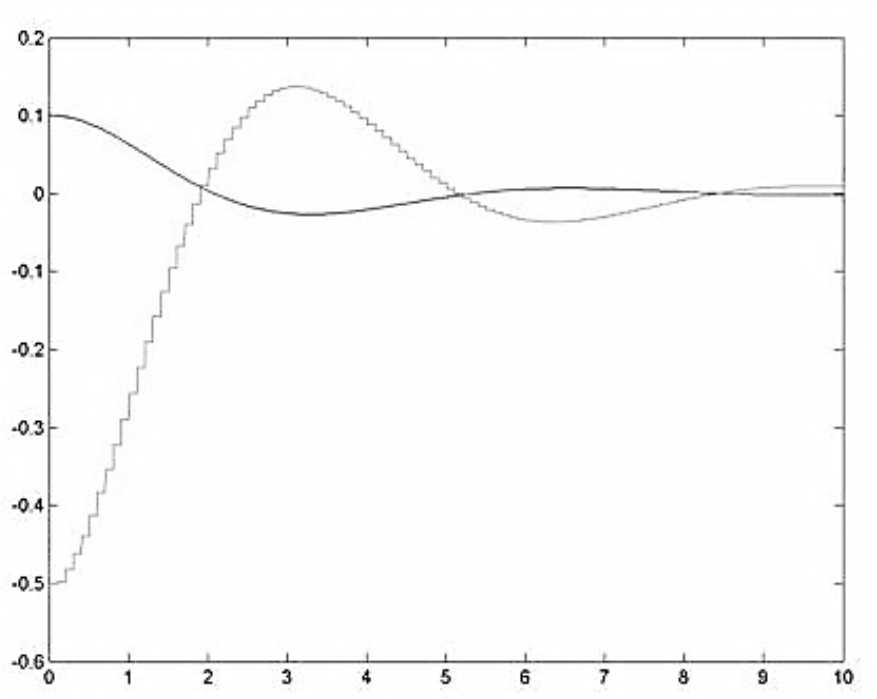


Рис. 6.15. Переходный процесс в дискретной системе
с постоянной временной задержкой

Контрольные вопросы

1. Назовите инструменты моделирования имитационной модели гибридной системы.

2. Перечислите основные составляющие «Stateflow»-диаграмм.

3. Что такое условие («Condition»)?

4. Какие виды декомпозиции используются в «Stateflow»?

5. Опишите принцип использования «Library Model» (библиотечная модель).

ЗАКЛЮЧЕНИЕ

Искусственный интеллект занял одно из ведущих положений в науке и практическом применении. Методы искусственного интеллекта широко используются в системах управления как сложными техническими объектами (роботы, станки, системы автоматизации), так и простыми устройствами (электромеханические устройства бытовой техники). Особое место в диагностике занимают нейронные сети и нечеткая логика, а при поддержке, например, сетями Петри или методами конструирования алгоритмов на основе генетики делают их исключительным средством для создания сложнейших систем, основанных на знаниях.

Область применения искусственного интеллекта в технике поистине безгранична: начиная с электромеханических игрушек и заканчивая сложнейшими космическими системами.

Создание интеллектуальных систем диагностирования нуждается в разработке структур, проверке их работоспособности, анализе функционирования и отладке алгоритмов диагностирования. Это может быть реализовано с использованием систем моделирования.

В структуру алгоритмов входит анализ состояния, выполненный каким-либо образом, с возможностью сформировать вывод, полученный с привлечением различных алгоритмов, построенных на основе нейронных систем, принять решение с использованием нечеткой логики и, если система полностью автоматическая, реализовать его. В противном случае вывести сообщение о рекомендуемых действиях для устранения дефекта в диагностируемой системе.

Для построения системы диагностирования необходимо иметь четыре основных подсистемы: программное обеспечение, пакет MATLAB, измерительную плату NI6221, модуль сопряжения BNC2120 и собственно объект диагностирования, в качестве которого рассматривается электромеханическая система с приводом постоянного тока. Особенность системы заключается в том, что управление работой привода, процессом измерений и обработкой информации осуществляется под управлением пакета «Simulink».

Объем пособия, которое рассматривается как первый шаг в освоении машинных методов использования искусственного интеллекта в системах диагностирования, очень мал, чтобы описать все подробности и тонкости применения интеллектуальных алгоритмов.

Для расширения знаний об интеллектуальных системах следует воспользоваться библиографическим списком, приведенным в пособии. Особое внимание целесообразно уделить электронному ресурсу URL:http:// matlab.exponenta.ru/, в котором подробно описывается как среда программирования MATLAB, так и пакеты NNTool и Fuzzy Logic.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Каллан, Роберт.* Основные концепции нейронных сетей : пер. с англ. / Роберт Каллан. – М. : Вильямс, 2001. – 260 с.

2. *Медведев, В. С.* Нейронные сети. Matlab 6 / В. С. Медведев, В. Г. Потёмкин. – М. : Диалог-МИФИ, 2002. – 496 с. – ISBN 5-86404-163-7.

3. *Омату, Сигеру.* Нейроуправление и его приложения / Сигеру Омату, Марзуки Халид, Рубия Юсоф ; пер. с англ. Н. В. Батина ; под ред. А. И. Галушкина, В. А. Птичкина. – М. : ИПРЖР, 2000. – 272 с. – ISBN 5-93108-006-6.

4. *Хайкин, Саймон.* Нейронные сети : Полный курс : пер. с англ. / Саймон Хайкин. – 2-е изд., знач. доп. – М. : Вильямс, 2006. – 1104 с. – ISBN 5-8459-08906.

5. Прикладные нечеткие системы / К. Асаи [и др.] ; под ред. Т. Тэрано, К. Асаи, М. Сугэно ; пер. с япон. Ю. Н. Чернышова. – М. : Мир, 1993. – 368 с.

6. *Дьяконов, В. П.* MATLAB 6.5 SPl/7 + Simulink 5/6. Основы применения / В. П. Дьяконов. – М. : СОЛОН-Пресс, 2005. – 800 с. – (Серия «Библиотека профессионала»). – ISBN 5-98003-181-2.

7. *Половко, А. М.* MATLAB для студента / А. М. Половко, П. Н. Бутусов. – СПб. : БХВ-Петербург, 2005. – 320 с. – ISBN 5-94157-595-5.

8. *Штовба, С. Д.* Введение в теорию нечетких множеств и нечеткую логику / С. Д. Штовба. – [Электронный ресурс]. – URL: http://matlab.exponenta.ru/fuzzylogic/book2/index.php (дата обращения: 02.10.2014).

9. Using MATLAB. Version 5.2. The Mathworks, Inc., 1997. – 531 p.

10. MATLAB 5.2 Product Family New Features. Version 5.2. The Mathworks, Inc., 1998. – 202 p.

11. Using MATLAB Graphics. Version 5.2. The Mathworks, Inc., 1997. – 372 p.

12. MATLAB Functions Reference (Volumes 1 and 2). Version 5. The Mathworks, Inc., 1998. – 586 p.

13. *Дьяконов, В. П.* Справочник по применению системы PC MatLab / В. П. Дьяконов. – М. : Физматлит, 1993. – 112 с.

14. *Потёмкин, В. Г.* Система MATLAB : справ. пособие / В. Г. Потёмкин. – М. : Диалог-МИФИ, 1997. – 350 с. – ISBN 5-86404-100-9.

15. *Гультяев, А. К.* MATLAB 5.2. Имитационное моделирование в среде Windows : практ. пособие / А. К. Гультяев. – СПб. : Корона-Принт, 1999. – 288 с. – ISBN 5-7931-0053-9.

16. *Дьяконов, В. П.* MATLAB 5. Система символьной математики / В. П. Дьяконов, И. В. Абраменкова. – М. : Нолидж, 1999. – 633 с. – ISBN 5-89251-069-7.

17. *Лазарев, Ю. Ф.* MATLAB 5.x. / Ю. Ф. Лазарев. – Киев : BHV, 2000. – 384 с. – (Серия «Библиотека студента»). – ISBN 966-552-068-7.

18. *Медведев, В. С.* Control system toolbox. MATLAB 5 для студентов / В. С. Медведев, В. Г. Потёмкин. – М. : Диалог-МИФИ, 1999. – 287 с. – ISBN 5-86404-136-X.

19. *Потёмкин, В. Г.* Введение в MATLAB / В. Г. Потёмкин. – М. : Диалог-МИФИ, 2000. – 347 с. – ISBN 5-86404-140-8.

20. *Потёмкин, В. Г.* Система инженерных расчетов. MATLAB 5.x : в 2 т. / В. Г. Потёмкин. – М. : Диалог-МИФИ, 1999. – 670 с.

21. *Рудаков, П. И.* Обработка сигналов и изображений. MATLAB 5.x. / П. И. Рудаков, В. И. Сафонов. – М. : Диалог-МИФИ, 2000. – 413 с. – (Серия «Пакеты прикладных программ»).

ПРИЛОЖЕНИЯ

Приложение 1

РАБОТА С ПАКЕТОМ «SIMULINK»

1. Состав библиотеки «Simulink»

1.1. «Sources» – источники сигналов

1.1.1. Источник постоянного сигнала «Constant»

Назначение: задает постоянный по уровню сигнал.

Блок имеет следующие параметры:

1. «Constant value» – постоянная величина.

2. «Interpret vector parameters as 1-D» – интерпретировать вектор параметров как одномерный (при установленном флажке). Данный параметр встречается у большинства блоков библиотеки «Simulink». В дальнейшем он рассматриваться не будет.

Значение константы может быть действительным или комплексным числом, вычисляемым выражением, вектором или матрицей.

Рис. П1.1 иллюстрирует применение этого источника и измерение его выходного сигнала с помощью цифрового индикатора Display.

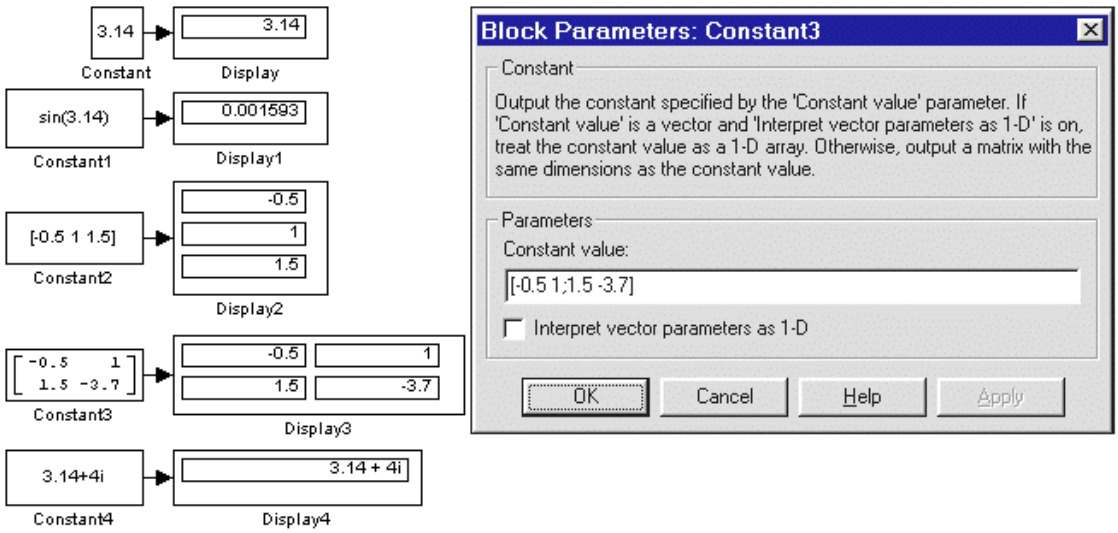


Рис. П1.1. Источник постоянного воздействия «Constant»

1.1.2. Источник синусоидального сигнала «Sine Wave»

Назначение: формирует синусоидальный сигнал с заданной частотой, амплитудой, фазой и смещением.

Для формирования выходного сигнала блоком могут использоваться два алгоритма. Вид алгоритма определяется параметром «Sine Type» (способ формирования сигнала):

- «Time-based» – по текущему времени;
- «Sample-based» – по величине шага модельного времени.

Формирование выходного сигнала по текущему значению времени для непрерывных систем. Выходной сигнал источника в этом режиме соответствует выражению

$y = Amplitude \cdot sin(frequency \cdot time + phase) + bias$.

Параметры блока:

- «Amplitude» – амплитуда;
- «Bias» – постоянная составляющая сигнала;
- «Frequency» (rads/sec) – частота(рад/с);
- «Phase (rads)» – начальная фаза (рад);
- «Sample time» – шаг модельного времени. Используется для согласования работы источника и других компонентов модели во времени. Параметр может принимать следующие значения:

а) 0 (по умолчанию) – используется при моделировании непрерывных систем;

б) >0 (положительное значение) – задается при моделировании дискретных систем. В этом случае шаг модельного времени можно интерпретировать как шаг квантования по времени выходного сигнала;

в) –1 – шаг модельного времени устанавливается таким же, как и в предшествующем блоке, т.е. блоке, откуда приходит сигнал в данный блок.

Этот параметр может задаваться для большинства блоков библиотеки «Simulink». В дальнейшем он рассматриваться не будет.

При расчетах для очень больших значений времени точность расчета выходных значений сигнала падает вследствие значительной ошибки округления.

Формирование выходного сигнала по текущему значению времени для дискретных систем. Алгоритм определения значения выходного сигнала источника для каждого последующего шага расчета определяется выражением (в матричной форме)

$$\begin{bmatrix} \sin(t+\Delta t) \\ \cos(t+\Delta t) \end{bmatrix} = \begin{bmatrix} \cos(\Delta t) & \sin(\Delta t) \\ \sin(\Delta t) & \cos(\Delta t) \end{bmatrix} \begin{bmatrix} \sin(t) \\ \cos(t) \end{bmatrix},$$

где Δt – постоянная величина, равная значению «Sample time».

В данном режиме ошибка округления для больших значений времени также уменьшает точность расчета.

Формирование выходного сигнала по величине модельного времени и количеству расчетных шагов на один период. Выходной сигнал источника в этом режиме соответствует выражению

$$y = Amplitude \cdot \sin[(k + Numberofoffsetsamples) / Samplesperperiod] + bias,$$

где k – номер текущего шага расчета.

Параметры блока:

• «Amplitude» – амплитуда;

• «Bias» – постоянная составляющая сигнала;

• «Samples per period» – количество расчетных шагов на один период синусоидального сигнала:

«Samples per period» = $2p$ / ($frequency \cdot Sampletime$);

• «Number of offset samples» – начальная фаза сигнала. Задается количеством шагов модельного времени:

«Number of offset samples» = $Phase \cdot Samplesperperiod / (2p)$;

• «Sample time» – шаг модельного времени.

В данном режиме ошибка округления не накапливается, поскольку «Simulink» начинает отсчет номера текущего шага с нуля для каждого периода.

На рис. П1.2 показано применение блока с разными значениями шага модельного времени («Sample time» = 0 для блока «Sine Wave 1» и «Sample time» = 0.1 для блока «Sine Wave 2»). Для отображения графиков выходных сигналов в модели использован виртуальный осциллограф («Scope»).

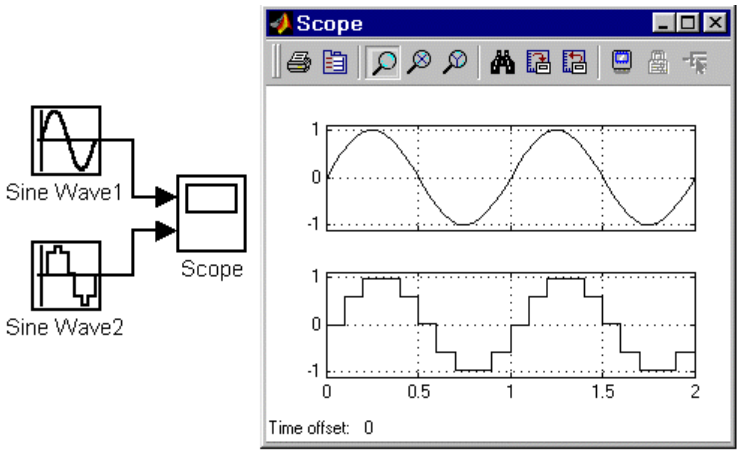


Рис. П1.2. Блок «Sine Wave»

1.1.3. Источник линейно изменяющегося воздействия «Ramp»

Назначение: формирует линейный сигнал вида *y = Slope·time + Initialvalue*.

Параметры блока:

• «Slope» – скорость изменения выходного сигнала;

• «Start time» – время начала формирования сигнала;

• «Initial value» – начальный уровень сигнала на выходе блока.

На рис. П1.3 показано использование данного блока.

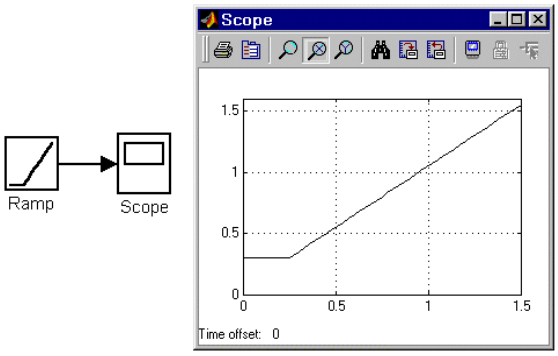


Рис. П1.3. Блок «Ramp»

1.1.4. Генератор ступенчатого сигнала «Step»

Назначение: формирует ступенчатый сигнал.

Параметры блока:

• «Step time» – время наступления перепада сигнала (с);

• «Initial value» – начальное значение сигнала;

• «Final value» – конечное значение сигнала.

Перепад может быть как в большую сторону (конечное значение больше, чем начальное), так и в меньшую (конечное значение меньше, чем начальное). Значения начального и конечного уровней могут быть не только положительными, но и отрицательными (например, изменение сигнала с уровня –5 до уровня –3).

На рис. П1.4 показано использование генератора ступенчатого сигнала.

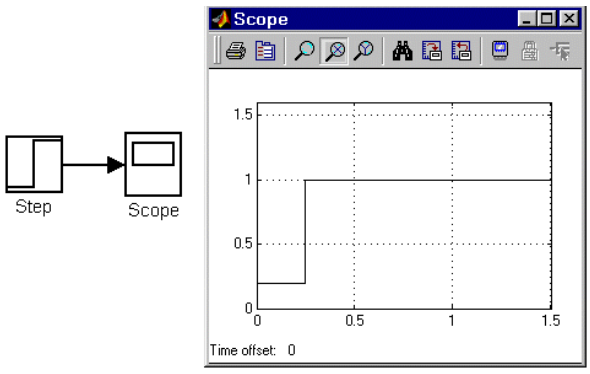


Рис. П1.4. Блок «Step»

1.1.5. Генератор сигналов «Signal Generator»

Назначение: формирует один из четырех видов периодических сигналов: «sine» – синусоидальный сигнал, «square» – прямоугольный сигнал, «sawtooth» – пилообразный сигнал, «random» – случайный сигнал.

Параметры блока:

• «Wave form» – вид сигнала;

• «Amplitude» – амплитуда сигнала;

• «Frequency» – частота (рад/с);

• «Units» – единицы измерения частоты. Может принимать два значения:

- Hertz – Гц;
- rad/sec – рад/с.

На рис. П1.5 показано применение этого источника при моделировании прямоугольного сигнала.

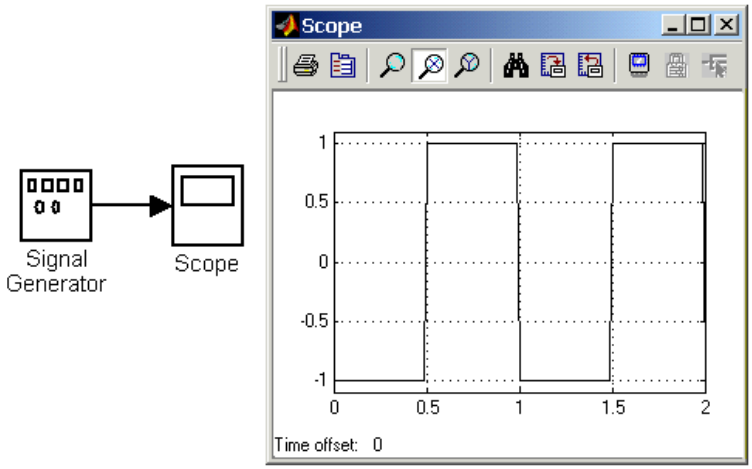


Рис. П1.5. Блок генератора сигналов

1.1.6. Источник случайного сигнала с равномерным распределением «Uniform Random Number»

Назначение: формирование случайного сигнала с равномерным распределением.

Параметры блока:

• «Minimum» – минимальный уровень сигнала;

• «Maximum» – максимальный уровень сигнала;

• «Initial seed» – начальное значение.

Пример использования блока и график его выходного сигнала представлены на рис. П1.6.

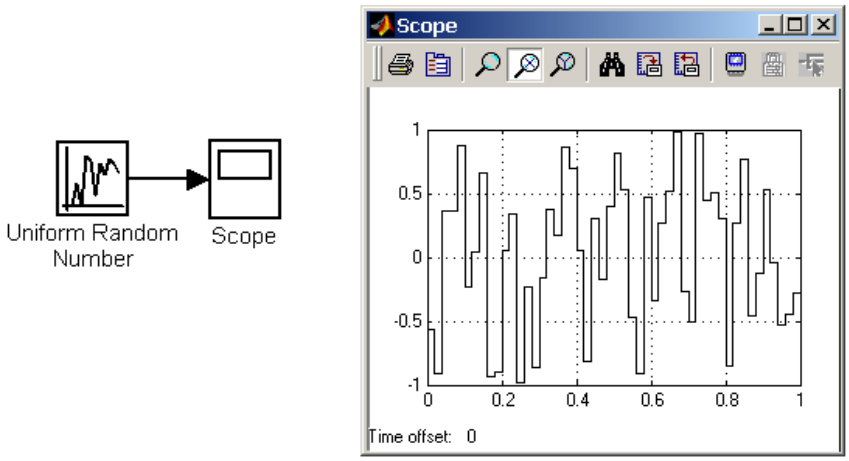


Рис. П1.6. Источник случайного сигнала с равномерным распределением

1.1.7. Источник случайного сигнала с нормальным распределением Random Number

Назначение: формирование случайного сигнала с нормальным распределением уровня сигнала (рис. П1.7).

Параметры блока:

- «Mean» – среднее значение сигнала;
- «Variance» – дисперсия (среднеквадратическое отклонение);
- «Initial seed» – начальное значение.

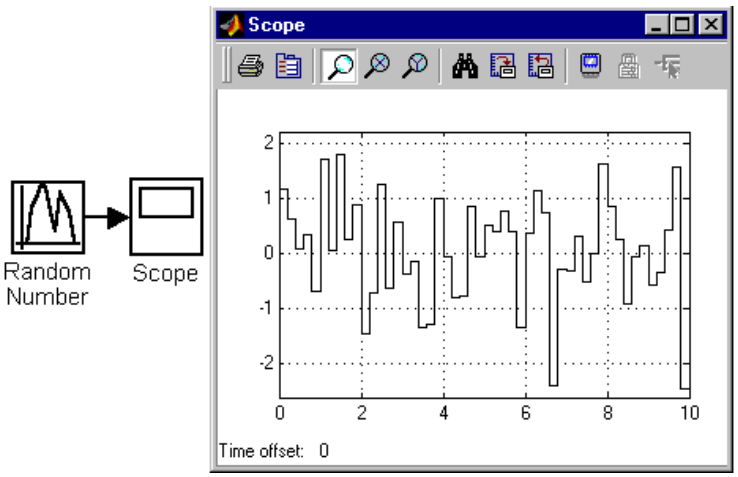


Рис. П1.7. Источник случайного сигнала с нормальным распределением

1.1.8. Источник импульсного сигнала «Pulse Generator»

Назначение: формирование прямоугольных импульсов.

Параметры:

- «Pulse Type» – способ формирования сигнала. Может принимать два значения:

- «Time-based» – по текущему времени;
- «Sample-based» – по величине модельного времени и количеству расчетных шагов;

• «Amplitude» – амплитуда;

• «Period» – период. Задается в секундах для «Time-based Pulse Type» или в шагах модельного времени для «Sample-based Pulse Type»;

• «Pulse width» – ширина импульсов. Задается в процентах по отношению к периоду для «Time-based Pulse Type» или в шагах модельного времени для «Sample-based Pulse Type»;

• «Phase delay» – фазовая задержка. Задается в секундах для Time-based Pulse Type или в шагах модельного времени для Sample-based Pulse Type;

• «Sample time» – шаг модельного времени. Задается для «Sample-based Pulse Type».

Пример использования «Pulse Generator» показан на рис. П1.8.

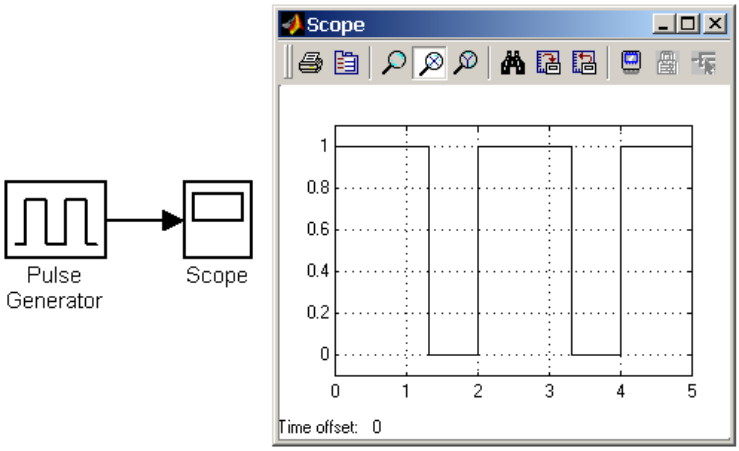


Рис. П1.8. Источник прямоугольных импульсов

1.1.9. Генератор линейно-изменяющейся частоты «Chirp Generator»

Назначение: формирование синусоидальных колебаний, частота которых линейно изменяется.

Параметры блока:

• «Initial frequence» – начальная частота (Гц);

• «Target time» – время изменения частоты (с);

• «Frequence at target time» – конечное значение частоты (Гц).

Пример использования блока показан на рис. П1.9.

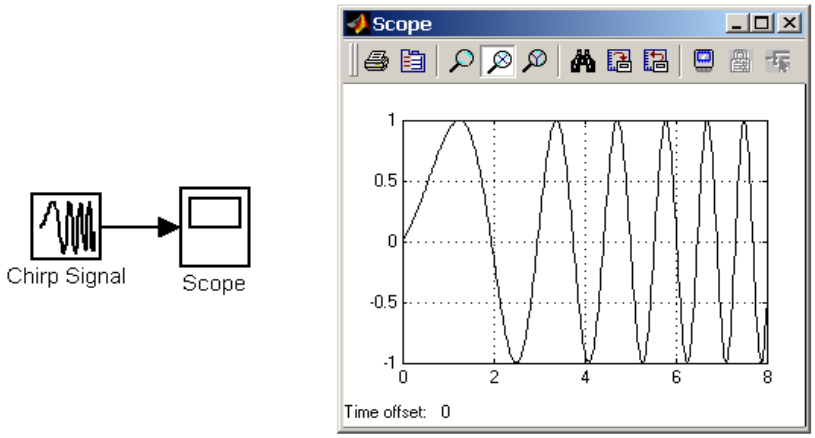


Рис. П1.9. Генератор линейно-изменяющейся частоты

1.1.10. Генератор белого шума «Band-Limited White Noice»

Назначение: создает сигнал заданной мощности, равномерно распределенной по частоте.

Параметры блока:

- «Noice Power» – мощность шума;
- «Sample Time» – модельное время;
- «Seed» – число, необходимое для инициализации генератора случайных чисел.

Рис. П1.10 показывает работу этого генератора.

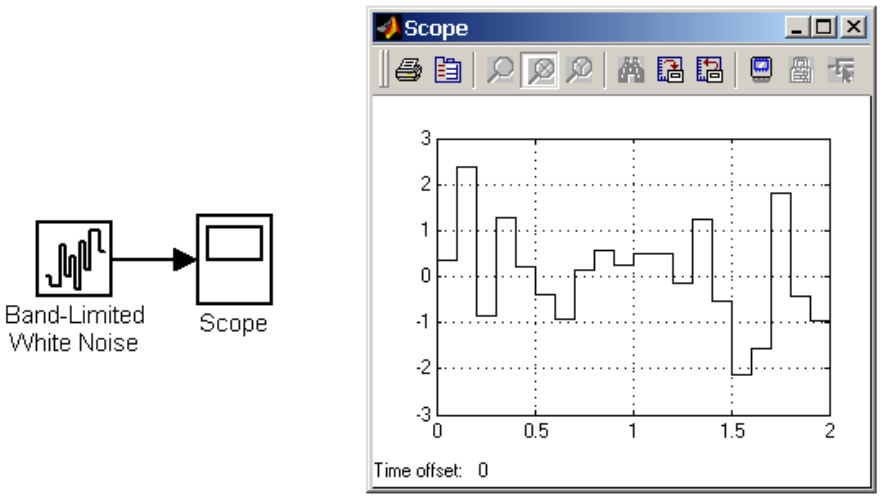


Рис. П1.10. Генератор белого шума

1.1.11. Источник временного сигнала «Clock»

Назначение: формирует сигнал, величина которого на каждом шаге расчета равна текущему времени моделирования.

Параметры блока:

• «Decimation» – шаг, с которым обновляются показания времени на изображении источника (в том случае, если установлен флажок параметра «Display time»). Параметр задается как количество шагов расчета. Например, если шаг расчета модели в окне диалога Simulation parameters установлен равным 0.01 с, а параметр «Decimation» блока «Clock» задан равным 1000, то обновление показаний времени будет производиться каждые 10 с модельного времени;

• «Display time» – отображение значения времени в блоке источника.

На рис. П1.11 показан пример работы данного источника

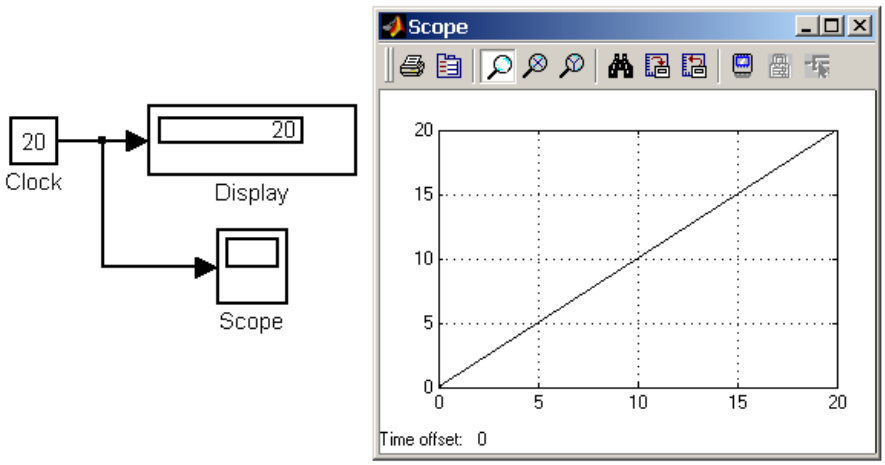


Рис. П1.11. Источник временного сигнала

1.1.12. Цифровой источник времени «Digital Clock»

Назначение: формирует дискретный временной сигнал.

Параметр блока: «Sample time» – шаг модельного времени (с).

На рис. П1.12 показана работа источника «Digital Clock».

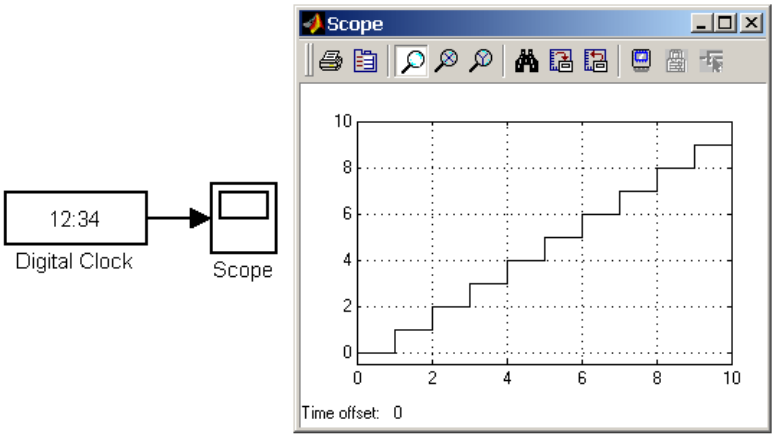


Рис. П1.12. Цифровой источник временного сигнала

1.1.13. Блок считывания данных из файла From File

Назначение: получение данных из внешнего файла.

Параметры блока:

• «File Name» – имя файла с данными;

• «Sample time» – шаг изменения выходного сигнала блока.

Данные в файле должны быть представлены в виде матрицы. Матрица должна состоять, как минимум, из двух строк. Значения времени записаны в первой строке матрицы, а в остальных строках находятся значения сигналов, соответствующие данным моментам времени. Значения времени должны быть записаны в возрастающем порядке. Выходной сигнал блока содержит только значения сигналов, а значения времени в нем отсутствуют. Если шаг расчета текущей модели не совпадает с отсчетами времени в файле данных, то «Simulink» выполняет линейную интерполяцию данных.

Файл данных (*mat*-файл), из которого считываются значения, не является текстовым. Структура файла подробно описана в справочной системе MATLAB. Пользователям «Simulink» удобнее всего создавать mat-файл с помощью блока «To File» (библиотека «Sinks»). На рис. П1.13 показан пример использования данного блока. Из файла data.mat считываются значения синусоидального сигнала.

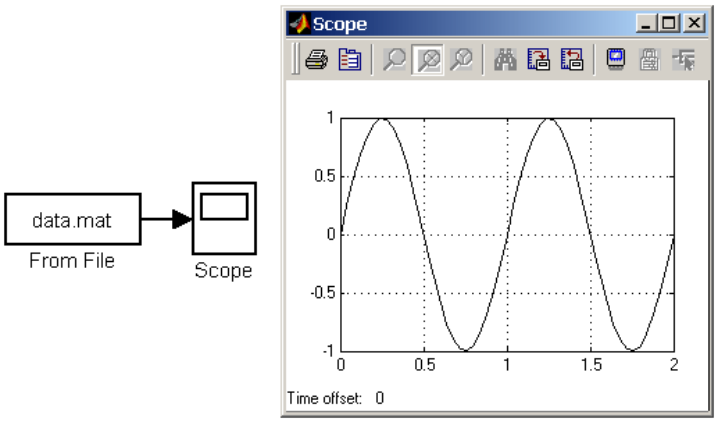


Рис. П1.13. Блок «From File»

1.1.14. Блок считывания данных из рабочего пространства «From Workspace»

Назначение: получение данных из рабочего пространства MATLAB.

Параметры блока:

• «Data» – имя переменной (матрицы или структуры), содержащей данные;

• «Sample time» – шаг изменения выходного сигнала блока;

• «Interpolate data» – интерполяция данных для значений модельного времени, не совпадающих со значениями в переменной «Data»;

• «Form output after final data value by» – вид выходного сигнала по окончании значений времени в переменной «Data»:

- «Extrapolate» – линейная экстраполяция сигналов;
- «Setting To Zero» – нулевые значения сигналов;
- «Holding Final Value» – выходные значения сигналов равны последним значениям;
- «Cyclic Repetition» – циклическое повторение значений сигналов. Данный вариант может использоваться, только если переменная «Data» имеет формат «Structure without time».

На рис. П1.14 показан пример использования данного блока. Данные в переменную simin рабочей области MATLAB загружаются из файла с помощью блока «Read data».

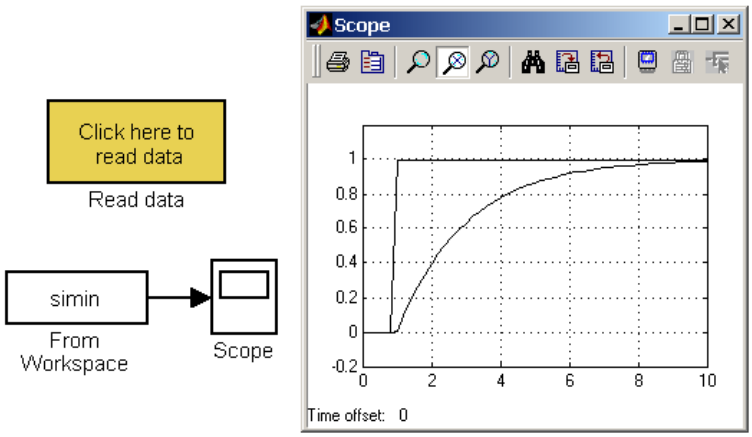


Рис. П1.14. Блок «From Workspace»

1.1.15. Блок сигнала нулевого уровня «Ground»

Назначение: формирование сигнала нулевого уровня.

Параметров данный блок не имеет.

Если какой-либо вход блока в модели не подсоединен, то при выполнении моделирования в главном окне MATLAB появляется предупреждающее сообщение. Для устранения этого на неподключенный вход блока можно подать сигнал с блока «Ground».

На рис. П1.15 даны примеры использования блока. В первом случае сигнал с блока «Ground» поступает на один из входов сумматора, а во втором – на один из входов блока умножения. Показания блоков «Display» подтверждают, что вырабатываемый блоком «Ground» сигнал имеет нулевое значение. Из рисунка также видно, что тип выходного сигнала блока устанавливается автоматически, в соответствии с типами сигналов, подаваемых на другие входы блоков (в данном случае – на входы блоков «Sum» и «Product»).

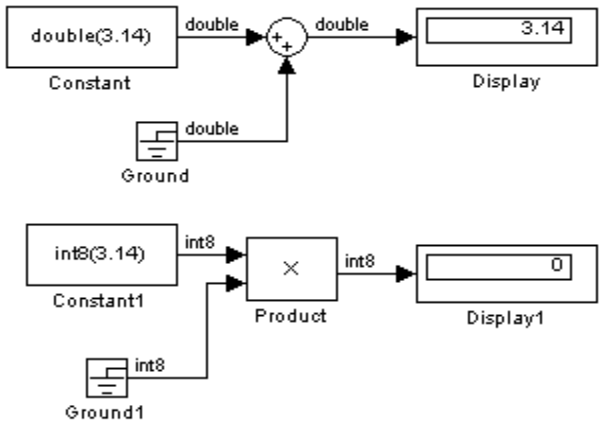


Рис. П1.15. Применение блока «Ground»

1.1.16. Блок периодического сигнала «Repeating Sequence»

Назначение: формирование периодического сигнала.

Параметры блока:

• «Time values» – вектор значений модельного времени;

• «Output values» – вектор значений сигнала для моментов времени, заданных вектором «Time values».

Блок выполняет линейную интерполяцию выходного сигнала для моментов времени, не совпадающих со значениями, заданными вектором «Time values». На рис. П1.16 показан пример использования блока для формирования пилообразного сигнала. Значения модельного времени заданы вектором [0 3], а значения выходного сигнала вектором [0 2].

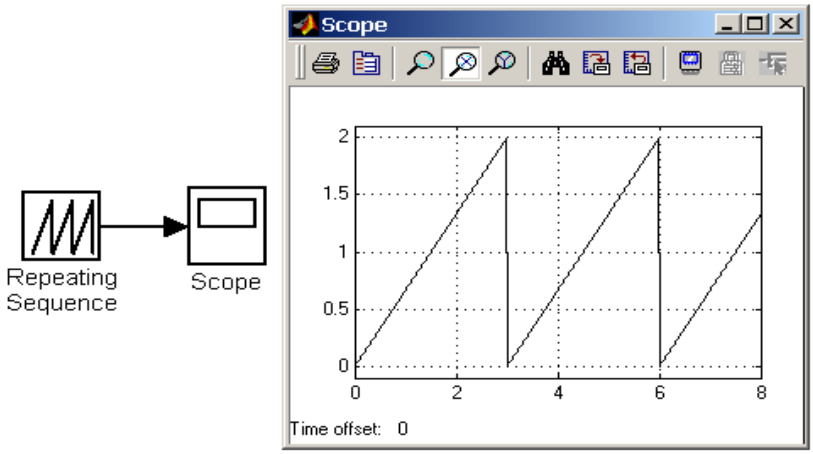


Рис. П1.16. Использование блока «Repeating Sequence»

1.1.17. Блок входного порта «Inport»

Назначение: создает входной порт для подсистемы или модели верхнего уровня иерархии.

Параметры блока:

• «Port number» – номер порта;

• «Port dimensions» – размерность входного сигнала. Если этот параметр равен –1, то размерность входного сигнала будет определяться автоматически;

• «Sample time» –шаг модельного времени;

• «Data type» – тип данных входного сигнала: *auto*, *double*, *single*, *int8*, *uint8*, *int16*, *uint16*, *int32*, *uint32* или *Boolean*;

• «Signal type» – тип входного сигнала:

auto – автоматическое определение типа;

real – действительный сигнал.

complex – комплексный сигнал;

• «Interpolate data» (флажок) – интерполировать входной сигнал. В случае, если временные отсчеты входного сигнала, считываемого из рабочей области MATLAB, не совпадают с модельным временем, то блок будет выполнять интерполяцию входного сигнала. При использовании блока «Inport» в подсистеме данный параметр не доступен.

Использование блока «Inport» в подсистемах. Блоки «Inport» подсистемы являются ее входами. Сигнал, подаваемый на входной порт подсистемы через блок «Inport», передается внутрь подсистемы. Название входного порта будет показано на изображении подсистемы как метка порта.

При создании подсистем и добавлении блока «Inport» в подсистему «Simulink» использует следующие правила:

При создании подсистемы с помощью команды «Edit/Create subsystem» входные порты создаются и нумеруются автоматически начиная с 1.

Если в подсистему добавляется новый блок «Inport», то ему присваивается следующий по порядку номер.

Если какой-либо блок «Inport» удаляется, то остальные порты переименовываются таким образом, чтобы последовательность номеров портов была непрерывной.

Если в последовательности номеров портов имеется разрыв, то при выполнении моделирования «Simulink» выдаст сообщение об

ошибке и остановит расчет. В этом случае необходимо вручную переименовать порты таким образом, чтобы последовательность номеров портов не нарушалась.

На рис. П1.17 показана модель, использующая подсистему, и схема этой подсистемы.

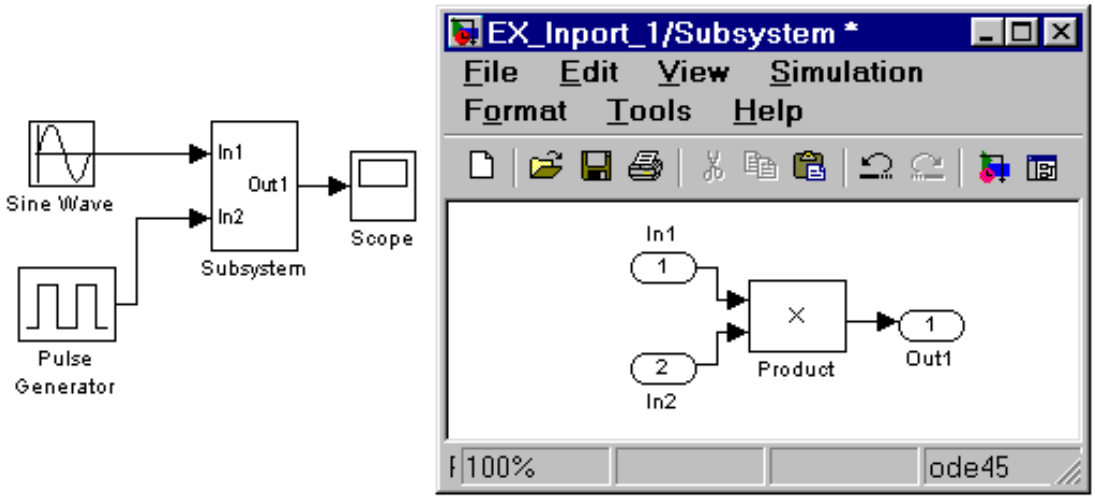


Рис. П1.17. Использование блока «Inport» в подсистеме

Использование блока «Inport» в модели верхнего уровня. Входной порт в системе верхнего уровня используется для передачи сигнала из рабочей области MATLAB в модель.

Для передачи сигнала из рабочего пространства MATLAB требуется не только установить в модели входной порт, но и выполнить установку параметров ввода на вкладке «Workspace I/O» окна диалога «Simulation parameters…» (должен быть установлен флажок для параметра «Inport» и задано имя переменной, которая содержит входные данные). Тип вводимых данных: «Array» (массив), «Structure» (структура) или «Structure with time» (структура с полем «время») задается на этой же вкладке.

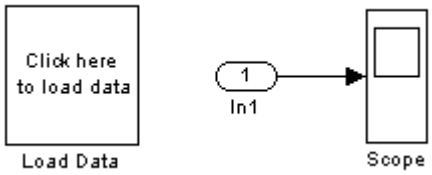


Рис. П1.18. Модель, считывающая входной сигнал из рабочего пространства MATLAB с помощью блока «Input»

На рис. П1.18 показана модель, считывающая входной сигнал из рабочего пространства MATLAB. Подсистема «Load Data» обеспечивает ввод данных из файла в рабочую область MATLAB.

1.2. «Sinks» – приемники сигналов

1.2.1. Осциллограф «Scope»

Назначение: строит графики исследуемых сигналов в функции времени. Позволяет наблюдать за изменениями сигналов в процессе моделирования.

Изображение блока и окно для просмотра графиков показаны на рис. П1.19.

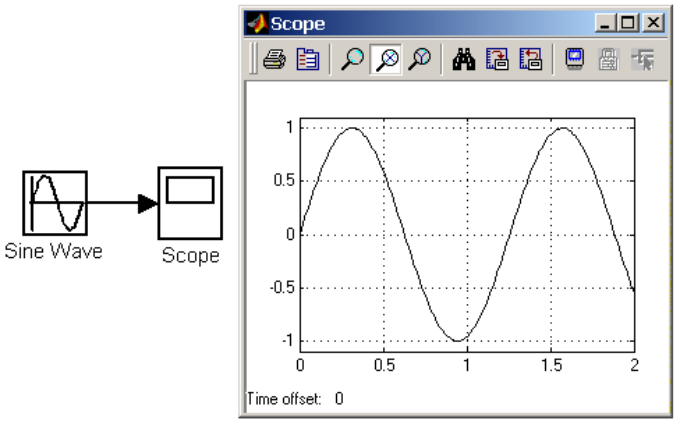


Рис. П1.19. Осциллограф «Scope»

Для того чтобы открыть окно просмотра сигналов, необходимо выполнить двойной щелчок левой клавишей «мыши» на изображении блока. Это можно сделать на любом этапе расчета (как до начала расчета, так и после него, а также во время расчета). В том случае, если на вход блока поступает векторный сигнал, то кривая для каждого элемента вектора строится отдельным цветом.

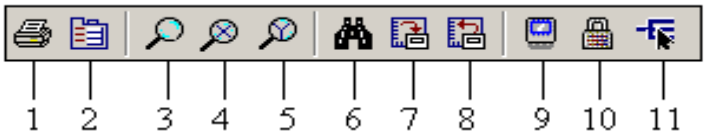


Рис. П1.20. Панель инструментов блока «Scope»

Настройка окна осциллографа выполняется с помощью панелей инструментов (рис. П1.20).

Панель инструментов содержит 11 кнопок:

1. «Print» – печать содержимого окна осциллографа.
2. «Parameters» – доступ к окну настройки параметров.
3. «Zoom» – увеличение масштаба по обеим осям.
4. «Zoom X-axis» – увеличение масштаба по горизонтальной оси.
5. «Zoom Y-axis» – увеличение масштаба по вертикальной оси.
6. «Autoscale» – автоматическая установка масштабов по обеим осям.

7. «Save current axes settings» – сохранение текущих настроек окна.

8. «Restore saved axes settings» – установка ранее сохраненных настроек окна.

9. «Floating scope» – перевод осциллографа в «свободный» режим.

10. «Lock/Unlock axes selection» – закрепить/разорвать связь между текущей координатной системой окна и отображаемым сигналом. Инструмент доступен, если включен режим «Floating scope».

11. «Signal selection» – выбор сигналов для отображения. Инструмент доступен, если включен режим «Floating scope».

Изменение масштабов отображаемых графиков можно выполнять несколькими способами:

1. Нажать соответствующую кнопку ( ,  или ) и щелкнуть один раз левой клавишей «мыши» в нужном месте графика. Произойдет 2,5-кратное увеличение масштаба.

2. Нажать соответствующую кнопку ( ,  или ) и, нажав левую клавишу «мыши», с помощью динамической рамки или отрезка указать область графика для увеличенного изображения. Рис. П1.21 поясняет этот процесс.

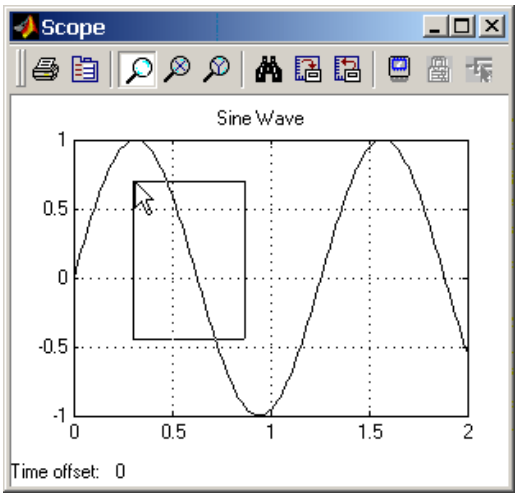


Рис. П1.21. Увеличение масштаба графика

3. Щелкнуть правой клавишей «мыши» в окне графиков и выбрать команду «Axes properties…» в контекстном меню. Откроется окно свойств графика, в котором с помощью параметров «Y-min» и «Y-max» можно указать предельные значения вертикальной оси. В этом же окне можно указать заголовок графика («Title»), заменив выражение «%<SignalLabel>» в строке ввода. Окно свойств показано на рис. П1.22.

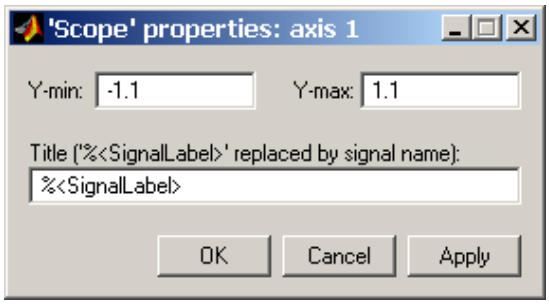


Рис. П1.22. Окно свойств графика

Параметры блока:

Параметры блока устанавливаются в окне «Scope parameters», которое открывается с помощью инструмента «Parameters» панели инструментов. Окно параметров имеет две вкладки:

- «General» – общие параметры;
- «Data history» – параметры сохранения сигналов в рабочей области MATLAB.

Вкладка общих параметров показана на рис. П1.23.

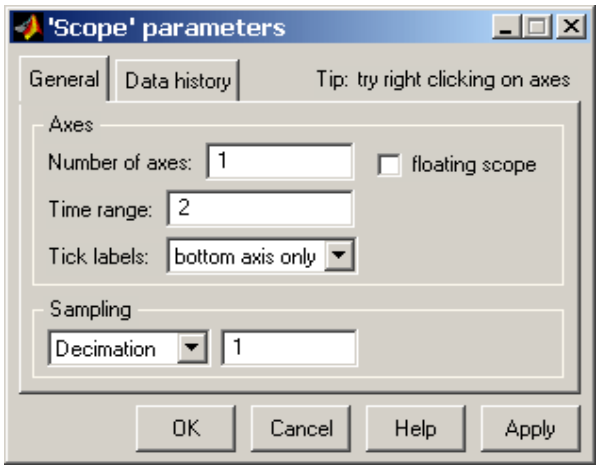


Рис. П1.23. Вкладка общих параметров «General»

На вкладке «General» задаются следующие параметры:

1. «Number of axes» – число входов (систем координат) осциллографа. При изменении этого параметра на изображении блока появляются дополнительные входные порты.

2. «Time range» – величина временного интервала, для которого отображаются графики. Если время расчета модели превышает заданное параметром Time range, то вывод графика производится порциями, при этом интервал отображения каждой порции графика равен заданному значению Time range.

3. «Tick labels» – вывод/скрытие осей и меток осей. Может принимать три значения (выбираются из списка):

- all – подписи для всех осей;
- none – отсутствие всех осей и подписей к ним;
- bottom axis only – подписи горизонтальной оси только для нижнего графика.

4. Sampling – установка параметров вывода графиков в окне. Задает режим вывода расчетных точек на экран. При выборе «Decimation» кратность вывода устанавливается числом, задающим шаг выводимых расчетных точек. На рис. П1.24 и П1.25 показаны графики синусоидальных сигналов, рассчитанных с фиксированным шагом 0.1 с. В окне блока «Scope» выводится каждая расчетная точка (параметр «Decimation» равен 1). На рис. П1.25 показан вывод каждо-

го второго значения (параметр «Decimation» равен 2). Маркерами на графиках отмечены расчетные точки.

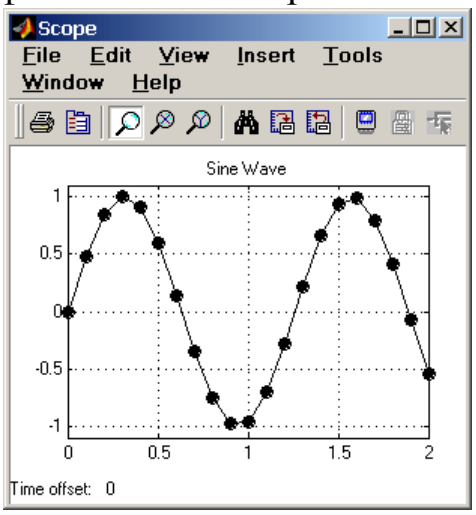


Рис. П1.24. Отображение синусоидального сигнала («Decimation» = 1)

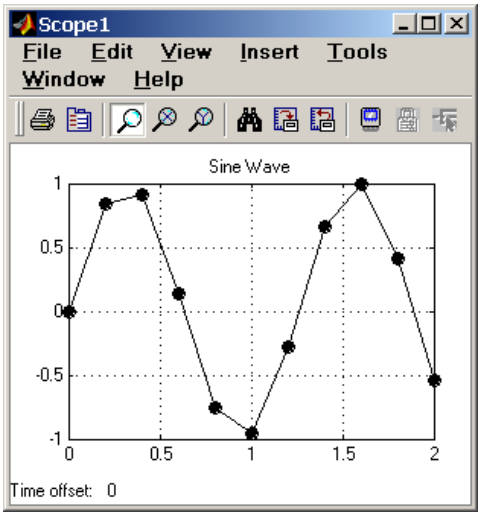


Рис. П1.25. Отображение синусоидального сигнала («Decimation» = 2)

В том случае, если режим вывода расчетных точек задается как «Sample time», то его числовое значение определяет интервал квантования при отображении сигнала. На рис. П1.26 показан график синусоидального сигнала для случая, когда значение параметра «Sample time» равно 0.1.

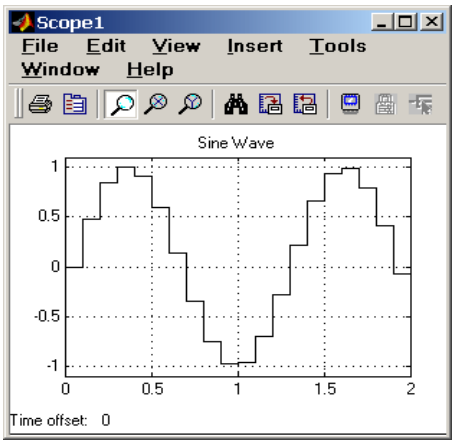


Рис. П1.26. Отображение синусоидального сигнала («Sample time» = 0.1)

5. Floating scope – перевод осциллографа в «свободный» режим (при установленном флажке).

На вкладке «Data history» (рис. П1.27) задаются следующие параметры:

1. «Limit data points to last» – максимальное количество отображаемых расчетных точек графика. При превышении этого числа начальная часть графика обрезается. В том случае, если флажок параметра «Limit data points to last» не установлен, то «Simulink» автоматически увеличит значение этого параметра для отображения всех расчетных точек.

2. «Save data to workspace» – сохранение значений сигналов в рабочей области MATLAB.

3. «Variable name» – имя переменной для сохранения сигналов в рабочей области MATLAB.

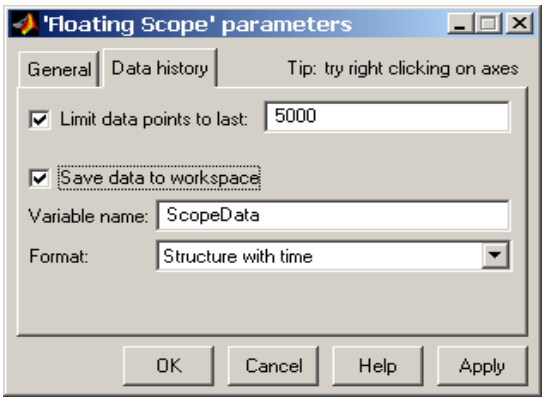


Рис. П1.27. Вкладка «Data history»

4. «Forma»t – формат данных при сохранении в рабочей области MATLAB. Может принимать значения:

- «Array» – массив;
- «Structure» – структура;
- «Structure with time» – структура с дополнительным полем «время».

1.2.2. Осциллограф «Floating Scope»

Осциллограф «Floating Scope» по сути есть обычный осциллограф «Scope», переведенный в «свободный» режим. Пример модели с осциллографом «Floating Scope» показан на рис. П1.28.

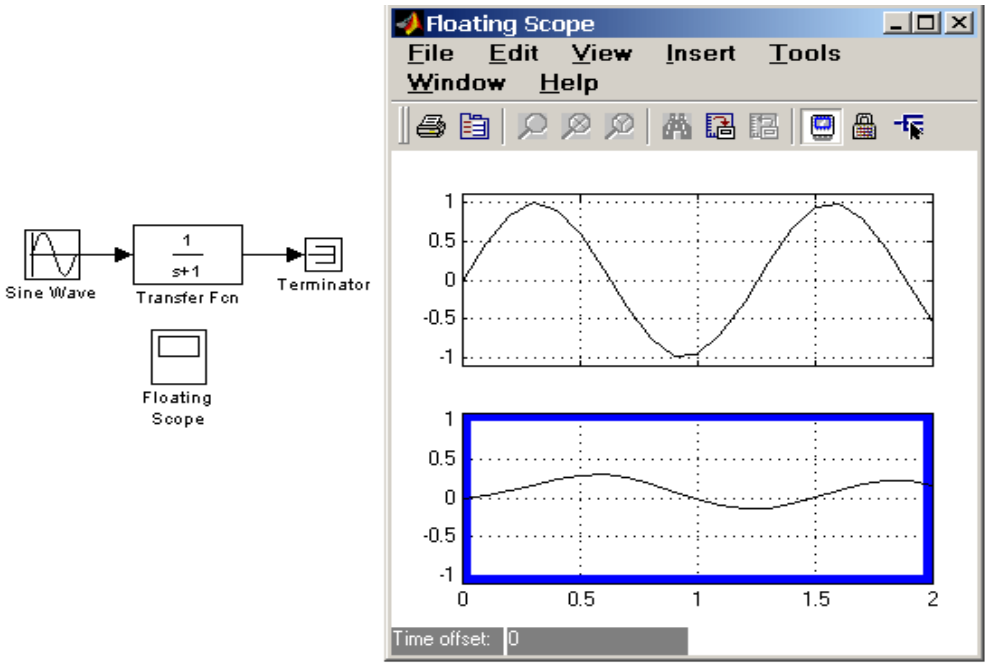


Рис. П1.28. Пример модели с осциллографом «Floating Scope»

В этом режиме блок осциллографа не имеет входов, а выбор отображаемого сигнала осуществляется с помощью инструмента «Signal selection» панели инструментов. Для выбора сигналов необходимо выполнить следующие действия:

1. Выделить систему координат, в которой будет отображаться график. Это достигается с помощью одиночного щелчка левой клавишей «мыши» внутри нужной системы. Выбранная система координат будет подсвечена по периметру синим цветом.

2. С помощью инструмента открыть окно диалога «Signal Selector» (рис. П1.29).

3. Отметить флажком имена блоков, сигналы с выхода которых требуется исследовать.

После выполнения расчета в окне блока «Floating Scope» будут отображены выбранные сигналы.

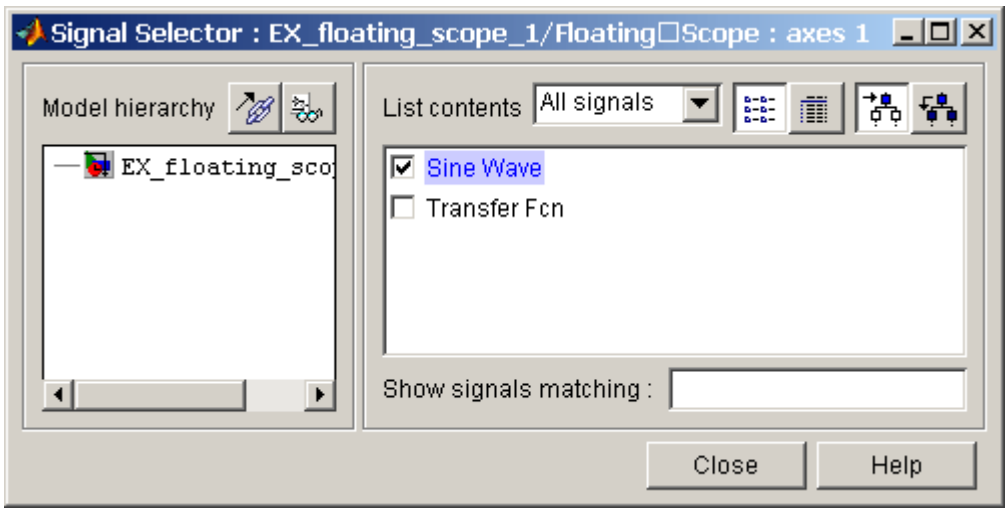


Рис. П1.29. Окно диалога «Signal Selector»

1.2.3. Графопостроитель «XY Graph»

Назначение: строит график одного сигнала в функции другого (график вида $Y(X)$).

Параметры блока:

- «x-min» – Минимальное значение сигнала по оси X;
- «x-max» – Максимальное значение сигнала по оси X;
- «y-min» – Минимальное значение сигнала по оси Y;
- «y-max» – Максимальное значение сигнала по оси Y;
- «Sample time» – шаг модельного времени.

Блок имеет два входа. Верхний вход предназначен для подачи сигнала, который является аргументом (X), нижний – для подачи значений функции (Y).

На рис. П1.30 в качестве примера использования графопостроителя показано построение фазовой траектории колебательного звена.

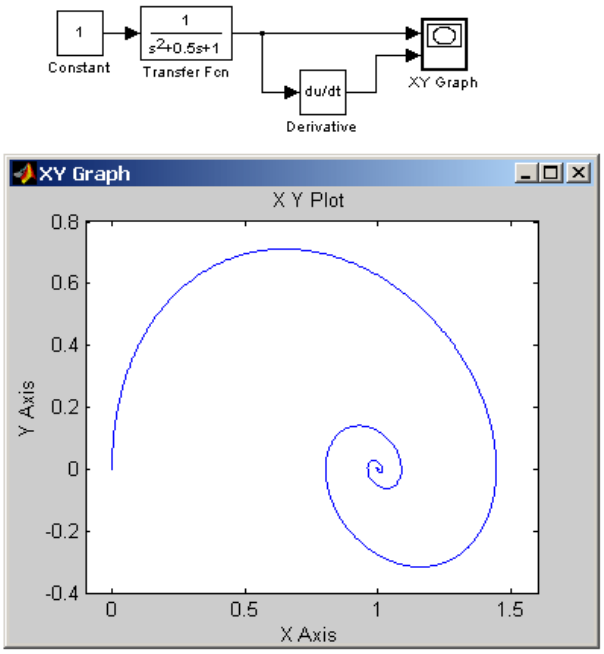


Рис. П1.30. Пример использования графопостроителя «XY Graph»

Графопостроитель можно использовать и для построения временных зависимостей. Для этого на первый вход следует подать временной сигнал с выхода блока «Clock». Пример такого использования графопостроителя показан на рис. П1.31.

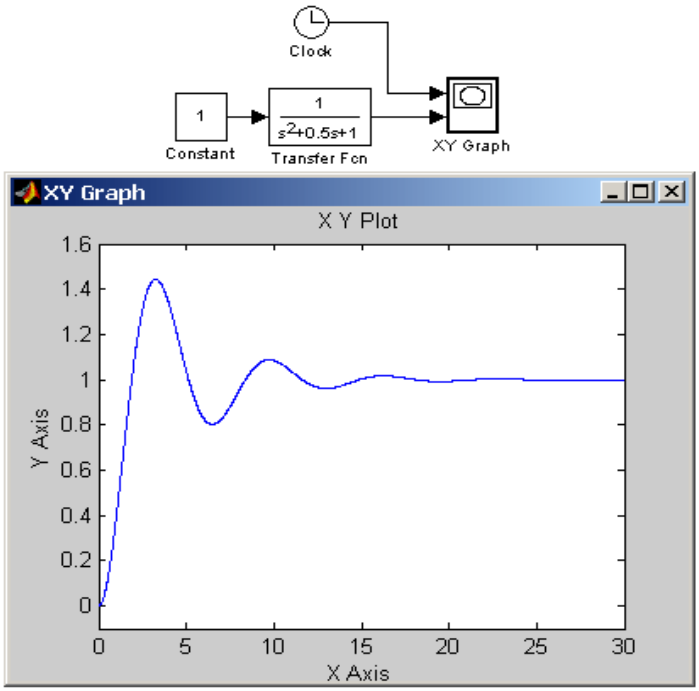


Рис. П1.31. Пример использования блока «XY Graph» для отображения временных зависимостей

1.2.4. Цифровой дисплей «Display»

Назначение: отображает значение сигнала в виде числа.

Параметры блока:

1. «Format» – формат отображения данных. Параметр «Format» может принимать следующие значения:

• «short» – 5 значащих десятичных цифр;

• «long» – 15 значащих десятичных цифр;

• «short_e» – 5 значащих десятичных цифр и 3 символа степени десяти;

• «long_e» – 15 значащих десятичных цифр и 3 символа степени десяти;

• «bank» – «денежный» формат. Формат с фиксированной точкой и двумя десятичными цифрами в дробной части числа;

2. «Decimation» – кратность отображения входного сигнала. При «Decimation» = 1 отображается каждое значение входного сигнала, при «Decimation» = 2 отображается каждое второе значение, при «Decimation» = 3 – каждое третье значение и т.д.

3. «Sample time» – шаг модельного времени. Определяет дискретность отображения данных.

4. «Floating display» (флажок) – перевод блока в «свободный» режим. В данном режиме входной порт блока отсутствует, а выбор сигнала для отображения выполняется щелчком левой клавиши мыши на соответствующей линии связи. В этом режиме для параметра расчета «Signal storage reuse» должно быть установлено значение «off» (вкладка «Advanced» в окне диалога «Simulation parameters…»).

На рис. П1.32 показано применение блока «Display» с использованием различных вариантов параметра «Format».

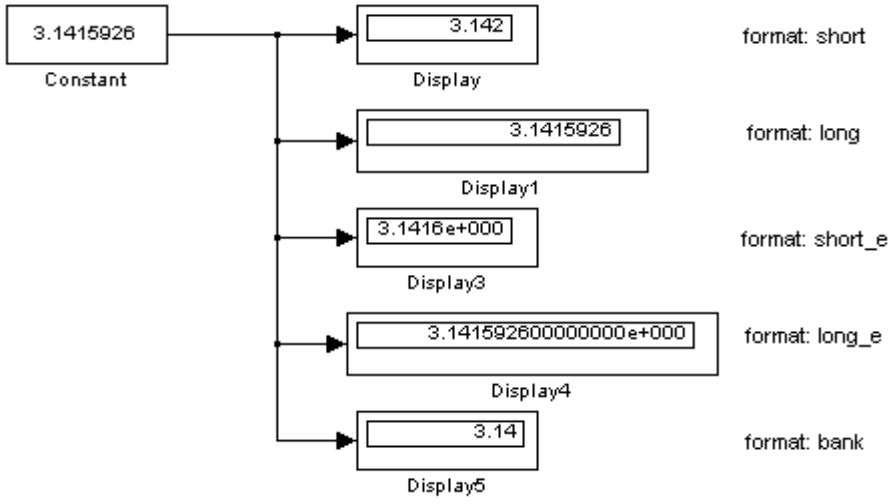


Рис. П1.32. Применение блока «Display» с использованием различных вариантов параметра «Format»

Блок «Display» может использоваться для отображения не только скалярных сигналов, но также векторных, матричных и комплексных. Рис. П1.33 иллюстрирует это. Если все отображаемые значения не могут поместиться в окне блока, в правом нижнем углу блока появляется символ ⇲, указывающий на необходимость увеличить размеры блока (см. блок «Display4» на рис. П1.33.

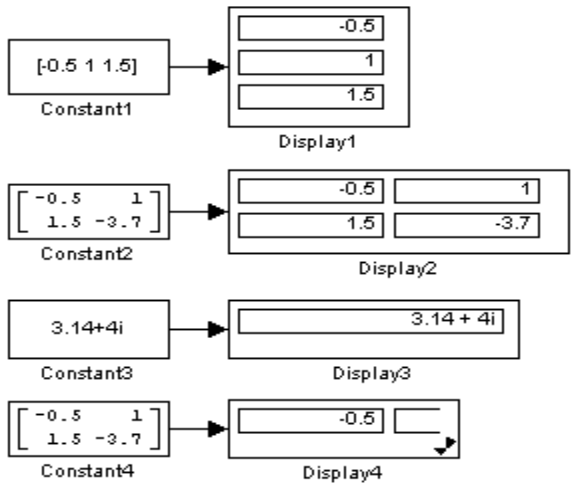


Рис. П1.33. Применение блока «Display» для отображения векторных, матричных и комплексных сигналов

1.2.5. Блок остановки моделирования «Stop Simulation»

Назначение: обеспечивает завершение расчета, если входной сигнал блока становится не равным нулю.

Параметров данный блок не имеет.

При подаче на вход блока ненулевого сигнала «Simulink» выполняет текущий шаг расчета, а затем останавливает моделирование. Если на вход блока подан векторный сигнал, то для остановки расчета достаточно, чтобы один элемент вектора стал ненулевым. На рис. П.1.34 показан пример использования данного блока. В примере остановка расчета происходит, если выходной сигнал блока «Transfer Function» становится большим или равным 0.99.

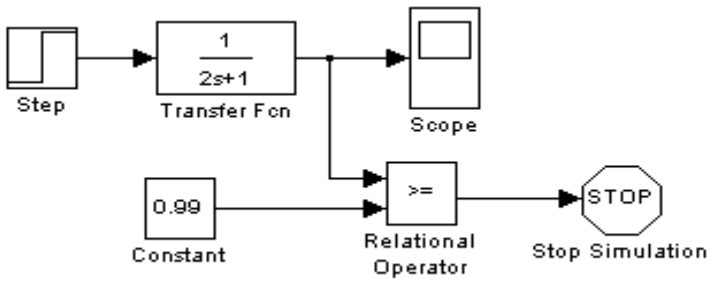


Рис. П1.34. Применение блока «Stop Simulation»

1.2.6. Блок сохранения данных в файле «To File»

Назначение: блок записывает данные, поступающие на его вход, в файл.

Параметры блока:

1. «Filename» – имя файла для записи. По умолчанию файл имеет имя *untitled.mat*. Если не указан полный путь файла, то файл сохраняется в текущей рабочей папке.

2. «Variable name» – имя переменной, содержащей записываемые данные.

3. «Decimation» – кратность записи в файл входного сигнала. При «Decimation» = 1 записывается каждое значение входного сигнала, при «Decimation» = 2 записывается каждое второе значение, при «Decimation» = 3 – каждое третье значение и т.д.

4. «Sample time» – шаг модельного времени. Определяет дискретность записи данных.

Данные в файле сохраняются в виде матрицы.

Значения времени записываются в первой строке матрицы, а в остальных строках будут находиться значения сигналов, соответствующих данным моментам времени.

Файл данных (*mat*-файл), в который записываются данные, не является текстовым. Структура файла подробно описана в справочной системе MATLAB. Пользователям «Simulink» удобнее всего считывать данные из *mat*-файла с помощью блока «From File» (библиотека «Sources»).

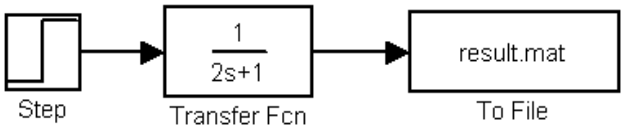


Рис. П1.35. Применение блока «To File»

На рис. П1.35 показан пример использования данного блока. Результаты расчета сохраняются в файле *result.mat*.

1.2.7. Блок сохранения данных в рабочей области «To Workspace»

Назначение:блок записывает данные, поступающие на его вход, в рабочую область MATLAB.

Параметры блока:

1. «Variable name» – имя переменной, содержащей записываемые данные.

2. «Limit data points to last» – максимальное количество сохраняемых расчетных точек по времени (отсчет ведется от момента завершения моделирования). В том случае, если значение параметра Limit data points to last задано как inf, то в рабочей области будут сохранены все данные.

3. «Decimation» – кратность записи данных в рабочую область.

4. «Sample time» – шаг модельного времени. Определяет дискретность записи данных.

5. «Save format» – формат сохранения данных. Может принимать значения:

6. «Matrix» – матрица. Данные сохраняются как массив, в котором число строк определяется числом расчетных точек по времени, а число столбцов – размерностью вектора, подаваемого на вход блока. Если на вход подается скалярный сигнал, то матрица будет содержать лишь один столбец.

7. «Structure» – структура. Данные сохраняются в виде структуры, имеющей три поля: time – время, signals – сохраняемые значения сигналов, block Name – имя модели и блока To Workspace. Поле time для данного формата остается не заполненным.

8. «Structure with Time» – структура с дополнительным полем (время). Для данного формата, в отличие от предыдущего, поле time заполняется значениями времени.

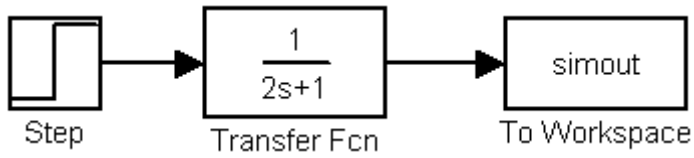


Рис. П1.36. Применение блока «To Workspace»

На рис. П1.36 показан пример использования данного блока. Результаты расчета сохраняются в переменной «simout».

Для считывания данных, сохраненных в рабочей области MATLAB, можно использовать блок «From Workspace» (библиотека «Sources»).

1.2.8. Концевой приемник «Terminator»

Назначение: блок используется для подачи сигнала с неиспользуемого выхода другого блока.

Параметров блок не имеет.

В том случае, если выход блока оказывается не подключенным ко входу другого блока, «Simulink» выдает предупреждающее сообщение в командном окне MATLAB. Для исключения этого необходимо использовать блок «Terminator». На рис. П1.37 показан пример использования концевого приемника. Извлекаемый с помощью блока Demux из матрицы второй элемент не никак не используется, поэтому он подается на вход блока Terminator.

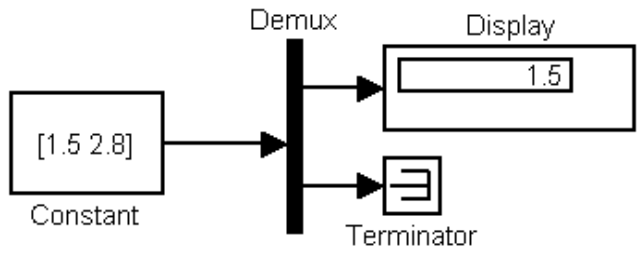


Рис. П1.37. Применение блока «Terminator»

1.2.9. Блок выходного порта Outport

Назначение: создает выходной порт для подсистемы или для модели верхнего уровня иерархии.

Параметры блока:

1. «Port number» – номер порта.

2. «Output when disabled» – вид сигнала на выходе подсистемы, в случае если подсистема выключена. Используется для управляемых подсистем. Может принимать значения (выбираются из списка):

- held – выходной сигнал подсистемы равен последнему рассчитанному значению.

- reset – выходной сигнал подсистемы равен значению, задаваемому параметром «Initial output».

3. Initial output – значение сигнала на выходе подсистемы до начала ее работы и в случае, если подсистема выключена. Используется для управляемых подсистем.

Использование блока «Outport» в подсистемах. Блоки «Outport» подсистемы являются ее выходами. Сигнал, подаваемый в блок «Outport» внутри подсистемы, передается в модель (или подсистему) верхнего уровня. Название выходного порта будет показано на изображении подсистемы как метка порта.

При создании подсистем и добавлении блока «Outport» в подсистему «Simulink» использует следующие правила:

1. При создании подсистемы с помощью команды «Edit/Create subsystem» выходные порты создаются и нумеруются автоматически, начиная с 1.

2. Если в подсистему добавляется новый блок «Outport», то ему присваивается следующий по порядку номер.

3. Если какой-либо блок «Outport» удаляется, то остальные порты переименовываются таким образом, чтобы последовательность номеров портов была непрерывной.

4. Если в последовательности номеров портов имеется разрыв, то при выполнении моделирования «Simulink» выдаст сообщение об ошибке и остановит расчет. В этом случае необходимо вручную переименовать порты таким образом, чтобы последовательность номеров портов не нарушалась.

На рис. П1.38 показана модель, использующая подсистему, и схема этой подсистемы.

В том случае, если подсистема является управляемой, то для ее выходных портов можно задать вид выходного сигнала для тех временных интервалов, когда подсистема заблокирована. На рис. П1.39 показана модель, использующая управляемую подсистему (схема подсистемы такая же, как и в предыдущем примере). Для первого выходного порта подсистемы параметр «Output when disabled» задан как «held», а для второго – как «reset», причем величина начального значения задана равной нулю. Графики сигналов показывают, что когда подсистема заблокирована, сигнал первого выходного порта остается неизменным, а сигнал второго становится равным заданному начальному значению (нулю).

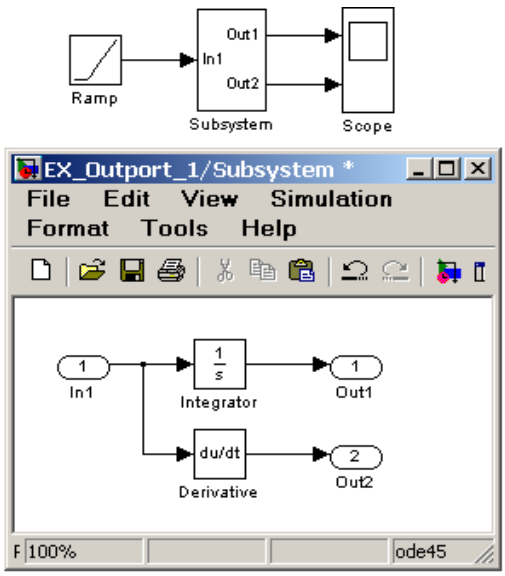


Рис. П1.38. Использование блока «Outport» в подсистеме

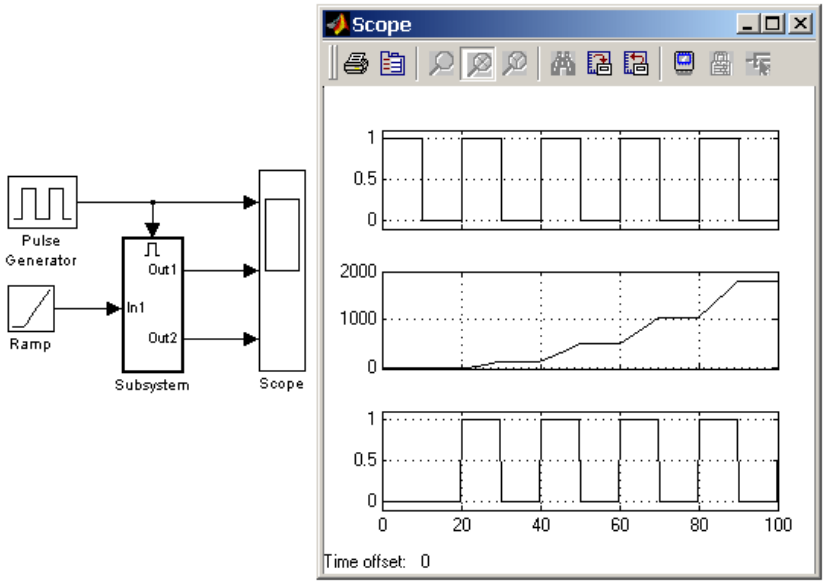


Рис. П1.39. Управляемая подсистема с различными настройками выходных портов

Использование блока «Outport» в модели верхнего уровня. Выходной порт в системе верхнего уровня используется в двух случаях:

1. Для передачи сигнала в рабочее пространство MATLAB.

2. Для обеспечения связи функций анализа с выходами модели.

Для передачи сигнала в рабочее пространство MATLAB требуется не только установить в модели выходные порты, но и выполнить установку параметров вывода на вкладке «Workspace I/O» окна диалога «Simulation parameters…» (должен быть установлен флажок для параметра «Output» и задано имя переменной для сохранения данных). Тип сохраняемых данных – «Array» (массив), «Structure» (структура) или «Structure with time» (структура с полем «время») задается на этой же вкладке.

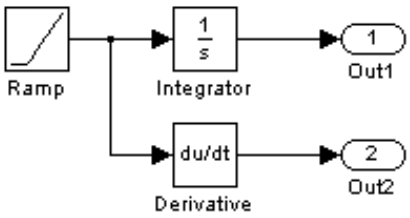


Рис. П1.40. Модель, передающая сигналы в рабочее пространство MATLAB с помощью блоков «Outport»

На рис. П1.40 показана модель, передающая сигналы в рабочее пространство MATLAB.

Блок «Outport» может использоваться также для связи модели с функциями анализа, например: «linmod или trim».

2. «Nonlinear» – нелинейные блоки

2.1. Блок ограничения Saturation

Назначение: выполняет ограничение величины сигнала.

Параметры блока:

1. «Upper limit» – верхний порог ограничения.

2. «Lower limit» – нижний порог ограничения.

3. «Treat as gain when linearizing» (флажок) – трактовать как усилитель с коэффициентом передачи, равным 1, при линеаризации.

Выходной сигнал блока равен входному сигналу, если его величина не выходит за порог ограничения. По достижении входным сигналом уровня ограничения выходной сигнал блока перестает изменяться и остается равным порогу. На рис. П1.41 показан пример использования блока для ограничения синусоидального сигнала, приводятся временные диаграммы сигналов и зависимость выходного сигнала блока от входного.

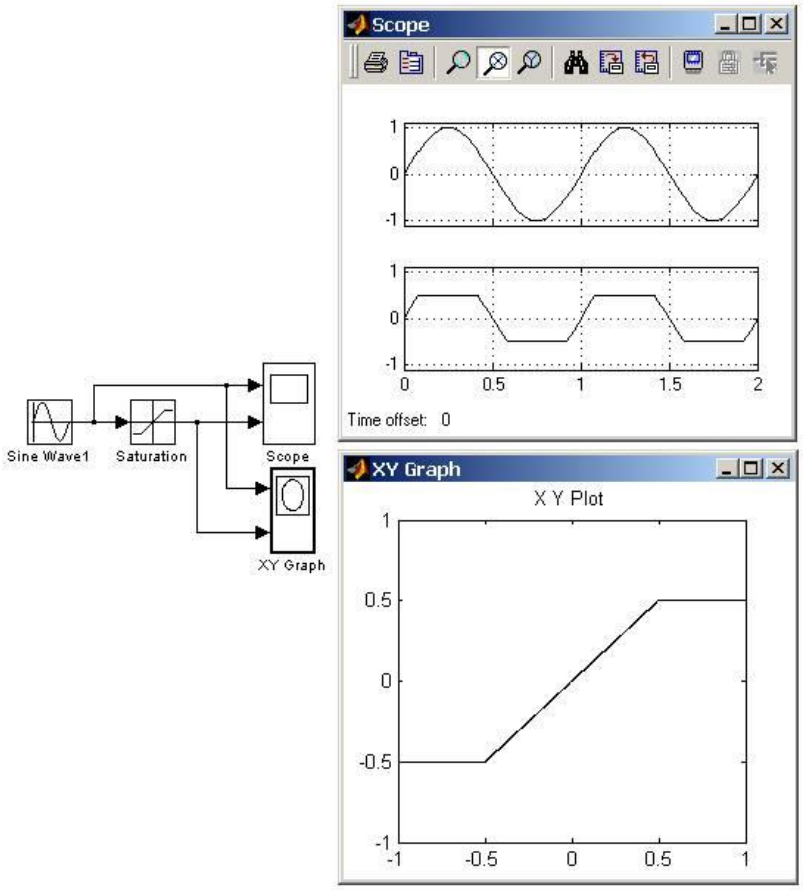


Рис. П1.41. Пример использования блока «Saturation»

2.2. Блок с зоной нечувствительности «Dead Zone»

Назначение: реализует нелинейную зависимость типа «зона нечувствительности (мертвая зона)».

Параметры блока:

1. «Start of dead zone» – начало зоны нечувствительности (нижний порог).

2. «End of dead zone» – конец зоны нечувствительности (верхний порог).

3. «Saturate on integer overflow» (флажок) – подавлять переполнение целого. При установленном флажке ограничение сигналов целого типа выполняется корректно.

4. «Treat as gain when linearizing» (флажок) – трактовать как усилитель с коэффициентом передачи, равным 1, при линеаризации.

Выходной сигнал блока вычисляется в соответствии со следующим алгоритмом:

1. Если величина входного сигнала находится в пределах зоны нечувствительности, то выходной сигнал блока равен нулю.

2. Если входной сигнал больше или равен верхнему входному порогу зоны нечувствительности, то выходной сигнал равен входному минус величина порога.

3. Если входной сигнал меньше или равен нижнему входному порогу зоны нечувствительности, то выходной сигнал равен входному минус величина порога.

На рис. П1.42 показан пример использования блока «Dead Zone».

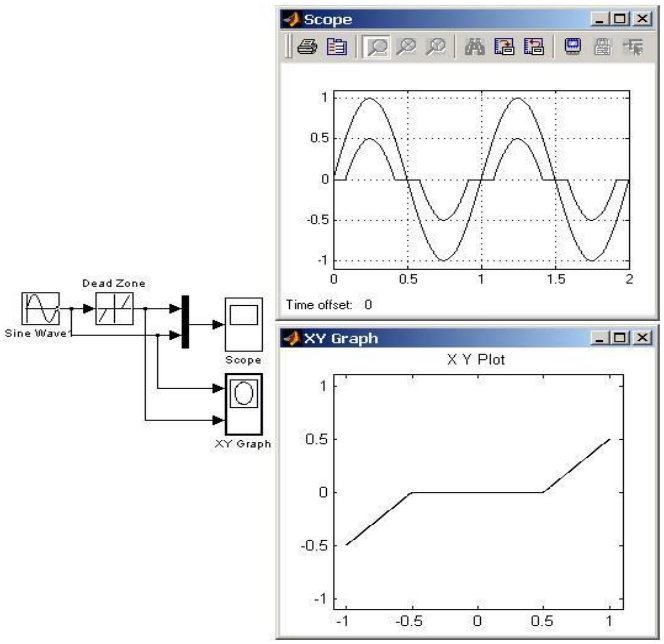


Рис. П1.42. Пример использования блока «Dead Zone»

2.3. Релейный блок «Relay»

Назначение: реализует релейную нелинейность.

Параметры блока:

1. «Switch on point» –порог включения. Значение, при котором происходит включение реле.

2. «Switch off point» – порог выключения. Значение, при котором происходит выключение реле.

3. «Output when on» – величина выходного сигнала во включенном состоянии.

4. «Output when off» – величина выходного сигнала в выключенном состоянии.

Выходной сигнал блока может принимать два значения. Одно из них соответствует включенному состоянию реле, второе – выключенному. Переход из одного состояния в другое происходит скачком при достижении входным сигналом порога включения или выключения реле. В том случае если пороги включения и выключения реле имеют разные значения, то блок реализует релейную характеристику с гистерезисом. При этом значение порога включения должно быть больше, чем значение порога выключения.

На рис. П1.43 показан пример использования блока «Relay». На временных диаграммах видно, что включение реле происходит при достижении входным сигналом величины 0.5, а выключение при -0.5.

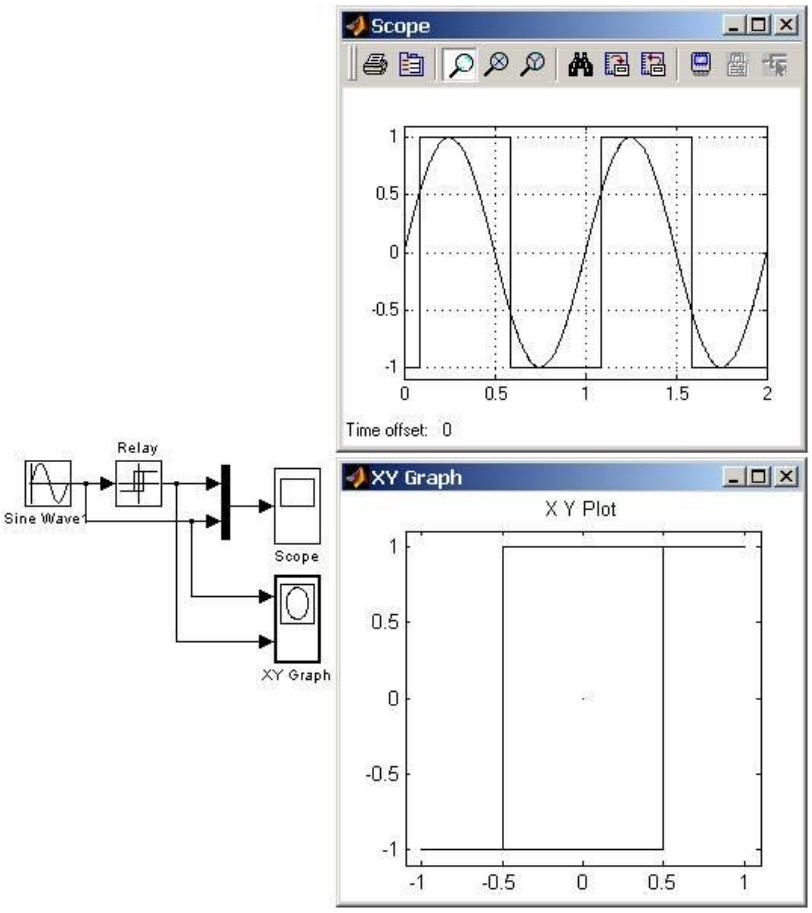


Рис. П1.43. Пример использования блока «Relay»

2.4. Блок ограничения скорости изменения сигнала «Rate Limiter»

Назначение: блок обеспечивает ограничение скорости изменения сигнала (первой производной).

Параметры блока:

1. «Rising slew rate» – уровень ограничения скорости при увеличении сигнала.

2. «Falling slew rate» – уровень ограничения скорости при уменьшении сигнала.

Вычисление производной сигнала выполняется по выражению

$$rate = \frac{u(i) - y(i-1)}{t(i) - t(i-1)},$$

где $u(i)$ – значение входного сигнала на текущем шаге;

$t(i)$ – значение модельного времени на текущем шаге;

$y(i–1)$ – значение выходного сигнала на предыдущем шаге;

$t(i–1)$ – значение модельного времени на предыдущем шаге.

Вычисленное значение производной сравнивается со значениями уровней ограничения скорости «Rising slew rate» и «Falling slew rate». Если значение производной больше, чем значение параметра «Rising slew rate», то выходной сигнал блока вычисляется по выражению

$$y(i) = \Delta tR + y(i-1),$$

где R – уровень ограничения скорости при увеличении сигнала.

Если значение производной меньше, чем значение параметра Falling slew rate, то выходной сигнал блока вычисляется по выражению

$$y(i) = \Delta tF + y(i-1),$$

где F – уровень ограничения скорости при уменьшении сигнала.

Если значение производной лежит в пределах между нижним и верхним уровнями ограничения, то выходной сигнал блока равен входному:

$$y(i) = u(i).$$

На рис. П.1.44 показан пример использования блока «Rate Limiter» при подаче на его вход прямоугольного периодического сигнала.

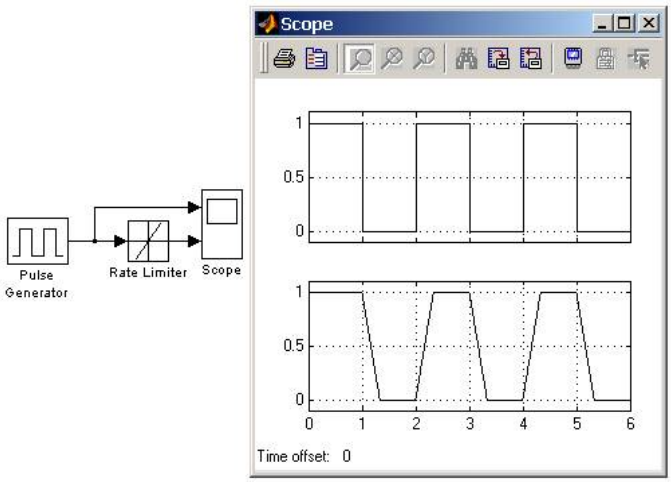


Рис. П1.44. Пример использования блока «Rate Limiter»

2.5. Блок квантования по уровню «Quantizer»

Назначение: блок обеспечивает квантование входного сигнала с одинаковым шагом по уровню.

Параметры блока: «Quantization interval» – шаг квантования по уровню.

На рис. П1.45 показан пример использования блока «Quantizer», выполняющего квантование по уровню синусоидального сигнала. Шаг квантования задан равным 0,5.

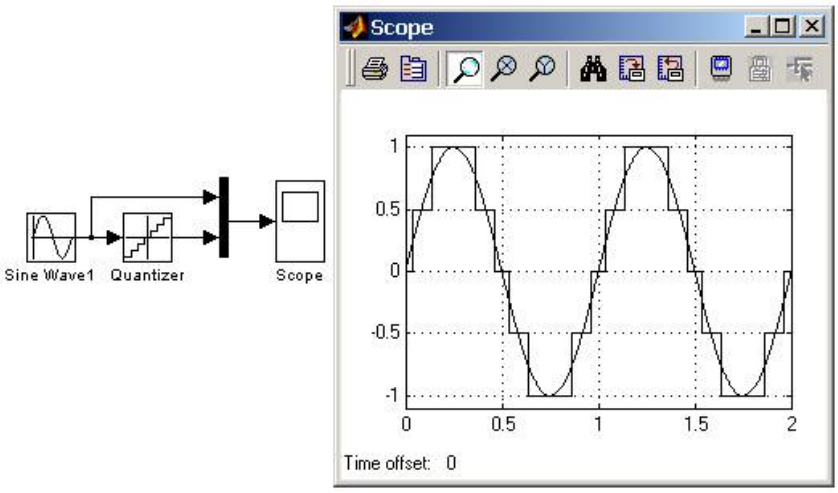


Рис. П1.45. Пример использования блока «Quantizer»

2.6. Блок сухого и вязкого трения «Coulomb and Viscous Friction»

Назначение: моделирует эффекты сухого и вязкого трения.

Параметры блока:

1. «Coulomb friction value (Offset)» – величина сухого трения.

2. «Coefficient of viscous friction (Gain)» – коэффициент вязкого трения.

Блок реализует нелинейную характеристику, соответствующую выражению:

$$y = sign(u)(Gain \cdot abs(u) \cdot Offset),$$

где u – входной сигнал, y – выходной сигнал, $Gain$ – коэффициент вязкого трения , $Offset$ – величина сухого трения.

На рис. П1.46 показан пример использования блока «Coulomb and Viscous Friction». Оба параметра блока заданы равными 1.

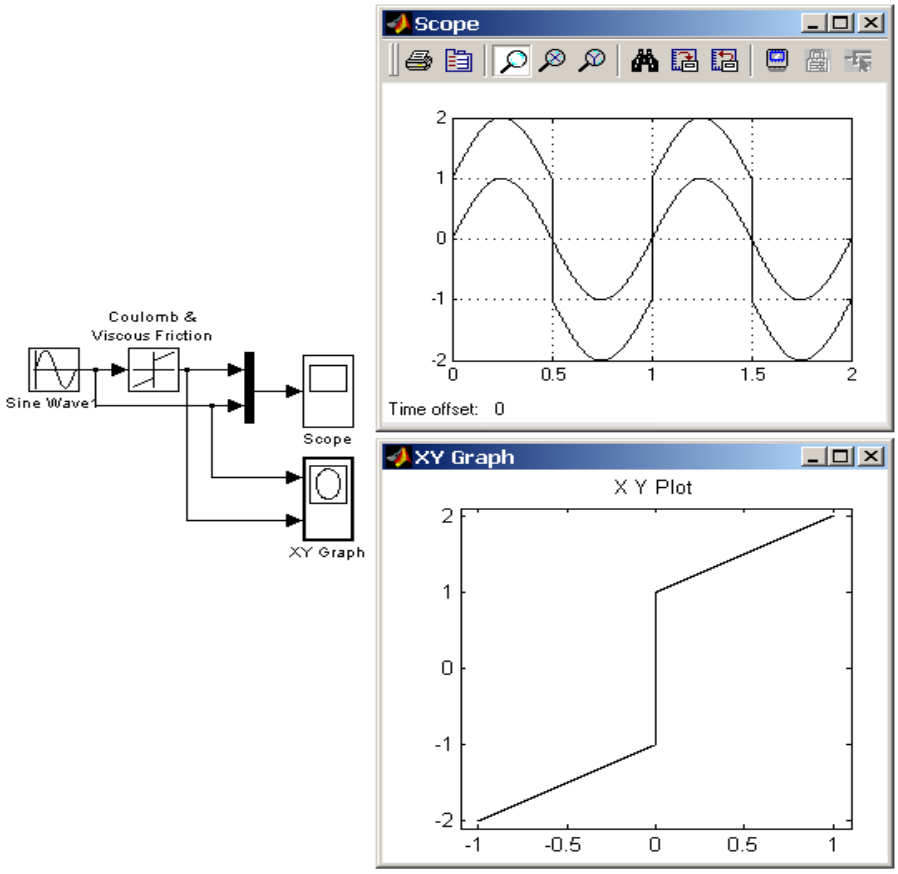


Рис. П1.46. Пример использования блока
«Coulomb and Viscous Friction»

2.7. Блок люфта «Backlash»

Назначение: моделирует нелинейность типа «люфт».

Параметры блока:

1. «Deaband width» – ширина люфта.
2. «Initial output» – начальное значение выходного сигнала.

Сигнал на выходе будет равен заданному значению «Initial output», пока входной сигнал при возрастании не достигнет значения «Deaband width»/2 (где U – входной сигнал), после чего выходной сигнал будет равен U – «Deaband width»/2. После того как произойдет смена направления изменения входного сигнала, он будет оставаться неизменным, пока входной сигнал не изменится на величину «Deaband width»/2, после чего выходной сигнал будет равен U + «Deaband width»/2.

На рис. П1.47 показан пример работы блока «Backlash». Входной сигнал блока гармонический с линейно возрастающей амплитудой.

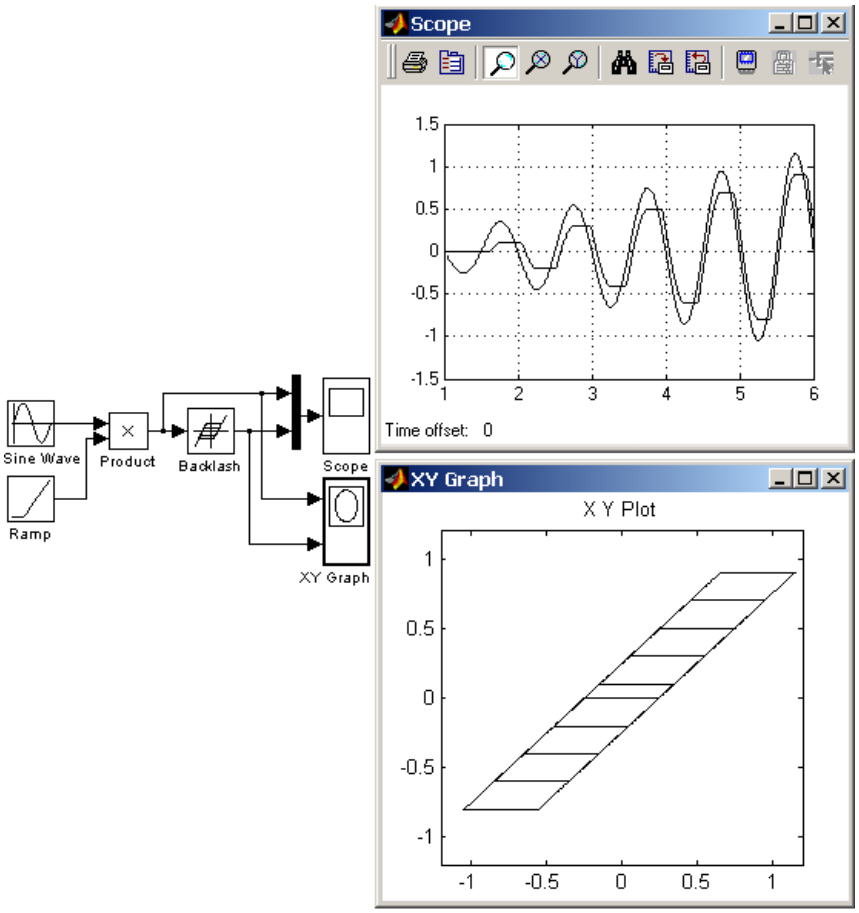


Рис. П1.47. Пример использования блока «Backlash»

2.8. Блок переключателя «Switch»

Назначение: выполняет переключение входных сигналов по сигналу управления.

Параметр блока: «Threshold» – порог управляющего сигнала.

Блок работает следующим образом.

Если сигнал управления, подаваемый на средний вход, больше, чем величина порогового значения «Threshold», то на выход блока проходит сигнал с первого (верхнего) входа. Если сигнал управления станет меньше, чем пороговое значение, то на выход блока будет поступать сигнал со второго (нижнего) входа.

На рис. П1.48 показан пример работы блока «Switch». В том случае, когда сигнал на управляющем входе ключа равен единице, на выход блока проходит гармонический сигнал, если же управляющий сигнал равен нулю, то на выход проходит сигнал нулевого уровня от блока «Ground». Пороговое значение управляющего сигнала задано равным 0,5.

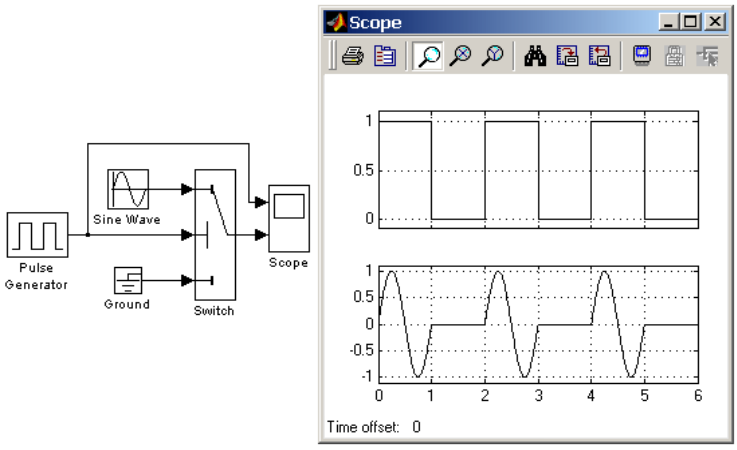


Рис. П1.48. Применение переключателя «Switch»

2.9. Блок многовходового переключателя «Multiport Switch»

Назначение: выполняет переключение входных сигналов по сигналу управления, задающему номер активного входного порта.

Параметр блока: «Number of inputs» – количество входов.

Блок многовходового переключателя «Multiport Switch» пропускает на выход сигнал с того входного порта, номер которого равен текущему значению управляющего сигнала. Если управляющий сигнал не является сигналом целого типа, то блок «Multiport Switch» производит отбрасывание дробной части числа, при этом в командном окне MATLAB появляется предупреждающее сообщение.

На рис. П1.49 показан пример работы блока «Multiport Switch». Управляющий сигнал переключателя имеет три уровня и формируется с помощью блоков «Constant», «Step», «Step1» и «Sum». На выход блока «Multiport Switch» в зависимости от уровня входного сигнала проходят гармонические сигналы, имеющие разные частоты.

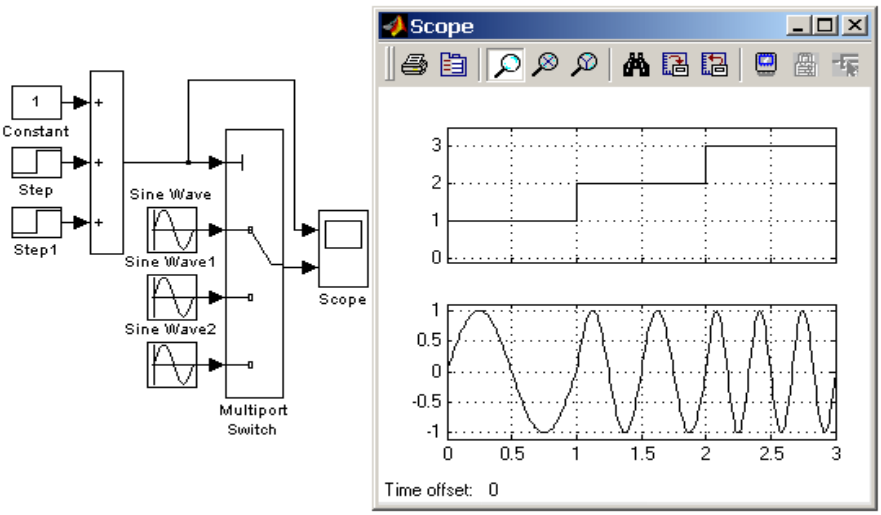


Рис. П1.49. Применение переключателя «Multiport Switch»

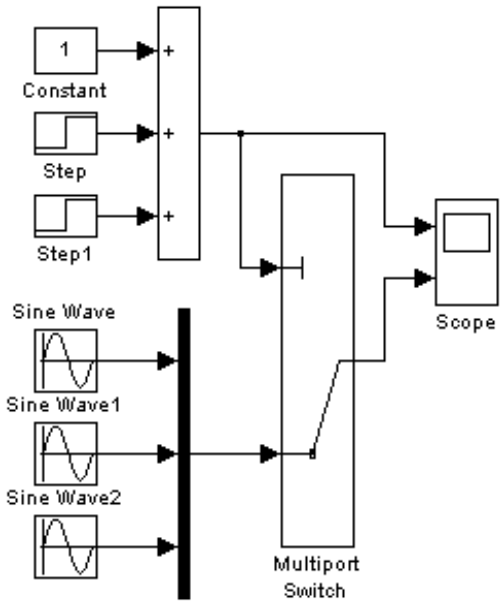


Рис. П1.50. Применение переключателя «Multiport Switch» при векторном входном сигнале

Количество входов блока «Multiport Switch» можно задать равным 1. В этом случае на вход блока необходимо подать векторный сигнал, а сам блок будет пропускать на выход тот элемент вектора, номер которого совпадает с уровнем управляющего сигнала.

На рис. П1.50 показан пример использования блока «Multiport Switch» при векторном сигнале. Временные диаграммы работы для данного примера совпадают с рассмотренными в предыдущем примере.

2.10. Блок ручного переключателя «Manual Switch»

Назначение: выполняет переключение входных сигналов по команде пользователя.

Параметров блок не имеет.

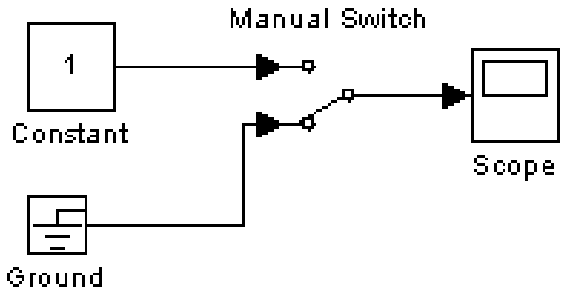


Рис. П1.51. Пример использования блока «Manual Switch»

Командой на переключение является двойной щелчок левой клавишей «мыши» на изображении блока. При этом изображение блока изменяется, показывая, какой входной сигнал в данный момент проходит на выход блока. Переключение блока можно выполнять как до начала моделирования, так и в процессе расчета.

На рис. П1.51 показан пример использования блока «Manual Switch».

3. «Signal & Systems» – блоки преобразования сигналов и вспомогательные блоки

3.1. Мультиплексор (смеситель) «Mux»

Назначение: объединяет входные сигналы в вектор.

Параметры блока:

1. «Number of Inputs» – количество входов.

2. «Display option» – способ отображения. Выбирается из списка:

- *bar* – вертикальный узкий прямоугольник черного цвета;

- *signals* – прямоугольник с белым фоном и отображением меток входных сигналов;

- *none* – прямоугольник с белым фоном без отображения меток входных сигналов.

Входные сигналы блока могут быть скалярными и (или) векторными.

Если среди входных сигналов есть векторы, то количество входов можно задавать как вектор с указанием числа элементов каждого вектора. Например, выражение [2 3 1] определяет три входных сигнала: первый – вектор из двух элементов, второй – вектор из трех элементов, и последний – скаляр. В том случае, если размерность входного вектора не совпадает с указанной в параметре «Number of Inputs», то после начала расчета «Simulink» выдаст сообщение об ошибке. Размерность входного вектора можно задавать как –1 (минус один). В этом случае размерность входного вектора может быть любой.

Параметр «Number of Inputs» можно задавать также в виде списка меток сигналов, например: «Vector1», «Vector2», «Scalar». В этом случае метки сигналов будут отображаться рядом с соответствующими соединительными линиями.

Сигналы, подаваемые на входы блока, должны быть одного типа (действительного или комплексного). Примеры использования блока «Mux» показаны на рис. П1.52.

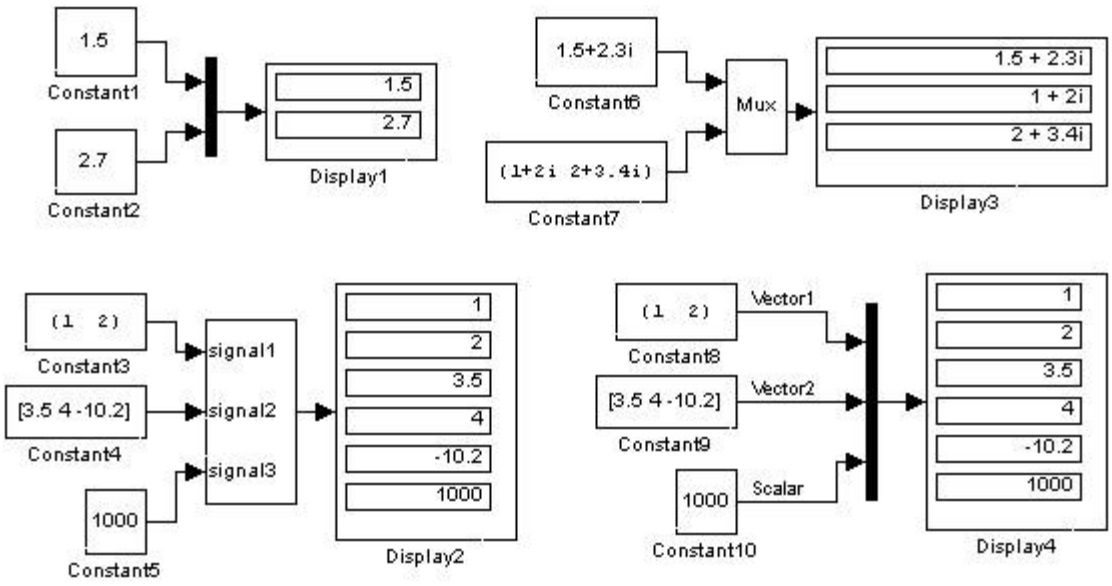


Рис. П1.52. Примеры использования блока «Mux»

3.2. Демультиплексор (разделитель) «Demux»

Назначение: разделяет входной векторный сигнал на отдельные составляющие.

Параметры блока:

1. «Number of Outputs» – количество выходов.

2. «Bus Selection Mode» (флажок) – режим разделения векторных сигналов.

Входным сигналами в обычном режиме является вектор, сформированный любым способом. Выходными сигналами являются скаляры или векторы, количество которых и размерность определяется параметром «Number of Outputs» и размерностью входного вектора.

Если количество выходов P (значение параметра «Number of Outputs») равно размерности входного сигнала N, то блок выполняет разделение входного вектора на отдельные элементы.

Если количество выходов P меньше, чем размерность входного сигнала N, то размерность первых P-1 выходных сигналов равна отношению N/P, округленному до ближайшего большего числа, а размерность последнего выходного сигнала равна разности между размерностью входного сигнала и суммой размерностей первых P-1 выходов. Например, если размерность входного сигнала равна 8, а количество выходов равно 3, то первые два выходных вектора будут иметь размерность $ceil(8/3) = 3$, а последний выходной вектор будет иметь размерность $8 – (3 + 3) = 2$.

Параметр «Number of Outputs» может быть задан также с помощью вектора, определяющего размерность каждого выходного сигнала. Например, выражение [2 3 1] определяет три выходных сигнала, первый сигнал – вектор из двух элементов, второй сигнал – вектор из трех элементов, и последний сигнал – скаляр. Размерность можно также задавать как –1 (минус один). В этом случае размерность соответствующего выходного сигнала определяется как разность между размерностью входного вектора и суммой размерностей заданных выходных сигналов. Например, если размерность входного вектора равна 6, а параметр «Number of Outputs» задан выражением [1 –1 3], то второй выходной сигнал будет иметь размерность 6 – (3 + 1) = 2.

Примеры использования блока «Demux» показаны на рис. П1.53.

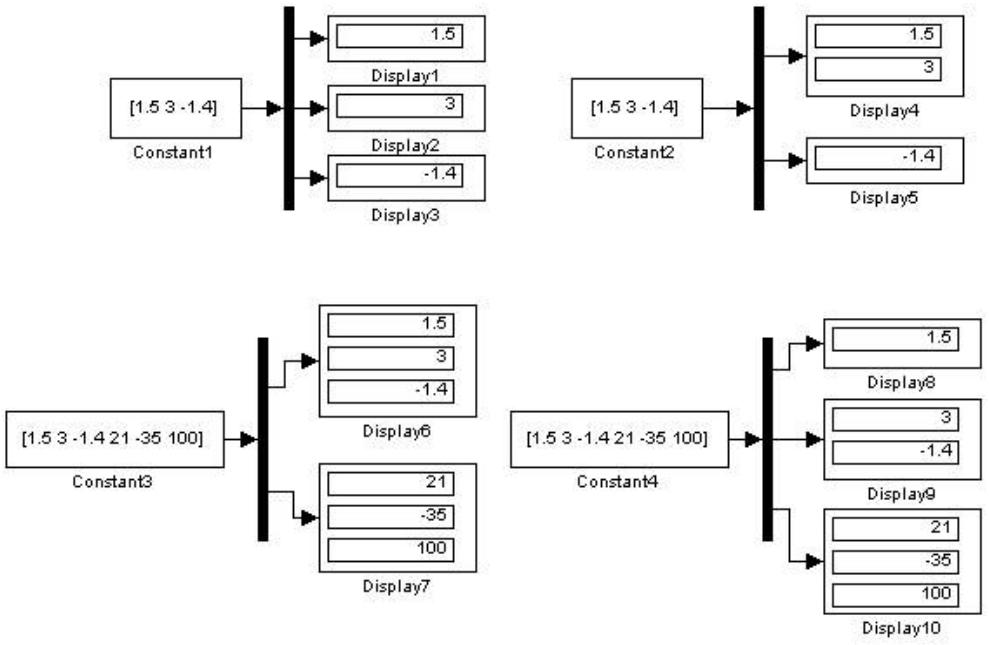


Рис. П1.53. Примеры использования блока «Demux»

В режиме «Bus Selection Mode» блок «Demux» работает не с отдельными элементами векторов, а с векторными сигналами в целом. Входной сигнал в этом режиме должен быть сформирован блоком «Mux» или другим блоком «Demux». Параметр «Number of Outputs» в этом случае задается в виде скаляра, определяющего количество выходных сигналов, либо в виде вектора, каждый элемент которого определяет количество векторных сигналов в данном выходном сигнале. Например, при входном сигнале, состоящем из трех векторов, параметр «Number of Outputs», заданный вектором [2 1], определит два выходных сигнала, первый из которых будет содержать два векторных сигнала, а второй – один.

Примеры использования блока «Demux» в режиме «Bus Selection Mode» показаны на рис. П1.54.

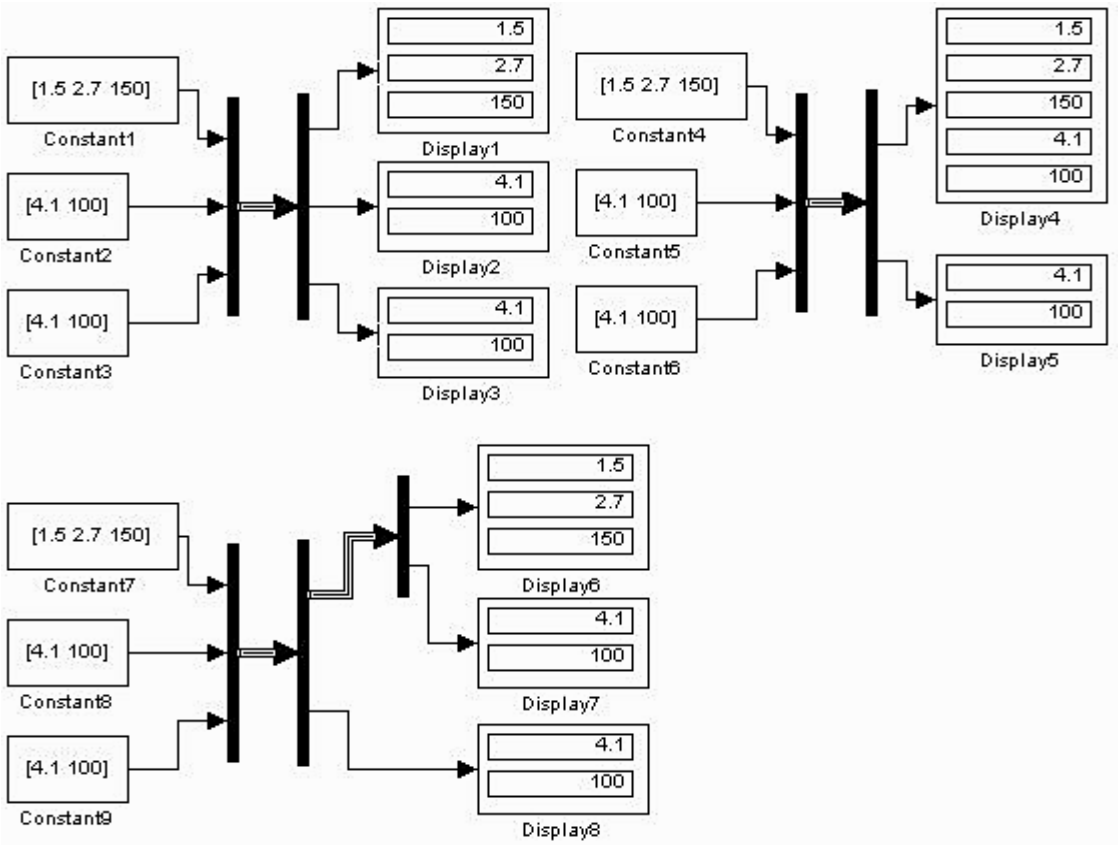


Рис. П1.54. Примеры использования блока «Demux»
в режиме «Bus Selection Mode»

3.3. Блок шинного формирователя «Bus Creator»

Назначение: формирует шину из сигналов различных типов.

Параметры блока:

«Signal naming options» – способ именования сигнала. Выбирается из списка:

«Inherit bus signal names from input ports» – наследовать имена входных сигналов.

«Require input signal names to match signals below» – требуется ввести имена сигналов.

«Number of inputs ports» – количество входных портов.

«Signals in bus» – список сигналов, объединяемых в шину.

«Rename selected signal» – новое имя выделенного сигнала. Параметр доступен, если выбрана опция «Require input signal names to match signals below».

Блок позволяет объединять любые сигналы (векторные, матричные, комплексные, действительные и целые разных типов) в единую шину. Такая шина позволяет сократить количество соединительных линий в модели. Для разделения шины на отдельные составляющие необходимо использовать блок «Bus Selector».

Окно параметров блока позволяет отыскать блок, который является источником сигнала. Для такого поиска необходимо выделить название сигнала в списке «Signals in bus» и нажать с помощью мыши кнопку «Find». Блок, являющийся источником выбранного сигнала, будет выделен цветом.

На рис. П1.55 показан пример формирования шины с помощью блока «Bus Creator» и окно параметров этого блока. Там же показан выделенный цветом источник сигнала «signal 2» – блок «Constant3», найденный с помощью изложенной выше процедуры.

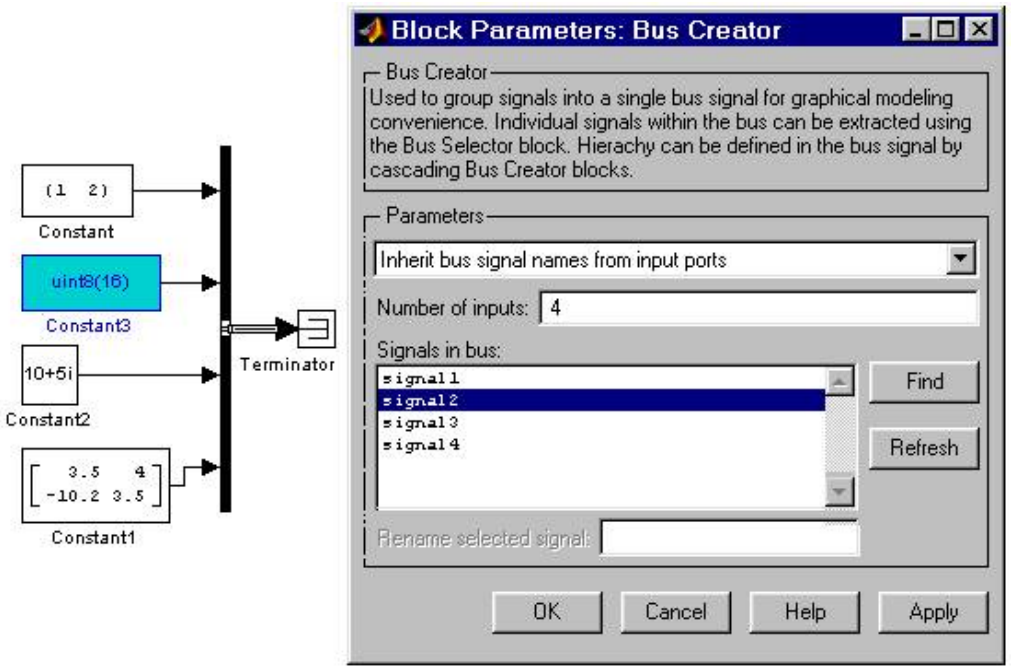


Рис. П1.55. Пример использования блока «Bus Creator»

3.4. Блок шинного селектора «Bus Selector»

Назначение: выделяет из шины требуемые сигналы.

Параметры блока:

1. «Signals in the bus» – имеющиеся в шине сигналы (входные сигналы).

2. «Selected signals» – выделенные сигналы (выходные сигналы).

3. «Muxed output» (флажок) – объединение выходных сигналов в один.

Шина может быть сформирована блоком «Mux» или «Bus Creator».

Для извлечения сигнала из шины необходимо открыть окно параметров блока, выделить сигнал в окне «Signals in the bus» и с помощью кнопки «Select» скопировать имя сигнала в окно «Selected signals». Для удаления сигнала из списка «Selected signals» необходимо выделить его имя в правом списке окна параметров блока и затем воспользоваться кнопкой «Remove».

С помощью кнопок «Up» и «Down» можно изменить порядок расположения сигналов в шине, перемещая их в окне «Selected signals» вверх или вниз соответственно.

Установка параметра «Muxed output» позволяет объединить сигналы в шину.

На рис. П1.56 показаны примеры использования блока «Bus Selector» и окно его параметров.

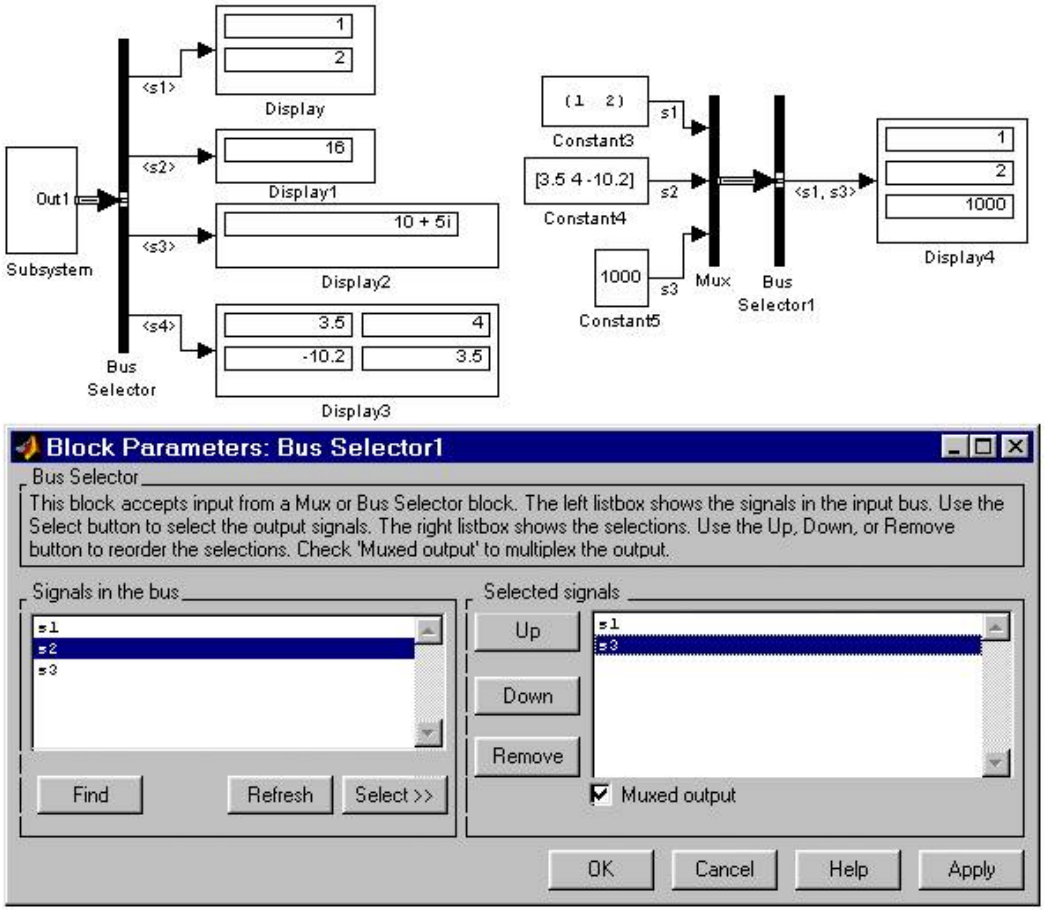


Рис. П1.56. Примеры использования блока «Bus Selector»

3.5. Блок селектора «Selector»

Назначение: выбирает из вектора или матрицы требуемые элементы.

Параметры блока:

1. «Input Type» – тип входного сигнала. Выбирается из списка:

- *vector* – вектор;

- *matrix* – матрица.

Список параметров блока изменяется в зависимости от типа входного сигнала.

2. «Source of element indices» – источник индексов элементов вектора. Выбирается из списка:

- *internal* – внутренний. Индексы выбираемых элементов вектора задаются параметром Elements;

- *external* – внешний. Индексы элементов вектора задаются с помощью внешнего входного сигнала.

3. «Elements» – список индексов элементов входного вектора, передаваемых на выход блока. Задается в виде вектора. Значение параметра –1 (минус один) предписывает выбор всех элементов вектора.

4. «Input port width» – размерность входного вектора.

5. «Source of row indices» – источник индексов строк элементов матрицы.

6. «Rows» – список индексов строк матрицы.

7. «Source of column indices» – источник индексов столбцов элементов матрицы.

8. «Columns» – список индексов столбцов матрицы.

Внешний вид блока изменяется в зависимости от установленных параметров блока. При выборе внешних источников индексов элементов на изображении блока появляются дополнительные входы, обозначенные следующими символами:

• *E* – вход сигнала, задающего индексы выбираемых элементов вектора;

• *R* – вход сигнала, задающего индексы строк матрицы;

• *C* – вход сигнала, задающего индексы столбцов матрицы.

Блок выбирает во входном векторе или матрице и передает на выход только те сигналы, которые определены в параметрах блока или заданы внешним входным сигналом.

На рис. П1.57 приведены примеры использования блока «Selector» для различных вариантов настройки блока.

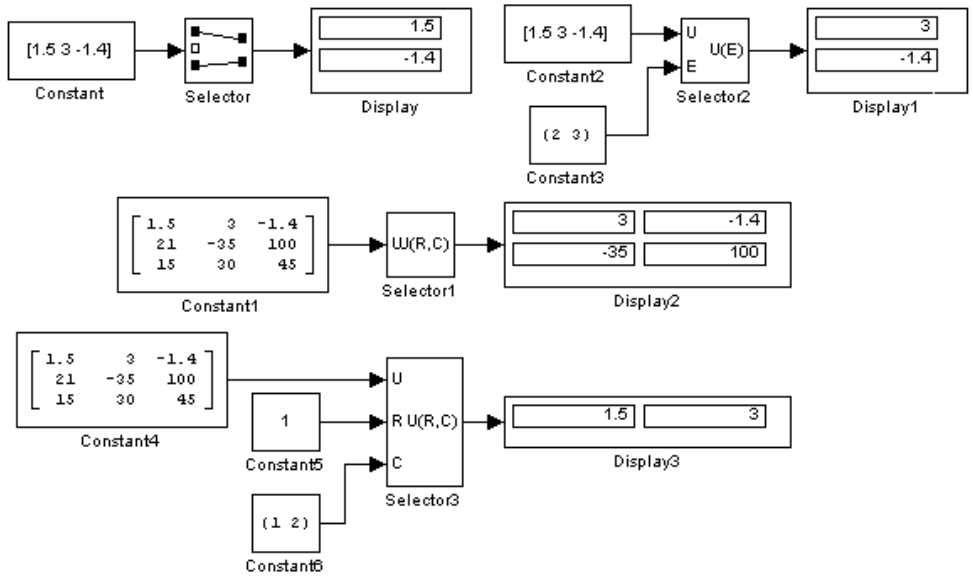


Рис. П1.57. Примеры использования блока «Selector»

3.6. Блок присвоения новых значений элементам массива «Assignment»

Назначение: заменяет элементы вектора или матрицы.

Параметры блока:

1. «Input Type» – тип входного сигнала. Выбирается из списка:

vector – вектор;

matrix – матрица.

Список параметров блока изменяется в зависимости от типа входного сигнала.

2. «Source of element indices» – источник индексов элементов вектора. Выбирается из списка:

internal – внутренний. Индексы выбираемых элементов вектора задаются параметром Elements;

external – внешний. Индексы элементов вектора задаются с помощью внешнего входного сигнала.

3. «Elements» – список индексов элементов входного вектора, передаваемых на выход блока. Задается в виде вектора. Значение параметра –1 (минус один) предписывает выбор всех элементов вектора.

4. «Source of row indices» – источник индексов строк элементов матрицы.

5. «Rows» – список индексов строк матрицы.

6. «Source of column indices» – источник индексов столбцов элементов матрицы.

7. «Columns» – список индексов столбцов матрицы.

Блок выполняет замену отдельных элементов первого входного массива на элементы второго входного массива в соответствии со списком индексов. Список индексов может задаваться как параметр блока или считываться из внешнего управляющего сигнала.

Внешний вид блока изменяется в зависимости от установленных параметров блока. При выборе внешних источников индексов элементов на изображении блока появляются дополнительные входы, обозначенные следующими символами:

• «E» – вход сигнала, задающего индексы выбираемых элементов вектора;

• «R» – вход сигнала, задающего индексы строк матрицы;

• «C» – вход сигнала, задающего индексы столбцов матрицы.

На рис. П1.58 приведены примеры использования блока «Assignment» для различных вариантов настройки блока.

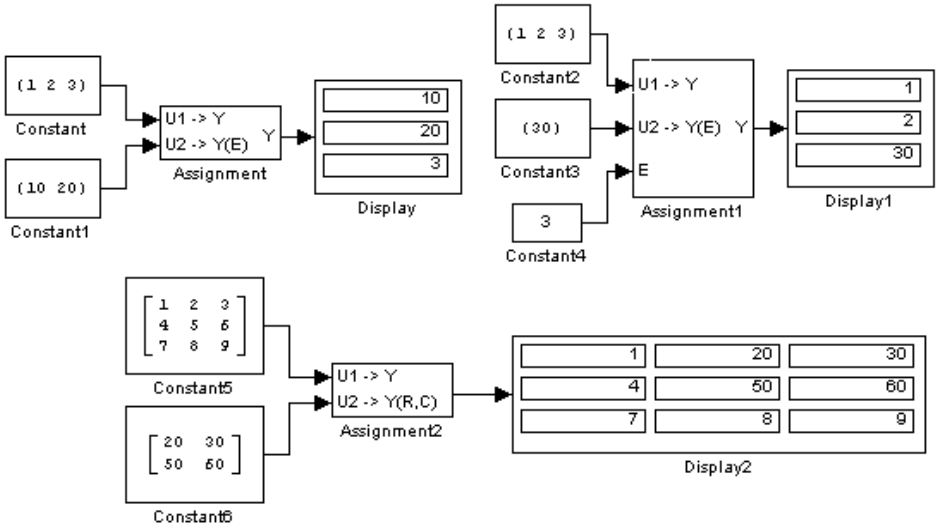


Рис. П1.58. Примеры использования блока «Assignment»

3.7. Блок объединения сигналов «Merge»

Назначение: блок выполняет объединение входных сигналов в единый векторный сигнал.

Параметры:

1. «Number of inputs» – количество входов.

2. «Initial output» – начальное значение выходного сигнала. Если этот параметр не задан, то на выход блока проходит сигнал, значение которого было вычислено последним.

3. «Allow unequal port widths» (флажок) – разрешить неодинаковую размерность входных портов.

4. «Input port offsets» – смещение входного сигнала. Задается в виде вектора, каждое значение которого определяет расположение соответствующего сигнала в выходном векторе.

Блок передает на выход значение сигнала, вычисленное последним.

С помощью параметра «Input port offsets» можно регулировать расположение входных сигналов в результирующем векторе.

Рамерность выходного сигнала определяется в соответствии с выражением $\max(w_1+o_1, w_2+o_2, \ldots, w_k+o_k)$, где w_k – размерность k-го входного сигнала, o_k – смещение k-го входного сигнала.

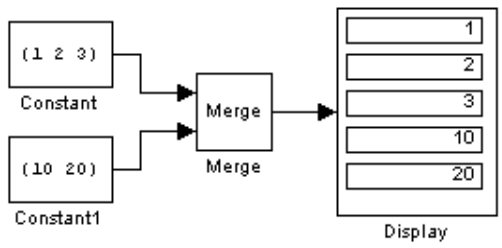


Рис. П1.59. Пример использования блока «Merge» для объединения входных сигналов

На рис. П1.59 приведен пример использования блока «Merge» для объединения двух векторов. Параметр «Input port offsets» в примере задан вектором [0 3].

Следующий пример (рис. П1.60) демонстрирует свойство блока пропускать на выход сигнал, который был вычислен последним.

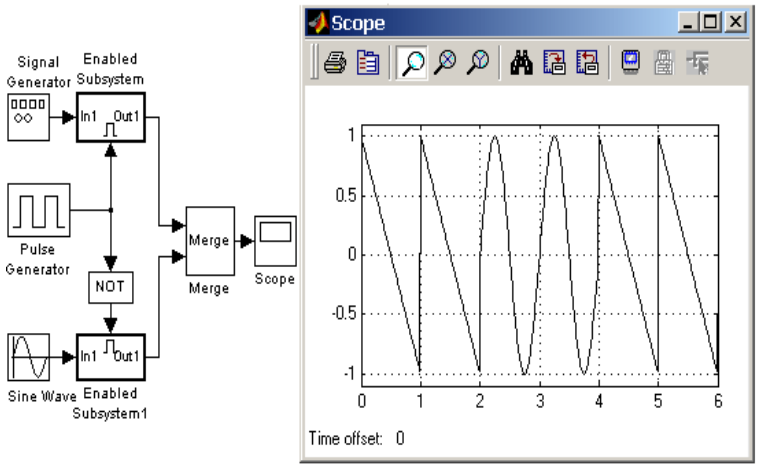


Рис. П1.60. Пример использования блока «Merge» для объединения выходных сигналов

В примере использованы блоки управляемых подсистем «Enabled Subsystem», которые выполняют вычисления только в том случае, если на управляющий вход подсистемы подан ненулевой сигнал. В данном примере подсистема не выполняет какие-либо вычисления, а лишь пропускает сигнал со своего входа на выход. Таким образом, на выход блока «Merge» поочередно проходят гармонический либо пилообразный сигналы.

3.8. Блок объединения сигналов в матрицу «Matrix Concatenation»

Назначение: блок выполняет объединение (конкатенацию) входных векторов или матриц.

Параметры блока:

1. «Number of inputs» – количество входов.

2. «Concatenation method» – способ объединения. Выбирается из списка:

- «Horizontal» – горизонтальный. Массивы объединяются добавлением новых массивов справа.
- «Vertical» – вертикальный. Массивы объединяются добавлением новых массивов снизу.

Примеры использования блока «Matrix Concatenation» приведены на рис. П1.61.

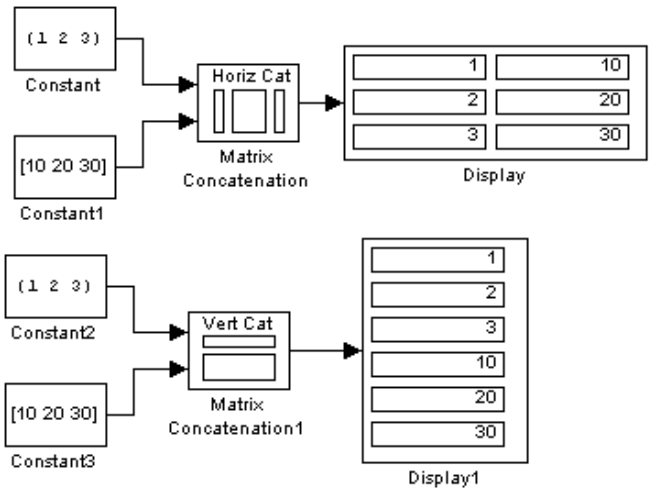


Рис. П1.61. Примеры использования блока «Matrix Concatenation»

3.9. Блок передачи сигнала «Goto»

Назначение: блок выполняет передачу сигнала к блоку «From».

Параметры блока:

1. «Tag» – идентификатор сигнала.

2. «Tag visibility» – признак видимости. Выбирается из списка:

- *local* – сигнал передается в пределах локальной подсистемы;

• *scoped* – сигнал передается в пределах локальной подсистемы и подсистемах нижнего уровня иерархии;

• *global* – сигнал передается в пределах всей модели.

Использование блока «Goto» совместно с блоком «From» обеспечивает передачу сигнала без линии связи. Для передачи могут использоваться сигналы любого типа.

В зависимости от выбранного параметра «Tag visibility» изменяется внешний вид блока.

Идентификатор сигнала помещается в квадратные скобки, если признак видимости имеет значение *local*. Например [*A*], где *A* – идентификатор сигнала.

Идентификатор сигнала помещается в фигурные скобки, если признак видимости имеет значение *scoped*. Например {*A*}.

Идентификатор сигнала отображается на пиктограмме блока без дополнительных символов, если признак видимости имеет значение *global*.

На рис. П1.62. показан «беспроводной» способ передачи сигнала от источника синусоидального сигнала к блоку «Scope» в подсистему.

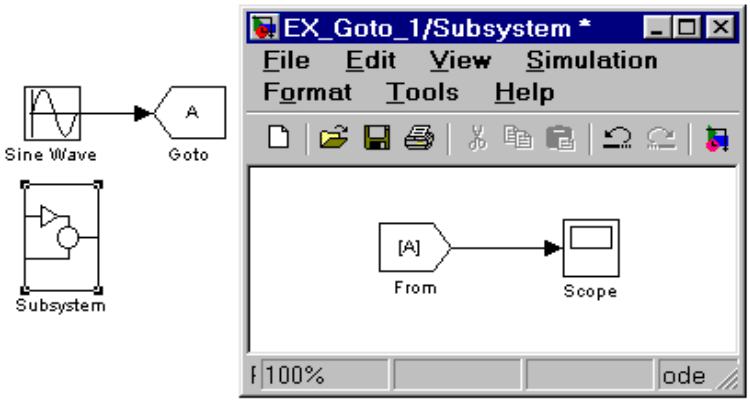


Рис. П1.62. Применение блока «Goto»

3.10. Блок приема сигнала «From»

Назначение: блок выполняет прием сигнала от блока «Goto».

Параметр блока: «Goto tag» – идентификатор принимаемого сигнала. Параметр должен совпадать с идентификатором, указанным в соответствующем блоке «Goto».

Использование блока «From» совместно с блоком «Goto» обеспечивает передачу сигнала без линии связи. Признак видимости сигнала отображается на пиктограмме блока таким же способом, что и у блока «Goto». В модели может быть сколь угодно много блоков «From», принимающих сигнал от одного блока «Goto».

На рис. П1.63 показан пример использования блоков «From» в модели. В примере один блок «Goto» передает сигнал трем блокам «From» (двум в основной модели и одному в подсистеме).

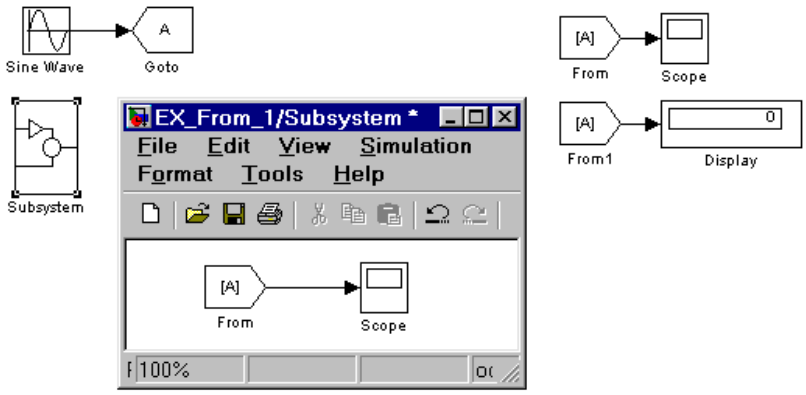


Рис. П1.63. Применение блока «From»

3.11. Блок признака видимости сигнала «Goto Tag Visibility»

Назначение: блок отображает признак видимости сигнала, передаваемого блоком «Goto».

Параметр блока: «Goto Tag» – идентификатор сигнала, передаваемого блоком «Goto». Блок необходимо включать в состав модели или подсистемы в том случае, если для передаваемых сигналов задана область видимости *scoped*. Блок помещается в те подсистемы, на которые распространяется область видимости передаваемых данных. Блок не участвует в передаче сигнала, а лишь отображает имя передаваемого сигнала.

Пример использования блока показан на рис. П1.64.

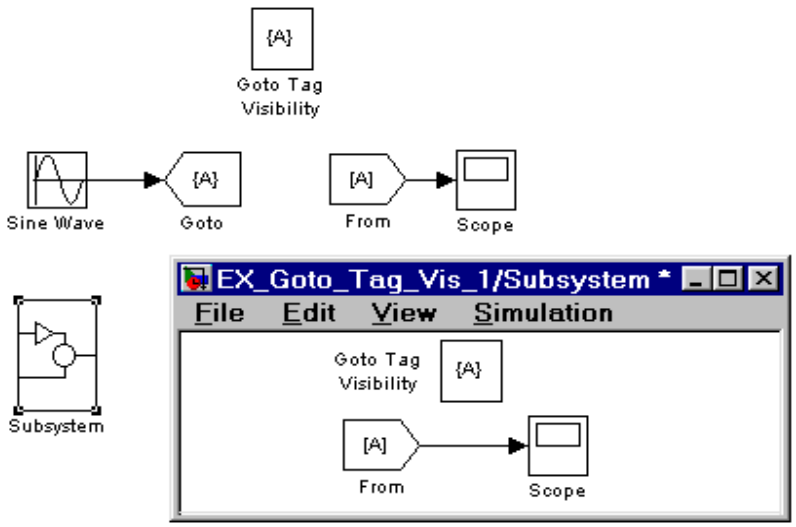


Рис. П1.64. Применение блока «Goto Tag Visibility»

3.12. Блок создания общей области памяти «Data Store Memory»

Назначение: блок создает поименованную область памяти для хранения данных.

Параметры блока:

1. «Data store name» – имя области памяти.

2. «Initial value» – начальное значение.

3. «Interpret vector parameters as 1-D» (флажок) – интерпретировать вектор параметров данных как одномерный вектор.

Блок используется совместно с блоками «Data Store Write» (запись данных) и «Data Store Read» (считывание данных).

Параметр «Initial value» задает не только начальное значение сигнала, но и его размерность. Например, если начальное значение сигнала задано матрицей [0 1; 2 3], то сохраняемый сигнал должен быть матрицей 2 × 2.

Если блок «Data Store Memory» расположен в модели верхнего уровня, то заданную им область памяти можно использовать как в самой модели, так и во всех подсистемах нижнего уровня иерархии. Если блок «Data Store Memory» расположен в подсистеме, то заданную им область памяти можно использовать в данной подсистеме и всех подсистемах нижнего уровня иерархии.

Блок работает с действительными сигналами типа «double».

Пример использования блока «Data Store Memory» совместно с блоками «Data Store Write» и «Data Store Read» показан на рис. П1.65.

3.13. Блок записи данных в общую область памяти «Data Store Write»

Назначение: блок записывает данные в поименованную область памяти.

Параметры блока:

1. «Data store name» – имя области памяти.

2. «Sample time» – шаг модельного времени.

Операция записи выполняется для значения сигнала, полученного на предыдущем шаге расчета.

В модели могут использоваться несколько блоков «Data Store Write», выполняющих запись в одну область памяти. Однако, если запись производится на одном и том же шаге расчета, то результат будет не предсказуем.

Пример использования блока «Data Store Write» совместно с блоками «Data Store Memory» и «Data Store Read» показан на рис. П1.65.

3.14. Блок считывания данных из общей области памяти «Data Store Read»

Назначение: блок считывает данные из поименованной области памяти.

Параметры:

1. «Data store name» – имя области памяти.
2. Sample time – шаг модельного времени.

Операция считывания выполняется на каждом шаге расчета.

В модели могут использоваться несколько блоков «Data Store Read», выполняющих считывание данных из одной и той же области памяти. Пример использования блока «Data Store Read» совместно с блоками «Data Store Memory» и «Data Store Write» показан на рис. П1.65. В примере используется триггерная подсистема, выполняющая вычисления по переднему фронту управляющего сигнала. Таким образом, запись значений в общую область памяти происходит только в моменты изменения управляющего сигнала в положительном направлении. В остальные моменты времени значения данных в области памяти не изменяются.

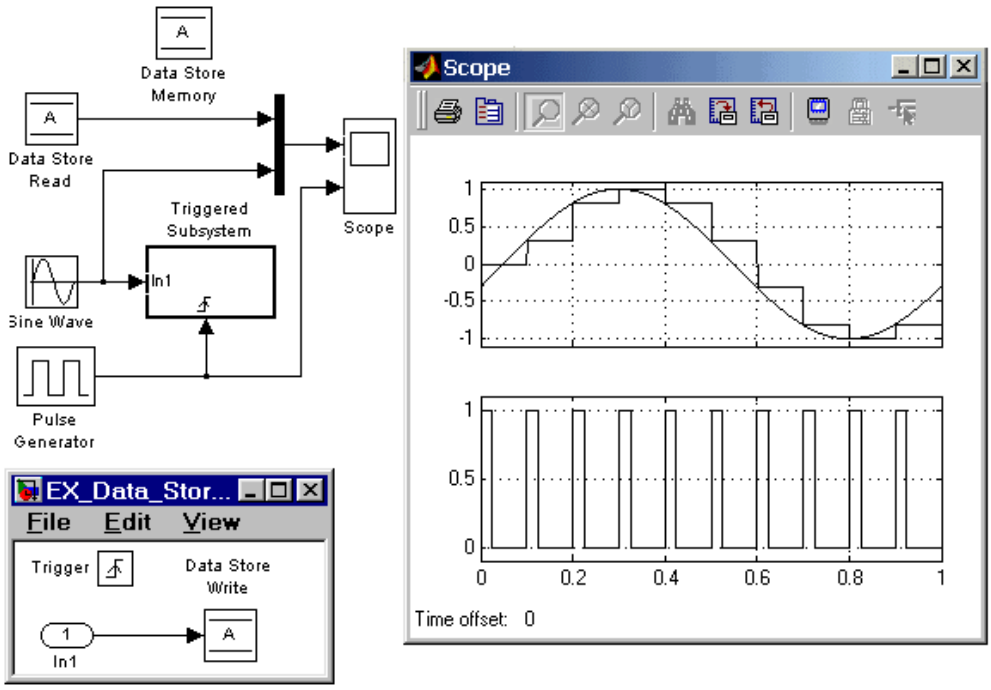


Рис. П1.65. Использование блоков«Data Store Memory», «Data Store Write» и «Data Store Read»

3.15. Блок преобразования типа сигнала «Data Type Conversion»

Назначение: блок преобразует тип входного сигнала.

Параметры блока:

1. «Data type» – тип данных выходного сигнала. Может принимать значения (выбираются из списка): *auto*, *double*, *single*, *int*8, *int*16, *int*32, *uint*8, *uint*16, *uint*32 и *boolean*.

2. «Saturate on integer overflow» (флажок) – подавлять переполнение целого. При установленном флажке ограничение сигналов целого типа выполняется корректно.

Значение *auto* параметра «Data type» используется в том случае, если необходимо установить тип данных такой же, как у входного порта блока, получающего сигнал от данного блока.

Входной сигнал блока может быть действительным или комплексным. В случае комплексного входного сигнала выходной сигнал также будет комплексным.

Блок работает со скалярными, векторными и матричными сигналами.

На рис. П1.66 показаны примеры использования блока «Data Type Conversion».

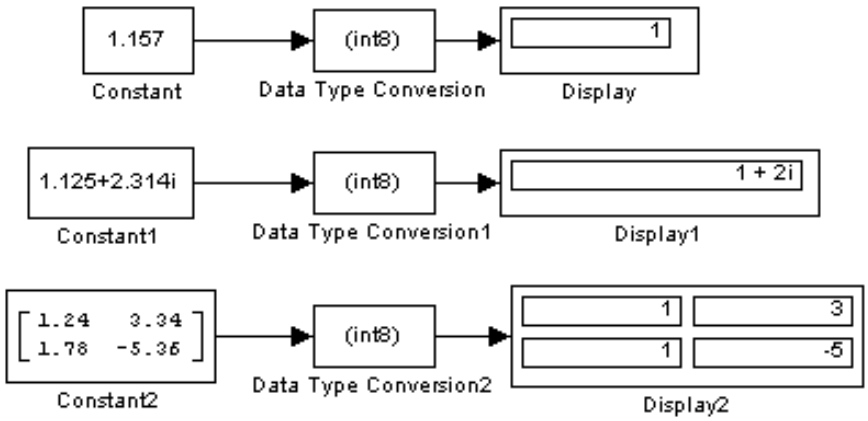


Рис. П1.66. Использование блока «Data Type Conversion»

3.16. Блок преобразования размерности сигнала «Reshape»

Назначение: блок изменяет размерность векторного или матричного сигнала.

Параметр блока «Output dimensionality» – вид размерности выходного сигнала. Выбирается из списка:

- «1-D array» – одномерный массив (вектор);
- «Column vector» – вектор-столбец;

- «Row vector» – вектор-строка;
- «Customize» – матрица или вектор заданной размерности. Для векторного выходного сигнала параметр задается как скаляр, определяющий число элементов выходного вектора. Для матричного выходного сигнала параметр задается как вектор, определяющий количество строк и столбцов выходной матрицы. Значение параметра должно соответствовать количеству элементов во входном массиве. В случае матричных сигналов данные выбираются из столбцов входной матрицы и последовательно заносятся в столбцы выходной матрицы;
- «Output dimensions» – значение размерности выходного сигнала. Параметр доступен, если вид размерности установлен как «Customize».

Примеры использования блока «Reshape» показаны на рис. П1.67.

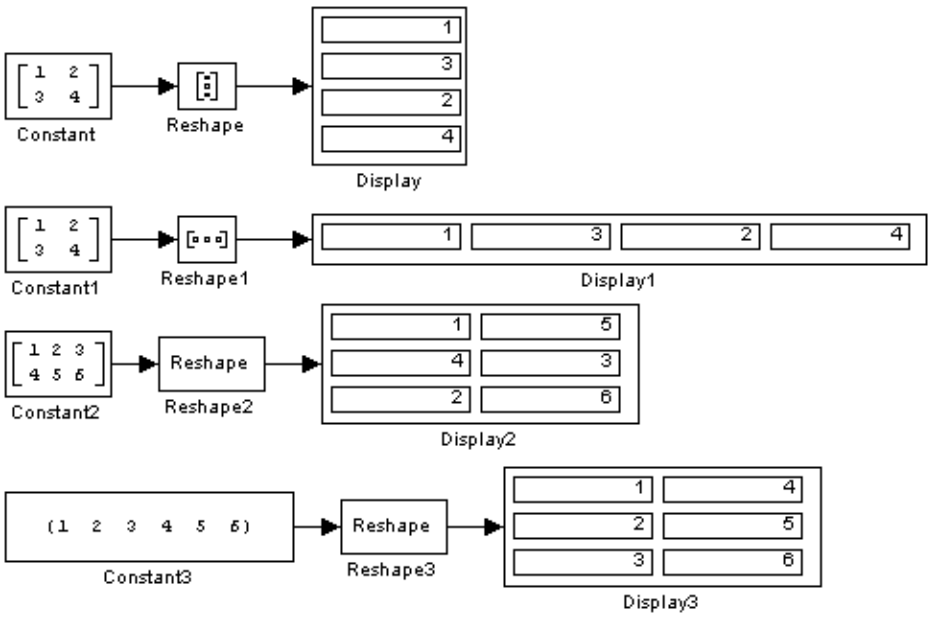


Рис. П1.67. Примеры использования блока «Reshape»

3.17. Блок определения размерности сигнала «Width»

Назначение: вычисляет размерность входного сигнала.

Параметров блок не имеет.

Входным сигналом блока может быть действительный или комплексный сигнал любого типа.

Выходной сигнал блока имеет тип *double*.

Примеры использования блока «Width» показаны на рис. П1.68.

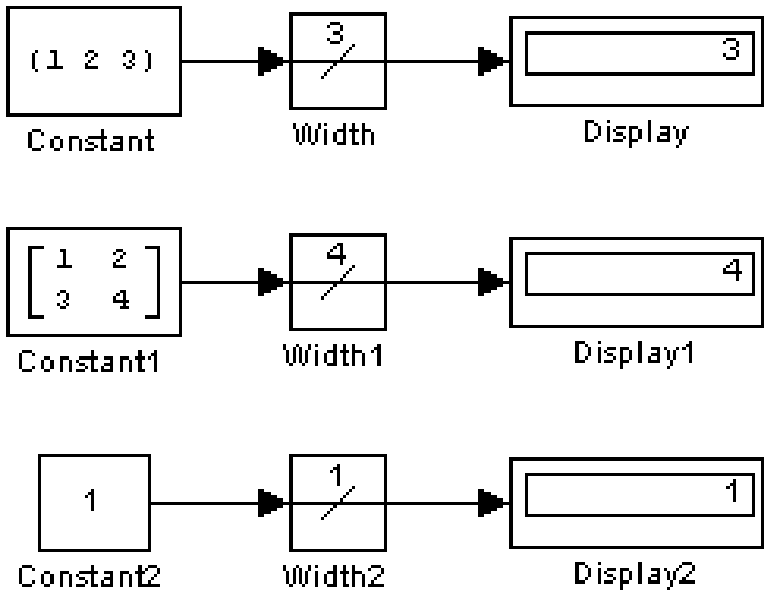


Рис. П1.68. Примеры использования блока «Width»

3.18. Блок определения момента пересечения порогового значения «Hit Crossing»

Назначение: определяет момент времени, когда входной сигнал пересекает заданное пороговое значение.

Параметры блока:

1. «Hitcrossingoffset» – порог. Значение, пересечение которого входным сигналом требуется идентифицировать.

2. «Hit crossing direction» – направление пересечения. Выбирается из списка:

- *rising* – возрастание;
- *failing* – убывание;
- *either* – оба направления.

3. «Show output port» (флажок) – показать выходной порт. В том случае, если этот флажок снят, то точка пересечения сигналом порогового уровня находится, но выходной сигнал блоком не генерируется.

В момент пересечения порогового уровня блок вырабатывает единичный сигнал длительностью в один шаг модельного времени. Пример использования блока «Hit Crossing» показан на рис. П1.69.

Блок определяет моменты пересечения в обоих направлениях синусоидальным сигналом уровня 0,5.

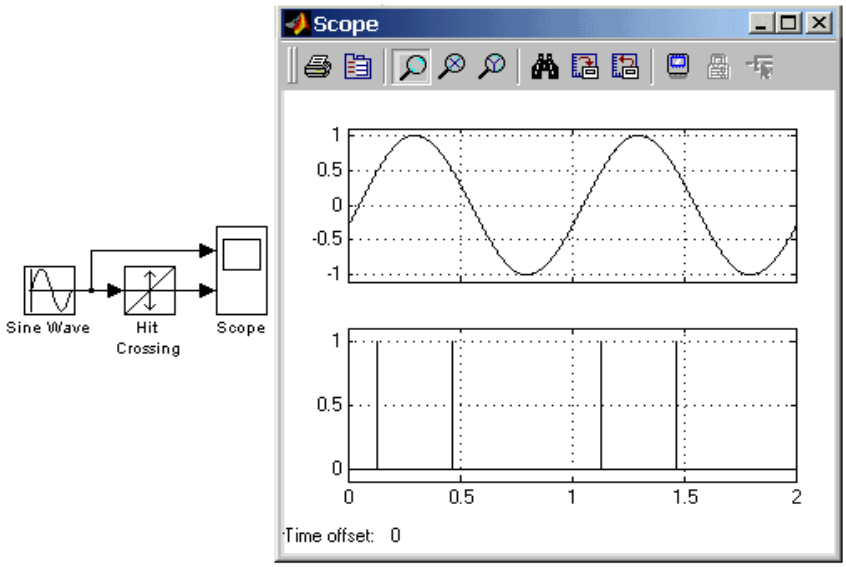


Рис. П1.69. Пример использования блока «Hit Crossing»

3.19. Блок установки начального значения сигнала «IC»

Назначение: задает начальное значение сигнала.

Параметр блока: «Initial value» – начальное значение.

Выходной сигнал блока «IC» равен значению параметра «Initial value» на первом шаге расчета вне зависимости от величины входного сигнала блока. На остальных расчетных шагах входной сигнал проходит на выход блока без каких-либо изменений.

Пример использования блока «IC» показан на рис. П1.70. В примере начальное значение сигнала задано равным 0,5. Шаг расчета задан равным одной секунде.

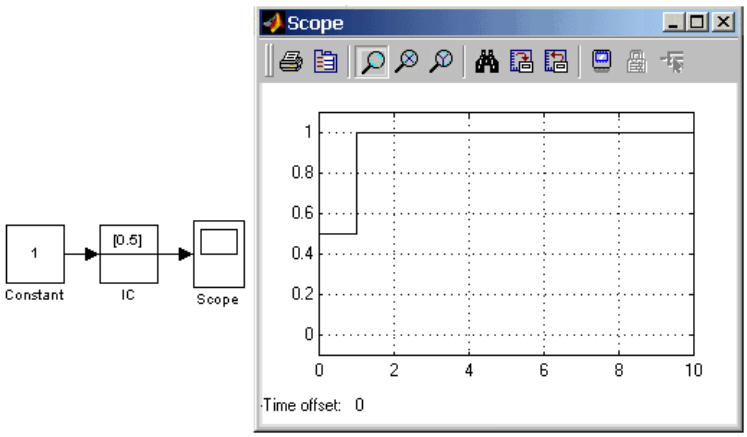


Рис. П1.70. Пример использования блока «IC»

3.20. Блок проверки сигнала «Signal Specification»

Назначение: выполняет проверку сигнала на соответствие заданным для сигнала параметрам.

Параметры блока:

1. «Dimension» – размерность сигнала. Задается скаляром, если входной сигнал векторный, или матрицей вида [*m n*] (*m* – количество строк, *n* – количество столбцов), если входной сигнал – матрица. Если значение параметра задано как –1 (минус 1), то проверка не производится.

2. «Sample time» – шаг модельного времени. Задается вектором вида [*period offset*], где *period* – значение шага модельного времени, *offset* – смещение. Если значение параметра задано как –1 (минус 1), то проверка не производится. Можно также задавать значение –1 (минус 1) и отдельно для параметров *period* или *offset*. В этом случае не будет проводиться проверка именно этих параметров.

3. «Data Type» – тип данных. Выбирается из списка: auto (проверка не производится), *double*, *single*, *int*8, *uint*8, *int*16, *uint*16, *int*32, *uint*32 или *boolean*.

4. «Signal type» – тип сигнала. Выбирается из списка: *auto* (проверка не производится), *real* или *complex*.

На пиктограмме блока отображаются проверяемые параметры сигнала и их значения. Пример использования блока «Signal Specification» показан на рис. П1.71.

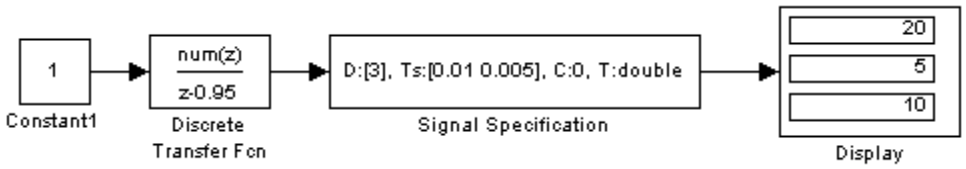


Рис. П1.71. Пример использования блока «Signal Specification»

3.21. Датчик свойств сигнала «Probe»

Назначение: блок позволяет получить численные значения параметров сигнала.

Параметры блока:

1. «Probe width» (флажок) – определение числа элементов в векторном или матричном сигнале.

2. «Probe Sample time» (флажок) – определение значения эталонного времени.

3. «Probe Complex Signal» (флажок) – определение типа сигнала (возвращает 1, если сигнал представлен в комплексном виде, и 0 в противном случае).

4. «Probe signal dimension» (флажок) – определение размерности сигнала.

Контролируются те параметры, для которых установлены флажки. Числом отмеченных флажков задается число выходов блока. Установка флажка для какого-либо параметра приводит к появлению на изображении блока порта, с которого можно считывать значение данного параметра сигнала. Пример использования блока «Probe» показан на рис. П1.72.

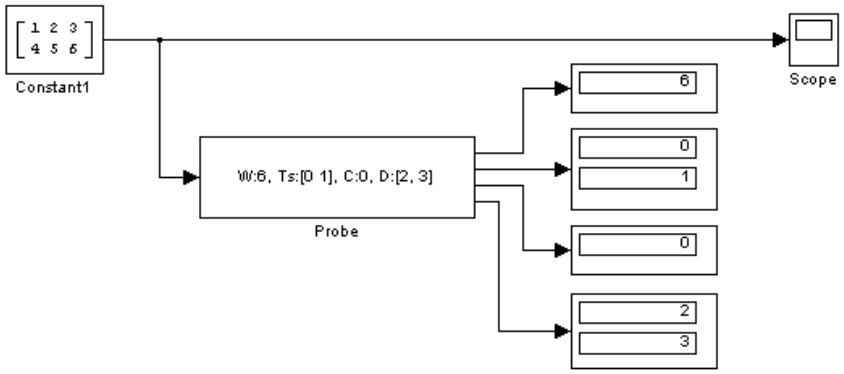


Рис. П1.72. Пример использования блока «Probe»

3.22. Блок, задающий количество итераций «Function-Call Generator»

Назначение: блок позволяет задать количество итераций на каждом шаге модельного времени для управляемой подсистемы.

Параметры блока:

1. «Sample time» – шаг модельного времени.

2. «Number of iterations» – количество итераций.

Блок используется совместно с управляемыми подсистемами «Function-CallSubsystem» или «TriggeredSubsystem». Для управляющих блоков внутри этих подсистем параметр «Trigger type» должен иметь значение *function-call*.

Пример использования блока «Function-Call Generator» показан на рис. П1.73. В примере использована управляемая подсистема, выходной сигнал которой увеличивается на единицу при каждом ее вызове. Для первой подсистемы блок «Function-Call Generator» задает количество итераций на каждом шаге, равное 1, а для второй – равное 3.

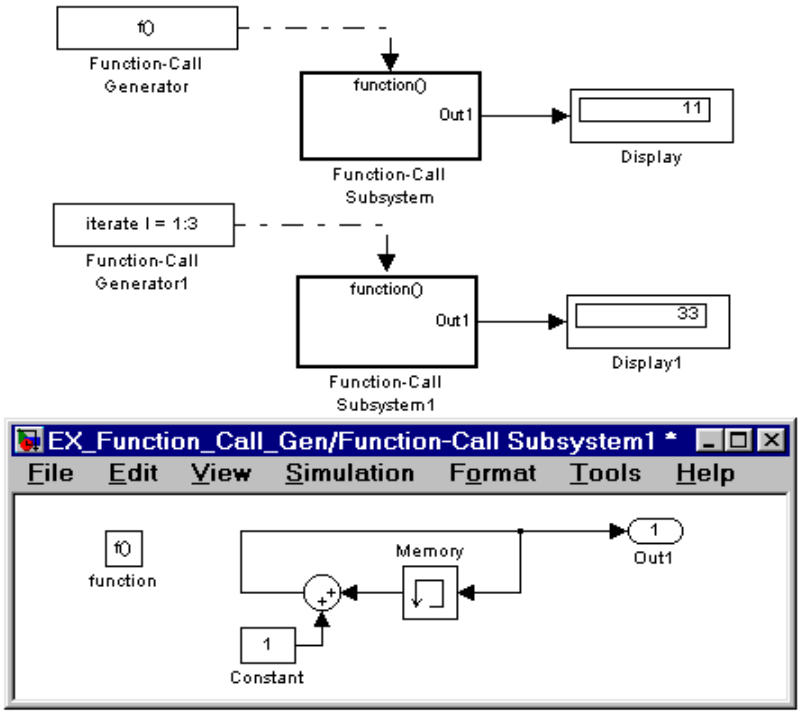


Рис. П1.73. Пример использования блока
«Function-Call Generator»

3.23. Информационный блок «Model Info»

Назначение: блок отображает информацию о модели.

Параметры блока:

1. «Model properties» – свойства модели.
2. «Created» – дата и время создания модели.
3. «Creator» – данные об авторе.
4. «Modified by» – данные о пользователе, вносившем изменения.
5. «Modified Date» – дата изменения.
6. «Modified Comment» – описание изменений.
7. «Model Version» – версия модели.
8. «Description» – описание модели.
9. «Last Modification Date» – дата последнего изменения.
10. «Horizontal text allignment» – способ выравнивания текста по горизонтали. Выбирается из списка:

- «Center» – по центру;
- «Left» – по левому краю;
- «Right» – по правому краю;
- «Show block frame» (флажок) – отобразить рамку блока.

Для отображения данных на пиктограмме блока необходимо с помощью кнопки --> скопировать нужный параметр из окна «Model properties» в окно редактирования. В блоке может отображаться статическая информация, которую пользователь вносит сам (например, данные об авторе, описание модели и т.п.) и динамически обновляемая информация (например, дата создания модели, дата последней модификации и т.п.). Динамически обновляемая информация представляется в окне блока как ссылка на переменную, которая ее содержит. Ссылка имеет вид %<имя_переменной>. Например, ссылка %<Last Modification Date> означает, что в требуемой позиции будет выведено значение переменной *Last Modification Date*, содержащей дату последней модификации модели.

На пиктограмме блока отображается также часть информации, заданная с помощью команды «Model Properties» меню «File» окна модели.

Пример использования блока «Model Info» показан на рис. П1.74. Там же показано окно параметров данного блока.

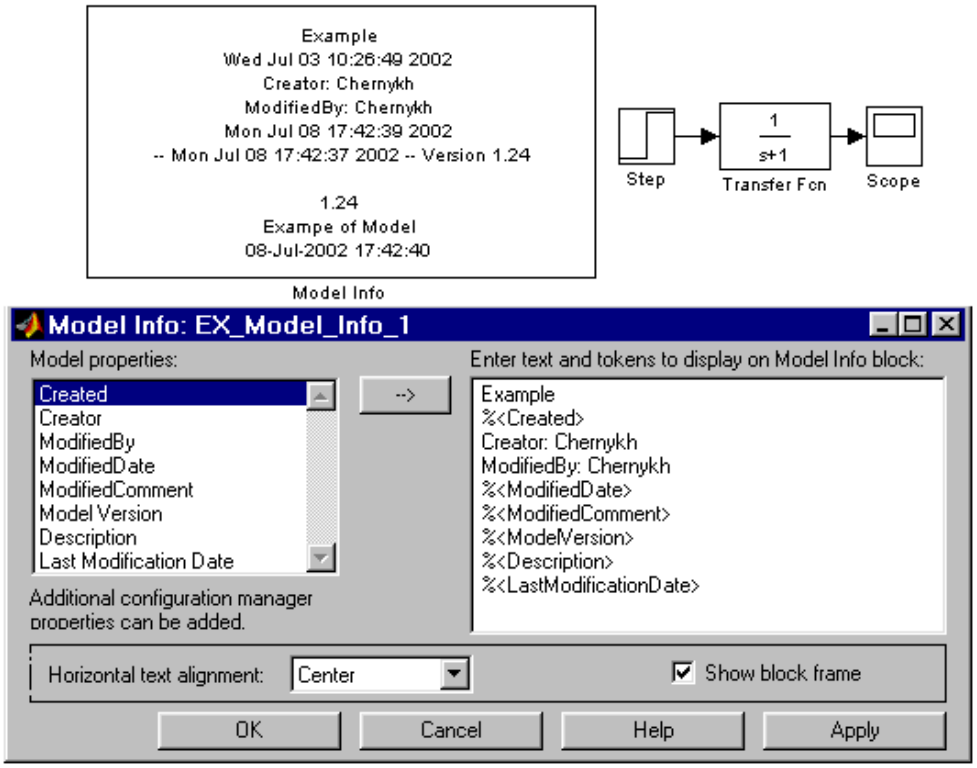


Рис. П1.74. Пример использования блока «Model Info»

Приложение 2

СОЗДАНИЕ И МОДЕЛИРОВАНИЕ НЕЙРОСЕТИ В MATLAB

Чтобы запустить «NNTool», необходимо выполнить одноимённую команду в командном окне MATLAB:>>*nntool*.

После этого появится главное окно NNTool, именуемое «Окном управления сетями и данными» («Network/Data Manager») (рис. П2.1).

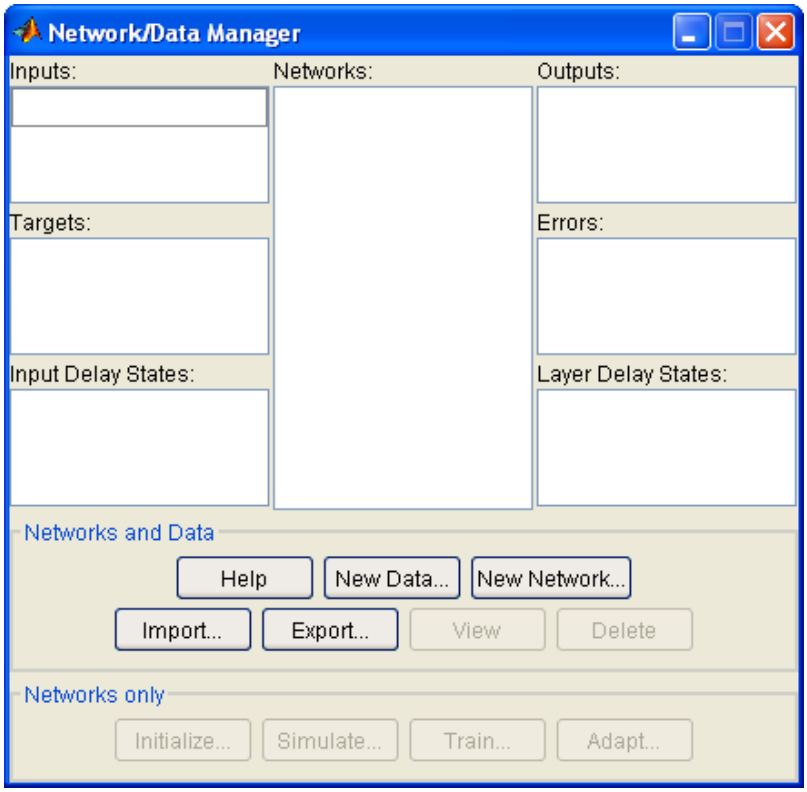


Рис. П2.1. Главное окно «NNTool»

Панель «Сети и данные» («Networks and Data») имеет функциональные клавиши со следующими назначениями:

- «Help» (Помощь) – краткое описание управляющих элементов данного окна;
- «New Data…» (Новые данные) – вызов окна, позволяющего создавать новые наборы данных;
- «New Network…» (Новая сеть) – вызов окна создания новой сети;
- «Import…» (Импорт) – импорт данных из рабочего пространства MATLAB в пространство переменных «NNTool»;
- «Export…» (Экспорт) – экспорт данных из пространства переменных «NNTool» в рабочее пространство MATLAB;

• «View» (Вид) – графическое отображение архитектуры выбранной сети;

• «Delete» (Удалить) – удаление выбранного объекта.

На панели «Только сети» («Networks only») расположены клавиши для работы исключительно с сетями. При выборе указателем мыши объекта любого другого типа эти кнопки становятся неактивными.

При работе с «NNTool» важно помнить, что клавиши «View», «Delete», «Initialize», «Simulate», «Train» и «Adapt» (изображены на рис. П2.1 как неактивные) действуют применительно к тому объекту, который отмечен в данный момент выделением. Если такого объекта нет либо над выделенным объектом невозможно произвести указанное действие, соответствующая клавиша неактивна.

Рассмотрим создание нейронной сети с помощью «NNTool» на примере.

Пусть требуется создать нейронную сеть, выполняющую логическую функцию «И».

Создание сети

Выберем сеть, состоящую из одного персептрона с двумя входами. В процессе обучения сети на её входы подаются входные данные и производится сопоставление значения, полученного на выходе, с целевым (желаемым). На основании результата сравнения (отклонения полученного значения от желаемого) вычисляются величины изменения весов и смещения, уменьшающие это отклонение.

Итак, перед созданием сети необходимо заготовить набор обучающих и целевых данных. Составим таблицу истинности для логической функции «И», где $P1$ и $P2$ – входы, а A – желаемый выход (таблица).

ТАБЛИЦА ИСТИННОСТИ ЛОГИЧЕСКОЙ ФУНКЦИИ «И»

$P1$	$P2$	A
0	0	0
0	1	0
1	0	0
1	1	1

Чтобы задать матрицу, состоящую из четырёх векторов-строк, как входную, воспользуемся кнопкой «New Data». В появившемся окне следует произвести изменения, показанные на рис. П2.2, и нажать клавишу «Create» (Создать).

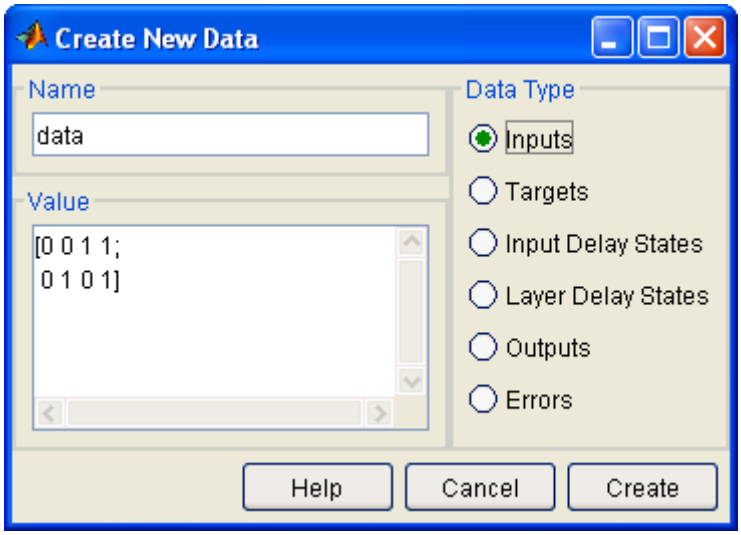


Рис. П2.2. Задание входных векторов

После этого в окне управления появится вектор *data* в разделе «Inputs». Вектор целей задаётся схожим образом (рис. П2.3).

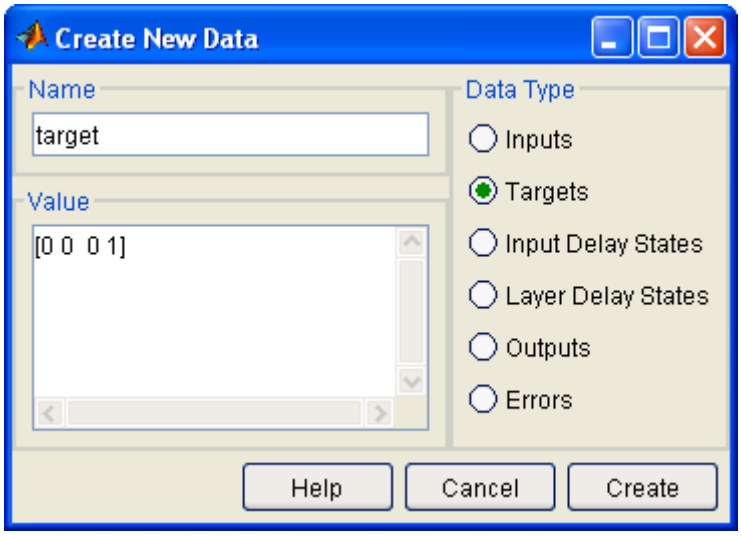


Рис. П2.3. Задание целевого вектора

После нажатия на «Create» в разделе «Targets» появится вектор *target*.

Данные в поле «Value» (Значение) могут быть представлены любым понятным MATLAB выражением. К примеру, предыдущее определение вектора целей можно эквивалентно заменить строкой вида

bitand([0 0 1 1], [0 1 0 1]).

Теперь следует приступить к созданию нейронной сети. Выбираем кнопку «New Network» и заполняем форму, как показано на рис. П2.4.

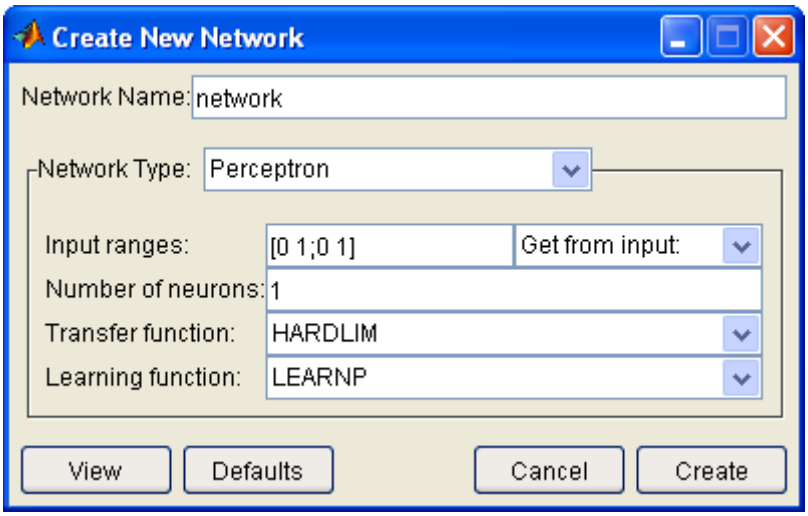


Рис. П2.4. Окно «Создание сети»

При этом поля несут следующие смысловые нагрузки:

- «Network Name» (Имя сети) – это имя объекта создаваемой сети;
- «Network Type» (Тип сети) – определяет тип сети и в контексте выбранного типа представляет для ввода различные параметры в части окна, расположенной ниже этого пункта. Таким образом, для разных типов сетей окно изменяет своё содержание;
- «Input ranges» (Входные диапазоны) – матрица с числом строк, равным числу входов сети. Каждая строка представляет собой вектор с двумя элементами: первый – минимальное значение сигнала, которое будет подано на соответствующий вход сети при обучении, второй – максимальное. Для упрощения ввода этих значений предусмотрен выпадающий список «Получить из входа» (Get from input), позволяющий автоматически сформировать необходимые данные, указав имя входной переменной;
- «Number of neurons» (Количество нейронов) – число нейронов в слое;

• «Transfer function» (Передаточная функция) – в этом пункте выбирается передаточная функция (функция активации) нейронов;

• «Learning function» (Функция обучения) – функция, отвечающая за обновление весов и смещений сети в процессе обучения.

С помощью клавиши «View» (Вид) можно посмотреть архитектуру создаваемой сети (рис. П2.5). Так мы имеем возможность удостовериться, все ли действия были произведены верно. На рис. П2.5 изображена персептронная сеть с выходным блоком, реализующим передаточную функцию с жёстким ограничением. Количество нейронов в слое равно одному, что символически отображается размерностью вектора-столбца на выходе слоя и указывается числом непосредственно под блоком передаточной функции. Рассматриваемая сеть имеет два входа, так как размерность входного вектора-столбца равна двум.

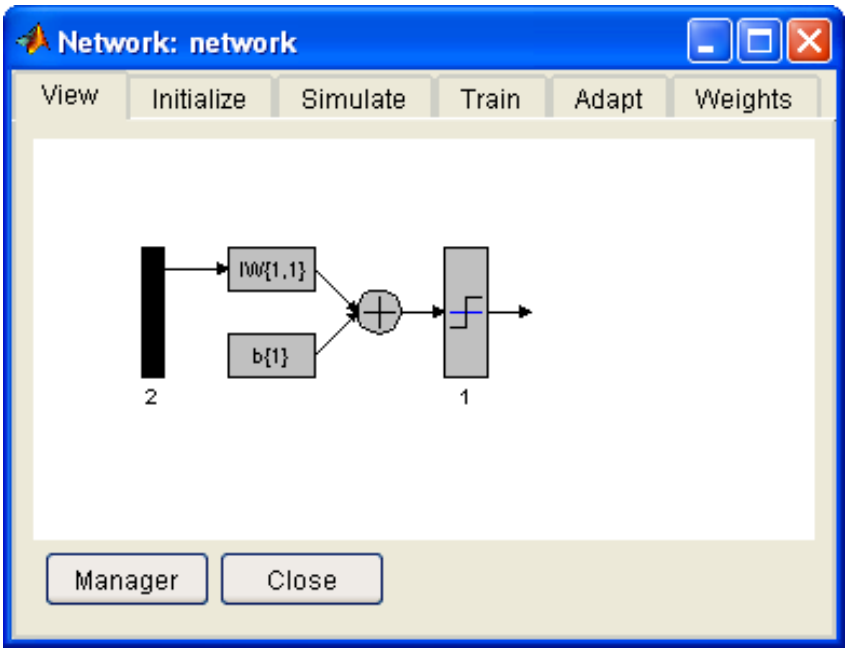


Рис. П2.5. Предварительный просмотр создаваемой сети

Итак, структура сети соответствует нашему заданию. Теперь можно закрыть окно предварительного просмотра, нажав клавишу «Close» (Закрыть), и подтвердить намерение создать сеть, нажав «Create» (Создать) в окне создания сети.

В результате проделанных операций в разделе «Networks» (Сети) главного окна «NNTool» появится объект с именем *network*.

Обучение

Наша цель – построить нейронную сеть, которая выполняет функцию логического «И». Очевидно, нельзя рассчитывать на то, что сразу после этапа создания сети последняя будет обеспечивать правильный результат (правильное соотношение «вход/выход»). Для достижения цели сеть необходимо должным образом обучить, т. е. подобрать подходящие значения параметров. В MATLAB реализовано большинство известных алгоритмов обучения нейронных сетей, среди которых представлено два для персептронных сетей рассматриваемого вида. Создавая сеть, мы указали «LEARNP» в качестве функции, реализующей алгоритм обучения (см. рис. П2.4).

Вернёмся в главное окно «NNTool». На данном этапе интерес представляет нижняя панель «Networks only» (Только сети). Нажатие любой из клавиш на этой панели вызовет окно, на множестве вкладок которого представлены параметры сети, необходимые для её обучения и прогона, а также отражающие текущее состояние сети.

Отметив указателем «мыши» объект сети *network*, вызовем окно управления сетью нажатием кнопки «Train». Перед нами возникнет вкладка «Train» окна свойств сети, содержащая, в свою очередь, ещё одну панель вкладок (рис. П2.6). Их главное назначение – управление процессом обучения. На вкладке «Training info» (Информация обучения) требуется указать набор обучающих данных в поле «Inputs» (Входы) и набор целевых данных в поле «Targets» (Цели).

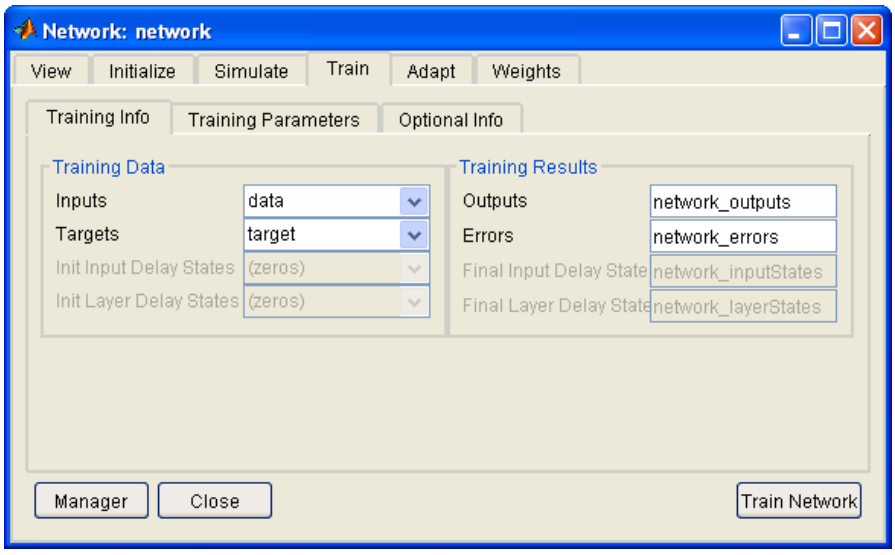


Рис. П2.6. Окно параметров сети, открытое на вкладке «Train» (обучение)

Поля «Outputs» (Выходы) и «Errors» (Ошибки) «NNTool» заполняет автоматически. При этом результаты обучения, к которым относятся выходы и ошибки, будут сохраняться в переменных с указанными именами. Завершить процесс обучения можно, руководствуясь разными критериями. Возможны ситуации, когда предпочтительно остановить обучение, полагая достаточным некоторый интервал времени. В то же время объективным критерием является уровень ошибки.

На вкладке «Training parameters» (Параметры обучения) для нашей сети (рис. П2.7) можно установить следующие поля:

- *epochs* (количество эпох) – определяет число эпох (интервал времени), по прошествии которых обучение будет прекращено. Эпохой называют однократное представление всех обучающих входных данных на входы сети;
- *goal* (достижение цели или попадание) – здесь задаётся абсолютная величина функции ошибки, при которой цель будет считаться достигнутой;
- *show* (период обновления) – период обновления графика кривой обучения, выраженный числом эпох;
- *time* (время обучения) – по истечении указанного здесь временного интервала, выраженного в секундах, обучение прекращается.

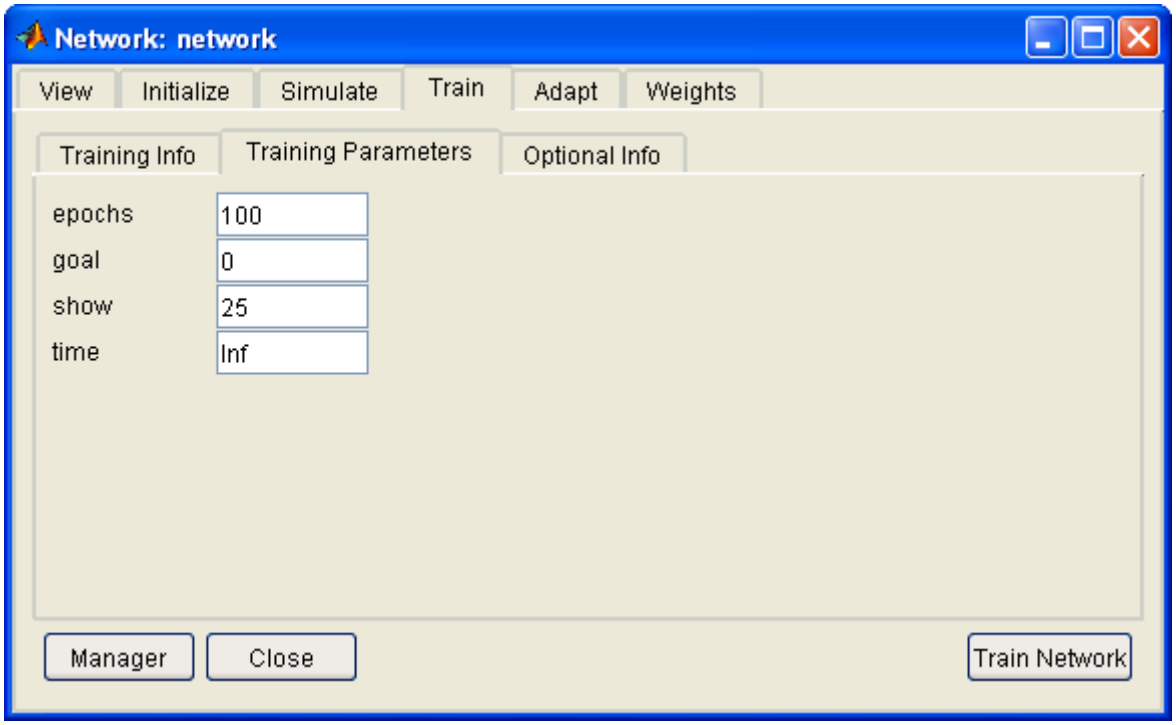


Рис. П2.7. Вкладка параметров обучения

Принимая во внимание тот факт, что для задач с линейно отделимыми множествами (а наша задача относится к этому классу) всегда существует точное решение, установим порог достижения цели, равный нулю. Значения остальных параметров оставим по умолчанию. Заметим только, что поле времени обучения содержит запись Inf, которая определяет бесконечный интервал времени (от английского Infinite – бесконечный).

Следующая вкладка «Optional Info» (Необязательная информация) показана на рис. П2.8.

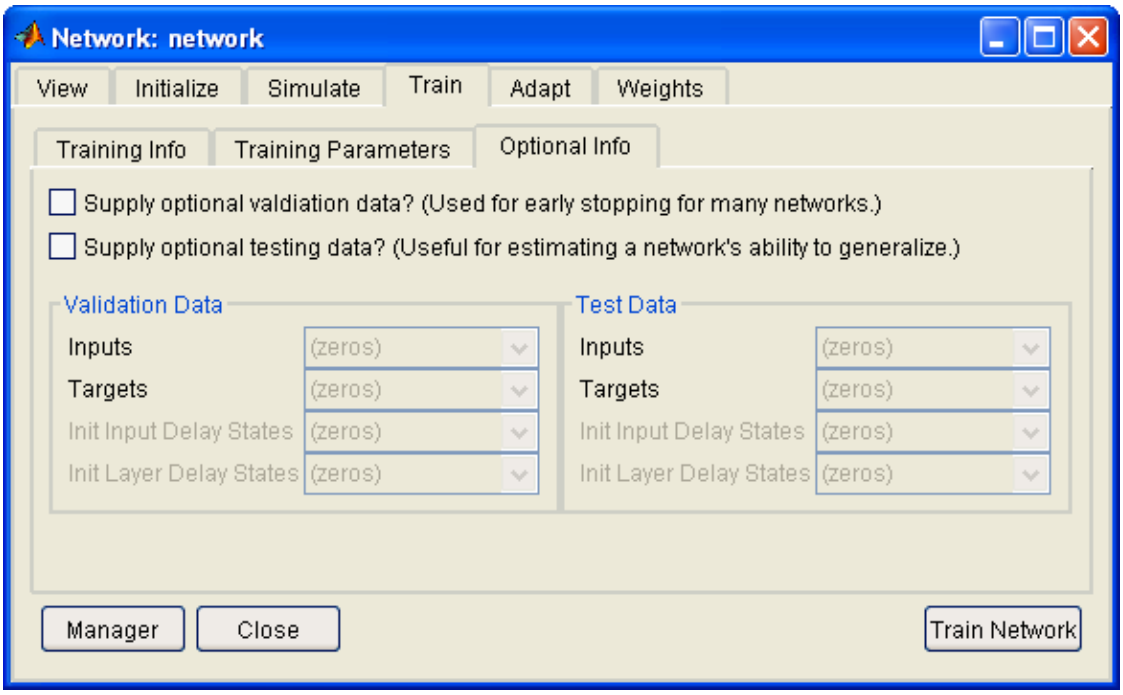


Рис. П2.8. Вкладка необязательной информации

Рассмотрим вкладку обучения («Train»). Чтобы начать обучение, нужно нажать кнопку «Train Network» (Обучить сеть). После этого, если в текущий момент сеть не удовлетворяет ни одному из условий, указанных в разделе параметров обучения («Training Parameters»), появится окно, иллюстрирующее динамику целевой функции – кривую обучения. В нашем случае график может выглядеть так, как показано на рис. П2.9. Кнопкой «Stop Training» (Остановить обучение) можно прекратить этот процесс. Из рисунка видно, что обучение было остановлено, когда функция цели достигла установленной величины (*goal* = 0).

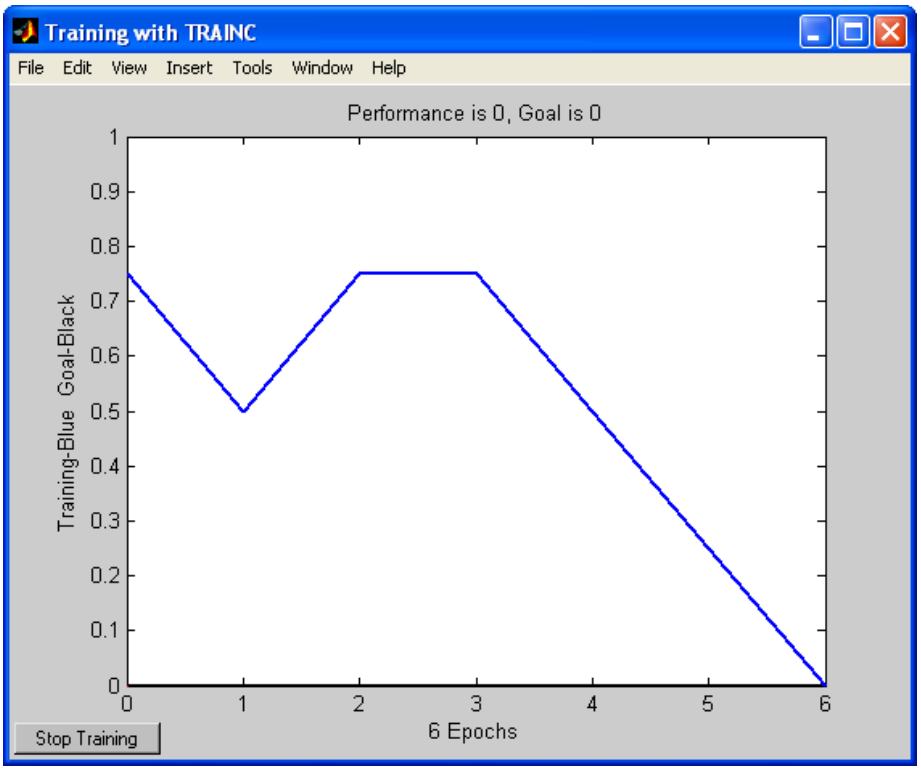


Рис. П2.9. Кривая обучения

Следует отметить, что для персептронов, имеющих функцию активации с жёстким ограничением, ошибка рассчитывается как разница между целью и полученным выходом.

Итак, алгоритм обучения нашёл точное решение задачи. В методических целях убедимся в правильности решения задачи путём прогона обученной сети. Для этого необходимо открыть вкладку «Simulate» (Прогон) и выбрать в выпадающем списке «Inputs» (Входы) заготовленные данные. В данной задаче естественно использовать тот же набор данных, что и при обучении *data*1. При желании можно установить флажок Supply «Targets» (Задать цели). Тогда в результате прогона дополнительно будут рассчитаны значения ошибки. Нажатие кнопки «Simulate Network» (Прогон сети) запишет результаты прогона в переменную, имя которой указано в поле «Outputs» (Выходы). Теперь можно вернуться в основное окно «NNTool» и, выделив мышью выходную переменную *network1*, нажать кнопку «View» (Просмотр). Содержимое окна просмотра совпадает со значением вектора целей – сеть работает правильно.

Следует заметить, что сеть создаётся инициализированной, т.е. значения весов и смещений задаются определённым образом. Перед

каждым следующим опытом обучения обычно начальные условия обновляются, для чего на вкладке «Initialize» (Инициализация) предусмотрена функция инициализации. Так, если требуется провести несколько независимых опытов обучения, инициализация весов и смещений перед каждым из них осуществляется нажатием кнопки Initialize «Weights» ( ). Инициализировать веса.

Вернёмся к вкладке «Optional Info» (Необязательная информация) (см. рис. П2.8). Чтобы понять, какой цели служат представленные здесь параметры, необходимо обсудить два понятия: переобучение и обобщение.

При выборе нейронной сети для решения конкретной задачи трудно предсказать её порядок. Если выбрать неоправданно большой порядок, сеть может оказаться слишком гибкой и может представить простую зависимость сложным образом. Это явление называется переобучением. В случае сети с недостаточным количеством нейронов, напротив, необходимый уровень ошибки никогда не будет достигнут. Здесь налицо чрезмерное обобщение.

Для предупреждения переобучения применяется следующая техника. Данные делятся на два множества: обучающее («Training Data») и контрольное («Validation Data»). Контрольное множество в обучении не используется. В начале работы ошибки сети на обучающем и контрольном множествах будут одинаковыми. По мере того как сеть обучается, ошибка обучения убывает, и пока обучение уменьшает действительную функцию ошибки, ошибка на контрольном множестве также будет убывать. Если же контрольная ошибка перестала убывать или даже стала расти, это указывает на то, что обучение следует закончить. Остановка на этом этапе называется ранней остановкой («Early stopping»).

Таким образом, необходимо провести серию экспериментов с различными сетями, прежде чем будет получена подходящая. При этом, чтобы не быть введённым в заблуждение локальными минимумами функции ошибки, следует несколько раз обучать каждую сеть.

Если в результате последовательных шагов обучения и контроля ошибка остаётся недопустимо большой, целесообразно изменить модель нейронной сети (например, усложнить сеть, увеличив число нейронов, или использовать сеть другого вида). В такой ситуации рекомендуется применять ещё одно множество – тестовое множество

наблюдений («Test Data»), которое представляет собой независимую выборку из входных данных. Итоговая модель тестируется на этом множестве, что даёт дополнительную возможность убедиться в достоверности полученных результатов. Очевидно, чтобы сыграть свою роль, тестовое множество должно быть использовано только один раз. Если его использовать для корректировки сети, оно фактически превратится в контрольное множество.

Установка верхнего флажка (см. рис. П2.8) позволит задать контрольное множество и соответствующий вектор целей (возможно, тот же, что при обучении). Установка нижнего позволяет задать тестовое множество и вектор целей для него.

Обучение сети можно проводить в разных режимах. В связи с этим в «NNTool» предусмотрено две вкладки, представляющие обучающие функции: рассмотренная ранее вкладка «Train» и «Adapt» (Адаптация). «Adapt» вмещает вкладку «Информация адаптации» («Adaption Info»), на которой содержатся поля, схожие по своему назначению с полями вкладки «Training Info» и выполняющие те же функции, и вкладку «Параметры адаптации» («Adaption Parameters»). Последняя содержит единственное поле *passes* (проходы). Значение, указанное в этом поле, определяет, сколько раз все входные векторы будут представлены в сети в процессе обучения.

Приложение 3

СПИСОК ОСНОВНЫХ ФУНКЦИЙ NEURAL NETWORK TOOLBOX

Функции сгруппированы по назначению.

Функции для анализа:

• *rrsurf* – поверхность ошибки нейрона с единственным входом;

• *maxlinlr* – максимальная скорость обучения для линейного нейрона.

Функции отклонения:

• *boxdist* – расстояние между двумя векторами;

• *dist* – евклидова весовая функция отклонения;

• *linkdist* – связанная фунция отклонения;

• *mandist* – весовая функция отклонения Манхеттена.

Функции графического интерфейса:

• *nntool* – вызов графического интерфейса пользователя.

Функции инициализации слоя:

• *initnw* – функция инициализации Нгуен-Видроу (Nguyen-Widrow);

• *initwb* – функция инициализации по весам и смещениям.

Функции обучения:

• *learncon* – обучающая функция смещений;

• *learngd* – обучающая функция градиентного спуска;

• *learngdm* – обучающая функция градиентного спуска с учетом моментов;

• *learnh* – обучающая функция Хэбба;

• *learnhd* – обучающая функция Хэбба с учетом затухания;

• *learnis* – обучающая функция instar;

• *learnk* – обучающая функция Кохонена;

• *learnlv1* – LVQ1 обучающая функция;

• *learnlv2* – LVQ2 обучающая функция;

• *learnos* – outstar обучающая функция;

• *learnp* – обучающая функция смещений и весов перцептрона;

• *learnpn* – обучающая функция нормализованных смещений и весов перцептрона;

• *learnsom* – обучающая функция самоорганизующейся карты весов;

• *learnwh* – правило обучения Уидроу-Хоффа (Widrow-Hoff).

Линейные функции поиска:

• *srchbac* – одномерная минимизация с использованием поиска с возвратом;

• *srchbre* – одномерная локализация интервала с использованием метода Брента (Brent);

• *srchcha* – одномерная минимизация с использованием метода Караламбуса (Charalambous);

• *srchgol* – одномерная минимизация с использованием золотого сечения;

• *srchhyb* – одномерная минимизация с использованием гибридного бисекционного поиска.

Функции вычисления производных от входов сети:

• *dnetprod* – вычисление производной от входов сети с перемножением входов;

• *dnetsum* – вычисление производной от входов сети с суммированием входов.

Входные функции сети:

• *netprod* – функция произведения входов;

• *netsum* – функция суммирования входов.

Функции инициализации сети:

• *initlay* – функция послойной инициализации сети.

Функции использования сети:

• *adapt* – разрешает адаптацию сети;

• *disp* – отображает свойства нейронной сети;

• *display* – отображает имена переменных и свойства сети;

• *init* – инициализация нейронной сети;

• *sim* – моделирование нейронной сети;

• *train* – тренировка нейронной сети.

Функции создания новой сети:

• *network* – создание нейронной сети пользователя;

• *newc* – создание конкурентного слоя;

• *newcf* – создание каскадной направленной сети;

• *newelm* – создание сети обратного распространения Элмана (Elman);

• *newff* – создание однонаправленной сети;

• *newfftd* – создание однонаправленной сети с входными задержками;

• *newgrnn* – создание обобщенной регрессонной нейронной сети;

• *newhop* – создание рекуррентной сети Хопфилда;

• *newlin* – создание линейного слоя;

• *newlind* – конструирование линейного слоя;

• *newlvq* – создание квантованной сети;

• *newp* – создание перцептрона;

• *newpnn* – конструирование вероятностной нейронной сети;

• *newrb* – конструирование сети с радиальным базисом;

• *newrbe* – конструирование точной сети с радиальными базисными функциями;

• newsom – создание самоорганизующейся карты.

Функции производных функционирования:

• *dmae* – средняя абсолютная ошибка вычисления производной;

• *dmse* – средне-квадратичная ошибка производной;

• *dmsereg* – средне-квадратичная ошибка производной w/reg;

• *dsse* – суммарная квадратичная ошибка производной.

Функции выполнения:

• *mae* – средняя абсолютная ошибка;

• *mse* – средне-квадратичная ошибка;

• *msereg* – средне-квадратичная ошибка w/reg;

• *sse* – суммарная квадратичная ошибка.

Функции графики:

• *hintonw* – график Хинтона для матрицы весов;

• *hintonwb* – график Хинтона для матрицы весов и векторов смещений;

• *plotbr* – график функционирования сети при регулярной тренировке (Bayesian);

• *plotep* – изображение положений весов и смещений на поверхности ошибки;

• *plotes* – изображение поверхности ошибок единичного входного нейрона;

• *plotpc* – изображение линии классификации в векторном пространстве перцептрона;

• *plotperf* – графическое представление функционирования сети;

• *plotpv* – графическое представление входных целевых векторов;

• *plotsom* – графическое представление самоорганизующейся карты;

• *plotv* – графическое представление векторов в виде линий, выходящих из начала координат;

• *plotvec* – графическое представление векторов различными цветами.

Функции предварительной и постобработки:

• *postmnmx* – ненормализованные данные, которые были нормализованы посредством prenmmx;

• *postreg* – линейный регрессионный анализ выходов сети по отношению к целевым значениям обучающего массива;

• *poststd* – ненормированные данные, которые были нормированы с помощью функции prestd;

• *premnmx* – нормирование данных в диапазоне от –1 до +1;

• *prepca* – анализ главных компонент для входных данных;

• *prestd* – нормирование данных к единичному стандартному отклонению и нулевому среднему;

• *tramnmx* – преобразование данных с предварительно вычисленными минимумом и максимумом;

• *trapca* – преобразование данных с использованием PCA матрицы, вычисленной с помощью функции prepca;

• *trastd* – преобразование данных с использованием предварительно *вычисленных* значений стандартного отклонения и среднего.

Функции поддержки «Simulink»:

• *gensim* – генерация блока «Simulink» для моделирования нейронной сети.

Топологические функции:

• *gridtop* – топологическая функция в виде сеточного слоя;

• *hextop* – топологическая функция в виде гексагонального слоя;

• *randtop* – топологическая функция в виде случайного слоя.

Функции тренировки:

• *trainb* – пакетная тренировка с использованием правил обучения для весов и смещений;

• *trainbfg* – тренировка сети с использованием квази–Ньютоновского метода BFGS;

• *trainbr* – регуляризация Bayesian;

• *traiinc* – использование приращений циклического порядка;

• *traincgb* – метод связанных градиентов Пауэлла-Била (Powell-Beale);

• *traincgf* – метод связанных градиентов Флетчера-Пауэлла (Fletcher-Powell);

• *traincgp* – метод связанных градиентов Полака-Рибира (Polak-Ribiere);

• *traingd* – метод градиентного спуска;

• *traingda* – метод градиентного спуска с адаптивным обучением;

• *traingdm* – метод градиентного спуска с учетом моментов;

• *traingdx* – метод градиентного спуска с учетом моментов и с адаптивным обучением;

• *trainlm* – метод Левенберга-Маркара (Levenberg-Marquardt);

• *trainoss* – одноступенчатый метод секущих;

• *trainr* – метод случайных приращений;

• *trainrp* – алгоритм упругого обратного распространения;

• *trains* – метод последовательных приращений;

• *trainscg* – метод шкалированных связанных градиентов.

Производные функций активации:

• *dhardlim* – производная ступенчатой функции активации;

• *dhardlms* – производная симметричной ступенчатой функции активации;

• *dlogsig* – производная сигмоидной (логистической) функции активации;

• *dposlin* – производная положительной линейной функции активации;

• *dpurelin* – приозводная линейной функции активации;

• *dradbas* – производная радиальной базисной функции активации;

• *dsatlin* – производная насыщающейся линейной функции активации;

• *dsatlins* – производная симметричной насыщающейся функции активации;

• *dtansig* – производная функции активации гиперболический тангенс;

• *dtribas* – производная треугольной функции активации.

Функции активации:

• *compet* – конкурирующая функция активации;

• *hardlim* – ступенчатая функция активации;

• *hardlims* – ступенчатая симметричная функция активации;

• *logsig* – сигмоидная (логистическая) функция активации;

• *poslin* – положительная линейная функция активации;

• *purelin* – линейная функция активации;

• *radbas* – радиальная базисная функция активации;

• *satlin* – насыщающаяся линейная функция активации;

• *satlins* – симметричная насыщающаяся линейная функция активации;

• *softmax* – функция активации, уменьшающая диапазон входных значений;

• *tansig* – функция активации гиперболический тангенс;

• *tribas* – треугольная функция активации.

Полезные функции:

• *calca* – вычисляет выходы сети и другие сигналы;

• *calca1* – вычисляет сигналы сети для одного шага по времени;

• *calce* – вычисляет ошибки слоев;

• *calce1* – вычисляет ошибки слоев для одного шага по времени;

• *calcgx* – вычисляет градиент весов и смещений как единственный вектор;

• *calcjejj* – вычисляет Якобиан;

• *calcjx* – вычисляет Якобиан весов и смещений как одну матрицу;

• *calcpd* – вычисляет задержанные входы сети;

• *calcperf* – вычисление выходов сети, сигналов и функционирования;

• *formx* – формирует один вектор из весов и смещений;

• *getx* – возвращает все веса и смещения сети как один вектор;

• *setx* – устанавливает все веса и смещения сети в виде одного вектора.

Векторные функции:

• *cell2mat* – объединяет массив элементов матриц в одну матрицу;

• *combvec* – создает все комбинации векторов;

• *con2seq* – преобразует сходящиеся векторы в последовательные векторы;

• *concur* – создает сходящиеся векторы смещений;

• *ind2vec* – преобразование индексов в векторы;

• *mat2cell* – разбиение матрицы на массив элементов матриц;

• *minmax* – вычисляет минимальные и максимальные значения строк матрицы;

• *normc* – нормирует столбцы матрицы;

• *normr* – нормирует строки матрицы;

• *pnormc* – псевдо-нормировка столбцов матрицы;

• *quant* – дискретизация величины;

• *seq2con* – преобразование последовательных векторов в сходящиеся векторы;

• *sumsqr* – сумма квадратов элементов матрицы;

• *vec2ind* – преобразование векторов в индексы.

Функции инициализации весов и смещений:

• *initcon* – «сознательная» функция инициализации;

• *initzero* – инициализация с установкой нулевых значений весов и смещений;

• *midpoint* – инициализация с установкой средних значений весов;

• *randnc* – инициализация с установкой нормализованных значений столбцов весовых матриц;

• *randnr* – инициализация с установкой нормализованных значений строк весовых функций;

• *rands* – инициализация с установкой симметричных случайных значений весов и смещений;

• *revert* – возвращение весам и смещениям значений, соответствующих предыдущей инициализации.

Функции весовых производных:

• *ddotprod* – производная скалярного произведения.

Весовые функции:

• *dist* – Евклидово расстояние;

• *dotprod* – весовая функция в виде скалярного произведения;

• *mandist* – весовая функция – расстояние Манхеттена;

• *negdist* – весовая функция – отрицательное расстояние;

• *normprod* – нормированное скалярное произведение.

Приложение 4

РАБОТА С FIS-РЕДАКТОРОМ

1. FIS-редактор

FIS-редактор предназначен для создания, сохранения, загрузки и вывода на печать систем нечеткого логического вывода, а также для редактирования следующих свойств:

• тип системы;

• наименование системы;

• количество входных и выходных переменных;

• наименование входных и выходных переменных;

• параметры нечеткого логического вывода.

Загрузка FIS-редактора происходит с помощью команды *fuzzy*. В результате появляется интерактивное графическое окно. На этом же рисунке также указаны функциональные назначения основных полей графичекого окна. В нижней части графического окна FIS-редактора расположены кнопки «Help» и «Close», которые позволяют вызвать окно справки и закрыть редактор соответственно.

FIS-редактор содержит 8 меню. Это три общесистемных меню («File», «Edit», «View») и пять меню для выбора параметров нечеткого логического вывода («And Method», «Or Method», «Implication», «Aggregation», «Defuzzification»).

Меню «File»

Это общее меню для всех GUI-модулей, используемых с системами нечеткого логического вывода. Общий вид меню показан на рис. П4.1.

Рис. П4.1. Меню «File»

С помощью команды «New FIS…» пользователь имеет возможность создать новую систему нечеткого логического вывода. При выборе этой команды появятся две альтернативы: «Mamdani» и «Sugeno», которые определяют тип создаваемой системы. Создать систему типа «Mamdani» можно также нажатием «Ctrl+N».

С помощью команды «Import» пользователь имеет возможность загрузить ранее созданную систему нечеткого логического вывода. При выборе этой команды появятся две альтернативы «From Workspace…» и «From disk», которые позволяют загрузить систему нечеткого логического вывода из рабочей области MATLAB и с диска соответственно. При выборе команды «From Workspace…» появится диалоговое окно, в котором необходимо указать идентификатор системы нечеткого логического вывода, находящейся в рабочей области MATLAB. При выборе команды «From disk» появится диалоговое окно (рис. П4.2), в котором необходимо указать имя файла системы нечеткого логического вывода. Файлы систем нечеткого логического вывода имеют расширение **.fis* . Загрузить систему нечеткого логического вывода с диска можно также нажатием «Ctrl+N» или командой *fuzzyFIS_name*, где *FIS_name* – имя файла системы нечеткого логического вывода.

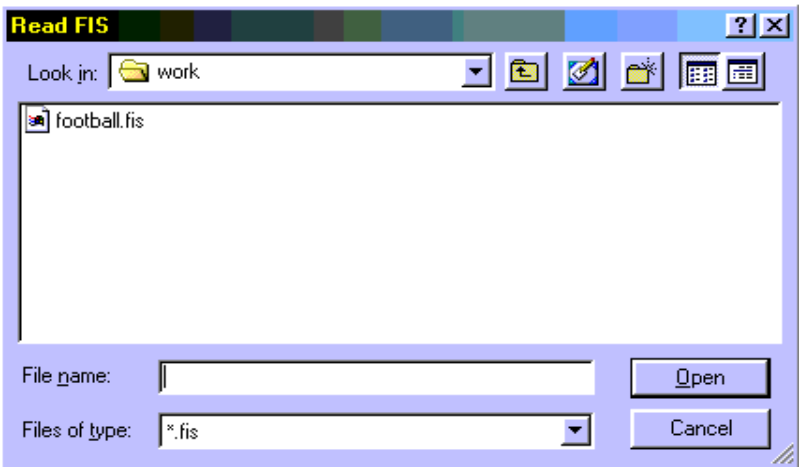


Рис. П4.2. Окно загрузки системы нечеткого логического вывода с диска

При выборе команды «Export» появятся две альтернативы «To Workspace…» и «To disk», которые позволяют скопировать систему нечеткого логического вывода в рабочую область MATLAB и на диск соответственно. При выборе команды «To Workspace…» появится диалоговое окно, в котором необходимо указать идентификатор системы нечеткого логического вывода, под которым она будет сохранена в рабочей области MATLAB. При выборе команды «To disk» появится диалоговое окно, в котором необходимо указать имя файла системы нечеткого логического вывода. Скопировать систему нечеткого логического вывода в рабочую область и на диск можно также нажатием «Ctrl+T» и «Ctrl+S» соответственно.

Команда «Print» позволяет вывести на принтер копию графического окна. Печать возможна также по нажатию «Ctrl+P».

Команда «Close» закрывает графическое окно. Закрытия графического окна происходит по нажатию «Ctrl+W» или однократного щелчка левой кнопки мыши по кнопке «Close».

Меню «Edit»

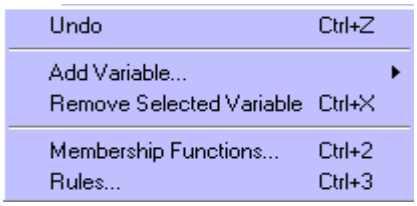


Рис. П4.3. Меню «Edit»

Общий вид меню приведен на рис. П4.3.

Команда «Undo» отменяет ранее совершенное действие. Выполняется также по нажатию «Ctrl+Z».

Команда «Add Variable…» позволяет добавить в систему нечеткого логического вывода еще одну переменную. При выборе этой

команды появятся две альтернативы «Input» и «Output», которые позволяют добавить входную и выходную переменную соответственно.

Команда «Remove Selected Variable» удаляет текущую переменную из системы. Признаком текущей переменной является красная окантовка ее прямоугольника. Назначение текущей переменной происходит с помощью однократного щелчка левой кнопки мыши по ее прямоугольнику. Удалить текущую переменную можно также с помощью нажатия «Ctrl+X».

Команда «Membership Function…» открывает редактор функций принадлежностей. Эта команда может быть также выполнена нажатием «Ctrl+2».

Команда «Rules…» открывает редактор базы знаний. Эта команда может быть также выполнена нажатием «Ctrl+3».

Меню «View»

Это общее меню для всех GUI-модулей, используемых с системами нечеткого логического вывода. Общий вид меню показан на рис. П4.4. Это меню позволяет открыть окно визуализации нечеткого логического вывода (команда «Rules» или нажатие клавиш «Ctrl+5») и окно вывода поверхности «входы-выход», соответствующей системе нечеткого логического вывода (команда «Surface» или нажатие клавиш «Ctrl+6»).

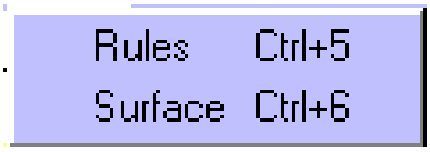


Рис. П4.4. Меню«View»

Меню«And Method»

Это меню позволяет установить следующие реализации логической операции «И»:

- *min* – минимум;
- *prod* – умножение.

Пользователь также имеет возможность установить собственную реализацию операции «И». Для этого необходимо выбрать команду «Custom…» и в появившемся графическом окне напечатать имя функции, реализующей эту операцию.

Меню «Or Method»

Это меню позволяет установить следующие реализации логической операции «ИЛИ»:

- *max* – умножение;
- *probor* – вероятностное «ИЛИ».

Пользователь также имеет возможность установить собственную реализацию операции «ИЛИ». Для этого необходимо выбрать команду «Custom…» и в появившемся графическом окне напечатать имя функции, реализующей эту операцию.

Меню «Implication»

Это меню позволяет установить следующие реализации импликации:

- *min* – минимум*;*
- *prod* – умножение.

Пользователь также имеет возможность установить собственную реализацию импликации. Для этого необходимо выбрать команду «Custom…» и в появившемся графическом окне напечатать имя функции, реализующей эту операцию.

Меню «Aggregation»

Это меню позволяет установить следующие реализации операции объединения функций принадлежности выходной переменной:

- *max* – максимум*;*
- *sum* – сумма*;*
- *probor* – вероятностное «ИЛИ».

Пользователь также имеет возможность установить собственную реализацию этой операции. Для этого необходимо выбрать команду «Custom…» и в появившемся графическом окне напечатать имя функции, реализующей эту операцию.

Меню «Defuzzification»

Это меню позволяет выбрать метод дефаззификации. Для систем типа Мамдани запрограммированы следующие методы:

- *centroid* – центр тяжести*;*

- *bisector* – медиана*;*
- *lom* – наибольший из максимумов*;*
- *som* – наименьший из максимумов*;*
- *mom* – среднее из максимумов.

Для систем типа «Сугэно» запрограммированы следующие методы:

- *wtaver* – взвешенное среднее*;*
- *wtsum* – взвешенная сумма.

Пользователь также имеет возможность установить собственный метод деффазификации. Для этого необходимо выбрать команду «Custom…» и в появившемся графическом окне напечатать имя функции, реализующей эту операцию.

2. Редактор функций принадлежности

Редактор функций принадлежности («Membership Function Editor») предназначен для задания следующей информации о термножествах входных и выходных переменных:

– количество термов;

– наименования термов;

– тип и параметры функций принадлежности, которые необходимы для представления лингвистических термов в виде нечетких множеств.

Редактор функций принадлежности может быть вызван из любого GUI-модуля, используемого с системами нечеткого логического вывода, командой «Membership Functions…» меню «Edit» или нажатием клавиш «Ctrl+2». В FIS-редакторе открыть редактор функций принадлежности можно также двойным щелчком левой кнопкой мыши по полю входной или выходной переменных. Общий вид редактора функций принадлежности с указанием функционального назначения основных полей графического окна приведен на рис. П4.5. В нижней части графического окна расположены кнопки «Help» и «Close», которые позволяют вызвать окно справки и закрыть редактор соответственно.

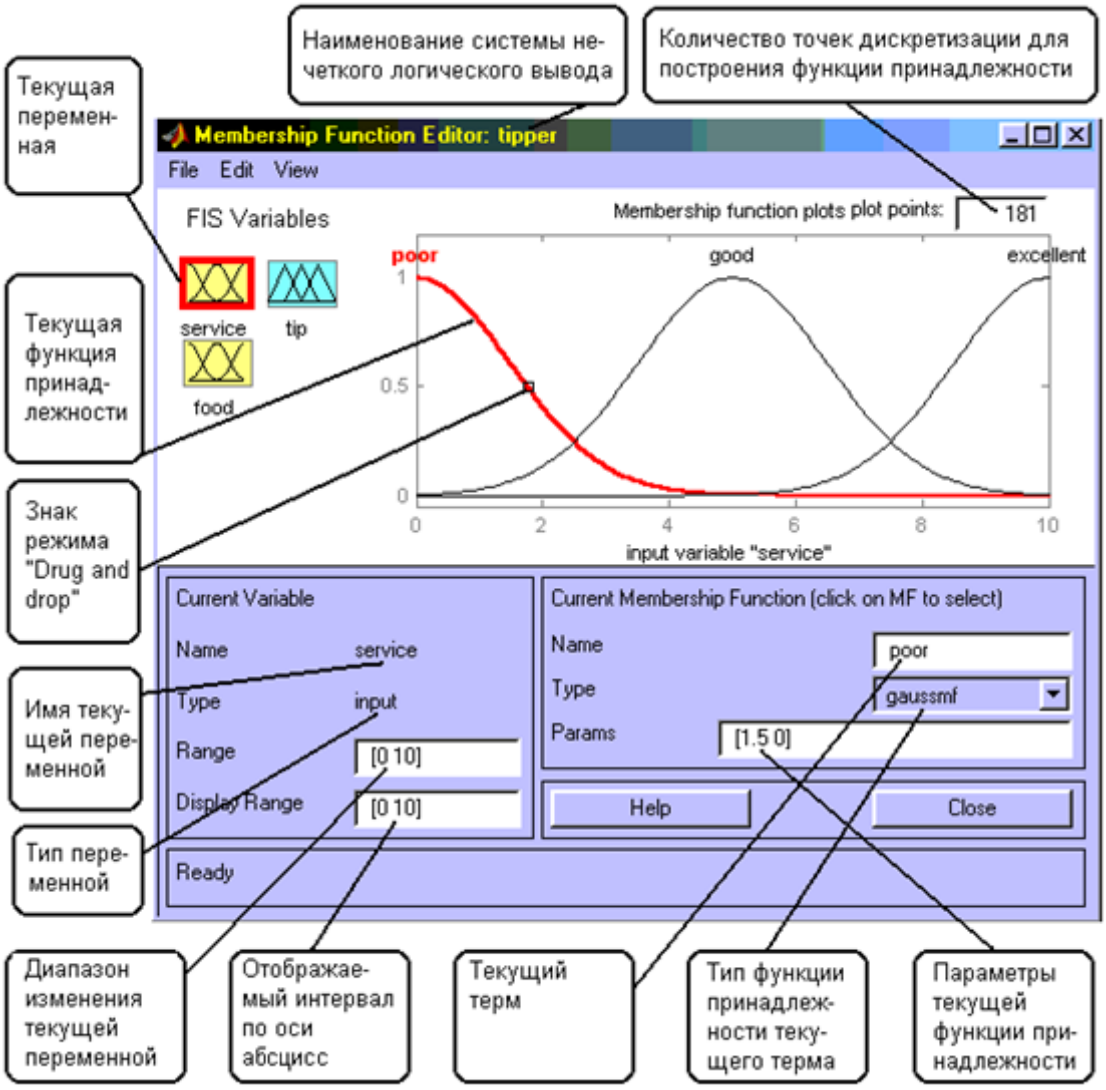


Рис. П4.5. Редактор функций принадлежности

Редактор функций принадлежности содержит четыре меню – «File», «Edit», «View», «Type» и четыре окна ввода информации – «Range», «Display Range», «Name» и «Params». Эти четыре окно предназначены для задания диапазона изменения текущей переменной, диапазона вывода функций принадлежности, наименования текущего лингвистического терма и параметров его функции принадлежности соответственно. Параметры функции принадлежности можно подбирать и в графическом режиме путем изменения формы функции принадлежности с помощью технологии «Drug and drop». Для этого необходимо позиционировать курсор мыши на знаке режима «Drug and drop» (см. рис. П4.5), нажать на левую кнопку мыши и

не отпуская ее изменять форму функции принадлежности. Параметры функции принадлежности будут пересчитываться автоматически.

Меню «File» и «View» одинаковые для всех GUI-модулей, используемых с системами нечеткого логического вывода.

Меню «Edit»

Общий вид меню приведен на рис. П4.6.

Рис. П4.6. Меню «Edit»

Команда Undo отменяет ранее совершенное действие. Выполняется также по нажатию «Ctrl+Z».

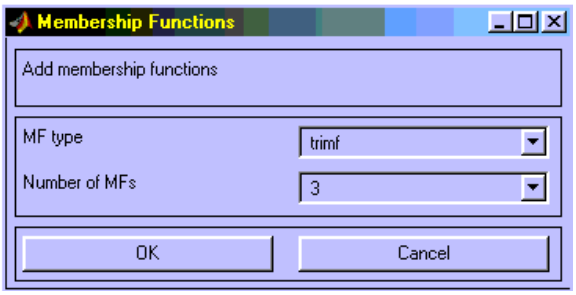


Рис. П4.7. Выбор количества термов
и типа функций принадлежности

Команда «AddMFs…» позволяет добавить термы в терммножество, используемое для лингвистической оценки текущей переменной. При выборе этой команды появится диалоговое окно (рис. П4.7), в котором необходимо выбрать тип функции принадлежности и количество термов. Значения параметров функций принадлежности будут установлены автоматически таким образом, чтобы равномерно покрыть область определения переменной, заданной в окне «Range».

При изменении области определения в окне «Range» параметры функций принадлежности будут промасштабированы.

Команда «Add Custom MF…» позволяет добавить один лингвистический терм, функция принадлежности которого отличается от встроенных. После выбора этой команды появится графическое окно (рис. П4.8), в котором необходимо напечатать лингвистически терм (поле «MF name»), имя функции принадлежности (поле «M-File function name») и параметры функции принадлежности (поле «Parameter list»).

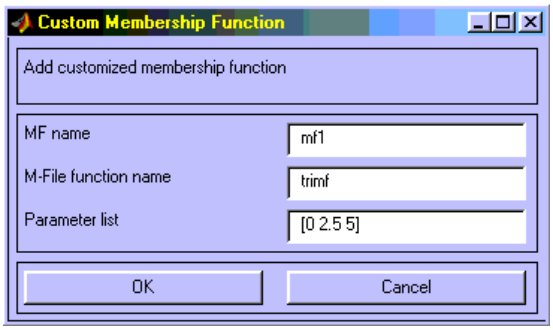


Рис. П4.8. Задание лингвистического терма с невстроенной функцией принадлежности

Команда «Remove Selected MF» удаляет текущий терм из терм-множества текущей переменной. Признаком текущей переменной является красная окантовка ее прямоугольника. Признаком текущего терма является красный цвет его функции принадлежности. Для выбора текущего терма необходимо провести позиционирования курсора мыши на графике функции принадлежности и сделать щелчок левой кнопкой мыши.

Команда «Remove All MFs» удаляет все термы из терм-множества текущей переменной.

Команда «FIS Properties…» открывает FIS-редактор. Эта команда может быть также выполнена нажатием «Ctrl+1».

Команда «Rules…» открывает редактор базы знаний. Эта команда может быть также выполнена нажатием «Ctrl+3».

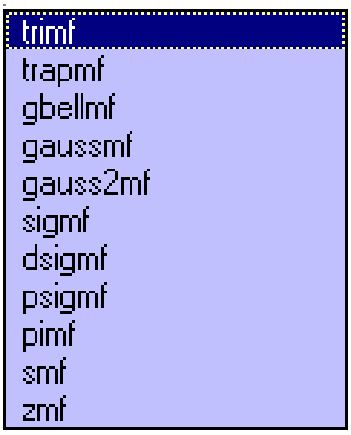


Рис. П4.9. Меню «Type»

Меню «Type»

Это меню позволяет установить тип функций принадлежности термов, используемых для лингвистической оценки текущей переменной. На рис. П4.9 приведено меню «Type», в котором указаны возможные типы функций принадлежности.

3. Редактор базы знаний

Редактор базы знаний («Rule Editor») предназначен для формирования и модификации нечетких правил. Редактор базы знаний может быть вызван из любого GUI-модуля, используемого с системами нечеткого логического вывода, командой «Rules…» меню «Edit» или нажатием клавиш «Ctrl+3». В FIS-редакторе открыть редактор базы знаний можно также двойным щелчком левой кнопкой мыши по прямоугольнику с названием системы нечеткого логического вывода, расположенного в центре графического окна.

Общий вид редактора базы знаний с указанием функционального назначения основных полей графического окна приведен на рис. П4.10. В нижней части графического окна расположены кнопки «Help» и «Close», которые позволяют вызвать окно справки и закрыть редактор соответственно.

Редактор функций принадлежности содержит четыре системных меню «File», «Edit», «View», «Options», меню выбора термов входных и выходных переменных, поля установки логических операций «И», «ИЛИ», «НЕ» и весов правил, а также кнопки редактирования и просмотра правил.

Для ввода нового правила в базу знаний необходимо с помощью мыши выбрать соответствующую комбинацию лингвистических термов входных и выходных переменных, установить тип логической связки («И» или «ИЛИ») между переменными внутри правила, установить наличие или отсутствие логической операции «НЕ» для каждой лингвистической переменной, ввести значение весового коэффициента правила и нажать кнопку «Add Rule». По умолчанию установлены следующие параметры:

- логическая связка переменных внутри правила – «И»;
- логическая операция «НЕ» – отсутствует;
- значение весового коэффициента правила – 1.

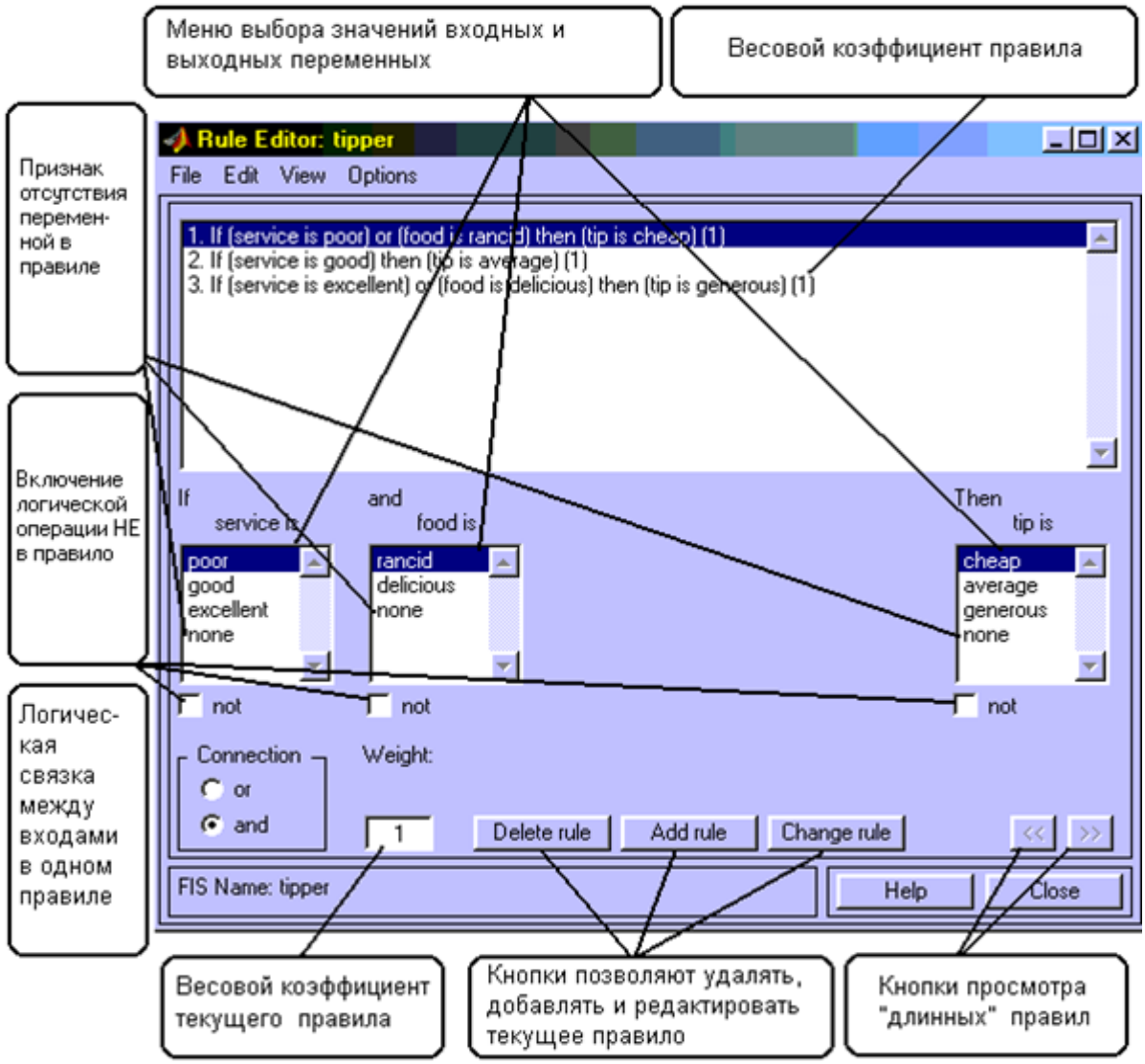


Рис. П4.10. Редактор базы знаний

Возможны случаи, когда истинность правила не изменяется при произвольном значении некоторой входной переменной, т.е. эта переменная не влияет на результат нечеткого логического вывода в данной области факторного пространства. Тогда в качестве лингвистического значения этой переменной необходимо установить none.

Для удаления правила из базы знаний необходимо сделать однократный щелчок левой кнопкой мыши по этому правилу и нажать кнопку «Delete Rule».

Для модификации правила необходимо сделать однократный щелчок левой кнопкой мыши по этому правилу, затем установить необходимые параметры правила и нажать кнопку «Edit Rule».

Меню «File» и «View» одинаковые для всех GUI-модулей, используемых с системами нечеткого логического вывода.

Меню «Edit»

Общий вид меню приведен на рис. П4.11.

Команда «Undo» отменяет ранее совершенное действие. Выполняется также по нажатию «Ctrl+Z».

Рис. П4.11. Меню «Edit»

Команда «FIS Properties…» открывает FIS-редактор. Эта команда может быть также выполнена нажатием «Ctrl+1».

Команда «Membership Function…» открывает редактор функций принадлежностей. Эта команда может быть также выполнена нажатием «Ctrl+2».

Меню «Options»

Это меню позволяет установить язык и формат правил базы знаний (рис. П4.12).

Рис. П4.12. Меню «Options»

При выборе команды «Language» появится список языков «English» (английский), «Deutsch» (немецкий), «Francais» (французский), из которого необходимо выбрать один.

При выборе команды «Format» появится список возможных форматов правил базы знаний: «Verbose» – лингвистический; «Symbolic» – логический; «Indexed» – индексированный. Различные форматы базы знаний демосистемы нечеткого логического вывода Tipper приведены на рис. П4.10 (формат правил «Verbose»), рис. П4.13 (формат правил «Symbolic») и рис. П4.14 (формат правил «Indexed»).

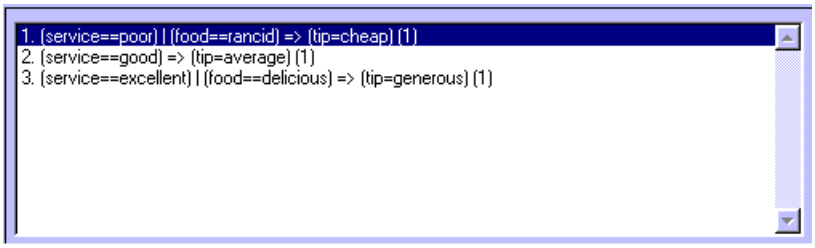


Рис. П4.13. База знаний в формате «Symbolic»

Рис. П4.14. База знаний в формате «Indexed»

4. Визуализация нечеткого логического вывода

Визуализация нечеткого логического вывода осуществляется с помощью GUI-модуля «Rule Viewer». Этот модуль позволяет проиллюстрировать ход логического вывода по каждому правилу, получение результирующего нечеткого множества и выполнение процедуры дефаззификации. «Rule Viewer» может быть вызван из любого «GUI-модуля», используемого с системами нечеткого логического вывода, командой «View rules…» меню «View» или нажатием клавиш «Ctrl+4». Вид «Rule Viewer» для системы логического вывода *tipper* с указанием функционального назначения основных полей графического окна приведен на рис. П4.15.

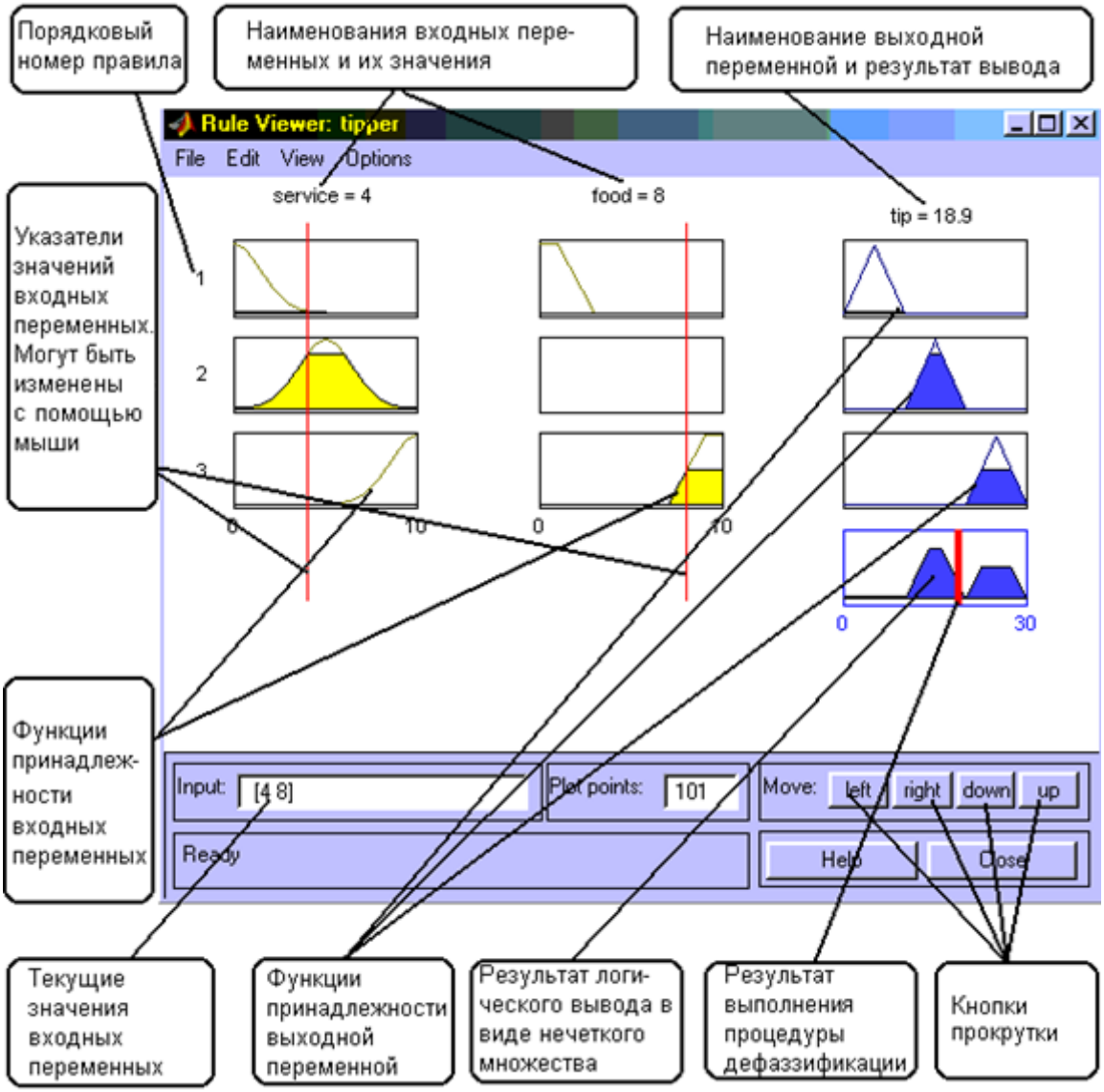


Рис. П4.15. Визуализация логического вывода для системы tipper с помощью «Rule Viewer»

«Rule Viewer» содержит четыре меню – «File», «Edit», «View», «Options», два поля ввода информации – «Input» и «Plot points» и кнопки прокрутки изображения влево-вправо («left-right»), вверх-вниз («up-down»). В нижней части графического окна расположены также кнопки «Help» и «Close», которые позволяют вызвать окно справки и закрыть редактор соответственно.

Каждое правило базы знаний представляется в виде последовательности горизонтально расположенных прямоугольников. При этом первые два прямоугольника (см. рис. П4.15) отображают функции принадлежностей термов посылки правила («ЕСЛИ» – часть правила), а последний третий прямоугольник соответствует функции принадлежности терма-следствия выходной переменной («ТО» – часть правила). Пустой прямоугольник в визуализации второго правила означает, что в этом правиле посылка по переменной *food* отсутствует (*foodisnone*). Желтая заливка графиков функций принадлежностей входных переменных указывает, насколько значения входов соответствуют термам данного правила. Для вывода правила в формате «Rule Editor» необходимо сделать однократный щелчок левой кнопки мыши по номеру соответствующего правила. В этом случае указанное правило будет выведено в нижней части графического окна.

Голубая заливка графика функции принадлежности выходной переменной представляет собой результат логического вывода в виде нечеткого множества по данному правилу. Результирующее нечеткое множество, соответствующее логическому выводу по всем правилам, показано в нижнем прямоугольнике последнего столбца графического окна. В этом же прямоугольнике красная вертикальная линия соответствует четкому значению логического вывода, полученного в результате дефаззификации.

Ввод значений входных переменных может осуществляться двумя способами: путем ввода численных значений в поле «Input»; с помощью мыши, путем перемещения линий-указателей красного цвета.

В последнем случае необходимо позиционировать курсор «мыши» на красной вертикальной линии, нажать на левую кнопку «мыши»и не отпуская ее переместить указатель на нужную позицию. Новое численное значения соответствующей входной переменной будет пересчитано автоматически и выведено в окно «Input».

В поле «Plot points» задается количество точек дискретизации для построения графиков функций принадлежности. Значение по умолчанию – 101.

Меню «File» и «View» одинаковые для всех GUI-модулей, используемых с системами нечеткого логического вывода.

Меню «Edit»

Общий вид меню приведен на рис. П4.16.

Рис. П4.16. Меню «Edit»

Команда «FIS Properties…» открывает FIS-редактор. Эта команда может быть также выполнена нажатием «Ctrl+1».

Команда «Membership Functions…» открывает редактор функций принадлежностей. Эта команда может быть также выполнена нажатием «Ctrl+2».

Команда «Rules…» открывает редактор базы знаний. Эта команда может быть также выполнена нажатием «Ctrl+3».

Меню «Options»

Меню «Options» содержит только одну команду «Format», которая позволяет установить один из следующих форматов вывода выбранного правило в нижней части графического окна:

- «Verbose» – лингвистический;
- «Symbolic» – логический;
- «Indexed» – индексированный.

5. Визуализация поверхности «вход – выход»

Визуализация поверхности «вход – выход» осуществляется с помощью GUI-модуля «Surface Viewer». Этот модуль позволяет вывести графическое изображение зависимости значения любой выходной переменной от произвольных двух (или одной) входных переменных. «Surface Viewer» может быть вызван из любого GUI-модуля, используемого с системами нечеткого логического вывода, командой «View surface…» меню «View» или нажатием клавиш «Ctrl+4». Общий вид модуля «Surface Viewer» с указанием функционального назначения основных полей графического окна приведен на рис. П4.17.

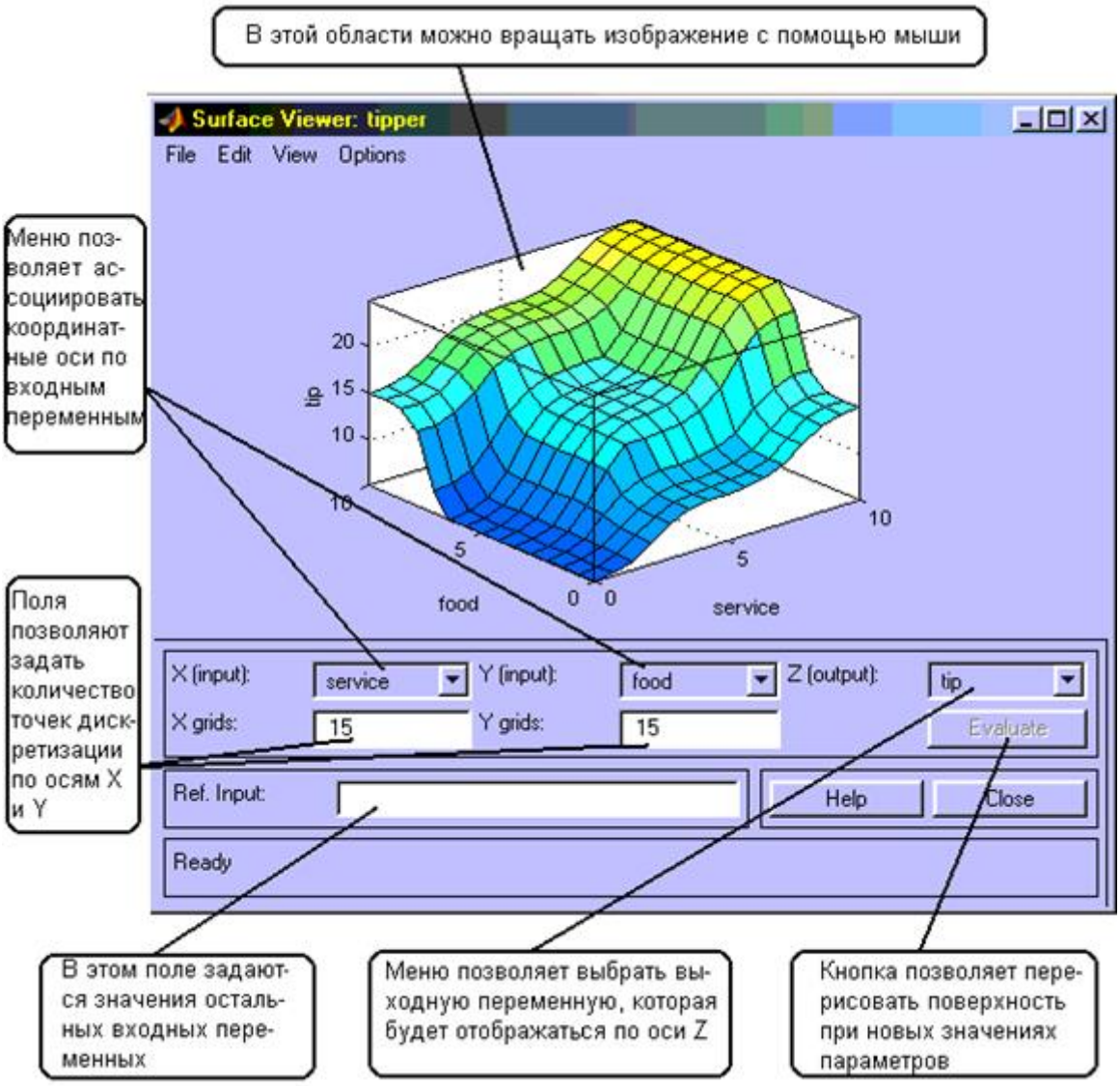


Рис. П4.17. Визуализация поверхности «входы – выход» для системы tipper с помощью «Surface Viewer»

«Surface Viewer» содержит четыре верхних системных меню – «File», «Edit», «View», «Options», три меню выбора координатных осей – «X (input)», «Y (input)», «Z (output)», три поля ввода ввода информации – «X girds», «Y girds», «Ref. Input» и кнопку «Evaluate» для построения поверхности при новых параметрах. В нижней части графического окна расположены также кнопки «Help» и «Close», которые позволяют вызвать окно справки и закрыть редактор соответственно.

«Surface Viewer» позволяет вращать поверхность «вход – выход» с помощью «мыши». Для этого необходимо позиционировать

курсор «мыши» на поверхности «вход-выход», нажать на левую кнопку мыши и не отпуская ее повернуть графическое изображение на требуемый угол.

Поля «X girds» и «Y girds» предназначены для задания количества точек дискретизации по осям *X* и *Y*, для построения поверхности «вход – выход». По умолчанию количество дискрет по каждой оси равно 15. Для изменения этого значения необходимо установить маркер на поле «X girds» («Y girds») и ввести новое значение.

Поле «Ref. Input» предназначено для задания значений входных переменных, кроме тех, которые ассоциированы с координатными осями. По умолчанию это значения середины интервалов изменения переменных. Для изменения этого значения необходимо установить маркер на поле «Ref. Input» и ввести новое значение.

Меню «File» и «View» одинаковые для всех GUI-модулей, используемых с системами нечеткого логического вывода.

Меню координатных осей

Меню «X (input)», «Y (input)», «Z (output)» позволяют поставить в соответствие осям координат входные и выходные переменные. При этом входные переменные могут отображаться только по осям *X* и *Y*, а выходные переменные только по оси *Z*. В «Surface Viewer» предусмотрена возможность построения однофакторных зависимостей «вход – выход». Для этого в меню второй координатной оси («X (input)» или «Y (input)») необходимо выбрать *none*.

Рис. П4.18. Меню «Options»

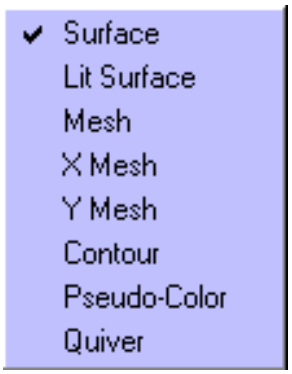


Рис. П4.19. Меню «Plot»

Меню «Edit»

Общий вид меню приведен на рис. П.4.16. Назначения команд меню описано выше.

Меню «Options»

Меню «Options» изображено на рис. П.4.18. Оно содержит команды «Plot», «Color Map» и «Always evaluate».

Команда «Plot» позволяет управлять форматом вывода поверхности «вход – выход». При выборе этой команды появляется меню (рис. П4.19), в котором необходимо выбрать формат вывода поверхности. На рис. П4.20 приведены поверхности «вход – выход» для системы *tipper* для всех поддерживаемых форматов.

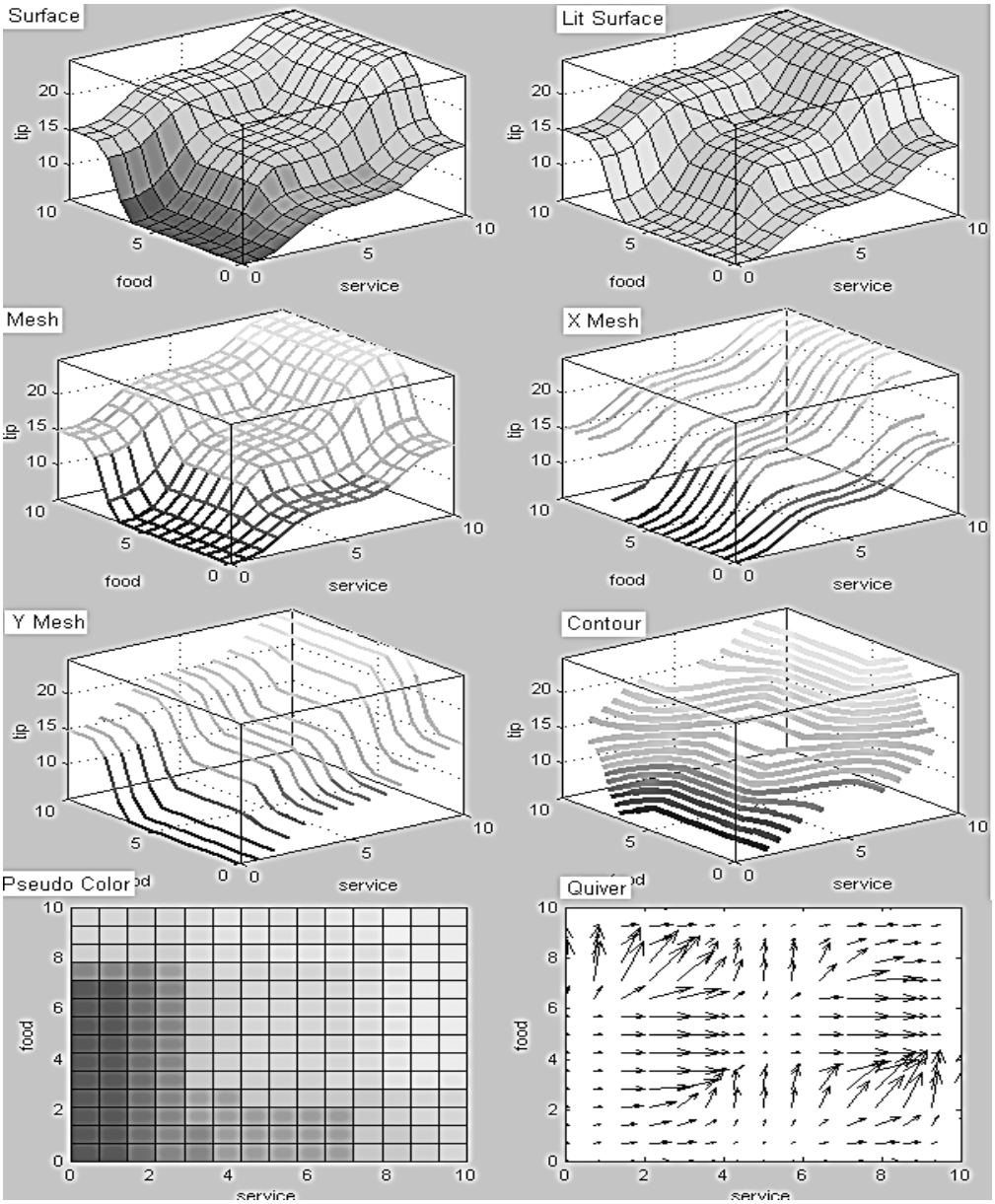


Рис. П4.20. Примеры форматов поверхности «вход – выход»

Команда «Color Map» позволяет управлять палитрой цветов при выводе поверхности «вход – выход». При выборе этой команды появляется меню, в котором необходимо выбрать одну из палитр:

• *default* – использовать палитру, установленную по умолчанию;

• *blue* – холодная сине-голубая палитра;

• *hot* – теплая палитра, состоящая из черного, красного, желтого и белого цветов;

• *HSV* – палитра насыщенных цветов: красный, желтый, зеленый, циан, голубой, маджента.

Команда «Always evaluate» позволяет установить или отменить режим автоматического, т.е. без нажатия кнопки «Evaluate», перерисовывания поверхности «вход – выход» при любом изменении параметров.

Приложение 5

СИСТЕМА КОМАНД ДЛЯ ПРОГРАММИРОВАНИЯ ФУНКЦИЙ НЕЧЕТКОЙ ЛОГИКИ

ADDMF

Добавляет функцию принадлежности к системе нечеткого логического вывода.

Синтаксис: *FIS_name = addmf*(*FIS_name, varType, varIndex, mfName, mfType, mfParams*)

Функцию принадлежности можно добавить только к существующей в рабочей области MATLAB системе нечеткого логического вывода. Другими словами, система нечеткого логического вывода должна быть каким-то образом загружена в рабочую область или создана с помощью функции *newfi*s. Функция *addmf* имеет шесть входных аргументов:

• *FIS_name* – идентификатор системы нечеткого логического вывода в рабочей области MATLAB;

• *varType* – тип переменной, к которой добавляется функция принадлежности. Допустимые значения – «input» – входная переменная и «output» – выходная переменная;

• *varIndex* – порядковый номер переменной, к которой добавляется функция принадлежности;

• *mfName* – наименование добавляемой функции принадлежности (терм). Задается в виде строки символов;

• *mfType* – тип (модель) добавляемой функции принадлежности. Задается в виде строки символов;

• *mfParams* – вектор параметров добавляемой функции принадлежности.

Порядковый номер функции принадлежности в системе нечеткого логического вывода соответствует порядку добавления с помощью функции *addmf*, т.е. первая добавленная функция принадлежности всегда будет иметь порядковый номер 1. С помощью функции *addmf* невозможно добавить функцию принадлежности к несуществующей переменной. В этом случае необходимо вначале добавить переменную к системе нечеткого логического вывода с помощью функции *addvar*.

Пример:

FIS_name=addmf(FIS_name, 'input', 1, 'низкий', 'trapmf', [150, 155, 165, 170])

Строка добавляет в терммножество первой входной переменной нечеткой системы *FIS_name* терм «низкий» с трапециевидной функцией принадлежности с параметрами [150, 155, 165, 170].

ADDRULE

Добавляет правила в базу знаний системы нечеткого логического вывода.

Синтаксис: *FIS_name= addrule (FIS_name, ruleList)*

Правила можно добавить только к существующей в рабочей области MATLAB системе нечеткого логического вывода. Функция *addrule* имеет два входных аргумента:

• *FIS_name* – идентификатор системы нечеткого логического вывода в рабочей области MATLAB;

• *ruleList* – матрица добавляемых правил.

Матрица правил должна быть задана в формате *indexed*. Количество строк матрицы *ruleList* равно количеству добавляемых правил, т.е. каждая строка матрицы соответствует одному правилу. Количество столбцов матрицы равно $m+n+2$, где m (n) – количество входных (выходных) переменных системы нечеткого логического вывода.

Первые m столбцов соответствуют входным переменным, т.е. задают ЕСЛИ-часть правил. Элементы этих столбцов содержат порядковые номера термов, используемых для лингвистической оценки

соответствующих входных переменных. Значение 0 указывает, что соответствующая переменная в правиле не задана, т.е. ее значение равно *none*.

Следующие *n* столбцов соответствуют выходным переменным, т.е. задают ТО-часть правил. Элементы этих столбцов содержат порядковые номера термов, используемых для лингвистической оценки соответствующих выходных переменных.

Предпоследний столбец матрицы содержит весовые коэффициенты правил. Значения весовых коэффициентов должны быть в диапазоне [0, 1].

Последний столбец матрицы задает логические связки между переменными внутри правил. Значение 1 соответствует логической операции «И», а значение 2 – логической операции «ИЛИ».

Пример:

FIS_name=addrule(FIS_name, [1 1 1 1 1; 1 2 2 0.5 1])

Строка добавляет в базу знаний системы *FIS_name* два правила, которые интерпретируются следующим образом:

Если вход1=MF1 и вход2=MF1, то выход1=MF1 с весом 1,
Если вход1=MF1 и вход2=MF2, то выход1=MF2 с весом 0.5,
где MF1 (MF2) – терм с порядковым номером 1 (2).

ADDVAR

Добавляет переменную в систему нечеткого логического вывода.

Синтаксис: *FIS_name= addvar (FIS_name, varType, varName, varBound).*

Переменную можно добавить только к существующей в рабочей области MATLAB системе нечеткого логического вывода. Функция *addrvar* имеет четыре входных аргумента:

• *FIS_name* – идентификатор системы нечеткого логического вывода в рабочей области MATLAB;

• *varType* – тип добавляемой переменной. Допустимые значения – «input» – входная переменная и «output» – выходная переменная;

• *varName* – наименование добавляемой переменной. Задается в виде строки символов;

• *varBound* – вектор, задающий диапазон изменения добавляемой переменной.

Порядковый номер переменной в системе нечеткого логического вывода соответствует порядку добавления с помощью функции *addvar*, т.е. первая добавленная переменная будет иметь порядковый номер 1. Входные и выходные переменные нумеруются независимо.

Пример:

FIS_name=addrule(FIS_name, 'input', 'Рост', [155 205])

Строка добавляет в систему нечеткого логического вывода FIS_name входную переменную 'Рост', заданную на интервале [155 205].

ANFIS

Настройка систем нечеткого логического вывода типа Сугэно.

Синтаксис:*[fis, error] = anfis(trndata)*

[fis, error] = anfis(trndata, initfis)

[fis, error, stepsize] = anfis(trndata, initfis, trnopt, dispopt, [], optmethod)

[fis, error, stepsize, chkfis, chkerror] = anfis(trndata, initfis, trnopt, dispopt, chkdata)

«ANFIS» является аббревиатурой «Adaptive Network – based Fuzzy Inference System». Это основная функция настройки систем нечеткого логического вывода типа Сугэно. Настройка представляет собой итерационную процедуру нахождения таких параметров системы нечеткого логического вывода, в частности параметров функций принадлежности, которые минимизируют расхождения между результатами логического вывода и экспериментальными данными, т. е. между действительным и желаемым поведениями системы. Экспериментальные данные, по которым настраиваются функции принадлежности, представляются в виде обучающей выборки.

В основу функции *anfis* положен один из первых методов построения нейро-нечетких систем для аппроксимации функций, предложенный в 1991 году Янгом (*Jang*). Для идентификации параметров системы нечеткого логического вывода типа Сугэно функция *anfis* может использовать метод обратного распространения ошибки, а также гибридный алгоритм, основанный на комбинации метода обратного распространения ошибки и метода наименьших квадратов. Функция *anfis* может использовать дополнительный аргумент для проверки модели – тестирующую выборку.

Функция *anfis* может имеет шесть входных аргументов:

1. *Trndata* – идентификатор обучающей выборки. Этот входной аргумент является обязательным. Обучающая выборка представляет собой матрицу, каждая строчка которой является парой «вход – выход». Последний столбец матрицы соответствует вектору значений выхода, а все остальные столбцы – входным данным. Заметим, что функцию *anfis* можно использовать для проектирования систем типа *SISO* (один вход – один выход) и типа *MISO* (много входов – один выход);

2. *Initfis* – идентификатор исходной системы нечеткого логического вывода, функции принадлежности которой будут настроены с помощью *anfis*. Исходная система нечеткого логического вывода должна быть системой типа Сугэно нулевого или первого порядка. Еще одним ограничением *anfis* является недопустимым использование типов функций принадлежностей и методов дефаззификации, определяемых пользователем. При отсутствии аргумента *initfis* функция *anfis* вызовет функцию *genfis*1, которая генерирует типовую систему нечеткого логического вывода. Эта типовая система будет использовать по две функции принадлежности гауссовского типа для каждой входной переменной. В случае, когда пользователь хочет получить систему с иным количеством функций принадлежности, необходимо задать в виде аргумента *initfis* количество функций принадлежности для оценки каждой входной переменной. Если количество функций принадлежности одинаково для всех входов, тогда достаточно указать только одно значение.

3. *Trnopt* – вектор параметров настройки:

• *trnopt*(1) – количество итераций (значение по умолчанию – 10);

• *trnopt*(2) – допустимая ошибка обучения (значение по умолчанию – 0);

• *trnopt*(3) – исходная длина шага (значение по умолчанию – 0.01);

• *trnopt*(4) – коэффициент уменьшения длины шага (значение по умолчанию – 0.9);

• *trnopt*(5) – коэффициент увеличения длины шага (значение по умолчанию – 1.1);

• *dispopt* – вектор, указывающий, какие промежуточные результаты обучения выводить в командное окно MATLAB во время настройки. Для вывода информации на экран необходимо установить соответствующую координату вектора в 1.

4. Вектор *dispopt* имеет четыре координаты:

• *dispopt*(1) – ANFIS-информация: количество функций принадлежности входных и выходной переменных и т.п.;

• *dispopt*(2) – ошибка обучения;

• *dispopt*(3) – длина шага, (выводится в случае его изменения);

• *dispopt*(4) – окончательный результат.

По умолчанию значения всех координат равны 1;

5. *Chkdata* – идентификатор тестирующей выборки. Тестирующая выборка используется для исследования свойства обобщения системы нечеткого логического вывода, т. е. способности системы выдавать правильные результаты для данных, отсутствующих в обучающей выборке. Формат тестирующей выборки такой же, как и обучающей. Обычно элементы тестирующей выборки несколько отличаются от элементов обучающей выборки. Тестирующая выборка особенно важна для задач обучения с большим количеством входов и (или) для задач с зашумленными данными. При формировании обучающей и тестирующей выборок необходимо стремиться к обеспечению свойств представительности выборок;

6. *Optmethod* – метод оптимизации, используемый для настройки. Допустимые значения: 1 – гибридный алгоритм и 0 – метод обратного распространения ошибки. По умолчанию применяется гибридный алгоритм, который использует метод обратного распространения ошибки (метод наискорейшего спуска) для настройки функций принадлежности входных переменных и метод наименьших квадратов для настройки функций принадлежности выхода. Точнее говоря, метод наименьших квадратов используется для нахождения коэффициентов линейных функций, связывающих входы и выход в каждом правиле. Метод по умолчанию используется всегда, за исключением случая, когда значение аргумента *optmethod* равно 0.

Функция *anfis* имеет пять выходных аргументов:

• *fis* – система нечеткого логического вывода, параметры которой настроены таким образом, что минимизируют расхождения между результатами логического вывода и экспериментальными данными из обучающей выборки;

• *error* – вектор, содержащий значения ошибки обучения на каждой итерации. Ошибка обучения рассчитывается как среднее квадратическое отклонение между результатами нечеткого логического вывода и значениями выходной переменной из обучающей выборки;

• *stepsize* – вектор, содержащий значения длины шага алгоритма оптимизации на каждой итерации;

• *chkfis* – система нечеткого логического вывода, параметры которой настроены таким образом, что минимизируют расхождения между результатами логического вывода и экспериментальными данными из тестирующей выборки. Для получения такой системы необходимо задать пятый входной аргумент функции *anfis* – тестирующую выборку;

• *chkerror* – вектор, содержащий значения ошибки тестирования на каждой итерации. Ошибка тестирования рассчитывается как среднее квадратическое отклонение между результатами нечеткого логического вывода и значениями выходной переменной из тестирующей выборки. Для расчета ошибки тестирования необходимо задать пятый входной аргумент функции *anfis* – тестирующую выборку.

Процесс настройки прекращается по достижению требуемой ошибки обучения настройки или после выполнения указанного количества итераций.

Функция *anfis* может быть вызвана с одним входным аргументом – обучающей выборкой и одним выходным аргументом – системой нечеткого логического вывода, параметры которой настроены по обучающей выборке. При вызове функции *anfis* значения неинициализированных аргументов принимаются равными по умолчанию. При инициализации только части аргументов, например, первого и шестого – обучающей и тестирующей выборок, значения остальных аргументов (для нашего примера – второго, третьего, четвертого и пятого) необходимо задать как [ ] или *NaN*. Значение [ ] или *NaN* указывает, что соответствующий аргумент принимает значения, установленные по умолчанию. Значения, установленные по умолчанию, могут быть изменены путем непосредственного редактирования файла *anfis.m*.

Пример:

Синтезируется и настраивается система нечеткого логического вывода *fis*, моделирующая зависимость $y=sin(x)$ в диапазоне [0, 1].

x=(0:0.04:1)';
y=sin(x);
trndata=[x y];
[fis, error, stepsize]=anfis(trndata).

CONVERTFIS

Преобразовывает систему нечеткого логического вывода, заданную в формате «Fuzzy Logic Toolbox v.1» в формат «Fuzzy Logic Toolbox v.2».

Синтаксис: *FIS_new= convertis (FIS_old)*

Функция преобразовывает систему нечеткого логического вывода, заданную матрицей «FIS_old» в формате «Fuzzy Logic Toolbox v.1» в структуру «FIS_new», в соответствии с форматом «Fuzzy Logic Toolbox v.2».

DEFUZZ

Дефаззификация нечеткого множества.

Синтаксис: *crisp = defuzz (x, mf, method)*

Выполняет операцию дефаззификации, т.е. преобразования нечеткого множества в четкое число. Функция *defuzz* имеет три входных аргумента:

- *x* – универсальное множество, на котором задано нечеткое множество, подлежащее дефаззификации;
- *mf* – вектор степеней принадлежности элементов множества *x* нечеткому множеству, подлежащему дефаззификации;
- *method* – метод дефаззификации.

Допустимые значения:

- '*centroid'* – центр тяжести;
- '*bisector'* – медиана;
- '*mom'* – центр максимумов;
- '*som'* – наименьший из максимумов;
- '*lom'* – наибольший из максимумов.

Если метод дефаззификации отличается от вышеуказанных, тогда он должен быть представлен в виде *m*-функции. В этом случае значения аргументов *x* и *mf* будут переданы этой функции для выполнения дефаззификации.

Пример:

Проводится дефаззификация нечеткого множества с трапециевидной функцией принадлежности с параметрами [0, 2, 4, 10]), заданного на универсальном множестве {0, 0.1, 0.2,…,10}.

x=0:0.1:10';
mf=trapmf(x, [0, 2, 4, 10]);
crisp=defuzz(x, mf, 'centroid').

DISCFIS

Дискретизация функций принадлежности всех термов, входящих в систему нечеткого логического вывода.

Синтаксис: *[XI, YI, XO, YO, R] = discfis (fis, numPts)*

Функция используется для ускорения нечеткого логического вывода путем дискретизации функций принадлежности всех термов, входящих в систему нечеткого логического вывода. Функция *discfis* может иметь два входных аргумента:

- *fis* – система нечеткого логического вывода;
- *numPts* – необязательный входной аргумент, задающий количество точек дискретизации функций принадлежности. Значение по умолчанию равно 181. Это означает, что все нечеткие множества представляются в виде 181 пары чисел «элемент универсального множества – степень принадлежности». При уменьшении точек дискретизации возрастает скорость выполнения логического вывода и уменьшается точность вычислений, и наоборот.

Функция *discfis* возвращает 5 выходных аргументов:

- *XI* – матрица абцисс-координат функций принадлежностей термов входных переменных. Размер матрицы *numPts* × *Ninp*, где *Ninp* – количество термов, используемых для лингвистической оценки входных переменных;
- *YI* – матрица степеней принадлежности элементов матрицы *XI* соответствующим термам. Размер матрицы *numPts* × *Ninp*, причем первый столбец содержит степени принадлежности первому терму первой входной переменной, а последний – последнему терму последней входной переменной;
- *XO* – матрица абцисс-координат функций принадлежности термов выходных переменных. Размер матрицы *numPts* × *Nout*, где *Nout* – количество термов, используемых для лингвистической оценки выходных переменных;

• *YO* – матрица степеней принадлежности элементов матрицы *XO* соответствующим термам. Размер матрицы *numPts* × *Nout*, причем первый столбец содержит степени принадлежности первому терму первой выходной переменной, а последний – последнему терму последней выходной переменной. Для системы типа Сугэно матрица *YO* является нулевой;

• *R* – список правил базы знаний в индексном формате.

Пример:

fis = readfis ('tipper');

[XI, YI, XO, YO, R] = discfis (fis).

Дискретизация функций принадлежностей демонстрационной системы нечеткого логического вывода «Tipper».

DSIGMF

Функция принадлежности в виде разности между двумя сигмоидными функциями.

Синтаксис*: y = dsigmf(x, params).*

Функция принадлежности в виде разности между двумя сигмоидными функциями задается формулой. Применяется для задания гладких асимметричных функций принадлежности.

Функция *dsigmf* имеет два входных аргумента:

• *x* – вектор, для координат которого необходимо рассчитать степени принадлежности;

• *params* – вектор параметров функции принадлежности. Порядок задания параметров – [*a*1 *c*1 *a*2 *c*2].

Функция *dsigmf* возвращает выходной аргумент *y*, содержащий степени принадлежности координат вектора *x*.

Пример:

```
x = 0: 0.1: 10;
y1 = dsigmf (x, [5 1 8 7] );
y2 = dsigmf (x, [5 4 5 7] );
y3 = dsigmf (x, [5 6 2 7] );
plot (x, [y1; y2; y3])
title ('dsigmf: a1=5, c2=7')
ylim([0 1.05])
legend('c1=1, a2=8', 'c1=4, a2=5', 'c1=6, a2=2')
```

Построение графиков функций принадлежности в виде разности между двумя сигмоидными функциями с разными параметрами на интервале [0, 10] представлено на рис. П5.1.

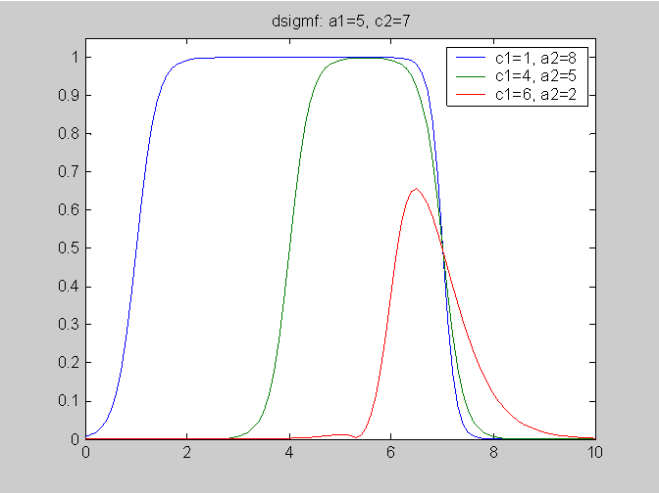


Рис. П5.1. Пример функции *dsigmf*

EVALFIS

Выполняет нечеткий логический вывод.

Синтаксис: *output = evalfis(input, fis)*

output = evalfis(input, fis, numPts)

[output, IRR, ORR, ARR] = evalfis(input, fis)

[output, IRR, ORR, ARR] = evalfis(input, fis, , numPts).

Функция *evalfis* может иметь три входных аргумента, первые два из которых обязательные:

- *input* – матрица значений входных переменных, для которых необходимо выполнить нечеткий логический вывод. Матрица должна иметь размер $M \times N$, где N – количество входных переменных; M – количество входных данных. Каждая строчка матрицы представляет один вектор значений входных переменных;
- *fis* – идентификатор системы нечеткого логического вывода;
- *numPts* – необязательный входной аргумент, задающий количество точек дискретизации функций принадлежности. Значение по умолчанию равно 101. Это означает, что все нечеткие множества представляются в виде 101 пары чисел «элемент универсального множества – степень принадлежности». При уменьшении точек дискретизации возрастает скорость выполнения логического вывода и уменьшается точность вычислений, и наоборот.

Функция *evalfis* может иметь четыре выходных аргумента:

- *output* – матрица значений выходных переменных, получаемая в результате нечеткого логического вывода для вектора входных значений *input*. Матрица имеет размер $M \times L$, где M – количество входных данных; L – количество выходных переменных в *fis*;

• *IRR* – матрица размером *NR* × *N*, где *NR* – количество правил в *fis*; *N* – количество входных переменных. Матрица содержит степени принадлежности входных значений термам, входящим в базу знаний;

• *ORR* – матрица размером *numPts* × (NR·L), где *numPts* – количество точек дискретизации; *NR* – количество правил в *fis*; *L* – количество выходных переменных в *fis*. Каждый столбец матрицы содержит функцию принадлежности выходной переменной, получаемую в результате вывода по одному правилу. Функция принадлежности дискретизируется на *numPts* точках и представляется в виде множества степеней принадлежности;

• *ARR* – матрица размером *numPts* × *L*, где *numPts* – количество точек дискретизации; *L* – количество выходных переменных в *fis*. Матрица содержит функции принадлежности выходных переменных, получаемые в результате нечеткого логического вывода по всей базе знаний. Функции принадлежности дискретизируются на *numPts* точках и представляются в виде множества степеней принадлежности.

Аргументы *IRR, ORR* и *ARR* являются необязательными, они содержат промежуточные результаты нечеткого логического вывода. В случае задания нескольких входных данных значения аргументов *IRR, ORR* и *ARR* будут рассчитаны только для последнего вектора входных данных. Эти аргументы используются, когда необходимо отследить процесс логического вывода или реализовать нестандартную процедуру нечеткого вывода.

Пример:

Первая строчка загружает демо-систему нечеткого логического вывода *tipper*, предназначенную для определения процента чаевых в ресторане. Вторая строчка рассчитывает размер чаевых, в случае если *service*=3 и *food*=8.

fis = readfis('tipper');

tip = evalfis([3 8], fis).

EVALMF

Вычисление значений произвольной функции принадлежности.

Синтаксис: *y = evalmf (x, params, type).*

Описание: позволяет вычислить значения произвольной функции принадлежности. Функция *evalmf* может иметь три входных аргумента:

• *x* – вектор, для координат которого необходимо рассчитать степени принадлежности;

• *params* – вектор параметров функции принадлежности, порядок задания которых определяется ее типом;

• *type* – тип функции принадлежности. Значение типа функции принадлежности может быть задано в виде строчки символов или числом:

1 –*'trimf'*; 7 –*'psigmf'*;
2 –*'trapmf'*; 8 –*'gbellmf'*;
3 –*'gaussmf'*; 9 –*'smf'*;
4 – ‘*gauss2mf*'; 10 –*'zmf'*;
5 –*'sigmf'*; 11 –*'pimf'*.
6 –*'dsigmf'*;

При задании другого типа функции принадлежности предполагается, что она определена пользователем и задана соответствующим *m*-файлом.

Функция *evalmf* возвращает выходной аргумент *y*, содержащий степени принадлежности координат вектора *x*.

Пример:

x = 0: 0.1: 10;y = evalmf (x, [0 3 9], 1);plot (x, y)title (‘Triangular membership function with parameters [0 3 9]').

Построение графика треугольной функции принадлежности с параметрами [0 3 9] на интервале [0, 10] представлено на рис. П5.2.

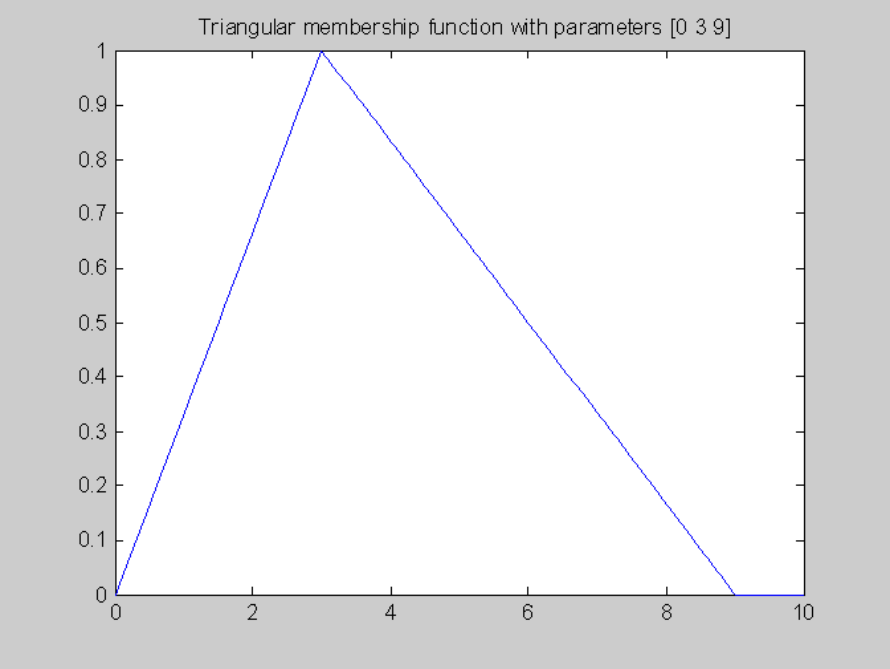


Рис. П5.2. Пример треугольной функции принадлежности с параметрами [0 3 9] на интервале [0, 10]

EVALMMF

Вычисление степеней принадлежностей для нескольких функций принадлежностей.

Синтаксис: *y = evalmmf (x, params, types).*

Описание: позволяет вычислить значения нескольких функций принадлежности нечетких множеств, заданных на одном и том же универсальном множестве. Функция *evalmmf* иметь три входных аргумента:

• *x* – вектор, для координат которого необходимо рассчитать степени принадлежности;

• *params* – матрица параметров функции принадлежности. Первая строка матрицы определяет параметры первой функции принадлежности, вторая строка – параметры второй функции принадлежности и т.д.;

• *types* – матрица типов функции принадлежности. Первая строка матрицы задает тип первой функции принадлежности, вторая строка – тип второй функции принадлежности и т.д. Значение типа функции принадлежности может быть задано в виде строчки символов или числом: 1 – *'trimf'*; 2 – *'trapmf'*; 3 – *'gaussmf'*; 4 – ‘*gauss2mf'*; 5 – *'sigmf'*; 6 – *'dsigmf'*; 7 – *'psigmf'*; 8 – *'gbellmf'*; 9 – *'smf'*; 10 – *'zmf'*; 11 – *'pimf'*. При задании другого типа функции принадлежности предполагается, что она определена пользователем и задана соответствующим *m*-файлом.

Функция *evalmmf* возвращает матрицу *y*, содержащую степени принадлежности координат вектора *x*. Первая строка матрицы содержит значения первой функции принадлежности, вторая строка – значения второй функции принадлежности и т.д.

Пример:

x = 0:0.2:10;para = [-1 2 3 4; 3 4 5 7; 5 7 8 0; 2 9 0 0];type = = str2mat('pimf', 'trapmf', 'trimf', 'sigmf');mf = evalmmf(x, para, type);

plot(x', mf');

ylim([0 1.05])

legend('pimf', 'trapmf', 'trimf', 'sigmf').

Построение графиков подобной, трапециевидной, треугольной и сигмоидной функций принадлежности на интервале [0, 10] представлено на рис. П5.3.

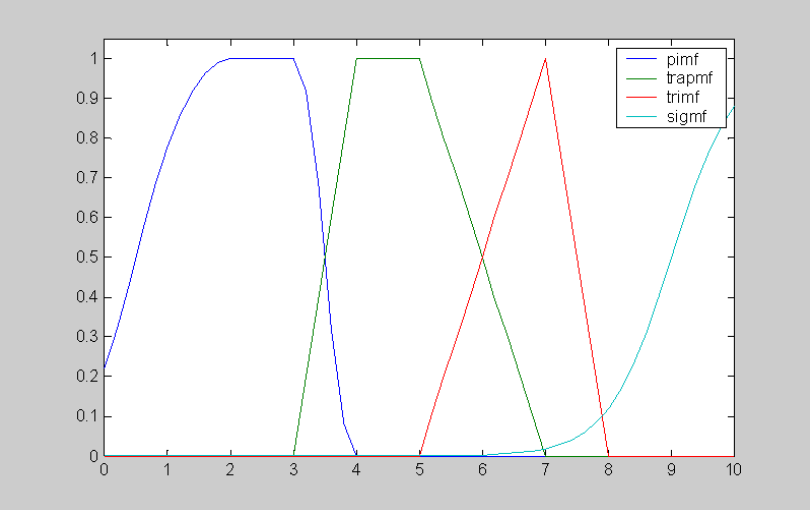


Рис. П5.3. Пример функции *evalmmf*

FCM

Кластеризация на основе нечеткого c-means алгоритма.

Синтаксис: *[V, M, obj_fcn] = fcm(X, c)*

[V, M, obj_fcn] = fcm(X, c, options).

Описание: на основе нечеткого c-means алгоритма выполняет кластеризацию данных. Этот алгоритм кластеризации предложил Джеймс Бэздэк (James Bezdek) в 1981 году.

Задача нечеткой кластеризации ставится следующим образом.

Дано: $x = (x_1, x_2, ..., x_n)^T$, объекты, подлежащие кластеризации (n – количество объектов). Каждый объект $x_k = (x_{k_1}, x_{k_2}, ..., x_{k_p})$ представляет собой точку в p-мерном пространстве признаков ($k = \overline{1, n}$); c – количество кластеров ($2 \le c < n$).

Необходимо каждому элементу множества X поставить в соответствие степени принадлежности c классам.

Элементы одного кластера должны быть так близки каждый каждому, как это только возможно, и одновременно кластеры должны быть на наибольшем удалении друг от друга. Для обеспечения управляемости процесса кластеризации необходимо использовать меру близости, в качестве которой обычно определяют расстояние между двумя объектами (точками в p-мерном пространстве) X_k и X_j в виде вещественной функции:

$d: X \times X \to R^+$ такой что:

$d(X_k X_{j,}) = d_{kj} \geq 0$

$d_{kj} = 0 \leftrightarrow X_k = X_j$

$d_{kj} = d_{jk}$

Дополнительно, если функция d удовлетворяет правилу треугольника, тогда эта функция является метрикой, хотя выполнение этого свойства не всегда необходимо для задач кластеризации.

Любое разбиение множества $x = (x_1, x_2, ..., x_n)^T$ на нечеткие подмножества $S_i (i = \overline{1, c})$ может быть полностью описано функцией принадлежности $\mu_{Si}: X \to [0,1]$.

Обозначим через μ_k – степень принадлежности объекта $x_k = (x_{k_1}, x_{k_2}, ..., x_{k_p})$ к подмножеству S_i, и через V_{cn} – множество всех действительных матриц размером $c \times n$. Тогда нечетким *c*-разбиением (или матрицей степеней принадлежности) называется матрица $M = [\mu_{ik}] \in V_{cn}$ при выполнении следующих условий:

1. $\mu_{ik} \in [0,1]$; $k = \overline{1, n}$, $i = \overline{1, c}$; 2. $\sum_{i=1}^{c} \mu_{ik} = 1$, $k = \overline{1, n}$; 3. $\sum_{k=1}^{n} \mu_{ik} \in (0, n)$, $i = \overline{1, c}$.

В отличие от четкого при нечетком *c*-разбиении любой объект одновременно принадлежит к различным кластерам, но с разной степенью:

$$\sum_{k=1}^{n} \mu_{ik} \in (0, n).$$

Условия 2 и 3 требуют только, чтобы сумма степеней принадлежности объекта ко всем кластерам была нормализована к 1, а также, чтобы количество кластеров, к которым принадлежит объект, не превышало *c*.

Обозначим центры кластеров, т.е. точки в *p*-мерном пространстве, вокруг которых сконцентрированы соответствующие объекты, через $V_i = (v_{i1} \vee v_{i2} \ldots \vee v_{in})$, $i = \overline{1, c}$.

При использовании эвклидового расстояния задача нечеткой кластеризации состоит в нахождении такой матрицы степеней принадлежности и таких координат центров кластеров, которые обеспечивают минимум следующего критерия:

$$\sum \sum (\mu_{ik})^m \left\| X_k - V_i \right\|^2 \to min;$$

где $V_i = \frac{1}{\sum_{k=1}^{n} \mu_{ik}} \sum_{k=1}^{n} (\mu_{ik})^m X_k$; центр i-го кластера, $i = \overline{1,c}$, m – так называемый экспоненциальный вес ($m >= 1$).

Значение экспоненциального веса устанавливается до начала кластеризации. Экспоненциальный вес m влияет на матрицу степеней принадлежности. Чем больше m, тем конечная матрица c-разбиения становится более «размазанной», и при $m \to \infty$ она примет вид M = [1/c], что является очень плохим решением, так как все объекты принадлежат ко всем кластерам с одной и той же степенью. Также экспоненциальный вес позволяет при формировании координат центров кластеров усилить влияние объектов с большими значениями степеней принадлежности и уменьшить влияние объектов с малыми значениями степеней принадлежности. На сегодня не существует теоретически обоснованного правила выбора значения m. Обычно устанавливают m = 2.

Аналитического решения задачи нахождения оптимальных координат центров кластеров и матрицы степеней принадлежности не существует, поэтому она решается численно. Один из итерационных алгоритмов решения этой задачи реализован в функции *fcm*.

Функция *fcm* может иметь три входных аргумента:

1) X – матрица, представляющая данные, подлежащие кластеризации. Каждая строка матрицы соответствует одному объекту (образу);

2) c – количество кластеров, которое должно быть получено в результате выполнения функции *fcm*. Количество кластеров должно быть больше 1 и меньше числа образов, заданных матрицей X;

3) *Options* – необязательный аргумент, устанавливающий параметры алгоритма кластеризации:

• *options*(1) – значения экспоненциального веса (значение по умолчанию – 2.0);

• *options*(2) – максимальное количество итераций алгоритма кластеризации (значение по умолчанию – 100);

• *options*(3) – минимально допустимое значение улучшения целевой функции за одну итерацию алгоритма (значение по умолчанию – 0.00001);

• *options*(4) – вывод промежуточных результатов во время работы функции *fcm* (значение по умолчанию – 1).

Для использования значений по умолчанию можно вести *NaN* в качестве значения соответствующей координаты вектора *options*.

Алгоритм кластеризации останавливается, когда выполнено максимальное количество итераций или когда улучшение значения целевой функции за одну итерацию меньше указанного минимально допустимого значения.

Функция *fcm* имеет три выходных аргумента:

1) *V* – матрица координат центров кластеров, полученных в результате кластеризации. Каждая строка матрицы соответствует центру одного кластера;

2) *M* – матрица степеней принадлежности образов к кластерам. Каждая строка матрицы соответствует функции принадлежности одного кластера;

3) *obj_fcn* – вектор значений целевой функции на каждой итерации алгоритма кластеризации.

Примечание: при описании нечеткого *c-means* алгоритма использована книга Zimmermann H.-J. Fuzzy Set Theory-and Its Applications. 3rd ed. Kluwer Academic Publishers, 1996. 435 p.

Пример:

Проводится кластеризация объектов, образующих фигуру типа «бабочка». Результаты кластеризации приведены на рис. П5.4.

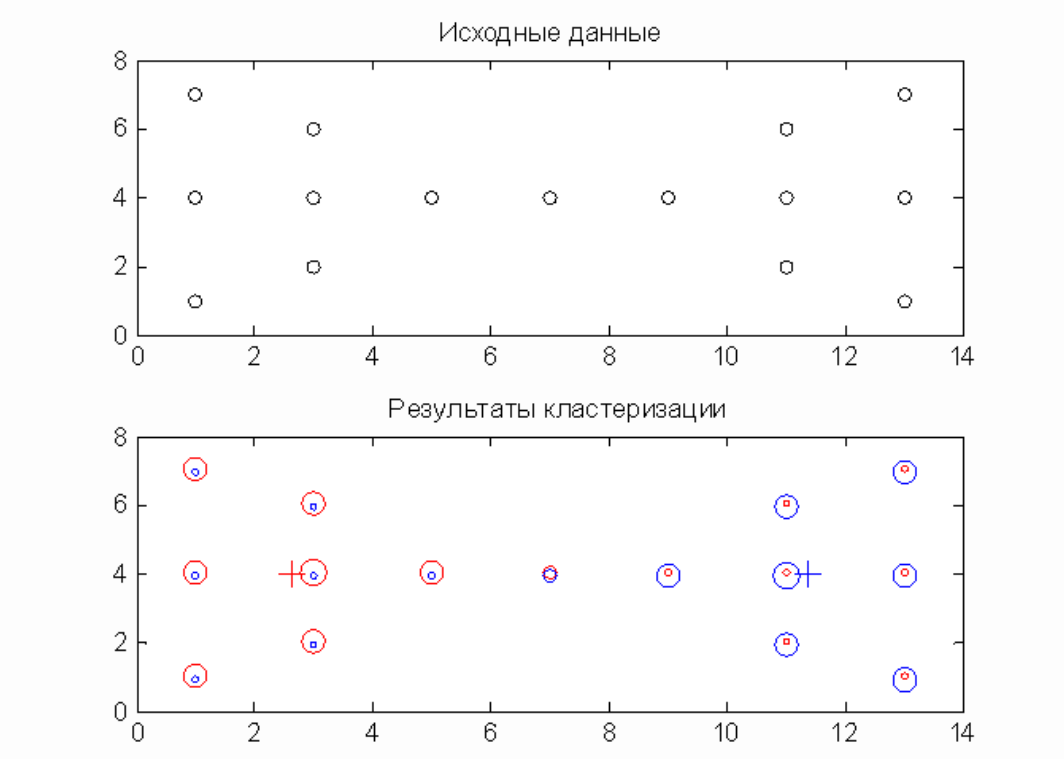


Рис. П5.4. Результаты кластеризации

Центры кластеров указаны маркером «+». Размер маркера «о» пропорционален степени принадлежности объекта кластеру. Расположенный в центре объект имеет одинаковые степени принадлежности к красному и синему кластерам. Это обеспечивает симметричное разбиение объектов по кластерам, что невозможно при четкой кластеризации.

```
X=[1 1; 1 4; 1 7; 3 2; 3 4; 3 6; 5 4; 7 4; 9 4; 11 2; 11 4; 11 6; 13 1; 13 4; 13 7];
[V M opt]=fcm(X, 2);
subplot(2,1,1);
plot(X(:,1), X(:,2), 'ko', 'markersize', 6);
text='Исходные данные';
title(text);
xlim([0 14]);
ylim([0 8]);
subplot(2,1,2);
plot(V(1,1), V(1,2), 'r+', 'markersize', 10);
hold on;
plot(V(2,1), V(2,2), 'b+', 'markersize', 10);
for i=1:15;
plot(X(i,1), X(i,2)+.05, 'ro', 'markersize', M(1,i)*8+2);
plot(X(i,1), X(i,2)-.05, 'bo', 'markersize', M(2,i)*8+2);
end;
xlim([0 14]);
ylim([0 8]);
text='Результаты кластеризации';
title(text).
```

FINDROW

Нахождение строки в матрице, совпадающей с входной строкой.

Синтаксис: *rownum = findrow (str, strmat)*

Описание: функция *findrow* возвращает номер строки матрицы *strmat*, содержащую стринг, заданный аргументом *str*. Если таких строк несколько, то функция *findrow* возвратит номера всех строчек.

Функция *findrow* не связана с нечеткими множествами и нечеткой логикой, она используется для выполнения алгоритмов обработки информации, необходимых для других функций «Fuzzy Logic Toolbox».

Пример: *rownum = findrow('два ', ['один'; 'два'; 'три']).*

Функция возвращает значение 2, которое означает, что стринг 'два ' содержится во второй строке матрицы ['один'; 'два '; 'три'].

FSTRVCAT

Конкатенация матриц различных размеров.

Синтаксис: *aout = fstrvcat(a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11)*

Описание: функция *fstrvcat* формирует матрицу *aout*, строки которой содержат векторы *a*1, *a*2, *a*3, …. Для того чтобы сформировать корректную матрицу, к некоторым векторам добавляется необходимое количество нулевых символов. Количество входных аргументов не должно превышать 11. Входные аргументы функции *fstrvcat* могут быть заданы как векторами, так и матрицами. Это позволяет формировать выходную матрицу *aout* практически любого размера.

Функция *fstrvcat* не связана с нечеткими множествами и нечеткой логикой, она используется для выполнения алгоритмов обработки информации, необходимых для формирования *fis*-структуры.

Пример:

aout = fstrvcat ('hi', 'blue', [110 101 101 100 108 101])

Функция fstrvcat возвращает следующую матрицу:

aout =

1041050000

9810811710100

110101101100108101,

строки которой содержат значения входных аргументов в формате *ASCII*.

FUZARITH

Выполнение нечетких арифметических операций.

Синтаксис: *c = fuzarith(x, a, b, operator)*

Описание: выполняет арифметические операции сложения, вычитания, умножения и деления над нечеткими числами.

Функция *fuzarith* имеет четыре входных аргумента:

• *x* – универсальное множество, на котором заданы нечеткие числа;

• *a* – вектор, задающий первый операнд. Представляет собой вектор степеней принадлежности элементов универсального множества первому нечеткому множеству. Другими словами, аргументы *x* и *a* образуют первое нечеткое число;

• *b* – вектор, задающий второй операнд. Представляет собой вектор степеней принадлежности элементов универсального множества второму нечеткому множеству, т.е. аргументы *x* и *b* образуют второе нечеткое число;

• *operator* – арифметическая операция:

‘*sum’* – сложение;

‘*sub’* – вычитание;

‘*prod’* – умножение;

‘*div’* – деление.

Выходной переменной функции *fuzarith* является вектор степеней принадлежности элементов универсального множества *x* результату выполнения нечеткой арифметической операции. Размерности векторов *x, a, b* и *c* должны быть одинаковыми.

Пример.

Рассчитывается нечеткое число *c* как произведение нечетких чисел *a* и *b* с гауссовскими функциями принадлежности, заданных на универсальном множестве {0, 0.01, …, 1}. Графики функций принадлежности показаны на рис. П5.5.

```
x=0:0.01:1;
a=gaussmf(x, [0.1 0.3]);
b=gaussmf(x, [0.2 0.6]);
c=fuzarith(x, a, b, 'prod');
plot(x, a, x, b, x, c);
legend('a', 'b', 'c=a*b').
```

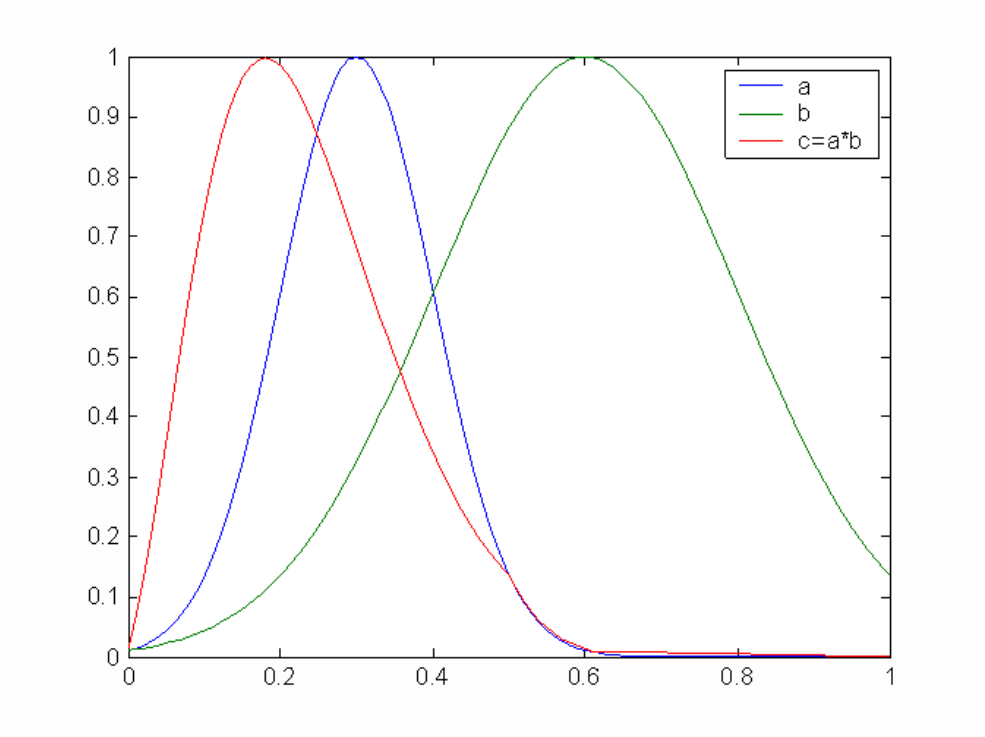


Рис. П5.5. Графики функций принадлежности

Приложение 6

ОТВЕТЫ НА ЗАДАНИЯ (к с. 91, 113)

4. Разработка нейроконтроллера:

1) b; 2) a; 3) d; 4) d; 5) d.

5. Нечеткие множества в системах диагностики:

1) d; 2) b; 3) a; 4) c; 5) b.

Учебное издание

ВЕСЕЛОВ Олег Вениаминович

САБУРОВ Павел Сергеевич

МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ДИАГНОСТИКЕ

Учебное пособие

Редактор Е. В. Невская

Технический редактор Н. В. Тупицына

Корректор Е. П. Викулова

Компьютерная верстка Л. В. Макаровой

Подписано в печать 07.10.15.
Формат 60×84/16. Усл. печ. л. 14,65. Тираж 100 экз.
Заказ
Издательство
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых.
600000, Владимир, ул. Горького, 87.