Нужная информация всегда под рукой!

Владимир Дьяконов
Владимир Круглов

- System Identification
- Nonlinear Control Systems
- Power System
- Communication
- Stateflow
- Real-Time Workshop

MATLAB

АНАЛИЗ, ИДЕНТИФИКАЦИЯ И МОДЕЛИРОВАНИЕ СИСТЕМ

Специальный справочник

ПИТЕР®

Содержание

Введение

Система MATLAB

Система MATLAB (MATrix LABoratory — матричная лаборатория) была создана специалистами фирмы MathWorks, Inc. как язык программирования *высокого уровня* для технических вычислений. Она вобрала в себя не только передовой опыт развития и современной компьютерной реализации численных методов, накопленный за последние три-четыре десятилетия, но и весь опыт становления математики за всю историю человечества. Особенно тщательно в MATLAB проработаны алгоритмы матричных операций, лежащие в основе большинства средств моделирования сложных систем. Эти высокоэффективные алгоритмы и реализующие их программные коды получили широкую известность и признание во всем мире, превратив систему MATLAB в один из самых мощных и эффективных инструментов для создания разнообразных программных комплексов, предназначенных для решения научно-технических задач.

Одним из самых важных и по достоинству оцененных качеств системы MATLAB является возможность ее модификации с целью решения все новых и новых научно-технических задач, которые в изобилии появляются благодаря прогрессу в науке, технике и образовании. Это достигается прежде всего созданием целого ряда пакетов расширения системы, охватывающих многие новые и практически полезные направления компьютерной математики. У системы MATLAB число этих пакетов составляет уже многие десятки, а документация по ним насчитывает десятки тысяч страниц.

Система MATLAB имеет открытую архитектуру, что дает пользователям полный доступ к ее кодам на гибком и мощном (и в то же время простом) языке программирования этой системы. Он является одним из лучших и высокоэффективных языков программирования для научно-технических расчетов, в том числе для создания удобных и очень наглядных визуально-ориентированных средств анализа, идентификации, построения и моделирования систем.

Пакет моделирования динамических систем Simulink

Современные версии системы MATLAB поставляются вместе с пакетом расширения Simulink, предназначенным для моделирования динамических систем, модели которых составляются из отдельных блоков (компонентов). Этот пакет является самым ярким представителем программ, созданных на основе системы MATLAB. При этом в нем реализованы принципы визуально-ориентированного программирования, что позволяет легко набирать нужные блоки и соединять их с целью составления модели анализируемой системы или устройства. При этом сложнейшие уравнения состояния, описывающие работу моделей, формируются автоматически.

По удобству графического пользовательского интерфейса, обилию моделей (блоков) компонентов в множестве библиотек, разнообразию виртуальных средств регистрации и визуализации результатов моделирования и, главное, по их надежности и достоверности Simulink выгодно отличается от множества других программ подобного назначения. Особенно это относится к открытости пакета и возможностям пополнения его библиотек. Вместе с базовой системой MATLAB, имеющей самые совершенные алгоритмы матричных вычислений и наиболее приспособленной для решения задач моделирования, Simulink становится мощнейшим инструментом познания реалий мира путем их моделирования. И эти возможности многократно усиливаются десятками пакетов расширения системы «MATLAB+Simulink».

Структура комплекса «MATLAB+Simulink+пакеты расширения»

Итак, система MATLAB и расширение Simulink поставляются в виде единого комплекса, в который могут входить различные *пакеты расширения*. Следует отметить, что большинство этих пакетов поставляется и продается фирмой MathWorks отдельно от комплекса «MATLAB+Simulink», причем стоимость некоторых пакетов заметно превышает стоимость самой базовой системы «MATLAB+Simulink». Впрочем, по всем нам хорошо известным причинам, этот вопрос наших пользователей чаще всего не волнует (в отличие от вопроса поиска редкой у нас документации по пакетам расширения системы MATLAB).

Рисунок 0.1 дает представление о комплексе «MATLAB+Simulink+пакеты расширения». На нем перечислены почти все пакеты расширения, вошедшие в 11-ю реализацию системы.

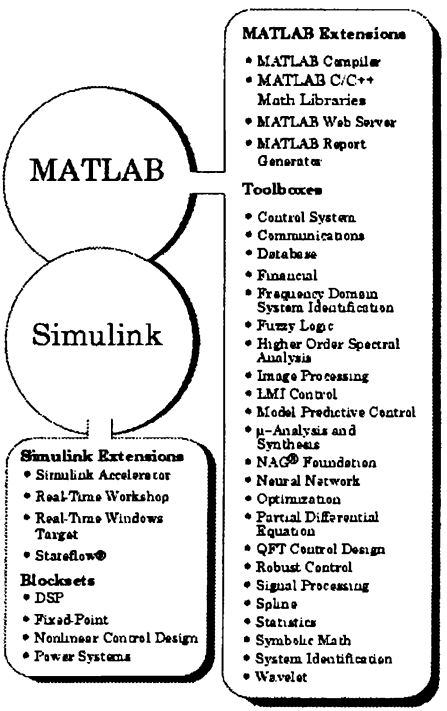


Рис. 0.1. Комплекс «MATLAB+Simulink+пакеты расширения»

Раздел MATLAB Extensions (Расширение MATLAB) содержит средства для подготовки исполняемых приложений MATLAB (с подключаемыми библиотеками и без них). Прежде всего в состав этих средств входят программные средства-компиляторы. Эти средства в данной книге не рассматриваются. Предполагается, что работа с описанными в ней пакетами расширения происходит в среде «MATLAB+Simulink» и создание исполняемых файлов по результатам этой работы не предполагается.

Раздел Toolboxes (Инструментальные средства) содержит самое большое число пакетов расширения системы MATLAB. Несколько пакетов расположено в разделе Blocksets, относящемся к главному пакету

расширения системы MATLAB — Simulink. В разделе расширений этого пакета — Simulink Extensions — имеется ряд средств, предназначенных для моделирования в реальном масштабе времени и событийно-управляемого моделирования. Для моделирования в реальном масштабе времени необходимо оснащение компьютера специальными адаптерами и датчиками. В связи с этим данные специфические вопросы рассматриваются в книге лишь обзорно.

Как видно из названий пакетов расширения системы «MATLAB+Simulink», они охватывают необычно широкий круг задач анализа и моделирования самых разнообразных систем, устройств и явлений реального мира. Данная книга посвящена описанию только тех пакетов расширения, которые имеют прямое отношение к идентификации, анализу и моделированию систем. Несмотря на это ограничение надо отметить, что в книге рассматривается довольно широкий круг вопросов, относящихся к идентификации, анализу, моделированию и проектированию самых разнообразных систем и устройств — в том числе механических, электрических, электронных, телекоммуникационных.

Мы, как правило, особо не оговариваем то важное обстоятельство, что все пакеты расширения системы MATLAB перед использованием нуждаются в инсталляции. Обычно она выполняется в ходе инсталляции системы MATLAB путем выбора имен нужных пакетов расширения. Следует также помнить, что большинство пакетов расширения распространяются как дополнительные программные продукты, причем их стоимость нередко довольно велика.

Временами возникает довольно курьезный вопрос — что первично, система MATLAB или система динамического моделирования Simulink? Несерьезность такой постановки вопроса становится совершенно очевидной из ответов на другие вопросы. Например, может ли работать система MATLAB без расширения Simulink? Или может ли Simulink работать без MATLAB? Ответ на первый вопрос однозначно положительный, а на второй — отрицательный. Так что философский вопрос о том, что появилось раньше — яйцо или курица — в данном случае просто неуместен.

Основой комплекса «MATLAB+Simulink+пакеты расширения» несомненно является матричная система MATLAB. Работа Simulink и других пакетов расширения без этой системы просто невозможна и им принадлежит вторичная, но вовсе не второстепенная роль. Многие из пакетов расширения системы «MATLAB+Simulink» — это серьезные программные продукты, вполне заслуживающие отдельного описания и знакомства с ними.

Литература по системе MATLAB и пакетам ее расширения

По самой системе MATLAB у нас уже опубликован ряд известных книг (ссылки на некоторые из них даны в каждой из глав данной книги). Ниже представлен список основных книг на русском языке, в которых описана базовая система MATLAB.

1. Дьяконов В. П. Справочник по применению системы PC MATLAB. — М.: Наука. Физматлит, 1993.
2. Дьяконов В. П. Компьютерная математика. Теория и практика. — М.: Нолидж, 2001.
3. Дьяконов В., Новиков Ю., Рычков В. Компьютер для студента. Самоучитель. — СПб: Питер, 2000.
4. Потемкин В. Г. MATLAB. Справочное пособие. — М.: ДИАЛОГ-МИФИ, 1997.
5. Потемкин В. Г. MATLAB 5 для студентов. — М.: ДИАЛОГ-МИФИ, 1998.
6. Потемкин В. Г. Система инженерных и научных расчетов MATLAB 5.x. Том 1 и 2. — М.: ДИАЛОГ-МИФИ, 1999.
7. Дьяконов В. П., Абраменкова И. В. MATLAB 5. Система символьной математики. — М.: Нолидж, 1999.
8. Дьяконов В. П. MATLAB. Учебный курс. — СПб: Питер, 2000.
9. Дьяконов В. П., Абраменкова И. В., Круглов В. В. MATLAB 5.3.1 с пакетами расширений. — М.: Нолидж, 2001.
10. Лазарев Ю. MATLAB 5.x. — К.: «Ирина», BHV, 2000.

Здесь уместно отметить, что первой книгой в России по системе MATLAB была книга [1], написанная еще в те времена, когда система MATLAB была далеко не самой популярной. В книгах [2, 3] система MATLAB описана вместе с рядом систем компьютерной математики, что позволяет оценить ее роль в сравнении с другими системами. Из приведенных выше книг на Интернет-сайте фирмы MathWorks пока представлена единственная русскоязычная книга — [7].

За рубежом опубликованы сотни книг, посвященных системе MATLAB и ее применению. Однако у нас книг по пакетам расширения все еще очень мало. Такие книги приведены в списках литературы в соответствующих главах. Практически нет книг с систематизированным описанием пакетов, объединенных общей направленностью. Этот серьезный пробел и призвана восполнить задуманная авторами серия

книг по системе MATLAB и важнейшим пакетам ее расширения. Первые две книги этой серии — учебный курс по базовой системе MATLAB 5.* (реализация 11) и справочник по математическим пакетам ее расширения — уже вышли.

Данная книга является третьей в этой серии. Она посвящена описанию пакетов идентификации, анализа и моделирования систем и устройств, то есть вопросам, имеющим фундаментальное значение в проектировании самых разнообразных устройств и изделий в самых разных областях науки и техники — вплоть до сложных систем, используемых в экономике, энергетике и промышленности.

Состав книги

Как и в упомянутых ранее книгах, одной из главных трудностей при подготовке данной книги были отбор и систематизация огромного по объему материала по системе MATLAB 5.* и пакетам ее расширения. Некоторые из описанных в данной книге пакетов расширения имеют описания, насчитывающие многие сотни страниц и представленные в справочной системе в виде HTML- и PDF-файлов. Поэтому естественно, что данная книга не претендует на полную и формальную замену этой громоздкой и трудно обозримой англоязычной документации (как электронной, так и обычной). Она скорее является кратким справочным руководством по описанным в ней пакетам расширения, содержащим тщательно отобранные и наиболее важные сведения. В этом и заключается ценность книг такого рода.

При этом авторы постарались для большинства пакетов все же дать описание практически всех их практически значимых функций, хотя и с разной степенью полноты такого описания. В книге нет описания самой системы MATLAB и поставляемого с ней пакета моделирования Simulink, поскольку оно уже было дано в упомянутых выше книгах.

В данную книгу включены только те пакеты расширения, которые имеют ярко выраженную направленность на идентификацию, анализ, исследование, разработку и моделирование систем. В книге рассмотрены следующие пакеты расширения:

- идентификации систем — System Identification;
- идентификации систем в частотной области — Frequency Domain Identifications;
- разработки систем управления — Control Systems;

- проектирования нелинейных систем управления — Nonlinear Control Systems;
- разработки электротехнических и энергетических систем — Power Systems;
- моделирования и проектирования систем связи — Communications;
- событийного моделирования систем — Stateflow;
- моделирования в реальном времени — Real-Time Workshop.

Разумеется, этим списком не исчерпывается состав пакетов расширения системы MATLAB в области идентификации, анализа и применения систем. Авторы были вынуждены отказаться от описания некоторых узкоспециализированных и громоздких систем (например, в области картографии) и систем, предназначенных для сбора информации с реальных физических устройств и ее обработки в реальном масштабе времени (кроме описанной обзорно системы Real-Time Workshop). Их описание, безусловно, может представлять тему для отдельных книг. Возможно, они появятся в будущем.

В главе 1 книги описан пакет идентификации систем — System Identification. В этой главе даны определения систем и основных их характеристик. Основное внимание уделено применению функций идентификации систем. Авторы сочли возможным дать достаточно полный теоретический материал по идентификации систем, учитывая, что данный пакет не очень знаком нашим пользователям.

Глава 2 посвящена пакету Frequency Domain Identification. Хотя область применения этого пакета — идентификация линейных систем в частотной области — является не столь широкой, как для пакета System Identification, она занимает большое место в практике анализа и исследования систем. Особенно важна идентификация систем в частотной области для радиоэлектронных систем.

Глава 3 посвящена разработке, проектированию и моделированию систем управления с помощью пакета расширения Control Systems. Это один из наиболее крупных пакетов расширения в системе «MATLAB+Simulink» и ему уделено в книге достаточно внимания. Разумеется, здесь не могло идти речи о полном описании пакета, поскольку оно должно быть темой отдельной и достаточно объемной монографии. Тем не менее, подавляющее большинство функций пакета описано, а для многих даны и примеры применения.

Глава 4 посвящена небольшому, но важному пакету проектирования нелинейных систем управления — Nonlinear Control Systems. Роль

моделирования нелинейных систем особенно велика, поскольку для них, как правило, не существует аналитических методов расчета и анализа. В этом случае возможность их численного моделирования становится решающей. Интересной возможностью этого пакета является оптимизация нелинейных систем по различным критериям и возможность коррекции их параметров с целью получения заданных характеристик систем. Всяческих похвал заслуживает оригинальный пользовательский интерфейс этого пакета.

Глава 5 впервые в нашей литературе описывает пакет Power Systems Blockset, предназначенный для моделирования электротехнических и энергетических устройств. Область применения этого пакета очень широка — от анализа отдельных электрических цепей до моделирования систем электропривода электрических машин различного типа и моделирования сложнейших физических и электромагнитных процессов в энергетических системах большой мощности. В пакете реализуются основные методы анализа таких систем и устройств: анализа статического режима работы, частотного и спектрального анализа и моделирования переходных процессов во временной области. Пакет описан полностью, а в отдельных разделах содержит материалы, отсутствующие даже в фирменном описании.

Глава 6 посвящена еще одному обширному пакету с необычайно обширными возможностями — Communications[1]. Этот пакет также впервые описывается в нашей литературе. Пакет описан достаточно полно, хотя это описание не претендует на роль исчерпывающего отражения возможностей пакета. Описание удалось сделать достаточно компактным, отказавшись от полного описания многих очень похожих блоков и примеров их применения, а также благодаря представлению моделей в виде копий экрана одновременно с окнами установки их параметров.

Наконец, глава 7 посвящена принципиально иному по назначению пакету моделирования систем — Stateflow. Базируясь на теории конечных автоматов, этот пакет обеспечивает событийное (ситуационное) моделирование систем с построением наглядных SF-диаграмм с элементами анимации, наглядно показывающих состояние отдельных объектов в ходе моделирования, а также демонстрирующих связи между ними, меняющиеся при возникновении событий. Этот па-

[1] Пакет Communications фактически делится на две части — набор функций MATLAB и набор блоков для Simulink. В MATLAB 6 это разделение оформилось окончательно — в списке компонентов системы отдельно фигурируют Communications Toolbox и Communications Blockset. — *Примеч. ред.*

кет существенно дополняет возможности пакета Simulink и других пакетов расширения системы MATLAB. Данная глава завершается кратким описанием еще одного практически важного пакета — мастерской реального времени Real-Time Workshop. С ее помощью можно готовить исполняемые файлы и файлы на языке C/C++ для совместной работы с внешними устройствами.

Графическая часть материала книги, включая схемы моделей и результаты моделирования, представлена в виде копий экрана, полученных для реально опробованных примеров. Это призвано повысить информационную насыщенность материала и способствует существенному уменьшению вероятности ошибок при представлении примеров моделирования. Тем не менее, авторы настоятельно рекомендуют сочетать чтение этой книги с реальной работой с описанными в них пакетами. Иначе их изучение очень напоминает попытки обучения плаванию без воды или даже парашютным прыжкам без парашюта.

Следует отметить, что материал книги носит цельный характер. Даже тем читателям, которых интересует лишь тот или иной конкретный пакет расширения, следует ознакомиться с книгой в целом. Это связано с тем, что многие важные определения в области теории и практики идентификации и моделирования систем изложены в первых главах книги, тогда как последующие главы нередко существенно расширяют представления об этих вопросах и дают множество интересных практических примеров моделирования систем и устройств.

Трудно переоценить роль этих вопросов в образовании, особенно в университетах и вузах, где ведутся курсы по анализу, идентификации, моделированию и проектированию систем. Хотя книга является справочной монографией и содержит результаты многолетнего исследования авторами систем класса MATLAB, авторы искренне надеются, что она окажется полезной и в качестве учебного пособия по таким курсам. Доводом в пользу этого является многолетний опыт педагогической и научной работы авторов. Вообще же книга рассчитана на широкий круг читателей — студентов и преподавателей вузов и университетов (прежде всего технических), инженеров и научных работников.

О версиях системы MATLAB

Для определенности отметим, что в этой книге рассмотрены версии пакетов расширения системы MATLAB 5.3.1 (11-я реализация). Практически это описание подходит для всех версий систем MATLAB 5.*

и даже для новейшей версии MATLAB 6.0 (реализация 12 — рис. 0.2), которая появилась во время подготовки рукописи данной книги, но пока находится в начальной стадии распространения и развития.

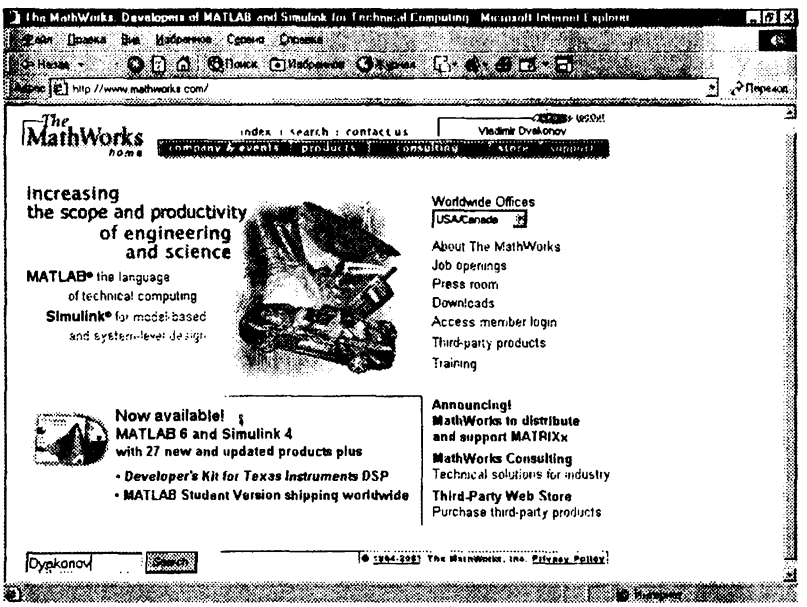


Рис. 0.2. Web-страница фирмы MathWorks с рекламой новейшей версии системы MATLAB 6.0 с пакетами расширения

Это вполне соответствует довольно распространенному мнению по поводу применения современных программных продуктов: надо использовать предпоследнюю их версию, ибо самая последняя версия обычно содержит ряд недоработок и даже ошибок. Тем не менее, как показала проверка новой версии MATLAB 6.0, особо существенных различий в последних версиях пакетов расширения, по сравнению с описанными в данной книге, пока нет. Впрочем, это справедливо только в части тех материалов, которые даны в данной книге. Описания новых возможностей версии MATLAB 6.0 (а их немало) и пакетов ее расширения, естественно, в данной книге нет.

Тем не менее, за редкими исключениями, описания пакетов расширения по идентификации, анализу и моделированию систем в данной книге вполне пригодны и для пользователей новейшей версии MATLAB 6. Переработка описания пакетов расширения только под

новую версию MATLAB 6.0 пока нецелесообразна, поскольку она распространена у нас (да и во всем мире) намного меньше, чем версии MATLAB 5.* и все еще находится в стадии развития. Очевидно, что для полного описания возможностей новой реализации системы MATLAB 6.0 и пакетов ее расширения нужна подготовка новых книг.

Предупреждения

Книги по компьютерной тематике пишутся по возможности быстро, иначе они неизбежно устаревают уже к моменту своего выхода в свет. Разумеется, в таких условиях трудно гарантировать, что в книге будут напрочь отсутствовать опечатки и недочеты. Авторы считают нужным предупредить читателей об этом, хотя сами они и издательство сделали все возможное, чтобы свести ущерб от спешки в ее подготовке к минимуму.

Работа с такой мощной математической системой, как MATLAB, и тем более с пакетами ее расширения, требует от читателя знания основ математики, а применительно к данной книге и основ системного анализа и системотехники. Без этого невозможно гарантировать правильное применение используемых в системе методов и корректность получаемых результатов. В связи с этим следует отметить, что данная книга не является исчерпывающим справочником по теории и методам применения систем и даже по описанным в ней пакетам расширения системы MATLAB. Она лишь излагает избранный материал по отдельным пакетам расширения с системной направленностью. Авторы берут на себя ответственность за то, какие именно из пакетов расширения можно отнести к этой категории.

Многие методы идентификации и анализа систем в книге описаны кратко, а некоторые (в основном узкоспециализированные) лишь упоминаются по названию. Несмотря на справочный характер книги, в ней, тем не менее, уделено достаточно внимания описанию теоретических положений для большинства из описанных пакетов — без этого возможности соответствующих пакетов расширения будут просто непонятны большинству читателей. Достаточно полно описаны и функции большинства пакетов.

Мы вынуждены предупредить читателя, что авторы и издательство не несут никакой ответственности за ошибки пользователей (особенно учащихся) в освоении системы и за моральный или даже экономический ущерб, который может иметь место вследствие ошибок и неудачного выбора математической системы для обучения и применения при решении конкретных задач пользователя.

Сказанное ни в коей мере не означает, что в данной книге или в системе MATLAB заведомо имеются серьезные ошибки и недочеты. Просто такое предупреждение отвечает нормам современного юридического права в отношении сложных программных продуктов и сопровождающей их документации.

Благодарности и адреса для связи

Эта книга написана в инициативном порядке на кафедре физической и информационной электроники Смоленского государственного педагогического университета (СГПУ) при поддержке ЗАО «Смоленский телепорт» (www.keytown.com) и на кафедре управления и информатики Смоленского филиала Московского энергетического института – технического университета (СФМЭИ). В ней частично отражены материалы работ одного из авторов (Дьяконова В. П.) по гранту Министерства общего и профессионального образования РФ («Применение современных систем компьютерной математики в решении фундаментальных задач естествознания») и по гранту Международной Соросовской программы образования в области точных наук (ISSEP, грант Соросовского профессора № p99-962 в области математики).

Авторы благодарят представителя фирмы MathWorks (создавшей систему MATLAB) Наоми Фернандес (Naomy Fernandes), генерального директора корпорации SoftLine (Россия) Игоря Боровикова и сотрудника этой фирмы Бориса Манзона за внимание к своей работе, ее поддержку и предоставление легальных версий системы MATLAB 5.3.1 и MATLAB 6.0 с рядом пакетов расширения для подготовки книг данной серии.

Авторы благодарят также Генерального директора ЗАО «Смоленский Телепорт» Григория Рухамина за предоставление услуг спутникового Интернета в ходе работы над книгой, что позволило посредством прямой оперативной связи с сайтом фирмы MathWorks, Inc. быть в курсе обновлений системы MATLAB и использовать самую свежую информацию. Андрею Пенькову авторы благодарны за отдельные примеры моделирования электронных схем.

Адреса для переписки

С авторами данной книги можно связаться по электронной почте: dyak@keytown.com (Дьяконов В. П.) и kruglov@sci.smolensk.ru (Круглов В. В.). Авторы заранее выражают признательность всем читате-

лям, которые готовы сообщить свое мнение о данной книге. Кроме электронной почты авторов замечания можно направлять по адресу: 214000, Смоленск, ул. Пржевальского, 4, СГПУ. Вы можете отправить свои письма и по адресу издательства, выпустившего книгу.

Связаться с фирмой MathWorks вы можете, посетив сайт www.mathworks.com. С официальным дилером фирмы в России — корпорацией SoftLine — можно связаться через сайт www.softline.ru.

Общие замечания по моделированию систем

Испытание готовых и отлаженных демонстрационных примеров, в том числе описанных в данной книге, может создать у малоопытного читателя иллюзию простоты моделирования. Набрал нужные блоки модели, соединил их друг с другом, запустил режим моделирования и почивай себе на лаврах, списывая результаты и перенося их в свою курсовую или дипломную работу, диссертацию, отчет или статью. Или (как сделано в этой книге) копируя их в буфер обмена Windows и затем редактируя в подходящем графическом редакторе.

Увы, близкая к этой идиллия возможна только для достаточно опытного пользователя, реально поработавшего с тем или иным пакетом расширения не один десяток часов и интуитивно чувствующего правильность (или неправильность) своих действий. Такой пользователь должен знать не только сведения о своей предметной области, но и особенности применения как системы «MATLAB+Simulink», так и используемых им пакетов расширения. В том числе пакетов, описанных в этой книге.

Малоопытный пользователь при переходе к моделированию своих систем или устройств скорее всего столкнется с множеством неожиданных ошибок. Наиболее характерными из них являются: неверное задание параметров моделей, нестыковка входных, выходных и управляющих параметров блоков, несоответствие блоков по типу, ошибочные записи математических выражений, неверный выбор метода моделирования и т. д. Никакая, даже самая обширная фирменная документация не способна отразить все нюансы ошибочного применения системы MATLAB с ее пакетами расширения. Поэтому и авторы данной книги вынуждены ограничиться в этой части описания изложением лишь некоторых «нравоучений» общего характера.

Довольно часто причиной ошибок является несоответствие типов блоков и их входных и выходных параметров. В таких случаях надо предусматривать переходные элементы. Наглядным примером тако-

го рода является переход от тока к напряжению путем включения резистора величиной 1 Ом в цепь тока.

Особенно часто нестыковка наблюдается при совместном использовании блоков из разных пакетов расширения, например из пакетов Power Systems и Simulink. Размерные величины, например используемые в пакете Power System Blockset, зачастую недопустимы для блоков Simulink, использующих безразмерные величины (например, при задании функций).

К сожалению, приходится признать, что популярная и массовая реализация системы MATLAB 5.* не лишена серьезных недостатков. Довольно часто в ней наблюдается нестыковка моделей компонентов, взятых из разных пакетов расширения. В этой связи стоит отметить, что даже беглое знакомство авторов этой книги с системой MATLAB 6.0 показало, что многие из этих недостатков у новейшей версии MATLAB с пакетами расширения уже устранены.

По-видимому, стоит разумно ограничить применение в создаваемых пользователем моделях систем и устройств компонентов из различных пакетов расширения. Как показывает практика, каждый из пакетов расширения имеет довольно широкую сферу применения и позволяет решать множество практически полезных задач. Совместное применение нескольких пакетов расширения системы MATLAB — это уже «высший пилотаж» в ее использовании, и он, как и в случае подготовки летчиков-асов, достигается только долгими тренировками и опытом практического применения данных средств.

Утешением для большинства пользователей MATLAB является то обстоятельство, что система тщательно диагностирует подготовленную модель и запускает ее на исполнение только после устранения всех обнаруженных ошибок. Сообщения об ошибках появляются в специальных окнах системы. Они носят достаточно подробный характер и позволяют принять меры по устранению ошибок и достижению корректной работы создаваемой модели системы или устройства.

В этой связи стоит отметить, что приведенные в книге и в справочной базе данных системы MATLAB примеры нуждаются не просто в просмотре, а во внимательном анализе их назначения и получаемых в результате моделирования результатов. Очень полезно модифицировать примеры, создавая на их основе новые (пусть даже и некорректные) модели и изучая их поведение. Все это и дает опыт, необходимый для полноценного и эффективного применения описанных в данной книге пакетов расширения системы «MATLAB+Simulink». И если это вам интересно, значит, вы являетесь потенциальным

пользователем этой системы и добьетесь нужных вам результатов! Ради этого и написана эта книга.

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу электронной почты компьютерной редакции издательства «Питер» comp@piter.com.

Мы будем рады узнать ваше мнение! Подробную информацию о наших книгах вы найдете на Web-сайте издательства http://www.piter.com.

Глава 1
Пакет System Identification Toolbox

Назначение пакета System Identification Toolbox

Пакет System Identification Toolbox (или просто System Identification) содержит средства для создания математических моделей линейных динамических объектов (систем) на основе наблюдаемых входных/выходных данных. Он имеет удобный графический интерфейс, помогающий организовать данные и создавать модели. Методы идентификации, входящие в пакет, применимы для решения широкого класса задач — от проектирования систем управления и обработки сигналов до анализа временных рядов. Основные свойства пакета следующие:

- простой и гибкий интерфейс;
- предварительная обработка данных, включая фильтрацию, удаление трендов и смещений;
- выбор диапазона данных для анализа;
- эффективные методы авторегрессии;
- возможности анализа отклика систем во временной и частотной областях;
- отображение нулей и полюсов передаточной функции системы;
- анализ невязок при тестировании модели.

Графический интерфейс пакета упрощает как предварительную обработку данных, так и диалоговый процесс идентификации модели. Операции загрузки и сохранения данных, выбора их диапазона, исключения смещений и трендов выполняются с минимальными усилиями и доступны из главного меню.

Представление данных и моделей систем и объектов организовано таким образом, что в процессе интерактивной идентификации пользователь легко может вернуться к предыдущему этапу работы. Для

начинающих пользователей существует возможность просматривать последующие этапы. Специалисту графические средства позволяют отыскать любую из ранее полученных моделей и оценить ее качество в сравнении с другими моделями.

Пакет поддерживает все традиционные виды моделей, включая модели передаточных функций, описания для переменных состояния (как для непрерывного, так и для дискретного времени) и другие, с произвольным числом входов и выходов.

Документация

Электронная документация по пакету представлена в справочной системе в формате HTML, а также в виде файлов формата PDF. Объем последней составляет 274 с. Для просмотра документации в формате HTML необходим броузер, например Internet Explorer 4/5, а в формате PDF — программа Adobe Acrobat Reader.

Рекомендуемая литература

Методы идентификации достаточно полно описаны в следующих монографиях:

1. Льюнг Л. Идентификация систем. Теория для пользователя. — М.: Наука, 1991.
2. Эйкхофф П. Основы идентификации систем управления. — М.: Мир, 1975. — 683 с.
3. Дейч А. М. Методы идентификации динамических объектов. — М.: Энергия, 1979. — 240 с.
4. Гроп Д. Методы идентификации систем. — М.: Мир, 1979. — 302 с.
5. Кашьяп Р. Л., Рао А. Р. Построение динамических стохастических моделей по экспериментальным данным. — М.: Наука, 1983. — 384 с.
6. Трофимов А. И., Егупов Н. Д., Дмитриев А. Н. Методы теории автоматического управления, ориентированные на применение ЭВМ. Линейные стационарные и нестационарные модели. Учебник для вузов. — М.: Энергоатомиздат, 1997. — 656 с.
7. Зеленский К. Х., Игнатенко В. Н., Коц А. П. Компьютерные методы прикладной математики. — К.: Дизайн-В, 1999. — 352 с.

Полную информацию о пакете System Identification можно получить, используя команду `help ident`.

Теоретические сведения

Основные понятия и определения

Человек в своей повседневной практической деятельности широко использует различные модели, которые применяются либо для изучения механизма явлений, происходящих в системах и объектах, либо для прогнозирования их функционирования. Так, в области теории управления широкое распространение получили математические модели объектов управления, используемые как для анализа, так и для синтеза систем регулирования.

Рассмотрим понятие «объект управления». Обычно под *объектом управления* понимается часть окружающего нас мира, поведение которой нас интересует и на которую мы можем целенаправленно воздействовать, то есть управлять ею.

Для облегчения работы с разнообразными объектами управления их разбивают на группы:

- статические объекты;
- динамические объекты;
- линейные объекты;
- нелинейные объекты;
- непрерывные объекты;
- дискретные объекты;
- стационарные объекты;
- нестационарные объекты;
- объекты с сосредоточенными параметрами;
- объекты с распределенными параметрами и т. д.

Под *моделью* обычно понимается выраженная в той или иной форме информация о наиболее существенных характеристиках объекта. По способу представления данной информации выделяют следующие типы моделей:

- *словесные*, или *вербальные* модели;
- *физические модели* (уменьшенные копии реальных объектов, иногда другой физической природы, позволяющие имитировать процессы в исследуемом объекте);
- *математические модели* (информация об исследуемом объекте или системе представляется в виде математических терминов).

В свою очередь, математические модели делятся на:

- графические;
- табличные;
- алгоритмические;
- аналитические.

В частности, аналитические модели представляют собой отражение взаимосвязей между переменными объекта в виде математической формулы или группы таких формул.

Моделирование основано на двух основополагающих признаках:

- на принципе практической ограниченности количества фундаментальных законов природы;
- на принципе подобия, означающем, что явления различной физической природы могут описываться одинаковыми математическими зависимостями.

Процедуру построения модели принято называть *идентификацией*, при этом данный термин обычно относится к построению аналитических математических моделей динамических объектов.

Динамический объект — это объект, выход которого зависит не только от текущего значения входных сигналов, но и от их значений в предыдущие моменты времени. Идентифицируемый объект принято представлять в виде, показанном на рис. 1.1, где t — время; $u(t)$ — контролируемый (иногда управляемый) входной сигнал; $\tilde{y}(t)$ — теоретический выход объекта; $y(t)$ — наблюдаемый выход объекта; $e(t)$ — аддитивная случайная помеха, отражающая действие неучитываемых факторов (шум наблюдения).

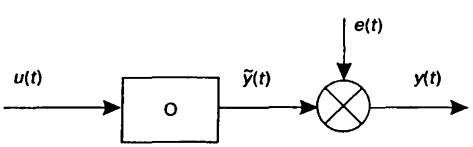


Рис. 1.1. Общее представление идентифицируемого объекта O

Обычно предполагают, что связь между входным и «теоретическим» выходным сигналами задается в виде некоторого оператора ψ *(оператор* — правило преобразования какой-либо функции в другую функцию):

$$\tilde{y}(t) = \Psi[u(t)],$$

при этом наблюдаемый выход объекта может быть описан соотношением

$$y(t) = \Psi [u(t)] + e(t).$$

Цель идентификации: на основании наблюдений за входным $u(t)$ и выходным $y(t)$ сигналами на каком-то интервале времени определить вид оператора, связывающего входной и теоретический выходной сигналы.

Иногда задачу определения математической зависимости можно решить чисто аналитическим путем.

Пример. Рассмотрим простейший сглаживающий фильтр (*RC*-цепь), показанный на рис. 1.2.

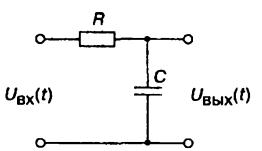


Рис. 1.2. Простейший сглаживающий фильтр

Исходя из известных законов электротехники, для него можно записать:

$$u(t) = RC\frac{dy(t)}{dt} + y(t),$$

где $U_{вх}(t) = u(t)$, $U_{вых}(t) = y(t)$.

Воспользоваться полученным соотношением (дифференциальным уравнением первого порядка) для определения выхода объекта при известном входном сигнале, однако, нельзя до тех пор, пока не установлены численные значения параметров R и C, входящих в модель. Более того, рассматривая другие примеры, можно прийти к выводу, что теоретический анализ с использованием известных физических законов, процессов и явлений, происходящих в объектах, дает возможность установить только структуру модели с точностью до ряда неизвестных параметров. Если такая структура (с точностью до вектора коэффициентов β) известна, то при известном входном сигнале $u(t)$ описание объекта можно представить в виде

$$y(t) = F(\beta, t) + e(t),$$

где F — функция известного вида, зависящая от β и времени t.

Последнее уравнение позволяет после проведения эксперимента, заключающегося в фиксации входного и выходного сигналов на ка-

ком-то интервале времени, провести обработку экспериментальных данных и каким-либо методом (например, методом наименьших квадратов) найти оценку вектора параметров β. Отметим, что при экспериментальном определении параметров модели необходимо обеспечить:

- подбор адекватной структуры модели;
- выбор такого входного сигнала, чтобы по результатам эксперимента можно было найти оценки всех параметров модели.

Наиболее просто задача определения параметров решается для *линейных* объектов, для которых выполняется принцип суперпозиции. Здесь можно выделить два случая:

1. Объект линеен по входному воздействию:
$$\tilde{y}(t) = \Psi[\beta, u_1(t) + u_2(t)] = \Psi[\beta, u_1(t)] + \Psi[\beta, u_2(t)] = \tilde{y}_1(t) + \tilde{y}_2(t).$$
2. Объект линеен по параметрам:
$$\tilde{y}(t) = \Psi[\beta_1 + \beta_2, u(t)] = \Psi[\beta_1, u(t)] + \Psi[\beta_2, u(t)] = \tilde{y}_1(t) + \tilde{y}_2(t).$$

В задачах идентификации под линейными объектами чаще понимают объекты, линейные по входному воздействию.

С учетом изложенного можно уточнить понятие идентификации.

Под *идентификацией динамических объектов* понимают процедуру определения структуры и параметров их математических моделей, которые при одинаковых входном сигнале объекта и модели обеспечивают близость выхода модели к выходу объекта при наличии какого-то критерия качества.

Обычно идентификация — многоэтапная процедура. Основные ее этапы следующие:

1. *Структурная идентификация* заключается в определении структуры математической модели на основании теоретических соображений.
2. *Параметрическая идентификация* включает в себя проведение идентифицирующего эксперимента и определение оценок параметров модели по экспериментальным данным.
3. *Проверка адекватности* — проверка качества модели в смысле выбранного критерия близости выходов модели и объекта.

Отметим, что в связи с многообразием объектов и различных подходов к их моделированию существует множество вариантов задачи параметрической идентификации, классификация которых показана на рис. 1.3.

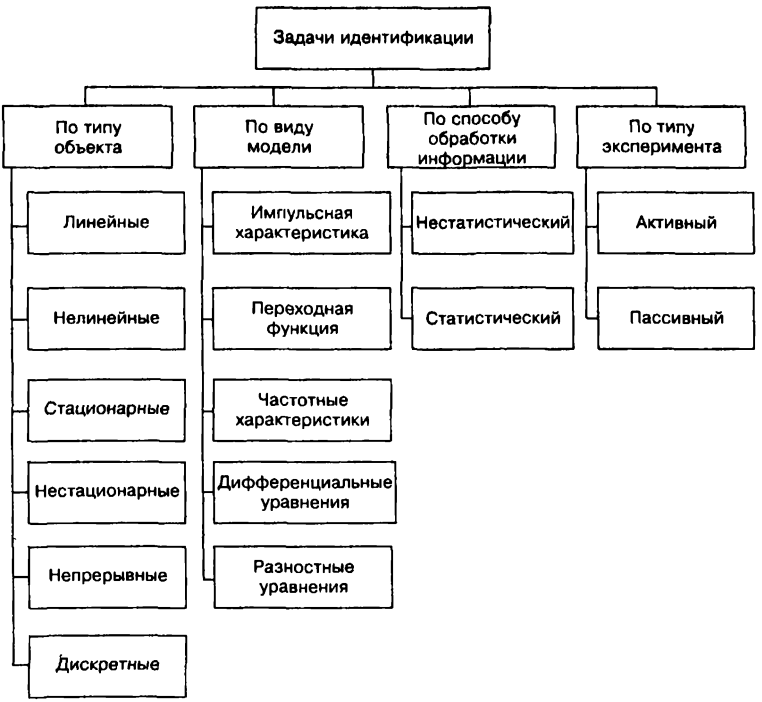


Рис. 1.3. Классификация задач идентификации

Дальнейшее изложение будет относиться к линейным стационарным динамическим объектам.

Виды моделей объектов

Рассмотрим основные виды моделей линейных непрерывных стационарных динамических объектов и их взаимосвязь (действием шума $e(t)$ пока пренебрегаем).

1. **Дифференциальное уравнение.** Наиболее универсальная модель, имеющая форму

$$\sum_{i=0}^{na} a_i y^{(i)}(t) = \sum_{j=0}^{nb} b_j y^{(j)}(t),$$

где na — порядок модели ($na > nb$), a_i и b_j — постоянные коэффициенты (параметры модели), $u^{(j)}(t)$ и $y^{(i)}(t)$ — производные, соответственно, входного и выходного сигналов.

2. **Передаточная функция.** Данная характеристика определяется как отношение преобразований Лапласа выходного и входного сигналов, что с учетом свойств данного преобразования и вышеприведенной формулы дает

$$W(p) = \frac{L\{y(t)\}}{L\{u(t)\}} = \frac{Y(p)}{U(p)} = \frac{\sum_{j=0}^{nb} b_j p^j}{\sum_{i=0}^{na} a_i p^i},$$

где $L\{\bullet\}$ — символ преобразования Лапласа, p — комплексная переменная.

3. **Импульсная характеристика** *(ИХ)* $w(t)$. Под ИХ понимается реакция предварительно невозмущенного объекта (то есть объекта с нулевыми начальными условиями) на входной сигнал в виде δ-функции.

4. **Переходная функция** $h(t)$. Это реакция предварительно невозмущенного объекта на входной сигнал в виде единичного скачка.

   Из теории управления известны следующие соотношения между этими характеристиками:

$$L\{w(t)\} = W(p), \quad w(t) = h'(t), \quad L\{h(t)\} = \frac{W(p)}{p}.$$

   При нулевых начальных условиях связь между выходным и входным сигналами описывается интегралом свертки:

$$y(t) = \int_{-\infty}^{\infty} w(t-\tau)u(\tau)d\tau,$$

   или, в операторной форме:

   $Y(p) = W(p) \cdot U(p)$.

5. **Частотные характеристики.** Частотные характеристики объекта определяются его комплексным коэффициентом передачи $W(j\omega) = W(p)|_{p=j\omega}$, который является Фурье-преобразованием ИХ.

   Модуль комплексного коэффициента передачи $|W(j\omega)| = A(\omega)$ представляет собой, как известно, амплитудно-частотную характеристику (АЧХ) объекта с передаточной функцией $W(p)$, а аргумент $\arg(W(j\omega)) = \varphi(\omega)$ — фазочастотную характеристику (ФЧХ).

   Графическое представление $W(j\omega)$ на комплексной плоскости при изменении частоты ω от 0 до ∞, то есть график амплитудно-фазовой характеристики (АФХ) в полярных координатах в отечественной литературе называется *годографом*, а в англоязычной — *диаграммой Найквиста*.

В теории управления часто используется логарифмическая амплитудно-частотная характеристика (ЛАЧХ), равная $20 \lg |W(j\omega)|$.

6. **Модель для переменных состояния.** При выборе n координат системы (объекта) в качестве переменных ее состояния (такими координатами, например, могут быть выходной сигнал $y(t)$ и $n - 1$ его производных) $x_i(t)$, $i = 1, 2, ..., n$, данную систему можно описать уравнениями для переменных состояния:

$$\mathbf{X}'(t) = \mathbf{A}\mathbf{X}(t) + \mathbf{B}u(t),$$

$$y(t) = \mathbf{C}\mathbf{X}(t) + Du(t),$$

где $\mathbf{X}(t) = [x_1(t), x_2(t), ..., x_n(t)]^T$ — вектор-столбец переменных состояния; **A**, **B**, **C** и D при скалярных $u(t)$ и $y(t)$ — соответственно матрица размера $n \times n$, векторы размера $n \times 1$ и $1 \times n$ и скаляр (при векторных $u(t)$ и $y(t)$ — матрицы соответствующих размеров).

Применение, при нулевых начальных условиях, к последним уравнениям преобразования Лапласа позволяет получить следующее выражение для передаточной функции:

$$W(p) = \mathbf{C}(p\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + D,$$

где **I** — единичная матрица.

Отметим, что все приведенные модели являются эквивалентными, то есть, зная любую из них, можно получить все остальные.

Для объектов, функционирование которых по тем или иным причинам представляется для дискретного времени $t_k = kT$ (в данном случае T — интервал дискретизации), то есть для дискретных объектов, наиболее общим видом описания является разностное уравнение (аналог дифференциального)

$$y_k + a_1 y_{k-1} + ... + a_{na} y_{k-na} = b_1 u_k + b_2 u_{k-1} + b_3 u_{k-2} + ... + b_{nb} u_{k-nb+1},$$

где $y_{k-i} = y[(k - i)T]$, $u_{k-j} = u[(k - j)T]$.

Связь между сигналами может быть отражена также через дискретную свертку:

$$y_k = \sum_{i=0}^{k} w_i u_{k-i},$$

где w_i — ординаты весовой решетчатой функции объекта, или, с использованием аппарата Z-преобразования

$$Y(z) = \sum_{k=0}^{\infty} y_k z^{-k}, \text{ где } z = e^{pT},$$

через дискретную передаточную функцию

$$W(z) = \frac{Y(z)}{u(z)} = \frac{B(z)}{A(z)},$$

которая определяется на основании разностного уравнения после применения к обеим частям этого уравнения Z-преобразования:

$$(1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{na} z^{-na}) Y(z) =$$
$$= (b_1 + b_2 z^{-1} + b_3 z^{-2} + \ldots + b_{nb} z^{-nb+1}) U(z).$$

Заметим, что Z-изображением решетчатой импульсной переходной характеристики является $W(z)$, то есть $Z\{w_i\} = W(z)$.

Отметим далее, что на практике в большинстве случаев измерение непрерывных сигналов производится в дискретные моменты времени, что представляет определенное удобство при последующей обработке данных на ЭВМ. Поставим вопрос: нельзя ли непрерывные объекты, хотя бы приближенно, отображать дискретными моделями?

Ответ на этот вопрос является утвердительным, в частности, возможны следующие способы перехода от непрерывных моделей к дискретным.

1. С применением Z-преобразования со следующей цепочкой переходов:

   $W(p) \to L^{-1}\{W(p)\} = w(t) \to w(kT) = w_k \to W(z) = Z\{w_k\}$.

2. С заменой разностями производных в дифференциальном уравнении, описывающем непрерывный объект,:

   $$\frac{dy(t)}{dt} \approx \frac{y_k - y_{k-1}}{T}; \quad \frac{d^2 y(t)}{dt^2} \approx \frac{y_k - 2y_{k-1} + y_{k-2}}{T^2} \text{ и т. д.}$$

   (данный подход дает приемлемую точность только при малых T).

3. С заменой $p = 2/T \cdot (z-1)/(z+1)$ (приближенный способ, предложенный А. Тастиным и называемый *билинейным преобразованием*), то есть

   $$W(p)\Big|_{p=\frac{2}{T}\frac{z-1}{z+1}} \to W(z).$$

Для дискретных объектов также может быть использовано описание через переменные состояния

$$\mathbf{X}_k = \mathbf{A}\mathbf{X}_{k-1} + \mathbf{B}u_{k-1},$$
$$y_k = \mathbf{C}\mathbf{X}_k + Du_k,$$

переходную функцию и частотные характеристики — так же, как и для непрерывных систем.

Укажем, что множитель $z^{-1} = e^{-pT}$ представляет собой *оператор задержки*, то есть $z^{-1}u_k = u_{k-1}$, $z^{-2}u_k = u_{k-2}$ и т. д. Принимая во внимание данное обстоятельство и обозначая моменты дискретного времени тем же символом t, что и непрерывное время (в данном случае $t = 0, 1, 2, ...$), приведем несколько распространенных моделей дискретных объектов для временной области, учитывающих действие шума наблюдения.

1. Модель авторегрессии AR (AutoRegressive) — считается самым простым описанием:

   $A(z)\, y(t) = e(t)$,

   где $A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + ... + a_{na} z^{-na}$.

2. ARX-модель (AutoRegressive with eXternal input) — более сложная:

   $A(z)\, y(t) = B(z)\, u(t) + e(t)$,

   или, в развернутом виде,

   $y(t) + a_1 y(t-1) + ... + a_{na} y(t-n) =$

   $= b_1 u(t) + b_2 u(t-1) + ... + b_{nb} u(t-m) + e(t)$.

   Здесь и ниже $e(t)$ — дискретный белый шум,

   $B(z) = b_1 + b_2 z^{-1} + ... + b_{bn} z^{-nb+1}$.

3. ARMAX-модель (AutoRegressive-Moving Average with eXternal input — модель авторегрессии скользящего среднего):

   $A(z)\, y(t) = B(z)\, u(t - nk) + C(z)\, e(t)$,

   где nk — величина задержки (запаздывания),

   $C(z) = 1 + c_1 z^{-1} + c_2 z^{-2} + ... + c_{nc} z^{-nc}$.

4. Модель «вход-выход» (в англоязычных источниках такая модель называется «Output-Error», то есть «выход-ошибка», сокращенно OE):

   $$y(t) = \frac{B(z)}{F(z)} u(t - nk) + e(t),$$

   где $F(z) = 1 + f_1 z^{-1} + f_2 z^{-2} + ... + f_{nf} z^{-nf}$.

5. Так называемая модель Бокса—Дженкинса (BJ):

   $$y(t) = \frac{B(z)}{F(z)} u(t - nk) + \frac{C(z)}{D(z)} e(t),$$

   полиномы $B(z)$, $F(z)$, $C(z)$ определены ранее,

   $D(z) = 1 + d_1 z^{-1} + d_2 z^{-2} + ... + d_{nd} z^{-nd}$.

Данные модели можно рассматривать как частные случаи обобщенной параметрической линейной структуры

$$A(z)y(t) = \frac{B(z)}{F(z)}u(t - nk) + \frac{C(z)}{D(z)}e(t),$$

при этом все они допускают расширение для многомерных объектов (имеющих несколько входов и выходов).

6. Модель для переменных состояния (State space):

   $x(t + 1) = \mathbf{A}x(t) + \mathbf{B}u(t),$

   $y(t) = \mathbf{C}x(t) + \mathbf{D}u(t) + v(t),$

   где **A**, **B**, **C**, **D** — матрицы соответствующих размеров, $v(t)$ — коррелированный шум наблюдений.

Возможна и другая (так называемая обновленная, или каноническая) форма представления данной модели:

$$x(t + 1) = \mathbf{A}x(t) + \mathbf{B}u(t) + \mathbf{K}e(t),$$

$$y(t) = \mathbf{C}x(t) + \mathbf{D}u(t) + e(t),$$

где **K** — некоторая матрица (вектор-столбец), $e(t)$ — дискретный белый шум (скаляр).

Методы оценивания

В общем случае оценивание параметров модели заданной структуры проводится, как отмечалось, путем минимизации выбранного критерия качества модели (чаще всего — среднего квадрата рассогласования выходов объекта и его постулируемой модели).

Рассмотрим несколько возможных подходов к такому оцениванию.

Оценивание параметрических моделей

Оценивание параметрических моделей (метод прогноза ошибки — Predictive Error Method, сокращенно PEM) заключается в следующем. Пусть модель исследуемого объекта имеет вид так называемой обобщенной линейной модели

$$y(t) = W(z)\,u(t) + v(t),$$

при этом шум $v(t)$ может быть представлен как

$$v(t) = H(z)\,e(t),$$

где $e(t)$ — дискретный белый шум, $H(z)$ — некоторый полином от z.

Из данных выражений следует, что

$$e(t) = H^{-1}(z)\,[y(t) - W(z)\,u(t)].$$

При выборе в качестве критерия (функции потерь) величины

$$V_N(W,H) = \sum_{t=1}^{N} e^2(t)$$

оценки коэффициентов полиномов модели могут быть найдены в результате решения следующей оптимизационной задачи (в общем случае нелинейной):

$$[\hat{W},\hat{H}] = \operatorname{argmin} \sum_{t=1}^{N} e^2(t).$$

Нахождение такого решения (различными численными методами нелинейной оптимизации), как правило, достаточно сложно и трудоемко.

Заметим, что еще более сложной является подобная процедура оценивания параметров модели для переменных состояния. Однако для ряда частных моделей существуют методы оценивания более простого вида. Рассмотрим их.

Оценивание импульсной характеристики

Предположим, что входной сигнал исследуемого (дискретного) объекта имеет нулевое математическое ожидание и является дискретным белым шумом, то есть имеет автокорреляционную функцию (АКФ)

$$R_u(\tau) = E\{u(t+\tau)u(t)\} = \begin{cases} \lambda, & \text{если } \tau = 0; \\ 0, & \text{иначе.} \end{cases}$$

где $E\{\bullet\}$ — оператор математического ожидания, $\lambda = \text{const}$ — интенсивность сигнала ($\lambda > 0$), и не коррелирован с шумом наблюдений (так что при любых t и τ справедливо $E\{e(t+\tau)u(t) = 0\}$).

Тогда для установившегося режима, исходя из дискретного аналога уравнения свертки, который запишем в форме

$$y(t) = \sum_{i=0}^{\infty} w(i)u(t-i) + e(t),$$

получим

$$R_{yu}(\tau) = E\{y(t+\tau)u(t)\} =$$

$$= E\left\{\sum_{i=0}^{\infty} w(i)u(t+\tau-i)\right\} + E\{e(t+\tau)u(t)\} =$$

$$= \sum_{i=0}^{\infty} w(i)\,E\{u(t+\tau-i)u(t)\} = \sum_{i=0}^{\infty} w(i)\,R_u(\tau-i).$$

Но в силу принятого предположения о виде АКФ входного сигнала в сумме в правой части от нуля отлично только слагаемое, соответствующее $\tau = i$, поэтому окончательно получаем

$$R_{yu}(\tau) = \lambda w(\tau),$$

где $R_{yu}(\tau)$ — взаимная корреляционная функция (ВКФ) выходного и входного сигналов.

Отсюда приходим к оценке ИХ по экспериментальным данным:

$$\hat{w}(\tau) = \frac{1}{\lambda N}\sum_{i=1}^{N} y(t+\tau)u(t),$$

где сумма в правой части с точностью до множителя представляет собой оценку взаимной корреляционной функции сигналов $y(t)$ и $u(t)$, находимую по выборкам $\{y(t)\}$, $\{u(t)\}$, $t = 1, 2, ..., N$.

В случае, когда входной сигнал $u(t)$ является случайным процессом, но не белым шумом, приведенным методом оценивания $w(t)$ можно воспользоваться, если предварительно, с помощью специального формирующего фильтра $\Phi(z)$ (так называемого обеляющего фильтра) преобразовать (хотя бы приближенно) $u(t)$ в сигнал типа белого шума:

$$u_{\Phi}(t) = \Phi(z)u(t).$$

Преобразовав таким же образом $y(t)$, можно воспользоваться (с использованием выборок $\{y_{\Phi}(t)\}$, $\{u_{\Phi}(t)\}$) приведенной выше формулой для нахождения оценки $\hat{w}(t)$.

Заметим, что рассмотренная процедура относится к методам корреляционного анализа.

Оценивание спектров и частотных характеристик

В данном случае объект (дискретный) представляется той же моделью, что и при оценивании параметрических моделей, а входной сигнал полагается случайным процессом с нулевым математическим ожиданием, спектральной плотностью $S_u(\omega)$, некоррелированным с шумом наблюдений $v(t)$, который в данном случае имеет некоторую спектральную плотность $S_v(\omega)$.

Для приведенной модели связи между спектрами сигналов описываются известными соотношениями:

$$S_y(\omega) = |W(e^{i\omega T})|^2 S_u(\omega) + S_v(\omega),$$

$$S_{yu}(\omega) = W(e^{i\omega T})S_u(\omega),$$

где $S_{yu}(\omega)$ — взаимная спектральная плотность сигналов $y(t)$ и $u(t)$, которые можно использовать для нахождения оценок комплексного коэффициента передачи объекта $W(e^{i\omega T})$ и спектра шума $S_v(\omega)$.

Методика оценивания такова:

1. Сначала по данным эксперимента определяют оценки авто- и взаимной корреляционной функций:

$$\hat{R}_u(\tau) = \frac{1}{N}\sum_{i=1}^{N} u(t+\tau)u(t),$$

$$\hat{R}_y(\tau) = \frac{1}{N}\sum_{i=1}^{N} y(t+\tau)y(t),$$

$$\hat{R}_{yu}(\tau) = \frac{1}{N}\sum_{i=1}^{N} y(t+\tau)u(t).$$

2. Затем находят оценки спектральных характеристик:

$$\hat{S}_u(\omega) = \sum_{-\tau=-M}^{M} \hat{R}_u(\tau) g_M(\tau) e^{-j\omega\tau},$$

$$\hat{S}_y(\omega) = \sum_{-\tau=-M}^{M} \hat{R}_y(\tau) g_M(\tau) e^{-j\omega\tau},$$

$$\hat{S}_{yu}(\omega) = \sum_{-\tau=-M}^{M} \hat{R}_{yu}(\tau) g_M(\tau) e^{-j\omega\tau},$$

где $g_M(\tau)$ — так называемое временное окно, а M — его ширина.

3. Наконец, требуемые оценки определяют по соотношениям

$$\hat{W}_N(e^{j\omega T}) = \frac{\hat{S}_{yu}(j\omega)}{\hat{S}_u(\omega)},$$

$$\hat{S}_v(\omega) = \hat{S}_y(\omega) - \frac{\left|\hat{S}_{yu}(j\omega)\right|^2}{\hat{S}_u(\omega)}.$$

Приведенная процедура относится к методам спектрального анализа.

ЗАМЕЧАНИЕ

Найденная оценка $\hat{W}_N(e^{j\omega T})$ комплексного коэффициента передачи является, вообще говоря, чувствительной к небольшим вариациям экспериментальных данных (вызванных, например, ошибками измерений).

Отметим, что в рассматриваемом пакете используются три внутренних вида матричного представления моделей: первый (для временных моделей) — так называемый тета-формат, второй (для частотных характеристик) — частотный формат, и третий — формат нулей и полюсов.

Адекватность моделей устанавливается применением таких критериев, как остаточная сумма квадратов ошибки — функция потерь (Loss fcn) — и так называемый теоретический информационный критерий Акейке (Akaike's Information Theoretic Criterion — AIC) — см. ниже.

Графический интерфейс System Identification Toolbox

Удобный графический интерфейс пакета System Identification запускается из режима командной строки командой ident. В результате ее исполнения появляется диалоговое окно, показанное на рис. 1.4.

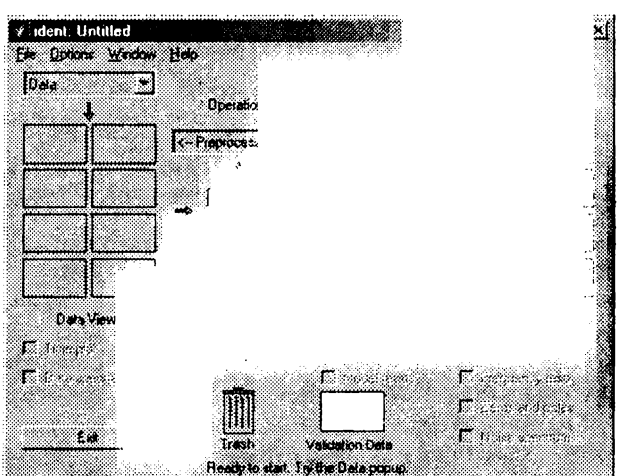


Рис. 1.4. Диалоговое окно графического интерфейса System Identification Toolbox

Работу с интерфейсом лучше всего рассмотреть на примере. В режиме командной строки введем команду

```
» load dryer2
```

Выполнение команды приводит к загрузке в рабочую среду MATLAB массивов данных u2 (входные данные) и y2 (выходные данные), от-

носящихся к такому объекту исследования, как фен (для сушки волос), при этом u2 — тепловая мощность, а y2 — температура нагретого воздуха.

В верхнем левом углу окна выберем в раскрывающемся списке Data (Данные) вариант Import. Это приведет к открытию диалогового окна, показанного на рис. 1.5.

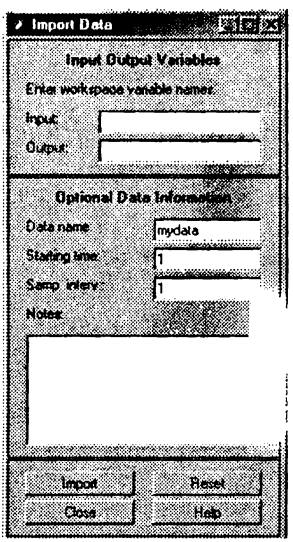


Рис. 1.5. Диалоговое окно импорта данных

Введем в нем имя u2 в поле Input (Вход), y2 — в поле Output (Выход), в поле Samp. inter. (Интервал дискретизации) зададим интервал дискретизации 0.08, в строке Data name (Имя данных) укажем произвольное название, например Фен, а в поле Notes (Примечания) — какой-либо поясняющий текст, что-нибудь вроде

```
%Первый опыт идентификации.
```

Затем нажмем кнопку Import. Далее нажатием кнопки Close (Закрыть) закроем окно импорта данных.

Указанные действия приведут к появлению в верхнем левом углу окна интерфейса значка в виде цветной линии, сопровождаемого надписью «Фен». Это означает, что импортируемые данные введены в среду интерфейса. Сообщения об этих данных также появляются в значках Working Data (Рабочие данные) и Validation Data (Данные для проверки модели).

Проведем исследование исходных данных, для чего установим флажок Time plot (Временной график) в левой нижней части окна интерфейса. Почти сразу появится графическое окно, содержащее графики сигналов $u2(t)$ и $y2(t)$ (рис. 1.6) — при этом график с надписью Output # 1 (Выход № 1) в данном случае относится к $y2(t)$, а график с надписью Input # 1 (Вход № 1) — к $u2(t)$.

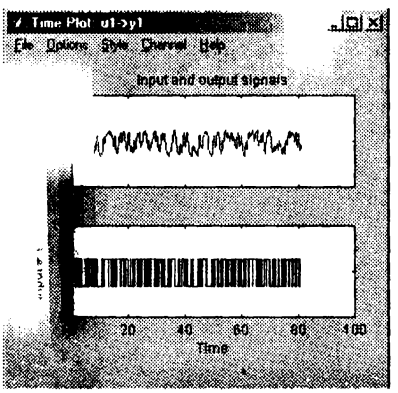


Рис. 1.6. Временные диаграммы сигналов u2(t) и y2(t)

В m-файле данного примера входной и выходной сигналы обозначены через u2 и y2. Однако при их отображении в окне графического интерфейса нумерация сигналов автоматически устанавливается с 1, поэтому в строке заголовка окна рисунка фигурируют обозначения u1 и y1 вместо u2 и y2.

Можно увеличить интересующие нас места графиков, чтобы получше их рассмотреть, — для этого необходимо мышью выделить некоторую прямоугольную область. Выделенный участок сигнала сразу же увеличится в размерах так, что будет занимать все соответствующее окно. Для возвращения к исходному масштабу необходимо дважды щелкнуть мышью на увеличенном графике. А вообще, изменение масштабов графика производится однократным щелчком мыши в области графика: левой кнопкой — для увеличения изображения, правой — для его уменьшения (такое изменение масштабов возможно, только если в меню Style (Стиль) окна Time plot выбран вариант Zoom).

Заметим, что меню окна имеет ряд команд, позволяющих достаточно подробно изучать вид входных и выходных сигналов.

Закроем окно Time plot и вернемся в основное окно интерфейса.

Проведем предварительную обработку сигналов исследуемого объекта, исключив из них постоянную составляющую. С этой целью активизируем раскрывающийся список Preprocess (Предварительная обработка) и выберем в нем вариант Remove means (Удалить среднее).

Результатом операции явится появление в одном из значков Data Views (Вид данных) в левой верхней части окна интерфейса информации о новых данных с именем Фенd. Можно опять активизировать окно Time plot и увидеть, что там появилось изображение двух новых сигналов, отличающихся от исходных отсутствием постоянной составляющей. Для их более удобного просмотра целесообразно воспользоваться командой меню окна Options ▸ Autorange (Автоматическое масштабирование). Чтобы просмотру не мешали исходные графики, можно убрать их, просто щелкнув мышью на значке с их представлением (в левой верхней части окна интерфейса), при этом сразу изменится и масштаб изображения.

Приступим теперь к построению модели, принимая в качестве данных для ее построения данные Фенd. Перетянем их мышью в область Working Data (в центре окна интерфейса). Если мы теперь хотим получить какую-то информацию об этих данных или изменить что-либо (например, их имя), щелкнем мышью дважды на соответствующем значке в группе Data Views. Появится диалоговое окно, показанное на рис. 1.7. Изменим имя данных, например, на ФенМ и закроем данное окно.

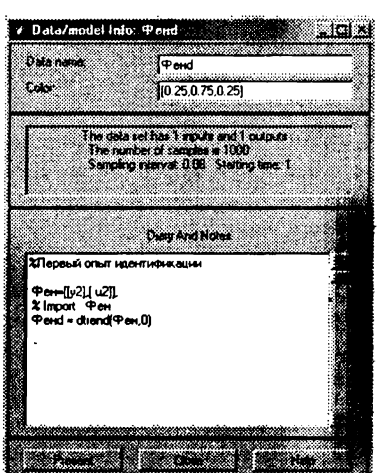


Рис. 1.7. Окно информации о выбранных данных

Активизируем теперь вариант Select Range (Выбор диапазона) из списка Preprocess. Это приведет к появлению окна, показанного на рис. 1.8.

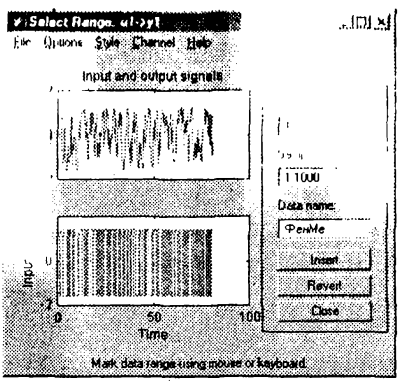


Рис. 1.8. Окно выбора диапазона

Диапазон можно задать либо в текстовом окошке Time span (Временной диапазон), либо с помощью мыши — выделяя прямоугольную область точно так же, как это проводилось для окна Time plot. Любым из этих способов укажем диапазон от 1 до 50 с и нажмем кнопку Insert (Вставить). Результат проделанной операции отразится появлением значка, символизирующего эти (усеченные) данные (с именем ФенМе) в окне интерфейса.

Повторим операцию задания диапазона с целью формирования данных для проверки модели — например, из оставшейся части начального диапазона, в результате чего появится еще один значок данных с именем ФенMv. Закроем окно выбора диапазона.

С помощью мыши перетащим данные ФенМе в область Working Data, а данные ФенMv — в область Validation Data (заметим, что если на каком-то этапе работы с данными допущена ошибка и мы хотим удалить неправильные данные, необходимо просто перетащить их мышью на значок Trash (Мусор) в центральной нижней части окна интерфейса, и они исчезнут).

Собственно, теперь можно приступать к нахождению оценки модели выбранного вида. Для этого вначале необходимо указать этот вид.

Начнем с оценивания переходной функции объекта. В раскрывающемся списке Estimate (Оценивание) выберем вариант Correlation Model (Корреляционная модель), что приведет к появлению соответствующего диалогового окна, показанного на рис. 1.9.

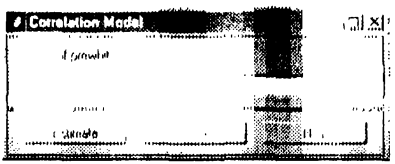


Рис. 1.9. Диалоговое окно корреляционной модели

Вообще-то в появившемся окне можно кое-что задать дополнительно, но в данном случае ничего не будем делать, только нажмем в нем кнопку Estimate (Оценить), после чего окно можно закрыть.

А в основном окне интерфейса в его правой части (Model Views — Вид моделей) при этом появится значок с надписью crad, означающий, что выбранная модель построена. Чтобы увидеть результат, установим флажок Transient resp (transient response — переходная функция) и увидим в появившемся окне (рис. 1.10) график переходной функции исследуемого объекта, найденной описанным выше корреляционным методом.

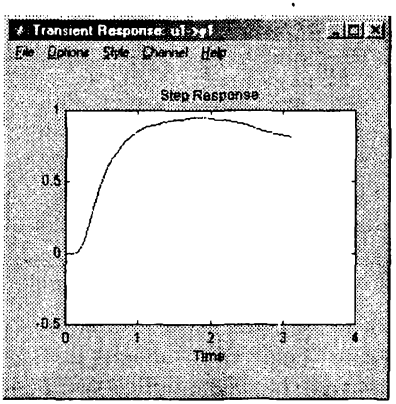


Рис. 1.10. Оценка переходной функции исследуемого объекта, найденная корреляционным методом

Можно провести детальное изучение данного графика, пользуясь доступными командами меню графического окна или с помощью мыши — так же, как это было проделано ранее при изучении входного и выходного сигналов. Но оценить качество полученной модели можно только в процессе ее сравнения с моделями других видов.

Укажем, что используя команду Options | Impulse response меню данного окна, вместо графика переходной функции можно вывести график ИХ.

Оценим модель следующего вида — частотные характеристики объекта. Выберем в списке Estimate вариант Spectral model и повторим только что проделанные операции по нахождению оценки модели. В результате в основном окне интерфейса (в его правой части — Model Views) появится еще один значок с именем spad, символически представляющий модель в виде частотных характеристик, найденных спектральным методом. Результат отображается активизацией графического окна Frequency resp (frequency response — частотные характеристики), показанного на рис. 1.11.

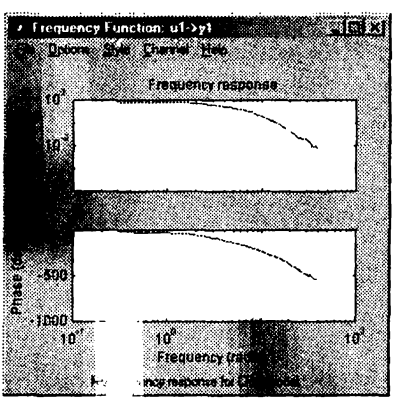


Рис. 1.11. Частотные характеристики, найденные спектральным методом

Перейдем теперь к оцениванию параметрических моделей, выбрав в списке Estimate вариант Parametric models. Данный выбор приведет к открытию диалогового окна задания структуры модели (рис. 1.12).

По умолчанию пользователю предлагается модель типа ARX (см. выше) с параметрами $na = 4$, $nb = 4$, $nk = 1$. Можно согласиться или не согласиться с данными значениями. В последнем случае параметры можно изменить непосредственно в строке окна или с помощью редактора порядка модели (Order editor), вызываемого нажатием соответствующей кнопки. Можно вообще выбрать другую параметрическую модель, воспользовавшись раскрывающимся списком в верхней части окна (возможный выбор — модели типа ARX, ARMAX, OE, BJ, State Space и модель, задаваемая пользователем).

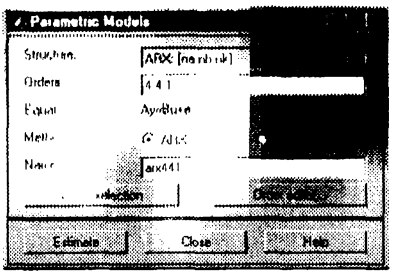


Рис. 1.12. *Диалоговое окно задания структуры модели*

В рассматриваемом примере сохраним значения по умолчанию (ARX, 4, 4, 1) и нажмем кнопку Estimate (Оценить). Результатом действия будет появление значка модели с названием arx441.

Воспользовавшись далее возможностями редактора порядка модели (Order editor), зададим теперь ARX-модель с параметрами $na = 2$, $nb = 2$, $nk = 3$ (последнее означает наличие в объекте запаздывания с величиной $nk \times T = 3 \times 0.08 = 0.24$ с) и повторим операцию оценивания. Это приведет к появлению значка еще одной модели с именем arx223 (рис. 1.13).

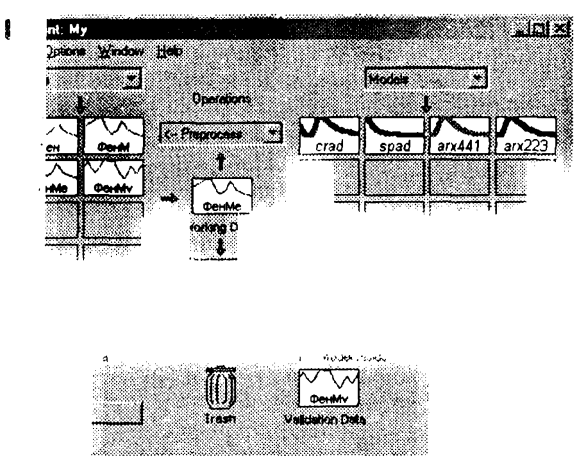


Рис. 1.13. Окно графического интерфейса с результатами построения моделей

Для сравнения полученных моделей активизируем окно Transient resp (рис. 1.14). В нем будут отображены графики переходных процес-

сов, соответствующие трем построенным моделям — корреляционной (crad), arx441 и arx223.

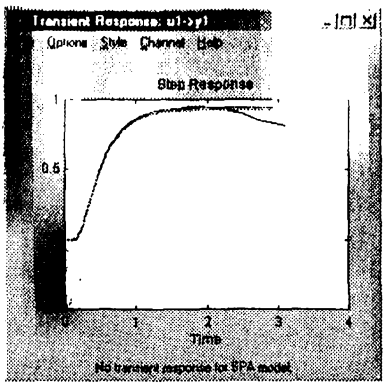


Рис. 1.14. Графики оценок переходного процесса для трех построенных моделей

Заметим, что однократный щелчок левой кнопкой мыши на графике модели в ее значке приведет к исчезновению соответствующего графика переходного процесса; повторный щелчок восстанавливает изображение. Двойной щелчок на значке приводит к открытию окна с информацией о модели (рис. 1.15).

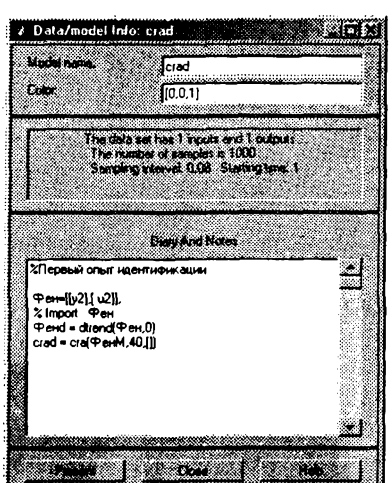


Рис. 1.15. Пример окна с информацией о модели

Вернемся к окну Transient resp. Оставляя в нем только по одному графику (то есть убирая остальные, как это описано выше), активизируем команду его меню Options ▸ Show 99% confidence intervals (Показать 99%-й доверительный интервал). Просмотрим последовательно переходные процессы с доверительными интервалами для трех построенных моделей — crad, arx441 и arx332. Сравнение покажет, что первая модель является наименее точной, а вторая и третья дают примерно одинаковые результаты.

Какую же модель выбрать в качестве итоговой? При прочих равных условиях, очевидно, более простую, а таковой здесь является (по числу оцениваемых коэффициентов) модель arx332.

Есть и еще один способ сравнения параметрических моделей — путем активизации графического окна Model output (соответствующий флажок расположен в средней нижней части окна интерфейса). Вид окна Model output для выбранных моделей arx441 и arx332 приведен на рис. 1.16.

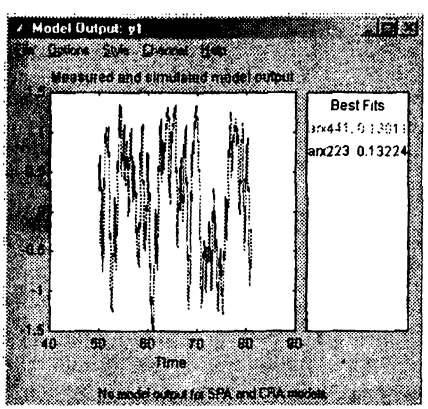


Рис. 1.16. Окно сравнения выходов моделей

В левой части данного окна приведены выходы объекта и указанных моделей, а в правой — величины, отражающие меру совпадения (среднеквадратичное рассогласование) экспериментальных и прогнозируемых данных. Как видно, несколько точнее оказывается модель arx441 (хотя и ненамного).

Возвратимся к процедуре оценивания параметрических моделей. Выберем структуру вида State Space. В появляющемся диалоговом окне задания структуры модели по умолчанию предлагается модель 4-го

порядка (используется обновленная форма модели для переменных состояния — см. выше). Можно, конечно, с этим согласиться, но можно избрать поисковый метод определения наилучшей структуры (порядка) модели, указав в строке Orders (Порядок) диапазон возможных значений порядка модели, например 1:10. Проделав это и нажав кнопку Estimate, в качестве результата выполненного действия получим открывшееся окно выбора порядка модели (рис. 1.17).

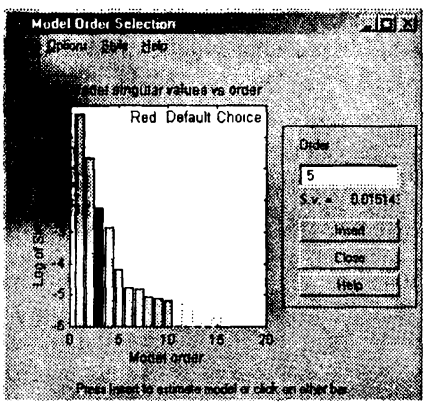


Рис. 1.17. Окно выбора порядка модели

График показывает вклад n-й составляющей вектора переменных состояния в выходной сигнал модели (значения n указаны по оси абсцисс). Целесообразно выбирать значение n таким, чтобы столбцы, соответствующие составляющим с номерами больше n, имели бы намного меньшую высоту, чем столбцы, расположенные слева от них. В рассматриваемом случае можно остановиться на $n = 4$.

Возвратимся далее в окно задания структуры модели и нажмем кнопку Estimate. Результатом явится появление значка новой модели с именем n4s4. Данную модель можно исследовать так же, как и полученные ранее модели.

Вообще-то для анализа моделей графический интерфейс пакета System Identification представляет весьма значительные возможности, среди которых отметим только возможность просмотра переходной функции модели с помощью программы LTI Viewer (детальное описание которой приведено в главе 3, посвященной пакету Control Systems).

Сохранение построенных моделей производится в два этапа:

- сначала модель вводится в рабочее окружение системы MATLAB (перетаскиванием значка модели в область To Workspace в центре рабочего окна интерфейса), при этом модель будет фигурировать в рабочем пространстве MATLAB под тем же именем, что и в среде интерфейса;
- затем модель сохраняется командой сохранения в режиме командной строки (как любая переменная MATLAB).

Отметим, что можно сохранить все рабочее пространство интерфейса (при его закрытии даже появится соответствующая подсказка) в виде файла с расширением (по умолчанию) sid, при этом в следующем сеансе работы можно загрузить все полученные результаты. Сохраненное рабочее пространство называется сессией.

Работа с пакетом в режиме командной строки

Возможности работы в режиме командной строки

Пакет System Identification располагает большим набором функций (команд), исполняемых из командной строки MATLAB и позволяющих, в принципе, решать задачи идентификации, не используя рассмотренную программу графического интерфейса. Все функции (команды) делятся на следующие группы:

- графического интерфейса;
- обработки и преобразования данных;
- отображения модели;
- непараметрического оценивания;
- параметрического оценивания;
- итерационного параметрического оценивания;
- задания структуры модели;
- изменения и уточнения структуры модели;
- выбора структуры модели;
- преобразования модели;
- извлечения информации о модели;
- проверки адекватности модели;
- прочие;
- демонстрации возможностей пакета.

Команды графического интерфейса

Данная группа содержит две команды: ident и midprefs.

Команда ident — команда открытия рассмотренного выше графического интерфейса пользователя. Она записывается в виде:

```
ident
ident(session,directory)
```

Команда открывает окно графического интерфейса. При второй форме записи открытие интерфейса сопровождается загрузкой сессии из файла с именем session, находящегося в директории directory. Примеры:

```
ident('iddata1.sid')
ident('mydata.sid','\matlab\data\cdplayer\')
```

Команда midprefs задает (изменяет) директорию для файла midprefs.mat, хранящего информацию о начальных параметрах графического интерфейса пользователя при его открытии. Пример:

```
midprefs('/home/ljung/matlab/')
```

Функции обработки и преобразования данных

В данную группу входят следующие четыре функции.

Функция detrend

Функция detrend удаляет тренд из набора данных. Она используется в виде:

```
zd = detrend(z)
zd = detrend(z,o,brkp)
```

Ее аргументы: z — матрица исходных данных, в которой столбцы соответствуют векторам данных; o (по умолчанию 0) — параметр, определяющий порядок тренда (нулевой или первый); если аргумент brkp не определен, из вектора данных вычитается линейная функция, соответствующая тренду; если данный аргумент определен, из исходных данных вычитается «ломаная линия» с точками разрыва производной при элементах, номера которых содержатся в brkp (только для линейных трендов). Возвращаемая величина: zd — матрица преобразованных данных.

В следующем примере:

```
zd(:,1) = detrend(z(:,1),1,119);
zd(:,2) = detrend(z(:,2));
```

удаляется V-образный тренд из данных первого столбца с пиком на 119-м элементе и тренд нулевого порядка (постоянная составляющая) — для элементов второго столбца матрицы исходных данных.

Функция idfilt

Функция idfilt

```
zf = idfilt(z,ord,Wn)
[zf,thf] = idfilt(z,ord,Wn,hs)
```

возвращает данные, фильтрованные с помощью фильтра Баттерворта. Эта функция создает фильтр Баттерворта порядка ord и осуществляет фильтрацию данных (по столбцам) матрицы данных z = [y u]. Если аргумент hs не задан и Wn содержит только один элемент, создается фильтр низких частот (ФНЧ) с частотой среза Wn.

Если hs ='high', создается фильтр верхних частот (ФВЧ) с частотой среза Wn.

Если Wn = [Wnl Wnh], то есть является двухэлементным вектором, и аргумент hs не определен, создается полосовой фильтр (ПФ) с полосой пропускания от Wnl до Wnh. Если же в этом случае задан аргумент hs = 'stop', создается заграждающий полосовой фильтр. Порядок фильтра в этих случаях — 2*ord.

Отметим, что частоты Wnl и Wnh задаются в относительных единицах, в долях от так называемой *частоты Найквиста*, определяемой как $\omega = \pi/T$ (или $f = 1/2T$), где T — выбранный интервал дискретизации.

Возвращаемые величины: zf — матрица отфильтрованных данных, thf — описание созданного фильтра в так называемом тета-формате (см. ниже).

Функция idinput

Функция idinput в формах

```
u = idinput(N)
u = idinput(N,type,band,levels)
u = idinput(N,'sine',band,levels,auxvar)
```

генерирует входной идентифицирующий сигнал. Она создает несколько разновидностей входного сигнала, наиболее часто используемых в задачах идентификации при возможности организации активного эксперимента. Аргумент N задает количество значений сигнала. Строковая переменная type задает тип сигнала и может принимать значения (по умолчанию — 'rbs'):

- type = 'rs' — задание случайного гауссова сигнала;
- type = 'rbs' — задание случайного бинарного (двоичного) сигнала;
- type = 'prbs' — задание псевдослучайного двоичного сигнала (ПСДС);

- type = 'sine' — задание полигармонического сигнала (в виде суммы синусоид).

Двухэлементный вектор-строка band = [wlow, whigh] задает диапазон частот сигнала (для сигналов типа 'rs', 'rbs' и 'sine'), при этом wlow — нижняя частота, а whigh — верхняя частота (частоты задаются в долях от частоты Найквиста — см. выше). Белый шум, таким образом, характеризуется значением band = [0, 1], которое является и значением по умолчанию.

Для сигнала типа ПСДС ('prbs') используется значение band = [twologp, M], при этом 1/M — длительность интервала постоянства сигнала, а значение 2^twologp-1 определяет период ПСДС. Максимальное значение периода соответствует twologp=18 (или twologp=0); в этом случае по умолчанию band = [0, 1].

Двухэлементный вектор levels = [minu, maxu] задает минимальный и максимальный уровни для сигналов типа 'rs', 'rbs' и 'sine'. Для сигнала типа 'rs' величина minu имеет значение, равное математическому ожиданию минус среднее квадратическое (или стандартное) отклонение, а maxu — математическому ожиданию плюс среднее квадратическое отклонение.

Для сигнала типа 'sine' возможно задание четвертого аргумента — auxvar = [no_of_sinusoids, no_of_trials], где no_of_sinusoids — число гармоник, образующих сигнал, а no_of_trials определяет, сколько итераций должно быть сделано, чтобы минимизировать амплитуду сигнала путем задания различным синусоидам произвольных начальных фаз. Значение по умолчанию — auxvar = [10, 10].

Возвращаемая величина — N-элементный вектор u.

Функция idresamp

Функция idresamp, записываемая в формах

```
zr = idresamp(z,R)
[zr, R_act] = idresamp(z,R,filter_order,tol)
```

выполняет преобразование сигнала путем изменения интервала его дискретизации.

Аргументы: z — матрица исходных данных (каждый столбец соответствует вектору данных), R — коэффициент изменения интервала дискретизации; при R>1 происходит прореживание исходного сигнала, при R<1 — его интерполяция. Допустимо любое положительное значение R, но оно будет заменено достигнутым приближением R_act.

- `filter_order` — порядок фильтра, который будет использован для предварительной фильтрации исходного сигнала (по умолчанию — 8);
- `tol` — допустимое относительное отклонение `R_act` от `R`. По умолчанию — 0.1.

Возвращаемые величины — матрица преобразованных сигналов `zr` и достигнутое значение `R_act`.

Функции отображения модели

Эта группа образована шестью функциями.

Функция bodeplot

`bodeplot` — функция отображения логарифмических частотных характеристик (диаграмма Боде):

```
bodeplot(g)
bodeplot([g1 g2 ... gn])
bodeplot(g,sd,C,mode)
```

Она имеет следующие аргументы:

- `g, g1, g2, ..., gn` — имена оценок частотных характеристик;
- `sd` — при `sd>0` к графикам добавляются соответствующие доверительные коридоры (по умолчанию `sd=0`);
- аргумент `C` задает выводимые характеристики:
  - `C = 'A'` — выводится только график АЧХ;
  - `C = 'P'` — выводится только график ФЧХ;
  - `C = 'B'` — выводятся оба графика;
- для получения всех графиков на одной и той же диаграмме задается `mode = 'same'`.

Возвращаются графики частотных характеристик с логарифмическими масштабами для АЧХ и оси частот.

Функция ffplot

Функция `ffplot` имеет такое же назначение, как предыдущая. Записывается в виде:

```
ffplot(g)
ffplot([g1 g2 ... gn])
ffplot(g,sd,C,mode)
```

Аргументы — как у предыдущей функции. Возвращаются графики частотных характеристик с логарифмическим масштабом для АЧХ, но линейным для оси частот (единицы частоты — Гц).

Функция idplot

Функция idplot графически отображает входные и выходные данные:

```
idplot(z)
idplot(z,int,T,ny,pc)
```

Аргументы функции:

- z = [y x] — матрица данных; первые столбцы соответствуют выходам, последние — входам объекта;
- ny — количество выходов, по умолчанию ny=1;
- int — вектор-строка, задающий количество точек графика. По умолчанию выводятся все исходные данные;
- T — интервал дискретизации (для задания правильного масштаба времени);
- pc — строковая переменная, определяющая вид представления входного сигнала (входных сигналов). При pc = 'pc' (значение по умолчанию) данный сигнал полагается кусочно-постоянным между моментами дискретизации, при pc = 'li' — линейно изменяющимся.

Функция nyqplot

Функция nyqplot осуществляет построение годографа (диаграммы Найквиста):

```
nyqplot(g)
nyqplot([g1 g2 ... gn])
nyqplot(g,sd,mode)
```

Аргументы g, g1, g2, ..., gn, sd, mode — такие же, как у функции bodeplot.

Возвращается график АФХ (годограф).

Функция present

Функция present возвращает информацию о модели (представленной в так называемом тета-формате):

```
present(th)
```

Здесь th — имя (идентификатор) параметрической модели. Возвращаются коэффициенты полиномов модели вместе с их стандартными отклонениями, значения функции потерь и критерия конечной ошибки прогноза, а также сообщение о методе оценивания модели.

Функция zpplot

Функция zpplot возвращает графическое изображение нулей и полюсов моделей объекта:

```
zpplot(zepo)
zpplot(zpform(zepo1,zepo2, .... zepon))
zpplot(zepo,sd,mode,axis)
```

Аргументы этой функции:

- zepo — матрица, содержащая информацию о нулях и полюсах модели объекта и получаемая с помощью функции th2zp (см. ниже);
- zpform(zepo1,zepo2, ..., zepon) — функция объединения информации о нулях и полюсах нескольких моделей одного и того же объекта с одними и теми же входами;
- при sd>0 к нулям и полюсам добавляются соответствующие доверительные области (по умолчанию sd=0);
- нули и полюсы разных моделей представляются на одном и том же графике; для дискретных моделей на графике также изображается единичная окружность;
- Когда в zepo содержится информация о нескольких входах, характер отображения определяется строковой переменной mode: при mode = 'sub' открываются одновременно графики, относящиеся ко всем входам; при mode = 'same' информация представляется в одном графическом окне, при этом переход от изображения нулей и полюсов для одного входа к изображению нулей и полюсов для другого входа производится при нажатии клавиши Enter; mode = 'sep' — то же, но предыдущий график убирается до обработки следующего входа. По умолчанию mode = 'sub';
- axis = [x1 x2 y1 y2] — задает масштабы осей. Задание axis = m — то же самое, что и axis = [-m m -m m].

Функции непараметрического оценивания

В данную группу входят функции covf, cra, etfe и spa. Рассмотрим их.

Функция covf

Функция covf выполняет расчет авто- и взаимных корреляционных функций совокупности экспериментальных данных:

```
R = covf(z,M)
R = covf(z,M,maxsize)
```

Аргументы:

- z — матрица данных размером N×nz, каждый столбец которой соответствует входному или выходному сигналу (обычно z=[y u]);
- M — максимальная величина дискретного аргумента, для которой рассчитываются корреляционные функции, минус единица;

- maxsize — параметр, определяющий максимально допустимый размер матриц.

Возвращаемая величина — матрица R размером $nz^2 \times M$ с элементами

$$R[i+(j-1)nz,\ k+1] = \hat{R}_{ij}(k) = \frac{1}{N}\sum_{t=1}^{N} z_i(t) z_j(t+k).$$

Функция cra

Функция cra определяет оценку ИХ методом корреляционного анализа для одномерного (один вход — один выход) объекта:

```
cra(z);
[ir,R,cl] = cra(z,M,na,plot);
cra(R);
```

Аргументы:

- z — матрица экспериментальных данных вида z = [y u], где y — вектор-столбец, соответствующий выходным данным, u — вектор-столбец входных данных;
- M — максимальное значение дискретного аргумента, для которого производится расчет оценки ИХ, по умолчанию M=20;
- na — порядок модели авторегрессии (степени многочлена $A(z)$), которая используется для расчета параметров «обеляющего» фильтра $\Phi(z)$, по умолчанию na=10. При na=0 в качестве идентифицирующего используется непреобразованный входной сигнал;
- plot — plot=0 означает отсутствие графика, plot=1 (по умолчанию) — график полученной оценки ИХ вместе с 99%-м доверительным коридором, plot=2 — выводятся графики всех корреляционных функций.

Возвращаемые величины: ir — оценка ИХ (вектор значений); cl — 99%-й доверительный коридор для оценки ИХ; R — матрица, элементы первого столбца которой — значения дискретного аргумента, элементы второго столбца — значения оценки автокорреляционной функции выходного сигнала (возможно, отфильтрованного), элементы третьего столбца — значения оценки автокорреляционной функции входного сигнала (возможно, «обеленного»), элементы четвертого столбца — значения оценки взаимной корреляционной функции.

Функция spa

Функция spa возвращает частотные характеристики одномерного объекта и оценки спектральных плотностей его сигналов для обобщенной линейной модели объекта (возвращая модель объекта в так называемом частотном формате):

```
[g,phiv] = spa(z)
[g,phiv,z_spe] = spa(z,M,w,maxsize,T)
```

Аргументы:

- `z` — матрица исходных данных — как в рассмотренных выше функциях;
- `M` — ширина временного окна (см. выше), по умолчанию `M = min(30, length(z)/10)`, где `length(z)` — число строк матрицы `z`;
- `w` — вектор частот, для которых производится расчет частотных характеристик, по умолчанию `[1:128]/128*pi/T`;
- `T` — интервал дискретизации;
- `maxsize` — параметр, определяющий максимальный размер матриц, создаваемых в процессе вычислений (оптимальный выбор этого значения позволяет добиться максимальной скорости расчетов).

Возвращаемые величины:

- `g` — оценка $W(e^{j\omega T})$ в частотном формате;
- `phiv` — оценка спектральной плотности шума $v(t)$;
- `z_spe` — матрица спектральных плотностей входного и выходного сигналов.

Рассмотрим пример. Пусть исходные данные содержатся в файле `dryer2.mat`. Воспользуемся функцией `spa` для нахождения оценок амплитудно- и фазочастотных характеристик объекта с выводом результата в форме графиков.

```
» %Загрузка данных
» load dryer2
» z=[y2 u2];
» g = spa(z); %Оценивание модели
» bodeplot(g) %Построение диаграммы Боде
```

Результат представлен на рис. 1.18 (оценка АЧХ построена в логарифмическом масштабе).

В продолжение примера ниже приведены функции, обеспечивающие вывод графиков АЧХ, ФЧХ и $S_v(\omega)$ вместе с доверительными коридорами шириной в три среднеквадратических отклонения, а также сами графики (рис. 1.19 и 1.20).

```
» w = logspace(-2,pi,128);
» [g,phiv] = spa(z,[],w);
» % (пустая матрица означает значение по умолчанию)
» bodeplot([g phiv],3)
```

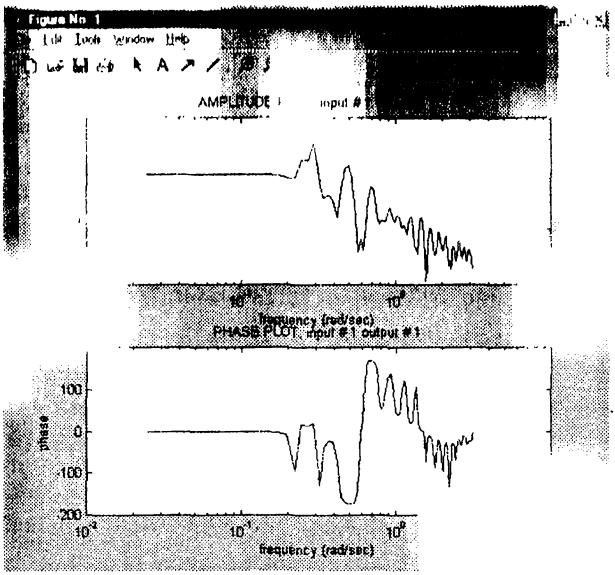


Рис. 1.18. Оценки АЧХ и ФЧХ к примеру использ

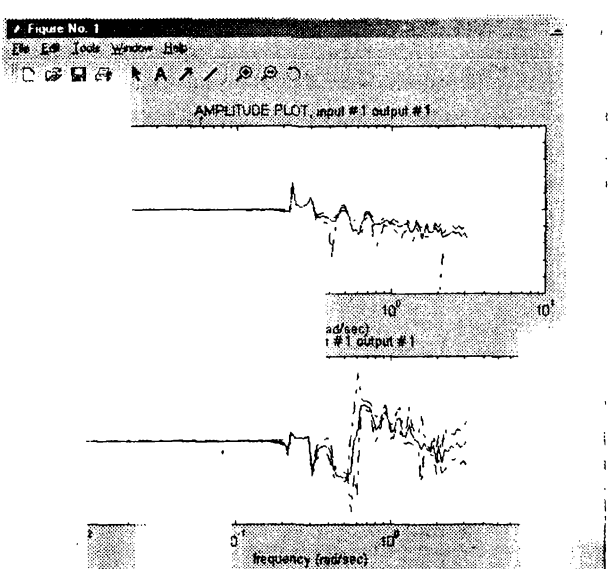


Рис. 1.19. Оценки АЧХ и ФЧХ вместе с соответствующими доверительными коридорами

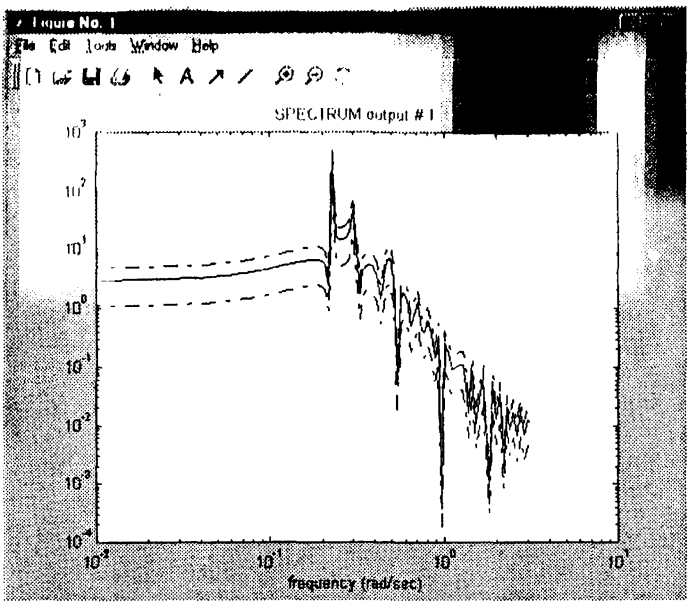


Рис. 1.20. График оценки S(ω) вместе с доверительным коридором

Функция etfe

Функция `etfe` так же, как предыдущая функция, возвращает оценку дискретной передаточной функции для обобщенной линейной модели (см. выше) одномерного объекта в частотной форме. Рекомендована к использованию для узкополосных объектов. Записывается в виде:

```
g = etfe(z)
g = etfe(z,M,N,T)
```

Аргументы:

- `z` — как для предыдущей функции;
- `M` — как для предыдущей функции;
- `N` — определяет диапазон частот для расчета (эта величина должна быть степенью 2) значений частотных характеристик согласно формуле `w = [1:N]/N*pi/T`. По умолчанию `N=128`;
- `T` — интервал дискретизации (по умолчанию `T=1`).

Возвращаемая величина $g = \hat{W}(e^{j\omega T})$.

В продолжение предыдущего примера (с его исходными данными) проведем сравнение результатов использования функций spa и etfe. Соответствующая программа и графики приведены ниже.

```
» ge = etfe(z);
» gs = spa(z);
» bodeplot([ge gs])
```

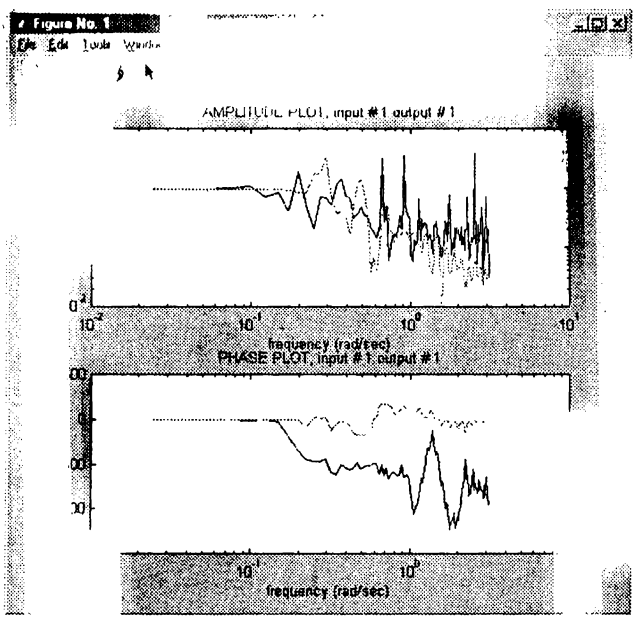


Рис. 1.21. Оценки АЧХ и ФЧХ объекта, полученные с помощью функций spa и etfe

Сравнение графиков на рис. 1.21 показывает, что более гладкими (и, наверное, более реалистичными) выглядят оценки частотных характеристик, полученные с помощью функции spa (на рисунке они представлены более светлыми кривыми).

Функции параметрического оценивания

Группа образована следующими 11 функциями.

Функция ar

Функция ar оценивает параметры модели авторегрессии (AR), то есть коэффициенты полинома $A(z)$, при моделировании скалярных временных последовательностей:

```
th = ar(y,n)
[th,refl] = ar(y,n,approach,win,maxsize,T)
```

Аргументы:

- `y` — вектор-столбец данных, содержащий `N` элементов;
- `n` — порядок модели (число оцениваемых коэффициентов);
- аргумент `approach` (строковая переменная) определяет метод оценивания:
  - `'fb'` — прямой-обратный метод (разновидность метода наименьших квадратов). Используется по умолчанию;
  - `'ls'` — метод наименьших квадратов (МНК);
  - `'yw'` — метод Юла—Уокера;
  - `'burg'` — метод Бэрга (комбинация МНК с минимизацией гармонического среднего);
  - `'gl'` — метод с использованием геометрического среднего.

  Если любое из данных значений заканчивается нулем (например, `'burg0'`), то вычисление сопровождается оцениванием корреляционных функций;
- Аргумент `win` (строковая переменная) используется в случае отсутствия части данных:
  - `win = 'now'` — используются только имеющиеся данные (используется по умолчанию — за исключением случая `approach = 'yw'`);
  - `window = 'prw'` — отсутствующие начальные данные заменяются нулями, так что суммирование начинается с нулевого момента времени;
  - `window = 'pow'` — последующие отсутствующие данные заменяются нулями, так что суммирование расширяется до момента времени `N+n`;
  - `window = 'ppw'` — и начальные, и последующие отсутствующие данные заменяются нулями. Используется в алгоритме Юла—Уокера;
- Аргумент `maxsize` определяет максимальную размерность задачи;
- `T` — интервал дискретизации.

Возвращаемые величины:

- `th` — информация о модели в так называемом тета-формате (внутреннем матричном формате представления параметрических моделей пакета System Identification);
- `refl` — информация о коэффициентах и функции потерь.

Пример использования функции ar для оценивания модели авторегрессии 4-го порядка приведен ниже.

```
» load dryer2; %Загрузка данных
» y=[y2(1:300)];
» th=ar(y,4); %Оценивание AR-модели
» present(th) %Вывод информации о модели
This matrix was created by the command AR     on 1/19 2001 at 9:8
Loss fcn: 0.010782   Akaike's FPE: 0.011074 Sampling interval 1
The polynomial coefficients and their standard deviations are
A =
    1.0000   -2.1716    1.5697   -0.3183   -0.0790
         0    0.0579    0.1379    0.1384    0.0583
```

Матрица A содержит коэффициенты полинома $A(z)$ (первая строка) и их стандартные отклонения (вторая строка). Текст сообщает о величине функции потерь и о значении интервала дискретизации. В данном случае он (по умолчанию) равен 1.

Функция armax

Функция armax оценивает параметры ARMAX-модели:

```
th = armax(z,nn)
th = armax(z,nn,'trace')
[th, iter_info] = armax(z,nn,maxiter,tol,lim,maxsize,T,'trace')
```

Аргументы функции:

- z = [y u] — матрица экспериментальных данных; в случае многомерного входа u представляет собой матрицу с числом столбцов, равным числу входов;
- nn = [na nb nc nk] — задаваемые параметры ARMAX-модели (степени полиномов и величина задержки);
- 'trace' — наличие этого аргумента задает выдачу информации о ходе итерационного процесса вычислений (трассировка);
- maxiter — определяет максимальное число итераций при решении оптимизационной задачи, по умолчанию maxiter=10;
- tol — определяет заданную погрешность вычислений при поиске минимума, по умолчанию tol=0.01;
- lim — так называемый критерий робастности, по умолчанию lim=1.6. При lim=0 используется обычный МНК;
- maxsize — определяет максимальную размерность задачи, по умолчанию maxsize=4096;
- T — интервал дискретизации, по умолчанию равен 1.

Возвращаемые величины:

- th — информация о модели в тета-формате;
- iter_info = [номер последней итерации, значение функции потерь на данной итерации] — информация о выполнении итерационного процесса оценивания модели.

Использование функции armax иллюстрируется следующим примером:

```
» load dryer2 %Загрузка данных
» z=[y2 u2];
» th=armax(z,[2 2 2 1]); %Оценивание ARMAX-модели
» present(th) %Вывод информации о модели
This matrix was created by the command ARMAX  on 1/19 2001 at 9:56
Loss fcn: 0.0071844   Akaike's FPE: 0.0072712 Sampling interval 1
The polynomial coefficients and their standard deviations are
B =
         0    0.0073    0.0451
         0    0.0021    0.0025
A =
    1.0000   -1.6490    0.7026
         0    0.0097    0.0088
C =
    1.0000   -0.2741   -0.4759
         0    0.0289    0.0285
```

В данном случае задана структура модели, все полиномы которой имеют 2-й порядок. Для выводимой информации, по-видимому, особых разъяснений не требуется.

Функция arx

Функция arx оценивает параметры моделей AR и ARX:

```
th = arx(z,nn)
th = arx(z,nn,maxsize,T)
```

Аргументы и возвращаемая величина идентичны используемым в предыдущей функции за тем исключением, что аргумент nn содержит только три параметра:

```
nn = [na nb nk]
```

Функция может использоваться для оценивания параметров многовходовой модели вида

$$A(z)y(t) = B_1(z)u_1(t-nk_1) + B_2(z)u_2(t-nk_2) + \ldots + B_{nu}(z)u_{nu}(t-nk_{nu}) + e(t),$$

при этом

- в представлении z=[y u] экспериментальных данных u = [u1, u2, …, unu] есть матрица, столбцы которой соответствуют входам;

- nb и nk – векторы-столбцы, определяющие порядки полиномов $B_j(z)$ и задержки nk_j для входов объекта.

Функция bj

Функция bj оценивает параметры модели Бокса–Дженкинса:

```
th = bj(z,nn)
th = bj(z,nn,'trace')
[th, iter_info]= bj(z,nn,maxiter,tol,lim,maxsize,T,'trace')
```

Аргументы и возвращаемые величины идентичны приведенным для функции armax за тем исключением, что в данном случае аргумент nn содержит пять параметров: nn = [nb nc nd nf nk], то есть степени четырех полиномов и величину задержки.

Функция canstart

Функция canstart оценивает коэффициенты канонической формы модели для переменных состояния:

```
th = canstart(z,orders,nu)
th = canstart(z,orders,nu,dkx)
```

Аргументы функции следующие:

- z — матрица данных;
- orders — скаляр, задающий порядок *n* модели;
- nu — число входов;
- dkx = [d, k, x] — трехэлементный вектор, в котором:
  - k=1 означает, что матрица **K** в модели полностью параметризована (то есть подлежит оцениванию); k=0 означает, что **K=0**;
  - d имеет тот же смысл, но по отношению к матрице модели **D**;
  - x=1 означает, что параметризован вектор начальных значений переменных состояния; при x=0 данные значения полагаются нулевыми.

По умолчанию dkx = [0, 1, 1].

Возвращаемая величина th — модель в тета-формате.

Приведем пример использования рассматриваемой функции с целью восстановления модели для переменных состояния второго порядка по имеющимся экспериментальным данным (один вход, один выход). Соответствующий фрагмент программы и результаты вычислений приведены ниже:

```
» load dryer2 %Загрузка данных
» z=[y2 u2];
```

```
» th=canstart(z,2,1); %Оценивание модели
» present(th) %Вывод информации о модели
This matrix was created by the command CANSTART on 1/19 2001 at 12:50
Loss fcn: 1   Akaike's FPE: 1 Discrete time model with sampling interval 1
The state-space matrices with standard deviations given as imaginary parts are
a =
        0              1.0000
  -0.6817 + 0.0000i   1.6191 + 0.0000i
b =
  -0.0316 + 0.0000i
   0.0415 + 0.0000i
c =
     1     0
d =
     0
k =
   0.4907 + 0.0000i
   0.4772 + 0.0000i
x0 =
   4.7505 + 0.0000i
   4.9671 + 0.0000i
lambda =
     1
```

Анализируя представленную в результате вычислений информацию, отметим, что в данном случае матрица системы **A** и матрицы (векторы) **B** и **K**, а также вектор начальных условий **x0** не содержат комплексные элементы (мнимые части элементов равны нулю). Через `lambda` обозначена интенсивность шума $e(t)$.

Функция ivar

Функция `ivar` оценивает параметры скалярной AR-модели

$$A(z)\, y(t) = v(t),$$

где шум $v(t)$ предполагается коррелированным случайным процессом типа процесса скользящего среднего, то есть имеющим представление $v(t) = H(z)e(t)$ (здесь $e(t)$ — белый шум), с использованием квазиоптимального метода инструментальной переменной (так называемого IV-метода). Функция записывается в виде:

```
th = ivar(y,na)
th = ivar(y,na,nc,maxsize,T)
```

где

- `y` — вектор данных;
- `na` — порядок модели авторегрессии (степень полинома $A(z)$);

- nc -- порядок полинома $H(z)$, по умолчанию na=nc;
- maxsize и T имеют тот же смысл, что и для функций armax и bj.

Возвращаемая величина th — модель в тета-формате.

Функция ivx

Функция ivx оценивает параметры ARX-модели методом инструментальной переменной:

```
th = ivx(z,nn,x)
th = ivx(z,nn,x,maxsize,T)
```

Функция аналогична функции arx и является ее альтернативой для случая, когда шум наблюдений является коррелированным. Аргумент x — матрица задаваемых инструментальных переменных. Остальные аргументы и возвращаемая величина аналогичны рассмотренным для функции arx.

ЗАМЕЧАНИЕ

Вместо функции arx рекомендуется использовать функцию iv4.

Функция iv4

Функция iv4 оценивает параметры ARX-модели с использованием четырехступенчатого метода инструментальной переменной. Может применяться для идентификации многомерных объектов:

```
th = iv4(z,nn)
th = iv4(z,nn,maxsize,T)
```

Функция аналогична функции arx и является ее альтернативой для случая, когда шум наблюдений является коррелированным.

Ниже приведен пример использования данной функции для идентификации объекта с одним выходом и двумя входами при различных задержках по входам (0 и 2), но при одинаковых степенях полиномов для данных входов (2):

```
» z = [y u1 u2];
» nb = [2 2]; %Задание степеней полиномов Bi(z)
» nk = [0 2]; %Задание задержек по входам
» %Оценивание модели
» th = iv4(z,[2 nb nk]);
```

Функция n4sid

Функция n4sid используется для оценивания параметров моделей для переменных состояния в канонической форме при произвольном числе входов и выходов:

```
th = n4sid(z)
[th,AO] = n4sid(z,order,ny,auxord,dkx,maxsize,T,'trace')
```

где:

- `z` — матрица экспериментальных данных;
- аргумент `order` задает порядок модели. Если данный аргумент вводится как вектор-строка, например, `order = [1:10]`, то предварительные расчеты выполняются по моделям всех заданных порядков (по умолчанию — от 1-го по 10-й) с выводом графика, позволяющего выбрать оптимальный порядок. Если `order = 'best'` (по умолчанию), выбирается модель «наилучшего» порядка (путем выбора наилучшей среди моделей 1-го—10-го порядков);
- `ny` — количество выходов, по умолчанию `ny = 1`;
- `auxord` — так называемый дополнительный порядок, используемый алгоритмом оценивания. Должен быть больше, чем порядок, задаваемый параметром `order` (по умолчанию `auxord = 1.2*order+3`. Если `auxord` вводится как вектор-строка (так же, как `order`), выбирается модель наилучшего порядка;
- `dkx`, `maxsize`, `T`, `'trace'` — как для функций `bj`, `canstart`, `ivar`.

Возвращаемые величины:

- `th` — имя (идентификатор) построенной модели в тета-формате;
- `AO` — наилучший порядок модели (определяемый с помощью задания `auxord`).

Функция oe

Функция `oe` оценивает параметры OE-модели:

```
th = oe(z,nn)
th = oe(z,nn,'trace')
[th, iter_info] = oe(z,nn,maxiter,tol,lim,maxsize,T,'trace')
```

Здесь аргумент `nn` задается в форме `nn = [nb nf nk]` и содержит информацию о степенях полиномов модели $B(z)$, $F(z)$ и о величине задержки nk.

Остальные аргументы и возвращаемые величины аналогичны рассмотренным для остальных функций данной группы.

Функция pem

Функция `pem` оценивает параметры обобщенной многомерной линейной модели вида

$$A(z)y(t) = \frac{B_1(z)}{F_1(z)}u_1(t-nk_1) + \frac{B_2(z)}{F_2(z)}u_2(t-nk_2) + \ldots + \frac{B_{nu}(z)}{F_{nu}(z)}u_{nu}(t-nk_{nu}) + \frac{C(z)}{D(z)}e(t).$$

Функция записывается в виде:

```
th = pem(z,nn)
th = pem(z,nn,'trace')
[th, iter_info] = pem(z,nn,index,maxiter,tol,lim,...
maxsize,T,'trace')
```

где:

- аргумент nn имеет представление nn = [na nb nc nd nf nk], где na, nb, nc, nd, nf, nk в многомерном случае являются векторами-столбцами;
- index — вектор-строка, содержащий индексы коэффициентов, которые должны быть оценены (по умолчанию оцениваются все коэффициенты).

Остальные аргументы и возвращаемые величины соответствуют рассмотренным выше.

Функции итерационного параметрического оценивания

В данную группу входят следующие 7 функций, позволяющие проводить оценивание коэффициентов моделей итерационными (рекуррентными) методами, при которых на каждой k-й итерации учитываются экспериментальные данные, соответствующие моменту времени k.

Функция rarmax

Функция rarmax реализует процедуру рекуррентного оценивания коэффициентов ARMAX-модели:

```
thm = rarmax(z,nn,adm,adg)
[thm,yhat,P,phi,psi] = rarmax(z,nn,adm,adg,th0,P0,phi0,psi0)
```

Здесь:

- z — матрица данных;
- nn — аргумент, задающий параметры ARMAX-модели, nn = [na nb nc nk];
- adm и adg — аргументы, задающие вид процедуры оценивания, например, значения adm ='ff' и adg = lam задают рекуррентный метод наименьших квадратов, adm = 'ug' и adg = gam — градиентный метод оценивания и т. п. (дополнительную информацию можно получить, используя команду help rarmax);

- th0 — вектор-строка, содержащий начальные значения оцениваемых параметров, по умолчанию — нулевой вектор;
- P0 — начальная матрица ковариаций оцениваемых коэффициентов, по умолчанию равна единичной матрице, умноженной на 10^4;
- phi0 и psi0 — соответственно, начальные значения вектора данных и вектора-градиента (по умолчанию — нулевые). Их размеры достаточно сложным образом зависят от параметров модели.

Возвращаемые величины:

- thm — матрица оценок коэффициентов модели; k-я строка матрицы содержит оценки, соответствующие моменту времени k в следующем порядке:

  ```
  thm(k,:) = [a1,a2,...,ana,b1,...,bnb];
  ```

- yhat — вектор прогнозируемых значений выхода (соответствующий вектору z(k,1));
- P, phi, psi — конечные значения матрицы ковариаций оценок коэффициентов, вектора данных и вектора-градиента.

Функция rarx

Функция rarx реализует процедуру рекуррентного оценивания коэффициентов AR и ARX-моделей:

```
thm = rarx(z,nn,adm,adg)
[thm,yhat,P,phi] = rarx(z,nn,adm,adg,th0,P0,phi0)
```

Аргументы и возвращаемые величины аналогичны рассмотренным для предыдущей функции.

Функция rbj

Функция rbj — то же, что предыдущая функция, но для модели Бокса—Дженкинса:

```
thm = rbj(z,nn,adm,adg)
[thm,yhat,P,phi,psi] = ...
rbj(z,nn,adm,adg,th0,P0,phi0,psi0)
```

Аргументы и возвращаемые величины — как у функции rarmax, с учетом того, что оценивается модель Бокса—Дженкинса.

Функция roe

Функция roe — то же, что предыдущая функция, но для OE-модели:

```
thm = roe(z,nn,adm,adg)
[thm,yhat,P,phi,psi] = roe(z,nn,adm,adg,th0,P0,phi0,psi0)
```

Аргументы и возвращаемые величины те же, что у функции rarmax.

Функция rpem

Функция rpem реализует рекуррентную процедуру оценивания параметров обобщенной линейной модели с несколькими входами, но с одним выходом:

```
thm = rpem(z,nn,adm,adg)
[thm,yhat,P,phi,psi] = rpem(z,nn,adm,adg,th0,P0,phi0,psi0)
```

Аргументы и возвращаемые величины те же, что у функции rarmax.

Функция rplr

Функция rplr имеет то же назначение, что и предыдущая функция. Отличие состоит в алгоритме оценивания. Функция записывается в виде:

```
thm = rplr(z,nn,adm,adg)
[thm,yhat,P,phi] = rplr(z,nn,adm,adg,th0,P0,phi0)
```

Функция segment

Функция segment осуществляет разбиение (сегментацию) данных и оценивание коэффициентов моделей для каждого сегмента в предположении, что в пределах каждого сегмента коэффициенты модели являются постоянными, но могут изменяться (скачкообразно) при переходе от сегмента к сегменту. Оцениваемые модели — AR, ARX, ARMAX. Функция записывается в виде:

```
segm = segment(z,nn)
[segm,V,thm,R2e] = segment(z,nn,R2,q,R1,M,th0,P0,l1,mu)
```

Здесь:

- z — матрица данных;
- аргумент nn определяет порядок модели:
  - для ARMAX-модели nn = [na nb nc nk];
  - для ARX-модели nn = [na nb nk];
  - для AR-модели nn = na;
- R2 — оценка дисперсии шума наблюдений;
- q — вероятность скачкообразного изменения модели в момент времени k, по умолчанию 0.01;
- R1 — предполагаемая матрица ковариаций оценок коэффициентов при их возможном скачкообразном изменении, по умолчанию — единичная матрица;
- M — число параллельных моделей, используемых в алгоритме, по умолчанию 5;

- th0 и P0 — смысл данных аргументов рассмотрен выше, при описании функции rarmax (здесь начальное значение P0 — единичная матрица, умноженная на 10);
- l1 - аргумент, определяющий существование параллельных моделей (каждая модель не исключается, по крайней мере, в течение l1 итераций), по умолчанию l1 = 1;
- mu — параметр алгоритма оценивания матрицы R2, по умолчанию 0.97.

Возвращаемые величины:

- segm — матрица, k-я строка которой содержит оценки коэффициентов, соответствующие моменту времени k (как возвращаемая матрица в функциях rarx и rarmax);
- thm — матрица, содержащая оценки коэффициентов, соответствующие моменту времени k без учета сегментации данных;
- V — сумма квадратов ошибок сегментированной модели;
- R2e — полученная оценка матрицы R2 для момента времени k.

Пример. Иллюстрируем возможности функции на примере аппроксимации синусоидального сигнала кусочно-постоянным, при этом на каждом сегменте принятая модель имеет вид

$$y(t) = b_1 \cdot 1.$$

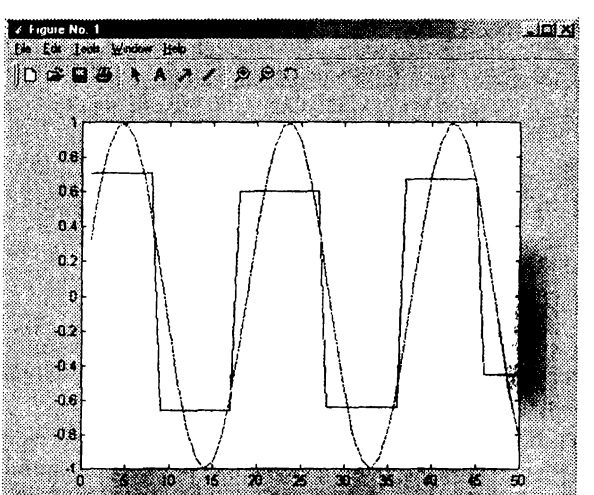


Рис. 1.22. Результаты аппроксимации синусоидального сигнала кусочно-постоянным, полученные с помощью функции segment

Соответствующая программа приведена ниже, результаты отражены на рис. 1.22.

```
» y = sin([1:50]/3)'; %Задание гармонического сигнала
» thm = segment([y,ones(size(y))],[0 1 1],0.1); %Сегментация данных
» plot([thm,y]) %Вывод результатов
```

Функции задания структуры модели

В эту группу входят следующие 6 функций.

Функция arx2th

Функция arx2th конструирует матрицу модели так называемого тета-формата по полиномам ARX-модели многомерного объекта:

```
th = arx2th(A,B,ny,nu)
th = arx2th(A,B,ny,nu,lam,T)
```

Исходная ARX-модель многомерного объекта имеет вид:

$$y(t) + A_1 y(t-1) + A_2 y(t-2) + \ldots + A_{na} y(t-na) =$$
$$= B_0 u(t) + B_1 u(t-1) + \ldots + B_{nb} u(t-nb) + e(t),$$

где $y(t)$ — вектор-столбец с ny элементами, $u(t)$ — вектор-столбец с nu элементами (то есть объект имеет nu входов и ny выходов), A_k и B_k — матрицы размеров ny×ny и nu×nu соответственно, $e(t)$ — вектор шума.

Аргументы функции следующие:

- A = [I A_1 A_2 … A_{na}], где I — единичная матрица;
- B = [B_0 B_1 … B_{nb}];
- ny и nu — количество выходов и входов модели (объекта);
- lam — матрица ковариаций шума $e(t)$, по умолчанию — единичная матрица;
- T — интервал дискретизации.

Функция canform

Функция canform конструирует каноническую форму модели для переменных состояния многомерного объекта:

```
ms = canform(orders,nu)
ms = canform(orders,nu,dkx)
```

Используется совместно с функцией mf2th (см. ниже). Аргументы аналогичны аргументам описанной выше функции canstart. Возвращаемая величина ms — матрица, отражающая информацию о структуре модели.

Функция mf2th

Функция mf2th преобразует структуру модели для переменных состояния в модель тета-формата:

```
th = mf2th(model,cd,parval)
th = mf2th(model,cd,parval,aux,lambda,T)
```

где:

- model — имя m-файла, содержащего информацию о том, как матрицы модели зависят от свободных (оцениваемых) параметров;
- cd — строковая переменная, задающая тип модели:
  - cd='c' — модель для непрерывного времени,
  - cd='d' — модель для дискретного времени;
- parval — вектор-строка, содержащий номинальные значения параметров;
- T — интервал дискретизации;
- aux — матрица вспомогательных переменных, используемых для различных целей и задаваемых в файле model;
- lambda — интенсивность шума.

Возвращаемая величина th — модель в тета-формате.

Функция modstruc

Функция modstruc определяет структуру модели для переменных состояния с частично незаданными коэффициентами матриц модели:

```
ms = modstruc(A,B,C,D,K)
ms = modstruc(A,B,C,D,K,x0)
```

Аргументы A, B, C, D, K, x0 — соответственно матрицы модели и вектор начальных условий (по умолчанию — нулевой). Известные коэффициенты матриц задаются числами, неизвестные (свободные), подлежащие оцениванию, — как NaN. Возвращаемая матрица ms содержит информацию о структуре модели для дальнейшего использования с помощью функции ms2th.

В следующем примере системы 2-го порядка неизвестные элементы содержатся в матрицах **A** и **B**:

```
» A=[NaN,0;0,NaN];          % Задание матрицы A
» B=[NaN;NaN];              % Задание матрицы B
» C=[1,1];                  % Задание матрицы C
» D=0;                      % Задание матрицы D
» K=[0;0];                  % Задание матрицы K
» ms=modstruc(A,B,C,D,K) % Формирование матрицы структуры модели
ms =
   NaN     0   NaN     1     0     0     0     1
     0   NaN   NaN     1     0     0     0     2
```

Функция ms2th

Функция ms2th преобразует матрицу структуры в модель тета-формата:

```
th = ms2th(ms)
th = ms2th(ms,cd,parval,lambda,T)
```

Аргументы и возвращаемая величина соответствуют описанным для предыдущих функций данной группы. Для примера определим следующую структуру модели (для непрерывного времени):

$$\dot{x} = \begin{bmatrix} \theta_1 & 0 \\ 0 & \theta_2 \end{bmatrix} x + \begin{bmatrix} \theta_3 \\ \theta_4 \end{bmatrix} u,$$

$$y = [1 \quad 1]x + e$$

с начальными значениями коэффициентов θ = [-0.2 -0.3 2 4] и оценим соответствующие свободные параметры:

```
» load dryer2 %Загрузка данных
» z=[y2 u2];
» %Задание структур матриц модели
» A=[NaN,0;0,NaN];
» B=[NaN;NaN];
» C=[1,1];
» ms=modstruc(A,B,C,0,[0;0]);
» th=ms2th(ms,'c',[-0.2,-0.3,2,4]);
» th=pem(z,th);
» present(th) %Отображение информации о модели
This matrix was created by the command PEM      on 1/20 2001 at 12:22
Loss fcn: 0.17279   Akaike's FPE: 0.17418 Continuous time model estimated using
sampling interval 1
The state-space matrices with standard deviations given as imaginary parts are
a =
  -0.2585 + 0.2844i        0
        0            -0.2816 + 0.3323i
b =
   3.5019 +93.6630i
  -3.5374 +93.6538i
c =
     1     1
d =
     0
k =
     0
     0
x0 =
     0
```

```
    0
ambda =
   0.1728
```

Функция poly2th

Функция poly2th конструирует модель тета-формата из исходной модели «вход-выход»:

```
th = poly2th(A,B)
th = poly2th(A,B,C,D,F,lam,T)
```

Исходная модель такая же, как и для рассмотренной функции pem (см. выше). Аргументы функции следующие:

- A, B, C, D, F — векторы-строки, содержащие коэффициенты соответствующих полиномов;
- T — интервал дискретизации;
- lam — дисперсия шума модели.

Возвращаемая величина th — модель в тета-формате.

Функции изменения и уточнения структуры модели

Для изменения и уточнения структуры модели служат следующие функции.

Функция fixpar

Функция fixpar фиксирует коэффициенты в структурах тета-моделей для переменных состояния и ARX, созданных с помощью функций ms2th и arx2th:

```
thn = fixpar(tho,matrix)
thn = fixpar(tho,matrix,elements,parval)
```

где:

- tho — исходная тета-модель;
- аргумент matrix (строковая переменная):
  - если исходная модель задана через переменные состояния, принимает одно из значений: 'A', 'B', 'C', 'D', 'K' или 'x0';
  - если исходная модель — ARX-типа, то параметр matrix принимает одно из значений: 'A1', 'A2', ..., 'B0', 'B1', ...;
- аргумент elements представляет собой матрицу из двух столбцов, содержащую индексы фиксируемых (то есть не подлежащих какому-либо последующему изменению) элементов матрицы, определяемой аргументом matrix, например, запись

```
thn = fixpar(tho,'A',[1,2;3,5]);
```

означает, что будут зафиксированы элементы A_{12} и A_{35}. По умолчанию фиксируются начальные значения параметров или их оценки (последние по времени);

- parval — вектор, элементы которого явно задают некоторые фиксируемые параметры (k-й элемент этого вектора задает величину k-го элемента в строке elements).

Возвращаемая величина thn — тета-модель с частично зафиксированными параметрами.

Функция sett

Функция sett явным образом задает интервал дискретизации в моделях:

```
modn = sett(mod,T)
```

Аргументы:

- mod — имя (идентификатор) модели в тета-формате или в частотном формате (см. описание функции spa);
- T — задаваемый интервал дискретизации.

Возвращаемая величина modn — модель объекта с новым значением T.

Функция thinit

Функция thinit устанавливает начальные (случайные) значения коэффициентов, подлежащих оцениванию в модели для переменных состояния (тета-формат) с последующим применением для такого оценивания функции pem. Записывается в виде:

```
th = thinit(th0)
th = thinit(th0,R,pars,sp)
```

где

- th0 — исходная тета-модель;
- аргументы R и pars задают параметры закона случайного задания коэффициентов модели по следующему правилу: коэффициент с номером k устанавливается равным величине pars(k) + e*sqrt(R(k)), где e — гауссова случайная величина с нулевым математическим ожиданием и единичной дисперсией. По умолчанию все элементы вектора R — единицы, а элементы вектора pars равны номинальным значениям коэффициентов модели;
- sp — строковая переменная с возможными значениями 'b' (стабильна как модель, так и прогнозируемый по ней выход), 's' (стабильна только модель) и 'p' (стабилен только прогнозируемый выход); по умолчанию sp='p'.

Возвращаемая величина th — тета-модель с заданными начальными значениями коэффициентов.

Функция unfixpar

Функция unfixpar является обратной по отношению к функции fixpar, то есть делает зафиксированные коэффициенты модели свободными:

```
thn = unfixpar(tho.matrix)
thn = unfixpar(tho.matrix.elements)
```

Аргументы и возвращаемая величина идентичны рассмотренным для функции fixpar.

Функции выбора структуры модели

Данная группа образована следующими четырьмя функциями.

Функция arxstruc

Функция arxstruc вычисляет функции потерь для ряда различных конкурирующих ARX-моделей с одним выходом:

```
v = arxstruc(ze,zv,NN)
v = arxstruc(ze,zv,NN,maxsize)
```

где:

- ze и zv — соответственно, матрицы экспериментальных данных для оценивания и верификации (проверки качества) моделей;
- NN — матрица задания конкурирующих структур со строками вида nn = [na nb nk];
- maxsize — служебный аргумент, определяющий максимально возможную размерность задачи.

Возвращаемая величина v — матрица, первые (верхние) элементы каждого столбца которой (кроме последнего) являются значениями функции потерь для ARX-моделей, структура которых отображается последующими элементами столбцов (то есть каждый столбец соответствует одной модели). Первый элемент последнего столбца — число значений экспериментальных данных для верификации моделей.

Последующий отбор наилучшей модели может быть произведен с помощью функции selstruc (см. ниже).

Функция ivstruc

Функция ivstruc вычисляет среднее квадратичное отклонение выходов объекта и ряда различных конкурирующих ARX-моделей:

```
v = ivstruc(ze,zv,NN)
v = ivstruc(ze,zv,NN,p,maxsize)
```

Рассматриваемая функция по назначению аналогична предыдущей, но отличается используемым алгоритмом оценивания – в данном случае используется метод IV.

Аргументы (кроме p) аналогичны аргументам функции arxstruc. Аргумент разрешает или запрещает (p=0) вывод так называемых условных чисел IV-матрицы.

Возвращаемая величина v практически идентична матрице, возвращаемой предыдущей функцией, но отличается от последней дополнительной (нижней) строкой, в которой приводятся условные числа IV-матрицы.

Функция selstruc

Функция selstruc осуществляет выбор наилучшей структуры модели из ряда возможных вариантов:

```
[nn,vmod] = selstruc(v)
[nn,vmod] = selstruc(v,c)
```

Здесь:

- v — матрица, возвращаемая функциями arxstruc или ivstruc;
- c — строковая переменная, определяющая вывод графика или критерий отбора наилучшей структуры:
  - при c = 'plot' выводится график зависимости функции потерь от числа оцениваемых коэффициентов модели;
  - если c = 'log', выводится график логарифма функции потерь;
  - при c = 'aic' график не выводится, но возвращается структура, минимизирующая так называемый теоретический информационный критерий Акейке (Akaike's Information Theoretic Criterion — AIC):

    $V_{mod} = V(1 + 2(d/N))$,

    где V — значение функции потерь, d — число оцениваемых коэффициентов модели, N — объем экспериментальных данных, используемых для оценивания,
  - при c = 'mdl' возвращается структура, обеспечивающая минимум так называемого критерия Риссанена минимальной длины описания (Rissanen's Minimum Description Length — MDL)

    $V_{mod} = V^*(1 + \log(N)\ (d/N))$;
  - при c, равном некоторому численному значению a, выбирается структура, которая минимизирует

    $V_{mod} = V(1 + a(d/N))$.

Возвращаемые величины:

- nn — выбранная структура;
- vmod — значение соответствующего критерия.

Рассмотрим следующий пример. Используя экспериментальные данные файла dryer2.mat, сравним ARX-модели порядков 1–5 с единичной задержкой и произведем отбор лучшей структуры:

```
» load dryer2 %Загрузка данных
» z=[y2 u2];
» NN=struc(1:5,1:5,1); %Задание пределов изменения порядка модели
» V=arxstruc(z(1:200,:),z(201:400,:),NN);
» nn=selstruc(V,0); %Выбор наилучшей модели
» th=arx(z,nn); %Оценивание коэффициентов выбранной модели
» present(th) %Представление модели
This matrix was created by the command ARX    on 1/20 2001 at 15:50
Loss fcn: 0.0014058   Akaike's FPE: 0.0014342 Sampling interval 1
The polynomial coefficients and their standard deviations are
B =
         0    0.0002    0.0060    0.0642    0.0617    0.0208
         0    0.0010    0.0012    0.0012    0.0023    0.0026
A =
    1.0000   -0.9662    0.0324    0.0257    0.1168   -0.0526
         0    0.0306    0.0420    0.0415    0.0317    0.0120
```

Как видно, в данном случае наилучшей оказалась модель 5-го порядка.

Функции преобразования модели

Данная группа образована десятью функциями.

Функция idmodred

Функция idmodred понижает порядок модели (в тета-формате):

```
thred = idmodred(th)
thred = idmodred(th,order,OE)
```

Функция преобразует исходную модель th, заданную в тета-формате, понижая ее порядок. Аргументы функции:

- th — исходная модель;
- order — желаемый порядок. Если order=[ ] (по умолчанию), то выводится график, показывающий, как управляемость и наблюдаемость зависят от порядка модели;
- если OE='oe', то рассчитывается выходная ошибка возвращаемой модели, в любом другом случае понижается порядок модели шума.

Возвращаемая величина thred — модель пониженного порядка.

Функция th2arx

Функция th2arx преобразует модель тета-формата в ARX-модель:

```
[A,B] = th2arx(th)
[A,B,dA,dB] = th2arx(th)
```

Возвращаемая ARX-модель имеет вид, рассмотренный при описании функции arx2th (см. выше), в частности,

```
A = [I A1 A2 ... Ana];
B = [B0 B1 ... Bnb];
```

элементы матриц dA и dB являются стандартными отклонениями элементов матриц A и B.

Функция th2ff

Функция th2ff вычисляет частотные характеристики и соответствующие стандартные отклонения по модели в тета-формате:

```
[g,phiv] = th2ff(th)
[g,phiv] = th2ff(th,ku,w,ky)
```

Здесь:

- th — исходная модель в тета-формате;
- w — вектор-строка, элементы которого являются частотами для расчета значений частотных характеристик;
- ku и ky — векторы-строки, которые (в случае многомерного объекта) задают соответственно номера входов и выходов, для которых рассчитываются оценки частотных передаточных функций.

Возвращаемые величины — как в функции spa (см. выше).

Функция th2par

Функция th2par возвращает вектор коэффициентов из модели тета-формата:

```
[par,P,lam] = th2par(th)
```

Аргумент th — модель в тета-формате. Возвращаемые величины:

- par — вектор коэффициентов модели следующего вида:
  - для обобщенной модели «вход-выход»

    $$par = [a_1 \ldots a_{na}\ b_1^1 \ldots b_{nb1}^1\ b_1^2 \ldots b_{nb2}^2 \ldots b_1^{nu} \ldots b_{nbnu}^{nu}\ c_1 \ldots c_{nc}$$
    $$d_1 \ldots d_{ndi}\ f_1^1 \ldots f_{nf1}^1 \ldots f_1^{nu} \ldots f_{nfnu}^{nu}];$$

  - для модели переменных состояния данный вектор состоит из перечисленных по строкам последовательно матриц **A**, **B**, **C**, **D**, **K** и **x0**;

- P — матрица ковариаций приведенных коэффициентов;
- lam — матрица ковариаций шума наблюдений.

Информация выводится только для свободных коэффициентов.

Функция th2poly

Функция th2poly преобразует матрицу модели тета-формата в матрицы обобщенной (многомерной) линейной модели (см. описание функции pem):

```
[A,B,C,D,F,lam,T] = th2poly(th)
```

Здесь lam — интенсивность шума наблюдений. Остальные величины пояснены выше.

Функция th2ss

Функция th2ss преобразует тета-модель в модель для переменных состояния:

```
[A,B,C,D,K,X0] = th2ss(th)
[A,B,C,D,K,X0,dA,dB,dC,dD,dK,dX0] = th2ss(th)
```

Смысл аргументов и возвращаемых величин — тот же, что в предыдущих функциях.

Функция th2tf

Функция th2tf преобразует модель тета-формата многомерного объекта в вектор передаточных функций, связанных с выбранным входом:

```
[num,den] = th2tf(th)
[num,den] = th2tf(th,iu)
```

где

- th — исходная модель тета-формата;
- iu — номер входа, для которого строятся передаточные функции (по умолчанию 1).

Возвращаемые величины:

- num — матрица, k-я строка которой содержит коэффициенты полинома числителя частной передаточной функции от выбранного (iu) входа к k-му выходу;
- den — вектор-строка, элементами которого являются коэффициенты знаменателя передаточных функций (одного и того же для всех частных передаточных функций).

К примеру, для непрерывного случая величины

```
num = [1 2]
den = [1 3 0]
```

соответствуют передаточной функции

$$W(p) = \frac{p+2}{p^2+3p}.$$

Аналогично, для дискретной модели при

```
num = [2 4 0]
den = [1 2 3 5]
```

имеем передаточную функцию вида

$$W(z) = \frac{2z^2+4z}{z^3+2z^2+3z+5}$$

или (что то же самое) вида

$$W(z) = \frac{2z^{-1}+4z^{-2}}{1+2z^{-1}+3z^{-2}+5z^{-3}}.$$

Функция th2zp

Функция th2zp рассчитывает нули, полюса и статические коэффициенты передачи (коэффициенты усиления) модели тета-формата в общем случае многомерного объекта:

```
[zepo,k] = th2zp(th)
[zepo,k] = th2zp(th,ku,ky,thresh)
```

Здесь:

- th — имя тета-модели;
- ku и ky — вектор-строки, содержащие номера входов ju и выходов jy, для которых будут рассчитываться частные передаточные функции и соответствующие им нули, полюса и коэффициенты усиления;
- thresh — пороговая величина, при превышении которой рассчитываемые нули будут полагаться равными бесконечности, по умолчанию thresh=100000.

Возвращаемые величины:

- zepo — матрица, в закодированном виде отражающая информацию о нулях и полюсах модели (в графическом виде данная информация может быть извлечена с помощью функции zpplot, в аналитическом — с помощью функции getzp);
- k — матрица с информацией о статических коэффициентах усиления частных передаточных функций. Первую строку этой матрицы составляют элементы, имеющие значения (jy-1)*1000+ju, вторую — соответствующие коэффициенты усиления.

Проведем сравнение полюсов двух ARX-моделей — 2-го и 3-го порядков, оценивание которых проводится по одним и тем же экспериментальным данным.

```
» load dryer2 %Загрузка данных
» z=[y2 u2];
» th2=arx(z,[2 2 1]); %Оценивание модели 2-го порядка
» th3=arx(z,[3 3 1]); %Оценивание модели 3-го порядка
» %Нахождение нулей и полюсов моделей
» zp2=th2zp(th2);
» zp3=th2zp(th3);
» %Графическое представление нулей и полюсов
» zpplot(zpform(zp2,zp3))
```

Результаты приведены на рис. 1.23 (график увеличен так, что отображены лишь нули — крестиками для модели 2-го порядка и квадратиками для модели 3-го порядка — и единичный круг. Данные результаты показывают, в частности, что обе модели являются устойчивыми, но у модели 3-го порядка запас устойчивости выше (ее полюса находятся дальше от границы круга).

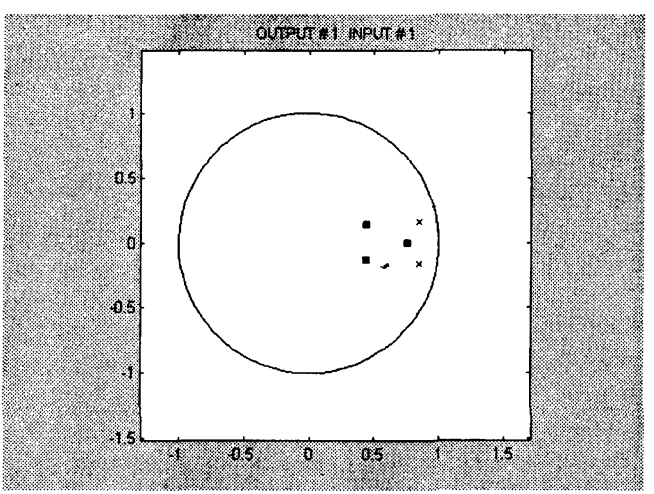


Рис. 1.23. Графическое представление полюсов рассматриваемых моделей 2-го и 3-го порядков

Функция thc2thd

Функция thc2thd преобразует модель из непрерывной формы в дискретную:

```
thd = thc2thd(thc,T)
```

Здесь:

- thc — исходная непрерывная тета-модель,
- T — интервал дискретизации,
- thd — возвращаемая дискретная тета-модель.

Функция thd2thc

Функция thd2thc является обратной по отношению к предыдущей, то есть преобразует дискретную модель в непрерывную:

```
thc = thd2thc(thd)
thc = thd2thc(thd,delay,NoP)
```

Смысл величин thc и thd пояснен выше; аргумент delay определяет способ учета задержки в дискретной модели:

- при delay = 'model' (по умолчанию) задержка учитывается звеном идеального запаздывания;
- при delay = 'del' задержка аппроксимируется непрерывным динамическим звеном;
- значение NoP=1 запрещает вывод матрицы ковариаций коэффициентов модели.

Функции извлечения информации о модели

В данную группу входят следующие 6 функций:

- getmfth — возвращает имя m-файла, созданного функцией mf2th;
- getncap — возвращает количество данных эксперимента, использованных для оценивания модели;
- getff — возвращает значения АЧХ и ФЧХ для заданных частотного диапазона и модели в частотном формате;
- gett — возвращает значение интервала дискретизации, заданного для модели;
- getzp — возвращает значения нулей и полюсов модели из матрицы zero-формата, созданной с применением функции th2zp;
- th2par — возвращает параметры модели исходя из ее тета-представления.

Для примера создадим ARX-модель 3-го порядка и выведем числовые значения ее нулей и полюсов:

```
» load dryer2 %Загрузка данных
» z=[y2 u2];
» th=arx(z,[3 2 1]); %Оценивание модели
```

```
» [zepo,k]=th2zp(th): %Формирование матрицы нулей и полюсов
» [ze,po] = getzp(zepo) %Вывод числовых значений
ze =
  -16.1043

po =
   0.8574
   0.5719 + 0.3084i
   0.5719 - 0 3084i
```

В выведенном результате ze — обозначение нуля, po — полюсов.

Более подробную информацию о данных функциях можно получить, используя справочную систему MATLAB.

Функции проверки адекватности модели

В данную группу входят следующие 5 функций.

Функция compare

Функция compare позволяет сравнить выходы модели и объекта: `compare(z,th);`

`[yh,fit] = compare(z,th,k,sampnr,leveladj)`

Здесь используются следующие аргументы:

- z — матрица данных;
- th — имя модели в тета-формате;
- k — количество предыдущих значений выхода объекта $y(t-k)$, используемых для прогноза по модели (по умолчанию не используются);
- sampnr — вектор-строка, элементы которого задают моменты времени, для которых будет строиться график;
- если leveladj='yes', то выходы как модели, так и объекта будут начинаться из нуля.

Возвращаемые величины:

- при использовании первой формы записи функции выводятся графики выходов объекта и построенной модели;
- при второй форме записи выводятся числовые значения прогноза модели yh и оценка адекватности модели fit, рассчитываемая по формуле

  `fit = norm(yh - y)/sqrt(length(y)),`

  где norm — в данном случае обозначение нормы вектора, length — количество элементов вектора, sqrt — обозначение операции извлечения квадратного корня.

Рассмотрим пример:

```
» load dryer2 %Загрузка данных
» z=[y2 u2];
» ze=z(1:250,:); %Формирование выборки для оценивания модели
» zv=z(251:500,:); %Формирование выборки для проверки модели
» th=armax(ze,[2 3 1 0]); %Оценивание ARX
» compare(zv,th,6); %Сравнение выходов модели и объекта
```

Здесь в качестве выхода объекта, с которым производится сравнение выхода модели, использована часть исходных данных — zv. Результат сравнения отображает рис. 1.24.

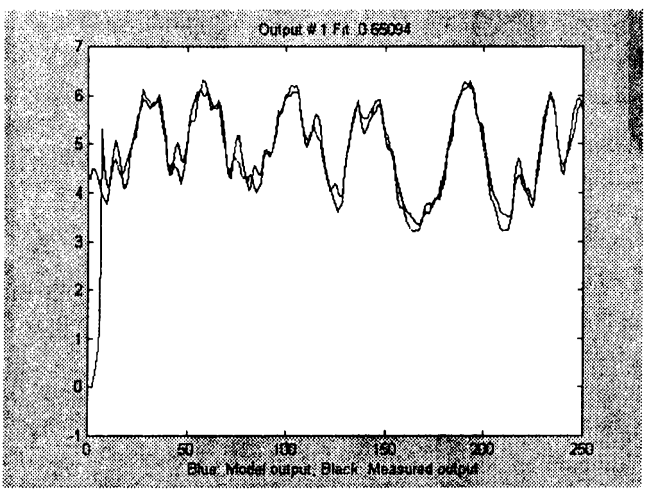


Рис. 1.24. Выходы объекта и модели, отображаемые с помощью функции compare

Функция idsim

Функция idsim возвращает выход модели тета-формата:

```
y = idsim([u e],th)
[y, ysd] = idsim(u,th)
```

Здесь:

- u — вектор значений входного сигнала (матрица входных сигналов в случае многомерного объекта);
- e — вектор значений шума (если не задан, принимается равным нулю);
- th — модель в тета-формате.

Возвращаемые величины:

- y — выход модели при заданном входе;
- ysd — соответствующие стандартные отклонения.

Функция pe

Функция pe прогнозирует ошибку модели при заданном входе и известном выходе объекта:

```
e = pe(z,th)
```

Здесь:

- z — матрица данных формата z = [y u];
- th — модель тета-формата;
- e — возвращаемый вектор значений ошибки, вычисляемых по формуле

  $e(t) = H^{-1}(z)[y(t) - W(z)u(t)]$.

Функция predict

Функция predict осуществляет прогноз выхода объекта по его тета-модели и с учетом информации о предыдущих фактических значениях выхода (рекомендована для расчета прогноза значений временной последовательности):

```
yp = predict(z,th)
[yp,thpred] = predict(z,th,k)
```

Аргументы z и th здесь те же, что и для предыдущей функции, k — количество используемых при прогнозе предшествующих фактических значений выхода, по умолчанию k=1.

Возвращаемые величины:

- yp — прогнозируемые значения;
- thpred — прогнозирующее звено в тета-формате.

Функция resid

Функция resid вычисляет остаточную ошибку для заданной модели и соответствующие корреляционные функции:

```
[e,r] = resid(z,th)
[e,r] = resid(z,th,M,maxsize)
resid(r)
```

Аргументы z, th, maxsize описаны ранее. Аргумент M задает число значений аргумента при расчете корреляционных функций.

Возвращаемые величины: e — вектор остаточных ошибок модели, r — матрица значений АКФ процесса $e(t)$ и значений ВКФ между остаточными ошибками и входами вместе с соответствующими 99%-ми

доверительными интервалами. Кроме возврата указанных значений выводятся графики данных функций. График может быть выведен повторно вызовом функции в форме `resid(r)`.

Для примера проведем оценивание ARX-модели 3-го порядка с последующим использованием рассматриваемой функции:

```
» load dryer2 %Загрузка данных
» z=[y2 u2];
» th=arx(z,[3 2 1]); %Оценивание ARX-модели
» [e,r]=resid(z,th) %Расчет остаточной ошибки и вывод графиков АКФ и ВКФ

e =
         0
         0
         0
    0.1374
   -0.1775
    0.0344
    0.0152
```

Здесь из соображений экономии приведены только несколько первых рассчитанных значений остаточных ошибок модели. Графики соответствующих АКФ и ВКФ представлены на рис. 1.25.

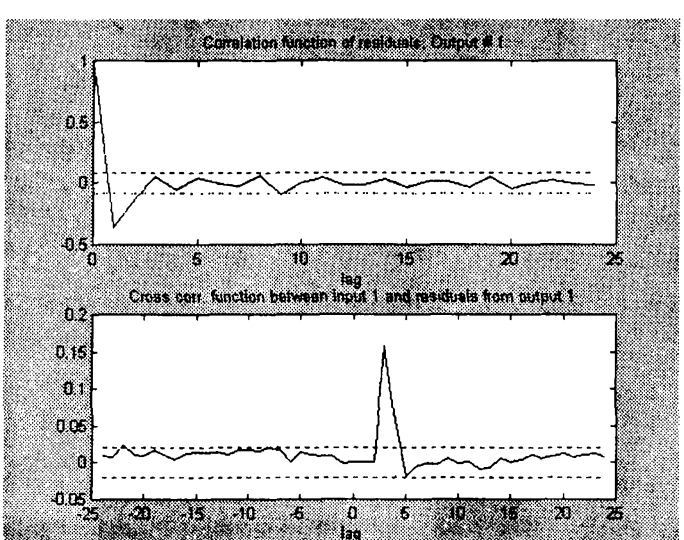


Рис. 1.25. Графики АКФ и ВКФ, полученные с помощью resid

Прочие функции

Ниже представлен ряд разнохарактерных функций, которые можно назвать прочими:

- Функция auxvar возвращает информацию о вспомогательных переменных iter_info, lim, maxiter, maxsize, tol и T. Для получения более подробной информации используйте команду

  ```
  help auxvar
  ```

- Функция freqfunc возвращает информацию об описании модели в частотном формате. Для получения более подробной информации используйте команду

  ```
  help freqfunc
  ```

- Функция idsimsd иллюстрирует неопределенность в моделировании выхода тета-модели:

  ```
  idsimsd(u,th)
  idsimsd(u,th,N,noise)
  ```

Функция возвращает график выходов N моделей (N по умолчанию равно 10) при заданных входе u и тета-модели th. Коэффициенты отмеченных моделей задаются случайно в соответствии с матрицей ковариаций тета-модели. Если noise='noise', то в модель вводится моделируемый шум наблюдений. При noise='nonoise' шум отсутствует.

- Функция nuderst задает шаг численного дифференцирования, используемый, в частности, в функциях th2ff и th2zp:

  ```
  nds = nuderst(pars)
  ```

  где:

  - pars — коэффициенты, вектор-строка;
  - nds: вектор-строка, k-й элемент которого представляет собой задаваемый шаг вариации k-го коэффициента.

- Функция theta возвращает информацию о тета-формате модели. Для получения более подробной информации используйте команду

  ```
  help theta
  ```

Демонстрация возможностей пакета

Демонстрация возможностей пакета осуществляется с помощью функции iddemo (без аргументов). При ее выполнении появляется следующее сообщение:

```
The SYSTEM IDENTIFICATION TOOLBOX is an analysis module
that contains tools for building mathematical models of
dynamical systems, based upon observed input-output data.
The toolbox contains both PARAMETRIC and NON-PARAMETRIC
MODELING methods.

Identification Toolbox demonstrations:

1) The Graphical User Interface (ident): A guided Tour.
2) Build simple models from real laboratory process data.
3) Compare different identification methods.
4) Model structure determination case study.
5) Spectrum estimation (Marple's test case).
6) Adaptive/Recursive algorithms.
7) Segmentation of data and models.
8) State-space and multi-output models.
9) Case studies
0) Quit
Select a demo number:
```

Данное сообщение предоставляет пользователю возможность выбрать одну из девяти демонстрационных программ, иллюстрирующих возможности пакета. Первая из них — работа с графическим интерфейсом пользователя — была, по сути, рассмотрена при изложении соответствующего раздела. Остальные являются весьма объемными, что делает нецелесообразным их приведение в тексте книги. Их изучение мы оставляем читателю.

Идентификация с использованием блоков Simulink

В состав библиотеки System ID Blocks блоков Simulink системы MATLAB 5.3.1 входят блоки, позволяющие проводить оценивание ряда типовых моделей (рис. 1.26):

- модели авторегрессии AR (AutoRegressive model estimator);
- ARX-модели (AutoRegressive with eXternal input model estimator);
- ARMAX-модели (AutoRegressive Moving Average with eXternal input model estimator);
- модели Бокса—Дженкинса BJ (Box—Jenkins model estimator);
- обобщенной линейной модели (General model estimator using Predictive Error Method);
- модели «вход-выход» OE (Output-error model estimator).

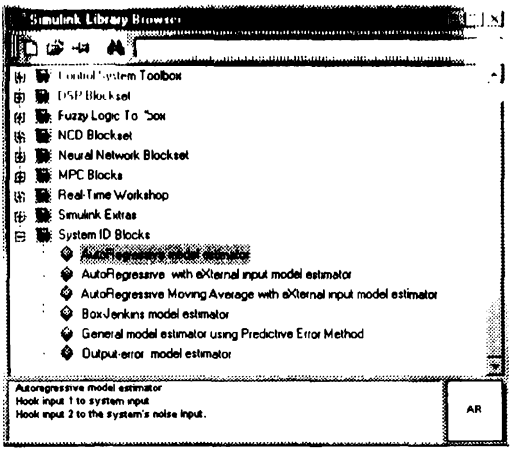


Рис. 1.26. *Библиотека блоков идентификации Simulink*

Правила работы с данными блоками аналогичны правилам для других блоков Simulink. В качестве примера на рис. 1.27 представлена модель Simulink для идентификации дискретного объекта 3-го порядка.

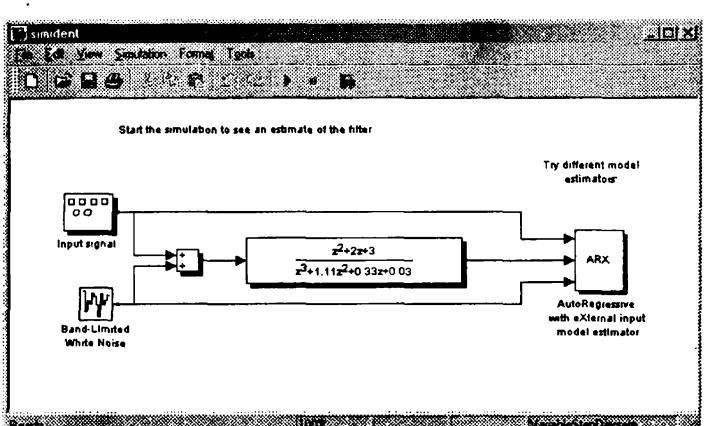


Рис. 1.27. Модель Simulink для идентификации дискретного объекта 3-го порядка

Щелчок на кнопке с треугольным значком приведет к запуску процесса моделирования, отражаемого изменяющимися во времени графиками, конечный вид которых приведен на рис. 1.28.

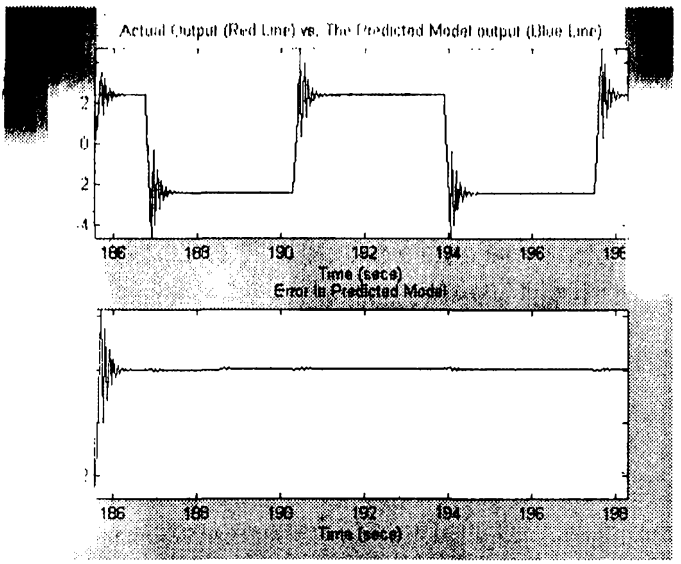


Рис. 1.28. Графики, иллюстрирующие процесс оценивания модели

Заметим, что полученная модель отображается в основном окне MATLAB. В данном случае конечный результат таков:

```
num/den =

                  z^3
-----------------------------------------
z^3 + 1.1052 z^2 + 0.32461 z + 0.027908
```

Сохранение результатов

Созданная модель с именем th сохраняется, как уже отмечалось, обычным образом — с помощью функции save, например:

```
save th
```

или

```
save имя_файла th
```

При этом будет создаваться файл с именем th или имя_файла с расширением mat.

Загрузка модели в следующем сеансе работы производится с помощью функции load.

Глава 2
Пакет Frequency Domain Identification

Назначение пакета Frequency Domain Identification

Возможности пакета

Пакет Frequency Domain Identification (пакет идентификации в частотной области) представляет специализированные средства для идентификации линейных динамических систем по их временному или частотному отклику. Частотные методы направлены в основном на идентификацию непрерывных систем, что является мощным дополнением к более традиционной дискретной методике, используемой в пакете System Identification Toolbox. Методы пакета могут быть применены к таким задачам, как моделирование электрических, механических, акустических и других устройств и систем. Свойства пакета:

- возможность моделирования и исследования периодических возмущений, псевдослучайных и случайных дискретных двоичных последовательностей;
- расчет доверительных интервалов для амплитуды и фазы либо нулей и полюсов;
- идентификация непрерывных и дискретных систем с неизвестным запаздыванием;
- диагностика модели, включая расчет прогнозируемых значений и невязок;
- преобразование моделей в формат моделей пакета System Identification Toolbox и обратно.

Используя частотный подход, можно добиться наилучшей модели в частотной области, избежать ошибок дискретизации, легко выделить постоянную составляющую сигнала, существенно улучшить соотношение сигнал/шум. Для получения возмущающих сигналов пакет предоставляет функции генерации полигармонических сигналов и двоичных последовательностей, случайных сигналов с заданным спектром.

Пакетом обеспечивается идентификация непрерывных и дискретных линейных динамических стохастических систем, автоматическая генерация требуемых входных сигналов, а также графическое отображение частотных характеристик моделей, спектров сигналов, нулей и полюсов исследуемой системы. Функции для тестирования включают вычисление невязок, передаточных функций, их нулей и полюсов, прогонку моделей с использованием тестовых данных.

Пакет в определенном смысле является дополнением к пакету System Identification, но может применяться независимо и совершенно самостоятельно. В то же время нельзя не отметить определенную сложность использования многих функций пакета ввиду необходимости задания большого количества аргументов.

Документация по пакету

Электронная документация по пакету Frequency Domain Identification представлена справочной системой (файлы формата HTML) и в виде описания в формате PDF с объемом 232 с.

Рекомендуемая литература

Методы идентификации в частотной области достаточно полно описаны в следующих монографиях:

1. Эйкхофф П. Основы идентификации систем управления. — М.: Мир, 1975.
2. Дейч А. М. Методы идентификации динамических объектов. — М.: Энергия, 1979.
3. Гроп Д. Методы идентификации систем. — М.: Мир, 1979.

Особенности идентификации систем в частотной области

Используемые модель и метод оценивания

Основные идеи и понятия идентификации динамических объектов (систем) были рассмотрены ранее, в предыдущей главе, поэтому остановимся лишь на тех особенностях методов идентификации в частотной области, которые реализованы в рассматриваемом пакете.

Первой такой особенностью является постулируемая структура объекта исследования, показанная на рис. 2.1.

На рисунке приняты следующие обозначения:

- $\Omega = j\omega$, где $\omega = 2\pi f$ — угловая частота;
- $W(\Omega) = W(j\omega)$ — комплексный коэффициент передачи;
- X и Y — обозначения, соответственно, входного и выходного сигналов объекта;
- X_m и Y_m — их измеряемые амплитуды;
- N_y и N_x — аддитивные случайные помехи (обычно гауссова типа), некоррелированные с входным сигналом и между собой и не зависящие от ω.

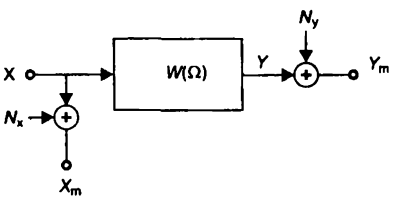


Рис. 2.1. Структура идентифицируемого объекта при идентификации в частотной области

В данном случае (в отличие от пакета System Identification) предполагается, что наблюдаемые выходной и входной сигналы регистрируются (измеряются) с помехами.

Второй особенностью является сама модель идентифицируемого объекта, которая представляется в виде

$$W(\Omega) = e^{-\Omega T_d}\frac{b_0 + b_1\Omega + b_2\Omega^2 + \ldots + b_{nn}\Omega^{nn}}{a_0 + a_1\Omega + a_2\Omega^2 + \ldots + a_{nd}\Omega^{nd}} = e^{-\Omega T_d}\frac{B(\Omega)}{A(\Omega)},$$

то есть в виде линейного непрерывного стационарного объекта с запаздыванием (T_d — величина запаздывания).

Третьей особенностью является характер экспериментальной информации: предполагается, что измерения входа и выхода производятся для частот ω_k ($k = 1, 2, \ldots, F$), при этом результатами измерений являются комплексные амплитуды X_{mk} и Y_{mk} (коэффициенты ряда Фурье). Иначе говоря, входной идентифицирующий сигнал предполагается полигармоническим.

С учетом структуры рис. 2.1 основные соотношения для исследуемого объекта могут быть записаны следующим образом:

$$Y_k = W(\Omega_k, \mathbf{P})X_k, \quad k = 1, 2, \ldots, F,$$

$$Y_{mk} = W(\Omega_k, \mathbf{P})(X_{mk} - N_{xk}) + N_{yk}, \quad k = 1, 2, \ldots, F,$$

где **P** — вектор неизвестных (подлежащих оцениванию) параметров модели; X_k и Y_k — комплексные амплитуды входного и выходного сигналов, не искаженных шумами, на частоте ω_k.

Наконец, последней особенностью является принятый метод оценивания.

Метод оценивания параметров модели, реализованный в пакете Frequency Domain Identification, базируется на приведенных допущениях о характере шумов и методе максимального правдоподобия, который в данном случае сводится к задаче минимизации критерия (так называемой *функции потерь*):

$$C_{WLS}(P) = \frac{1}{2}\sum_{k=1}^{P} g_k \left| e^{-\Omega_k T_d} X_{mk} A(\Omega_k, P) - Y_{mk} B(\Omega_k, P) \right|^2,$$

где g_k — некоторые весовые коэффициенты, то есть модификации взвешенного метода наименьших квадратов.

Оценка дисперсии невязок модели

$$e_k = e^{-\Omega_k T_d} X_{mk} A(\Omega_k, P) - Y_{mk} B(\Omega_k, P)$$

при этом производится по формуле

$$\begin{aligned}\mathrm{var}\{e_k\} &= E\{\overline{N}_{xk} N_{xk}\}\left|A(\Omega_k, P)\right|^2 + E\{\overline{N}_{yk} N_{yk}\}\left|B(\Omega_k, P)\right|^2 \\ &= 2\sigma_{xk}^2 \left|A(\Omega_k, P)\right|^2 + 2\sigma_{yk}^2 \left|B(\Omega_k, P)\right|^2.\end{aligned}$$

Здесь черта сверху означает комплексно-сопряженную величину, σ_{xk}^2 и σ_{yk}^2 — дисперсии шумов на входе и выходе объекта соответственно.

Весовые коэффициенты в приведенном выше критерии выбираются равными $g_k = 1/\mathrm{var}\{e_k\}$.

Представленные формулы составляют сущность использованного в пакете метода оценивания, названного ELiS (Estimator for Linear Systems — буквально «Оцениватель для линейных систем»). Нетрудно видеть, что для своего использования данный метод требует представления сигналов системы в частотной форме, то есть в форме комплексных амплитуд.

Данный метод развит и для более сложной ситуации, когда шумы наблюдений являются коррелированными.

Отметим достоинства ELiS:

- оцениваемые параметры имеют асимптотически нормальное распределение, даже если шумы наблюдений не являются гауссовыми;

- метод является робастным по отношению к законам распределений данных шумов;
- получаемая модель обеспечивает удобный и быстрый переход к передаточной функции объекта;
- метод обеспечивает быструю сходимость процедуры оценивания;
- метод является помехоустойчивым и обеспечивает удовлетворительное качество оценивания даже при малых отношениях сигнал/шум.

Недостатком метода следует считать необходимость организации активного идентифицирующего эксперимента с входным полигармоническим сигналом (что не всегда и не для всех объектов возможно).

Типы файлов

Следующей особенностью пакета являются используемые в нем типы файлов. Для представления данных и результатов идентификации применяются файлы следующих пяти типов:

1. Файлы, содержащие векторы данных из временной области (так называемые временные файлы) с расширениями tim, tnt, tbn.
2. Так называемые Фурье-файлы, содержащие массивы данных в форме комплексных амплитуд (с расширениями fou, fnt, fbn).
3. Файлы дисперсий, содержащие массивы дисперсий экспериментальных данных (с расширениями var, vnt, vbn).
4. Параметрические файлы, содержащие параметры передаточных функций (с расширениями par, pnt, pbn).
5. Файлы ковариаций, содержащие ковариационные матрицы оцениваемых параметров (с расширениями cov, cnt, cbn).

Все перечисленные файлы могут быть или текстовыми (с расширениями tim, tnt, fou, fnt, var, vnt, par, pnt, cov, cnt), или бинарными (с расширениями tbn, fbn, vbn, pbn, cbn) файлами MATLAB.

Приведем примеры текстовых файлов.

Пример 1. Файл данных во временной области.

```
%Имя файла: data.tim, тип: временной V1.0, 06-Ноябрь-1999, 11:51
%Электрическая машина
2 %Количество входов
1 %Количество выходов
2 %Количество экспериментов
5 %Количество опытов
0 %Начальное время (с)
```

```
1000 %Конечное время (с)
%
%            Результаты измерений
%          Time       Входы      Выход
1 %Эксперимента 1
0.00       1.00       2.00       0.0125
250.00     1.00       3.00       0.2345
500.00     1.00       4.00       2.456
750.00     1.00       5.00       15.63
1000.0     1.00       6.00       5.678
%
2 %Эксперимент 2
0.00       1.00       2.00       0.0125
250.00     1.00       3.00       0.2545
500.00     1.00       4.00       2.876
750.00     1.00       5.00       16.63
1000.0     1.00       6.00       4.678
%
%Конец файла
```

Пример 2. Файл ковариаций.

```
%Имя файла: data.cov, тип: ковариационный V1.0, 06-Ноябрь-1999, 11:51
%Механическая система 2-го порядка
%
1.18e-2        1.62e-3        1.93e-3
1.24e-2        1.456e-3
2.19e-2
%Конец файла
```

В пакете Frequency Domain Identification имеются функции импорта/экспорта файлов всех типов и преобразования (если это возможно) из одного типа в другой (см. ниже).

Функции пакета Frequency Domain Identification

В рассматриваемый пакет входят около 50 функций, объединенные в следующие группы:

- проектирования идентифицирующих сигналов;
- предварительной обработки данных;
- оценивания;
- представления результатов;
- проверки адекватности модели;

- чтения/записи данных;
- преобразования моделей;
- прочие.

Функции проектирования идентифицирующих сигналов

В состав этой группы входят следующие 8 функций.

Функция dibs

Функция dibs синтезирует сигнал в виде псевдослучайной дискретной бинарной последовательности с нулевым математическим ожиданием. Она записывается в виде:

```
bitser = dibs(N,dt,freqv,ampv)
bitser,ampopt,Puf,Ptot] = dibs(N,dt,freqv,ampv,trialno,graphmod)
```

Здесь:

- N — длина последовательности;
- dt — интервал дискретизации;
- freqv и ampv — векторы-строки, задающие частоты и соответствующие значения спектральной плотности проектируемого сигнала (по умолчанию ampv = ones(size(freqv)));
- trialno — число, определяющее количество запусков алгоритма синтеза при случайных начальных условиях, по умолчанию 25; данные каждой итерации выводятся на дисплей, если это не отменено значением graphmod = 'nograph' (по умолчанию graphmod = 'graph').

Возвращаемые величины:

- bitser — синтезированные значения бинарного сигнала (имеющие значения +1 или −1);
- ampopt — комплексные амплитуды возвращаемого сигнала для частот freqv, нормируемые таким образом, чтобы мощность моделируемого сигнала равнялась заданной мощности, определяемой по вектору ampv;
- Puf — мощность сигнала на частотах freqv (в долях от общей мощности сигнала);
- Ptot — общая мощность сигнала, найденная по вектору ampv.

Кроме того, возвращаются график сигнала во временной форме, непрерывный сигнал, обладающий заданным спектром, спектр моделируемого сигнала, заданный спектр.

Функция dibsimpr

Функция dibsimpr итерационным образом «улучшает» синтезированную с помощью предыдущей функции бинарную последовательность bits0, приближая ее спектр к заданному:

```
bitser,ampopt] = dibsimpr(bits0,dt,freqv,ampv)
bitser,ampopt,Puf,Ptot] = ...
    dibsimpr(bits0,dt,freqv,ampv,itno,graphmod)
```

Аргументы и возвращаемые величины — как у предыдущей функции. Число итераций для «улучшения» сигнала задается аргументом itno (по умолчанию 1).

Рассмотрим пример. Пусть требуется смоделировать бинарную дискретную последовательность с равномерным спектром в диапазоне частот от 500 Гц до 2 кГц (с дискретностью частот 100 Гц). В качестве частоты дискретизации выберем учетверенное значение верхней частоты заданного диапазона. Соответствующая программа приведена ниже.

```
» freqv=[500:100:2e3]; %Задание частот
» fs=4*2e3; %Расчет частоты дискретизации
» N=round(fs/100); %Расчет длины последовательности: N = 80
» bits0=dibs(N,1/fs,freqv,[],10); %Моделирование с помощью
                                  %функции dibs
» bitser=dibsimpr(bits0,1/fs,freqv); %Моделирование с помощью
                                    %функции dibsimpr
dibsimpr cycle 0, worst amplitude: 66.5%   useful power: 76.5%
Cycles (79):
.........:.........:.........:.........:.........:
.........:.........:.........
dibsimpr cycle 1, worst amplitude: 72.2% (+5.7%)
   useful power: 75.2%
```

Результаты использования функций dibs и dibsimpr в данном примере отражены на рис. 2.2 и 2.3, соответственно.

На рисунках показаны перечисленные при описании функции dibs сигналы и спектры. Кроме того, на них (см. рис. 2.2) представлена следующая информация:

- trial — порядковый номер запуска алгоритма моделирования, при котором был получен наилучший (с точки зрения качества синтезируемого сигнала) результат;
- iter — порядковый номер итерации при этом запуске;
- trials — общее количество запусков, определяемое аргументом trialno;

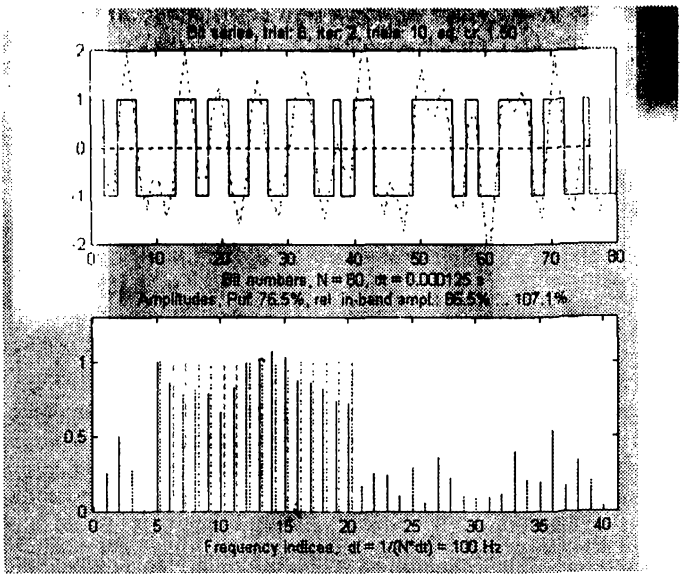


Рис. 2.2. Отображение сигнала, смоделированного с помощью функции dibs

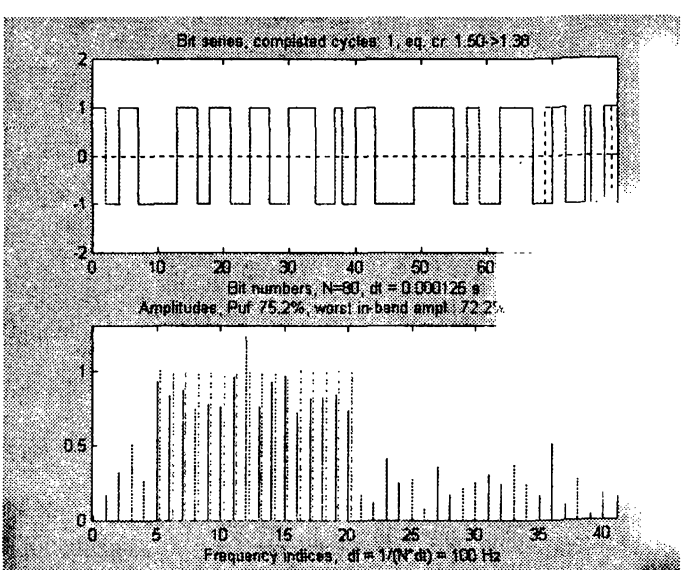


Рис. 2.3. Тот же сигнал, «улучшенный» с помощью функции dibsimpr

- eq. cr — эквивалентный *пик-фактор* (equivalent crest factor — *показатель формы*), определяющий «качество» возвращаемого процесса и рассчитываемый по формуле

  ```
  eqcr = max(abs(ampv./ampopt))
  ```

  где ./ — оператор MATLAB для поэлементного деления векторов (идеальное «качество» имеет место при eqcr = 1).

На графике частотных спектров заданные амплитуды отображены пунктирными линиями, фактические — сплошными линиями.

Приведены следующие значения:

- Puf — часть общей мощности сигнала, которая сконцентрирована на заданных частотах;
- rel. in-band ampl. — минимальное и максимальное значения относительных амплитуд abs(ampv./ampopt) в процентах (фактические амплитуды масштабируются, чтобы иметь ту мощность сигнала, которая задана вектором ampv).

Ось частоты масштабируется в «индексах частоты» df = 1/(N*dT).

Аналогичная информация приводится на графиках, возвращаемых функцией dibsimpr.

Функция lin2qlog

Функция lin2qlog возвращает вектор-столбец частот, соотносящихся друг с другом в заданном отношении (то есть выполняет преобразование линейной шкалы частот в логарифмическую):

```
fqlog = lin2qlog(freqv,rf)
[fqlog,df,cdmax,freqind] = lin2qlog(freqv,rf)
```

Здесь:

- freqv — вектор-строка, содержащий исходный (равномерно возрастающий) набор частот;
- rf — отношение, примерно с которым (или большим) должны возрастать отобранные частоты.

Возвращаемые величины:

- fqlog — вектор-столбец отобранных значений частот;
- df — интервал между соседними частотами в исходном их наборе;
- cdmax — множитель, позволяющий определить наименьший общий период гармоник с отобранными частотами в соответствии с выражением 1/(cdmax*df);

- freqind — индексы отобранных частот (их порядковые номера в ряду, который задается вектором freqv).

Примеры:

```
» fqlog = lin2qlog([1:16],3)
fqlog =
     1
     3
     9
    16
» [fqlog,df,cdmax,freqind]=lin2qlog([2:16],2)
fqlog =
     2
     4
     8
    16
df =
     1
cdmax =
     2
freqind =
     1
     3
     7
    15
```

Функция log2qlog

Функция log2qlog возвращает заданное число частот из некоторого набора исходных:

```
fqlog = log2qlog(freqv,mhno)
[fqlog,df,cdmax,freqind] = log2qlog(freqv,mhno)
```

Здесь аргумент freqv — как в предыдущей функции, mhno — заданное число частот. Возвращаемые величины аналогичны рассмотренным для предыдущей функции, но здесь отбор производится по соотношению

```
max(freqv)*[1:mhno]/mhno
```

Пример:

```
» fqlog=log2qlog([1:16],5)
fqlog =
    3.2000
    6.4000
    9.6000
   12.8000
   16.0000
```

Функция mlbs

Функция mlbs генерирует бинарную последовательность максимальной длины (псевдослучайный двоичный сигнал, ПСДС):

```
bitseries = mlbs(log2N)
[bitseries,nextstnum] = mlbs(log2N,bitno,startnum)
```

Здесь:

- log2N — количество разрядов сдвигового регистра для формирования ПСДС (минимальное значение — 2, максимальное — 30);
- bitno — количество бит генерируемой последовательности, по умолчанию 2^log2N-1 (это значение равно периоду ПСДС);
- startnum — начальное состояние сдвигового регистра.

Возвращаемые величины:

- bitseries — значения ПСДС;
- nextstnum — конечное состояние сдвигового регистра (может использоваться для продолжения моделирования).

Пример:

```
» bitseries = mlbs(3) %Формирование ПСДС с периодом 7 тактов
bitseries =
    -1
    -1
     1
    -1
     1
     1
     1
```

Функция msinclip

Функция msinclip синтезирует полигармонический сигнал с минимальным значением пик-фактора (показателя формы) (см. выше) с использованием алгоритма отсечения путем итерационного подбора фазовых сдвигов:

```
cx = msinclip(freqv,ampv)
[cx,crx,crxmax,cry,crymax] = ...
    msinclip(freqv,ampv,tf,gmod,itno,ovs,N,clO)
```

Аргументы функции следующие:

- freqv и ampv — векторы-строки, задающие частоты (которые должны быть кратными некоторой величине df; минимальное число частот — 2) и соответствующие абсолютные величины желаемых комплексных амплитуд (они вдвое меньше, чем величины вещественных амплитуд);

- tf — вектор-строка, содержащий значения комплексного коэффициента передачи на заданных частотах; при введении данного аргумента минимизация проводится по сигналам «вход-выход»;
- gmod — строковая переменная, которая может принимать значения:
  - 'nograph' — при этом не выводятся графики синтезируемого сигнала, отражающие итерационный процесс минимизации;
  - 'lastgraph' — выводится график, соответствующий последней итерации;
  - 'graph10' или 'graph100' — график выводится после каждой 10-й или 100-й итерации;
  - 'graph' — выводится результат каждой итерации;
  - 'graphZOH' или 'ZOH' — выходной сигнал формируется как кусочно-постоянный;
- itno — определяет число итерационных циклов;
- аргумент ovs косвенно определяет точность вычисления пик-фактора; в частности, частота дискретизации устанавливается по соотношению $fs \geq ovs*2*f_{max}$;
- N — задаваемая длина временной последовательности;
- последний аргумент, c10, позволяет пользователю задать параметр алгоритма минимизации как число из интервала (0,1).

Значения по умолчанию:

- ampv = ones(length(freqv),1);
- tf = []; '
- gmod = 'graph10';
- itno = 250;
- ovs = 16 (при синтезе кусочно-постоянного сигнала ovs = 1).

Возвращаемые величины:

- вектор cx содержит комплексные амплитуды синтезированного сигнала;
- crx — пик-фактор синтезированного сигнала;
- crxmax — максимально возможное значение пик-фактора (коэффициента формы).

Пример использования рассматриваемой функции представлен ниже, а результат синтеза сигнала показан на рис. 2.4.

```
» %Синтез сигнала, состоящего из 5 гармоник
» [cx,crestx]=msinclip([1:5]',ones(5,1))
```

```
Bandlimited multisine design
df=1, fmax=5*df, fsmin=12*df
Prescribed oversampling: 16
Sample number rounded up to nearest power of 2, oversampling: 25.6
FFT length in msinclip: 256
Iteration=200, crestx=1.5532, crestxmax=1.5558, found in cycle 200, clipx=0.892
cx =
   0.9992 - 0.0390i
  -0.1477 - 0.9890i
  -0.9394 + 0.3428i
   0.3832 - 0.9237i
```

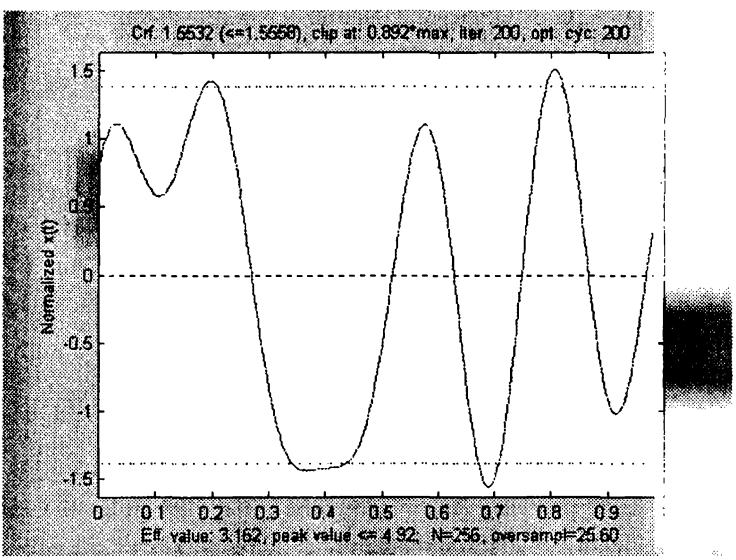


Рис. 2.4. Сигнал, синтезируемый с помощью функции msinclip

Функция msinprep

Функция msinprep синтезирует полигармонический сигнал и выдает его значения на периферийное устройство:

```
xtim = msinprep(freqv,cx)
[xtim,df] = msinprep(freqv,cx,N,fs,dev)
```

Здесь:

- freqv — вектор-строка заданных частот;
- cx — вектор-строка заданных комплексных амплитуд (например, вектор, полученный с помощью функции msinclip);

- N — длина временной последовательности. Если значение N не задано, оно определяется по соотношению N = fs/df, где df — максимальный общий делитель частот, заданных в векторе freqv;
- fs — частота дискретизации, определяемая как fs = N*df. Если значение N не задано, частота дискретизации определяется как fs = = df*2*(max(freqv)/df+1);
- dev — строковая переменная, определяющая периферийное устройство для вывода синтезированного сигнала. Возможные значения:
  - dev = 'screen' — вывод на дисплей;
  - dev = 'DAC' — вывод на цифро-аналоговый преобразователь (значение по умолчанию).

Возвращаемые величины:

- xtim — вектор значений синтезированного сигнала;
- df — максимальный общий делитель частот, заданных в векторе freqv.

Функция optexcit

Функция optexcit возвращает оптимальный спектр сигнала для идентификации объекта, то есть решает задачу планирования оптимального активного эксперимента:

```
X = optexcit(pdat,freqv)
[X,CR,fsv,vXwdev,Fiw] = ...
    optexcit(pdat,freqv,vdat,fixpind,X0,Ncyc,Fiw,pd)
```

где:

- pdat — вектор параметров передаточной функции объекта (имя соответствующего файла);
- freqv — вектор заданных частот;
- vdat — вектор или матрица ковариаций входного и выходного сигналов;
- fixpind — вектор, содержащий индексы зафиксированных параметров в векторе-строке;
- pdat = [num,denom,delay] (здесь num — вектор-строка коэффициентов числителя передаточной функции, denum — ее знаменателя, delay — величина запаздывания). Если fixpind='n', то все параметры передаточной функции считаются свободными, то есть подлежащими оцениванию;
- X0 — вектор начальных значений амплитуд сигнала с оптимальным спектром;

- Ncyc — число итераций алгоритма поиска оптимального спектра;
- Fiw — массив матриц вспомогательной информации, полученный на предыдущих этапах оптимизации и используемый для ускорения расчетов текущего этапа;
- аргумент pd определяет частоту обновления графиков (pd = 0 — графики обновляются, pd = inf — графики не выводятся).

Значения по умолчанию:

- vdat = [1,1];
- fixpind = [nn+nd,nn+nd+1] — для передаточных функций, заданных для аргумента $\Omega = j\omega$; fixpind = [nn+1,nn+nd+1] — для передаточных функций, заданных для аргумента $e^{-j\omega T}$ (nn — число коэффициентов числителя, nd — число коэффициентов знаменателя передаточной функции; коэффициент a_0 в знаменателе и величина запаздывания T_d считаются фиксированными, то есть заданными). При fixpind = [ ] все коэффициенты считаются свободными;
- X0 = ones(F,1)/sqrt(F), где F — длина вектора частот;
- Ncyc = 1.

Возвращаемые величины:

- X — вектор амплитуд сигнала с оптимальной спектральной плотностью;
- CR — нижняя граница Крамера—Рао для матрицы ковариаций оцениваемых параметров;
- fsv — масштабирующий вектор, используемый для уточнения границы Крамера—Рао по формуле CRscaled = CR.*(fsv*fsv');
- vXwdev — двухэлементный вектор-строка, содержащий минимальное и максимальное значения дисперсионной функции;
- Fiw — см. описание входных параметров. Если данный массив слишком велик, возвращается пустой вектор, а пересчет Fiw производится на каждой итерации.

Функции предварительной обработки данных

Функция modifyfv

Функция modifyfv осуществляет фильтрацию входных данных путем пропускания их через динамическое звено, передаточная функция которого равна инверсной передаточной функции известного сомножителя передаточной функции объекта:

```
Fdatm = modifyfv(pdat,Fdat)
[Fdatm,vdatm] = modifyfv(pdat,Fdat,vdat,plotmode)
```

Если передаточная функция идентифицируемого объекта частично известна — в форме некоторого ее сомножителя — то нет необходимости оценивать коэффициенты этой известной части, в связи с чем для исключения такой процедуры входной сигнал модифицируют, пропуская через фиктивное динамическое звено с передаточной функцией, равной обратной (инверсной) передаточной функции известного сомножителя.

Аргументы этой функции следующие:

- `pdat` — вектор параметров известного сомножителя (имя соответствующего файла);
- `Fdat` — массив комплексных амплитуд входного сигнала или имя соответствующего Фурье-файла;
- `vdat` — массив ковариаций входных данных или имя соответствующего файла дисперсий;
- `plotmode` — строковая переменная, определяющая масштаб оси частот на графике выводимого результата:
  - при значениях `'lin'` или `'linpb'` — линейный масштаб (по умолчанию `plotmode = 'linpb'`);
  - при `'log'` или `'logpb'` — логарифмический масштаб;
  - при любом другом значении график не выводится.

Возвращаемые величины `Fdatm` и `vdatm` — соответственно, массив амплитуд преобразованного сигнала и массив его дисперсий.

Функция tim2fou

Функция `tim2fou` преобразует данные из временной области в частотную:

```
[x,y] = tim2fou(tdat,freqv)
[x,y,fv] = tim2fou(tdat,freqv,expi)
```

где:

- `tdat` — вектор данных или имя содержащего их m-файла (например, `tdat = 'gettim'`) или временного файла (с расширением `tbn`, `tim` или `tnt`);
- `freqv` — вектор заданных частот;
- `expi` — вектор-строка, содержащий номера используемых экспериментов.

Возвращаемые величины:

- x и y — комплексные амплитуды входного и выходного сигналов при их частотном представлении;
- fv — вектор фактически используемых частот.

Функция varanal

Функция varanal производит усреднение экспериментальных значений комплексных амплитуд и рассчитывает оценки их ковариаций:

```
[vx,vy,cxy] = varanal(Fdat,expi)
[vx,vy,cxy,mx,my,Na,Np,cfl,dv,sd] = ...
    varanal(Fdat,expi,synch,T,inp,outp)
```

Здесь:

- Fdat — содержит коэффициенты Фурье-разложений входного и выходного сигналов;
- expi — вектор-строка, содержащий номера используемых экспериментов;
- T — величина периода (для периодических сигналов);
- synch — строковая переменная, значение которой 'synch' позволяет «синхронизировать» входные и выходные данные, что приводит к повышению качества оценок;
- inp и outp — номера (или векторы-строки, элементы которых являются номерами) входов и выходов, для которых проводится обработка.

Возвращаемые величины:

- vx и vy — векторы, элементы которых являются дисперсиями оценок комплексных амплитуд входа и выхода;
- cxy — вектор ковариаций между входом и выходом, рассчитываемый по формуле $0.5E\{\overline{N}_x N_x\}$;
- mx и my — векторы средних значений амплитуд;
- Na — количество усредняющих экспериментов;
- Np — общее количество экспериментальных данных (Na и Np могут отличаться при введении синхронизации);
- cfl — множитель (двухэлементный вектор-строка) для определения 95 %-го доверительного интервала для дисперсий. Ширина такого интервала определяется по соотношению (cfl-1)*sqrt(varx(k)*vary(k)), где sqrt — обозначение квадратного корня, varx(k) и vary(k) — обозначения дисперсий;

dv — аргумент, используемый при введении синхронизации. Представляет собой вектор задержек, удовлетворяющих неравенству -T/2 < delay ≤ T/2;

sd — граница Крамера—Рао для величин задержек.

Функции оценивания

В данную группу входят следующие 8 функций.

Функция elis

Функция elis выполняет оценивание параметров передаточной функции, реализуя описанный выше метод ELiS:

```
elis(Fdat,vdat,rppar)
[pvect,fit,Cp,CR,cfv] = elis(Fdat,vdat,rppar,fixp,rpalg,rppl,initp,rpfs)
```

Здесь:

- Fdat — массив комплексных амплитуд входного и выходного сигналов или имя соответствующего Фурье-файла;
- vdat — массив ковариаций входных и выходных данных или имя соответствующего файла дисперсий;
- rppar — вектор параметров, элементы которого имеют значения:
  - rppar(1) — строковая переменная, задающая область Ω (значение 's') или $e^{j\omega T}$ (значение 'z');
  - rppar(2) — порядок числителя;
  - rppar(3) — порядок знаменателя;
  - rppar(4) = fs: масштабирующий общий множитель для шкалы частот;
  - rppar(5) = 'a';

  (должны быть заданы по крайней мере rppar(1)...rppar(3));
- аргумент fixp определяет фиксированные (неоцениваемые) параметры и представляет собой массив размером M×2, первый столбец которого задает номера фиксированных параметров вектора [num, denom,delay], а второй — значения параметров. Пусть, например, имеем передаточную функцию, в которой порядок числителя — 4, знаменателя — 8, а величина запаздывания равна нулю. Вектор параметров в этом случае включает в себя 4+1+8+1+1=15 элементов. Фиксация a_0 = 1 и T_d = 0 осуществляется заданием fixp = = [6,1;15,0], а фиксация только нулевой задержки — заданием fixp = [15,0].

Векторный аргумент `rpalg` задает алгоритм поиска минимума и па раметры итерационной процедуры. Его элементы:

- `rpalg(1)` — строковая переменная со значениями:
  - `'g'` (задается алгоритм Ньютона—Гаусса);
  - `'l'` (алгоритм Левенберга—Марквардта — LM);
  - `'s'` (так называемый алгоритм декомпозиции сингулярны чисел);
  - `'m'` (модификация LM — с алгоритмом декомпозиции сингу лярных чисел);
  - `'r'` (Ньютона—Рафсона);
- `rpalg(2)` определяет способ определения начальных значений:
  - `'l'` (обычный метод наименьших квадратов — МНК);
  - `'w'` (взвешенный МНК);
  - `'s'` (взвешенный МНК и алгоритм декомпозиции сингулярны чисел);
  - `'f'` (файл или вектор параметров);
  - `'e'` (метод уравнений ошибки пакета Signal Processing);
- `rpalg(3)` — задает максимальное число итераций;
- `rpalg(4)` — процесс вычислений останавливается, если относитель ная вариация функции потерь меньше, чем данная величина;
- `rpalg(5)` — процесс вычислений останавливается, если относитель ная вариация всех параметров меньше, чем данная величина.

Следующие элементы относятся к параметрам алгоритма Левенбер га—Марквардта:

- `rpalg(6)` — число последовательных уменьшений функции потер до испытания со значением параметра алгоритма `lambda = 0`;
- `rpalg(7)` — начальное значение `lambda`;
- `rpalg(8)` — минимально возможное значение `lambda`, при которо возможен останов алгоритма.

Аргумент `rppl` определяет формат графика:

- `rppl(1)` — определяет частоту обновления графика (например, по сле каждой 10-й или 100-й итерации и т. п.);
- `rppl(2:5) = [fmin,fmax,amin,amax]` — вектор, задающий масштаб осей графика;

- rppl(6) — масштаб оси частот, 'l' — линейный, 'o' — логарифмический;
- rppl(7) — задание вычисления ('c') нулей числителя (при rppl(7) = 'n' нули не вычисляются);
- rppl(8) — то же для полюсов знаменателя (возможные значения 'c' или 'n');
- rppl(9:12) — четырехэлементный вектор-строка для задания осей графика нулей и полюсов;
- rppl(13) — определяет число возможных наборов нулей и полюсов, отображаемых на графике.

Аргумент initp может иметь различное назначение: это может быть вектор начальных значений параметров или имя файла с такими параметрами; если initp — скаляр, то это величина запаздывания.

Аргумент rpfs — имена генерируемых файлов.

Возвращаемые величины:

- pvect — вектор параметров модели;
- fit — вектор, элементы которого содержат информацию о качестве полученной модели:
  - fit(1) — значение функции потерь;
  - fit(2) — теоретическое значение функции потерь;
  - fit(3) — 95 %-я точка теоретического распределения функции потерь;
  - fit(4) — число частот;
  - fit(5) — число свободных параметров;
  - fit(6) — число проделанных итераций;
  - fit(7) — последнее изменение функции потерь;
  - fit(8) — последний максимум относительного изменения параметров;
  - fit(9) — последнее значение параметра lambda;
  - fit(10) — аппроксимация ошибки модели;
  - fit(11) — среднее абсолютное значение передаточной функции;
  - fit(12) — критерий Акейке (Akaike criterion — см. главу 1);
  - fit(13:15) — нормы обусловленности ряда вспомогательных матриц алгоритма;
  - fit(16) — значение масштабирующей частоты;

- Cp — оценка матрицы ковариаций параметров;
- CR — нижняя граница Крамера–Рао для матрицы ковариаций параметров;
- cfv — вектор значений функции потерь в каждом цикле. В случае использования алгоритма Левенберга–Марквардта cfv — матрица из двух столбцов, при этом второй столбец содержит значения параметра алгоритма lambda.

Кроме этого, выводятся графики АЧХ, ФЧХ и нулей/полюсов.

Необходимо указать, что использование рассмотренной функции достаточно сложно и нуждается в составлении соответствующего файла отчета.

Пример использования функции elis приведен ниже, а выводимые функцией графики показаны на рис. 2.5.

```
» [pvect,fit,Cp]=elis('inpchans.ebn'); %Оценивание модели и построение графиков
elis is running ...
elis is loading run parameters from file 'inpchans.ebn', date: 20-Dec-93
impfou loading data from binary file 'inpchan.fbn'
   Experiment: 1, frequencies: 49, from 4.000e+002 to 1.960e+004 Hz
   Comments: Data acquisition channel, sigmax=3.1e-6, sigmay=31e-6
   Number of frequencies: 49
   Maximum of the frequency vector: 19600 Hz
   Scaling angular frequency for internal calculations: 125000 rad*Hz
   Number of estimated parameters: 26, model: 12/12, s-domain
Start time: 16:40:43
   Setting the starting values, mode: svd ...
   Norm=1
   Cost function: 2509.4, cfth: 36.5
   Total run time: 0.01 min, plots: 0.01 min
------------------
Time: 16:40:44
   Beginning the iteration algorithm: svd
  Norm=1, delay fixed
------------------
Iteration cycle: 1
   Costf: 2125.1, cfth: 36.5, dcf: -384.35, stop for |dcf/cf| < 1e-006
   Total run time: 0.02 min, 0.02 min/cycle, plots: 0.02 min
Iteration cycle: 2
   Costf: 2124.8, cfth: 36.5, dcf: -0.26775, stop for |dcf/cf| < 1e-006
   Total run time: 0.03 min, 0.02 min/cycle, plots: 0.03 min
Iteration cycle: 3
   Costf: 2124.8, cfth: 36.5, dcf: -0.00038294, stop for |dcf/cf| < 1e-006
   Total run time: 0.04 min, 0.01 min/cycle, plots: 0.04 min
Iteration termination criterion is met
Variation of the cost function is sufficiently small
```

```
A last cycle follows for calculation of the covariances
   Total run time: 0.05 min. 0 01 min/cycle, plots: 0.04 min
Total run time: 0.05 min, time for plots: 0.04 min
```

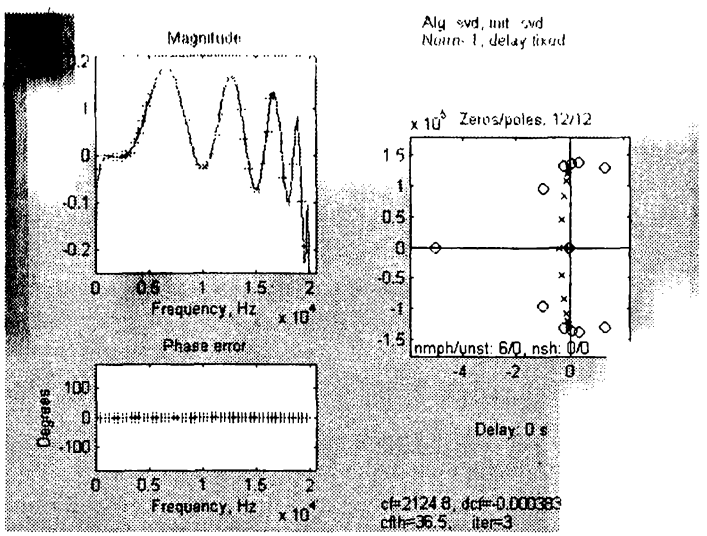


Рис. 2.5. Графики, возвращаемые функцией elise

Функция eliscost

Функция eliscost вычисляет величину функции потерь:

```
costf = eliscost (pdat,Fdat,vdat)
```

Аргументы функции:

- pdat — вектор параметров модели (см. описание функции exppar далее в разделе «Функции чтения/записи данных»);
- Fdat — вектор Фурье-коэффициентов данных (см. описание функции expfou далее в разделе «Функции чтения/записи данных»);
- vdat — ковариации данных (см. описание функции expvar далее в разделе «Функции чтения/записи данных»).

Возвращаемая величина costf — значение функции потерь.

Функция elisml

Функция elisml оценивает параметры передаточной функции объекта методом максимального правдоподобия:

```
[pvect,fit,Cp]=elisml(Fdat,vdat,nord,dord,delay,dfx)
```

Рассматриваемая функция использует функцию `elis` с большинством аргументов по умолчанию, что несколько упрощает процедуру оценивания. Ее аргументы:

- `Fdat` — входные и выходные данные в Фурье-формате (`[freq,x,y]`, массив размером N×3, где N — количество анализируемых частот);
- `vdat` — ковариации входного и выходного сигналов (или массив размером N×2 вида `[varx,vary]`, или массив размером N×3 вида `[varx, vary,covxy]`, или двухэлементный вектор-строка вида `[vx0,vy0]`, или трехэлементный вектор-строка вида `[vx0,vy0,cxy0]`;
- `nord` — порядок числителя передаточной функции;
- `dord` — порядок ее знаменателя;
- `delay` — величина запаздывания в секундах (по умолчанию 0);
- `dfx` = `'f'` для фиксированного параметра `delay`, `'v'` — для оцениваемого (по умолчанию `'f'`).

Возвращаемые величины:

- `pvect` — вектор параметров модели (для его расшифровки используется функция `imppar`, см. далее);
- `fit` — информация о качестве модели (см. описание функции `elis` выше);
- `Cp` — оценка матрицы ковариаций параметров.

Функция elistper

Функция `elistper` оценивает параметры передаточной функции с использованием данных (полигармонических сигналов), представленных во временной области:

```
[pvect,fit,Cp,Fdat,vdat]=elistper(xt,yt,fs,fv,nord,dord,delay,dfx,mr,pd)
```

Здесь:

- `xt` и `yt` — соответственно, векторы значений входного и выходного сигналов объекта во временной области;
- `fs` — частота дискретизации экспериментальных данных;
- `fv` — вектор частот гармоник входного сигнала;
- `nord`, `dord`, `delay`, `dfx` — как у предыдущей функции;
- `mr` — аргумент для коррекции «неидеальных ситуаций», например, в случае неточно заданных `fs` и/или `fv`; как правило, используется по умолчанию;
- `pd` — определяет частоту обновления графиков (`pd = 0` — графики обновляются, `pd = inf` — графики не выводятся).

Функция elisqa

Функция elisqa предназначена для создания (в режиме диалога с пользователем) файла начальных значений параметров для функции elis.

```
rpfout = elisqa(rpf,defaults)
```

Здесь:

- rpf — имя mat-файла для генерируемых параметров; расширение по умолчанию — ebn;
- defaults — имя mat-файла со значениями параметров по умолчанию.

Пример вызова функции:

```
elisqa('elisqtst.ebn','inpchans.ebn');
```

Функция elrpf2v

Функция elrpf2v преобразует файл параметров для исполнения функции elis в вектор таких параметров:

```
[rppar,fixp,rpalg,rppl,initp,rpfs]=elrpf2v(rpfile)
```

Здесь:

- rpfile — имя файла параметров (с расширением ebn);
- rppar — вектор описания модели;
- fixp — описание фиксируемых параметров;
- rpalg — вектор описания алгоритма оценивания;
- rppl — вектор параметров графика;
- initp — начальные значения параметров;
- rpfs — имена генерируемых файлов.

Возвращаемые величины: см. описание функции elis.

Функция elrpv2f

Функция elrpv2f является обратной по отношению к предыдущей, то есть преобразует вектор параметров функции elis в файл:

```
elrpv2f(rpfile,rppar,fixp,rpalg,rppl,initp,rpfs)
```

Аргументы — как для функции elis.

Функция gmean

Функция gmean возвращает геометрическое среднее комплексных чисел:

```
gm = gmean(X)
```

Здесь аргумент X — матрица комплексных чисел. Возвращаемая величина gm — вектор-строка, элементы которого являются геометрическими средними комплексных чисел по столбцам матрицы X.

Функции представления результатов

Для представления результатов при идентификации систем в частотной области служат приведенные ниже функции.

Функция expvect

Функция expvect экспортирует векторы данных в файл текстового формата (ASCII) для последующего графического отображения:

```
expvect(file,v1)
expvect(file,v1,digitnum)
expvect(file,v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12,digitnum)
```

Данная функция возвращает данные в текстовом формате, например, для последующего построения графиков с помощью соответствующих программ инженерной графики. Она имеет следующие аргументы:

- file — имя создаваемого файла;
- v1...v12 — вещественные векторы данных;
- digitnum — количество цифр для представления чисел (по умолчанию 7).

Возвращаемая величина — текстовый файл с указанным именем.

Приводимая ниже программа создает набор отметок времени (вектор t) и набор комплексных чисел x, а затем сохраняет данные в виде файла с именем data.txt.

```
» t = [1:100]; x = 100*ones(100,2)+rand(100,2)+j*rand(100,2);
» expvect('data.txt',t,real(x(:,2)),imag(x(:,2)),8)
```

Функция ploteltf

Функция ploteltf возвращает графики АЧХ и ФЧХ комплексного коэффициента передачи:

```
ploteltf(pdat1)
[ha1,ha2,fsc] = ploteltf(pdat1,pdat2,Fdat,fscale,msc,cdl,cc,cn,rec,expi,ntx)
```

Здесь:

- pdat1 и pdat2 — векторы параметров модели (см. далее описание функции exppar в разделе «Функции чтения/записи данных») или имена файлов, содержащих такие параметры;
- Fdat — массив частот и соответствующих Фурье-коэффициентов входного и выходного сигналов (см. описание функции elisml);

- fscale — определяет масштаб оси частот:
  - fscale = 'lin' — линейный масштаб частот (по умолчанию);
  - fscale = 'linF' — линейный масштаб с максимальной частотой из Fdat;
  - fscale = ['lin',f1,f2] — линейный масштаб с диапазоном частот от f1 до f2;
  - fscale = 'log' — логарифмический масштаб;
  - fscale = 'logF' — логарифмический масштаб с максимальной частотой из Fdat;
  - fscale = ['log',f1,f2] — логарифмический масштаб с диапазоном частот от f1 до f2;
- msc — задает масштабирование на графике АЧХ (при значении 'full' — по всем точкам, при значении 'passb' — только для точек, где выход объекта отличен от нуля);
- cdl — матрица ковариаций (или вектор ковариаций, или имя файла), относящаяся к pdat1;
- cc — коэффициент для определения доверительного интервала (множитель стандартного отклонения);
- cn — число точек графика для отображения доверительных интервалов;
- rec = 'abc', где первая буква аббревиатуры abc относится к pdat1, вторая — к pdat2, третья — к Fdat. Значения символов задают вычисление и вывод графика соответствующей обратной характеристики (при rec = 's' такой расчет не производится, при rec = 'r' расчет проводится до построения графика);
- expi — количество отображаемых на графике экспериментов;
- если ntx = 'notext', то имя файла не отображается.

Большинство приведенных аргументов может быть опущено.

Значения по умолчанию:

```
pdat2=''; Fdat=''; fscale='lin'; msc='full'; cdl=''; cc=2; cn=32; rec='sss';
expi=[]; ntx=''
```

Возвращаемые величины:

- ha1, ha2 — дескрипторы графиков АЧХ и ФЧХ, создаваемые в целях последующего отображения графиков;
- fsc — единица масштаба частот.

Кроме того, функция производит построение графиков АЧХ и ФЧХ.

Использование функции иллюстрируют приведенная ниже программа и рис. 2.6, 2.7.

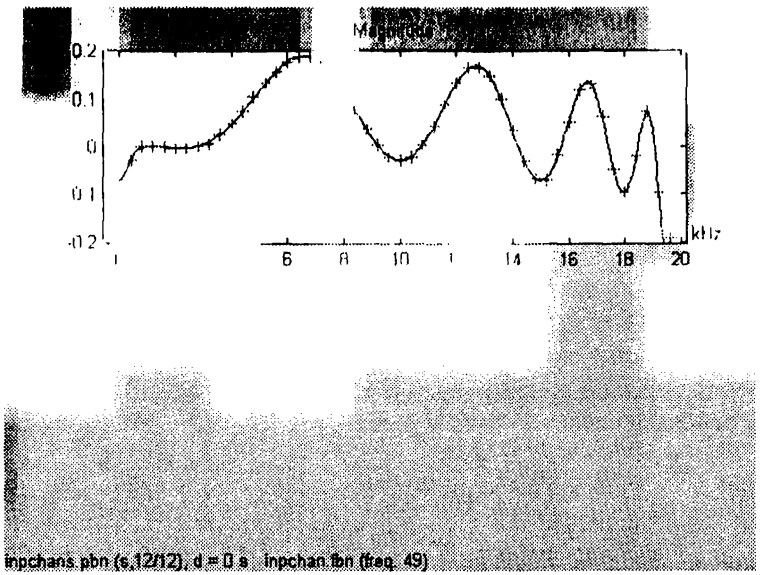


Рис. 2.6. График, построенный с помощью функции ploteltf

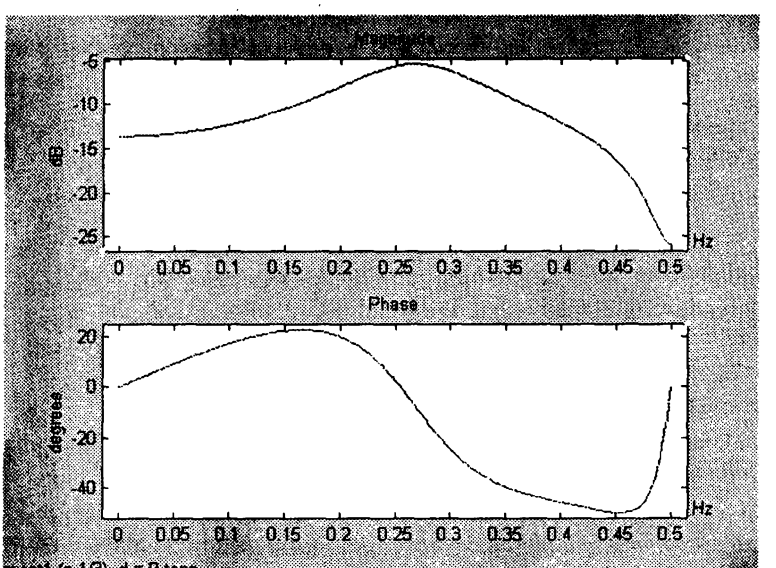


Рис. 2.7. Графики к примеру применения функции ploteltf

```
» %Загрузка данных и построение рис. 2.6
» ploteltf('inpchans.pbn','','inpchan.fbn','linF','full+');
imppar loading data from binary file 'inpchans.pbn'
   Orders: 12/12, delay: 0.0000e+000 s, s-domain
impfou loading data from binary file 'inpchan.fbn'
   Experiment: 1, frequencies: 49, from 4.000e+002 to 1.960e+004 Hz
   Comments: Data acquisition channel, sigmax=3.1e-6, sigmay=31e-6
» %Задание передаточной функции
» num=[1.1.1];denom=[4,3,2,1];
» %Построение графиков рис. 2.7
» [hal,ha2,fsc]=ploteltf(exppar('z',num,denom,0));
```

Функция plotelpz

Функция plotelpz отображает график нулей/полюсов передаточной функции вместе с соответствующими доверительными эллипсами:

```
plotelpz(pdat)
[rnum,rdenom] = plotelpz(pdat,axv,cdat,Pc,ntx,parr,zarr,domain,da,dp,plm)
```

Здесь:

- pdat — вектор параметров или имя соответствующего файла (см. далее описание функции exppar в разделе «Функции чтения/записи данных»);
- cdat — матрица (массив) ковариаций или имя ковариационного файла;
- Pc — доверительный уровень для построения доверительных эллипсов (если Pc=0, то доверительные эллипсы на графике не отображаются);
- axv — четырехэлементный вектор параметров графика. Если данный аргумент опущен, график масштабируется таким образом, чтобы отобразить все нули и полюсы, при axv = 'z' максимальные значения оцифровки осей определяются вектором [-2,2,-2,2], при axv = 'p' отображаются все нули и полюсы при одинаковых масштабах осей;
- ntx — строковая переменная, содержащая любой текст, который требуется отобразить на графике (в случае ntx = 'notext' никакой текст не выводится);
- zarr — массив нулей;
- parr — массив полюсов;
- domain = 's' или 'z' — области для нулей и полюсов (соответственно, для непрерывных или дискретных передаточных функций);
- da — строковая переменная для указания вида вычислений ('anal' — аналитические, 'num' — численные);

- dp — вариации параметров при численных вычислениях;
- plm — строковая переменная, позволяющая отобразить взаимные изменения пар нулей и полюсов (значения 'mc' или 'mp' — в этом случае после каждого графика делается пауза).

Значения по умолчанию: axv=[]; cdat=''; ntx=''; da = 'anal'.

Возвращаемые величины:

- график нулей и полюсов;
- rnum — вектор-столбец, содержащий корни полинома числителя (нули) передаточной функции;
- rdenom — вектор-столбец, содержащий корни полинома знаменателя (полюсы) передаточной функции.

Примеры использования рассмотренной функции приведены ниже, а результаты показаны на рис. 2.8 и 2.9.

```
» plotelpz('inpchanz.pbn'); %Пример 1 – см. рис. 2.8
impparloading data from binary file 'inpchanz.pbn'
   Orders: 14/14, delay: 0.000 taps, z-domain, fs=5.120e+004 Hz
»
» [rnum,rdenom]=plotelpz('inpchans.pbn','inpchans.cbn');%Пример 2 – см. рис. 2.9
impparloading data from binary file 'inpchans.pbn'
   Orders: 12/12, delay: 0.0000e+000 s, s-domain
impcovloading data from binary file 'inpchans.cbn'
   Number of parameters: 27, fixed: 1
```

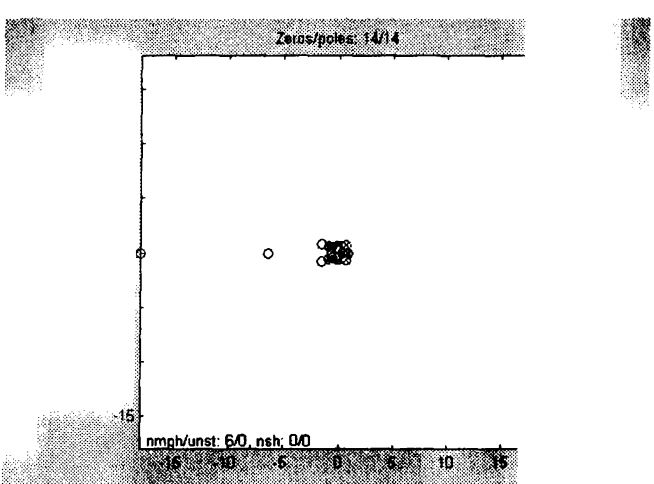


Рис. 2.8. График нулей и полюсов к примеру применения функции plotelpz

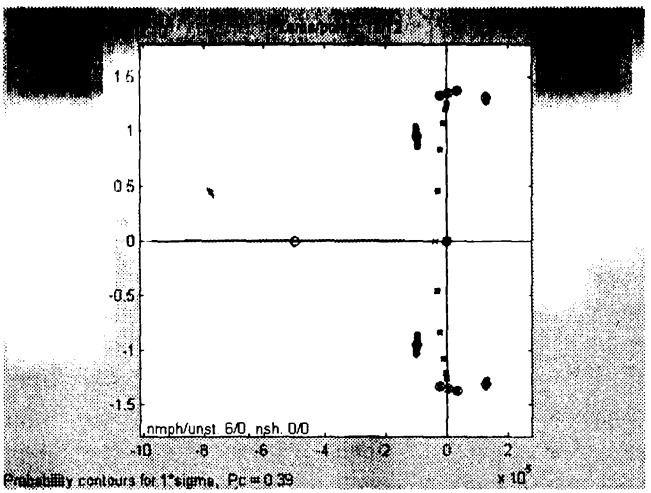


Рис. 2.9. Нули и полюсы вместе с доверительными областями

Функция tfcalc

Функция `tfcalc` вычисляет значения комплексного коэффициента передачи для заданных частот:

```
tf=tfcalc(pdat,freqv)
```

где `pdat` — вектор параметров (см. далее описание функции `expfou` в разделе «Функции чтения/записи данных») и `freqv` — вектор частот.

Возвращаемая величина `tf` — вектор значений комплексного коэффициента передачи.

Функции проверки адекватности модели

К данной группе отнесены следующие 7 функций.

Функция fdcovpzp

Функция `fdcovpzp` возвращает параметры и ковариации передаточной функции по заданным нулям и полюсам:

```
[pvect,Cp] = fdcovpzp(zv,stdz,pv,stdp,rzp,g,stdg,domain,fs,da,dzp)
```

Здесь:

- `zv` — вектор-столбец нулей передаточной функции;
- `stdz` — массив стандартных отклонений нулей; три столбца содержат, соответственно, стандартные отклонения для действительных

частей, стандартные отклонения для мнимых частей и коэффициенты корреляции между действительными и мнимыми частями;

- pv — вектор-столбец полюсов передаточной функции;
- stdp — массив стандартных отклонений полюсов; три столбца содержат, соответственно, стандартные отклонения для действительных частей, стандартные отклонения для мнимых частей и коэффициенты корреляции между действительными и мнимыми частями;
- rzp — матрица коэффициентов корреляции элементов вектора, который содержит вещественные и мнимые части каждого нуля и полюса и основные коэффициенты числителя и знаменателя:
  `[z1r;z1i;z2r;...;p1r;p1i;p2r;...;n1,d1];`
- g — статический коэффициент усиления передаточной функции;
- stdg — стандартное отклонение статического коэффициента усиления;
- fs — частота дискретизации (в случае дискретной модели);
- da — строковая переменная для указания вида вычислений (`'anal'` — аналитические, `'num'` — численные);
- dzp — величины изменений полюсов и нулей в направлениях собственных векторов матрицы ковариаций.

Значения по умолчанию:

```
g=[1;1]; stdg=[0;0;0]; domain='s'; fs=1; da='num'; dzp=1;da = 'anal';
```

Возвращаемые величины:

- pvect — вектор параметров (числитель, знаменатель, запаздывание) — см. далее описание функции exppar в разделе «Функции чтения/записи данных»;
- Cp — матрица ковариаций вектора pvect.

Функция simfou

Функция simfou возвращает Фурье-коэффициенты выходного сигнала по заданным входному сигналу и передаточной функции:

```
[x,y] = simfou(pdat,freqv,x0,vdat)
[x,y] = simfou(pdat,freqv,x0,vdat,expno)
```

Здесь x0 — вектор Фурье-коэффициентов входного сигнала (если он пустой, всем амплитудам присваиваются единичные значения); expno — число используемых экспериментов, по умолчанию 1. Остальные аргументы рассматривались выше, при описании функции tfcalc.

Возвращаемые величины x и y — соответственно, Фурье-коэффициенты входного и выходного сигналов.

Функция simtime

Функция simtime моделирует данные во временной области:

```
[xt,yt] = simtime(pdat,u,numi,denomi,numo,denomo)
[xt,yt] = simtime(pdat,u,numi,denomi,numo,denomo,mode)
```

Здесь:

- pdat — см. описание функции tfcalc;
- u — вектор значений входного сигнала во временной области;
- numi — числитель передаточной функции формирующего фильтра входного шума наблюдений;
- denomi — знаменатель передаточной функции формирующего фильтра входного шума наблюдений;
- numo — числитель передаточной функции формирующего фильтра выходного шума наблюдений;
- denomo — знаменатель передаточной функции формирующего фильтра выходного шума наблюдений;
- mode — строковая переменная, определяющая характер входного сигнала:
  - mode = 'periodic' (по умолчанию) — входной сигнал рассматривается как периодический с представленными значениями, отражающими один период;
  - mode = 'transient' — моделируется реакция объекта при нулевых начальных условиях.

Возвращаемые величины xt и yt — соответственно, векторы значений входного и выходного сигналов модели во временной области.

Функция rdueelis

Функция rdueelis вычисляет остаточную ошибку построенной модели:

```
[rx,ry] = rdueelis(pdat,cdat,Fdat,vdat,expi)
[rx,ry,ryx,vryx,xe,ye] = rdueelis(pdat,cdat,Fdat,vdat,expi,inp,outp)
```

Здесь:

- pdat — вектор параметров или имя соответствующего файла (см. далее описание функции exppar в разделе «Функции чтения/записи данных»);
- cdat — матрица (массив) ковариаций или имя ковариационного файла;

- Fdat — массив комплексных амплитуд входного сигнала или имя соответствующего Фурье-файла;
- vdat — вектор или матрица ковариаций входного и выходного сигналов;
- expi — вектор-строка, содержащий номера используемых экспериментов;
- inp и outp — номера входного и выходного портов, по умолчанию 1.

Возвращаемые величины:

- rx, ry, ryx, vryx, xe, ye — соответственно, комплексная остаточная ошибка для векторов амплитуд входа и выхода, значений комплексного коэффициента передачи (определяемых по формуле ym./xm), вектор вариаций vryx, оценки комплексных амплитуд входного и выходного сигналов.

Функция stdpz

Функция stdpz возвращает стандартные отклонения нулей и полюсов оценки передаточной функции:

```
[zv,stdz,pv,stdp] = stdpz(pdat,cdat)
[zv,stdz,pv,stdp,rzp] = stdpz(pdat,cdat,zv0,pv0)
[zv,stdz,pv,stdp,rzp,g,stdg,dps] = stdpz(pdat,cdat,zv0,pv0,da,plm,dp,axv)
```

Здесь:

- pdat, cdat — рассматривались при описании функции plotelpz;
- zv0 и pv0 — векторы, элементами которых являются, соответственно, нули и полюсы передаточной функции;
- da — строковая переменная для указания вида вычислений ('anal' — аналитические, 'num' — численные);
- plm — строковая переменная, определяющая характер отображения: при значении 'mc' графики будут появляться один сразу после другого, а при значении 'mp' — после паузы;
- dp — вариации параметров при численных вычислениях;
- axv — четырехэлементный вектор параметров графика (см. выше описание функции plotelpz).

Возвращаемые величины:

- zv — вектор нулей передаточной функции;
- stdz — массив стандартных отклонений нулей; три столбца содержат, соответственно, стандартные отклонения действительных частей, стандартные отклонения мнимых частей и коэффициенты корреляции между действительными и мнимыми частями;

- pv — вектор-столбец полюсов передаточной функции;
- stdp — массив стандартных отклонений полюсов; три столбца содержат, соответственно, стандартные отклонения действительных частей, стандартные отклонения мнимых частей и коэффициенты корреляции между действительными и мнимыми частями;
- rzp — матрица коэффициентов корреляции элементов вектора, который содержит вещественные и мнимые части каждого нуля и полюса и основные коэффициенты числителя и знаменателя:

  ```
  [z1r;z1i;z2r;...;p1r;p1i;p2r;...;n1,d1];
  ```

- g — статический коэффициент усиления передаточной функции;
- stdg — стандартное отклонение статического коэффициента усиления;
- dps — предлагаемые значения dp.

Функция stdtf

Функция stdtf возвращает стандартные отклонения АЧХ и ФЧХ:

```
[tf,stda,stdph,stdri,rtf] = stdtf(freqv,pdat,cdat)
```

Аргументы freqv, pdat, cdat уже рассматривались (см. описание функций plotelpz и tfcalc). Возвращаемые величины:

- tf — значения комплексного коэффициента передачи на заданных частотах;
- stda — стандартные отклонения значений АЧХ;
- stdph — стандартные отклонения значений ФЧХ;
- stdri — массив стандартных отклонений значений комплексного коэффициента передачи: первый столбец содержит стандартные отклонения для вещественных частей данных отклонений, второй — стандартные отклонения для мнимых частей, а третий — коэффициенты корреляции между вещественными и мнимыми частями;
- rtf — матрица коэффициентов корреляции вектора, содержащего вещественные и мнимые части значений комплексного коэффициента передачи:

  ```
  [Re(tf1);Im(tf1);Re(tf2);...];
  ```

Функция stdtfm

Функция stdtfm возвращает стандартные значения непараметрической оценки передаточной функции:

```
[tfm,stdAm,stdphm] = stdtfm(Fdat,vdat)
```

Аргументы Fdat и vdat рассматривались при описании функции rdueelis. Возвращаемые величины:

- tfm — непараметрическая оценка передаточной функции объекта;
- stdAm — стандартные отклонения оценок значений АЧХ;
- stdph — стандартные отклонения оценок значений ФЧХ.

Функции чтения/записи данных

В данную группу входят следующие 12 функций.

Функция elisfcnv

Функция elisfcnv преобразует файлы данных из одного формата в другой:

```
elisfcnv(fromfile,tofile)
```

где fromfile — имя преобразуемого файла; tofile — имя возвращаемого файла.

Используемые файлы могут быть любого из рассмотренных выше типов файлов данных с расширениями cbn, cnt, cov, fbn, fnt, fou, pbn, pnt, par, tbn, tnt, tim, vbn, vnt, var.

Функция expcov

Функция expcov осуществляет запись ковариаций в вектор ковариаций или в ковариационный файл:

```
cvect = expcov(coeffcovar,fixpind)
[cvect,Cp] = expcov(coeffcovar,fixpind,filename,comments,fdate)
```

Здесь:

- coeffcovar — матрица ковариаций оценок параметров модели размером $n \times n$, где n равняется сумме порядков числителя и знаменателя передаточной функции плюс единица (для задержки);
- fixpind — вектор, содержащий индексы зафиксированных параметров в векторе-строке;
- pdat = [num,denom,delay] (здесь num — вектор-строка коэффициентов числителя передаточной функции, denum — вектор-строка коэффициентов ее знаменателя, delay — величина запаздывания);
- filename — имя возвращаемого файла (с расширением cbn или cnt);
- comments — строка комментариев;
- fdate — строковая переменная с указанием даты и времени.

Возвращаемые величины:

- cvect — вектор, содержащий данные в формате cnt-файла;
- Cp — матрица ковариаций всех параметров модели.

Функция expfou

Функция expfou возвращает в рабочее пространство MATLAB и/или в соответствующий Фурье-файл данные эксперимента (или нескольких экспериментов) в виде вектора:

```
Fvect = expfou(freqvect,x,y)
Fvect = ...
expfou(freqvect,x,y,i,expno,filename,comments,fdate,digitnum)
```

Аргументы функции:

- freqvect — вектор частот;
- x — вектор комплексных амплитуд входного сигнала (или массив таких амплитуд при нескольких входах);
- y — вектор комплексных амплитуд выходного сигнала (или массив таких амплитуд при нескольких выходах);
- i — номера используемых экспериментов;
- expno — общее число экспериментов;
- filename — имя возвращаемого файла (с расширением fbn или fnt);
- comments — строка комментариев;
- fdate — строковая переменная с указанием даты и времени;
- digitnum — число цифр для представления данных.

Возвращаемая величина Fvect — вектор, отражающий данные эксперимента.

Функция exppar

Функция exppar возвращает параметры модели в виде вектора или файла (для использования в функции elis):

```
pvect = exppar(domain,num,denom)
pvect = ...
exppar(domain,num,denom,delay,fs,filename,comments,fdate)
```

Здесь:

- domain — строковая переменная, имеющая значение 's' для непрерывной модели и 'z' для дискретной модели;
- num — вектор коэффициентов числителя передаточной функции (упорядоченных по степеням p для непрерывной модели и z^{-1} — для дискретной);

- denom — вектор коэффициентов знаменателя передаточной функции;
- delay — величина запаздывания;
- fs — частота дискретизации для дискретных моделей или единица масштаба оси частот для непрерывных моделей;
- filename — имя возвращаемого файла (с расширением pbn или pnt);
- comments — строка комментариев;
- fdate — см. выше описание функции expfou.

Возвращаемая величина pvect — вектор параметров модели, предназначенный для использования в функции elis.

Функция exptim

Функция exptim возвращает данные, представленные во временной области в виде вектора или файла для использования в функции elis:

```
tvect = exptim(timevect,xt,yt)
tvect = ...
exptim(timevect,xt,yt,i,expno,filename,comments,fdate,digitnum)
```

Здесь:

- timevect — вектор временных отметок (значений дискретного времени);
- xt и yt — векторы (массивы для многомерного случая) значений входного и выходного сигналов объекта.

Остальные аргументы — как у двух предыдущих функций. Возвращаемая величина tvect — вектор данных (для использования в функции elis).

Функция expvar

Функция expvar возвращает набор ковариаций в форме вектора и
файла ковариаций:

```
vvect = expvar(varx,vary)
vvect = ...
expvar(varx,vary,covxy,filename,comments,fdate,Ffile)
```

Здесь:

- varx, vary и covxy — векторы, содержащие, соответственно, дисперсии вещественных частей Фурье-коэффициентов входного сигнала, выходного сигнала и коэффициенты корреляции между входом и выходом;
- filename — имя возвращаемого файла (с расширением vbn или vnt);

- comments, fdate — как у предыдущих функций;
- Ffile — соответствующий Фурье-файл для организации перекрестной проверки.

Возвращаемая величин vvect — вектор дисперсий и корреляций или файл ковариаций.

Функция impcov

Функция impcov осуществляет чтение (и ввод в рабочее пространство MATLAB) ковариационной матрицы из вектора ковариаций или из файла; является обратной по отношению к функции expcov:

```
coeffcovar = impcov(cdat)
[coeffcovar,fixpind,comments,fdate] = impcov(cdat,nofixp)
```

Здесь параметр nofixp — строковая переменная, задание которой в виде 'nofixp' приводит к исключению строк и столбцов матрицы cdat с нулевыми элементами; по умолчанию nofixp=''. Остальные входные и выходные параметры — те же, что у функции expcov.

Функция impfou

Функция impfou осуществляет чтение (и ввод в рабочее пространство MATLAB) Фурье-коэффициентов из вектора таких коэффициентов или из Фурье-файла. Является обратной по отношению к функции expfou. Записывается в виде:

```
[freqvect,x,y] = impfou(Fdat)
[freqvect,x,y,expno,comments,fdate] = impfou(Fdat,expi)
```

Аргументы и возвращаемые величины уже были описаны.

Функция imppar

Функция imppar осуществляет чтение параметров модели из вектора таких параметров или из файла, возвращаемого функцией elis. Является обратной по отношению к функции exppar.

Записывается в виде:

```
[domain,num,denom,delay,fs] = imppar(pdat)
[domain,num,denom,delay,fs,comments,fdate] = imppar(pdat,fsc)
```

Описание аргументов и возвращаемых величин было дано выше.

Функция imptim

Функция imptim осуществляет чтение данных из временного вектора или файла. Является обратной по отношению к функции exptim. Записывается в виде:

```
[timevect,xt,yt] = imptim(tdat)
[timevect,xt,yt,expno,comments,fdate] = imptim(tdat,expi)
```

Описание аргументов и возвращаемых величин было дано выше.

Функция impvar

Функция impvar осуществляет чтение (и ввод в рабочее пространство MATLAB) ковариаций из вектора ковариаций или ковариационного файла. Является обратной по отношению к функции expvar. Записывается в виде:

```
[varx,vary] = impvar(vdat)
[varx,vary,covxy,comments,fdate] = impvar(vdat)
```

Описание аргументов и возвращаемых величин было дано выше.

Функция loadasc

Функция loadasc загружает данные из текстового файла в переменные с заданными именами:

```
data = loadasc(filename)
data = loadasc(filename,ASCIItype)
```

Здесь:

- filename — имя исходного текстового файла;
- ASCIItype — строковая переменная, указывающая тип файла:
  - 'flat' — текстовый файл без комментариев;
  - 'text' (по умолчанию) или 'array' — текстовый файл с комментариями;
  - 'string' — данные представлены строковыми переменными.

Возвращаемая величина data — вектор-столбец данных.

Функции преобразования моделей

Для преобразования моделей служит ряд функций, перечисленных и описанных ниже.

Функция elis2tha

Функция elis2tha преобразует ELiS-модель в модель тета-формата, используемого в пакете System Identification:

```
theta = elis2tha(pdat,cdat,varet)
theta = elis2th(pdat,cdat,varet,numn,denomn)
```

Здесь:

- pdat, cdat — описанные ранее аргументы;
- varet — дисперсия шума наблюдений, приведенного к выходу объекта;

- numn — числитель передаточной функции формирующего фильтра шума наблюдений;
- denomn — знаменатель передаточной функции формирующего фильтра шума наблюдений.

Значения по умолчанию: numn=1, denomn=1.

Возвращаемая величина theta — модель объекта в тета-формате.

Функция tha2elis

Функция tha2elis является обратной по отношению к предыдущей:

```
[num,denom,delay,fs] = tha2elis(theta)
[num,denom,delay,fs,vary,ccovar] = tha2elis(theta,N,freqv)
```

Большинство аргументов и возвращаемых величин описано выше; аргумент N — длина массива данных (число точек дискретного спектра).

Возвращаемая величина ccovar — матрица ковариаций вектора [num, denom,delay].

Другие функции

Ряд функций различного назначения представлен ниже.

Функция digitnum

Функция digitnum возвращает количество цифр, используемых для представления чисел:

```
dignum=digitnum(x)
```

где x — некоторое число. Пример:

```
» digitnum(5.62)
ans =
     3
```

Функция fdiddemo

Функция fdiddemo осуществляет запуск демонстрационных примеров применения пакета. При выполнении функции в рабочем окне MATLAB появляется меню, предлагающее пользователю выбрать один из 13 примеров (или завершить выполнение функции):

```
You can choose from the following demonstrations:
     1) Identification example, time domain data
     2) Simulation example to show usage of TB
     3) Some problems for self-study
     4) Ident. procedure for flexible robot arm
     5) Optimal excitation signal design
     6) Crest factor minimization
     7) Time function preparation: msinprep
```

```
8) Maximum length binary sequence design
9) Discrete interval binary sequence design
10) Geometric mean of complex numbers
11) Closest point pairs on the 2D plain
12) Synchronization and averaging of data
13) Processing of measured data of the book

0) Exit from fdiddemo

Your choice (1):
```

Функция iterctrl

Функция `iterctrl` осуществляет контроль за ходом итерационного процесса при выполнении функций типа `elis`:

```
checked=iterctrl(action,parent,items,visibility)
```

Функция `iterctrl` позволяет пользователю изменять итерационный процесс выполнения соответствующей функции с использованием выпадающего меню, в частности, вводить паузы между итерациями, сохранять последний график, изменять график, завершать итерационный процесс и т. д.

Аргумент `action` — строковая переменная, определяющая выполняемое действие:

- `'initialize'` — создать и инициализировать меню (по умолчанию);
- `'checked'` — вернуть метки пунктов меню;
- `'Iteration'` — получить информацию, если пункт меню выбран;
- `'Abort'`, `'Cancel'`, `'Continue'`, `'Finish'`, `'Hold graph'`, `'Keyboard'`, `'Matlab prompt'`, `'Pause'`, `'Plot'`, `'Print'`, `'Zoom'` — обычные пункты меню MATLAB;
- `'delete'` — удалить меню.

Следующие аргументы используются, только если выбрано действие `'initialize'`:

- `parent` — дескриптор родительского объекта для создаваемого меню (по умолчанию `gcf`);
- `items` — определить список подчиненных меню (по умолчанию `'all'`);
- `visibility` = `'on'` или `'off'` — отображает или не отображает выполнение итераций, по умолчанию `'on'`.

Возвращаемая величина `checked` — строковое меню.

Функция loadvar

Функция loadvar загружает единственную переменную из mat-файла:

```
variable = loadvar(matfile,varname)
```

Здесь matfile — имя mat-файла с расширением по умолчанию mat, и varname — имя переменной в данном файле. Возвращаемая величина variable — имя переменной, которой будет присвоено значение из mat-файла.

Функция pairs

Функция pairs определяет ближайшие пары элементов в двух комплексных векторах с использованием венгерского алгоритма:

```
indab = pairs(a,b)
[indab,cycle,digits] = pairs(a,b,p,maxcycle,digitsreq,D)
```

Здесь:

- a — первый комплексный вектор;
- b — второй комплексный вектор с числом элементов не меньшим, чем в векторе a;
- p — показатель степени, используемый при вычислении расстояния между элементами (по умолчанию 2);
- maxcycle — максимальное число итерационных циклов алгоритма (по умолчанию inf);
- digitsreq — количество цифр для округленных значений чисел;
- D — матрица весов; если не задана, используется расстояние (с показателем p).

Возвращаемые величины:

- indab — номера элементов вектора b, ближайших к последовательно перечисленным элементам вектора a;
- cycle — количество выполненных итераций;
- digits — количество цифр, используемых на последнем шаге итеративной минимизации.

В процессе выполнения функции для каждого j-го элемента вектора a в соответствии с заданным расстоянием $|a_j - b_k|^p$ ищется индекс k элемента вектора b, наиболее близкого к a_j. Такая операция проводится для всех элементов вектора a с общим критерием

$$\sum_{j=1}^{n_a} |a_j - b_j|^p,$$

при этом набор полученных индексов k возвращается в виде вектора indab.

Пример:

```
» a=[1 2 3];
» b=[7 4 2 5];
» indab = pairs(a,b)
indab =
     3
     2
     4
```

Легко видеть, что в данном случае ближайшим к первому элементу вектора a (равному 1) является 3-й элемент вектора b (равный 2) и т. д.

Функция savevar

Функция savevar сохраняет указанную переменную в уже существующем mat-файле:

```
savevar(matfile,varname)
savevar(matfile,varname,varvalue)
```

Данная функция является обратной по отношению к функции loadvar. Аргументы matfile и varname — как для функции loadvar; varvalue — значение для varname. Если аргумент varvalue не используется, переменная varname будет удалена из mat-файла.

Функция yesinput

Функция yesinput предназначена для «интеллектуального» ввода данных:

```
answer = yesinput(question,default)
answer = yesinput(question,default,possib)
```

Эта функция возвращает запрос о вводе данных с подсказкой (вариантом по умолчанию). Аргументы функции: question — строка, содержащая запрос на ввод; default — значение вводимых данных по умолчанию, и possib — варианты ввода данных. Пример применения:

```
» order=yesinput('Введите порядок фильтра',10,[0,12]);
Введите порядок фильтра (10):
```

Здесь по умолчанию предлагается 10-й порядок. Требуемый выбор подтверждается нажатием клавиши перевода строки.

Функция ywalk

Функция ywalk возвращает передаточную функцию дискретного рекурсивного фильтра, наилучшим образом соответствующую заданной АЧХ:

```
[B,A] = ywalk (n,F,M)
```

где `n` — порядок фильтра; `F` и `M` — соответственно, векторы частот и значений АЧХ.

Возвращаемые величины `B` и `A` — векторы коэффициентов числителя и знаменателя дискретной передаточной функции фильтра

$$\frac{B(z)}{A(z)} = \frac{b_1 + b_2 z^{-1} + \ldots + b_n z^{-(n-1)}}{1 + a_1 z^{-1} + \ldots + a_n z^{-n}}.$$

Заинтересованный пользователь может более подробно ознакомиться с функциями и возможностями пакета при рассмотрении демонстрационных примеров пакета и используя функцию `help fdident`.

Глава 3
Пакет Control System Toolbox

Свойства пакета Control System

Пакет Control System Toolbox (или просто Control System) предназначен для моделирования, анализа и проектирования систем автоматического управления, как непрерывных, так и дискретных. Функции пакета реализуют методы исследования динамических систем, основанные на использовании передаточных функций и моделей для переменных состояния. Частотные и временные характеристики, нули и полюсы системы легко вычисляются и отображаются в виде графиков и диаграмм. В пакете реализованы:

- полный набор средств для анализа одномерных и многомерных динамических систем (объектов);
- временные характеристики: передаточная и переходная функция, реакция на произвольное воздействие;
- частотные характеристики: диаграммы Боде, Найквиста, Николса и др.;
- разработка замкнутых систем регулирования;
- проектирование регуляторов;
- характеристики моделей: управляемость, наблюдаемость, понижение порядка моделей;
- поддержка систем с запаздыванием.

Дополнительные функции позволяют конструировать более сложные модели. Временной отклик может быть рассчитан для импульсного входа, единичного скачка или произвольного входного сигнала.

Интерактивная среда для отображения временного и частотного откликов предоставляет пользователю графические управляющие элементы для одновременного отображения данных откликов и переключения между ними. Можно вычислять различные характеристики откликов, такие, например, как время регулирования.

Пакет Control System содержит средства для синтеза систем с обратными связями. Среди традиционных методов: анализ особых точек,

определение требуемого коэффициента усиления. Среди современных методов — аналитическое конструирование регуляторов и т. п.

Документация по пакету

Электронная документация по пакету Control System Design представлена в формате HTML в справочной системе. Имеется также описание в формате PDF, содержащее 649 с. Ввиду ограниченного объема книги отдельные функции пакета в данной главе описаны упрощенно — указаны их имена и назначения.

Рекомендуемая литература

Методы теории управления и правила работы с пакетом описаны, в частности, в следующих книгах:

1. Изерман Р. Цифровые системы управления. — М.: Мир, 1984.
2. Справочник по теории автоматического управления/Под ред. А. А. Красовского. — М.: Наука, 1987.
3. Топчеев Ю. И. Атлас для проектирования систем автоматического регулирования: Учебное пособие для втузов. — М.: Машиностроение, 1989.
4. Трофимов А. И., Егупов Н. Д., Дмитриев А. Н. Методы теории автоматического управления, ориентированные на применение ЭВМ. Линейные стационарные и нестационарные модели. Учебник для вузов. — М.: Энергоатомиздат, 1997.
5. Медведев В. С., Потемкин В. Г. Control System Toolbox. MATLAB 5 для студентов/Под общ. ред. к. т. н. В. Г. Потемкина. — М.: ДИАЛОГ-МИФИ, 1999.
6. Лазарев Ю. Ф. MATLAB 5.x. — К.: Издательская группа BHV, 2000.

Рассмотренные ниже функции пакета соответствуют его версии 4.2.1 (для MATLAB 5.3).

Классы вычислительных объектов пакета

Основными вычислительными объектами рассматриваемого пакета являются:

- родительский объект (класс) LTI (Linear Time-Invariant Systems — линейные, инвариантные во времени системы, в отечественной литературе обычно называемые линейными стационарными системами);

- дочерние объекты (подклассы класса LTI), соответствующие четырем видам моделей:

1. В так называемой tf-форме (в форме передаточной функции):

$$W(p) = \frac{b_1 p^m + b_2 p^{m-1} + \ldots + b_m p + b_{m+1}}{a_1 p^n + a_2 p^{n-1} + \ldots + a_n p + b_{n+1}}, \quad m \le n$$

(при описании дискретных объектов комплексная переменная p заменяется на z).

2. В zpk-форме (путем задания нулей, полюсов и коэффициента усиления):

$$W(p) = K \frac{(p - z_1)(p - z_2)\ldots(p - z_m)}{(p - p_1)(p - p_2)\ldots(p - p_n)}$$

(при описании дискретных объектов комплексная переменная p заменяется на z).

3. В ss-форме (в виде системы дифференциальных уравнений для переменных состояния):

$x'(t) = Ax(t) + Bu(t)$,

$y(t) = Cx(t) + Du(t)$

(для дискретных объектов — системы разностных уравнений).

4. В frd-форме — в виде набора (вектора) частот ω_k и соответствующих значений комплексного коэффициента передачи $W(j\omega_k)$.

В зависимости от выбора модели линейный объект (система) может быть задан либо парой многочленов (числитель и знаменатель передаточной функции), либо тройкой параметров (нули, полюса, обобщенный коэффициент передачи), либо четверкой параметров (A, B, C, D) для моделей в пространстве состояний. Для описания одномерных (в англоязычной литературе — SISO) систем используются одномерные и двумерные массивы, в случае многомерных объектов — массивы ячеек.

Пакет Control System обеспечивает создание структуры данных для модели любого из перечисленных видов в форме массива ячеек, не зависящего от конкретного представления модели. Это позволяет манипулировать линейной системой как единым объектом, а не набором данных в виде векторов или матриц.

Функции пакета

Общая характеристика

В состав пакета Control System входит более ста различных функций, объединенных в следующие группы:

- вызова графического интерфейса;
- создания моделей стационарных систем;
- извлечения данных;
- получения информации об отдельных характеристиках модели;
- преобразования моделей;
- «арифметических» операций с моделями;
- моделей динамики;
- временной задержки;
- моделей для переменных состояния;
- отклика во времени;
- частотного отклика;
- композиции систем;
- традиционного проектирования систем;
- аналитического конструирования регуляторов;
- решения матричных уравнений;
- демонстрационных программ.

Вызов графического интерфейса

В данную группу входят две функции — `ltiview` и `rltool`.

Функция ltiview

Функция `ltiview` открывает специальное средство просмотра временных и частотных характеристик модели — LTI-viewer (возможная, хотя и не совсем адекватная версия перевода — *интерактивный обозреватель*).

Запись:

```
ltiview
ltiview(plottype,sys)
ltiview(plottype,sys,extras)
ltiview(plottype,sys1,sys2,...,sysN)
ltiview(plottype,sys1,sys2,...,sysN,extras)
ltiview(plottype,sys1,PlotStyle1,sys2,PlotStyle2,...)
```

Функция `ltiview` без аргументов запускает интерактивный обозреватель, при этом на дисплее появляется рабочее окно обозревателя (рис. 3.1).

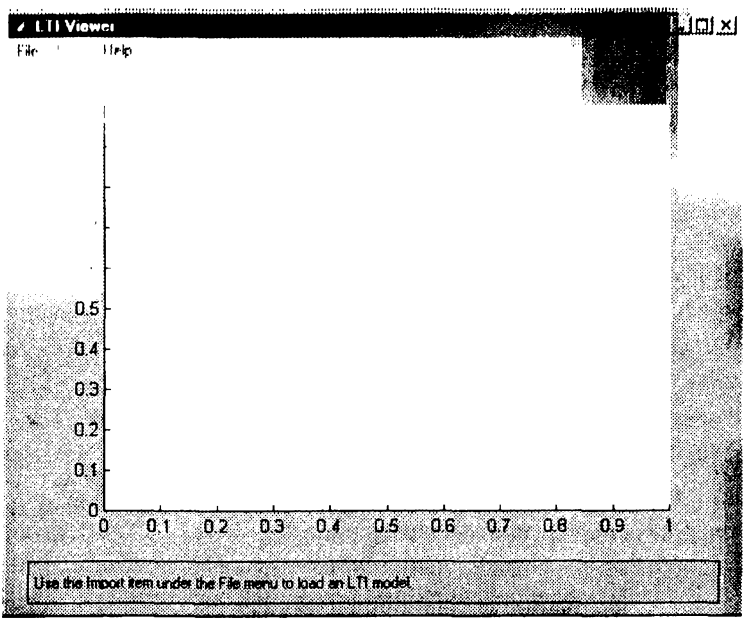


Рис. 3.1. Основное графическое окно интерактивного обозревателя

Выполнение функции в формате `ltiview(plottype,sys)` вызывает запуск обозревателя, отображающего заданные аргументом `plottype` характеристики модели с именем `sys`. Возможные значения `plottype`:

- `'step'` — отображается реакция на единичный скачок, то есть переходная функция;
- `'impulse'` — импульсная характеристика (ИХ);
- `'bode'` — диаграмма Боде, то есть ЛАЧХ и ФЧХ;
- `'nyquist'` — диаграмма Найквиста, то есть годограф;
- `'nichols'` — годограф Николса;
- `'pzmap'` — карта нулей и полюсов;
- `'sigma'` — отображается зависимость сингулярных чисел комплексного коэффициента передачи системы от частоты;

- 'initial' — отображается реакция системы на ненулевые начальные условия (модели для переменных состояния);
- 'lsim' — отображается реакция системы на произвольный входной сигнал.

Аргументы PlotStyle1 и extras обеспечивают некоторые дополнительные возможности обозревателя, sys1, sys2, ..., sysN — позволяют отображать характеристики нескольких моделей одновременно.

Интерактивный обозреватель взаимодействует с собственной рабочей средой, которая формируется независимо от рабочей среды системы MATLAB. Поэтому в случае вызова рассматриваемой функции без задания аргументов исследуемую модель необходимо поместить в рабочую среду обозревателя.

Набор LTI-моделей, используемых для демонстрационных целей в версии MATLAB 5.3, размещен в файле данных LTIexamples.mat. Выполним загрузку этого файла в рабочую среду системы MATLAB:

```
» load LTIexamples
```

Для обмена данными между рабочими средами MATLAB и LTI-Viewer используются команды меню окна обозревателя File ▸ Import (Файл ▸ Импорт) и File ▸ Export (Файл ▸ Экспорт). Выберем первую из них. Данный выбор приведет к открытию диалогового окна выбора системы (модели) для загрузки (рис. 3.2).

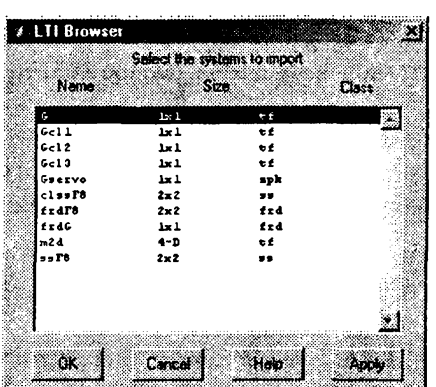


Рис. 3.2. Окно загрузки модели

В окне представлены имена моделей (столбец Name), количество их входов и выходов (столбец Size) и тип (столбец Class). Выберем, например, модель с именем G, подтвердив выбор нажатием кнопки OK.

Окно загрузки модели при этом закроется, а в окне обозревателя отобразится график переходной функции модели (рис. 3.3).

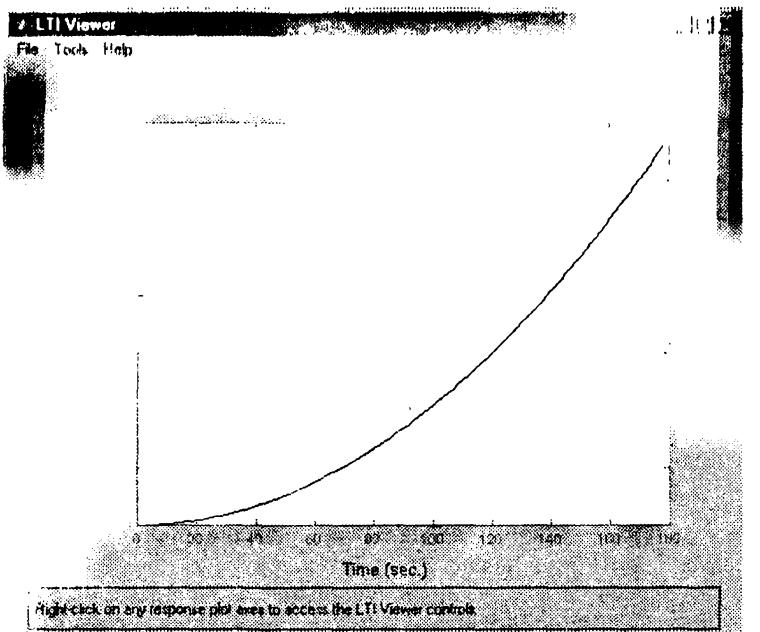


Рис. 3.3. LTI-Viewer с графиком переходного процесса

Если теперь подвести указатель мыши к произвольной точке данного графика и нажать левую кнопку, то в результате этого действия появится всплывающее сообщение об имени системы, величинах абсциссы и ординаты графика; если проделать то же, но с использованием правой кнопки мыши, получим сообщение об имени системы и номерах входа и выхода, для которых определена отображаемая характеристика.

Активизация пункта меню Tools ▸ Viewer Configuration приведет к появлению диалогового окна (рис. 3.4), позволяющего выбрать набор интересующих нас характеристик для их отображения.

Возможен выбор и вывод графиков до 6 характеристик одновременно. Выберем, например, две из них — ИХ (impulse) и ЛАЧХ-ФЧХ (Bode). В результате окно интерактивного обозревателя примет вид, показанный на рис. 3.5.

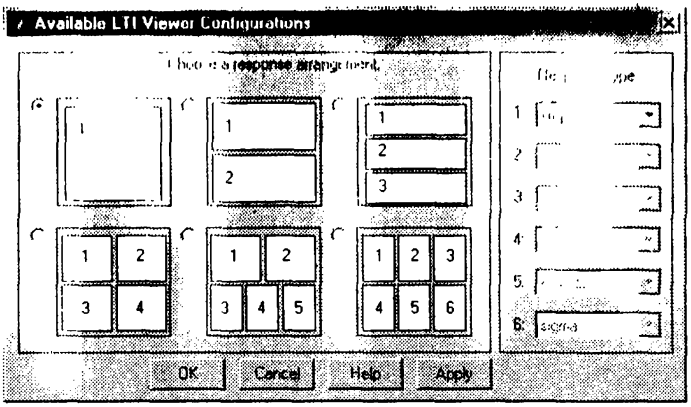


Рис. 3.4. Диалоговое окно задания видов характеристик

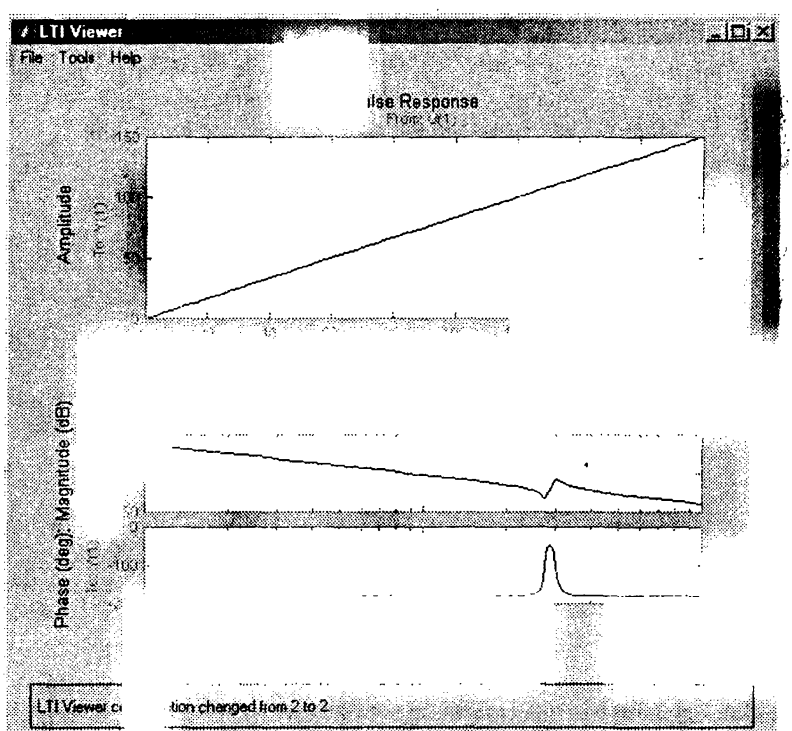


Рис. 3.5. Графики переходной и частотных характеристик

Отметим, что щелчок правой кнопкой мыши в области любого из графиков приведет к появлению контекстного меню, позволяющего менять вид характеристики (Plot Type), цвет и стиль линий (Systems), определять некоторые числовые характеристики графиков — наибольшее значение, время установления и т. п. (Characteristics), изменять масштаб (Zoom) или наносить сетку (Grid) (рис. 3.6).

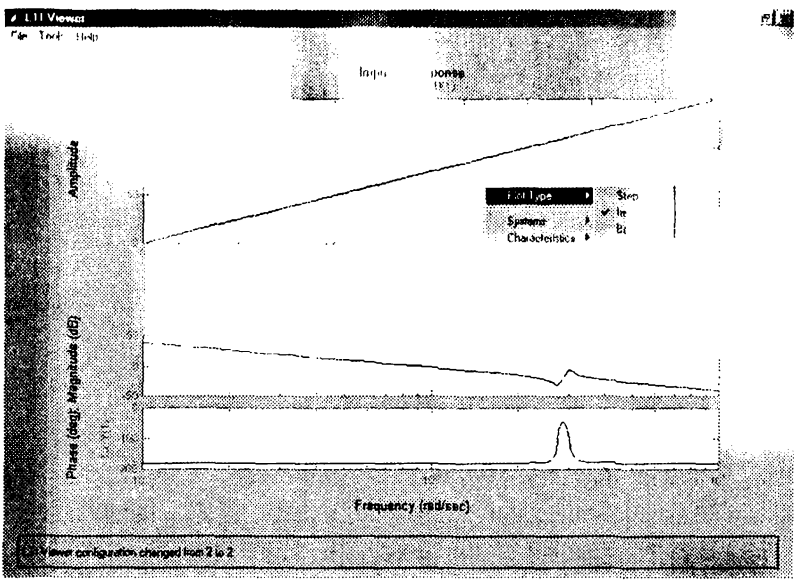


Рис. 3.6. Меню изменений параметров графика

Заканчивая рассмотрение свойств и возможностей интерактивного обозревателя, остановимся еще на нескольких пунктах его меню.

Выбор команды меню Tools ▸ Response Preferences (Сервис ▸ Свойства графиков) приводит к появлению одноименного окна (рис. 3.7). В нем можно установить диапазоны изменения координат, задать единицы измерения амплитуды и частоты, установить собственные определения времени установления и времени возрастания переходного процесса и т. д.

Выбор пункта меню Tools ▸ Linestyle Preferences (Сервис ▸ Свойства и стили линий) приводит к появлению диалогового окна, показанного на рис. 3.8, которое позволяет установить цвет, толщину и стиль линий для графического изображения. Для этого достаточно выбрать нужный объект и установить требуемый параметр.

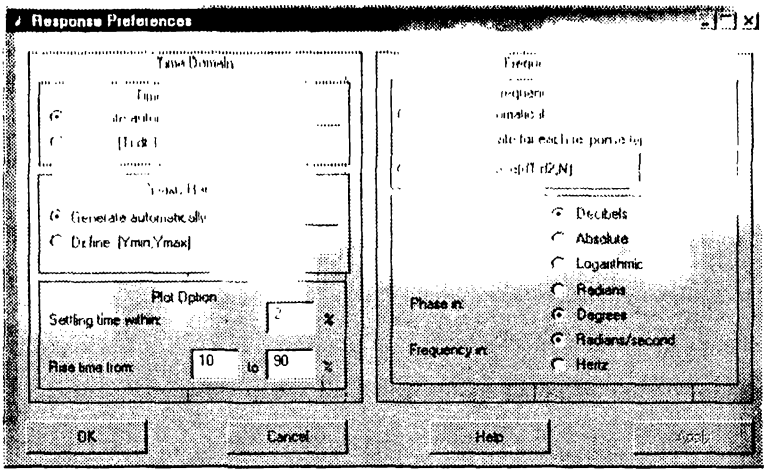


Рис. 3.7. Окно задания свойств графиков

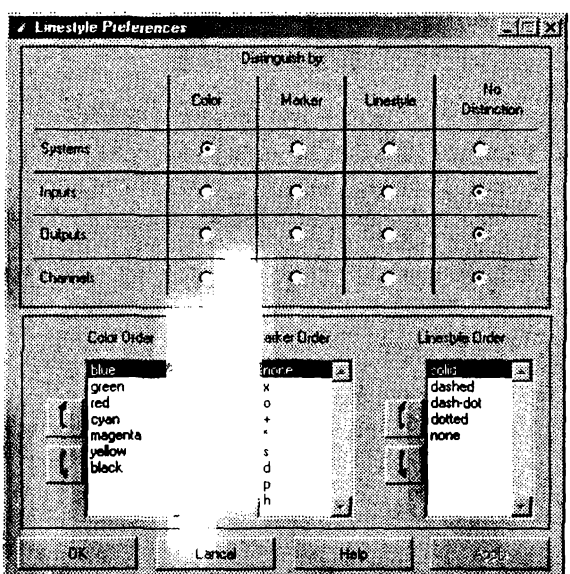


Рис. 3.8. Окно задания свойств и стилей линий

Другие пункты меню являются достаточно стандартными и не требуют особых пояснений.

Функция rltool

Функция `rltool` открывает графический интерфейс, позволяющий проектировать корректирующее звено в замкнутой одномерной системе управления методом корневого годографа (МКГ) см. рекомендуемую литературу. Функция записывается в следующих формах:

```
rltool
rltool(sys)
rltool(sys,comp)
rltool(sys,comp,LocationFlag,FeedbackSign)
```

Здесь:

- `sys` — имя модели одномерного объекта;
- `comp` — имя (идентификатор) корректирующего звена-компенсатора;
- `LocationFlag` — переменная, задающая позицию компенсатора в системе: 1 — в прямом тракте системы, 2 — в цепи обратной связи;
- `FeedbackSign` — тип обратной связи (−1 — отрицательная обратная связь, 1 — положительная обратная связь).

Выполнение рассматриваемой функции без аргументов приводит к появлению основного окна интерфейса (рис. 3.9).

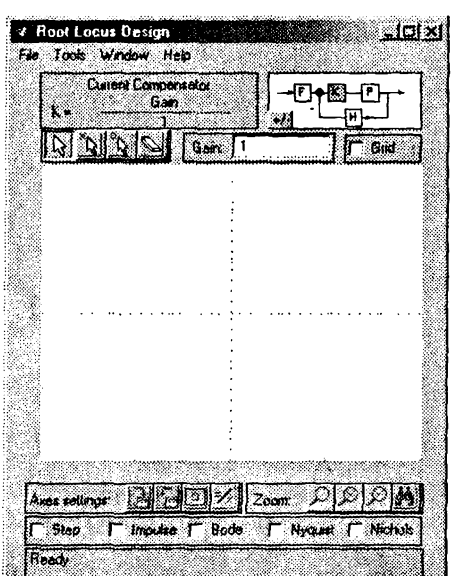


Рис. 3.9. Окно интерфейса, реализующего метод корневого годографа

Работу с интерфейсом лучше всего рассмотреть на примере. Предварительно в режиме командной строки выполним загрузку файла с набором LTI-моделей:

```
load LTIexamples
```

Затем в меню интерфейса выберем команду File ▸ Import Model (Файл ▸ Импорт модели). Выполнение данной команды меню приведет к открытию диалогового окна загрузки моделей и задания структуры системы (рис. 3.10).

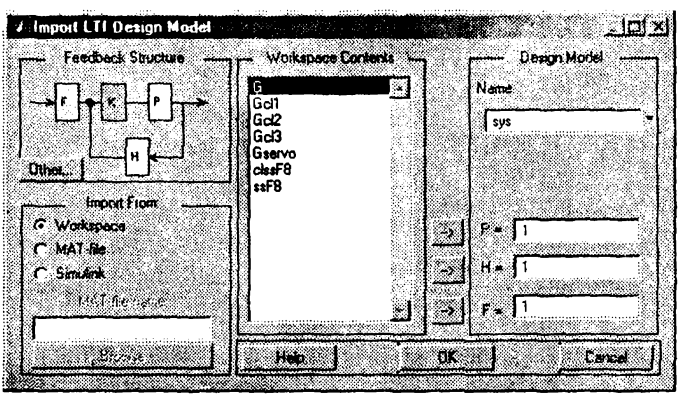


Рис. 3.10. Окно загрузки модели и задания структуры системы

Обратим внимание, что в верхнем левом углу окна приведена структурная схема замкнутой системы. В ней *F*, *H*, *P* — звенья, модели которых можно загружать (с помощью переключателя Import From) из рабочей среды MATLAB, из mat-файла или из среды Simulink. Звено же, обозначенное буквой *K*, — это как раз то компенсирующее динамическое звено, структуру и параметры которого нужно определить. В исходной схеме данное звено включено в прямой тракт замкнутой системы. Нажатием кнопки Other (Другая) можно перейти к альтернативной структуре, в которой компенсатор введен в цепь обратной связи.

Сохранив исходную структуру, выберем модели для загрузки. В качестве модели звена *P* примем модель с именем G. Данный выбор проведем, выбрав в списке модель G и нажав кнопку со стрелкой, указывающей на звено *P* (в правой нижней части окна). Модели звеньев *F* и *H* изменять не будем (по умолчанию это пропорциональные звенья с единичным коэффициентом усиления). Зададим теперь какое-либо имя создаваемой системе, например mysys (в поле Name), под-

твердим указанные действия нажатием кнопки OK. Окно загрузки модели при этом закроется, а основное окно интерфейса приобретет вид, показанный на рис. 3.11.

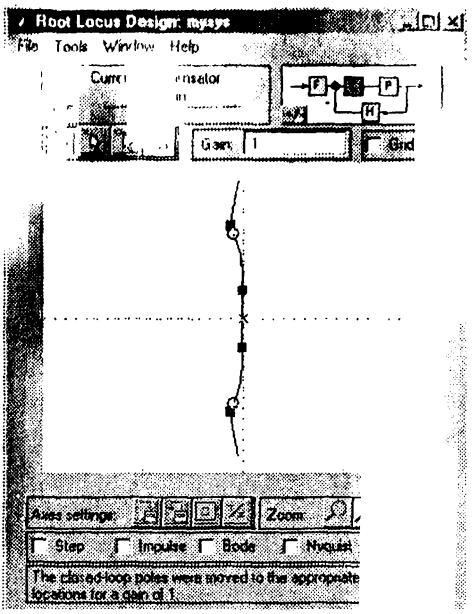


Рис. 3.11. Окно интерфейса МКГ с загруженной моделью объекта

В графической части окна (на комплексной плоскости нулей и полюсов) отображены полюсы замкнутой системы (квадратиками), ее нули (кружочками) и линии их перемещения при изменении коэффициента усиления компенсатора (Gain) от указанного (заданного) значения до ∞. Как представляется, система с таким расположением полюсов не будет обладать хорошим качеством переходного процесса — они расположены слишком близко к границе устойчивости (к правой половине комплексной плоскости). В этом легко убедиться, просмотрев график переходного процесса замкнутой системы путем установки флажка Step в нижней части интерфейса — данный выбор приведет к открытию окна интерактивного обозревателя (рис. 3.12).

Действительно, как видно из графика переходного процесса, исходная система находится на границе устойчивости с временем затухания переходного процесса около $3{,}5 \cdot 10^4$ с. Заметим, что с корректирующим звеном пропорционального типа сколь-нибудь существенно

улучшить динамические свойства системы не удастся (кстати, при заданной структуре компенсатора нули и полюсы можно передвигать непосредственно на графике при помощи мыши; при этом соответственно изменяется значение коэффициента усиления компенсатора в поле Gain). Поэтому попробуем построить компенсатор инерционно-форсирующего типа, добавив один полюс и один ноль с помощью кнопок с обозначением крестика и кружочка, расположенных над графическим окном слева и разместив при этом (с помощью мыши) нули и полюсы, например, так, как это показано на рис. 3.13.

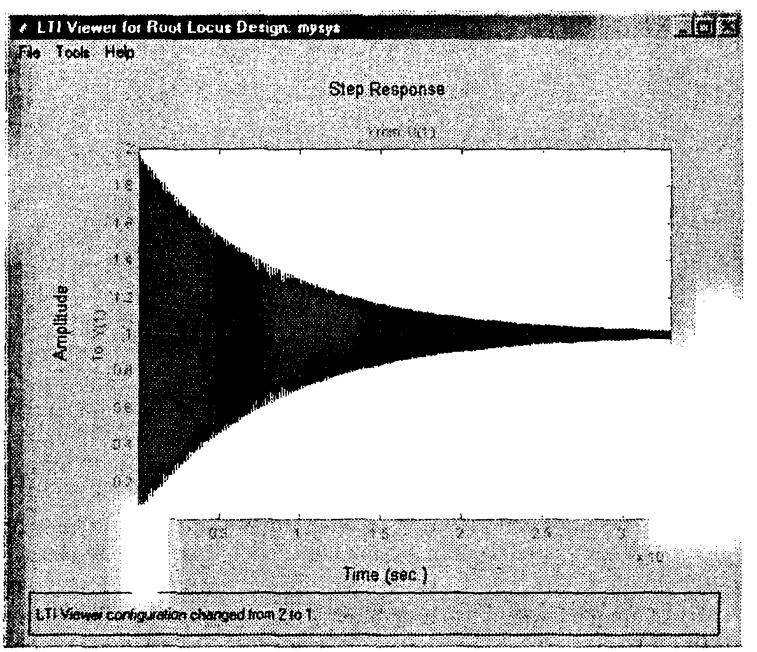


Рис. 3.12. Переходный процесс в нескорректированной системе

Заметим, что график переходного процесса (рис. 3.14) при такой коррекции выглядит существенно лучше.

Если такой вид переходного процесса нас устраивает, на этом процедуру синтеза можно и закончить. Заметим, что передаточная функция компенсатора в данном случае имеет вид

$$K(p) = 1 \cdot \frac{p + 0.22}{p + 0.68}.$$

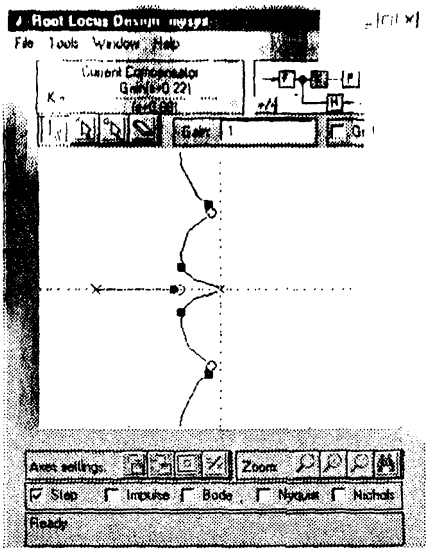


Рис. 3.13. Нули и полюсы в скорректированной системе

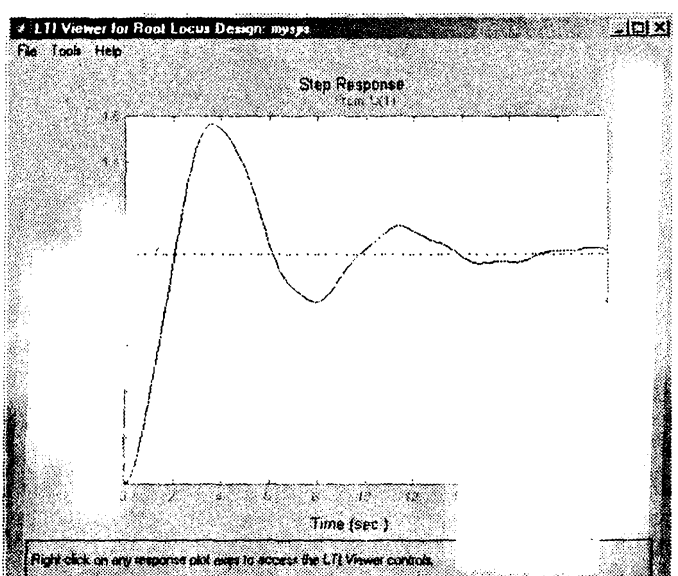


Рис. 3.14. График переходного процесса в скорректированной системе

Заметим далее, что интерфейс МКГ обладает многочисленными сервисными функциями. Щелкнем, например, дважды на блоке *P* структурной схемы. При этом раскроется окно с информацией о моделях блоков *F*, *H*, *P* (рис. 3.15).

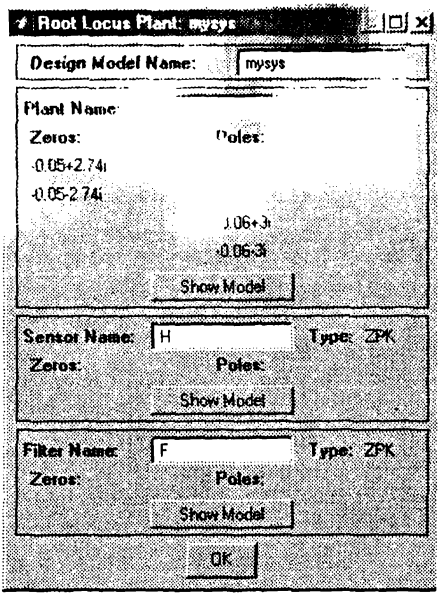


Рис. 3.15. Окно с информацией о моделях блоков F, H, P

Более детальную информацию о передаточных функциях можно получить при нажатии кнопок Show Model (Показать модель). Так, информация о передаточной функции звена *P* имеет вид, показанный на рис. 3.16.

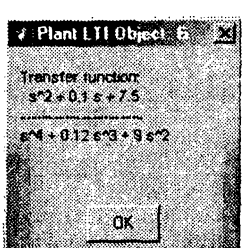


Рис. 3.16. Информация о передаточной функции звена P

Довольно много возможностей предоставляет пользователю меню интерфейса. Например, используя команду меню File ▸ Draw Simulink Diagram (Файл ▸ Изобразить диаграмму Simulink), непосредственно перейдем к моделированию функционирования системы в среде Simulink. Пример подобного моделирования показан на рис. 3.17 и 3.18.

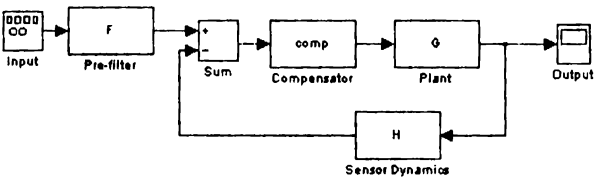


Рис. 3.17. Модель исследуемой системы в среде Simulink

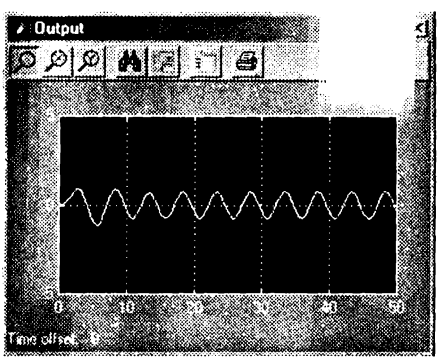


Рис. 3.18. Реакция скорректированной системы на гармонический сигнал с единичной амплитудой и частотой 0,2 Гц

Возложив приятную задачу изучения других возможностей интерфейса МКГ на заинтересованного пользователя, отметим в заключение, что выполнение функции `rltool` с заданием аргументов приведет к открытию интерфейса с соответствующими загруженными моделями.

Создание моделей стационарных систем

В данную группу входят следующие 9 функций.

Функция tf

Функция `tf` создает модель в виде передаточной функции:

```
sys = tf(num,den)
sys = tf(num,den,Ts)
```

```
sys = tf(M)
sys = tf(num,den,ltisys)
sys = tf(num,den,'Property1',Value1,...,'PropertyN',ValueN)
sys = tf(num,den,Ts,'Property1',Value1,...,'PropertyN',ValueN)
sys = tf('s')
sys = tf('z',Ts)
tfsys = tf(sys)
tfsys = tf(sys,'inv') %Только для ss-моделей
```

Здесь:

- num и den — векторы-строки (массивы для многомерных моделей) коэффициентов полиномов числителя и знаменателя, расположенных в порядке убывания степеней p и z;
- Ts — интервал дискретизации (при создании дискретных моделей);
- M — скаляр или матрица; при использовании функции tf с данным аргументом создается модель пропорционального звена с коэффициентом пропорциональности M (соответственно, скалярная модель или матрица таких моделей);
- ltisys или sys — имя (идентификатор) LTI-модели;
- 'Property1', Value1, ..., 'PropertyN', ValueN — набор пар «свойство/значение»;
- 's' — определяет задание непрерывной передаточной функции;
- 'z' — определяет задание дискретной передаточной функции;
- 'inv' — определяет быстрый алгоритм преобразования модели для переменных состояния в передаточную функцию.

Возвращаемая величина (sys или tfsys) — передаточная функция:

- при записи sys = tf(num,den) создается непрерывная передаточная функция num(p)/den(p);
- при записи sys = tf(num,den,ltisys) создается дискретная передаточная функция при интервале дискретизации Ts;
- при записи sys = tf(M) формируется матрица коэффициентов передачи;
- при записи sys = tf(num,den,ltisys) формируется LTI-модель со свойствами, наследуемыми от LTI-модели ltisys.

Информацию о свойствах модели можно получить, используя команду (функцию) ltiprops.

Приведем примеры применения функции tf.

Пример 1. Сформируем одномерную непрерывную передаточную функцию

$$W(p)=\frac{p}{p^2+2p+10}$$

с именем w. В данном случае num = [1 0], den = [1 2 10], и формирование $W(p)$ задается следующей записью:

```
  w=tf([1 0],[1 2 10])
 Transfer function:
      s
--------------
s^2 + 2 s + 10
```

Пример 2. Создадим матрицу передаточных функций непрерывного объекта с одним входом («Ток») и двумя выходами («Момент» и «Угловая скорость») вида

$$W(p)=\begin{bmatrix}\dfrac{p+1}{p^2+2p+2}\\[2ex]\dfrac{1}{p}\end{bmatrix}.$$

```
» W=tf(num,den,'inputn','Ток','outputn',{'Момент' 'Угловая
      скорость'},'variable','p')
 Transfer function from input "Ток" to output...
                   p + 1
 Момент:     -------------
             p^2 + 2 p + 2

                    1
 Угловая скорость:  -
                    p
```

Пример 3. Сформируем модель матричной дискретной передаточной функции

$$W(z)=\begin{bmatrix}\dfrac{1}{z+0.3} & \dfrac{z}{z+0.3}\\[2ex]\dfrac{-z+2}{z+0.3} & \dfrac{3}{z+0.3}\end{bmatrix}$$

при интервале дискретизации Ts = 0,2 с:

```
» nums={1 [1 0];[-1 2] 3};
» Ts=0.2;
» W=tf(nums,[1 0.3],Ts)
 Transfer function from input 1 to output...
```

```
          1
#1:  -------
     z + 0.3

     -z + 2
#2:  -------
     z + 0.3
Transfer function from input 2 to output...
          z
#1:  -------
     z + 0.3

          3
#2:  -------
     z + 0.3
Sampling time: 0.2
```

Функция zpk

Функция zpk создает модель в виде передаточной функции по ее з данным нулям и полюсам:

```
sys = zpk(z,p,k)
sys = zpk(z,p,k,Ts)
sys = zpk(M)
sys = zpk(z,p,k,ltisys)
sys = zpk(z,p,k,'Property1',Value1,...,'PropertyN',ValueN)
sys = zpk(z,p,k,Ts,'Property1',Value1,...,'PropertyN',ValueN)
sys = zpk('s')
sys = zpk('z')
zsys = zpk(sys)
zsys = zpk(sys,'inv') % Только для ss-моделей
```

Здесь z и p — соответственно, векторы-строки (массивы для мног мерных объектов) полюсов и нулей передаточной функции (матриц передаточных функций), а k — обобщенный коэффициент усилени Остальные аргументы — как у предыдущей функции. Возвращаемы величины — модели объектов (систем) в zpk-форме.

Функция zpk(sys) преобразует произвольную LTI-модель в модел вида zpk. Так, команда формирует модель объекта с одним входом двумя выходами при заданных нулях и полюсах, приведенных в ар гументах функции:

```
» W=zpk({[];[2 3]},{1;[0 -1]},[-5;1])
Zero/pole/gain from input to output...
           -5
#1:    -----
       (s-1)
```

```
        (s-2) (s-3)
#2:     -----------
         s (s+1)
```

Функция ss

Функция ss формирует модель в пространстве состояний:

```
sys = ss(a,b,c,d)
sys = ss(a,b,c,d,Ts)
sys = ss(d)
sys = ss(a,b,c,d,ltisys)
sys = ss(a,b,c,d,'Property1',Value1,....,'PropertyN',ValueN)
sys = ss(a,b,c,d,Ts,'Property1',Value1,....,'PropertyN',ValueN)
sys_ss = ss(sys)
sys_ss = ss(sys,'minimal')
```

Здесь a, b, c, и d — матрицы модели для переменных состояния, а 'minimal' — аргумент, задание которого формирует модель, для которой все переменные состояния являются управляемыми и наблюдаемыми. Остальные аргументы — как у двух предыдущих функций. Возвращаемая величина — модель в ss-форме. Функция ss(sys) преобразует произвольную модель, заданную в tf- или zpk-форме, в ss-модель.

Функция dss

Функция dss создает модель в неявной форме Коши:

$$E\dot{x} = Ax + Bu,$$
$$y = Cx + Du$$

для непрерывного объекта и

$$Ex(n+1) = Ax(n) + Bu(n),$$
$$y(n) = Cx(n) + Du(n)$$

для дискретного с заданным интервалом дискретизации Ts. Эта функция записывается в виде:

```
sys = dss(a,b,c,d,e)
sys = dss(a,b,c,d,e,Ts)
sys = dss(a,b,c,d,e,ltisys)
sys = dss(a,b,c,d,e,'Property1',Value1,....,'PropertyN',ValueN)
sys = dss(a,b,c,d,e,Ts,'Property1',Value1,....,'PropertyN',ValueN)
```

Аргумент e задает матрицу E модели. Остальные аргументы — как у рассмотренных выше функций. Возвращаемая величина — ss-модель приведенного вида.

Для иллюстрации применения этой функции создадим модель рассматриваемого подкласса для объекта, описываемого соотношениями

$$5\dot{x} = x + 2u,$$
$$y = 3x + 4u$$

при наличии временной задержки в 0,1 с, с названием входа «Напряжение» и указанием в примечании, что модель является примером.

Данная задача решается следующим образом:

```
» sys=dss(1,2,3,4,5,'td',0.1,'inputname','Напряжение','notes','Пример')
a =
                    x1
          x1        1

b =
              Напряжение
          x1        2

c =
                    x1
          y1        3

d =
              Напряжение
          y1        4

e =
                    x1
          x1        5

Input delays (listed by channel): 0.1
Continuous-time model.
```

Функция frd

Функция frd создает частотную модель объекта в frd-форме:

```
sys = frd(response,frequency)
sys = frd(response,frequency,Ts)
sys = frd
sys = frd(response,frequency,ltisys)
sysfrd = frd(sys,frequency)
sysfrd = frd(sys,frequency,'Units',units)
```

Здесь frequency — вектор-строка частот ω_k, response — вектор-строка (массив в многомерном случае) значений комплексного коэффициента передачи и units — строковая переменная, задающая размерность частоты (по умолчанию 'rad/s', другое возможное значение 'Hz').

Остальные аргументы — как у рассмотренных выше команд данной группы. Возвращаемая величина — frd-модель. При записи

```
sysfrd = frd(sys,frequency,'Units',units)
```

модель sys (tf-, zpk- или ss-вида) преобразуется в frd-модель. Пример:

```
» freq = logspace(1,2); %Генерация вектора частот с 50-ю элементами
» resp = .05*(freq).*exp(i*2*freq); %Значения комплексного коэффициента
передачи
» sys = frd(resp,freq); %Формирование frd-модели
```

Функция filt

Функция `filt` создает tf-модель в формате цифрового фильтра, то есть в виде дискретной передаточной функции, числитель и знаменатель которой являются полиномами от z^{-1}:

```
sys = filt(num,den)
sys = filt(num,den,Ts)
sys = filt(M)
sys = filt(num,den,'Property1',Value1,...,'PropertyN',ValueN)
sys = filt(num,den,Ts,'Property1',Value1,...,'PropertyN',ValueN)
```

Аргументы этой функции соответствуют описанным выше, за тем исключением, что коэффициенты полиномов числителя num и знаменателя den должны быть упорядочены в порядке возрастания степеней z^{-1}.

Рассмотрим пример создания цифрового фильтра с двумя входами и одним выходом (интервал дискретизации не задан):

```
» num = {1 , [1 0.3]};
» den = {[1 1 2] ,[5 2]};
» W = filt(num,den,'inputname',{'Канал_1' 'Канал_2'})

Transfer function from input "Канал_1" to output:
       1
-----------------
1 + z^-1 + 2 z^-2

Transfer function from input "Канал_2" to output:
1 + 0.3 z^-1
------------
 5 + 2 z^-1

Sampling time: unspecified
```

Функция set

Функция set задает свойства LTI-моделей:

```
set(sys,'Property',Value)
set(sys,'Property1',Value1,'Property2',Value2,...)
set(sys,'Property')
set(sys)
```

Рассматриваемая функция предназначена для того, чтобы LTI-модели с именем sys присвоить свойство 'Property' со значением Value.

Функция set(sys,'Property1',Value1,'Property2',Value2,...) позволяет присвоить значения нескольким свойствам с помощью одной команды.

Функция set(sys,'Property') выводит список допустимых значений для свойства 'Property'.

Функция set(sys) выводит для модели sys полный список свойств и их допустимых значений.

В качестве примера создадим следующую одномерную непрерывную ss-модель:

```
» sys = ss(1,2,3,4)
 a =
                    x1
          x1         1

 b =
                    u1
          x1         2

 c =
                    x1
          y1         3

 d =
                    u1
          y1         4

 Continuous-time model.
```

Дополним эту модель следующими свойствами:

- запаздывание td по входу, равное 0,1 с;
- имя входной переменной «Момент»;
- заменим значение d на нулевое;
- сохраним значение коэффициента передачи модели, используя свойство 'Userdata'.

Перечисленные свойства устанавливаются функцией

```
» set(sys,'inputd',0.1,'inputn','Момент','d',0,'user',dcgain(sys))
```

Проверим установки, используя функцию get (описание данной функции см. ниже):

```
» get(sys)
                    a: 1
                    b: 2
                    c: 3
                    d: 0
                    e: []
            StateName: {''}
                   Ts: 0
           InputDelay: 0.1
          OutputDelay: 0
        ioDelayMatrix: 0
            InputName: {'Момент'}
           OutputName: {''}
           InputGroup: {0x2 cell}
          OutputGroup: {0x2 cell}
                Notes: {}
             UserData: -2
```

Функция ltimodels

Функция `ltimodels` возвращает общую информацию о различных типах LTI-моделей:

```
ltimodels
ltimodels(modeltype)
```

Здесь аргумент `modeltype` — строковая переменная со значениями `'tf'`, `'zpk'` или `'ss'`.

При записи `ltimodels(modeltype)` возвращается дополнительная информация о модели указанного типа.

Функция ltiprops

Функция `ltiprops` возвращает информацию о свойствах LTI-моделей:

```
ltiprops
ltiprops(modeltype)
```

Использование аналогично предыдущей функции.

Извлечение данных

Данную группу образуют следующие 6 функций.

Функция tfdata

Функция `tfdata` возвращает числитель и знаменатель передаточной функции (ПФ).

Запись:

```
[num,den] = tfdata(sys)
[num,den] = tfdata(sys,'v')
[num,den,Ts] = tfdata(sys)
```

Здесь аргумент 'v' используется для возвращения результата в виде векторов-строк, а не в виде описания размерностей массива ячеек. Остальные аргументы аналогичны рассмотренным для других функций.

Функция zpkdata

Функция zpkdata возвращает нули, полюсы и обобщенные коэффициенты передачи для каждого канала LTI-модели с именем sys:

```
[z,p,k] = zpkdata(sys)
[z,p,k] = zpkdata(sys,'v')
[z,p,k,Ts,Td] = zpkdata(sys)
```

Функция ssdata

Функция ssdata возвращает матрицы (и величину интервала дискретизации в дискретном случае) ss-модели:

```
[a,b,c,d] = ssdata(sys)
[a,b,c,d,Ts] = ssdata(sys)
```

Функция dssdata

Функция dssdata возвращает информацию о матрицах (и о величине интервала дискретизации) ss-модели в неявной форме Коши (см. описание функции dss):

```
[a,b,c,d,e] = dssdata(sys)
[a,b,c,d,e,Ts] = dssdata(sys)
```

Функция frdata

Функция frdata возвращает информацию о комплексном коэффициенте передачи объекта. Является обратной по отношению к функции frd. Записывается в виде:

```
[response,freq] = frdata(sys)
[response,freq,Ts] = frdata(sys)
[response,freq] = frdata(sys,'v')
```

Смысл всех аргументов и возвращаемых величин пояснен выше.

Функция get

Функция get возвращает информацию о текущих свойствах модели:

```
Value = get(sys,'PropertyName')
get(sys)
```

Получение информации об отдельных характеристиках модели

Данную группу образуют следующие функции, которые приведены здесь для справки и представлены лишь своими именами:

- `class` — возвращает информацию о типе модели (tf, zpk, ss или frd);
- `isa` — осуществляет проверку, является ли LTI-модель моделью заданного типа;
- `size` — возвращает информацию о размерах матриц модели;
- `ndims` — возвращает информацию о размере вектора переменных состояния модели `x`;
- `isempty` — осуществляет проверку, является ли LTI-модель пустой;
- `isct` — осуществляет проверку, является ли LTI-модель непрерывной;
- `isdt` — осуществляет проверку, является ли LTI-модель дискретной;
- `isproper` — осуществляет проверку, является ли LTI-модель правильной;
- `issiso` — осуществляет проверку, является ли LTI-модель одномерной;
- `reshape` — изменяет размеры матрицы (массива).

Преобразование моделей

К функциям преобразования моделей относятся следующие функции, для краткости описания указанные только именами:

- `tf` — преобразует LTI-модель в tf-форму;
- `zpk` — преобразует LTI-модель в zpk-форму;
- `ss` — преобразует LTI-модель в ss-форму;
- `frd` — преобразует LTI-модель в frd-форму;
- `chgunits` — изменяет размерность частоты в frd-модели;
- `c2d` — преобразует непрерывную модель в дискретную;
- `d2c` — преобразует дискретную модель в непрерывную;
- `d2d` — изменяет интервал дискретизации в дискретной модели.

«Арифметические» операции с моделями

Функции этой группы позволяют создавать и изменять структуры динамических систем. Внешне они подобны арифметическим операторам, но операндами в данном случае являются LTI-модели.

- + и - — сложение и вычитание LTI-моделей (параллельное соединение);
- * — умножение LTI-моделей (последовательное соединение);
- \ — левое деление (sys1\sys2 равносильно inv(sys1)*sys2);
- / — правое деление (sys1/sys2 равносильно sys1*inv(sys2));
- ^ — возведение LTI-модели в степень (последовательное соединение нескольких одинаковых LTI-моделей);
- ' — операция *pertransposition*, означающая замену матрицы системы $\mathbf{A}(p)$ на матрицу $[\mathbf{A}(-p)]^{\mathrm{T}}$; для дискретных моделей — замена матрицы $\mathbf{A}(z)$ на матрицу $[\mathbf{A}(z^{-1})]^{\mathrm{T}}$;
- .' — транспонирование модели;
- [..] — горизонтальное/вертикальное объединение LTI-моделей;
- inv — обращение LTI-модели (замена входов на выходы и наоборот).

Рассмотрим примеры использования данных функций.

Пример 1. Создание одномерной модели:

```
» w=tf([1 0],[1 2 10]) % Создание модели
Transfer function:
          s
  -----------------
s^2 + 2 s + 10

» w' % Pertransposition
Transfer function:
         -s
------------------
s^2 - 2 s + 10

» inv(w) %Обращение

Transfer function:
s^2 + 2 s + 10
------------------
          s
» w^2 %Возведение в степень

Transfer function:
                        s^2
-------------------------------------------
s^4 + 4 s^3 + 24 s^2 + 40 s + 100
```

Пример 2. Создание модели дискретного объекта с двумя входами и двумя выходами:

```
» nums={1 [1 0];[-1 2] 3};
» Ts=0.2;
» W=tf(nums,[1 0.3],Ts)

Transfer function from input 1 to output...
          1
 #1:  ---------
       z + 0.3

         -z + 2
 #2:  ---------
       z + 0.3

Transfer function from input 2 to output...
          z
 #1:  ---------
       z + 0.3

          3
 #2:  ---------
       z + 0.3

Sampling time: 0.2

» W' %Pertransposition

Transfer function from input 1 to output...
            z
 #1:  ------------
       0.3 z + 1

            1
 #2:  ------------
       0.3 z + 1

Transfer function from input 2 to output...
         2 z - 1
 #1:  ------------
       0.3 z + 1

           3 z
 #2:  ------------
       0.3 z + 1

Sampling time: 0.2
```

```
» inv(W) %Обращение

Transfer function from input 1 to output...
          3 z + 0.9
 #1:  ----------------
        z^2 - 2 z + 3

         z^2 - 1.7 z - 0.6
 #2:  --------------------
          z^2 - 2 z + 3

 Transfer function from input 2 to output...
        -z^2 - 0.3 z + 6.328e-016
 #1:  -------------------------------
                 z^2 - 2 z + 3

           z + 0.3
 #2:  ----------------
        z^2 - 2 z + 3

Sampling time: 0.2
```

Модели динамики

В данную группу входят следующие 9 функций.

Функция pole

Функция pole возвращает вектор, элементами которого являются полюсы LTI-модели:

```
p = pole(sys)
```

Пример:

```
» pole(w)
ans =
  -1.0000 + 3.0000i
  -1.0000 - 3.0000i
```

Функция zero

Функция zero возвращает вектор, элементами которого являются нули LTI-модели:

```
z = zero(sys)
[z,gain] = zero(sys)
```

Функция, вызванная в форме `[z,gain] = zero(sys)`, кроме нулей возвращает еще и обобщенный коэффициент усиления.

Функция pzmap

Функция pzmap возвращает карту нулей и полюсов:

```
pzmap(sys)
[p,z] = pzmap(sys)
```

В последней форме записи график не выводится, а возвращаются векторы, элементы которых являются нулями и полюсами модели.

Функция dcgain

Функция dcgain возвращает статический коэффициент передачи (матрицу таких коэффициентов в многомерном случае) LTI-модели:

```
k = dcgain(sys)
```

Рассмотрим пример. Пусть дискретная передаточная функция имеет вид

$$W(z)=\begin{bmatrix} 1 & \dfrac{p-1}{p^2+p+3} \\ \dfrac{1}{p+1} & \dfrac{p+2}{p+3} \end{bmatrix}.$$

Создадим ее tf-модель, а затем найдем матрицу статических коэффициентов усиления:

```
» % Создание tf-модели
» W=[1 tf([1 -1],[1 1 3]);tf(1,[1 1]) tf([1 2],[1 -3])]

Transfer function from input 1 to output...
 #1:  1

           1
 #2:  ------
      s + 1

Transfer function from input 2 to output...
            s - 1
 #1:  --------------
      s^2 + s + 3

      s + 2
 #2:  -------
      s - 3

» % Нахождение матрицы коэффициентов усиления
» dcgain(W)
ans =
    1.0000   -0.3333
    1.0000   -0.6667
```

Используя свойства преобразования Лапласа, нетрудно убедиться в правильности ответа.

Функция norm

Функция norm предназначена для вычисления норм типа $\| \|_2$ или $\| \|_\infty$ для непрерывной или дискретной LTI-модели (норма равна бесконечности для неустойчивых систем):

```
norm(sys)
norm(sys,2)
norm(sys,inf)
norm(sys,inf,tol)
[ninf,fpeak] = norm(sys)
```

Отметим, что норма $\|W(p)\|_2$ устойчивой непрерывной системы с передаточной матрицей $W(p)$ — это квадратный корень из среднего значения квадрата импульсной характеристики системы, а при переходе к преобразованию Лапласа данная норма, в соответствии с теоремой Парсеваля, определяется соотношением

$$\|W(p)\|_2 = \sqrt{\frac{1}{2\pi}\int_{-\infty}^{\infty} \mathrm{tr}\{W^*(i\omega)W(i\omega)\}d\omega},$$

где tr{•} — обозначение следа матрицы, $W^*(j\omega)$ — матрица, сопряженная по отношению к $W(j\omega)$.

Норма $\| \|_\infty$ равна максимальному значению модуля частотной характеристики:

- для одномерных моделей

  $$\|W(p)\|_\infty = \max_\omega |W(j\omega)|;$$

- для многомерных моделей

  $$\|W(p)\|_\infty = \max_\omega \sigma_{\max} |W(j\omega)|,$$

  где $\sigma_{\max}$ — максимальное сингулярное число матрицы $W(j\omega)$, то есть неотрицательный корень квадратный из максимального собственного числа матрицы $W^*(j\omega)W(j\omega)$;

- для дискретной модели

  $$\|W(z)\|_\infty = \max_{\theta\in[0,\pi\theta]} \sigma_{\max} |W(e^{j\theta})|.$$

Аргументы функции norm следующие:

- sys — имя модели;
- 2 — задание нормы $\| \|_2$;

- `inf` — задание нормы $\| \|_\infty$;
- `tol` — точность при расчете нормы, по умолчанию `tol = 1e-2`.

Возвращаемые величины:

- `ninf` — норма модели (по умолчанию — $\| \|_2$),
- `fpeak` — частота, на которой норма достигает максимального значения.

Функция covar

Функция `covar` возвращает матрицу ковариаций сигнала на выходе устойчивой линейной модели с постоянными параметрами при действии на входах возмущений типа белого шума:

```
P = covar(sys,W)
[P,Q] = covar(sys,W)
```

Здесь `sys` — имя модели и `W` — матрица интенсивностей входного сигнала. Возвращаемые величины: `P` — матрица ковариаций выходов и `Q` — матрица ковариаций переменных состояния.

Функция damp

Функция `damp` возвращает собственные частоты и коэффициенты демпфирования LTI-модели:

```
[Wn,Z] = damp(sys)
[Wn,Z,P] = damp(sys)
```

Функция `damp` предназначена для расчета собственных чисел и коэффициентов демпфирования, соответствующих полюсам LTI-модели `sys`. Если функция вызывается без выходных аргументов, то на дисплей выводится таблица собственных значений `P`, соответствующих им собственных частот `Wn` и коэффициентов демпфирования `Z`.

Функция esort

Функция `esort` сортирует полюсы непрерывной LTI-модели в порядке убывания значений их действительной части:

```
s = esort(p)
[s,ndx] = esort(p)
```

Неустойчивые полюсы располагаются в начале списка. В этой функции `p` — вектор-строка полюсов модели; `s` — вектор, содержащий отсортированные полюсы; `ndx` — вектор, содержащий индексы, использованные при сортировке.

Функция dsort

Функция dsort сортирует полюсы дискретной LTI-модели в порядке убывания их модулей. Неустойчивые полюсы располагаются в начале списка:

```
s = dsort(p)
[s,ndx] = dsort(p)
```

Описание и использование — как для предыдущей функции.

Временная задержка

В группу функций временной задержки входят следующие функции, которые представлены их именами:

- hasdelay — возвращает информацию о том, имеется ли в системе временная задержка;
- totaldelay — возвращает матрицу суммарных задержек от входов к выходам модели;
- delay2z — вводит задержку в k интервалов дискретизации в дискретной системе путем умножения ее передаточной функции на z^{-k};
- pade — осуществляет Паде-аппроксимацию звена запаздывания.

Рассмотрим примеры применения этих функций.

Пример 1. Рассмотрим применение функции delay2z:

```
» % Создание дискретной модели
» z=tf('z',-1);
» sys=(-.4*z -.1)/(z^2+1.05*z+.08)
 Transfer function:
-0.4 z - 0.1
-------------------
z^2 + 1.05 z + 0.08

Sampling time: unspecified
» %Задание задержки в один интервал дискретизации
» sys.Inputd = 1
 Transfer function:
           -0.4 z - 0.1
z^(-1) * -------------------
         z^2 + 1.05 z + 0.08

Sampling time: unspecified
» %Использование функции delay2z
» sys=delay2z(sys)
 Transfer function:
```

```
     -0.4 z - 0.1
---------------------
z^3 + 1.05 z^2 + 0.08 z

Sampling time: unspecified
```

Пример 2. Использование функции pade для создания динамического звена с передаточной функцией в виде отношения двух полиномов, аппроксимирующего звена временного запаздывания. Величина задержки 1 с, порядок аппроксимирующего звена — 2.

```
» [num,den] = pade(1,2)
num =
     1    -6    12

den =

     1     6    12
```

Отметим, что полученная аппроксимация соответствует известной формуле

$$e^{-p} \approx \frac{p^2 - 6p + 12}{p^2 + 6p + 12}.$$

Модели для переменных состояния

Теперь рассмотрим (см. ниже) функции моделей для переменных состояния.

Функции rmodel и rss

Функции rmodel и rss формируют устойчивые непрерывные модели со случайными параметрами:

```
sys = rss(n)
sys = rss(n,p)
sys = rss(n,p,m)
sys = rss(n,p,m,s1,....,sn)
[num,den] = rmodel(n)
[A,B,C,D] = rmodel(n)
[A,B,C,D] = rmodel(n,p,m)
```

Здесь n, p, m — соответственно порядок модели (число переменных состояния), число выходов и число входов. По умолчанию — один вход и один выход.

Функция sys = rss(n,p,m,s1,...,sn) формирует массив ss-моделей со случайными параметрами размером s1×s2×...×sn.

Функция [num,den] = rmodel(n) возвращает числитель и знаменатель передаточной функции устойчивой непрерывной модели n-го порядка.

Функции drmodel и drss

Функции drmodel и drss формируют устойчивые дискретные модели со случайными параметрами (интервал дискретизации не задается):

```
sys = drss(n)
sys = drss(n,p)
sys = drss(n,p,m)
sys = drss(n,p,m,s1,...sn)

[num,den] = drmodel(n)
[A,B,C,D] = drmodel(n)
[A,B,C,D] = drmodel(n,p,m)
```

Описание аналогично описанию предыдущих функций.

Функция ss2ss

Функция ss2ss осуществляет преобразование ss-модели при переходе к новому базису:

```
sysT = ss2ss(sys,T)
```

Для заданной ss-модели sys рассматриваемая функция выполняет преобразование вектора состояния $\psi = \mathbf{T}x$ и создает эквивалентную модель sysT, описываемую системой уравнений:

$$\dot{\psi} = \mathbf{TAT}^{-1}\psi + \mathbf{TB}u,$$

$$y = \mathbf{CT}^{-1}\psi + \mathbf{D}u.$$

Функция sysT = ss2ss(sys,T) возвращает преобразованную модель sysT, используя исходную модель sys и матрицу преобразования T (данная матрица должна быть невырожденной). Функция применима как к непрерывным, так и к дискретным моделям.

Функция canon

Функция canon возвращает так называемую каноническую форму ss-модели:

```
csys = canon(sys,'type')
[csys,T] = canon(sys,'type')
```

Функцией поддерживаются две канонические формы: модальная и присоединенная.

В модальной канонической форме действительные собственные значения матрицы A расположены на ее главной диагонали, а паре комплексно-сопряженных собственных значений соответствует блок

размером 2×2, также расположенный на диагонали этой матрицы. Для системы с собственными значениями (λ_1, $\sigma \pm j\omega$, λ_2) модальная матрица A имеет вид

$$\begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \sigma & \omega & 0 \\ 0 & -\omega & \sigma & 0 \\ 0 & 0 & 0 & \lambda_2 \end{bmatrix}.$$

В присоединенной канонической форме данная матрица имеет вид

$$\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & -a_n \\ 1 & 0 & 0 & \cdots & 0 & -a_{n-1} \\ 0 & 1 & 0 & \cdots & 0 & -a_{n-2} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 0 & -a_2 \\ 0 & 0 & 0 & \cdots & 1 & -a_1 \end{bmatrix},$$

где a_1–a_n — коэффициенты характеристического многочлена системы

$$\gamma(p) = p^n + a_1 p^{n-1} + \ldots a_{n-1} p + a_n.$$

Аргументы этой функции следующие:

- sys — имя исходной ss-модели;
- 'type' — строковая переменная, задающая тип канонической формы ('modal' — модальная форма, 'companion' — присоединенная).

Возвращаемые величины: csys — преобразованная модель, T — матрица преобразования, связывающая вектор состояния в канонической форме с вектором состояния исходной модели.

Функция ctrb

Функция ctrb формирует матрицу управляемости для модели в пространстве состояний:

```
Co = ctrb(A,B)
Co = ctrb(sys)
```

Здесь:

- sys — имя ss-модели;
- A и B — матрицы этой модели.

Система является управляемой, если матрица управляемости имеет полный ранг. Возвращаемая величина Co — матрица управляемости.

имеющая n строк и $n \times m$ столбцов (m — количество входов) и описываемая соотношением

$$\mathbf{Co} = [\mathbf{B}\ \mathbf{AB}\ \mathbf{A}^2\mathbf{B}\ \ldots\ \mathbf{A}^{n-1}\mathbf{B}].$$

Для примера проверим, является ли управляемой система 2-го порядка с матрицами

$$A = \begin{bmatrix} 1 & 1 \\ 4 & -2 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}.$$

Проверка осуществляется с использованием следующих функций:

```
» A=[1 1;4 -2];
» B=[1 -1;1 -1];
» Co=ctrb(A,B) %Вычисление матрицы управляемости
Co =

     1    -1     2    -2
     1    -1     2    -2
» rank(Co) %Вычисление ранга матрицы
ans =

     1
```

В данном случае ранг матрицы управляемости равен 1, а порядок системы — 2. Следовательно, система не является полностью управляемой.

Функция ctrbf

Функция ctrbf формирует так называемую каноническую форму управляемости:

```
[Abar,Bbar,Cbar,T,k] = ctrbf(A,B,C)
[Abar,Bbar,Cbar,T,k] = ctrbf(A,B,C,tol)
```

Здесь аргумент tol позволяет задать точность вычислений. Если матрица управляемости для пары {**A**, **B**} имеет ранг $r < n$, где n — порядок матрицы **A**, то существует преобразование подобия вида

$$\overline{\mathbf{A}} = \mathbf{TAT}^T, \quad \overline{\mathbf{B}} = \mathbf{TB}, \quad \overline{\mathbf{C}} = \mathbf{CT}^T,$$

где **T** — унитарная матрица, при котором преобразованная система имеет блочно-треугольную форму с неуправляемыми модами, расположенными в верхнем левом углу:

$$\overline{\mathbf{A}} = \begin{bmatrix} \mathbf{A}_{uc} & 0 \\ \mathbf{A}_{21} & \mathbf{A}_c \end{bmatrix}, \quad \overline{\mathbf{B}} = \begin{bmatrix} 0 \\ \mathbf{B}_c \end{bmatrix}, \quad \overline{\mathbf{C}} = [\mathbf{C}_{uc} \quad \mathbf{C}].$$

Такая форма называется канонической формой управляемости.

Пара матриц $\{A_c, B_c\}$ является управляемой, так что справедливо соотношение

$$C_c(pI - A_c)^{-1}B_c \quad C(pI - A)^{-1}B,$$

где I — единичная матрица, то есть передаточная функция всей системы совпадает с передаточной функцией ее управляемой части, а все моды, соответствующие собственным значениям матрицы A_{uc}, являются неуправляемыми.

Функция `[Abar,Bbar,Cbar,T,k] = ctrbf(A,B,C)` преобразует ss-модель, описываемую тройкой матриц `[A, B, C]`, в каноническую форму управляемости `[Abar, Bbar, Cbar]`. Матрица `T` описывает преобразование подобия, а элементы вектора `k` указывают количество управляемых мод, выделенных на каждом шаге расчета матрицы преобразования. Число ненулевых элементов вектора `k` показывает, сколько итераций потребовалось для расчета матрицы `T`, а величина `sum(k)` указывает число канонических переменных состояния, соответствующих управляемой части матрицы `Abar`.

Функция obsv

Функция `obsv` формирует матрицу наблюдаемости для модели в пространстве состояний:

```
Ob = obsv(A,C)
Ob = obsv(sys)
```

Здесь

- `sys` — имя ss-модели;
- `A` и `C` — матрицы этой модели.

Система является наблюдаемой, если матрица наблюдаемости имеет полный ранг. Возвращаемая величина `Ob` — матрица наблюдаемости, имеющая p строк (p — количество выходов) и n столбцов и описываемая соотношением

$$Ob = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}.$$

Функция obsvf

Функция `obsvf` формирует так называемую каноническую форму наблюдаемости:

```
[Abar,Bbar,Cbar,T,k] = obsvf(A,B,C)
[Abar,Bbar,Cbar,T,k] = obsvf(A,B,C,tol)
```

Если матрица наблюдаемости для пары {**A**, **C**} имеет ранг $r < n$, где n — порядок матрицы **A**, то существует преобразование подобия вида

$$\overline{\mathbf{A}} = \mathbf{TAT}^{\mathrm{T}}, \quad \overline{\mathbf{B}} = \mathbf{TB}, \quad \overline{\mathbf{C}} = \mathbf{CT}^{\mathrm{T}},$$

где **T** — унитарная матрица, при котором преобразованная система имеет блочно-треугольную форму с ненаблюдаемыми модами, расположенными в верхнем левом углу:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{n0} & \mathbf{A}_{12} \\ 0 & \mathbf{A}_0 \end{bmatrix}, \quad \overline{\mathbf{B}} = \begin{bmatrix} \mathbf{B}_{n0} \\ \mathbf{B}_0 \end{bmatrix}, \quad \overline{\mathbf{C}} = \begin{bmatrix} 0 & \mathbf{C}_0 \end{bmatrix}.$$

Такая форма называется канонической формой наблюдаемости.

Пара матриц $\{\mathbf{A}_o, \mathbf{C}_o\}$ является наблюдаемой, так что справедливо соотношение

$$\mathbf{C}_o(p\mathbf{I} - \mathbf{A}_o)^{-1}\mathbf{B}_o = \mathbf{C}(p\mathbf{I} - \mathbf{A})^{-1}\mathbf{B},$$

то есть передаточная функция всей системы совпадает с передаточной функцией ее наблюдаемой части, а все моды, соответствующие собственным значениям матрицы $\mathbf{A}_{no}$, являются ненаблюдаемыми.

Функция `[Abar,Bbar,Cbar,T,k] = obsvf(A,B,C)` преобразует ss-модель, описываемую тройкой матриц `[A, B, C]`, в каноническую форму наблюдаемости `[Abar, Bbar, Cbar]`. Матрица `T` описывает преобразование подобия, а элементы вектора `k` указывают количество наблюдаемых мод, выделенных на каждом шаге расчета матрицы преобразования. Число ненулевых элементов вектора `k` показывает, сколько итераций потребовалось для расчета матрицы `T`, а величина `sum(k)` указывает число канонических переменных состояния, соответствующих наблюдаемой части матрицы `Abar`.

Функция gram

Функция `gram` позволяет вычислить функции Грама для оценки управляемости и наблюдаемости системы, называемые, соответственно, грамианами управляемости и наблюдаемости. Грамианы применяются для исследования свойств управляемости и наблюдаемости моделей систем, заданных в пространстве состояний, а также для построения их минимальных реализаций. Они более удобны для вычислений, чем матрицы управляемости и наблюдаемости (матрица **A** модели должна быть устойчивой).

Для непрерывной ss-модели грамиан управляемости определяется интегралом

$$\mathbf{G}_C = \int_0^\infty e^{\mathbf{A}\tau}\mathbf{B}\mathbf{B}^{\mathrm{T}}e^{\mathbf{A}^{\mathrm{T}}\tau}d\tau,$$

а грамиан наблюдаемости — интегралом

$$\mathbf{G}_0 = \int_0^\infty e^{\mathbf{A}^{\mathrm{T}}\tau}\mathbf{C}^{\mathrm{T}}\mathbf{C}e^{\mathbf{A}\tau}d\tau.$$

Для дискретных моделей аналогами грамианов управляемости и наблюдаемости служат выражения:

$$\mathbf{G}_C = \sum_{k=0}^{\infty}\mathbf{A}^k\mathbf{B}\mathbf{B}^{\mathrm{T}}(\mathbf{A}^{\mathrm{T}})^k, \quad \mathbf{G}_0 = \sum_{k=0}^{\infty}(\mathbf{A}^{\mathrm{T}})^k\mathbf{C}^{\mathrm{T}}\mathbf{C}\mathbf{A}^k.$$

Грамиан управляемости положительно определен тогда и только тогда, когда пара матриц {**A**, **B**} является управляемой. Аналогично грамиан наблюдаемости положительно определен тогда и только тогда, когда пара матриц {**A**, **C**} является наблюдаемой.

Функция `Gc = gram(sys,'c')` вычисляет грамиан управляемости для непрерывной или дискретной ss-модели.

Функция `Go = gram(sys,'o')` вычисляет грамиан наблюдаемости для непрерывной или дискретной ss-модели.

Функция ssbal

Функция `ssbal` выполняет масштабирование ss-моделей:

```
[sysb,T] = ssbal(sys)
[sysb,T] = ssbal(sys,condT)
```

Рассматриваемая функция выполняет масштабирование матриц ss-модели, используя преобразование подобия с диагональной матрицей **T** и скаляром α такими, что матрица

$$\begin{bmatrix} \mathbf{T}\mathbf{A}\mathbf{T}^{-1} & \mathbf{T}\mathbf{B}/\alpha \\ \alpha\mathbf{C}\mathbf{T}^{-1} & 0 \end{bmatrix}$$

имеет малые числа обусловленности по отношению к задаче на собственные значения.

Функция `[sysb,T] = ssbal(sys)` возвращает масштабированную модель `sysb`, описываемую четверкой $\{\mathbf{T}\mathbf{A}\mathbf{T}^{-1}, \mathbf{T}\mathbf{B}/\alpha, \alpha\mathbf{C}\mathbf{T}^{-1}, \mathbf{D}\}$, и матрицу преобразования **T**, такую что $\psi = \mathbf{T}\mathbf{x}$, где ψ — новый вектор состояния модели.

Функция `[sysb,T] = ssbal(sys,condT)` задает верхнюю границу числа обусловленности `condT` для матрицы `T`. Поскольку масштабирование при плохо обусловленной матрице `T` может приводить к росту оши-

бок округления, задание величины condT дает возможность контроля данных ошибок.

Функция balreal

Функция balreal возвращает сбалансированную реализацию ss-модели, которая имеет равные грамианы управляемости и наблюдаемости:

```
sysb = balreal(sys)
[sysb,g,T,Ti] = balreal(sys)
```

Здесь аргумент sys — имя исходной (несбалансированной) модели. Возвращаемые величины:

- sysb — сбалансированная модель;
- g — вектор, содержащий диагональные элементы сбалансированного грамиана;
- T и Ti — матрица преобразования подобия и обратная ей матрица.

Функция modred

Функция modred используется (обычно вместе с функцией balreal) для понижения порядка модели:

```
rsys = modred(sys,elim)
rsys = modred(sys,elim,'mdc')
rsys = modred(sys,elim,'del')
```

Здесь:

- sys — имя исходной модели;
- elim — вектор, элементы которого являются номерами подлежащих удалению переменных состояния;
- 'mdc' — задает метод понижения размерности, заключающийся в приравнивании производных удаляемых переменных состояния нулю и решении системы уравнений для определения их установившихся значений;
- 'del' — задает метод понижения размерности, при котором просто удаляются переменные состояния, определяемые вектором elim. Этот способ не гарантирует сохранения значения коэффициента передачи, но более точно аппроксимирует переходные процессы в модели.

Возвращаемая величина rsys — модель, имеющая порядок на length(elim) меньше, чем исходная.

Рассмотрим пример. Пусть исходная модель имеет передаточную

$$W(p) = \frac{p^3 + 11p^2 + 36p + 26}{p^4 + 14.6p^3 + 74.96p^2 + 1537p + 9965}.$$

Понизим ее порядок, построив вначале с помощью функции **balreal** ее сбалансированную реализацию:

```
» W = tf([1 11 36 26],[1 14.6 74.96 153.7 99.65]);
» [Wb,g] = balreal(W);
» g'

ans =

    0.1394    0.0095    0.0006    0.0000
```

Три последних диагональных элемента грамиана здесь имеют малые значения по сравнению с первым элементом, поэтому порядок модели можно понизить до первого, удалив соответствующие переменные состояния. С этой целью используем функцию **modred** в двух модификациях: в первом случае гарантируем сохранение коэффициента передачи, во втором — просто удалим переменные состояния:

```
» Wmdc=modred(Wb,2:4,'mdc');
» Wdel=modred(Wb,2:4,'del');
```

Сравним теперь частотные характеристики редуцированных моделей с частотными характеристиками исходной модели (результат показан на рис. 3.19):

```
» bode(W,'-',Wmdc,'x',Wdel,'*')
```

Как следует из рис. 3.19, частотные характеристики редуцированной модели **Wdel** более точно, чем характеристики модели **Wmdc**, воспроизводят частотную характеристику исходной системы.

Проведем аналогичное сравнение для переходных процессов (результаты показаны на рис. 3.20):

```
» step(W,'-',Wmdc,'-.',Wdel,'--')
```

Из рисунка видно, что редуцированная модель **Wdel** точнее аппроксимирует переходный процесс, а модель **Wmdc** — установившееся значение.

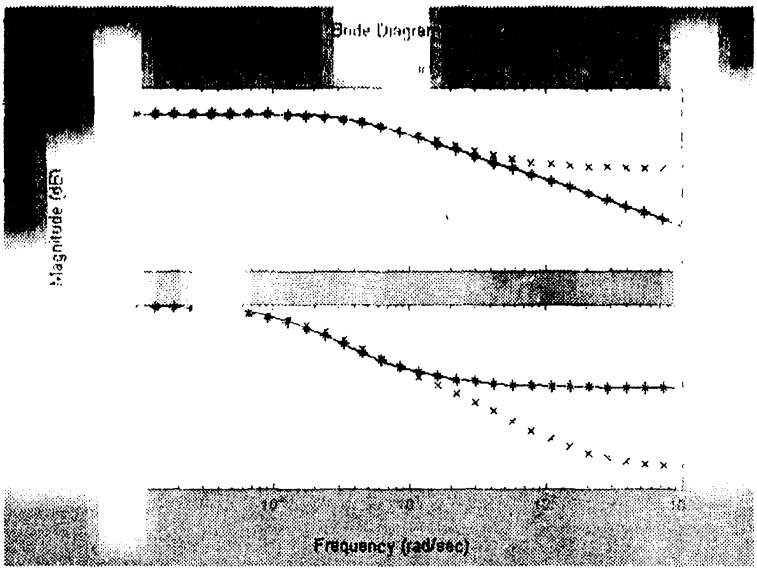


Рис. 3.19. Частотные характеристики исходной и редуцированных моделей

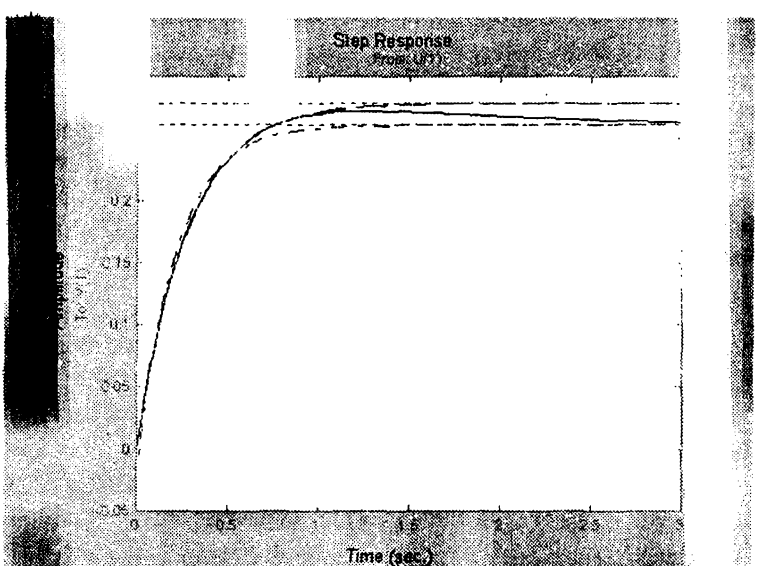


Рис. 3.20. Переходные характеристики исходной и редуцированных моделей

Функция minreal

Функция minreal строит минимальную реализацию для некоторой модели. При этом удаляются неуправляемые или ненаблюдаемые переменные состояния и выполняется сокращение совпадающих нулей и полюсов. Результирующая модель имеет минимальный порядок и те же самые частотные характеристики, что и первоначальная модель системы. Функция записывается в виде:

```
sysr = minreal(sys)
sysr = minreal(sys,tol)
```

Смысл аргументов и возвращаемой величины пояснен выше.

Функция sminreal

Функция sminreal выполняет понижение порядка системы за счет удаления переменных состояния, не оказывающих влияния на отклик системы, вызываемый входным сигналом:

```
msys = sminreal(sys)
```

Отклик во времени

В группу функций для расчета и построения отклика систем во времени входят следующие 6 функций.

Функция step

Функция step рассчитывает и строит реакцию модели на единичную ступенчатую функцию, то есть возвращает переходную функцию системы:

```
step(sys)
step(sys,t)
step(sys1,sys2,....,sysN)
step(sys1,sys2,....,sysN,t)
step(sys1,'PlotStyle1',....,sysN,'PlotStyleN')
[y,t,x] = step(sys)
```

Для моделей, заданных в пространстве состояний, начальные условия принимаются нулевыми. Аргументы функции следующие:

- sys, sys1, sys2, ..., sysN — имена моделей, для которых строятся переходные функции (все модели должны иметь одинаковое число входов и выходов);
- t — аргумент, задающий момент окончания моделирования — либо в форме t = Tfinal (в секундах), либо в форме t = 0:dt:Tfinal. Для дискретных моделей значение dt должно равняться интервалу дискретизации, для непрерывных моделей — быть достаточно малым, чтобы учесть наиболее быстрые изменения переходного процесса;

- 'PlotStyle1', 'PlotStyle2', ..., 'PlotStyleN' — строковые переменные, задающие стили графиков (типы линий) при выводе нескольких графиков одновременно.

Возвращаемые величины:

- график (графики) переходных процессов;
- y, x, t — соответственно, векторы, содержащие значения переходного процесса, переменных состояния и моментов времени; при возвращении данных величин график переходного процесса не отображается.

В качестве примера применения этой функции построим графики переходных процессов для следующей модели 2-го порядка:

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}-0.5572 & -0.7814\\ 0.7814 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}1 & -1\\ 0 & 2\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix},$$

$$y = [1.9291 \quad 6.4493]\begin{bmatrix}x_1\\ x_2\end{bmatrix}.$$

```
» a=[-0.5572 -0.7814;0.7814 0]; % Формирование модели
» b=[1 -1;0 2];
» c=[1.9691 6.4493];
» sys=ss(a,b,c,0);
» step(sys) % Построение переходных процессов
```

Графики данных процессов приведены на рис. 3.21.

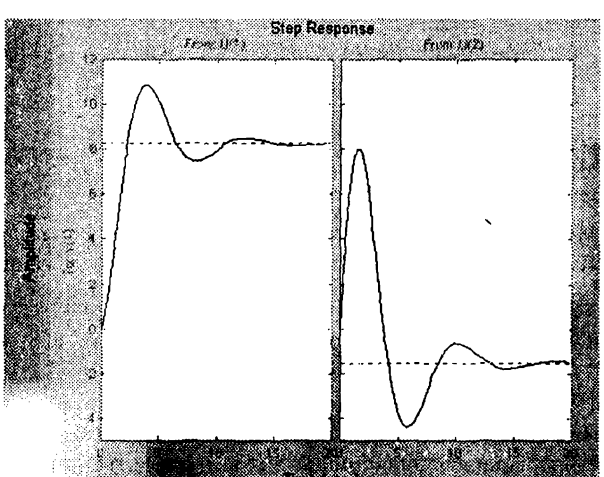


Рис. 3.21. Графики переходных процессов в системе 2-го порядка

Функция impulse

Функция `impulse` возвращает импульсную характеристику модели и нескольких моделей:

```
impulse(sys)
impulse(sys,t)
impulse(sys1,sys2,....,sysN)
impulse(sys1,sys2,....,sysN,t)
impulse(sys1,'PlotStyle1',....,sysN,'PlotStyleN')
[y,t,x] = impulse(sys)
```

Смысл аргументов и возвращаемых величин — как у предыдущ функции, за тем исключением, что здесь все относится к ИХ.

Функция initial

Функция `initial` возвращает графики переходных процессов для модели при ненулевых начальных условиях:

```
initial(sys,x0)
initial(sys,x0,t)
initial(sys1,sys2,....,sysN,x0)
initial(sys1,sys2,....,sysN,x0,t)
initial(sys1,'PlotStyle1',....,sysN,'PlotStyleN',x0)
[y,t,x] = initial(sys,x0)
```

Здесь `x0` — вектор-строка начальных условий для переменных состояния. Смысл остальных аргументов и возвращаемых величин — как у функции `step`.

Функция lsim

Функция `lsim` возвращает реакцию LTI-модели на произвольное входное воздействие:

```
lsim(sys,u,t)
lsim(sys,u,t,x0)
lsim(sys1,sys2,....,sysN,u,t)
lsim(sys1,sys2,....,sysN,u,t,x0)
lsim(sys1,'PlotStyle1',....,sysN,'PlotStyleN',u,t)
[y,t,x] = lsim(sys,u,t,x0)
```

Здесь `t` — вектор-строка моментов времени, `u` — вектор-строка значений входного сигнала, остальные аргументы и возвращаемые величины — как у предыдущих функций данной группы.

Функция gensig

Функция `gensig` генерирует скалярный сигнал заданного типа и с заданным периодом:

```
[u,t] = gensig(type,tau)
[u,t] = gensig(type,tau,Tf,Ts)
```

Здесь:

- type — строковая переменная, задающая тип генерируемого сигнала и имеющая следующие возможные значения:
  - 'sin' — синусоида,
  - 'square' — периодический прямоугольный сигнал,
  - 'pulse' — периодические импульсы.

  Все генерируемые сигналы имеют единичную амплитуду;
- tau — период генерируемого сигнала (в секундах);
- Tf — продолжительность сигнала, с;
- Ts — период дискретности (в секундах) для генератора импульсов.

Возвращаемые величины:

- t — вектор значений времени;
- u — соответствующий вектор значений генерируемого сигнала.

Рассмотрим следующий пример, в котором рассматриваемая функция используется для генерации периодического прямоугольного сигнала с периодом 5 с, продолжительностью 30 с и периодом дискретизации 0,1 с:

```
[u,t]=gensig('square',5,30,0.1); %Формирование сигнала
```

Отобразим теперь график сигнала (рис. 3.22):

```
plot(t,u)
axis([0 30 -1 2])
```

Затем для объекта, описываемого матрицей передаточных функций

$$W(p)=\begin{bmatrix}\dfrac{2p^2+5p+1}{p^2+2p+1}\\ \dfrac{p-1}{p^2+p+5}\end{bmatrix},$$

найдем реакцию на сформированный сигнал:

```
%Задание модели объекта
W=[tf([2 5 1],[1 2 3]);tf([1 -1],[1 1 5])];
%Определение реакции
lsim(W,u,t)
```

Результат показан на рис. 3.23.

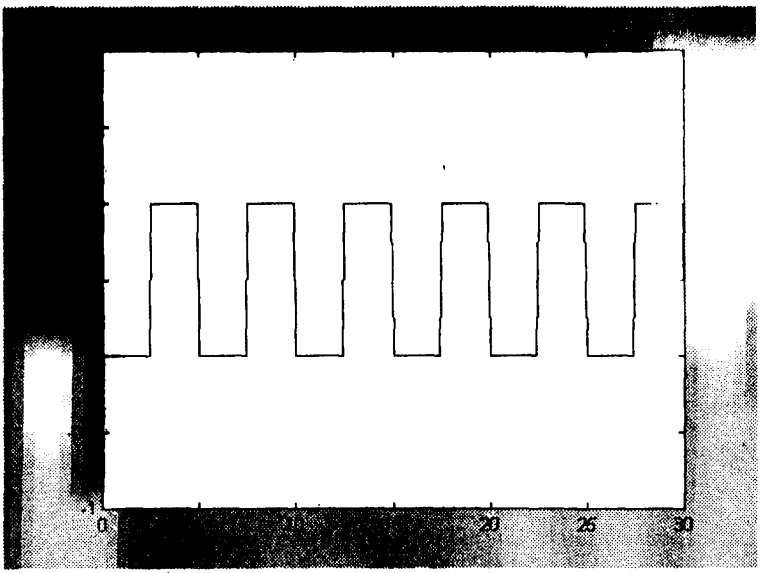


Рис. 3.22. Пример использования функции gensig

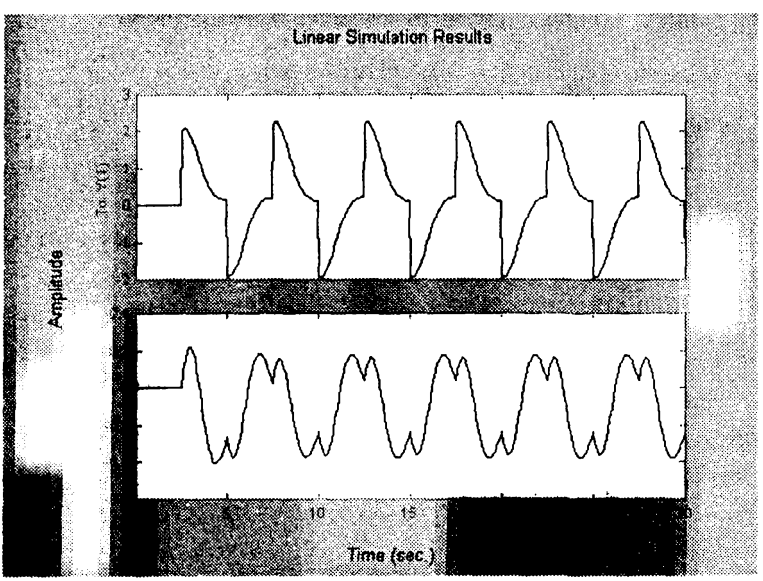


Рис. 3.23. Реакции двумерного объекта на периодический прямоугольный сигнал

Функция stepfun

Функция stepfun возвращает сигнал типа «единичный скачок»:

```
stepfun(t,t0)
```

Здесь t — вектор-строка моментов времени, и t0 — некоторый момент времени. Возвращаемая величина — сигнал, соответствующий функции единичного скачка, происходящего в момент t0.

Частотный отклик

Следующая группа функций служит для расчета и построения функции частотного отклика систем.

Функция bode

Функция bode возвращает диаграммы Боде — графики логарифмической амплитудно-частотной характеристики (ЛАЧХ) и фазочастотной характеристики (ФЧХ):

```
bode(sys)
bode(sys,w)
bode(sys1,sys2,...,sysN)
bode(sys1,sys2,...,sysN,w)
bode(sys1,'PlotStyle1',...,sysN,'PlotStyleN')
[mag,phase,w] = bode(sys)
```

Здесь аргумент w задает диапазон частот (w = [wmin,wmax]), остальные аргументы — такие же, как у ранее рассмотренных функций. Возвращаемые величины:

- диаграммы Боде для моделей с заданными именами;
- mag, phase, w — соответственно, массивы значений амплитуд и фаз и вектор частот.

Функция sigma

Функция sigma осуществляет построение частотной характеристики сингулярных чисел модели:

```
sigma(sys)
sigma(sys,w)
sigma(sys,w,type)
sigma(sys1,sys2,...,sysN)
sigma(sys1,sys2,...,sysN,w)
sigma(sys1,sys2,...\sysN,w,type)
sigma(sys1,'PlotStyle1',...,sysN,'PlotStyleN')
[sv,w] = sigma(sys)
sv = sigma(sys,w)
```

Для непрерывных моделей рассматриваемая функция вычисляет частотную характеристику сингулярных чисел (определение сингулярных чисел см. выше, при описании функции norm) матрицы $W(j\omega)$. Для дискретных моделей с дискретной матричной передаточной функцией $W(z)$ вычисляется частотная характеристика матрицы $W(e^{j\omega T_s})$ для диапазона частот от 0 до π/Ts. Частотные характеристики сингулярных чисел являются обобщением понятия логарифмических частотных характеристик и оказываются весьма полезными при анализе робастности.

Аргументы этой функции следующие:

- sys, sys1, sys2,..., sysN — имена моделей;
- w — диапазон частот, w = {wmin,wmax};
- 'PlotStyle1', ..., 'PlotStyleN' — аргументы, определяющие стили графиков (типы линий);
- аргумент type задает модификацию частотной характеристики сингулярных чисел:
  - 1 — для $W^{-1}(j\omega)$;
  - 2 — для $\mathbf{I} + W(j\omega)$;
  - 3 — для $\mathbf{I} + W^{-1}(j\omega)$.

Возвращаемые величины:

- графики частотных характеристик сингулярных чисел для моделей с указанными именами;
- sv — массив значений сингулярных чисел;
- w — соответствующий вектор частот.

Функция nyquist

Функция nyquist возвращает диаграмму Найквиста (годограф):

```
nyquist(sys)
nyquist(sys,w)
nyquist(sys1,sys2,...,sysN)
nyquist(sys1,sys2,...,sysN,w)
nyquist(sys1,'PlotStyle1',...,sysN,'PlotStyleN')
[re,im,w] = nyquist(sys)
[re,im] = nyquist(sys,w)
```

Аргументы — как у предыдущей функции. Возвращаемые величины:

- диаграмма Найквиста (в многомерном случае — для каждого канала «вход-выход»);

- re и im — соответственно, векторы значений вещественной и мни мой частей комплексного коэффициента передачи для заданного набора частот.

Функция nichols

Функция nichols возвращает годограф Николса:

```
nichols(sys)
nichols(sys,w)
nichols(sys1,sys2,....,sysN)
nichols(sys1,sys2,....,sysN,w)
nichols(sys1,'PlotStyle1',....,sysN,'PlotStyleN')
[mag,phase,w] = nichols(sys)
[mag,phase] = nichols(sys,w)
```

Аргументы — те же, что у предыдущей функции. Возвращаемые ве личины:

- годографы Николса;
- mag и phase — массивы значений модуля и фазы комплексного ко эффициента передачи;
- w — соответствующий вектор частот.

Для примера построим годограф Николса для одномерной системь с передаточной функцией

$$W(p)=\frac{-4p^4+48p^3-18p^2+250p+600}{p^4+30p^3+282p^2+525p+60}$$

и нанесем на график диаграмму (сетку) Николса:

```
» % Формирование модели
» num=[-4 48 -18 250 600];
» den=[1 30 282 525 60];
» W=tf(num,den);
» %Построение годографа Николса
» nichols(W);
» ngrid
```

Результат показан на рис. 3.24.

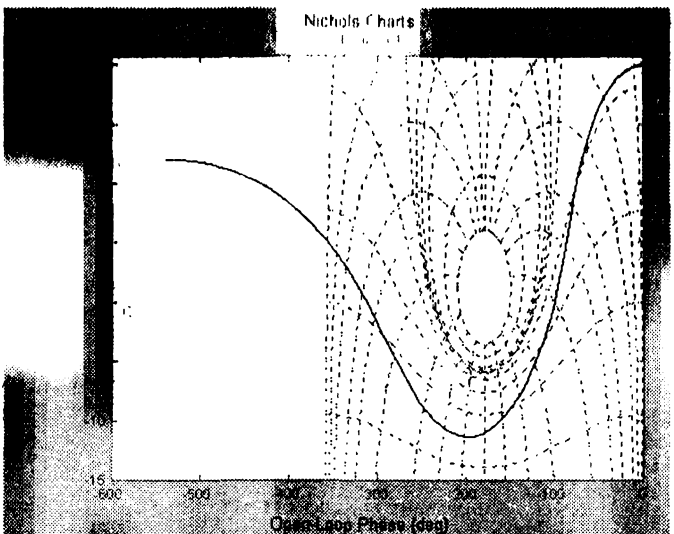


Рис. 3.24. Пример годографа Николса

Функция margin

Функция `margin` вычисляет запасы устойчивости по фазе и амплитуде:

```
[Gm,Pm,Wcg,Wcp] = margin(sys)
[Gm,Pm,Wcg,Wcp] = margin(mag,phase,w)
margin(sys)
```

Функция `margin(sys)` строит логарифмические частотные характеристики разомкнутой системы с указанием запасов ее устойчивости.

Функция `[Gm,Pm,Wcg,Wcp] = margin(sys)` рассчитывает запас устойчивости по амплитуде (`Gm`), по фазе (`Pm`) и соответствующие частоты для одномерной разомкнутой системы `sys`. Если имеется несколько точек пересечения амплитудной характеристики на уровне 0 дБ и фазовой на уровне −180°, то возвращаются наименьшие значения запасов устойчивости.

Функция `[Gm,Pm,Wcg,Wcp] = margin(mag,phase,w)` возвращает значения запасов устойчивости, вычисленных по значениям массива точек (`mag,phase,w`), где `mag` — амплитуда, `phase` — фаза, `w` — частота. В этом случае для определения запасов устойчивости необходимо выполнить интерполяцию для определения соответствующих точек пересечения.

Функция freqresp

Функция freqresp вычисляет значения комплексного коэффициента передачи для заданного набора частот:

```
H = freqresp(sys,w)
```

Здесь sys — имя модели, и w — вектор-строка значений частот. Возвращаемая величина H — массив значений $W(j\omega)$.

Функция evalfr

Функция evalfr вычисляет значение комплексного коэффициента передачи для одного заданного комплексного значения частоты:

```
frsp = evalfr(sys,f)
```

Здесь sys — имя модели, и f — заданное значение частоты. Возвращаемая величина frsp равна $W(f)$.

Функция linspace

Функция linspace возвращает вектор-строку линейно возрастающих частот:

```
linspace(x1,x2)
linspace(x1,x2,N)
```

Здесь x1 и x2 — верхняя и нижняя границы диапазона частот, и N — количество задаваемых значений (по умолчанию 100). Возвращается вектор-строка, элементами которого являются частоты, эквидистантно расположенные на отрезке [x1, x2]. Пример, иллюстрирующий работу с этой функцией, дан ниже:

```
» linspace(0,10,6)
ans =
     0     2     4     6     8    10
```

Функция logspace

Функция logspace возвращает вектор-строку логарифмически возрастающих частот:

```
logspace(d1,d2)
logspace(d1,d2,N)
```

Этой функцией создается вектор-строка, элементами которого являются частоты, логарифмически равномерно возрастающие на отрезке от 10^{d1} до 10^{d2}. Число элементов N по умолчанию равно 50. Пример:

```
» logspace(0.1,1,6)
ans =
   1.2589    1.9055    2.8840    4.3652    6.6069   10.0000
```

Функция ngrid

Функция ngrid (без аргументов) наносит диаграмму (сетку) Николса на одноименный годограф.

Композиция систем

Функции данной группы позволяют формировать системы различной структуры, используя в качестве элементов такой структуры LTI-модели.

- append — объединение LTI-систем путем объединения входов и выходов;
- parallel — обобщённое параллельное соединение (см. также процедуру «сложения» систем в разделе «Арифметические операции с моделями»);
- series — обобщенное последовательное соединение (см. также процедуру «умножения» систем в разделе «Арифметические операции с моделями»);
- feedback — образование замкнутой структуры из двух систем;
- lft — образование системы с перекрестными связями (соединение систем так называемой звездой Редхеффера);
- connect — построение LTI-модели на основе структурной схемы;
- conv — умножение двух полиномов;
- ord2 — формирование модели 2-го порядка;
- stack — создание массива моделей (все модели имеют одинаковое число входов и выходов);
- augstate — дополняет выходы переменными состояния модели.

Более подробная информация о данных функциях может быть получена с помощью справочной системы MATLAB или командой help имя_функции.

Традиционное проектирование систем

Следующая группа функций служит целям традиционного проектирования систем.

Функция rlocus

Функция rlocus выполняет расчет и построение корневого годографа по известной LTI-модели разомкнутой одномерной системы sys:

```
rlocus(sys)
rlocus(sys,k)
```

```
[r,k] = rlocus(sys)
r = rlocus(sys,k)
```

Здесь sys — имя модели, и k — начальное значение коэффициента усиления (вектор-строка таких значений для многомерного объекта) звена обратной связи. Возвращаемые значения:

- график корневого годографа;
- r — массив полюсов при изменении k от 0 до ∞;
- k — соответствующий массив значений коэффициентов усиления.

Рассмотрим следующий пример. Построим корневой годограф системы с передаточной функцией

$$W(p)=\frac{2p^2+5p+1}{p^2+2p+3}.$$

```
» W=tf([2 5 1],[1 2 3]); %Формирование модели
» rlocus(W)
```

Результат показан на рис. 3.25.

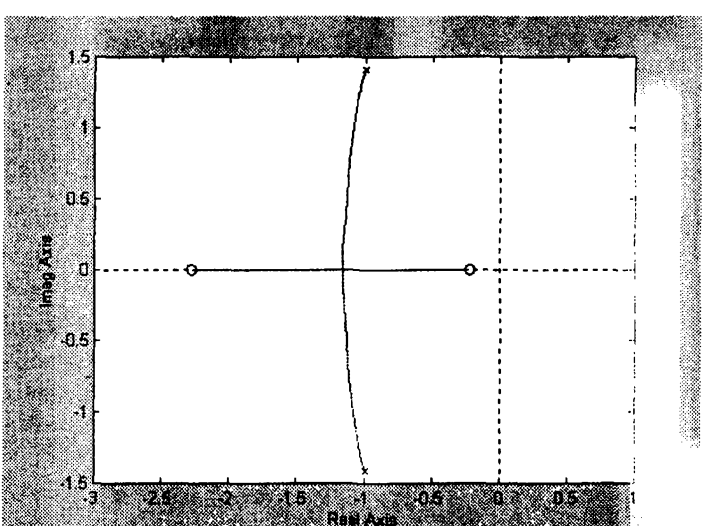


Рис. 3.25. Результат выполнения функции rlocus

Функция rlocfind

Функция rlocfind осуществляет подбор коэффициента передачи звена обратной связи по заданному набору полюсов:

```
[k,poles] = rlocfind(sys)
[k,poles] = rlocfind(sys,p)
```

Функция `[k,poles] = rlocfind(sys)` выполняется в интерактивном режиме после выполнения функции `rlocus`, при этом на комплексной плоскости указывается желаемое положение полюса.

Функция `[k,poles] = rlocfind(sys,p)` аналогична, но желаемые полюсы задаются в векторе-строке `p`.

Возвращаемые величины: обобщенный коэффициент усиления `k` и вектор полюсов `poles`.

Рассмотрим пример. Используя модель из предыдущего примера, зададим полюс, равный −2:

```
» [k,poles]=rlocfind(W,-2)
k =
    3.0000
poles =
   -2.0000
   -0.4286
```

Результат расчета: при $k = 3$ полюсы замкнутой системы равны −2 и −0,4286.

Функция acker

Функция `acker` возвращает значение вектора усиления обратной связи, обеспечивающего желаемое расположение полюсов для одномерных систем:

```
k = acker(A,B,p)
```

Здесь аргументы `A` и `B` — матрицы ss-модели, `p` — вектор желаемых полюсов.

Возвращаемая величина `k` — вектор коэффициентов усиления обратной связи.

Функция `place` по назначению аналогична предыдущей функции, но может использоваться как для одномерных, так и для многомерных систем:

```
K = place(A,B,p)
[K,prec,message] = place(A,B,p)
```

Аргументы этой функции те же, что у функции `acker`. Возвращаемые величины:

- `K` — матрица коэффициентов усиления;
- `prec` — число точных десятичных цифр у полюсов найденной замкнутой системы;

- message — предупреждение, выдаваемое, если некоторые полюсы замкнутой системы отличаются от желаемых более, чем на 10 %.

Функция estim

Функция estim формирует наблюдающее устройство (наблюдатель):

```
est = estim(sys,L)
est = estim(sys,L,sensors,known)
```

Функция estim(sys,L) возвращает ss-модель наблюдателя для непрерывной модели (все входы модели предполагаются случайными и неизмеряемыми, а все выходы измеряемыми) в соответствии с уравнениями

$$\begin{cases} \dot{\bar{\mathbf{x}}} = \mathbf{A}\bar{\mathbf{x}} + \mathbf{L}(\mathbf{y} - \mathbf{C}\bar{\mathbf{x}}), \\ \begin{bmatrix} \bar{\mathbf{y}} \\ \bar{\mathbf{x}} \end{bmatrix} = \begin{bmatrix} \mathbf{C} \\ \mathbf{I} \end{bmatrix} \bar{\mathbf{x}}, \end{cases}$$

где $\bar{\mathbf{x}}$, $\bar{\mathbf{y}}$ — оценки переменных состояния и выхода объекта.

Наблюдатель для дискретной модели описывается аналогичными уравнениями.

Функция est = estim(sys,L,sensors,known) работает с объектами управления более общей структуры, позволяя учитывать измеряемые сигналы управления **u**, случайные измеряемые входы **w**, а также измеряемые выходы **y** и неизмеряемые выходы **z**:

$$\begin{cases} \dot{\bar{\mathbf{x}}} = \mathbf{A}\bar{\mathbf{x}} + \mathbf{B}_1\mathbf{w} + \mathbf{B}_2\mathbf{u}, \\ \begin{bmatrix} \mathbf{z} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{C}_1 \\ \mathbf{C}_2 \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{D}_{11} \\ \mathbf{D}_{21} \end{bmatrix} \mathbf{w} + \begin{bmatrix} \mathbf{D}_{12} \\ \mathbf{D}_{22} \end{bmatrix} \mathbf{u}. \end{cases}$$

Индексы векторов sensors и known определяют, какие выходы являются измеряемыми и какие входы известными. Полученный наблюдатель est использует для вычисления оценок переменных состояния как вектор **u**, так и вектор **y**:

$$\begin{cases} \dot{\bar{\mathbf{x}}} = \mathbf{A}\bar{\mathbf{x}} + \mathbf{B}_1\mathbf{w} + \mathbf{B}_2\mathbf{u} + \mathbf{L}(\mathbf{y} - \mathbf{C}_2\bar{\mathbf{x}} - \mathbf{D}_{22}\mathbf{u}), \\ \begin{bmatrix} \bar{\mathbf{y}} \\ \bar{\mathbf{x}} \end{bmatrix} = \begin{bmatrix} \mathbf{C}_2 \\ \mathbf{I} \end{bmatrix} \bar{\mathbf{x}} + \begin{bmatrix} \mathbf{D}_{22} \\ 0 \end{bmatrix} \mathbf{u}. \end{cases}$$

Функция reg

Функция reg формирует динамический регулятор (компенсатор) rsys для заданной в пространстве состояний модели объекта управления sys, матрицы коэффициентов обратных связей по переменным со-

стояния K и матрицы L коэффициентов обратных связей наблюдателя:

```
rsys = reg(sys,K,L)
rsys = reg(sys,K,L,sensors,known,controls)
```

Уравнения регулятора имеют вид

$$\begin{cases} \dot{\bar{\mathbf{x}}} = [\mathbf{A} - \mathbf{LC} - (\mathbf{B} - \mathbf{LD})\mathbf{K}]\bar{\mathbf{x}} + \mathbf{Ly}, \\ \mathbf{u} = -\mathbf{K}\bar{\mathbf{x}}. \end{cases}$$

Аргумент controls определяет, какие входы являются измеряемыми, остальные аргументы — как у предыдущей функции. Возвращаемая величина — модель регулятора в ss-форме.

Функции sgrid и zgrid

Функции sgrid и zgrid строят сетки постоянных коэффициентов демпфирования от 0 до 1 с шагом 0,1 и собственных частот от 0 до 10 рад/с с шагом 1 рад/с на плоскостях p и z, соответственно:

```
sgrid
sgrid(z,wn)
zgrid
zgrid(z,wn)
```

Функции sgrid(z,wn) и zgrid(z,wn) строят сетки постоянных коэффициентов демпфирования и собственных частот для значений, указанных в векторах z и wn.

Аналитическое конструирование регуляторов

Для аналитического конструирования регуляторов служит группа перечисленных ниже функций. Она содержит следующие 7 функций.

Функция lqr

Функция lqr осуществляет синтез оптимального регулятора для непрерывной системы:

```
[K,S,e] = lqr(A,B,Q,R)
[K,S,e] = lqr(A,B,Q,R,N)
```

Функция [K,S,e] = lqr(A,B,Q,R,N) рассчитывает матрицу коэффициентов обратных связей K, такую, что закон управления $\mathbf{u} = -\mathbf{Kx}$ минимизирует квадратичный критерий качества:

$$J(\mathbf{u}) = \int_0^\infty (\mathbf{x}^T\mathbf{Qx} + \mathbf{u}^T\mathbf{Ru} + 2\mathbf{x}^T\mathbf{Nu})dt$$

для непрерывной ss-модели. Если аргумент N опущен, то по умолчанию N = 0.

В дополнение к матрице коэффициентов обратных связей K возвращается решение S непрерывного уравнения Риккати

$$\mathbf{A}^{\mathrm{T}}\mathbf{S}+\mathbf{SA}-(\mathbf{SB}+\mathbf{N})\mathbf{R}^{-1}(\mathbf{B}^{\mathrm{T}}\mathbf{S}+\mathbf{N}^{\mathrm{T}})+\mathbf{Q}=0$$

и собственные значения матрицы замкнутой системы e = eig(A – BK). При этом матрица K определяется выражением

$$\mathbf{K}=\mathbf{R}^{-1}(\mathbf{B}^{\mathrm{T}}\mathbf{S}+\mathbf{N}^{\mathrm{T}}).$$

Функция dlqr

Функция dlqr осуществляет синтез оптимального регулятора для дискретной системы:

```
[K,S,e] = dlqr(A,B,Q,R)
[K,S,e] = dlqr(A,B,Q,R,N)
```

Смысл аргументов и возвращаемых величин — как у предыдущей функции.

Функция lqry

Функция lqry осуществляет синтез оптимального регулятора с ограничением на выходные сигналы:

```
[K,S,e] = lqry(sys,Q,R)
[K,S,e] = lqry(sys,Q,R,N)
```

Здесь sys — имя модели объекта. Остальные аргументы и возвращаемые величины — как у двух предыдущих функций. Отличие данной функции от функции lqr состоит в виде используемого критерия оптимизации. В данном случае

$$J(\mathbf{u})=\int_0^{\infty}(\mathbf{y}^{\mathrm{T}}\mathbf{Q}\mathbf{y}+\mathbf{u}^{\mathrm{T}}\mathbf{R}\mathbf{u}+2\mathbf{y}^{\mathrm{T}}\mathbf{N}\mathbf{u})dt.$$

Функция lqrd

Функция lqrd осуществляет синтез дискретного регулятора для непрерывной системы:

```
[Kd,S,e] = lqrd(A,B,Q,R,Ts)
[Kd,S,e] = lqrd(A,B,Q,R,N,Ts)
```

Рассматриваемая функция предназначена для расчета дискретных регуляторов с обратными связями по всем переменным состояния с динамическими характеристиками, аналогичными характеристикам непрерывного регулятора, найденного с помощью функции lqr.

Функция, вызванная в форме [Kd,S,e] = lqrd(A,B,Q,R,N,Ts), рассчитывает дискретный закон управления

$$\mathbf{u}(n)=-\mathbf{K}_d\mathbf{x}(n),$$

который минимизирует критерий качества, эквивалентный непрерывному критерию, используемому в функции lqr.

Аргумент Ts — интервал дискретизации, остальные аргументы и возвращаемые величины аналогичны используемым в функции lqr.

Функция kalman

Функция kalman осуществляет синтез фильтра Калмана:

```
[kest,L,P] = kalman(sys,Qn,Rn,Nn)
[kest,L,P,M,Z] = kalman(sys,Qn,Rn,Nn) % Только для дискретных моделей
[kest,L,P] = kalman(sys,Qn,Rn,Nn,sensors,known)
```

Рассматриваемая функция выполняет синтез фильтра Калмана для оценки переменных состояния объекта управления на основе данных о случайных внешних возмущениях и ошибках измерений.

Для непрерывного случая предполагается заданной следующая модель объекта управления:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{G}\mathbf{w} & \text{(уравнение состояний)} \\ \mathbf{y}_v = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} + \mathbf{H}\mathbf{w} + \mathbf{v} & \text{(уравнение наблюдений)} \end{cases}$$

с известными входами $\mathbf{u}$ и возмущениями по входам $\mathbf{w}$ и измерениям $\mathbf{v}$, которые являются «белым шумом» со следующими характеристиками:

$$M\{\mathbf{w}\} = M\{\mathbf{v}\} = \mathbf{0},$$
$$M\{\mathbf{w}(t)\mathbf{w}^{\mathrm{T}}(\tau)\} = \mathbf{Q}_n\delta(t-\tau),$$
$$M\{\mathbf{v}(t)\mathbf{v}^{\mathrm{T}}(\tau)\} = \mathbf{R}_n\delta(t-\tau),$$
$$M\{\mathbf{v}(t)\mathbf{w}^{\mathrm{T}}(\tau)\} = \mathbf{N}_n\delta(t-\tau).$$

Синтез наблюдателя для оценивания вектора переменных состояния объекта проводится исходя из минимизации установившейся ошибки оценивания

$$P = \lim_{t\to\infty} M\{(\mathbf{x}-\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})^{\mathrm{T}}\}.$$

Оптимальным решением является фильтр Калмана, описываемый уравнениями

$$\begin{cases} \dot{\bar{\mathbf{x}}} = \mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\mathbf{u} + \mathbf{L}(\mathbf{y}_v - \mathbf{C}\bar{\mathbf{x}} - \mathbf{D}\mathbf{u}), \\ \begin{bmatrix} \bar{\mathbf{y}}_v \\ \bar{\mathbf{x}} \end{bmatrix} = \begin{bmatrix} \mathbf{C} \\ \mathbf{I} \end{bmatrix}\bar{\mathbf{x}} + \begin{bmatrix} \mathbf{D} \\ 0 \end{bmatrix}\mathbf{u} + \mathbf{H}\mathbf{w} + \mathbf{v}, \end{cases}$$

где матрица коэффициентов обратных связей $\mathbf{L}$ определяется на основе решения алгебраического матричного решения Рикатти.

Аналогично определяется фильтр Калмана и для дискретного случая.

Аргументы данной функции следующие:

- sys — имя модели объекта;
- Qn, Rn, Nn — матрицы, определяющие характеристики возмущений и поясненные выше;
- sensors и known — векторы, определяющие, какие выходы системы измеряются и какие входы известны (все другие входы предполагаются случайными и неизвестными).

Возвращаемые величины:

- kest — ss-модель фильтра Калмана;
- L и P — матрицы приведенного вида;
- M — обновленная матрица обратных связей;
- Z — ковариационная матрица ошибок оценивания в установившемся режиме.

Функция kalmd

Функция kalmd осуществляет синтез дискретного фильтра Калмана для непрерывной системы:

```
[kest,L,P,M,Z] = kalmd(sys,Qn,Rn,Ts)
```

Рассматриваемая функция рассчитывает дискретный фильтр Калмана с динамическими характеристиками, подобными характеристикам непрерывного фильтра, рассчитанного с помощью функции kalman. Фактически эта функция предназначена для цифровой реализации непрерывного фильтра. Аргумент Ts — интервал дискретизации. Остальные аргументы и возвращаемые величины — как у функции kalman.

Функция lqgreg

Функция lqgreg формирует так называемый линейный квадратичный гауссов регулятор (ЛКГ-регулятор):

```
rlqg = lqgreg(kest,K)
rlqg = lqgreg(kest,K,'current') % Только для дискретных моделей
rlqg = lqgreg(kest,K,controls)
```

Рассматриваемая функция формирует ЛКГ-регулятор путем соединения рассчитанного с помощью функции kalman фильтра Калмана и матрицы коэффициентов обратных связей, рассчитанной с помощью функций lqr, dlqr или lqry. ЛКГ-регулятор является оптимальным и минимизирует некоторый квадратичный критерий качества, оценивающий эффективность регулирования и затраты на управление.

ЛКГ-регулятор синтезируется как динамический объект, обеспечивающий фильтрацию шумов измерений выходных сигналов объекта управления.

В непрерывном варианте данный регулятор имеет следующее описание:

$$\begin{cases} \dot{\bar{\mathbf{x}}} = [\mathbf{A} - \mathbf{LC} - (\mathbf{B} - \mathbf{LD})\mathbf{K}]\bar{\mathbf{x}} + \mathbf{L}\mathbf{y}_v, \\ \mathbf{u} = -\mathbf{K}\bar{\mathbf{x}}, \end{cases}$$

где $\mathbf{y}_v$ — вектор измеряемых выходов объекта управления.

Аналогичный вид имеет и описание регулятора для дискретного варианта.

Аргументы функции следующие:

- `kest` — имя фильтра Калмана, синтезированного с помощью функции `kalman`;
- `K` — матрица обратных связей (найденная с помощью функций `lqr`, `dlqr` или `lqry`);
- аргумент `'current'` задает расчеты сигнала управления на основе текущего измерения, то есть не в режиме реального времени;
- вектор индексов `controls` указывает, какие входы наблюдателя являются измеряемыми сигналами управления `ud`.

Возвращаемая величина — ss-модель ЛКГ-регулятора с именем `rlqg`.

Решение матричных уравнений

Следующие функции используются для решения матричных уравнений с повышенной эффективностью.

Функция lyap

Функция `lyap` возвращает решение непрерывного матричного уравнения Ляпунова:

```
X = lyap(A,Q)
X = lyap(A,B,C)
```

Функция `X = lyap(A,Q)` находит решение для матричного уравнения Ляпунова вида

$$\mathbf{AX} + \mathbf{XA}^T + \mathbf{Q} = \mathbf{0},$$

где $\mathbf{A}$ и $\mathbf{Q}$ — квадратные матрицы одинаковых размеров. Решение $\mathbf{X}$ является симметричной матрицей, если таковой является матрица $\mathbf{Q}$.

Функция `X = lyap(A,B,C)` возвращает решение обобщенного уравнения Ляпунова или уравнения Сильвестра вида

$$\mathbf{AX} + \mathbf{XB} + \mathbf{C} = \mathbf{0}.$$

Матрицы **A**, **B**, **C** должны иметь согласованные размеры, но не обязательно быть квадратными.

Функция dlyap

Функция `dlyap` возвращает решение так называемого дискретного уравнения Ляпунова:

```
X = dlyap(A,Q)
```

Рассматриваемая функция решает дискретное уравнение Ляпунова

$$\mathbf{A}^T\mathbf{XA} - \mathbf{X} + \mathbf{Q} = \mathbf{0},$$

где **A** и **Q** — квадратные матрицы.

Функция care

Функция `care` находит решение непрерывного алгебраического уравнения Рикатти:

```
[X,L,G,rr] = care(A,B,Q)
[X,L,G,rr] = care(A,B,Q,R,S,E)
[X,L,G,report] = care(A,B,Q,[],[],'report')
[X1,X2,L,report] = care(A,B,Q,[],[],'implicit')
```

Функция `[X,L,G,rr] = care(A,B,Q)` находит единственное решение алгебраического уравнения Рикатти:

$$\mathrm{Ric}(\mathbf{X}) = \mathbf{A}^T\mathbf{X} + \mathbf{XA} - \mathbf{XBB}^T\mathbf{X} + \mathbf{Q} = \mathbf{0}$$

в виде такой матрицы **X**, что все собственные значения матрицы замкнутой системы $\mathbf{A} - \mathbf{BB}^T\mathbf{X}$ расположены в левой полуплоскости комплексной переменной p. Кроме этого, функция возвращает:

- собственные значения `L` матрицы $\mathbf{A} - \mathbf{BB}^T\mathbf{X}$;
- матрицу коэффициентов обратных связей $\mathtt{G} = \mathbf{B}^T\mathbf{X}$;
- относительную невязку решения `rr`, определяемую выражением

$$rr = \frac{\|\mathbf{Ric}(\mathbf{X})\|}{\|\mathbf{X}\|}.$$

Функция `[X,L,G,rr] = care(A,B,Q,R,S,E)` решает обобщенное уравнение Рикатти вида

$$\mathrm{Ric}(\mathbf{X}) = \mathbf{A}^T\mathbf{XE} + \mathbf{E}^T\mathbf{XA} - (\mathbf{E}^T\mathbf{XB} + \mathbf{S})\mathbf{R}^{-1}(\mathbf{B}^T\mathbf{XE} + \mathbf{S}^T) + \mathbf{Q} = \mathbf{0}.$$

Здесь матрица коэффициентов обратных связей G рассчитывается следующим образом:

$$\mathbf{G} = \mathbf{R}^{-1}(\mathbf{B}^T\mathbf{X}\mathbf{E} + \mathbf{S}^T),$$

а собственные числа L определяются путем решения обобщенной проблемы собственных значений для матриц **A** – **BG** и **E**

```
L = eig(A - BG, E)
```

Функция `[X,L,G,report] = care(A,B,Q,...,'report')` возвращает сообщение об ошибке, когда решения уравнения Рикатти не существует. Возвращаемая величина `report` может принимать следующие значения:

- −1, когда пара матриц **Γ**, **E** имеет обобщенные собственные значения на мнимой оси или очень близко к ней. Здесь **Γ** — матрица Гамильтона вида

$$\Gamma = \begin{bmatrix} \mathbf{A} & -\mathbf{B}\mathbf{B}^T \\ -\mathbf{Q} & -\mathbf{A}^T \end{bmatrix};$$

- −2, когда матрица $\mathbf{X}_1$ вырожденная и не существует конечного решения $\mathbf{X} = \mathbf{X}_2\mathbf{X}_1^{-1}$.

Возможно применение рассматриваемой функции в форме

```
[X1,X2,L,report] = care(A,B,Q,...,'implicit')
```

когда матрица X возвращается в неявной форме в виде пары матриц $\{\mathbf{X}_1, \mathbf{X}_2\}$. Заметим, что при таком формате в случае успешного решения возвращается значение `report = 0`.

Функция dare

Функция dare возвращает решение так называемого дискретного алгебраического уравнения Рикатти:

```
[X,L,G,rr] = dare(A,B,Q,R)
[X,L,G,rr] = dare(A,B,Q,R,S,E)
[X,L,G,report] = dare(A,B,Q,....,'report')
[X1,X2,L,report] = dare(A,B,Q,....,'implicit')
```

Функция `[X,L,G,rr] = dare(A,B,Q,R)` находит единственное решение алгебраического уравнения Рикатти

$$\mathrm{Ric}(\mathbf{X}) = \mathbf{A}^T\mathbf{X}\mathbf{A} - \mathbf{X} + \mathbf{A}^T\mathbf{X}\mathbf{B}(\mathbf{B}^T\mathbf{X}\mathbf{B} + \mathbf{R})^{-1}\mathbf{B}^T\mathbf{X}\mathbf{A} + \mathbf{Q} = \mathbf{0}$$

в виде такой матрицы X, что все собственные значения матрицы замкнутой системы

$$\mathbf{A}_c = \mathbf{A} - \mathbf{B}(\mathbf{B}^T\mathbf{X}\mathbf{B} + \mathbf{R})^{-1}\mathbf{B}^T\mathbf{X}\mathbf{A}$$

расположены внутри единичной окружности на плоскости комплексной переменной *z*. Кроме этого, функция возвращает:

- собственные значения L матрицы $\mathbf{A}_c$;

- матрицу коэффициентов обратных связей $G = (B^TXB + R)^{-1}B^TXA$;
- относительную невязку решения rr, определяемую выражением

$$rr = \frac{\|Ric(X)\|}{\|X\|}.$$

Функция [X,L,G,rr] = dare(A,B,Q,R,S,E) решает обобщенное уравнение Рикатти вида

$$Ric(X) = A^TXA + E^TXE - (A^TXB + S)(B^TXB + R)^{-1}(B^TXA + S^T) + Q = 0.$$

В данном случае матрица коэффициентов обратных связей равна

$$G = (B^TXB + R)^{-1}(B^TXA + S^T),$$

а собственные числа L определяются путем решения обобщенной проблемы собственных значений для матриц $A - BG$, E.

Функция [X,L,G,report] = dare(A,B,Q,…,'report') возвращает сообщение об ошибке, когда решения уравнения Рикатти не существует. Возвращаемая величина report имеет такие же значения, как и для предыдущей функции.

Возможно применение функции в форме [X1,X2,L,report] = dare(A,B,Q,…,'implicit'), когда матрица X возвращается в неявной форме в виде пары матриц $\{X_1, X_2\}$. Заметим, что при таком формате в случае успешного решения возвращается значение report = 0.

Вызов демонстрационных примеров

Для ознакомления с пакетом Control System можно использовать следующие функции (команды), задаваемые в режиме командной строки:

- ctrldemo — введение в Control System Toolbox;
- jetdemo — проектирование системы автоматического управления углом рыскания реактивного самолета;
- diskdemo — проектирование контроллера привода жесткого диска;
- milldemo — система автоматического управления прокатным станом;
- kalmdemo — проектирование и моделирование фильтра Калмана.

С каждым из этих примеров связан соответствующий m-файл, запуск которого производится с помощью одной из указанных команд. Текст пояснений в примерах — на английском языке. Данные примеры доступны и через главное меню MATLAB (пункт Help ▸ Examples and Demos, раздел Toolboxes ▸ Control System).

Дополнительную информацию можно получить, используя команду `help control`.

В качестве примера рассмотрим задачу проектирования контроллера привода жесткого диска.

Схема, поясняющая работу привода, приведена на рис. 3.26.

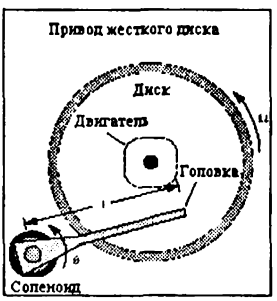


Рис. 3.26. Схема привода жесткого диска

Дифференциальное уравнение, описывающее динамику движения магнитной головки жесткого диска, записывается в следующем виде:

$$J\frac{d^2\theta}{dt^2} + C\frac{d\theta}{dt} + K\theta = K_i i,$$

где J — момент инерции магнитной головки, C — коэффициент вязкого трения в подшипниках, K — коэффициент жесткости пружины, K_i — моментный коэффициент двигателя, θ — угловое положение головки, i — ток якоря двигателя.

Применяя к данному уравнению преобразование Лапласа, найдем передаточную функцию от управляющей переменной i к выходной переменной θ:

$$W(p) = \frac{K}{Jp^2 + Cp + K}.$$

Используя далее числовые значения $J = 0{,}01$ кг·м², $C = 0{,}004$ Н·м/(рад/с), $K = 10$ Н·м/рад, $K_i = 0{,}05$ Н·м/А, сформируем следующую tf-модель:

```
» J=0.01; C=0.004; K=10; Ki =0.05;
» num=Ki;
» den=[J C K];
» W=tf(num,den)
Transfer function:
```

```
        0.05
-----------------------
0.01 s^2 + 0.004 s + 10
```

Задача состоит в синтезе цифрового регулятора, обеспечивающего точное позиционирование магнитной головки. Для ее решения сначала перейдем от непрерывной модели объекта к дискретной с интервалом дискретизации 0,005 с, используя экстраполятор нулевого порядка:

```
» Ts=0.005;
» Wd=c2d(W,Ts,'zoh')

Transfer function:
6.233e-005 z + 6.229e-005
------------------------
  z^2 - 1.973 z + 0.998

Sampling time: 0.005
```

Для проверки качества выполненной аппроксимации сравним частотные характеристики исходной непрерывной и полученной дискретной моделей (рис. 3.27):

```
» bode(W,'-',Wd,'--')
```

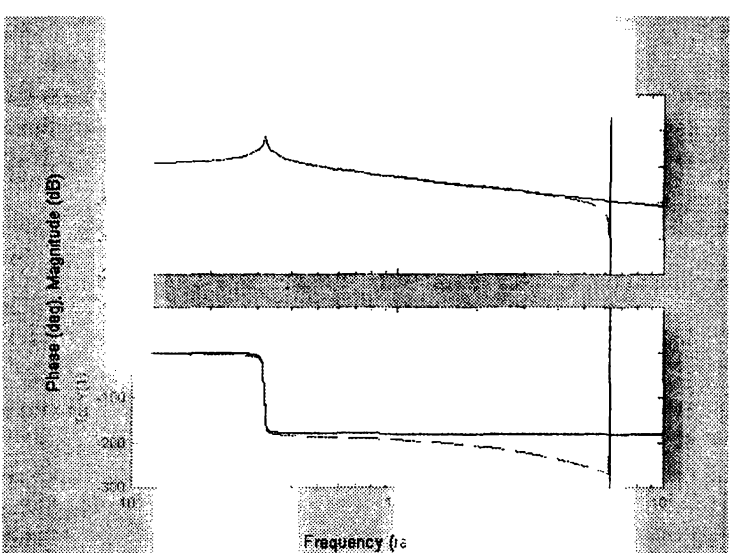


Рис. 3.27. Частотные характеристики непрерывной и дискретной моделей

Построим теперь переходную функцию для дискретной системы (рис. 3.28):

```
» step(Wd)
```

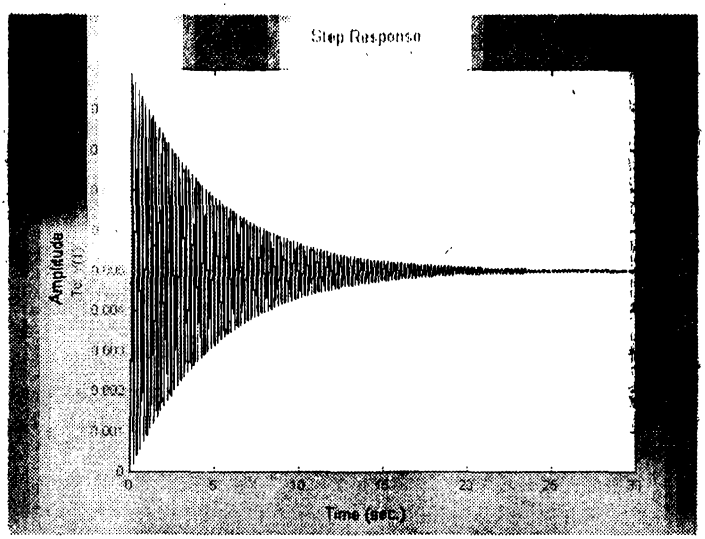


Рис. 3.28. Переходный процесс в дискретной системе

В системе присутствует ярко выраженная колебательность, причиной которой является слабое демпфирование. В этом легко убедиться, вычислив полюсы дискретной модели:

```
» %Полюсы дискретной модели
» damp(Wd)

      Eigenvalue          Magnitude  Equiv. Damping  Equiv. Freq. (rad/s)

9.87e-001 + 1.57e-001i  9.99e-001    6.32e-003          3.16e+001
9.87e-001 - 1.57e-001i  9.99e-001    6.32e-003          3.16e+001
```

Полюсы расположены вблизи единичной окружности и имеют малое демпфирование.

Такой результат позволяет конкретизировать задачу проектирования: необходимо рассчитать регулятор, который увеличит демпфирование данных полюсов. Наиболее простой вариант решения этой задачи — подбор коэффициента усиления системы, который можно осуществить с помощью метода корневого годографа (рис. 3.29):

```
» rlocus(Wd), zgrid
```

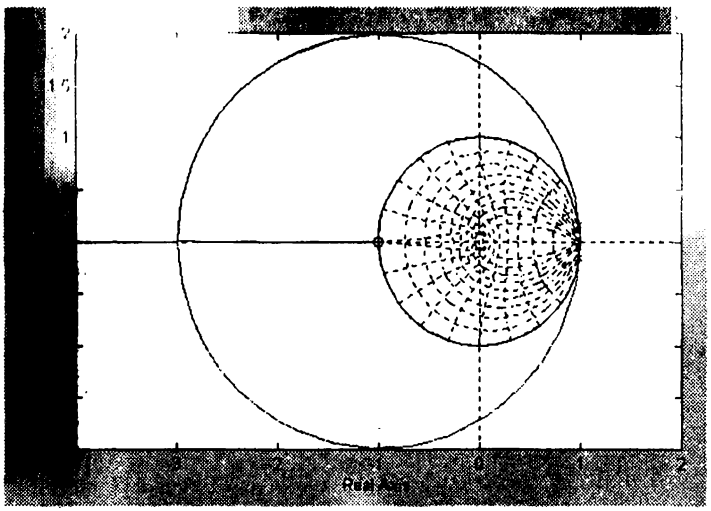


Рис. 3.29. Корневой годограф исследуемой системы

Из корневого годографа следует, что с увеличением коэффициента усиления полюсы замкнутой системы быстро выходят за пределы единичной окружности и система становится неустойчивой. Поэтому введем некоторую динамическую коррекцию в виде дискретного компенсатора с передаточной функцией

$$D(z) = \frac{z+a}{z+b},$$

где $a = -0{,}85$; $b = 0$.

Сформируем разомкнутую модель контура как последовательное соединение моделей объекта и компенсатора:

```
» D=zpk(0.85,0,1,Ts) %Формирование модели компенсатора

Zero/pole/gain:
(z-0.85)
--------
   z

Sampling time: 0.005
» oloop=Wd*D %Формирование разомкнутого контура

Zero/pole/gain:
```

```
6.2328e-005 (z+0.9993) (z-0.85)
-------------------------------
   z (z^2 - 1.973z + 0.998)

Sampling time: 0.005
```

Определим, как компенсатор модифицирует логарифмическую частотную характеристику магнитной головки (рис. 3.30):

```
» bode(Wd,'--',oloop,'-')
```

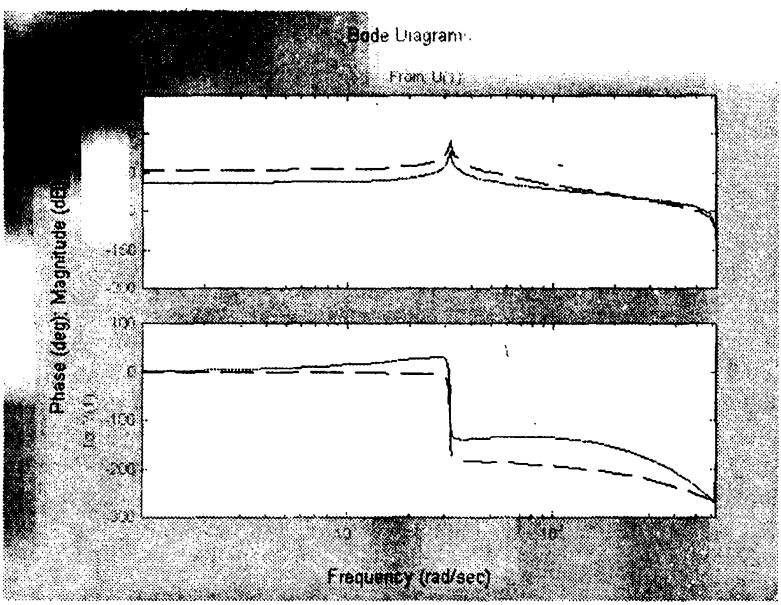


Рис. 3.30. Частотные характеристики магнитной головки и разомкнутого контура

На рис. 3.30 частотная характеристика магнитной головки показана пунктирной линией, а частотная характеристика контура с компенсатором — сплошной линией.

Из анализа характеристик следует, что компенсатор сдвигает фазовую характеристику вверх (вводит опережение) в диапазоне частот от 10 до 1000 рад/с.

Построим теперь корневой годограф для контура с компенсатором (рис. 3.31):

```
» rlocus(oloop), zgrid
```

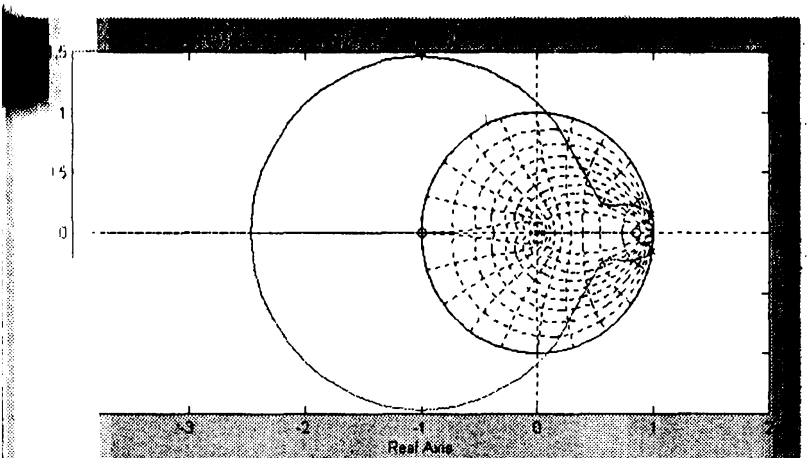


Рис. 3.31. Корневой годограф для контура с компенсатором

Теперь полюсы остаются внутри единичной окружности для большего диапазона значений коэффициента усиления.

Используя далее функцию rlocfind, зададим некоторое оптимальное, на наш взгляд, расположение полюсов:

```
[k,poles]=rlocfind(oloop)
Select a point in the graphics window

selected_point =
   0.4378 - 0.0088i
k =
  3.8643e+003
poles =
   0.6471 + 0.2199i
   0.6471 - 0.2199i
   0.4380
```

Определим демпфирование замкнутой системы:

```
ddamp(poles,Ts)

    Eigenvalue          Magnitude  Equiv. Damping  Equiv. Freq. (rad/s)

.47e-001 + 2.20e-001i  6.83e-001    7.58e-001        1.00e+002
.47e-001 - 2.20e-001i  6.83e-001    7.58e-001        1.00e+002
.38e-001               4.38e-001    1.00e+000        1.65e+002
```

Чтобы оценить результаты синтеза, сформируем замкнутую систему и построим для нее переходную функцию (рис. 3.32):

```
» cloop=feedback(oloop,k);
» step(cloop)
```

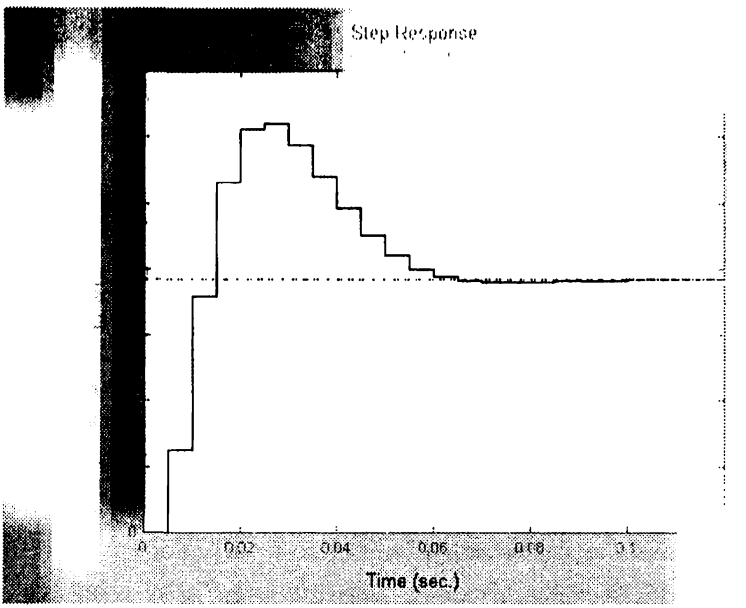


Рис. 3.32. Переходный процесс в замкнутой системе с компенсатором

Полученный переходный процесс имеет время установления около 14 периодов дискретизации, то есть 0,07 с. По современным требованиям к управлению магнитными головками это большая величина, но надо иметь в виду, что динамика головки характеризовалась очень малым демпфированием.

Оценим робастность полученной системы. Классическими критериями робастности являются, как известно, запасы устойчивости по фазе и амплитуде. Для их определения воспользуемся функцией `margin`, предварительно учтя найденное с помощью функции `rlocfind` значение `k`:

```
» olk=k*oloop;
» [Gm,Pm,Wcg,Wcp]=margin(olk);
» Margins = [Gm Wcg Pm Wcp]
Margins =
    4.0402  296.7989   43.8146  101.4708
```

Запасы устойчивости можно отобразить и на графике (рис. 3.33):

```
» margin(olk)
```

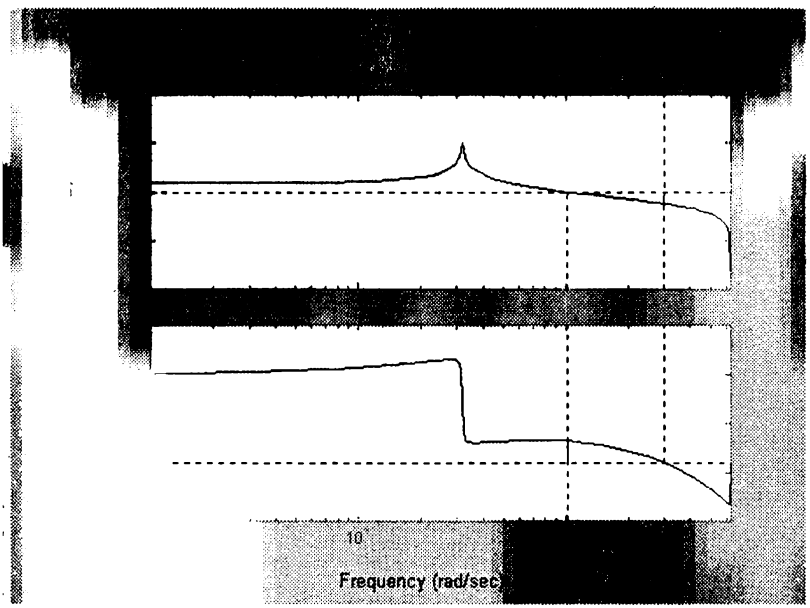


Рис. 3.33. ЛАЧХ и ФЧХ с показанными запасами устойчивости

Рассмотренная система является робастной, поскольку имеет запас устойчивости по амплитуде, равный

```
» 20*log10(Gm)
ans =
   12.1280 %Дб
```

и запас устойчивости по фазе, равный 43°.

Работа в среде Simulink

Кроме отмеченной выше возможности создания моделей Simulink с помощью графического интерфейса МКГ, создание моделей с применением LTI-объектов возможно непосредственно в среде Simulink, в частности, с применением блоков набора Control System Toolbox (рис. 3.34), графические изображения которых показаны на рис. 3.35.

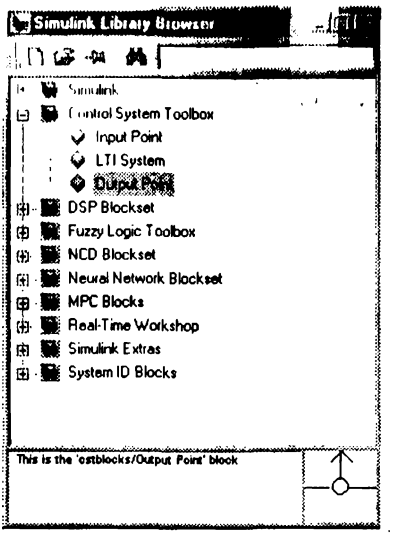


Рис. 3.34. Набор блоков Control System Toolbox

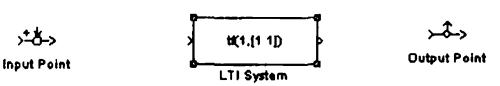


Рис. 3.35. Блоки Control System Toolbox

Использование моделей пакетов идентификации

К сожалению, в системе MATLAB 5.3.1 (и, естественно, в ее более ранних версиях) не реализована возможность прямого использования моделей (например, моделей тета-формата), получаемых с помощью функций пакетов System Identification и Frequency Domain System Identification. Поэтому единственным способом применения результатов идентификации для решения задач управления в рамках Control System Toolbox — это использование информации о полиномах передаточной функции, матрицах **A**, **B**, **C**, **D** модели для переменных состояния или частотных характеристиках с последующим формированием модели LTI-типа.

Глава 4
Пакет Nonlinear Control Design

Назначение пакета и документация по нему

Назначение пакета Nonlinear Control Design Blockset

Пакет прикладных программ для построения нелинейных систем управления Nonlinear Control Design (NCD) Blockset реализует метод динамической оптимизации. Этот инструмент, строго говоря, представляющий собой набор блоков, разработанных для использования с Simulink, автоматически настраивает параметры моделируемых систем, основываясь на определенных пользователем ограничениях на их временные характеристики.

Пакет использует метод Click And Drag («щелкни и тяни») для изменения временных ограничений. Он реализует следующие возможности:

- легкую настройку переменных;
- указание неопределенных параметров систем;
- интерактивную оптимизацию;
- моделирование методом Монте-Карло;
- поддержка проектирования как одномерных, так и многомерных систем управления;
- моделирование подавления помех;
- моделирование процессов слежения;
- моделирование объектов с запаздыванием;
- решение других задач управления.

Средства пакета являются весьма полезным дополнением к средствам моделирования линейных систем, предоставляя пользователю именно те возможности, которые отсутствуют в ранее рассмотренных пакетах расширения системы MATLAB.

Документация по пакету Nonlinear Control Design

Электронная документация по пакету Nonlinear Control Design представлена в формате HTML в справочной системе. Имеется также описание в формате PDF объемом 106 с. Таким образом, данный пакет является небольшим в сравнении с ранее рассмотренными пакетами, что ничуть не принижает его значения, учитывая отмеченные выше новые возможности пакета.

Рекомендуемая литература

Для знакомства с вопросами теории и практики нелинейных систем управления рекомендуется следующая литература.

1. Топчеев Ю. И. Атлас для проектирования систем автоматического регулирования: Учебное пособие для втузов. — М.: Машиностроение, 1989.
2. Дьяконов В. П., Абраменкова И. В. MATLAB 5.0/5.3. Система символьной математики. — М.: Нолидж, 1999.
3. Гультяев А. К. MATLAB 5.2. Имитационное моделирование в среде Windows: Практическое пособие. — СПб.: КОРОНА принт, 1999.
4. Гультяев А. К. Визуальное моделирование в среде MATLAB. Учебный курс. — СПб.: Питер, 2000.

Основные правила работы с NCD

Пакет расширения NCD является частью пакета Simulink и наследует все его приемы работы. Это, в частности, относится к вызову библиотек пакета NCD, их применению для построения моделей нелинейных систем и запуску процесса моделирования. В связи с этим воздержимся от подробного описания этих приемов.

Данный пакет, по сути, является специализированной оптимизирующей программой для решения задачи оптимизации при наличии ограничений в форме неравенств и использующей в качестве алгоритма оптимизации последовательное квадратичное программирование.

Описание блоков NCD

Рисунок 4.1 показывает окно броузера библиотек, демонстрирующее доступ к библиотекам пакета NCD. Сокращение RMS означает среднеквадратическую величину (root mean square).

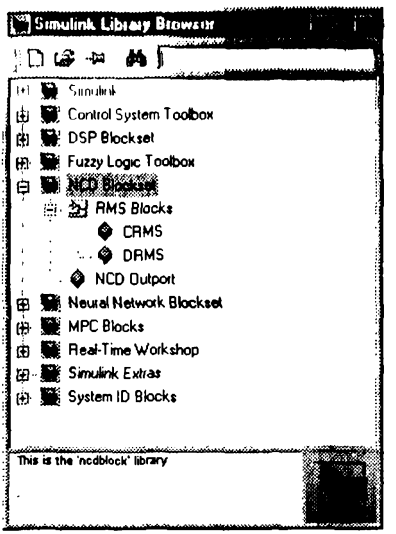


Рис. 4.1. Состав NCD Blockset

Пакет NCD Blockset содержит следующие блоки (рис. 4.2):

- блок CRMS (Continuous RMS);
- блок DRMS (Discrete RMS);
- блок NCD Outport.

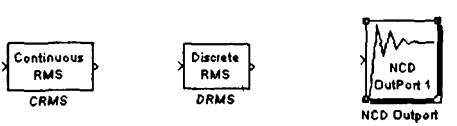


Рис. 4.2. Блоки пакета NCD Blockset

Рассмотрим их подробнее.

Блок CRMS реализует математическую зависимость

$$y(t) = \sqrt{\frac{1}{t}\int_0^t u^2(\tau)\,d\tau}, \text{ при } t > 0,$$

где $u(t)$ — входной сигнал блока, $y(t)$ — его выходной сигнал. Следует отметить, что для эргодических случайных процессов с нулевым математическим ожиданием выходной сигнал блока при $t \to \infty$ является среднеквадратическим (стандартным) отклонением.

Блок DRMS, по сути, реализует такую же зависимость, что и блок CRMS, но для сигналов, определенных в дискретные моменты времени:

$1/\sqrt{2} \approx 0.707$ при $n > 0$.

Пример с демонстрацией функционирования данных блоков содержится в файле rmsdemo. Для его запуска в режиме командной строки MATLAB введем команду:

```
» rmsdemo
```

В результате исполнения этой команды появится рабочее окно Simulink с моделью, содержащей источник синусоидального сигнала с единичной амплитудой, к выходу которого подключены рассмотренные блоки (рис. 4.3). С помощью блока Mux их выходы подключаются ко входу виртуального осциллографа, что задает отображение двух временных зависимостей — на выходе блоков CRMS и DRMS.

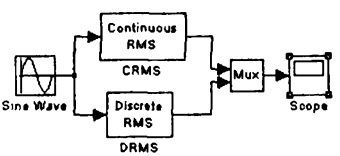


Рис. 4.3. Модель, иллюстрирующая функционирование блоков CRMS и DRMS

Активизируем блок Scope (двойным щелчком мыши) и запустим процесс моделирования. Его результат отображен на рис. 4.4, где плавная линия — выход блока CRMS, а ступенчатая — блока DRMS. При цветном дисплее они отличаются цветом линий.

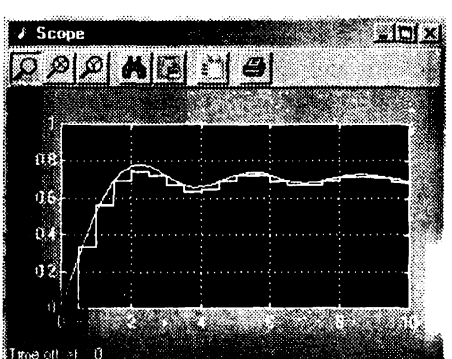


Рис. 4.4. Выходные сигналы блоков CRMS и DRMS

Как видно, с течением времени выходы обоих блоков стремятся к одному и тому же установившемуся значению — действующему значению синусоиды с единичной амплитудой, равному $1/\sqrt{2} \approx 0.707$. В то же время в случае блока CMRS это приближение получается плавным, а в случае блока DRMS — дискретным.

Рассматриваемые блоки могут применяться, в частности, в системах моделирования, где качество функционирования целесообразно оценивать интегральным квадратичным критерием или стандартным отклонением ошибки.

Блок NCD Output является основным в рассматриваемом наборе блоков. Он имеет свое рабочее окно и меню и позволяет в интерактивном режиме выполнять следующие операции:

- задавать требуемые ограничения во временной области на любой сигнал оптимизируемой системы;
- указывать параметры, подлежащие оптимизации;
- указывать неопределенные параметры;
- проводить параметрическую оптимизацию системы с учетом заданных ограничений.

Работа с данным блоком будет рассмотрена далее на примерах.

Правила работы с пакетом NCD

Типовой сеанс работы в среде Simulink с использованием возможностей и блоков NCD Blockset состоит из ряда стадий, описанных ниже.

1. В среде Simulink создается модель исследуемой динамической системы (в общем случае нелинейной).
2. Входы блоков NCD Outport соединяются с теми сигналами системы, на которые накладываются ограничения. Этими сигналами могут быть, например, выходы системы, их среднеквадратические отклонения и т. д.
3. В режиме командной строки MATLAB задаются начальные значения параметров, подлежащих оптимизации.
4. Двойным щелчком на пиктограмме NCD Output данные блоки «раскрываются».
5. При помощи мыши нужным образом изменяются конфигурации и размеры областей ограничений для нужных сигналов системы.
6. С помощью меню блока NCD Output задается интервал дискретизации (один или два процента от длительности процесса моделиро-

вания) и указываются имена (идентификаторы) параметров системы, подлежащих оптимизации.

7. Задаются неопределенные параметры системы, указываются их номинальные значения.
8. При необходимости сформированные ограничения сохраняются в виде файла с помощью команды меню Save (позднее они, естественно, могут быть загружены с помощью команды Load).
9. Процесс оптимизации системы инициализируется нажатием кнопки Start.

Таким образом, фактически здесь реализуются обычные приемы работы с Simulink. В связи с этим мы не будем затрагивать их более подробно и перейдем к описанию практики применения пакета NCD на характерных для него примерах.

Примеры моделирования и оптимизации

Оптимизация коэффициента передачи И-регулятора

Рассмотрим пример параметрической оптимизации замкнутой системы автоматического регулирования, используя структуру, описанную в файле ncdtut1. Для запуска этого примера в режиме командной строки выполним команду

```
» ncdtut1
```

Это приведет к открытию окна Simulink с моделью исследуемой системы (рис. 4.5).

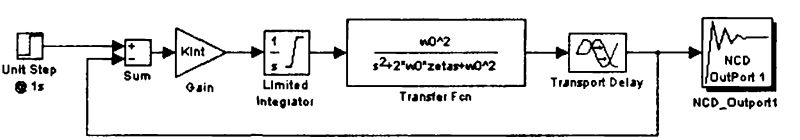


Рис. 4.5. Модель исследуемой системы

Данная система представляет собой замкнутую структуру, состоящую из:

1) объекта регулирования с передаточной функцией

$$\frac{\omega_0^2 e^{-p\tau}}{p^2 + 2\omega_0 \xi p + \omega_0^2},$$

который в модели отображен звеном 2-го порядка (блок Transfer Fcn) и звеном транспортного запаздывания (блок Transport Delay) со значениями $\tau = 1$ с, $\omega_0 = 1$ с$^{-1}$, $\xi = 1$ и обозначениями w0 = ω_0, zetas = ξ;

2) интегрального регулятора (И-регулятора), отображаемого последовательно соединенными пропорциональным звеном с коэффициентом пропорциональности Kint и интегрирующим звеном с ограничением на выходной сигнал (блок Limited Integrator);
3) контура обратной связи и звена сравнения Sum.

В модель также введены источник входного сигнала в виде единичного скачка и NCD-блок типа NCD Output, подключенный к выходу системы (здесь данный блок имеет имя NCD OutPort 1).

Нетрудно видеть, что в данном случае контролируемым сигналом является реакция системы на единичный скачок, то есть ее переходная функция. Настраиваемым (оптимизируемым) параметром является коэффициент Kint, а ограничения, накладываемые на переходную функцию, формулируются следующим образом:

- максимальное перерегулирование — не более 10 %;
- время нарастания — не более 10 с;
- длительность переходного процесса — не более 30 с.

При решении задачи оптимизации учтем, что первые два ее этапа (см. список выше) уже выполнены. Для выполнения третьего этапа в командной строке MATLAB наберем:

```
» % Задание числовых параметров системы
» zeta=1;
» w0=1;
» % Задание начального значения настраиваемого параметра
» Kint=0.3;
```

Далее двойным щелчком мыши откроем рабочее окно блока NCD OutPort 1. Вид данного окна приведен на рис. 4.6.

В графической части окна показаны границы контролируемого сигнала, устанавливаемые по умолчанию. Легко заметить, что они не соответствуют заданным ограничениям, поэтому изменим их. Для этого, используя указатель мыши, переместим вертикальные и горизонтальные линии ограничений до положений, показанных на рис. 4.7 и соответствующих нашим требованиям. Внимательно сравните рис. 4.6 и 4.7, чтобы понять суть изменений.

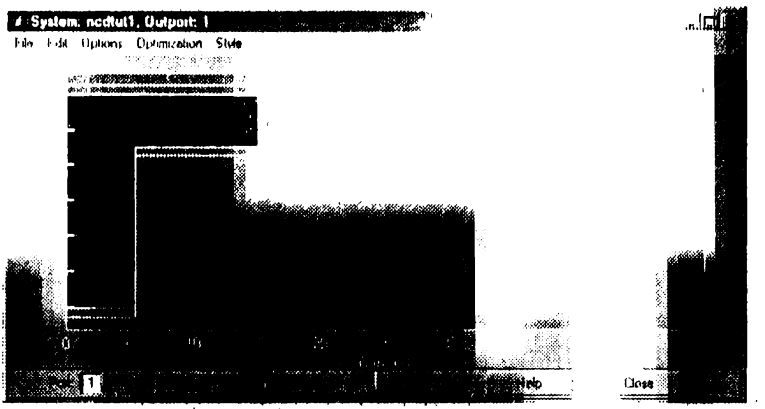


Рис. 4.6. Окно блока NCD Outport с временными ограничениями по умолчанию

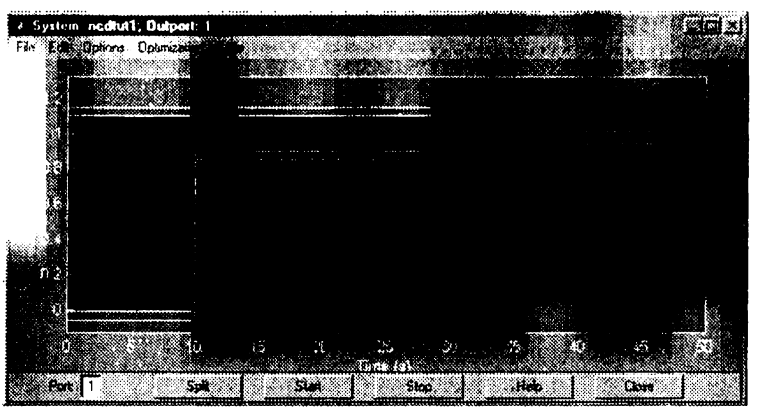


Рис. 4.7. Временные ограничения, соответствующие условиям задачи

Заметим, что точную установку линий ограничения можно провести, выбирая требуемые линии с помощью щелчка левой кнопкой мыши (выбранная линия изменяет цвет — становится белой) и используя затем команду меню Edit ▸ Edit constraint. В появляющемся окне редактора ограничений (Constraint Editor) в строке снизу необходимо задать начальную и конечную точки прямой в формате [x1 y1 x2 y2], нажав затем кнопку Done (рис. 4.8).

Точно такие же действия можно обеспечить, не используя пункты меню, а просто щелкнув на соответствующей линии правой кнопкой мыши.

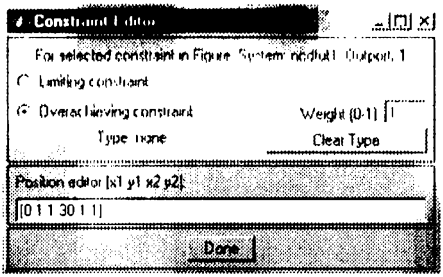


Рис. 4.8. Окно редактора ограничений

Следующий этап — указание переменных, подлежащих оптимизации. Выбор команды меню Optimization ▸ Parameters приведет к открытию диалогового окна задания настраиваемых параметров и интервала дискретизации — Optimization Parameters (рис. 4.9).

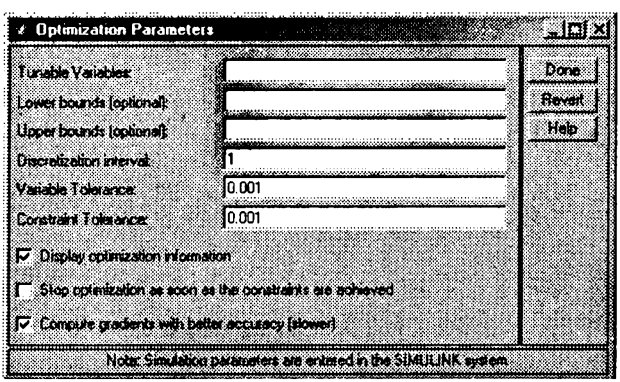


Рис. 4.9. Окно задания настраиваемых параметров и интервала дискретизации

В верхнем поле данного окна (Tunable Variables) введем имя настраиваемого параметра Kint (если таких параметров несколько, то их имена разделяются пробелами или запятыми), а величину интервала дискретизации (Discretization interval) установим равной 0.5 (ввод завершается нажатием кнопки Done).

Теперь все готово для решения задачи, и процесс поиска такого решения можно запустить нажатием кнопки Start в панели инструментов Simulink. Данный поиск иллюстрируется начальной и конечной формами переходного процесса, показанными на рис. 4.10, а также представленной ниже выходной информацией MATLAB.

```
Processing uncertainty information.
Uncertainty turned off.
Setting up call to optimization routine.
Done plotting the initial response.
Start time: 0  Stop time: 50.
There are 205 constraints to be met in each simulation.
There are 1 tunable variables.
There are 1 simulations per cost function call.
Creating a temporary SL model tp335930 for computing gradients...
Creating simulink model tp335930 for gradients...Done
f-COUNT     MAX{g}         STEP  Procedures
    3     0.184675           1
    6    0.0642839           1   Hessian modified twice
    9    0.0166645           1   Hessian modified twice
   12   -0.0055491           1   Hessian modified twice
   13  -0.00586138           1   Hessian modified twice
Optimization Converged Successfully
Active Constraints:
   124
   165
```

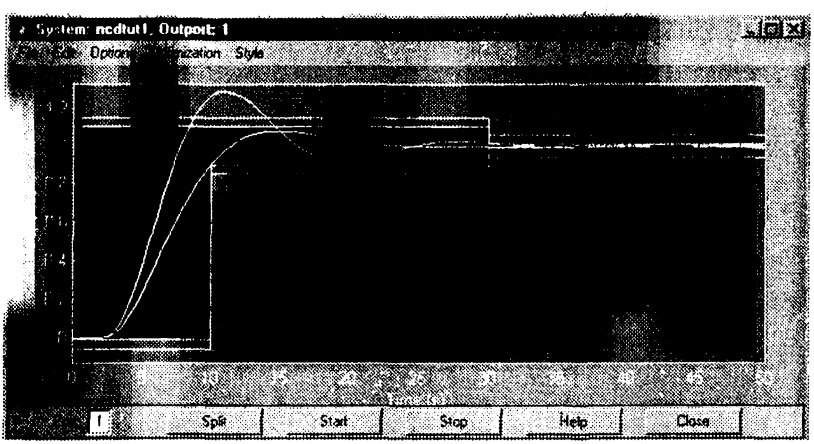


Рис. 4.10. Иллюстрация процесса оптимизации

Выведенные сообщения носят в основном служебный характер (главное из них говорит о том, что процесс поиска решения завершился успешно — `Optimization Converged Successfully`). Что же касается формы переходного процесса для оптимизированной системы, то из рис. 4.10 видно, что она полностью соответствует заданным ограничениям. Несмотря на довольно сложный характер переходной ха-

рактеристики, имеющей заметные колебания, она (в отличие от исходного варианта) прекрасно вписывается в заданные ограничительные рамки.

Найденную оптимальную величину параметра Kint можно узнать, набрав в командной строке имя данного параметра:

```
» Kint
Kint =
    0.1864
```

Задачу оптимизации можно усложнить, введя неопределенные параметры. К таким параметрам обычно относят какие-то параметры объекта регулирования, точные значения которых неизвестны или могут претерпевать изменения (например, вследствие изменения внешних условий).

В рамках рассматриваемого примера предположим, что коэффициент zeta может изменяться в пределах 5 % от своего номинального значения, а коэффициент w0 — в пределах от 0,7 до 1,45. Для задания подобной неопределенности воспользуемся командой меню Optimization ▸ Uncertainty окна NCD Output. Данный выбор приведет к открытию диалогового окна задания неопределенных переменных (Uncertain Variables). Введем указанные значения так, как изображено на рис. 4.11.

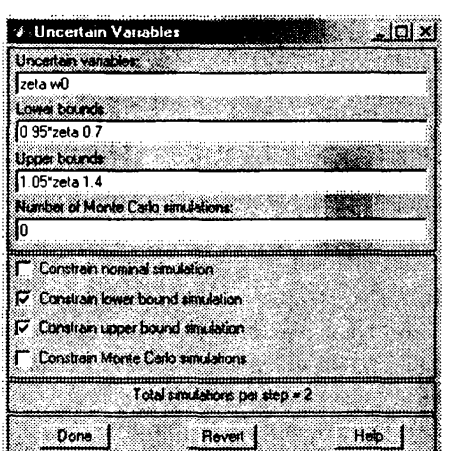


Рис. 4.11. *Окно задания неопределенных переменных*

Заметим, что по умолчанию используется номинальное значение параметра (Constrain nominal simulation). Для введения неопределен-

ности необходимо задать нижнюю (Constrain lower simulation) и/ил верхнюю (Constrain upper simulation) границы диапазона неопределен ности. Следующий флажок (Constrain Monte Carlo simulations, см. рис. 4.11 позволяет провести моделирование для нескольких случайных зна чений указанных параметров внутри отмеченной зоны (метод Мон те-Карло). Число таких значений задается в поле Number of Monte Carl simulations; не рекомендуется выбирать его очень большим ввиду воз можного значительного увеличения времени счета.

Завершив задание неопределенных переменных нажатием кнопк Done, можно продолжить оптимизацию системы в новых условиях Нажмем для этого кнопку Start в панели инструментов Simulinl и результат выполнения оптимизации отразится на временной диа грамме. Она примет вид, представленный на рис. 4.12.

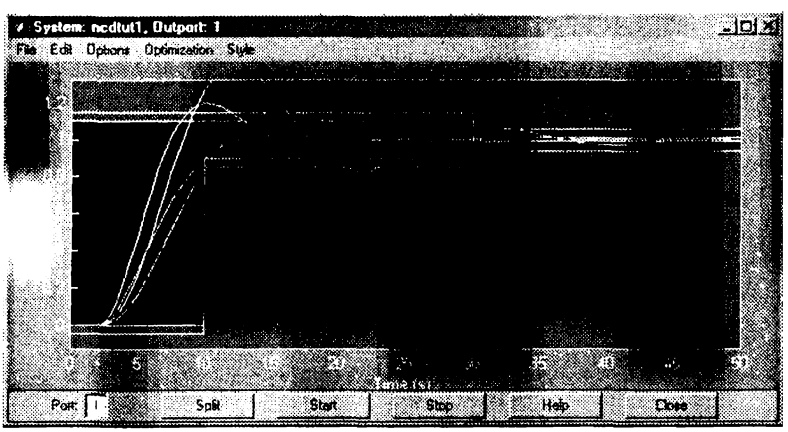


Рис. 4.12. Иллюстрация процесса оптимизации при наличии неопределенных параметров

На данном рисунке две светлых кривых соответствуют переходны процессам в системе при нижних и верхних граничных значения неопределенных параметров и начальном значении настраиваемог коэффициента Kint, две более темных кривых — переходным процес сам для тех же граничных значений, но уже при оптимальном Kin Кстати, теперь это оптимальное значение становится другим:

```
» Kint
Kint =
    0.1834
```

Оно отличается от найденного ранее значения, хотя и незначительно

Укажем, что удаление графиков процессов в окне блока NCD Output производится с помощью команды меню Edit | Delete plots (или одновременным нажатием клавиш Ctrl+X).

Меню окна блока NCD Output

Кратко остановимся на других командах меню окна блока NCD Output.

Меню File (Файл) содержит стандартные команды Load (Загрузить), Close (Закрыть), Save (Сохранить) и Print (Печатать). Действия, выполняемые при выборе любой из данных команд, относятся к графикам заданных временных ограничений, отображаемым в основном окне рассматриваемого блока.

Меню Edit (Правка) содержит уже рассмотренные команды Edit constraint и Delete plots, а также команду Undo (Отменить), производящую отмену последнего действия.

Меню Options (Настройка) содержит следующие команды:

- Initial response (Начальный отклик). Выбор данной команды приводит к выводу (в основном окне блока NCD Output) отклика исследуемой системы при начальных значениях ее параметров.
- Reference input (Задающий вход). Выбор данной команды приводит к открытию диалогового окна (рис. 4.13), в котором при желании — для вывода соответствующего графика — можно указать параметры входного задающего сигнала системы. Никакого влияния на процессы моделирования и оптимизации данные параметры не оказывают (на рис. 4.13 показаны установки по умолчанию).

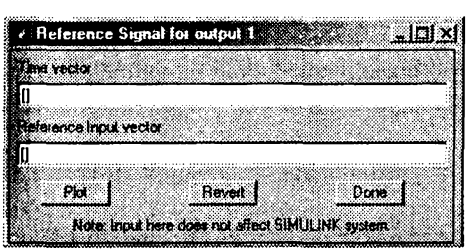


Рис. 4.13. Окно ввода параметров задающего воздействия

- Step response (Характеристики переходного процесса). Выбор этой команды приводит к открытию диалогового окна, показанного на рис. 4.14, в котором задаются параметры переходного процесса, такие как его длительность (Settling time), время нарастания (Rise time), максимальное перерегулирование (Percent overshoot), макси-

мальное «недорегулирование» (Percent undershoot), соответствующие уровни определения данных характеристик (Percent settling и Percent rise, в процентах), а также начальное и конечное время моделирования (Step time и Final time), желаемые начальное и конечное значения выхода (Initial output и Final output). Все эти параметры пользователь может изменять.

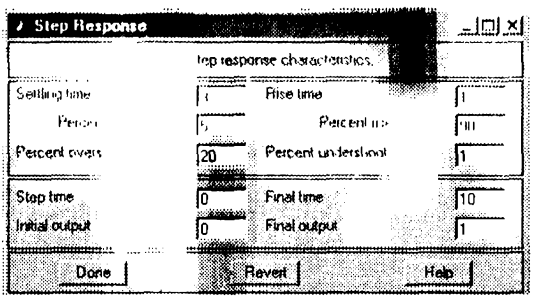


Рис. 4.14. Окно задания характеристик переходного процесса

- Time range (Временной диапазон). Выбор данной команды приводит к открытию диалогового окна, показанного на рис. 4.15, в котором можно задать или изменить диапазон времени моделирования и метку оси времени, то есть параметры оси абсцисс.

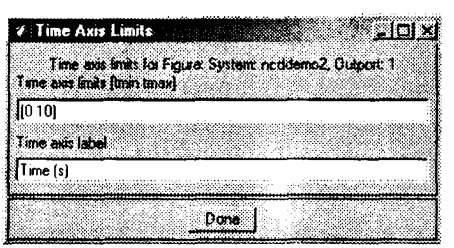


Рис. 4.15. Окно задания временного диапазона графика

- Y-Axis (Ось Y). То же, что и для предыдущей команды, но по отношению к оси ординат (рис. 4.16).
- Refresh (Обновить) — перерисовать все временные ограничения.

Меню Optimization (Оптимизация) содержит рассмотренные ранее команды Parameters и Uncertainty, а также команду Start (Старт), выбор которой запускает процесс моделирования и оптимизации системы

(это аналогично нажатию кнопки Start в окне Simulink или кнопки Start в панели кнопок в нижней части окна блока NCD Output). Команда Stop (Стоп) останавливает процесс моделирования (аналогичное действие выполняется при нажатии кнопки Stop в панели кнопок блока NCD Output).

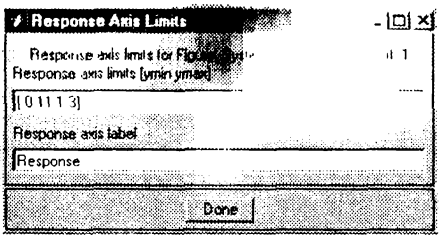


Рис. 4.16. Окно задания диапазона значений оси ординат

Наконец, последним является меню Style (Стиль). Здесь имеются следующие команды:

- Grid (Сетка) — установка сетки на графике заданных ограничений;
- Snap (Привязка) — при установке данного режима линии временных ограничений можно проводить не под любым углом к оси абсцисс, а только под углом, кратным 22,5°;
- Hot-key help (Горячие клавиши) — вывод информации о «горячих» клавишах и их комбинациях;
- Readme.m (Файл справки) — вывод файла справки об окне блока NCD Output.

Из пяти кнопок панели, расположенной в нижней части окна рассматриваемого блока, четыре (Start, Stop, Help и Close) в дополнительных пояснениях не нуждаются. Что же касается пятой кнопки Split (Расщепить), то ее нажатие, при предварительном выборе с помощью левой кнопки мыши какой-либо ограничивающей линии, приводит к «расщеплению» этой линии на две одинаковые по длине половинки с возможностью последующего редактирования отдельно каждой из них.

Настройка параметров ПИД-регулятора

Другой пример использования блоков пакета NCD Blockset содержится в файле ncddemo1. Запустив этот файл, в окне Simulink мы увидим модель системы (рис. 4.17).

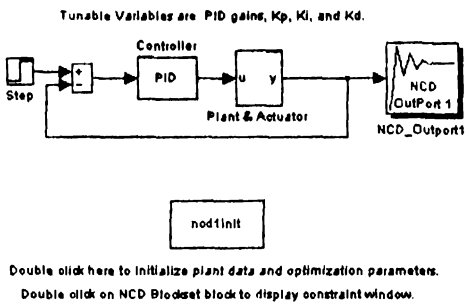


Рис. 4.17. Модель оптимизируемой системы

Основными элементами замкнутой системы являются:

- объект регулирования (блок Plant & Actuator);
- ПИД-регулятор (Controller);
- цепь обратной связи и узел сравнения.

Кроме них в модель входят блок задающего воздействия (в виде единичного скачка) Step и блок NCD Output (NCD OutPort 1).

Дважды щелкнув на блоке Plant & Actuator, раскроем его (рис. 4.18).

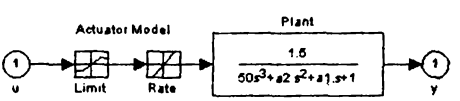


Рис. 4.18. Структура объекта регулирования

Как видно из рис. 4.18, в состав объекта регулирования входят нелинейности типа «Ограничение» с уровнями ограничения –2, 2 (блок Limit), динамического ограничения коэффициента усиления (блок Rate), осуществляющего ограничение величины

$$\frac{u(t_i)-y(t_{i-1})}{t_i-t_{i-1}}$$

диапазоном [-0.8, 0.8], а также линейное динамическое звено с передаточной функцией

$$W(p)=\frac{1.5}{50p^3+a2\cdot p^2+a1\cdot p+1},$$

где коэффициент $a2$ может принимать значения в диапазоне [40, 50] с номинальным значением $a2 = 43$, а коэффициент $a1$ — в диапазоне [0.5, 2] относительно номинального значения $a1 = 3$.

Постановка задачи оптимизации в данном случае такова: при заданных структуре объекта управления и неопределенностях его параметров необходимо найти значения коэффициентов *Kp*, *Ki* и *Kd* регулятора, при которых в представленной замкнутой структуре переходный процесс будет иметь:

- максимальное перерегулирование — не более 20 %;
- время нарастания — не более 10 с;
- длительность переходного процесса — не более 30 с.

Дважды щелкнем на блоке инициализации модели ncd1init, а затем раскроем блок NCD OutPort 1. Открытие далее окон задания настраиваемых параметров и неопределенных переменных (рис. 4.19 и 4.20) дает возможность увидеть соответствующие величины, относящиеся к сформулированной задаче.

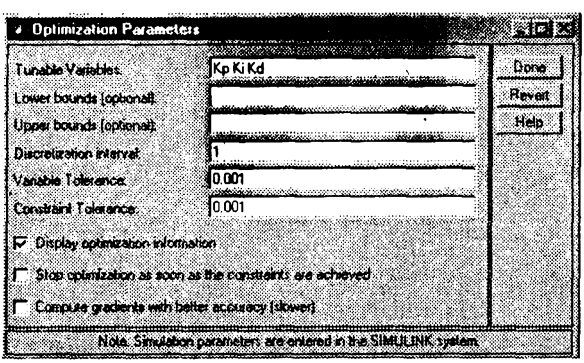


Рис. 4.19. *Окно задания настраиваемых параметров*

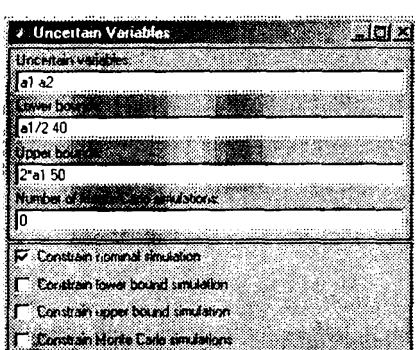


Рис. 4.20. Окно задания неопределенных переменных

Отметим, что моделирование в данном случае будет проводиться для номинальных значений коэффициентов a1 и a2 и при начальных значениях настраиваемых параметров Kp = 0,63; Ki = 0,0504; Kd = = 1,9688. Такие значения выбраны в соответствии с методикой настройки ПИД-регуляторов Зиглера–Николса (Ziegler–Nichols method), согласно которой:

1. Коэффициенты Ki и Kd устанавливаются равными нулю, а коэффициент Kp увеличивается до тех пор, пока система не потеряет устойчивость.
2. Предельное значение Kp обозначается как Ku, а период автоколебаний — как Pu.
3. Задаются следующие значения коэффициентов регулятора:

   $Kp = 3 \cdot Ku/5$, $Ki = 6 \cdot Ku/(5 \cdot Pu)$, $Kd = 3 \cdot Ku \cdot Pu/40$.

Запустим процесс оптимизации, нажав кнопку Start. Полученный результат показан на рис. 4.21.

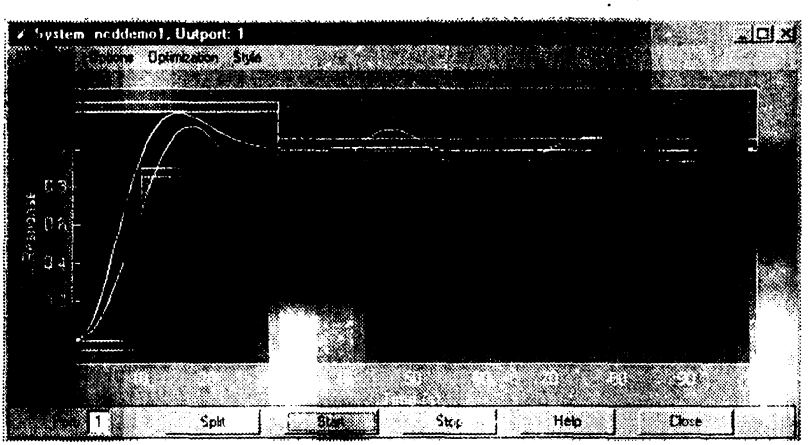


Рис. 4.21. Иллюстрация процесса оптимизации

Найденные оптимальные значения при этом равны:

```
» Kp
Kp =
    1.3353

» Ki
Ki =
    0.1547
```

```
» Kd
Kd =
    8.3280
```

Как видим, они существенно отличаются от начальных.

Проведем теперь оптимизацию, учитывая интервальную неопределенность коэффициентов $a1$ и $a2$, то есть установив сначала в окне задания неопределенных переменных флажки Constrain lower simulation и Constrain upper simulation, а затем повторно запустив процесс моделирования. Результат для этого случая отражен на рис. 4.22; ниже приведены также значения параметров регулятора (светлые линии на рисунке соответствуют исходной системе при граничных значениях неопределенных параметров, темные — оптимизированной системе).

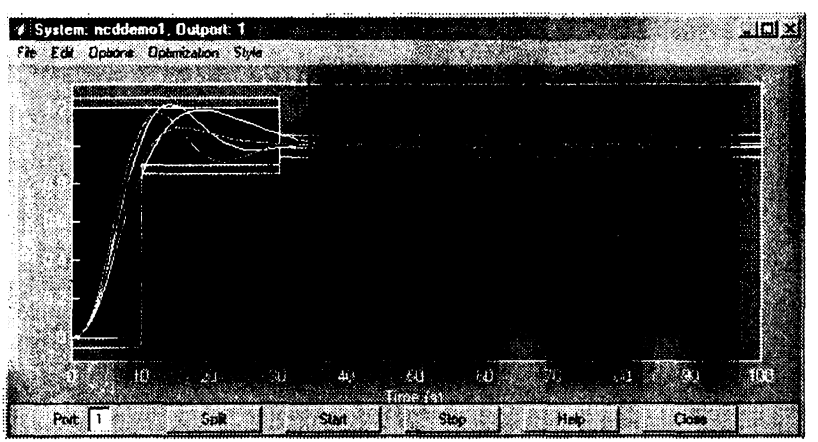


Рис. 4.22. Иллюстрация процесса оптимизации при неопределенных параметрах a1 и a2

```
» Kp
Kp =
    1.6792

» Ki
Ki =
    0.1285

» Kd
Kd =
    8.2518
```

Как видно, по сравнению с предыдущим случаем здесь имеются определенные отличия.

Процесс исследования системы можно продолжить, вводя, например, дополнительные ограничения или изменяя имеющиеся и т. п. Оставляем это увлекательное занятие заинтересованному читателю.

Настройка параметров комплексного регулятора

Следующим примером (отраженным в файле ncddemo2) является пример настройки параметров регулятора сложной структуры.

Запустив данный файл, в окне Simulink мы увидим модель системы (рис. 4.23).

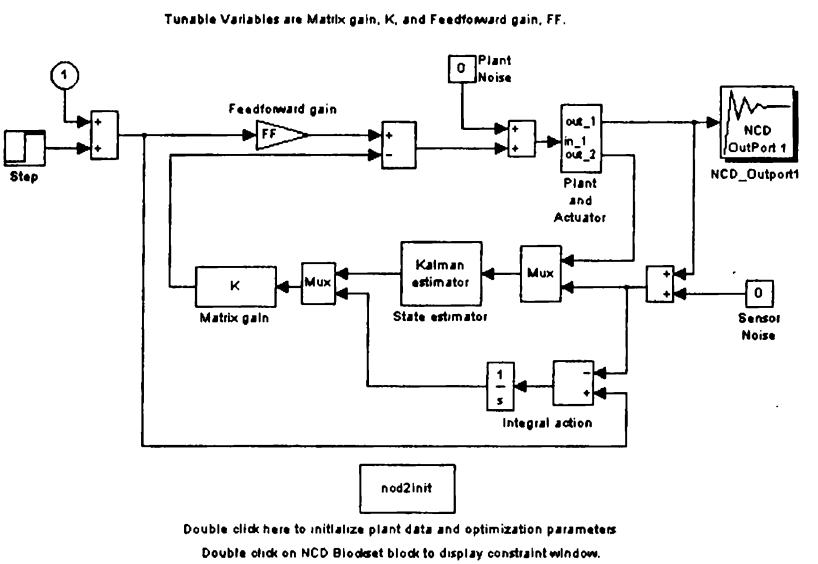


Рис. 4.23. Модель системы регулирования

Объект управления (Plant and Actuator) (рис. 4.24) здесь представляет собой последовательно соединенные нелинейности типов «Ограничение» (Saturation) «Динамическое ограничение коэффициента усиления» (Rate Limiter, см. выше) и линейное динамическое звено (Plant), описание которого задано через переменные состояния.

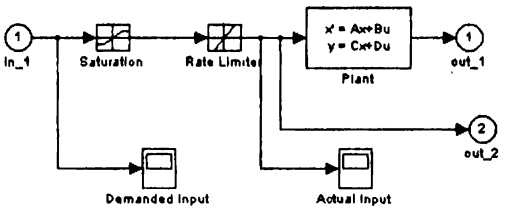


Рис. 4.24. Структура объекта управления

Характеристики звеньев: уровни ограничения −5 и +5, динамическое ограничение определяется диапазоном [−10, 10], описание линейного динамического звена имеет вид

$$\dot{\mathbf{x}} = \begin{bmatrix} -1.0285 & 0.9853 & -0.9413 & 0.0927 \\ -1.2903 & -1.0957 & 2.8689 & 4.7950 \\ 0.1871 & -3.8184 & -2.0788 & -0.9781 \\ 0.4069 & -4.1636 & 2.5407 & -1.4236 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 6.6389 \\ 0 \\ 0 \end{bmatrix} \mathbf{u} = \mathbf{Ax} + \mathbf{Bu},$$

$$\mathbf{y} = [-1.7786 \quad 1.1390 \quad 0 \quad -1.0294]\mathbf{x} = \mathbf{Cx}.$$

Предполагается, что все элементы матрицы **A** могут изменяться в диапазоне [0.5, 2] от своих номинальных значений.

Регулятор в данном случае имеет сложную структуру: в его состав входят как обычный И-регулятор (блоки Integral action), так и наблюдатель в виде фильтра Калмана (Kalman estimator), а также многомерное пропорциональное звено (Matrix gain) с матричным коэффициентом усиления **K**. Для повышения быстродействия в систему введена дополнительная прямая связь от задающего воздействия (пропорциональное звено Feedforward gain с коэффициентом усиления *FF*). Действие внешних возмущений отражено в модели источниками шумовых сигналов Plant Noise и Sensor Noise (для упрощения задачи сигналы данных источников приняты равными нулю).

Синтез системы сводится к нахождению оптимальных значений коэффициентов **K** и *FF* таких, при которых:

- перерегулирование не превышает 20 %;
- время установления — не более 1 с;
- длительность переходного процесса — не более 3 с.

Рисунок 4.25 иллюстрируют установку перечисленных временных ограничений. Он дает шаблон для будущих построений.

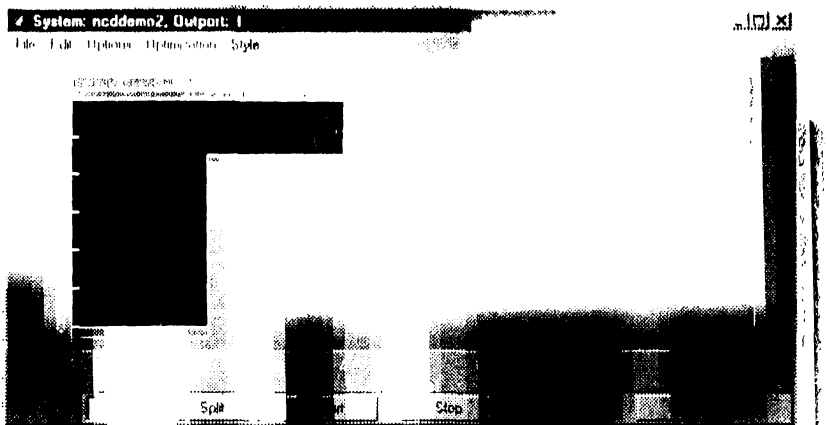


Рис. 4.25. Окно блока NCD OutPort 1 с требуемыми временными ограничениями

Следующий этап — установку настраиваемых параметров — иллюстрирует рис. 4.26. Параметры устанавливаются в окне Optimization Parameters (Параметры оптимизации). Обратите внимание, что верхний и нижний пределы могут не задаваться.

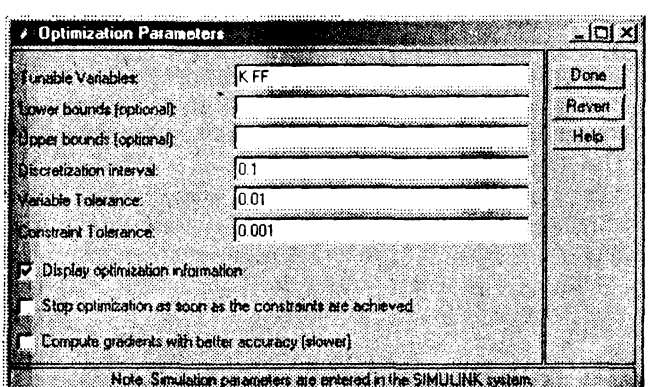


Рис. 4.26. Окно задания настраиваемых параметров

Наконец, на рис. 4.27 показана установка параметров неопределенных переменных. Она выполняется в окне Uncertain Variables. В нашем случае неопределенной переменной является матрица системы A, обозначенная на рис. 4.27 как a.

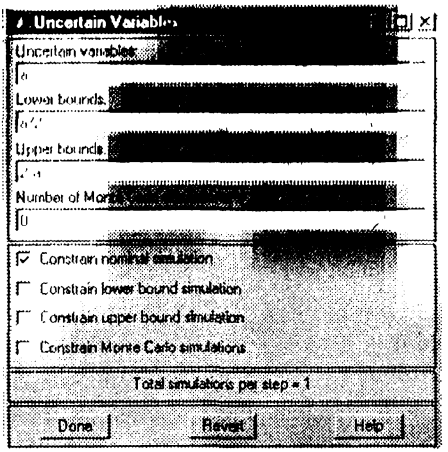


Рис. 4.27. Окно задания неопределенных переменных

Если инициализировать модель (см. рис. 4.23) двойным щелчком мыши на блоке ncd2init, а затем перейти в окно блока NCD OutPort 1 и нажать в нем кнопку Start, то запустится процесс моделирования и оптимизации системы, результаты которого показаны на рис. 4.28.

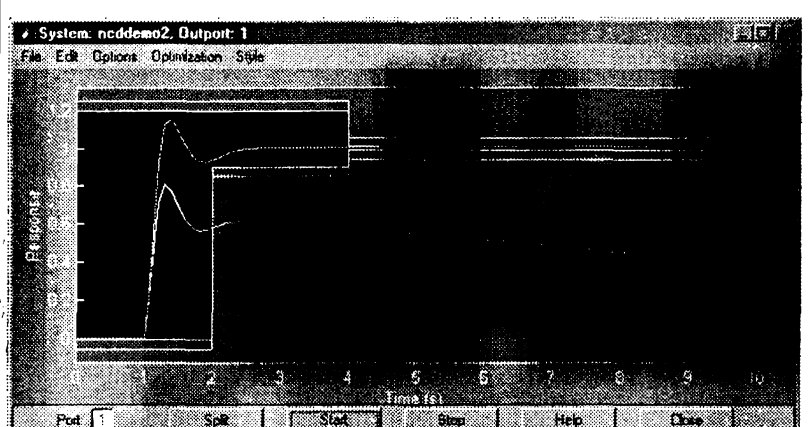


Рис. 4.28. Иллюстрация процесса оптимизации

Как видно, при исходных (начальных) значениях параметров системы вид ее переходного процесса был весьма далек от желаемого, но итоговый переходный процесс укладывается в заданные временные

ограничения. Найденные оптимальные значения параметров, как и раньше, можно просмотреть в режиме командной строки MATLAB:

```
» K
K =
   -1.0585    0.9378    0.0188   -0.2413    0.0786

» FF

FF =
    1.2729
```

Заметим, что при установленных параметрах (см. рис. 4.27) найденное решение соответствует номинальным значениям элементов матрицы A. Для исследования поведения системы с учетом неопределенности данной матрицы необходимо выполнить действия, описанные в предыдущем примере.

Настройка параметров ПИ-регулятора для многомерного объекта

Наконец, еще одним примером (отраженным в файле ncddemo3) является пример настройки параметров пропорционально-интегрального регулятора (ПИ-регулятора) для многомерного объекта. В качестве такого объекта выбран газотурбинный двигатель. Его математическое описание предоставлено в стандартной форме для переменных состояния, при этом предполагается, что в общем случае (как и в предыдущем примере), элементы матрицы A модели объекта могут изменяться от половины до двух своих номинальных значений. Объект имеет два входных воздействия, пять переменных состояния и два выхода. Структура системы представлена на рис. 4.29.

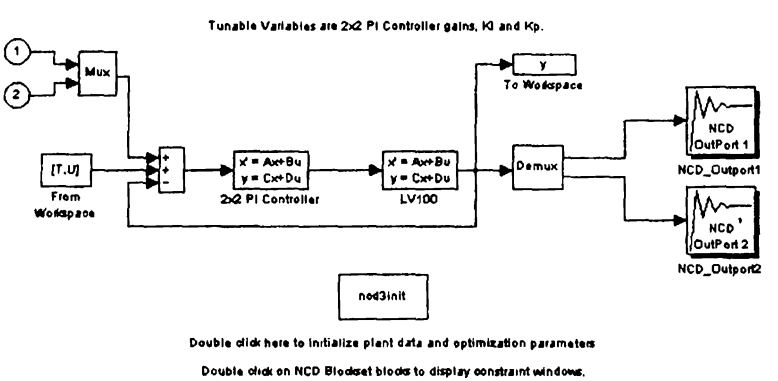


Рис. 4.29. Системы с многомерным объектом регулирования

В данном случае настраиваемыми параметрами являются матрицы коэффициентов **Ki** и **Kp** двухканального ПИ-регулятора (блок **2×2 PI Controller**). Условия настройки:

- время нарастания — не более 1 с;
- нулевое перерегулирование в первом канале и не более 10 % — во втором канале;
- длительность переходного процесса — не более 3 с.

Задающие воздействия имеют вид прямоугольных импульсов.

Здесь контролируются оба выхода объекта и поэтому используются два блока **NCD Output**. Задаваемые в них ограничения представлены, соответственно, на рис. 4.30 и 4.31.

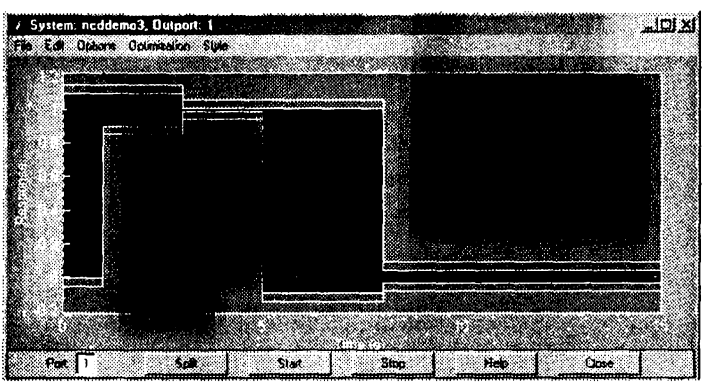


Рис. 4.30. Временные ограничения первого канала

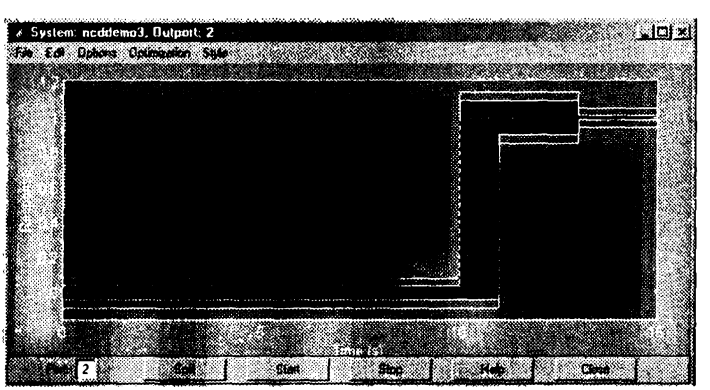


Рис. 4.31. Временные ограничения второго канала

Для обоих блоков NCD Output указаны одни и те же настраиваемые и неопределенные параметры (рис. 4.32 и 4.33).

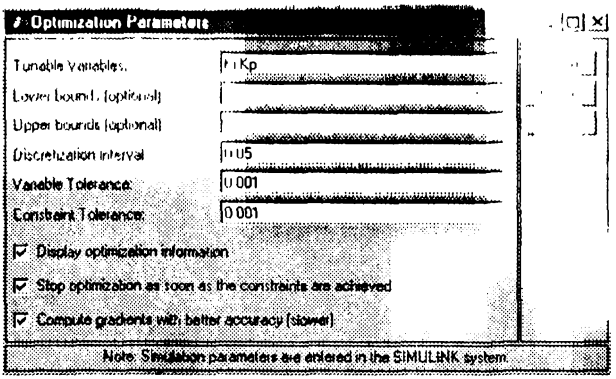


Рис. 4.32. Окно задания настраиваемых параметров системы

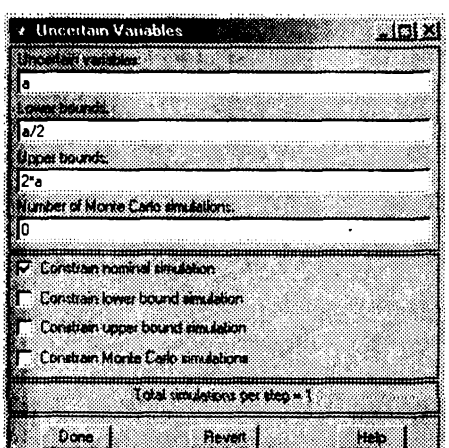


Рис. 4.33. Окно неопределенных параметров (переменных)

Инициализация системы (двойным щелчком на блоке ncd3init), а затем запуск процесса моделирования и оптимизации приводит к требуемому результату (кстати сказать, в данном примере поиск оптимальных значений параметров проходит существенно дольше, чем в предыдущих примерах — из-за большей размерности и сложности задачи). Полученные результаты, например, для первого канала

(рис. 4.34), следует признать очень хорошими (здесь оптимизация проведена для номинальных значений элементов матрицы A).

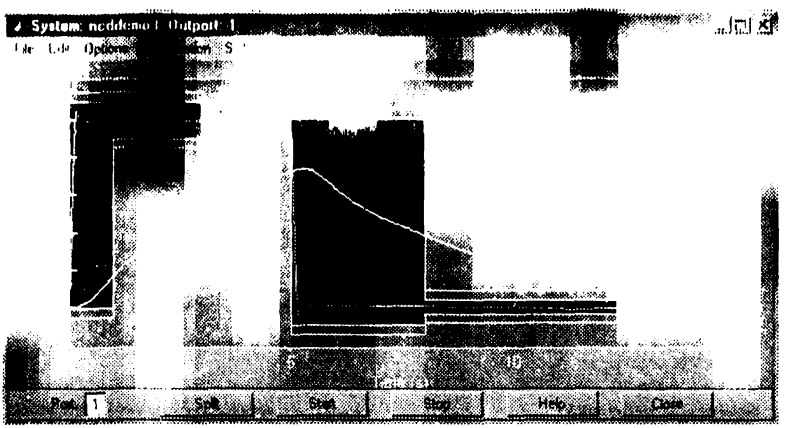


Рис. 4.34. Начальная и конечная формы отклика первого канала

Найденные оптимальные значения таковы:

```
» Ki
Ki =
    9.6062    0.2193
   32.1497  -44.6145

» Kp
Kp =
    1.5119    6.3274
   87.3971  -35.3085
```

Отметим, что в файлах ncddemo4, ncdtut2, ncdtut2old содержатся еще три примера, иллюстрирующие работу и применение блоков пакета NCD Blockset. Дополнительную информацию можно получить, используя команду help NCD.

Особенности решаемых задач

При решении различных оптимизационных задач с помощью пакета NCD Blockset следует иметь в виду следующие особенности этого пакета:

- пакет может использоваться для решения задач оптимизации переходного процесса, идентификации, настройки параметров сис-

темы с использованием квадратичного критерия качества — для случаев, когда все сигналы в исследуемой системе на этапе ее оптимизации являются детерминированными, а шумы наблюдений и измерений отсутствуют. При необходимости рекомендуется провести исследование оптимизированной системы с добавлением указанных шумов;

- оптимизация следящих систем проводится, вообще говоря, не в режиме их нормального функционирования (то есть при отслеживании произвольного входного воздействия), а при входном сигнале типа единичного скачка;
- могут рассматриваться задачи оптимизации многомерных объектов с одновременным заданием временных ограничений на ряд сигналов системы (что требует использования нескольких блоков NCD Output);
- проблема «повторяющихся параметров» (пример этого показан на рис. 4.35, где параметр K присутствует дважды — в числителе и знаменателе передаточной функции регулятора) в рассматриваемом пакете решается однократным заданием повторяющегося параметра среди других оптимизируемых параметров;

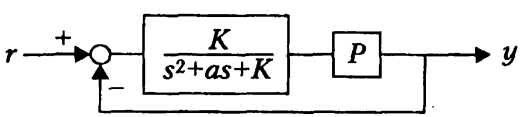


Рис. 4.35. Пример системы, где параметр K повторяется дважды

- задача оптимизации регуляторов одинаковой структуры для двух подсистем сводится к рассмотренной проблеме повторяющихся параметров;
- пакет может также использоваться для оптимизации размещения нулей и полюсов передаточной функции на комплексной плоскости. Для этого надо просто использовать блоки Simulink Zero-pole или Transfer Fcn библиотеки Continuous и объявить (с помощью диалогового окна настраиваемых параметров) требуемые нули и полюса настраиваемыми, с заданием, разумеется, их предельных (наибольших и наименьших) значений. В случае комплексных полюсов их задание возможно двумя способами: в форме `(s+a+bj)(s+a-bj)` или в форме `(s^2+as+b)`. В первом случае мы имеем дело с повторяющимся параметром, во втором — нет. Но в первом случае задание ограничений представляется более простым.

Функции и команды NCD Blockset

В рассматриваемый пакет входит следующий набор недокументированных функций и команд, выполняемых в режиме командной строки MATLAB.

Функции управления диалоговыми окнами:

- `coneddlg` — вызов окна редактора ограничений;
- `paramdlg` — вызов окна задания настраиваемых параметров;
- `rangedlg` — вызов окна задания диапазонов осей координат;
- `refdlg` — вызов окна задания параметров опорного сигнала;
- `stepdlg` — вызов окна задания параметров переходного процесса системы;
- `uncerdlg` — вызов окна задания неопределенных переменных.

Функции данной группы, выполняемые при значении аргумента, равном нулю (например, `stepdlg(0)`), дублируют соответствующие команды меню окна блока `NCD Output`. Функции выполняются только при открытом данном окне.

Функции главного интерфейса NCD Blockset:

- `ncdblock` — вызов блоков NCD;
- `slblocks` — определение библиотеки блоков NCD Blockset;
- `optblock` — открытие блока NCD Output;
- `optfig` — создание временных ограничений.

Например, выполнение функции `ncdblock` приведет к появлению окна Simulink с блоками NCD (рис. 4.36).

Функции оптимизации:

- `initresp` — возвращает график временной характеристики при начальных значениях параметров модели;
- `costfun` — вычисляет функцию потерь;
- `gradfun` — вычисляет градиент функции потерь;
- `nlinopt` — запускает алгоритм оптимизации;
- `copymdl` — создает дополнительную модель, используемую для нахождения градиента;
- `tvarset` — S-функция для задания настраиваемых параметров.

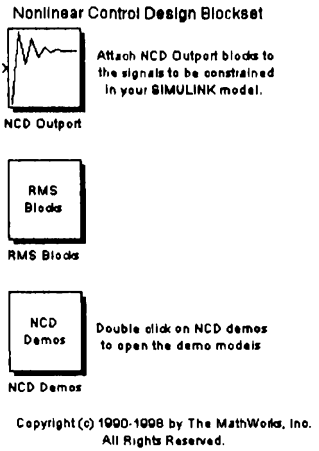


Рис. 4.36. Окно Simulink, появляющееся при выполнении функции ncdblock

Первая из указанных функций дублирует соответствующую команду меню главного окна блока NCD Output, остальные используются в процессе оптимизации и, по сути, являются вызываемыми подпрограммами данного блока.

Функции вызова демонстрационных примеров:

- ncddemo — библиотека демонстрационных примеров;
- ncddemo1 — оптимизация ПИД-регулятора;
- ncddemo2 — оптимизация регулятора сложной структуры;
- ncddemo3 — оптимизация ПИ-регулятора для многомерной системы;
- ncddemo4 — оптимизация системы управления перевернутым маятником;
- rmsdemo — демонстрация блоков CRMS и DRMS.

Функции обучения:

- ncdtut1 — пример проектирования системы управления;
- ncdtut2 — пример идентификации системы;
- ncdtut2old — вариант примера идентификации системы.

Файл ncdtut2old.mat содержит данные для ncdtut2old.

Утилиты для функций демонстрации и обучения:

- ncd1init — задание ограничений и начальных условий для ncddemo1;

- ncd2init — задание ограничений и начальных условий для ncddemo2;
- ncd3init — задание ограничений и начальных условий для ncddemo3;
- ncd4init — задание ограничений и начальных условий для ncddemo4;
- penddata — задание ограничений и начальных условий для ncdtut2.

Файлы описаний (текстовые файлы с расширением .hlp):

- hotkey — информация об используемых комбинациях клавиш;
- mainncd — общее описание блоков NCD;
- paramdlg — описание окна задания настраиваемых параметров;
- readncd — то же, что файл Readme.m;
- stepdlg — описание диалогового окна для задания параметров переходного процесса системы;
- uncerdlg — описание диалогового окна задания неопределенных переменных.

В пакет входит также набор утилит оптимизации и графического интерфейса пользователя (GUI), используемых в качестве подпрограмм основной программы блока NCD Output.

Более подробную информацию о каждой из перечисленных функций можно получить, вызвав в режиме командной строки MATLAB команду help имя_функции.

Глава 5
Пакет моделирования энергетических систем Power System Blockset

Назначение пакета и доступ к нему

Назначение пакета Power System Blockset

Пакет Power System Blockset служит для моделирования энергетических (силовых) систем и устройств — от простых электрических цепей постоянного и переменного тока до сложных линий электропередачи большой мощности, преобразовательных устройств на современной элементной базе и электрических машин с системами управления. Другими словами, энергетические системы и устройства характерны объединением электрических цепей с электромеханическими устройствами и различными преобразователями электрической энергии.

Для работы пакета необходимы базовая система MATLAB и пакет моделирования Simulink.

Функционально полный набор библиотек и компонентов, превосходный графический интерфейс пользователя системы Simulink, обширные вычислительные возможности базовой матричной системы MATLAB, высокая степень достоверности моделирования энергетических устройств и систем и превосходное качество визуализации результатов моделирования — все это сделало Power System Blockset лучшим среди пакетов такого рода, доступных для установки на персональных компьютерах среднего класса с операционной системой Windows 95/98.

При построении моделей энергетических устройств используется метод «Click and Drag» («щелкни и тяни»), позволяющий легко создавать графические представления моделей в виде привычных для энергетиков блок-схем. Они создаются в окнах моделей пакета Simulink и затем запускаются на исполнение.

Доступ к библиотекам пакета Power System Blockset

Возможности пакета Power System Blockset прежде всего определяются компонентами входящих в него библиотек. Доступ к ним обычно осуществляется из среды Simulink. При запуске последнего появляется окно броузера библиотек Simulink, показанное на рис. 5.1 справа. В нем можно выбрать библиотеку Power System Blockset. С помощью контекстного меню правой кнопки мыши можно открыть основное окно с разделами библиотеки компонентов Power System Blockset.

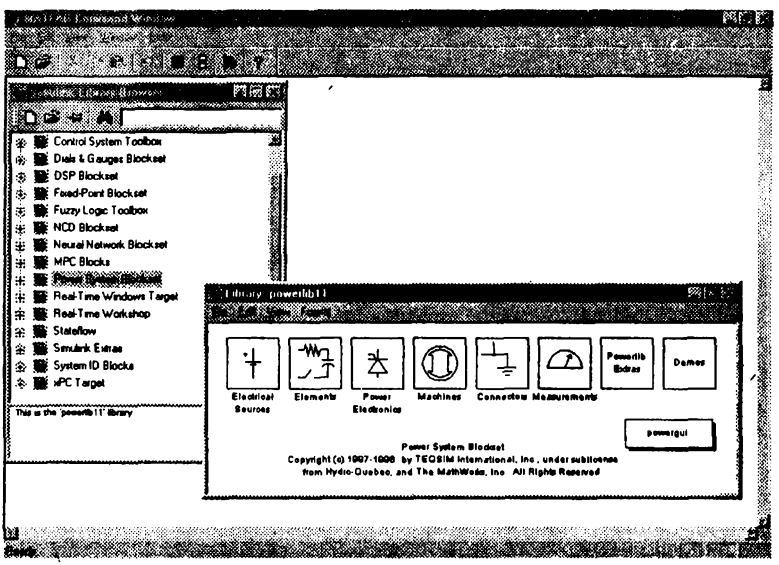


Рис. 5.1. Доступ к библиотекам пакета Power System Blockset

Окно с разделами библиотеки компонентов можно также вызвать, исполнив в окне системы MATLAB команду `powerlib`. Библиотеки пакета Power System Blockset содержат достаточно представительные наборы компонентов, на основе которых строится графическая модель энергетической системы. После этого запускается сеанс имитации ее работы обычными средствами совместно работающих пакетов «MATLAB+Simulink».

Состав библиотек Power System Blockset

Как видно из рис. 5.1, в состав Power System Blockset входят библиотеки следующего назначения:

- Electrical Sources — источники электрической энергии и сигналов;
- Elements — линейные и нелинейные компоненты электрических и электронных устройств;
- Power Electronics — устройства энергетической электроники;
- Machines — электрические машины;
- Connectors — подключающие устройства;
- Measurements — измерительные и контрольные устройства;
- Powerlib Extras — специальные энергетические устройства;
- Demos — примеры моделирования энергетических устройств.

Их применение позволяет создавать модели самых различных энергетических установок и выполнять их моделирование в режиме работы виртуальных устройств. Это дает наглядное представление о работе реальных энергетических систем.

Правила подготовки моделей

Правила подготовки моделей при использовании пакета Power System Blockset ничем не отличаются от тех, которые применяются в пакете Simulink. Поэтому лишь кратко напомним их. Вначале готовится модель энергетической системы или устройства. Для этого в окно модели Simulink переносятся модели необходимых компонентов с использованием метода Click and Drag. После этого (или по мере ввода моделей компонентов) производится соединение компонентов с помощью соединительных элементов, входящих в пакеты Simulink и Power System Blockset. В необходимых случаях к моделям компонентов подключаются виртуальные измерительные и регистрирующие приборы, например осциллографы.

При необходимости с помощью окон задания параметров задаются параметры моделей компонентов. С помощью команды Simulation Parameters можно задать также параметры моделирования. Обычно при первом запуске сеанса моделирования используются параметры, задаваемые по умолчанию. Однако, особенно для параметров компонентов, это может быть недопустимо. Вызов окна установки параметров моделей компонентов удобнее всего осуществлять с помощью мыши, установив указатель на модель компонента и дважды щелкнув левой кнопкой мыши. Если просто задержать указатель мыши на значке компонента, то появится всплывающая подставка на желтом фоне с указанием установленных параметров.

По завершении подготовки модели она запускается на исполнение командой Run в меню Simulation окна модели Simulink. Можно также

запускать модель на исполнение с помощью кнопки с треугольником в панели инструментов окна модели Simulink. Для остановки процесса моделирования можно использовать кнопку с изображением прямоугольника.

Исполнение модели сопровождается подготовкой отдельных окон регистрирующих и измерительных приборов. Иногда они не видны, поскольку оказываются закрытыми другими окнами. Можно вызвать окна регистрирующих приборов на просмотр, используя имеющиеся в Windows средства переключения окон — например, совместно нажимая клавиши Alt+Tab.

Параметры и единицы их измерения

При указании параметров моделей компонентов и физических единиц для их измерения используются следующие обозначения (в скобках даны обозначения, принятые в отечественной литературе):

Параметр	Единица измерения	Сокращенное обозначение
Time (время)	second	s (с)
Length (длина)	meter	m (м)
Mass (масса)	kilogram	kg (кг)
Energy (энергия)	joule	J (Дж)
Current (ток)	ampere	A (А)
Voltage (напряжение)	volt	V (В)
Frequency (частота)	Hertz	Hz (Гц)
Active power (активная мощность)	watt	W (Вт)
Apparent power (полная мощность)	volt ampere	VA (ВА)
Reactive power (реактивная мощность)	volt-ampere reactive	var (вар)
Impedance (импеданс)	ohm	Ω (Ом)
Resistance (сопротивление)	ohm	Ω (Ом)
Inductance (индуктивность)	henry	H (Гн)
Capacitance (емкость)	farad	F (Ф)
Flux linkage (поток сцепления)	volt second	V·S (В·с)
Rotation speed (скорость вращения)	radians per second	rad/s (рад/с)

Параметр	Единица измерения	Сокращенное обозначение
Rotation speed (скорость вращения)	revolutions per minute	rpm (об./мин)
Torque (вращающий момент)	newton meter	N·m (Н·м)
Inertia (инерция)	kilogram (meter)^2	kg·m^2 (кг·м^2)
Friction factor (коэффициент трения)	newton meter second	N·m·s (Н·м·с)

В этой таблице приведены лишь основные параметры и единицы их измерения. Некоторые из параметров будут рассмотрены по мере описания моделей.

Документация и литература по пакету

Документация

Электронная документация по пакету имеется в справочной системе в виде файлов форматов HTML и PDF. Объем документации в формате PDF составляет 294 с.

Рекомендуемая литература

При работе с пакетом Power System Blockset может быть полезна следующая литература:

1. Дьяконов В. П. MATLAB. Учебный курс. — С.Пб.: Питер, 2001.
2. Бачурин В. В., Дьяконов В. П. и др. Схемотехника устройств на мощных полевых транзисторах. — М.: Радио и связь, 1994.
3. Дьяконов В. П., Абраменкова И. В. MATLAB 5. Система символьной математики. — М.: Нолидж, 1999.
4. Филиппов Е. Нелинейная электротехника. — М.: Энергия. 1976.
5. Влах И., Сингхал К. Машинные методы анализа и проектирования электронных цепей. — М.: Радио и связь, 1988.
6. Келехсаев Б. Г. Нелинейные преобразователи и их применение. М.: Солон-Р, 1999.
7. Mohan N., Power Electronic, Converters, Applications and Design, John Wiley & Sons, Inc., New York, 1995.
8. Krause P. C., Wasynczuk O., Sudhoff S. D., Analysis of Electric Machinery, IEEE Press, 1995.

9. Rajagopalan V., Computer-Aided Analysis of Power Electronic Systems, Marcel Dekker, Inc., New York,1987.

Библиотека источников электрической энергии Electrical Sources

Состав библиотеки Electrical Sources

Источники электрической энергии являются первичными компонентами энергетических систем и устройств. Большинство электротехнических устройств является потребителями энергии, вырабатываемой этими источниками, либо ее преобразователями. Особенно часто используются преобразователи одного вида энергии в другой. Например, широко применяются преобразователи переменного тока в постоянный (выпрямители), постоянного тока в переменный (инверторы), регуляторы и стабилизаторы напряжения и тока и др.

В связи с этим пакет Power System Blockset имеет модели источников, позволяющие имитировать реальные источники электроэнергии. Дважды щелкнув на значке библиотеки Electrical Sources, можно открыть окно этой библиотеки. Оно представлено на рис. 5.2.

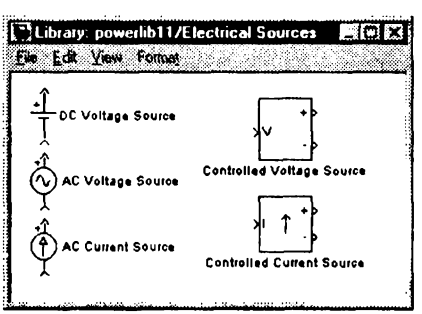


Рис. 5.2. Окно библиотеки источников Electrical Sources

В нем представлено пять типов источников электрической энергии:

- AC Current Source — источник переменного тока;
- AC Voltage Source — источник переменного напряжения;
- DC Voltage Source — источник постоянного напряжения
- Controlled Current Source – регулируемый источник тока;
- Controlled Voltage Source – регулируемый источник напряжения.

Эти компоненты образуют функционально полный набор источников электрической энергии и минимальный набор источников сигналов (в других расширениях MATLAB кроме источников синусоидальных сигналов можно задавать великое множество иных источников сигналов самой различной формы). Этот набор вполне достаточен для проектирования энергетических устройств. Ниже представлено полное описание всех компонентов библиотек с простыми примерами их применения. С этих примеров рекомендуется начать освоение пакета Power System Blockset.

Источник переменного тока

Идеальный (с бесконечно большим внутренним сопротивлением) источник переменного тока с заданной амплитудой, частотой и фазой (AC Current Source) задает ток, который меняется по синусоидальному закону и описывается выражением

$$I = \text{Amplitude} \times \sin(2\text{pi} \times \text{Frequency} \times t + \text{Phase} \times \text{pi}/180).$$

Этот компонент имеет окно задания параметров, представленное на рис. 5.3. Для вызова этого окна достаточно дважды щелкнуть на значке компонента в модели системы или устройства. Обратите внимание на то, что амплитуда тока (Peak amplitude) задается в амперах, частота (Frequency) в герцах и фаза (Phase) в градусах (а не в радианах).

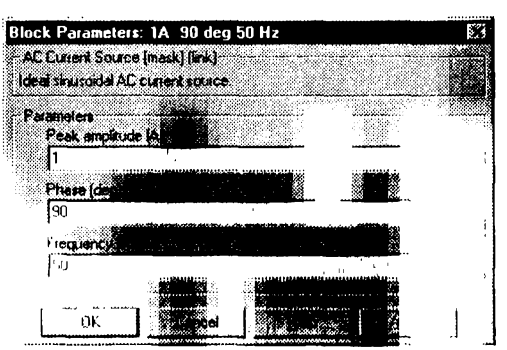


Рис. 5.3. Окно задания параметров источника переменного тока

Рисунок 5.4 показывает пример применения модели источника переменного тока. Здесь показан случай суммирования двух токов в резисторе с номиналом 50 Ом. Этот пример можно найти в файле psbaccurrent.mdl.

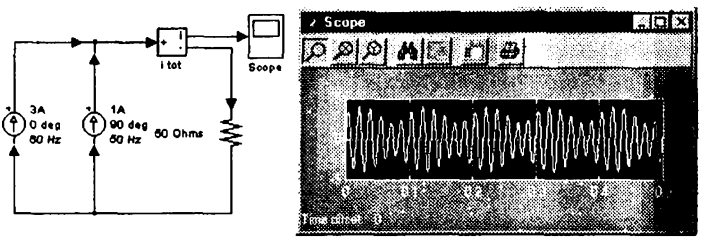


Рис. 5.4. Схема, демонстрирующая суммирование токов от двух источников (слева) и осциллограмма тока в нагрузочном резисторе (справа)

В результате суммирования двух токов с разными частотами (50 и 60 Гц) на результирующем токе наблюдаются биения с разностной частотой. Следует отметить, что хотя осциллограмма тока с биениями напоминает модулированный ток, он таковым не является — модуляция возможна только в нелинейной системе, а приведенная в данном примере простейшая система является линейной.

Источник переменного напряжения

Идеальный (с нулевым внутренним сопротивлением) источник переменного напряжения с заданной амплитудой, частотой и фазой (AC Voltage Source) задает напряжение, меняющееся по синусоидальному закону и описываемое выражением

$$U = \text{Amplitude} \times \sin(2\text{pi} \times \text{Frequency} \times t + \text{Phase} \times \text{pi}/180).$$

Параметры источника задаются с помощью окна, представленного на рис. 5.5. Правила его вызова уже описывались. Амплитуда, на этот раз для напряжения (Peak amplitude), задается в вольтах.

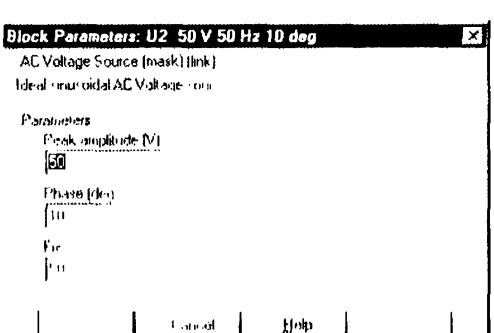


Рис. 5.5. Окно задания параметров источника переменного напряжения

Приведенный на рис. 5.6 пример (файл psbacvoltage.mdl) демонстрирует суммирование двух напряжений — U1 и U2 — переменного тока на резисторе с сопротивлением 100 Ом.

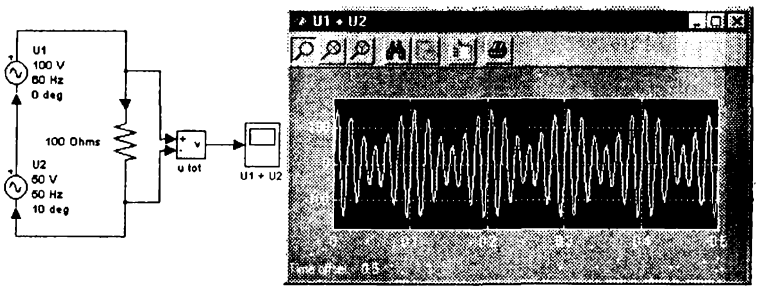


Рис. 5.6. Схема, демонстрирующая суммирование напряжений от двух источников напряжения (слева) и осциллограмма результирующего напряжения

В результате сложения двух переменных напряжений — U1 и U2 — с частотами 50 и 60 Гц видны характерные биения с разностной частотой 10 Гц. Данная цепь также является линейной.

Источник постоянного напряжения

Источник постоянного напряжения DC Voltage Source задается только значением его параметра — выходным напряжением E (такой источник также называется идеальным источником постоянного напряжения). Напряжение во времени не меняется. Окно для установки единственного параметра показано на рис. 5.7.

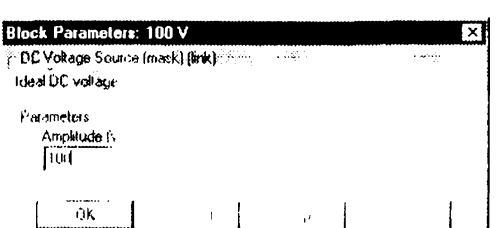


Рис. 5.7. Окно задания параметра источника постоянного напряжения

Пример применения источника постоянного напряжения представлен на рис. 5.8. Этот пример имеется в файле psbdcvoltage.mdl. В данном случае анализируется процесс заряда конденсатора $C = 50$ мкФ

через резистор R = 50 Ом от источника постоянного напряжения 100 В после включения ключа Breaker спустя время, равное 20 мс после начала моделирования. Осциллограф иллюстрирует экспоненциальный процесс заряда конденсатора.

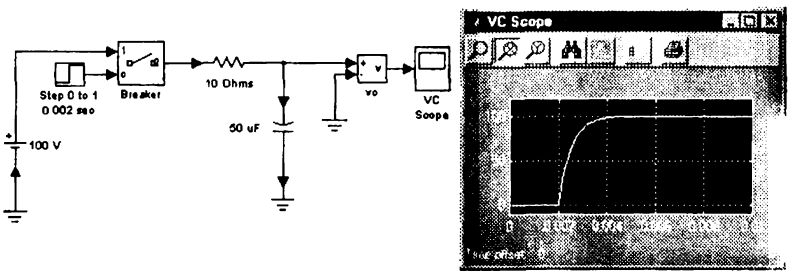


Рис. 5.8. Пример моделирования заряда конденсатора через резистор от источника постоянного напряжения

В этой модели, впервые обеспечивающей наблюдение переходного процесса, использованы некоторые компоненты, которые будут описаны ниже. Однако это не должно вызывать заметных затруднений, поскольку назначение этих компонентов вполне очевидно. Следует отметить, что вообще идеология работы с пакетом предполагает интуитивное применение большинства компонентов, используемых для построения моделей электрических систем и устройств.

Управляемый источник тока

Управляемый источник тока Controlled Current Source задает во внешней цепи ток, который зависит от начального тока и величины управляющего тока. В окне параметров этого источника (рис. 5.9) задается только начальный ток (Initial current, по умолчанию равный 0). При этом временная зависимость тока определяется временной зависимостью управляющего тока.

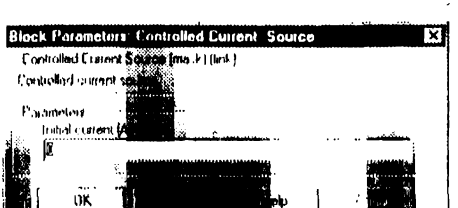


Рис. 5.9. Окно задания параметра управляемого источника тока

Пример применения управляемого источника тока представлен на рис. 5.10 (файл psbcontrolcurr.mdl). Обратите внимание на то, что управляющий сигнал для управляемого источника тока создается операциями суммирования и умножения с помощью двух источников переменного тока и одного источника постоянного тока. Для этого используются соответствующие блоки системы Simulink.

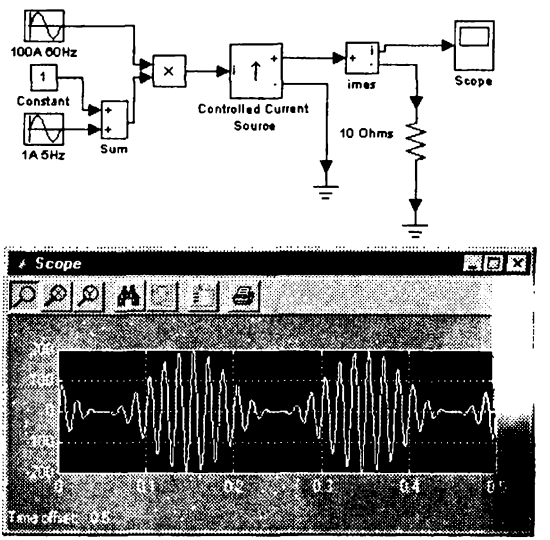


Рис. 5.10. Пример применения управляемого источника тока

Суть работы схемы в этом примере вполне очевидна — на резисторе в 10 Ом получается модулированный ток. Источником модулируемого тока является источник 100 А 60 Гц, модулирующим является источник 1 А 5 Гц. Операция перемножения токов ведет к тому, что цепь в данном примере становится нелинейной.

Управляемый источник напряжения

Управляемый источник напряжения Controlled Voltage Source задает на зажимах внешней цепи напряжение, которое зависит от начального напряжения и величины управляющего сигнала. В окне параметров этого источника (рис. 5.11) задается только начальное напряжение (Initial voltage).

Пример применения управляемого источника напряжения представлен на рис. 5.12. Этот пример содержится в файле psbcontrolvolt.mdl.

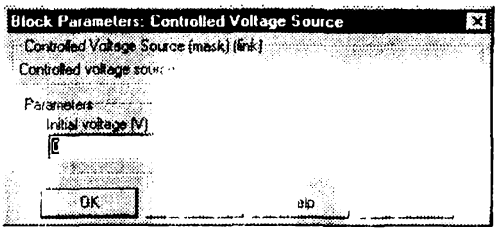


Рис. 5.11. Окно задания параметра управляемого источника напряжения

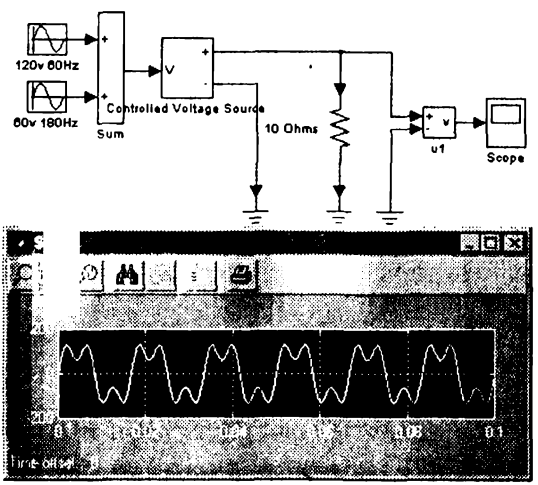


Рис. 5.12. Пример применения управляемого источника напряжения

Суть работы схемы в этом примере также вполне очевидна — получаем ток, представленный первой и третьей гармониками с частотами 60 и 180 Гц и разными амплитудами.

Управляемые источники напряжения и тока позволяют моделировать такой важный класс электрических цепей, как *параметрические цепи*, параметры которых зависят от времени.

Соединительные элементы

Состав соединительных элементов

Состав библиотеки соединительных элементов представлен на рис. 5.13.

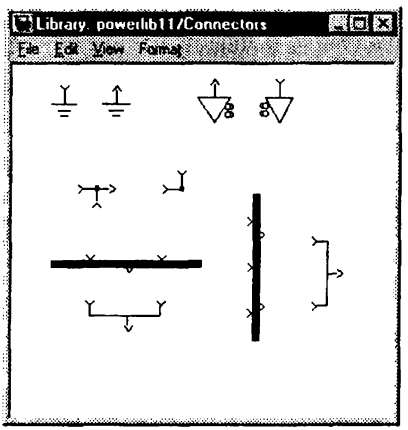


Рис. 5.13. Библиотека соединительных элементов

Большинство соединительных элементов имеют вполне очевидное назначение. Например, заземляющие элементы служат для подключения других элементов к общему проводу — земле. Эти элементы не имеют параметров и в их окне параметров предусмотрено лишь инвертирование изображения элементов. Более детально эти элементы рассматривать не имеет смысла ввиду очевидности их назначения.

Нейтраль

Нейтраль — это общий для всех сложных цепей провод. Относительно нейтрали отсчитываются потенциалы разных точек цепи. Нейтраль не следует путать с землей. Земля (блок Ground) также может рассматриваться как нейтраль с номером узла 0.

Обычно нейтраль на схеме электрической цепи не показывают и характеризуют ее номером узла. Таким образом, нейтраль имеет единственный параметр (номер узла). Она представляется на схеме как большой треугольник с острием, обращенным вниз, и номером узла рядом.

Рисунок 5.14 показывает применение заземляющего элемента и нейтралей в схеме трехфазной сети переменного тока. Этот пример имеется в файле psbneutral.mdl.

Данный пример поясняет возникновение напряжений на одной из фаз относительно нейтрали и земли энергетической системы. Следует отметить, что основные возможности в организации соединений между компонентами электрических цепей обеспечивает система

Simulink. Поэтому набор соединительных элементов в пакете Power System Blockset ограничен лишь специфическими для силовых цепей компонентами.

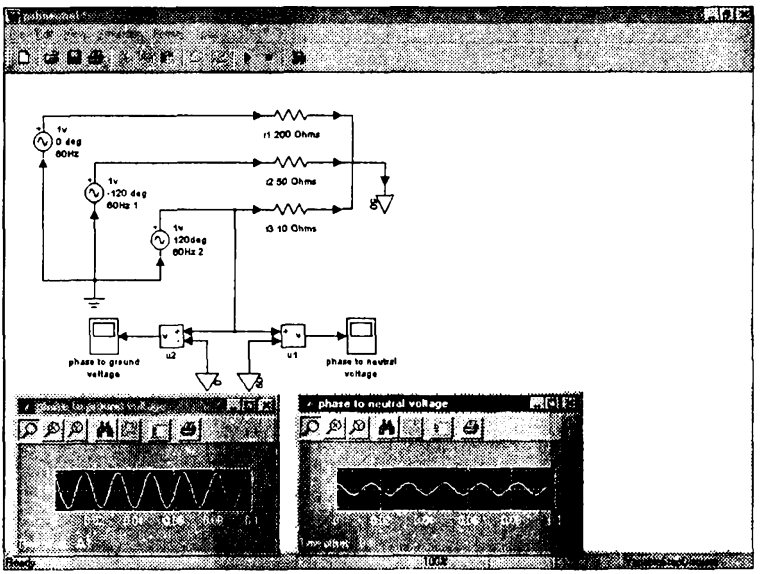


Рис. 5.14. Применение заземляющего элемента и нейтралей

Блок шин

Блок шин Bus является идеальным переходником от одиночной шины к ряду шин. Окно параметров этого блока (рис. 5.15) содержит всего два параметра — число входных шин (Number of inputs) и число выходных шин (Number of outputs).

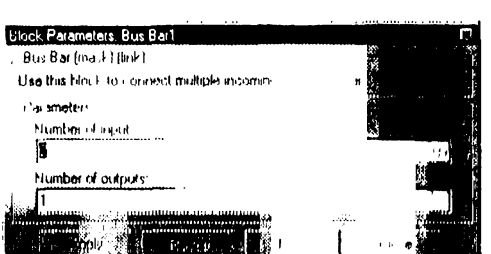


Рис. 5.15. Окно параметров блока шин

Пример применения блока шин показан на рис. 5.16 (файл psbground.mdl). В этом примере блок шин используется для объединения токов от трех входных цепей в одной выходной шине. Эти цепи образованы последовательным контуром R1L1C1, резистором R2 и конденсатором C2.

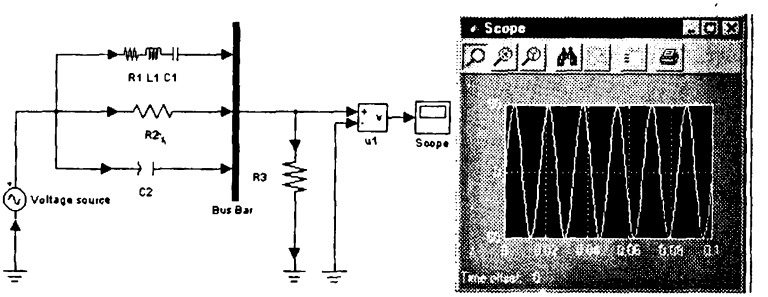


Рис. 5.16. Пример применения блока шин

В данном случае на модели опущено указание номиналов компонентов. Напоминаем, что они заданы в окнах установки параметров. Поскольку данный пример взят из справочной системы MATLAB, то выявление этих параметров не составит труда для читателя.

Библиотека компонентов

Основная библиотека компонентов содержит ряд моделей, имеющих достаточно универсальный характер. С помощью одной такой модели можно, как правило, создать модели нескольких компонентов. Окно библиотеки компонентов представлено на рис. 5.17.

Эта библиотека содержит несколько характерных компонентов:

- Series RLC Branch — последовательная RLC-цепь;
- Series RLC Load — последовательная RLC-цепь — нагрузка;
- Parallel RLC Branch — параллельная RLC-цепь;
- Parallel RLC Load — параллельная RLC-цепь — нагрузка;
- Linear Transformer — линейный трансформатор;
- Saturable Transformer — насыщающийся (нелинейный) трансформатор;
- Mutual Inductance — блок взаимной индуктивности;
- Surge Arrester — ограничитель пиковых напряжений;

- Breaker — управляемый выключатель;
- PI Section Line — линия с сосредоточенными параметрами;
- Distributed Parameters Line — линия с распределенными параметрами.

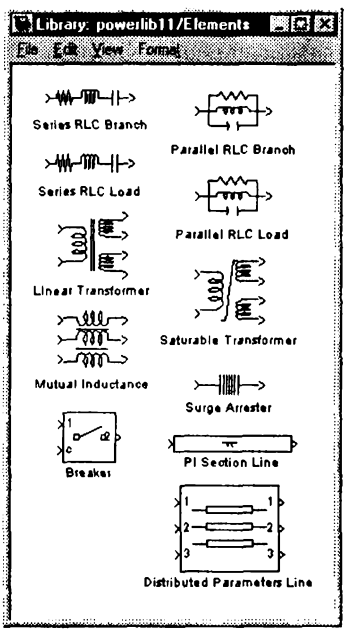


Рис. 5.17. Окно библиотеки компонентов Elements

Подробнее эти элементы будут рассмотрены ниже.

Последовательные и параллельные RLC-цепи

В состав библиотеки входят две последовательные и две параллельные RLC-цепи. Они могут задаваться как обычные и нагрузочные цепи. Обычные цепи (последовательная Series RLC Branch и параллельная Parallel RLC Branch) задаются всего тремя параметрами: сопротивлением R, индуктивностью L и емкостью C. Ввиду очевидности этих параметров окна из задания не приводятся.

У нагрузочных цепей (также последовательной Series RLC Load и параллельной Parallel RLC Load) задаются допустимые мощности рассеивания — активная для резистора и реактивные для индуктивности и конденсатора (рис. 5.18).

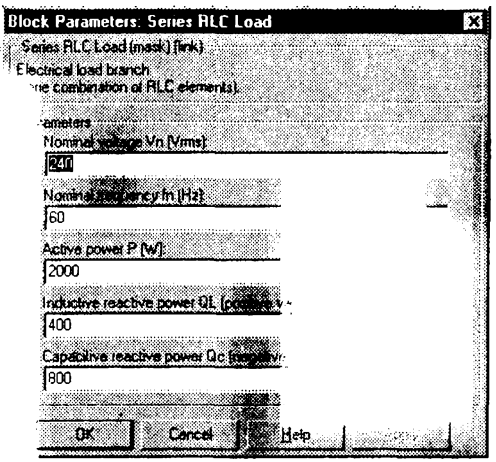


Рис. 5.18. Окно установки параметров нагрузочной последовательной RLC-цепи

Последовательные и параллельные RLC-цепи могут использоваться для моделирования колебательных контуров и создания эквивалентов нагрузки.

Отдельные элементы R, L и C

Для ввода отдельных элементов (резистора R, конденсатора C и индуктивности L) можно использовать любую из RLC-цепей, задав соответствующее значение нужному параметру и значение, соответствующее отсутствию элементов, для других параметров. Например, если надо с помощью последовательной RLC-цепи задать только резистор R, следует указать L = 0 (индуктивность при этом исчезнет и будет заменена проводником) и C = inf (inf означает бесконечное значение емкости, что превращает ее также в проводник). Это правило модификации распространяется и на другие сложные компоненты, например на цепочки RsCs, гасящие выбросы в моделях ключей (они будут описаны ниже).

Благодаря этому правилу число простых моделей в пакете Power System Blockset сокращено. Кроме того, это правило позволяет быстро модернизировать отдельные цепи, например превращать резистор в RL- или RLC-цепь, не вводя новые компоненты в уже составленную схему, а просто указав нужные параметры в окне установки параметров RLC-цепей.

Примеры моделирования RLC-цепей

Рисунок 5.19 показывает пример (файл psbseriesload.mdl) моделирования последовательной нагрузочной цепи при подключении ее к источнику переменного напряжения 240 В с частотой 60 Гц и внутренним сопротивлением 2 Ом, имитируемым резистором R. Измеряются напряжение на нагрузочной цепи и ток в ней. Блок powergui обеспечивает графический интерфейс пользователя и контроль параметров модели в целом.

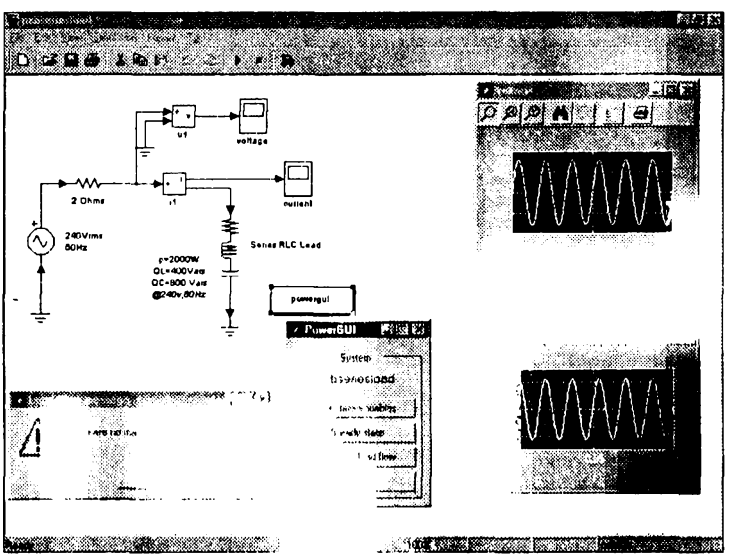


Рис. 5.19. Пример моделирования нагрузочной последовательной RLC-цепи, подключенной к источнику переменного напряжения

Активизация значка powergui выводит окно этого интерфейса — оно показано внизу рис. 5.19 вместе с окном вывода сообщений об ошибках. Окно PowerGUI позволяет контролировать статус переменных модели и параметры ее компонентов. Это поясняет рис. 5.20, на котором представлены соответствующие окна для контроля и изменения параметров модели. Более подробно работа с блоком powergui будет описана в дальнейшем.

На рис. 5.21 представлен пример моделирования комбинированной цепи, которая образована конденсатором емкостью 4,42 мкФ и неполной параллельной RL-цепью. Этот пример имеется в файле

psbparalbranch.mdl. Как видно из окна параметров этой цепи, также показанного на рис. 5.21, для исключения из параллельной цепи конденсатора С достаточно задать его емкость равной нулю.

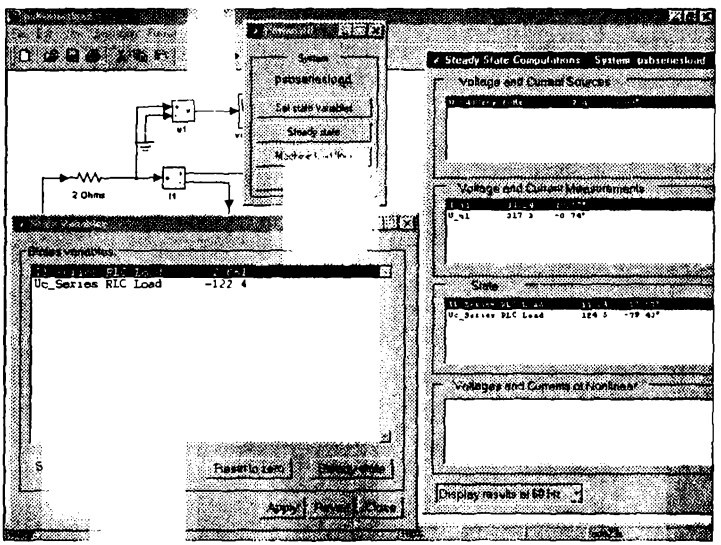


Рис. 5.20. Контроль статуса переменных и параметров модели рис. 5.19

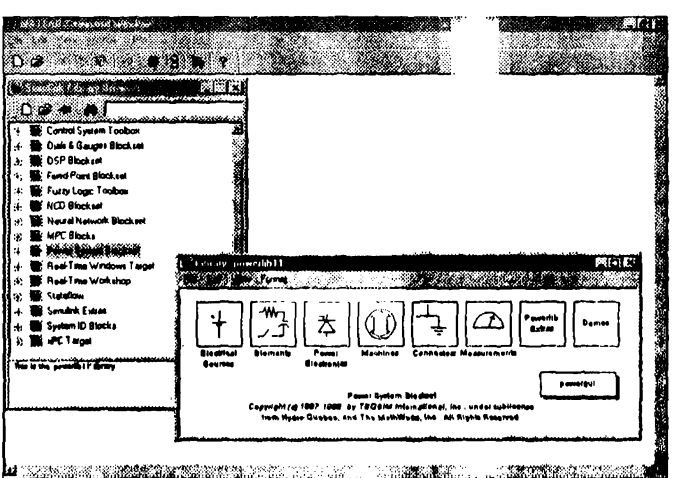


Рис. 5.21. Пример моделирования комбинированной цепи

В данном случае комбинированная цепь настроена на частоту резонанса $f_p = 1/(2\pi\sqrt{LC})$ для 11-й гармоники (660 Гц) переменного тока с частотой 60 Гц, источник которого нагружен на данную цепь.

Контролируется напряжение на этой цепи. Поскольку частота резонанса цепи далека от частоты источника, то цепь представляет собой большое по модулю реактивное сопротивление, что ведет к большому напряжению на ней. Напряжение на цепи остается синусоидальным и имеет частоту источника (60 Гц), что является следствием линейности данной модели (как известно, в линейных цепях возникновение новых частот спектра исключено).

Применение пакета Control System Toolbox

Весьма интересной является возможность совместного использования пакетов Power System Blockset и рассмотренного ранее пакета Control System Toolbox. Рассмотрим это на примере вычисления полного сопротивления цепи рис. 5.21. Импеданс этой цепи задается выражением:

$$Z(s) = \frac{V(s)}{I(s)} = \frac{RLCs^2 + Ls + R}{LCs^2 + RCs}.$$

Ниже представлен текст m-файла, который с помощью функции данного пакета расширения power2sys и стандартных средств графики MATLAB обеспечивает вычисление модуля импеданса (полного сопротивления) цепи и фазового сдвига с построением их логарифмических зависимостей от частоты:

```
[A,B,C,D] = power2sys('psbparalbranch');
freq = logspace(1,4,500);
w = 2*pi*freq;
[Z,phaseZ] = bode(A,B,C,D,1,w);
subplot(2,1,1)
loglog(freq,Z)
grid
title('11th harmonic filter')
xlabel('Frequency, Hz')
ylabel('Impedance Z')
subplot(2,1,2)
semilogx(freq,phaseZ)
xlabel('Frequency, Hz')
ylabel('phase Z')
grid
```

Эти характеристики представлены на рис. 5.22. Из этих зависимостей отчетливо видно, что полное сопротивление комбинированной

цепи минимально на ее частоте резонанса (660 Гц) и весьма велико на частоте источника переменного тока.

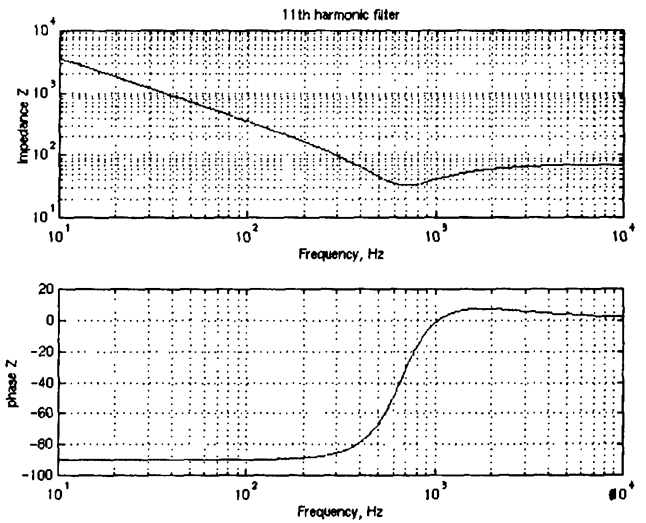


Рис. 5.22. Логарифмические зависимости модуля и фазы полного сопротивления комбинированной цепи, представленной на рис. 5.21

Этот прием открывает большие возможности для всестороннего исследования свойств электрических цепей в различных областях их применения.

Модель линейного трансформатора

В пакете Power System Blockset имеется модель линейного трансформатора Linear Transformer, представленная на рис. 5.23. Она задана индуктивностью Lm и сопротивлением потерь в сердечнике Rm первичной обмотки трансформатора, а также омическими сопротивлениями Ri и индуктивностями рассеивания Li всех обмоток трансформатора (i = 1, 2, 3).

При описании параметров трансформатора используется дополнительная система параметров, принятая в индустрии Запада и называемая в описании пакета pu-системой. При этом вводятся обозначения, представленные на рис. 5.24 с примерами их расчета. Запись обозначений здесь не вполне корректна, так как при формальном математическом подходе получается $pu = R_{base} = L_{base}$, что является грубой

ошибкой. На самом деле под 1 pu в системе pu-единиц понимаются разные параметры.

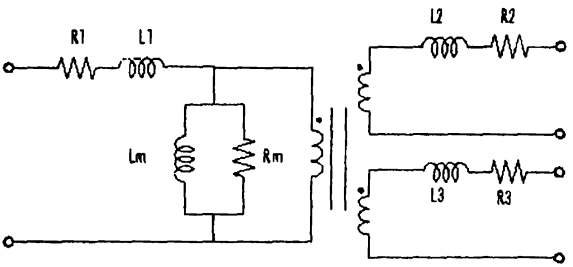


Рис. 5.23. Модель линейного трансформатора

$$R_{\text{base}} = 1\,\text{pu} = \frac{(V_n)^2}{P_n};$$

$$L_{\text{base}} = 1\,\text{pu} = \frac{R_{\text{base}}}{2\pi f_n};$$

$$R_{\text{base}} = \frac{(424.35 \cdot 10^3)^2}{250 \cdot 10^6} = 720.3\Omega;$$

$$L_{\text{base}} = \frac{720.3}{2\pi 60} = 1.91H;$$

$$R_1 = 0.02\,\text{pu} \cdot 720.3\,\Omega = 1.44\Omega;$$

$$L_1 = 0.08\,\text{pu} \cdot 1.91\ H = 0.1528\,H;$$

$$R_m = 500\,\text{pu} \cdot 720.3\,\Omega = 36 \cdot 10^5\,\Omega$$

Рис. 5.24. Формулы пересчета параметров R_{base} и L_{base} для трансформатора

Вид окна установки параметров линейного трансформатора представлен на рис. 5.25. Обратите внимание на то, что некоторые параметры трансформатора задаются списками (векторами).

На рис. 5.26 показан пример моделирования трансформатора линейной подстанции с мощностью 75 кВт. Этот пример можно найти в файле psbtransformer.mdl.

Напряжение на входе трансформатора 14 400 В, на выходе 240 В и 2×120 В. Осциллографы контролируют ток в первичной обмотке и напряжение на одной из вторичных обмоток.

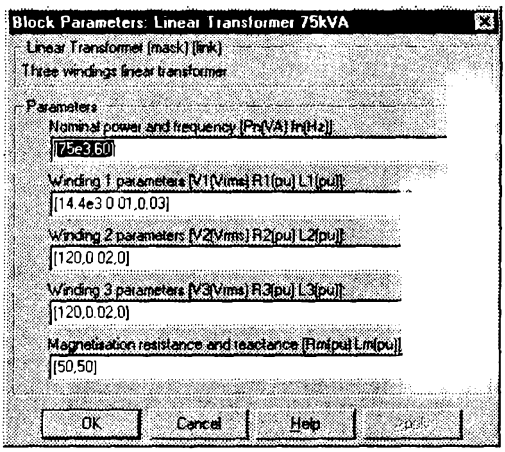


Рис. 5.25. Окно установки параметров линейного трансформатора

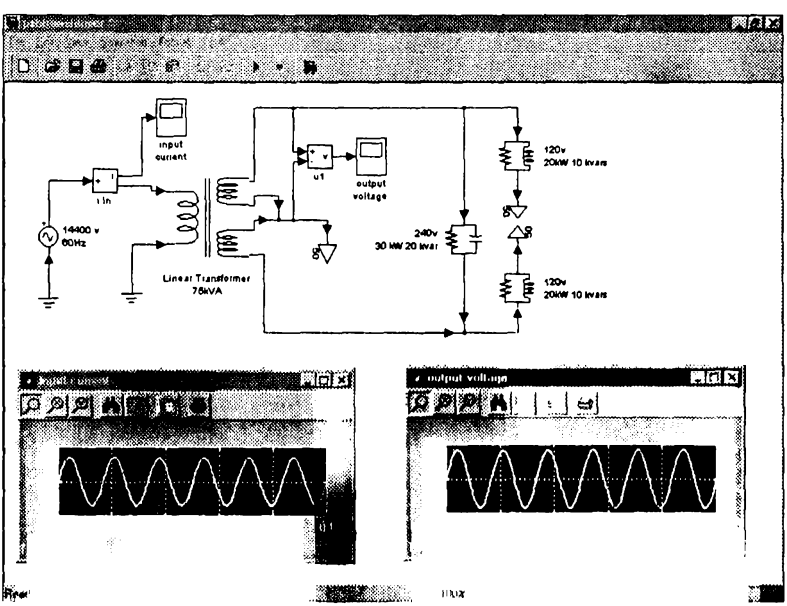


Рис. 5.26. Пример моделирования трансформатора линейной подстанции

Модель линейного трансформатора в значительной мере идеализирована.

Модель нелинейного трансформатора

Модель нелинейного трансформатора Saturable Transformer (рис. 5.27) отличается от модели линейного трансформатора тем, что индуктивность первичной обмотки Lm заменена на нелинейную индуктивность Lsat. Для нелинейных трансформаторов также может использоваться pu-система параметров (см. рис. 5.24).

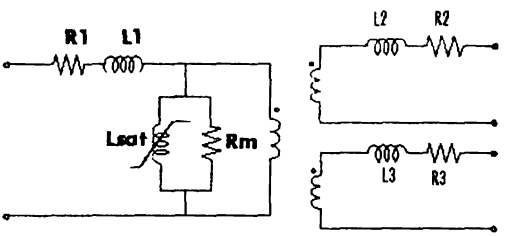


Рис. 5.27. Модель нелинейного трансформатора

Нелинейность трансформатора учитывается зависимостями, представленными на рис. 5.28. Допускается два вида этой зависимости, отличающихся числом указываемых опорных точек и поведением зависимости в области малых токов.

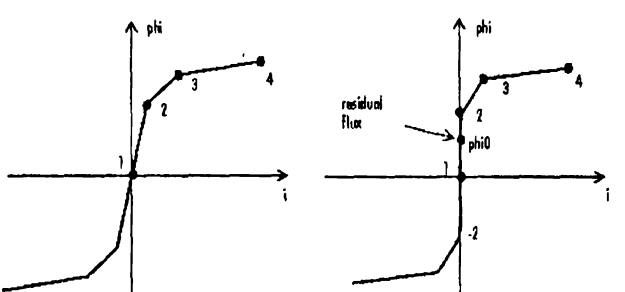


Рис. 5.28. Модели нелинейности трансформатора

На рис. 5.29 представлен пример моделирования одной фазы мощной трансформаторной подстанции с установочной мощностью 1000 МВт. Там же показано окно установки параметров нелинейного трансформатора. Обратите внимание на то, что здесь ряд параметров также задается в виде списков (векторов). Дополнительно задаются параметры нелинейности трансформатора.

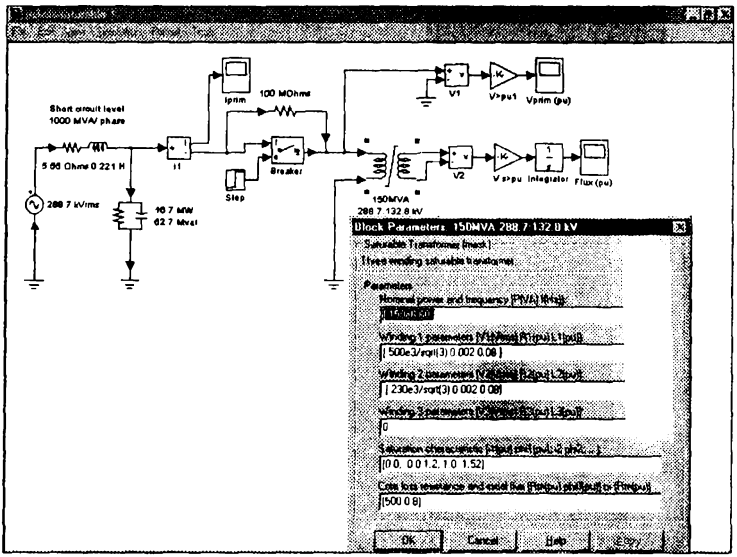


Рис. 5.29. Модель трансформаторной подстанции и окно установки параметров нелинейного трансформатора

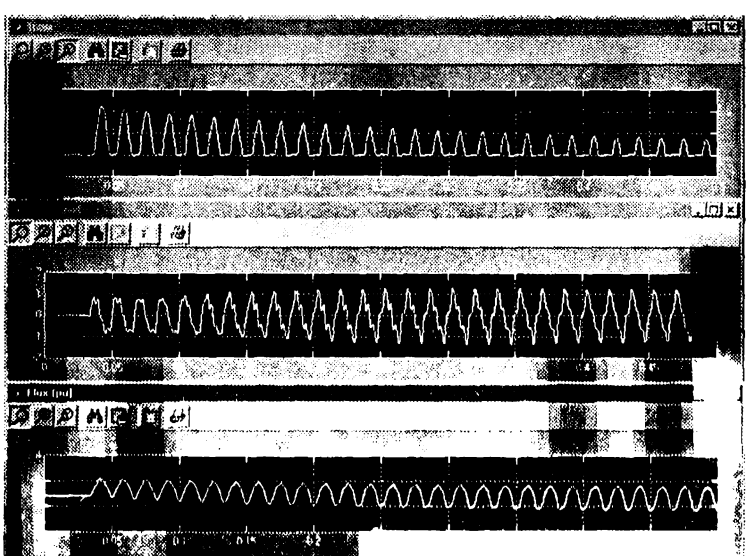


Рис. 5.30. Результаты моделирования системы с мощным нелинейным трансформатором

Нетрудно заметить (рис. 5.30), что переходный процесс в системе с нелинейным трансформатором намного сложнее, чем в схеме с линейным трансформатором. При этом он длится примерно два десятка периодов переменного напряжения входного источника электропитания.

К сожалению, модель нелинейного трансформатора не учитывает гистерезиса петли намагничивания сердечника трансформатора и потому является лишь первым приближением к учету явлений, связанных с нелинейностью трансформатора. Однако пользователь может создать свою модель трансформатора, в том числе учитывающую гистерезис.

Блок взаимной индуктивности

Катушки индуктивности и даже отдельные проводники, расположенные вблизи друг от друга, имеют перекрывающиеся магнитные поля, что создает эффект взаимной индуктивности. Для моделирования взаимной индуктивности в пакете Power System Blockset служит блок (модель) взаимной индуктивности Mutual Inductance на основе идеального трансформатора (рис. 5.31). Он соответствует теоретической модели взаимной индуктивности.

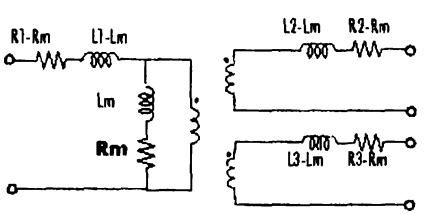


Рис. 5.31. Блок взаимной индуктивности

Пример моделирования электрической цепи с блоком взаимной индуктивности представлен на рис. 5.32 (файл psbmutual.mdl). Там же представлено окно установки параметров блока взаимной индуктивности. Обратите внимание на то, что параметры обмоток трансформатора, имитирующего взаимную индуктивность, задаются списками (векторами).

В этом примере создается двухчастотное напряжение на RL-нагрузке, причем как первая (60 Гц), так и третья (180 Гц) гармоники создаются благодаря взаимной индуктивности, представленной блоком Mutual Inductance.

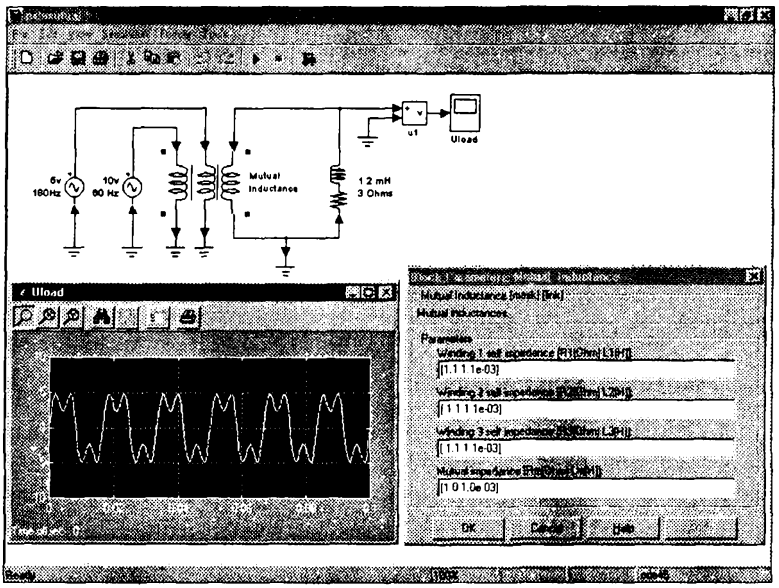


Рис. 5.32. Пример моделирования электрической цепи с блоком взаимной индуктивности

Нелинейный ограничитель пиковых напряжений

Многие радиоэлектронные устройства весьма чувствительны даже к кратковременным выбросам питающего напряжения. Такие выбросы нежелательны и в силовых цепях, поскольку провоцируют развитие электрического пробоя и возникновение дуги. Выбросы могут быть обусловлены ударами молнии или характером переходных процессов в электрических цепях. Для защиты от подобных выбросов были созданы специальные нелинейные резистивные устройства — ограничители пиковых напряжений. У нас их называют также *варисторами*. Эти устройства основаны на управляемом электрическом пробое и способны выдерживать без разрушения кратковременные выбросы тока до 500–1000 А.

В пакете Power System Blockset можно задавать ограничители пиковых напряжений Surge Arrester с вольтамперной характеристикой, которая описывается следующим выражением:

$$\frac{V}{V_{\text{ref}}} = K_I \left(\frac{I}{I_{\text{ref}}} \right)^{V_{\text{exp}}}.$$

Такую вольтамперную характеристику можно представить в виде, показанном на рис. 5.33 — слева в обычном масштабе и справа в логарифмическом. Из этого рисунка четко видна возможность разбиения вольтамперной характеристики на несколько характерных участков, что и используется при ее задании в окне установки параметров ограничителя.

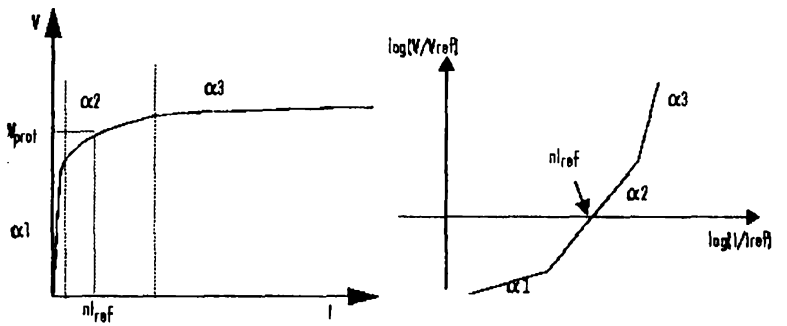


Рис. 5.33. Вольтамперные характеристики нелинейного ограничителя в линейном (слева) и логарифмическом (справа) масштабах

На рис. 5.34 представлены модель электрической системы с варистором, окно установки его параметров и результаты моделирования. Этот пример имеется в файле psbarrester.mdl. Обратите внимание на то, что помимо осциллограмм напряжений, иллюстрирующих эффективное ограничение выбросов, представлено окно графопостроителя, в котором построена вольтамперная характеристика варистора.

Применение варисторов является одним из самых простых и эффективных методов ограничения выбросов напряжения. Наряду с мощными варисторами, упомянутыми выше, выпускаются подобные ограничители средней и даже малой мощности, например для защиты бытовых радиоэлектронных устройств.

Варистор является типичным представителем нелинейных устройств, как и рассмотренный ранее нелинейный трансформатор. Многие нелинейные устройства, в отличие от линейных, не имеют аналитических решений для уравнений, описывающих их состояние. Это делает моделирование нелинейных цепей на компьютерах (в том числе с помощью пакета Power System Blockset) особенно ценным для практики.

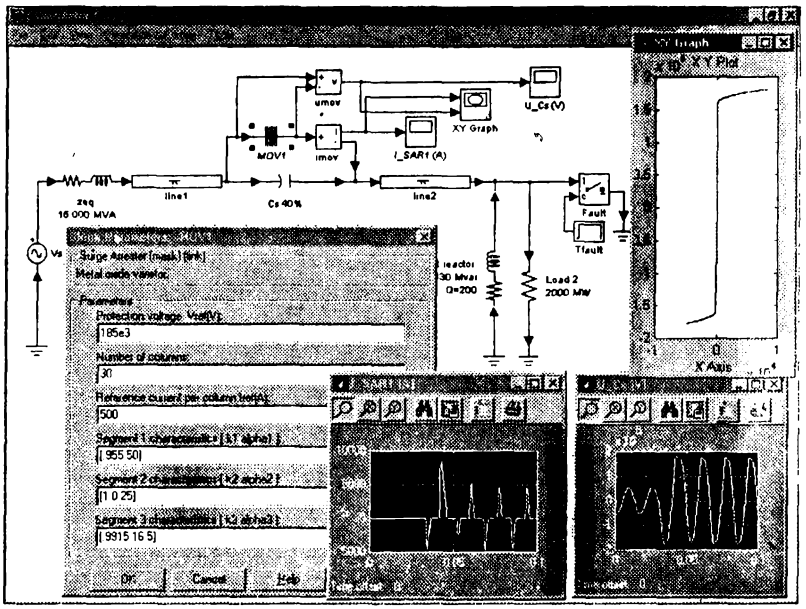


Рис. 5.34. Моделирование электрической системы с варистором – ограничителем выбросов напряжения

Выключатель

Для имитации выключателей (рубильников) служит управляемый выключатель — Breaker. Он обеспечивает включение или выключение переменного тока. Пример применения этого элемента уже был показан ранее (см. рис. 5.8). Приведем еще один пример (рис. 5.35, файл psbbreaker.mdl).

Здесь наряду с моделью показаны окно установки параметров ключа и результаты моделирования цепи. Обратите внимание на то, что возможно задание паразитного сопротивления и индуктивности выключателя, что позволяет при необходимости моделировать эффекты, связанные с неидеальностью выключателя.

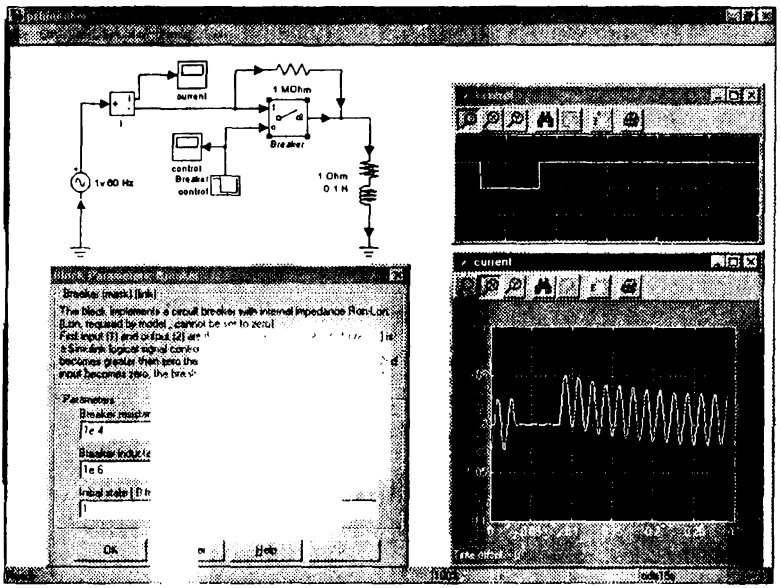


Рис. 5.35. Пример моделирования цепи с ключом

Линии передачи с сосредоточенными постоянными

В пакете Power System Blockset имеется возможность задания двух типов линий передачи. Первый тип — это линии PI Section Line, структура которых представлена на рис. 5.36. Это линия с сосредоточенными параметрами. Она может приближенно представлять и линию с распределенными параметрами. В последнем случае длина линий задается в километрах и параметрами линий являются распределенные индуктивности и емкости. Они измеряются, соответственно, в Г/км и Ф/км. Взаимная индуктивность у таких линий не учитывается.

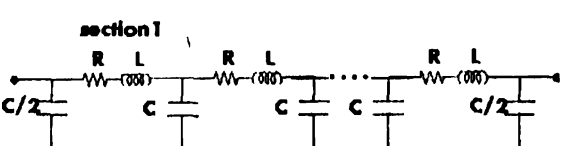


Рис. 5.36. Линия типа PI Section Line

Максимальная частота переменного тока, который может протекать через линию данного типа, составляет $f_{max} = N\,v/(8l)$, где N — число секций PI-линии, $v = 1/\sqrt{LC}$ — скорость распространения волны в линии, и l — длина линии. Например, для линии с v = 300 000 км/с максимальная частота переменного тока составит всего 375 Гц при длине линии в 100 км и использовании одной секции.

На рис. 5.37 показана модель электрической системы с линией передачи рассматриваемого типа. Там же приведены окно установки параметров линии и осциллограммы, иллюстрирующие результат моделирования данной системы. Рассматривается случай подключения линии к источнику переменного напряжения в момент, когда амплитуда его положительной полуволны почти достигла максимума.

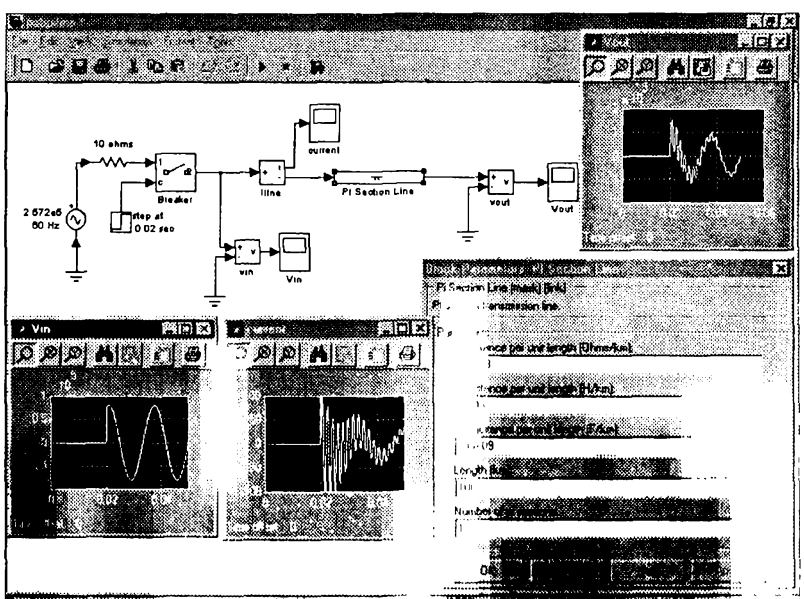
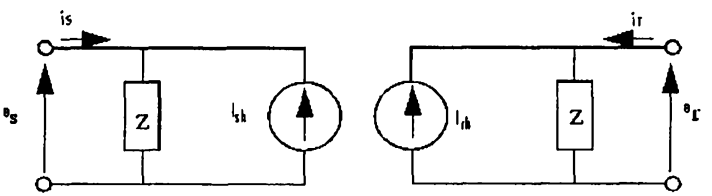


Рис. 5.37. Пример моделирования электрической системы с линией типа PI Section Line

Полезно обратить внимание на довольно сложные переходные процессы, возникающие при подключении источника переменного напряжения к линии. Помимо колебаний, происходящих с частотой этого источника, виден более высокочастотный переходной процесс. Он особенно заметен на осциллограмме тока линии.

Линии передачи с распределенными параметрами

Другой возможный тип линий — линии с распределенными параметрами Distributed Parameters Line. Теоретическая модель такой линии с соответствующими расчетными выражениями представлена на рис. 5.38. Смысл этих выражений очевиден для специалистов, применяющих подобные линии, и потому здесь детально не обсуждается.



$$I_{sh}(t) = \left(\frac{1+h}{2}\right)\left[\frac{1}{Z}e_r(t-\tau)+hi_r(t-\tau)\right]+\left(\frac{1-h}{2}\right)\left[\frac{1}{Z}e_s(t-\tau)+hi_s(t-\tau)\right]$$

$$I_{rh}(t) = \left(\frac{1+h}{2}\right)\left[\frac{1}{Z}e_s(t-\tau)+hi_s(t-\tau)\right]+\left(\frac{1-h}{2}\right)\left[\frac{1}{Z}e_r(t-\tau)+hi_r(t-\tau)\right]$$

$$Z = Z_C + \frac{R}{4} \qquad h = \frac{Z_C - \frac{R}{4}}{Z_C + \frac{R}{4}} \qquad Z_C = \sqrt{\frac{L}{C}} \qquad \tau = d\sqrt{LC}$$

Рис. 5.38. Теоретическая модель линии с распределенными параметрами

Окно установки параметров распределенной линии представлено на рис. 5.39. В нем можно задать число фаз линии, рабочую частоту, значения распределенных индуктивности и емкости, а также длину линии.

Пример (файл psbdistline.mdl) моделирования электрической системы с распределенной линией представлен на рис. 5.40. Там же даны осциллограммы, представляющие результат моделирования. В этом примере моделируется процесс включения источника переменного напряжения с частотой 60 Гц на отрезок линии длиной 300 км.

Сложность переходного процесса в данном примере обусловлена проявлением эффектов отражения, возникающих из-за рассогласования линии в ее начале и в конце. Линия является согласованной, когда она нагружена на резистор с номиналом, равным волновому сопротивлению линии $Z_C = \sqrt{L/C}$. Для линий электропередачи согласование не используется, поскольку ведет к большим потерям энергии в цепях согласования.

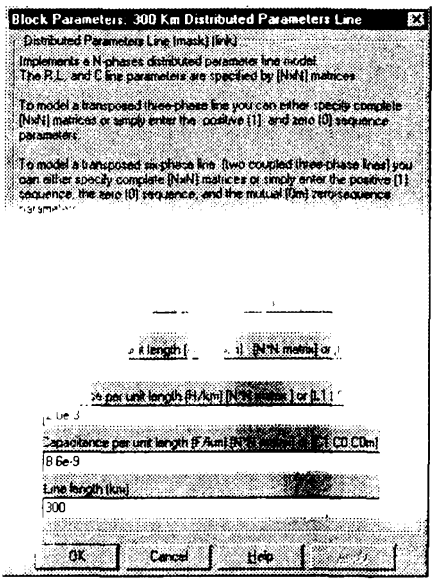


Рис. 5.39. Окно установки параметров распределенной линии

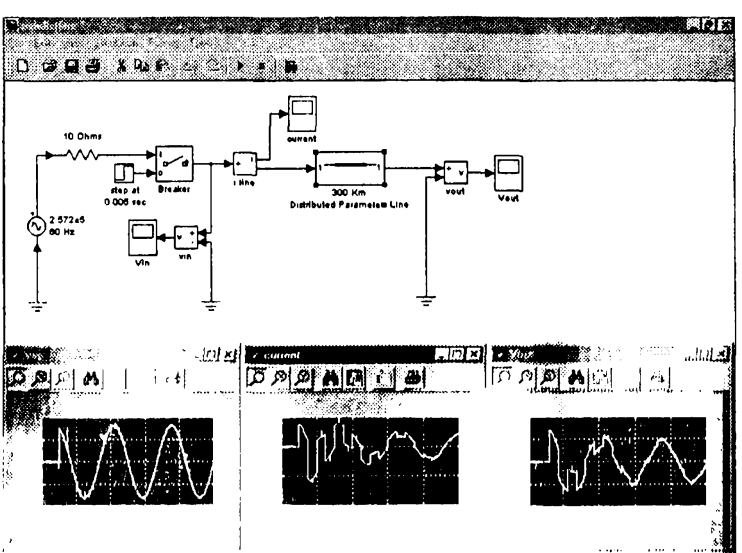


Рис. 5.40. Пример моделирования электрической системы с распределенной линией

Распределенные линии обладают несколько лучшими частотными свойствами по сравнению с линиями с сосредоточенными параметрами. Это позволяет при заданной частоте переменного тока передавать электрическую энергию на большие расстояния. В распределенных линиях радиотехнических устройств, в отличие от линий передачи электроэнергии, обычно используется режим согласования на входе и на выходе. В таком режиме линия на входе и выходе нагружается на сопротивления, равные волновому сопротивлению линии. Это минимизирует отражения сигналов в линиях и позволяет передавать их с минимальными искажениями формы. Однако потери в трактах передачи в этом случае значительны.

Ключевые элементы энергетической электроники

Состав библиотеки энергетической электроники

Современная силовая электроника основана на импульсном способе преобразования электрической энергии, обеспечивающем высокий коэффициент полезного действия преобразовательных устройств. Учитывая это обстоятельство, в библиотеку рассматриваемого пакета включен достаточно представительный набор блоков ключевых устройств. Активизация значка Power Electronics открывает окно с пиктограммами моделей управляемых ключей (рис. 5.41).

Представлены следующие типы ключей:

- Ideal Switch — идеальный (точнее, управляемый) ключ;
- Mosfet — мощный полевой транзистор с изолированным затвором;
- Gto — силовой биполярно-полевой блок;
- Diode — полупроводниковый диод;
- Thyristor — упрощенная модель тиристора;
- Detailed thyristor — детальная модель тиристора.

Некоторое удивление может вызвать отсутствие здесь модели биполярного транзистора. Отчасти это связано с тем, что в последнее время силовые биполярные транзисторы заменяются мощными полевыми транзисторами, а отчасти и тем, что пользователь (в случае необходимости) может составить субмодель биполярного транзистора, например, на основе модели идеального ключа.

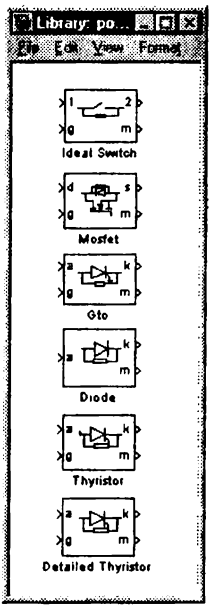


Рис. 5.41. Состав библиотеки компонентов энергетической электроники

Модификация моделей ключевых устройств

Все модели ключевых элементов имеют имитацию гасящей выбросы напряжения последовательной цепи RsCs, которая подключается к силовым выводам моделей. Задание бесконечного значения для Cs и нулевого для Rs превращает модель прибора в короткозамыкающую перемычку (значок прибора при этом заменяется значком проводника). Это позволяет имитировать пробой ключевого прибора, что нередко резко меняет характер работы моделируемого устройства. Модели имеют также выход m для подключения измерительных приборов. На этом выходе формируется список мгновенных значений тока, протекающего через прибор, и напряжения на нем в процессе моделирования. Рассмотрим параметры этих компонентов и примеры их применения.

Управляемый ключ

Идеальный ключ Ideal Switch на самом деле моделирует ключ, который во включенном состоянии имеет сопротивление Ron и индуктивность Lon. Сопротивление Ron позволяет приближенно учитывать

статические потери в ключе во включенном состоянии, а индуктивность Lon — инерционные процессы переключения ключа. Задание Lon = 0 недопустимо, поскольку ведет к прекращению решения системы уравнений, описывающих работу электрических цепей, из-за операции деления на ноль. Сопротивление ключа в выключенном состоянии считается равным бесконечности. Если оно должно быть конечным, то это легко моделируется включением резистора соответствующего номинала параллельно зажимам ключа.

В исходном состоянии ключ может быть закрыт или открыт в зависимости от параметра Initial State. Кроме того, может задаваться последовательная RsCs-цепь, которая включается параллельно ключу для ограничения выбросов напряжения и подавления дуги. Окно задания параметров ключа представлено на рис. 5.42.

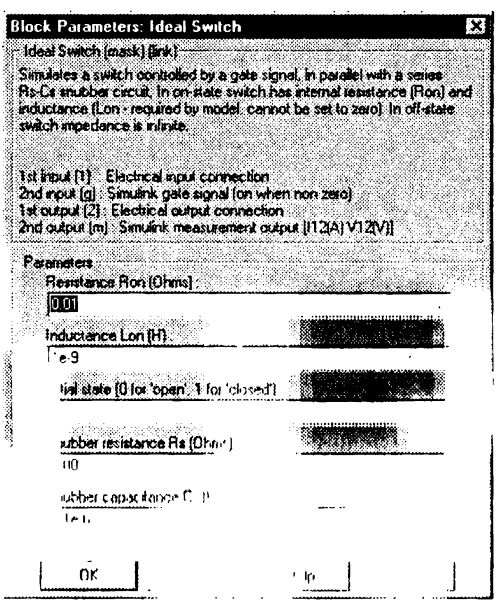


Рис. 5.42. Окно задания параметров ключа

Пример электрической цепи с применением управляемого ключа, параметры которого указаны на рис. 5.42, приведен на рис. 5.43 (файл psbswitch.mdl). Модель иллюстрирует процесс отключения параллельного RLC-контура от источника переменного напряжения 120 В 60 Гц с помощью ключа. Нетрудно заметить, что процесс отключения со-

провождается явно выраженным затухающим высокочастотным переходным процессом.

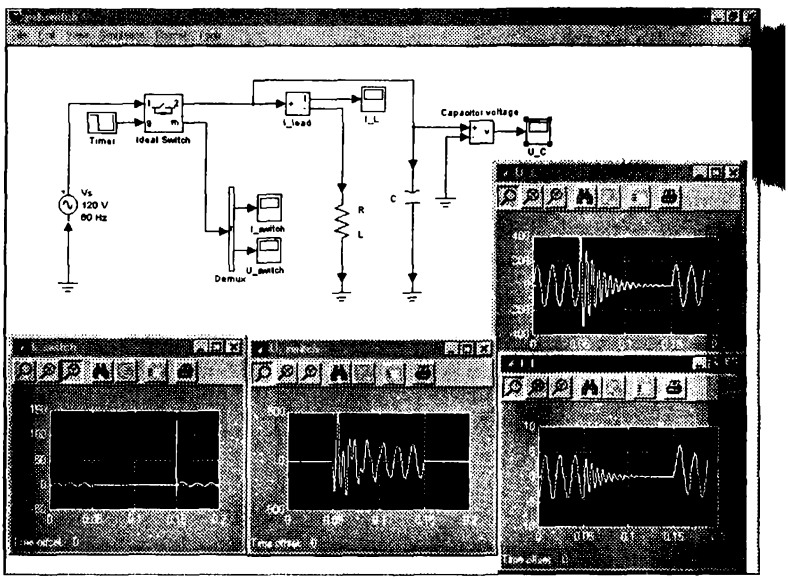


Рис. 5.43. Пример моделирования цепи с идеальным ключом

Приведенный на рис. 5.43 пример моделирования показывает, что название «идеальный ключ» (в оригинале — Ideal Switch) является неточным и сужает возможности модели. На самом деле модель описывает *реальный* ключ с его основными паразитными параметрами. Они и создают заметные высокочастотные колебательные процессы, наблюдаемые при моделировании цепей с такими ключами.

Полупроводниковый диод

Модель диода Diode представляет собой последовательно соединенные источник напряжения Vf, падающего на диоде в прямом включении, резистор Ron и паразитную индуктивность Lon, когда диод проводит ток (в прямом направлении). В обратном направлении сопротивление диода считается бесконечно большим. Предусмотрена возможность включения параллельно диоду гасящей цепи RsCs, содержащей последовательно включенные резистор и конденсатор. Эта цепь является внутренней и на схеме модели не показана. Все эти возможности дает окно установки параметров диода, представленное на рис. 5.44.

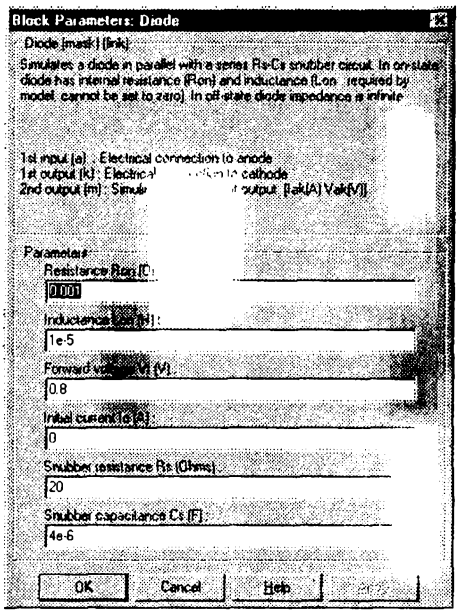


Рис. 5.44. Окно задания параметров диода

Рисунок 5.45 дает пример моделирования однополупериодного диодного выпрямителя, работающего на RL-нагрузку с небольшой индуктивностью. Этот пример есть в файле psbdiode.mdl. Большое число осциллограмм позволяет достаточно полно судить о физических процессах, протекающих в данной схеме (а точнее, о результатах их моделирования).

Следует отметить, что модель диода довольно приближенная. Так, источник напряжения Vf дает ненулевое напряжение на диоде при токе через него, равном нулю (на самом деле это напряжение, естественно, равно нулю). Прямая ветвь диода может лишь приближенно описываться прямой, наклон которой задается номиналом резистора Ron (в действительности ток диода экспоненциально зависит от напряжения на нем). Наконец, неточно учитывается инерционность диода, связанная с накоплением в его базе избыточных зарядов неосновных носителей и их рассасыванием при выключении. Учет этих явлений индуктивностью Lon является довольно грубым приближением. Тем не менее, принятые для модели приближения вполне соответствуют тем, что применяются в практике использования диодов при проектировании силовых устройств.

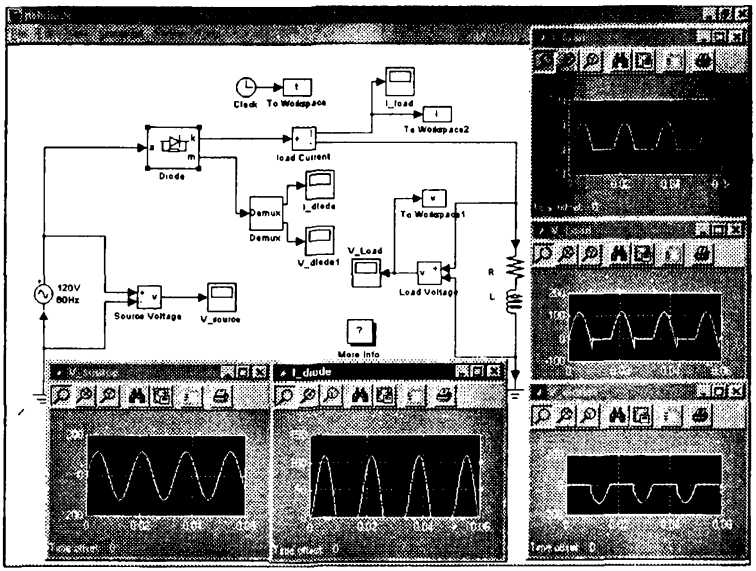


Рис. 5.45. Пример моделирования однополупериодного диодного выпрямителя

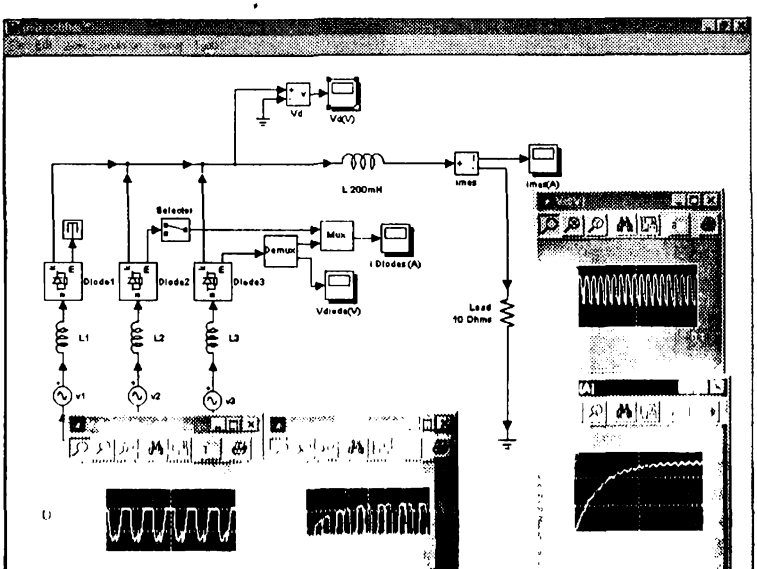


Рис. 5.46. Пример моделирования трехфазного выпрямителя

Другой пример (рис. 5.46) иллюстрирует моделирование трехфазного диодного выпрямителя с мощностью около 1,5 кВт. Нетрудно заметить, что такой выпрямитель дает почти постоянный ток в нагрузке 10 Ом с установившемся значением около 12,5 В и сравнительно небольшими пульсациями. Последнее является следствием выпрямления всех трех фаз, сдвинутых на угол 120° друг относительно друга.

Трехфазные выпрямители, благодаря их высоким энергетическим показателям и малым пульсациям выпрямленного напряжения, широко используются в промышленности. Область применения однофазных выпрямителей весьма ограничена (в основном бытовыми устройствами).

Мощный полевой транзистор с изолированным затвором

Мощный полевой транзистор с изолированным затвором в последнее время стал основным типом переключающих транзисторов малой и средней (а отчасти и большой) мощности. В пакете Power System Blockset предусмотрена простая модель мощного полевого транзистора Mosfet. Фактически он рассматривается как силовой ключ с резистором Ron и индуктивностью Lon во включенном состоянии и бесконечно большим сопротивлением в выключенном состоянии. Можно также задать включение параллельно транзистору диода, открытого при закрытом транзисторе и характеризующегося сопротивлением Rd. Наконец, можно задать включенную к выводам стока и истока цепь, содержащую последовательно соединенные конденсатор Cs и резистор Rs. Все эти параметры задаются в окне установки параметров, представленном на рис. 5.47.

Рисунок 5.48 показывает пример моделирования квазирезонансного преобразователя на мощном полевом транзисторе, переключающемся в моменты прохождения тока через нуль. Этот пример содержится в файле psbmosconv.mdl.

Переключение ключевого прибора (в нашем случае полевого транзистора) в моменты перехода тока в колебательном контуре через нуль минимизирует динамические потери в ключе и позволяет получить приемлемый коэффициент полезного действия на высоких частотах преобразования — вплоть до сотен килогерц и даже нескольких мегагерц.

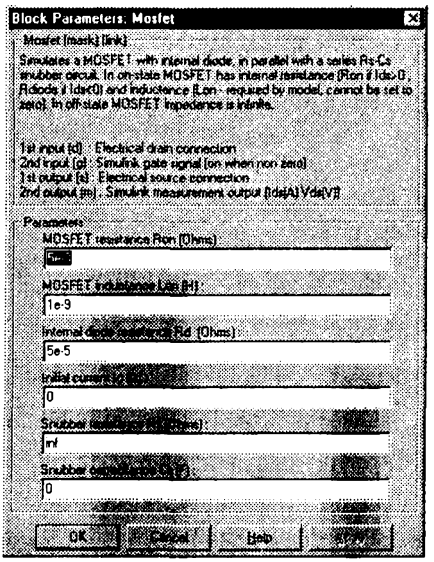


Рис. 5.47. Окно установки параметров мощного полевого транзистора

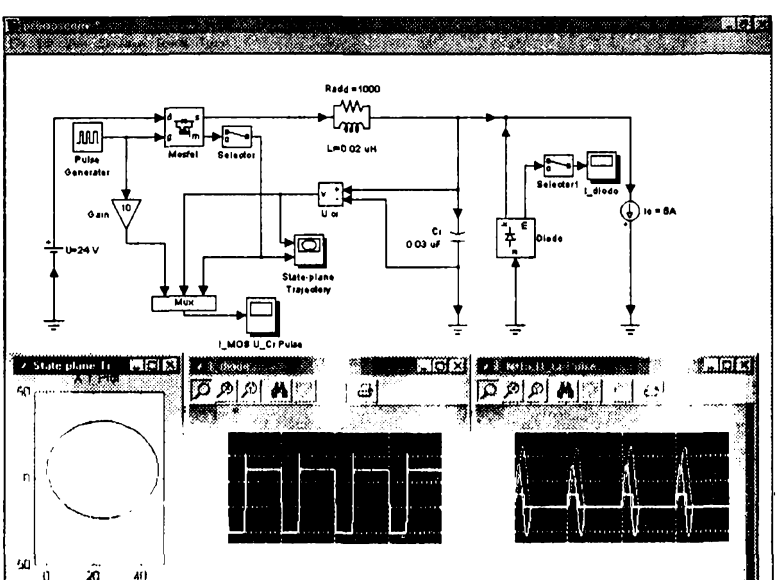


Рис. 5.48. Моделиро[illegible] инвертора напряжения на мощном полевом транзисторе

Подобные преобразователи в последние годы нашли широкое применение при построении источников электропитания без низкочастотного силового трансформатора, поскольку они обеспечивают меньший уровень генерации высших гармоник, чем источники электропитания с импульсным регулированием. Преобразователи этого типа имеют высокий КПД и малые удельные (на единицу мощности) габариты и вес.

Упрощенная модель тиристора

Модель тиристора Thyristor также построена на основе идеального ключа с элементами, имитирующими остаточные параметры включенного тиристора. Это сопротивление во включенном состоянии Ron, индуктивность Lon и падение напряжения в прямом направлении Vf. В выключенном состоянии (обратное направление) сопротивление прибора считается равным бесконечности. Тиристор выключается при управляющем сигнале, равном нулю, а также в тех случаях, когда прямой ток тиристора падает до нуля или напряжение на нем становится обратным. Предусмотрено также моделирование параллельно включенной (между анодом и катодом) последовательной RsCs-цепочки. Можно также задавать начальный ток Ic, с которого начинается моделирование (по умолчанию 0, это означает, что моделирование начинается при закрытом тиристоре).

Окно задания параметров упрощенной модели тиристора показано на рис. 5.49. Задание Lon = 0 недопустимо.

Пример модели с тиристором представлен на рис. 5.50. Обратите внимание на то, что модель тиристора имеет надпись Thyristor, что указывает на применение упрощенной модели тиристора. Моделирование выполняется при следующих параметрах: R = 1 Ом; L = 10 мГн; Ron = 0,001 Ом, Lon = 10^{-5} Гн, Vf = 0,8 В, Ic = 0 А, Rs = 20 Ом, Cs = $4 \cdot 10^{-6}$ Ф. Параметры моделирования: решатель дифференциальных уравнений ode15s; относительная погрешность Relative tolerance — 10^{-3}; абсолютная погрешность Absolute tolerance — 10^{-3}.

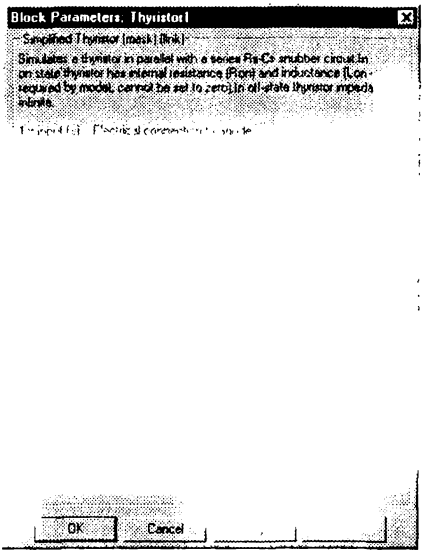


Рис. 5.49. Окно задания параметров тиристора

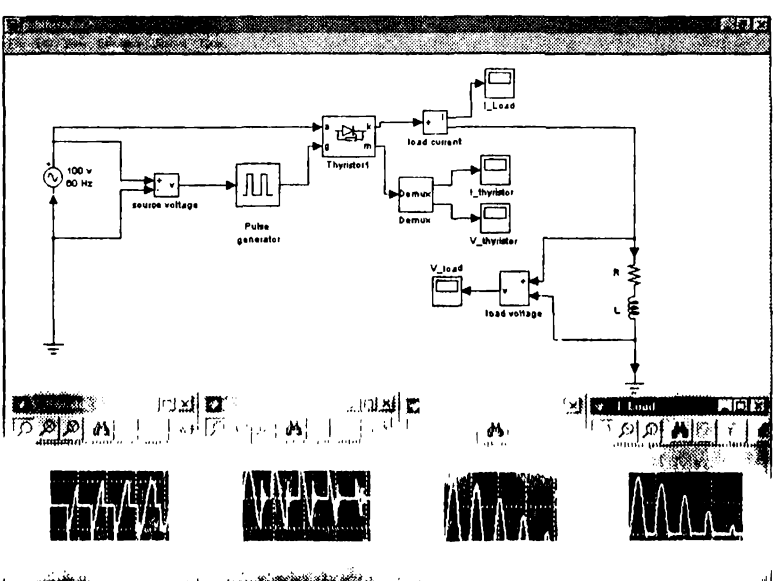


Рис. 5.50. Пример моделирования цепи с тиристором

Детальная модель тиристора

Возможна более детальная модель тиристора — Detail Thyristor. Ее окно задания параметров имеет два дополнительных параметра — ток Il, при котором тиристор выключается, и время выключения Tq (в секундах), которое характеризует задержку выключения. Эта модель позволяет более точно моделировать переходные процессы в схемах с тиристорами, что важно при моделировании устройств, работающих с повышенными частотами переключения — сотни герц и выше. Для моделирования устройств, работающих на частоте промышленной сети (50 или 60 Гц) вполне удовлетворительные результаты дает упрощенная модель тиристора.

Управляемый биполярно-полевой модуль

Биполярно-полевые силовые модули Gto — сравнительно новый и перспективный тип мощных силовых ключевых приборов. Они имеют управляющие сигналы малой мощности и способны переключать большие напряжения и токи, чем полевые транзисторы с изолированным затвором. Однако эти приборы имеют сравнительно большое время выключения.

Приборы характеризуются теми же параметрами, что и тиристоры, но у них имеются два новых специфических параметра — время Tf спада тока до уровня 0,1 от тока в момент перехода от включения к выключению и время Tt окончательного спада тока до нуля. Временная зависимость спада тока приближенно описывается двумя линейными участками с указанными длительностями.

На рис. 5.51 показана схема преобразователя постоянного напряжения, в котором используется модуль Gto (файл psbbuckconv.mdl). Осциллограммы поясняют работу этой системы. Нагрузкой преобразователя являются RL-цепь и источник постоянного напряжения E (первичным источником является источник напряжения U).

Модули Gto обычно используются в преобразовательных устройствах, работающих на частоте промышленной сети переменного тока 50 или 60 Гц. Работе на более высоких частотах препятствует отмеченная выше инерционность процесса выключения. Они также находят применение в импульсных преобразователях постоянного тока с невысокими частотами преобразования.

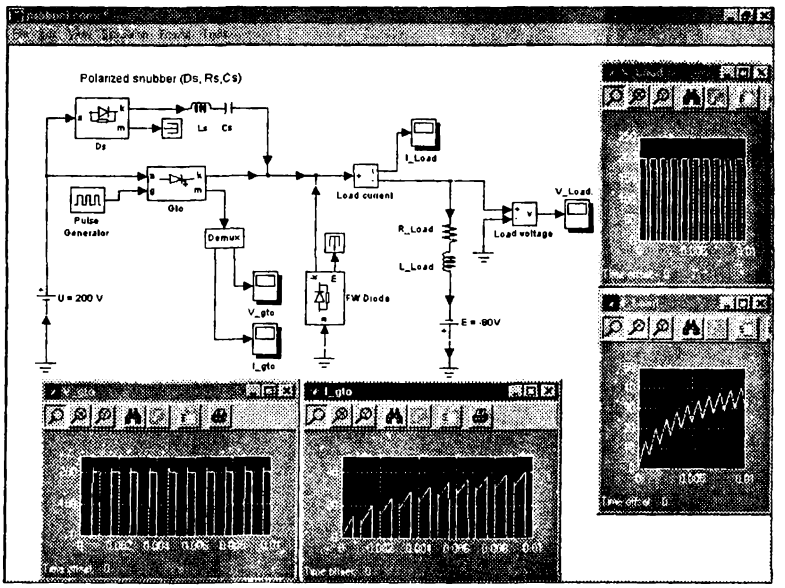


Рис. 5.51. Моделирование преобразователя постоянного напряжения с модулем Gto

Моделирование электрических машин и схем управления ими

Как уже отмечалось, библиотека пакета Power System Blockset содержит модели ряда электрических машин — постоянного тока, синхронных и асинхронных машин переменного тока. Машины могут моделироваться в режимах мотора или генератора. Это открывает возможности моделирования как самих машин, так и достаточно сложных схем управления ими (устройств электропривода). Ввиду ограниченности объема данной книги мы не будем останавливаться на деталях моделирования электрических машин (многие из них были рассмотрены выше) и опишем ряд практических примеров моделирования, вошедших в библиотеку справочных материалов по пакету Power System Blockset.

Моделирование мотора постоянного тока со стартером

Рисунок 5.52 показывает окно установки параметров двигателя постоянного тока. Ввиду очевидности этих параметров (по крайней мере для специалистов по электрическим машинам) не будем описы-

вать их подробно. Для вызова окна достаточно дважды щелкнуть на значке с изображением двигателя DC_Motor. Напоминаем, что это правило распространяется и на другие компоненты модели как описываемого здесь устройства, так и других устройств и систем.

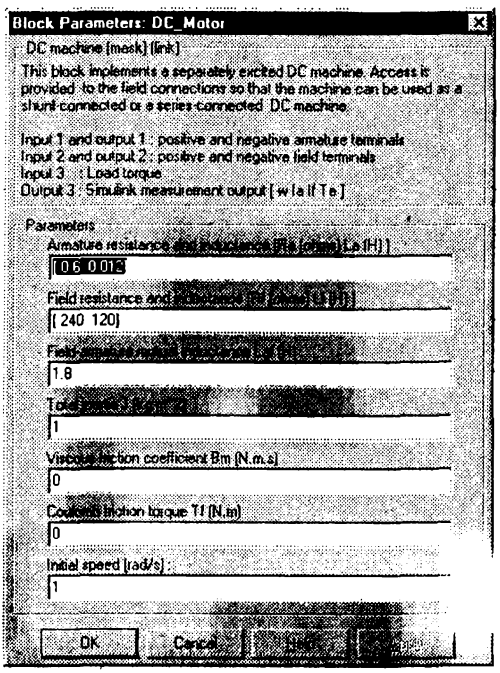


Рис. 5.52. Панель установки параметров двигателей постоянного тока

Рисунок 5.53 показывает пример моделирования двигателя (мотора) постоянного тока с использованием трехшаговой пусковой схемы — стартера. Это позволяет избежать больших перегрузок питающей сети по току, возникающих при запуске еще не раскрутившегося двигателя. Данный прием характерен для достаточно дешевых двигателей, находящих самое широкое применение в быту и на производстве.

Трехшаговый стартер не является стандартной моделью пакета Power System Blockset. Он создается системой моделирования Simulink как подсистема. Для вызова окна подсистемы (рис. 5.54) и редактирования ее достаточно дважды щелкнуть на значке подсистемы Motor Starter. Как нетрудно заметить, стартер представляет собой просто три коммутируемых во времени с помощью идеальных ключей резисто-

ра, последовательно включенных в цепь мотора. На практике для их коммутации используются обычные реле подходящей мощности.

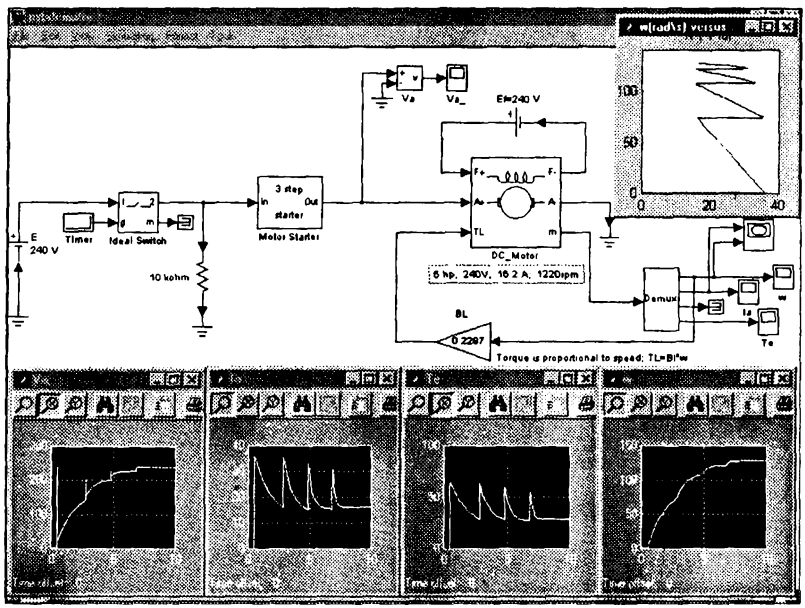


Рис. 5.53. Моделирование двигателя постоянного тока с трехшаговым стартером

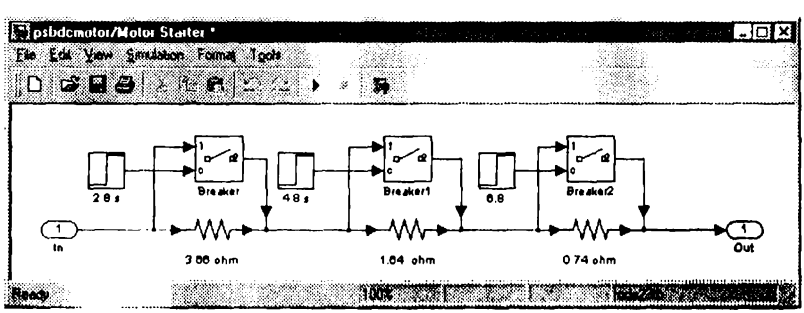


Рис. 5.54. Подсистема стартера двигателя постоянного тока

Несмотря на простоту моделируемой схемы, переходные процессы в ней достаточно сложны (см. рис. 5.53). Это типичный пример моделирования системы с переменными во времени параметрами.

Моделирование мотора постоянного тока с электронным регулятором

В более ответственных случаях для запуска двигателей постоянного тока используются электронные регуляторы с импульсным регулированием энергии, обеспечивающим ее малые потери. На рис. 5.55 показан пример моделирования двигателя постоянного тока с регулятором, построенным на модуле Gto. Как видно из осциллограмм работы данного регулятора, он обеспечивает не только быстрый разгон двигателя (с некоторым перерегулированием), но и стабилизацию скорости вращения.

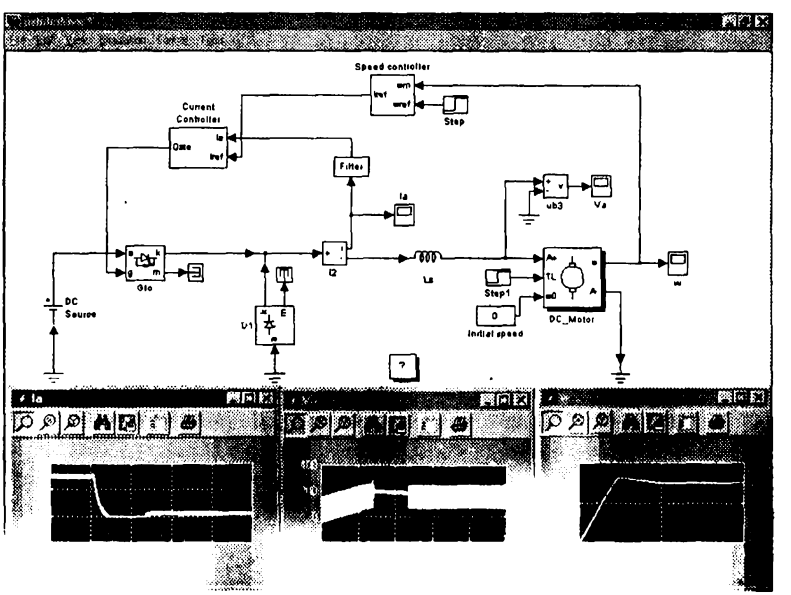


Рис. 5.55. Модель двигателя постоянного тока с электронным импульсным регулятором

Этот пример показывает возможности моделирования сложных замкнутых систем с обратными связями, содержащими столь непростые и резко нелинейные технические устройства, как мощный двигатель постоянного тока и импульсный регулятор. Высокая точность и наглядность моделирования таких устройств, несомненно, относится к достоинствам системы MATLAB и ее расширений (в нашем случае — Simulink и Power System Blockset).

Моделирование синхронных машин — генераторов

Рисунок 5.56 демонстрирует простой пример моделирования синхронной машины, работающей на трехфазную сеть переменного напряжения. Такая работа характерна для современных энергосистем.

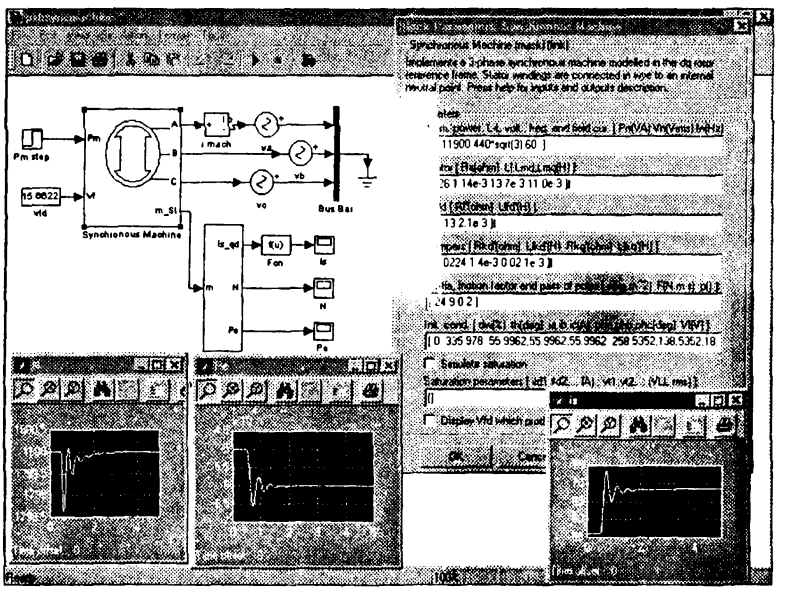


Рис. 5.56. Пример моделирования синхронной электрической машины

На рис. 5.56 помимо осциллограмм, демонстрирующих работу машины, показано окно установки параметров машины.

Еще один пример моделирования синхронной машины представлен на рис. 5.57. В этой модели используется ряд подсистем, имеющих свои модели (см. ниже). Таким образом, данная модель является многокомпонентной системой.

На рис. 5.58 показана та же модель с открытыми окнами двух подсистем — трехфазного объекта Line и подсистемы Subsystem.

Возможность задания подсистем характерна для пакета Simulink, который используется (наряду с базовой системой MATLAB) совместно с пакетом Power System Blockset. Эту возможность трудно переоценить, особенно когда целью моделирования является доработка отдельных компонентов сложных технических систем и устройств.

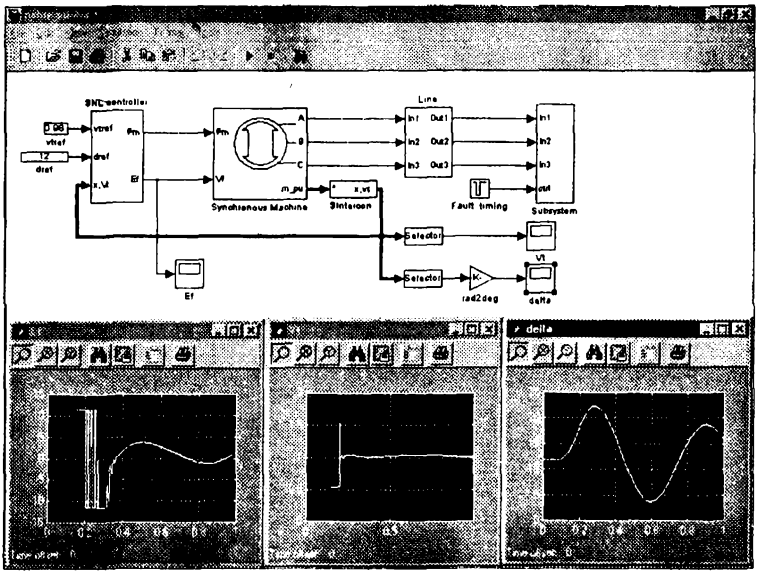


Рис. 5.57. Многокомпонентная модель синхронной машины

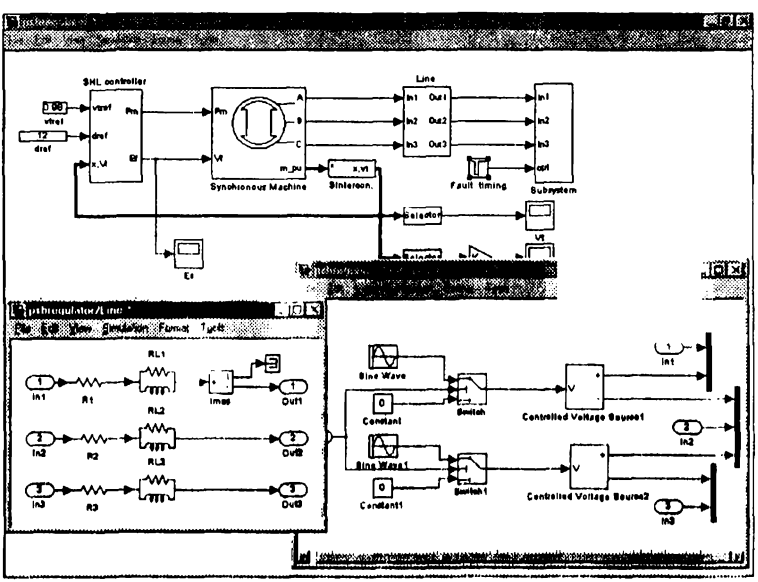


Рис. 5.58. Модель рис. 5.57 с открытыми окнами двух подсистем

Моделирование мощной синхронной машины гидравлической турбины

Рисунок 5.59 демонстрирует пример моделирования мощной синхронной машины в составе крупной электростанции с гидравлической турбиной. Машина-генератор выдает электроэнергию в трехфазную высоковольтную сеть (13,8 кВ, 210 МВт). С помощью трансформатора напряжение сети повышается до 230 кВ. С подсистемами этой модели заинтересованный читатель может ознакомиться самостоятельно.

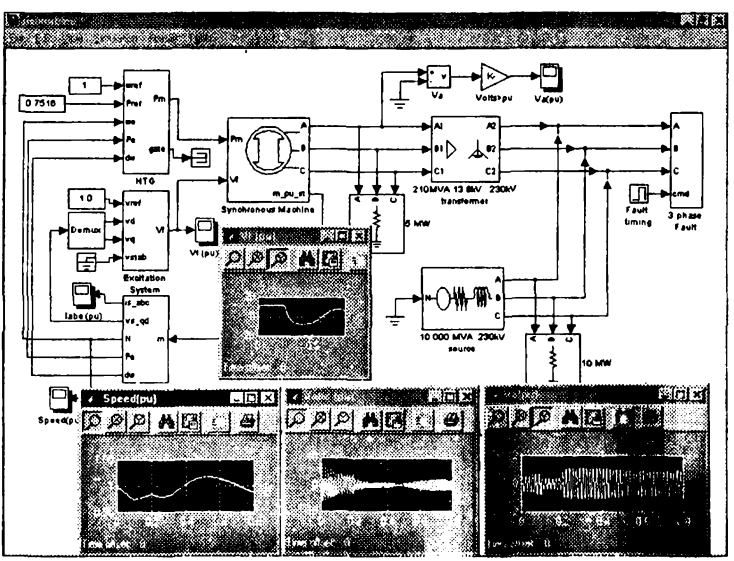


Рис. 5.59. Пример моделирования мощной синхронной машины

Приведенные осциллограммы наглядно иллюстрируют сложные процессы работы синхронной машины.

Моделирование синхронной машины — двигателя

Еще один пример моделирования синхронной машины — двигателя с магнитом представлен на рис. 5.60. Осциллограммы изменения во времени ряда параметров системы демонстрируют весьма сложный характер переходных процессов.

Подсистемы этой машины и окно установки ее параметров представлены на рис. 5.61.

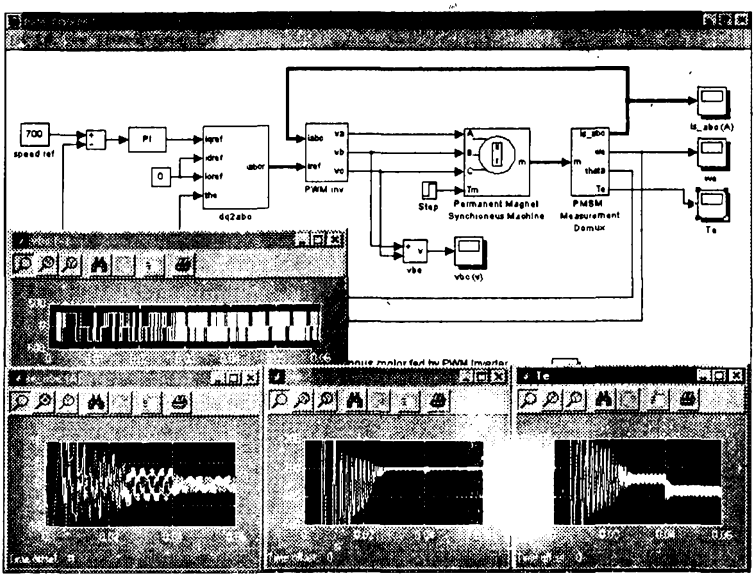


Рис. 5.60. Пример моделирования-синхронной машины – двигателя с магнитом

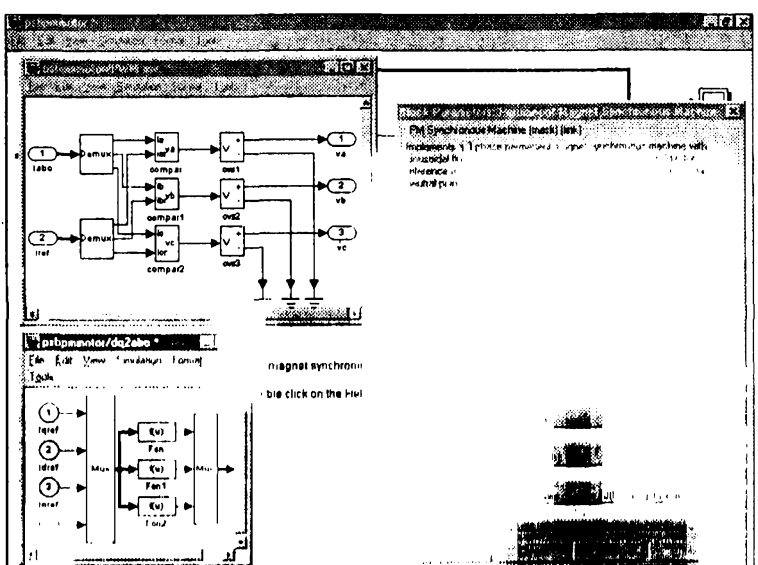


Рис. 5.61. Подсистемы и окно установки параметров для синхронной машины с маг

Приведенные примеры наглядно показывают возможности моделирования с помощью пакета Power System Blockset синхронных машин различного типа с системами их привода, что открывает возможности детального изучения этих сложных устройств без привлечения к этому натурных испытаний или, по крайней мере, с уменьшением их продолжительности.

Моделирование асинхронных машин

Асинхронные машины получили широкое распространение — в основном как двигатели для электрического привода. Это связано с их неприхотливостью и простотой. Такие машины получают питание от импульсных преобразователей или от промышленной сети переменного тока.

Пример моделирования асинхронного двигателя, питаемого от трехфазной сети импульсов, созданной импульсным преобразователем релейного типа (часть схемы, размещенная под моделью машины), показан на рис. 5.62. Осциллограммы характеризуют работу системы на этапе разгона двигателя.

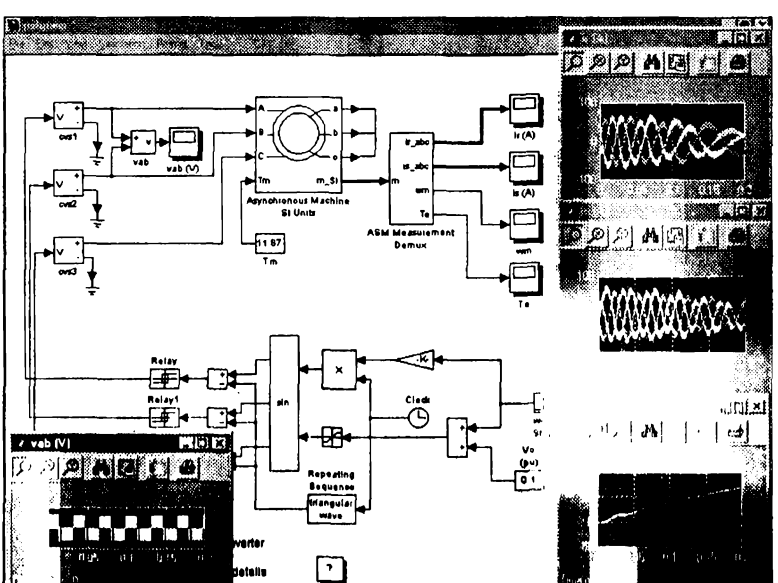


Рис. 5.62. Моделирование привода асинхронного двигателя, питаемого от трехфазного импульсного источника

На рис. 5.63 показана та же модель с открытыми окнами установки параметров асинхронной машины и выбора измеряемых параметров. Последнее окно дает представление о системе параметров асинхронной машины, доступных для регистрации с помощью виртуальных измерительных приборов.

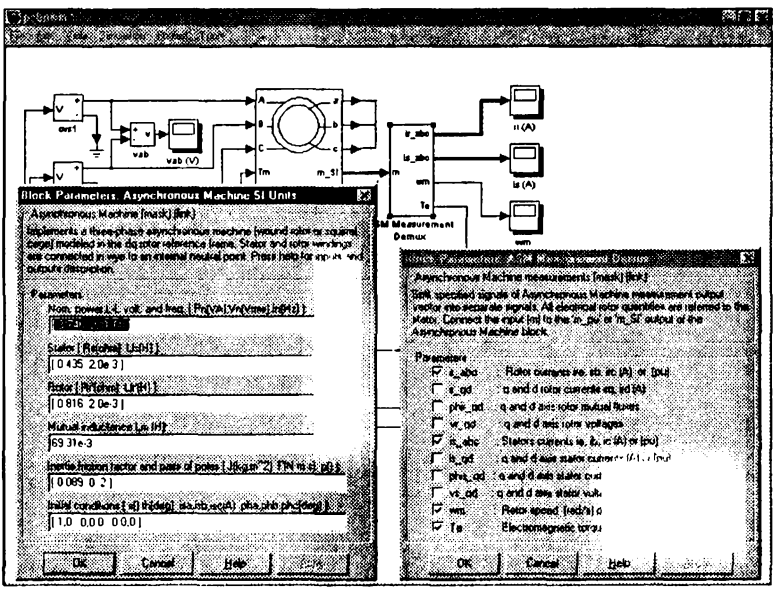


Рис. 5.63. Модель привода асинхронного двигателя с открытыми окнами установки параметров асинхронной машины и выбора измеряемых параметров

Следует отметить, что асинхронные машины в пакете Power System Blockset имеют две модели, названные pu Units 1 (с трехфазным входом) и pu Units (со входом, имеющим две фазы с нейтралью).

Применение графического интерфейса пользователя

В одном из примеров (см. рис. 5.19 и 5.20) мы уже отмечали возможность применения средств графического интерфейса пользователя (GUI) для управления процессом моделирования системы или устройства. Эта возможность существует не только в данном примере, но и во всех других — как входящих в библиотеку, так и состав-

ляемых пользователем. Для использования средств GUI достаточно встроить блок GUI в любую модель.

Подключение GUI к модели системы или устройства

Чтобы подключить средства GUI к конкретной модели системы или устройства, надо при загруженной (или составленной) модели открыть окно броузера библиотек. В разделе Power System Blockset этого окна можно найти компонент с именем powergui. Этот компонент надо перетащить мышью и разместить в окне выбранной модели на любом свободном месте. Рисунок 5.64 показывает, как это делается, на примере модели резонансного преобразователя (конвертора) на мощном полевом транзисторе.

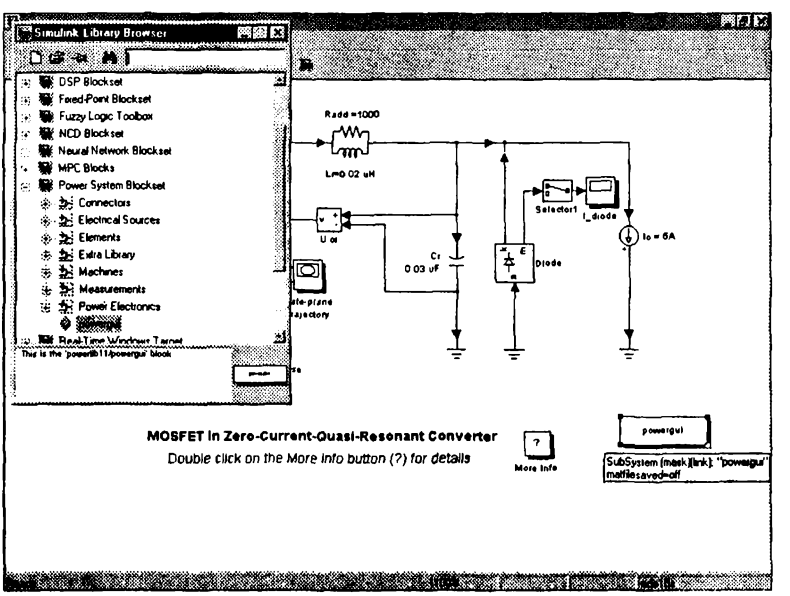


Рис. 5.64. Установка компонента powergui в окно выбранной модели

Блок powergui виден в правом нижнем углу рис. 5.64 вместе с всплывающей подсказкой, появляющейся при установке на него указателя мыши. После этого активизация компонента powergui (как всегда, двойным щелчком левой кнопки мыши) вызывает появление небольшого окна с меню GUI. Это окно показано внизу рис. 5.65 в окне выбранной модели.

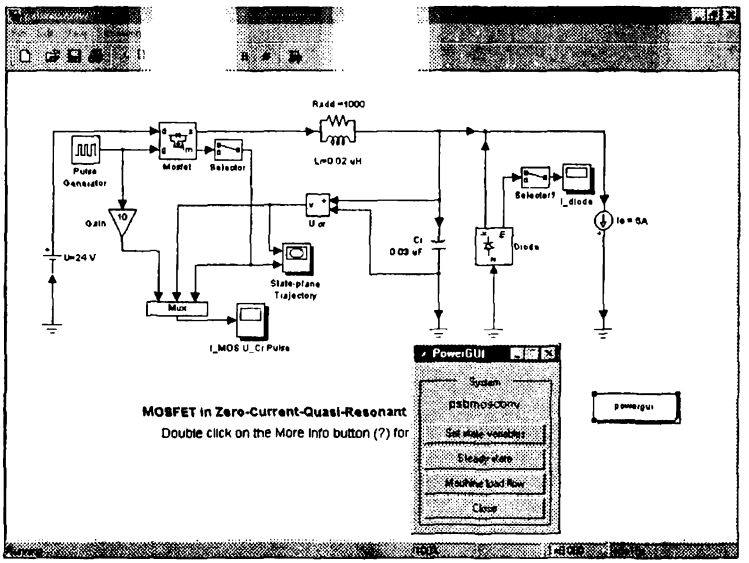


Рис. 5.65. Модель с окном компонента powergui

Управление моделью системы с помощью GUI

Как видно из рис. 5.65, окно GUI имеет ряд кнопок, позволяющих контролировать ряд параметров модели. На рис. 5.66 демонстрируется возможность контроля значений переменных с помощью окна State Variables, появляющегося при щелчке на кнопке Set State Variables (Установка значений переменных). Изменение начальных значений переменных позволяет выполнить расчет цепи при этих значениях.

В этом окне можно наблюдать за значениями переменных. Кнопка Reset to zero позволяет установить нулевые значения переменных. Поле Set to позволяет задать для выделенной переменной произвольное значение. Для этого, введя это значение, нужно нажать кнопку Apply (Применить). Кнопка Revert возвращает исходное значение переменной.

Активизация кнопки Steady State (Установившееся состояние) позволяет оценить параметры установившегося режима моделируемой системы или устройства. При этом выводится окно, показанное в левой части рис. 5.67. В этом окне указаны напряжения и токи источников сигналов, показания виртуальных измерительных приборов, данные установившегося режима, а также напряжения и токи нелинейных компонентов системы или устройства.

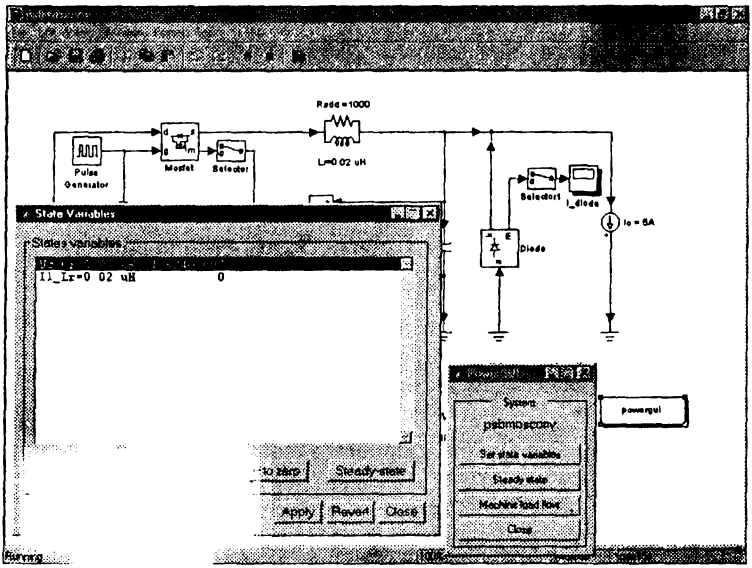


Рис. 5.66. Контроль за значениями переменных модели

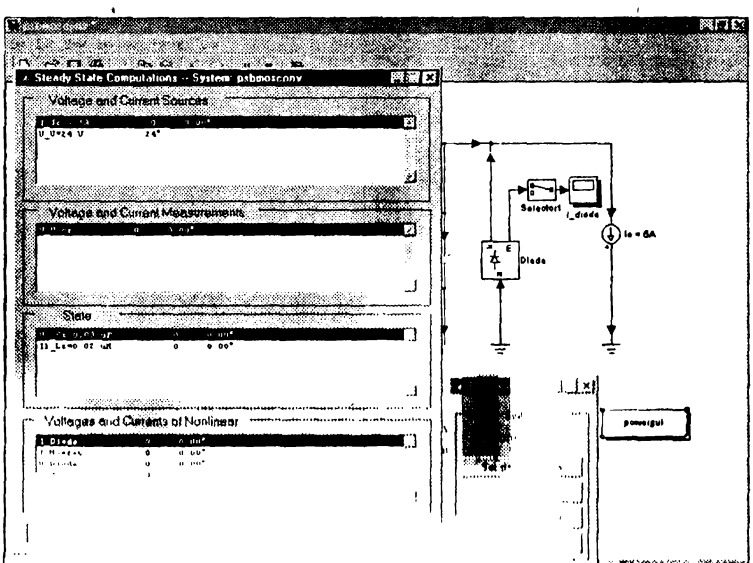


Рис. 5.67. Контроль установивш[illegible] состояния системы

Кнопка Machine load flow служит для контроля параметров пуска электрических машин. Если таковых в составе модели нет, то выводится окно с сообщением об этом (см. рис. 5.19 — окно внизу слева). При наличии электрической машины в составе модели выводится окно с параметрами машины (рис. 5.68).

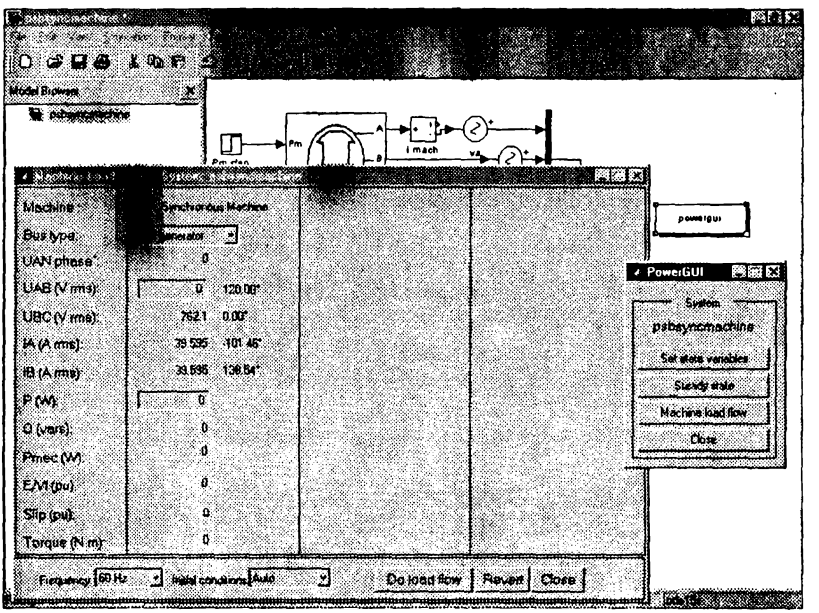


Рис. 5.68. Окно вывода параметров пуска электрической машины

Кнопка Close закрывает окно PowerGUI. Использование средств GUI облегчает управление состоянием системы или устройства и позволяет обеспечить различные режимы моделирования.

Вывод краткого описания модели

К средствам GUI можно отнести также компонент в виде вопросительного знака и надписи More Info (Дополнительная информация). Этот компонент имеется в ряде демонстрационных примеров и служит для вывода развернутого наименования модели (рис. 5.69).

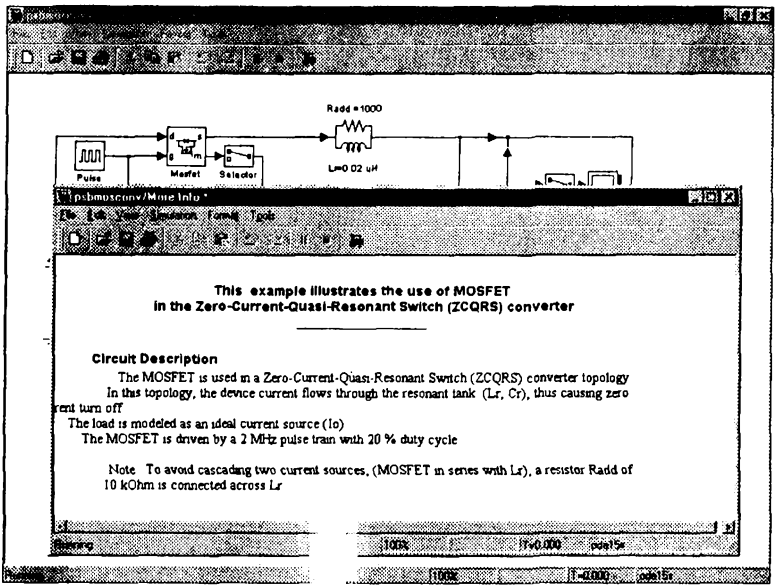


Рис. 5.69. Пример вывода дополнительной информации о модели

Дополнительные примеры моделирования энергетических систем и устройств

Моделирование линии передачи электроэнергии с компенсаторами

Рисунок 5.70 показывает пример моделирования переходных процессов в высоковольтной линии передачи электроэнергии с напряжением 735 кВ, имеющей два участка длиной по 150 км каждый и индуктивные компенсаторы.

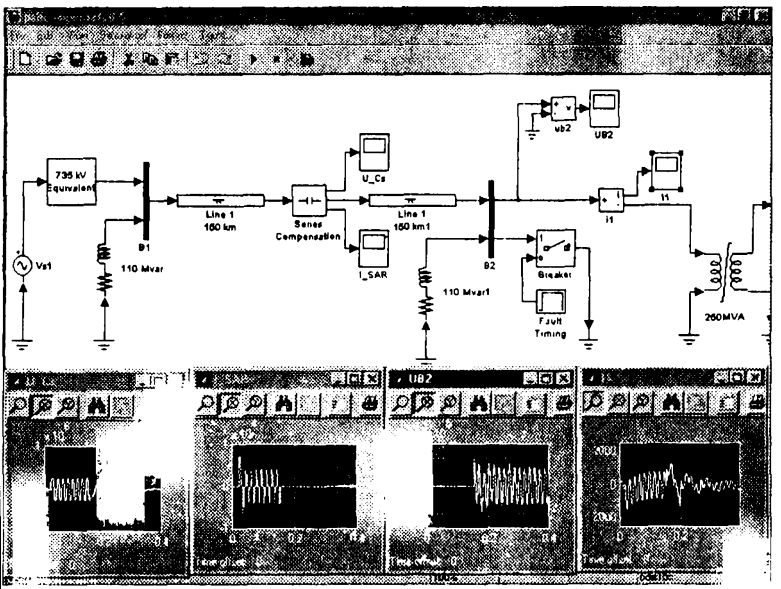


Рис. 5.70. Моделирование линии электропередачи с компенсаторами

Сравнение двух типов однофазных линий передачи электроэнергии

Нередко бывает необходимо провести сравнение различных вариантов технических решений. Рисунок 5.71 показывает модель для сравнительной оценки двух однофазных линий передачи — с сосредоточенными и распределенными параметрами. Обе линии выбраны с равной длиной в 200 км. С помощью выключателей производится кратковременное отключение линий от источника переменного тока с напряжением 1 кВ.

Сравнение переходных процессов для линий двух типов показывает, что хотя разница и есть, но назвать ее существенной в данном случае трудно.

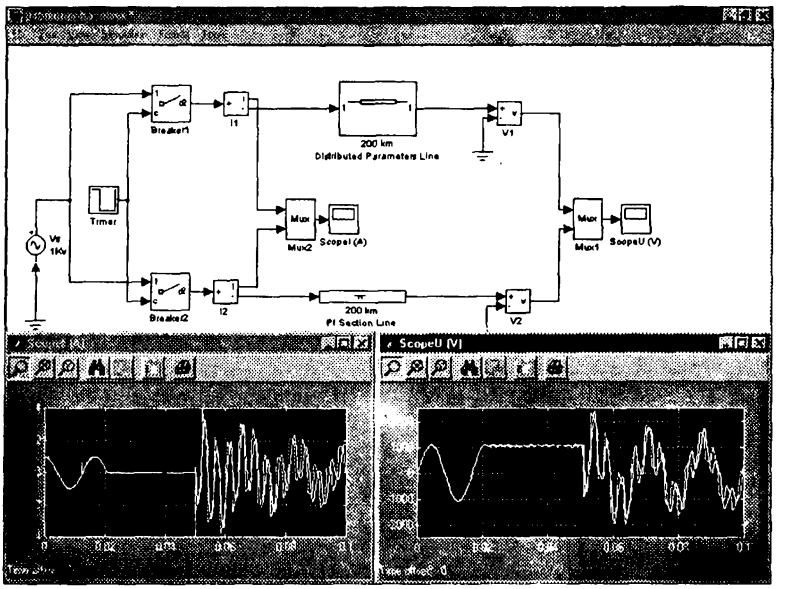


Рис. 5.71. Сравнение двух типов однофазных линий передачи

Подготовка субмодели ШИМ

Описанная выше модель преобразователя (см. рис. 5.48) идеализирована и не содержит элементов автоматического управления. Обычно в преобразователях такого рода для управления ключом используется широтно-импульсный модулятор (ШИМ). Рисунок 5.72 показывает одну из реализаций подобного ШИМ, а также управляющий треугольный сигнал на его входе и выходные импульсы модулятора.

Эта модель построена с применением средств пакета Simulink. Обратите внимание на то, что все пояснения даны на русском языке, поскольку модель создавалась заново и не является встроенной в библиотеку примеров системы Simulink.

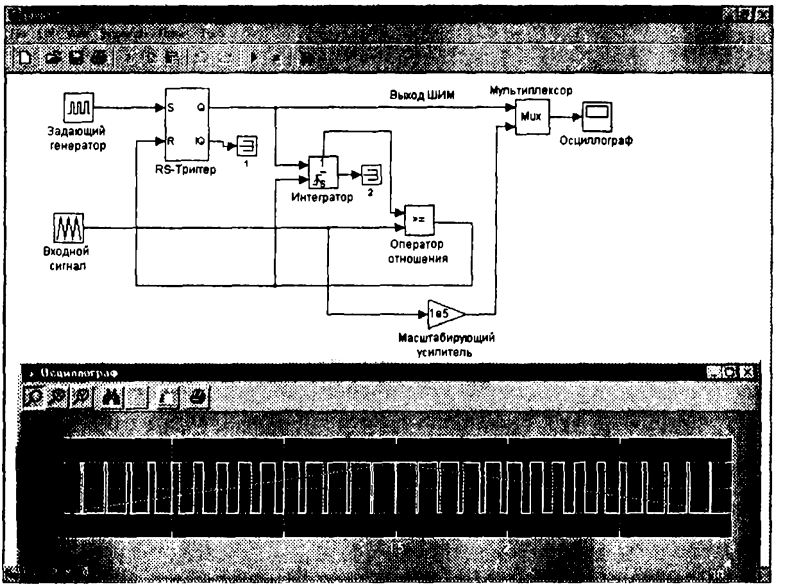


Рис. 5.72. Подсистема ШИМ

Моделирование преобразователя с ключом на полевом транзисторе, управляемым от ШИМ

На рис. 5.73 показана модель импульсного преобразователя с идеальным ключом, управляемым с выхода ШИМ. Этот преобразователь является типичной замкнутой системой регулирования, которая стремится установить выходное напряжение равным опорному напряжению стандартной бортовой сети 27 В (блок 3). Если выходное напряжение превышает опорное, скважность импульсов ШИМ уменьшается и ключевой регулятор понижает напряжение на выходе.

Осциллограмма выходного напряжения, представленная на рис. 5.73, показывает, что сначала имеет место заметное перерегулирование, дающее короткий всплеск напряжения на выходе примерно до 46 В. Затем напряжение на выходе опускается до уровня примерно 27 В и пульсирует около него с частотой, равной частоте модуляции. Поведение данной системы вполне характерно для систем такого рода и свидетельствует о желательности применения дополнительных мер по коррекции динамических процессов при запуске подобного устройства.

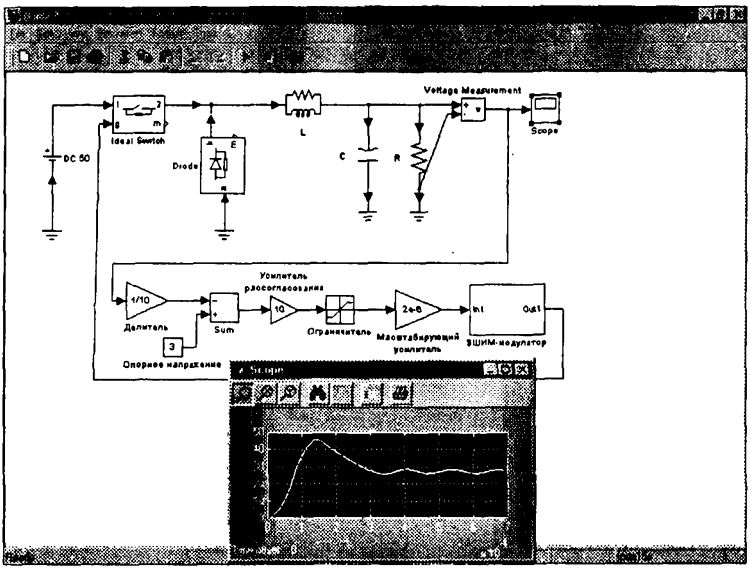


Рис. 5.73. Моделирование преобразователя с идеальным ключом, управляемым от ШИМ

Следует отметить, что затраты времени на моделирование преобразовательных устройств быстро растут по мере усложнения модели. Типовая схема современного бестрансформаторного источника вторичного электропитания от сети 220 В 50 Гц (такие схемы применяются, например, в источниках питания компьютеров) может потребовать полного времени моделирования (до выхода на стационарный режим) порядка часа даже при использовании компьютеров с процессорами класса Pentium II и Pentium III. Однако, учитывая дешевизну работы компьютера, даже такое время моделирования следует признать вполне приемлемым (известны случаи, когда моделирование сложных систем занимало многие дни и даже месяцы, оставаясь вполне рентабельным делом).

Дополнительные возможности пакета Power System Blockset

Функция power2sys

Для детального анализа моделируемых энергетических систем и электрических цепей в MATLAB имеется специальная функция power2sys. Синтаксически эта функция записывается в следующем виде:

```
POWER2SYS('simwin');
[A,B,C,D,x0, state_var ,inputs,outputs,uss,xss,yss,freqyss,Hlin]=
    POWER2SYS('simwin');
[A,B,C,D,x0,state_var,inputs,outputs,uss,xss,yss,freqyss,Hlin]=
    POWER2SYS('simwin','n');
```

Здесь `simwin` — имя файла моделируемой системы. Эта функция рассматривает линейную часть модели анализируемой схемы и создает ее описание. Оно базируется на следующем матричном уравнении:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u},$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}.$$

Здесь **A**, **B**, **C** и **D** — матрицы, которые формирует Simulink при составлении матричной модели анализируемой схемы. Смысл этих и других параметров поясним примером, представленным ниже.

Запустим схему из файла psbnetsim2.mdl. Это можно сделать, указав имя файла прямо в командной строке MATLAB или загрузив файл с помощью команды Open меню окна Simulink. Соответствующая схема (модель) появится в окне Simulink (рис. 5.74). Запустив модель на исполнение, получим осциллограмму напряжения на RL-цепи, возникающего при включении ключа.

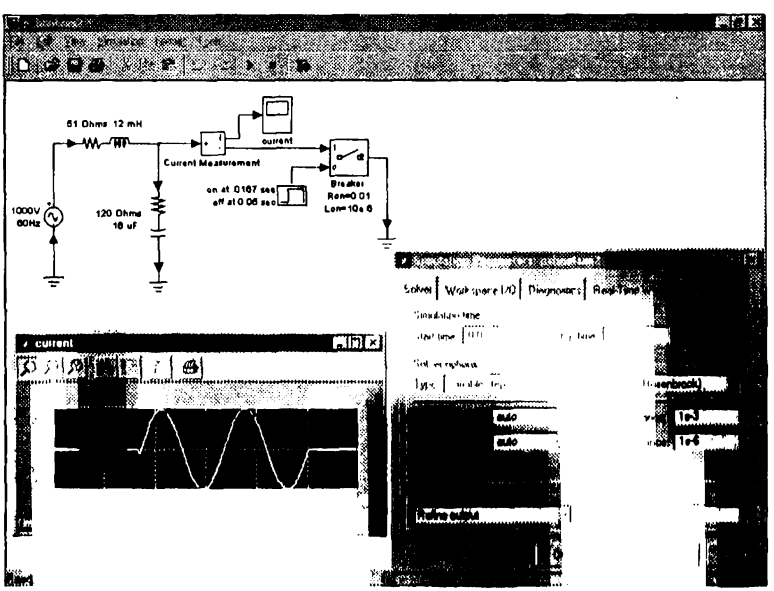


Рис. 5.74. Модель из файла psbnetsim2.mdl

Теперь, перейдя в окно MATLAB, исполним команду power2sys:

```
»[A,B,C,D,x0,state_var,inputs,outputs,uss,xss,yss,freqyss,Hlin]=
 POWER2SYS(' psbnetsim2')
```

Появятся данные о схеме (точнее, о ее линейной части) в следующем виде:

```
Power System Blockset processing psbnetsim2 ...
Computing state-space representation of linear electrical circuit...
(2 states : 2 inputs : 2 outputs)
A =
 1.0e+004 *
   0 6.2500
 -0.0083 -1.4250
B =
 1.0e+004 *
   0 -6.2500
 0.0083 1.0000
C =
  1 120
  0  0
D =
  0 -120
  0  1
x0 =
 -513.1443
 2.9190
state_var =
Uc_120 Ohms 16 uF
Il_51 Ohms 12 mH
inputs =
U_1000V 60Hz
I_Breaker Ron=0.01 Lon=10e-6
outputs =
U_Breaker Ron=0.01 Lon=10e-6
I_Current Measurement
uss =
  1000
   0
xss =
 1.0e+002 *
 4.8392 - 5.1314i
 0.0310 + 0.0292i
yss =
 1.0e+002 *
 8.5535 - 1.6287i
  0
freqyss =
```

```
 60
Hlin =
 0.8553 - 0.1629i  -44.3596 + 4.4368i
 0                   1.0000
```

Эти данные вряд ли представят интерес для большинства пользователей, которым важен лишь конечный результат моделирования. Однако читатели, знакомые с техникой моделирования систем, найдут в них немало интересного, поскольку эти результаты отражают не столько внешние, сколько внутренние аспекты процесса моделирования. Помимо указанной функции в пакет Power System Blockset входит функция `circ2ss`, готовящая статическую модель анализируемой системы. Эта функция вызывается также из командной строки MATLAB. Ввиду ограниченного ее применения детально она не рассматривается, подробное описание этой функции можно найти в справочной системе пакета Power System Blockset.

Анализ цепей в частотной области

Мы уже отмечали возможность анализа цепей в частотной области. Показанная на рис. 5.75 модель дает еще один пример частотного анализа электрической цепи. Она реализуется с помощью компонента Frequency Analysis Z(f).

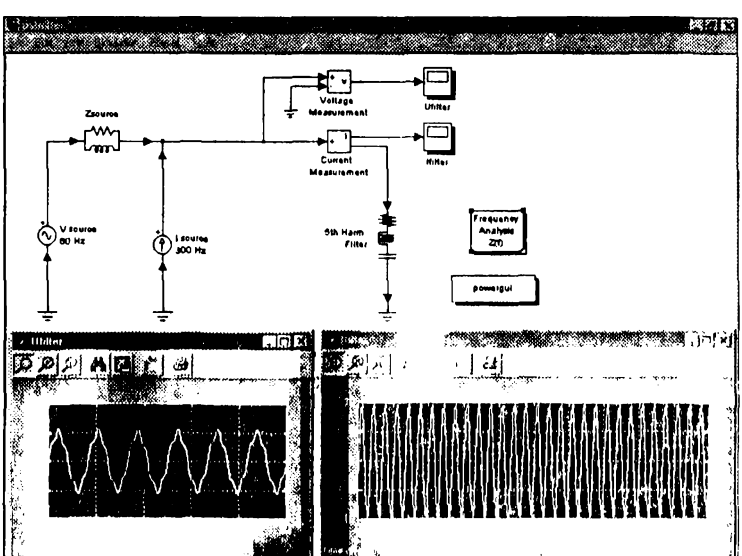


Рис. 5.75. Модель цепи и временные диаграммы ее работы

Активизируя значок Frequency Analysis Z(f), можно наблюдать окно с амплитудно-частотной (АЧХ) и фазочастотной (ФЧХ) характеристиками анализируемой цепи (рис. 5.76). Там же показано окно свойств компонента Frequency Analysis Z(f).

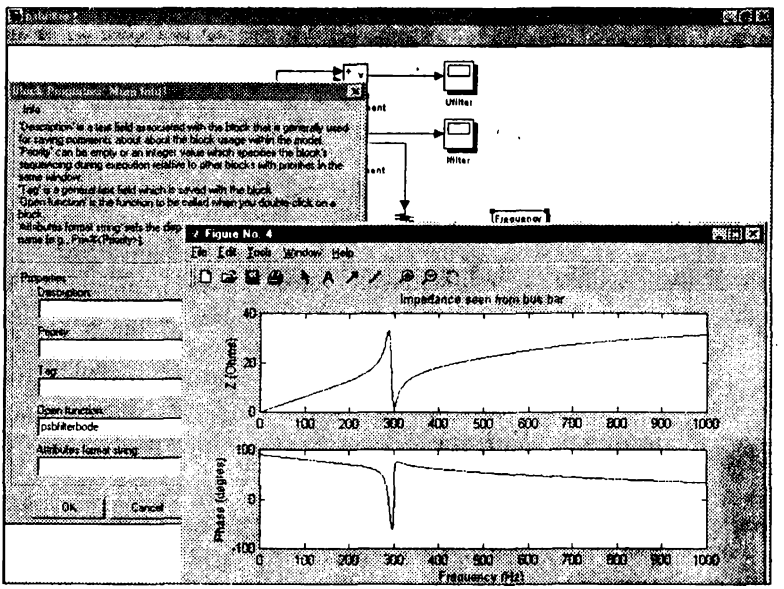


Рис. 5.76. Результаты частотного анализа цепи рис. 5.75

Анализ цепей во временной области

Анализ цепей во временной области — самая распространенная задача, решаемая средствами пакетов Simulink и Power System Blockset. Выше приводилось множество примеров такого анализа. Еще один пример, представленный на рис. 5.77, демонстрирует средства пакета Power System Blockset, предназначенные для этого вида моделирования. Здесь моделируется переходный процесс в достаточно сложной цепи, содержащей ряд реактивных элементов и находящейся под воздействием источника переменного напряжения.

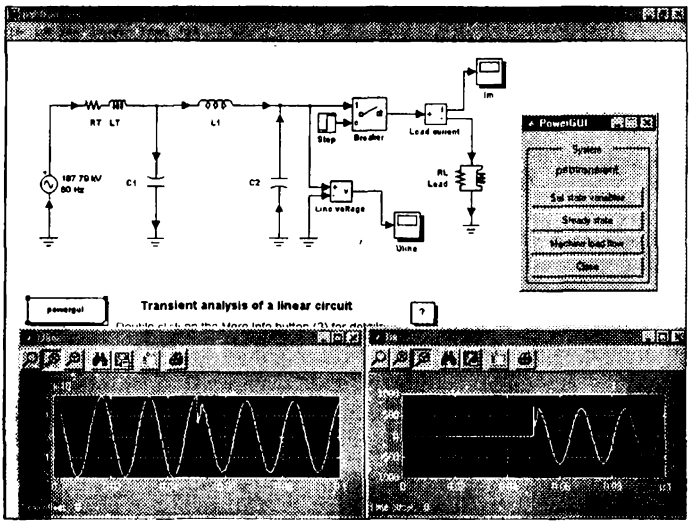


Рис. 5.77. Пример анализа цепи во временной области

Дополнительные библиотеки

Внимательный читатель, возможно, подметил, что выше были приведены примеры моделирования электрических машин постоянного тока, хотя в библиотеке они не были указаны (точнее, была указана лишь модель машины со встроенными магнитами). Эти машины, как и многие другие дополнительные модели, входят в особый раздел библиотек — Extra Library. Набор компонентов, входящих в этот раздел, представлен на рис. 5.78 в окне броузера библиотек.

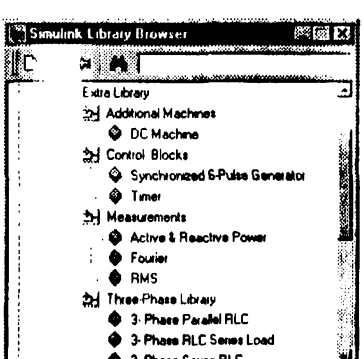


Рис. 5.78. Набор компонентов дополнительных библиотек

На рис. 5.79 представлены окна библиотек раздела Extra Library. В них даны обозначения дополнительных компонентов этого раздела. В отличие от основных библиотек, в разделах Extra Library нет ссылок на демонстрационные примеры. Более того, описание этого блока в документации отсутствует. В этой главе, однако, мы дадим его, хотя и выборочно.

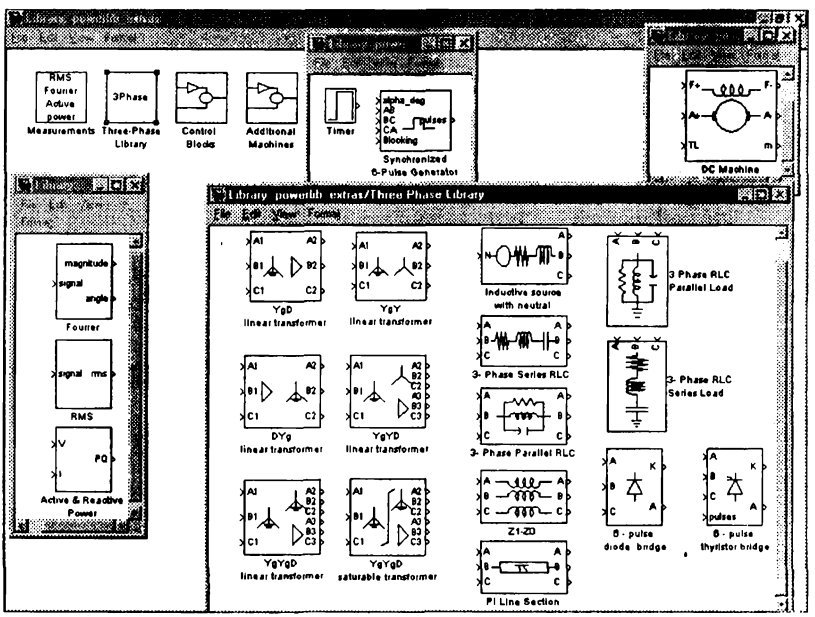


Рис. 5.79. Состав компонентов дополнительных библиотек

Нетрудно заметить, что наиболее обширным является раздел библиотеки, посвященный трехфазным системам и устройствам. Это и понятно — такие устройства получили в промышленности очень широкое распространение.

Моделирование трехфазных линий передачи

В качестве примера применения средств дополнительной библиотеки рассмотрим моделирование переходных процессов в трехфазных линиях с сосредоточенными и распределенными параметрами. Этот пример показан на рис. 5.80.

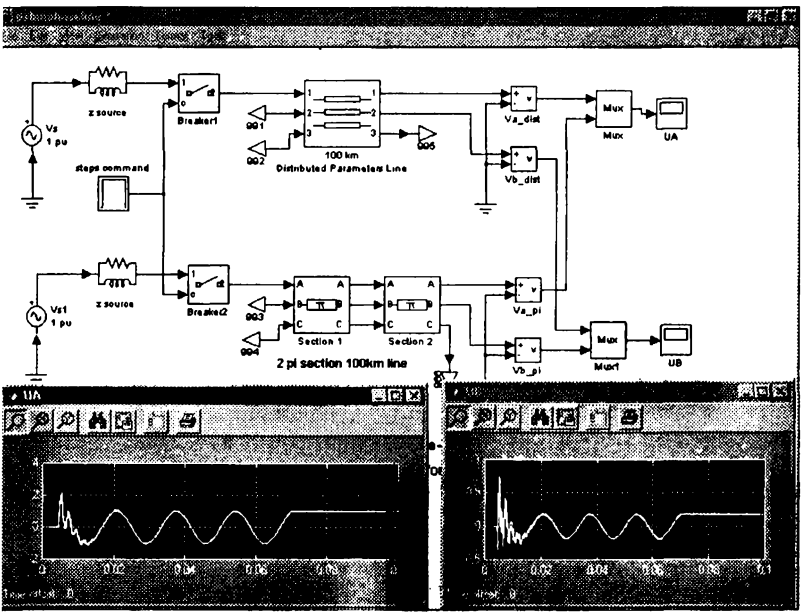


Рис. 5.80. Сравнение переходных процессов в трехфазных линиях разного типа

В этом примере моделируется трехфазная линия длиной 100 км с напряжением 735 кВ при мощности 30 000 МВА. Сравниваются два варианта линии — линия из двух секций с сосредоточенными параметрами и линия с сосредоточенными параметрами.

Моделирование сложной энергетической системы

В заключение рассмотрим моделирование сложной энергетической системы, показанной на рис. 5.81. Эта система содержит эквивалент мощного трехфазного источника (315 кВ, 5000 МВА), мощный реактивный фильтр, 6-фазный тиристорный преобразователь, модули постоянного тока и ряд других узлов. Набор средств соответствует мощной современной электростанции. Представленные осциллограммы свидетельствуют о сложности переходных процессов в этой системе.

Рисунок 5.82 показывает одну из подсистем моделируемой энергосистемы. Это мощный трехфазный реактивный фильтр. В фильтре использован целый ряд трехфазных компонентов из библиотеки Extra Library.

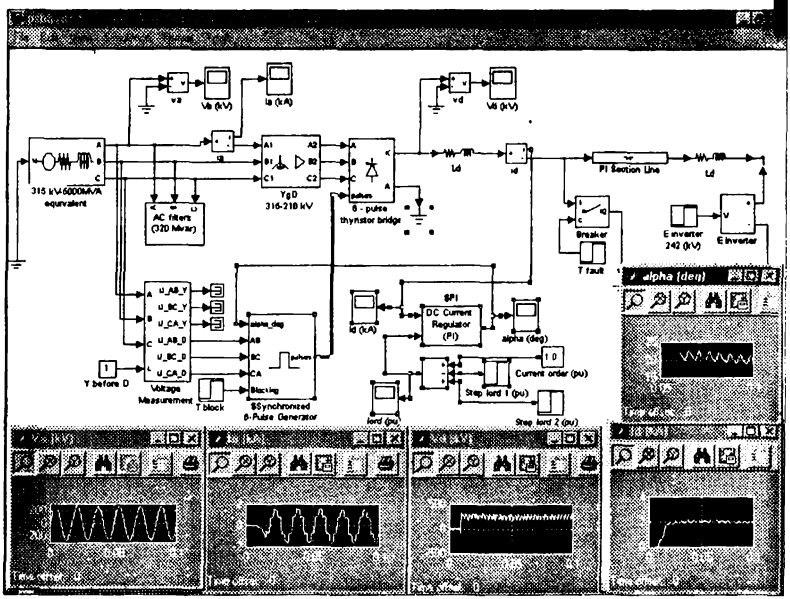


Рис. 5.81. Моделирование сложной энергосистемы большой мощности

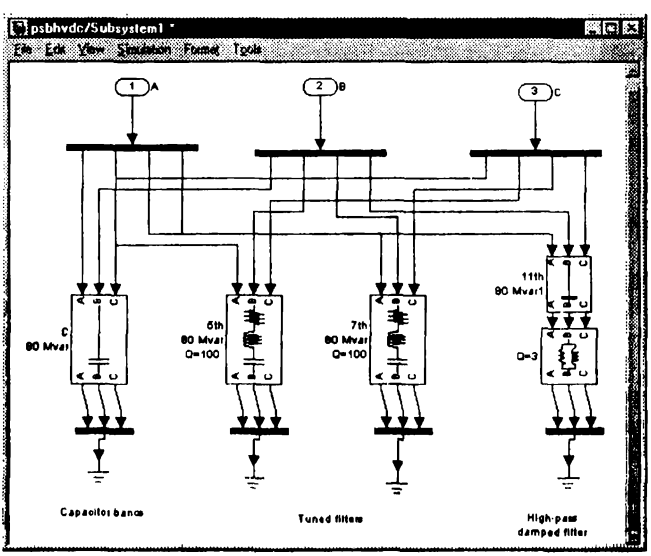


Рис. 5.82. Подсистема мощного реактивного фильтра

Еще одна подсистема — трехфазного измерительного блока — представлена на рис. 5.83. Для иллюстрации возможности установки параметров блока на рис. 5.83 показаны также два окна установки параметров для низкодобротной (Q = 2) корректирующей цепи и масштабирующего устройства.

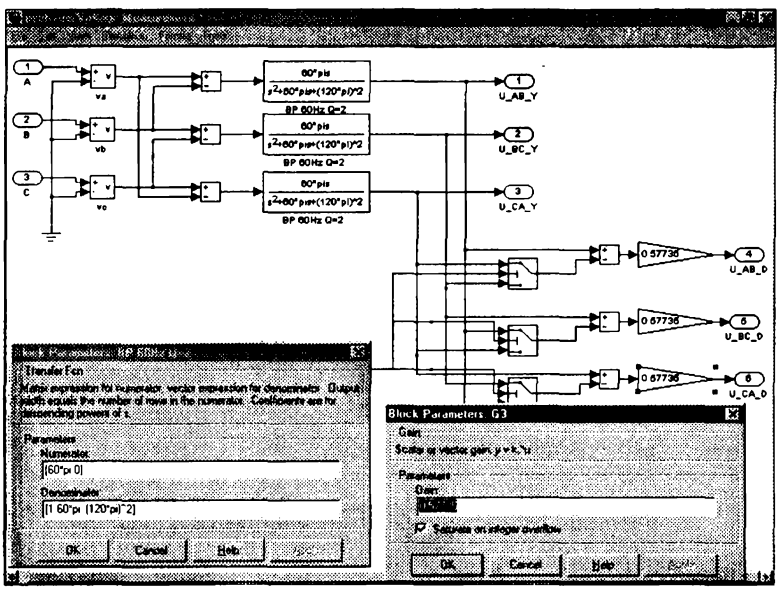


Рис. 5.83. Подсистема трехфазного измерительного блока

Еще одна подсистема — $PI — представлена на рис. 5.84. Это достаточно сложная подсистема, имеющая в своем составе интегратор, масштабирующий усилитель и ряд других систем. Окна настройки трех компонентов этой подсистемы также показаны на рис. 5.84.

Таким образом, система рис. 5.81 содержит множество разнохарактерных и взаимодействующих друг с другом устройств, что сильно затрудняет анализ переходных процессов в ней. При настройке по умолчанию моделирование системы оказалось невозможным и потребовало смены метода решения системы дифференциальных уравнений, описывающих ее работу. Смена метода осуществляется с помощью окна установки параметров системы Simulink. Рисунок 5.85 показывает это окно с раскрытым списком возможных методов решения систем дифференциальных уравнений, описывающих работу моделируемой энергосистемы.

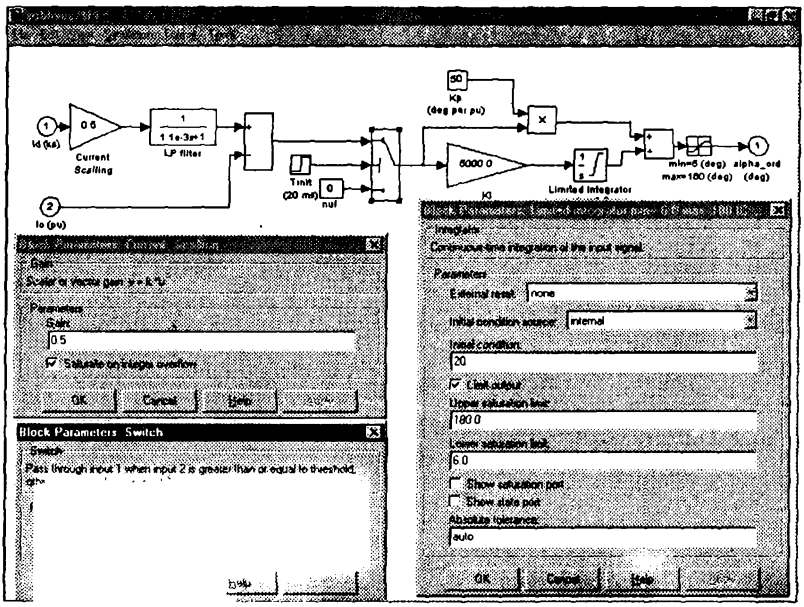


Рис. 5.84. Подсистема $PI

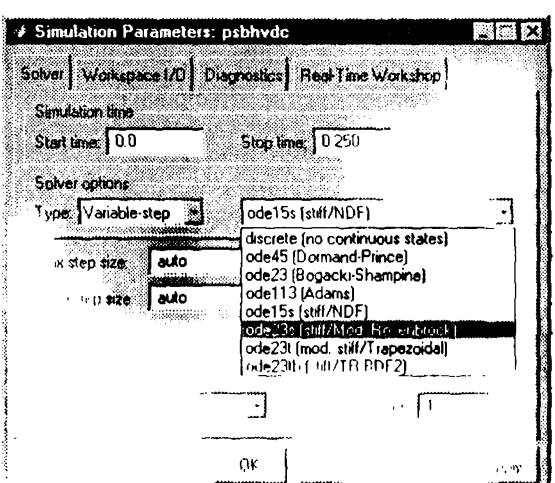


Рис. 5.85. Выбор метода решения системы ДУ моделируемой энергосистемы

Результаты моделирования, представленные на рис. 5.81, были получены только после установки метода `ode23f`, представляющего собой модифицированный метод Розенброка для решения жестких систем дифференциальных уравнений. Этот пример хорошо иллюстрирует необходимость правильного выбора метода решения при моделировании как электронных, так и любых других устройств и систем.

Глава 6
Пакет Communications Blockset

Назначение пакета и доступ к его ресурсам

Назначение пакета

Пакет Communications Blockset — один из самых крупных пакетов расширения системы MATLAB, предназначенных для исследования, моделирования и проектирования коммуникационных систем и устройств. Пакет представлен обширным набором моделей, включающих в себя описания важнейших коммуникационных элементов: источников сигналов с различными законами распределения, квантователей, модуляторов, демодуляторов, виртуальных устройств индикации и многих других устройств.

Пакет рассматривается как часть среды моделирования Simulink. Он также использует средства ядра базовой матричной системы MATLAB и допускает совместную работу с функциями (компонентами), входящими в другие пакеты расширения системы MATLAB, в том числе описанные выше. Все это делает возможности системы MATLAB в области моделирования коммуникационных и связных устройств поистине уникальными и удивительными.

Вместе с пакетом Communication Blockset могут использоваться многие другие пакеты расширения:

- Signal Processing Toolbox — пакет обработки и моделирования сигналов;
- DSP Blockset — пакет обработки и моделирования цифровых систем и сигналов;
- Quantized Filtering Toolbox — пакет моделирования и разработки цифровых фильтров;
- Image Processing Toolbox — пакет обработки изображений.

Применение всех этих пакетов открывает огромные возможности в моделировании и проектировании самых разнообразных устройств связи и цифровой обработки информации.

Документация по пакету и замечания по его описанию

Электронная документация по пакету представлена в виде двух файлов формата PDF. Один из файлов представляет собой краткое описание особенностей пакета и его отличий от предшествующей версии, а другой — руководство пользователя (User's Guide). Объем документов составляет, соответственно, 106 и 718 с. Уже одно это делает невозможным исчерпывающее описание пакета в данной книге, поскольку в ней, при значительно меньшем общем объеме, описан еще целый ряд пакетов. Указанная документация поставляется и в форме обычных книг.

Тем не менее, пакет описан достаточно полно для знакомства с ним. Это достигнуто за счет исключения многих повторений, чем «грешит» обширная фирменная документация, и, главным образом, за счет лишь описания *основных* возможностей доступа к ресурсам пакета. Число примеров применения пакета мы вынуждены ограничить разумным минимумом. Таким образом, представленное описание не заменяет фирменных руководств по данному пакету и не является прямым переводом последних. В то же время, оно дает детальный обзор пакета и призвано стимулировать читателя к самостоятельному знакомству с теми материалами, которые мы были вынуждены опустить из ограниченного объема данной книги. Тем не менее, представляется, что изложенный ниже материал вполне достаточен для освоения работы с данным обширным и чрезвычайно полезным расширением системы MATLAB+Simulink.

Литература

Для дополнительного изучения возможностей пакета и связанных с ними теоретических вопросов можно рекомендовать следующую литературу:

1. Дьяконов В. П. Компьютерная математика. Теория и практика. — М.: Нолидж, 2001.
2. Дьяконов В. П., Смердов В. Ю. Бытовая и офисная техника связи. — М.: Солон-Р, 1999.
3. Лагутенко О. И. Модемы. Справочник пользователя. — М.: Лань, 1997.
4. Блейхут Р. Теория и практика кодов, контролирующих ошибки. — М.: Мир, 1986.
5. Витерби А. Д., Омура Дж. К. Принципы цифровой связи и кодирования. — М.: Радио и связь, 1982.

6. Боккер П. Передача данных (техника связи в системах телеобработки данных). В 2-х тт. — М.: Связь, 1980 (т.1); Радио и связь, 1981 (т. 2).
7. Адаптивные фильтры/Под ред. К. Ф. Н. Коуэна и П. М. Гранта. — М.: Мир, 1988.
8. Окунев Ю. Б. Цифровая передача информации фазоманипулированными сигналами. — М.: Радио и связь, 1991.
9. Kondoz A. M. Digital Speech. Chichester, England: John Wiley & Sons, 1994.
10. Sklar B. Digital Communications: Fundamentals and Applications. Englewood Cliffs, N.J.: Prentice-Hall, 1988.
11. Кларк Дж., мл., Кейн Дж. Кодирование с исправлением ошибок в системах цифровой связи. — М.: Радио и связь, 1987.
12. Lin S., Costello D. J., Jr. Error Control Coding: Fundamentals and Applications. Englewood Cliffs, N.J.: Prentice-Hall, 1983.
13. Wesley P. W., Weldon E. J., Jr. Error-correcting Codes, 2nd ed. Cambridge, Mass.: MIT Press, 1972.
14. van Lint J. H. Introduction to Coding Theory. New York: Springer-Verlag, 1982.
15. Gitlin R. D., Hayes J. F., Weinstein S. B. Data Communications Principles. New York: Plenum, 1992.
16. Jeruchim M. C., Balaban P., Sam Shanmugan K. Simulation of Communication Systems. New York: Plenum Press, 1992.
17. Proakis J. G. Digital Communications, 3rd ed. New York: McGraw-Hill, 1995.
18. Korn I. Digital Communications. New York: Van Nostrand Reinhold, 1985.
19. Oppenheim A. V., Schafer R. W. Discrete-Time Signal Processing. Englewood Cliffs, N.J.: Prentice Hall, 1989.
20. Kailath T. Linear Systems. Englewood Cliffs, N.J.: Prentice-Hall, 1980.
21. Lang S. Algebra. Third Edition. Reading, Mass.: Addison-Wesley, 1993.

Основы работы с пакетом

Будучи пакетом расширения системы «MATLAB+Simulink», пакет Communications Blockset допускает три основных вида работы:

- визуально-ориентированную подготовку моделей и их исполнение в среде Simulink;

- применение функций пакета в командах, исполняемых в командном режиме MATLAB;
- применение функций пакета в составе m-файлов MATLAB.

Основным и наиболее удобным видом работы является визуально-ориентированная подготовка моделей путем перетаскивания компонентов мышью из окон библиотек в окно создаваемой модели коммуникационной системы с последующим соединением этих компонентов друг с другом. При этом используются все возможности пакета Simulink, включая имеющиеся в нем модели компонентов общего назначения — например, модели соединений между блоками, виртуальные осциллографы и прочие регистрирующие приборы. Полученная модель после установки ее параметров запускается, затем изучается и корректируется.

Весьма практичным является подход, когда модель создается не «с нуля», а путем модификации одного из подходящих демонстрационных примеров. Это позволяет сократить число операций ввода моделей компонентов и их соединения друг с другом, а также уменьшить число настраиваемых параметров. Обилие демонстрационных примеров дает широкие возможности для применения такого подхода. Более того, нередко удается подобрать почти готовый пример. Особенно пригодится этот подход студентам. Ничего плохого в нем нет, ибо если учащийся смог самостоятельно подстроить готовый пример под решение своей задачи, значит, он выполнил свою учебную миссию и разобрался в работе с пакетом и теми моделями, которые нужны для решения задач пользователя.

Работа с функциями пакета из командной строки полезна в основном для изучения отдельных функций. От пользователя, только приступающего к изучению пакета Communications Blockset, трудно ожидать виртуозного владения техникой работы в командном режиме. Для этого прежде всего надо основательно знать базовую систему MATLAB.

Подготовка m-файлов с применением функций пакета Communications Blockset — это уже «высший пилотаж» в части использования пакета. Этот вид работы позволяет создавать свои программы на языке MATLAB. Для тех, кто намерен заниматься такой работой, эта глава не более чем введение в технику применения пакета Communications Blockset. Такие пользователи должны уверенно разбираться в обширной англоязычной обычной и электронной документации как по пакету Communications Blockset, так и по всему комплексу «MATLAB+ Simulink».

В целом пакет рассчитан на самообучение пользователя в процессе работы с ним. Это достигается за счет включения в пакет множества заранее созданных моделей компонентов и довольно наглядных примеров их применения.

Вызов библиотек пакета из среды MATLAB

Для вызова окна с разделами библиотек пакета достаточно в командном режиме MATLAB исполнить команду

```
» commlib;
```

Появится окно, показанное на рис. 6.1. В этом окне расположены значки с основными разделами библиотеки коммуникационных устройств и примеров их применения. Активизация того или иного значка (двойным щелчком) открывает окно соответствующего раздела библиотеки. В качестве примера на рис. 6.1 под основным окном представлены окна с разделами Channel Coding и Modulation.

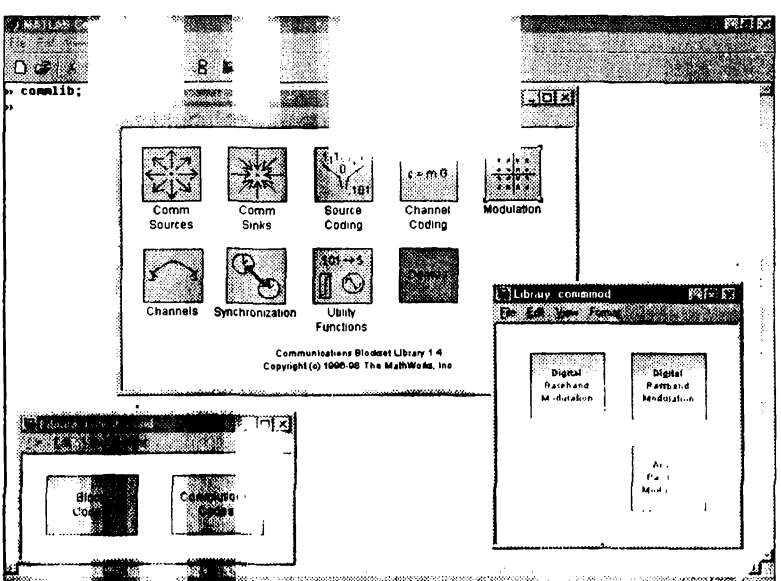


Рис. 6.1. Окна библиотеки пакета Communications Blockset и нескольких ее разделов

В дальнейшем мы рассмотрим состав всех разделов библиотеки этого пакета.

Вызов графического интерфейса пользователя

Для доступа к графическому интерфейсу пользователя пакета достаточно исполнить команду

```
» commgui;
```

Появится основное окно графического интерфейса, представленное на рис. 6.2. Это окно используется в ряде демонстрационных примеров. В окнах библиотек представлены значки множества примеров. В основном окне доступ к примерам дает значок Demos, в окнах разделов библиотеки примеры представлены значками с названиями, но в конце их стоит слово demo. Значки примеров имеют голубой фон, тогда как компоненты моделей имеют значки с белым фоном.

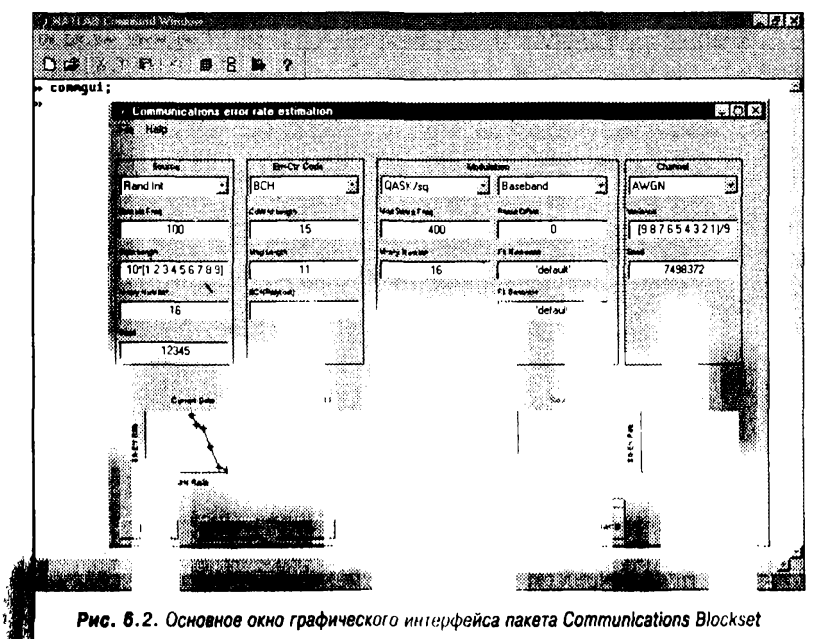


***Рис. 6.2.** Основное окно графического интерфейса пакета Communications Blockset*

При таком вызове часть областей в основном окне пуста. Данные в них появляются после запуска той или иной модели, выводящей данные в эти области.

Окно GUI имеет ряд групп элементов управления, в верхних частях которых размещены списки, обеспечивающие выбор компонентов следующих типов:

- Source — выбор источников сигналов;
- Err-Ctr Code — выбор корректирующего кода;
- Modulation — выбор вида модуляции;
- Channel — выбор канала связи.

С помощью этих средств можно быстро провести анализ поведения системы при разных вариантах ее построения, например, при изменении источника входного сигнала, выборе иного типа модуляции и т. д.

Вызов полного списка функций

С помощью приведенной ниже команды можно вызвать список всех функций пакета:

```
» help comm;
 Communications Toolbox.
 Version 1.4  (R11) 10-Jul-1998

 What's new.
  Readme     - New features, bug fixes, and changes in this version.

 Simulink Block Library.
  commlib    - Open the Communications Toolbox Simulink Block Library.

 Signal Sources and Display Functions.
  biterr     - Bit error number and bit error rate computation.
  eyescat    - Eye-pattern diagram or scatter plot.
  randint    - Random integer matrix generator.
  randbit    - Binary noise generator.
  symerr     - Symbol error number and symbol error rate computation.

 Source Coding.
  compand    - Source code mu-law or A-law compressor or expander.
  dpcmdeco   - Differential pulse code modulation decode computation.
  dpcmenco   - Differential pulse code modulation encode computation.
  dpcmopt    - Differential pulse code modulation parameter optimization.
  lloyds     - Scalar quantization optimization using training data.
  quantiz    - Quantization index and quantized output value.

 Error-Control Coding.
  bchpoly    - BCH code generator polynomial.
  cyclgen    - Cyclic code generator and parity-check matrices.
  cyclpoly   - Cyclic generator polynomial.
  decode     - Decode computation for error-control coding methods.
  encode     - Encode computation for error-control coding methods.
```

```
  gen2par   - Generator matrix to parity-check matrix conversion.
  gen2abcd  - Convert a convolution function into state-space form.
  gfweight  - Calculates Hamming weight.
  hammgen   - Hamming code generator and parity-check matrices.
  htruthtb  - Truth table from generator or parity check matrix.
  oct2gen   - Convolution code octal form to binary form conversion.
  rsdecof   - Decode a Reed-Solomon code encoded text file.
  rsencof   - Encode a text file using Reed-Solomon code.
  rspoly    - Produce Reed-Solomon code generator polynomial
  sim2gen   - Convolution code transfer function in octal from a Simulink model.
  sim2logi  - Simulink model to convolution code logic representation.
  sim2tran  - Simulink model to convolution code transfer function.
  viterbi   - Viterbi convolution code decode with limited memory.

 Lower-Level Functions for Error-Control Coding.
  bchcore   - The kernel computation part for BCH decode.
  bchdeco   - BCH decode computation.
  bchenco   - BCH code encode computation.
  convdeco  - Convolution code decode.
  convenco  - Convolution code encode.
  errlocp   - Error-location polynomial for BCH and RS code.
  rscore    - Reed-Solomon decode core.
  rsdeco    - A user-interfaced Reed-Solomon decode calculation.
  rsdecode  - Decode Reed-Solomon code.
  rsencode  - Encode Reed-Solomon code.
  rsenco    - A user-interfaced Reed-Solomon encode calculation.

 Modulation/Demodulation.
  ademod    - Analog demodulation computation.
  ademodce  - Demodulate from a complex envelope of analog modulated signal.
  amod      - Analog modulation.
  amodce    - Complex envelope of analog modulation.
  apkconst  - Compute/plot ASK/PSK constellation.
  ddemod    - Demodulate a digital modulated signal.
  ddemodce  - Demodulate from complex envelope of digital modulated signal.
  demodmap  - Digital demodulation demapping.
  dmod      - Digital modulation.
  dmodce    - Complex envelope of digital modulation.
  modmap    - Digital modulation mapping.
  qaskdeco  - Decode a two-dimensional code to one dimension message.
  qaskenco  - Encode a message to two-dimensional form for QASK use.

 Special Filters.
  hank2sys  - Convert Hankel matrix to linear system.
  hilbiir   - Hilbert transform IIR filter.
  imp2sys   - FIR filter to IIR filter conversion.
  rcosflt   - Filtering signal using raised cosine filter.
```

```
  rcosine   - Design raised cosine filter.

 Lower-Level Functions for special filters.
  rcosfir   - Raised cosine FIR filter (used by rcosine).
  rcosiir   - Raised cosine IIR filter (used by rcosine).

 Galois Field Computation.
  flxor     - Exclusive OR calculation.
  gfadd     - GF additive computation.
  gfconv    - GF polynomial convolution (multiply) computation.
  gfcosets  - Cyclotomic cosets generator.
  gfdeconv  - GF polynomial deconvolution (dividing) computation.
  gfdiv     - GF dividing computation.
  gffilter  - GF filtering computation.
  gflineq   - Computer X in A X = B in GF(p) field.
  gfminpol  - Find minimal polynomials.
  gfmul     - GF multiplicative computation.
  gfplus    - GF(2^p) additive computation.
  gfpretty  - GF polynomial presentation.
  gfprimck  - Test GF irreducible and primitive properties.
  gfprimdf  - Output default primitive polynomial at a given degree.
  gfprimfd  - Find GF primitive polynomial.
  gfrank    - Find the rank of a matrix in Galois field.
  gfrepcov  - GF polynomial conversion.
  gfroots   - Find roots of a polynomial in GF(p^m) field.
  gfsub     - GF subtraction computation.
  gftrunc   - GF polynomial truncation processing.
  gftuple   - GF m-tuple representation and power representation.

 Utilities.
  bi2de     - Binary to positive decimal integer conversion.
  blkdiag   - Block diagonal concatenation of input arguments (MATLAB Toolbox).
  callhelp  - Display help text section.
  checkinp  - Function default value check.
  closebut  - Close all but specified figure window.
  de2bi     - Positive decimal integer to binary conversion.
  vec2mat   - Convert vector to matrix by a given column number.

 Graphical User Interface example.
  commgui   - An MATLAB GUI example of bit-error rate computation.

 See also COMMMASKS, COMMSFUN, COMMSIM.
```

Обратите внимание на имеющуюся в конце этого сообщения ссылку на дополнительные команды, открывающие возможность дополнительного знакомства со средствами пакета.

Здесь приведены англоязычные наименования соответствующих команд и функций в том виде, как они выводятся системой MATLAB. Большинство этих наименований вполне очевидно специалистам и даже учащимся, изучающим курсы основ теории связи и передачи информации. Тем не менее, в дальнейшем, по мере описания функций пакета, соответствующие термины будут даны в русскоязычном написании.

Вызов справки по конкретной функции

Зная имена функций, можно вызвать информацию по любой из них командой `help f_name`, где `f_name` — имя функции. Например, для функции `bi2de`, предназначенной для преобразования двоичного числа в десятичное, это даст следующее:

```
» help bi2de:
 BI2DE Converts binary to decimal format.
    D = BI2DE(B) converts a binary representation vector B to a decimal
    presentation D. When B is a matrix, the output D is a column vector
    with each element being the transfer of a row of B.
    The first element in B represents the lowest binary bit. For example
    bi2de([1 0]) results in 1; bi2de([0 1]) results in 2.

    D = BI2DE(B, P) converts a P-based vector to a decimal.

    See also DE2BI.
```

Естественно, что описание функций дано на английском языке, что при серьезной работе с пакетом может вызывать определенные трудности.

Вызов списка библиотек и примеров Simulink

Доступ к списку Simulink-файлов пакета Communications Blockset осуществляется командой `help commsim`:

```
 help commsim;
Communications Toolbox Simulink files.

Top level file for the Communications Toolbox library.
 commlib   - Communications Toolbox top level library.

Secondary level files for communications toolbox library.
 com_case  - Case study and examples.
 com_chan  - Channel sub-library.
 com_code  - Error-control code sub-library.
 com_frmt  - Source coding sub-library.
 com_mlac  - Multi-access sub-library.
```

```
 com_modu  - Modulation / demodulation sub-directory.
 com_sour  - Source/sink sub-library.
 com_sync  - Synchronization sub-library.
 com_util  - Utility sub-library.

Lower level files for communications toolbox library.
 comcdbch  - BCH error-control code sub-library.
 comcdcon  - Convolution error-control code sub-library.
 comcdcyc  - Cyclic error-control code sub-library.
 comcdham  - Hamming error-control code sub-library.
 comcdlin  - Linear block error-control code sub-library.
 comcdrs   - Reed-Solomon error-control code sub-library.
 comcdint  - Interleave, scrambler sub-library.
 com_cdmo  - Analog modulation, passband simulation sub-library.
 com_cemo  - Analog modulation, baseband simulation sub-library.
 com_ddmo  - Digital modulation, passband simulation sub-library.
 com_ddm2  - Additional digital modulation, passband simulation sub-library.
 com_mocd  - Digital modulation, passband mapping sub-library.
 com_moce  - Digital modulation, baseband simulation sub-library.
 com_momp  - Digital modulation, baseband mapping sub-library.

Source/sink library demonstration files.
 ttrigwks  - Read/write workspace variables
 ttrigfil  - Read/write files
 tsteyesc  - Eye-pattern Diagram and Error-meter
 tstbterr  - Error-meter
 turand    - Uniform random noise generator
 tgrand    - Gaussian random noise generator
 tirand    - Random integer generator
 tprand    - Poisson integer generator
 tbrand    - Binary vector generator
 tstrayl   - Rayleigh noise generator
 tstricia  - Rician noise generator

Source coding library demonstration files.
 tstquans  - quantization encode and decode
 tstdpcms  - DPCM encode and decode
 tstquant  - enable quantization encode and decode

Error-Control Coding Library.
 tsthammc  - Hamming code vector I/O
 tsthamsq  - Hamming code sequential I/O
 tstbchcd  - BCH code vector I/O
 tutbchcd  - BCH code vector I/O, 2
 tstbchsq  - BCH code sequential I/O
 tstrscod  - RS code integer vector I/O
 tstrscbv  - RS code binary vector I/O
```

```
tstrsinv  - RS code integer sequential I/O
tstrsbnv  - RS code binary sequential I/O
tstcyclc  - Cyclic code, Hamming code, and linear block code vector I/O
tstcycsq  - Cyclic code sequential I/O
tstlinco  - Linear block code vector I/O
tstlinsq  - Linear block code sequential I/O
tstconcd  - Convolution code vector I/O
tutconcd  - Convolution code vector I/O, 2
tstconsq  - Convolution code Sequential I/O
fig_10_9  - Block diagram for building convolution code transfer function
exconvcd  - Block diagram for building convolution code transfer function
testinterleave - A basic example of interleaving
tstrsbch  - Combination of R-S code, interleave and BCH code
testscrm  - Scrambler and descrambler
dsss      - Direct sequence spread spectrum system

Modulation and demodulation demonstration files.
test_am2  - DSB-SC AM/ADM passband and baseband
test_qa2  - QAM passband and baseband
test_fm2  - FM passband and baseband
test_fm2  - PM passband and baseband
testssba  - SSB AM passband and baseband
testamc2  - AM with carrier passband and baseband
tstam2cd  - MASK map/demap, passband
tstqa2cd  - QASK S-map map/demap, passband
tstps2cd  - QASK C-map map/demap, passband
tstps3cd  - QASK A-map map/demap, passband
tstfs2cd  - MFSK map/demap, passband
tstpk2cd  - MPSK map/demap, passband
tmpam2cd  - MASK map/demap, baseband
tmpqa2cd  - QASK S-map map/demap, baseband
tmpps2cd  - QASK C-map map/demap, baseband
tmpps3cd  - QASK A-map map/demap, baseband
tmpfs2cd  - MFSK map/demap, passband
tmppk2cd  - MPSK map/demap, passband
tddmmask  - MASK modem, passband
tddmqask  - QASK S-map modem, passband
tutcdwmo  - QASK S-map with BCH code, passband.
tddps2cd  - QASK C-map modem, passband
tddps3cd  - QASK A-map modem, passband
tddmmfsk  - MFSK modem, passband
tddmmpsk  - MPSK modem, passband
tstam2ce  - MASK modem, baseband
tcemqask  - QASK S-map modem, baseband
tceps2cd  - QASK C-map modem, baseband
tceps3cd  - QASK A-map modem, baseband
tcemmfsk  - MFSK modem, baseband
```

```
  tcemmpsk  - MPSK modem, baseband
  tdpsk2    - DPSK modem, passband
  tdpsk     - DPSK map/demap, passband
  tbpsk     - BPSK modem, passband
  tbpsk2    - BPSK map/demap, passband
  tqpsk     - QPSK modem, passband
  tqpskmap  - QPSK map/demap, passband
  toqpsk    - OQPSK modem, passband
  toqpskmp  - OQPSK map/demap, passband
  tmsk      - MSK modem, passband

Multiple access demonstration files.
  tsttdma   - Digital TDMA
  tsttdmaa  - Time-sharing multiplex
  tstfdma   - FDMA
  tstcdma   - CDMA

Special filters demonstration files.
  tstrcos   - Raised cosine filter
  tstrcos2  - Raised cosine filter
  tstsinc   - Sinc filter
  tsthilbt  - Hilbert transform filter

Synchronization demonstration files.
  tstpll    - Phase locked loop

Utilities and miscellany library demonstration files.
  testregi  - Register shift and triggered buffer down.
  tstdist   - Triggered signal switch

  See also COMM, COMMMASKS, COMMSFUN.
```

Здесь названия библиотек и демонстрационных примеров также даны, естественно, на английском языке. Наиболее важные из них будут рассмотрены в дальнейшем.

Доступ к библиотеке пакета Communications Blockset

Доступ к библиотеке Communications Blockset возможен и даже, пожалуй, наиболее удобен прямо из пакета моделирования Simulink. В MATLAB 5.3.1 для этого достаточно запустить пакет Simulink, например, активизировав кнопку его запуска в панели инструментов окна системы MATLAB или введя в командной строке окна MATLAB команду **simulink**. Появится окно броузера библиотек, показанное на рис. 6.3 в окне системы MATLAB слева.

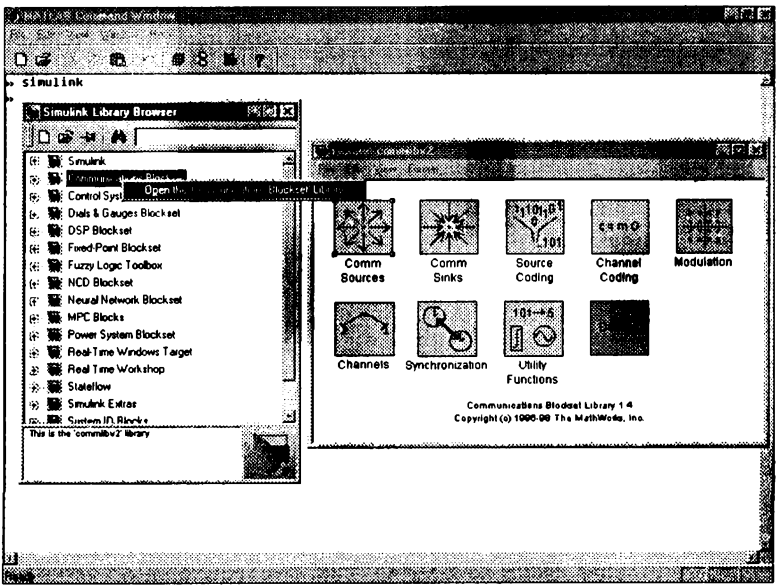


Рис. 6.3. Окно броузера библиотек и окно со значками библиотек пакета Communications Blockset

Используя правую кнопку мыши при выделенной в списке библиотеке Communications Blockset, можно выполнить единственную команду — Open the 'Communications Blockset' Library. Она открывает окно с графическим представлением библиотек пакета Communications Blockset — оно показано на рис. 6.3 в правой части главного окна MATLAB.

Доступ к mdl-файлам из пакета Simulink

Мощь пакета Communications Blockset определяется не только обширной библиотекой его компонентов, из которых можно создавать модели практически любых устройств и систем связи, но и широчайшим набором уже подготовленных моделей таких устройств и систем. Они представлены в виде файлов с расширением .mdl и находятся в директории *\TOOLBOX\COMM\COMMSIM\, где * — основная директория пакета. Вход в эту директорию удобнее всего выполнить из окна загрузки файлов в пакете Simulink. Для этого достаточно исполнить команду Open из меню File или активизировать кнопку этой команды в панели инструментов пакета Simulink.

Окно Open в полностью открытом и растянутом на весь экран виде представлено на рис. 6.4. В этом окне видно главное «богатство» па-

кета Communications Blockset — более двухсот файлов с библиотеками компонентов и множеством примеров их применения. Среди этих примеров можно найти модели большинства применяемых на практике систем и устройств.

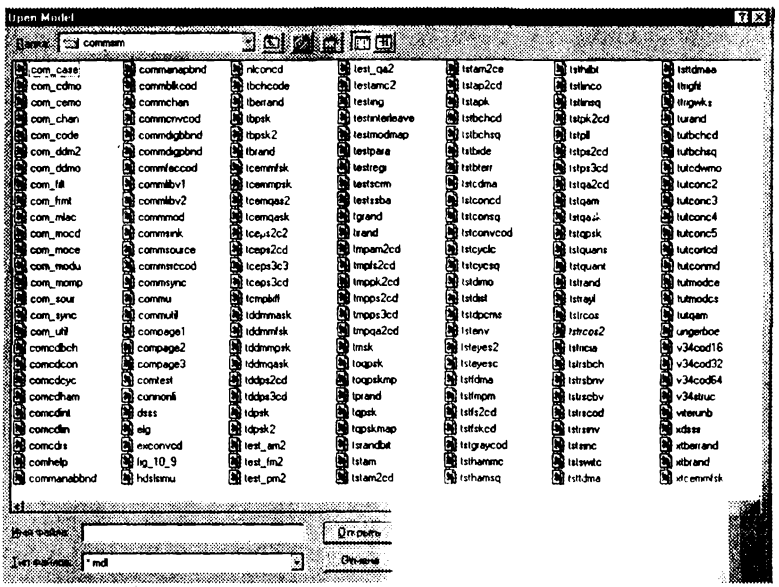


Рис. 6.4. Окно загрузки файлов с расширением .mdl в среде пакета Simulink

Следует иметь в виду, что все эти файлы написаны на языке программирования системы MATLAB, поэтому они являются подлинным кладезем профессионально составленных примеров программирования. Пользователь даже средней квалификации может редактировать и модернизировать эти файлы, записанные в текстовом формате и потому легко читаемые и загружаемые в любой текстовый редактор, поддерживающий формат ASCII.

Обзор библиотек компонентов пакета Communications Blockset

Библиотека пакета Communications Blockset содержит множество моделей компонентов, с помощью которых можно моделировать как самые простые, так и весьма сложные устройства и системы связи.

Несмотря на обилие входящих в нее средств, не надо забывать, что они составляют лишь подмножество обширного набора средств моделирования, которые имеются в пакете Simulink. При этом обеспечиваются простой графический интерфейс пользователя и визуально-ориентированный способ подготовки моделей устройств и систем, полностью аналогичные применяемым в пакете Simulink.

Библиотека источников сигнала

Активизируя значок Comm Sources основного окна библиотек (см. рис. 6.1), можно открыть окно с источниками сигналов. Это окно показано на рис. 6.5. Там же показаны перенесенная мышью в окно пакета Simulink модель одного из источников сигнала (Pulse divide sample time) и окно установки ее параметров. Последнее доступно после переноса модели объекта из окна библиотеки в окно Simulink. Для вызова окна установки параметров достаточно выполнить на значке объекта двойной щелчок.

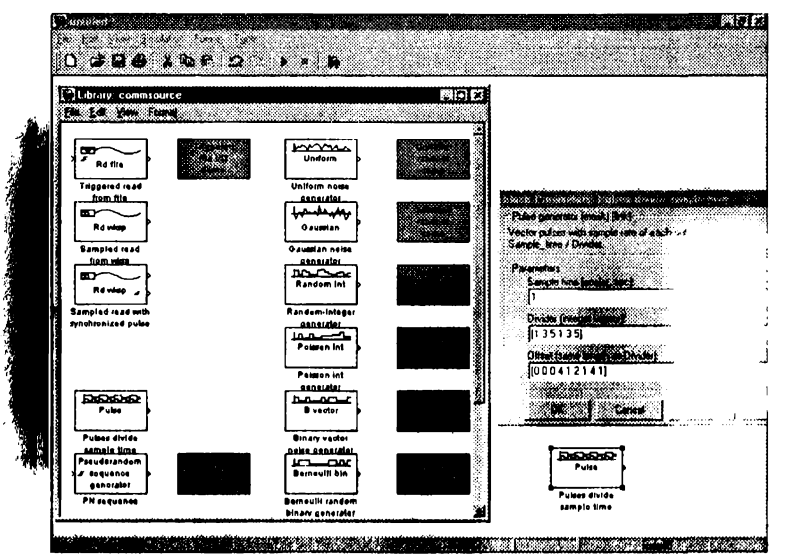


Рис. 6.5. Выбор источника сигнала

В пакете имеется представительный набор источников. Его можно разбить на две группы. Первая группа содержит источники детерминированных сигналов, характеристики которых задаются считыванием данных из файла, из переменных рабочей области MATLAB, от

синхронизирующих импульсов и т. д. Эти источники расположены в левой части окна. Вторая группа источников — это генераторы случайных сигналов с заданными законами распределения и с определенными характеристиками, например с равномерным распределением (Uniform noise generator), с гауссовым шумом (Gaussian noise generator), со случайными целыми числами (Random-integer generator) и т. д. Краткие названия источников приводятся под значками их моделей, а полное описание источников дается в окнах их параметров (см. пример на рис. 6.5).

Ввиду общеизвестности законов распределения сигналов источников мы не будем обсуждать эти законы. Соответствующие расчетные формулы приведены как в литературе по статистической радиотехнике, так и в справочной системе пакета.

Для каждого источника предусмотрен демонстрационный пример. Он вызывается активизацией значка примера, имеющего голубой фон, позволяющий сразу отличить значок примера от значка модели (с белым фоном). В каждом примере приводится модель небольшой коммуникационной системы, позволяющая исследовать свойства источника, в частности при изменении его параметров. Рисунок 6.6 показывает работу с демонстрационным примером для источника Uniform noise generator — генератора шума с равномерным распределением.

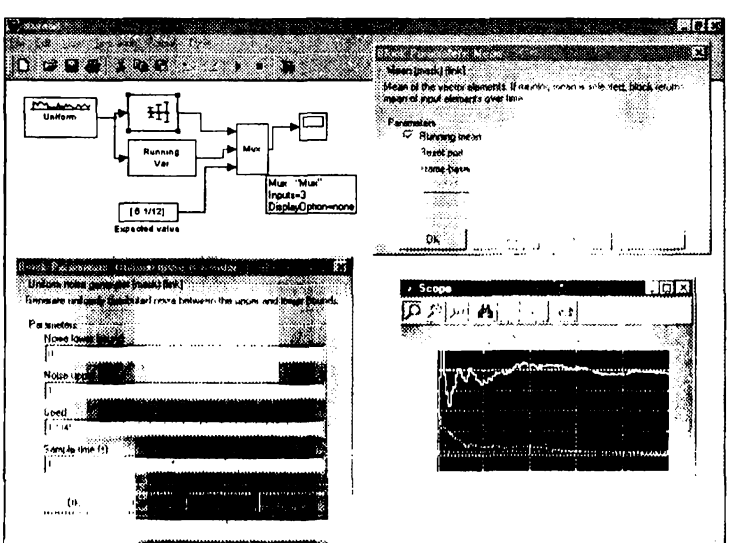


Рис. 6.6. Демонстрационный пример использования источника сигнала

Как видно из рис. 6.6, демонстрационный пример дает исчерпывающую информацию о самом источнике и испытательной системе для него, выводя окна установки параметров для моделей системы и пользуясь всплывающими подсказками, появляющимися в желтом прямоугольнике при установке указателя мыши на тот или иной объект. На рис. 6.6 всплывающая подсказка показана для блока микшера Mux, служащего для построения на экране виртуального осциллографа трех кривых, что имитирует работу многоканального осциллографа.

Аналогичным образом можно задать любой другой источник, установить его параметры и опробовать в виртуальной работе. Читателю рекомендуется проделать это самостоятельно по образу и подобию описанного примера.

Библиотека регистрирующих устройств

Библиотека устройств регистрации представлена довольно скромно — всего четырьмя устройствами. Окно с ними показано на рис. 6.7 слева. Внизу представлен пример применения виртуального регистрирующего устройства Sample-time eye-diagram scatter. Его окно выводится активизацией соответствующего значка в окне библиотеки. Кроме того, на рисунке показано окно установки параметров данного регистрирующего устройства.

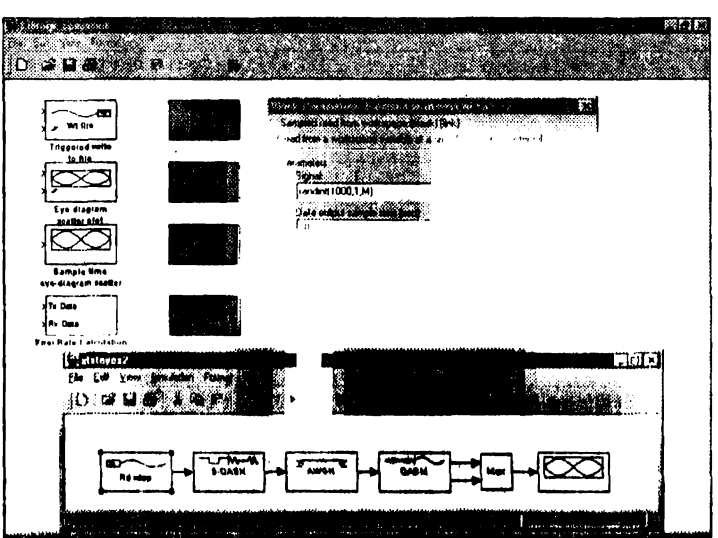


Рис. 6.7. Окно библиотеки устройств регистрации и пример применения одного из них

Работу регистратора Sample-time eye-diagram scatter поясняет рис. 6.8. Нетрудно заметить, что строятся временная диаграмма (так называемая *глазковая диаграмма — eye diagram*; слева) и распределение точек на плоскости (*диаграмма рассеяния — scatter plot*). Графики такого рода широко используются во многих демонстрационных примерах описываемого пакета расширения. Мы рассмотрим их по мере описания этих примеров.

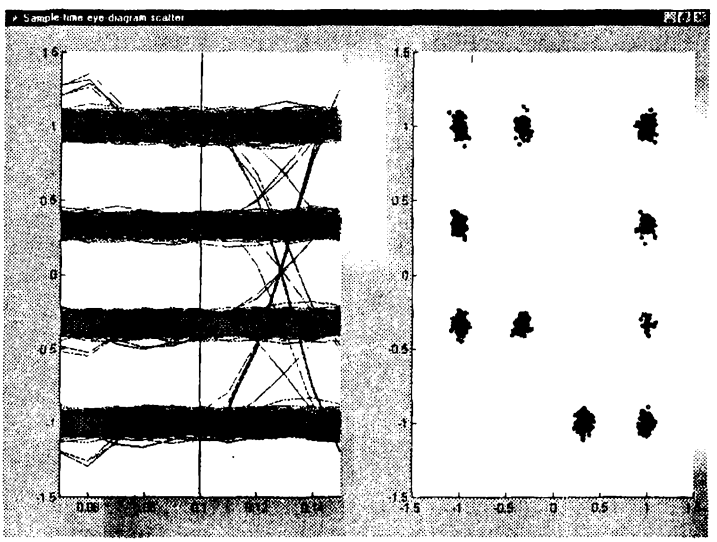


Рис. 6.8. Пример работы регистратора Sample-time eye-diagram scatter

В дальнейшем мы неоднократно будем наблюдать работу различных виртуальных регистрирующих устройств. Однако надо заметить, что ограниченный набор регистраторов в этом разделе библиотеки вовсе не означает скудности возможностей пакета в этой части — не надо забывать, что можно использовать множество виртуальных регистрирующих устройств пакета Simulink, в состав которого входит описываемый здесь пакет. Так, с работой виртуальных многоканальных осциллографов пакета Simulink мы сталкивались уже многократно.

Библиотека кодирования и декодирования источников сигнала

Раздел библиотеки Source Coding содержит модели кодирующих и декодирующих устройств для источников сигнала. Большая их часть –

это квантователи того или иного вида, обеспечивающие тот или иной способ преобразования аналоговой информации в дискретную и обратно. Окно этого раздела, представленное на рис. 6.9 слева, дает представление об имеющихся моделях этих устройств и демонстрационных примерах их применения.

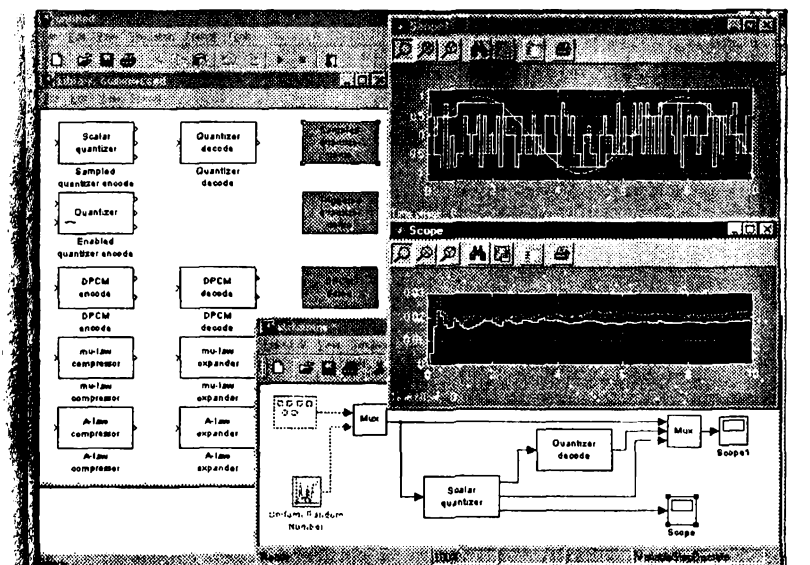


Рис. 6.9. Окно кодирующих и декодирующих устройств и пример использования следящего квантователя с постоянным шагом по уровню

Один из таких примеров (применение квантователя с постоянным шагом по уровню) представлен на рис. 6.9 снизу. Там же даны осциллограммы, поясняющие его работу.

Еще один пример использования квантователя, на этот раз с шагом, постоянным по времени, представлен на рис. 6.10. Здесь показаны процессы как кодирования, так и декодирования сигнала (верхняя осциллограмма), а также искажения, вносимые квантователем (нижняя осциллограмма). Окна установки параметров квантователя и декодера, также показанные на этом рисунке, позволяют судить о простоте настройки этих блоков.

По образцу и подобию приведенных примеров читатель может ознакомиться с другими типами кодирующих устройств и примерами их применения.

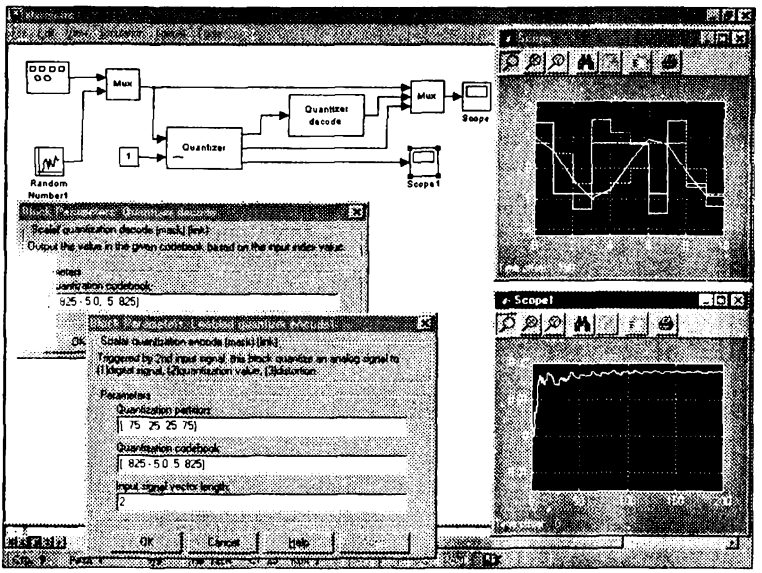


Рис. 6.10. Пример кодирования и декодирования сигнала с применением квантования с постоянным шагом по времени

Библиотеки модуляторов

Модуляторы — пожалуй, самые распространенные устройства в технике связи. В этом нет ничего удивительного, поскольку именно модуляция (процесс наложения информации на несущее колебание) обеспечивает саму принципиальную возможность осуществления беспроводной связи. Поэтому библиотека моделей модуляторов в пакете Communications способна поразить воображение даже специалиста в этих областях обилием предлагаемых средств и простотой их использования.

Библиотека модуляторов делится на четыре раздела, каждый из которых имеет свое окно (рис. 6.11). Всего библиотека содержит свыше 50 моделей модуляторов, каждая из которых сопровождается примером применения.

Рисунок 6.12 показывает пример моделирования модулятора, осуществляющего модуляцию пилообразного сигнала с использованием одной боковой полосы спектра (Single Side Band, SSB). Этот вид модуляции позволяет вдвое сократить полосу частот, занимаемых сигналом, использовать возможность подавления несущей и, в конечном счете,

существенно повысить отношение сигнал/шум в тракте передачи-приема информации. Осциллограммы показывают исходный сигнал и результат его восстановления (после прохождения блоков модулятора и демодулятора) при использовании верхней боковой полосы спектра.

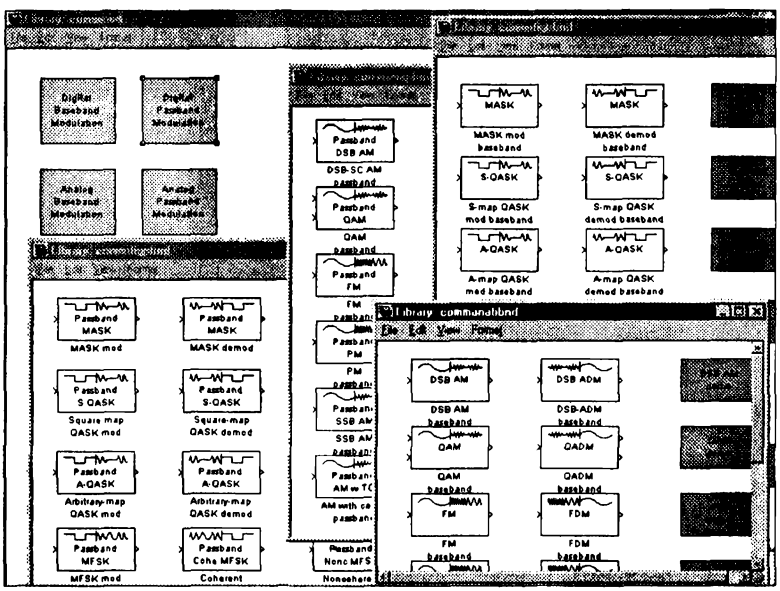


Рис. 6.11. Главное окно библиотеки модуляторов и окна ее разделов

В этом примере используется виртуальный анализатор спектра. Его окна для вывода спектров верхней и нижней полос представлены на рис. 6.13. Этот «прибор» показывает временные диаграммы модулированного сигнала, а также амплитудный и фазовый спектры для случаев использования нижней и верхней боковых полос. Следует отметить, что настоящие анализаторы спектра — очень дорогие и редкие приборы, которые есть далеко не в каждой радиотехнической лаборатории. Это делает возможность компьютерного анализа спектра сигналов весьма полезной и целесообразной.

Читатель может познакомиться с другими примерами моделирования модуляторов самостоятельно, поскольку, как уже отмечалось, примеры имеются для всех моделей модуляторов, имеющихся в данной библиотеке.

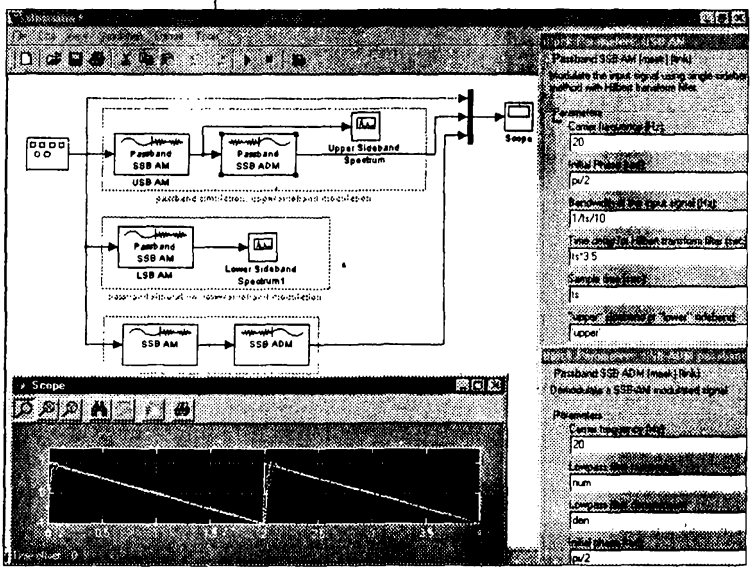


Рис. 6.12. Пример моделирования тракта с однополосным модулятором

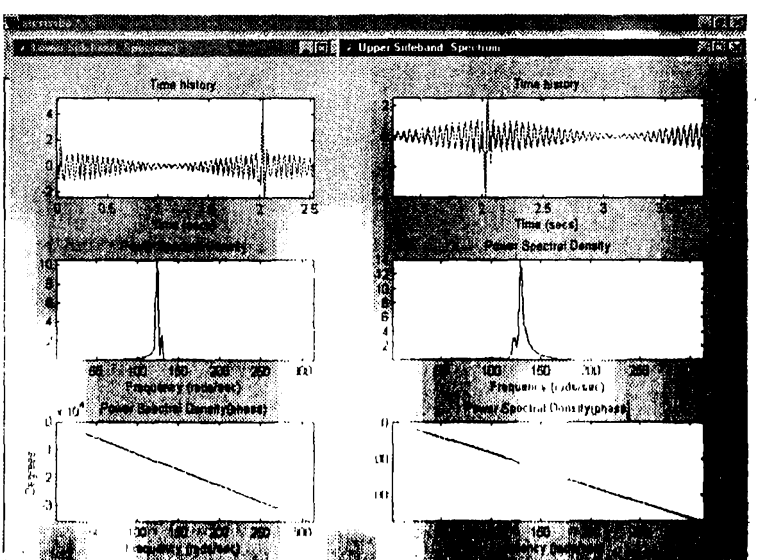


Рис. 6.13. Окна анализатора спектра

Библиотеки кодирования канала

Окно библиотеки кодирования канала Channel Coding (рис. 6.14 в левом верхнем углу) содержит два значка: Block Codes и Convolutional Codes. Последнее открывает небольшое окно (см. рис. 6.14 сверху справа) с моделями устройств для осуществления кодирования и декодирования с использованием сверточных кодов. Окно демонстрационного примера, показывающего реализацию сверточного кодирования, также показано на рис. 6.14 (внизу).

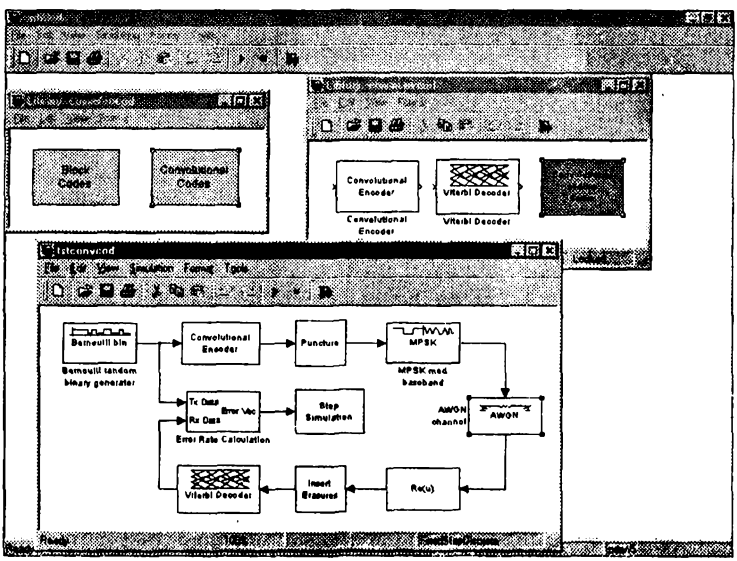


Рис. 6.14. Окно библиотеки кодирования канала и средства работы со сверточными кодами

В разделе Block Codes (рис. 6.15) имеется множество моделей специальных кодирующих и декодирующих устройств, известных под названием «кодеки». Они отличаются в основном видом кода, что и видно по названиям моделей.

Рисунок 6.16 показывает пример построения кодека на основе целочисленного векторного кодирования Рида—Соломона. Помимо осциллограммы, выявляющей ошибки декодирования, на рис. 6.16 представлено окно установки параметров источника сигнала Rd wksp. Обратите внимание на цифровой индикатор работы кодека, расположенный над блоком виртуального осциллографа. При запуске модели он показывает текущее состояние вектора ошибок.

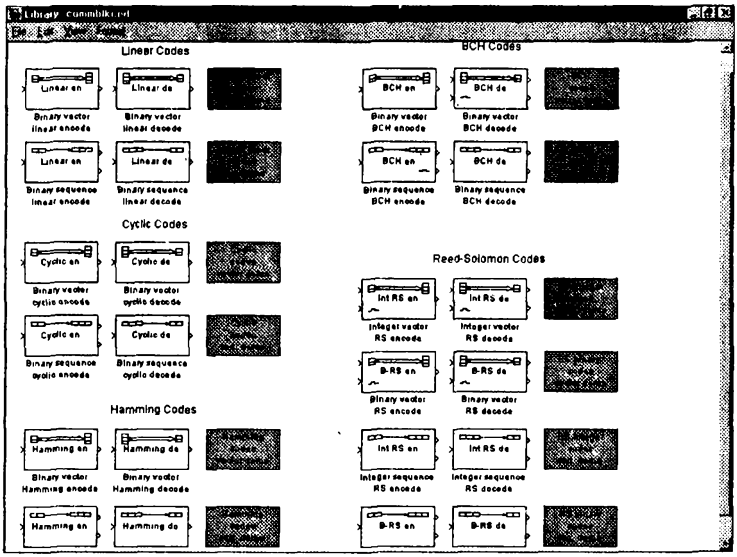


Рис. 6.15. Окно раздела блочных кодов библиотеки кодирования канала

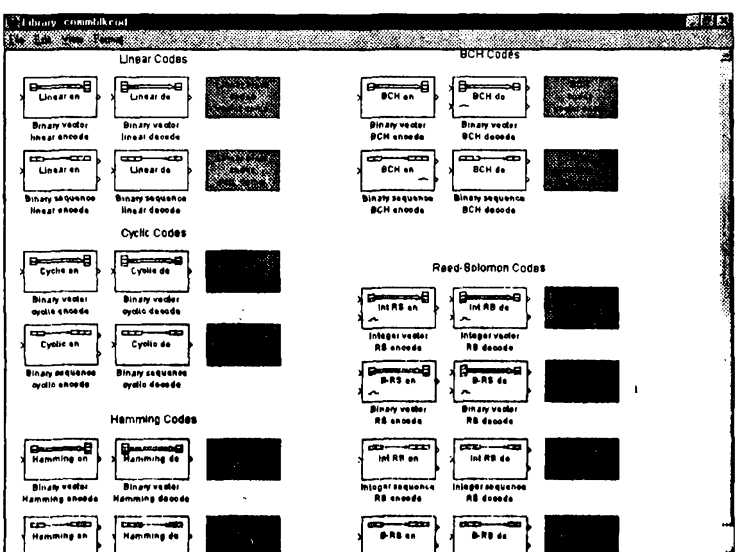


Рис. 6.16. Пример построения кодека на основе целочисленного векторного кодирования Рида–Соломона

С другими примерами построения кодеков читатель может ознакомиться самостоятельно.

Библиотека каналов

Библиотека каналов Channels представлена окном, показанным на рис. 6.17. В нем имеются значки как моделей, так и демонстрационных примеров.

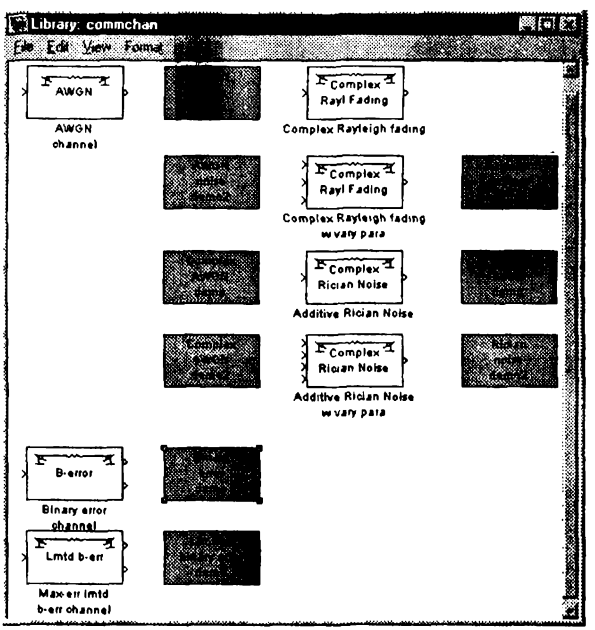


Рис. 6.17. Окно библиотеки каналов

На рис. 6.18 показан пример использования средств библиотеки каналов — моделируется бинарный канал с ошибками.

Обратите внимание на значок BCH code view table, расположенный под моделью канала. Его активизация открывает окно просмотра таблицы кодов, которое представлено на рис. 6.19. Назначение параметров N, K и T указано под таблицей.

С другими примерами применения моделей этой библиотеки читатель может ознакомиться самостоятельно.

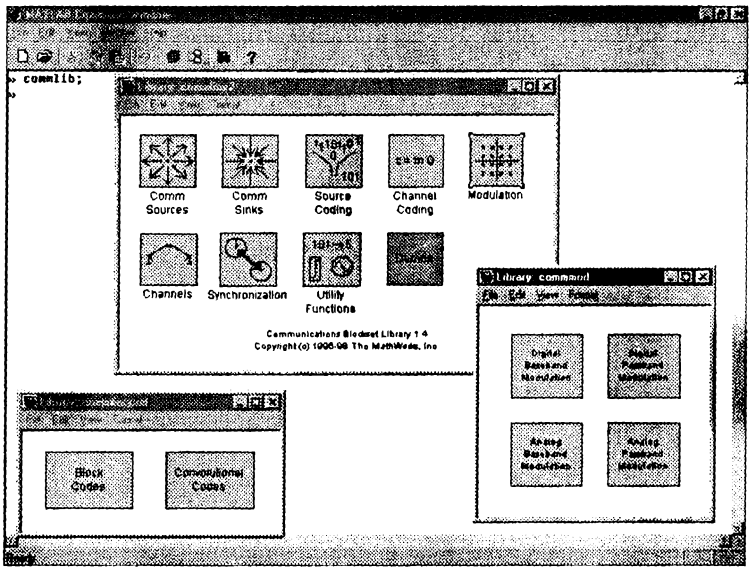


Рис. 6.18. Пример моделирования бинарного канала с ошибками

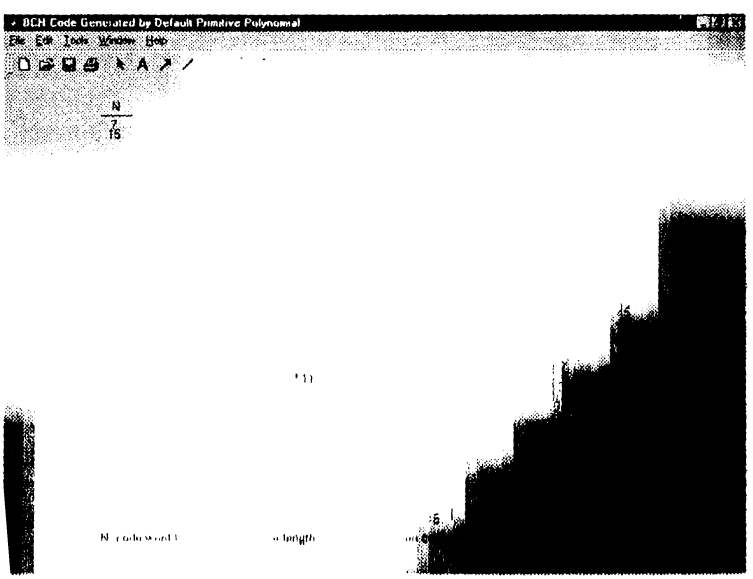


Рис. 6.19. Окно просмотра таблицы кодов

Библиотека модулей синхронизации

Библиотека модулей синхронизации Syncronization представлена окном, показанным на рис. 6.20. Как нетрудно заметить, этот раздел библиотеки содержит четыре модуля фазовой автоподстройки частоты (ФАПЧ; английское сокращение PLL расшифровывается как Phase-Locked Loop) и один демонстрационный пример.

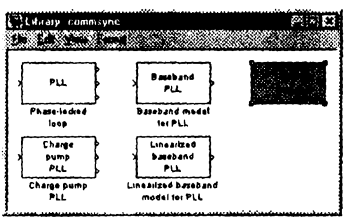


Рис. 6.20. Окно библиотеки модулей синхронизации

Пример применения модулей синхронизации показан на рис. 6.21. В этом примере, помимо демонстрации работы ФАПЧ, интересен многоканальный вывод сигналов (панель справа) с учетом фазовых сдвигов, вносимых всеми компонентами (их в этом примере 6).

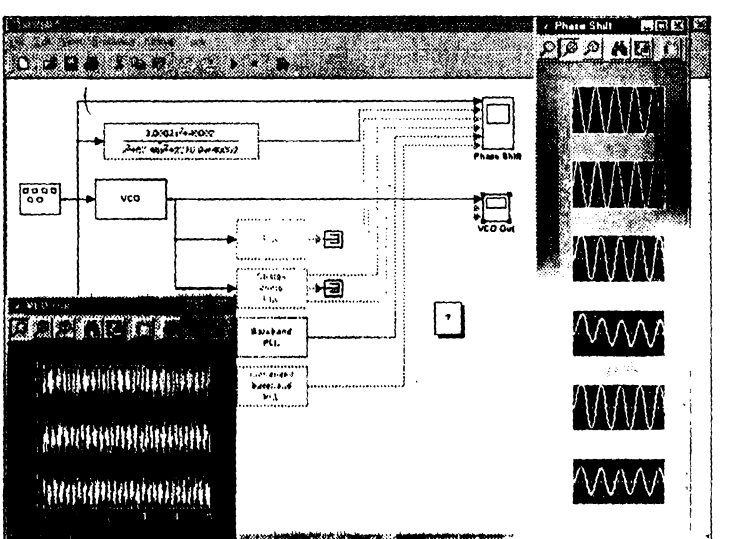


Рис. 6.21. Пример применения модулей синхронизации

Библиотека модулей синхронизации выглядит довольно скромной. Возможно, это связано с тем, что устройства синхронизации при проектировании систем связи рассматриваются как второстепенные, хотя на практике неустойчивая работа синхронизации часто служит причиной неустойчивой работы таких систем в целом.

Библиотека утилит и функций

Окно библиотеки утилит и функций Utility Functions показано на рис. 6.22. В этом разделе библиотеки представлено довольно разношерстное семейство коммуникационных устройств. С их набором можно ознакомиться по названиям моделей.

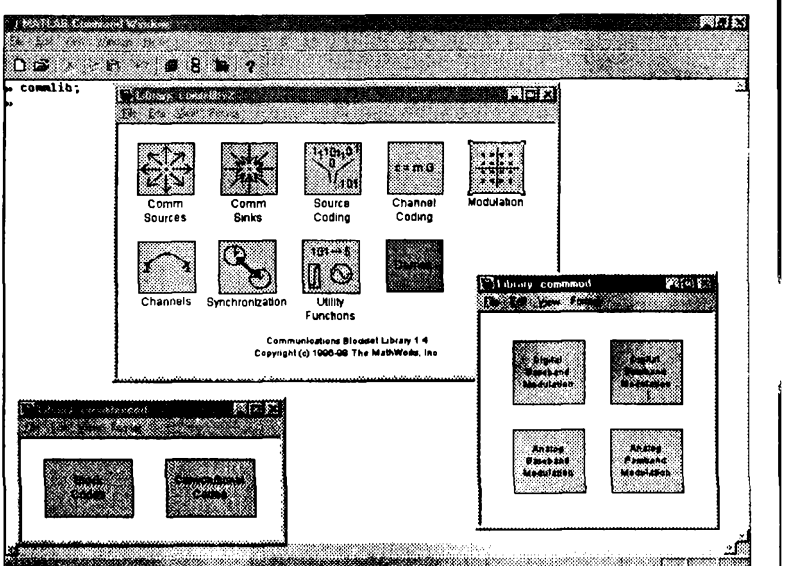


Рис. 6.22. Окно библиотеки утилит и функций

Утилиты и функции этой библиотеки предоставляют большие возможности в обработке сигналов. Поэтому рассмотрим примеры их применения более подробно.

Рисунок 6.23 даст пример применения блока Register Shift (Сдвигающий регистр). Такие блоки находят широкое применение в цифровых устройствах. Работу модели иллюстрируют временные диаграммы, полученные с помощью виртуального осциллографа. Обратите внимание на окно задания параметров блока Register Shift.

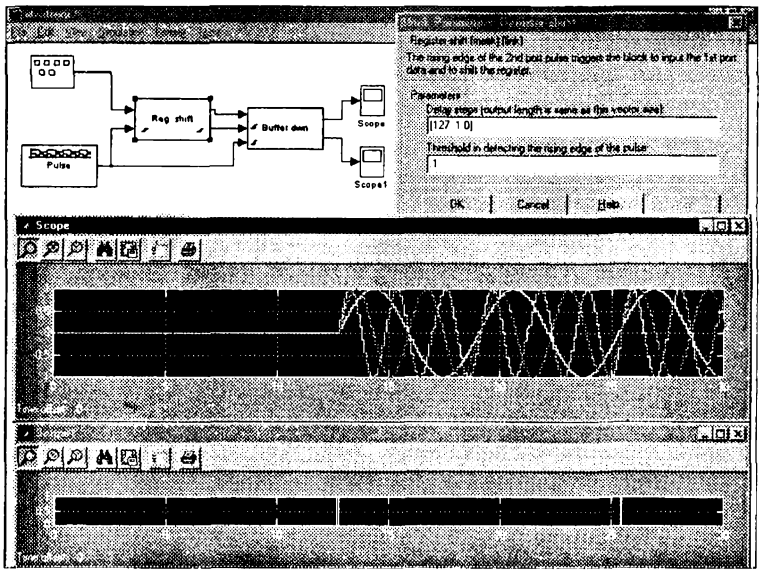


Рис. 6.23. Пример применения блока Register Shift

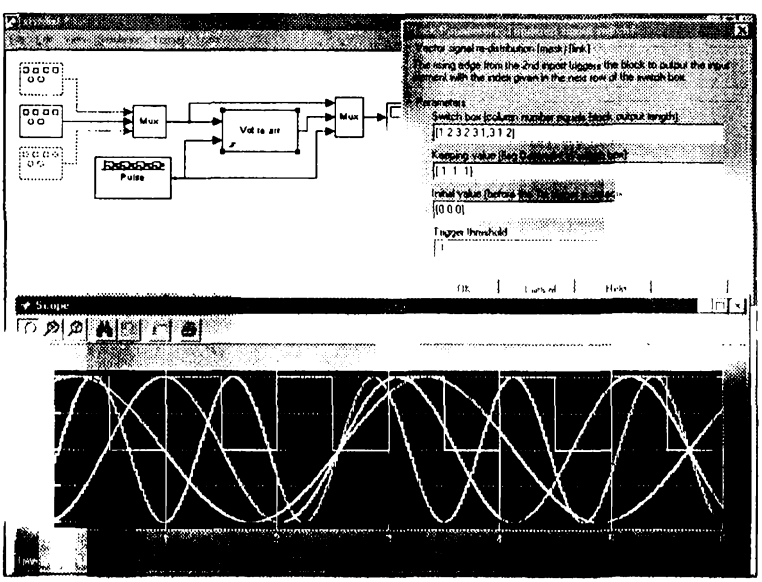


Рис. 6.24. Пример применения блока Vct re-arr

Следующий пример (рис. 6.24) иллюстрирует применение блока Vct re-arr. Как показывают приведенные осциллограммы, данная модель обеспечивает квантование трех синусоидальных сигналов с разными частотами при разных шагах квантования по уровню.

Рисунок 6.25 показывает пример применения блока Switch box demo. Этот блок формирует симметричные разнополярные импульсы — прямоугольные (меандр) и пилообразные. Их осциллограммы представлены слева, а окна установки параметров блоков Switch box и Signal Generator даны справа. В них видно, какие параметры установлены для моделирования в этом примере.

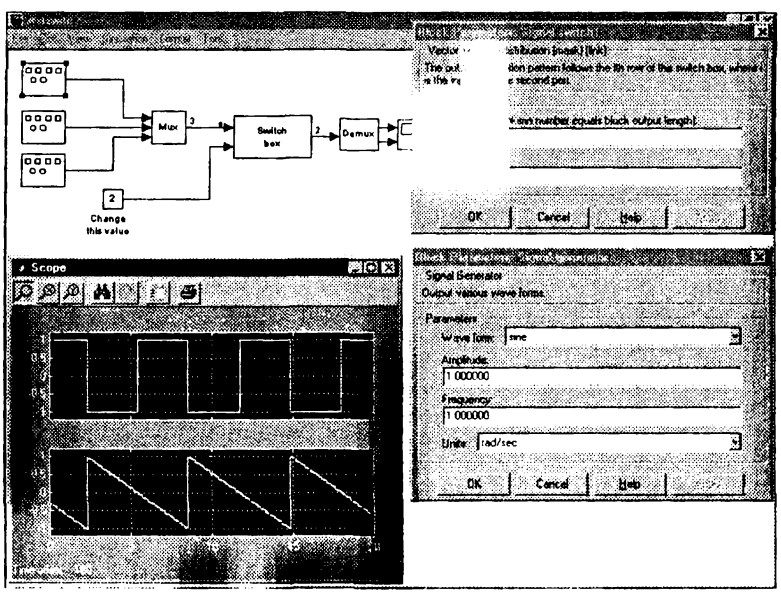


Рис. 6.25. Пример применения блока Switch box

Пример, представленный на рис. 6.26, показывает применение блоков Block Interleave совместно с рядом других блоков. Стоит особо обратить внимание на блок задержки синхронизирующих импульсов с тремя выходами (он представлен в верхней части схемы модели).

Этот пример интересен и применением графических средств представления результатов работы — используется стандартное окно двумерной графики системы MATLAB. Оно представлено на рис. 6.27.

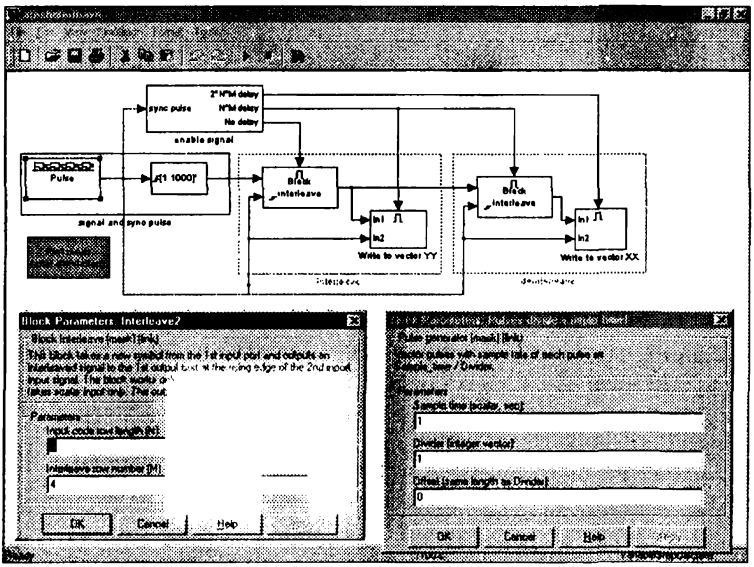


Рис. 6.26. Пример применения блоков Block Interleave

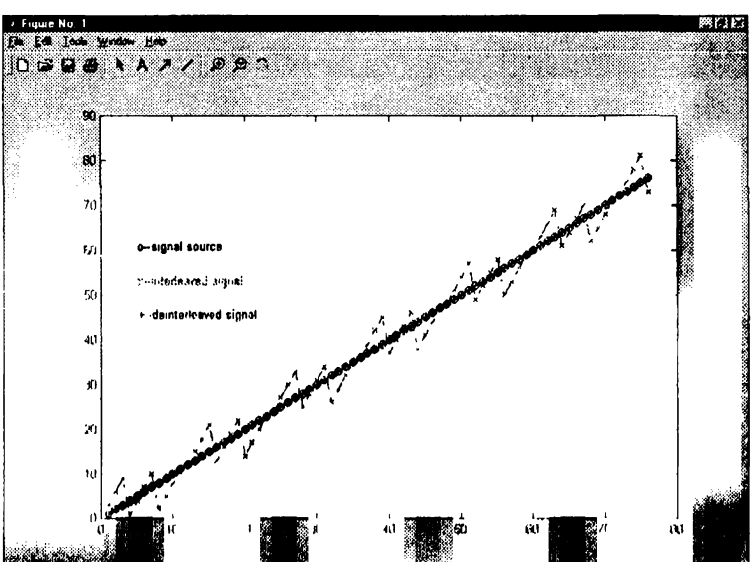


Рис. 6.27. Графическое представление результатов моделирования для примера рис. 6.26

Представление библиотек в виде функциональной схемы

В предшествующей версии MATLAB вызов основного окна с перечнем разделов библиотек осуществлялся довольно оригинально — появлялась обобщенная схема системы связи, каждый блок которой при активизации его мышью вызывал появление соответствующего раздела библиотеки. В MATLAB 5.3.1 от этого отказались в пользу вывода стандартного окна с перечнем разделов библиотек (см. рис. 6.3).

Тем не менее, эта интересная деталь интерфейса пакета сохранилась. Но теперь для вызова подобного представления библиотек надо в командном окне MATLAB исполнить команду `commlibv1`. Окно с перечнем разделов библиотек в виде функциональной схемы, появляющееся при этом, представлено на рис. 6.28. В этом окне также представлено окно устройств синхронизации, которое выводится, если активизировать мышью один из блоков функциональной схемы — Synchronization.

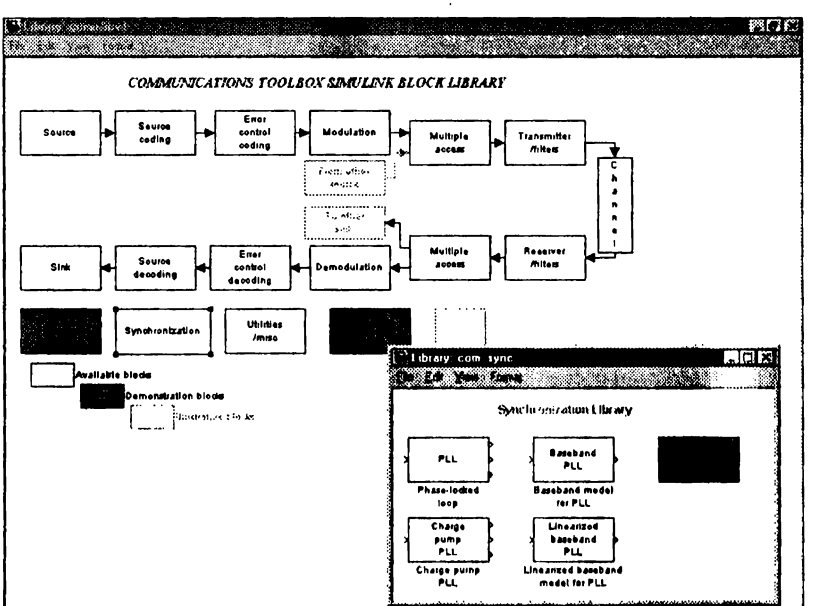


Рис. 6.28. Окно разделов библиотеки в виде функциональной схемы

Представленная на рис. 6.28 функциональная схема дает прекрасное представление об общем характере устройств, которые могут моделироваться с помощью пакета. Активизируя другие блоки функциональной схемы, можно убедиться в разнообразии моделей устройств, которые содержатся в рассмотренных выше библиотеках пакета Communications Blockset. Заметим, что стандартное окно с разделами библиотеки (см. рис. 6.3) вызывается командой `commlibv2`. Команды `commlibv1` и `commlibv2` являются функциями пакета MATLAB, которые могут вызываться из командной строки. Ряд других функций пакета Communications Blockset такого рода мы рассмотрим в следующем разделе.

MATLAB-функции пакета Communications Blockset

Обзор MATLAB-функций пакета

Пакет Communications Blockset имеет множество функций, доступных из командной строки системы MATLAB. Это позволяет пользователю пакета не только «передвигать и вставлять» модели компонентов в общую модель проектируемого устройства или системы но и полноценно использовать средства пакета в своих программных разработках. MATLAB-функции пакета классифицируются по тому же принципу, что и библиотеки моделей компонентов. Поэтому приведенный ниже материал существенно дополняет описание библиотек.

Чтобы функции пакета Communications Blockset были доступны из среды MATLAB, следует установить пакет при инсталляции системы MATLAB и проследить за тем, чтобы в Path-броузере был открыт доступ к этим функциям (рис. 6.29). Обычно при правильной инсталляции MATLAB с выбранными пакетами расширения это обеспечивается автоматически.

Как видно из рис. 6.29, в правом окне Path-броузера можно наблюдать полный список m-файлов функций и файлов DLL-библиотек представленных в алфавитном порядке. Ниже мы рассмотрим эти функции по их категориям. С помощью команды `help name` можно вывести детальные сведения о каждой функции, включая правила их синтаксического задания. Поэтому объем приводимого ниже материала по синтаксису задания функций ограничен.

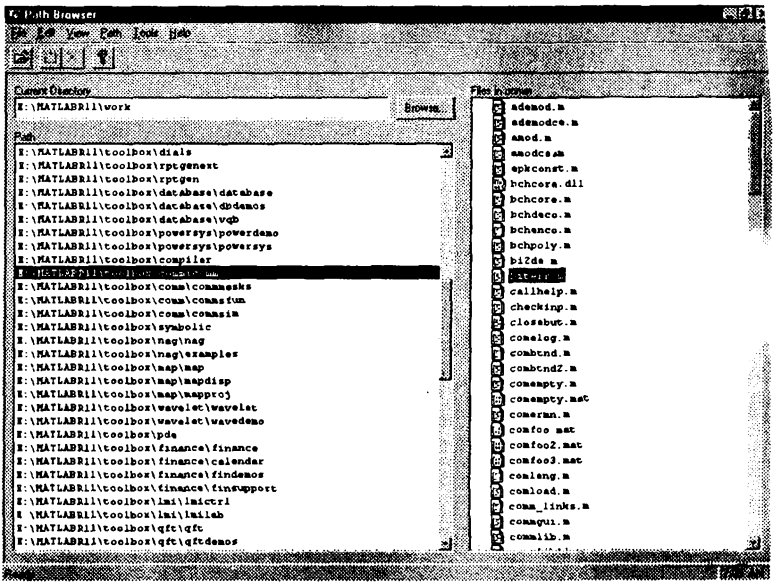


Рис. 6.29. Окно Path-броузера системы MATLAB с указанием пути к библиотеке функций пакета Communications Blockset

Функции построения источников сигналов и анализа ошибок

Для построения источников сигналов и анализа ошибок пакет Communications Blockset имеет ряд функций, указанных ниже:

- `biterr` — число ошибочных битов и вероятность их появления;
- `eyescat` — вывод глазковой диаграммы (eye-pattern diagram) или диаграммы рассеяния (scatter plot);
- `randbit` — генерация матрицы со случайными битами;
- `randint` — генерация матрицы со случайными целыми числами;
- `symerr` — число ошибочных символов и вероятность их появления.

Функция `randint` в основной форме записи `randint(m,n,range,seed)` генерирует матрицу размером $m \times n$ с целыми случайными числами в диапазоне, заданном необязательным параметром `range` (по умолчанию 1). Инициализация генератора случайных чисел производится значением параметра `seed` (тоже необязательный параметр). Подобный формат имеет и функция `randbit`. Приведем примеры их применения, а также применения функций `biterr` и `symerr`:

```
» randbit(3,3,1,5)
ans =
   0   1   0
   1   0   0
   0   1   0
» X=randint(3,3,8)
X =
   4   1   5
   3   7   4
   6   2   2
» Y=randint(3,3,10)
Y =
   1   6   6
   2   8   8
   5   8   4
» symerr(X,Y)
ans =
   9
» A=[1,2;3,4]
A =
   1   2
   3   4
» biterr(X,Y)
ans =
  20
» B=[1,2;3,5]
B =
   1   2
   3   5
» biterr(A,B)
ans =
   1
» symerr(A,B)
ans =
   1
```

В этих примерах обратите внимание на действие функции **symerr** — вначале она показывает наличие 9 ошибок при сравнении двух случайных матриц **X** и **Y**, во втором случае — наличие только одной ошибки при сравнении матриц **A** и **B**. Внимательный просмотр значений матриц убеждает в правильности анализа.

Пример применения функции **eyescat** представлен на рис. 6.30. Работа этой функции иллюстрируется также приведенным ранее рис. 6.8 и примером, представленным на рис. 6.31.

Приведенные примеры показывают обширные возможности функции **eyescat** для графического представления результатов моделирования.

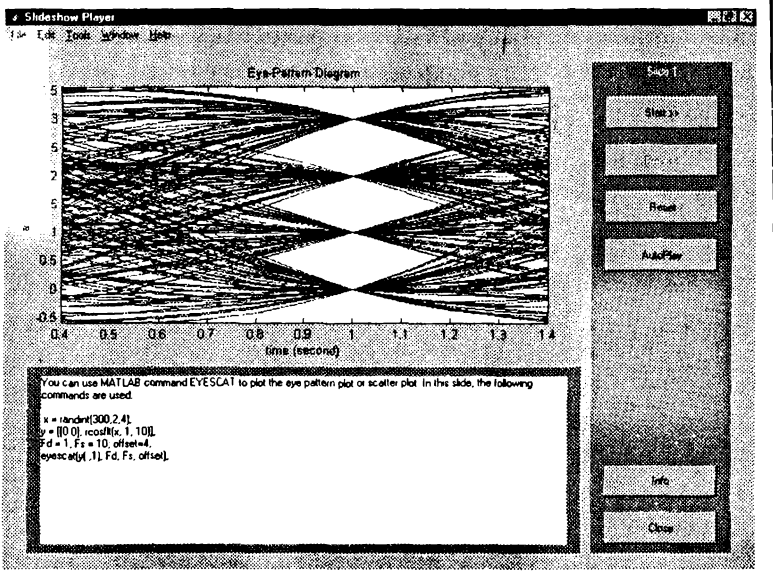


Рис. 6.30. Пример применения функции eyescat

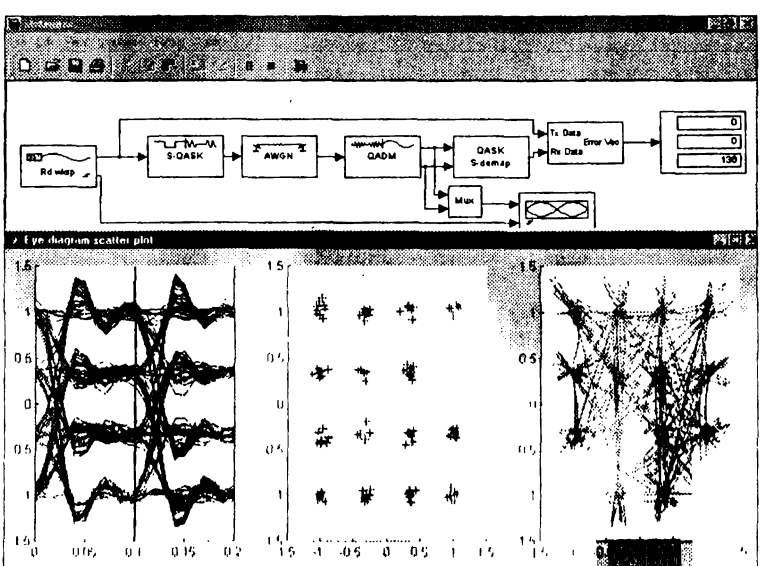


Рис. 6.31. Иллюстрация выполнения примера xtsleyescat

Функции кодирования источника сигнала

Ряд функций кодирования источников сигналов представлен ниже:

- compand — сжатие или восстановление динамического диапазона сигнала согласно стандарту μ-law или A-law;
- dpcmdeco — декодирование при использовании дифференциальной импульсно-кодовой модуляции;
- dpcmenco — обеспечивает кодирование при использовании дифференциальной импульсно-кодовой модуляции;
- dpcmopt — оптимизация параметров дифференциальной импульсно-кодовой модуляции;
- lloyds — скалярная оптимизация параметров квантования на основе алгоритма Ллойда;
- quantiz — осуществляет квантование.

Приведем примеры применения первой и последней функций этого раздела. Для функции compand пример приведен ниже:

```
mu = 255;
sig = -4:.1:4;
sig = exp(sig);
V = max(sig);
[index,quants,distor] = quantiz(sig,0:floor(V),0:ceil(V));
compsig = compand(sig,mu,V,'mu/compressor');
[index,quants] = quantiz(compsig,0:floor(V),0:ceil(V));
newsig = compand(quants,mu,max(quants),'mu/expander');
distor2 = sum((newsig-sig).^2)/length(sig);
[distor, distor2]
ans =
  0.5348  0.0397
```

Пример квантования синусоидального сигнала с помощью функции quantiz представлен ниже:

```
t = [0:.1:2*pi];
sig = sin(t);
partition = [-1:.2:1];
codebook = [-1.2:.2:1];
[index,quants] = quantiz(sig,partition,codebook);
plot(t,sig,'x',t,quants,'.')
axis([-.2 7 -1.2 1.2])
```

Рисунок 6.32 показывает результат выполнения этого примера. На нем показаны исходная синусоида (крестиками) и результат ее квантования (точками).

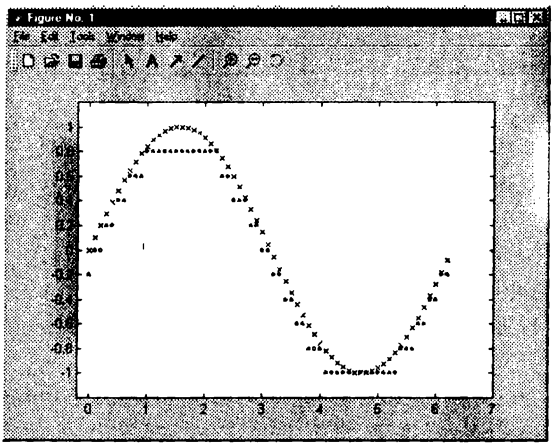


Рис. 6.32. Пример квантования синусоиды функцией quantize

В данном случае отчетливо видно, что квантование идет при равных шагах по уровню, что приводит к изменению длительности «ступенек» квантованного сигнала.

Помехоустойчивое кодирование

Для работы с корректирующими кодами служат следующие функции:

- `bchpoly` — создает полиномиальное представление бинарных БЧХ-кодов;
- `cyclgen` — создает порождающую матрицу и матрицу проверок четности для циклического кода;
- `cyclpoly` — создает полиномиальное представление для циклического кода;
- `decode` — обеспечивает исправление ошибок при декодировании различными методами;
- `encode` — обеспечивает кодирование с использованием различных корректирующих кодов;
- `gen2abcd` — преобразует функцию передачи сверточного кода в набор матриц **A**, **B**, **C**, **D**, задающих этот код в пространстве состояний;
- `gen2par` — обеспечивает преобразование матрицы проверки четности в порождающую матрицу и наоборот;

- hammgen — создает порождающую матрицу и матрицу проверок четности для кодов Хемминга;
- htruthtb — генерирует таблицу истинности для кодов Хемминга;
- oct2gen — преобразует матрицу функции передачи сверточного кода из восьмеричной формы в двоичную;
- rsdecof — обеспечивает декодирование текстового ASCII-файла, закодированного с использованием кода Рида—Соломона;
- rsencof — обеспечивает кодирование текстового ASCII-файла с использованием кода Рида—Соломона;
- rspoly — создает полиномиальное представление кода Рида—Соломона;
- sim2gen — преобразует модель сверточного кодера, созданную в Simulink, в функцию передачи данного сверточного кода (для линейных кодов);
- sim2logi — преобразует модель сверточного кодера, созданную в Simulink, в функцию передачи данного сверточного кода (для нелинейных кодов);
- sim2tran — преобразует модель сверточного кодера, созданную в Simulink, в функцию передачи данного сверточного кода (для линейных и нелинейных кодов).

Приведем пример использования функций кодирования encode и декодирования decode:

```
m = 4; n = 2^m-1; k = 5; t = 3;
msg = ones(10,1);
code = encode(msg,n,k,'bch');
noisycode = code;
noisycode(1:2) = bitxor(noisycode(1:2),[1 1]');
noisycode(16) = bitxor(noisycode(16),1);
[newmsg,err,ccode,cerr] = decode(noisycode,n,k,'bch',t);
disp('Transpose of err is'); disp(err')
disp('Transpose of cerr is'); disp(cerr')
Transpose of err is
   2  2  2  2  2  1  1  1  1  1
Transpose of cerr is
 Columns 1 through 12
   2  2  2  2  2  2  2  2  2  2  2  2
 Columns 13 through 24
   2  2  2  1  1  1  1  1  1  1  1  1
 Columns 25 through 30
   1  1  1  1  1  1
```

Следует отметить, что для большинства функций этой категории в фирменном описании, несмотря на большой его объем, примеры не приводятся. Возможно, это связано с тем, что данные функции служат «для внутреннего употребления» — они используются для реализации соответствующих моделей в виде блоков Simulink.

Низкоуровневые функции помехоустойчивого кодирования

К низкоуровневым функциям помехоустойчивого кодирования относятся следующие функции:

- `bchcore` — ядро алгоритма декодирования кодов БЧХ;
- `bchdeco` — декодирование кодов БЧХ;
- `convenco` — кодирование с использованием сверточного кода;
- `errlocp` — вычисление полиномиального представления локатора ошибок для кодов Рида–Соломона и БЧХ;
- `rsdeco` — декодирование кодов Рида–Соломона с использованием различных форматов представления данных;
- `rsdecode` — декодирование кодов Рида–Соломона с экспоненциальным форматом представления данных;
- `rsenco` — кодирование с использованием кодов Рида–Соломона и различных форматов представления данных;
- `rsencode` — кодирование с использованием кодов Рида–Соломона и экспоненциального формата представления данных;
- `viterbi` — декодирование сверточного кода с использованием алгоритма Витерби.

Эти функции самостоятельного интереса, как правило, не представляют, поскольку используются функциями более высокого уровня. В связи с этим ограничимся указанием их имен и назначения.

Модуляция и демодуляция

Как отмечалось, модели процессов модуляции и демодуляции занимают центральное место в арсенале средств моделирования коммуникационных устройств. В связи с этим они имеют свои аналоги среди функций, доступных для использования в командном режиме работы системы MATLAB. Ниже представлены функции этой группы и их назначение:

- `ademod` — аналоговая демодуляция (функция, обратная `amod`);
- `ademodce` — аналоговая демодуляция комплексной огибающей (функция, обратная `amodce`);

- amod — аналоговая модуляция с выбором метода модуляции;
- amodce — аналоговая модуляция комплексной огибающей с выбором метода модуляции;
- apkconst — вычисляет и строит диаграммы круговых сигнально-кодовых конструкций (созвездий), используемых при квадратурной (QASK) модуляции;
- ddemod — цифровая демодуляция (функция, обратная dmod);
- ddemodce — цифровая демодуляция комплексной огибающей (функция, обратная dmodce);
- demodmap — преобразование аналоговых параметров модулированного сигнала в дискретный код (функция, обратная modmap);
- dmod — цифровая модуляция с возможностью выбора метода модуляции;
- dmodce — цифровая модуляция комплексной огибающей с выбором метода модуляции;
- modmap — преобразование дискретного кода в аналоговые параметры модуляции;
- qaskdeco — преобразование аналоговых параметров модулированного сигнала в дискретный код при использовании квадратурной модуляции (QASK) с квадратной сигнально-кодовой конструкцией (функция, обратная qaskenco);
- qaskenco — преобразование дискретного кода в аналоговые параметры модуляции при использовании квадратурной модуляции (QASK) с квадратной сигнально-кодовой конструкцией.

Каждая из функций имеет несколько форм записи. Например, функция amod записывается в следующих десяти формах:

```
y = amod(x,Fc,Fs,'amdsb-sc');
y = amod(x,Fc,Fs,'amdsb-tc',offset);
y = amod(x,Fc,Fs,'amssb/opt');
y = amod(x,Fc,Fs,'amssb/opt',num,den);
y = amod(x,Fc,Fs,'amssb/opt',hilbertflag);
y = amod(x,Fc,Fs,'qam');
y = amod(x,Fc,Fs,'fm',deviation);
y = amod(x,Fc,Fs,'pm',deviation);
y = amod(x,Fc,[Fs phase],...);
[y,t] = amod(...);
```

Здесь:

- x — представление сигнала;
- Fc — несущая частота;

- Fs — частота модулирующего сигнала;
- offset = -min(min(x));
- deviation — девиация частоты (по умолчанию 1);
- hilbertflag — флаг для задания фильтрации по Гильберту;
- num — числитель передаточной характеристики фильтра Гильберта;
- denum — знаменатель передаточной характеристики фильтра Гильберта;
- phase — фазовый угол.

В апострофах указывается параметр, определяющий тип модуляции:

- 'amdsb-tc' — амплитудная модуляция с двумя боковыми полосами при наличии несущей;
- 'amdsb-sc' — амплитудная модуляция с двумя боковыми полосами и подавлением несущей;
- 'amssb' или 'amssb/up' — амплитудная модуляция с одной боковой полосой и подавлением несущей;
- 'qam' — квадратурная амплитудная модуляция;
- 'fm' — частотная модуляция;
- 'pm' — фазовая модуляция.

В дальнейшем мы не будем приводить все формы записи функций, поскольку этого не позволяет объем данной главы. Кроме того, с форматом записи любой функции нетрудно ознакомиться с помощью справки по ней, и это не требует серьезного знания английского языка. С работой функций мы познакомимся на практических примерах. Приведенные для функции amod параметры являются характерными для большинства других функций этой группы.

Следующий пример (m-файл) показывает моделирование процесса аналоговой модуляции для двух сигналов с помощью функции amod, а затем их демодуляцию с помощью функции ademod:

```
Fc = 25; Fs = 100; t = [0:1/Fs:10]';
x = [cos(t),t/5];
y = amod(x,Fc,Fs,'amdsb-tc',.3);
z1 = ademod(y,Fc,Fs,'amdsb-tc',.3);
z2 = ademod(y,Fc,Fs,'amdsb-tc');
plot(t,z1,'b',t,z2,'r--')
```

В конце примера строятся графики этих сигналов — гармонического и линейно нарастающего, а также сигналов, полученных после моделирования процессов модуляции и демодуляции. Эти графики показаны на рис. 6.33.

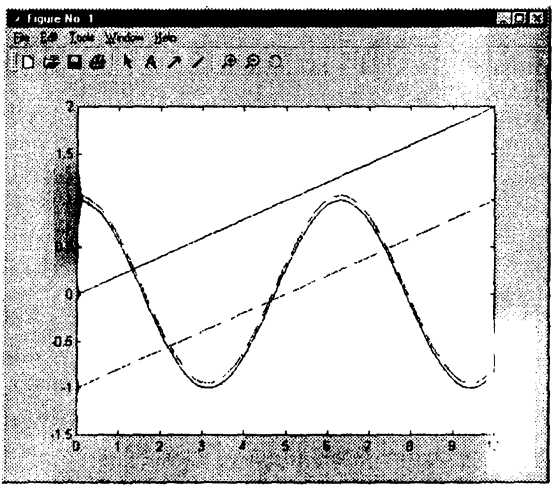


Рис. 6.33. Графики двух сигналов до и после процесса аналоговой модуляции-демодуляции

Нетрудно заметить, что, за исключением некоторого (особенно заметного на линейно нарастающем сигнале) сдвига по уровню, временная зависимость сигнала после прохождения им тракта модуляции-демодуляции повторяет временную зависимость исходного сигнала. Это говорит о малых искажениях формы сигнала.

Следующий пример демонстрирует процесс однополосной модуляции и демодуляции гармонического сигнала с использованием фильтра Гильберта, заданного функцией `hilbiir`:

```
Fs = 100;
[numh,denh] = hilbiir(1/Fs,15/Fs,15);
t = [0:1/Fs:5]'; x = cos(t);
y = amodce(x,[Fs pi/4],'amssb/time',numh,denh);
z = ademodce(y,[Fs pi/4],'amssb');
d = ceil(log10(sum(abs(x-z))/length(x)))
d =
  -16
```

Приведенный ниже пример показывает процесс моделирования цифровой модуляции и демодуляции:

```
M = 16; Fc = 10; Fd = 1;
Fs = 50; x = randint(100,1,M);
y = dmod(x,Fc,Fd,Fs,'ask',M);
ynoisy = y + .01*randn(Fs/Fd*100,1);
z = ddemod(ynoisy,Fc,Fd,Fs,'ask',M);
s = symerr(x,z)
```

```
s =
    0
```

Здесь с помощью функции randint создается цифровое сообщение из 100 случайных бинарных кодов. Затем оно используется при модуляции, выполняемой с помощью функции dmod, что создает вектор y «чистого» цифрового сигнала. К этому сигналу подмешивается небольшой по уровню шум, созданный с помощью функции randn. В результате получается вектор зашумленного цифрового сообщения ynoisy. Этот вектор демодулируется с помощью функции ddemod. В заключение функция symerr выполняет сравнение исходного вектора x и вектора, полученного после процессов модуляции-демодуляции. Равенство s нулю после исполнения этого программного фрагмента говорит о том, что шум в данном случае абсолютно не сказался на передаче сигнала в ходе цифровой модуляции и демодуляции.

Функция apkconst строит комбинированное (в виде окружностей и точек) графическое представление амплитуд и фаз, используемых при квадратурной модуляции с круговыми сигнально-кодовыми конструкциями. Например, при исполнении в командной строке MATLAB следующей команды

```
» apkconst ([2 3 5],[1 4 5],[0 pi pi/4])
```

будет выведено окно с диаграммой, показанной на рис. 6.34.

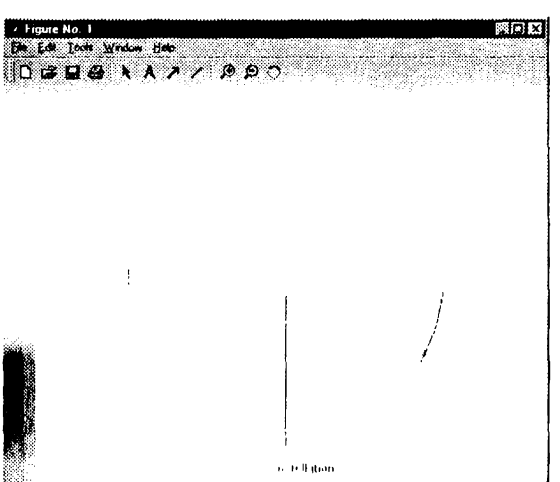


Рис. 6.34. Пример построения круговой сигнально-кодовой конструкции с помощью функции apkconst

Вызов функции

```
» qaskenco([0 3 4 3 2 5],8)
```

строит график, показанный на рис. 6.35. Этот график создает «квадратное созвездие» точек, представляющих амплитуды и фазы цифрового сигнала, заданного массивом — параметром функции. Каждая точка помечается своим номером. Элементы массива (в нашем случае вектора) могут быть целыми числами, имеющими значения от 0 до $m - 1$ (в нашем случае $m = 8$).

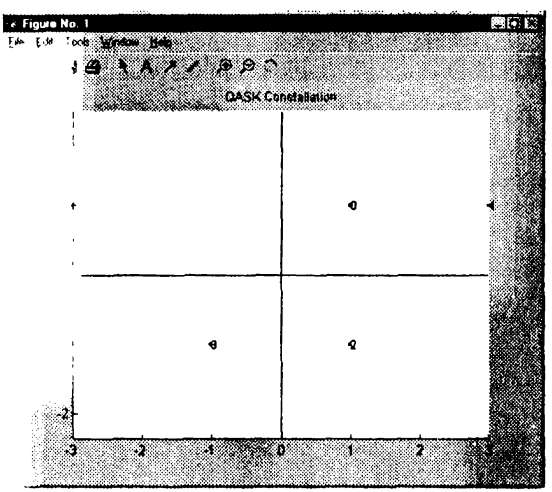


Рис. 6.35. Пример квадратурной цифровой модуляции

В заключение приведем еще пару примеров использования функций qaskenco и qaskdeco:

```
[inphase,quadr] = qaskenco([0 3 5 3 2 5],8);
inphase'
ans =
     1    -1    -3    -1     1    -3
quadr'
ans =
     1    -1     1    -1    -1     1
msg = [0 3 5 3 2 5]'; M = 8;
[inphase,quadr] = qaskenco(msg,M);
newmsg = qaskdeco(inphase,quadr,M)
newmsg =
     0
     3
```

```
5
3
2
5
```

Для более полного знакомства с различными формами записи описанных выше функций рекомендуется пользоваться фирменным описанием и справочной системой. В последней имеется множество примеров, иллюстрирующих применение различных видов модуляции. Все они построены по принципу уже описанных примеров: задается генератор модулирующего сигнала, осуществляется модуляция заданного вида, а затем соответствующая демодуляция. В заключение исходный сигнал сравнивается с сигналом, полученным после демодуляции. Читатель может просмотреть эти примеры и выяснить, какой вид модуляции наиболее пригоден для решения его конкретных задач.

Специальные фильтры

Помимо модуляторов и демодуляторов при построении систем связи важное место принадлежит фильтрам. Фильтры в большинстве случаев необходимы для очистки полезного сигнала от шумов и различных других паразитных составляющих.

Из функций, относящихся к области проектирования фильтров, и, кстати, заметно отличающихся по набору даже у близких версий пакета Communications Blockset, отметим несколько наиболее важных:

- `hilbiir` — расчет рекурсивного фильтра, приближенно выполняющего преобразование Гильберта;
- `hank2sys` — преобразует матрицу Ханкеля в передаточную функцию дискретной системы;
- `imp2sys` — создает линейную модель дискретной системы по ее импульсной характеристике;
- `rcosfir` — расчет нерекурсивного фильтра с косинусоидальным сглаживанием АЧХ (raised cosine);
- `rcosflt` — фильтрация сигнала с использованием фильтра с косинусоидальным сглаживанием АЧХ;
- `rcosiir` — расчет рекурсивного фильтра с косинусоидальным сглаживанием АЧХ;
- `rcosine` — расчет фильтра с косинусоидальным сглаживанием АЧХ с возможностью выбора типа фильтра.

Ввиду ограниченного объема книги мы не будем детально рассматривать свойства тех или иных фильтров, а ограничимся несколькими

примерами построения фильтров средствами пакета Communications Toolbox. Краткое описание фильтров можно найти в справочной системе пакета и в документации по нему.

Фильтр Гильберта имеет импульсную характеристику следующих видов:

$h(t) = 1/\pi t$ фильтр без задержки,

$h(t) = 1/\pi(t - dly)$ фильтр с задержкой.

На рис. 6.36 представлен пример построения характеристики фильтра Гильберта по команде

```
» hilbiir(.2,1.3)
```

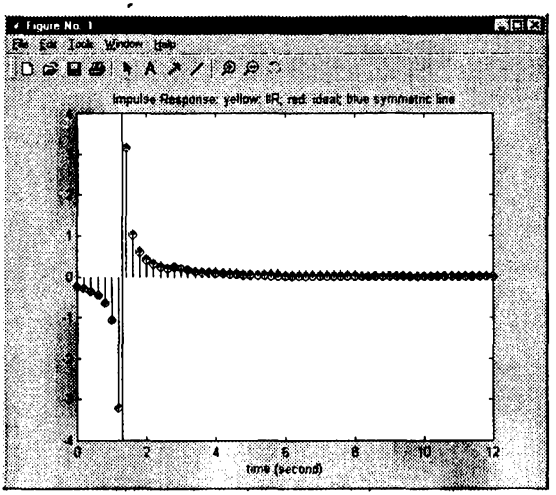


Рис. 6.36. Пример построения характеристики фильтра Гильберта

Еще один пример применения этой функции рассматривался выше (фильтр Гильберта использовался при моделировании однополосной модуляции).

Следующий пример демонстрирует применение функции hank2sys:

```
» h = hankel([1 0 1]);
» [num,den,sv] = hank2sys(h,0,.01)
num =
     0         1.0000    0.0000    1.0000
den =
     1.0000    0.0000   -0.0000    0.0000
sv =
```

```
1.6180
1.0000
0.6180
```

Функция `imp2sys` служит для построения линейной модели дискретной системы с заданной импульсной характеристикой `impulse` при заданной погрешности `tol` и порядке `ord`. При задании `tol>1` величина `ceil(tol)` есть порядок создаваемой системы. По умолчанию погрешность `tol=0.01`. Возможны четыре формы записи этой функции:

```
[num,den] = imp2sys(impulse,tol,ord);
[num,den,sv] = imp2sys(…);
[a,b,c,d] = imp2sys(…);
[a,b,c,d,sv] = imp2sys(…);
```

где `a`, `b`, `c` и `d` — матрицы, используемые системой MATLAB при составлении уравнений, описывающих модель системы. Пример применения функции `imp2sys` приведен ниже:

```
[a,b,c,d] = drmodel(5,1,1);
imp = dimpulse(a,b,c,d,1,100);
[a1,b1,c1,d1] = imp2sys(imp,5);
imp1 = dimpulse(a1,b1,c1,d1,1,100);
plot([imp,imp1]);
```

Построенный в этом примере график импульсных характеристик двух систем представлен на рис. 6.37. Обе построенные характеристики на графике практически совпадают.

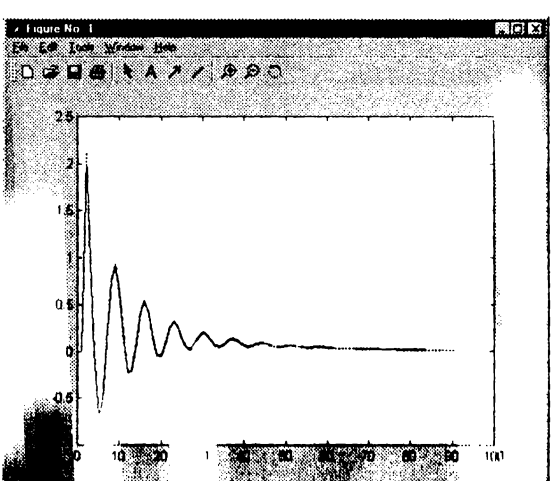


Рис. 6.37. Иллюстрация применения функции imp2sys

Функция rcosiir создает так называемый фильтр с косинусоидальным сглаживанием АЧХ (raised-cosine filter), импульсная характеристика которого описывается выражением:

$$h(t) = \frac{\sin\left(\frac{\pi t}{T}\right)}{\frac{\pi t}{T}} \cdot \frac{\cos\left(\frac{\pi R t}{T}\right)}{1 - \frac{4R^2 t^2}{T^2}}.$$

Здесь R — коэффициент сглаживания АЧХ (ширина зоны сглаживания), T — период дискретизации входного сигнала. Эта функция записывается в одном из следующих форматов:

```
[num,den] = rcosiir(R,T_delay,rate,T,tol);
[num,den] = rcosiir(R,T_delay,rate,T,tol,filter_type);
rcosiir(…);
rcosiir(…,color);
[num,den,sample_time] = rcosiir(…);
```

Здесь помимо указанных выше параметров используются следующие:

- T_delay — время задержки;
- rate — коэффициент повышения частоты дискретизации от входа к выходу;
- tol — погрешность преобразования.

Параметры могут не указываться, в этом случае они задаются по умолчанию: T_delay=3, rate=5, T=1 и tol=0.01. Ниже представлен пример применения функции rcosiir:

```
rcosiir(0,10);
subplot(211); hold on;
subplot(212); hold on;
col = ['r-';'g-';'b-';'m-';'c-';'w-'];
R = [8,6,4,3,2,1];
for i = R
   rcosiir(0,i,[],[],[],[],col(find(R==i),:));
end;
```

Этот пример строит графики АЧХ и ФЧХ данного фильтра, представленные на рис. 6.38 для различных значений временной задержки (параметра T_delay), берущихся из вектора R.

Этими примерами мы ограничимся. С другими видами фильтров читатель может познакомиться самостоятельно.

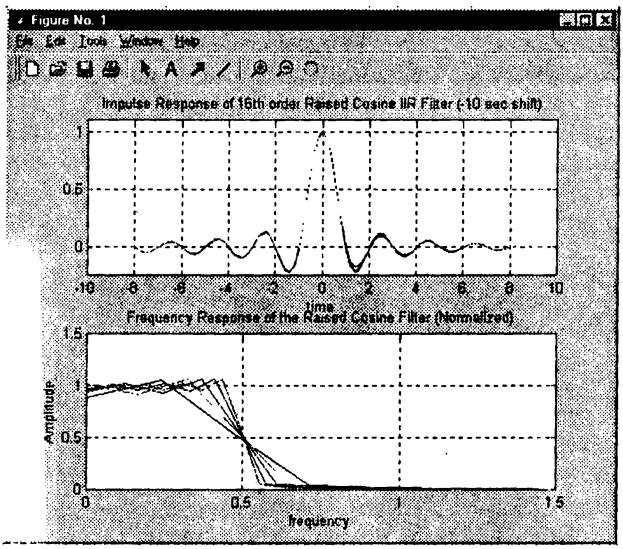


Рис. 6.38. Пример построения фильтра функцией rcosiir

Утилиты и другие функции

К утилитам относится ряд разнохарактерных функций:

- `bi2de` — преобразование двоичных векторов в десятичные числа;
- `blkdiag` — создание блочной диагональной матрицы;
- `callhelp` — возвращает заданную секцию из текстового файла справки;
- `checkinp` — проверка входных параметров и замена пустых параметров значениями по умолчанию (до 10 параметров);
- `de2bi` — преобразование целых десятичных чисел в двоичные векторы;
- `vec2mat` — преобразует вектор в матрицу с заданным числом столбцов.

Следующие примеры иллюстрируют применение некоторых из этих функций:

```
» bi2de([1 1 0 1 0 1])
ans =
  43
» de2bi(43)
```

```
ans =
   1   1   0   1   0   1
» blkdiag([1,2;3,4],[5,6;7,8])
ans =
     1     2     0     0
     3     4     0     0
     0     0     5     6
     0     0     7     8
» vec2mat([1 2 3],1)
ans =
   1
   2
   3
» vec2mat([1,2,3],2)
ans =
     1     2
     3     0
» vec2mat([1,2,3],3)
ans =
     1     2     3
» x1=1;x2=2;z1=1;z2=2; [y1,y2]=checkinp(x1,x2,z1,z2)
y1 =
   1
y2 =
   2
» x1=1;x2='';z1=1;z2=4; [y1,y2]=checkinp(x1,x2,z1,z2)
y1 =
     1
y2 =
     4
```

Последние примеры применения функции checkinp требуют некоторого пояснения. Эта функция возвращает набор параметров yi, проверяя ввод xi. Если все xi введены, то yi принимают значения соответствующих xi (см. первый пример применения этой функции). Но если какие-то xi пусты, то для yi используются соответствующие значения по умолчанию — zi (см. второй пример применения этой функции). Значение i не может превышать 10.

Ряд функций пакета Communications Toolbox реализует возможности полей Галуа — от проверки чисел на простоту до осуществления операций свертки и фильтрации. В связи с ограниченным применением этого весьма специфического аппарата мы не будем описывать относящиеся к нему функции. Заинтересованный читатель может познакомиться с ними самостоятельно.

Комплексные демонстрационные примеры

Доступ к примерам

Приведенные выше примеры ориентированы в основном на демонстрацию отдельных моделей. Однако пакет Communications Blockset содержит ряд комплексных примеров, рассмотрение которых позволяет сделать выводы о характере решаемых пакетом задач и качестве визуализации получаемых при этом результатов. Для доступа к этим примерам надо в окне MATLAB исполнить команду меню Help ▸ Examples and Demos. Появится окно демонстрационных примеров, открытое для запуска примеров, в том числе примеров пакета Communications Blockset. Рисунок 6.39 показывает это окно в момент, предшествующий запуску первого примера из группы примеров данного пакета.

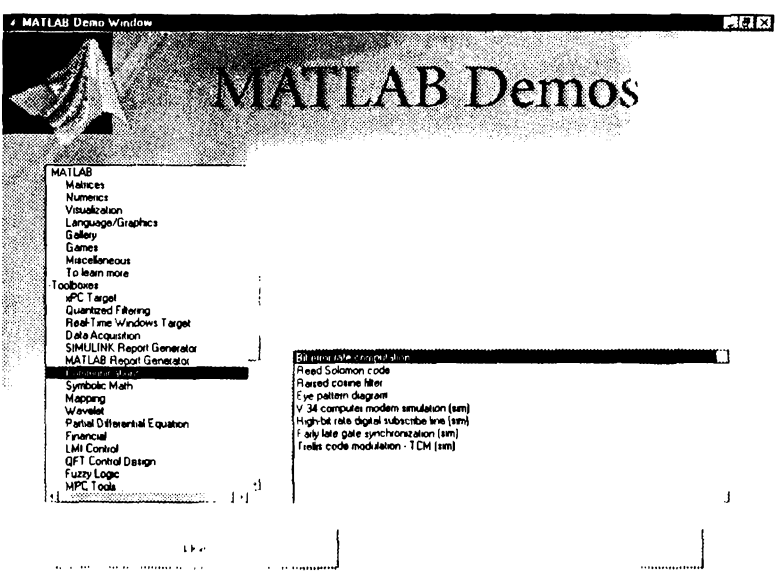


Рис. 6.39. Окно демонстрационных примеров MATLAB Demos

В левой части окна расположен список категорий демонстрационных задач (в нашем случае это Toolboxes ▸ Communications), а в правой - список для выбора примеров.

Моделирование фильтра

На рис. 6.40 показан один из примеров — моделирование фильтра с косинусоидальным сглаживанием АЧХ (Raised cosine filter).

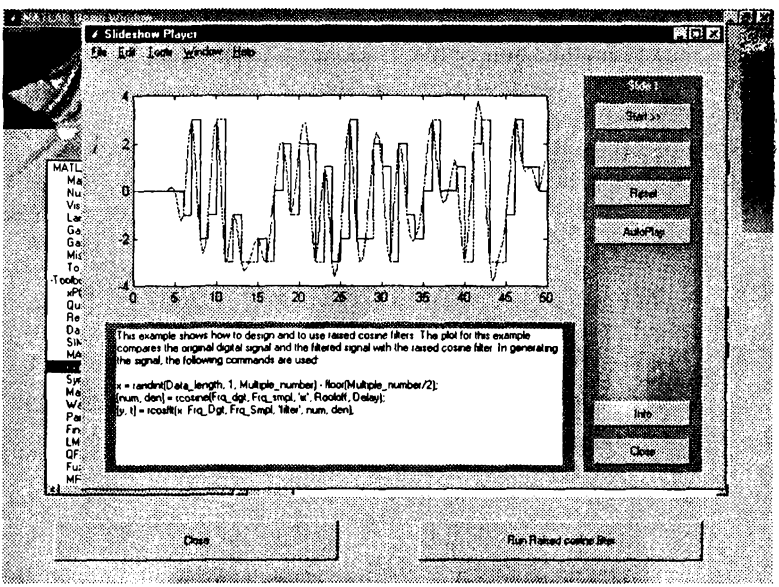


Рис. 6.40. Демонстрационный пример Raised cosine filter

Для продолжения моделирования нужно нажать кнопку Start>> в окне, показанном на рис. 6.40. Кнопка AutoPlay запускает режим автоматического просмотра всех кадров демонстрационного примера. Кнопка Info позволяет получить дополнительную информацию о примере.

Моделирование модема

Рассмотрим еще один пример такого рода — моделирование работы модема, реализующего протокол V.34. Рисунок 6.41 демонстрирует модель модема и одну из ее подсистем.

Этот пример демонстрирует разнообразие средств представления информации. Одно из таких средств, до сих пор нами не рассмотренное, показано на рис. 6.42. Это панель контроля параметров модема.

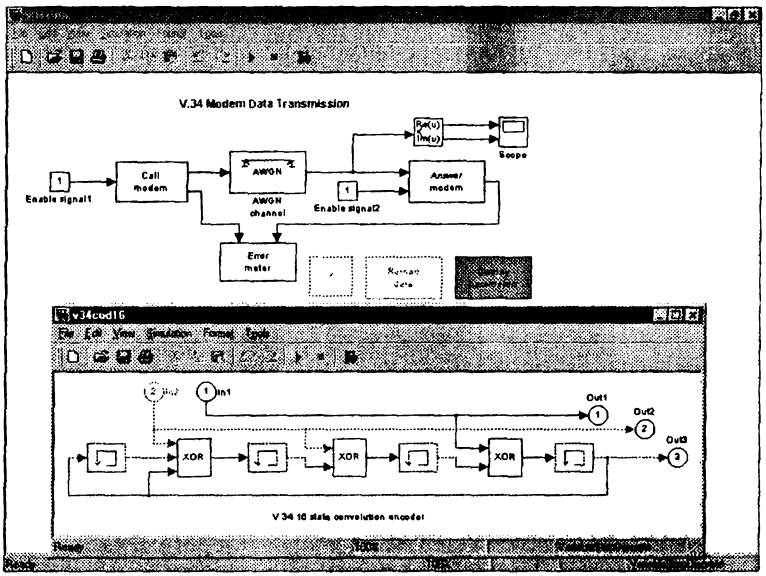


Рис. 6.41. Модель модема и одна из ее подсистем

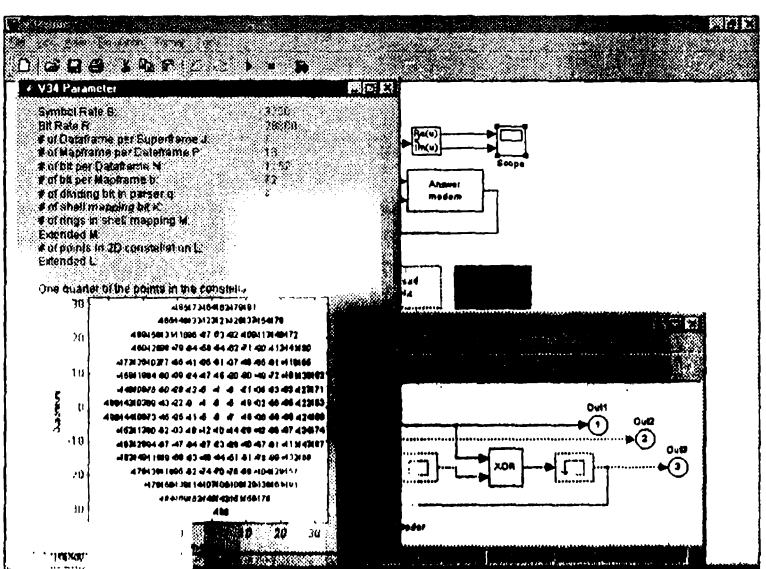


Рис. 6.42. Панель контроля параметров модема

Другие средства (рис. 6.43) более традиционные. Это панель контроля ошибок, дающая динамическое представление о прохождении сложного цифрового сигнала по тракту модема, и обычный виртуальный (в данном случае двухканальный) осциллограф. Динамика представления заключается в постоянной смене данных таблицы и осциллограмм в ходе моделирования.

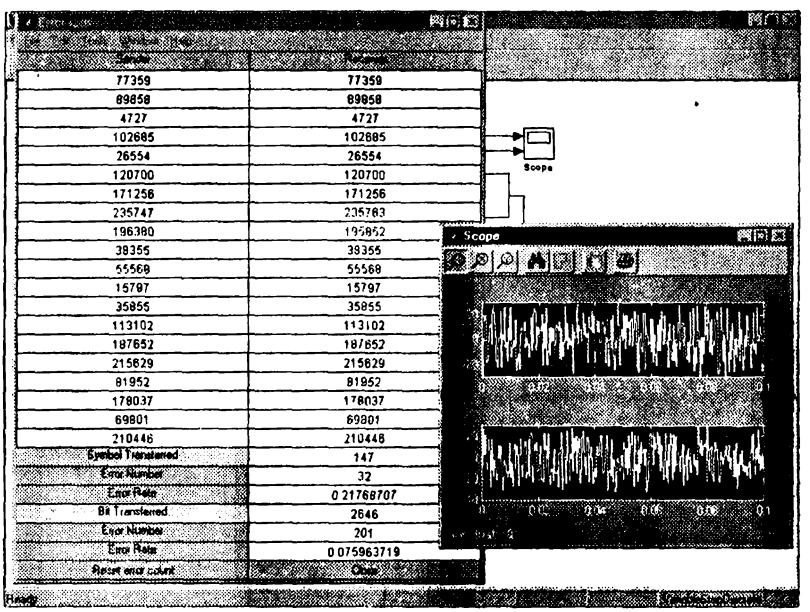


Рис. 6.43. Средства оперативного контроля работы модема

Моделирование высокоскоростной цифровой линии связи

Пример моделирования работы высокоскоростной цифровой линии связи, использующей амплитудно-импульсную модуляцию, представлен на рис. 6.44. Линия состоит из источника сигнала, передатчика, канала связи и приемника. Субмодуль цифрового приемника (Receiver) также представлен на рис. 6.44, вместе с осциллограммами, характеризующими форму исходного и прошедшего по линии сигналов.

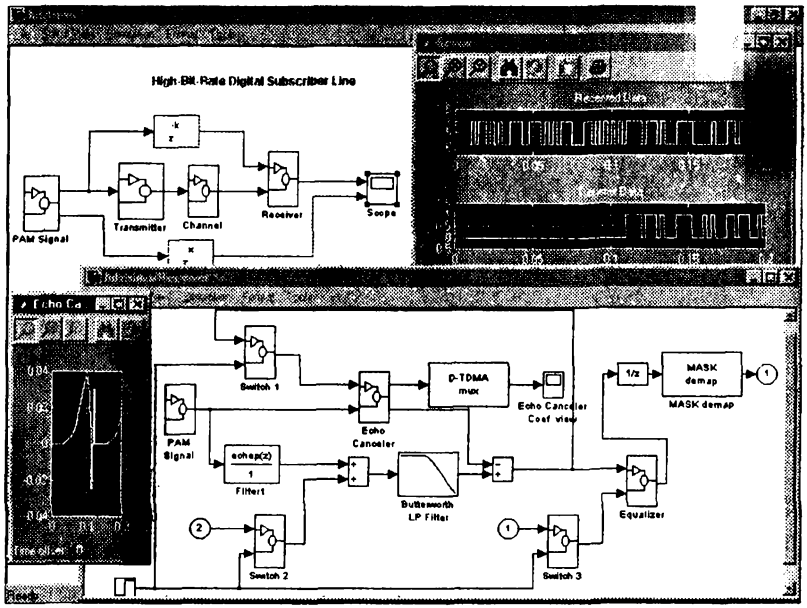


Рис. 6.44. Пример моделирования высокоскоростной цифровой линии связи

Моделирование системы синхронизации

Пример моделирования специальной системы синхронизации Early-late gate synchronization приведен на рис. 6.45. На этом рисунке внизу показана подсистема цифрового интегратора, а справа — результат работы измерителя ошибок.

Осциллограммы, иллюстрирующие работу системы синхронизации, представлены на рис. 6.46. В данном случае панель виртуального осциллографа системы MATLAB, использованного для отображения временных диаграмм, развернута на весь экран. Она показывает высокую разрешающую способность осциллографа. Хорошо видны даже мелкие детали сложных осциллограмм (к примеру, средней на рис. 6.46).

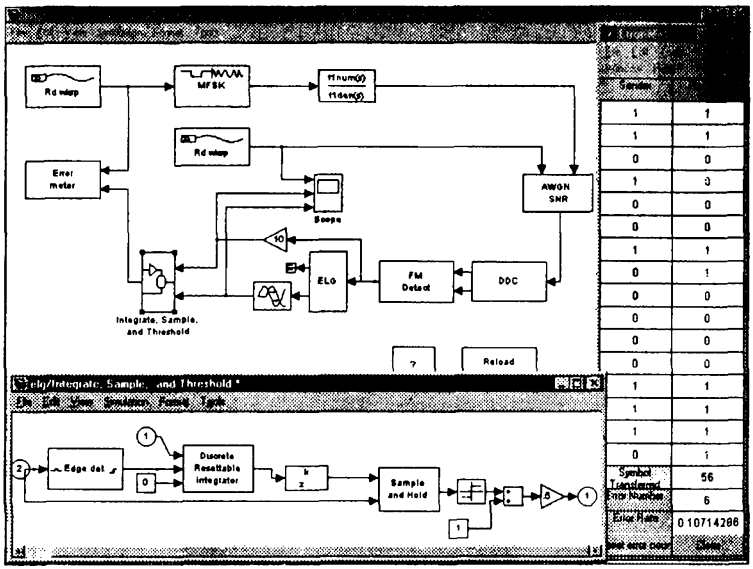


Рис. 6.45. Пример моделирования системы синхронизации Early-late gate synchronization

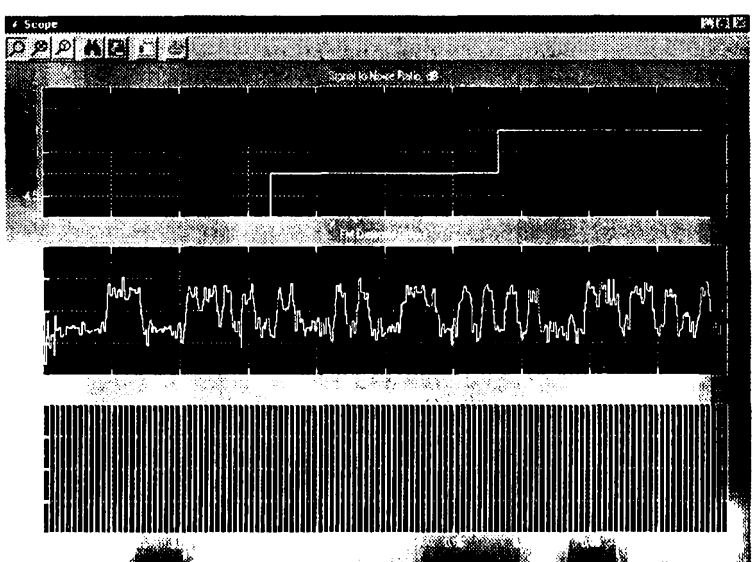


Рис. 6.46. Осциллограммы сигналов в системе синхронизации Early-late gate synchronization

Представление комментариев к примерам

Многие примеры применения пакета Communications Blockset содержат специальные комментарии, оформленные в виде блока с вопросительным знаком. Выполнив двойной щелчок на таком блоке, можно открыть окно с аннотационным описанием моделируемой системы или устройства. Так, на рис. 6.47 представлено такое описание для системы синхронизации из предыдущего примера (см. рис. 6.45).

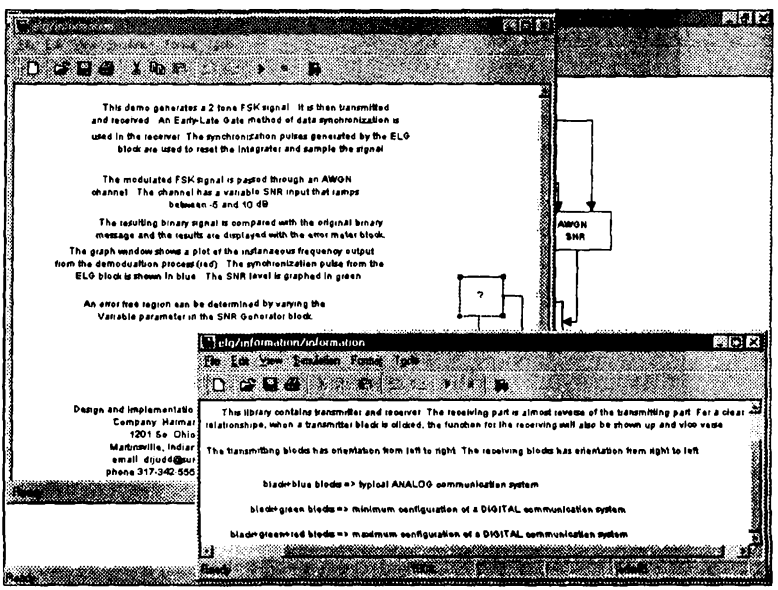


Рис. 6.47. Аннотационное описание системы Early-late gate synchronization

Внутри окна описания имеется еще один блок описания. Его окно также представлено на рис. 6.47. Естественно, что для фирменных демонстрационных примеров это описание дается на английском языке. Однако ничто не мешает нашему пользователю в своих примерах готовить аннотации на русском языке или переводить на русский язык уже имеющиеся аннотации.

С другими демонстрационными примерами читатель может легко познакомиться самостоятельно. После этого остается сделать последний шаг — приступить к моделированию своих систем связи.

Глава 7
Пакеты Stateflow и RTW

Назначение пакета событийного моделирования Stateflow

Взаимодействие системы MATLAB с пакетами расширения

Система «MATLAB+Simulink+Описанные пакеты расширения» является достаточной основой для идентификации, анализа и моделирования систем и устройств в среде «MATLAB+Simulink». Однако система MATLAB предлагает еще ряд дополнительных возможностей моделирования. Прежде всего, они реализованы двумя особо важными компонентами — пакетом расширения Stateflow, предназначенным для событийного моделирования систем и устройств, и лабораторией работы в реальном масштабе времени Real-Time Workshop (RTW). Взаимодействие этих компонентов с системой «MATLAB+Simulink» и друг с другом представлено на рис. 7.1. Не следует путать эту систему взаимодействия с моделируемой системой.

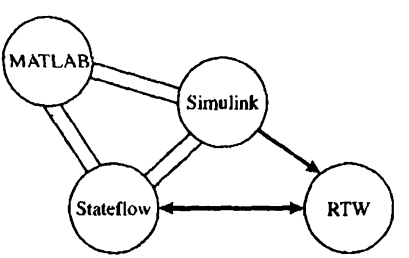


Рис. 7.1. Взаимодействие компонентов Stateflow и RTW с системой «MATLAB+Simulink»

В этой главе рассматривается пакет расширения Stateflow. Предполагается, что при инсталляции системы MATLAB была произведена и установка пакета Stateflow. Этот пакет может использоваться совместно с любыми пакетами расширения Simulink. Коротко, в основном как средство создания исполняемых файлов для моделей систем, описан и пакет RTW.

Назначение пакета Stateflow

Пакет событийного моделирования Stateflow основан на теории конечных автоматов. Он позволяет представить функционирование системы на основе цепочки правил: «если произошло данное событие, то будет то-то и то-то». Пакет может использоваться как самостоятельно, так и в составе моделей, создаваемых в Simulink и в других пакетах расширения системы «MATLAB+Simulink». Это открывает обширные возможности в визуализации поведения сложных систем в ходе их моделирования. Демонстрационные примеры этого пакета, описанные в конце главы, дают наглядное тому подтверждение.

Пакет Stateflow предназначен, прежде всего, для анализа, моделирования и проектирования таких систем, как:

- детерминированные системы управления;
- диспетчерская служба различных транспортных средств (автомобильного, железнодорожного и воздушного движения);
- периферийные устройства и контроллеры для компьютеров;
- средства графического интерфейса пользователя (GUI);
- элементы человеко-машинного интерфейса (Men Machine Interface — MEI);
- гибридные системы на основе средств ряда пакетов расширения (Simulink, Control System, Digital Signal Processing и др.).

Также данный пакет может использоваться для разработки наглядных уроков по моделированию и проектированию систем.

Пакет имеет свой графический интерфейс пользователя, позволяющий создавать SF-модели в виде графических SF-диаграмм динамического типа. Это означает, что в ходе моделирования на диаграмме видны характерные изменения, например построение диаграмм переходов, изменение цвета блоков в зависимости от их активности и т. д. Это позволяет наглядно отслеживать поведение системы в ходе моделирования. К сожалению, в примерах книги трудно в полной мере отразить динамику изменения SF-диаграмм, поскольку цветовую раскраску передать нельзя, а для отображения динамики можно представлять лишь отдельные кадры.

Графический интерфейс пользователя пакета Stateflow имеет следующие компоненты:

- графический редактор SF-диаграмм;
- Stateflow-проводник для анализа SF-диаграмм (Stateflow Explorer);

- навигатор (Stateflow Finder) для поиска в SF-диаграммах нужных объектов;
- отладчик Stateflow-моделей;
- генератор кодов для работы совместно с расширением Real-Time Workshop.

Эти компоненты обеспечивают интуитивно понятные и простые приемы работы с пакетом Stateflow.

Документация и рекомендуемая литература

Документация по пакету расширения Stateflow достаточно обширна и представлена в HTML- и PDF-форматах. Объем документации (User Guide) в PDF-формате составляет 413 с. По вопросам, связанным с применением пакета расширения Stateflow, можно рекомендовать следующую литературу:

1. Гультяев А. Имитационное моделирование в среде Windows. Практическое пособие. — СПб.: КОРОНА-Принт, 1999.
2. Гультяев А. Визуальное моделирование в среде MATLAB. Учебный курс. — СПб.: Питер, 2000.
3. Harel D. Statecharts: A Visual Formalism for Complex Systems, Science of Computer Programming 8, 1987, pages 231–274.
4. Hatley D. J., Pirbhai I. A. Strategies for Real-Time System Specification, Dorset House Publishing Co., Inc., NY, 1988.

Доступ к средствам Stateflow

Для обеспечения доступа к средствам пакета расширения Stateflow пакет нужно инсталлировать. Это делается в ходе инсталляции системы MATLAB установкой флажка для приложения Stateflow Coder в списке компонентов системы MATLAB. После этого в окне броузера библиотек, вызываемого при запуске системы Simulink, появится раздел Stateflow.

На первый взгляд средства пакета Stateflow очень просты. Они используются совместно с составленной пользователем моделью системы или устройства путем вставки в них блока Stateflow с именем Chart (Диаграмма). Доступ к блоку Chart обеспечивается из броузера библиотек. Найдя библиотеку Stateflow, нужно открыть ее окно, используя меню правой кнопки мыши, содержащее единственную команду. Рисунок 7.2 показывает это окно в полностью раскрытом состоянии.

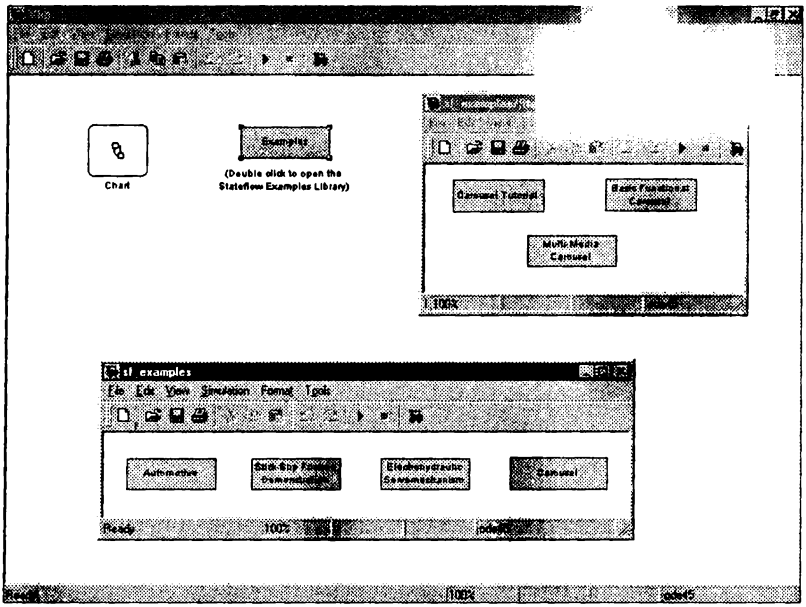


Рис. 7.2. Окна библиотеки и примеров применения системы Stateflow

В окне библиотеки (основном) присутствуют только два компонента — Chart (Диаграмма) и Examples (Примеры). Компонент Chart служит для вставки в Simulink-модели блоков вызова SF-диаграмм — изначально пустых.

Если активизировать значок Examples, то появится окно со значками примеров применения пакета Stateflow. Оно показано на рис. 7.2 снизу. В этом окне имеются четыре примера применения пакета Stateflow, причем последний пример, Carousel, разбит на три примера (окно с ними также показано на рис. 7.2).

Подключение блока Chart к Simulink-модели

Средства пакета Stateflow могут использоваться как самостоятельно (см. ниже), так и в составе моделей в пакете расширения Simulink. Все, что надо сделать для подключения блока Chart к Simulink-модели, — это перетащить его мышью в эту модель и подключить (опять же с помощью мыши) к нужному месту. Рисунок 7.3 показывает простую Simulink-модель, состоящую из источника сигнала, к которому подключен осциллограф. В нее внедрен блок Chart пакета расшире-

ния Stateflow. Он подключен к выходу источника сигнала и задает свойство модели как выключателя.

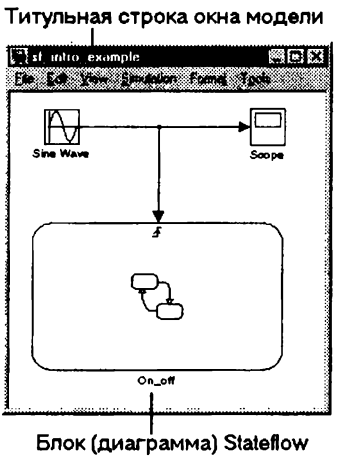


Рис. 7.3. Простая Simulink-модель с блоком вызова SF-диаграммы

Изначально активизация этого блока вызывает появление пустого окна редактора SF-диаграмм. Его можно использовать для создания SF-диаграмм и их отладки с целью получения от них нужных функций. Совокупность всех SF-блоков, имеющихся в составе Simulink-модели, образует *Stateflow-машину*. Как превратить эту модель в реально работающую модель выключателя, мы рассмотрим в дальнейшем. Пока же ограничимся констатацией того, что включение блока Chart позволяет превратить модель Simulink в комбинированную модель, позволяющую строить SF-диаграмму.

Что такое SF-диаграмма

SF-диаграмма — это диаграмма, создаваемая средствами графического интерфейса пакета расширения Stateflow. Она служит для визуального представления работы моделируемой системы. Это достигается анализом всех стадий ее работы с указанием активных и пассивных в данное время блоков, а также переходов между ними по результатам анализа тех или иных условий. При этом блоки различаются цветом и толщиной линий, которыми они представлены. Последнее, кстати, позволяет дать представление о динамике работы SF-диаграмм даже в книгах с черно-белыми иллюстрациями.

Прежде чем описывать средства создания SF-диаграмм, рассмотрим типичную SF-диаграмму некой абстрактной системы. Такая диаграмма с обозначенными на ней объектами представлена на рис. 7.4.

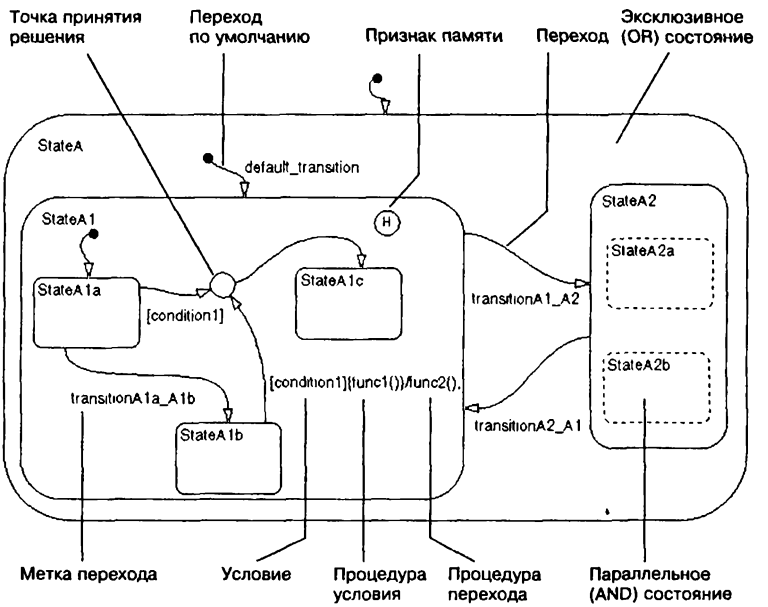


Рис. 7.4. Вид Stateflow-диаграммы

На рис. 7.4 показаны основные типы объектов SF-диаграмм. Позже мы рассмотрим их более подробно.

Работа с редактором SF-диаграмм

Первоначально блок Chart представляет пустую диаграмму. Естественно, что такая диаграмма никаких действий не иллюстрирует. Это просто заготовка для создания действующей диаграммы. Если активизировать мышью пустой блок Chart, то появится окно редактора SF-диаграмм, в котором можно начать подготовку диаграммы.

Рисунок 7.5 показывает этот процесс в самом начале. Здесь вы видите окно броузера библиотек Simulink и пустое окно модели Simulink, в которое перетащен блок Chart. Внизу показано окно редактора SF-диаграмм, которое открывается, если выполнить на блоке Chart и двойной щелчок.

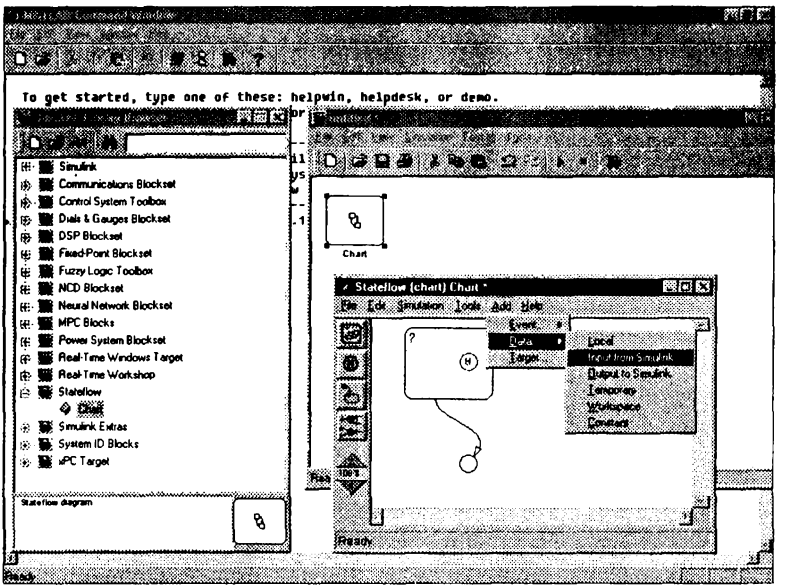


Рис. 7.5. Подготовка к созданию Stateflow-диаграммы

Как видно из рис. 7.5, редактор SF-диаграмм имеет обычный для окон системы MATLAB вид. Основные средства построения диаграмм сосредоточены в меню Add (Добавить), которое показано раскрытым. Эти средства дублированы в панели инструментов, которая располагается вдоль левой границы окна. Четыре ее кнопки имеют следующее назначение (сверху вниз):

- State — установка состояния (в том числе внутри другого состояния — см. рис. 7.5);
- History junction — установка признака состояния с памятью;
- Default transition — установка перехода по умолчанию;
- Connective junction — установка признака альтернативы.

Наименования кнопок (Name), их обозначения на диаграммах (Notation) и вид значков инструментальной панели (Toolbar Icon) представлены на рис. 7.6.

Каждая кнопка панели инструментов имеет изображение создаваемого ею блока. Эти блоки можно переносить мышью в окно SF-диаграммы, а затем соединять их друг с другом также с помощью мыши удерживая левую кнопку нажатой. При этом в общем случае строят

ся изгибающиеся стрелки, причем изгиб можно формировать также перемещением мыши. Ранее на рис. 7.5 в окне SF-диаграммы было показано построение четырех основных графических объектов. Перейдем к их более детальному описанию.

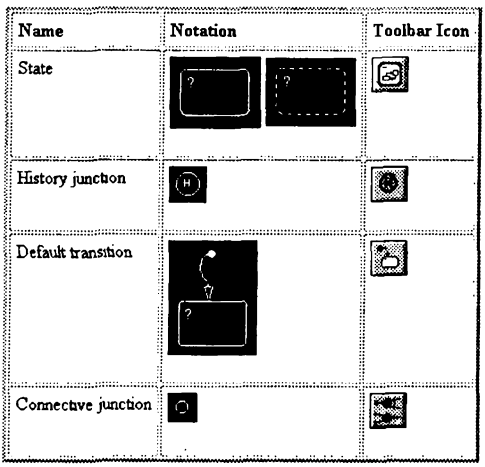


Рис. 7.6. Подготовка к созданию Stateflow-диаграммы

Основные объекты SF-диаграмм

Состояния

Важным объектом SF-диаграмм являются *состояния* (state). Это графический объект в виде прямоугольника с заметно скругленными углами, обычно построенный синими линиями. Активность или пассивность состояния динамически изменяется в зависимости от происходящих событий. Каждое состояние имеет состояние-родителя и может иметь дочерние состояния (состояния более низкого уровня). Дочернее состояние изображается вложенным в родительское. Если состояние является единственным, то его родителем является сама SF-диаграмма, называемая также *корневой диаграммой*.

Основное состояние, которое является внешним по отношению к встроенным в него состояниям, называется *эксклюзивным*. Если строится единственное состояние, то оно и будет эксклюзивным.

Состояния могут быть *нейтральными* (neutral) и *занятыми* (engaged). Имеется два типа состояний — параллельные, то есть существующие

одновременно (AND) и взаимно исключающие друг друга (OR). К примеру, взаимно исключающими друг друга являются состояния «Включено» и «Выключено» двустабильного выключателя.

Признак состояния с памятью

Состояние может иметь *признак памяти* или свою хронологию (history), которая обеспечивает определение будущего перехода в другое состояние на основе информации о прошлом системы. Признак памяти — это функция, имеющая приоритет в выполнении. Он выглядит как красная окружность с буквой «H» внутри.

Переходы

Переходы (transition) — еще один важный класс графических объектов SF-диаграмм. Они отражают переходы от одного объекта к другому и представляются обычно красными стрелками. Переходы не имеют своей кнопки в панели инструментов. Они создаются мышью, как только вы укажете (при нажатой левой кнопке) объект, от которого должна исходить стрелка перехода, и начнете двигать указатель к тому объекту, к которому направляется переход. Переходы имеют *метки*, которые описывают обстоятельства или условия, при которых происходит переход из одного места диаграммы в другое. Например, метка clutch_engaged сопровождает переход от нейтрального состояния к занятому.

Признаки альтернативы

Для указания альтернативных путей перехода систем из одного состояния в другое служат *признаки альтернативы* (connective junction). Это графические объекты в виде черных окружностей, закрашенных внутри красным цветом и имеющих стрелку перехода. Применение таких объектов упрощает построение SF-диаграмм, исключая необходимость пересечения переходов. Кроме того, альтернативные пути переходов способствуют повышению эффективности работы SF-диаграмм и облегчают генерацию программного кода.

События

Событие (event) — важнейшее понятие в пакете Stateflow. Возникновение события меняет статус связанных с ним состояний и может запустить заданное действие или переход. При этом события распространяются сверху вниз, от родительских состояний к дочерним.

Событие не является графическим объектом. Однако для визуализации события можно использовать метки переходов, связанных с теми или иными событиями. Таким образом, каждое событие должно быть определено с помощью того или иного логического *условия*.

Рисунок 7.7 иллюстрирует построение двух состояний — нейтрального neutral и занятого engaged. Затем между ними создается переход clutch_engaged. После этого внутри занятого состояния создаются два дочерних состояния: first и second. Между ними организован переход по условию (в данном случае, если скорость speed превышает заданную величину threshold). Обратите внимание на то, что условие записывается в квадратных скобках. Наконец, вводится признак памяти для занятого объекта (в его правом верхнем углу).

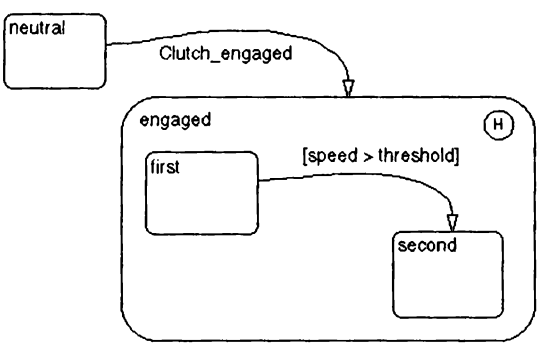


Рис. 7.7. Пример подготовки простой SF-диаграммы

События имеют *свойства*. Главным из них является область видимости события. События, классифицируемые по области видимости, могут быть следующих типов:

- *локальные* — видимые только в пределах данной SF-диаграммы;
- *входные* — передаваемые в SF-диаграмму из Simukink-модели;
- *выходные* — передаваемые из SF-диаграммы в Simulink-модель;
- *экспортируемые* — передаваемые из SF- или Simulink-модели во внешнюю программу;
- *импортируемые* — получаемые из внешних программ.

Редактор SF-программ имеет меню Add (см. рис. 7.5), с помощью которого можно задать тип события и указать его свойства.

Процедуры

Процедура (action) не является графическим объектом. Есть две модели конечных автоматов для процедур:

- *Модель Мура* (Moore), связывающая процедуры с состояниями;
- *Модель Мили* (Mealy), связывающая процедуры с переходами.

Для описания процедур служит специальный язык процедур — Action Language. При этом процедура может быть задана как вызов функции, задание определенного события или перехода и т. д. Пример записи процедур, относящихся к переходу, был дан на рис. 7.4. Здесь можно выделить процедуру условия и процедуру перехода.

Пример задания процедуры, связанной с состоянием объекта, приведен на рис. 7.8.

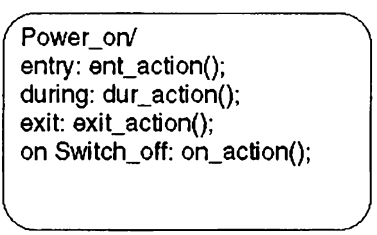


Рис. 7.8. Пример задания процедуры, связанной с состоянием объекта

При записи процедур используется семантика пакета Stateflow. К сожалению, из-за ограниченного объема книги подробно семантика записи действий не рассматривается. Такого рассмотрения, кстати, нет и в объемном фирменном описании. В нем семантика показана на множестве практических примеров. Демонстрационные примеры Stateflow неплохо иллюстрируют семантику записи процедур, и мы призываем читателя, заинтересованного в глубоком знакомстве с этим вопросом, внимательно проанализировать их. Часть из этих примеров описана ниже.

Данные

Данные (data) представляют в SF-модели числовые значения. Данные не являются графическими объектами и непосредственно на SF-диаграмме не указываются. Они могут создаваться на любом уровне иерархии модели и имеют свойства. Данные, классифицируемые по области видимости, могут быть следующих типов:

- *локальные* — видимые только в пределах данной SF-диаграммы;
- *входные* — передаваемые в SF-диаграмму из Simulink-модели;
- *выходные* — передаваемые из SF-диаграммы в Simulink-модель;
- *временные* (*промежуточные*);
- *сохраняемые* — хранящиеся в рабочей области MATLAB;
- *константы*;
- *экспортируемые* — передаваемые из SF- или Simulink-модели во внешнюю программу;
- *импортируемые* — получаемые из внешних программ.

Для создания и модификации данных служит команда Data меню Add редактора SF-диаграмм. Это же можно сделать с помощью рассмотренного чуть далее проводника Stateflow.

Описание объектов

При создании состояния в левом верхнем углу его графического образа появляется знак вопроса. На его место вводится описание состояния — в простейшем случае его имя. Общая структура состояния следующая:

```
name/
entry:
during:
exit:
on event_name:
```

Смысл этих определений дан ниже:

- during, или du — действия, которые выполняются как часть процедуры пребывания в некотором состоянии;
- entry, или en — действия, которые выполняются как часть процедуры входа в состояние;
- entry(state_name), или en(state_name) — генерация локального события при входе в состояние state_name;
- exit, или ex — действия, которые выполняются как часть процедуры выхода из состояния;
- exit(state_name), или ex(state_name) — генерация локального события при выходе из состояния state_name;
- on event_name — процедура, задающая выполнение каких-либо действий при возникновении события event_name.

Пример описания состояния дан на рис. 7.8. Описания других объектов также вводятся по запросу в виде вопросительного знака.

Процедуры в SF-диаграммах описываются на специальном языке Action Language. Он построен на основе синтаксиса языка C и содержит арифметические и логические операторы и функции, некоторые специальные функции и функции пользователя. Отметим следующие функции:

- `change(data_name)`, или `chg(data_name)` — генерация локального события в случае, когда меняется значение `data_name`;
- `in(state_name)` — функция условия, дающая значение `true` в случае, когда состояние `state_name` является активным;
- `send(event_name,state_name)` — посылает спецификацию события `event_name` состоянию `state_name` (прямая передача события);
- `matlab(evalString,arg1,arg2,...)`, или `ml(...)` — процедура, выполняющая вычисления, записанные в строке `evalString` в нотации системы MATLAB, с перечисленными вслед за ней аргументами;
- `matlab.MATLAB_workspace_data`, или `ml` — процедура, предоставляющая доступ к переменным рабочей области MATLAB.

Примеры описаний проще всего взять из демонстрационных примеров. Особо надо отметить применение символа `t` для обозначения времени, например, в выражениях вида `[t - On_time > Duration]`.

Подготовка SF-диаграмм

Подготовка модели Simulink с заготовкой SF-диаграммы

Попробуем составить и испытать в работе простую модель электрического выключателя, который в исходном состоянии выключен. Спустя некоторое время мы вручную включаем его, и напряжение поступает на нагрузку, подключенную через выключатель к источнику переменного напряжения. Для моделирования такого устройства пригодна модель, показанная ранее на рис. 7.3. Однако тут настало время показать, как же создается такая модель с самого начала.

Прежде всего, надо запустить расширение Simulink, например, активизируя соответствующую кнопку на панели инструментов MATLAB. Как обычно, появится окно броузера библиотек Simulink. Откроем новое окно Simulink-модели. В библиотеках Simulink найдем источник переменного напряжения и осциллограф (нагрузку). Перетащим их мышью в окно модели и соединим выход источника переменного

напряжения со входом осциллографа. Все это и показано на рис. 7.9 в левом верхнем углу окна Simulink-модели.

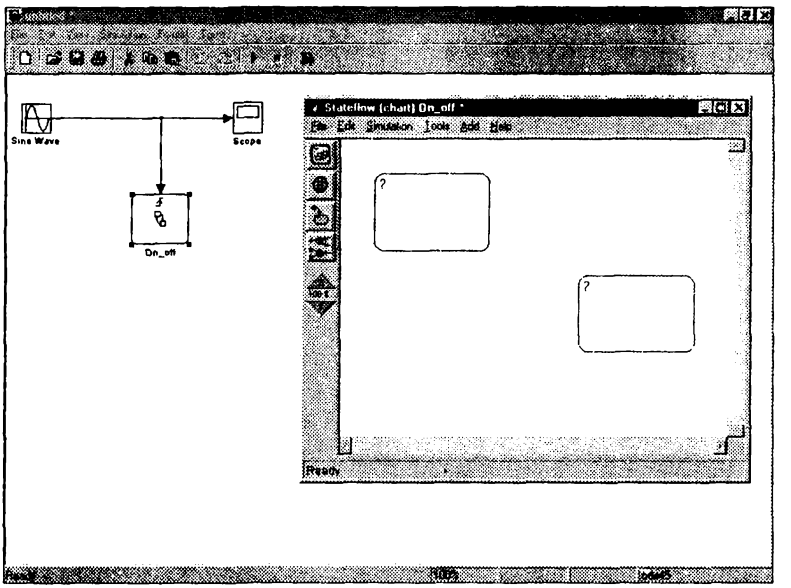


Рис. 7.9. Подготовка модели выключателя с диаграммой Stateflow

Теперь откроем раздел библиотек Stateflow и перенесем в окно Simulink-модели блок SF-диаграммы, представленный на рис. 7.9 под Simulink-моделью. Подключим этот блок к проводнику, соединяющему источник переменного напряжения с входом осциллографа. Установив указатель мыши в конец подписи блока Chart и щелкнув ее левой кнопкой, можно перейти в режим редактирования подписи. Заменим ее на более смысловую — On-off.

Начало подготовки SF-диаграммы в редакторе

Теперь двойным щелчком на SF-блоке On-Off можно открыть окно редактора SF-диаграмм и начать создавать в нем диаграмму. Начало этого процесса было продемонстрировано на рис. 7.9, где окно редактора SF-диаграмм показано в правой части окна редактора Simulink.

Используя первую сверху кнопку панели инструментов, перенесем в окно редактора два графических объекта-состояния. Их можно видеть в окне редактора SF-диаграмм, показанном на рис. 7.9 справа.

Пока оба состояния не имеют имен — вместо каждого имени стоит вопросительный знак. Поместив на него указатель и щелкнув левой кнопкой мыши, можно вместо вопросительного знака получить место ввода с текстовым курсором (в виде мигающей вертикальной черты) и ввести имя данного состояния. Введем имена Power_on и Power_off. При этом модель выключателя будет иметь вид (пока неполный), показанный на рис. 7.10.

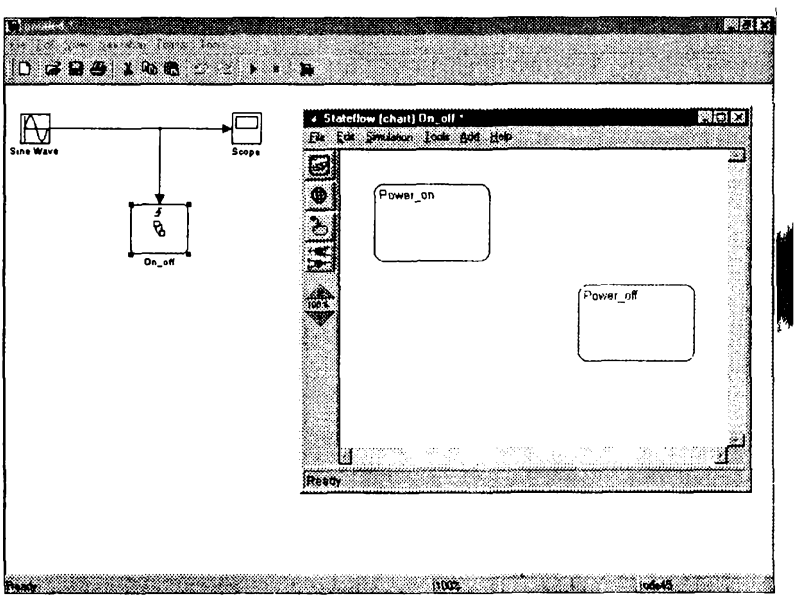


Рис. 7.10. Модель выключателя с SF-диаграммой, содержащей два состояния

Если вам не нравится местоположение созданных состояний или не устраивают их размеры, то вы можете их изменить. Поместив указатель мыши в середину скругленного прямоугольника, можно перетаскивать прямоугольник по окну диаграммы. Перетаскивая же округлые углы (указатель мыши при этом принимает вид двухсторонней стрелки), можно растягивать прямоугольник состояния, меняя его размеры.

Создание переходов между состояниями

Далее можно приступать к созданию переходов между состояниями. Для перехода от состояния **Power_on** к состоянию **Power_off** надо ус-

тановить указатель мыши на выход блока Power_on и, нажав левую кнопку мыши, начать строить стрелку перехода, перемещая ее ко входу блока Power_off. В отличие от моделей Simulink, графические объекты состояний не имеют четко обозначенных мест ввода и вывода, так что их можно выбирать произвольно. Например, можно построить переход от правой границы прямоугольника состояния Power_on к левой границе прямоугольника Power_off.

После этого надо построить стрелку обратного перехода от состояния Power_off к состоянию Power_On. К этому моменту создаваемая SF-диаграмма будет иметь вид, представленный на рис. 7.11 в правом окне.

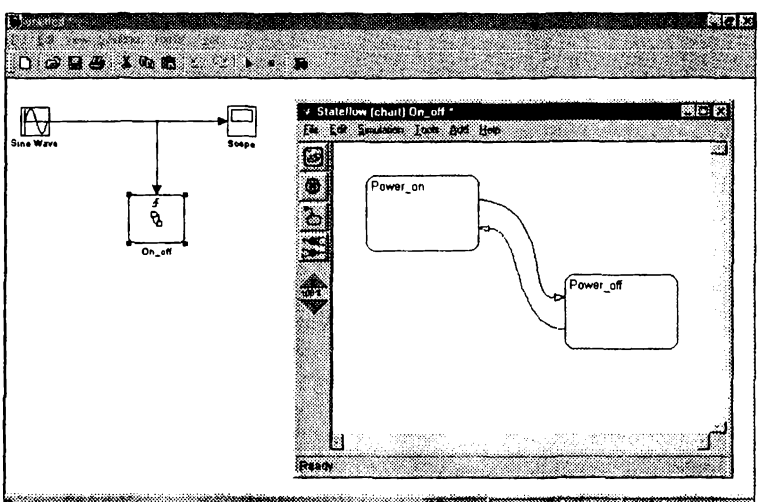


Рис. 7.11. Модель выключателя с SF-диаграммой, дополненной переходами

Обратите внимание, что строящаяся стрелка состояния изгибается при перемещении мыши и этот изгиб можно в дальнейшем корректировать, поместив на стрелку указатель мыши и перемещая мышь при нажатой левой кнопке.

Установка названий переходов

Если поместить на линию любого перехода указатель мыши и нажать левую ее кнопку, то около стрелки перехода появится вопросительный знак, на место которого можно ввести название перехода. Пусть это будет имя Switch.

Установка альтернативного перехода

Кроме того, нам надо указать переход к Simulink-модели у состояния Power_off. Зададим его как альтернативный переход с помощью соответствующей кнопки панели инструментов редактора SF-диаграмм. В итоге SF-диаграмма примет окончательный вид, приведенный на рис. 7.12.

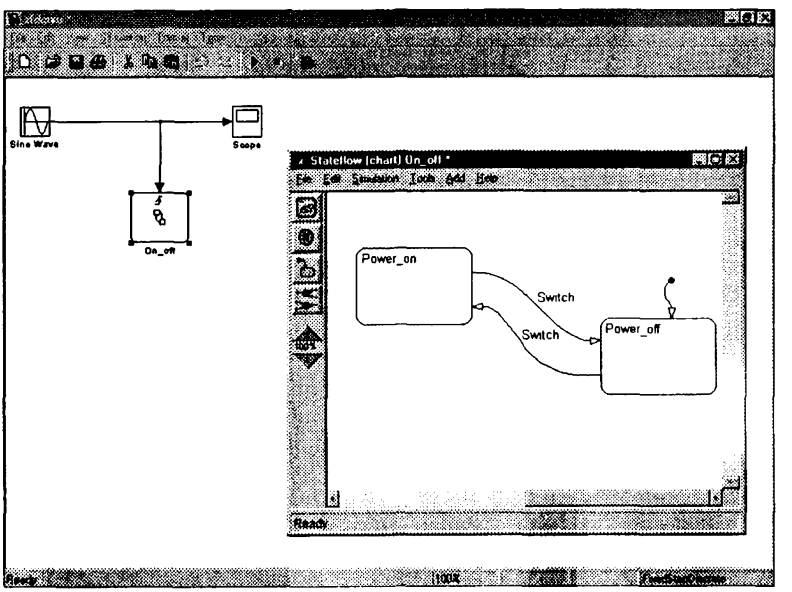


Рис. 7.12. Модель выключателя с законченной SF-диаграммой

На этом графическая часть подготовки модели выключателя заканчивается. Целесообразно сохранить Simulink-модель вместе с SF-диаграммой (как это делается, читателю, добравшемуся до этого места, пояснять, видимо, не стоит). Запишем модель под именем, например, sfdemo. Теперь можно приступить к установке ее параметров и пробному запуску.

Установка параметров SF-диаграммы с помощью проводника

Параметры состояний SF-диаграммы можно установить в окне графического редактора диаграмм с помощью команды Event меню Add. Эта команда открывает меню с необходимыми установками. Именно так было задано управление от блоков Simulink в примере, показан-

ном на рис. 7.3 (о типе управления говорит значок внутри блока диаграммы у острия входящей стрелки).

Однако здесь мы познакомимся с еще одним способом установки параметров — с помощью проводника Stateflow Explorer. Для его запуска используется команда меню Tools ▸ Explore. Она вызывает появление окна проводника, которое показано на рис. 7.13 в правом верхнем углу.

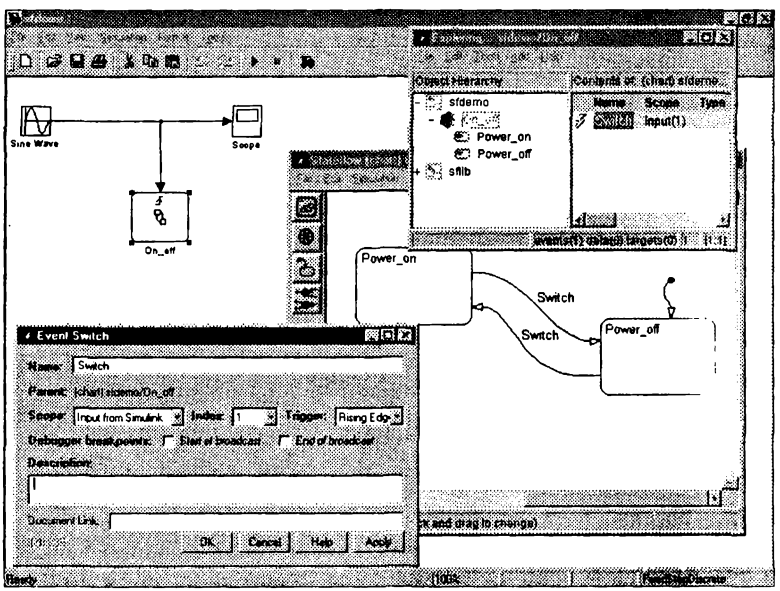


Рис. 7.13. Установка параметров SF-диаграммы

В левой половине окна проводника отражена иерархия объектов — Object Hierarchy. В нем можно найти записанную ранее модель sfdemo и входящую в нее SF-диаграмму On_off с объектами Power_on и Power_off. Выделив объект On_off, надо исполнить команду Add ▸ Event с помощью меню проводника. При этом появится окно Event, с помощью которого можно установить необходимые параметры. Прежде всего нужно ввести имя Switch в поле Name. В переключателе областей видимости Scope надо установить Input from Simulink, поскольку SF-диаграмма будет работать от Simulink-модели. Наконец, в списке Trigger надо выбрать вариант Rising Edge. После этих установок окна события и проводника можно закрыть.

Запуск и отладка SF-диаграмм

Установка параметров запуска

Перед запуском Simulink-модели следует установить параметры моделирования. Для этого в окне модели Simulink надо исполнить команду меню Simulation ▸ Parameters. Необходимые установки представлены на рис. 7.14.

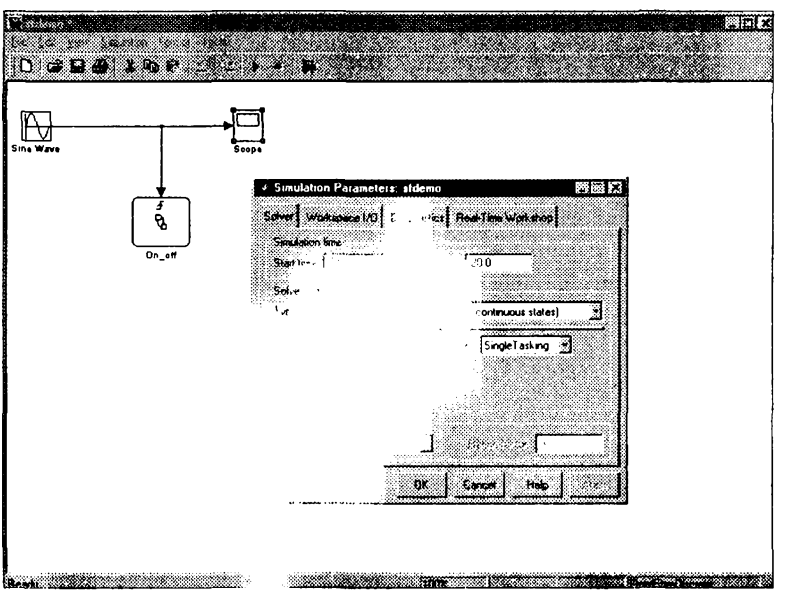


Рис. 7.14. Установка параметров запуска

На вкладке Solver надо установить начальное время Start time и конечное время Stop time, затем тип работы Fixed-step (с фиксированным шагом) и discrete (поскольку работа выключателя носит дискретный характер).

Запуск модели

Запуск производится, как обычно, командой Start из меню Simulation или активизацией кнопки Start в панели инструментов. При этом можно наблюдать работу модели и SF-диаграммы. Для этого, как правило, надо щелкнуть на блоке SF-диаграммы, чтобы она появилась на

переднем плане окна моделей Simulink. Можно запускать диаграмму и отдельно, как описано ниже.

Работа с отладчиком SF-диаграмм

Поскольку данная модель проста, то моделирование происходит быстро, так что оказывается трудно уследить за его результатом. Поэтому целесообразно воспользоваться специальным отладчиком SF-диаграмм. Его окно появляется при исполнении в окне SF-диаграммы команды меню Tools ▸ Debug. Вид окна показан на рис. 7.15.

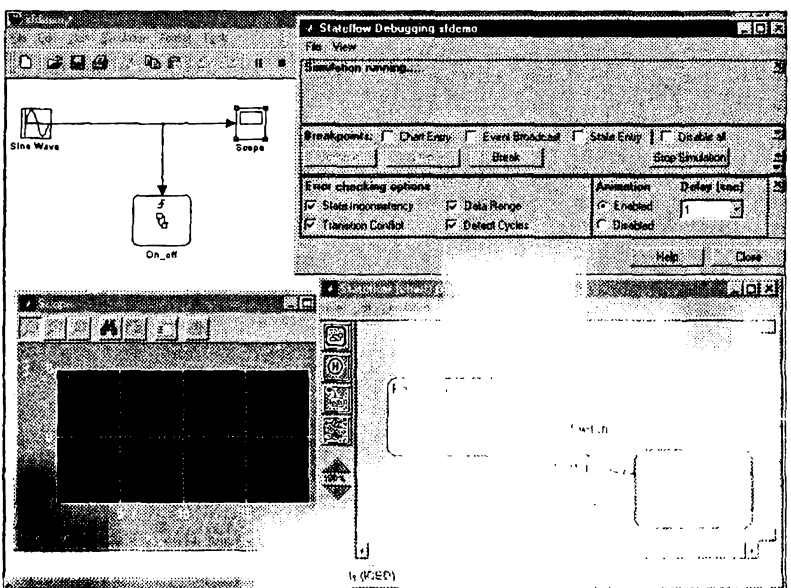


Рис. 7.15. Начало моделирования

Рисунок 7.15 иллюстрирует начало моделирования после нажатия кнопки Start отладчика (она после этого меняет название на Continue). Нетрудно заметить, что вначале активизируются альтернативный переход и состояние Power_off. Они отображаются жирными красными линиями (цвет на черно-белом рис. 7.15, естественно, не виден, но жирность линий просматривается отчетливо). Активность состояния Power_off означает, что выключатель отключает нагрузку (в нашем случае осциллограф) от источника переменного тока. Поэтому осциллограммы на экране осциллографа нет.

Рисунок 7.16 показывает некоторый промежуточный кадр динамики SF-диаграммы. Видно, что теперь активным становится переход от состояния Power_off в состояние Power_on, что означает активизацию процесса включения нагрузки. На экране осциллографа можно увидеть появление осциллограммы.

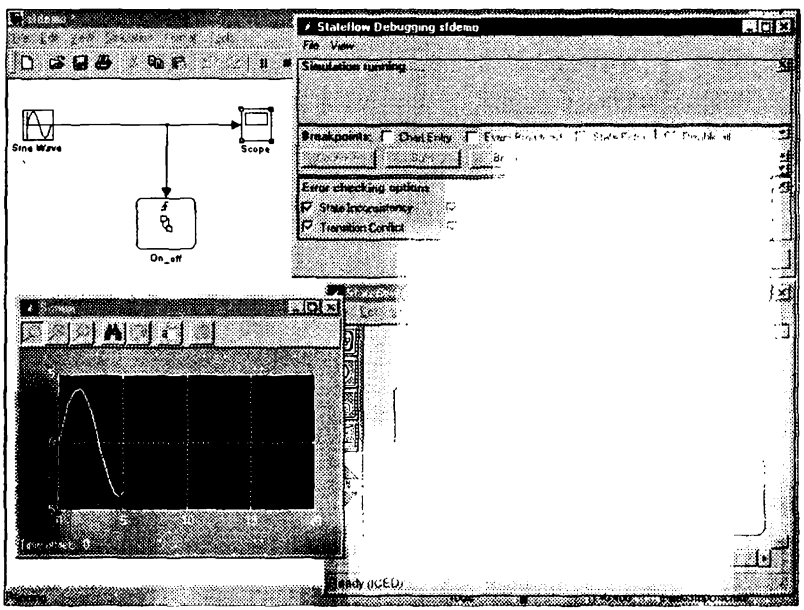


Рис. 7.16. Процесс включения выключателя

Наконец, рис. 7.17 показывает процесс по завершении всех стадий изменения SF-диаграммы. Этот процесс означает устойчивое включенное состояние, так что теперь переменное напряжение все время подается на нагрузку — осциллограф. Выделение объектов SF-диаграммы прекращается.

Итак, нами выполнен весь цикл операций по составлению и запуску достаточно простой Simulink-модели выключателя со встроенными в модель SF-машиной и SF-диаграммой. Мы настойчиво советуем читателям провести этот опыт самостоятельно, прежде чем пытаться начинать составлять свои собственные SF-модели и диаграммы. Это позволит почувствовать последовательность операций по подготовке и запуску моделей. На практике любая «мелочь» может помешать успешному запуску и привести к появлению окон с сообщениями

о тех или иных ошибках. Разбираться в них, как правило, намного сложнее, чем организовать правильное составление и запуск моделей.

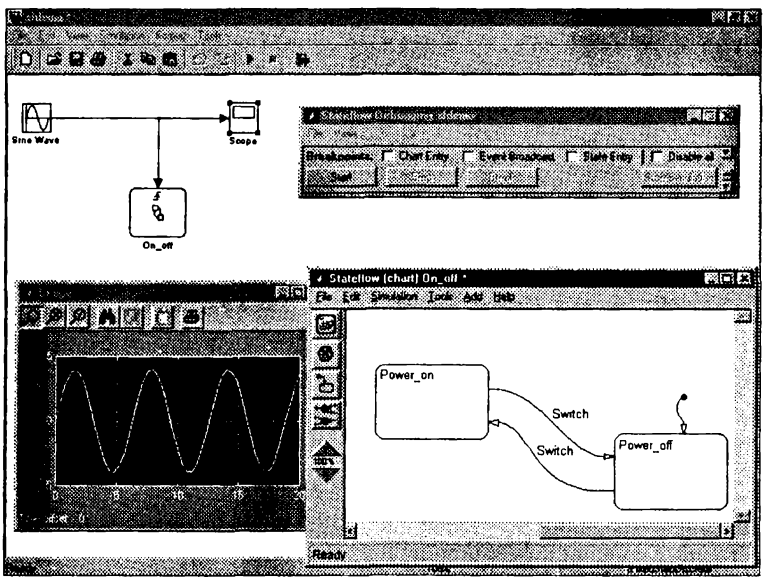


Рис. 7.17. Модель с SF-диаграммой в конце моделирования

Работа с отладчиком SF-диаграмм достаточно очевидна. Отметим несколько ее особенностей, которые полезно знать. Прежде всего, это возможность выполнения моделирования по шагам с помощью кнопки Step. Анимацию SF-диаграмм можно отключить, установив переключатель Animation в положение Disabled (анимация отключена) вместо Enabled (анимация включена). Наконец, можно замедлить темп моделирования, установив нужное время задержки Delay (по умолчанию оно равно 1 с). Поле Error checking options позволяет отключить различные режимы проверки ошибок. По умолчанию они все включены. Отключение проверки ошибок целесообразно производить только после окончательной отладки SF-диаграммы.

Средства отладки SF-диаграмм

SF-модель с точки зрения системы «MATLAB+Simulink» является типичной S-функцией. По записи S-функций имеется специальное фирменное руководство. Однако визуально-ориентированное програм-

мирование, используемое в пакетах Simulink и Stateflow, позволяет и без этого создавать достаточно надежные программы и сводит к минимуму (хотя и не исключает вообще) возможности возникновения синтаксических ошибок. Тем не менее, Stateflow имеет развитые средства отладки программ. Главное из них — синтаксический анализатор, который запускается командой Tools ▸ Parse из меню редактора SF-диаграмм. Открывающееся при этом окно анализатора показано на рис. 7.18 в правом верхнем углу.

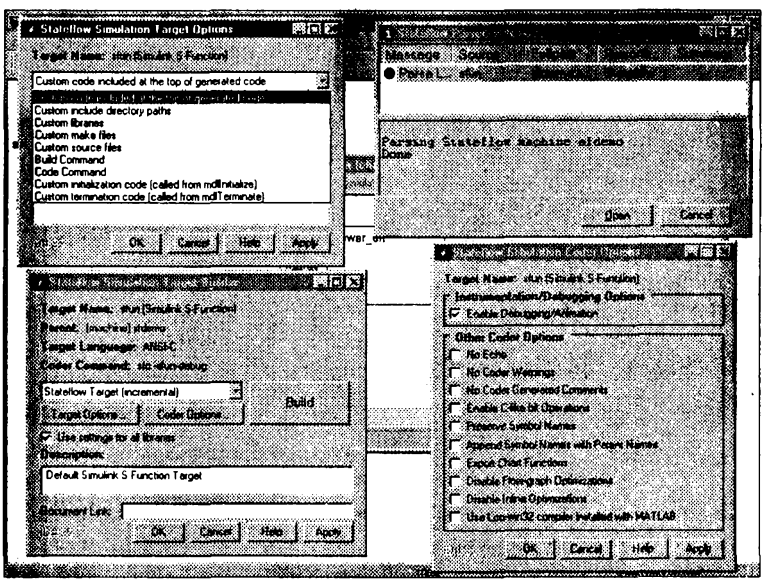


Рис. 7.18. Средства отладки SF-диаграмм

В этом окне имеется информация о текущей SF-диаграмме и наличии ошибок, если они есть. В нашем примере ошибок нет, поскольку он был корректен. Однако надо иметь в виду, что анализатор анализирует только *синтаксические* ошибки. Более каверзные *семантические* ошибки, например, в выборе алгоритмов построения SF-диаграмм, не выявляются. Устранение таких ошибок — дело пользователя.

Для выделенного в окне анализатора сообщения можно открыть окно построителя целевого кода — Simulation Target Builder. Это окно показано на рис. 7.18 в нижнем левом углу. Краткое описание вопросов генерации кодов будет дано в конце этой главы, поэтому пока от описания этого окна мы воздержимся.

Кнопка Target Options открывает окно параметров построителя целевого кода. На рис. 7.18 оно расположено в верхнем левом углу с открытым списком возможных параметров. Другая кнопка — Coder Options — открывает окно параметров специального компонента Stateflow — генератора программного кода Stateflow Coder. Это окно показано на рис. 7.18 в правом нижнем углу. Здесь прежде всего надо отметить флажок Enable Debugging/Animation. Этот режим разрешает анимацию SF-диаграммы и по умолчанию включен. Ряд других флажков по умолчанию снят; они служат для управления процессом генерации кода.

Кнопка Build в окне построителя открывает окно, подобное окну Parse, но с именем Build в строке заголовка.

Таким образом, рис. 7.18 показывает все основные средства отладки SF-диаграмм. Естественно, что эти средства нужны только в том случае, когда при работе с SF-диаграммами возникают проблемы. Управление режимами создания кода полезно и для обеспечения должного уровня надежности программ.

Поиск объектов SF-диаграмм

SF-диаграммы многих моделируемых систем, в том числе приведенных в демонстрационных примерах, очень сложны, и разобраться в них не так-то просто. Помощь в этом оказывает средство поиска объектов. Оно вызывается командой Tools ▸ Find из меню окна редактора SF-диаграмм. Эта команда открывает окно поиска объектов, показанное на рис. 7.19.

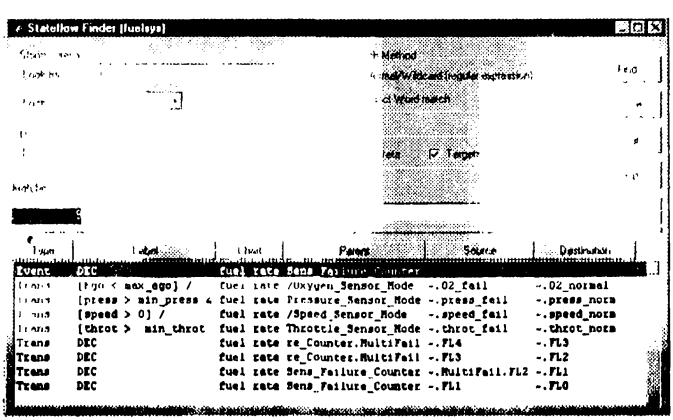


Рис. 7.19. Окно поиска объектов в SF-диаграммах

В окне поиска можно задать имя объекта, который должен быть найден. Внизу окна появляется список всех мест, в которых обнаружено заданное имя, и приводятся данные о каждом таком месте. С другими возможностями этого окна можно ознакомиться самостоятельно.

Оформление SF-диаграмм

Выбор стиля

Масштаб SF-диаграмм можно менять с помощью переключателя, расположенного в панели инструментов под кнопками вывода графических объектов. Возможно также задание разных стилей SF-диаграмм, различающихся цветовым оформлением. Для этого в меню Edit есть команда Style. Она выводит окно выбора стиля диаграммы, показанное на рис. 7.20.

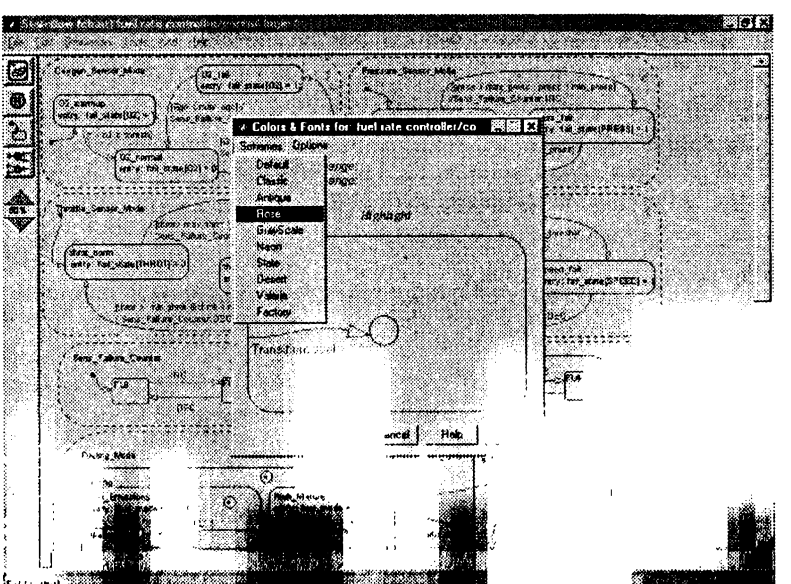


Рис. 7.20. Выбор стиля SF-диаграммы

Выбор стилевой схемы производится из меню, которое на рис. 7.20 представлено в открытом состоянии.

Установка размера символов

Имеется также возможность установить размер символов для надписей, которые имеются на SF-диаграмме. Для этого достаточно выделить нужный объект и воспользоваться командой меню Edit ▸ Set Font Size. В раскрывшемся подменю остается выбрать нужный размер шрифта. Не следует злоупотреблять выбором больших размеров у символов, поскольку при этом надпись может вылезти за пределы объекта, например за пределы прямоугольника состояния.

Демонстрационные примеры пакета Stateflow

Рекомендуемые правила работы

При ознакомлении с демонстрационными примерами пакета Stateflow целесообразно придерживаться следующих рекомендаций:

- загрузите пример в окно пакета Simulink;
- ознакомьтесь с моделью примера и выясните, есть ли в ней подсистемы;
- просмотрите блок-схемы подсистем и разберитесь с их работой;
- произведите пробный запуск примера;
- если моделирование идет слишком долго, остановите его и проверьте установку параметров моделирования (в частности, убедитесь, что время моделирования не задано бесконечным). При необходимости измените параметры;
- снова запустите пример на моделирование;
- по завершении моделирования выведите осциллограммы и иные предусмотренные в модели иллюстрации ее работы;
- постарайтесь осмыслить полученные результаты;
- выведите SF-диаграмму и запустите ее на выполнение (лучше это делать с помощью отладчика SF-диаграмм);
- проанализируйте SF-диаграмму, отметьте те ее детали, которые являются новыми для вас и разберитесь с ними;
- запишите результаты своего исследования.

Только очень наивный читатель может подумать, что стоит ему подготовить модель с SF-диаграммой и запустить сеанс моделирования, как можно будет сразу «снимать с нее пенки» в виде практически полезных результатов. На самом деле придется затратить немало часов для разбора как демонстрационных, так и своих собственных

примеров, прежде чем даже самые простые модели начнут нормально работать. При серьезном занятии событийным моделированием недостаточно ни материалов данной книги, ни фирменных руководств и справочной системы — лишь большой практический опыт моделирования позволяет получать результаты быстро и надежно.

Ниже представлено несколько весьма поучительных (для знакомства как с техникой моделирования в пакете Simulink, так и с применением пакета Stateflow) примеров. Мы рассмотрим их в порядке повышения сложности моделей.

Брусок, скользящий с трением

В качестве довольно простого примера применения пакета Stateflow рассмотрим моделирование скольжения бруска по поверхности при наличии трения. Это типичная физическая задача, демонстрирующая полезность моделирования физических явлений. Итак, пусть имеется брусок, лежащий на поверхности. Пусть через пружину на него действует линейно нарастающая сила. Из-за трения брусок при малых усилиях будет оставаться неподвижным, затем сорвется с места и начнет двигаться. Если вектор силы периодически меняет направление, то трение порождает гистерезис зависимости положения бруска от действующей на него силы.

Simulink-модель этой простой физической системы показана на рис. 7.21. Там же представлены результаты моделирования — осциллограммы для временных зависимостей силы и координаты бруска, а также фазовый портрет его движения, явно показывающий упомянутый выше гистерезис.

SF-диаграмма этой модели также предельна проста, что позволяет легко разобраться с ее особенностями. Она показана на рис. 7.22.

Данная диаграмма является прекрасным примером для разбора особенностей построения SF-диаграмм.

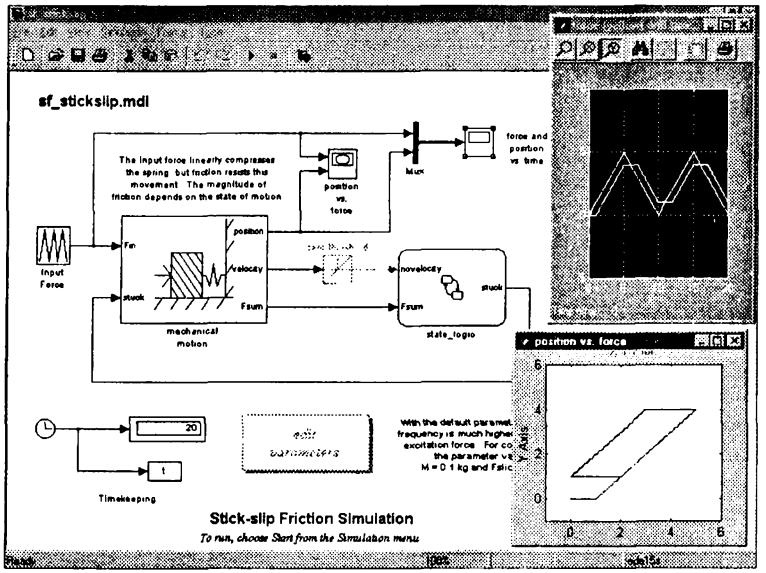


Рис. 7.21. Модель бруска, скользящего с трением

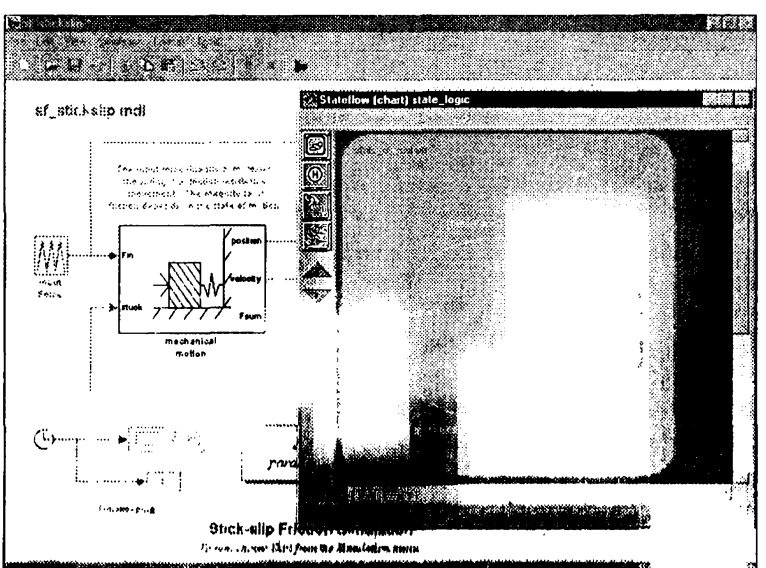


Рис. 7.22. SF-диаграмма модели бруска, скользящего с трением

Поведение автомобиля

Рассмотрим еще один пример — пример моделирования поведения автомобиля (файл sf_car). Соответствующая модель показана на рис. 7.23. Там же даны осциллограммы, показывающие поведение системы при возникновении на дороге чрезвычайного обстоятельства. Какого именно обстоятельства, читателю полезно разобраться самому. Мы же ограничимся техникой представления этого примера.

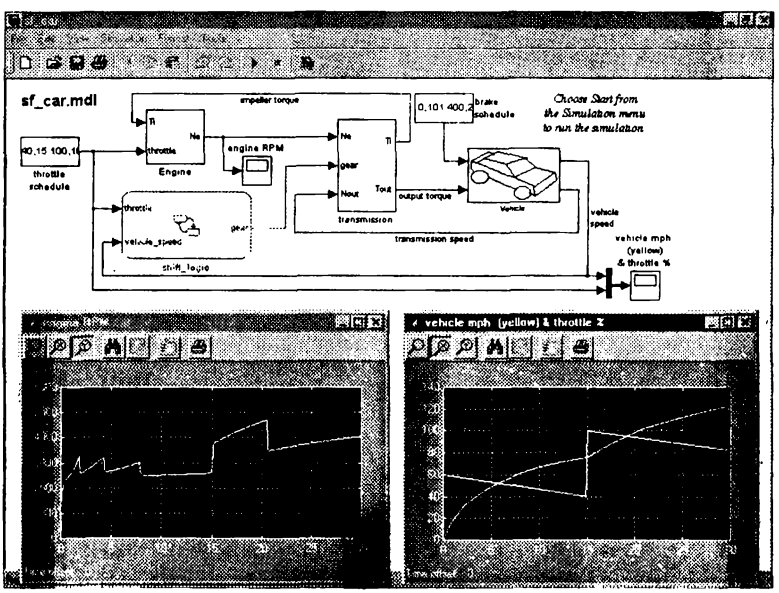


Рис. 7.23. Моделирование системы автомобиля

В отличие от рассмотренных ранее примеров, эта модель содержит ряд подсистем (рис. 7.24). Последние, в свою очередь, используют много интересных компонентов, демонстрирующих возможности системы Simulink. Однако, несмотря на наличие подсистем, этот пример все еще довольно прост и удобен для разбора.

Наконец, на рис. 7.25 представлена SF-диаграмма этой модели. Здесь прежде всего надо отметить создание особого объекта - *очереди*, которая видна в верхней части SF-диаграммы. В очереди имеет место последовательная активизация участков под действием управляющих событий.

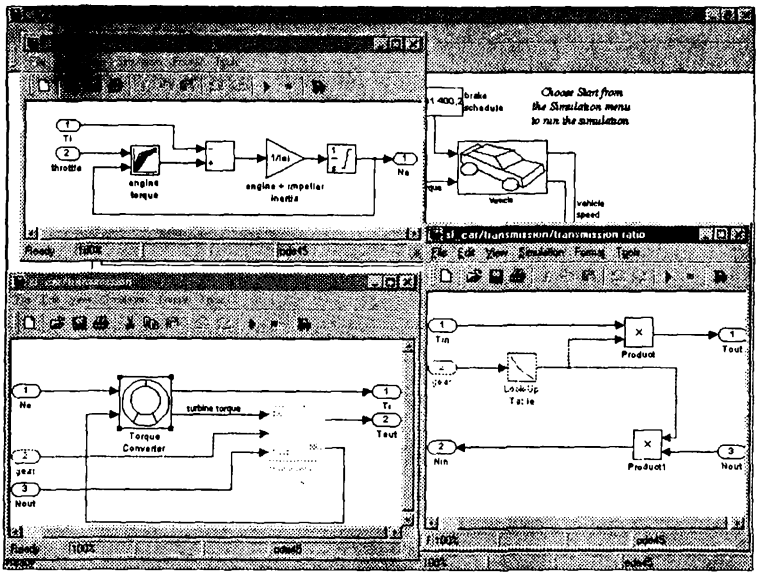


Рис. 7.24. Подсистемы модели поведения автомобиля

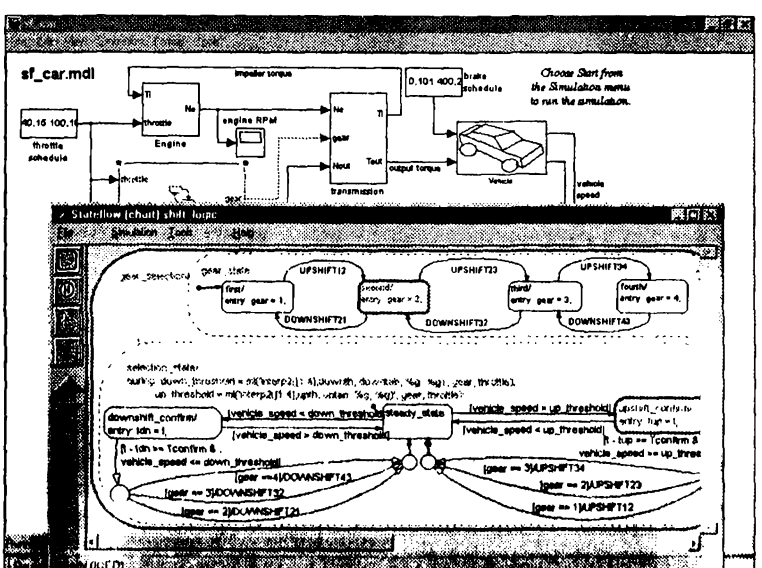


Рис. 7.25. SF-диаграмма моделирования поведения автомобиля

Наверное, трудно себе представить более важную задачу, чем сохранение жизни водителя и пассажиров автомобилей при возникновении аварийных ситуаций на дорогах. Этот пример показывает ценность моделирования подобных ситуаций. Мы намеренно оставляем за читателем более подробный разбор этого примера, да и других примеров. Такой анализ позволит разобраться в деталях моделирования задач, представленных демонстрационными примерами.

Электрогидравлический механизм

Еще один пример моделирования технического устройства представлен на рис. 7.26. На этот раз моделируется электрогидравлический сервомеханизм. Такие механизмы широко применяются в промышленности и на транспортных средствах.

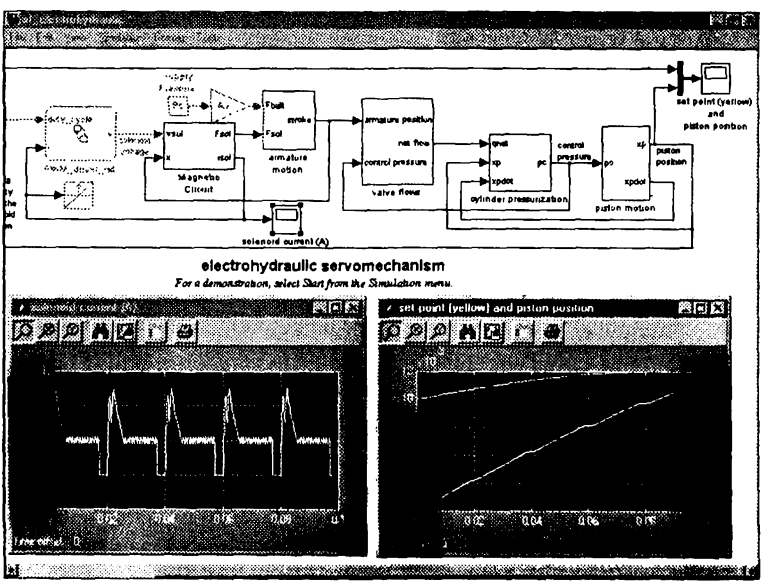


Рис. 7.26. Модель электрогидравлического сервопривода

Вы можете самостоятельно ознакомиться с подсистемами этой модели. На рис. 7.27 показан пример применения проводника SF-диаграмм для данной модели. Чем сложнее модель, тем больше данных о ней можно получить с помощью проводника. В этом нетрудно убедиться, сравнив рис. 7.27 с рис. 7.13, который иллюстрирует применение проводника при работе с более простой моделью.

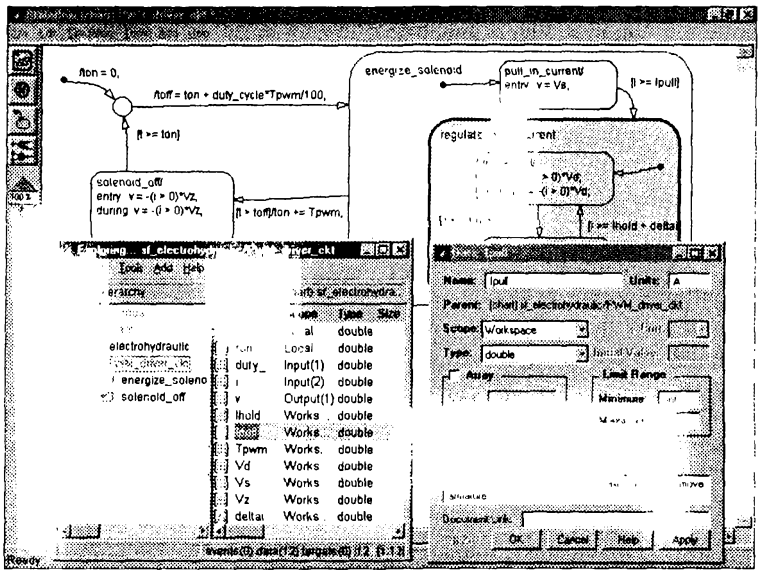


Рис. 7.27. Работа обозревателя SF-диаграмм с моделью электрогидравлического сервопривода

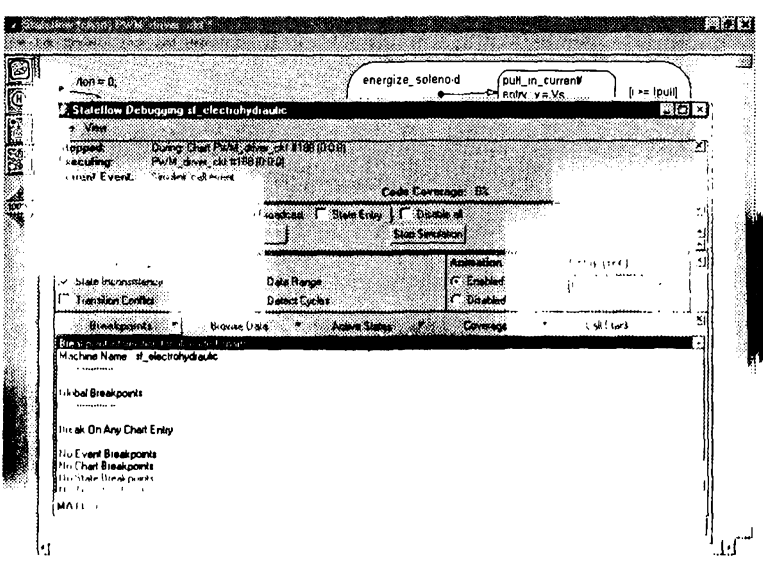


Рис. 7.28. Пошаговое выполнение SF-диаграммы электрогидравлического сервопривода

Следующий рисунок (рис. 7.28) показывает применение отладчика SF-диаграмм для пошагового анализа SF-диаграммы. Это позволяет спокойно изучить каждый шаг диаграммы.

Следует отметить, что с помощью соответствующих команд из меню View окна отладчика можно задать различные размеры окна — на рис. 7.28 его размер максимален. Минимальные размеры стоит устанавливать, если отладчик используется только для управления выполнением SF-диаграммы, например в пошаговом режиме. Не забывайте и о возможности оперативно менять масштаб представления SF-диаграммы с помощью переключателя в левой части ее окна.

Карусель

Для обучения работе с SF-диаграммами среди демонстрационных примеров приведен один особенно оригинальный и поучительный — моделирование карусели (Carousel, см. рис. 7.2). Этот пример дан в трех вариантах — учебном и двух демонстрационных. Оставив учебный пример читателю для самостоятельного разбора (он для этого и создан), возьмем вариант, представленный файлом sf_carousel, модель которого показана на рис. 7.29.

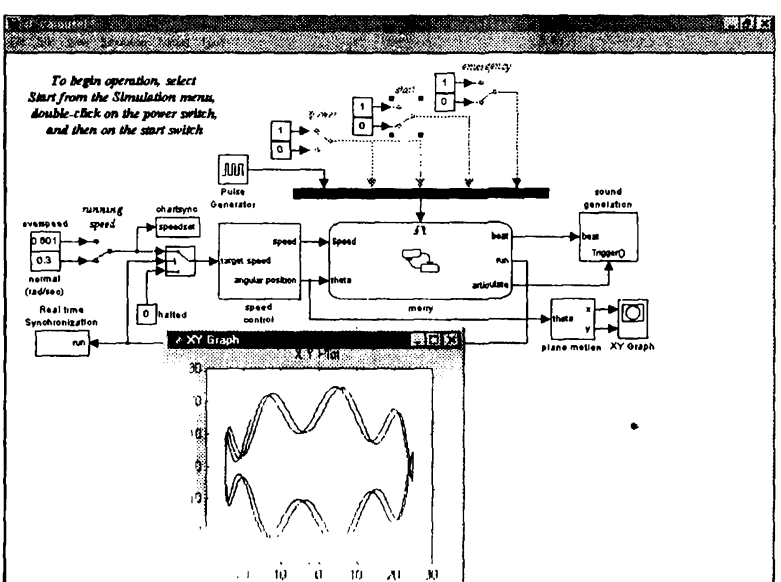


Рис. 7.29. Модель карусели

Этот пример демонстрирует работу довольно сложной модели, причем не только в виде графика, построенного графопостроителем, но и в виде звуков музыкального автомата, по заданной программе исполняющего популярную мелодию.

Обратите внимание на применение в этом примере ряда ручных переключателей из библиотеки компонентов системы Simulink. Чтобы запустить пример, мало дать общую команду Start для запуска моделирования. После этого нужно вручную (мышью) переключить переключатель power в состояние 1, а затем переключатель start также в положение 1 (верхнее). После всего этого можно наслаждаться видом строящейся траектории движения карусельных лошадок, звучанием музыки и работой SF-диаграммы. Разумеется, музыку вы услышите, только если ваш компьютер оснащен звуковой картой и колонками (или наушниками).

В том, что этот пример далеко не так прост, как кажется на первый взгляд, убеждает набор его подсистем, показанный на рис. 7.30. Эти подсистемы содержат довольно много занимательных и поучительных конструкций, изучение которых очень полезно для серьезного знакомства с пакетами Simulink и Stateflow.

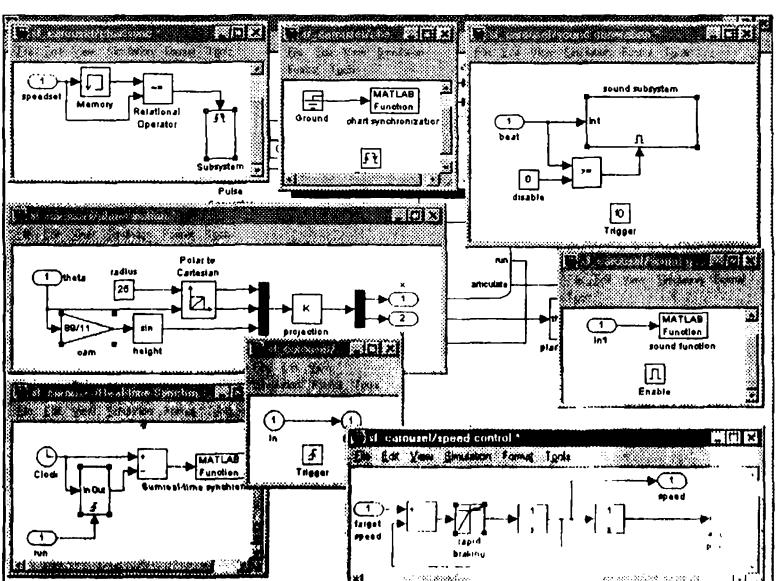


Рис. 7.30. Набор подсистем примера sl_ ...usel

Кадр работающей SF-диаграммы показан на рис. 7.31. Хотя сам по себе этот пример достаточно сложный, его SF-диаграмма довольно проста для изучения. Разумеется, это надо делать на «живом» компьютере, а не по приведенному на рис. 7.31 фрагменту.

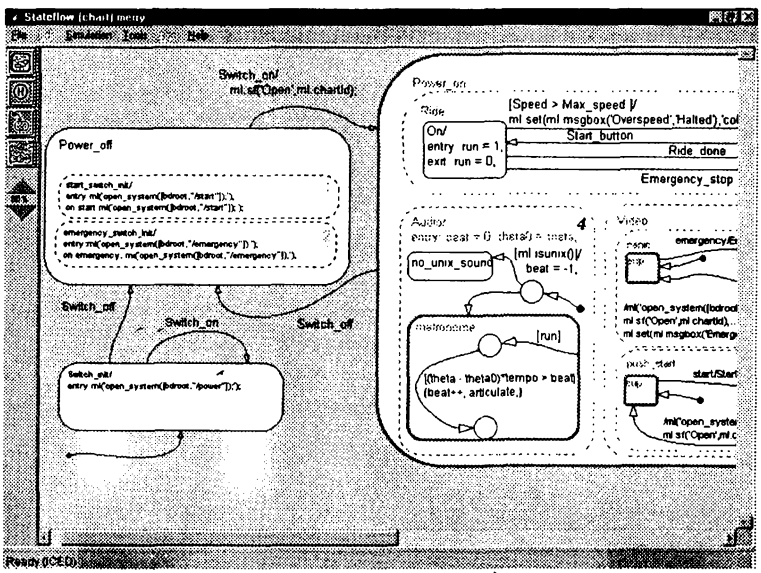


Рис. 7.31. Кадр SF-диаграммы примера sf_carousel

С другим демонстрационным примером группы Carousel вы можете ознакомиться самостоятельно.

Отказоустойчивая система управления подачей топлива

В современных скоростных транспортных средствах, например в самолетах и вертолетах (не говоря уже о космических кораблях и ракетах), важное значение имеет проектирование отказоустойчивых систем подачи топлива. В пакете Stateflow есть пример такого рода — fuelsys. Загрузив его в окно Simulink, можно увидеть модель, довольно простую на первый взгляд. Рисунок 7.32 показывает эту модель после ее исполнения и вывода контрольных осциллограмм.

Эта модель также содержит множество подсистем, причем куда более сложных, чем в модели карусели. Рисунок 7.33 дает лишь общее представление о них. К тому же, стоит отметить, что в этом примере подсистемы уже не помещаются в поле рисунка, да и показаны они не все.

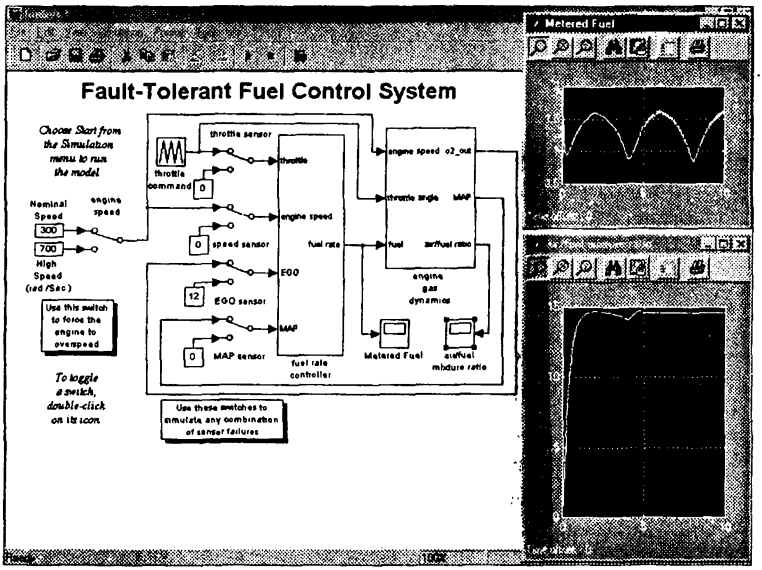


Рис. 7.32. Модель системы подачи топлива

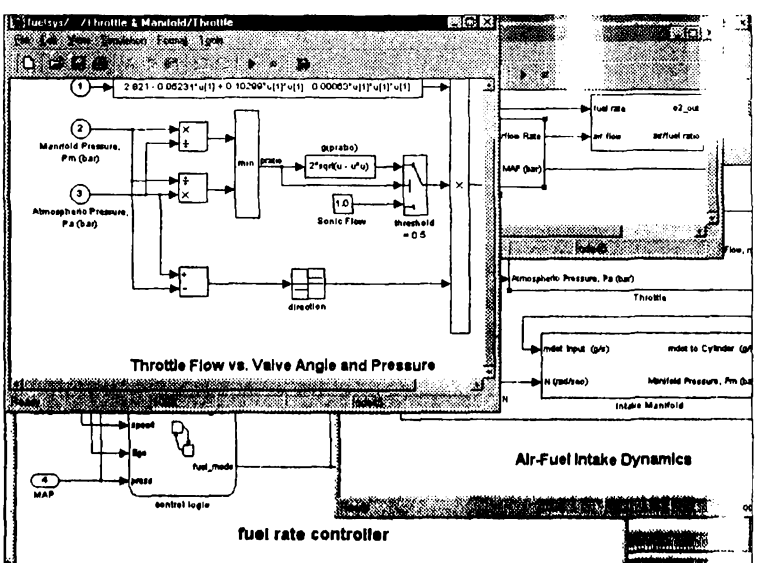


Рис. 7.33. Некоторые подсистемы модели, показанной на рис. 7.32

Кадр работающей SF-диаграммы представлен на рис. 7.34. Здесь также можно найти множество интересных примеров задания описаний различных объектов диаграммы, в частности ее состояний и управляемых событиями переходов.

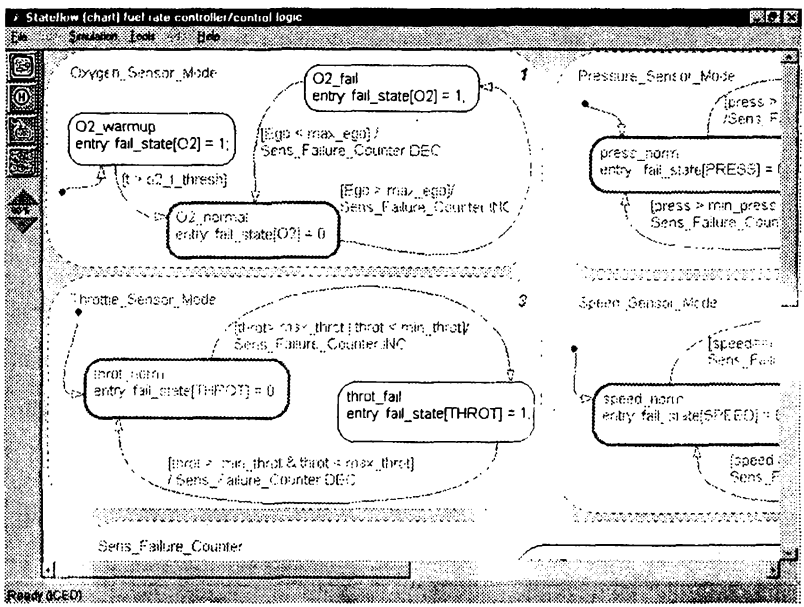


Рис. 7.34. SF-диаграмма для системы подачи топлива

На этом мы заканчиваем рассмотрение очень полезной и интересной системы событийного моделирования Stateflow. Надо полагать, что читатель отдаст должное уникальным возможностям этого пакета.

Real-Time Workshop

Назначение пакета RTW

Пакет расширения Real-Time Workshop (RTW) – это лаборатория реального времени. Этот пакет позволяет создавать файлы на языке C/C++ и даже исполняемые (с расширением .exe) файлы для программ системы «MATLAB+Simulink». При этом обеспечивается автоматический подбор необходимых библиотек и программных модулей

с последующей компиляцией и сборкой. Эти файлы обычно используются для управления внешними устройствами, работающими в реальном масштабе времени, например роботами или компьютеризированными измерительными комплексами.

Процесс создания исполняемых кодов моделей имеет некоторые ограничения. Так, в состав компилируемых нельзя включать блоки Fcn (функции MATLAB) и S-функции (программные модули на языке программирования C или MATLAB). Эти модули должны быть предварительно преобразованы в MEX-файлы. В моделях для компиляции нельзя задавать переменный шаг интегрирования дифференциальных уравнений, что заметно снижает возможности моделирования существенно нелинейных систем. Не стоит забывать также о необходимости установки самого пакета и обеспечения его бесконфликтной работы с компилятором.

Кроме того, на компьютере должен быть установлен один из следующих компиляторов:

Платформа	Компилятор	Файл шаблона
Windows	MS Visual C/C++	grt_vc.tmf
Windows	Borland C/C++	grt_bcx.tmf
Windows	Walcon C	grt_watc.tmf
UNIX	Любой компилятор ANSI C	grt_unix.tmf

Все это указывает на то, что задача получения исполняемых кодов не так проста, как кажется с первого взгляда.

Установка пакета

Пакет RTW должен быть установлен вместе с системой MATLAB. Признаком того, что это было сделано, является появление меню Tools в окне Simulink и увеличение числа команд в меню Simulation. К вновь добавленным относятся две команды — Normal и External (рис. 7.35).

Меню Tools, показанное в открытом виде на рис. 7.36, также имеет три команды, относящиеся к пакету RTW. Это команды RTW Build, RTW Options и External Mode Control Panel. Их назначение мы рассмотрим ниже.

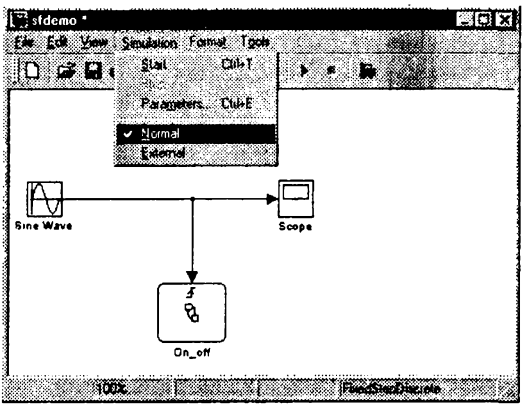


Рис. 7.35. Окно Simulink с открытым меню Simulation

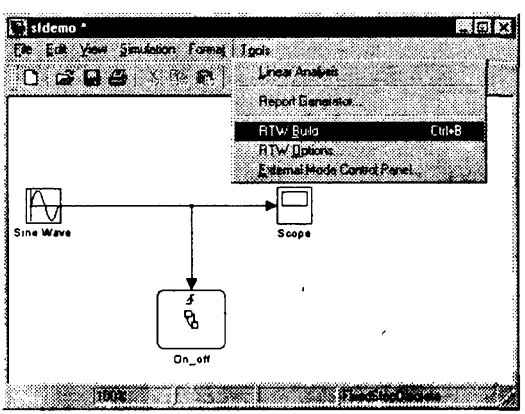


Рис. 7.36. Окно Simulink с открытым меню Tools

Принцип работы

Работа пакета RTW поясняется диаграммой, представленной на рис. 7.37. Коротко опишем ее. Вначале в системе «MATLAB+Simulink» готовится файл некоторой модели с обобщенным именем model.mdl. При необходимости для его подготовки могут привлекаться другие пакеты расширения, в частности все пакеты, описанные в этой книге. Файл модели вместе с системным файлом — шаблоном (system.tmf) — поступает на вход утилиты Build пакета RTW. Эта утилита создает промежуточный файл model.rtw в языково-независимом

текстовом формате. Файл содержит все необходимые для функционирования модели данные, например установки параметров, типы и размерности сигналов, дискретность модельного времени и т. д.

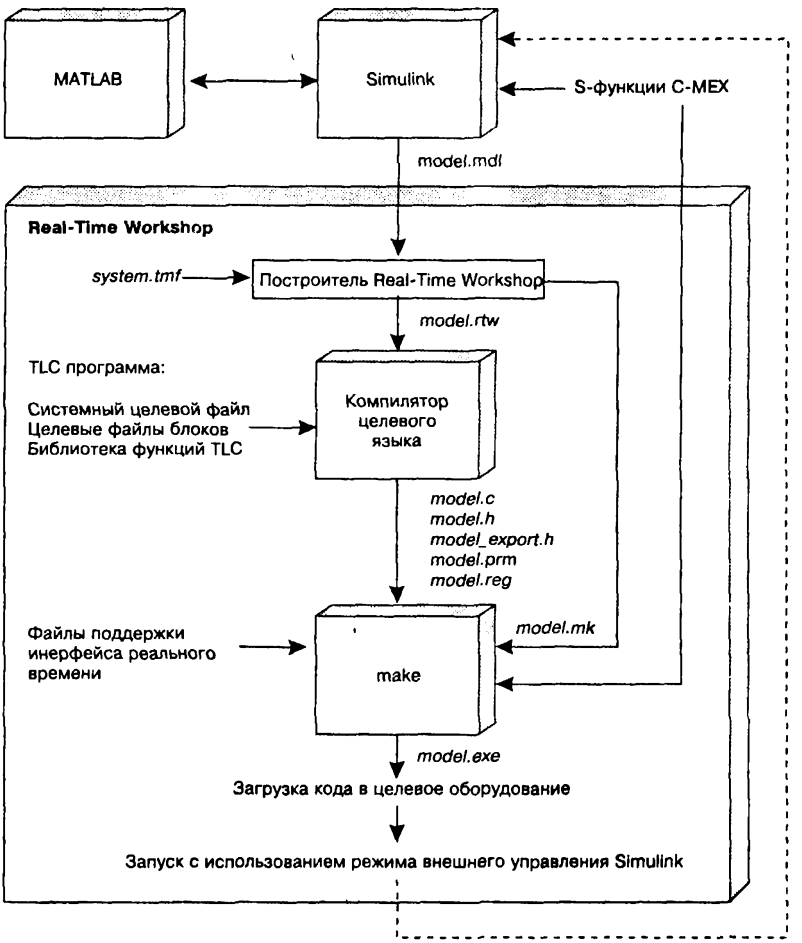


Рис. 7.37. Диаграмма работы пакета Real-Time Workshop

Данный файл вместе с необходимыми для компиляции другими файлами (включая системные и библиотечные) поступает в компилятор TLC (Target Language Compiler). Этот компилятор создает набор из

пяти файлов, перечисленных на рис. 7.37. Назначение этих файлов следующее:

- model.c — файл, задающий вход в программу (точка входа);
- model.h — файл заголовка, описывающий связи между блоками модели;
- model_export.h — файл заголовка с информацией об экспортируемых сигналах и параметрах модели;
- model.prm — файл с параметрами блоков S-модели;
- model.reg — файл описания параметров модельного времени для S-модели.

Все эти файлы поступают на вход утилиты Make, которая и создает исполняемый файл с расширением .exe (то есть файл, исполняемый в реальном масштабе времени). Состав файлов, используемых утилитой Make при моделировании управления летательным аппаратом F14, представлен на диаграмме рис. 7.38.

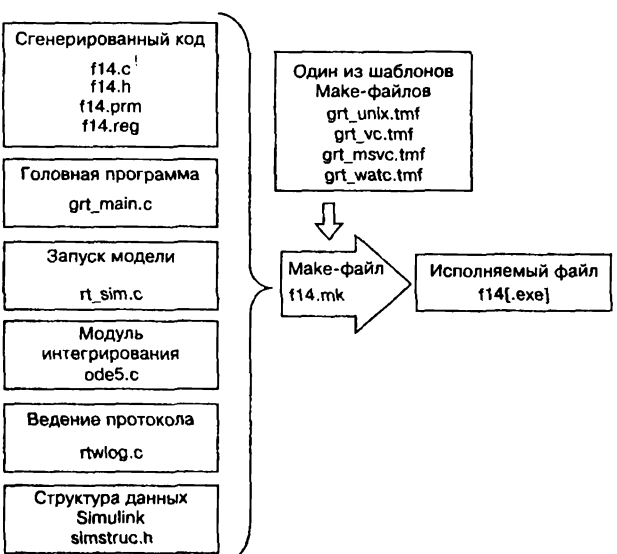


Рис. 7.38. Состав файлов при работе утилиты Make

Полученный исполняемый файл может быть запущен как из командной строки системы MATLAB, так и без нее (из сеанса MS-DOS)

Набор файлов, созданных пакетом RTW, размещается в директории WORK.

Что дает компиляция моделей

Заметим, что исполнение моделей в системе «MATLAB+Simulink» обычно происходит в интерпретирующем режиме. Это означает, что каждая команда или функция сначала анализируется, и только затем исполняется, что ведет к относительно медленной работе системы. Впрочем, следует отметить, что, даже будучи интерпретатором, MATLAB выгодно отличается от других систем компьютерной математики повышенной скоростью работы. Это достигнуто усовершенствованием используемых в MATLAB матричных алгоритмов.

Компиляция означает, что этапы интерпретации и сборки становятся не нужны, поскольку они уже были выполнены в процессе создания исполняемых файлов. В результате скорость вычислений значительно возрастает — от нескольких раз до нескольких десятков раз. Это позволяет заметно ускорить моделирование типовых систем, особенно когда нужно проводить его многократно с изменением тех или иных параметров. А при использовании системы MATLAB для управления внешними системами и устройствами, работающими в реальном масштабе времени, компиляция моделей просто жизненно необходима.

Заметим, что в подавляющем большинстве системы компьютерной математики не позволяют создавать исполняемые в реальном масштабе времени файлы своих моделей. Это означает, что работать с моделями можно только при загруженной математической системе. Система MATLAB в этом отношении уникальна, поскольку позволяет (при полном наборе нужных для этого средств, включая внешний компилятор C/C++) создавать исполняемые файлы. Это свойство особенно ценно при использовании созданных файлов совместно с внешними программами и периферийным оборудованием, которое может быть подключено к компьютеру.

Однако стоит сказать и о трудностях такого подхода. Помимо уже указанных в этой главе проблем необходимо упомянуть самые разные виды нестыковки между как аппаратными, так и программными компонентами систем, на аппаратном уровне работающих в реальном масштабе времени. Поэтому описание технологии проектирования и практической отладки таких систем далеко выходит за рамки задачи этой книги, а приведенный ниже материал является лишь ознакомительным.

Взаимодействие с внешними устройствами

На приведенной диаграмме (см. рис. 7.37) схематично показано, как происходит работа с внешними устройствами. Файл model.exe используется с их платами расширения (то есть с аппаратными средствами — hardware). Внешние устройства используются для запуска пакета Simulink в режиме внешнего управления (см. команду External в меню Simulation на рис. 7.35). Таким образом, в данном случае образуется замкнутая система управления внешними устройствами. Она дает широкие возможности для программного управления ими и обработки поступающей из них информации.

Режим внешнего управления позволяет осуществлять динамическую настройку параметров исполнения программ, созданных пакетом RTW, на основе применения механизма API (Application Program Interface, интерфейс прикладных программ), имеющегося у операционных систем класса Windows 95/98/NT. Реализуемая пакетом система разработки программного кода получила название *среды быстрого макетирования* (rapid prototyping program framework).

Если говорить кратко, работа системы «MATLAB+Simulink» в режиме внешнего управления происходит следующим образом. Вначале, при соответствующих начальных установках параметров, выполняется моделирование и строится SF-диаграмма. После этого Simulink переходит в ждущий режим и ожидает модификации параметров. Это может осуществляться пользователем вручную (в среде Simulink) или по сигналам от внешнего оборудования. Опуская описание самого механизма такой передачи, отметим, что таким образом реализуется замкнутый цикл управления моделью с возможностью вмешательства оператора, осуществляющего общее управление.

Подобный режим управления имеет свои ограничения в части изменения параметров. Так, нельзя менять следующие характеристики системы:

- число состояний;
- число входов и выходов любых блоков;
- шаг модельного времени;
- алгоритм интегрирования для непрерывных систем;
- имена модели и отдельных ее блоков;
- параметры блоков типа Fcn (функция).

Иными словами, можно моделировать лишь системы с неизменяемой структурой. Тем не менее, пакет RTW (а также ряд других па-

кетов) позволяет вести эффективную работу с самыми различными периферийными устройствами, такими как устройства для обработки сигналов от различных датчиков, устройства контроля напряжений и токов, аппаратные средства для управления роботами (в том числе промышленными и бытовыми) и др.

Как правило, периферийные устройства такого рода требуют приобретения довольно специфичных и дорогих аппаратных средств (плат расширения), стоимость которых нередко намного превышает стоимость современного компьютера с установленным на нем базовым пакетом «MATLAB+Simulink». Но и возможности такого применения системы MATLAB становятся почти фантастическими!

Подготовка к созданию исполняемого файла

Для создания исполняемого файла в нормальном режиме моделирования надо прежде всего запустить MATLAB, Simulink и те пакеты расширения, которые нужны для создания компилируемой модели. Возьмем, к примеру, модель ключа с блоком построения SF-диаграммы, представленную на рис. 7.39. Затем надо опробовать модель и убедиться в ее правильной работе в интерпретирующем (обычном) режиме работы системы MATLAB.

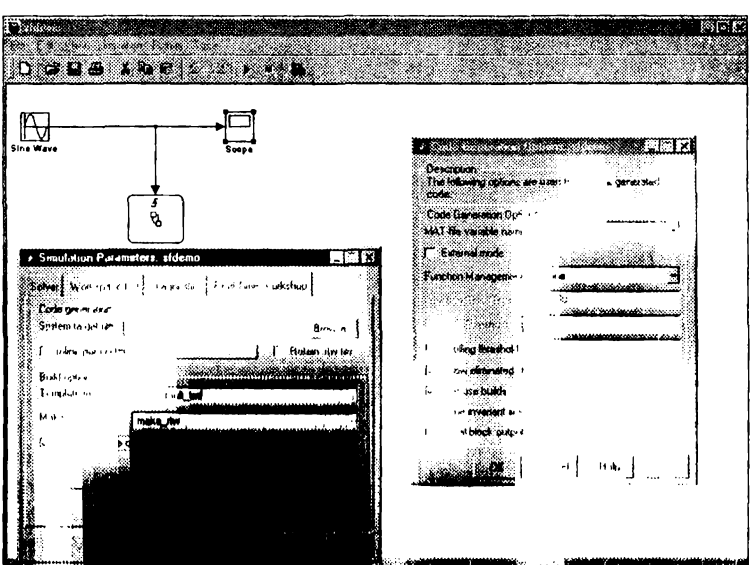


Рис. 7.39. Подготовка модели к компиляции

Далее следует просмотреть параметры модели (как это делается, описывалось выше) и параметры пакета RTW. Для этого в меню Simulation выберите команду Parameters и в появившемся окне установки параметров перейдите на вкладку Real-Time Workshop (см. рис. 7.39 слева).

Рассмотрим основные установки, производимые на этой вкладке. В поле System target file указывается имя tlc-файла, который будет использоваться для создания RTW-кода. По умолчанию это файл grt.tlc, но это имя можно изменить. Выбор tlc-файла зависит от типа создаваемого программного кода и вида целевого оборудования (target). Кнопка Browse открывает окно со списком всех tlc-файлов (рис. 7.40). В этом окне можно выбрать файл, подходящий для решения конкретной задачи. Кнопка OK данного окна фиксирует выбор файла.

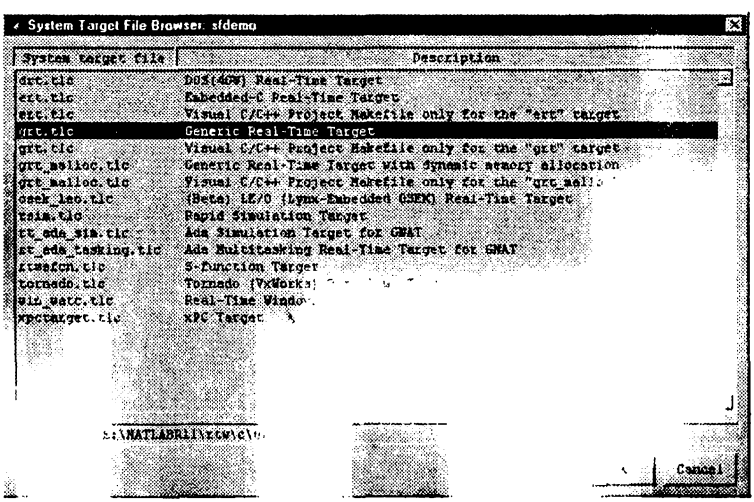


Рис. 7.40. Окно со списком tlc-файлов

Если установить флажок Inline parameters, то на место имен переменных будут подставляться их значения и во всех блоках будет использована единая величина шага модельного времени (Sample time). При этом будет доступна кнопка Tunable Parameters (Подстраиваемые параметры), позволяющая изменить обычно неизменяемые параметры модели (рис. 7.41). Включенные в список переменные становятся настраиваемыми и для них можно указать класс хранения.

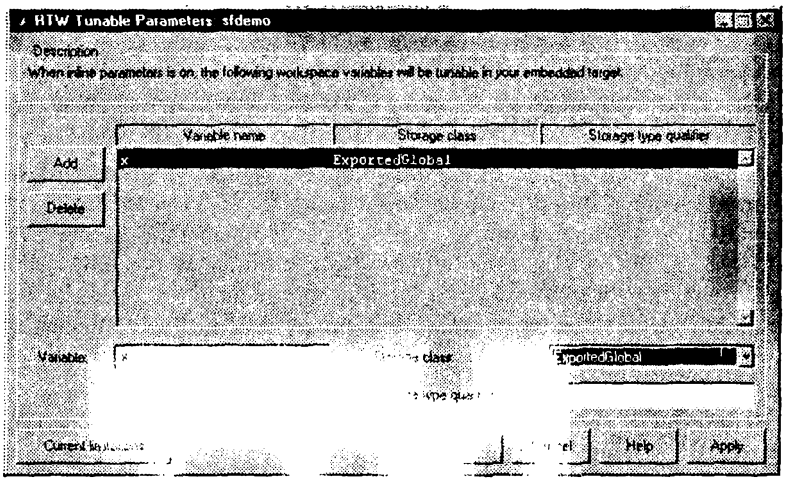


Рис. 7.41. Окно установки настраиваемых параметров

Флажок Retain .rtw file позволяет сохранить в текущей папке промежуточное описание RTW-модели в виде одноименного rtw- файла.

Кнопка Options открывает окно для установки дополнительных параметров работы пакета RTW. Это окно было показано выше (см. рис. 7.39 справа). Вид окна зависит от выбранного tlc-файла.

Из элементов этого окна отметим лишь наиболее важные. Параметр Loop rolling threshold задает порог генерации циклов. Если число проходов цикла меньше указанного порога, цикл заменяется повторяющимися фрагментами кода. Флажок Verbose builds задает вывод в командном окне MATLAB сообщений о выполнении этапов генерации программного кода. Флажок Inline invariant signals задает встраивание в программный код неизменяемых сигналов в виде констант. С другими элементами этого окна заинтересованный читатель может ознакомиться самостоятельно.

Активизация кнопки Stateflow Options окна установки параметров RTW открывает окно параметров, показанное на рис. 7.42 слева. Кнопки этого окна Target Options и Coder Options выводят окна, также показанные на рис. 7.42. Обычно необходимости менять эти параметры нет; все флажки по умолчанию сброшены. Однако в случае необходимости настройка параметров возможна, хотя и выходит за рамки нашего рассмотрения.

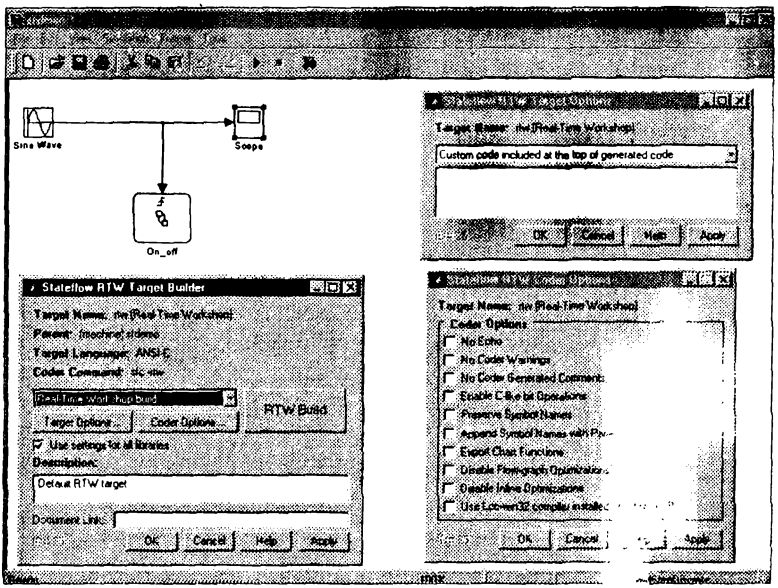


Рис. 7.42. Окна параметров

Запуск процесса компиляции производится активизацией кнопки Build на вкладке Real-Time Workshop окна установки параметров модели (см. рис. 7.39). Можно также сделать это из окна параметров Stateflow RTW Target Builder (см. рис. 7.42; для выполнения компиляции служит большая кнопка RTW Build). В процессе компиляции на передний план выводится окно системы MATLAB, в котором можно проследить за процессом создания файлов, образующихся в ходе компиляции (рис. 7.43).

Как уже отмечалось, для полного завершения компиляции и создания исполняемого файла с расширением .exe необходим компилятор языка C/C++. Если он не установлен, будет выдано сообщение об ошибке (в отдельном окне и в окне системы MATLAB). Ошибка возможна и в том случае, если путь к компилятору не включен в список путей, используемых операционной системой для поиска программ (переменная PATH).

При успешной компиляции исполняемый файл и сопровождающие его файлы размещаются в директории WORK, расположенной в директории системы MATLAB. Вместе с исполняемым файлом создаются пусковой файл с расширением .bat и файл на языке C с расшире-

нием .c. Исполняемый файл можно запустить прямо в командном окне системы MATLAB, указав имя файла после восклицательного знака, то есть в виде !Имя_файла. Заметим, что такой знак означает вызов команды операционной системы. Командой save Имя_файла можно записать результаты. Здесь надо использовать то же имя, которое было выбрано для исполняемого файла.

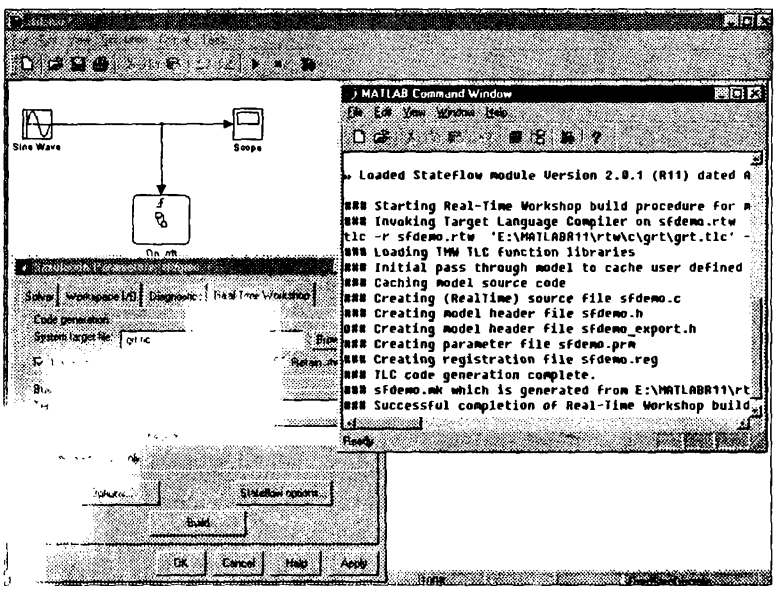


Рис. 7.43. Иллюстрация процесса компиляции

В заключение следует отметить, что далеко не каждая модель сразу компилируется. Как правило, чтобы компиляция прошла успешно, большинство моделей нуждается в доработке. Отчасти это связано с отмеченными выше ограничениями, налагаемыми на параметры компилируемой модели. К примеру, большинство моделей из демонстрационных примеров просто не работает при фиксированном шаге по времени, а между тем только такой шаг разрешен при компиляции. Возникают проблемы и с выбором метода моделирования.

Кроме того, успешная компиляция вовсе не означает, что исполняемый файл будет нормально работать. Правильнее использовать полученный на языке C файл модели для его последующей отладки и компиляции в компиляторе языка C/C++. Таким образом, создание корректно работающих исполняемых файлов все еще остается уде-

лом опытных программистов, и вряд ли эту работу стоит рекомендовать начинающим пользователям. Для них куда более привычной и родной будет среда «MATLAB+Simulink», в которой проблем с отладкой моделей намного меньше.

Вопросы отладки моделирования во внешнем режиме работы мы опускаем, поскольку они далеко выходят за рамки данной книги и требуют рассмотрения как применяемых в этом случае специфических внешних устройств, так и деталей сетевых протоколов, на которых они базируются (например TCP/IP). Для полноты обзора средств RTW отметим, что в его состав может включаться расширение RTW Ada Coder, позволяющее готовить коды на языке Ada (стандарт Ada95). Полученные файлы могут выполняться на компьютерах нескольких платформ, при этом используется описанный выше графический интерфейс пользователя.

Алфавитный указатель

F

G

I

K

L

M

N

O

P

Q

R

S

T

U

V

Y

Z

А

Б

В

Г

Д

З

И

К

Л

M

Н

О

П

P

С

Т

У

Ф

Ц

Ч

Ш

Э

КНИГА-ПОЧТОЙ

ЗАКАЗАТЬ КНИГИ ИЗДАТЕЛЬСКОГО ДОМА «ПИТЕР» МОЖНО ЛЮБЫМ УДОБНЫМ ДЛЯ ВАС СПОСОБОМ:

- по телефону: **(812) 294-01-04;**
- по электронному адресу: **postbook@piter.com;**
- на нашем сервере: **www.piter.com;**
- по почте: **197198, Санкт-Петербург, а/я 619 ЗАО «Питер Пост».**

ВЫ МОЖЕТЕ ВЫБРАТЬ ОДИН ИЗ ДВУХ СПОСОБОВ ДОСТАВКИ И ОПЛАТЫ ИЗДАНИЙ:

- Наложенным платежом с оплатой заказа при получении посылки на ближайшем почтовом отделении. Цены на издания приведены ориентировочно и включают в себя стоимость пересылки по почте **(но без учета авиатарифа)**. Книги будут высланы нашей службой **«Книга — почтой»** в течение **двух недель** после получения заказа или выхода книги из печати.
- Оплата наличными при курьерской доставке **(для жителей Санкт-Петербурга и Москвы)**. Курьер бесплатно доставит заказ по указанному адресу в удобное для вас время в течение трех дней. Такой заказ лучше оформлять по телефону.

ПРИ ОФОРМЛЕНИИ ЗАКАЗА УКАЖИТЕ:

- фамилию, имя, отчество, телефон, факс, e-mail;
- почтовый индекс, регион, район, населенный пункт, улицу, дом, корпус, квартиру;
- название книги, автора, код, количество заказываемых экземпляров.

Вы можете заказать бесплатный журнал «Клуб Профессионал».

ИЗДАТЕЛЬСКИЙ ДОМ ПИТЕР®
WWW.PITER.COM